SYMANTEC.™

Visual Cafe™ for Java™ Windows Edition

User's Guide

Symantec Visual Cafe™ for Java™ User's Guide

The software described in this book is furnished under a license agreement and may be used only in accordance with the terms of the agreement.

Copyright Notice

Copyright © 1997 Symantec Corporation.

All Rights Reserved.

Released: 8/97 for Visual Cafe for Java 2.0

This document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior consent in writing from Symantec Corporation, 10201 Torre Avenue, Cupertino, CA 95014.

ALL EXAMPLES WITH NAMES, COMPANY NAMES, OR COMPANIES THAT APPEAR IN THIS MANUAL ARE IMAGINARY AND DO NOT REFER TO, OR PORTRAY, IN NAME OR SUBSTANCE, ANY ACTUAL NAMES, COMPANIES, ENTITIES, OR INSTITUTIONS. ANY RESEMBLANCE TO ANY REAL PERSON, COMPANY, ENTITY, OR INSTITUTION IS PURELY COINCIDENTAL.

Every effort has been made to ensure the accuracy of this manual. However, Symantec makes no warranties with respect to this documentation and disclaims any implied warranties of merchantability and fitness for a particular purpose. Symantec shall not be liable for any errors or for incidental or consequential damages in connection with the furnishing, performance, or use of this manual or the examples herein. The information in this document is subject to change without notice.

Trademarks

Symantec Visual Cafe, Symantec, and the Symantec logo are U.S. registered trademarks of Symantec Corporation.

Other product names mentioned in this manual may be trademarks or registered trademarks of their respective companies and are the sole property of their respective manufacturers.

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

SYMANTEC LICENSE AND WARRANTY

The software which accompanies this license (the "Software") is the property of Symantec or its licensors and is protected by copyright law. While Symantec continues to own the Software, you will have certain rights to use the Software after your acceptance of this license. Except as may be modified by a license addendum which accompanies this license, your rights and obligations with respect to the use of this Software are as follows:

• You may:

(i) use one copy of the Software on a single computer;

(ii) make one copy of the Software for archival purposes, or copy the software onto the hard disk of your computer and retain the original for archival purposes;

(iii) use the Software on a network, provided that you have a licensed copy of the Software for each computer that can access the Software over that network;

(iv) after written notice to Symantec, transfer the Software on a permanent basis to another person or entity, provided that you retain no copies of the Software and the transferee agrees to the terms of this agreement; and

(v) if a single person uses the computer on which the Software is installed at least 80% of the time, then after returning the completed product registration card which accompanies the Software, that person may also use the Software on a single home computer.

• You may not:

(i) copy the documentation which accompanies the Software;

(ii) sublicense, rent or lease any portion of the Software;

(iii) reverse engineer, decompile, disassemble, modify, translate, make any attempt to discover the source code of the Software, or create derivative works from the Software; or

(iv) use a previous version or copy of the Software after you have received a disk replacement set or an upgraded version as a replacement of the prior version, unless you donate a previous version of an upgraded version to a charity of your choice, and such charity agrees in writing that it will be the sole end user of the product, and that it will abide by the terms of this agreement. Unless you so donate a previous version of an upgraded version, upon upgrading the Software, all copies of the prior version must be destroyed.

• Sixty Day Money Back Guarantee:

If you are the original licensee of this copy of the Software and are dissatisfied with it for any reason, you may return the complete product, together with your receipt, to Symantec or an authorized dealer, postage prepaid, for a full refund at any time during the sixty day period following the delivery to you of the Software.

• Limited Warranty:

Symantec warrants that the media on which the Software is distributed will be free from defects for a period of sixty (60) days from the date of delivery of the Software to you. Your sole remedy in the event of a breach of this warranty will be that Symantec will, at its option, replace any defective media returned to Symantec within the warranty period or refund the money you paid for the Software. Symantec does not warrant that the Software will meet your requirements or that operation of the Software will be uninterrupted or that the Software will be error-free.

THE ABOVE WARRANTY IS EXCLUSIVE AND IN LIEU OF ALL OTHER WARRANTIES, WHETHER EXPRESS OR IMPLIED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE OTHER RIGHTS, WHICH VARY FROM STATE TO STATE.

• Disclaimer of Damages:

REGARDLESS OF WHETHER ANY REMEDY SET FORTH HEREIN FAILS OF ITS ESSENTIAL PURPOSE, IN NO EVENT WILL SYMANTEC BE LIABLE TO YOU FOR ANY SPECIAL, CONSEQUENTIAL, INDIRECT OR SIMILAR DAMAGES, INCLUDING ANY LOST PROFITS OR LOST DATA ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE EVEN IF SYMANTEC HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

SOME STATES DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

IN NO CASE SHALL SYMANTEC'S LIABILITY EXCEED THE PURCHASE PRICE FOR THE SOFTWARE. The disclaimers and limitations set forth above will apply regardless of whether you accept the Software.

• U.S. Government Restricted Rights:

RESTRICTED RIGHTS LEGEND. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or subparagraphs (c) (1) and (2) of the Commercial Computer Software-Restricted Rights clause at 48 CFR 52.227-19, as applicable, Symantec Corporation, 10201 Torre Avenue, Cupertino, CA 95014.

• General:

This Agreement will be governed by the laws of the State of California. This Agreement may only be modified by a license addendum which accompanies this license or by a written document which has been signed by both you and Symantec. Should you have any questions concerning this Agreement, or if you desire to contact Symantec for any reason, please write:

Symantec Customer Service, 175 W. Broadway,
Eugene, OR 97401

Language Addendum

If the Software is a Symantec language product, then you have a royalty-free right to include object code derived from the libraries in programs that you develop using the Software and you also have the right to use, distribute, and license such programs to third parties without payment of any further license fees, so long as a copyright notice sufficient to protect your copyright in the program is included in the graphic display of your program and on the labels affixed to the media on which your program is distributed. You also have a royalty-free right to include unmodified Symantec Class files required by your programs. The Symantec Class files are in the following installed zip file in the Visual Cafe directory – VisualCafe\redist\symbeans.jar. The Java Virtual Machine (VM) or Just In Time (JIT) compiler are not freely distributable.

C O N T E N T S

Section I Using Visual Cafe

Chapter 1 Welcome to Visual Cafe

Features 1-2
Visual Page 1-2
Visual Cafe Web Development Edition 1-4
Visual Cafe Professional Development Edition 1-6
Visual Cafe Database Development Edition 1-6
What's new in Visual Cafe 2.0 1-7
Contextual Menus 1-7
ZIP archive and JAR (Java ARchive) support 1-7
Open by Name 1-7
Version Compatibility 1-8
Project and environment management 1-8
Drag-and-drop 1-9
Display options 1-9
Debugger 1-9
Class Browser and Editor 1-9
New and enhanced Wizards 1-9
LiveUpdate 1-10
Visual Cafe documentation 1-10
Visual Cafe Getting Started and Tour 1-10
Online Help 1-10
User's Guide 1-11
Prerequisites for using Visual Cafe 1-11
System requirements 1-12
Starting Visual Cafe 1-12
The Visual Cafe environment 1-13
Windows 1-13
Toolbars 1-14
Visual Cafe windows 1-15
Editors 1-15
How much Java do I need to know to use Visual Cafe? 1-15
What's next? 1-16

Chapter 2 Developing in Visual Cafe

Putting Visual Cafe to work 2-1
- Understanding Visual Cafe components 2-2
- Forms hold your Java program together 2-3
- Projects keep your work together 2-4
- Using workspaces to customize your work environment 2-4

Debugging with Visual Cafe 2-4
- Symantec's Just-in-time compiler and the Visual Cafe environment 2-5
- Sun Microsystems' Java Compiler and JDK 2-5

Overview of creating a Java program 2-5
- Overview of creating an applet 2-6
- Overview of creating an application 2-7
- Overview of creating a dbAWARE applet or application 2-7

Chapter 3 Working with projects and workspaces

What is a project? 3-1
- The project file and the project folder 3-2
- Basic concepts of working with a project 3-3
- Looking at the Objects view 3-3
- Looking at the Packages view 3-6
- Looking at the Files view 3-8

Creating projects by using templates 3-8
- Choosing the project template 3-9
- Constructing a logically arranged project folder 3-10
- Looking at the files in the project folder 3-12
- Adding files to a project 3-13
- Working on multiple projects 3-14
- Compiling projects 3-15
- Defining project options 3-15

Beginning to work with a project 3-15
- Creating a new project 3-15
- Opening an existing project 3-16
- Saving, renaming, and copying a project 3-17
- Adding files to and removing files from projects 3-17
- Migration from Visual Cafe 1.0 to Visual Cafe 2.0 3-22
- Closing a project 3-23

Using the Project window 3-24
- Dragging-and-dropping into the Project window 3-24

Viewing the files in a project 3-26
Adding a new file to a project 3-26
Adding an existing file to a project 3-27
Deleting a file from a project 3-27
Copying a file of a project 3-28
Working with components in a project 3-29
Viewing the components and HTML files in a project 3-29
Adding a component 3-30
Copying a component 3-30
Renaming a component 3-31
Deleting a component 3-32
Working with packages 3-33
Viewing the packages in a project 3-33
Adding packages to Visual Cafe 3-34
Setting project-level options 3-34
Project Options 3-35
Project types 3-40
Specifying what applets to run and the HTML file 3-41
Making applets run in the Applet Viewer or a browser 3-42
Specifying the main class to run for an application 3-43
Specifying arguments for application execution 3-44
Specifying whether to parse imports 3-44
Specifying whether to clear messages before builds 3-45
Setting compiler options for a project 3-45
Specifying general compiler options 3-46
Specifying file search paths and the output folder 3-47
Using Version Control 3-51
Setting remote debugging options for a project 3-52
Defining the Visual Cafe startup mode 3-52
Automating source file backups 3-53
Defining a new default template 3-54
Creating a project template and adding it to the library 3-54
Deleting a project template 3-55
Use Workspaces to customize your work environment 3-55
Working with subprojects 3-58
Adding subprojects 3-58
Objects View 3-59
Files view 3-59

Project options and subprojects 3-60
Importing source code 3-61
The Visual Cafe main file menus 3-61
The File menu 3-61
New Project… 3-62
New File 3-63
Open… 3-63
The Edit menu 3-65
The Search menu 3-66
The Insert menu 3-69
The Object menu 3-70
The Window menu 3-71
The Help menu 3-72

Chapter 4 Working with Java Source Code

Using the Class Browser 4-1
Grouping and sorting classes and members 4-3
Controlling the display of inherited methods 4-4
Navigating the panes 4-5
Navigating the Classes pane 4-5
Using the Classes pane 4-5
Finding a class 4-6
Adding a class 4-7
Adding a subclass from the Class Browser 4-7
Viewing and editing the source code for a class 4-10
Adding a method from the Class Browser 4-10
Navigating the Members pane 4-11
Navigating the Source pane 4-15
Configuring the Class Browser and Hierarchy Editor 4-16
Navigating the Hierarchy Editor 4-17
Working with subclasses 4-19
Changing inheritance relationships 4-20
Deleting inheritance relationships 4-20
Using the Class Attributes dialog box 4-20
Using the Insert Class Wizard 4-20
Using the Source Editor 4-22
Creating a new document 4-23
Adding code to a Java source file 4-23

Editing source code 4-24
Correcting your source code 4-25
Viewing a component's Java source code 4-25
Enhancing an object's Java source code 4-25
Binding code to a form or component 4-26
Binding code to a menu command 4-27
Programming hot keys 4-27
Moving around in a file with the Search menu 4-32
Searching through and comparing multiple files 4-33
Adding a method from the Source window 4-33
Enhancing Java code for a component 4-34
Specifying the search file type and location 4-34
Specifying the file search path 4-35
Setting Advanced Search Criteria 4-37
Adding packages to Visual Cafe 4-37
Using the Package view 4-38
Working with events 4-39
Using events with your own components 4-39
Adding an event to a component 4-39
Editing an event handler 4-40
Deleting an event handler 4-41
Editing event methods 4-41
Working with the Interaction wizard 4-41
Changes between the JDK 1.0 and 1.1 event models 4-42
Creating an interaction 4-44
Changing an existing interaction 4-46
Deleting an interaction 4-47
Using the Classes, and Hierarchy Editor menus 4-50
Using the Classes menu 4-50
Using the Hierarchy Editor right-click menu 4-51
Using Macros in Visual Cafe 4-52
Recording and playing the Default Macro 4-52
Saving the default macro and using it with other macros 4-53
Using recorded macros other than the default macro 4-53
Using the ScriptMaker dialog box 4-53

Chapter 5 Including Visual Components

What are Visual Cafe forms? 5-1

Understanding the container class 5-2
Working with basic user interface components 5-4
Creating component layouts 5-4
Working in the Form Designer 5-5
Displaying graphics in the Form Designer 5-6
Creating Java code 5-6
Designing a GUI with Visual Cafe 5-7
Adding forms to the project 5-8
Adding components to a form 5-9
Arranging components on your forms 5-14
Modifying component properties 5-19
Creating component interactions 5-20
Creating menus with the Menu Designer 5-21
Using the Menu Designer pop-up menu 5-22
Adding a menu to a form 5-23
Copying a menu 5-23
Adding a menu bar to a frame or dialog box 5-24
Adding menus to a menu bar 5-24
Adding menu items to menus 5-25
Adding submenus to menu items 5-25
Editing a menu structure 5-26
Editing menu bars and menus 5-26
Associating command keys and menu items 5-27
Binding code to a menu item 5-28
Working with the Component Palette 5-28
Using the InvisibleHTMLLink component 5-29
Using the MultiList component 5-29
Using the ScrollingPanel container 5-30
Using the TabPanel container 5-31
Using the TreeView component 5-32
Building a custom Palette 5-33
Creating a Palette tab 5-33
Adding components to the Palette 5-33
Moving Palette components 5-35
Deleting components from the Palette 5-35
Working with the Component Library 5-36
Viewing a component’s Java source 5-37

Chapter 6 Compiling, running, and deploying your program

Introduction 6-1
Concepts of applets and applications 6-1
- Applets 6-1
- Advantages and disadvantages 6-2
- Applications 6-3

Compiling and running 6-4
- Setting compiler options 6-5
- Correcting your source code 6-7

Deploying Java programs 6-7
- Deploying your applet 6-8
- Including your applet in a Web page 6-10
- Deploying your application 6-13

Determining what class files an applet or application needs 6-16
- Using the JAR command to get the class files your Java program needs 6-16
- Using SJ to determine what class files your Java program needs 6-17

Configuring UNIX-based Web servers 6-17
Building your Java program 6-18
- Building your project with commands in the Project menu 6-18

Messages window 6-19

Chapter 7 Working with JavaBeans

JavaBeans and Java 7-1
JavaBean terminology 7-2
Basic JavaBean structure 7-2
The JavaBeans services 7-2
- Property management 7-3
- Introspection 7-4
- Event handling 7-6
- Persistence 7-7
- Application builder support 7-8

Creating a Bean 7-8
- Bean design fundamentals 7-9
- Testing your Bean 7-9

Adding and using Beans in Visual Cafe 7-9
- Using JAR files 7-10
- Creating a JAR file 7-10
- Expanding a JAR file 7-11

Converting components (description files) to JavaBeans 7-11
Adding a JavaBeans component to the Component Library 7-13
Creating a JavaBean component 7-14
Adding Visual Cafe information to a JavaBean 7-15
SymantecBeanDescriptor 7-16
ConnectionDescriptor 7-16
Code samples 7-17
Adding JavaBeans to the Component Library 7-18
Deleting JavaBeans from the Component Library 7-20
Viewing and changing JavaBean properties 7-20
Packaging and deploying JavaBeans 7-21
Visual Cafe's tools for building Beans 7-21
Bringing Beans into the Visual Cafe environment 7-22
Bean associates 7-22
Appearance 7-23
Renaming a Bean 7-23
Hierarchical properties 7-23
Internationalization properties 7-23

Chapter 8 Debugging Java Programs

Using the Debug workspace 8-2
Overview of the Breakpoints window 8-3
Overview of the Variables window 8-3
Overview of the Watch window 8-4
Overview of the Threads window 8-4
Overview of the Call Stack window 8-5
Overview of the Messages window 8-5
Overview of the Source window 8-6
Running the Debugger 8-6
Starting a debugging session 8-7
Scrolling in the Source Editor 8-7
Using the Debug toolbar 8-8
Stepping through code 8-8
Pausing a program 8-9
Stopping a program 8-9
Stepping into a method 8-9
Using Debug > Step Into 8-9
Stepping over a method 8-10

Using Debug Step Over 8-10
Stepping out of a method 8-10
Toggling a breakpoint 8-10
Watching a variable 8-10
Running to the first line 8-11
Running the program to the end 8-12
Running to the cursor location 8-12
Resuming a program 8-12
Restarting a program 8-12
Handling exceptions 8-13
Throwing exceptions 8-13
Catching exceptions 8-13
Setting exceptions in Visual Cafe 8-14
Changing source code 8-15
Using the Source window 8-15
Working with breakpoints 8-16
Setting a breakpoint on a line number 8-17
Setting simple breakpoints 8-18
Setting a breakpoint on a method name 8-18
Setting a breakpoint on a variable or expression 8-19
Setting a conditional breakpoint 8-19
Clearing breakpoints 8-20
Enabling or disabling a breakpoint 8-20
Ignoring all breakpoints 8-21
Viewing the source for a breakpoint 8-21
Stepping through code when the program is paused 8-22
Using the Variables window 8-23
Viewing the value of a variable 8-23
Viewing type information for a variable 8-24
Modifying a variable in the Variables window 8-24
Using expressions in the Watch window 8-24
Adding a variable to the Watch window 8-25
Modifying a variable or expression in the Watch window 8-25
Deleting a variable or expression from the Watch window 8-26
Using the Call Stack window 8-26
Viewing parameters for a method on the Call Stack window 8-27
Viewing variables for a method on the call stack 8-27
Viewing source for a method on the call stack 8-28

Ending a debugging session 8-28
Debugging threads 8-28
Using the Threads window 8-29
Debugging a single thread 8-31
Debugging remotely 8-34
Setting up for remote debugging 8-34
Starting remote applets or application debugging 8-34
Ending remote applets or application debugging 8-35
Using Debugger-specific menus 8-35
Overview of the Project menu 8-35
Overview of the Debug menu 8-36
Overview of the Insert menu 8-37
Overview of the Breakpoints menu 8-38
Overview of the Variables menu 8-39
Overview of the Threads menu 8-39
Overview of the Calls menu 8-39
Overview of the Source menu 8-40
Overview of the Window menu 8-41

Chapter 9 Fine-Tuning Visual Cafe

Setting environment options 9-1
Setting environment options in the General tab 9-1
Setting debugging options for the environment 9-3
Specifying text formatting for Visual Cafe windows 9-5
Mapping Visual Cafe commands to key sequences 9-8
Specifying key editing options for the Source Editor 9-10
Customizing the display font and color in Visual Cafe windows 9-11
Specifying backup and save options 9-13
Specifying code editing options 9-15
Controlling toolbar position and visibility 9-17
Enabling ValueTips 9-17
Customizing Class Browser and Class Hierarchy window editing 9-18
Changing editor properties 9-18
Updating Visual Cafe with LiveUpdate 9-19
Using LiveUpdate over the Internet 9-20
Using LiveUpdate with your modem 9-20
Uninstalling LiveUpdate upgrades 9-23
Troubleshooting Visual Cafe for Windows 9-24

Limitations of the Java language 9-24
Common programming errors 9-25
Compiler errors 9-25
Using Visual Cafe to locate compiler errors 9-26
Cross-platform considerations 9-26
Visual Cafe cross-development 9-26
Browser issues 9-26
When do you have to write your own code? 9-27
Event handling 9-27
Disabling automatic code generation in Visual Cafe 9-27
Disabling code that is automatically generated 9-28
How to tell when your Visual Cafe environment becomes corrupted 9-28

Section II Professional Features

Chapter 10 Creating Native Win32 Java Applications

Creating native executables and DLLs 10-2
Setting project options for native applications 10-3
Registering DLLs using SNJREG 10-12
Debugging Native Win32 Java applications 10-13
Deploying native Win32 applications, DLLs and libraries 10-13
Converting Java applications from bytecode to native Win32 10-14
Considerations when creating native Win32 Java applications 10-15
Linking native Java applications 10-15
The main class in bytecode and native applications 10-16
Working with samples of native applications 10-17
Example of creating an executable file 10-17
Example of creating an executable with a DLL 10-18

Chapter 11 Incremental Debugging and Importing Projects

Incremental debugging 11-1
Importing projects from Cafe 11-2
Importing Visual J++ projects into Visual Cafe 11-3
Considerations when importing Visual J++ projects 11-5

Section III Database Connectivity

Chapter 12 Developing a dbAWARE project

Overview 12-2
Setting the Database Environment Options 12-2
About the dbAWARE components 12-3
About the database browser, dbNAVIGATOR 12-10
The dbAWARE wizards 12-11
Database manipulation functions 12-12
Creating a dbAWARE project 12-12
Starting dbANYWHERE 12-13
Creating the database for your project 12-14
Defining your data source 12-15
Using the dbAWARE project wizard 12-19
Using the Add Table wizard 12-28
Using dbNAVIGATOR in form development 12-29
Connecting to a server using dbNAVIGATOR 12-30
Connecting to a database 12-30
Adding a dbAWARE text field to a form 12-31
Refreshing dbNAVIGATOR 12-32
Adding a database grid component 12-32
Disconnecting from a database 12-34
Changing grid attributes 12-34
Changing foreground and background cell colors 12-35
Changing cell fonts 12-35
Changing Grid column attributes 12-36

Chapter 13 Using dbANYWHERE

About dbANYWHERE 13-1
What is dbANYWHERE? 13-1
How the dbANYWHERE architecture works 13-2
Configuring dbANYWHERE 13-3
One machine for local databases 13-4
Two machine configuration for remote databases 13-4
Two machine configuration for local and remote databases 13-5
Three machine configuration 13-5
Using the dbANYWHERE packages 13-6

Using the dbANYWHERE packages for development 13-7
Setting up the dbANYWHERE packages for a deployed application 13-7
Configuring dbANYWHERE as a Windows NT service (Windows NT only) 13-8
Running the service 13-9
Viewing messages 13-9
Connecting to a data sources 13-10
Connecting to a Sybase SQL Anywhere and Watcom data source 13-10
Connecting to an Informix data source 13-11
Connecting to Microsoft Access data source by means of JDBC 13-11
Connecting to an ODBC SQL-based data source by means of JDBC 13-12
Connecting to an ODBC Xbase data source by means of JDBC 13-13
Connecting to Microsoft SQL data source by means of JDBC 13-13
Connecting to Sybase SQL data source by means of JDBC 13-14
Connecting to Oracle data source by means of JDBC 13-14
Testing a data source connection 13-15
Using the dbANYWHERE tools 13-16
Using the DataSource tool 13-16
Editing the data source 13-17
Using the dbANYWHERE Admin tool 13-18
Connecting to a dbANYWHERE server 13-19
Checking dbANYWHERE server statistics 13-19
Checking dbANYWHERE connections 13-20
Testing link performance 13-21
Changing dbANYWHERE properties 13-22
Setting the dbANYWHERE server URL 13-23
Logging messages 13-23
Log menu 13-25
Allocating dbANYWHERE resources 13-26
Allowing remote administration 13-27
License 13-27
LiveUpdate 13-29

INDEX i

I

Using Visual Cafe

CHAPTER 1

Welcome to Visual Cafe

Symantec's Visual Cafe family of products is the first visual Rapid Application Development (RAD) tool designed exclusively for the Java programming language. Visual Cafe is a complete form-centric development environment that provides you with a rich set of What-You-See-Is-What-You-Get (WYSIWYG) tools and components necessary to develop, debug, and deploy high-performance Web applets, and stand-alone Java applications. Additional tools such as JavaBean and Java component libraries, graphic libraries, templates, and graphic editors provide the complete solution for the Java developer or sophisticated Web developer.

Visual Cafe is available in three editions:

- Visual Cafe Web Development Edition (WDE) provides the most complete solution for the Web developer with technically sophisticated needs. This edition is also for programmers who are new to Java. The Visual Cafe WDE includes support for the professional who writes, debugs, and uses HTML scripting and wants to take advantage of the power of the Java language and push their Web pages to the edge of technology.
- Visual Cafe Professional Development Edition (PDE) is for Developers who need the latest and most powerful Java features available in their development environment. It provides a unique customizable environment with wizards, reusable Java components, class libraries, project level management tools, incremental debugging, and remote debugging to speed your development process.
- Visual Cafe Database Development Edition (dbDE) is for Database Application Developers who want full database connectivity. Featuring a sophisticated set of high level tools and wizards, Visual Cafe dbDE allows you to quickly select database tables for use in your application,

automatically generate the associated Java code to manage the associations, adds the appropriate components to the application form, and binds components with the dbANYWHERE Server.

All three editions of Visual Cafe also contains Visual Page, Symantec's visual Web design tool, Netscape Navigator, and full JDK 1.1 support.

Features

This section describes the features of Visual Page and each edition of the Visual Cafe family of products.

Visual Page

Visual Page is a tool for creating and publishing documents for the Web. It's a what-you-see-is-what-you-get (WYSIWYG) environment that includes a visual designer, source code editor, and publishing utility. Visual Page is fully integrated into Visual Cafe to provide a complete development environment to design, develop, test, and deploy your Java projects.

Accessible and powerful feature set

Unlike most other HTML-based Web page editors, Visual Page displays your Web page so you see it as it's created. You can drag and drop blocks of text, graphics, and other Web-based media onto a Web page, and Visual Page creates the necessary HTML code for you.

Extensive HTML authoring tools

Visual Page doesn't require that you enter complex HTML code—entering and modifying content is quick and easy.

If you prefer to directly control your HTML code, you can view and edit the HTML code generated by Visual Page in the source editor. Changes made in the source editor are automatically reflected in the main editing window.

State-of-the-art media support

Visual Page offers a wide range of advanced Web media tools. Frames, tables, QuickTime movies, Navigator plug-ins, Java applets, JavaScript code and CGI-based forms can all be inserted effortlessly into your Web pages.

Extensive template and sample set

The Visual Page Designer's Toolkit consists of a number of templates and sample documents. The templates cover a wide range of document styles, and the sample documents cover a number of document types (such as product plans and registration forms). These templates can be used as the basis for your business or personal Web sites, saving you hours of design time in the process.

One-step publishing

In addition to its sophisticated layout tools, Visual Page includes several Web site management tools. With Visual Page, you can create your Web site on your local hard drive, and then publish the entire site to a Web server, without ever leaving Visual Page. No third party utility packages are necessary. When you need to perform routine maintenance on your Web site (for example, deleting unneeded files), you can do that within Visual Page as well.

Visual Page is also a useful tool for testing the links between pages in your site. Using Visual Page's preview mode, you can test and correct links simply and quickly.

Visual Page Documentation

Visual Page comes with a complete set of documentation and online help to assist you in the process of learning and using Visual Page. These documents are described below.

User s Guide

This may be the document you turn to most frequently as you work with Visual Page, as it contains step-by-step information on using Visual Page's features.

The contents of the User's Guide cover creating Web pages, advanced Web features, site publishing, and site maintenance. A glossary, covering commonly used Web-related terms, is also included.

Visual Page Getting Started Guide and Tour

The Visual Page Getting Started Guide consists of installation instructions for Visual Page, and a Tour. The Tour is designed to familiarize you with

the main features of Visual Page by guiding you through the process of building a working Web page. It requires no previous experience in Web page development, is a quick way to learn Visual Page—and it's fun!

Online help

Visual Page has an extensive online help system, providing all of the procedures for building a Web page. To access Visual Page's Help, choose Help Topics from the Help menu. From dialog boxes, toolbars, and windows, you can press F1 to access Help that is specific to your current Visual Page activity.

Visual Cafe Web Development Edition

Featuring a sophisticated set of high level, two-way tools, Visual Cafe WDE is the leading Java development environment. Two-way tools allow you to work in the Source Code Editor, or visually in the Form Editor and your project is updated as you work. Source code is generated for you while you work.

Support for JDK 1.1 (new in version 2.0)

With version 1.1 of the Java Development Kit, also known as JDK 1.1, you can write applets and applications that conform to the 1.1 Java Core API. It includes improvements in functionality, performance, and quality over JDK 1.0.2.

JDK 1.1 offers new capabilities: Internationalization, signed applets, JAR file format, AWT (window toolkit) enhancements, JavaBeans(tm) component model, networking enhancements, Math package for large numbers, Remote Method Invocation, Reflection, database connectivity, Native Interface, Object Serialization, Inner Classes, and performance enhancements.

Form-centric development environment

The form-centric development environment provides a powerful integrated form design tool. With form-centric development, you can drag and drop components as well as visually create their interactions between them.

Component Palette

The configurable component palette contains your favorite form components. You can drag and drop standard Visual Cafe components as well as add your own custom components to the Component Palette.

Component Library

The Component Library is a repository of components for forms, windows, and databases. You can take advantage of the libraries supplied by Visual Cafe or create and add your own. Since the library is extensible, Visual Cafe provides support for add-in third party components.

Interaction Wizard

The Interaction Wizard guides you visually through specifying actions based on form components events.

Automatic code generation

Form and property sheet modifications generate real-time source code. Source code is automatically updated and generated as you work in your projects. In addition, Visual Cafe allows you to have complete control over your source code at any time.

Debugger

The source-level debugger can be used to debug applications that are built using Visual Cafe. The debugger uses browser-style windows to display information about the process currently being debugged. One feature of the debugger is a Stack Crawl pane that contains a list of all stack frames for the program counter location in your code.

These windows include breakpoint, calls, messages, variables, and threads.

Hierarchical view

The Project, Class Editor, and Hierarchy windows incorporate a tree view to collapse and expand the various scopes, presenting you with visual relationships among the many parts of even the most complex projects.

Insert Class Wizard

With Visual Cafe's Insert Class Wizard, you can effortlessly integrate your own custom classes or third-party classes.

Visual Cafe Professional Development Edition

The Visual Cafe PDE includes all the standard features of Visual Cafe WDE, plus enhanced debugging tools that offer remote debugging and multithread support, as well as a unique incremental debugger that lets you keep coding while you debug.

Importing projects from Microsoft Visual J++

Reuse your Microsoft Visual J++ projects by importing them into Visual Cafe PDE and add power to them that you couldn't do before.

Native building and debugging native applications

You can now build Win32 applications and DLLs and debug them in Visual Cafe and add then add the DLLs to the Windows registry.

Incremental Debugging

You can fix your program's bugs in the debugger without having to recompile all the time.

Visual Cafe Database Development Edition

Use Visual Cafe dbDE to create industrial strength Java applications and connect to multiple databases with the click of a button. It's intuitive, powerful and easy to learn.

The Visual Cafe PDE includes all the standard features of Visual Cafe WDE and PDE, plus these powerful features:

Speed development with visual database tools

Visual Cafe dbDE allows you to produce powerful database applications instantly. It's visual database tools, such as the dbAware Project Wizard, walk you through the entire process of creating a database aware applet or application. In addition, you can create web forms on top of existing tables in minutes, display meta-data graphically, generate code automatically and

alter it for complex situations and choose from a library of 20 customizable database templates.

Get open database connectivity

Everything you need to connect to most databases is provided in Visual Cafe dbDE, including the dbANYWHERE Workgroup Server. The dbANYWHERE server supports JDBC and provides native support for Oracle, Informix, Sybase and Microsoft SQL Server databases, and ODBC support for over 30 databases.

Offer swift, scalable data access—affordably

The dbANYWHERE server's true three-tier architecture lets thin applications run on clients, with no client-side drivers required, and dbANYWHERE's data caching enables bandwidth flexibility. All clients need is a Java-enabled browser, so management is a snap and client software costs are reduced.

What's new in Visual Cafe 2.0

Contextual Menus

Contextual menus are pop-up menus that show commands that can act on the object you on.

ZIP archive and JAR (Java ARchive) support

Archives are supported in JDK 1.1 in two types. The .zip file, which was also supported in Java 1.0, is a basic archive format that only appends .class files together into one long file. The .JAR format allows compaction, encryption, and signing. It also extends the archive format by allowing multiple file types to be included. JAR files allow you to contain such things as sound and images that you want to be included with your compiled classes in the archive.

Open by Name

This new feature enables you to search for any .java or .class file. You can type partial names, and in the case of multiple matches, you can select the one you want to open.

Version Compatibility

In general, any applet or application that ran in JDK 1.0.2 should run correctly in Visual Cafe 2.0. Incompatibilities might be found where functionality has changed between component versions.

Of course, applets that depend on any new JDK 1.1 APIs will not work in browsers that support only 1.0.2, such as Internet Explorer 3.0, Netscape 3.0, and the alpha and pre-beta1 versions of the HotJava browser. However, generally, applets relying only on APIs defined in 1.0.2 (but compiled with the JDK 1.1 compiler) will run on 1.0.2 browsers. This backwards compatibility has not been extensively tested and cannot be guaranteed.

Project and environment management

Visual Cafe provides a host of advanced project and environment management features. The highlights are described below.

Project types

Project types are templates from which new projects are created. Templates define the Java source and HTML files that the project contains and the project's option settings. Additional project types have been included in Visual Cafe to facilitate native and database-aware development.

Project window

The Project window displays information about the project's organization and status. The Project window has been revised in this release to make it easier to control window presentation.

Event model migration utility

With the Migrate utility, you can migrate your Java files that use the JDK 1.0 event model to the JDK 1.1 event model.

Enhanced macro editing

Visual Cafe lets you define a custom keystroke sequence for many Visual Cafe editing operations and macros in your projects.

Drag-and-drop

Not only does Visual Cafe fully support drag-and-drop, but this feature has been extending to dragging and dropping JAR files across projects. JAR files may be added to and removed from a project by dragging files and folders to and from the Windows Explorer and other open projects.

Display options

You can now customize the display of the Project window using display options. This display may optionally include icons, debugging, make, project status, code and data size, file location (full source path), modification date, source's translator, immediate group container of source file, kind of each entry (for example, source file or group), and font and size. The Project window can be displayed by the group hierarchy or sorted by any of the displayed categories.

Debugger

The Visual Cafe debugger has some new features as well. In addition to full JDK 1.1 support, Visual Cafe includes incremental debugging. Incremental debugging allows you to make changes in your source code as you fix bugs without recompiling and restarting your program!

Class Browser and Editor

The Class Browser and Editor are designed with object-oriented program development in mind. The Class Browser and Editor now include support for JDK keywords.

New and enhanced Wizards

New Wizards have been added as well as enhancing existing ones. The Interaction Editor now supports JavaBeans. The new Insert Class Wizard helps you to quickly get custom and 3rd party classes into your Java programs. You can convert description files (the blueprints of your custom classes) to JavaBean info and implement your custom components with the new JDK.

LiveUpdate

LiveUpdate is a feature in many Symantec products that connects you to the Symantec Update Center and checks to see if their are updates to Visual Cafe waiting for you. If their are product updates available, LiveUpdate downloads the latest version and installs it for you so you can get back to work quickly.

Visual Cafe documentation

Visual Cafe comes with extensive documentation and online help to assist you in the process of developing in Visual Cafe. These documents are described below.

Visual Cafe Getting Started and Tour

The Visual Cafe Getting Started Guide consists of installation instructions, support information, and a tour of the main features of Visual Cafe.

You may want to work through the Tour, which takes less than an hour, before beginning work in Visual Cafe. The Tour is designed to familiarize you with the main features of Visual Cafe by guiding you through the process of building a working Java applet. It requires no previous experience with Java, is a quick way to learn Visual Cafe -- and it's fun!

Online Help

Visual Cafe has extensive online help, providing all of the procedures for building Java applets and applications. To access Visual Cafe's Help, choose Help Topics from the Help menu. The online help is also context-sensitive, which means that you can press F1 in most parts of Visual Cafe and receive information that is specific to your current activity.

You can also access information about the Visual Cafe components, and review the Java API documentation from Sun Microsystems. Information on an individual component is available by typing the name of the component in the Index tab of Help, by choosing Components Reference from the Help Contents tab, or from the Component Library by selecting a component and pressing F1.

User's Guide

This manual may be the document you turn to most frequently as you work with Visual Cafe. It contains both conceptual information as well as step-by-step procedures. It contains three parts: Developing in Visual Cafe, Professional Features, and Database Connectivity.

Part One: Developing in Visual Cafe

Part One introduces Visual Cafe and takes you through the basic process of creating an application or applet with Visual Cafe. Information in this section applies to all three Visual Cafe editions, including Visual Cafe WDE.

Part Two: Professional features

Part Two provides a complete reference to the features unique to Visual Cafe PDE.

Part Three: Database connectivity

Part Three is made up of information specific to Visual Cafe dbDE.

Conventions used in the Visual Cafe User's Guide

This book uses the following typographic conventions:

- Names of files, code fragments, resource names, class names, method names, variables, and information you type appear in `code` face. Metanames appear in *italic*.
- All numbers are decimal unless otherwise noted. Hexadecimal numbers are written with the prefix `0x` as in `0x3EFA`.
- Keys you press at the same time are shown as follows: Control-Z, Shift-Control-G. Please note that even though the letter keys are listed in uppercase, do not hold down the Shift key when executing these key combinations unless the Shift key is listed as part of the combination.

Prerequisites for using Visual Cafe

This book assumes that you know, or are learning how to program in Java. If you do not know how to program in Java, you can refer to one of the many new books that teach Java programming.

System requirements

Visual Cafe's minimum system requirements are as follows:

- Intel (compatible) 486 or higher CPU (Pentium 90 and higher recommended)
- One of the following operating systems:
  - Windows 95
  - Windows NT Workstation (version 4.0 or higher)
  - Windows NT Server (version 4.0 or higher)
- 16 megabytes RAM (for Windows 95)
  32 megabytes RAM (for Windows NT)
- 30-100 megabytes free hard disk space
- CD-ROM drive recommended
- Color monitor (SVGA recommended)

Internet connectivity

In addition to the above system requirements, it is recommended that you have Internet and Web accounts with an Internet Service Provider (ISP) in order to test your Java applets and applications on the Web.

You can also use Visual Cafe over a high-speed Internet connection, such as Ethernet, that is using TCP/IP protocol.

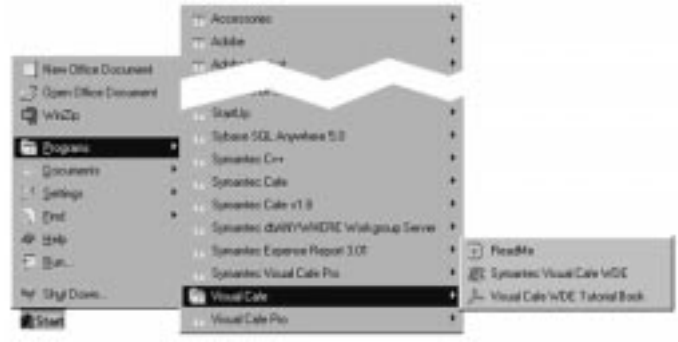
Starting Visual Cafe

As soon as you launch Visual Cafe, you can begin developing your applet or application. If you haven't installed Visual Cafe yet, follow the instructions in the Getting Started Guide that comes with the Visual Cafe package.

To start Visual Cafe:

1. Open the Start Menu.

2 Locate the Symantec Visual Cafe entry under Programs.



3 Select Visual Cafe. The Visual Cafe environment is displayed on your desktop.

4 Locate the Visual Cafe folder on your system.

5 Locate the Visual Cafe program icon and double-click on it.

The Visual Cafe environment

The Visual Cafe environment contains windows, toolbars, and editors that make developing your Java applet or application easy.

Windows

Visual Cafe windows are the areas on your screen that you use to develop your programs, including monitoring the status of your projects at any time. These windows include:

Form Designer – Drag and drop components, as well as arrange them to this main development area.

Project window – Work with the different parts of your project in this tabbed window. It's views include objects, packages, and files.

Property List – Lists and lets you control the properties for your components. You can resize, name, change visibility, assign values, and change colors and fonts of your components.

Source Editor – The Source Editor is where you add and customize source code for your project in any phase of the development cycle.

Breakpoints – Set breakpoints while debugging to validate parameters and states.

Variables – Use this window to monitor and change variables in your expressions as you debug your code.

Watch – In the Watch window, you can watch the debugger step through your code and watch it evaluate selected expressions.

Threads – Allows you to monitor and debug threads in your Java program.

Call Stack – Provides you a window to watch the debugger step through functions, methods, and classes.

Messages – Collects all the information from Visual Cafe as it is running or debugging your project.

Toolbars

Visual Cafe provides an extensive set of toolbars. Toolbars can be docked in the Visual Cafe window or floated on the screen. Docked toolbars offer easy access to your favorite components. Hide your toolbars for a clutter-free design environment if you are more comfortable using menu commands.

These toolbars are available in the Visual Cafe window:

- **Standard** – Contains buttons for working with files, printing, and copy and pasting.
- **Palette** – Contains buttons for adding components to a form.
- **Layout** – Contains buttons for arranging the placement of components on the form.
- **Views** – Contains buttons for debugging views.
- **Debug** – Contains buttons for debugging actions.
- **Workspace** – Selects the debugging or editing workspace mode.

As you change your environment within Visual Cafe, access to other functions is enabled. For example, when you work in the Class Browser or Hierarchy Editor, the Classes menu is enabled to assist you in working with classes.

Visual Cafe windows

From the Views toolbar, you can access these windows:

- Class Browser
- Hierarchy Editor
- Property List
- Breakpoints
- Variables
- Watch
- Threads
- Call Stack
- Messages

Editors

Visual Cafe has three editors for creating and managing Java projects. You can use these editors to control the development of your projects, such as editing source code and manipulating classes.

- **Source Editor** – a tool for editing source code. For more information about working with Java source code, see Chapter 4, "Working with Java Source Code."
- **Class Browser** – a three-pane window that lists all of the classes, methods and data items contained in your program. For more information about using the Class Browser, see "Using the Class Browser" on page 4-1.
- **Hierarchy editor** – provides a visual representation of the classes in your project, and their inheritance relationships. For more information about using the Hierarchy Editor, see "Navigating the Hierarchy Editor" on page 4-17.

How much Java do I need to know to use Visual Cafe?

It is possible to develop Java programs without having to write a single line of source code. There are many places to obtain Java programs, such as books, the Internet, and even your friends and colleagues. You can drop

these programs into Visual Cafe and have an applet, application, or JavaBean ready to use.

However, sometimes these programs need simple to extensive modifications and this is where real challenges to your programming expertise occur.

You should have a basic understanding of object-oriented programming, such as C++. Many of the principle s and concepts of Java are based upon those found in C++. There are also some vast differences between Java and C++ that you should know about, as well.

You should also have a basic understanding of cross-platform operating system concepts. This knowledge is useful, for example, when developing multi-threaded Java programs because all platforms handle threading differently.

If you are using the Professional Development Edition, you need to know basic Microsoft Windows programming and techniques to develop native, 32-bit applications and libraries.

If you are using the Database Development Edition, a basic understanding of Structured Query Language (SQL) and client-server models is necessary to build more complex database-aware Java programs.

Finally, a basic understanding of Hypertext Markup Language (HTML) is helpful to understand and develop the relationship between your Java applets and Web pages.

It is beyond the scope of Visual Cafe to teach Java programming, although you can learn more about Java by using Visual Cafe. You can find many excellent books and tutorials that extensively explore the Java language. You can also search the Internet with your favorite search engine and find many well-written Java tutorials and summaries, as well as find abundant resources to guide you on your way to becoming a Java developer.

You can also participate in special user interest groups, often called a “SIG”, for your favorite platforms in your local area

What’s next?

This chapter has introduced you to the Visual Cafe integrated debugging and development environment and the main interface. Before you actually begin developing your Java applet or application, take a look at Chapter 2,

"Developing in Visual Cafe", which provides information on basic features of Visual Cafe and includes an overview of creating an applet and application. Remember, if you are new to Java development, the Visual Cafe Tour is a great place to start.

CHAPTER 2

Developing in Visual Cafe

Visual Cafe helps you find and fix development problems, reference the rules of the Java language, automatically create and update source code, and optimize the entire process of creating Java applets and applications. Visual Cafe facilitates these tasks by providing an Integrated Development and Debugging Environment (IDDE). It is possible to design, develop, and build an applet or application without having to write a single line of source code. In other words, Visual Cafe tools provide everything you need to create and develop a Java program.

Putting Visual Cafe to work

When a development environment is said to be integrated, this means that the tools in the environment work together. For example, the compiler might find an error in the source code. In addition to displaying the error, and with a double click on the highlighted error, Visual Cafe opens the source file in the text editor and jumps to the exact line in the source code where the error occurred.A large part of applet and application development involves adding and arranging components on forms, frames, and applets. Visual Cafe provides tools to make designing forms a simple process.

After you have started Visual Cafe, you begin creating Java programs using Visual Cafe's many powerful tools. These tools include:

- **Toolbar** the central control panel for Visual Cafe.
- **Form Editor** the primary tool you use to create your programs by dragging and dropping components.
- **Component Palette** the central place where often used components can be accessed.

- **Property List** controls the location, behavior, and look of the components in your program.
- **Component Library** the repository for all components, including your custom ones.
- **Class Browser and Hierarchy Editor** allows you to view objects with a hierarchical display as well as provide access to your source code.
- **Class Browser** allows you to view objects with a hierarchical display and provide access to your source code.
- **Insert Class Wizard** helps you insert the right class in the right place in your project.
- **Interaction Wizard** adds functionality to your components by creating interactions for them.

Understanding Visual Cafe components

There are three types of components in Visual Cafe: visual objects, non-visual objects, and containers. A visual component is visible at runtime and lets users interact with your applet or application; it has a screen position, a size, and a foreground and background color. Examples of visual components are forms, applets, and buttons. An invisible component is not visible at runtime, such as a Timer, or has different display properties, such as a MenuBar. Some components can contain other components, such as an application window containing a button; these components are called containers.

With Visual Cafe, you add visual components to your forms and frames to assemble applets and applications. Visual components can accept input from a user and perform specific actions. Visual components can also be used to display the results of an action. You can use standard graphical user interface (GUI) techniques for dragging and dropping components into and among other components. Visual Cafe also has unique drag-and-drop behavior in the Project and Source windows.

Visual components generally have the following attributes:

A set of properties

Governs how components display and behave. Properties are accessible from the Property List.

A visual element

Defines the appearance of a component at run-time. Shown as an icon in the Project window, and directly editable by double-clicking and opening the component.

One or more interactions

The ability to quickly build a relationship between two components is called an interaction. As you create interactions by connecting components, Visual Cafe automatically generates code for the relationship, allowing you to assemble interactive applets and applications without writing code.

Interactions are not an integral part of all components.

A set of event method(s)

In Visual Cafe, interactions are implemented as methods and imply an event notification. This Java code is accessible from the Source Window.

Each of these component attributes has its own editor, from the Form Designer to the Interaction Editor, making it easy to manipulate visual components. You create cross-platform Java applets and applications by first designing the user-interface components, and then using the various Visual Cafe tools to automate the process of deriving classes, creating interactions, coordinating text messages, and mapping functions to visual components and messages. You can do most of these tasks by setting properties to define, refine, and control the appearance and behavior of your visual components.

Forms hold your Java program together

Forms are containers for components that allow a user to interact with your program. You use forms when creating applets. You use frames, another type of form, when creating a stand-alone application.

An example of an application using a frame is the JavaPad application, included on your Visual Cafe CD. Frames generally include toolbars, menubars, and other windows.

Visual Cafe provides several tools to help you create forms, including the Form Designer. Examples of forms are applets, windows, and dialog boxes.

Projects keep your work together

When developing an applet or application in Visual Cafe, you work mainly with projects. A project is a collection of files that make up your Java applet or application. You create a project to manage and organize these files.

When you open a new Visual Cafe project, all the default files required to begin development are created and included as part of your project. In addition, project elements may include HTML files, frames, and forms.

When you save your work, Visual Cafe saves the entire project to the project folder as a .vep file along with all the changes and other files you may have added to your project.

Using workspaces to customize your work environment

A large number of views are available in the Visual Cafe environment. Visual Cafe lets you save a configuration of windows as a unique workspace. A workspace provides a convenient way to switch from one screen layout view to another.

A workspace is a saved arrangement of windows. Because the various tools in Visual Cafe are displayed in many individual windows, workspaces are used to group together the windows that have related functions. Workspaces provide a means of organizing the multiple windows of the development environment into logical groups. For example, when you edit source code you could use a workspace that displays the text editor, the Messages window, and the Project window. When you are debugging your program, you could use another workspace that displays windows for the different debugging tools.

Debugging with Visual Cafe

One of the most powerful tools in the Visual Cafe environment is the integrated Debugger. The Debugger allows you to watch your programs execute line by line. As your program executes, you can observe the various components of the program to see how they are behaving. By using the Debugger, you can monitor:

- The values stored in variables
- Which methods are being called
- The order in which program events occur

When you first create your project, you specify the type of compiler you want to use. Visual Cafe provides two compilers: Symantec's Just-In-Time compiler and Sun Microsystems' Java compiler. These two compilers are detailed in the following sections.

Symantec's Just-in-time compiler and the Visual Cafe environment

The Symantec Just-in-time compiler that comes with Visual Cafe was written specifically for the Java language and is generally faster than Sun's command line compiler. Instead of typing a string of commands at the command prompt, you click on a single icon to compile and execute a program.

Visual Cafe displays information in the Windows environment instead of at the DOS command line. Visual Cafe also provides useful, meaningful messages indicating how the compiling process is progressing.

In addition, Visual Cafe provides many GUI based development tools not found in the JDK. Visual Cafe's graphical development environment makes Java programming easier and faster.

Sun Microsystems' Java Compiler and JDK

When Sun Microsystems developed Java, they also created a compiler to convert Java source code into .class files. The Java compiler developed by Sun is called "javac.exe", and is run from a DOS command line. To compile a Java program you need to run javac.exe, passing the name of the source file, along with any other required parameters.

Sun's Java compiler also comes with the Java Developers Kit, or JDK. The JDK is a collection of tools to help compile, debug, and test Java programs. The JDK, like Sun's compiler, uses a command line interface.

Overview of creating a Java program

Developing a Visual Cafe applet or application happens in two stages: designing the program and developing it. In the first stage, you design, create, and implement the graphic user interface (GUI) of your program. You also make a very simplistic arrangement of all the components needed. The second stage is when you bring your project to life by adding and modifying source code, debugging, re-designing, and testing your

project. If you are creating database-aware applets or applications, there are several additional steps you must do as part of the design stage.

Overview of creating an applet

When you start up Visual Cafe, Visual Cafe creates all the project files required for a Visual Cafe applet or application, including source files and the Visual Cafe project file. Visual Cafe then loads these project files, and you can immediately begin working on your project.

To create an applet:

1. Create a project with an applet template. Visual Cafe provides you with several templates ready for your use.

   See Chapter 3, “Working with projects and workspaces.”

2. Design the user interface by adding forms and components to your project, customize the component properties, and create component interactions.

   This process is described in detail in Chapter 5, “Including Visual Components.”

3. If necessary, modify the Java source code.

   See Chapter 4, “Working with Java Source Code.”

4. Set project options.

   See Chapter 3, “Working with projects and workspaces.”

5. Run the applet in Visual Cafe.

   This step is explained further in Chapter 6, “Compiling, running, and deploying your program.”

6. If needed, debug the applet.

   Chapter 8, “Debugging Java Programs”, contains detailed information on debugging in Visual Cafe.

7. Add the applet to your HTML page. You may also want to Test run the HTML page on your local machine, then across the intranet or Internet.

   See Chapter 6, “Compiling, running, and deploying your program”, for more details.

8. Deploy the applet.

   Deploying is described in Chapter 6, “Compiling, running, and deploying your program.”

Overview of creating an application

Creating an application in Visual Cafe is very similar to creating an applet.

To create an application:

1. Create a project with an application template. Visual Cafe provides you with several templates ready for your use.

   See Chapter 3, "Working with projects and workspaces."

2. Follow steps 2 through 4 of creating an applet as explained in the section above.

   These steps are detailed in the same chapters as referenced.

3. Test run the application in Visual Cafe.

   Compiling and running a project is explained in Chapter 6, "Compiling, running, and deploying your program."

4. If needed, debug the application. You should also test the application outside of Visual Cafe once its been debugged.

   Chapter 8, "Debugging Java Programs", contains detailed information on debugging in Visual Cafe.

5. Deploy the application.

   Deploying is described in Chapter 6, "Compiling, running, and deploying your program."

Overview of creating a dbAWARE applet or application

A database-aware applet or application can be used to store and retrieve data from Web-based forms.

To create a database-aware applet or application:

1. Start dbANYWHERE.

   See Chapter 12, "Developing a dbAWARE project", for details.

   Note: There are several configurations which have to do with the location of your dbAWARE project, dbANYWHERE, and your database. Before you start the dbANYWHERE application, it must be configured to suit your needs. The configuration options are explained in Chapter 13, "Using dbANYWHERE.".

   ---

2. Create your database by using the tools shipped with Visual Cafe or by using proprietary database tools.

A brief overview of how to do this is provided in Chapter 12, "Developing a dbAWARE project."

3 Identify the database you want to use for the project's data source by using the ODBC administrator tool or the dbANYWHERE Data Source tool. These tools are installed along with Visual Cafe and appear on the Windows Start menu.

See Chapter 12, "Developing a dbAWARE project." for more information.

4 Create the dbAWARE project using the dbAWARE Project Wizard.

See "Using the dbAWARE project wizard," Chapter , for details.

5 If you are creating a dbAWARE applet, follow steps 2 through 8 as explained in the Overview of Creating an Applet section above.

These steps are detailed in the same chapters as referenced.

6 If you are creating a dbAWARE application, follow steps 2 through 5 as described in the Overview of Creating an Application section above.

These steps are detailed in the same chapters as referenced.

CHAPTER 3

Working with projects and workspaces

Visual Cafe provides an easy yet sophisticated system to manage the collection of files that make up a project. A project is the primary component of Visual Cafe. A workspace is a saved arrangement of windows. Because the various tools in Visual Cafe are displayed in many individual windows, workspaces are used to group together the windows that have related functions. For more information about workspaces, see "Use Workspaces to customize your work environment" on page 3-55.

What is a project?

A project is a set of files that, when assembled by Visual Cafe, produce an application or applet. In a typical project, there are source code files, class files, and documentation files. A single project generally is used to create a single target.

Note: The application or applet that Visual Cafe builds from a project is referred to as the project's *target.*

A project is the starting point of every Java applet and application created in Visual Cafe. The Project window shows each item in a project. Visual Cafe lets you choose among three ways to view your project:

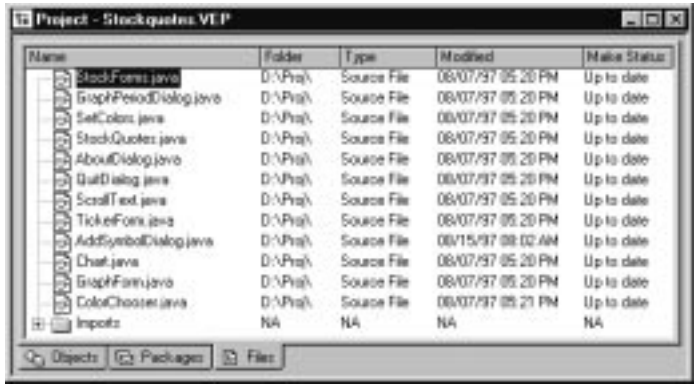
- a list of objects (and, optionally, HTML files)
- a list of packages
- a list of files

Projects provide vital organization to your programs, particularly as they grow more complex.

The project file and the project folder

The central element of a project is the *project file*. The project file contains all information necessary for management of the project, such as locations of the project entries, and additional information such as compiler options and browser data.

Projects speed development by letting you recompile only the source files that have changed since the last time the project was built. Visual Cafe manages the project for you by automatically analyzing the dependencies of the source files and updating the project information each time you build the project. So, if you make changes to only two files in the project shown here, and then direct Visual Cafe to build the project, it recompiles only those two files.



In general, most project entries specific to a project, together with the project file, are kept in a folder referred to as the *project folder*. However, a project's entries do not all have to reside in the project folder. In addition, a project may include entries that are located in the project folder of another project; this permits sharing of code.

An *entry* is the name of a given component or other object. If you add an entry to a project, Visual Cafe adds the corresponding file(s) to the project. A given entry might have a `.java` file initially, but building generates a `.class` file from the `.java` file. Both the `.java` and `.class` files are associated with the entry.

When you organize a target's files in this manner (as a project), Visual Cafe can assume full management responsibility. In contrast to traditional "make" systems, this strategy frees you from the bookkeeping involved in accessing project entries and building the target. Because Visual Cafe keeps track of all project entries of the project, the features of Visual Cafe are smoothly integrated. For example, if an error occurs during compilation, you can click to open a Source window containing the source code with the questionable line of code highlighted.

Also, Visual Cafe automatically determines those project entries that need to be rebuilt following changes to any project entry(ies).

Basic concepts of working with a project

Creating an application or applet in Visual Cafe is a multistep process involving Visual Cafe, the Visual Cafe Debugger, and (for applets only) the AppletViewer.

Starting a new application or applet with Visual Cafe begins with the creation of a project.

The names of the files needed to create one or more Java applets, applications, or related Web pages containing applets are entered in a Visual Cafe project. For example, a project could include any of the following elements:

- an animation applet
- a word processing application
- the Web pages and applets that make up an entire Web site.

Before you can construct an applet or application with Visual Cafe, you need to create a new project, specifying one of the predefined project templates. Then the Project window opens with the project name in the title bar. Each Java program created in Visual Cafe must be part of a project.

By default, each new project is named "Untitled1" until you save it with a name.

Looking at the Objects view

If you click the Objects tab in the Project window, you see the components in your project and any HTML files that have been added to the project. Java *components* are like C++ controls: they are user interface elements,

such as windows, menus, buttons, lists, and so on. If the tabs in the window show icons but no text, open the window wider and the text will also appear on the tabs.

Some components can contain other components, such as an application window containing a button. A that contains other components is called a *container*. In the Project window, the components in a container appear subordinate to the container, like a file system display shows files subordinate to their folders. The containers at the top level are separate Java files in your project (called Visual Cafe forms), while the components in the containers are Java code within the container Java file.

All Visual Cafe components are organized in the Visual Cafe *Component Library* (also known as the *Component Palette*).

The Objects view is the view many people use most often during Java program development with Visual Cafe. With it, you can keep track of the components your project contains, as well as open a variety of editors. In addition, one way to add a standard Java component or a Visual Cafe component to your project is to drag it from the Component Palette to the Project window. Then an instance of the component class, an object, appears. See "Viewing the components and HTML files in a project" on page 3-29, and "Dragging-and-dropping into the Project window" on page 3-24 for more information.

Project contents

The different types of entries that can be included in a project are source files, projects, documentation files, and groups. Each type is handled

differently by Visual Cafe when the target is built and when the entry is accessed from within Visual Cafe. Visual Cafe uses filename extensions to identify the various types of entries.

Source files

Visual Cafe can process Java (`.java`), class (`.class`), and documentation source files. The Java and documentation files are text files.

Projects

Visual Cafe also allows you to include other projects in a project. Including projects lets you group together sets of related entries and access all included projects' entries within Visual Cafe. Included projects are built when the project containing them is built. Thus, you can develop a suite of applications by having one project for each application and one additional project that includes all the individual projects; the entire suite of applications can then be built with one command.

Documentation files

You can add any documentation files to the project to make them readily accessible during development. These files will neither, however, be included in the final target nor be involved in any way with building. By including these files directly in the project, you make them always available for reference and modification. You can use any application to create these files as long as they are saved as text files.

Typical files you might want to include are informational files on the design of your project.

Groups

To better organize entries within your project, Visual Cafe allows you to create groups. Groups are similar in concept to folders in Windows. By placing your project entries into groups, you make it easier to locate individual entries, especially with a large project. Like folders, groups can be nested. The placement of project entries into groups has no effect on the final target. Further, the location of an entry in a group has no bearing on the file's location on the disk.

Organizing files and folders

Installing Visual Cafe establishes a specific plan of folder organization. This plan is set up to allow Visual Cafe to quickly and unambiguously locate your project's entries. Specifically, Visual Cafe looks for your project's entries in the folders of two paths, the *system path* and the *project path*.

The system path

Components that will be used in many projects should be placed in folders of the system path. In the folder where Visual Cafe is installed (usually the `Visual Cafe for Windows` folder), there is a `Java Libraries` folder. Within that folder, there is a `classes` folder. The default system path is that `classes` folder along with all its subfolders. You can add paths to the system path, and you can modify the system path, by using the Search Paths option from the Project Options window.

The project path

Components that are specific to a particular project belong in its *project path*. The project folder, along with all subfolders it contains, is the default project path. The project folder is the folder that contains the project file. You can add paths to the project path, and you can modify the project path, by using the Search Directories option from the Project Options window.

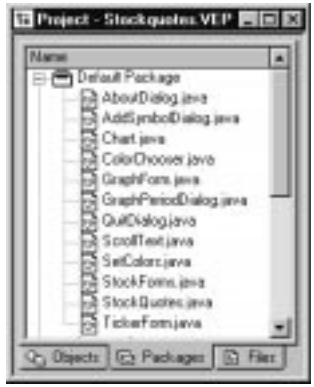
Typically, a project file resides in a project folder along with all files specific to the particular project. The folder may also contain subfolders. Setting up your project's entries in this way helps reduce the time it takes Visual Cafe to search for files, and reduces the likelihood of confusion due to duplicate file names. You can expect to have many project folders.

When you first add a file to a project, Visual Cafe notes the tree (folder hierarchy) to which the file belongs. Thus, you can move files in and out of folders and create and rename folders without having to tell Visual Cafe exactly where the files are located. If you move files later on, Visual Cafe first looks in this tree.

Looking at the Packages view

A Java package is a group of related classes that can be used by programs that import that package or any file in that package. It is similar to a C

library. The Packages view always shows the standard Java packages that Java programs require.



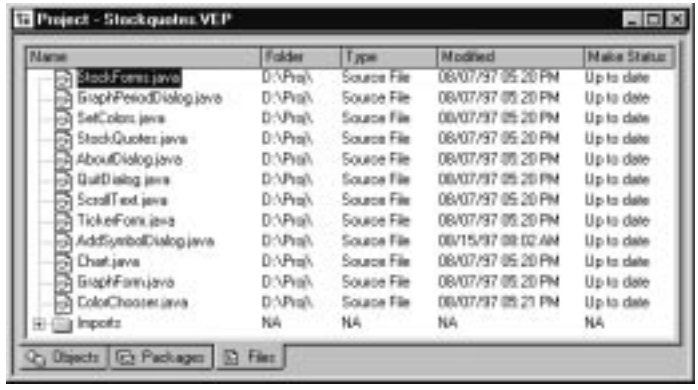
When components are added to a project, a Default Package, containing the `.java` files for those objects, appears. If you add a Visual Cafe component, the Symantec package also displays in the Package view; it contains the Java source file for each Visual Cafe component you include (specifically, it contains the source code for the component class that your project object is based on).

You can also make other packages available to Visual Cafe projects by adding them to the Visual Cafe class path. Then if you add an import statement or use part of the package in your Java source code, the package appears in the Packages view. See "Adding packages to Visual Cafe" on page 3-34.

Usually, only advanced Java programmers are interested in this view. See "Viewing the packages in a project" on page 3-33 for more information.

Looking at the Files view

The Files view lists all of the files contained in your project. At the top level are the files for the top-level components in your project, as well as any HTML files you may have added to the project.



In the `Imports` folder are all of the Java source files for the packages your project uses. Whereas in the Packages view the files are listed by package, in the `Imports` folder the files are listed alphabetically.

As with the Packages view, usually only advanced Java programmers are interested in the Files view. See "Viewing the files in a project" on page 3-26 for more information.

Creating projects by using templates

This section describes the use of project types, and the templates that are contained in each, when creating a project. It discusses the different project types and how to create projects based upon them.

This section assumes you've already correctly installed Visual Cafe. Before working with Visual Cafe, you should create a common folder to contain your specific project folders. You can name this common folder anything you like, such as `My Projects`, and you can place it anywhere you like as long as it is outside of the system path (see the previous section for more information).

Choosing the project template

When creating a new project, you must select a project template for it. Project templates determine those project entries that are to be initially added to a project and those configuration options that are to be initially provided. Using project templates reduces the amount of overhead involved with creating projects. You can also create your own project types, as described in "Creating a project template and adding it to the library" section in this chapter.

The following components are in the standard Visual Cafe project templates:

- Empty Project – No objects.
- Basic Java Bean – This template contains all the basic files you need to get started in building JavaBeans and deploying them.
- Basic Application – A Frame, AboutDialog, and QuitDialog component. The Frame container is a special kind of window; this one has been set up as a main application window. In this template, it contains a menu bar and an Open File dialog box, which is a standard Java component. The About and Quit dialog boxes are Visual Cafe components that you can customize like templates; they are tied to the About and Exit menu items.
- Basic Applet – An Applet component. The Applet container is a type of Panel component that is designed to appear in an HTML file (such as a Web page) viewed in a Web browser (such as Netscape Navigator).

Professional Development Edition

The Visual Cafe Professional Development Edition adds two more component templates:

- Basic Win32 Dynamic Link Library – This template specifies that the program is a native DLL.
- Basic Win32 GUI Application – This template specifies that the program is a native, stand-alone executable.
- Basic Win32 Console Application – Has no GUI features uses console.

Database Development Edition

The Visual Cafe Database Development Edition adds two more component templates beyond those of the Professional Development Edition:

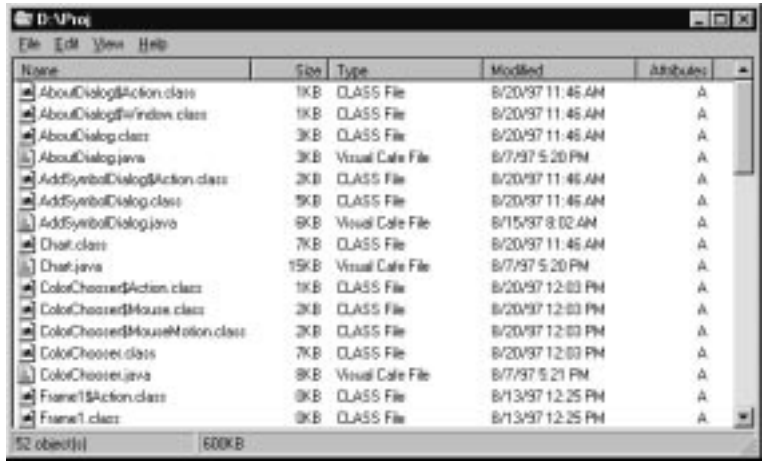
- dbAWARE Template Wizard – This template supplies everything you need to select or build a table in a specified data source.
- dbAWARE Project Wizard – This wizard guides you step by step in creating a project framework for developing Java projects that use database connectivity.

You can also create your own project templates. See "Creating projects by using templates" on page 3-8 for more information.

Constructing a logically arranged project folder

The first time you save a project, the files it contains are stored in the folder you specify. For efficient project management, you should store all the files of a given project in a separate folder dedicated to that project. This folder

is called the *project folder*. Doing so will make deploying your applets, applications, and Web pages much easier.



The Java and Symantec packages are automatically available during development with Visual Cafe. These files are not added to your project when you save it the first time; there are a number of ways you can make imports from these packages available to your applets and applications during deployment.

If your project has a lot of files, you can group the files in folders subordinate to the project folder. For example, if your project has many graphics files, you could store these files in a separate folder subordinate to the project folder. Remember that when you create your Java programs and HTML files, you specify where they should look for certain files, such as graphics files. So whenever you move files in the project folder, you must be aware of any dependencies that your Java programs and HTML files have on file locations.

For more information on deployment, for more details, see “Deploying Java programs” on page 6-7.

Looking at the files in the project folder

Visual Cafe creates several files that contain information about your project and stores them in the project folder:

Extension	Description
`.vpj`	The Visual Cafe project
`.vep`	Visual Cafe project options and file list
`.ve2`	Secondary project information
`.cdb`	Compiled database that Visual Cafe uses to track compilation dependencies (created after compilation)

These files are not needed for deployment. The name of the `.vep` file appears in the title bar of the Project window; the rest of the project files do not appear in the Project window.

Note: In Visual Cafe 1.0, you opened a project by opening the `.vpj` file. In Visual Cafe 2.0, you open a project by opening the `.vep` file.

Visual Cafe can also create these files and store them in the project folder:

Extension	Description
`.java`	A Java source file, such as for an applet
`.class`	A compiled version of a Java source file
`.html`	An HTML file

Double-clicking a `.java` or `.html` file in the Project window opens that file in a Visual Cafe editor. For deploying applications, you need the compiled version of your Java files (the `.class` files). For deploying applets, you need both the `.class` files and any HTML files you want to use for your Web pages. In addition, you also need your graphics files.

The Java source files (`.java`) and compiled Java files (`.class`) have the same file names, but different extensions. You see only the `.java` files in the Project window, because they are used for development, while `.class` files are used for deployment.

Adding files to a project

You can add `.html`, `.java`, and `.vep` project files to a project. If you have Visual Cafe Professional Development edition, you can add DLLs as well.

The only way an HTML file will appear in the Project window is if you manually add it. You can add HTML files to the project as a helpful organizational tool, but it is not required.

You can also add Java source (`.java`) files. For example, if you wanted to import a Java source file for an applet that you created in a product other than Visual Cafe, you could place the file in a project folder, then add the `.java` file to the Project window. Visual Cafe will attempt to translate the file into its visual environment.

If you add the project (`.vep`) file of a project to the current, open project, the added project becomes a subproject of the open project it resides in.

You can add ZIP files and DLLs to a project and, from your Java code, import any packages or class files it contains. Note that packages in a ZIP file or DLL do not appear in the Packages view of the Project window.

Remember that you should store all the files of a given project in a separate folder dedicated to that project. Doing so will make deploying your applets, applications, and Web pages much easier. See "Working with components in a project" on page 3-29 for more information.

Although Visual Cafe automatically creates many Java source files for you in its visual environment, in some cases you may want to add `.java` files directly. For example, if you wanted to import a Java source file for an applet that you created in a product other than Visual Cafe, you could place the file in a project folder, then add the `.java` file to the Project window. Visual Cafe will attempt to translate the file into its visual environment.

For more information, see "Adding an existing file to a project" on page 3-27 and "Adding a new file to a project" on page 3-26.

To add files to the Visual Cafe environment:

1. From the Insert menu, choose Files Into Project.

   The Project Files dialog box appears.
2. Select the file(s) that you want to add.
3. Click Add or Add All, as appropriate.

4 Click OK.

The file(s) appear in the Project window.

To add files from the Windows Explorer or other file system window:

Drag Java or HTML files from the Explorer into the Project window.

Working on multiple projects

You can open multiple projects simultaneously; they are displayed in separate Project windows. This feature helps you to easily navigate between projects.

Sharing files across multiple projects

For efficient project management, you should store all the files of a given project in a separate folder dedicated to that project. Doing so will make deploying your applets, applications, and Web pages much easier.

You can reuse a file if it does not contain code that was automatically generated by Visual Cafe. Just add the file to the open project. For more information, see "Adding an existing file to a project" on page 3-27.

Any file that does contain code that was automatically generated by Visual Cafe should not be shared between multiple projects. The file can change in one project, causing version problems in any other projects it belongs to. Instead, you have a number of options. For example, you can copy the file into the new project folder and add it to the project, create a new project template tailored to your requirements, add your own custom components to the Component Library, cut and paste Java code, and so on. For more information, see "Copying a file of a project" on page 3-28 and "Creating a project template and adding it to the library" on page 3-54.

Note: File names are relative to the current project. For example, if you add a file called `foo.java` and it resides in a subfolder (called `foo1`) of the project folder, the file name is stored as `foo1\foo.java` and is not fully qualified. If `foo1` was a folder at the same level as the project folder, the file would be stored as `..\foo1\foo.java`.

Compiling projects

Projects speed development by compiling only the source files that have changed since the last time the project was built. Visual Cafe manages the project for you by automatically analyzing the dependencies of the source files and updating the project information each time you build the project.

Defining project options

You can customize your project by setting project options such as the project release type, runtime arguments, compiler settings, and search folders. See "Setting compiler options for a project" on page 3-45.

Beginning to work with a project

You can create an entirely new project, open an existing project, or derive a new project from an existing one by saving or renaming or copying. If you want to work on an existing project that was created under release 1.0, you must migrate it to release 2.0 first.

Creating a new project

The files needed to create one or more Java applets, applications, or related Web pages containing applets are stored in a Visual Cafe project. A project also helps you manage and organize your files as you develop your Java programs.

To create a new project:

1. From the File menu, choose New Project.

   The New Project dialog box appears.

2. Select the project template you want as the base for the new project.

   The default template is indicated by an asterisk. You can easily change the default template by selecting a template and clicking Set Default.

3. Click OK.

   A new Project window opens with the selected template loaded. All objects in the template are added to the project. Only one project appears in a Project window.

4. To save the project, from the File menu, choose Save As.

   The Save As dialog box appears.

The project and all of the files contained within it should be in the same folder. For easier project management, you should save each project in its own folder.

5 Select a folder and type the project name in the File name field, then click Save.

The Visual Cafe project file must have the extension `.vep`. If you type just the first part of the file name, Visual Cafe will add the `.vep` extension for you.

Note: The first time you save a project, all of the files it contains are saved to the folder you specify. After you save a project once, saving just the project does not save other files, such as applets. From the File menu, choose Save All to save all files in a project.

Opening an existing project

To open an existing project from within Visual Cafe:

1 From the File menu, choose Open.

The Open dialog box appears.

2 Navigate to the project folder.

3 Make sure Visual Cafe Project is shown in the Files of type field.

The projects display.

4 Select a project from the list.

5 Click Open.

The project opens with the last saved window configuration.

To open an existing project from the Windows Explorer or other file system window:

Double-click the project file, which has the extension `.vep`.

Note: In Visual Cafe 1.0, you opened a project by opening the `.vpj` file. In Visual Cafe 2.0, you do so by opening the `.vep` file.

Saving, renaming, and copying a project

The first time you save a project, all of the files it contains are saved to the folder you specify. After you save a project once, saving just the project saves the project files only, not other files, such as applets. From the File menu, choose Save All to save all files within a project.

For efficient project management, you should store all the files of a given project in a separate folder dedicated to that project. Doing so will make deploying your applets, applications, and Web pages much easier. If your project has a lot of files, you can group the files in folders subordinate to the project folder.

Adding files to and removing files from projects

The project file contains various project entries that are added by default when the project is created. The specific project type that you choose determines the default entries that are added to the project file.

To add new project entries to a project file:

1 From the Insert menu, choose Files into Project.

The **Insert Files** dialog box appears.

You use the Insert Files dialog box to add any types of files to your project. Note that only those files that have not previously been added to the project are listed in the upper scrolling list.

2. Choose either Net Files (`*.java`,`*.html`) or All Files (`*.*`) from the Files of Type pop-up menu.

   If Net Files is chosen, only files in the project that have the specified extensions are displayed in the upper scrolling list. If All Files is chosen, all files are listed.

3. Navigate to the appropriate folder and either double-click the name of the file in the upper scrolling list or select the name and click Add.
4. The name of the file is added to the lower scrolling list.
5. Repeat step 3 for each additional file you want to add to the project.
6. When you have specified all the files you want to add to the project, click OK.

The files are added to the project and their names are displayed in the Project window.

To save a new project

1. Activate the Project window, then from the File menu, choose Save As. The Save As dialog box appears.
2. Select a folder and type the Save As field
3. The Visual Cafe project file must have the extension `.vep`. If you type just the first part of the file name, Visual Cafe will add the `.vep` extension for you.
4. Click Save. The new file name appears in the title bar.

To remove project files:

Choose Remove "*filename*" from the Project menu, where *filename* is the name of the selected project entry. You can select multiple project entries in the Project window, in which case the menu item is titled Remove Selected Items.

The project entries are removed from the project and their names are removed from the Project window.

To remove project entries using drag and drop, drag the entries from the Project window to the Trash. This only removes the entries from the project; it does *not* delete the files.

Creating groups

A group has the same relation to Visual Cafe as a folder has to the Windows Explorer: it represents a collection of related files. Like folders, groups can be nested. Groups are added to the Component Library.

As you add project entries to a project, you should create new groups to help keep these entries organized and help you locate individual entries.

To create a new group:

1. Choose Window, then Component Library from the Visual Cafe file menu.
2. Choose Group from the Insert menu to add a new group to your project.
3. A new folder appears in the Component Library.
4. Enter the name of the new group and click OK.

   The new group is displayed in the Component Library.

Saving only the project file

To save only the project and not its files:

Activate the Project window, then from the File menu, choose Save.

Only the project file is saved, not files in a project, such as applets.

If you are adding a file or set of files to the group using the Files into Project command in the Insert menu, be sure the group is first highlighted in the Project window.

To display the contents of a group, click the positive sign located to the left of the group name.

To hide the contents of a group, click the negative sign.

It is also possible to create groups by dragging folders to the Project window. Entire group hierarchies can be added in this process. All the files within a folder are added to the group corresponding to the folder in which they are located. Note that the same rules of filtering apply as when dragging and dropping individual files.

To add a hierarchy of folders and files, drag the folder icon containing the hierarchy from the Windows Explorer to the Project window.

The folder hierarchy is used only as a template for establishing groups and the locations of project entries within these groups. Project entry files can be located anywhere in the project tree, regardless of the group in which the entry resides.

Working with multiple projects

Visual Cafe allows you to have multiple projects open at the same time. One of the currently open projects is designated as the main project. When only one project is open, it is automatically designated as the main project. You can specify the project that should be the main project using the Switch Main Project submenu in the Project menu. The main project's name is listed with a checkmark. To designate a project as the main project, choose its name from the Switch Main Project submenu. If the project you choose is not open, it is opened automatically.

When working with several projects at the same time, it is important to know the project that will be affected by project-related commands you might choose. Visual Cafe applies the following rules to determine the project that will be affected:

1. If the frontmost window is a Project window, the command affects the project to which the window belongs.
2. If the frontmost window is not a Project window, the command affects the main project.

Viewing active projects

The active project has its name in the Visual Cafe title bar. The Class Browser is also project-dependent. If you have a Class Browser open for each open project, you can identify which project is active by looking at the Visual Cafe title bar for each Class Browser window you select.

This feature is helpful when viewing .java files. For example, if you open two or more .java files that are in two or more different projects, look at the Visual Cafe title bar to see which project is associated with them. Being able to tell which .java file belongs to which project is helpful when debugging, for example, because when you set breakpoints, they are associated with that project.

To save all the files of a project:

Activate the Project window, then from the File menu, choose Save All.

project files and all files in the project are saved.

To save one file of a project:

After you save a project once, you can save a file within the project separately from other files.

1. Open the file in an editor and activate that window.

   Remember that a top-level container in the Objects view is associated with a Java source file.
2. From the File menu, choose Save.

   The Save menu item is enabled only if there are changes to save.

To save and rename a project:

1. Activate the Project window, then from the File menu, choose Save As.

   The Save As dialog box appears.

2 Type the project name in the File name field, then click Save.

The Visual Cafe project file must have the extension `.vep`. If you type just the first part of the file name, Visual Cafe will add the `.vep` extension for you.

Note: In Visual Cafe 1.0, you opened a project by opening the `.vpj` file. In Visual Cafe 2.0, you open a project by opening the `.vep` file.

---

3 Using operating system tools, delete from the project folder the project files that have the old name and the extensions `.vep`, `.vpj`, `.ve2`, and `.cdb` (if present).

To copy a project:

1 If Visual Cafe is running, make sure that the Project window for the project is closed.

You cannot copy files while they are in use.

2 In Windows, copy the files in the project and paste them in a new folder, preserving the folder structure.

3 Drag the files and drop them where you want to copy them.

4 If all of your files are in one project folder, you could just copy the project folder, then rename it.

To delete a project:

1 If Visual Cafe is running, make sure that the Project window for the project is closed.

You cannot delete files while they are in use.

2 From the Windows operating system, delete the project folder and all files in the project (except for imports).

Migration from Visual Cafe 1.0 to Visual Cafe 2.0

There are two aspects to migrating from Visual Cafe 1.0 to 2.0: migrating the project, and migrating to the new event model.

Migrating a project from version 1.0 to 2.0

In Visual Cafe 1.0, you opened a project by opening the `.vpj` file. In Visual Cafe 2.0, you open a project by opening the `.vep` file.

Visual Cafe 2.0 project files (the `.vep`, `.vpj`, `.ve2`, and `.cdb` files) are not backward-compatible with Visual Cafe 1.0 project files. If you save a 1.0 project in the 2.0 software, you cannot convert the project back to 1.0. So you might want to save your 1.0 projects under a different name in the same folder. That way you will still have your 1.0 project files.

If you create an applet using JDK 1.1, the Web browser must support this version of the JDK or the applet will not run in the browser.

Migrating Java source files from the JDK 1.0 to the JDK 1.1 event model

Visual Cafe offers a utility that automatically converts a Java source file from the JDK 1.0 event model to the JDK 1.1 event model.

To migrate a project file:

1. Open the file in the Source window.
2. From the Project menu, choose Migrate.

   Visual Cafe parses the `handleEvent` and `action` methods. From these methods, new code is generated to handle the events in the JDK 1.1 event model. This includes generating the required listeners and adapters, and registering the listeners. For menus with menuItems created as quoted literals, such as `menu1.add("Open")`, a MenuItem object is created to support the new event model.
3. Look over your code and make modifications as needed. For example, you might have code that you need to move from the `handleEvent` and `action` methods.

Closing a project

When a project closes, all windows associated with the project close (except the Property List, which clears). If any Build Error, Search Results, or Class Browser windows are open for that project, they are closed along with the Project window.

To close the active project:

From the File menu, choose Close while the Project window is active.

You are prompted to save any unsaved changes. Visual Cafe remains open so you can work on other projects.

To close all projects and exit Visual Cafe:

From the File menu, choose Exit.

You are prompted to save any unsaved changes. The project and Visual Cafe windows close.

If any Message, Search Results, or Class Browser windows are open for that project, they are closed along with the Project window.

Using the Project window

This section describes how you control project behavior, such as setting options for building and running an applet, for configuring Java applications and applets, libraries, for compiling, and for project display.

The Project window helps you to quickly edit the items in a project. To access other editor windows from the Project window:

Double-click…	To get this editor…
component	Form Designer
menu	Menu Designer
Java or HTML file	Source window

- Components and menus are displayed in the Objects view.
- Java files are displayed in the Packages and Files view.
- HTML files are displayed in the Objects and Files view.

Dragging-and-dropping into the Project window

You can use standard graphical user interface (GUI) techniques for dragging and dropping components and files into the Objects view of the Project window.

When copying or moving components, if a box appears over a container in the Project window, the component is inserted within the container. If a line appears under a component, the new component appears after the component that is underlined. A + appears over the cursor when a copy operation is being performed.

Containers are placed at the top level or in another container, depending on where you drag the component or on the characteristics of the component. Containers at the top level mean a new `.java` file is added to your project; components added to a container mean that code is generated in the `.java` file for the top-level container.

Visual Cafe does not allow inappropriate copies and moves, such as dropping a component into a container when that component must be at the top level.

Drag from…	Into…	Result…
Component Library or Palette	Project window	Copies the component to the new project (in other words, instantiates the component).
Project window	Same Project window	Moves the location of the item. Or press CONTROL and drag an item to copy it. For example, you can move a component to another container or copy a component within the same container. You can also reorder the components in the Project window list, which affects how components overlap on a form (the z-order), the tab order, and the order they are declared in the Java code.
Project window	Different Project window	Copies the file or component to the new project.
Form Designer	Project window	Within the same project, moves the location of the component; or press CONTROL and drag to copy a component.When dragging to a different project, copies the component.
Menu Designer	Project window	Within the same project, moves the location of the menu item; or press CONTROL and drag to copy a menu item.When dragging to a different project, copies the menu item.

Drag from…	Into…	Result…
Windows Explorer or other file system window	Project window	Adds the file to the project. (However, it does not copy the file to the project folder.)

Viewing the files in a project

The Files view lists all of the files contained in your project. At the top level are the files for the components in your project, as well as any HTML files you may have added to the project.

In the `Imports` folder are all of the Java source files for the packages your project uses. Whereas in the Packages view the files are listed by package (see "Viewing the packages in a project" on page 3-33), in the `Imports` folder the files are listed alphabetically.

HTML files are also listed in the Objects view (see "Viewing the components and HTML files in a project" on page 3-29).

Looking at the Files view

Click the Files tab in the Project window.

Adding a new file to a project

Visual Cafe lets you create a new text file from within its environment. You can add HTML code to create HTML files and Java code to create Java source code files; you can add these files to your project.

To add a new file to a project:

1. From the File menu, choose New File.

   An empty Source window opens.
2. From the File menu, choose Save As.
3. In the Save As dialog box, type the file name (including the appropriate extension) and select the Add to Project option.
4. Click Save.

   The file name appears in the title bar.
5. In the Source window, add appropriate code or text.

Adding an existing file to a project

The only way an HTML file will appear in the Project window is if you manually add it. You can add HTML files to the project as a helpful organizational tool, but it is not required. Adding HTML files to your project allows you to see how your Java applet works.

You can also add Java source files (with the extension `.java`). For example, if you wanted to import a Java source file for an applet that you created in a product other than Visual Cafe, you could place the file in a project folder, then add the `.java` file to the Project window. Visual Cafe will attempt to translate the file into its visual environment.

Remember that you should store all the files of a given project in a separate folder dedicated to that project. Doing so will make deploying your applets, applications, and Web pages much easier. You should also avoid sharing files between projects.

Note: File names are relative to the current project. For example, if you add a file called `foo.java` and it resides in a subfolder (called `foo1`) of the project folder, the file name is stored as `foo1\foo.java` and is not fully qualified. If `foo1` was a folder at the same level as the project folder, the file would be stored as `..\foo1\foo.java`.

To add files from the Visual Cafe environment:

1. From the Insert menu, choose Files into Project.

   The Project Files dialog box appears.
2. Select the file(s) that you want to add.
3. Click Add or Add All, as appropriate.
4. Click OK.

   The file(s) appear in the Project window.

To add files from the Windows Explorer or other file system window:

Drag Java or HTML files from the Explorer into the Project window.

Deleting a file from a project

As you are developing your project, you can delete HTML and Java source files as needed.

To remove a file by pressing Delete:

In the Files view of the Project window, select a file and press DELETE.

To remove files by using the Project Files dialog box:

1. Make the Project window active.
2. In the Packages or Files view, right-click the window to display the pop-up menu, then choose Insert/Remove Files.

   Or while any view is displayed, from the Insert menu, choose Files into Project.
3. In the Project Files dialog box, select one or more files from the list at the bottom pane of the dialog box.
4. Click Remove.
5. Click OK.

Notes:

When you remove a Java source file from a project, all associated visual elements that are created in the Java code are also removed.

Deleting a file from a project does not delete it from the project folder. You must manually delete it.

You cannot remove imported files.

Copying a file of a project

You can copy and paste a file from one project to another, or within the same project. The file is duplicated and placed in the target project folder.

To copy a file of a project:

1. Open the project(s) you want to use and click the Packages or Files tab.
2. In the Project window, select the file that you want to copy.
3. From the Edit menu, choose Copy (or click the toolbar Copy button).
4. Activate the Project window that you are moving the file to.
5. From the Edit menu, choose Paste (or click the toolbar Paste button).

The file appears in the Project window. If you are pasting into the same project that you copied the file from, the file is renamed to prevent file name conflicts.

Working with components in a project

Following are some tasks you can perform with components in a project:

- Viewing components and HTML files
- Adding a component
- Copying a component
- Renaming a component
- Deleting a component

This section explains how to perform each of these tasks.

Viewing the components and HTML files in a project

The Objects view of the Project window shows the components in your project and, if they were added to the project, the HTML files. The components in a container appear subordinate to the container, like a file system display. The containers at the top level are separate Java files in your project, while the components in the containers are Java code within the container Java file. The Objects view is the view many people use most often during Java program development with Visual Cafe.

To look at the Objects view:

Click the Objects tab in the Project window.

Adding a component

Visual Cafe provides several ways to add components to your project. When components are added directly to a form in the Form Designer, they are also added to the associated project.

You can drag a component from the following areas into the Project window, Form Designer, or Menu Designer:

- the Component Palette or Component Library
- a Project window or Form Designer of the same project (press CONTROL and drag to copy the component)
- a Project window or Form Designer of a different project

Note: When you copy a component within the same or a different project, only component code that is automatically generated by Visual Cafe is copied. This does not include custom code and interactions.

To add a component by using an Insert menu item:

1. While the Project window or Form Designer is active, choose a menu item from the Insert menu to add components to your project.
2. Right-click the Project window and choose an Insert menu item.

The component is added to the container that is currently selected in the Project window.

Note: If the parse fails, you can see a file in the Packages and Files view, but not see an object in the Objects view of the Project window. You probably need to correct the Java code to get the file to parse. See "Adding code to a Java source file" on page 4-23 for more information.

Copying a component

You can copy and paste a component listed in the Objects view from one project to another, or within the same project:

- If you copy and paste a top-level container, the corresponding Java file is duplicated and placed in the target project folder.

- If you copy and paste a component in a container, the Java code of that component is placed in the Java file of the top-level container. Only component code that is automatically generated by Visual Cafe is placed in the Java file. This does not include custom code and interactions.

Visual Cafe allows only appropriate copies; for example, a component that is not a container cannot be copied to the top level.

To copy and paste a component:

1. Open the project(s) you want to use.
2. Click the Objects tab in the Project window, then select the component that you want to copy.
3. From the Edit menu, choose Copy (or click the toolbar Copy button).
4. Click the Objects tab in the Project window that you are moving the component to.
5. If you want to paste the component within another container, select that container.
6. While the target Project window is active, from the Edit menu, choose Paste (or click the toolbar Paste button).

   The component appears in the Project window. If needed, the component is renamed to prevent name conflicts.

To copy and drag a component:

- Within the same project, press CONTROL and drag the component to the location you want it.
- To a different project, drag the component to the location you want it.

Renaming a component

You can rename the components listed in the Objects view of a project. If you rename a top-level container, the corresponding Java file name changes. If you rename a component in a container, the Java code of the top-level container changes for that component. Visual Cafe changes the name in the entire Java source file, even in custom code.

To rename a component from the Project window:

Click the Objects tab in the Project window, select the component name, press TAB, and retype the name in the edit frame. Press ENTER or click somewhere else when you are finished.

or

Click the Objects tab in the Project window, slow double-click the component name and type the new name in the edit frame. Press ENTER or click somewhere else when you are finished.

To rename a component from the Property List:

Make your project active, then change the Name property of the component in the Property List.

Deleting a component

If you delete a top-level container in the Objects view of the Project window, the corresponding java file is deleted from the project and the other views. The file is not deleted from the folder on your hard disk.

Note: If you want to delete a file, you should first delete it from the project then delete it using Windows operating system commands. Do not delete a file using operating system commands before first deleting it from the project.

If you delete a component within a container, the code automatically generated by Visual Cafe is deleted. You must manually delete any custom code or interactions. Similarly, if you delete a component's code from a Java file, the component is removed from the Project window after you click out of the Source window or save the Java file.

Note: To avoid deleting the wrong code or not all of the code, it is recommended that you delete components from the Project window instead of from the source code. Then delete from the source code any custom code or interactions.

To delete a component:

1. Click the Objects tab in the Project window.
2. Select the component and press the DELETE key.

   If you delete a component from a container, you must manually delete any custom code or interactions involving that component.

   If the component is a top-level container, the corresponding Java file is not deleted from the folder on hard disk. You must manually delete it.

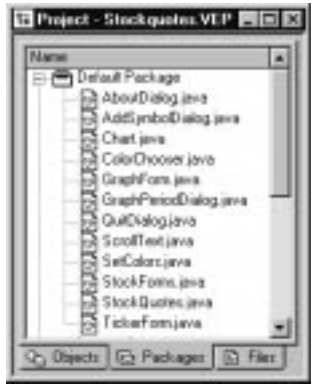
Working with packages

The following are actions you can perform as you work with packages:

- Viewing the packages in a project
- Adding packages to Visual Cafe

Viewing the packages in a project

The Packages view lists the Java source files, grouped as Java packages, in your project. The Packages view always shows the standard Java packages that Java programs require. When components are added to a project, a Default Package, containing the java files for those objects, appears. If you add a Visual Cafe component, the Symantec package also appears in the Package view; it contains the Java source file for each Visual Cafe component you include.



You can also make other packages available to Visual Cafe projects by adding them to the Visual Cafe class path. See “Adding packages to Visual Cafe” on page 3-34.

Usually, only advanced Java programmers are interested in the Packages view.

To see the Packages view:

Click the Packages tab in the Project window.

You can expand and collapse your view of a package like you would for a hierarchy of folders and files in the Windows file system.

Note: Classes that have not been added to a package display under the Default Package. Only source files that have been added to the project are listed.

Adding packages to Visual Cafe

The standard Java packages and the Symantec Visual Cafe packages are available to you as you create Java programs in the Visual Cafe environment. You can also make other packages available to Visual Cafe projects by adding them to the Visual Cafe class path. For example, you could add third-party or your own packages containing components or utilities. You can set the class path for a project or for the Visual Cafe environment. See "Specifying file search paths and the output folder" on page 3-47 for more information.

Remember to keep the folder structure of your package intact, and make sure that your file names use the same uppercase and lowercase letters, exactly as they were before copying the package.

If you add a Java `import` statement or use part of the new package in your Java source code, the package appears in the Packages view.

Setting project-level options

The most complex command in the Project menu is Options. Choosing this command opens a multipage dialog box. In the Project Options dialog box, Visual Cafe lets you perform the following tasks to set options.

To set options that apply to a single project:

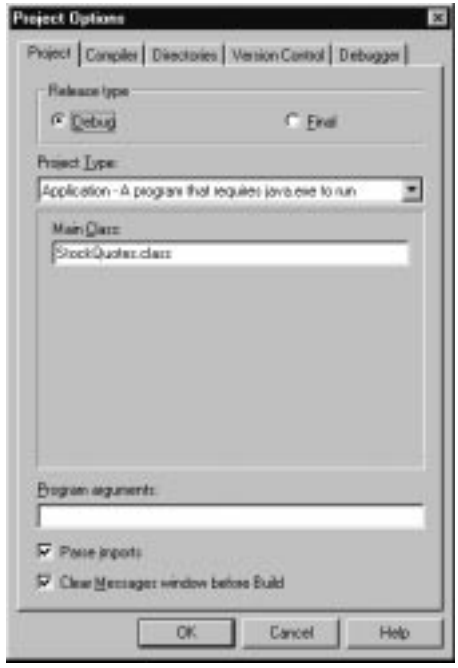
1. Activate the Project window of the project you want to work with.
2. From the Project menu, choose Options.

To set environment options:

For information about setting environment options, "Setting environment options" on page 9-1.

Project Options

This command opens the Project Options dialog box in which you configure the options settings for the selected project. Options are project specific, unlike the preferences you set using the Environment Options dialog box. Options settings remain bound to a project even when you close and later reopen the project. They control general project behavior, including build and run settings. The preferences in the Project Preferences dialog box affect all projects.



The Project Options dialog box has the following options tabs:

Project

- Specifying whether builds are debug or final
- Setting the project type to applet or application
- Specifying what applets to run and the HTML file
- Making applets run in the Applet Viewer or a browser
- Specifying the main class to run for an application
- Specifying arguments for application execution

- Specifying whether to clear messages before builds
- Specifying whether to parse imports

Compiler

- Specifying Java optimizations
- Generating Debug information
- Specifying Sun's Java compiler instead of Symantec's

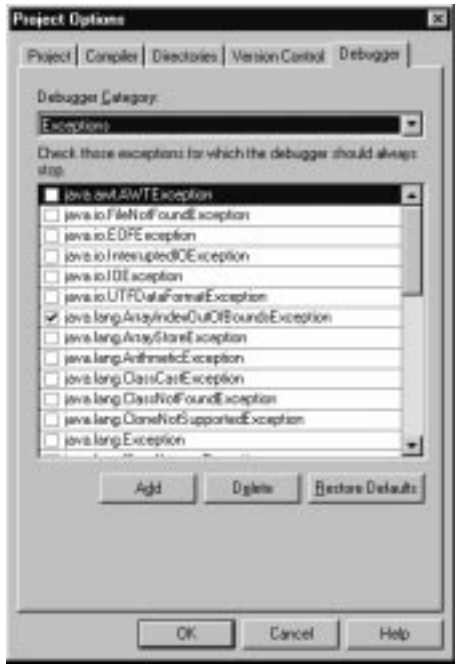
- Setting messages options

Directories

- Specifying file search paths and the output folder (directory) for class, output, source, and library files, as well as managing folders.

Version Control

- Provides a means of working with an existing version control system in your development environment



Debugger

- Setting Debugging options for exceptions in a project
- Remote Debugging options

Each options page is represented by a tab at the top of the dialog box. By default, when the Project Options dialog box appears, it displays the Project options page. Clicking a tab selects the corresponding page. You can move freely between options pages as you configure options. You can move around the options on each page by using the tab key. You can change tabs by using the right and left arrow keys.

Each options page contains Cancel and Save buttons. Cancel performs its standard Windows operation. Clicking Save saves the options settings for the current editing session; these options settings are then available for selection–for example, in another editing session or from the Project window.

Other common elements on all options pages include the Help area and the Options pop-up menu.

Specifying whether builds are debug or final

Visual Cafe has Debug and Final project option sets that let you specify how you want to compile your Java code. By default, for Debug the compiler includes debug information, while for Final no debug information is included and Java optimizations for speed and compactness are performed. Debug information enables you to use all Visual Cafe debug features when you debug your Java programs, but makes the compiled code larger.

In the Project Options dialog box, the Compiler and folders tabs show the release type option sets. For more information on the options, see "Setting compiler options for a project" on page 3-45 and "Specifying file search paths and the output folder" on page 3-47.

To set the release type:

1. Activate the Project window of the project you want to work with.
2. From the Project menu, choose Options.
3. In the Project Options dialog box, click the Project tab.
4. Select one of the following options:

Select…	To do this…
Debug	By default, build an executable that contains debugging information.
Final	By default, build a more compact executable that is optimized and contains no debugging information.

   You can change the default, as described in the next procedure.

5. Click OK.

   The change takes effect the next time you compile your Java program.

To change a release type option set:

1. From the Project Options dialog box, set the release type to Debug or Final.
2. Click the Compiler and Directories tabs and set the options for that release type.

3 Click OK.

The change takes effect the next time you compile your Java program.

Project types

If you started developing a project based on the empty project template, you can still tell Visual Cafe what kind of project type you want it to be.

Clicking either radio button in the Project Type group displays a set of options for the project. The contents of this subpage vary according to which program type radio button you click: Application or Applet.

To set the project type to applet, application, JavaBean, Win32 application, or DLL

1 Activate the Project window of the project you want to work with.

2 From the Project menu, choose Options.

3 In the Project Options dialog box, click the Project tab.

4 Select one of the following options:

Select…	To specify this…
Applet	The project is an applet. When you run a project, the Starter HTML setting determines which HTML file is used. The HTML file determines which applets are run. See "Specifying what applets to run and the HTML file" on page 3-41.To specify that your applets should run in your default Web browser, select the Execute applet in default HTML viewer option. Deselect it if you want to run applets in the Applet Viewer associated with the Visual Cafe environment. For more information, see "Making applets run in the Applet Viewer or a browser" on page 3-42.
Application	The project is a standalone application. To run the application from Visual Cafe, the main class must also be specified. See "Specifying the main class to run for an application" on page 3-43.
Win32 Application (Professional edition only)	The project is a native, standalone executable. To run the application from Visual Cafe, the main class must also be specified.

Select…	To specify this…
Win32 DLL (Professional edition only)	The project is a native Dynamic Link Library (DLL). You can specify the program to use to run and debug the DLL. You can also set the library name, which is by default the project name appended with the appropriate extension.

5 Click OK.

The change takes effect next time you run your project.

Specifying what applets to run and the HTML file

An applet is launched from an HTML file that has an applet tag. Visual Cafe can automatically create an HTML file with applet tags for all the applets in your project. When you run your project from Visual Cafe, you can specify what HTML file to use to display your applets. Here are some scenarios:

- If you run your project with the automatically generated HTML file in the Applet Viewer, all of your applets appear in separate windows.
- If you run your project with the automatically generated HTML file in a Web browser, all of your applets appear in an otherwise blank browser window.
- If you run your project with your own HTML file in the Applet Viewer, each applet that has an applet tag will appear in a separate window.
- If you run your project with your own HTML file in a Web browser, the HTML file appears in a browser window, including the applets as specified in the file.

If you want to test the files for a Web site, you could specify the home page as the starter HTML and run the page from a Web browser. Then you could access the other pages from this page to make sure your applets work.

To test the files for a Web site:

1 Activate the Project window of the project you want to work with.

2 From the Project menu, choose Options.

3 In the Project Options dialog box, click the Project tab.

4 Specify an HTML file in one of these ways:

- Choose (Automatic) to run all of the applets from an automatically generated HTML file that is blank.

- Choose one of your own HTML files from the pop-up menu. HTML files that you added to your project automatically appear in the pop-up menu.
- Click … to browse for an HTML file.
- Type in the field to specify a file name or a URL, for example, `http://myserver.someplace.com/somefolder/some.htm`.

5 Click OK.

The change takes effect next time you run your project.

Making applets run in the Applet Viewer or a browser

When running applets from Visual Cafe, you can launch your applets in the Applet Viewer associated with the Visual Cafe environment or in the Web browser of your choice. The Web browser must be set up to be the default Web browser for your computer.

Running applets in the Applet Viewer can be faster, because the browser does not have to start up. However, it is a good idea to test your applets in popular browsers before deployment.

Note: When you run an applet in the debugger, it is always run in the Applet Viewer.

To specify where applets are to run:

1 Activate the Project window of the project you want to work with.

2 From the Project menu, choose Options.

3 In the Project Options dialog box, click the Project tab.

4 Select Applet as the Project Type, if needed, then select Execute applet in default Web browser if you want to run the applet in a browser or deselect it if you want to run the applet in the Applet Viewer.

5 Click OK.

The change takes effect next time you run your project.

Tip: When this option is selected, Visual Cafe looks for the application you have associated with the file extensions `.htm` and `.html` (Hypertext Document file type). If an association does not exist, you must define one.

To set file associations (including the open action) in Windows 95 and Windows NT 4.0 and higher, from a Windows file system window (such as the Explorer), choose View, then Options, then click the File Types tab.

A browser might have already set the association for you, for example, the file type Netscape Hypertext Document.

Specifying the main class to run for an application

When you run an application from the command line, you type the name of the Java file that contains the main method. To run an application from within the Visual Cafe environment, you must specify the starting point of your application (the Java class file) so Visual Cafe knows how to run your application. If you started a project with the Basic Application template or inserted a Java file that already had a main method in it, the main class was already specified for you. If there is no entry in this field, Visual Cafe tries to use the project name.

If your application also accepts arguments on the command line (the main method takes arguments), you need to specify those. See "Specifying arguments for application execution" on page 3-44.

To specify the main class to run for an application:

1. Activate the Project window of the project you want to work with.
2. From the Project menu, choose Options.
3. In the Project Options dialog box, click the Project tab.
4. Select Application, if needed, then type the name of the class file in the Main Class field; for example, `Frame1` or `Frame1.class` are both acceptable. If the class is inside a package other than the Default Package, you need to type *`package_name.class_name`* in this field.

   You can enter one name only.
5. Click OK.

   The change takes effect next time you run your project.

Note: If you rename a class that appears in this field, Visual Cafe updates the field for you.

Specifying arguments for application execution

If your application accepts arguments on the command line (the main method takes arguments), you need to specify them so Visual Cafe can run your application from its environment. For example, the Sun Java compiler is written in Java and takes command line arguments.

You also need to specify the main class. See "Specifying the main class to run for an application" on page 3-43.

To specify command line arguments:

1. Activate the Project window of the project you want to work with.
2. From the Project menu, choose Options.
3. In the Project Options dialog box, click the Project tab.
4. Select Application, if needed. In the Program arguments field, type any arguments that should be passed to the program when you run it.

   Delimit the arguments with a space.
5. Click OK.

   The change takes effect next time you run your project.

Specifying whether to parse imports

Visual Cafe ships with the standard Java package imports from the Sun Microsystems Java Developers Kit (JDK) in a preparsed form. These imports might be required by applets and applications in order to execute.

By default, if you import other packages (including the Symantec packages), Visual Cafe will parse them so you can, for example, look at them in the Class Browser, view them in the Project window, and see them in the Form Designer, if applicable. Not parsing imports requires less computer resources and makes Visual Cafe run faster. You might want to disable import parsing if, for example, you import a lot of third-party packages and your computer runs very slowly as a result of the parsing.

To specify whether to parse inputs:

1. Activate the Project window of the project you want to work with.

2 From the Project menu, choose Options.

3 In the Project Options dialog box, click the Project tab.

4 Select Parse imports to specify that imports be parsed automatically as you work with the project. Or deselect it to not parse these imports.

5 Click OK.

The change takes effect immediately if you select the option. If you deselect the option, no more imports are parsed.

Specifying whether to clear messages before builds

By default, Visual Cafe clears the Messages window before each build. This makes it easier to see what messages apply to the current build. However, you can specify that the window not be cleared so you can see messages from previous builds and compare build messages.

To specify that the Messages window not be cleared before each build:

1 Activate the Project window of the project you want to work with.

2 From the Project menu, choose Options.

3 In the Project Options dialog box, click the Project tab.

4 Select Clear Messages window before build to clear the Messages window before each project build. Deselect this option to not clear the window.

5 Click OK.

The change takes effect next time you compile your project.

Setting compiler options for a project

From the Compiler tab of the Project Options dialog box, you can control what compiler information is sent to the Messages window, which Java compiler to use, and if Java optimizations are performed. This view changes the option set for the Debug or Final release type. For example, if you have the debug option set selected and change an option in the compiler page, Visual Cafe changes that option for the debug option set, not the final option set. For more information, see "Specifying whether builds are debug or final" on page 3-39.

To access the Compiler view:

1. Activate the Project window of the project you want to work with.
2. From the Project menu, choose Options.
3. In the Project Options dialog box, click the Compiler tab.

Specifying general compiler options

There are several general compiler options that you can specify.

Show compiler warnings

Display compiler warnings in the Messages window. During development, you may want to look at items identified by warnings. (Default: enabled)

Show all Java messages

In the Messages window, display detailed messages that you might be interested in if you need more information; for example, if you have import problems these messages trace where your classes are coming from to help you resolve any import and class path problems. (Default: disabled)

When using the Sun Java compiler, this option causes the compiler to report diagnostic messages about its own execution. This option is ignored by the Symantec compiler. (Default: disabled)

Show progress messages

Display compiler progress messages in the Messages window. (Default: disabled)

Show dependencies

Display file dependencies, such as imports, in the Messages window. (Default: disabled)

Generate debug information

Create debugging information used by the Visual Cafe debugger. For example, this option lets you see local variables during debugging. (Default: enabled for Debug and disabled for Final release types)

Use Java optimizations

Optimize the Java executable for a more compact executable that runs faster. (Default: enabled for Final and disabled for Debug release types)

Select the Disable function inlining option if needed. Inlining means that Visual Cafe takes a function's code and imbeds it in the calling function instead of calling the function. Inlining increases execution speed but also increases executable size. (Default: not selected)

Use the Sun Java compiler

Use the Sun Java compiler, `javac.exe`. When this option is cleared, the Symantec Java compiler, which is faster, is used. Note that you cannot compile native applications and DLLs with the Sun Java compiler. (Default: disabled)

Specifying file search paths and the output folder

From the Directories tab of the Project Options dialog box, you can add and remove folders (directories) that Visual Cafe uses to locate source and class files and control where Visual Cafe puts compiler output files. This view changes the option set for the Debug or Final release type. See "Specifying whether builds are debug or final" on page 3-39 for more information.

You can also set class path information for the entire Visual Cafe environment. Remember that Visual Cafe does not inherit Windows class path information unless it is set up to do so.

To access the Directories view:

1. Activate the Project window of the project you want to work with.
2. From the Project menu, choose Options.
3. In the Project Options dialog box, click the Directories tab.

The source search path applies to Java source files and any text file that can be opened in the Source window.

Note: You can set the source search path for the entire Visual Cafe environment from the Environment Options dialog box and with the `javainc` statement in the `\VisualCafe\Bin\sc.ini` file.

To specify source file search paths:

1. Go to the Directories view of the Project Options dialog box.
2. Choose Source Files in the Show directories for field.

   A list of folders (directories) that can contain source files appears. The folder order affects the search order: the topmost folder is the first to be searched. The default setting, the project folder, does not appear in the list; it is the first folder in the search order.
3. Modify the list as needed:
   - To change the order in which folders are searched, select a folder and move it with the Up Arrow and Down Arrow buttons.
   - To delete a folder from the list, select the folder and click Delete.
   - To add a folder to the list, select the blank entry (marked by an empty box) at the bottom of the list and type the folder name, including the full path. Or click the New button (located above the text box), then select a folder by clicking the … browse button that appears in the field. You can also use the New button to insert a new entry above the selected entry.
4. Click OK.

The change takes effect immediately.

To specify library file search paths (native only)

If a native application or DLL uses libraries, you need to make sure Visual Cafe can find them. To specify locations to look for libraries, follow these steps:

1. Go to the Directories view of the Project Options dialog box.
2. Choose Library files in the Show directories field.

   A list of folders (directories) that can contain library files appears. The order affects the search order: the topmost folder is the first to be searched. The default is to search folders specified for the Windows LIB environment variable; these folders do not appear in the list and are last in the search order.
3. Modify the list as needed:
   - To change the order that folders (directories) are searched, select a folder and move it with the Up Arrow and Down Arrow buttons.
   - To delete a folder from the list, select the folder and click Delete.
   - To add a folder to the list, select the blank entry (marked by an empty box) at the bottom of the list and type the folder name,

including the full path. Or click the New button (located above the text box), then select a folder by clicking the … browse button that appears in the field. You can also use the New button to insert a new entry above the selected entry.

4 Click OK.

The change takes effect the next time you compile.

Specifying the output folder for a project

You can specify where compiler output files, such as class files, are stored for this project.

Note: When you run a project in the Applet Viewer, the output folder is temporarily added to the class path.

To specify the output folder:

1 Go to the Directories view of the Project Options dialog box.

2 Choose Output folder in the Show directories for field.

The folder appears. If no folder is specified, Visual Cafe uses the default: a `.class` file is placed in the same location as the corresponding `.java` file. If you do specify an output folder, all output files are placed in this folder and any package file hierarchy structures are created as well.

3 Type a folder and full path in the field, or select a folder by clicking the … browse button that appears in the field.

4 Click OK.

The change takes effect the next time you compile your project.

Specifying class search paths for a project

You can tell Visual Cafe where to look for the class files used by a project; for example, where to find the packages you are using. You can make up your own custom list of folders, let Visual Cafe automatically generate the class path based on the `.java` files you have added to your project (including those created when you add top-level components), and use the class path set for the Visual Cafe environment. If you specify one or more of these options, the search order is your list, the automatically generated list, then the class path for the environment.

To specify the class search paths:

1. Go to the Directories view of the Project Options dialog box.
2. Choose Class Path in the Show directories for field.

   A list of folders (directories) appears. The folder order affects the search order: the topmost folder is the first to be searched.
3. Modify the list as needed:
   - To change the order that folders are searched, select a folder and move it with the Up Arrow and Down Arrow buttons.
   - To delete a folder from the list, select the folder and click Delete.
   - To add a folder to the list, select the blank entry (marked by an empty box) at the bottom of the list and type the folder name, including the full path. Or click the New button (located above the text box), then select a folder by clicking the … browse button that appears in the field. You can also use the New button to insert a new entry above the selected entry.
4. To generate the class path based on the files in the project, select Autogenerate class path. Otherwise, deselect it.

   The default is selected. When this option is selected, if a file is not in the project folder, the file path is added to the class path.
5. To append the Visual Cafe environment class path to the project class path, select Append class path. If you deselect it, the class path defined for the Visual Cafe environment is not used.

   The default is selected.
6. Click OK.

The change takes effect immediately.

Setting the class path for the Visual Cafe environment

Visual Cafe gets its class path information from this file: `\VisualCafe\Bin\sc.ini`. You can modify the `sc.ini` file to set class path information globally for your Visual Cafe environment. You should append your information to the classpath statement, not delete the default classpath statement, because it can affect the operation of Visual Cafe.

Having the Visual Cafe environment inherit the class path from the Windows environment

You can set up the `sc.ini` file so that Windows class path information is inherited by the entire Visual Cafe environment. However, realize that the class path information might not be useful for Visual Cafe projects and can cause delays whenever Visual Cafe searches for the source corresponding to a class file when your Java programs are built. Some products that use class files do not have their own `ini` file, so they put their class path information in the Windows environment.

For Windows 95, the class path is defined in the autoexec.bat file. Windows NT can also use the class path in an `autoexec.bat` file, if it is present.

For Windows NT 4.0 and higher, you can set the CLASSPATH variable by opening Control Panel, then choosing System, then clicking the Environment tab, and entering a value so it appears in the System or User Variables list box.

Inheriting the Windows class path setting

1. In the Environment section of `\VisualCafe\Bin\sc.ini,` add the following specification at the end of the classpath statement:

   ```
   ;%classpath%
   ```

2. Restart Visual Cafe for the change to take effect.

Setting the class path for a Web browser

Before deployment, you might want your Web browser to be able to locate the Visual Cafe classes when the browser is launched outside of the Visual Cafe environment. The Visual Cafe installer can set this class path information for you. To set it manually, make sure the following is part of the class path information for your Windows environment:

```
C:\visualcafe\java\lib\vecclass.zip
```

Using Version Control

You can use version control software that integrates into the Visual Cafe environment. If you have installed version control software on your computer, you will be able to select it in the Visual Cafe project options. You enable version control software on a per-project basis.

See the documentation provided with your version control system for more information.

1. Activate the Project window of the project you want to work with.
2. From the Project menu, choose Options.
3. In the Project Options dialog box, click the Version Control tab.
4. Choose the version control software you want to use.

   You must properly install the software before you can use it.
5. Click OK.

The change takes effect immediately.

Note: If you enable a version control system or switch version control systems for a project, the files are checked into the new version control system as new files.

Setting remote debugging options for a project

From the Debugger tab of the Project Options dialog box, you can do the following:

- From General, you can specify if you want to enable remote debugging. This allows you to debug a program from a remote site. For more information, see "Debugging remotely" on page 8-34.
- From Exceptions, you can view and customize how the debugger works with exceptions for this project. For more information, see . . "Handling exceptions" on page 8-13.

Defining the Visual Cafe startup mode

The startup mode defines what processing you want done when you start a new Visual Cafe session.

1. From the File menu, choose Environment Options, then click the General tab.
2. In the On Startup area, select the appropriate option.

Select…	To specify this…
Create a new project	Open a new project each time Visual Cafe starts.

Select…	To specify this…
Open the last project	Open the project that was active the last time Visual Cafe exited.
Do nothing	Specify that no project opens and that you will select an appropriate action after Visual Cafe starts.

Automating source file backups

You can control whether files in a project are automatically backed up each time that a save is performed. You can also specify the location and name of the backup files.

Note: Only Java and HTML files that have changed are backed up. For example, if you save all files in a project, only the files that have changed are backed up.

To automate project backups:

1. From the File menu, choose Environment Options, then click the Backup tab.
2. Select Backup files on Save.
3. Select the location and name of the backup files.

Select…	To specify this…
Create BAK file	Create one or more backup files that are named `file.bak`.
Copy to folder	Copy the source files to the specified folder. You can type the folder with the full path into the text box, or click … to select a folder from a dialog box. There is no default.
Invoke OnBackup script	Run your own macro, created with the Visual Cafe macro utility and containing an `OnBackupFile` statement. All macros are placed in the `\VisualCafe\Bin\Macs` folder.

Defining a new default template

You can select a template to use as the default for all new projects.

1. From the File menu, choose New Project.

   The current default template is highlighted and an asterisk is placed next to the template name.
2. Select a template.
3. Click Set Default.
4. Click OK.

A new project is created based on the selected template. This template will be used for subsequent new projects until the default is changed.

Creating a project template and adding it to the library

Project templates are used as the basis of new projects. When you create a project from a template, the project initially contains the files and components in the template. You can create your own templates as the starting point of new projects.

Important: When you save a project as a template, make sure all of your source files (both `.java` and `.html`) are in the same project folder or a subordinate folder.

---

1. Create a project to be used as the template. Add any source code filesthat you want to add.
2. With the current project in a Project window, choose Project, then choose Create Project Template.

   The Create Template dialog box appears.
3. Select the group that you want the template to belong to in the Component Library.

   If you do not select a group, the template is added to the Project Templates group.
4. Type a name and description for the template in the appropriate fields.

   The name and description is displayed for the template in the New Project dialog box. The description appears when the template is listed in the Component Library window.
5. Click OK.

Visual Cafe copies the project and stores it as a project template. The template is now available from the New Project dialog box.

Deleting a project template

The project templates that ship with Visual Cafe cannot be deleted. You can delete a project template that you create as long as it is not the current default template. After you delete it, it no longer appears in the New Project dialog box.

1 Open the Component Library.

2 Display the contents of the Project Templates folder.

3 From the template list.

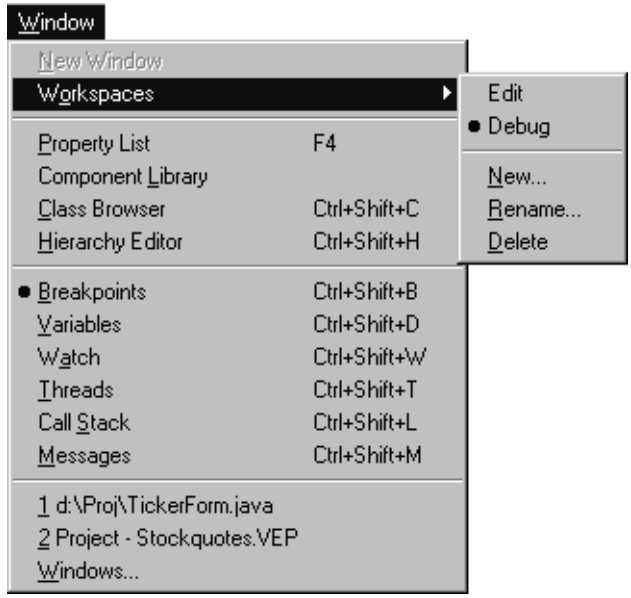
Use Workspaces to customize your work environment

A large number of views are available in the Visual Cafe environment. Visual Cafe lets you save a configuration of windows as a unique workspace. A workspace provides a convenient way to switch from one screen layout view to another.

A workspace is a saved arrangement of windows. Because the various tools in Visual Cafe are displayed in many individual windows, workspaces are used to group together the windows that have related functions. Workspaces provide a means of organizing the multiple windows of the development environment into logical groups. For example, when you edit source code you could use a workspace that displays the text editor, the Messages Window, and the Project window. When you are debugging your program, you could use another workspace that opens windows for the different debugging tools.

Workspaces are task-oriented as opposed to project-oriented. You create workspaces for different tasks, such as designing, editing or debugging.

You can then access the workspaces from the Workspace toolbar or from the Window menu under Workspaces



Visual Cafe supplies you with two workspaces; Edit and Debug. The Debug workspace automatically loads when you run your application and opens necessary debugging windows. When the processing ends, the development environment returns to the Edit workspace.

You can create, delete, or rename workspaces. Only global view windows are saved in a workspace.

Workspaces can be loaded automatically while running a project and when you stop the project. They are saved dynamically. For example, if the Property List window is open in the Edit workspace and you close that window while in the Edit workspace, the change is saved permanently. This feature ensures that as you run and stop your programs, the window state remains as you last left them.

Visual Cafe supports the extended mouse functions of the Windows 95 interface. Right-clicking on various components of the windows opens context-sensitive pop-up menus. As you work with Visual Cafe, try right-clicking on the different parts of the screen. You may discover some useful shortcuts!

You can create, delete, and rename a workspace by using the Workspaces menu option. Visual Cafe automatically saves changes to a workspace configuration when you exit the workspace.

To change to a different workspace:

Use either of these methods:

- From the Window menu, choose Workspaces, then choose one of the workspace names displayed in the submenu.
- From the Workspace toolbar, select the appropriate workspace name.

To save your current Visual Cafe window arrangement as a new workspace:

1. Configure the screen as you like by opening the windows you need and positioning and sizing them to suit your requirements.
2. From the Window menu, choose Workspaces, then choose New.
3. In the New Workspace dialog box, type a new name.
4. Click OK.

Visual Cafe creates a new workspace and displays it in the toolbar Workspace field. The workspace is also added to the Workspace list.

To rename a workspace:

1. From the Window menu, choose Workspaces, then choose Rename.
2. In the Rename Workspace dialog box, type a new name.
3. Click OK.

Visual Cafe changes the workspace name and displays it in the toolbar Workspace field. The name also changes in the Workspace list.

To delete a workspace:

Caution: Deleting a workspace immediately deletes the workspace named in the toolbar Workspace field.

Note: You cannot delete the last remaining workspace.

From the Window menu, choose Workspaces, then choose Delete.

The workspace is deleted and the next workspace in the listing is activated.

Working with subprojects

Visual Cafe lets you add subprojects to a project. When you compile the parent project, the subproject gets compiled and saved as well. When you run a project that has subprojects, only the parent project is run. To run and debug a subproject, you need to open the project in its own Project window.

You add subprojects to a project by using the Insert Files dialog box. You choose the project file type and then select a project file and add it. Subprojects appear in both the Objects view and Files view of the Project window.

Subprojects only appear in the files view of the Projects window.

Note: There is currently no particular order in which subprojects are compiled.

Adding subprojects

1. Open the project you want to add subprojects to.
2. From the File menu, choose Open.

   The Open dialog box appears.
3. Navigate to the project folder.
4. Make sure Visual Cafe Project is shown in the Files of type field.

   The projects display.
5. Select a project from the list.
6. Click Open.

   The project opens with the last saved window configuration.
7. Add the `.vep` file of a project to the parent project.

The project you added becomes a subproject of the parent project. It appears in the Files view of the Project window.

Objects View

Subprojects are not expandable in any of the Project window views to show the visual elements within the subproject. Double-clicking the subproject object opens the project in its own project window.

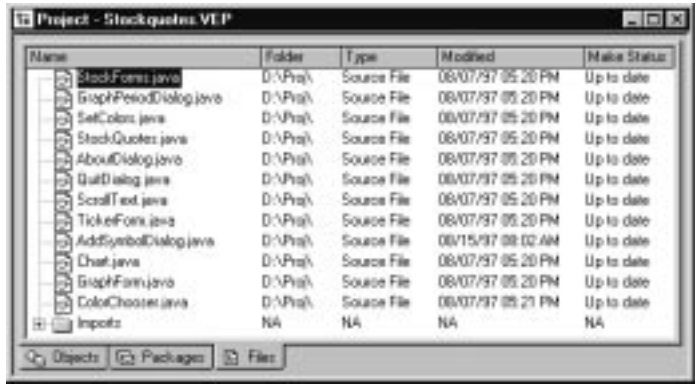
The column headers behave like they do in the Windows Explorer: you click the header to sort by that column. Clicking the same column twice in succession reverses the sort order.

Note: the project icon is not a final icon.

Files view

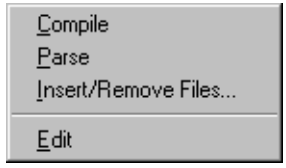
The Files view displays all the files within a project. Imports can be shown or hidden. Subprojects are visible in Files view, but not expandable. You

can double-click a file to open it in the Source Editor, or double-click the project icon to open the subproject in a separate Project window.



The Files view allows you to see the file name, location, type, modification date and time, and its make status. You can remove a file from a project by selecting the file and pressing DELETE.

In Files view, the following context menu is available when you right-click a source file:



Project options and subprojects

Project options can be set for final and debug projects, and subprojects. The page shows the settings that are common between the selected projects.

If the settings differ between the selected projects, the settings indicate that state.

The triangle in the upper right corner of the window is a pop-up menu that contains any menu commands that are appropriate for that view.

Importing source code

Java source code for all kinds of applets and applications are already written and available for you to download from the Internet, compile, and execute. Also, many books written on Java and Visual Cafe contain sample programs for you to use right away. To import source code from outside of Visual Cafe:

1. Make sure that the source file is an extension of the Applet class. If so, then create a Basic Applet project.
2. Insert the file.
3. Remove the `Applet1.java` file.

If the applet hangs, or Visual Cafe cannot compile it, then there may be some problem in the code.

The Visual Cafe main file menus

This section describes the Visual Cafe menu commands.

The File menu

This section provides a detailed explanation of all commands in the Visual Cafe File menu. The project types and options available from the New Project dialog box are described, along with the options in the File Open dialog box. The final sections of the chapter cover creating and modifying project types.

You use the Visual Cafe File menu to create new source code files and to open existing ones. You also use this menu to save and print your source code and to create new project files as you need them.

New Project…

Opens the New Project dialog box, from which you can create a project based on the default project types or one that you have developed and saved.

To create a new project, choose a project type and click the OK button. You can set your most often-used project type as the default project type every time you start a new project.

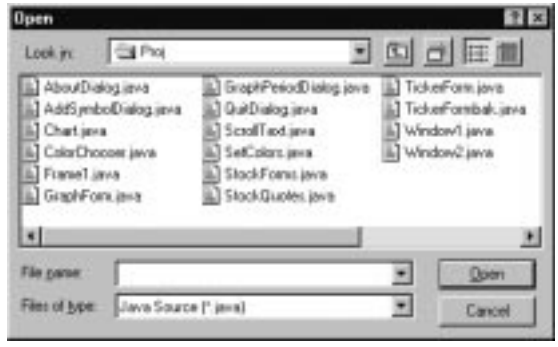
Project types are templates from which new projects are created. They define the libraries, resources, and source code files that the project initially contains, as well as the project's options settings. Visual Cafe provides templates for many common project types. You can create your own project types and display them in the New Project dialog box.Once a project is created from a project type, you can add files, remove files, and change settings as needed. For details, see "Project types" on page 3-40.

New File

Opens an empty and untitled Editor window. You must save the contents of the new window before you can add it to a project. Once the file has been saved, you can add it to the active project by choosing Add "filename" or Add Files from the Project menu.

Open...

Opens a File Open dialog box, which you can use to open text and project files.



Visual Cafe lets you have more than one project open at a time. However, only one project at a time is considered to be active. You can make any open project–or any project that you have opened since you opened Visual Cafe–the active project by choosing its name from the Window menu of the Visual Cafe menu. If you choose a project that is not currently open from the Window menu, Visual Cafe opens that project and makes it the active project.

Files of Typelist box

The Files of Type list box, located at the bottom of the File Open dialog box, is used to set a file filter for displaying files

You can open several types of files:

- Custom (`.java` and `.html` files)
- Visual Cafe Project (`.vep`)
- Cafe Project
- Visual J++ (`.dsw`, `.dsp`)
- Java Source (`.java`)
- HTML (`.html`)
- All files (`*.*`)

Save / Save All

Save saves the contents of the frontmost Editor window to disk. If the file has not yet been saved to disk, a standard File Save dialog box appears for naming and placing the file.

Choosing the Save All command saves the contents of all open windows that have unsaved modifications. This includes any text in the Worksheet window (if open) and any changes made to source code in Class Browser windows.

Save As…

Saves a copy of the contents of the frontmost Editor window with a new name. If the file has been added to the project, the version of the file with the new name replaces the older version in the project file.

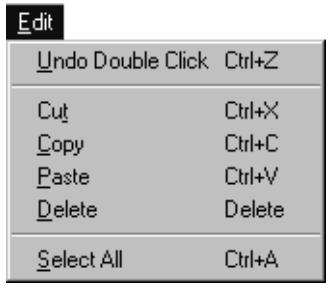
Close Project Close Project closes the frontmost window. You can perform the same function by clicking the close box on the right side of the window's title bar.

Exit You can also quit the Visual Cafe program by clicking the close window button in the upper right corner of the Visual Cafe window.

Printer Setup…Opens the standard Printer Setup…dialog box for your printer. Print… Prints the contents of the frontmost window. You can print the following types of windows: Editor windows, the active pane of a Class Browser window, and Project windows.

The Edit menu

The Edit Menu contains standard Windows functionality, with added functionality in the Debug workspace -- the Delete and Select All functions are enabled. Keyboard shortcuts are listed to the right of each menu item.



Undo

This command cancels the last command you made.

Redo

This undoes the Undo command.

Cut

This item removes selected text from your screen and moves it to the Clipboard on your system.

Copy

This command copies the selected items to the clipboard, but does not remove them from the screen.

Paste

This command inserts the contents of your system's Clipboard where you position the cursor.

Delete

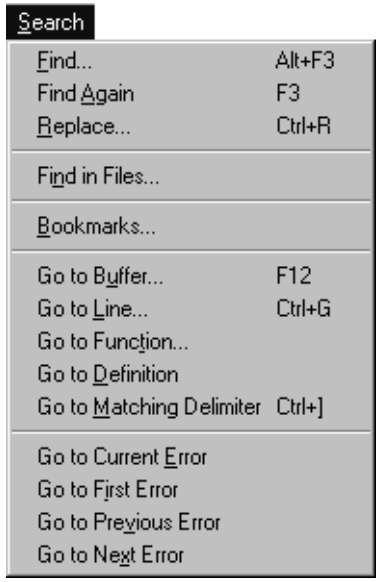
This command removes the selected item and does not move or copy it to the Clipboard. You can use the Undo command to restore a deleted one if it was your most recent command.

Select All

This command highlights all editable objects in the active window.

The Search menu

The search menu contains commands for searching and locating items within your project files very quickly.



Find...

Opens the Find dialog box, where you can enter search criteria for the file in the active editing window.

Find Again

Finds the next occurrence of the text string previously defined with the Find command.

Replace...

Opens the Replace dialog box, where you can specify a search string and replacement text for the file in the active editing window.

Find in Files...

Opens the Find in Files dialog box, where you can search for text across multiple files. The scope of the file search can be defined.

Compare Files...

Opens the Compare Files dialog box, where you can specify two file to compare. The differences are presented in editing windows, allowing you to view the lines that are different.

Bookmarks...

Opens the Bookmark dialog box, where you can add, remove, or go to a bookmark. You can set and move to as many as ten different locations in your source files. Bookmarks are saved through the current Visual Cafe session only.

Go to Buffer...

Opens the Go to Buffer dialog box, where you can change the options for the current edit buffer or those of another.

Go to Line...

Opens the Go to Line window where you can specify a line number to move to. The Source pane is scrolled to the requested line. If any text is currently selected, the selection is extended to include that line.

Go to Function

Opens the Go to Function window so that you can select an available function from the list. The edit window's focus is moved to the selected function location.

Go to Definition

Opens the selected method or data member in the Class Browser. The associated code block appears in the editing pane. If there are multiple occurrences of the selection in the source file, then the Members window opens for a selection.

Go to Marking Delimiter

Finds the delimiter that matches the delimiter to the right of the current insertion point. The insertion point is moved to the front of the matching delimiter. This command can find matching parentheses, square brackets, or braces.

Go to Current Error

Go to First Error

Go to Next Error

Go to Previous Error

The Go to error commands move the editing window focus to the location of the corresponding error.

The Insert menu

With the Insert menu, you can use commands to add items to the current project.

Form

Opens the Insert Form dialog box so that you can insert a form from the Component Library into the current project.

Applet

Inserts an applet if there are no saved applet objects, or opens the Insert Applet dialog box where you can select an applet from a list of applet templates.

Component

Opens the Insert Component dialog box so you can add one or more objects from the Component Library to the active form or to the project.

Class

Opens the Insert Class Wizard, where you can add and define a new class or interface for the current project.

Member

Opens the Insert Member dialog box, where you can declare a method to be added to the current class.

Group

Adds a group to the current project or group. Groups can only be inserted at the root level of the Component Library window or inside other groups.

Files into Project

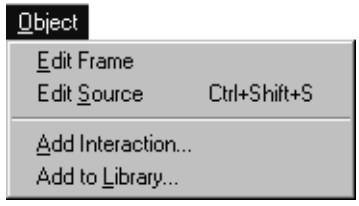
Opens a dialog box where you can select one or more files to be added to the project. You can also remove files from the project with this dialog box.

Component into Library

Opens a dialog box where you can add an external component to the Component Library.

The Object menu

The Object menu is available in the menu bar when the Project window is active.



Edit Frame

Opens the selected item in its corresponding editor. This command is enabled only when the selected object is a form or visual object. The menu name changes to reflect the type of object selected. The object could be an applet, frame, dialog, or any component type.

Edit Source

Opens the Source window for the selected object. Allows you to view the code associated with an object.

Add Interaction

Opens the Interaction Wizard where you can create an interaction for objects in your project. This command uses the selected object as the trigger component.

Add To Library

Opens the Add to Library dialog box, where you can save the selected object in the Component Library.

The Window menu

New Window

Creates a new instance of the active source window. Each new instance of the same window is incremented to indicate the number of open windows.

Workspaces

- Edit
- Debug

Selecting a workspace from the list saves your current workspace and configures your environment to the new layout.

New…allows you to create a new workspace.

Rename…allows you to rename a workspace.

Delete allows you to delete a workspace.

You can activate the display of these tools:

- Property List
- Component Library
- Hierarchy Editor
- Class Browser

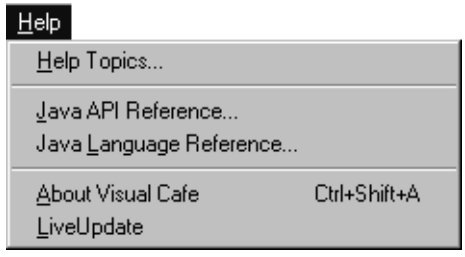
You can activate the display of these debugging windows:

- Watch
- Breakpoint
- Messages
- Threads
- Variables
- Call Stack
- Recently Used Windows

  To display one of the windows that you used recently, you can select the window name from the numbered list on the menu.

Windows… displays a list of recently used windows.

The Help menu



Help Topics

Opens the Content listing for the Visual Cafe online help. The online help provides task and context sensitive information.

Java API Reference

A reference of all Java packages, classes and interfaces for each package. Variables, constructors, and methods are also included.

Java Language Reference

Opens the Content listing for the Java Language Reference.

About Visual Cafe

Displays version and copyright information for this release.

LiveUpdate

Helps you update your software across the Internet through a Symantec Web site. Before you can use LiveUpdate, you must register at the Symantec update center; after registering, you are given a file that enables LiveUpdate. See "Updating Visual Cafe with LiveUpdate" on page 9-19.

CHAPTER 4

Working with Java Source Code

Classes are the foundation of object oriented programming. To make working with classes easier, Visual Cafe provides several powerful tools for working with classes:

- Class Browser
- Hierarchy Editor
- Event Class Wizard
- Source Code Editor

This chapter describes how to use these tools, as well as how to add your own packages or third-party packages to Visual Cafe, and how to work with events and interactions.

Using the Class Browser

The Class Browser is a three-pane window that lists all of the classes, methods and data items contained in your program. This tool provides abstraction from the underlying source files by letting you navigate and edit your classes and members quickly. In the Class Browser you are free from the clutter of other member implementations in the same source file. The Class Browser helps you work with your source code in an organized way. It's not very easy to read a 27-page source file filled with references to multiple classes. You need to be able to keep track of what each of the classes does, what data is in each class, and how the methods within the classes work.

The Class Browser window shows the class hierarchy in your project and lets you quickly navigate and add classes, modify extension relationships, and view and edit class member declarations and definitions. The Class Browser window shows data and methods for each class and an edit area for working directly with the body of a method.

You cannot switch a Class Browser window to a different project after it has been opened. To examine the classes of a different project, make the other project the active project, then open a new Class Browser window.

You only see the methods and data members for the selected class, and only the Java code for the selected member. The isolation of member source code provides an extra degree of security by ensuring that you do not unintentionally change code outside of the object's scope.

Both the Classes and Members panes support keyboard incremental searches. As you type the name of a class or member, the list of matching objects is refined until the desired class is automatically selected.

Any change made to classes or inheritance relationships is automatically changed in the associated source code and saved. Saves are done as you move between members or classes.

The bottom editing pane provides the same editing features as the Source window.

Right-mouse pop-up menus are available for each pane. You can select multiple items in the pane by using Shift-click.

To open a Class Browser window, choose Class Browser from Visual Cafe's Window menu. If you choose this command with a Class Browser window already open, the window becomes the frontmost window.

To open a new Class Browser window when one or more Class Browser windows are already open for a project, choose New Window from the Window menu.

You may change the relative size of the panes by dragging the size bars. Once you have established a new relative size for a pane, it is maintained when the window is resized. The following types of information are displayed in the active window:

- Classes, which shows all classes defined for the project after compiling the project
- Members, which shows the member functions for a selected class
- Source, which shows the source code for a class, member, or data item

Grouping and sorting classes and members

Use the following functions to group and sort classes and members. Right-click in the Class Browser and select Options...

Using the Class Options, Group/Sort tab

Use this tab to change the ordering of classes and member elements in the Class Browser.

Specifying group classes

Specify how the classes in the Classes pane are to be grouped: alphabetically, hierarchically, or by package.

Sorting members

Specify how the members in the Members pane are to be sorted in their group. Grouping of members is controlled with Group Members. The None option sorts the elements based on the order in which they were created.

Group members

This defines the grouping of members in the Member pane. The By Kind option groups the elements as Methods or Data. The By Access option groups the elements by their access type.

Controlling the display of inherited methods

Use the following functions to control the display of inherited methods.

Using the Class Options, Inheritance tab

Use this option to toggle the display of inherited methods in the Class Browser's Members pane.

Showing inherited methods

Use this option to toggle the display and activates two sub-options.

Using full method names

Use this option to add the method's package and class name to the method names in the listing.

Showing overridden methods

Use this option to add methods that are overridden by methods in the class to the Member listing.

Navigating the panes

The Classes, Source, and Members panes are all lists. You scroll a list until an item is visible. You can also type the first few letters of the item's name and the list automatically scrolls to the first item that begins with those letters.

Navigating the Classes pane

The Classes pane displays a list of your project's Java classes. This view also provides an outline in which subclasses display indented and below their parent. A class that implements multiple interfaces appears below each interface class.

You can change the ordering of the classes by using the Options command in the pop-up menu. You access this pop-up menu by right-clicking in a pane, then selecting Options. This command opens the Class Options dialog box's Group tab. By default, the classes appear alphabetically by full package name. The classes in the default package are displayed first.

To add a class, use the pop-up menu, then select Insert Class command. This command is the same as choosing Class from the main Insert menu.

If you select a class in the Classes pane, the member functions it implements are displayed in the Members Functions pane. The data members it defines are also displayed in the MembersData pane. Inherited member functions and data members are not displayed. You may display a class declaration in the Source pane by double-clicking its name in the Classes pane.

Classes do not appear in the Classes pane until the source code file that contains their definition has been compiled at least once.

Using the Classes pane

When you display the Classes pane in graphical order, you can specify (by means of the Options button) the following options:

Using the Show All Classes option

Select this option to display the entire class hierarchy. If you select a class and deselect this option, the display focuses on the selected class showing only its direct base class and direct descendents. In addition, you can "follow" the class hierarchy by clicking on each graphically displayed base/descendent class.

To change the focus from the currently displayed class, you can type the name of any other class while the class pane is active. When you type the name, that class becomes the focus of the pane regardless of the class currently displayed in the window.

Using the Show Implements option

When you select this option, multiple inheritance lines are drawn in the hierarchy. The Class Browser uses the first class mentioned in a class definition as the direct base class of the class.

Finding a class

Click in the Classes pane, then type the class name. For example, to locate java.awt.FlowLayout, type flo. Press TAB to go to the next entry with that letter sequence.

As you type, selections are made to match the text you enter. As you continue typing, the search is refined.

The class you are searching for is highlighted.

Notes:

In the Class Browser, the search is conducted by package and only expanded packages are searched. So if you want to locate a class in a particular package, you need to expand the package by clicking the +, then start typing. If you want to exclude a package from a search, then collapse the package (click the -).

If the class you want is not displayed in the pane, you might need to change what classes are displayed, as described next.

To change which classes are displayed in the Classes pane:

1. In the Classes pane, choose Classes, then Options, or right-click and choose Options.
2. In the Class Options dialog box, set the options you want:
   - In the Group/Sort tab, specify whether you want to group classes alphabetically, hierarchically, or by package.
   - In the Filter tab, specify where you want to show imported classes.

To create a new class with the Class Wizard:

1. In the Classes pane, optionally select a class you want the new class to extend.
2. Right-click in the Classes pane and choose Insert Class.

   The Insert Class Wizard appears.
3. Complete the Insert Class Wizard. "Using the Insert Class Wizard" on page 4-20 for more details.

   The new class appears in the Class Browser display and is added to the project.

Adding a class

You can add a custom class that extends an existing class, the parent class, or a developer-defined class (extensions of java.lang.Object).

Classes can be added at any time by choosing Class from the Insert menu.

You can add methods to a component from the Class Browser or Source window.

Adding a subclass from the Class Browser

As an alternative to using the Insert menu, you can add a class within the Class Browser.

To add a subclass from the Class Browser:

1. Select a class as the base class of the your new class.
2. Right-click in the Classes pane to display the pop-up menu.
3. Select the command Insert Class.
4. Complete the Insert Class Wizard.

The new class name appears in classes pane. The source pane shows the class declaration.

The Insert Class Wizard makes creating new Java classes and interfaces easier and more foolproof by setting up a complete prototype for you.

To edit a class or interface with the Edit Class Wizard:

1. In the Classes pane, select a class you want view or edit.
2. Choose Classes Edit Class, or right-click in the Classes pane and choose Edit Class.

   The Edit Class Wizard appears.
3. Complete the Edit Class Wizard. On the first page of the wizard, specify the options you need:

Options	Description
Type	Select Class if you are creating a class or Interface if you are creating an interface.
Name	Type the name of the class or interface.
Source	Type the complete path to the Java file. Click the Edit button to browse. In the Edit Class File dialog box, you can specify the package and Java file name, and the file path is displayed for you.
Package	Choose a package to add the class or interface to, or None to not add it to an existing package.
Extends	If you are defining a class, choose a class to extend from.
Access	Select whether you want to make access to your class through Package, Public, Private or Protected. Public is available only if the class name and file name are the same; Protected and Private are for inner classes only.
Final or Abstract	Select Final class or Abstract class.

4. Click Finish if you are finished with the definition, or continue by clicking Next.

The next page of the wizard appears, where you can choose the interfaces to implement.

5 From the Available interfaces list, select the interfaces you want to implement and click the downward-pointing arrow.

To move an interface from the lower list box to the upper one, select the interface and click the upward-pointing arrow.

6 Click Finish if you are finished with the definition, or continue by clicking Next.

The next page of the wizard appears, where you can choose the methods to override. Required methods already appear in the Override these methods list box.

7 From the Available methods list, select the methods you want to override and click the downward-pointing arrow.

To move a method from the lower list box to the upper one, select the method and click the upward-pointing arrow. You cannot move methods that are required.

8 If you want to review or change part of the definition, click Back.

You can go back to previous pages of the wizard.

9 Click Finish when you are finished with the definition.

The new class is inserted in the active project.

10 Complete the definition of the class or interface in the Source window or Class Browser.

11 Click OK.

The new class name appears in classes pane. The source pane shows the class declaration.

Tip: To delete a class, select the class and press Delete.

Classes defined in subprojects are not displayed in a project's Class Browser window—except for base classes from which one or more classes in a project are derived. These base classes are listed in italic in the Classes pane. You cannot examine these classes in the Members or Source panes. To browse classes defined in a subproject, make the subproject the frontmost project, then open a new Class Browser window.

Viewing and editing the source code for a class

You can view source code in the bottom editing pane of the Class Browser or in a Source window. The bottom editing pane has the same editing options as the Source window.

To display source code in the bottom editing pane, select a class in the Classes pane then a member in the Members pane.

To display source code in a Source window, from the Classes pane, select a class then choose Classes, > Go to Source, or right-click a class and choose Go to Source.

To delete a class inheritance:

1. In the Classes pane, choose Classes, then Options, or right-click and choose Options.
2. In the Group/Sort tab, specify a Class Grouping of Hierarchically, then click OK.

   When you are showing classes by a Hierarchical view and a class implements one or more interfaces, it displays below each interface.
3. In the Classes pane, select a class, then press DELETE.

   If you delete a class that is specified below an interface, that interface is deleted from the class. If you delete a class below a class, the class now inherits directly from java.lang.Object.

To delete a class:

In the Classes pane, select a class, then press DELETE.

Adding a method from the Class Browser

The Class Browser is a convenient way to add methods.

To add a method from the Class Browser:

1. In the Classes list, select the class that will contain the new method.
2. From the Insert menu, choose Member.

   The Insert Member dialog box appears.
3. Specify the member declaration.

   For data items, enter the type and member name (for example, `int nCats`). A trailing semicolon is optional. The member declaration is placed into the class declaration.

Note: The Source File field is not editable. It displays the name of the source file into which the member definition will be placed.

4 Indicate the base class access type. You can also select the appropriate member group from the members list.

This dialog box is also available with the Insert Member command on the Class Browser, Members pane pop-up menu.

5 From the Insert menu, choose Member, or use the right-mouse Insert Member command.

6 Complete the Insert Member dialog box.

7 Click OK.

The new member is selected in the Members list and the declaration appears in the editing pane.

Navigating the Members pane

The Members pane displays a list of methods and data elements in the selected class. Clicking on a member causes its associated Java source code definition or declaration to display in the editing pane.

By default, the methods and data members display in two groups with each element sorted alphabetically within the group. Each element has a color-coded icon to indicate its access privileges (green = public, yellow = protected, blue = package, and red = private.)

You can change the display of elements in the Members pane by using the Options command in the pop-up menu. This command opens the Class Options dialog box's Group/Sort tab.

To locate a method or data variable in the Members pane:

1 Select a class in the Classes pane.

Note: If the class you want is not displayed in the pane, you might need to change what classes are displayed, as described previously.

2 Click in the Members pane, then type the method or data variable name. For example, to locate the CENTER data variable for the FlowLayout class, type cen. Press TAB to go to the next entry with that letter sequence.

The method or data variable you are searching for is highlighted. The source code is displayed in the bottom editing pane.

Each member has a color-coded icon to indicate its access privileges (green is public, yellow is protected, blue is package, and red is private).

Note: If the method or data variable you want is not displayed in the pane, you might need to change what members are displayed, as described next.

To change what members are displayed in the Members pane:

1. In the Members pane, right-click and choose Options. Or while the Class Browser is the active window, choose Classes, then Options.
2. In the Class Options dialog box, set the options you want:
   - In the Group/Sort tab, specify whether you want to sort members by access (public, private, protected, and package), alphabetically, or not at all.
   - In the Group/Sort tab, specify whether you want to group members by kind or by access.
   - In the Filter tab, specify what members you want to show (choose from public, private, protected, package, final, static, and regular) and if you want to display their types.
   - In the Inheritance tab, specify whether you want to show inherited methods, and if so, if you want to display full method names or show overridden methods.

To create a new member from the Members pane:

1. In the Class pane, select a class you want to add a method or data variable to.

   Note: If the class you want is not displayed in the pane, you might need to change what classes are displayed, as described previously.

   ---

2. Right-click in the Members pane and choose Insert Member. Or while the Class Browser is the active window, choose Insert Member.

   The Insert Member dialog box appears.
3. Type the declaration and choose the access type, then click OK.

For example, int myVar ()is a valid declaration you could type. The member appears in the Members pane.

4 Select the member in the Members pane to view and edit source code in the bottom editing window.

To view attributes of a member from the Members pane:

1 In the Class pane, select a class that contains the member.

Note: If the class you want is not displayed in the pane, you might need to change what classes are displayed, as described previously.

---

2 Select the member in the Members pane.

Note: If the method or data variable you want is not displayed in the pane, you might need to change what members are displayed, as described previously.

---

3 Right-click in the Members pane and choose Member Attributes. Or while the Class Browser is the active window, choose Classes, then Member Attributes.

4 The Member Attributes dialog box appears.

5 Change the access type if needed, then click OK.

To delete a member from the Members pane:

1 In the Class pane, select a class that contains the member.

Note: If the class you want is not displayed in the pane, you might need to change what classes are displayed, as described previously.

---

2 Select a member or shift-click multiple members in the Members pane.

Note: If a method or data variable you want is not displayed in the pane, you might need to change what members are displayed, as described previously.

---

3 Right-click in the Members pane and choose Delete Member. Or while the Class Browser is the active window, choose Classes Delete Member.

The member is deleted from the class.

You can view source code in the bottom editing pane of the Class Browser or in a Source window. The bottom editing pane has the same editing options as the Source window.

To view or edit the source code of a member:

Do either of the following:

- To display source code in the bottom editing pane, select a class in the Classes pane then a member in the Members pane.
- To display source code in a Source window, from the Classes pane, select a class then choose Classes Go to Source, or right-click a class and choose Go to Source.

To rename a class or member:

1. Select the class or member in the Classes or Members pane. Then click it again.
2. When a larger edit box appears, type the new name or edit the existing name.

   The constructor is renamed for you.

To add a class or member name by dragging it into source code:

Select the class or member in the Classes or Members pane, then drag it into the bottom editing pane or a Source window.

The class or member name is added at the location you drop it.

Using the Member Attributes dialog box

Use this window to change the access type of selected members.

To access the Member Attributes dialog box:

1. Click on a member in the Members pane.
2. Choose Classes, then Member attributes.

The Members Attributes dialog box opens.

The class declaration is modified and the Members display is updated to reflect the change.

If you are editing several members simultaneously and the original access specifiers are not identical, a Don't Change option displays. This option lets you change member attributes without affecting the original access of each member.

Navigating the Source pane

The Source pane displays the source code for a class definition, a member function definition, or a data member definition. All editing operations available in source file Editor windows are available in the Source pane. Any editing operation done within the Source pane is synchronized with all open source file Editor windows.

Adding a method from the Source pane

Event handlers and other source code can be added directly in the Source pane.

To add a method from the Source pane:

1. Open the Source pane for the applet or form.
2. In the Source panel, select an event method from the Event drop-down list.

   A code template and the appropriate event handler logic is added automatically to the source code.
3. Replace the placeholder text "// to do: place event handler code here" with appropriate Java code.
4. Save the file.

Editing event handler methods

There are several ways to access an object's Java source code for event handlers.

To edit event handler methods from the Form Designer:

1. Do either of the following:
   - Double-click the form background or a component.
   - Select the object and from the Edit Source menu, choose Object.
2. From the Source window Events drop-down list, select the event that you want to edit.
3. Enhance the code block as needed.

To edit event handler methods from the Class Browser:

1. Select the object class that you want to enhance.
2. In the Members pane, double-click on the event that you want to edit.

3 In the editing pane, enhance the code block as needed.

Note: The Class Browser window does not automatically keep member function's definitions synchronized with their definitions. If you change a member function's declaration or definition in the Source pane, you must manually update the other part to match.

Configuring the Class Browser and Hierarchy Editor

You can enable multiple component selection and select confirmation options for both the Class Browser and the Class Hierarchy.

To configure the Class Browser and Hierarchy Editor:

1 From the Tools menu, choose Environment Options. Click the Editing tab.

2 In the Class Browser and Class Hierarchy area, select or clear appropriate options:.

Option	Description
Confirm Delete Member	Display a confirmation message before deleting a class member.
Confirm Inheritance Change	Display a confirmation message before applying inheritance changes.
Multiple Selection	Display a confirmation message before applying a change to multiple selected objects. Yes - allow multiple select. No: disallow multiple selects. Confirm: allow but confirm changes to multiple select.

Tip: You can also enable a horizontal scroll bar in the Class Browser source pane by selecting the scroll bar option in the Source window area.

Navigating the Hierarchy Editor

The Hierarchy Editor is a visual tool that provides you with a visual representation of the classes in your project, and their inheritance relationships. You can optionally show imports as well.

All Java programs have a hierarchical structure. You can use the Hierarchy Editor to directly manipulate the class relationships of your projects, by dragging and dropping from one class to another.

You can change inheritance by clicking the line between a parent and base class, then dragging the line by its anchor to another base class.

Double-clicking a class opens the Class Browser on the selection.

You can view all classes used by the project by choosing the View Imports command on the pop-up menu. You may also view class data and methods using the Hierarchy Editor's Member and Source Windows.

You can extend existing classes just by clicking and dragging. You can also create a new parent-child relationship by clicking a class and dragging to the desired parent class. If you double-click a class, that class is displayed in the Class Editor so you can view its data and methods.

Changes made to classes or inheritance relationships are automatically changed in the underlying source code and in all open windows in the Visual Cafe environment. For that reason, you should be careful when making changes with the Hierarchy Editor.

Note: Visual Cafe does not let you change the structure of the existing Java hierarchy.

To display the Hierarchy Editor:

While the project you want to view is active, choose Window, then Hierarchy Editor.

To enable and disable viewing imports:

While the Hierarchy Editor is the active window, choose Hierarchy View Imports or right-click and choose View Imports to toggle the display.

To locate a class in the display:

While the Hierarchy Editor is the active window, type the class name. For example, to locate java.awt.FlowLayout, type flo.

As you type, selections are made to match the text you enter. As you continue typing, the search is refined.

The class you are searching for is highlighted.

To change the inheritance hierarchy:

Click the line between a parent and base class, then drag the line by its anchor to another base class.

To remove an inheritance:

Select the line between a parent and base class, then right-click and choose Remove Inheritance.

The class now extends directly from `java.lang.Object`.

To create a new class:

1. Optionally select a class you want the new class to extend.
2. Drag from the class into the background space, or right-click and choose Insert Class.

   The Insert Class Wizard appears.
3. Complete the Insert Class Wizard. See "Using the Insert Class Wizard" on page 4-20 in this chapter.

   The new class appears in the Hierarchy Editor display and is added to the project.

To edit an existing class:

1. Select a class you want view or edit.
2. Right-click and choose Edit Class. Or choose Hierarchy, then Edit Class.

   The Edit Class Wizard appears.
3. Complete the Edit Class Wizard. See "Using the Insert Class Wizard" on page 4-20 in this chapter for more information.

To view a class in a Source window

Select a class then choose Hierarchy, then Go to Source, or right-click a class and choose Go to Source.

To view a class in the Class Browser

Double-click a class in the Hierarchy Editor.

Working with subclasses

As an alternative to using the Insert menu, you can add a subclass within the Hierarchy Editor, Project window, or Class Browser.

To add a subclass from the Hierarchy Editor:

1. Select the parent/base class.
2. Drag the selection and release it in any whitespace of the window background.
3. Complete the Insert Class wizard. See "Using the Insert Class Wizard" on page 4-20 in this chapter for more information.

To add classes to an existing package from within the Project window:

1. In the Project window, click the Packages tab.
2. Select the package to which you want to add the class.
3. Choose Insert, then Class.
4. Complete the Insert Class wizard. See "Using the Insert Class Wizard" on page 4-20 in this chapter for more information.

To add a class within the Class Browser:

1. Select a class as the base class of the your new class.
2. Right-click in the Classes pane to display the pop-up menu.
3. Select the command Insert Class
4. Complete the Insert Class wizard. See "Using the Insert Class Wizard" on page 4-20 in this chapter for more information.

   The new class name appears in classes pane. The source pane shows the class declaration.

Note: To delete a class, select the class and press DELETE.

Changing inheritance relationships

You can change inheritance relationships by clicking on the line between a parent and base class, then dragging the line by its anchor to another base class.

Double-clicking on a class opens the Class Browser on the selection.

You can view all classes used by the project by choosing the View Imports command on the pop-up menu.

Deleting inheritance relationships

Deleting the inheritance relationship between a class and its parent can be done in the Hierarchy Editor.

To delete an inheritance relationship:

1. Select the line that links a class and its parent.
2. Right-click on the window to display the pop-up menu.
3. Choose Delete Inheritance.

Using the Class Attributes dialog box

You can change the class name and its base class.

To change the attributes for the selected class:

Do either of the following:

- Choose Classes, then Class Attributes
- Hierarchy, then Class Attributes

Using the Insert Class Wizard

The Insert Class Wizard makes creating new Java classes and interfaces easier and more foolproof by setting up a complete prototype for you.

To define a new class or interface with the Create Class Wizard:

1. Do one of the following:
   - From the Insert menu, choose Class.

- Select a class or interface in the Classes pane of the Class Browser, then right-click and choose Insert Class.
- Select a class or interface in the Hierarchy Editor, then drag a line from it.

If you select a class, that class becomes the default class to inherit from.

To edit a class or interface with the Edit Class Wizard:

1. Do either of the following:
   - Select a class or interface in the Classes pane of the Class Browser, then choose Class Attributes from the Classes menu or right-click and choose Class Attributes.
   - Select a class or interface in the Hierarchy Editor, then right-click and choose Class Attributes.
2. On the first page of the wizard, specify the options you need:

Option	Description
Type	Select Class if you are creating a class or Interface if you are creating an interface.
Name	Type the name of the class or interface.
Source	Type the complete path to the Java file. Click the Edit button to browse. In the Edit Class File dialog box, you can specify the package and Java file name, and the file path is displayed for you.
Package	Choose a package to add the class or interface to, or None to not add it to an existing package.
Extends	If you are defining a class, choose a class to extend from.
Access	Select whether you want to make access to your class through Package, Public, Private, or Protected. Public is available only if the class name and file name are the same; Protected and Private are for inner classes only.
Final or Abstract	Select Final class or Abstract class.

3. Click Finish if you are finished with the definition, or continue by clicking Next.

   The next page of the wizard appears, where you can choose the interfaces to implement.

4. From the Available interfaces list, select the interfaces you want to implement and click the downward-pointing arrow.

   To move an interface from the lower list box to the upper one, select the interface and click the upward-pointing arrow.

5. Click Finish if you are finished with the definition, or continue by clicking Next.

   The next page of the wizard appears, where you can choose the methods to override. Required methods already appear in the Override these methods list box.

6. From the Available methods list, select the methods you want to override and click the downward-pointing arrow.

   To move a method from the lower list box to the upper one, select the method and click the upward-pointing arrow. You cannot move methods that are required.

7. If you want to review or change part of the definition, click Back.

   You can go back to previous pages of the wizard.

8. Click Finish when you are finished with the definition.

   The new class is inserted in the active project.

9. Complete the definition of the class or interface in the Source window or Class Browser.

Using the Source Editor

The source editor supplies standard Windows functions for cutting, copying, pasting, and deleting text. These functions can be accessed through either the Edit menu or the pop-up menu. In addition, by clicking and dragging a block of selected text, you can reposition the text anywhere in the buffer. Press the Control key while releasing the block to copy rather than move the block. When you are dragging text blocks around in this way, a small outlined box and “+” is drawn next to the cursor to indicate this mode.

Text selection is accomplished using one of three standard techniques: clicking and dragging the mouse, shift-clicking, or ALT-clicking. The ALT+click drag performs a column select.

Creating a new document

You can create documents from within Visual Cafe. You may want to create documents for custom HTML pages and Java files.

To create a new document, open the File Menu and choose New File. Create your document in the Source window and save it. When you save the file, be sure to select the correct document type: HTML, Java, definition.

Adding code to a Java source file

Code can be added to a class source code from the Source window or Class Browser editing pane.

Although you can create many applets with Visual Cafe that do not require any custom Java code, you may want to add code for advanced applets and for applications.

Custom code may be added for error handling, event control, and complex component relationships and behavior.

You can bind code to any component and to menu commands.

Caution: Custom code is not deleted from the source file when you delete a component. You must manually maintain any file that contains custom code.

Guidelines and warnings

- Whenever possible, make object changes using Visual Cafe, versus adding code directly to the source file. For example, adding components, classes or class members, and changing component properties can be done through the Component Palette, using a menu selection, or making a change in the Property List window.
- Do not add to or modify custom code within code blocks that are regenerated. Visual Cafe places special comments in the source code to indicate the beginning and end of blocks of code that it needs to manage. These code blocks start with //{{ and end with //}}. For example, if you add a button to a form, Visual Cafe generates the following code:

```
//{{DECLARE_CONTROLS
```

```
java.awt.Button button1;
//}}

//{{INIT_CONTROLS
button1 = new java.awt.Button("button");
button1.reshape(63,77,88,30);
add(button1);
//}}
```

Avoid moving these code blocks. But if you do, be sure to move the entire code block, including the special comments.

Editing source code

The purpose of the Visual Cafe source editor is to create, examine, and modify your project's source files. Because these files are standard text files, you can, in principle, use any source editor to work with them. The source editor is designed to work in concert with other Visual Cafe tools.

The Visual Cafe Source window and the Class Browser editing pane both share the same editing functionality. The editor uses standard Windows editing commands and has special features that make working with Java files easier. For example, the editor can automatically indent or unindent after braces and can check delimiters.

In addition, the source editor can display keywords, preprocessor directives, and comments in special font styles and colors. This technique helps track errors in source code while you are editing. For example, an unmatched comment (/* without a matching */) turns a large part of the code a different color, making it obvious where the problem lies. Also, keywords and preprocessor directives are easier to spot when they are in a different color or font style. Misspelled keywords can be caught immediately when they remain displayed in the default font.

Source windows are an integral part of the Visual Cafe environment and work together with other Visual Cafe windows to make application development easier. For example, Visual Cafe automatically saves all files open in Source windows when you rebuild your project. During compilation, error messages are displayed in the Messages window; when you double-click on an error message, Visual Cafe opens a Source window on the corresponding source file, if necessary, and then jumps to the line in the source code that caused the error.

Correcting your source code

If there are syntax errors in your source code, Visual Cafe flags them in the Messages window after a compile. You can easily navigate to each error directly from the Messages window.

To go directly to a syntax error from the Messages window:

1. From the Window menu, choose Messages to bring the Messages window to the front.
2. Double-click on any error message to go to that error.

The file containing the error opens at the offending line within a Source window. Once the file opens, you can work on your source code.

Viewing a component's Java source code

You can open a component's Java source file to view the component's behavior.

To view a component's Java source code:

Do either of the following:

- Click on the class you want to view
- In the Form Designer, double-click on the component.

The source code appears in the Source Pane.

Enhancing an object's Java source code

With Visual Cafe's extensive Component Library, you can create many applets and applications without writing any Java Code. However, there may be specific behaviors that require custom code to be added to the Visual Cafe source file.

You may want to add or modify an object's Java code to:

- add event behavior
- perform exception handling
- implement runtime security for applications
- extend a method

Before you add custom code, you should create your project and use Visual Cafe's powerful visual tools to create your forms, applets, menu

bars, and components. After you add the objects to your project, arrange components in the Form Designer, test run the forms, and modify the look of individual components with the Property List.

When you are visually satisfied with your project, you can then add any necessary code enhancements using the Source window or Class Browser.

Caution: Visual Cafe only maintains the code that it automatically generates. This means that when you add code to an object and later delete the object from the form or project, only the automatically generated code is removed during the object deletion. You must edit the appropriate Java source files to remove your own code references.

To modify a component's source code:

1. Do either of the following:
   - In the Form Designer, double-click the component.
   - Select the component in the Source window or Form Designer, then choose Edit Source from the pop-up menu.
2. In the Source window, add your custom code to the file.

Binding code to a form or component

Every form and component has a set of events that users can trigger at runtime. To bind code to an event, you create an event method in the source file and add custom code or use the Interaction Wizard. When the event occurs, the code in your event method executes.

To bind code to a form or component:

1. In the Form Designer, open the Source window for the object using either of these methods:
   - Double-click on the appropriate component or on the form background.
   - Select an object and choose Edit Open Source from the Object menu.
2. In the Source window, select the event that you want to bind to from the Event/Method drop-down list.
3. The method template is added to the source file. The cursor insertion point is positioned inside the template. Code is also added to the

HandlerEvent method so that when an event occurs in the object the code block get executed.

4 Enter the appropriate code into the method template.

Binding code to a menu command

Code can be bound to a menu item just as it can be bound to form or component. Menu items respond to one event, Action. This event occurs when the user selects the menu command.

To bind code to a menu command:

1 Open the menu bar in the Menu Editor.

2 Select the menu item.

3 Choose Edit Source from the pop-up menu.

4 In the Source window, select the onAction event from the Event/ Method drop-down list.

5 Add the appropriate Java code to the event code block.

Programming hot keys

You can configure buttons to accept keyboard input, such as CONTROL+C. When certain keys are pressed, it would be the same as clicking the button. Here is a code sample:

```
/*This simple extension of the java.awt.Frame class contains all
 the elements necessary to act as the main window of an
 application.*/
import java.awt.*;
public class Frame1 extends Frame
{
    public Frame1()
    {
    // This code is automatically generated by Visual Cafe
    // when you add components to the visual environment.
    // It instantiates and initializes the components. To
    // modify the code, only use code syntax that matches
    // what Visual Cafe can generate, or Visual Cafe
    // may be unable to back parse your Java file into
    // its visual environment.

    //{{INIT_CONTROLS
```

```
setLayout(null);
setVisible(false);
setSize(insets().left + insets().right + 405,insets().top +
 insets().bottom + 305);
openFileDialog1 = new java.awt.FileDialog(this);
openFileDialog1.setMode(FileDialog.LOAD);
openFileDialog1.setTitle("Open");
//$$ openFileDialog1.move(36,276);
setTitle("A Basic Application");
//}}

//{{INIT_MENUS
mainMenuBar = new java.awt.MenuBar();

menu1 = new java.awt.Menu("File");
miNew = new java.awt.MenuItem("New"); menu1.add(miNew);
miOpen = new java.awt.MenuItem("Open..."); menu1.add(miOpen);
miSave = new java.awt.MenuItem("Save"); menu1.add(miSave);
miSaveAs = new java.awt.MenuItem("Save As...");
 menu1.add(miSaveAs);
menu1.addSeparator();
miExit = new java.awt.MenuItem("Exit"); menu1.add(miExit);
mainMenuBar.add(menu1);

menu2 = new java.awt.Menu("Edit");
miCut = new java.awt.MenuItem("Cut"); menu2.add(miCut);
miCopy = new java.awt.MenuItem("Copy"); menu2.add(miCopy);
miPaste = new java.awt.MenuItem("Paste"); menu2.add(miPaste);
mainMenuBar.add(menu2);

menu3 = new java.awt.Menu("Help");
mainMenuBar.setHelpMenu(menu3);
miAbout = new java.awt.MenuItem("About..");
 menu3.add(miAbout);
mainMenuBar.add(menu3);setMenuBar(mainMenuBar);
//$$ mainMenuBar.move(4,277);
//}}

//{{REGISTER_LISTENERS
SymWindow lSymWindow = new SymWindow();
addWindowListener(lSymWindow);
```

```
    SymAction lSymAction = new SymAction();
    miOpen.addActionListener(lSymAction);
    miAbout.addActionListener(lSymAction);
    miExit.addActionListener(lSymAction);
    SymKey lSymKey = new SymKey();
    addKeyListener(lSymKey);
    //}}
    }

    public Frame1(String title)
    {
        this();
        setTitle(title);
    }
    public synchronized void show()
    {
        move(50, 50);
        super.show();
    }
    static public void main(String args[]) {
    (new Frame1()).show();
    }
    public void addNotify()
    {
        Dimension d = etSize();
        super.addNotify();
        if (fComponentsAdjusted)
            return;
// Adjust components according to the insets
        setSize(insets().left + insets().right + d.width,
          insets().top + insets().bottom + d.height);
        Component components[] = getComponents();
        for (int i = 0; i < components.length; i++) {
            Point p = components[i].getLocation();
            p.translate(insets().left, insets().top);
            components[i].setLocation(p);
        }
        fComponentsAdjusted = true;
}
//Used for addNotify check.
```

```
boolean fComponentsAdjusted = false;

//{{DECLARE_CONTROLS
java.awt.FileDialog openFileDialog1; //}}

//{{DECLARE_MENUS
java.awt.MenuBar mainMenuBar;
java.awt.Menu menu1;
java.awt.MenuItem miNew;
java.awt.MenuItem miOpen;
java.awt.MenuItem miSave;
java.awt.MenuItem miSaveAs;
java.awt.MenuItem miExit;
java.awt.Menu menu2;
java.awt.MenuItem miCut;
java.awt.MenuItem miCopy;
java.awt.MenuItem miPaste;
java.awt.Menu menu3;
java.awt.MenuItem miAbout;
//}}

class SymWindow extends java.awt.event.WindowAdapter
{
    public void windowClosing(java.awt.event.WindowEvent event)
    {
        Object object = event.getSource();
        if (object == Frame1.this)
            Frame1_WindowClosing(event);
    }
}

void Frame1_WindowClosing(java.awt.event.WindowEvent event)
{
    hide();          // hide the Frame
    dispose(); // free the system resources
    System.exit(0); // close the application
}
class SymAction implements java.awt.event.ActionListener
{
    public void actionPerformed(java.awt.event.ActionEvent event)
```

```
    {
        Object object = event.getSource();
        if (object == miOpen)
        miOpen_Action(event);
        else if (object == miAbout)
        miAbout_Action(event);
        else if (object == miExit
        miExit_Action(event);
    }
}

void miAbout_Action(java.awt.event.ActionEvent event)
{
    //{{CONNECTION
    // Action from About Create and show as modal
    (new AboutDialog(this, true)).show();
    //}}
}
void miExit_Action(java.awt.event.ActionEvent event)

{
    //{{CONNECTION
    // Action from Exit Create and show as modal
    (new QuitDialog(this, true)).show();
    //}}
}

void miOpen_Action(java.awt.event.ActionEvent event)
{
    //{{CONNECTION
    // Action from Open... Show the OpenFileDialog
     openFileDialog1.show();
    //}}
}

class SymKey extends java.awt.event.KeyAdapter
{
    public void keyPressed(java.awt.event.KeyEvent event)
    {
        Object object = event.getSource();
```

```
        if (object == Frame1.this)
        Frame1_KeyPress(event);
    }
}

void Frame1_KeyPress(java.awt.event.KeyEvent event)
{
    // to do: code goes here.
    System.out.println("Button Key Press :" +
     event.getKeyText(event.getKeyCode()));
    }
}
```

Moving around in a file with the Search menu

You can move the insertion point in a file by using the mouse or keyboard in the standard Windows text-editing manner. In addition, you can jump to specific points in a file using commands from the Search menu.

The following are common tasks that you do while editing. Other "Go To" functionality is available on the Search menu.

Jumping to a matching delimiter

A common problem in source code is parentheses (( )), brackets ([]), and braces ({}) that don't match. To find the other half of a pair of these delimiters, position the insertion point in front of one of the delimiters and choose Search, then Go to Matching Delimiter. The insertion point moves to the other half of the pair.

Delimiter checking can also be done automatically using the source format options. The Source Editor looks for any parenthesis, bracket, or brace, including text in strings and comments.

To jump to a specific line:

1. From the Search menu, choose Go To Line.

   The Go To Line dialog box appears.

2. Type the line number in the text box and click OK.

   The editor moves the insertion point to the beginning of the specified line.

To jump to a function:

1. From the Search menu, choose Go to Function.

   The Go to Function dialog box appears.

2. Select a function name from the scrolling list or type in a function name.

   The insertion point then moves to the beginning of the specified function.

Searching through and comparing multiple files

The global find feature of the Source Editor provides a powerful means of locating a string in any set of files.

To search for a string in multiple files:

From the Search menu, choose Find in Files. You can choose to search:

- All source files in the current project
- All files listed in the Search window (which opens after the first search)
- All files matching the criteria you specify, including filename, directory, date, time, and file attributes

To compare two files:

From the Search menu, choose Compare Files.

Adding a method from the Source window

Event handlers and other source code can be added directly in the Source window.

To add a method from the Source window:

1. Open the Source window for the form.
2. In the Source window, select an event handler from the Events/ Methods drop-down list.

   The appropriate event handler logic is added automatically to the source code.
3. Replace the placeholder text "// to do: place event handler code here" with appropriate Java code.
4. Save the file.

Enhancing Java code for a component

With Visual Cafe's extensive Component Library, you can create many applets and applications without writing any Java code. However, there may be specific behaviors that require custom code to be added to the Visual Cafe source file.

You may want to add or modify Java code for a component to:

- add event behavior
- perform exception handling
- implement runtime security for applications
- extend a method

Before you add custom code, you should create your project and use Visual Cafe's powerful visual tools to create your forms. You can add components to your project, arrange components in the Form Designer, modify the look of individual components with the Property List, and add interactions between components or a component and itself with the Interaction Wizard. It is a good idea to do as much as you can with the Visual Cafe tools before you add your custom code.

When you are visually satisfied with your project, you can then add any necessary code enhancements using the Source window or Class Browser.

Caution: Visual Cafe only maintains the code that it automatically generates. This means that when you add code to an component and later delete the component from the form or project, only the automatically generated code is removed during the component deletion. You must edit the appropriate Java source files to remove your own code references.

Specifying the search file type and location

You can specify the search criteria, the type of files to search, and the files location on the Name & Location tab of the Find in Files dialog box.

To specify search criteria:

1 Select the appropriate search criteria options at the bottom of the window.

To	Do
Search using regular expressions	Choose what you want to search for from the pop-up menu of regular expressions. This option is the default.
Search with wildcard symbol	Select the Match Wildcards option.
Search with exact case matching	Select the Match case option.
Search for an exact match to a whole word	Select the Match whole word only option. This option limits matches to files that contain the search criteria string preceded and followed by a space, tab, or punctuation character, or search string that start or end a line.

2 Specify the search pattern in the Find what field. The drop-down list displays the previous sixteen search strings. If Regular expression is selected, you can use the more button to select valid regular expression characters.

3 Specify the types of files to search by entering file extensions or selecting extensions from the drop-down list.

4 Select the scope of the search. You can specify a folder, the current project, or the last set of files found in a previous search.

5 To expand a search into subfolders, select the Search subfolders option.

Specifying the file search path

From the Directories tab of the Project Options dialog box, you can add and remove directories that Visual Cafe uses to locate source and class files and control where Visual Cafe puts compiler output files. This view changes the option set for the Debug or Final release type. See "Specifying whether builds are debug or final" on page 3-39, for more information.

You can also set class path information for the entire Visual Cafe environment. Remember that Visual Cafe does not inherit Windows class path information unless it is set up to do so.

To access the Directories view:

1. Activate the Project window of the project you want to work with.
2. From the Project menu, choose Options.
3. In the Project Options dialog box, click the Directories tab.

Specifying source file search paths

The source search path applies to Java source files and any text file that can be opened in the Source window.

Note: You can set the source search path for the entire Visual Cafe environment from the Environment Options dialog box and with the javainc statement in the \VisualCafe\Bin\sc.ini file.

To specify a source file search path:

1. Go to the Directories view of the Project Options dialog box.
2. Choose Source files in the Show directories field.

   A list of directories that can contain source files appears. The directory order affects the search order: the topmost directory is the first to be searched. The default setting, the project directory, does not appear in the list; it is the first directory in the search order.
3. Modify the list as needed:
   - To change the order that directories are searched, select a directory and move it with the Up Arrow and Down Arrow buttons.
   - To delete a directory from the list, select the directory and click Delete.
   - To add a directory to the list, select the blank entry (marked by an empty box) at the bottom of the list and type the directory name, including the full path. Or click the New button (located above the text box), then select a directory by clicking the Browse button that appears in the field. You can also use the New button to insert a new entry above the selected entry.
4. Click OK.

   The change takes effect immediately.

Setting Advanced Search Criteria

You specify file attribute and modification search criteria from the Find in Files dialog box's Advanced tab.

Set attribute criteria by enabling the appropriate attribute options. This narrows the search scope.

File attributes are Archive, Read Only, System, and Hidden.

Note: The Attributes checkboxes are 3-state checkboxes. If an attribute is enabled, then files with the given attribute are searched. If an attribute is disabled, then files without the given attribute are searched. If an attribute is grayed, the attribute is ignored when Visual Cafe decides which files to search.

Specify modification criteria by selecting the Files created or modified option.

Set the appropriate values in the Date and Time fields. The field value "ignore" disables the associated data or time field.

Date: Select Ignore to ignore the date. Otherwise, specify a date and one of the options. For instance, specify "Is" and 11/6/94 to search files last modified on November 6, 1994, or "Greater" and 4/1/90 to search files last modified after April 1, 1990.

Time: Select Ignore to ignore the time. Otherwise, specify a time and one of the options.

Adding packages to Visual Cafe

The standard Java packages and the Symantec Visual Cafe packages are available to you as you develop Java programs in the Visual Cafe environment. You can also make other packages available to Visual Cafe projects by adding them to the Visual Cafe class path. For example, you could add third-party or your own packages containing components or utilities. You can set the class path for a project or for the Visual Cafe environment.

To add packages to Visual Cafe, the package needs to be in the classpath. Remember to keep the directory structure of your package intact, and

make sure that your file names use the same upper and lower case letters, exactly as they were before copying the package.

If you add a Java import statement or use part of the new package in your Java source code, the package appears in the Packages view.

If you declare your application or applet as a part of another package, you must specify the output directory in the Package Destination line Directories tab of the Project options window.

Using the Package view

For Visual Cafe version 2.0, Packages view supports drag and drop as well as package creation. Click on the Packages tab to start using these sophisticated features.

Using drag-and-drop

In Packages view, you can drag and drop files between packages, drag files from a file source—like the Windows Explorer—and to add them to a project, and you can drag files to the trash to remove them from the project. You can also press Delete to remove files from a project. Any of these actions can be undone.

Creating packages

To create a new package you choose Package from the Insert menu. When you choose this command a new package is created and added to the end of the list. The name it made is immediately editable. If necessary, the list is automatically scrolled to display the new package.

The Package Group command in the Insert menu is only available in Packages view. It replaces the Group command which is available from the Object Library window.

When you create a new package a placeholder is created into which you can drag files. If you don't add any files to the package, the package placeholder remains accessible in Packages view while the project is open. When the project is closed, the reference is deleted and when you reopen the project, the empty package is no longer in the list.

Using the Context menu

The following context menu is available in Package view:

Tip: The default view is always on top. You can change the default view by right-clicking on the tab, then selecting the make-default option at the bottom of the pop-up window.

You can also make a view the only view in the Project window by selecting the options with checkmarks. This action removes the checkmark and the particular view from the Project window. To restore a particular view to the Project window, right-click any existing tab and select the view you want. The view is restored.

Working with events

An event is an asynchronous signal that a source program element sends to target program element to notify the target that some specific behavior has occurred. Java defines the events it supports in the Events class, which extends the Object class.

Using events with your own components

In Visual Cafe, you can usually route an event to a target component and specify the action to be taken, via the Interaction Wizard.

Some Visual Cafe components require you to add code to your source code file. To handle a standard event in a way unique for a component, override the action( ) method in your target component. Pass that standard event to `action( )` instead of `eventHandler( )` in your target component.

A component does not need to handle all events passed to it. Events retrieved but not handled by a target component are passed to the component's parent object. Events eventually pass to the component class, which discards events it doesn't recognize.

Adding an event to a component

Event-handler code is automatically added to a component's source when you add an event to a component.

Visual Cafe changes the source code in four ways when you add an event handler from the Source window:

- First, Visual Cafe generates an adapter or listener implementation for the event. If one was already generated, it is used with the new interaction.
- Second, in the adapter/listener class, Visual Cafe generates a check for the object requesting the handling of the event and a call to the event handler.
- Third, Visual Cafe instantiates this adapter/listener class and generates the registration (specifically, object.addtypeListener or object.addtypeAdapter) after the comment tag REGISTER_LISTENERS. If the adapter/listener class has already been instantiated, it is used.
- Fourth, an event handler is generated. If the event handler already exists, it is used.

If instead you use the Interaction Wizard to create an interaction, Visual Cafe makes the four modifications mentioned previously, plus the interaction specified in the wizard is generated in the event handler. See "Working with the Interaction wizard" on page 4-41 for more information.

To add an event handler from the Source window:

1. In the Objects drop-down list of the Source window, choose a component.
2. In the Events/Methods drop-down list, choose the event or method.

   Existing events and methods are shown in bold. If you choose an event or method that is not bold, it is created for you.
3. In the event handler, replace the placeholder text // to do: code goes here with appropriate Java code.

Editing an event handler

You can conveniently edit event handlers from the Class Browser and Source window.

To edit an event handler from the Class Browser

- Select a class in the Classes pane then a member in the Members pane, and edit the member in the bottom Source pane. See "Finding a class" on page 4-6 for more information.

To edit an event handler in the Source window

1. In the Objects drop-down list of the Source window, choose an object.
2. In the Events/Methods drop-down list, choose the event or method.

   Existing events and methods are shown in bold. If you choose an event or method that is not bold, it is created for you.
3. Edit the Java code.

Deleting an event handler

Deleting an event-handler is the same as deleting an interaction. See Using the Interaction Wizard for more information.

Editing event methods

There are several ways to access an object's Java source code for event editing.

To edit event methods from the Form Designer:

1. Do either of the following:
   - Double-click the form background or a component.
   - Select the object and from the Object menu, choose Edit Source.
2. From the Source window Events drop-down list, select the event that you want to edit.
3. Enhance the code block as needed.

To edit event methods from the Class Browser:

1. Select the object class that you want to enhance.
2. In the Members pane, double-click on the event that you want to edit.
3. In the editing pane, enhance the code block as needed.

Working with the Interaction wizard

One of the most powerful features of Visual Cafe is the ability to quickly build a relationship between two components. Anytime a button is pushed, or a box is enabled, a program action called an event is generated. To link those events to a corresponding action in your program, you need an event handler that responds to that event.

In Visual Cafe, this event handler is called an interaction. The Interaction Wizard lets you graphically build relationships between components, or between a component and itself. For example, double-clicking an item in a list box could remove the item from the list box and add it to another list box. Also, you can create an interaction on an animation component where a mouse click anywhere within the boundaries of a component starts an animation in that component; another click ends the animation. In a slide show, a button click could cause the next image to be displayed. Visual Cafe automatically generates the necessary code for the specified relationship.I

Changes between the JDK 1.0 and 1.1 event models

In the Visual Cafe 1.0, the key elements of an interaction are the trigger event and the action component. The trigger event is the originator of the interaction and is associated with a component. The trigger event determines when the interaction happens. The action component is the component on which a defined “action” happens. The action specifies what to do when a condition is met.

For example, you can connect a button (the trigger component) to a text box (the action component) so that when the user clicks the button (the trigger event), the associated text box is enabled for data input (the action).

An interaction does not have to contain two components. You can create an interaction where the trigger and action component are the same. Interactions are implemented as methods in the component’s container class.

In the 1.1 event model, when an interaction is created, code for the event handler, listener registration, and adapter/listener class with a call is automatically generated. If you want to delete an interaction, you must manually delete all of this code.

Visual Cafe changes the source code in five ways when you create an interaction:

1. Visual Cafe generates an adapter or listener implementation for the event. If one was already generated, it is used with the new interaction.
2. In the adapter/listener class, Visual Cafe generates a check for the object requesting the handling of the event and a call to the event handler.

3. Visual Cafe instantiates this adapter/listener class and generates the registration (specifically, object.addtypeListener or object.addtypeAdapter) after the REGISTER_LISTENERS tag. If the adapter/listener class has already been instantiated, it is used.
4. An event handler is generated. If the event handler already exists, it is used.
5. The interaction specified in the wizard is generated in the event handler.

For example, if you have a button, called NextButton, that displays the next image of a slide show, called VacationSlides, the event handler could look like this and appear toward the end of the Java source file:

```
void NextButton_Action(java.awt.event.ActionEvent event)
{
    // to do:  code goes here.
    //{{CONNECTION
    // Go to the SlideShow's next image
    VacationSlides.nextImage();
    //}}
}
```

More toward the middle of the Java source file is the listener registration:

```
//{{REGISTER_LISTENERS
SymAction lSymAction = new SymAction();
NextButton.addActionListener(lSymAction);
//}}
```

Toward the end of the Java source file is the adapter/listener class with a call to the event handler:

```
class SymAction implements java.awt.event.ActionListener
{
    public void actionPerformed(java.awt.event.ActionEvent
     event)
    {
        Object object = event.getSource();
        if (object == NextButton)
            NextButton_Action(event);
    }
}
```

Then, if you add an interaction between VacationSlides and a Label component, Visual Cafe generates a new event handler for VacationSlides, and the listener registration code changes:

```
//{{REGISTER_LISTENERS
SymAction lSymAction = new SymAction();
NextButton.addActionListener(lSymAction);
VacationSlides.addActionListener(lSymAction);
//}}
```

And the adapter/listener class code changes:

```
class SymAction implements java.awt.event.ActionListener
{
    public void actionPerformed(java.awt.event.ActionEvent
      event)
    {
        Object object = event.getSource();
        if (object == NextButton)
        NextButton_Action(event);
        else if (object == VacationSlides)
        VacationSlides_SlideChanged(event);
    }
}
```

Creating an interaction

With the Interaction Wizard, you can create an interaction:

- by connecting two components on a form in the Form Designer
- by connecting two components in the Project window
- by connecting two components across the Project window and Form Designer (such as a button in the Form Designer and a dialog component in the Project window)
- by connecting a component to itself
- by connecting a form and a component contained by the form

Note: The components must be within the same project.

To create an interaction relationship between two components:

1 In the Form Designer or Project window, select the trigger component by using one of these methods:

- Right-click the component and select Add Interaction.
- Click the Interaction tool icon, then click the trigger component.

2 Drag a line from the trigger component to the action component within the same project.

The action component is highlighted to help you identify which component you selected. If a component will not highlight, the interaction could be inappropriate.

- To connect a component to itself, double-click the component.

After you release the mouse button, the Interaction Wizard appears.

Tip: Before you release the mouse button, you can press ESCAPE to cancel the interaction, or move the interaction line to another component.

---

3 In the Interaction Wizard, verify that the component names are correct.

The trigger component should appear in the Start an interaction field title; if it is wrong, you need to start over.

The action component should appear in the Select the item you want to interact with field; if it is wrong, choose another component.

4 In the Start an interaction field, select the event that activates the interaction.

5 In the Choose what you want to happen field, select an action to invoke when the interaction event occurs.

- If the action takes no parameters, the interaction is finished.
- If the action takes parameters, the Next button is enabled. When you click Next, a second window displays allowing you to fill in those parameters.

6 In the second window, specify the interaction condition.

Tip: A variable must be of the type listed for the second radio button, A type constant or expression, for it to appear. If you define a variable of this type and the variable is in scope, it will appear in the variable list.

---

7 Click Finish.

In the Java source file, code is generated for the call handler, listener registration, and adapter/listener class.

In the source code, Visual Cafe performs five edits:

- First, Visual Cafe generates an adapter or listener implementation for the event. If one was already generated, it is used with the new interaction.
- Second, in the adapter/listener class, Visual Cafe generates a check for the object requesting the handling of the event and a call to the event handler.
- Third, Visual Cafe instantiates this adapter/listener class and generates the registration (specifically, object.addtypeListener or object.addtypeAdapter) after the REGISTER_LISTENERS tag. If the adapter/listener class has already been instantiated, it is used.
- Fourth, an event handler is generated. If the event handler already exists, it is used.
- Fifth, the interaction specified in the wizard is generated in the event handler.

To connect components with the Interaction Wizard:

1. Select a component in the Form Designer or Project window.
2. Choose Object, then Add Interaction.
3. Complete the Interaction Wizard as described in the previous procedures.

Changing an existing interaction

Interactions are implemented as methods in the container class of a component. The interaction methods are called an event handlers.

To change an existing interaction:

1. Make the project active by clicking the Project window.
2. Choose Window, then Class Browser

   You should see a listing of project classes in the Classes pane.

   Note: If the class you want is not displayed in the pane, you might need to change what classes are displayed by choosing Classes, then Options.

   ---

3. In the Classes list, click the container class that contains the trigger component.

   In the Members pane, you should see a list of methods and variables associated with the container class.

4 Identify the appropriate event handler. Event handlers are named using a combination of the component name and the trigger event; for example, `NextButton_Action`.

Note: If you rename a component after an event handler is created, the method is not renamed.

---

5 Display the method's source code by clicking the event handler. The interaction code is marked by the //{{CONNECTION comment and a comment that explains the interaction.

For example:

```
void NextButton_Action(java.awt.event.ActionEvent event)
{
    // to do:  code goes here.
    //{{CONNECTION
    // Go to the SlideShow's next image
    VacationSlides.nextImage();
    //}}
}
```

6 In the editing pane, make any source code changes you need.

Deleting an interaction

To delete an interaction, you delete the interaction in the event handler, as well as the applicable portions of the listener registration and the event handler call in the adapter/listener class.

For example, if you have a button, called `NextButton`, that displays the next image of a slide show, called `VacationSlides`, the event handler could look like this and appear toward the end of the Java source file:

```
void NextButton_Action(java.awt.event.ActionEvent event)
{
    // to do:  code goes here.
    //{{CONNECTION
    // Go to the SlideShow's next image
    VacationSlides.nextImage();
    //}}
}
```

More toward the middle of the Java source file would be the listener registration:

```
//{{REGISTER_LISTENERS
SymAction lSymAction = new SymAction();
NextButton.addActionListener(lSymAction);
//}}
```

Toward the end of the Java source file would be the adapter/listener class with a call to the event handler:

```
class SymAction implements java.awt.event.ActionListener
{
    public void actionPerformed(java.awt.event.ActionEvent
     event)
    {
        Object object = event.getSource();
        if (object == NextButton)
            NextButton_Action(event);
    }
}
```

If you then add an interaction between VacationSlides and a Label component, Visual Cafe would generate a new event handler for VacationSlides, and the listener registration code would change:

```
//{{REGISTER_LISTENERS
SymAction lSymAction = new SymAction();
NextButton.addActionListener(lSymAction);
VacationSlides.addActionListener(lSymAction);
//}}
```

And the adapter/listener class code would change:

```
class SymAction implements java.awt.event.ActionListener
{
    public void actionPerformed(java.awt.event.ActionEvent
     event)
    {
    Object object = event.getSource();
    if (object == NextButton)
        NextButton_Action(event);
    else if (object == VacationSlides)
        VacationSlides_SlideChanged(event);
    }
}
```

To delete the event handler in the Class Browser:

1. Make the project active by moving focus to the project's Project window.
2. From the Window menu, choose Class Browser.

   A list of project classes appears in the Classes pane.
3. In the Classes pane, click the container class that contains the trigger component.

   In the Members pane is a list of methods associated with the container class.

   Note: If the class you want is not displayed in the pane, you might need to change what classes are displayed by choosing Classes, then Options.

   ---
4. Select the appropriate event handler for the interaction. An event handler is named using a combination of the component's name and the trigger event; for example, `NextButton_Action`.

   Note: If you rename a component after an event handler is created, the method is not renamed.

   ---
5. Delete the interaction in the event handler code. Or delete the entire event handler if it only handles one interaction.
   - To delete the event handler, right-click the event handler and choose Delete Member.

To delete the event handler in the Source window:

1. Open the source file containing the event handler in the Source window.
2. In the Objects drop-down list of the Source window, choose an object.
3. In the Event/Methods drop-down list, choose the event.

   Existing events and methods are shown in bold.
4. Delete the interaction in the event handler code. Or delete the entire event handler if it only handles one interaction.

To delete the applicable portions of the listener registration and the call in the adapter/listener class:

1. Open the Source window.
2. Delete the applicable portions of code.

Consult a Java book for more information. In general, these are the edits you need to make:

- In the adapter/listener class, if there is only one call to an event handler, delete the adapter/listener class. If there is more than one call to an event handler, delete the check for the object requesting the handling of the event and the call to the event handler. This is an if or else if statement; make sure the code you are left with is syntactically correct.
- In the listener registration (marked by the REGISTER_LISTENERS comment), delete the object.addtypeListener or object.addtypeAdapter call for that component, if the listener registration is not used by another event tied to that component.

Using the Classes, and Hierarchy Editor menus

This reference section provides detailed descriptions of the commands in Visual Cafe's Window menu when one or more of these tools is in use. The Window menu contains commands to help manage projects that you open in Visual Cafe's Class Browser and Hierarchy Editor windows.

Using the Classes menu

The following options are available from the Visual Cafe Classes menu. This menu is available when the Class Browser is active. Use the Insert menu or pop-up menus in the Class Browser panes to add new classes and members.

Using the Edit Class command

Use this option to open the Edit Class Wizard, where you can change the definition of a class or interface.

Using the Member Attributes command

Use this option to display the Member Attributes dialog box, where you can change the member's access type for the selected method or data element.

Using the Delete Member command

Use this command to delete the selected member from its class. If the selected member is a variable, like "Button1", this command removes the associated visual object.

Using the Go to Source command

Use this command to open the class's Java file in the Source window.

Using the Options command

Use this command to open the Class Options dialog box, where you can expand or refine the elements in the class listing and define how classes or members are sorted.

Using the Hierarchy Editor right-click menu

When you right mouse click an object in the Project window, Visual Cafe displays a pop-up menu with these commands:

Using the Class Attributes command

Use this command to open the Edit Class Wizard, where you can change the definition of a class or interface.

Using the Remove Inheritance command

This command is enabled when you select the line connecting two classes in the window.

The line represents a connection between two classes. When you delete the connection, the inheritance changes such that the object now extends `java.lang.object`.

Using the Go to Source command

Use this command to open the Source window with focus on the selected package or class.

Using the View Imports command

In addition to the local classes, the Hierarchy Editor displays the classes that are imported into the project.

Using Macros in Visual Cafe

As you use the Source editor to create and edit program code, you may find yourself performing some tasks again and again. You can automate the task by turning on the Macro Recorder and having it create a macro for you.

Record macro Choose this command before beginning a task in an editing window. Your keystrokes and mouse actions are then recorded. After completing the task, choose Stop Recording from the Macro Menu. You can save the macro for later use.

Play After creating a macro, repeat the task you recorded by playing the macro.

Recording and playing the Default Macro

Visual Cafe lets you record a single default macro and then use it repeatedly throughout your project. You might want to use this feature if you only need a single macro to do a repetitive task.

To record the default macro:

1. From the Tools menu, choose Macro, then Record Macro.

   A dialog box appears and asks if you want to write over the existing default macro.
2. Click OK.
3. Perform the necessary actions that you want recorded by the Macro editor.
4. When you are done creating your macro, choose Tools, Macro, then Stop Recording.

   The macro is recorded, but is saved only in memory. See the next session on how to save macros for later use.

To play the default macro:

From the Tools menu, choose Macro, then Play.

Saving the default macro and using it with other macros

After you have recorded the default macro, you can save it for later use by naming it and saving it to your hard drive.

To save the default macro for later use:

1. Record the macro. See the previous section, “Recording and playing the default macro.”
2. Choose Tools, Macro, then ScriptMaker.

   In the ScriptMaker dialog box, the <default> entry must be highlighted.
3. Click the Rename... button.

   The Rename/Duplicate Macro dialog box appears.
4. In the Rename/Duplicate Macro dialog box, enter a name for the macro and a filename.
5. Click the OK button.

Using recorded macros other than the default macro

To run a macro other than the default one:

Choose Tools, then Macro, then ScriptMaker. Select the macro you want to run from the menu.

Using the ScriptMaker dialog box

You can use the ScriptMaker dialog box to copy and rename macros.

Choose Tools, then Record Macro command to record your editing keystrokes. The file is saved as the <DEFAULT> macro. You can then use this dialog box to duplicate the macro and name the duplicate for saving.

You cannot rename the default script; you must duplicate it first. If DEFAULT.MAC is deleted from disk, Visual Cafe Basic creates a stub file when it starts up.

Macros

Lists existing macros that are associated with the current project. ScriptMaker macro files have the extension .MAC. The default macro is highlighted by default. Click a macro to select it. Double-click to open a macro in an editor.

Properties

The display name and file name of the selected macro.

Display in Menu

Select this option if you want the current (highlighted) macro to appear by name at the bottom of the Tools Macro menu.

Reorder Commands

Allows you to change the order of the macros which appear in the Macro menu. Click the up arrow to move the selected macro up in the menu; click the down arrow to move it down in the menu. The default macro always stays at the top of the list.

Done Button

Click Done to save all changes to the project's macros.

Edit Button

Click Edit to open a macro in an editor.

Rename Button

Click Rename to rename the current macro. The Rename/Duplicate Macro dialog box appears. Rename is not active when the default macro is highlighted.

Duplicate Button

Click Duplicate to create a copy of the current macro. The Rename/Duplicate Macro dialog box appears.

Delete Button

Click Delete to delete the current macro. Delete is not active when the default macro is highlighted.

CHAPTER 5

Including Visual Components

Visual Cafe makes it much easier to design your graphical user interface (GUI) by allowing you to visually lay out your applets and windows. To design your GUI, you must first create a Visual Cafe project. Then you can:

- Add components, such as windows, dialog boxes, text, buttons, and graphics, to the project. You can add components to a project from the Component Library or Palette.
- Open top-level components, or forms, in the Form Designer, then visually arrange components on the form. You can drop components into the Project window or directly into the Form Designer, and then arrange them in the Form Designer.
- Set component properties, such as color and text, in the Property List. These changes are immediately reflected in the Form Designer.
- Create interactions between components. For example, if a user clicks a Clear button, a text field clears.

What are Visual Cafe forms?

Forms are the basis for creating a user interface to your applets and applications. Forms can be windows that display information and can receive user input. A form can contain labels that display text, or can contain components that provide interaction with the program. Some examples of components are:

- Check boxes
- Radio buttons

- Text boxes
- Scroll bars

You can also use forms as containers for items that are not visible components in a user interface. For example, you can have a form in an application that serves as a container for other components that will be used in other programs. Or, you can have a container containing code that handles events like mouse interactions.

Understanding the container class

Some components can contain other components, such as text, buttons, graphics, charts, and so on. These components are called containers.

In general, while components are independent objects, there is a certain parent and child relationship that exists between containers and other components.

Java defines containers as components that hold any number of components, including other containers. Containers hold and organize your components, but they also contain code for event handling and many essential things such as controlling the cursor image and the program's icon. All containers allow for operations such as adding, removing, and painting their contents, or components. Containers allow for the grouping of related components together and treating them as a single unit, simplifying the amount of programming you have to do.

To better understand the idea of containers, think of all visual components as children of the form on which they are displayed. Most components inherit the read-only Parent property, which displays the form. The placement of visual components is also relative to the Parent form. Visual components cannot be moved outside the boundaries of the parent. Moving a form moves the components as well.

The top-level containers, or forms, are Applet, Frame, Window, and some dialogs. When displayed in the Form Designer, the size of the window is the bounds of the form.

Other containers are panels, MenuBar, Menu, and some dialogs. These containers can be contained by a form, but are not forms themselves.

You can use a container in two ways:

- You can subclass Container if you override all abstract members, but the Window container typically provides this basic functionality.
- All Containers inherit Container data members and methods. Any Container you subclass can call any member it needs, provided you override that member in your subclass.

In Visual Cafe, applications are built on Frame containers and Applets are built on Applet containers. The Form Designer window becomes an Applet or a Frame as selected on the Project window. Other specialized containers are listed in the following table.

Container	Function
Window	A blank modal container that must be set into a Frame.
Frame	Extends Window. It uses the BorderLayout manager, supports a title and menu bars, can be minimized, and must be a parent container.
Panel	Uses a FlowLayout manager, and can be placed inside other containers including other panels.
Applet	A Panel that can be run by another program and supports multimedia features. Applet containers are parents to applet programs. Applets cannot be nested.

You can build a GUI using Visual Cafe, create an application or an applet project using the Project window. Depending on what you selected, Visual Cafe automatically provides you with either a Frame container or an Applet container as the parent container for your project. Continue by dragging container and component icons from the Palette to the Form Designer window as necessary.

You can also build a GUI in project source code, by instantiating a Container object and all the Component objects you need. Call the container's getLayout method to add a layout manager. Then call that Container object's Add method to place Component objects into your Container object. Finally, call the container Show method to make the container and all components visible.

Working with basic user interface components

The Java Abstract Window ToolKit, or AWT, is a portable GUI library to allow you to create visual front ends. This portable GUI library is cross-platform for developing and running applications and applets.

The AWT supplies many classes for you to develop your Java programs. It is the crucial link between your Java program and the native GUI of the operating system. The AWT performs a very high level of abstraction because there has to be a common denominator in order for your Java program to be portable across platforms. It is a Java package that can be used in any Java program by importing java.awt.* using the import keyword. Visual Cafe does this automatically for you.

In the structure of the AWT, components are added to and then arranged by layout managers in containers. There are a variety of event handling, menu, fonts, and graphics classes in addition to components and containers. The AWT also works well in conjunction with the networking and threads classes.

Creating component layouts

The Java language provides layouts (or layout managers) which help you arrange components inside a container, like a form or a panel. Layouts provide a method of automatically arranging components within a container so that they display nicely on different platforms, screens, and at different resolutions. The following table provides an overview of the layouts.

Layout	Description
BorderLayout	Arranges the components in a center, north, south, east, and west orientation.
CardLayout	Arranges the components on several cards. Only one card is visible at a time. This allows you to flip through the cards.
FlowLayout	Arranges buttons from left to right until there is no more room. This layout has the option of align center, align left, and align right.
GridLayout	Arranges components in definable rows and columns.
GridBagLayout	Arranges objects by size and space.

The way AWT components actually appear on the screen is determined by the order in which components are added to the panel that contains them, and the layout manager that panel is using to display them on the screen. The layout manager determines and allocates which components within that panel will be displayed.

Visual Cafe supports all Java layouts, as well as creating your own. You can add a layout to a form (or panel) by setting the form's layout property. Immediately when you change the property, the components inside the layout are arranged based on their creation order and the specified layout. You can rearrange components in the layout by dragging and dropping to the desired location.

Working in the Form Designer

The Form Designer is a separate Visual Cafe window that displays a form. The window size is the size of your form. Components contained by the form appear on the form. For example, if you run an applet in a Web browser, the applet will appear the same size as it does in the Form Designer window. The components contained by the form appear as shown in the Form Designer. While working with Visual Cafe, you can have multiple Form Designer windows open at once—one per form.

The Form Designer uses an integrated Java Virtual Machine to run Java code at design time. This allows the Form Designer to provide a true what-you-see-is-what-you-get (WYSIWYG) design environment, including the accurate representation of complex layouts using the various Java layout managers. It also allows the Form Designer to run Java components and

applets at design time so that the effects of an on-screen animation can be accounted for in the layout design.

Some components can only appear at the top level in the Objects view of the Project window. These are the Visual Cafe forms, including the Applet, Frame, Window, and some dialog components. If you look in the Component Library, the forms provided with Visual Cafe are all in one Forms group. A form can contain other components, such as text, buttons, and graphics.

Displaying graphics in the Form Designer

When you have applets that have an image (`.gif` file) drawn using paint(), it appears in browsers and in the AppletViewer. However, it may not appear in the Form Designer.

To have your images display at design time in the Form Designer, use the ImageViewer component.

Creating Java code

A top-level component corresponds to a Java source file. Visual Cafe automatically creates the Java code and updates the code as you visually design your GUI.

The top-level component for an applet is the Applet component. The init method, which is called by another program (such as a Web browser), is

the entry point of the applet. If you use the Basic Applet project template provided with Visual Cafe, the applet is already set up for you programmatically.

The top-level component for an application with a GUI is the Frame component. The main method, usually called from the command line, is the entry point of the application. If you use the Basic Application template provided with Visual Cafe, the main window is already set up for you programmatically.

Other top-level components are the Window component and some dialog components. When you create a Java program, you can display these top-level components from an Applet or Frame, for example.

Component is the parent class from which all visual components and containers extend. This class provides a protocol defining objects that will possess size and position, can be rendered onto the screen, and can respond to events.

Component subclasses the set of visual components present on the Visual Cafe Palette. It also subclasses the container class, which in turn subclasses specialized containers supporting applets, pop up windows and applications. Component is an abstract class and can not be instantiated.

With Visual Cafe, you can use a component in two ways:

- You can subclass Component directly, provided you override all abstract members.
- All containers inherit Component data members and methods. Any visual component you subclass can call any member it needs.

By default, Visual Cafe provides an Applet component for creating applets and a Frame component for creating applications.

Designing a GUI with Visual Cafe

With Visual Cafe, you can add graphical user interface (GUI) elements to your applets or applications and define how those elements should interact with your applet or application.

Every time you drag and drop a visual component onto the form in the Form Designer, or edit properties for a component using the Property List, Visual Cafe updates the source file for you.

In general, to design your graphical user interface (GUI), you follow these basic steps:

1. Add forms to the project.
2. Add components to your forms
3. Arrange components.
4. Modify component properties.
5. Create component interactions.

The next sections cover each step in more detail.

Adding forms to the project

When you created the project, your project template might have already added forms to your project. Some components can only appear at the top level in the Objects view of the Project window. These are the Visual Cafe forms, including the Applet, Frame, Window, and some dialog components. If you look in the Component Library, the forms provided with Visual Cafe are all in one Forms group. A form can contain other components, such as text, buttons, and graphics.

The form for an applet is the Applet component. The form for an application with a GUI is the Frame component.

Other forms are the Window component and some dialog components. When you create a Java program, you can display these forms from an Applet or Frame, for example.

You can add a form to a project by using the Insert menu, or by using drag-and-drop.

To add a form using the Insert menu:

1. While the project is selected, choose Form from the Insert menu.
2. Select a form template.
3. Click OK.

   The new form is added to the project and the Form Designer opens.

To add a form using drag-and-drop:

Drag a form from any of the following areas into the Project window:

- the Component Palette
- the Component Library
- the same Project window (press CONTROL and drag to copy the form)
- a different Project window

Note: If the parse fails, you can see a file in the Packages and Files view, but not see an object in the Objects view of the Project window. You probably need to correct the Java code to get the file to parse. See "Adding code to a Java source file" on page 4-23, for more information.

Adding components to a form

After you add a form to a project, you can add components to it in the Form Designer.

To add a component to a form, select the component from the Component Palette and drag it into the Form Designer. The Component Palette contains a variety of components that you can add to forms including standard Java window components. Visual Cafe also includes an extensive list of custom and third-party components.

Components provide the user interface to your applet or application.

Note: Double-click on a form component to open the Property List for the component or an appropriate editor.

The following table describes the relationship between form types and their valid components.

Form type	Valid components
Frame	Menus and all components
Applet	All components, except menus and forms

To add a component to a form:

1. Add the component by using one of these methods:
   - use the Insert menu
   - drag the component from the Palette to the Project window
   - drag the component from the Palette to the Form Designer
   - drag the component from the Component Library to the Form Designer
2. Size the component.
3. While the component is selected, type in a name for the component. The name should not contain spaces.
4. Change component properties as needed.
5. From the File menu, choose Save to save changes.

Note: Renaming a component must be done using the Name property in the Property List.

Copying components

You can copy form components and menu bars to another form, within the same form, or to another project by:

- dragging and dropping the component to the target form
- copying and pasting the component with menu commands and the toolbar
- selecting the component and using the Copy and Paste pop-up menu commands

When you copy a component, the following points apply:

- The object's bound events are not moved to the new location.
- You must manually move any code to the new form.

- Forms and components must have unique names. If necessary, an object is renamed when pasted.
- If you copy and paste a top-level container, the corresponding Java file is duplicated and placed in the target project directory.
- If you copy and paste a component in a container, the Java code of that component is placed in the Java file of the top-level container. Only component code that is automatically generated by Visual Cafe is placed in the Java file. This does not include custom code and interactions.
- Visual Cafe only allows appropriate copies; for example, a component that is not a container cannot be copied to the top level. While dragging, a circle with a line through it means the operation is not allowed.

You can copy components between the Project window (Objects tab), Form Designer, and Menu Designer as needed.

To copy and paste a component:

1. Open the project(s) you want to use.
2. Click the Objects tab in the Project window, or open the Form Designer or Menu Designer, then select the component that you want to copy.
3. From the Edit menu, choose Copy, or right-click and choose Copy, or click the toolbar Copy button.
4. If you want to paste the component within another container, select that container.
5. While the target Project window (Object tab), Form Designer, or Menu Designer is active, from the Edit menu, choose Paste; or right-click and choose Paste, or click the toolbar Paste button.

   The component appears in the Project window. If needed, the component is renamed to prevent name conflicts.

To copy and drag a component:

Do either of the following:

- Within the same project, press CONTROL and drag the component to the location you want it.
- To a different project, drag the component to the location you want it.

Moving components between forms

You can move a component from one form to another.

To move a component between forms:

Do either of the following:

- In the Project window, drag the component on top of the new form.
- In the Form Designer, cut and paste the component.

When you move a component, the following points apply:

- Moving a component to another form does not move the associated code. You need to delete any code that references the component and add it to the new form's source code.
- In the Form Designer, dragging a component from one form window to another moves the component.
- The object's bound events are not moved to the new location. You must manually move the code to the new class.
- Forms cannot be moved into another form.

Deleting components from a form

You can delete a component from a form at any time.

To delete a component:

Do either of the following:

- Select the component and press DELETE.
- From the Project window, expand the object listing of the form that you want to edit, select the component, and press DELETE.

Changing components

If you place components in your project, and you would like to change them to another type of component, Visual Cafe lets you to do this easily.

For example, you may have placed one type of button on your form, but now you want to change the kind of button. Instead of hacking the source code, one way to do it is by deleting the components all together, and then dragging new ones on to your form.

To change one component to another type of component:

In the Property List, select the component's name from the Property List's drop-down list.

To select multiple components, you must select them in the Form Designer by CONTROL-clicking the desired components, SHIFT-clicking, or dragging to select.

If the Form Designer is open, any changes that you make are dynamically reflected.

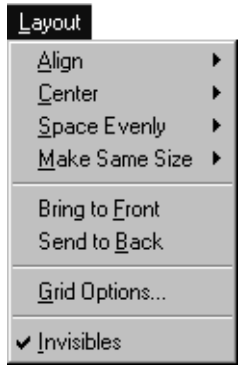
Displaying invisible components

Some components are visible only at design time; that is, they are not visible to the user at runtime. A menu bar is an example of a component that is invisible at runtime.

Invisible components allow access to canned functionality such as File Open dialog boxes. They are platform specific and share a set of basic functionality. You can display the dialog box and use its existing functionality.

To display or hide invisible components:

From the Layout menu, choose Invisibles.



Tip: To make form development easier, turn display of invisible components off.

Notes:

Invisible components never display at runtime.

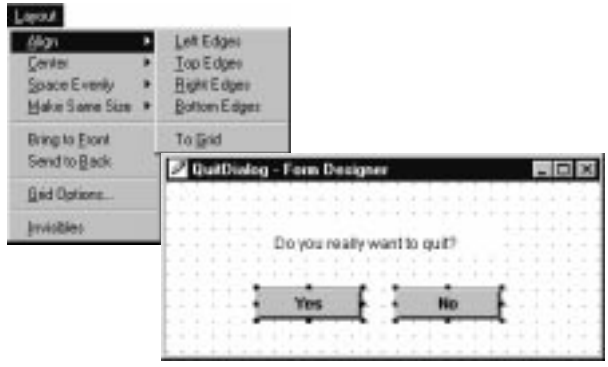
When you add an invisible component to a form, all invisible components display automatically.

Tabbing between fields on a form

You can allow users to tab between fields on a form by placing the fields in a KeyPressManagerPanel container. The tab order of the components contained by the panel is listed in the Project window. To change the tab order, change the order of components in the Project window list.

Arranging components on your forms

Visual Cafe provides layouts (also called layout managers) to help you organize components on a form or in a panel. You can group components in panels on your form and use different layout managers to suit the components contained by the panel.



To change the layout of a form:

1. Display the Form Designer by double-clicking the form in the Objects view of the Project window. Or, you can right-click the form, and choose Edit form.

   Now you need to use the default layout tools or choose a layout manager for the form.

2. Open the Property List for a form or panel.

3 Select a layout for the Layout property.

The Form Designer immediately reflects the layout by rearranging the components based on the order in which they appear in the Project window. Or, you might have to position components first.

4 Rearrange the components in the Form Designer, if needed.

Notes:

You can also arrange objects with the Layout menu's Align, Center, Size, and Space menu commands and the Form Designer grid. (These commands are only available when you do not specify a layout manager for the form.)

You can move components by pixel spacing with the right, left, up, and down arrow keys.

Using the layout manager of None

When you choose a layout of None, components are placed by exact pixel positions on a form or panel. In the Form Designer, you can do the following:

- Drag components on the Form Designer to position them.
- Explicitly set x and y coordinates in the Property List.
- Arrange objects with the Align, Center, Space Evenly, Make Same Size, Bring to Front, and Send to Back menu items of the Layout menu.
- Use the Form Designer grid, which you can set by choosing Layout Grid Options. See Manipulating the Form Designer grid for more information.
- Move a component pixel-by-pixel by selecting the component on the Form Designer and pressing the right, left, up, and down arrow keys as needed.

You should test your layout by running it on different operating systems and screens, as applicable. For maximum portability, it is sometimes best to not overlap components in your layout.

Arranging components in BorderLayout

Use BorderLayout to arrange components in a center, north, south, east, and west orientation. It positions components based on their preferred sizes and the constraints of the container size.

To arrange components in BorderLayout:

1. Choose BorderLayout for the Layout property of a form or panel.
2. If components are already on the form or panel, rearrange components, as needed. Set the Position property for each component you want to position.

   Note: In the BorderLayout, no components should have the same Placement value.

3. In the Property List, choose the form or panel component, then set the Horizontal Gap and Vertical Gap properties to adjust the layout.
4. For each component you want to add, add the new component, then set its Placement property to place it on the form or panel.

   When you first add a component, its position property is blank (which is the same as CENTER).
5. Test your layout by running it at different form sizes and resolutions. For example, you can resize the form when you run it from Visual Cafe.

Arranging components in CardLayout

Use CardLayout to arrange components on several cards. Only one card is visible at a time, which allows you to flip through the cards.

Note: At runtime, you have to programmatically implement flipping through the cards.

A component will be sized to take up an entire card; if you want multiple components on a card, place components on panels. That way, the panel will take up the entire card.

To arrange components in CardLayout:

1. Choose CardLayout for the Layout property of a form or panel.

If components are already on the form or panel, each component directly subordinate to the form or panel becomes a separate card. The cards are placed in the order they appear in the Project window.

2. In the Property List, choose the form or panel component, then set the Horizontal Gap and Vertical Gap properties to adjust the layout.
3. If components are already on the form or panel, rearrange components in the Form Designer or Project window, as needed:
   - To change the card order, change the component order in the Project window.
   - To flip between cards in the Form Designer, right-click, then choose Previous Card or Next Card.
4. Add components as needed and rearrange them.
5. Test your layout by running it at different form sizes and resolutions. For example, you can resize the form when you run it from Visual Cafe.

Arranging components in FlowLayout

Use FlowLayout to arrange components in rows from left to right. You can specify center, left, or right alignment, as well as the horizontal and center gaps between components.

To arrange components in FlowLayout:

1. Choose FlowLayout for the Layout property of a form or panel.

   The Form Designer immediately reflects the layout by rearranging the components based on the order in which they appear in the Project window.
2. In the Property List, choose the form or panel component, then set the Alignment, Horizontal Gap, and Vertical Gap properties to adjust the layout.
3. If components are already on the form or panel, rearrange components in the Form Designer or Project window, as needed.
4. Add components as needed and rearrange them.
5. Test your layout by running it at different form sizes and resolutions. For example, you can resize the form when you run it from Visual Cafe.

Arranging components in GridLayout

Use GridLayout to arrange components in components in definable rows and columns. This layout is similar to FlowLayout except each component is in an area of equal size. You can specify the number of rows and columns, as well as the horizontal and center gaps between components.

To arrange components in GridLayout:

1. Choose GridLayout for the Layout property of a form or panel.

   The Form Designer immediately reflects the layout by rearranging the components based on the order in which they appear in the Project window.

2. In the Property List, choose the form or panel component, then set the Rows, Columns, Horizontal Gap, and Vertical Gap properties to adjust the layout.

   If you set either Rows or Columns to zero, GridLayout computes the other value for you. If both Rows and Columns are nonzero, the Rows value is used.

3. If components are already on the form or panel, rearrange components in the Form Designer or Project window, as needed.
4. Add components as needed and rearrange them.
5. Test your layout by running it at different form sizes and resolutions. For example, you can resize the form when you run it from Visual Cafe.

Arranging components in GridBagLayout

Use GridBagLayout to arrange components by size and space. Like GridLayout, GridBagLayout treats the form or panel as a grid of cells. Unlike GridLayout, however, a component can occupy more than one cell.

To arrange components in GridBagLayout:

1. Choose GridBagLayout for the Layout property of a form or panel.

   The Form Designer immediately reflects the layout by rearranging the components based on the order in which they appear in the Project window.

2. Rearrange and add components as needed. In the Property List, choose a component, then set the Grid Bag Constraints properties to adjust its place in the layout. You can think of these properties as “suggestions” that GridBagLayout uses.

3 Press F1 on a Grid Bag Constraints property to get a description of it. Insets specify how much space to leave between the borders of a component and its display area.

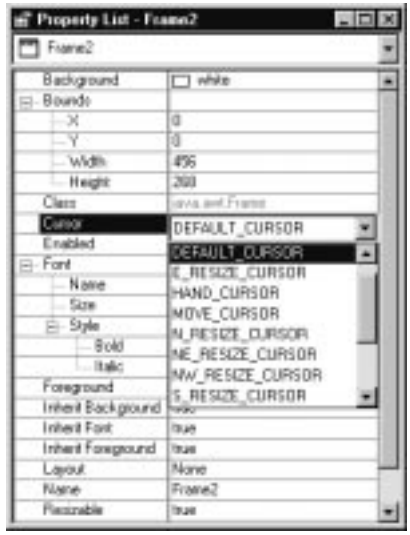
4 Test your layout by running it at different form sizes and resolutions. For example, you can resize the form when you run it from Visual Cafe.

Modifying component properties

Each component has a set of properties that define its look and behavior. Visual Cafe provides a Property List from which you can directly modify these properties. When you change a property, the source code and Form Designer are immediately updated. If you change a property in the Component Library, you see the changes the next time you use the object.

The Property List allows you quick access to all of the components within the scope of the current form or applet. It shows the name of the selected item and lists the component's properties. As you add components to a form, or set properties for the components on the form, the Form Designer shows exactly what the form will look like when you run it.

To view or change the properties of components in a project, select one or more components in the Form Designer or Project window.



If you select multiple components at the same time, only the subset of the properties common to all selected components are displayed.

To view or change the properties of components in the Component Library and Palette, select one or more components in the Component Library. Remember that if you change the properties of a component in the Component Library, the change now appears every time you add that component to a project. Any projects that already contain the component before the change are not affected.

To modify a component's properties:

1. From the Window menu, choose Property List to display the Property List.
2. Select a component in the Form Designer, Project window, Component Library, or from the Property List pull-down menu. To select multiple components, CONTROL-click in the Form Designer, Project window, or Component Library, or Shift-click or drag in the Form Designer.

   When multiple components are selected, only their common properties are shown and editable. In the Property List, you see the heading title "Multiple Selection."
3. To edit a property, click the right column, double-click the left column, or use TAB in the Property List.

   The right column displays a list of valid values or makes the text string editable.

   Properties with multiple values are marked with a plus sign (+); for example, the Font property. Click the + to expand the list.

   Properties that are defined using a dialog box are marked with the ellipsis button (…). Click the … button to display the dialog box.
4. Press ENTER or click somewhere else to make the change.

   The change is applied to all selected components.

Note: Press ESCAPE to cancel an edit and return the property to its previous value.

Creating component interactions

One of the most powerful features of Visual Cafe is the ability to quickly build a relationship, or interaction, between two components. Interactions are implemented as methods and imply an event notification.

An interaction is created by connecting two components on a form. For example, you may want to connect a dialog box (the action component) to

a button (the trigger component). You can connect components from within the Form Designer or the Project window. (You can also create an interaction relationship between a form and a component by using similar steps.)

As you create interactions, Visual Cafe automatically generates code for the relationship. This means that you can assemble interactive applets and applications without writing code.

For more information about working with interactions, see “Working with the Interaction wizard” on page 4-41.

Creating menus with the Menu Designer

You can add menu bars to frames and dialog boxes (they inherit from MenuContainer); applets do not support menu bars. To create a menu bar, you can do the following:

- Add a MenuBar component to a frame or dialog box.
- Add one or more Menu components to the MenuBar container.
- Add one or more MenuItem components to a Menu container. You can also add CheckboxMenuItem components.
- Add one or more submenus to a menu item.
- Add command key equivalents to menu items.

- Add interactions between menu items and other components in your project. For example, specifying that a menu item open a dialog within the same project.
- Bind custom code to menu items.

The Frame component in the Basic Application template already has a menu bar, which you can modify and enhance.

The Menu Designer makes it easier to create a menu bar by letting you edit a visual representation of the menu bar. You can move items in the Menu Designer to visually change the menu structure.

To open the Menu Designer:

1. Double-click a MenuBar component in the Project window or Form Designer.
2. In the Project window, select the MenuBar component, then choose Edit MenuBar from the Object menu, or right-click and choose Edit MenuBar.

Using the Menu Designer pop-up menu

When you right mouse click in the Menu Designer, Visual Cafe displays a pop-up menu with these commands:

Cut/Copy/Paste – These edit commands perform standard Windows functions.

Insert Menu – This command inserts a menu in the menu bar; you can then add menu items to the menu.

Insert Menu Item – This command inserts a menu item into the selected menu that is in a menu bar.

Create Submenu – This command lets you create a submenu for the selected menu item.

Edit Source – This command opens the source file for the menu bar so you can bind code to menu items.

Add Interaction – This command opens the Interaction Wizard so you can define an interactive behavior between the selected menu item and another component in the same project.

Properties – This command displays the list of properties for the selected menu component.

Adding a menu to a form

Menus are added to dialogs and frames in the Form Designer, and then edited in the Menu Editor. Applets do not accept menu bars.

You can also add a menu by choosing Object from the Insert menu.

To add a menu to a form:

1. Click the MenuBar component in the Component Palette and drag it into the open form.
2. Double-click on the menu bar object to open the Menu Editor.

Copying a menu

Copying menus allows you to reuse menus that are common among forms and projects.

To copy a menu:

1. To open the Menu Editor, double-click the menu bar component on a form or in the Project window.
2. In the Menu Editor, click on the menu that you want to copy.
3. From the Edit menu, choose Copy.
4. Open the target form and double-click the target menu bar component.

   The component appears in the Menu Editor.
5. Select a location for the new menu. You can paste a menu to the top-level or within a menu to create a hierarchical menu.
6. From the Edit menu, choose Paste.

Adding a menu bar to a frame or dialog box

You can add the MenuBar component to a frame or dialog box in a variety of ways.

To add a menu bar to a frame or dialog box:

Do one of the following:

- Drag a MenuBar component from the Component Library or Palette into the Project window or Form Designer.
- While the Project window or Form Designer is active, choose Component from the Insert menu.
- Right-click the frame or dialog box in the Project window and choose Insert Component.
- Copy and paste it within the same project or between projects.

To open the Menu Designer:

1. Double-click a MenuBar component in the Project window or Form Designer.
2. In the Project window, select the MenuBar component, then choose Edit Menu from the Object menu, or right-click and choose Edit Menu.

Adding menus to a menu bar

You can add a Menu component to a MenuBar component.

To add a menu to a menu bar:

1. Do one of the following:
   - Drag a Menu component from the Component Library or Palette into its position in the Project window or Menu Designer.
   - While the menu bar is selected in the Menu Designer, right-click and choose Insert Menu.
   - Right-click the menu bar in the Project window and choose Insert Component.
   - While the menu bar is selected in the Project window or Menu Designer, choose Component from the Insert menu.
   - Copy and paste it within the same project or between projects.
2. In the Property List, set the menu properties, including the menu name, in the Label property.

The Help Menu property lets you integrate the menu item into an existing Help menu that is part of the operating system, for example.

Adding menu items to menus

You can add a MenuItem or CheckboxMenuItem component to a Menu component.

To add a menu item to a menu:

1. Do one of the following:
   - Drag a component from the Component Library or Palette into its position in the Project window or Menu Designer.
   - While a menu item is selected in the Menu Designer, right-click and choose Insert Menu Item.
   - Right-click the menu in the Project window and choose Insert Component.
   - While a menu is selected in the Project window or Menu Designer, choose Component from the Insert menu.
   - Copy and paste it within the same project or between projects.
2. In the Property List, set the menu item properties, including the menu item name in the Label property.

Tip: To add more menu items while in the Menu Designer, you can now select the bottom menu item in the list and press ENTER.

Adding submenus to menu items

You can add a MenuItem or CheckboxMenuItem component to a MenuItem component.

To add a submenu to a menu item:

1. In the Menu Designer, right-click a menu item and choose Create Submenu.

   A submenu appears.
2. While the submenu is selected, in the Property List set the submenu properties, including the name of the menu in the Label field

Tip: To add more submenu items, you can now select the bottom submenu item in the list and press ENTER.

To add a CheckboxMenuItem component as a submenu:

1. Drag it from the Component Library or Palette to its position in the Project window or Menu Designer.
2. In the Project window or Menu Designer, select the parent menu item or a submenu, then choose Component from the Insert menu.

Editing a menu structure

You can change the menu structure as needed.

To move items in the menu structure:

Move items in the Menu Designer or Project window to visually change it.

To delete a menu or menu item:

Select the component in the Menu Designer or Project window, then press DELETE or choose Edit Delete.

Note: If you delete a component from a container, you must manually delete any custom code or interactions involving that component.

Editing menu bars and menus

Menus are edited using the Menu Editor and Property List. Each menu is a subcomponent of the menu bar.

To edit a menu or menu bar:

1. Add a menu bar to the form, or if the menu bar already exists, do either of the following:
   - In the Form Designer, double-click on the menu object.
   - In the Project window, double-click on the menu bar icon.

   Note the menu placeholder in the menu bar window.
2. Open the Property List by choosing Property List from the Window menu.

3 Do either of the following:
   - Select the Label property and enter the menu caption.
   - Highlight the menu placeholder and start typing. The text is added to the Label property.
4 To add more menu items:
   - press ENTER to move down to the next menu item.
   - choose Insert Menu Item from the pop-up menu to insert a command before the selected command.
5 Right-click on the menu item to display the pop-up menu.
6 Choose Create Submenu.
7 Bind code to the appropriate menu items.
8 Define any interactions by selecting the menu item and choosing Add Interaction from the pop-up menu.

Associating command keys and menu items

You can quickly add command keys to menu items.

To associate a command key with a menu item:

1 Select a menu item in the Project window or Menu Designer.
2 In the Property List, expand the Menu Shortcut property and specify the command key(s) in the Key Code and Use Shift Key fields.

   The Key Code field lets you specify what keys you want to use, for example, VK_P selects CTRL+P. In the Menu Designer and Project window, this key sequence displays as “CTRL+Kanji.”

   If you want the SHIFT key to be part of the command key sequence, choose true; otherwise, choose false.
3 Verify your command keys by running your Java program:
   - From the Project menu, choose Execute to run the project with no debug processing.
   - From the Project menu, choose Run in Debugger to run the project and have access to all debugging functionality.

Note: You can compile and run a project any time during its development cycle. Visual Cafe automatically saves files in the project before running.

Binding code to a menu item

Code can be bound to a menu item just as it can be bound to a component. Menu items respond to one event: ActionEvent. This event occurs when the user selects the menu command.

To bind code to a menu item:

1. Open the menu bar in the Menu Designer or Project window.
2. Select the menu item.
3. Choose Edit Source from the pop-up menu.
4. In the Source window, select the Action event from the Event/Method drop-down list.
5. Add the appropriate Java code to the event handler.

Working with the Component Palette

The Component Palette contains a variety of components that you can add to forms. The Palette can contain visual objects, program modules, and form templates.

You can control the Palette's position and visibility by docking, floating, resizing, and hiding.

The Interaction Tool allows you to create an interaction between two components by visually connecting their behavior.

The Selection Tool is enabled by default. Click on this icon to return to object select mode. A component is selected only if it is completely surrounded by the selection rectangle.

You can also customize the Palette to contain the objects that you use most often.

The following sections describe common components in the Component Palette that you can add to your forms.

Using the InvisibleHTMLLink component

An InvisibleHTMLLink component lets you set up a jump to a URL. Because it is invisible, InvisibleHTMLLink is ideal for creating a clickable area over an image that lets you jump to a new location in an HTML file.

To use the InvisibleHTMLLink component:

1. In the Component Palette, click Additional, then click InvisibleHTMLLink and click and drag to draw a rectangle on the Form Designer.

   Alternatively, you can drag InvisibleHTMLLink from the Component Library and resize and reposition it.

   A new InvisibleHTMLLink appears in the Form Designer.
2. In the Property List, double-click the HTML Link URL property.
3. In the HTML Link URL dialog box, type the URL.

Notes:

Anchors (#) are supported if you want to jump to a specific location within an HTML file.

Different browsers use a different z-order when determining how components overlap in an applet. The InvisibleHTMLLink must be on top for users to be able to click it. To ensure compatibility with different browsers, it is a good idea to "sandwich" InvisibleHTMLLinks on top and beneath components they overlap. Use Layout Sent to Back or Send to Front.

Different browsers handle anchors (#) differently. For example, while Netscape Navigator requires one # symbol, Internet Explorer requires two (##). If you are using relative URLs, the Visual Cafe component will handle both cases for you; it is sufficient to type one #. However, if you are using absolute URLs, the Visual Cafe component cannot handle the difference for you.

Using the MultiList component

A MultiList component lets you create a table with rows and columns.

To use the MultiList component:

1. In the Component Palette, click Additional, then click MultiList and drag it to the Form Designer.

   Alternatively, you can drag MultiList from the Component Library.

   A new MultiList appears in the Form Designer.
2. Resize the new MultiList component as needed.
3. In the Property List, click the Column Headings property, then type your column names. Press CTRL+ENTER between labels, then ENTER after the last label.
4. Click the List Items property, then type the list items according to these guidelines:
   - Press CTRL+ENTER between rows, then ENTER after the last row. (Each row should be on its own line.)
   - Type a semicolon (;) to indicate a new column in a row, for example, john; cheryl; tim would specify text for three columns in one row.
   - Add rows in order from top to bottom.
5. Optionally set the Heading and Cell properties to customize the look of your table.

Using the ScrollingPanel container

A ScrollingPanel container has scroll bars and can contain one panel, which can contain multiple components.

To set up a ScrollingPanel container:

1. In the Component Palette, click Panels, then click ScrollingPanel and drag it to the Form Designer.

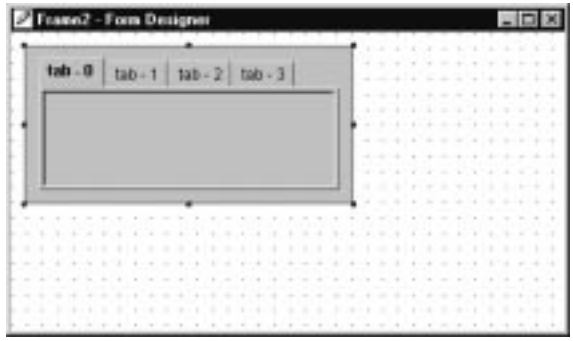
   Alternatively, you can drag ScrollingPanel from the Component Library.

   A new ScrollingPanel appears in the Form Designer.
2. Resize the new ScrollingPanel component to be the size you want it to be.
3. Design another panel.
4. Drag the other panel onto the ScrollingPanel.

The panel now appears within the scrolling region. You can test it in the Applet Viewer.

Using the TabPanel container

A TabPanel component can contain multiple panel components. You can access a particular panel by clicking its tab, which can appear on the top or bottom of the TabPanel.



To set up a TabPanel container:

1. In the Component Palette, click Panels, then click TabPanel and drag it to the Form Designer.

   Alternatively, you can drag TabPanel from the Component Library.

   A new TabPanel appears in the Form Designer.
2. Resize the new TabPanel component as needed.
3. If the background color is white, you could set the Background property of TabPanel to a color, such as lightGray, to see the tabs better.
4. Drag panel components onto the TabPanel.

   A tab is automatically added for each panel, from left to right. You can change the tab order by changing the order of the panels listed in the Project window.

   Be sure to drag panels onto the TabPanel, and not a panel it contains. For example, you can drop a panel next to one of the tabs.

   Tip: To make a tab the default active tab, enter its number in the TabPanel Active Tab property. The tabs are numbered starting from 0 at the left.

   ---
5. In the Property List, choose the TabPanel component from the menu, then click the Tab Labels property.

6 Type the tab labels. Press CTRL+ENTER between labels, then ENTER when you are finished entering labels.

The labels are added in order. The top label is the rightmost label, and the bottom label is the leftmost label.

7 If you want the tabs to appear on the bottom rather than the top, set the Tabs On Bottom property to true.

8 To add components to a panel contained by a TabPanel, click a tab in the TabPanel, then drag components onto this panel.

After clicking a tab, you can click the panel associated with that tab to better see its dimensions. All panels contained by TabPanel are the same size. You must resize the TabPanel to change the size of all the panels it contains.

Using the TreeView component

Use the TreeView container to create a hierarchical list of different text elements. The list can be expanded and contracted by users, similar to a directory structure of files.

To use the TreeView component:

1 In the Component Palette, click Utility, then click TreeView and drag it to the Form Designer.

Alternatively, you can drag TreeView from the Component Library.

A new TreeView appears in the Form Designer.

2 Resize the new TreeView component as needed.

3 In the Property List, click the Items property, then type your labels according to these guidelines:

- Press CTRL+ENTER between labels, then ENTER when you are finished entering labels. (Each label should be on its own line.)
- Add labels in order from top to bottom.
- Create a hierarchy by adding spaces to the front of a label (spaces within a label are allowed). For example, no space is the root level, one space is the next level into the hierarchy, two spaces indicates one more level in, and so on. The label will be subordinate to the first label above it that is at a higher level in the hierarchy.

4 Look through the hierarchy in the Form Designer, then make adjustments to the Items property as needed.

Building a custom Palette

Visual Cafe provides a extensive collection of custom and third-party objects that you can quickly add to your forms, including grids and tabbed dialog boxes.

Palette customization includes:

- creating object groups
- adding objects to the Palette
- deleting objects from the Palette
- removing a tab from the Palette

You can add and remove objects on the Palette from the Environment Options dialog box Component Palette tab.

When groups are added to the Palette, they display on the Palette as tabs. Palette tabs can display horizontally only.

The left list box shows all the objects in the Component Library. The right-hand list represents the Palette itself.

Creating a Palette tab

You can add tabs to your custom Palette to help organize components.

To create a Palette tab:

1. Click the Component Library window.
2. From the Insert menu, choose Group.

   A new folder appears in the Component Library window.
3. Edit the group item name and press Return to enter the new group name.
4. Add your components to the new group folder.
5. Drag the group to the Component Palette. A new Tab for your group appears.

Adding components to the Palette

The Palette contains reusable objects. You can add any object from the Component Library to the Palette and easily use the object in your applets and applications.

You can add objects to the Palette with these tools:

- Component Library
- Project window
- Component Palette tab in the Environment Options dialog box

Adding Palette objects from the Component Library

You can drag an object from the Component Library to the Palette to use in your applets and applications.

To add objects to the Palette from the Component Library:

Drag the object from the Component Library and drop it on the tab of the Palette where you want the icon stored. Duplicates are not allowed in the same tab.

Note: Dragging and dropping a group from the Component Library creates a tab on the Palette. The tab contains all components in the group.

Adding Palette objects from the Project window

While working in the Project window, you can quickly add objects to your custom Palette.

To add Palette objects from the Project window:

1. Select the object.
2. Drag and drop the object onto the tab of the Palette where you want the object to be stored.

Adding Palette objects from the Component Palette tab

The Component Palette tab provides an interface to help you quickly customize your Palette.

To customize your Palette from the Component Palette tab:

1. From the Tools menu, choose Environment Options, then click the Component Palette tab,

   or

Right-click and choose Customize Palette from the Component Palette's pop-up menu.

2 Do either of the following:

- Select an object or group from the Component Library pane and drag it onto a Palette Group or into the general Palette area.

  Dragging an object adds it to the current Palette group. Dropping the object onto another object adds the selected object as a sibling of the same group.

- Select a Group in the Palette pane. Select an object in the Component Library pane. Then click the Add button.

The selected object is added as a member of the current group.

Moving Palette components

Palette components can be moved within their group or to another group.

To move Palette components:

1 Hold down the Control key.

2 Click the component you want to move.

Within the Palette toolbar, you can drag and drop components within the same tab to reorganize their display.

Deleting components from the Palette

As you customize your Palette, you may need to remove objects that you no longer need on a frequent basis.

Caution: Deleting a tab deletes the group and all objects within the tab.

Deleting a component from the Palette:

1 Click an object on the Palette.

2 Right-click and select Remove Component.

Deleting a component from the Environment Options dialog box:

1 From the Tools menu, choose Environment Options > Palette tab.

2 Select the object to delete from the Palette Pane.

3 Click Remove or press the DELETE key.

Working with the Component Library

If you have modified a Visual Cafe component and want to place it in the Component Library, all you have to do is drag the component from the Project window, Objects view, into the Component Library. However, if you want to add custom components that are not based on the Visual Cafe components, you need to follow these steps.

Note: You only need to add components to the Component Library if you want to visually select and use the custom component with Visual Cafe views. Custom components can be referenced in the program's Java code without being integrated into Visual Cafe.

To add a custom component to the Component Library:

1. Create the component class Java file (`.class`).
2. Create a description file for the component (`.desc`).
3. Create a small icon for the component (`.ico`).

   Create a 32 by 32 standard Windows icon, but only the 16 by 16 upper left region of the icon will be used and displayed by Visual Cafe.
4. Add the icon and description files to your Visual Cafe environment.

   The description file and component icon must be copied to the Visual Cafe bin\component directory.
5. Make sure the class file is in the location you want to store it.

   This location must be in your class path. Remember that class files are case-sensitive. If you want to add the class file to a package, the class file must be in the package directory.
6. Optionally add the compiled component class to a package in your project.
7. Display the Component Library by choosing Window Component Library.
8. From the Insert menu, choose Component into Library.

   An Open dialog box appears.
9. Select the class file, then click Open.

   An Add to Library dialog box appears.
10. Select a group, then click OK.

    The component is added to the Component Library.

As you are developing your project, you can delete user-created objects for the Component Library. Deleting a component from the Library also removes the component from the Palette.

To delete an object from the Component Library:

1 In the Component Library, select the object to be deleted.

2 From the Edit menu, choose Cut (or press the DELETE key).

Viewing a component's Java source

You can open a component's Java source file to enhance the component behavior with custom Java code.

To view a component's Java source:

1 Do either of the following:

- In the Form Designer, double-click on the component.
- Select the component in the Source window or Form Designer, then choose Object, then Open Source from the pop-up menu.

2 In the Source window, add your custom code to the file.

CHAPTER 6

Compiling, running, and deploying your program

Introduction

Visual Cafe applets and applications are cross-platform Java programs that you design, develop, and build using the drag-and-drop features of Visual Cafe. Both are executed by a Java Virtual Machine, but applets run only within a Web page, while applications run on their own. You can use Visual Cafe to compile a Java program to bytecode, exercise it to verify its behavior, debug it as necessary, and eventually deploy it to users.

Concepts of applets and applications

The program coding used to create applets and applications are fundamentally similar. Both applets and applications are created using the same basic programming concepts. Because the Web browser is responsible for running an applet, program instructions for applets have a different organization than the instructions for applications.

Applets

Java applets, like all Java programs, are made up of source code that is compiled into a class file. A reference to the class file is placed in a Web page. The Web page is downloaded across the Internet using a Java-capable Web browser. As the bytecode contained in the class file is read, the Web browser's Java interpreter converts the bytecode into machine specific instructions, executes the program, and displays it.

Another feature of applets is that when they are executed within a Java-capable browser, the applet adopts the look and feel of the client machine's operating system and native interface controls. This means that a Java applet will look and feel like it was written for the Macintosh when it runs in the Macintosh environment, Windows applets will have the Windows look to them, and UNIX applets will reflect the look of the various flavors of UNIX.

Applets, however, cannot read from or write to a user's local hard drive. To add this capability to your applet, Symantec has created the database solution dbAnywhere.

When you create an applet with Visual Cafe, you create a subclass of the class `Applet` in the `java.applet` package. This `Applet` class enables your applet to work within a Web browser and to utilize the capabilities of the Abstract Window Toolkit. The AWT allows your applet to include user interface (UI) elements, to handle mouse and keyboard events and to display to the screen. Although your applet can utilize as many classes as it needs, the main `Applet` class triggers the execution of the applet. Its signature is as follows:

```
public class myClass extends java.applet.Applet {
    . . .
}
```

Java requires that your `Applet` subclass be declared as `public`. This is only true of the main `Applet` class; all other classes may be declared `public` or `private` as you desire.

Advantages and disadvantages

Java applets have a significant advantage over applications because the Web browser software handles many of the functions that are required to make the applet run.

Standalone applications have overhead that must be written to make the program run properly; a stand-alone program must be able to start and stop the program, perform memory management, and handle display functions. Although this requires additional programming work, applications do not depend on the presence of a Web browser, nor do they have limitations on the kind of operations they can perform.

Because the browser software provides this functionality for applets automatically, applets are an attractive type of Java program to work with.

Applet limitations

Sun designed Java to restrict the kinds of operations applets can perform. Limitations are imposed on applets so that a destructive program from a remote computer cannot steal information or cause damage to your system. To prevent applets from being destructive, Java enforces the following limitations:

- Applets cannot read from or write to the file system of the computer viewing the applet. This prevents damage to files and the spread of viruses.
- Applets cannot run any programs (or parts of programs like DLLs or shared files) on the viewer's computer. This prevents an applet from calling destructive programs that do not have the limitations of the applet.
- Applets can only establish connections between the server computer where the applet is stored and the client's computer. This restriction prevents the applet from connecting the client's computer to another server without the viewer's knowledge.

Java applications do not have these limitations and can be used to build fully functional software programs.

Applications

An application is a Java program that runs with a Java Virtual Machine (Java VM) or a Java compatible browser that is installed on the client system. An application is not displayed on a Web page. Java applications can be built because the Java language includes useful features that are not found standard in other languages. For example, Java comes with existing libraries of program code that make networking and graphics operations easier than ever.

Java applications, unlike applets, can read from and write to the client machine's hard drive. Applications can create their own frames, title bars, and menus.

Because applications are Java programs that run on their own, applications can be as large or as small as you want or need them to be. The only class that is needed to run an application is the `main` method. When you run your compiled Java class with the Java interpreter in Visual Cafe, the `main` method is called first.

The signature for the `main` method always looks like this:

```
public static void main(String args[]) {. . .}
```

The parts of this line of code have the following meanings:

- The `public` keyword means that the method can be "seen" or used by other classes and components. The `main` method of your application must be declared `public` for the application to run.
- The keyword `static` specifies a storage class.
- The keyword `void` tells the `main` method to not return anything.
- The `main` method takes one argument, which is an array of strings. This array is used for command-line arguments outside of Visual Cafe.

  For example, the body of the `main` method contains all the code your application needs to start executing. This includes code for variable initialization or component instantiation

Compiling and running

You can compile and run a project any time during its development cycle. Visual Cafe automatically saves files in the project before running.

- From the Project menu, choose Execute to run the project with no debug processing.
- From the Project menu, choose Run in Debugger to run the project and have access to all debugging functionality.

See "Setting project-level options" on page 3-34 for information about defining general project and run-time options.

Setting compiler options

From the Compiler tab of the Project Options dialog box, you can control what compiler information is sent to the Messages window, which Java compiler to use, JavaDoc options, Input Encodings, and whether Java optimizations are performed. Click the Project tab to access the option set for the Debug or Final release type.

To set compiler options:

1. Activate the Project window of the project you want to work with.
2. From the Project menu, choose Options.
3. In the Project Options dialog box, click the Compiler tab.

4 Select the options you want:

Select...	To specify this...
Use Java Optimizations	Optimize the Java executable for a more compact executable that runs faster. (Default: enabled for Final and disabled for Debug release types)
	Select the Disable function inlining option if needed. Inlining means that Visual Cafe takes a function's code and imbeds it in the calling function instead of calling the function. Inlining increases execution speed but also increases executable size. (Default: not selected)
Generate Debug Information	Create debugging information used by the Visual Cafe debugger. For example, this option lets you see local variables during debugging. (Default: enabled for Debug and disabled for Final release types)
Show compiler warnings	In the Messages window, display detailed messages that you might be interested in if you need more information; for example, if you have import problems these messages trace where your classes are coming from to help you resolve any import and class path problems. (Default: enabled)
Show progress messages	Show compiler progress messages in the Messages window. (Default: disabled)
Show dependencies	Display file dependencies, such as imports, in the Messages window. (Default: disabled)
Show all Java messages	In the Messages window, display detailed messages that you might be interested in if you need more information; for example, if you have import problems these messages trace where your classes are coming from to help you resolve any import and class path problems. (Default: disabled)

Select…	To specify this…
Use the Sun Java compiler	Use the Sun Java compiler, `javac.exe`. When this option is cleared, the Symantec Java compiler, which is faster, is used. (Default: disabled)

5 Click OK.

The change takes effect next time you compile your project.

Correcting your source code

If there are syntax errors in your source code, Visual Cafe flags them in the Messages window after a compile. You can easily navigate to each error directly from the Messages window.

1 From the Window menu, choose Messages to bring the Messages window to the front.

2 Double-click on any error message to go to that error.

The file containing the error opens at the offending line within a Source window. Once the file opens, you can work on your source code.

Deploying Java programs

Java is a programming language that allows you to design, build, and execute applications and applets. Java programs must be coded, compiled,

and debugged similarly to development in any other computer programming language before they can be used.

Deploying your applet

After you complete an applet in Visual Cafe, you are ready to deploy it on a Web site. You need to know how your particular Web site is set up to get your applet up and running. However, here are some guidelines. Because your project can contain more than one applet that appears in related Web pages, these guidelines are for setting up a project that contains one or more applets.

Important: The classes needed by Visual Cafe components are stored in `symbeans.jar`. You do not want to deploy using `symbeans.jar`, because it will affect the performance of your applets. Instead, you want to deploy using just the classes needed by the components in your applets. These classes can be packaged in a JAR file or not.

To deploy your applets in a JAR:

1. Use the Project JAR command to create a JAR file containing your applets and all supporting files, including Symantec class files. See Chapter 7, “Working with JavaBeans” for more information.

   Tip: Your applets should use relative URLs for graphics files so you can easily move your applets to different computers.

   ---

2. Add the variable `ARCHIVE="name.jar"` to the applet tags in your HTML files. You can specify multiple JAR files by delimiting them with a comma (,).

   Tip: Remember that the applet tags specify where your applets are relative to the location of the HTML files. So you must place the JAR file in the same relative location.

   ---

3. On your local computer, test your Web pages by opening them in a Web browser from outside of the Visual Cafe environment.
4. Put your JAR file and HTML files in a directory on the Web server, as they appeared in your Visual Cafe project directory.
5. After completing the setup of your Web site, test your Web pages from remote computers. You can also test it with different operating systems and Web browsers.

To deploy your applets outside of a JAR:

Here is one way:

1. Use the JAR command under the Project menu to create a JAR file containing your applets and all supporting files, including Symantec class files. See Chapter 7, “Working with JavaBeans” for more information.

   Tip: Your applets should use relative URLs for graphics files so you can easily move your applets to different computers.

2. Create a new directory for your deployment files.
3. Expand the JAR file into the new directory. See Chapter 7, “Working with JavaBeans” for more information.

   Note: If you want to put the class files your applets use into a JAR file, create the JAR file then add the variable `ARCHIVE="name.jar"` to the applet tags in your HTML files. You can specify multiple JAR files by delimiting them with a comma (,).

4. Copy the HTML files for your Web pages into the new directory.

   Tip: Remember that the applet tags specify where your applets are relative to the location of the HTML files.

5. On your local computer, test your Web pages by opening them in a Web browser from outside of the Visual Cafe environment.
6. Put your files in a directory on the Web server, as they appeared in the directory on your local computer.
7. After completing the setup of your Web site, test your Web pages from remote computers. You can also test it with different operating systems and Web browsers.

Here is another way:

1. Create a deployment directory on your local computer.
2. Put all of your applet class files, HTML files, and other supporting files (such as graphics files) in the deployment directory, as they appeared in your Visual Cafe project directory.

   Tip: Your applets should use relative URLs for graphics files so you can easily move your applets to different computers.

3 Enable your applet to access the class files it needs, including the Symantec class files. (Remember that the Web browser should provide access to standard Java class files.) See "Determining what class files an applet or application needs" on page 6-16 for more information.

   If you want to put the class files your applets use into a JAR file, create the JAR file then add the variable `ARCHIVE="name.jar"` to the applet tags in your HTML files. You can specify multiple JAR files by delimiting them with a comma (,).

   If the class files are not in a JAR file, you can place the files in the deployment directory, preserving the directory structure and case of the class names

4 On your local computer, test your Web pages by opening them in a Web browser from outside of the Visual Cafe environment.

5 Put your files in a directory on the Web server, as they appeared in the directory on your local computer.

6 After completing the setup of your Web site, test your Web pages from remote computers. You can also test it with different operating systems and Web browsers.

Including your applet in a Web page

HTML stands for HyperText Markup Language. A markup language is made up of a system of tags that preserves a document's structure so that it can be disassembled, moved electronically, and reassembled when it reaches its final destination. The "Markup" part of HTML refers to tags that affect the way information in a document will be displayed. By using different HTML tags, you control text editing, formatting, how graphics and animations are displayed.

Java is an interpreted, Object Oriented programming language with a syntax and structure similar to C++, designed by Sun specifically for the Internet. One of the fundamental features about the Java programming language is that it is platform independent. Java applets and applications will run on any computer platform that has Java installed on it.

An HTML document can contain links to Java programs so that you can display a Java program alongside regular text in a Web page Because HTML now supports Java, users get the best of both worlds. Visual Cafe can provide the necessary HTML you need to run and view your applet within Visual Cafe. However, Visual Cafe is not intended as a tool for writing HTML documents. You can achieve this task with Symantec's Visual Page HTML editing application.

HTML, Java, and the World Wide Web

Applets are designed to be embedded in Web pages. Basic operations such as starting, stopping, and displaying the applet are all handled by the Web browser. In order to tell the Web browser to display the applet, you have to put certain information about the applet in the HTML file.

For the Web browser to be able to display an applet, it requires some basic information. provided through the use of the <APPLET> tag. Within the scope of the <APPLET> tag you specify where to find the class file, and how large to make the display space within the Web page. Like many HTML tags, <APPLET> marks the beginning of it scope and </APPLET> marks the end. The <APPLET> tag is a link to a class file that contains bytecode. The Web browser interprets the bytecode and displays the applet in the Web page.

In addition, there are three attributes for the Applet tag: `CODE`, `WIDTH`, and `HEIGHT`.

Viewing and editing HTML files

If HTML files are included in your project, you can view the file and edit its HTML code from within Visual Cafe.

To edit an HTML file:

1. Open the file by using one of these methods:
   - In the Project window, double-click on the HTML file name.
   - From the File menu, choose Open, and select the file.
2. In the Source window, make any necessary edits.
3. Save the file (from the File menu, choose Save).

Adding an applet to an HTML page

After assembling your applet and testing it in Visual Cafe, you are ready to add the applet to your HTML page.

Using the <APPLET> tag, you specify (at a minimum) the location of the `Applet` subclass and the dimensions of the applet's on-screen display area. When a Java-capable browser encounters an <APPLET> tag, it reserves on-screen space for the applet, loads the `Applet` subclass onto the computer the browser is executing on, and creates an instance of the `Applet`

subclass. Next, the browser initializes the applet, the applet is off and running.

The bold lines of the following listing comprise the <APPLET> tag that includes the "Hello World" applet in an HTML page.

```
<HTML>
<HEAD>
<TITLE> A Simple Program </TITLE>
</HEAD>
<BODY>
Here is the output of my program:
<APPLET CODE="HelloWorld.class" WIDTH=150 HEIGHT=25>
</APPLET>
</BODY>
</HTML>
```

The <APPLET CODE="HelloWorld.class" WIDTH=150 HEIGHT=25> tag specifies that the browser load the class whose compiled code is in the file named HelloWorld.class. The browser looks for this file in the same folder (directory) as the HTML document that contains the tag.

When the browser finds the class file, it loads, creates, and displays an instance of the class. If you include an <APPLET> tag twice in one HTML page, the browser loads the class file once and creates and displays two instances of the class.

The WIDTH and HEIGHT attributes are like the same attributes in an <IMG> tag: They specify the size in pixels of the applet's display area. Most browsers do not let the applet resize itself to be larger or smaller than this display area.

To ensure that the required Symantec custom classes are available to your applets at all times, move the \symantec folder structure to your HTML folder. The symantec folder is found in \java\lib of the Visual Cafe installation folder.

Passing parameters to applets from the HTML file

It is often helpful to have an applet receive information from an HTML document. This allows people to customize how applets appear in their Web pages. To pass information from an HTML document to a Java applet, use the <PARAM> tag.

The <PARAM> can appear between the HTML <APPLET> and </APPLET> tags. The <PARAM> tag has two attributes, NAME and VALUE, which are used to pass data to a Java applet.

```
<APPLET CODE="MyApplet.class" WIDTH=100 HEIGHT=100>
<PARAM NAME="Color" VALUE="red">
<PARAM NAME="Number" VALUE="81">
</APPLET>
```

The HTML file can pass multiple parameters. After the parameters are set in the HTML file, an applet can retrieve them through the `getParameter` method

```
String x=getParameter("Color");
```

In the example above, we declare a variable `x` of type `String` to hold the `Color` parameter retrieved from the HTML document. It is necessary to specify the name of the parameter to retrieve from the HTML file.

The parameter is named `Color`:

```
<PARAM NAME="Color" VALUE="red">
```

The name of the parameter is specified as the argument passed to `getParameter` in the applet code.

```
getParameter("Color");
```

This results in the `getParameter` method returning the value “red” from the HTML file.

Deploying your application

After you complete an application in Visual Cafe, you are ready to deploy it. Requirements for different applications vary. However, here are some guidelines:

To deploy your application in a JAR:

1. Use the JAR command under the Project menu to create a JAR file containing your application and all supporting files, including Symantec class files. See “Using JAR files” on page 7-10, for more information.

Tip: Your application should use relative URLs for graphics files so you can easily move your application to different computers.

---

2 On your local computer, test your application by running it from outside of the Visual Cafe environment. To start your application, the JAR file must be in the class path. For example, you could type at a DOS prompt:

```
set classpath=%classpath%;name.jar
```

Then you can run your application:

```
java application-name
```

`java` invokes `java.exe` and, if needed, includes the complete path, for example, `\visualcafe\java\bin\java`. Your application name is the same as the name of the frame for the main application window without the class extension; it is case-sensitive and you might need to include the complete path.

For example:

```
set classpath=%classpath%;Amazing.jar
\visualcafe\java\bin\java AmazingTour
```

3 Test your application from remote computers. You can also test it with different operating systems.

Note: Your users need to obtain or you need to provide the Java virtual machine and standard Java class files. The Symantec Java virtual machine must be licensed from Symantec.

To deploy your application outside of a JAR:

Here is one way:

1 Use the JAR command under the Project menu to create a JAR file containing your application and all supporting files, including Symantec class files. See "Using JAR files" on page 7-10.

Tip: Your application should use relative URLs for graphics files so you can easily move your application to different computers.

---

2 Create a new directory for your deployment files.

3 Expand the JAR file into the deployment directory. See "Using JAR files" on page 7-10, for more information.

Note: You can put the class files your application uses into a JAR file, but remember that the JAR file must be added to the class path on the computer the application runs on.

---

4 On your local computer, test your application by running it from outside of the Visual Cafe environment.

To run your application, type at the DOS command line:

```
java application-name
```

`java` invokes `java.exe` and, if needed, includes the complete path, for example, `\visualcafe\java\bin\java`. Your application name is the same as the name of the frame for the main application window without the class extension; it is case-sensitive and you might need to include the complete path.

For example:

```
\visualcafe\java\bin\java AmazingTour
```

5 Test your application from remote computers. You can also test it with different operating systems.

Note: Your users need to obtain or you need to provide the Java virtual machine and standard Java class files. The Symantec Java virtual machine must be licensed from Symantec.

Here is another way:

1 Create a deployment directory on your local computer.

2 Put all of your application class files and other supporting files (such as graphics files) in the deployment directory, as they appeared in your Visual Cafe project directory.

Tip: Your application should use relative URLs for graphics files so you can easily move your application to different computers.

---

3 Enable your application to access the class files it needs, including the Symantec class files. (Remember that the Java virtual machine should come with the standard Java class files.) See "Determining what class files an applet or application needs" on page 6-16 for more information.

You can put the class files your application uses into a JAR file, but remember that the JAR file must be added to the class path on the computer the application runs on.

If the class files are not in a JAR file, you can place the files in the deployment directory, preserving the directory structure and case of the class names

4 On your local computer, test your application by running it from outside of the Visual Cafe environment.

To run your application, type at the DOS command line:

```
java application-name
```

`java` invokes `java.exe` and, if needed, includes the complete path, for example, `\visualcafe\java\bin\java`. Your application name is the same as the name of the frame for the main application window without the class extension; it is case-sensitive and you might need to include the complete path.

For example:

```
\visualcafe\java\bin\java AmazingTour
```

5 Test your application from remote computers. You should also test it with different operating systems.

Note: Your users need to obtain or you need to provide the Java virtual machine and standard Java class files. The Symantec Java virtual machine must be licensed from Symantec.

Determining what class files an applet or application needs

You can learn what class files you need by creating a JAR file with the Visual Cafe JAR utility or by using SJ.

Remember that when you deploy you need to keep the class directory structure intact and you must use the same case in the class names. (Class names are case-sensitive.)

Using the JAR command to get the class files your Java program needs

When you create a JAR file by choosing JAR under the Project menu, Visual Cafe adds the class files your Java programs require. You can use the JAR file to deliver your Java programs, or expand the JAR file with `jar.exe` to extract the class files from the JAR file. See “Creating a JAR file” on page 7-10, and “Expanding a JAR file” on page 7-11 for more information.

Using SJ to determine what class files your Java program needs

The sj.exe utility enables you to easily figure out which class files are used by your applet or application. After your Java applet or application is finished, enter the following command at the DOS prompt (make sure that \visualcafe\bin is in your path):

```
sj.exe -make -cdb mainclass.cdb -depend listname.dep
 mainclass.java
```

mainclass.java is the name of the main class Java file in your project.

listname.dep is the file where the list of classes used by your applet or application will be generated. sj.exe also lists the name of the ZIP file (if any) where the class file was found. This helps you figure out what class files you would need to unzip from each class ZIP file used by your applet or application. The filename must have a .dep extension.

Even though the standard java.* class files are logged in this file, you are not actually required to copy those to your Web server for your applets, as they are usually available with most Web browsers. For applications, the standard Java class files are part of Java Runtime Environment, which you can download from JavaSoft.

sj.exe takes additional command line options, so you can add additional class paths, and so on. See Compiling from the command line, in Chapter for more information.

Configuring UNIX-based Web servers

The following instructions for system administrators are guidelines only; they have been tested with Apache. After this is set up, all applets in the user's home HTML directory would have access to the Symantec classes.

To configure a UNIX-based Web server:

1. Create a UNIX directory, such as /home/symantecclasses. Make sure all users have read access to the directory.
2. Copy the Symantec directory, including all subdirectories, to this UNIX directory. For Visual Cafe Pro applications that use database features, you also need to unzip Dbaw.zip and Dbaw_awt.zip into the Symantec directory and Sql.zip into a Symjava directory, which you must also copy to the UNIX directory.

3 Create a symbolic link from `/home/symantecclasses/symantec` to the user's home html directory. For example:

```
ln -s /home/symantecclasses/symantec
/home/joeuser/public_html
```

joeuser can now run applets from Web pages without the need to copy the full Symantec class structure to his `public_html` directory.

Building your Java program

This section documents the commands of the Project's File menu. The final section of this chapter describes the use of the Messages window.

Many of the commands in this menu, such as Compile, operate on the currently selected file or files. If the frontmost window is an Editor window, the selected file is the source file in that window. If the frontmost window is a Project window, all files currently selected in that Project window will be affected.

Building your project with commands in the Project menu

The Project menu contains all the commands that turn source code into object code. You use the commands in this menu to precompile header files to allow for faster builds, to turn source code into object code, and to link the object code in a project into an application or library.

You use the Project menu commands to perform the following primary building functions:

Execute

When you choose Execute, Visual Cafe compiles and runs the current project with no debugging. Applets are run in the Symantec Applet viewer by default. You set the default viewer in the Project tab of the Project Options dialog box.

Run in Debugger

With this command, Visual Cafe runs the program in debug mode and stops at the first breakpoint.

Step into

This command begins running the program by stepping into the first line of source code. When used on an applet, this command takes you the applet `void init()` method if one is implemented.

Build Applet/Application

You can build the applet or application with this command, but it does not execute the resulting program after building it.

Compile

If the frontmost window is an Editor window or a Project window with one or more source files selected, this command is named Compile. Selecting it compiles the file (or files, if more than one source file is selected in the Project window). If the frontmost window is an Editor window containing a file that is not already in the Project window, the file is added to the project if it is successfully compiled.

Parse All

Parses all files in the project.

Messages window

When an error (or warning) occurs while a compiler compiles or precompiles a source code file, Visual Cafe displays the compiler's message in the Messages window for that file's project. The Messages window

opens automatically any time errors are detected during compilation. You can also open the window by choosing Messages from the Window menu.

The contents of the Messages window are saved when the window is closed and are displayed again when the window is next opened. The Messages window displays compilation errors and warnings for all files in the project. Each error or warning is displayed on two lines: the first lists the file and line number where the error or warning occurred and the second gives the message itself.

Error messages and warning messages for all files in the project are listed in the Messages window. The most recent messages are displayed at the bottom of the list. If a file for which the Messages window was displaying errors or warning messages is recompiled, the existing messages are deleted from the window and any new messages are added at the end of the list.

CHAPTER 7

Working with JavaBeans

One of the biggest endeavors of software developers has been to develop a way to make source code reusable in other development projects that are next door or across the Internet. With the development of object-oriented programming and languages like Java, this endeavor is starting to pay rewards.

JavaBeans and Java

Until recently, however, the Java language did not offer true reusability. True reusability allows other developers to use your code without having to recompile it for a particular platform. Moreover, true reusability means that other developers can integrate your code into their projects without having to recompile their source code. Thus, the JavaBeans API was developed to help solve these issues in software development.

Here are some of the features of JavaBeans components (“Beans”) that make them very attractive to software developers:

- Beans are discrete. Beans are typically small and have very specific functionality. Beans can be used with other Beans to make larger and more complex Java programs. Note that although many Beans are small, a Bean is not restricted in size. For example, a Bean could be a spreadsheet or a fully functional spelling checker.
- Beans have reusability. Beans can be reused over and over again in unlimited numbers of Java programs. Examples of reusability are beans that perform various financial transactions.
- Beans can be configured visually in some kind of visual tool, such as Visual Cafe. Bean properties can be easily configured to allow your Java Bean to interact with other Beans that have specific properties.

These specific properties can be passed on to a Java Bean builder application, such as Visual Cafe.

- Beans can communicate with other component models. For example, you can construct Bean Bridges to interact with Microsoft's Active X modules.

JavaBean terminology

Throughout this manual (and in other trade publications) the terms *JavaBean*, *JavaBeans component*, and *Bean* all refer to the same thing. Also keep in mind that the term *JavaBeans* typically refers to the component technology itself and not to multiple Beans.

Basic JavaBean structure

In the simplest sense, a JavaBeans component consists of two fundamental parts: methods and data. This is no different than other objects in object-oriented environments. The data portion of the Bean describes the Bean's state, the methods provide the interface to manipulate the Bean's state.

In addition, a Bean is capable of responding to events.

A Bean's methods can be public or private. A Bean's public methods are often grouped together according to their functionality. These groups are know as interfaces. It is through these interfaces that the Bean communicates to the outside world.

The JavaBeans services

For all the fantastic things that JavaBeans components can accomplish, you might think that there is some highly complex system at work. Actually, Beans have been designed to be fairly simple to use. This lack of complexity is the power of the JavaBeans API.

The JavaBeans API is an extension of the Java language. JavaBeans is ultimately a programming interface which means that all of its features are implemented as extensions to the Java class library. Specifically, the component model that is described by the JavaBeans specification contains five major services:

- Property Management

- Introspection
- Event handling
- Persistence
- Application builder support

Property management

Properties of a Bean reflect the internal state of a Bean and constitute the data part of the Bean's structure. *Properties* are discrete, named attributes of a Bean that determine its appearance and behavior. Properties include attributes that are associated with the Bean, such as color, size, or the label string.

Properties can be changed in a number of ways: at runtime through setter and getter methods; by scripting; or by the BeanInfo Editor.

The following are some examples of how Bean properties can be accessed:

- Programmatically via public accessor methods
- Visually through Visual Cafe's property sheets
- Through persistent storage and retrieval of a Bean
- As object fields in scripting environments (VBScript or JavaScript)

Accessor methods

Accessor methods are the primary means by which a Bean's properties are exposed. An *accessor method* is a public method defined in the Bean that reads/writes the property value.

An accessor method is a public method defined in the Bean that reads and writes the property value in the Bean. A Bean property typically has a pair of accessor methods called getter and setter methods. A *getter method* gets (reads) the property value; a *setter method* sets (writes) the property value. Each property must have a getter method, the setter method is optional (although typically included). The getter method returns the property value, the setter method sets the property value.

Indexed properties

In addition to single-value properties, JavaBeans supports *indexed properties*—properties that represent an array of values. Indexed properties

in Beans are similar to indexed properties in Java, you access a specific value using an integer index into the array.

Indexed properties are useful when a Bean needs to maintain a group of properties. For example, a Bean may want to maintain an array of the methods of another Bean obtained through introspection of that other Bean.

Bound and constrained properties

JavaBeans supports two advanced level mechanisms for working with properties—bound and constrained properties. A *bound property* is a property that provides notifications to any interested party when the property value changes. A *constrained property* enables an interested party to perform a validation on a new property value before accepting the change.

Actually, any property can be designated as bound or constrained. Bound and constrained properties provide notifications to an interested party based on changes to that party. An *interested party* is an application, applet, or another Bean that needs to know about changes in the property.

Bound properties are defined at the component level, meaning that the Bean is responsible for specifying that a property is bound. For example, the visibility property of a Bean could be a bound property

A constrained property enables an interested party to perform a validation on a changed property value prior to the Bean accepting the change. A constrained property is useful to enable an interested party to control how the Bean is modified. For example, a date may be designated as constrained where the application containing the Bean needs to limit the valid range of dates.

Bound and constrained properties are not mutually exclusive. That is a property may be designated as bound or constrained—or neither or both.

Introspection

Introspection is the ability for a Bean to make public (or “publish”) the operations, methods, and properties it supports, as well be able to discover operations, methods, and properties of other Beans.

Note: Although the introspection services in JavaBeans is designed primarily for use by application builder tools such as Visual Cafe, they are a separate service because they can be used independently of Visual Cafe.

Introspection calls on two API processes: the Java Reflection API, and the Java Serialization API. The Java Reflection API is a set of classes that look into a class file and examines the properties, methods, events, and properties of Beans. The Java Serialization API is used to store the class, including its state. Visual Cafe uses these two Java API functions to allow easy creation and modification of Beans.

Reflection and design patterns

Beans can determine information about a Bean's properties, methods, and events by analyzing the Bean using a set of low-level reflection services. *Reflection* is the process of querying a Bean to determine information about its public facilities and functionality.

These services gather the Bean's information by applying simple design patterns. *Design patterns* are rules used to determine information about a Bean from its reflected method names and signatures. The JavaBeans introspection facilities match them based on a specified design pattern of the method names and automatically determine the property they access.

Design patterns rely on the convention of method names and signatures conforming to a standard convention as defined in the JavaBeans specification. This approach to introspection encourages Bean developers to use consistent naming conventions.

Explicit Bean information

Design patterns are not required, or strictly enforced—you are free to use whatever naming conventions you want. If you choose to do so, Beans must use the explicit introspection facility. You must provide specific information about the Bean including a property list, method list, and event list. This information goes into a `BeanInfo` class that must be included with the Bean in its Java Archive (JAR) file. Although this method is not automatic, there may be some situations where this approach is advantageous.

The Introspector

The Introspector service provides the facilities to consolidate these two approaches. First, the introspector traverses the inheritance tree of a Bean trying to determine the explicit Bean information for all parent Beans. If at any point, the explicit information is undefined, the introspector reverts to the reflection services using design patterns to determine the external Bean information.

The JavaBeans Introspector Service provides the best of both worlds. First it tries to use information explicitly provided by the Bean's developer. If that information is not provided, the introspector relies on design patterns to try to get the Bean information that way.

Finally, the introspector supports a combination of the two approaches. For example, the methods in a Bean could be explicitly defined in the BeanInfo class, but the events and properties could be determined using design patterns.

Event handling

A Java Bean must have a system of event handling for the methods of one Bean to call the methods of another Bean. The event handling facilities determine how Beans respond to changes in state and how these changes propagate to applications (and other Beans). The event handling in JavaBeans works with the concepts of event sources and event listeners.

An *event source* is a Bean capable of generating events. An *event listener* is an application (or Bean) capable of responding to events. An e*vent state object is* used to store information associated with an event.

Event sources and event listeners are connected using an event registration mechanism that is part of JavaBeans. This registration links the source with one or more listeners. When the source generates an event, a designated method is called on the listener with an event state object sent as an argument. Event state objects carry information related to the event with them.

Unicast and multicast event sources

In practice most event sources are multicast event sources. That is, more than one listener can be registered with the event source.

A *unicast event source* is an event source that can generate events for a single listener. *Multicast event source* is an event source that can generate events for more than one listener.

The main difference between the two is that a unicast event source throws an exception when an attempt is made to register more than one listener.

Note: You should avoid using unicast event sources unless there is a specific reason to do so.

Event adapters

JavaBeans provides the event adapter mechanism for those situations where you need more control than the standard source/listener model provides. Event adapters sit between a source and its listeners, providing specialized event delivery behavior. An *event adapter* is an intermediary placed between an event source and listener that provides additional event delivery behavior.

Event adapters allow you to implement highly specialized event handing tailored to unique application design.

Persistence

Persistence is the ability for the component to remember and store the state of the component long after you are finished working with it. When you change the state of your Bean, you might want your Bean to store or *persist* the changed state. Bean states can change because of some action affecting it during development or runtime.

There are two ways to store Beans: automatically through the Java Object Serialization mechanism, or through a future externalizing stream that permits the Bean total control of its persistence.

Beans store information about their internal state such as their properties and their look and feel so that the Bean can easily re-create itself when necessary. Also, a Bean can store the properties of other Beans that it references, but this practice can be dangerous because the referencing Bean assumes that the other Beans have persisted themselves.

Bean storage

Beans are stored in JAR (Java ARchive) format. These JAR files are Zip files that contain another component called a manifest file. This manifest file contains additional information about what else is in the JAR file.

Application builder support

The application builder support service that is built into JavaBeans is what enables Visual Cafe to smoothly integrate your Bean into your container (application or another Bean). These facilities are what enables Visual Cafe to construct your Bean with little or no programming effort.

An intended side effect of the application builder support is the separation of design-time and run-time code. It would be wasteful to bundle the design-time specific code with your Bean for distribution (think of the Web bandwidth). Separating the design-time code into a separate file solves this problem.

Property editors and sheets

Visual Cafe supports the editing and manipulation of a Bean's properties through property sheets. A *property sheet* is a Visual Cafe user interface that contains property editors for each of the exported properties of the Bean. A *property editor* is a Visual Cafe user interface that enables the visual editing of a single property.

Creating a Bean

A Bean is typically a small development project, in terms of the volume of code. However, you want to design your Bean to be reused—that's one of the main reasons for creating Beans. It is important for you to fully evaluate the functional specifications of the Bean before you begin writing any code. This has always been good programming practice, however it is especially crucial because of the nature of JavaBeans components and how they are used. For example, you want your Bean to be backward compatible as it evolves so you need to make sure that the interfaces to the Bean are extensible enough to provide room for added functionality.

Bean design fundamentals

The design process does not need to be complex. At a minimum you should consider questions such as the following:

What does the Bean do?

The answer to this question helps you to clearly identify what the Bean is to accomplish as a reusable piece of software.

How is the Bean used?

The answer to this question depends upon the functionality you have in mind. For example, you might be planning a Bean as a specific part of an application. It might be visible or non-visible. It might be a customizable component.

No matter what you are planning for the Bean, be sure you have a good idea of how the Bean is to be used by the end user.

How might the Bean be modified?

This question is typically overlooked by Bean developers. You should try to determine potential upgrade features or changes and how those relate to the current state of the Bean. Performing this exercise might reveal that a small change in the design may provide for increased adaptability.

What kind of Interface does your Bean need?

How will your Bean need to get input and handle event notification?

Testing your Bean

As you develop your Bean, you can incrementally test and debug the functionality using Visual Cafe. Visual Cafe is a feature rich development environment for creating applications with Beans.

Adding and using Beans in Visual Cafe

You can add Beans to the Visual Cafe environment as a class, from within a zip, or from a JAR file. You add the Bean from the Add Bean command from the Edit menu.

A Java Archive (JAR) file is a compressed archive file that complies with the JavaBeans standard. It is the primary method for delivering JavaBeans components.

A JAR file contains one or more related Beans, and any support files, including classes, icons, graphics, sounds, HTML documentation, serialization files, and internationalization files. A JAR tool, called `jar.exe` on Windows computers, archives and extracts JAR files and is provided with JDK 1.1.

Using JAR files

In Visual Cafe, to use the JavaBeans components in a JAR file, you must first add the file to the Component Library. Then you can add the components to your projects. If HTML documentation was included in the JAR for a Bean and you want to look at it, you need to expand the JAR by using `jar.exe`.

Creating a JAR file

Creating a JAR involves adding a class, classes, a zip file, or a JAR file to the Components folder. You then open the Beans tab in the Preferences dialog box and select the class or classes that you just added to the Global Classpath. The Beans are introspected and show up in the Component Library and optionally in the Component Palette. By default all extensions of `java.awt.Component` should show up without much intervention from you.

Visual Cafe provides a tool you can use within its environment to quickly create JAR files.

To create a JAR file:

1. While the project you want to work with is active, choose the JAR command from the Project menu.
2. In the JAR name field, type the name and full path that you want the JAR file to have. Click … to browse.

   Note: You cannot add to an existing JAR file, but you can create a new JAR file.

3. Click More to display the files and set options for them.

   The JAR utility sets up much of the JAR for you.

Initially, the display includes all of the classes in your project, any classes they depend on, and other files you have added to the project. Graphics files associated with Visual Cafe components might also be added for you.

- To add more files, click Add Files and select the files to add.
- To remove a file, select the file and click Remove.

4 Select a file to specify options for it:

- Select Is Bean if the class is a Bean.
- Select Design Time if it is a class only needed at design time, such as a BeanInfo file.
- To specify dependencies within the JAR, click Depends and select a file that the class depends on, such as a graphics file or another support file. (It will appear subordinate to the class in the display; this command is the same as the depends flag in the manifest file.) A file must be added to the JAR before you can specify dependencies with it; clicking Depends does not let you add files.

5 Click OK to create the JAR.

Expanding a JAR file

Visual Cafe provides a tool you can use within its environment to quickly create JAR files. To expand JAR files, use the `jar.exe` utility in the `java\bin` subdirectory. For example, to expand the file `Amazing.jar`, enter the following at a DOS prompt:

```
jar -xf Amazing.jar
```

Converting components (description files) to JavaBeans

In previous versions of Visual Cafe, you needed to create a separate description file with an extension of `.desc`. This file contained the explicit instructions on how the Bean was supposed to function and what results to expect.

With a Bean, description files are no longer needed, and if you want to reuse a component that you developed previously, or to use third-party components, you need to convert your description file to a Bean. Visual Cafe has a utility called the Description File Converter to help you with the conversion process.

Visual Cafe provides a utility for converting Visual Cafe description files so you can implement the JavaBeans standard with your custom components.

To convert a component to a Bean:

1 Run the Description File Converter.

To do so, double-click the `DescToBeanInfo.bat` batch file, which is in the `bin\DescFileConverter` directory of the main Visual Cafe directory.

The Description File Converter dialog box appears. You can view the Java version by going to the Help menu and choosing Environment.

2 Click the Location tab.

3 In the Description File Directory field, specify the full path to the directory containing one or more description files you want to convert. You can use the Browse button to specify the directory.

4 If the component icons are with the description files, select `.ico` Files Are With `.desc` Files. Otherwise, in the Icon File Directory field, specify the location of the corresponding icon files, if present.

5 In the Output Directory field, specify the full path to a directory where you want your output files (BeanInfo and `gif`) to go.

When the description file is converted, the name of the output file is the first part of the Bean name (without the extension) appended with `BeanInfo.java`; the icon files are converted to `gif` files.

6 Select relative or absolute.

If you select relative, the fully qualified class name is used to create a directory structure subordinate to the output directory. For example, if the output directory is `c:\temp` and the class name is `symantec.beans.Beans`, `Beans.java` will be placed in the directory `c:\temp\symantec\beans`.

If you select absolute, all files are placed in the directory you specify. This could potentially cause file name conflicts, because the package directory structure is not preserved.

7 Click the Selection tab and select the files you want to convert. Shift-click to select multiple files. Click Select All Listed Files to select all files.

8 From the File menu, choose Convert.

A dialog box appears when the conversion is complete. The files display in the output directory; if the path was relative, the files are in a directory subordinate to the output directory. For each component entry in a description file, a `BeanInfo.java` file is created. For each

icon file, a 16 by 16 and 32 by 32 GIF file is created. For the latter, the 16 by 16 image is expanded to a 32 by 32 size.

Adding a JavaBeans component to the Component Library

You might get a Bean from your colleague, from your favorite software source, or even from the Internet. Where do you put it?

To use a JavaBeans component, you must add it to the Component Library. You can add class or JAR files. The Bean must comply with the JavaBeans standard for it to be added. After you add a Bean:

- The component appears in the Component Library.
- If an icon was specified in the `BeanInfo` file, the component uses that icon. If the `BeanInfo getIcon` method returns NULL, Visual Cafe uses the icon of a base class already in the Component Library. Visual Cafe examines the classes the JavaBeans component inherits from, and picks the class deepest in the inheritance hierarchy. The icon of this class is used with your new JavaBeans component.
- If the get and set methods conform to the JavaBeans design pattern, the component properties will appear in the Property List window.
- Visual Cafe derives the interactions displayed in the Interaction Wizard. It determines the interactions in one of two ways:
  - The interactions are specified directly in the `BeanInfo` class.
  - Visual Cafe uses introspection to look at the public methods and derives interactions from them.

To add a JavaBean to the Component Library:

1. Make sure the component class or JAR file is in the location you want to store it.

   Note: For class files, this location must be in your class path. For example, c:\VisualCafe\java\lib is in your class path. Remember that class files are case-sensitive. If you want to add the class file to a package, the class file must be in the package directory.

2. Display the Component Library by choosing going to the Window menu and choosing Component Library.

3. From the Insert menu, choose Component into Library. This menu item is available only when a project is open.

   An Open dialog box appears.

4. Select the class or JAR file, then click Open.

   For a class file, an Add to Library dialog box appears.

   For a JAR file, Visual Cafe inserts the Beans into the Component Library. The Beans are put in a group with the same name as the JAR file, unless another group name was specified in the Bean.

5. For a class file, select a group, then click OK.

   The component appears in the Component Library.

6. If you want to add the component to the Component Palette, select the component in the Component Library, right-click over the component, and choose Add to palette from the pop-up menu.

The component is added to the Component Library. You can verify that it is there.

Here are some considerations about inserting Beans:

- If Visual Cafe does not accept the Bean, it is not added to the Component Library. It most likely does not comply with the JavaBeans standard.
- If you modify a JAR file after it is in the Component Library, the change is not recognized until you restart Visual Cafe, or you delete the JAR and add it to the Component Library again.
- After you insert a JAR file, you should not move it to a new location. If you move it, Visual Cafe will be unable to find it.

Creating a JavaBean component

You can create JavaBeans components within the Visual Cafe environment. Visual Cafe provides tools to make your job easier. However, realize that you must be very familiar with the JavaBeans standard before you attempt to create custom JavaBeans components.

Tip: You can create a component template by dragging a component from the Project window (Objects view) to the Component Library. The source file is copied by Visual Cafe, so you do not have to keep the files in the same location. A component and a component template appear the same in the Component Library, and you add them to projects in the same way.

1. Create a new project for developing one or more JavaBeans components.

   You can start with the Basic JavaBean project template to get started quickly. See Chapter 3, "Working with projects and workspaces."for more information on starting a project.

2. If you are converting a custom component that has a description file, convert the description file. See "Converting components (description files) to JavaBeans" on page 7-11.

3. Create the Bean according to the JavaBeans standard.

   This step includes adding to the project any support files, such as classes, icons, graphics, sounds, HTML documentation, serialization files, and internationalization files. If you add support files to the project, they are automatically included when you create a JAR file from the project with the Visual Cafe JAR utility.

4. Optionally add to the `BeanInfo` some information for better integrating into the Visual Cafe environment. "Adding Visual Cafe information to a JavaBean" on page 7-15.

5. To package the Bean(s) in a JAR file, choose JAR from the Project menu. See "Creating a JAR file" on page 7-10

6. To use and test the Bean within the Visual Cafe environment, add it to the Component Library. See "Adding a JavaBeans component to the Component Library" on page 7-13

Adding Visual Cafe information to a JavaBean

JavaBeans has introspection, the ability to read JavaBeans classes directly with the Core Reflection API using the Introspector class. This information is stored in a `BeanInfo` object and includes data such as properties, events, and all the accessible methods. In addition, you can add information about how a Bean should integrate into the Visual Cafe environment. The integration information is provided through two Visual

Cafe classes: `symantec.itools.beans.SymantecBeanDescriptor` and `symantec.itools.beans.ConnectionDescriptor`.

A quick description and code samples follow; for more information, see the Java API reference, which is available from the Help menu.

SymantecBeanDescriptor

The `SymantecBeanDescriptor` class extends from `java.beans.BeanDescriptor` and is the main way to supply Visual Cafe integration information. The `SymantecBeanDescriptor` methods let you tell the Visual Cafe environment about the following:

- The name of a Component Library group (folder) to put the Bean in
- The name of a Component Palette tab to put the Bean in
- Whether to not allow users to drop other components into this Bean (if this Bean derives from `java.awt.Container`)
- Visual Cafe flags, such as INVISIBLE for specifying invisible components
- Visual Cafe connections (used by the Interaction Wizard) that are not tied to a specific method

ConnectionDescriptor

The `ConnectionDescriptor` class extends from `java.beans.FeatureDescriptor`. A ConnectionDescriptor encapsulates a Visual Cafe connection, which is used by the Interaction Wizard. A Visual Cafe connection defines an interaction, or connection, for a Bean. It implies a relationship between objects (or between an object and itself) involving either event notification or data transmission. The Interaction Wizard allows users to graphically build these relationships between objects, and Visual Cafe is able to generate the code for the specified relationship based on the underlying connection information encapsulated in the ConnectionDescriptor.

The connection is made of five pieces:

- **Form** – Determines whether the value of a connection is INPUT or OUTPUT. An input connection defines an interaction that sets data or initiates the execution of a method; an output connection defines an interaction that returns data.
- **Type** – Sets the Java type of the input or output value of the connection.

- **Expression** – Defines the code that is generated to create the connection. Properties and the following replacement variables are allowed in the code string:
  - *%name%* – name of the class/component
  - *%class%* – full class name of the class/component
  - *%arg%* – method argument used for output connection data
- **Initialization** – Defines any initialization code that needs to be present prior to the code generated from the connection expression.
- **Description** – Supplies an English description of the connection. This string appears in the Interaction Wizard. *%class%* is allowed.

A Visual Cafe connection looks like a method call. However, a connection can be more than that. It can be a "meta-method." Any valid code expression can be used to generate the connection, for example, the connection "Toggle pause" might have

```
%name%.setPaused(!%name%.isPaused());
```

as the code expression. (In this expression, *%name%* is a replacement variable for the name of the component class). A connection does not have to be tied to a method, for example, the connection "Point the button arrow LEFT" might have the code expression *%name%*.LEFT.

The `ConnectionDescriptor` methods allow you to set the form, type, expression, initialization, and description of a particular connection. If a connection is tied to a specific method, its ConnectionDescriptor is associated with that method's `MethodDescriptor`. Currently, the association uses the `MethodDescriptor`'s inherited method `setValue`, passing in a Vector of all `ConnectionDescriptor` objects to be associated with that method. All connections that are not tied to a specific method have their `ConnectionDescriptor` objects associated with the Bean's `SymantecBeanDescriptor` object.

Code samples

Following are three code samples that show three different ways of implementing the `BeanInfo` information specific to Visual Cafe. Here is the first sample:

```
public BeanDescriptor getBeanDescriptor() {


   SymantecBeanDescriptor bd = new
 SymantecBeanDescriptor(beanClass);
```

```
    bd.setCanAddChild(false);
    bd.setFolder("Additional");
    bd.setToolbar("Additional");

    bd.addConnectionDescriptor(new ConnectionDescriptor("output",
 "int", "",
                                    "%name%.BORDER_REGULAR",
                                    "BORDER_REGULAR"));

    bd.addConnectionDescriptor(new ConnectionDescriptor("output",
 "int", "",
                                    "%name%.BORDER_NONE",
                                    "BORDER_NONE"));

    return (BeanDescriptor) bd;
    }
Here is a second style:
ConnectionDescriptor cd = new ConnectionDescriptor( );
    cd.setForm(ConnectionDescriptor.OUTPUT);
    cd.setType(“int”);
    cd.setExpr(“%name%.BORDER_NONE"” );
    cd.setShortDescription(“BORDER_NONE”);

bd.addConnectionDescriptor(cd);
Here is a third style:
ConnectionDescriptor cd = new
 ConnectionDescriptor(ConnectionDescriptor.OUTPUT );
    cd.setType(“int”);
    cd.setExpr(“%name%.BORDER_NONE"” );
    cd.setShortDescription(“BORDER_NONE”);

bd.addConnectionDescriptor(cd);
```

Adding JavaBeans to the Component Library

To use a JavaBeans component, you must add it to the Component Library. You can add class or JAR files. The Bean must comply with the JavaBeans standard for it to be added. After you add a Bean:

- The component will appear in the Component Library.

- If an icon was specified in the `BeanInfo`, the component uses that icon. If the `BeanInfo getIcon` method returns NULL, Visual Cafe uses the icon of a base class already in the Component Library. Visual Cafe examines the classes the JavaBeans component inherits from, and picks the class deepest in the inheritance hierarchy. The icon of this class is used with your new JavaBeans component.
- If the get and set methods conform to the JavaBeans design pattern, the component properties will appear in the Property List window. You can edit custom properties as a text field, a drop-down list, or in a dialog box, depending on the property. The dialog box appears if you click a field value, then click the … button.
- Visual Cafe derives the interactions displayed in the Interaction Wizard. It determines the interactions in one of two ways:
- The interactions are specified in the `BeanInfo` class.
- If the interactions are not specified in the `BeanInfo` class, Visual Cafe uses introspection to look at the public methods and derives interactions from them.

To add a JavaBeans component to the Component Library, follow these steps:

1 Make sure the component class or JAR file is in the location you want to store it.

Important: For class files, this location must be in your class path. For example, `\VisualCafe\java\lib` is in your class path. Remember that class files are case-sensitive. If you want to add the class file to a package, the class file must be in the package directory.

---

2 Choose Insert Component into Library. This menu item is available only when a project is open.

An Open dialog box displays.

3 Select the class or JAR file, then click Open.

For a class file, an Add to Library dialog box displays.

For a JAR file, Visual Cafe inserts the beans into the Component Library. The beans are put in a group with the same name as the JAR file, unless another group name was specified in the Bean.

4 For a class file, select a group, then click OK.

The component appears in the Component Library. You should verify that it is there. (You can display the Component Library by choosing Window, then Component Library.)

Notes:
If a Bean is not added to the Component Library, it most likely does not comply with the JavaBeans standard.

If you move or modify a class or JAR file after it is in the Component Library, the change is not recognized until you restart Visual Cafe, or you re-add it.

Some components are made of more than one class file, such as if the Java source file for a component has inner classes. You need to make sure these class files are in the class path. You do not need to add inner classes to the Component Library.

Deleting JavaBeans from the Component Library

As you are developing your project, you can delete user-created objects in the Component Library. Deleting a component from the Library also removes the component from the Palette.

To delete a JavaBean from the Component Library:

1. In the Component Library, select the object to be deleted.
2. From the Edit menu, choose Cut; or press the DELETE key.

Viewing and changing JavaBean properties

Each JavaBean has a set of properties that define its look and behavior. Visual Cafe provides a Property List from which you can directly modify these properties. When you change a property, the source code and Form Designer are immediately updated.

To view or change the properties of JavaBeans in a project, select one or more components in the Form Designer or Project window.

To view or change the properties of JavaBeans in the Component Library and Palette, select one or more JavaBeans in the Component Library. Remember that if you change the properties of a component in the Component Library, the change now appears every time you add that

component to a project. Any projects that already contain the component before the change are not affected.

To modify JavaBean properties:

1. To display the Property List, Go to the Window menu and choose Property List.
2. Select a component in the Form Designer, Project window, Component Library, or from the Property List pull-down menu. To select multiple components, CONTROL-click in the Form Designer, Project window, or Component Library, or Shift-click or drag in the Form Designer.

   When multiple components are selected, only their common properties are shown and editable. In the Property List, you see the heading title “Multiple Selection”.
3. To edit a property, click the right column, double-click the left column, or use TAB in the Property List.

   The right column displays a list of valid values or makes the text string editable.

   Properties with multiple values are marked with a plus sign (+); for example, the Font property. Click the + to expand the list.

   Properties that are defined using a dialog box are marked with the ellipsis button (…). Click the … button to get the dialog box.
4. Press ENTER or click somewhere else to make the change.

Tip: Press ESCAPE to cancel an edit and return the property to its previous value.

Packaging and deploying JavaBeans

Visual Cafe provides a utility to help you deploy your JavaBeans and other Java programs. With the `SJ.EXE` utility, you can put your Java program into a tidy package for deployment and distribution.

Visual Cafe’s tools for building Beans

With Visual Cafe, you are able to create Beans and add descriptive information (modify `BeanInfo`) about them to better fit the environment in which you are going to deploy them. The descriptive information is exportable to other environments.

You can also use existing Beans and add descriptive information (modify `BeanInfo`) about them to better fit the environments in where you are going to deploy them. The descriptive information is exportable to other environments, unless the Bean has an explicit `BeanInfo`, in which case the information is exportable only to other users of the Visual Cafe 2.0 environment.

Bringing Beans into the Visual Cafe environment

There are a number of ways a Bean can exist before it is brought into your Visual Cafe environment. Here are some common Bean cases:

- A Bean comes in as a class file with no explicit `BeanInfo`
- A Bean comes in as a class file with a `BeanInfo` that is not a subclass of s`ymantec.itools.beans.BeanInfo`
- A Bean comes in as a class file with a `BeanInfo` that is a subclass of `symantec.itools.beans.BeanInfo`
- A Bean comes in with source or is created within us and has a `BeanInfo` that is not a subclass of `symantec.itools.beans.BeanInfo`
- A Bean comes in with source or is created within us and has a `BeanInfo` that is a subclass of `symantec.itools.beans.BeanInfo`

BeanInfo for standard AWT components

The `symantec.itools.beans.infos` class contains `symantec.itools.beans.BeanInfo` classes for the standard AWT components. Visual Cafe 2.0 adds the `symantec.itools.beans.infos` to the `BeanInfoSearchPath` through the `setBeanInfoSearchPath` method of Introspector.

Bean associates

A Bean will have a number of files that will be associated with it. If a Bean has an explicit `BeanInfo` class, then this is one. If a Bean has one or more Resource bundles associated with it then this/these is also one. An explicit `BeanInfo` may also have associated files `.gif`, `.jpg`, or resource bundles - these are also regarded as associates of the Bean. Associated `.gif` and `.jpg` files are added to the project and should appear anywhere every file that is added to the project appears.

Appearance

Bean Associates do not appear in the object view (see target customer). They do appear in the packages and file views where appropriate.

Renaming a Bean

When a Bean is renamed, Bean Associates that are found through some mangling of the Bean name need to be renamed as well. The explicit `BeanInfo` of a Bean and any associated ResourceBundles would be examples of these.

Hierarchical properties

Expansion of hierarchical properties will occur in the `symantec.itools.beans.BeanInfo` class.

Internationalization properties

Internationalization properties will have a derivable key for the resource bundle. This will be *membername* + _ + *name of property*.

CHAPTER 8

Debugging Java Programs

After writing a program, it is not unusual to find that it doesn't work the way that you expected. Just because a program compiles without errors, does not mean that it will work correctly. Finding and correcting problems within a program is called debugging. When you run your program, you may encounter a variety of errors, including compile errors (code construction, syntax errors), run-time errors that occur after you start the program (dividing by zero or writing to a file that doesn't exist), and logic errors (program doesn't do what you want it to do).

The Visual Cafe Debugger:

- Provides a Windows graphical user interface
- Provides a graphical representation of data structures
- Supports breakpoints
- Lets you drag and drop to execute commands
- Supports incremental debugging, allowing you to make changes while a program is paused in the debugger and continue running with the new changes
- Provides a Messages window separate from the Debugger, to display the status of your debugging session.

Note: Your program must successfully compile into a `.class` file before the Debugger can open it.

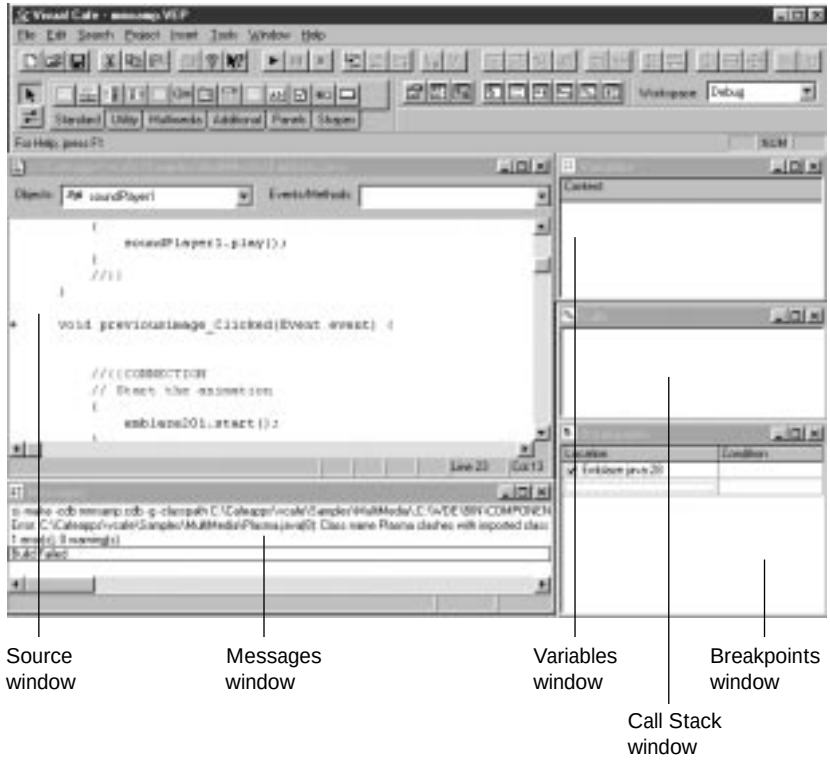
When you run the Debugger, Visual Cafe switches into debug mode. While you are in debug mode you cannot compile your source files, or rebuild your project. To try out your code changes you must first stop the

Debugger and recompile your program. Once the program has recompiled, you may restart the Debugger.

Using the Debug workspace

After building your project you can enter debugging mode by choosing Project, then Run in the Debugger.

The Debug workspace is automatically loaded, but not enabled by default. The Debug workspace includes the Call Stack, Variables, Messages, and Breakpoints windows, which are opened automatically in the same size and position as they were when the project was last run.



While debugging your program, you can use the Source, Class Browser, Breakpoints, Variables, Call Stack, Threads, and Messages windows to help you isolate problems and resolve them. The Source and Class Browser windows are available at all times; the other views are only available after the debugger has begun to execute the program. Visual Cafe by default

runs your Java program in the Applet Viewer. By running your program in the Applet Viewer, Visual Cafe eliminates the need for you to compile the `.class` file, open a Java-compatible Web browser, write the corresponding HTML code and load it into the Web browser, and reload the applet over and over again.

The Debugging views consist of the following windows.

Overview of the Breakpoints window

The Breakpoints window is used to display a list of breakpoints currently set in the source code. Breakpoints are flags you can insert into the code at specific points that cause the program execution to pause. When the Debugger goes through the instructions, it stops whenever it encounters a breakpoint. You can check on the value of the data and other program conditions.

The Breakpoints view allows you to view a list of all breakpoints in the currently active project. It lists source file and line number of the breakpoint. The current breakpoint is highlighted in the highlight color. The check column can be used to enable and disable listed breakpoints. Double-clicking on a breakpoint in the Breakpoint window goes to the source file and displays the selected breakpoint.

The Breakpoints window is accessible through the Breakpoints menu item in the Window menu.

Overview of the Variables window

Your program's variables are displayed in the Variables window, which displays local variables, global variables, and objects that are local to the current method. You can examine objects and array data elements as well as simple data types.

When you pause the execution of your program, you can modify the value of a variable and then continue execution with the new value in place.

You can modify the value of a variable from either the Variables window or the Watch window.

The Variables window is accessible through the Variables menu item in the Window menu.

Overview of the Watch window

The Watch window lets you specify variables and expressions that you want to watch continuously while debugging your program. In this window, you can examine the contents of a class variable. To watch all the variables accessible to a method, simply drag the method from the Call Stack window into the Watch window. You can examine objects and array data elements as well as simple data types.

You can add a variable name or an expression to the Watch window by clicking in a cell and typing the name or expression. Variables can be modified by double-clicking in the Value cell and entering a new value. After modifying a variable, you can continue debugging from the current line without having to stop and restart the debug session.

The Watch window is accessible through the Watch menu item in the Window menu.

Overview of the Threads window

The Threads window displays the threads that your program has created. A thread is a process that is controlled by your program. It is possible to

create programs that have more than one process (thread), running at the same time.

Note: If you are using multiple threads in your program, each thread has its own call chain. To see the call chain for an individual thread, drag the thread from the Threads window and drop it on the Call Stack window.

The Threads window is accessible through the Threads menu item in the Window menu.

Overview of the Call Stack window

The Call Stack window lists the method calls a program has made since it began running. This list is known as the call stack. The methods in the call stack are displayed in reverse order from the last (most recent) call on top to the first (initial) call on the bottom. Each entry lists the name of the method followed by the name of the class which contains the method.

The Call Stack window is accessible through the Call Stack menu item in the Window menu.

Overview of the Messages window

This window displays all messages from Visual Cafe. For example, if an error is encountered during parsing, the error message displays in this window. You can customize what messages the Message window does or doesn't display. To customize the output to the Messages window, use the Options menu item in the Project Menu. For example, one of the options in the Options menu is the ability to turn off the Java compiler warnings.

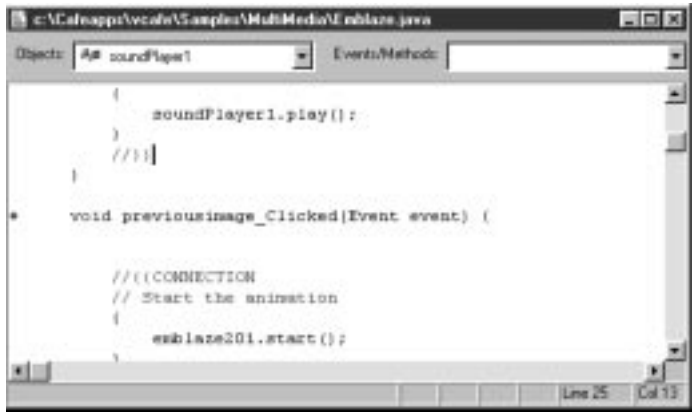
Another function of the Messages window is to notify you of any run-time exceptions, as well as act as a standard output display device. You can redirect textual information directly to the Messages window as you are executing and debugging your Java program.

Double-click on any error message to open the file in the Source window with focus on the selected error.

The Messages window is accessible through the Messages menu item in the Window menu.

Overview of the Source window

The Source window is the primary debugging view: it is where you add, edit, and view code. When debugging an application, Visual Cafe automatically opens the Source window containing the current line of code.



Running the Debugger

When you run your program in the Debugger, you hand control over to the program and interact with it as a user, but you can still debug it. When

you run your program outside the Debugger, you are executing your program as a finished program and you cannot break into debug mode.

In Visual Cafe, you can choose to run your program in one of four ways. You can run your program:

- until it terminates normally or reaches a breakpoint or an exception.
- until execution reaches the cursor location in the Source window.
- so it pauses as it is about to execute the first line.
- from its current debug point until it terminates, ignoring any breakpoints that are set.

Starting a debugging session

You launch the Debugger within Visual Cafe. Your project must be open in the Project window, so the Debugger can have access to symbolic debugger information.

To start a debugging session:

1. Open the project that you want to debug.
2. From the Project menu, choose Run in Debugger (or press F5).

Scrolling in the Source Editor

You can scroll through your code in the Source Editor and always have immediate control over your program. You can configure the display of the source code editor to assist you with the following tasks.

To scroll to a specific line in your code:

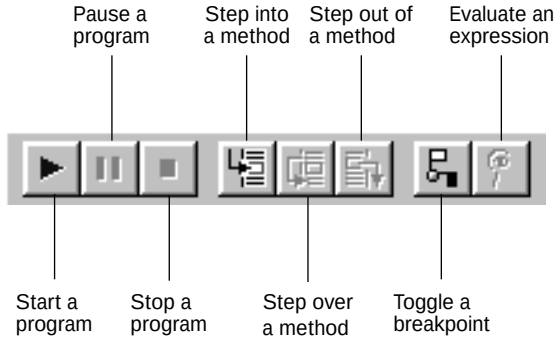
1. From the Object menu, choose Edit Source to open the source code for your project.
2. From the Debugger's Search menu, choose Go To Line (CONTROL-G).
3. When the dialog box is displayed, type the number of the line to which you want to go.
4. Click OK.

To print the file displayed in the Source window, choose Print from the Debugger's File menu.

To edit the file in the Source window, choose Edit from the Source menu. Visual Cafe moves that project to the foreground and opens an Editor window for the file.

To open a different source file in the Source window pane, click that source file in the Project window, then choose Object, then Edit Source.

Using the Debug toolbar



The Debug toolbar provides easy access to the commands that you use most often when debugging your program. From the Debug toolbar, you can do the following:

- Start a program
- Pause a program
- Stop a program
- Step into a method
- Step over a method
- Step out of a method
- Toggle a breakpoint
- Evaluate an expression

For more information about using these commands, see “Stepping through code” on page 8-8.

Stepping through code

The Debugger uses several commands to control execution. To make it easier to debug applications, you can invoke some of these commands by

choosing commands from the Debug menu, or using CONTROL-keyboard equivalents.

To step through code:

Do either of the following:

- Use the commands in the Debugger's Debug menu.
- Use the Debug Toolbar

Pausing a program

In Visual Cafe, you can pause a running program and switch to debug mode. The effect on your program is the same as if a breakpoint is hit.

When an unhandled exception is encountered, the program pauses automatically at the line where the error occurred.

Stopping a program

You can completely stop a program (as opposed to pausing temporarily). Stopping a program terminates it completely in order to work further on it. Execution of a stopped program cannot be continued, and must be restarted from the beginning.

To stop debugging a program:

From the Debug menu, choose Stop Debugging.

Stepping into a method

After your code has hit a breakpoint, you can choose to single step by stepping into any method call contained in the next line to be executed. This causes the single step to land at the first line of the called method.

Using Debug > Step Into

If you are paused in Debug Mode, choose Step Into from the Debug menu to start debugging and pause at the first line.

When used on an applet, this command takes you to the applet `void init( )` method if one is implemented.

If the next line to be executed does not contain a method call, Step Into performs a single step of the line.

Stepping over a method

You can step over a method call contained in your code after the Debugger has hit a breakpoint. This causes the single step to execute the called method in its entirety and land on the next line of code in your source.

Using Debug Step Over

If the next line to be executed does not contain a method call, Step Over performs a single step of the line.

Stepping out of a method

If you hit a breakpoint in a method and want to execute the rest of that method returning to the caller, you can choose to step out of the currently executing method.

Toggling a breakpoint

Toggling a breakpoint is a quick way to reverse the state of a breakpoint on any line of source code in the Source window.

To toggle a breakpoint in the Source window:

1. Click on the line where you want to toggle the breakpoint.
2. Click on (Toggle Breakpoint) the Breakpoint to turn it off in the Debug Toolbar.

   If the breakpoint is set, a red diamond appears in the left margin next to the line of code; if it is reset, the red diamond is cleared.

Watching a variable

You can watch a variable or expression while debugging by adding the variable name to the Watch window.

There are two ways to do this: directly from the Watch window, or from the Variables window.

Caution: Because the program is paused, do not enter a watch expression that relies on calls to other components.

To watch a variable from the Watch menu:

1. From the Window menu, choose Watch to open the Watch window.
2. Type a variable name or an expression directly into the Watch field of the Watch window.

To watch a variable from the Variables window:

1. Right-click on the variable you want to watch.
2. Choose Watch Variable from the context menu.

   The Watch window evaluates and immediately displays the value of the variable or expression.

To move through a block of code quickly:

1. Select a line.
2. From the Debug menu, choose Continue to Cursor.

   The Debugger goes as soon as the command is selected. This command has the same effect as setting a temporary breakpoint at the selected line; that is, it starts execution of your code and stops at the selected line.

   To jump ahead to a selected line without executing any intervening code, chose Skip To Here from the Debug menu. This allows you to skip over code that you know contains bugs but that is not crucial to the rest of the code's operation.

   Note: Use this feature with caution, especially when debugging optimized code.

Running to the first line

Visual Cafe lets you run your program and pause just before the first line executes. This puts you in debug mode before anything has been done to your program's state.

To run to the first line of the program:

From the Debug menu, choose Step Into.

Running the program to the end

You can run your program from its current point until the end, ignoring any breakpoints that are set. This allows you to check for any suspected exception conditions.

To run to the end of the program:

From the Debug menu, choose Continue to End.

If any kind of exception occurs, the program will pause at the point of the violation, unless otherwise specified. You can specify options for handling exceptions in the Debugger Tab of the Project Options window, then choose Category, then Exceptions.

Running to the cursor location

You can run your program until execution reaches the cursor location, after which the program pauses. This is a handy way to continue from a breakpoint to a line you are inspecting in the Source window. You can also run to a cursor location without having to stop at all the breakpoints you may have set.

To run to the cursor location:

From the Debug menu, choose Continue to Cursor.

If the selected line does not get executed before the end of the program, your program won't pause.

Resuming a program

You can resume execution when your program is paused because of a breakpoint, an exception, or because you manually paused execution. When you resume your program, execution is continued from the current location.

To resume debugging a program:

From the Debug menu, choose Continue.

Restarting a program

Restarting a program is when you start execution over again from the beginning. This is different from continuing a program where you restart execution after a breakpoint.

To restart debugging a program from the beginning:

From the Debug menu, choose Restart.

When Restart is selected while a program is executing, it does a Stop > Run in Debugger. Otherwise, it does a Stop, then Step Into.

Handling exceptions

The Java language uses exceptions as a means to process errors in its programs. An exception is an event that occurs during the execution of your program that interferes with, disrupts, or stops the normal flow of instructions.

Throwing exceptions

Many types of errors force the Java run time system to throw exceptions—from simple programming errors to a hard disk crash. When such an error occurs within a Java method, the method creates an exception component and passes it to the Java run time system. This exception component contains important information about the exception, including its type and the state of the program when the error occurred. The run time system tries to find a piece of code to handle the error. This process of creating an exception component and passing it to the Java run time system is called throwing an exception.

Catching exceptions

After a method throws an exception, the Java run time system finds a way to handle the exception. One set of possible tools to handle exceptions is the set of methods in the call stack of the method where the error occurred. The Java run time system scans backwards through the call stack, starting in the method where the error occurred, until the Java run time system locates a method that has a suitable exception handler. An exception handler is suitable if the type of the exception thrown is the same as the type of exception handled by the exception handler. The exception moves up through the call stack until a suitable handler is found and one of the calling methods handles the exception. This is called catching the exception. If the runtime system searches all of the methods on the call stack without encountering a suitable exception handler, the runtime system as well as the Java program terminate.

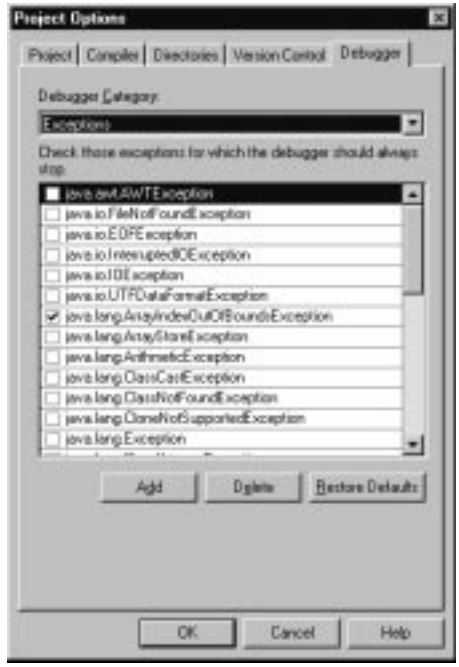
Setting exceptions in Visual Cafe

You can view and set the exceptions options by choosing Exceptions from the Debug options drop-down list in the Project Options dialog box. To stop whenever a particular exception occurs, check the box to the left of the exception name. When this options is set, the program stops whenever the exception is encountered regardless of whether your program handles the error or not. The default behavior for exceptions is for the Debugger only to stop if a particular exception is not handled in the source code.

When running your program, you can choose to have all exceptions break into the Debugger, or to have only unhandled exceptions break into the Debugger.

To prepare to handle exceptions:

1. From the Project menu, choose Options.
2. In the dialog box that appears, click the Debugger tab.



3. Choose how you want to handle exceptions, and add exceptions to the list.

To make all exceptions break into the Debugger:

1. Select the exception in the list.
2. Click in the Action column.
3. Click the down-arrow button and choose Stop Always.

To make only unhandled exceptions break into the Debugger:

Select the exception in the list.

You check a checkbox next to an item that you want ALWAYS to pause the Debugger. The default behavior is for unchecked items to pause the Debugger if they are not handled in source code.

The Restore Defaults button restores the defaults for all exceptions, thereby disabling the stopping at exceptions (all checkmarks are deleted).

Changing source code

If there are syntax errors in your source code, Visual Cafe flags them in the Messages window after a compile. You can easily navigate to each error directly from the Messages window.

To navigate to each error from the Messages window:

1. From the Window menu, choose Messages to bring the Messages window to the front.
2. Double-click on any error message to go to that error.

The file containing the error opens at the offending line within a Source window. Once the file opens, you can work on your source code.

Using the Source window

The Source window is the primary debugging view, it is where you see your code at its current point of execution. When you debug your program, Visual Cafe automatically opens the Source window containing the current line of code, if possible. Sometimes Visual Cafe can't open the Source window because you may not have all the source, for example.

When you are in debug mode, the Source window view provides extra functionality allowing you to manipulate breakpoints and specify variables to watch in the Watch window.

Working with breakpoints

Watching an entire program execute from beginning to end helps you understand the program flow. However, it is more likely that there are only specific parts of the program which you want to observe. It makes sense to step through the few specific lines of code which have behavior you want to observe rather than the entire program. In these cases you can use breakpoints.

A breakpoint is a flag (really a little piece of code) placed in the source code that tells the Debugger to pause execution of the program. This tiny piece of code is invisible to you, but it is represented as a diamond. Breakpoints allow you to examine your program line-by-line as your program executes in the debugger. The compiler recognizes the breakpoint and treats it as a part of your program. Breakpoints are only important to the compiler when you are testing and debugging your program. Breakpoints are ignored when you compile your program into a finished `.class` file.

You can set breakpoints anywhere in your code in order to stop execution at a particular line and regain control after starting your program. When your program breaks on a breakpoint you can examine variable values, single step your program, or examine the state of your program in any way you want.

In Visual Cafe, you can set a breakpoint on:

- The current line in the Source window
- A specific line number
- A method name
- The condition of a variable or expression

Tip: The fastest way to set a breakpoint is to position the cursor on the line of code where you want the breakpoint to appear, then press the F9 key.
Managing breakpoints

Visual Cafe provides many ways to manage breakpoints, allowing you to stop execution of a program at a specific location or on a predetermined condition, such as a variable being assigned a particular value.

Breakpoints are manipulated from within the Breakpoints window, which shows all breakpoints currently defined in your program. You can view or

modify the different parameters of any breakpoint and enable or disable the currently defined breakpoints.

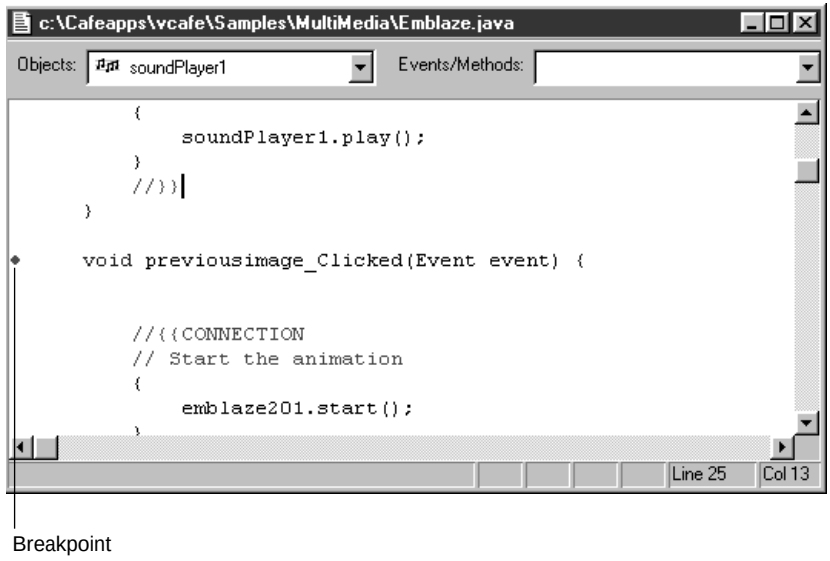
Breakpoints are saved with the source and are made visible each time you open the project. The state of each breakpoint—enabled or disabled—is also saved with the project.

Currently, Disable Breakpoints are represented by hollow diamonds in the Source window. Use the Breakpoint window to evaluate the state of breakpoints in your program.

Setting a breakpoint on a line number

You can have your program break every time a specific line within your current source file is about to be executed. You set breakpoints in the Source Code pane of the Class Browser, or in a Source window.

The diamond indicators that appear to the left of each executable statement indicate the places in the code where you have set breakpoints, as shown in the following figure.



Setting simple breakpoints

You can set a simple breakpoint on a specific line within your current source file.

To set a simple breakpoint:

1. Do either of the following:
   - Click a statement marker circle.
   - Click in the line to select it, then choose Set Breakpoint from the Source menu.

   The circle changes from light blue to red to indicate that a breakpoint has been set.
2. From the Source menu, choose Set Conditional Breakpoint.
3. In the Break at panel, click on Line Number.
4. Enter a line number in the text box.
5. Click OK.

   The breakpoint is added to the list in the Breakpoints window.

When code is running, the Debugger stops just before executing the first statement where a breakpoint has been set. A set breakpoint is indicated by a red circle to the left of the statement. The current statement arrow points at that statement.

Setting a breakpoint on a method name

You can have your program break every time a specific method is executed by setting a breakpoint on a method name.

To set a breakpoint on a method name:

1. From the Source menu, choose Set Conditional Breakpoint.
2. In the Break at panel, click on Method Name.
3. Enter a method name in the text box.
4. Click OK.

   The breakpoint is added to the list in the Breakpoints window.

Setting a breakpoint on a variable or expression

You can have your program break whenever a variable reaches a specific value or an expression of your choice becomes true.

A breakpoint symbol in the left margin of the Source window indicates a breakpoint on the adjacent line.

To set a breakpoint on a variable or expression:

1. From the Source menu, choose Set Conditional Breakpoint.
2. In the Break panel, click on When Expression Is True.
3. Enter a Boolean expression into the text box.
4. Click OK.

   The breakpoint is added to the list in the Breakpoints window.

Setting a conditional breakpoint

A conditional breakpoint pauses the execution of your program when a specified condition is met. In Visual Cafe, when you set a conditional breakpoint, it is added to the breakpoint list in the Breakpoints window along with the condition you specify.

To set a conditional breakpoint:

1. In the Source Code window, click on the line where the breakpoint should occur.
2. From the Source menu, choose Set Conditional Breakpoint.
3. Click When Expression Is true.
4. Type your breakpoint condition into the text box. This condition must be an expression that evaluates to true. When the expression is true, the breakpoint occurs.
5. Click Add.

The breakpoint is added to the list in the Breakpoints window.

Modifying a conditional breakpoint

You modify a conditional breakpoint by changing the conditions of the breakpoint in the Breakpoints window.

To modify a conditional breakpoint:

1. Choose Breakpoints from the Window menu to display the Breakpoints window.
2. Click on the Condition field of the breakpoint you want to change.
3. Type the new condition into the Condition field.
4. Press ENTER to save the change.

Clearing breakpoints

Clearing a breakpoint deletes it from the set of breakpoints you have defined. This is distinct from disabling a breakpoint which retains it in your breakpoint list in a temporarily disabled state.

You clear a breakpoint either at a source line (in the Source window or from the Class Browser, or from the list maintained in the Breakpoints window.

To clear a breakpoint at a line of source code:

1. Select the breakpoint by clicking in the line where your breakpoint is set (the breakpoint is denoted with a stop symbol in the left margin).
2. From the Source menu, choose Clear Breakpoint.

To clear a breakpoint from the Breakpoints window:

Select one or more breakpoints in the list and press DELETE.

When the breakpoint is cleared, the stop symbol is removed from the left margin of the line of source code, and the item is removed from the Breakpoint window.

Enabling or disabling a breakpoint

The checkbox preceding each breakpoint in the Breakpoints window indicates whether or not that breakpoint is enabled. When checked, the breakpoint is enabled.

You can enable or disable a breakpoint by clicking on the breakpoint's checkbox, or by selecting the breakpoint and pressing the spacebar when the breakpoint is selected.

To enable a breakpoint:

Find the breakpoint in the list and click the empty check box corresponding to that breakpoint. When the box is checked the breakpoint is enabled.

To disable a breakpoint:

Find the breakpoint in the list and click the checked box corresponding to that breakpoint. When the box is empty the breakpoint is disabled.

To toggle a breakpoint using the spacebar:

You can press the SPACEBAR to toggle the state of a selected breakpoint in the Breakpoints window.

Note: Disabling a breakpoint does not delete it from the breakpoint list.

Ignoring all breakpoints

There's an easy way in Visual Cafe to ignore all breakpoints you have set without changing the state of any of the breakpoints in the Breakpoints window. You can do this two ways.

- Running to program end
- Running to the cursor location

To ignore all breakpoints by running to program end:

From the Debug menu, choose Continue to End.

Ignores breakpoints and runs the program to termination.

To ignore all breakpoints by running to the cursor location:

From the Debug menu, choose Continue to cursor.

Ignores breakpoints until execution hits the current line at which point execution pauses.

Viewing the source for a breakpoint

In Visual Cafe, you can select any of the breakpoints listed in the Breakpoints window and view the source code associated with that breakpoint.

To view the source for a breakpoint:

1. From the Window menu, choose Breakpoints to open the Breakpoints window.
2. Click on the breakpoint for which you want to see the source code.
3. From the Breakpoint menu, choose Go to Source.

The focus in the Source window will be set to the specified breakpoint.

Stepping through code when the program is paused

After your program has hit a breakpoint, you can step through your lines of code one at a time using three techniques.

Step Into

When you step into your code, the Debugger executes one line of code including any jumps to other methods until it reaches the next line of code. This method allows you to step through the program, executing every statement completely. If the program is running and you are paused in the program, this command steps to the next source code statement. If the current line is a method call, Step In steps inside the method. If the method is an another source file, that file is opened to the next line of source code.

When you step into a method call, the Debugger jumps to the code for that method, and steps through it as well. If the code for the method is in a different file, the Debugger opens that file, and jumps right to the code for the method.

Stepping into every line of code, for every method called by your program is not usually the best way to debug a program. For example, let's say your program calls the println method. We know the println method works, because it is part of the Java language. Stepping into method calls like println throughout a program can quickly result in a chaos of open windows. A more useful approach to debugging, is to step into some parts of the code and to step over other parts. You can avoid stepping through known working code by stepping over method calls. However, if you stepped in where you should have stepped over, you could step out.

Step Over

If the current statement is a method call, the program executes to the next statement following the call. Step Over executes the program to the next

statement (unless a breakpoint or an exception is encountered before execution reaches that point). This function is handy when you are sure that a method is not causing problems.

Step Out

This command executes to the current method's return address, unless a breakpoint or an exception is encountered before execution reaches that point.

Using the Variables window

The Variables window shows the variables that are active in the current context. A context is the particular portion of your program on which Visual Cafe is focusing. This window is useful for examining errors that occur when your program passes parameters to methods. You can modify these variables directly in this window. When you pause the execution of your program, you can modify the value of a variable and then continue execution with the new value in place.

Your program's variables are displayed in the Variables window which displays local variables, global variables, and objects that are local to the current method. You can examine objects and array data elements as well as simple data types and expressions.

You can examine any of the variables in your program while in debug mode.You can modify the value of a variable from either the Variables window or the Watch window.

Viewing the value of a variable

You can view a variable by selecting its name from the Variables window.

To view the value of a variable:

1. From the Window menu, choose Variables.

   The Variables window opens, displaying all the variables in the current context of the program.
2. Click on the variable you want to view.
3. To expand an object to see its contents, click the plus sign to the left of the object you want to expand.

Viewing type information for a variable

You can view a variable's type from the Variables window.

To view type information for a variable:

1. From the Variables menu, choose Watch.

   The Variables window opens, displaying all the variables in the current context of the program.
2. Click on the variable whose type you want to view.

   The variable's type is shown in the Type column.

Modifying a variable in the Variables window

You can modify a variable by selecting its name from the Variable window and typing a value into the value column. You can modify a variable in the Variables window only when your program is paused in the debugger.

To modify a variable in the Variables window:

1. From the Window menu, choose Variables > Watch.

   The Variables window opens, displaying all the variables in the current context of the program.
2. Click on the variable you want to change.
3. Click in the Value column.
4. Type the new value in the Value box.
5. Press ENTER to save the change.

To modify the value of an array or any structured type, edit the individual array's fields or elements. You cannot edit an entire array or structure at once.

Using expressions in the Watch window

In Visual Cafe, you can specify variables and expressions that you want to watch at breaks while debugging and executing your program. The variables that you elect to watch are displayed in the Watch window, which is only available when you pause a running program.

Caution: Since the program is paused, do not enter a watch expressions that rely on calls to other components.

Adding a variable to the Watch window

You can watch a variable or expression while debugging by adding the variable name to the Watch window. There are two ways to do this: directly from the Watch window, or from the Variables window.

To add a variable from the Watch window:

1. From the Window menu, choose Watch to display the Watch window.

   All the variables and expressions you have chosen to watch are displayed in the Watch window.

   Start typing a variable name or an expression directly into the Watch field of the Watch window. As you begin to type, the window changes to edit mode for the column of the selected row.
2. Press ENTER to save or ESCAPE to leave the item unchanged.

To add a variable from the Variables window:

1. Right-click on the variable you want to watch.
2. Choose Watch, then Variable from the context menu.

   The Watch window evaluates and immediately displays the value of the variable or expression. You can also modify the value of a variable using the Watch window.

Modifying a variable or expression in the Watch window

You can change the variable name or expression you elected to watch at run time directly from the Watch window. The Watch window can also be used for modifying variables in place while your program is in debug mode.

To locate the variable you want to modify:

1. From the Window menu, choose Watch.
2. Click on the variable or expression you want to change.

   All the variables and expressions you have chosen to watch are displayed in the Watch window.

To change the value of a variable:

1. Click in the Value field.
2. Type the new value in the Value box.

To change the variable or expression to watch:

1. Click in the Watch field.
2. Edit the variable or expression.
3. Press ENTER to save the change.

To modify the value of an array or any structured type, edit the individual array's fields or elements. You cannot edit an entire array or structure at once.

Caution: Since the program is paused, do not enter a watch expressions that rely on calls to other components.

Deleting a variable or expression from the Watch window

When you've finished watching a run-time variable or expression, you can delete it directly from the Watch window.

To delete a variable or expression:

1. From the Window menu, choose Watch.

   All the variables and expressions you have chosen to watch are displayed in the Watch window.
2. Click on the variable or expression you want to delete from the list.
3. Press DELETE.

   The selected entry is deleted from the Watch window. You can also delete multiple entries.

Using the Call Stack window

The Call Stack window displays all active calls. A call is a reference made from one class to methods in another class. The Call Stack window shows all the method calls that have started, but not have completed execution.

The call chain is the sequence of functions that were called to get to the current function. You can access the functions in the call chain through the Call Stack window.

When debugging in Visual Cafe, you can view the stack of methods in your application that have started but not completed. These pending methods are displayed in the Call Stack window. The currently executing method appears at the top of the stack and older function calls below that. You can also see the parameter types and values for each method on the call stack.

The active method is indicated by a black arrow next to it.

Viewing parameters for a method on the Call Stack window

You can view the parameters passed to a method on the call stack when that method was called. Visual Cafe allows the display of both values and types for each parameter passed.

To open the Call Stack window:

From the Window menu, choose Call Stack.

To view parameter types:

From the Calls menu, choose View Parameter Types.

This action toggles the display of parameter types in the Procedure column of the Call Stack window.

To view parameter values:

1. From the Calls menu, choose View Parameter Values.

   This action toggles the display of parameter values in the Procedure column of the Call Stack window.
2. Enter the number of items in the array and the base index of the array.
3. Click OK.

Viewing variables for a method on the call stack

You can view the values of variables for any method on the call stack. Only the variables in scope at the time the method entered the stack can be viewed.

To view the variables on the call stack:

1. From the Window menu, choose Call Stack to open the Call Stack window.
2. Click on the Method whose variables you want to view.
3. From the Calls menu, choose Go to Variables.

   The Variables window lists the variables in the selected call.

Viewing source for a method on the call stack

Visual Cafe allows you to choose any method entered on the call stack and view the source code for that method.

To view the source for a method on the call stack:

1. From the Window menu, choose Call Stack to open the Call Stack window.
2. Click on the Method whose code you want to view.
3. From the Calls menu, choose Go to Source.

   The Source window for the selected call opens.

Ending a debugging session

You can end a debugging session at any time.

To end a debugging session:

Do either of the following:

- Quit the application by closing the project or Visual Cafe.
- Choose Stop from the Debugger's Debug menu.

Debugging threads

Java is a multithreaded language, which means that Java allows for more than one sequence of execution at a time. You might want to use threads to allow your Java program to talk to more than one client across networks.

A thread is a process in a program that has a beginning and an end. In applications, the main method is responsible for indicating the beginning

and end of the program. In applets, the Web browser uses various methods to control the program flow.

However, programs are not limited to performing a single process. Java programs can use threads to perform multiple processes simultaneously. This is called multithreading. For example, suppose you are using a text editor to type a letter and you want to save your changes before you continue. If the text editor program is single-threaded, when you save the file, the rest of the program must wait until the file is completely written to the hard disk.

In a multithreaded application, the process that saves the file can be an independent thread with its own beginning and end. When you save the file, the file-saving thread starts and runs in parallel with the application's other processes. You can continue to type your letter as a copy of the file is written to disk in the background. Multithreaded programs run faster and are more convenient than single threaded applications.

Visual Cafe uses the Threads window to help you keep track of all the threads that you put into your program. The threads window lists all the known threads at any breakpoint. You must pause the program in order to use the Threads window. You can also use the Threads window to control the focus of the Variables window so that the Variables window displays only variables that are associated with a particular thread and discard others.

When debugging a multithreaded program in Visual Cafe, you can work with a single thread in exclusion of others. All the extant threads your program has created display in the Threads window, along with the state of each thread.

You can switch between threads to debug or view the source for a selected thread from the Threads window. You can also update the Call Stack window with a single thread's call chain and update the Variables window to show only the variables within the current thread.

Using the Threads window

The Threads window presents at a glance all the currently existent threads that your program has created, together with their states. The Threads window provides no benefits when debugging a single-thread application. It is only when debugging a multi-threaded application that the Threads window provides indispensable services. This window lets you:

- easily switch between threads to debug

- update the Source window with a thread's current location
- update the Call Stack window with a thread's call chair

The current are listed one per row. The currently selected thread is highlighted. You can change the selection by clicking on a different row, or by using the arrow keys. The primary thread is identified by a bold arrow in the left margin. This thread receives user input and is automatically created by the operating system when a process (an instance of an application) is created. The active thread—the one from which the debugger regained control—is identified by a normal arrow in the left margin. The columns of the Threads window have the following significance:

Column	Meaning
ID	Thread ID number. Thread IDs are unique within a process.
Status	Status. The Thread Status is the state of the thread inside the Virtual Machine. If you suspend a thread and the status reads "at breakpoint", the thread is at a breakpoint in the VM. Possible values of this field are: Frz – the thread is "frozen" (suspended) Thw – the thread is "thawed" (resumed or not suspended) Suspended -- the thread is suspended and waiting to be resumed. At Breakpoint -- the thread is waiting to be executed. Running -- the current thread running in the Virtual Machine. Condition Waiting -- the thread has been locked by another thread. Zombie -- a thread has been created and is waiting to be run for the first time.

The Threads view allows the user to see a list of threads running, and suspend and resume threads. Comprised of two columns, the threads view

lists the threads and addresses of all spawned threads. The current thread is shown by selection. Double-clicking on a thread sets the context to that thread. The stack crawl window shows that thread.

Debugging a single thread

If you want to focus your debugging efforts on a specific thread to eliminate the behavior of others, you can set the focus to that thread and work on it alone.

To debug a single thread:

1. From the Window menu, choose Threads to open the Threads window.
2. Click the thread you want to work on.
3. From the Threads menu, choose Set Focus.

   The focus of the Debugger is set to the selected thread.
4. Continue debugging that thread in the Call Stack window, Variables window, or Source windows.

Suspending a thread

If you're working on a multithreaded program, you can suspend any specific thread if you suspect it of causing unwanted side effects while your program is running or while you are debugging.

To suspend a thread:

1. From the Window menu, choose Threads to open the Threads window.
2. Click the thread you want to suspend.
3. Choose the Thread, then Suspend.

   The selected thread is suspended.

Resuming a suspended thread

If you've suspended any threads to temporarily eliminate their behavior from your program, you can resume any one you choose from the list displayed in the Threads window.

To resume a suspended thread:

1. From the Window menu, choose Threads to open the Threads window.
2. Click the thread you want to resume.
3. From the Threads menu, choose Resume.

   The selected thread is resumed.

Suspending other threads

If you want to focus your debugging attention on a single thread, Visual Cafe allows you to suspend other threads to narrow down the behavior of your program.

To suspend other threads:

1. From the Window menu, choose Threads to open the Threads window.
2. Click the thread you want to want to work on.

Caution: You're choosing all threads other than this one to be suspended.

---

3. From the Threads menu, choose Suspend Others.

   All other threads in the Debugger except the one that is selected are suspended.

Resuming other suspended threads

If you've chosen to suspend any threads in a multithreaded program, you can resume all other threads at any time.

To resume other suspended threads:

1. From the Window menu, choose Thread to open the Threads window.
2. Click the thread you do not want to resume.

Caution: You're allowing threads other than this one to resume.

---

3. From the Threads menu, choose Resume Others.

   All other threads in the Debugger except the one that is selected are resumed.

Viewing the source code for a selected thread

If you want to look at the source code for a specific thread, you can do so from the Threads window.

To view the source code for a selected thread:

1. From the Window menu, choose Thread to open the Threads window.
2. Click the thread you want to focus on.
3. From the Threads menu, choose Set Focus.

   Set Focus updates the Source window.
4. Display the Source window.

   The thread's source code is visible in the Source window.

Viewing the active variables in a thread

You can view the values of variables for any thread currently executing.

To view the active variables in a thread:

1. From the Window menu, choose Threads to open the Threads window.
2. Click the thread you want to focus on.
3. From the Threads menu, choose Set Focus.

   Set Focus updates the Variables window.
4. From the Window menu, choose Variables.

   The Variables window opens displaying the chosen thread's variables.

Viewing the call stack for a thread

If you're working on a specific thread and want to see the call stack for that thread alone, Visual Cafe allows you to do so.

To view the call stack for a thread:

1. From the Window menu, choose Threads to open the Threads window.
2. Click the thread whose call stack you want to view.
3. From the Threads menu, choose Set Focus.

   Set Focus updates the Call Stack window.
4. From the Window menu, choose Call Stack.

The Call Stack window opens on the thread you selected.

5 Click on the Method whose code you want to view.

Debugging remotely

One of the unique features of Visual Cafe is running a program on one machine and debugging it remotely on another.

Setting up for remote debugging

Before beginning a remote debugging session, you must properly configure the local and remote machines.

To configure the local and remote machines:

1 Visual Cafe or Visual Cafe Pro must be installed and running on both machines.

2 The TCP/IP networking protocol must be installed on both computers and running.

3 Identical copies of the project, including all source files to be debugged, must be on both machines.

Starting remote applets or application debugging

You can debug an applet or application remotely on another machine. You cannot debug native applications or DLLs remotely.

Note: The class files you are debugging reside on the remote computer and not on the computer where you are running the debugger. So, to have a good debug session, you need to be sure that the remote computer has the debug build of the classes you need, and so on. Also, the remote virtual machine needs to have the class path set in the environment before running `caferemote.exe`. Make sure the class path is set correctly in `autoexec.bat` for Windows 95 or in the Control Panel for Windows NT; `caferemote.exe` can also read `sc.ini`.

To debug an applet or application remotely:

1 Open a console session (DOS window) on the remote machine by moving to the directory containing the class and HTML files of the program you intend to debug and executing `caferemote`.

Caferemote displays some version information, and then an agent password and the machine's IP address.

2 On the machine that is running the Debugger, load the project corresponding to the class and HTML files you are debugging on the remote host machine.

3 From the Project menu, choose Options.

4 If you are remotely debugging an applet, from the Project tab, specify an HTML file, including the full path.

5 From the Debugger tab, choose General from the Category list.

6 Check Enable Remote Debugging and enter the Host Address and Agent Password as reported by caferemote.

Field	Description
Host Address	The IP address of the remote host machine.
Agent Password	The remote machine's password, which is returned by caferemote.exe run on the remote computer.

7 Click OK.

Remote debugging is now enabled.

Ending remote applets or application debugging

On the remote machine, select the Caferemote console window and press "CONTROL-C". This terminates the debug session currently in process.

To continue, you need to re-execute caferemote in order to get a new agent password.

Using Debugger-specific menus

This section describes the menus and their functionality while using the debugger.

Overview of the Project menu

The following commands are available from the Visual Cafe Project menu.

The **Execute** command compiles and runs the current project with no debugging. Applets are run in the Symantec Applet viewer by default. You set the default viewer in the Project tab of the Project Options dialog box.

The **Run in Debugger** command runs the program in debug mode and stops at the first breakpoint.

The **Step Into** command begins running the program by stepping into the first line of source code. When used on an applet, this command takes you the applet `void init( )` method if one is implemented.

The **Build Applet** or **Build Application** command builds the applet or application.

The **Compile Item** command compiles the active source file.

The **Parse All** command parses all files in the project.

The **Add Item** command adds the active file in the Source window to the project.

The **Create Project Template** command displays the Create Template dialog box so you can add a template to the Component Library. Once added, the template is available for the New Project command.

The **Switch Project** command lets you select an open project as the new active main project. The main project is the one that the Project menu commands apply to.

The **Options** command displays the Project Options dialog box where you can define debugger, compiler, project, and directory options.

Overview of the Debug menu

The Debug menu replaces the Project menu when you start debugging. To display the Debug menu, choose Run in Debugger from the Project menu.

The following commands are available from the Visual Cafe Debug menu.

The **Continue** command runs a paused program.

The **Pause** command temporarily stops the execution of a program while it is running and switches to debug mode. To begin the program again at the current location, choose Continue.

When an un-handled exception is encountered, Visual Cafe pauses automatically at the line where the error occurred.

The **Stop** command terminates the program execution.

The **Restart** command restarts the program. Debugging is restarted from the first line.

The **Step Into** command steps into the next line of source code. This command steps to the next source code statement even if it is contained within a method.

The **Step Over** command steps over the current method and stops when the method returns. Executes the program to the next statement, unless a breakpoint or exception is encountered before the next statement. If the current statement contains a method call, the entire method is called before control is returned.

The **Step Out** command executes the current method until it returns to its caller, unless a breakpoint or exception is encountered before execution reaches that point.

The **Continue to Cursor** command continues running a paused program while ignoring any breakpoints prior to the cursor location, then stops at the cursor location. When the cursor is reached, the program pauses and the debugger is invoked. If the selected line does not get executed before the end of the program, the program does not break.

The **Continue to End** command continues running a paused program, ignoring all breakpoints, from the pause point until the normal termination point. If any type of exception occurs, the program breaks at the point where the exception is called.

The **Options** command displays the Project Options dialog box, where you can set debugging options under the Debugger tab.

Overview of the Insert menu

The following options are available from the Visual Cafe Insert menu. This menu allows you to create new project objects and add forms, files, and objects to the project.

The **Form** command displays the Insert Form dialog box so that you can insert a form from the Component Library into the current project.

The **Applet** command inserts an applet if there are no saved applet objects, or opens the Insert Applet dialog where you can select an applet from a list of applet templates.

The **Component** command displays the Insert Component dialog box so you can add one or more objects from the Component Library to the active form or to the project.

The **Class** command opens the Insert Class Wizard, where you can add and define a new class or interface for the current project.

The **Member** command displays the Insert Member dialog box, where you can declare a method to be added to the current class.

The **Group** command adds a group to the current project or group. Groups can only be inserted at the root level of the Component Library window or inside other groups.

The **Files into Project** command (Insert/Remove Files) displays a dialog box where you can select one or more files to be added to the project. You can also remove files from the project with this dialog box.

The **Component into Library** command displays a dialog box where you can add an external component to the Component Library.

Overview of the Breakpoints menu

The Breakpoints menu is available when your applet or application is running. To display the Breakpoints window and this menu, choose Breakpoints from the Window menu.

The following commands are available from the Visual Cafe Breakpoints menu, or from the Breakpoint window's pop-up menu.

The **Clear** or **Clear All** command clears the selected breakpoint or all breakpoints.

The **Enable** or **Enable All** command enables the selected breakpoints or all breakpoints.

The **Disable** or **Disable All** command disables the selected breakpoints or all breakpoints.

The **Go to Source** command opens the Source window with focus on the line of code where the breakpoint is set.

Overview of the Variables menu

To display the Variables window and this menu, choose Variables from the Window menu.

The **Evaluate Expression** command lets you evaluate variables and expressions from a dialog, modify their values, and add an entry to the Watch window.

Overview of the Threads menu

To display the Threads window and this menu, choose Threads from the Window menu.

The following commands are available from the Visual Cafe Threads menu.

The **Suspend** command suspends the execution of the selected thread.

The **Resume** command resumes execution of the selected thread from the previous suspend point.

The **Suspend Others** command suspends all other threads in the debugger except for the selected thread.

The **Resume Others** command resumes all other threads in the debugger except for the currently selected thread.

The **Set Focus** command sets the focus of the debugger to the selected thread. The Call Stack, Variables, and Source windows are updated.

Overview of the Calls menu

To display the Call Stack window and the Calls menu, choose Calls from the Window menu.

The following commands are available from the Visual Cafe Calls menu.

The **View Parameter Values** command toggles the display of parameter values in the Method column of the Call Stack window.

The **View Parameter Types** command toggles the display of parameter types in the Method column.

The **Go to Source** command opens the Source window for the selected call.

The **Go to Variables** command updates and displays the Variables window with a list of variables in the selected method call.

The **Set Focus** command sets the focus of the debugger to the selected method call. This option also updates the Variables to show the active variables in the method and Source windows to show the method call so that you can step in if you want.

Overview of the Source menu

The following options are enabled when the Source window is the current window. These commands affect the debug setting or cursor placement in the Source window.

To display the Source window and this menu, select a file in the Project window, then choose Edit Source from the Object menu, or right-click and choose Edit Source.

The following commands are available from the Visual Cafe Source menu.

The **Evaluate Expression** command lets you evaluate variables and expressions from a dialog box, modify their values, and add an entry to the Watch window.

The **Set Breakpoint** command sets or clears a breakpoint on the current line. The toggling is based on whether or not a breakpoint is set for the current line of source code.

If no breakpoint is set, a breakpoint is set on the current line and a stop icon appears in the left margin next to the current line. If there is a breakpoint on the current line, the breakpoint is removed.

The **Set Conditional Breakpoint** command displays the Conditional Breakpoint dialog box, where you can set a breakpoint that occurs if a particular expression evaluates to true.

The **Indent** or **Unindent** command indents or unindents the selected text block.

The **Uppercase** or **Lowercase** command converts the selected text to upper or lower case.

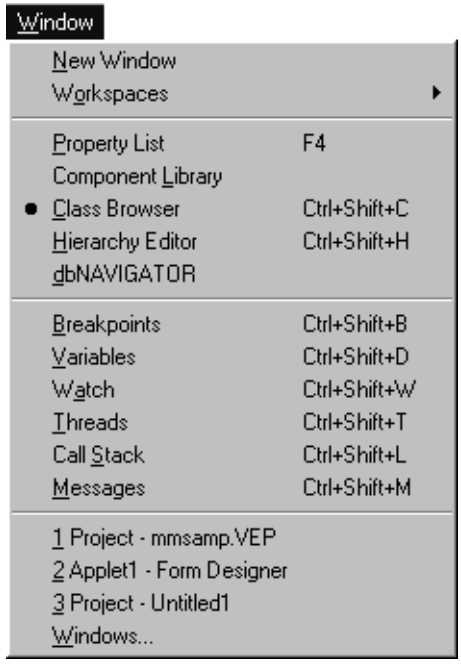
The **Tabs to Spaces** command changes all tab characters in the selected text to spaces. The number of spaces used to replace each tab character

depends on the current edit buffer's Tab Width value in the Format Options dialog box.

The **Spaces to Tabs** command changes spaces in the selected text to tab characters. The number of spaces used to create each tab character depends on the current edit buffer's Tab Width value in the Format Options dialog box.

The **Format Options** command displays the Format Options dialog box, where you can set editing options for the current edit buffers.

Overview of the Window menu



The following options are available from the Visual Cafe Window menu:

The **New Window** command creates a new instance of the active source window. Each new instance of the same window is incremented to indicate the number of open windows.

The **Workspaces** command includes the following commands on its submenu that let you work with your current workspace:

- The <workspace name> command lets you save your current workspace and configure your environment to the new layout.
- The **New** command lets you create a new workspace.
- The **Rename** command lets you rename a workspace.
- The **Delete** command lets you delete a workspace.

You can activate the display of these tools using their corresponding commands:

- Property List
- Component Library
- Class Browser
- Hierarchy Editor

You can activate the display of these debugging windows using their corresponding commands:

- Breakpoints
- Variables
- Watch
- Threads
- Call Stack
- Messages

The **Recently Used Windows** command displays one of the windows that you used recently, you can select the window name from the numbered list on the menu.

The **Windows** command displays a list of recently used windows.

CHAPTER 9

Fine-Tuning Visual Cafe

This chapter describes various ways to enhance and troubleshoot your development experience with Visual Cafe. The first section describes the ways you can configure the Visual Cafe environment to your development preferences. Also included is a section on how to use the Symantec LiveUpdate feature to upgrade your existing version of Visual Cafe.

The second section discusses common problems in programming in the Java language and with Visual Cafe in general.

Setting environment options

Visual Cafe allows you to set options that are global to all of your projects.

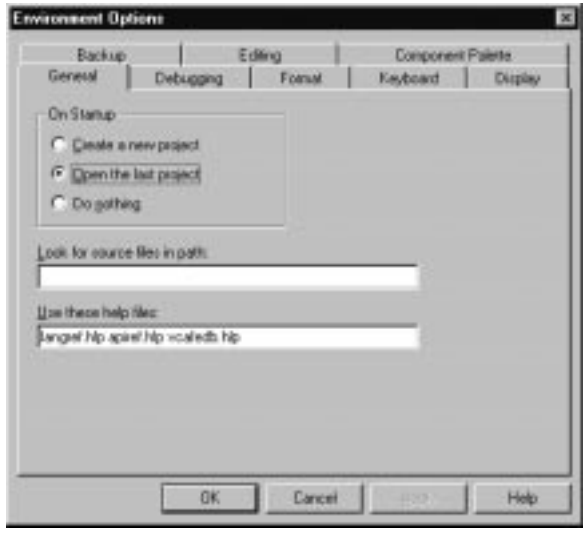
Note: To customize a single project, use the Project Options dialog box.

Setting environment options in the General tab

From the General tab of the Environment Options dialog box, you can specifying the following environment options:

- Defining the Visual Cafe startup mode
- Finding Java source files
- Defining the Help File Set

To open the Environment Options General view, choose Choose Tools, Environment Options, then General tab.



Defining the Visual Cafe startup mode

The startup mode defines what processing you want done when you start a new Visual Cafe session.

To establish the startup mode:

1. From the Tools menu, choose Environment Options, then click the General tab.
2. In the On Startup area, select the appropriate option.

Select…	To specify this…
Create a new project	Open a new project each time Visual Cafe starts.
Open the last project	Open the project that was active the last time Visual Cafe exited; if there is none, a new project is created. (This is the default.)
Do nothing	Specify that no project opens and that you will select an appropriate action after Visual Cafe starts.

Finding Java source files

You can specify the path to locate source files if they are not in the project. This setting is for the entire Visual Cafe environment.

To specify the project path:

Note: You can set the source search path for a project from the Project Options dialog box, and for the entire environment with the `javainc` statement in the `\VisualCafe\Bin\sc.ini` file.

---

1. From the Tools menu, choose Environment Options, then click the General tab.
2. In the Look for source files in path field, type the path to locate source files if they are not in the project.

Defining the Help file set

The Help file set is a group of WinHelp `hlp` and `cnt` files that Visual Cafe uses at runtime.

To define the Help file set:

1. From the Tools menu, choose Environment Options, then click the General tab.
2. In the Help Files field, enter the **hlp** file names.

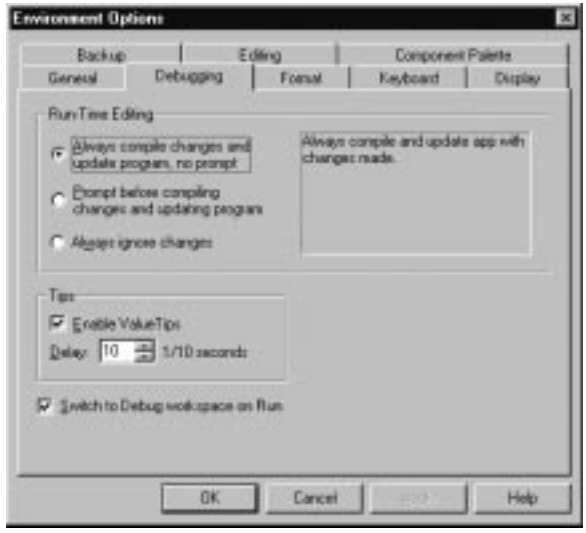
Note: These directories are automatically searched by Visual Cafe: the installation directory, and the release `\bin` and `\help` directories.

Setting debugging options for the environment

There are several Visual Cafe debugger options that apply to the entire environment.

To set environment options for debugging, from the Tools menu, choose Environment Options, then click the Debugging tab. The panel offers the

Run-Time Editing option group, the Tips group, and the Switch to Debug workspace on Run option.



The Run-Time Editing option group

Visual Cafe Professional Development Edition offers incremental debugging (also called run-time editing). While your program is running in the Visual Cafe debugger, you can edit your Java program and immediately see the effects of the code change.

Three options are available:

- Always compile changes and update program, no prompt — Enables run-time editing so that choosing a Save menu item causes new changes to be compiled and reflected in an executing program. If the program is paused, resuming it causes all changes to be applied and saved. (This is the default.)
- Prompt before compiling changes and updating program — Enables run-time editing the same as the first option, except that you are prompted first, so you can choose to stop run-time editing.
- Always ignore changes — Disables the run-time editing feature. (Visual Cafe version 1.0 worked this way.)

The Tips option group

To enable or disable ValueTips at debug time:

1. Select or clear the Enable ValueTips option.
2. If selected, set the delay time before a tip displays.

The Switch to Debug workspace on Run option

Select this option to cause Visual Cafe to go to the workspace named "Debug" when you run in the debugger (from the Project menu, choose Run in Debugger).

Deselect this option if you want to remain in the current workspace. The default is selected.

Specifying text formatting for Visual Cafe windows

You can define a format style for different file types displayed in the Visual Cafe windows. The initial file types that are available are Java and HTML.

To open the Environment Options Format view, from the Tools menu, choose Environment Options, then click the Format tab.

To set format options for files with a certain extension:

1 Choose the file extension you want to customize.

- To create a new file extension entry, click New and enter the extension in the dialog box.
- To remove an extension from the list, choose it then click Delete.

<UNTITLED> is the same type you get after choosing New File from the File menu.

<UNKNOWN> is a file extension that does not have specific format options set for it. This is the type of file that you get when you save a file as an "unknown" type.

2 Set the options you want to apply to files with this extension. To highlight language keywords, choose the language from the Language keywords drop-down list.

Option	Description
Word wrap	Enables word wrap. While typing, lines that extend beyond the right margin are automatically broken at the last word boundary before the margin.
Check delimiters	If you type a right parenthesis, square bracket, or brace, the editor briefly highlights the corresponding left delimiter. If no matching delimiter is found, an error message is displayed in the status bar.
Indent after brace	If the last character typed on a line is a left brace, the next line is automatically indented by an extra tab stop. This option only works if Indent Automatically is selected in the buffer.
Indent automatically	Automatically indent on newline. When you press ENTER, the editor positions the cursor directly below the first character in the previous line.
Change tabs to spaces	Tabs are inserted into the text as an appropriate number of spaces, rather than as tab characters.
Remove trailing spaces	Trailing spaces and tabs are removed from the end of each line when a file is saved.

Option	Description
Enable custom keywords	You can maintain a set of custom keywords that highlight in a specified manner within the edit window. There is one custom keywords list that applies to all file types that have the Enable custom keywords option. Click Edit Custom to specify custom keywords. To set highlighting for custom keywords, click the Display tab.
Indent comments at	If selected, the editor automatically indents the comment to a specified alignment column when you type "//" or "/*" to start a comment. You can specify the alignment column in the adjacent text box.
Tab width	Specifies the number of columns between tab stops (1-16).The default is four character widths. This value may be overridden locally in each buffer.
Right margin	Specifies the column that acts as the right margin for word wrapping. (1-512). This value may be overridden locally in each buffer.

To specify custom keywords:

Custom keywords are recognized in the Source window or pane. They are highlighted in red by default. You can change the display attributes of custom keywords in the Environment Options Display tab. There is one custom keywords list that applies to all file types that have the Enable custom keywords option.

- Click Edit Custom.

  The Custom Keywords dialog box opens so that you can enter the new keywords.

Using the Enter New File Extension dialog box

From the Tools menu, choose Environment Options, then click the Format tab, then click New.

Enter the file extension that you want to add to the format type list.

When the extension is available from the list, you can define a format style for the file type. Two predefined file types are Java and HTML.

Using the Custom Keywords dialog box

From the Tools menu, choose Environment Options, then click the Format tab, then click Edit Custom.

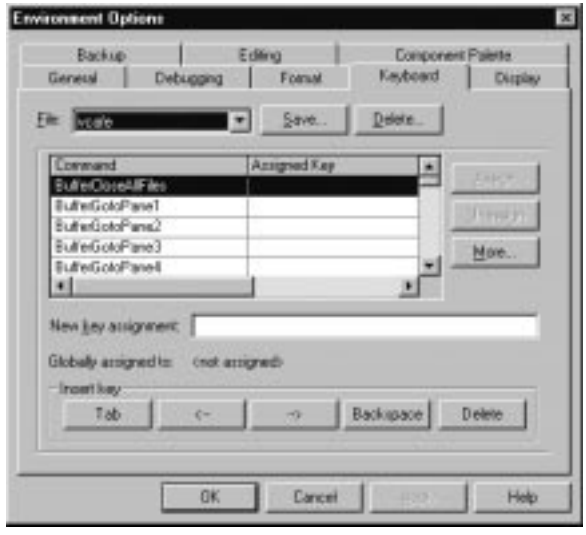
- Use the Custom Keywords dialog box to add keywords to the list of words recognized in the Source window and pane.
- To add a new keyword, type the keyword into the text box and click Add.
- To remove a keyword from the list, click the keyword in the list and click Remove.

Mapping Visual Cafe commands to key sequences

Visual Cafe lets you define a custom keystroke sequence for many Visual Cafe editing operations and macros. Macros are stored in the command list as macro*filename*.

Note: Help on the commands is available in the Macro help. This help is available when you are editing macros. For example, from the Tools menu, choose Macro, then ScriptMaker, then click Edit; in the window that displays, press F1 to get the Macro help contents.

In the Keyboard view of the Environment Options, you can also specify editing options for the Source Editor.



To access the Environment Options Keyboard view:

From the Tools menu, choose Environment Options, then click the Keyboard tab.

To choose a key file to use in the Visual Cafe environment:

Choose a file in the File field.

To delete a key file:

Choose a file in the File field, then click Delete.

To map a command to a key sequence:

1. Choose a key file.
2. From the command list, select a command.

   If an item has more than one key assignment, the item appears multiple times in the list.

   Tip: Click the column header to sort the list by command or key assignment, and by forward or reverse alphabetical order.

   ---

3. To specify a key assignment, click in the New Key Assignment field and specify the key sequence as follows:
   - Press the key sequence.
   - Click a button in the Insert Key area.

     If the value that you enter is already mapped to a command or macro, that command name appears in the Assigned To area.

     Tip: A command can have multiple key assignments. If the command already has a key assignment and you specify another assignment, the command now has two separate assignments.

     ---

4. Assign the key sequence to the command by clicking Assign.

   The assignments are automatically saved in an untitled file.
5. To save the settings to a file, click Save.

   In the dialog box, specify the name of a new or existing **key** file. Saving to a file allows you to reload or distribute the key assignments.

To delete the key assignment for a command:

- Select the command in the Command list and click Unassign.

To copy the command assignment list as text:

- Right-click in the Command list and select Copy All.

You can paste the list into another window, such as the Source window.

To modify what commands are shown:

- Right-click in the Command list and choose Unbound commands, All, Member, Text, or Class.

  Unbound commands toggles the display of commands without key assignments.

  Member, Text, and Class toggle display of commands that start with these words.

To specify key editing options for the Source window and pane:

Click More and select options in the dialog box.

Specifying key editing options for the Source Editor

These option settings apply to all Source windows and panes.

To specify editing options:

1. From the Tools menu, choose Environment Options, then click the Keyboard tab.
2. Click More.

3 Select the options you want:

Option	Description
Virtual cursor	When checked, you can position the cursor anywhere in the file, regardless of the placement of the line endings. When unchecked, you cannot position the cursor past the end of a line. This box is unchecked by default.
Brief menu accelerators	Disables menu keys. After this is selected, any new windows will not have any underscores beneath the top-level menu items.
Brief-compatible selection	If you choose this option, the editor stays in selection mode when you use the arrow keys.
Typing replaces selection	When checked, the source editor follows the Windows standard convention of replacing any selected text with the first character typed or pasted. When unchecked, typing or pasting inserts the text to the left of the current selection. This box is checked by default.
Normal selection for debugging	Enables normal selection of text when in debugging mode. If cleared, you can drag from the source window to the Variables, Watch, and Thread windows while debugging.
Cut and copy line without selection	When checked, you can quickly cut or copy the current line using the standard cut or copy keystrokes (without having to first select the line). When unchecked, you can cut or copy a line by using the keys assigned to the EditorCutLine or EditorCopyLine functions. This box is unchecked by default.

Using the Save Key Bindings dialog box

Use the Save Key Binding dialog box to save a **key** file.

- From the Tools menu, choose Environment Options, then click the Keyboard tab, then click Save.

Customizing the display font and color in Visual Cafe windows

Visual Cafe allows you to set the font, style, and colors for development environment windows, Visual Cafe editors, and window items. You can

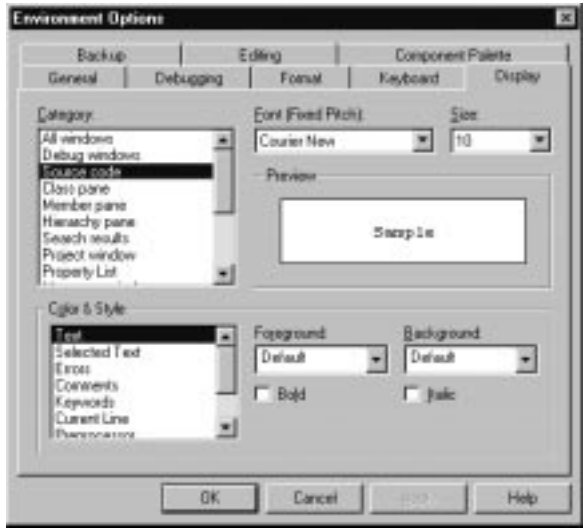
also change the appearance of elements in your source code; for example, comments, keywords, current line, and breakpoints.

Note: When you select All Windows, the Color and Style list displays a set of common components. Common component changes apply to all objects that contain the component.

There is only one Text entry; changing the color and style for it changes the text color for all windows.

To access the Environment Options Display view:

- From the Tools menu, choose Environment Options, then click the Display tab.



To set the font or font size:

1. In the Category list box, select the item whose appearance you want to modify.
2. Choose a value for font or font size.

To set the color and style:

1. In the Category list box, select the item whose appearance you want to modify.

2 Select an item in the Color & Styles list box, then choose a value for Foreground or Background, or select Bold or Italic.

The default setting is the default text color as set in the Windows control panel.

Here are explanations of items in the Color & Style list:

Item	Description
Text	Text other than the type defined below.
Selected text	The text and background of selected text.
Errors	Lines where compiler errors are found.
Comments	Java comments.
Keywords	Java keywords.
Current Line	The line that contains the insert point.
Preprocessor	Java preprocessor directives.
Custom keywords	Special keywords that you define.
Execution Line	During debugging, the current execution line.

Specifying backup and save options

Visual Cafe provides several backup and save options:

- Saving files for recovery purposes
- Automating source file backups
- Setting the Scope of the Undo Command

Saving files for recovery purposes

You can automate the saving of files in temporary locations so you can recover changes if there is a system failure.

To establish file saving:

1 From the Tools menu, choose Environment Options, then click the Backup tab.

2 Select Save Automatically and choose a time interval.

When selected, each modified edit buffer is saved at regular intervals in a temporary file in the `temp` home directory. Under normal circumstances, these temporary files are automatically deleted when the editor exits. If a system crash or power failure occurs, the editor will not exit normally and these temporary files are not deleted. This permits you to recover work that would otherwise be lost. The temporary files created by the autosave function have name that begin with the character ~, followed by a unique number and the extension `.sav`; for example, `~4289352.sav`.

For identification purposes, the autosave function adds a line in the following format at the beginning of each temporary file:

```
; Visual Cafe AUTOSAVE C:\DIR\FILENAME.EXT 08-12-95
 7:35 pm
```

This line contains the complete path name of the corresponding file and the date and time of the autosave, formatted as a Java language comment. The rest of the temporary file, starting with line 2, stores the contents of the buffer at the time the autosave was performed.

To recover from a system crash or power failure, examine each file with the extension `.sav` in the `temp` directory.

Automating source file backups

You can control whether files in a project are automatically backed up each time that a save is performed. You can also specify the location and name of the backup files.

Note: Only Java and HTML files that have changed are backed up. For example, if you save all files in a project, only the files that have changed are backed up.

To automate project backups:

1. From the Tools menu, choose Environment Options, then click the Backup tab.
2. Select Backup files on Save.
3. Select the location and name of the backup files:

Select…	To specify this…
Create BAK file	Create one or more backup files that are named *file*.bak.

Select…	To specify this…
Copy to directory	Copy the source files to the specified directory. You can type the directory with the full path into the text box, or click **…** to select a directory from a dialog box. There is no default.
Invoke OnBackup script	Run your own macro, created with the Visual Cafe macro utility and containing an **OnBackupFile** statement. All macros are placed in the \VisualCafe\Bin\Macs directory.

Setting the scope of the Undo command

The Undo command is available from the Edit menu and has standard Windows functionality. The default number of undoable actions is 100.

To set the scope of the Undo command:

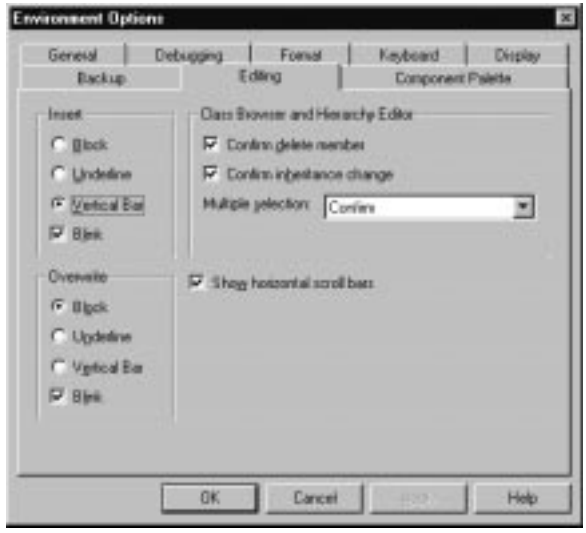
1. From the Tools menu, choose Environment Options, then click the Backup tab.
2. In the Save actions for undo field, enter the number of previous actions that you want Visual Cafe to store for each window.

Specifying code editing options

The Editing environment options of Visual Cafe allow you to change standard editing functionality for Source windows, the Class Browser, and the Hierarchy Editor.

In a Source window or pane, you can toggle between Insert mode and Overwrite mode by pressing the Insert key. In Overwrite mode, the new characters overwrite the existing text. The Editing tab of the Environment

Options enables you to specify the appearance of the Insert and Overwrite modes.



To access the Environment Options Editing view:

From the Tools menu, choose Environment Options, then click the Editing tab.

Controlling the cursor style

The Insert and Overwrite groups offer you options that govern the cursor style of the respective modes.

Option	Description
Block	Covers the current character.
Underline	Underlines the current character.
Vertical bar	The standard insertion point cursor, displays between characters.
Blink	When selected, the cursor blinks. You can set the blink rate in the Windows control panel.

Setting the cursor style

To set the cursor style for the insert and overwrite cursor:

1. From the Tools menu, choose Environment Options, then click the Editing tab.
2. In the Insert or Overwrite definition area, select or clear the Blinking option.
3. Select the cursor style: Block, Underline, or Vertical Bar.

To show horizontal scroll bars:

Select this option to display a horizontal scroll bar at the bottom of Source windows and panes (it is selected by default.)

Controlling toolbar position and visibility

Visual Cafe toolbars are at the top of the Visual Cafe main window. You can control toolbar position and visibility.

To float a toolbar:

1. Drag the toolbar from the top of the Visual Cafe window onto your desktop.
2. Double-click somewhere in the toolbar background.

To dock a toolbar:

1. Drag the toolbar to the top of the Visual Cafe window.
2. Double-click somewhere in the toolbar background.

To hide or show a toolbar:

1. Right-click at the top of the Visual Cafe window and select the toolbar name.
2. Click the close box on the toolbar.

Enabling ValueTips

ValueTips display the value of a variable in the Source window when you place your cursor over the variable at debug time.

To enable ValueTips:

1. From the Tools menu, choose Environment Options.

2 Click the Debugging tab.

3 Select or clear the Enable Value Tips option.

4 If selected, set the delay time before a tip displays.

Customizing Class Browser and Class Hierarchy window editing

You can enable multiple component selection and select confirmation options for both the Class Browser and the Class Hierarchy.

To customize Class Browser and Class Hierarchy window editing:

1 From the Tools menu, choose Environment Options, then click the Editing tab.

2 In the Class Browser and Class Hierarchy area, select or clear appropriate options:

Option	Description
Confirm Delete member	Display a confirmation message before deleting a class member.
Confirm Inheritance change	Display a confirmation message before applying inheritance changes.
Multiple selection	Display a confirmation message before applying a change to multiple selected objects. Yes - allow multiple select. No: disallow multiple selects. Confirm: allow but confirm changes to multiple select.

Note: You can also enable a horizontal scroll bar in the Class Browser source pane by selecting the Show Horizontal scroll bar option in the Source window area.

Changing editor properties

You can change the behavior of the Visual Cafe editors to suit your individual work needs.

To make local changes (local to an editing session):

From the Source menu, choose Format Options.

The following items can be customized:

- Font Style and Size

  Sets the appearance of text in Source windows.

- Colors

  Both foreground and background color for the selected window can be customized. You can also preset the colors for certain text items. The items for which you can change the color include: breakpoint text, code window text, comment text, current statement text, custom keyword text, error text, identifier text, keyword text, next statement text, and selection text.

- Tab to Space Conversion

  Converts tab characters to spaces as you type. You can also preset the number of spaces used for each tab character.

- Automatic Source Formatting

  Automatically formats code as it is entered, indenting lines of code and adding closing braces and parentheses.

Updating Visual Cafe with LiveUpdate

LiveUpdate is a Symantec technology that allows its registered users to upgrade selected products online. LiveUpdate keeps track of packages that are downloaded to your computer so that it can determine the appropriate upgrade for your software.

You can update your version of Visual Cafe by initiating the process through Visual Cafe over a TCP/IP based network connection to the Internet, such as your office workstation. You can also use LiveUpdate with a TCP/IP-based connection with your online or Internet service provider.

LiveUpdate also works through a modem connection to a Bulletin Board System (BBS) in case you do not have access to the Internet or your access is restricted by firewalls.

After you connect to the LiveUpdate center, LiveUpdate presents you with the most recent update to your version and edition of Visual Cafe. You must run LiveUpdate again to install subsequent updates. For example, if you have Visual Cafe 1.0d on your hard drive, and you want to upgrade to

1.1, you must install Visual Cafe 1.0e and subsequent versions of Visual Cafe before updating to the 1.1 version.

Using LiveUpdate over the Internet

If you already have a working connection to the Internet, you select the Use Existing Internet Connection option. No further setup is required.

If you do not have a working connection to the Internet or you wish to connect by modem instead, you should proceed to the Select a Modem wizard panel and select your modem (and the communication port your modem is using) from the available choices.

Using LiveUpdate with your modem

If you have a TCP/IP-based connection to the Internet through your Internet service provider, you can still use LiveUpdate to update your copy of Visual Cafe. Your dialup or ISDN account with your Internet service provider must be TCP-IP based, such as PPP, SLIP, or CSLIP accounts. You cannot use LiveUpdate through a Shell account.

You can also use LiveUpdate if you have an account with an online service such as America Online (AOL) or Compuserve. You must have the appropriate software installed and configured to use TCP/IP.

Configuring your modem

When LiveUpdate starts, the first wizard panel contains a button titled Modem Setup. Pressing this button allows you to change your modem settings at any time. If you choose not to setup your modem initially and use your existing Internet connection, you will still be able to setup a modem in the future.

Identifying your modem

If you have an external modem, examine it for a manufacturer and/or model label. Sometimes this information is on the underside of the modem. If you cannot identify your modem in this manner, or if you have an internal modem, it is suggested you try selecting Standard Modem Types from the list of manufacturers and then select the Standard modem model that most closely matches your modem's speed, if known. For example, if you have a 14,400 baud (14.4) modem you might try selecting 14000 bps Modem as the model.

If you do not know the speed of your modem, it is suggested that you first try 9600 bps Modem. If that does not work, try 2400 bps Modem. As a final alternative, try selecting Hayes as the manufacturer and either Compatible (default) or Compatible (alternate) for the model. If none of these selections result in a successful connection to a LiveUpdate service, you will need to consult your computer equipment vendor or company MIS/ Help Desk for assistance in determining your modem type.

If your modem manufacturer is listed but the particular model is not, you should select Other from the model list. This choice will usually enable your modem to properly connect, download, and disconnect.

If you find your modem manufacturer in the list but not your model, select Standard Modem Types from the list of manufacturers and then select the Standard modem model that most closely matches your modem's speed. For example, if you have a 14,400 baud (14.4) modem you might try selecting 14000 bps Modem as the model.

Configuring your modem INIT string

If LiveUpdate has your modem manufacturer and model listed, but you'd like to modify the INIT string to better suit your preferences, LiveUpdate allows you to do this by pressing the Edit String button on the Select a Modem wizard panel.

While it is suggested that only advanced users who know the implications of changing their modem's INIT strings actually attempt to do so, LiveUpdate allows you to partially or completely change the supplied INIT string. LiveUpdate does require E0 and V1to be present in the INIT string for proper operation. Again, most users will never need to change the supplied INIT strings, and this provision is provided for the few users who need or want to modify the default string.

If you've made modifications to the default INIT string but now would like to restore the original string for your particular modem, you can do this by reselecting your modem manufacturer and model from the list.Your custom INIT string will be replaced by the default string for the selected modem.

Selecting the COM port

In the Select a Modem wizard panel, you are asked to select the COM port for your modem. If you aren't sure which is the correct COM port, press the Find My Modem button. The LiveUpdate setup wizard will attempt to locate any modem(s) you have attached to any of the four basic COM

ports. If LiveUpdate finds a modem, it places a red dot next to the corresponding COM port.

If you have two (or more) modems on your system, LiveUpdate allows you to choose the correct one if you know the COM port number of the modem you want to use. To select the corresponding COM port, highlight that COM port in the listbox. If you don't know the COM port, press the Find My Modem button and LiveUpdate will place red dots corresponding to all found modems next to the appropriate COM port number or numbers. Select the COM port from among the entries marked with a red dot.

Selecting dialing parameters

In the Number to Dial wizard panel, there are a number of countries in a listbox. You select the appropriate country depending on where you are geographically at the time you run LiveUpdate. Users in the United States and Canada choose the service in Eugene, Oregon. Users in New Zealand, for example, should connect to the service in Australia. If you are a laptop user and travel extensively, you may need to change the service location to which you connect based on the service that is closest to you at the time you run LiveUpdate.

If your company phone system requires that you dial a "9" or other code to access an outside line In the Number to Dial wizard panel, there is an edit box at the bottom of the panel with the caption, LiveUpdate will dial. Normally, this edit box contains the phone number of the service to be dialed. However, you can add whatever codes before or after the number that you might need. Specifically, you may enter codes to access outside lines, dial country codes, enter a calling card to bill the phone call to, disable call waiting, access alternate long distance carriers as well as any other required codes.

If you have call waiting on your phone line, you can disable call waiting so that you won't get disconnected if you receive a phone call while you are using LiveUpdate. The code required to disable call waiting varies depending on your phone system. Commonly, codes such as *70, 70#, or 1170, if dialed before the phone number, temporarily disable call waiting. For the specific code to disable call waiting in your area, contact your local phone company.

If you need to dial a particular access code, then wait for a few seconds before dialing the remainder of the number. You can tell LiveUpdate to pause by entering a comma (,) in the LiveUpdate dial edit box after the last

digit where you want the pause to occur. The comma causes a modem-dependent pause period. Commonly, it causes a pause of a few seconds. To increase the time, add additional commas.

Troubleshooting a LiveUpdate connection

If you are informed that LiveUpdate detected a problem while retrieving your software update, run LiveUpdate again.

While a number of errors may have occurred, the single most likely problem is that you have insufficient hard disk space to accommodate the size of the software update being downloaded, especially with larger updates. The updates are compressed, so they require additional hard drive space beyond their original (compressed) size. If there is not enough space for the final files, this error may be displayed. If this error occurs, you should clear some hard drive space (usually on the drive that contains Windows) and try LiveUpdate again.

LiveUpdate informs you if you have the latest software update for your product. When you next use LiveUpdate, it will retrieve any updates that have been released since you last ran LiveUpdate.

Uninstalling LiveUpdate upgrades

After you have uninstalled previous software upgrades, your Windows 95 or Windows NT registry may still be “dirty” and therefore not allow LiveUpdate to install upgrades to Visual Cafe.

LiveUpdate includes a utility within the Visual Cafe directory to “clean up” the registry for Windows 95 or Windows NT and allow you to use LiveUpdate again to install upgrades to Visual Cafe.

Using LUCLEAN.EXE

`LUCLEAN.EXE` removes any changes LiveUpdate leaves behind in the Windows registry. `LUCLEAN.EXE` is non-destructive to any component of the operating system.

To run LUCLEAN.EXE:

1. Open a DOS session window
2. Switch to the `Visual Cafe` directory
3. Type `LUCLEAN.EXE /LU`

4 Restart your computer and run LiveUpdate again.

Troubleshooting Visual Cafe for Windows

As with all development tools, your success with them can be dependent on a number of things, and Visual Cafe is no exception. Your programs may or may not work depending on your system configuration, Visual Cafe environment settings, and your development and coding styles.

The following section provides some clues to solve the most frequently encountered development problems with the Java language and Visual Cafe.

Limitations of the Java language

Java has some architectural limitations that keep it from performing certain tasks.

Java and case sensitivity

Java compilers and interpreters are case sensitive, meaning that upper case letters are distinguished from lower case letters. `Hello.java` is not the same file as `hello.java`. The different use of capital letters indicates that these files are not the same.

When a `.java` source file is compiled, the compiler creates a file with a `.class` extension. The name that the compiler assigns to that `.class` file is taken from the class definition in the source file. Essentially, the class you define in the source code becomes the name of your program. Like all things in Java, the class definition is case sensitive.

Because of the case sensitive nature of Java code, compilers, and interpreters, it is important to be careful and consistent when you name projects, source files, and classes.

To prevent problems resulting from case sensitivity, get in the habit of always naming the project and the main source file with the same case sensitive name as the class you are creating.

Hardware limitations

Another limitation of the Java language is its inability to communicate directly with computer hardware, such as video cards and modems, or any

other devices that use some kind of port on a computer. The Java VM is blind to system-specific resource details, making it impossible for Java to access hardware.

Such tasks as printing and modem control are useless with Java, but JDK 2.0 is supposed to remedy these problems.

Common programming errors

The most common compiler errors result from improper syntax. The compiler can only understand source code that has been formatted correctly. Some of the most common mistakes are:

- Misspelled words (keywords, methods)
- Missing, or incorrect separators (periods, braces, semicolons)
- Improper use of case (capitalization)

One compiler error can generate multiple error messages. For example, you could misspell the name of a method that you call from your program. Although you misspelled it only once, your program may reference the misspelled method multiple times. The result is multiple compiler errors from a single typing error.

Sometimes the compiler error message will indicate exactly what caused the problem. You may have to inspect the troublesome line of code character-by-character to locate the source of the problem.

Compiler errors

A compiler error occurs when the compiler does not understand a portion of your source code. Even in the simplest program, it is easy to write code that will result in a compiler error.

There is a difference between compiler errors and programming errors. For example, if source code fails to compile successfully, you have a compiler error. If the source code compiles successfully but does not run as you expect, you have a programming error. It is possible for a program to compile successfully and still not run. Fixing the logic of your program to get it to run the way you want is called debugging your program.

The Messages Window displays all of the compiler errors encountered each time you build or compile a program.

Using Visual Cafe to locate compiler errors

When the compiler encounters an error in the source code, the error is displayed in the Messages Window. Double-click the error message to jump to the line containing the error. The editor highlights the error. The integration between the Messages Window and the Source Editor saves you from searching through every line of your code to find an error.

Cross-platform considerations

Probably the most powerful feature of the Java language is being able to run Java programs on a variety of platforms and Web browsers. For example, it is common to develop your Java program with Visual Cafe for Macintosh and then run your Java program on the Windows or UNIX platforms.

Visual Cafe cross-development

The `.java` and `.class` files that Visual Cafe for Windows and Macintosh are cross-compatible. However, the project files that are generated on each platform are not cross-compatible. If you want to transfer `.java` and `.class` files between platforms, create new projects and add the files.

These incompatibilities will be removed in future releases of Visual Cafe for each platform.

Browser issues

Although Visual Cafe is designed to be an efficient and fast development tool, you must also develop your Java applet to run on other combinations of platforms and Web browsers.

Besides the built-in security restrictions that are in Java applets, your Java applet will probably have a different look and feel (and sometimes behave) differently on other platforms and browsers. Here are some things to consider while you are developing your Java program.

- Select or design your GUI (graphical user interface) components carefully. For example, a button being viewed with the Macintosh version of Netscape will look different when you run the same program in Microsoft Internet Explorer for Windows NT. The same

button could look different between the different versions of Netscape Navigator for the various flavors of UNIX.

- Understand how your platform and browsers handle threads. Macintosh, Windows 95, Windows NT, and UNIX operating systems each handle threading differently, as well as their respective browsers.
- Each platform and browser has its own way of terminating applets. For example, when you start an applet in one browser, move to another page, then move back to the original page, the applet still runs and is consuming resources. In other browsers the same applet will terminate when you move to another Web page and then back. As of this writing, this inconsistency is not due to a Java language-specific problem, but rather a lack of standards for implementing Java capabilities in Web browsers.

When do you have to write your own code?

Generally, Visual Cafe generates all the source code you need to quickly create basic applets and applications without having to write much of your own source code. Visual Cafe also provides the framework for developing much more complicated applets and applications, eliminating the need for writing the routine and tedious source code and allowing you to develop and write the more sophisticated aspects of your Java program.

Event handling

One such situation where you will need to write your own code is when you need to implement event handling that can't be done through the Interaction Wizard.

Disabling automatic code generation in Visual Cafe

If you don't need to use some of the RAD features of Visual Cafe, such as automatic code generation, you can disable it.

You can removing the `//{{INIT_CONTROLS` line and its corresponding closing `//}}`. Also, remove the `//{{DECLARE_CONTROLS` line and its corresponding closing `//}}`. Removing these lines disables the Form designer and Visual Cafe stops altering your code.

However, this fix is not permanent. If you add anything to the applet by dragging it from the Component toolbar or the Component library, the

sections will be added back to the project and you will have to remove them again.

Disabling code that is automatically generated

You can tell Visual Cafe not to execute code that it generates. Place the Visual Cafe generated code between a block like this:

```
if(false) {
//{{INIT_CONTROLS
.
.
.
//}}
}
```

Visual Cafe sees your code at design time, but at runtime, the code that is automatically generated is ignored.

How to tell when your Visual Cafe environment becomes corrupted

When you start Visual Cafe, if your Component Palette is empty, or if you get an error saying "There are no starter templates in your repository", your Visual Cafe environment is probably corrupted.

To restore your Visual Cafe environment back to its working state:

1. Quit Visual Cafe.
2. In the `VisualCafe\Bin` directory delete the files:
   `local.rps`
   `vcafe.reg`
   `Visual Cafe.vws`
3. Restart Visual Cafe.

II

Professional Features

CHAPTER 10

Creating Native Win32 Java Applications

Visual Cafe Professional Development Edition lets you create native Win32 Java applications that run without using the virtual machine. Native Win32 Java applications offer the following advantages over bytecode Java applications:

- Speed – Because you do not need to run native Win32 applications using a Java virtual machine, applications will run faster.
- Packaging – Unlike bytecode Java applications in which all the class files must be available in order to run the application, Native Win32 Java projects can create an executable (EXE) file that contains all the class information.
- Compatibility with already existing executable and dynamic link library (DLL) files – You can include code already written in C or C++ in your native Win32 Java applications with minor modifications.
- Portability – Application projects written for native Win32 still generate all the class files necessary to port the Java code to another platform.

In addition to creating new native Win32 Java applications, you can also convert your existing bytecode applications into native Win32 code.

Support for native development includes several Sun and Symantec Java API packages already converted to DLLs. The following table lists the

import library files, the DLL files, and what packages are contained within each file.

Library File	DLL File	Java API Package Type
Snjrt11.LIB	Snjrt11.DLL	Core API
Snjbeans11.LIB	Snjbeans11.DLL	Beans
Snjres11.LIB	Snjres11.DLL	Resource support including audio
Snjnet11.LIB	Snjnet11.DLL	Networking and Internet support
Snjmath11.LIB	Snjmath11.DLL	Math
Snjzip11.LIB	Snjzip11.DLL	Compression
Snjsec11.LIB	Snjsec11.DLL	Security
Snjrmi11.LIB	Snjrmi11.DLL	RMI
Snjawt11.LIB	Snjawt11.DLL	AWT
Symbeans.LIB	Symbeans.DLL	Visual Cafe Components

To find out what classes are specifically contained in each DLL, see `packlst.txt` in the `Redist` directory of Visual Cafe. This file will always have the most current information regarding the classes stored in each DLL.

This chapter contains an overview of the steps to create native Win32 Java applications or convert existing bytecode applications to native, things to remember while working with native applications, and an example of creating a simple native executable with a DLL.

Creating native executables and DLLs

The process of creating a native executable or DLL file is very similar to the process used to create bytecode Java applications. In general, the development cycle is made of these basic steps:

1. Create a new project.

   To get started quickly, you can use the Basic Win32 GUI Application, Basic Win32 Console Application or Basic Win32 Dynamic Link Library project template.

2. Design the user interface.

See Chapter 5, “Including Visual Components” for more information.

3 Enhance Java source code.

See Chapter 4, “Working with Java Source Code.”

4 Set project options.

See the section "Setting project options for native applications" in this chapter.

5 Test run the application or DLL in Visual Cafe.

See Chapter 6, “Compiling, running, and deploying your program.”

6 Debug the application or DLL , if needed.

See "Debugging Native Win32 Java applications" in this chapter.

7 Test run the application or DLL outside of Visual Cafe.

See Chapter 6, “Compiling, running, and deploying your program.”

8 Deploy the application or DLL.

See Chapter 6, “Compiling, running, and deploying your program.”

9 (Optional) Register the DLL using SNJREG.EXE, if necessary.

See "Registering DLLs using SNJREG" in this chapter.

An example of creating a simple native executable with a DLL can be found in the section “Example of creating an executable file” on page 10-17.

Setting project options for native applications

When you create a native Win32 Java executable or DLL you need to set the project options so Visual Cafe recognizes that you are developing Native Win32 Java applications. You also need to set options to tell Visual Cafe to debug a native Win32 Java application.

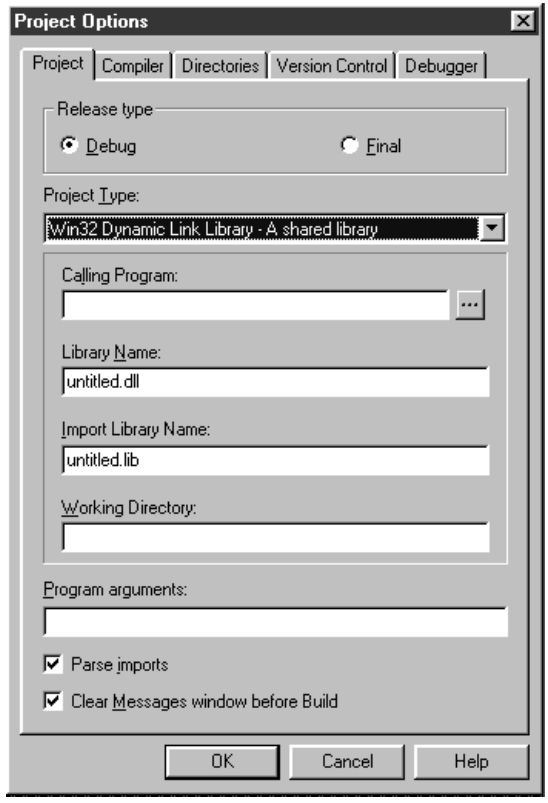
This section contains information on setting project options specific to building native applications. For information on general project options, see Chapter 3, “Working with projects and workspaces.”

Specifying the name of a native application or DLL

Visual Cafe provides a default file name of untitled to executable and DLL files. The default file name is untitled.exe or untitled.dll. You can change the name from the default by performing the following steps.

To specify the name of a native executable or library:

1. Activate the Project window of the project you want to work with.
2. From the Project menu, choose Options.
3. In the Project Options dialog box, click the Project tab.



4. Set the Project Type field to Win32 Application or Win32 Dynamic Link Library.
5. Enter the executable or library file name in the Application or Library name field.

   The application name is by default the project name, appended with the `.exe` extension.

   The library name is by default the project name, appended with the `.dll` extension.

The import library name is the name of the import library file that gets created along with your DLL. It is by default the project name, appended with the `.lib` extension.

6 Click OK.

The change takes effect the next time you run your project.

Specifying a program for running and debugging a DLL

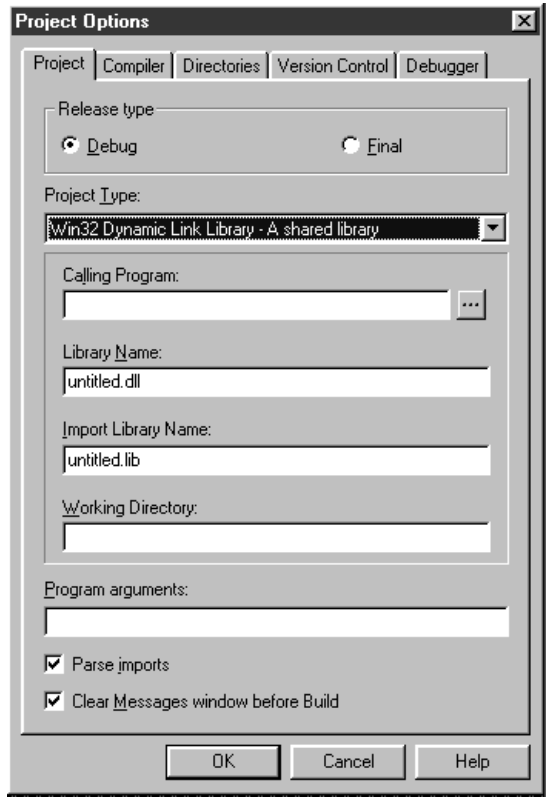
Because DLLs are called by an executable file to run, you need to specify the name and path of the executable file so Visual Cafe can run the DLLs for testing or debugging.

To specify an executable file for running or debugging a DLL:

1 Activate the Project window of the project you want to work with.

2 From the Project menu, choose Options.

3 In the Project Options dialog box, click the Project tab.



4. Set the Project Type field to Win32 Dynamic Link Library.
5. Type the name of the executable in the Calling program field.
6. Click OK.

   The change takes effect the next time you debug the DLL.

In the Calling program field, you specify the name and fully qualified path to the executable. If you do not specify a fully qualified path, Visual Cafe looks in the project directory, then through the directories in your Windows PATH environment variable.

When the calling program tries to load the DLL, it looks for the DLL in the following order:

- The directory from which the application loaded. This is the project directory or, if you specified a full path, the directory where the program resides.
- The current working directory, if different from the directory from which the application loaded.
- For Windows 95, the Windows system directory. For Windows NT, the 32-bit Windows system directory, then the 16-bit Windows system directory.
- The Windows directory.
- The directories that are listed in the Windows PATH environment variable.

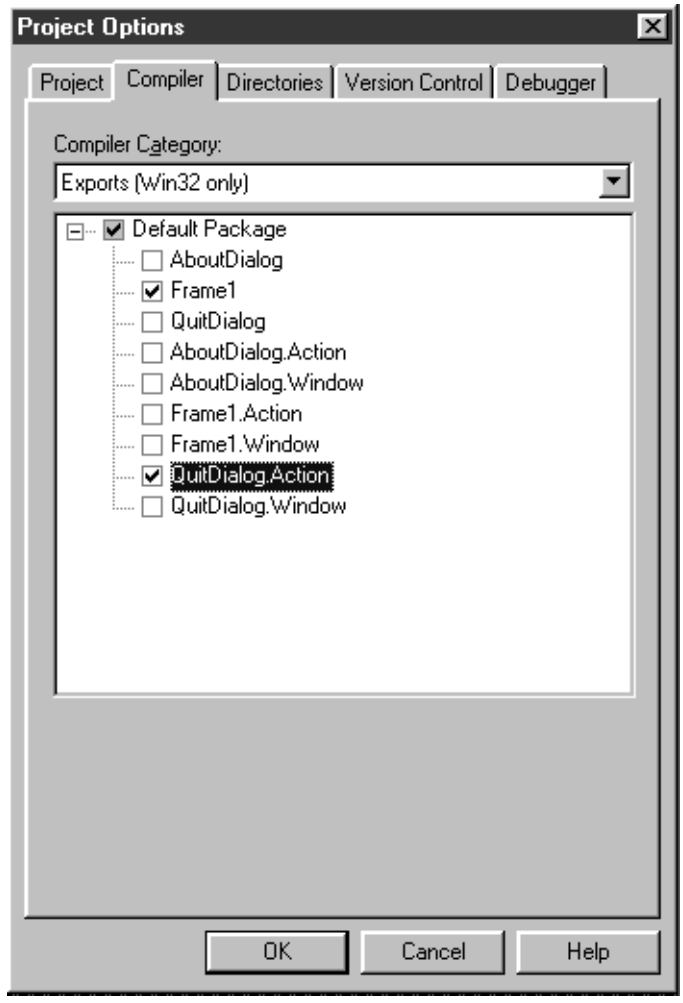
Specifying a class or package to be exported

When you build DLLs you need to tell Visual Cafe which classes or packages can be used by calling programs.

To specify a class or package to be exported:

1. Activate the Project window of the project you want to work with.
2. From the Project menu, choose Options.

3 In the Project Options dialog box, click the Compiler tab.



4 Set the Compiler Category to Exports (Win32 only).

5 Choose the classes or packages you want to make available to other projects by clicking on them.

6 Click OK.

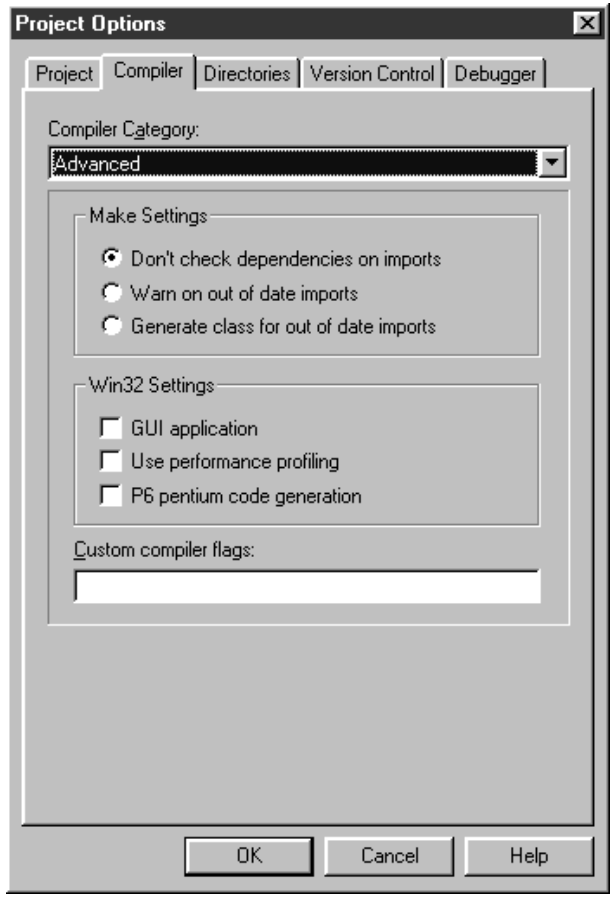
The change takes effect the next time you build your project.

Specifying advanced Win32 compiler options

Visual Cafe provides you with additional options that allow you control aspects of how Visual Cafe compiles files.

To specify a class or package to be exported:

1. Activate the Project window of the project you want to work with.
2. From the Project menu, choose Options.
3. In the Project Options dialog box, click the Compiler tab.



4. Set the Compiler Category to Advanced.

5 Click the checkboxes of the items you want to select. There are three options:

Option	Description
GUI application	Suppresses the console window. The default setting is disabled.
Use performance profiling	Attributes the code with profiling calls so the executable can generate profiling information. The default setting is not selected.
P6 Pentium code generation	Generate P6 Pentium code. The default setting is to generate P5 Pentium code.

6 Click OK.

The change takes effect the next time you build your project.

Including library files to be compiled in the main project

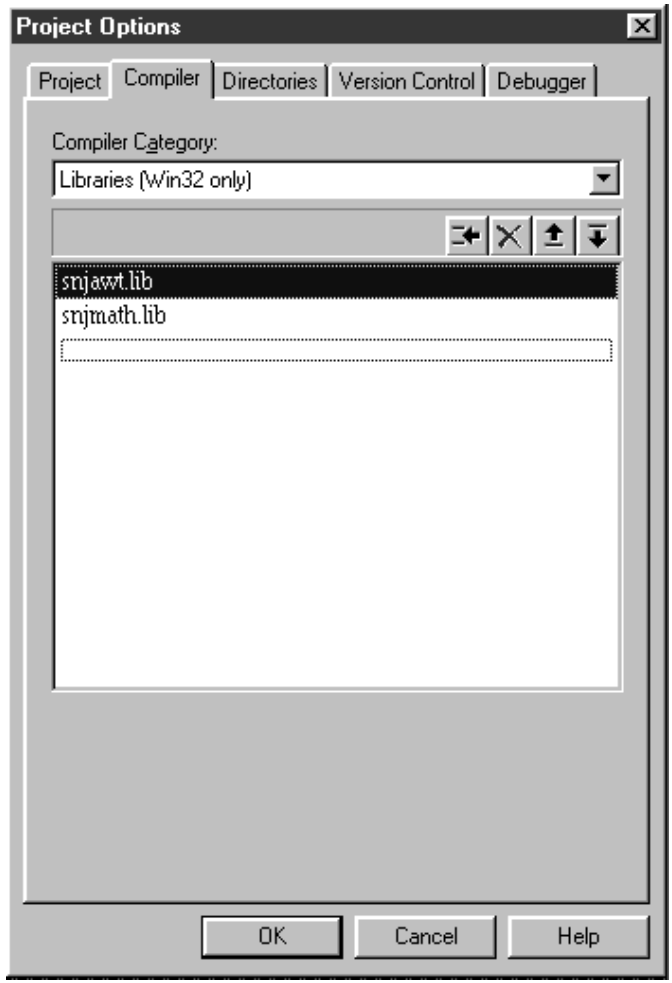
In order to build the executable file and the associated DLL files, Visual Cafe needs to know the names of the import library files for the associated DLLs.

To include import library files:

1 Activate the Project window of the project you want to work with.

2 From the Project menu, choose Options.

3 In the Project Options dialog box, click the Compiler tab.



4 Set the Compiler category to Libraries (Win32 only).

5 Add the library file.

6 Click OK.

The change takes effect the next time you run your project.

Making a library file available to a project

Since a library file may be in a different directory from the main project, you can specify the directory in which the library files you want to use are

located. If you have already specified all the library files you are including in a project, you do not need to specify a directory.

To set the directory:

1. Activate the Project window of the project you want to work with.
2. From the Project window, choose Options.
3. In the Project Options dialog box, click the Directories tab.

4. Set the Show Directories for Option field to Library files.
5. Select from the list of directories available or add the directory where the library files are located.
6. Click OK.

The change takes effect the next time you run your project.

Registering DLLs using SNJREG

Unlike bytecode Java applications, native Win32 Java applications cannot use the CLASSPATH to find classes and packages. Instead, the Windows path is searched to locate classes and packages in DLLs. Classes that are dynamically loaded must have their names stored in the Windows registry. The native classes and packages Symantec provides with Visual Cafe are already registered when Visual Cafe is installed.

If you create a DLL whose class or package names are not stored in the registry, the packages or classes in the DLL might not be found by an application that loads these classes dynamically at runtime and you will receive an error message. SNJREG allows you to enter the package or class name and which DLL files contain the package or class.

SNJREG has the following parameters:

Parameter	Description
-class	DLLs register individual classes, not packages. Using -class will erase any previous DLL references assigned to a class already registered.
-noprompt	Disables prompting the user before making changes to existing entries
-nowarn	Turns off all warnings
-verbose	Reports all registrations made
-reg *file.reg*	Creates a registry file where *file.reg* is the name you give the new registry file

To register a package in a DLL file using SNJREG:

1. Open a DOS Window.
2. Change to the directory where the DLL files are located.
3. Enter:

   ```
   SNJREG [options] file1.dll file2.dll file3.dll
   ```

   where `options` can be any of the parameters in the table above and `file1.dll, file2.dll`, and `file3.dll` are the names of the

DLL files from which you are registering the classes or packages. You may enter the names of as many DLL files as you want. If the DLL files are not in the current directory, make sure you enter the full path for each file.

Debugging Native Win32 Java applications

You can debug native applications and DLLs in Visual Cafe the same as you would debug bytecode programs. For DLLs, you need to specify the calling program you want to use to run and debug your DLL. See "Specifying a program for running and debugging a DLL" on page 10-5 for more information.

Note the following differences between debugging native and bytecode programs:

- The Calls and Threads windows are different for native and bytecode programs.
- When debugging native code while incremental debugging is enabled, you can add new methods, while with bytecode you cannot.
- You cannot perform remote debugging.

For more information on debugging applications, see Chapter 8, "Debugging Java Programs."

Deploying native Win32 applications, DLLs and libraries

To deploy your native Win32 application, you need the DLLs the program requires. Visual Cafe comes with the standard DLLs you need to run your Java programs. The redistributable Visual Cafe DLLs are in the redist directory. You only need snjrmiregistry.exe if you have a remote method invocation (RMI) executable that is like one of the samples, which does not start the naming server itself (The other RMI sample does not require snjrmiregistry.exe.). You also need any DLLS that your DLL requires, including DLLs you created.

You need to register any dynamically loaded DLLs on the computer where the executable will run or the DLL is used. To do so, you can use the SNJREG tool. To do so, you can use the SNJREG tool. See Registering DLLs with SNJREG for more information.

To create a program that uses your custom DLL, you need to provide the class files, the library file (which is created when the DLL is created), and the DLL. The class files must be on the class path and the library should be

specified in the compiler options for the project. To run a program that uses a DLL, all you need is the DLL, and you need to register it.

Converting Java applications from bytecode to native Win32

If you already have Java applications that you want to run in the WIn32 environment , you can convert them from bytecode to native Win32 code for easier packaging and increased speed benefits. See the sample `Conversion` in the Samples\Symantec\Tutorials\Win32 directory.

To convert your bytecode Java application to native Win32:

1. Open the project you want to convert.
2. From the Project menu, choose Options.
3. In the Project Options dialog box, click the Project tab.
4. Change the Project Type field to Win32 Application or Dynamic Link Library.
   - Choose Win32 Application if the project does not contain any files you want to use with other native Win32 applications.
   - Choose Win32 Dynamic Link Library if you will be sharing the code with other native Win32 applications.
5. Click OK.

Based on the choices you make in the Project Tab, you may need to change additional Project Options in order to build your application successfully. See “Setting project options for native applications” on page 10-3 and “Considerations when creating native Win32 Java applications” on page 10-15 for more information. If you need to register DLLs, see “Registering DLLs using SNJREG” on page 10-12. If your project has errors when you build it, see “Debugging Native Win32 Java applications” on page 10-13. See Chapter 3, “Working with projects and workspaces” for information on setting class file directories.

Note: SNJRT11.DLL does not need to be added to converted applications. It is added at compile time for native Win32 applications.

Considerations when creating native Win32 Java applications

There are several things you can do when developing native Win32 Java applications to minimize build errors. Keep the following in mind when creating native Win32 Java applications:

- Keep all the files in a project. Build any DLLs as subprojects that you add to the main project.
- Use SNJREG to register any Java DLLs you create.
- When compiling DLLs in the command line, use the -export option.
- Remember that `SNJRT11.LIB` is automatically linked to your project.
- During runtime of the executable, make sure that the path includes the proper folder containing the DLLs.
- Statically linked DLLs have to be in the Windows path. For an example, see the `staticDLL` sample in the Samples\Symantec\Tutorials\Win32 folder.
- Before running an application that dynamically loads classes during runtime, you must register the DLLs using SNJREG. For example, if you use `Class.forName(className)`, to access a class in a DLL, you need to register the DLL. The sample, dynamicDLL, in the Samples\Symantec\Tutorials\Win32 folder illustrates this.
- If you link object files that contain static members to different library or DLL files in a project, be aware that there is now more than one copy of the static member.
- Do not include any library files in the Files tab of the Project window. Include library files in a project by using the Compiler tab in the Project Options. For more information, see "Including library files to be compiled in the main project" on page 10-9.

In addition to the considerations listed above, you must keep in mind that native Win32 Java application development uses linking when compiling and that the main class is treated differently in native Win32 development.

Linking native Java applications

There may be extra steps you need to perform for a native project to link that you don't need to do for a bytecode project. In bytecode, Java uses the classpath to dynamically and automatically resolve references to classes. However, in native Win32 applications the Java classpath is not

used to resolve these references. They must be resolved by static linking, which means they are resolved when the application is built.

Resolution by static linking occurs in the traditional compile/link mechanism. The object (`.obj`) files, which are generated by the sj compiler for each Java class, are linked together by the linker which resolves the references. The project system supplies the linker with the object files and libraries necessary to build your native Java executable. Linker errors will occur if the linker was unable to resolve some reference made to a class in your project. Typically, an object file corresponding to a class that your project uses did not get passed to the linker by the project system. Standard Java runtime libraries and Symantec components are automatically supplied by the project system.

To prevent linker errors, you need to add the Java source file for any missing classes to your project. Other possible solutions include adding the missing object files or import libraries to your project. Any symbols in source files, object files or import libraries that are in your project will be passed to the linker and used to resolve references when you build an application. See “Setting project options for native applications” on page 10-3 for more information.

As in bytecode, native Java applications allow programs to dynamically load arbitrary classes at runtime, without requiring them to be statically linked. However classes that are not statically linked into your application at build time must exist in a separately compiled dynamic linked library (DLL) and be registered in the Windows registry by using the SNJREG tool. For more information, see “Registering DLLs using SNJREG” on page 10-12.

The main class in bytecode and native applications

The *main class* is the name of the class with a main method. For both a bytecode Java application and a native Win32 Java application, the main class is the starting point of execution. However, you should note these differences between bytecode and native applications:

- When you run a bytecode application from the command line, you type the name of the Java file that contains the main method as the first argument to Java.exe. For a native Win32 application, you run the application outside the Visual Cafe environment as you would any other executable and use the application name, not the main class name. See "Specifying a program for running and debugging a DLL" and "Specifying the name of a native application or DLL" for more information.

- An application must have a main class containing a main method with this signature:

  ```
  static public void main(String args[])
  ```

For a bytecode application, if it does not have a method of this format, the application can compile but will not run. For a native application, the application cannot link or run.

Working with samples of native applications

To illustrate how to create native Win32 Java applications, Symantec has provided several simple sample applications. These applications demonstrate how to create an EXE file, how to create an EXE and DLLs, what type of project requires you to register a DLL, working with the RMI registry, resource binding, and working with C code and JNI. The samples are located in the Samples\Symantec\Tutorials\Win32 directory. The following examples use the *.java files found in the sample called Exe. If you choose to use any of the samples, copy them to a new folder before making any changes to the code or the project options.

Example of creating an executable file

Creating a native Win32 Java application is very similar to creating a bytecode application. The example below builds a simple Win32 application that prints the word Hello.

To create a Win32 executable file:

1. Create a new project by choosing New project from the file menu and selecting Win32 console application.
2. Create a source file called Main.java that contains the following code:

   ```
   public class Main
   {
       public static void main(String args[] )
       {
           Hello hi = new Hello();
           hi.printHello();
       }
   }
   ```
3. Add a second source file called Hello.java with the following code:

   ```
   public class Hello
   ```

```
{
    public void printHello()
    {
        System.out.println("Hello");
    }
}
```

4. Delete the file `SimplCon.java`.
5. From the Project menu, choose Options, and using the Project Options dialog box, type `Main` in the main class field, and type `simple.exe` in the Application name field.
6. Compile and execute the application by choosing Execute from the Project menu.
7. Save the project to new directory called `simple`.

When you compile a native Win32 executable, the .java files are compiled into both class files and object files. Next, the object files are linked together to create an EXE file.

Example of creating an executable with a DLL

Let's take this example further by building an executable and a DLL file. The two files `Hello.java` and `Main.java` stay the same.

To create an executable and a DLL file:

1. Save the Simple project to a new name and directory, both called Simple3 and remove all files with the name Hello from the Simple3 directory.
1. Create a project for a DLL by choosing New Project from the File menu. Use the resulting dialog box to select Basic Win32 Dynamic Link Library as the project type.
2. Add `Hello.java` from the `Simple` directory to the project, and delete `SimplDLL.java`.

Note: If you were creating a DLL project from scratch, you would not delete the `SimplDLL.java` file, but use it as a template.

---

3. Save the project to a new directory called Simple2.
4. Set the project options for the DLL by choosing Options from the Project menu, and using the Project Options dialog box to select `simple3.exe` as calling program, `Hello.dll` as Library Name, and

`Hello.lib` as Import Library name. See "Setting project options for native applications" on page 10-3 for more information.

5. Make the Hello class exportable. See "Specifying a class or package to be exported" on page 10-6 for more information.
6. Build the project to generate `Hello.lib` and `Hello.DLL`.

After creating the DLL ,you need to add the Simple2 project as a subproject to the main project, Simple3.

To add the DLL to the Simple3 project:

1. Create a new WIn32 application project.
2. Delete all files from the project and add Main.java.
3. From the Project menu, choose Options, and using the Project Options dialog box, type `Main` in the main class field, and type `simple.exe` in the Application name field.
4. From the Insert menu, choose Files into Project.
5. Select `simple2.vep` and click Add.
6. Click OK to add the VEP file to the project.
7. Save the project as Simple3 in a new directory called Simple3.
8. Make `Hello.lib` a recognized library file. See "Including library files to be compiled in the main project" on page 10-9 for more information.
9. Build the simple3 project and execute.

What happens in this example is similar to building an EXE alone. The major differences are:

- more project options need to be set.
- the build order of the subproject and project must be considered when a DLL file is linked with an EXE file.
- `.lib` files are linked to the exe instead of linking multiple object files to form an exe.

CHAPTER 11

Incremental Debugging and Importing Projects

Visual Cafe Professional Development Edition provides many debugger tools including incremental debugging. Incremental debugging, also referred to as run-time editing, allows you to run your applet or application and debug it without starting over from the beginning of the code every time you locate and fix a bug.

Visual Cafe Professional Development Edition can not only read projects from older versions of Visual Cafe, it can also import Cafe and Visual J++ projects. This gives you more flexibility since you no longer have to rebuild project files created in these products. Importing Cafe projects includes the importing of any subprojects inside a Cafe project.

Incremental debugging

Visual Cafe has several incremental debugging features that make debugging your Java programs more efficient. While your program is executing or paused in the Visual Cafe debugger, you can edit your Java program and immediately see the effects of the code change. This is also called run-time editing. The code is compiled and saved automatically.

You enable run-time editing by choosing Environment Options > Debugging from the Tools menu. See Chapter 9, “Fine-Tuning Visual Cafe.”

Choose Update Now from the Debug menu to force an incremental update and save all files. Use this command when you want to save all the changes you have made to your files.

Choose Restart Method from the Debug menu to take you back to the method that called the currently active method.

Note: You should not think of the Restart Method command as a type of undo command, because it cannot undo some edits, such as variable edits. It does not undo side effects of code that was run; an example could be if part of a program runs two times and causes an exit. Use the command when you want to restart the method you are currently debugging.

Another incremental debugging feature is if you change a portion of code that is active (anywhere on the call stack), the code is compiled and saved, and you get a dialog box asking what action you want to take:

- Restart the program – Start the program from the beginning in the debugger.
- Restart the active method – Start the active method again. (This option is only valid if you perform run-time editing while your program is paused.)
- Continue – Continue with the old code until the next time the code becomes active. Ignore breakpoints in this code until the code becomes active again.
- Stop debugging – Exit the debugger.

The recommended action is already selected for you in the dialog box.

Note: When debugging native Win32 code, you can add new methods, while with bytecode you cannot.

For more information on debugging applications in Visual Cafe, see Chapter 8, “Debugging Java Programs.”

Importing projects from Cafe

If you previously owned Cafe and are moving to Visual Cafe, you may want to import your Cafe projects to take advantage of Visual Cafe’s rapid application development (RAD) environment.

To import a project:

1 From the File menu, choose Open.

The Open dialog box appears.

2 Choose Cafe projects as the file type.

3 Locate the correct directory and select the project file.

4 Click OK.

Your project is opened in Visual Cafe. All subprojects in the project are available.

Caution: Visual Cafe projects can not be read by Cafe. You can still move the source files between the two products.

Importing Visual J++ projects into Visual Cafe

Your Visual J++ project is converted automatically when you open the workspace (`dsw`) or project (`dsp`) file with Visual Cafe. If you open the workspace file, some of your workspace options (stored in an `opt` file) are preserved; these settings are not imported if you open the project file directly.

When Visual Cafe converts a project through the workspace file, it gives the project the same name as the Visual J workspace file and saves it immediately in the same directory as the workspace. If your workspace contains multiple projects, first convert the projects that are in their own directories first, then convert the project stored in the same directory as the workspace file. For a more complete list of considerations, see "Considerations when importing Visual J++ projects" on page 11-5.

Note: It is best to work from a copy of your Visual J++ files instead of the originals.

To import a project through the dsw workspace file:

1 From the File menu, choose Open.

The Open dialog box appears.

2 Choose Visual J++ Workspace Files in the Files of type field.

3 Locate the directory that contains the `dsw` file for the project you want to import.

The `dsw` and `dsp` files appear.

4 Select the `dsw` file from the list, then click Open.

5. If there is more than one project in the workspace, a dialog box appears that asks what project you want to convert. Select the project, then click OK.
6. If you have more than two configurations defined for the project, or if you have two configurations and neither of them are named "Debug," a dialog box appears that asks what configuration you want to use for the Visual Cafe Debug and Final option sets. Select the configuration and click OK in each dialog box.

   The project is converted and opens in a Visual Cafe Project window.
7. Review the Visual Cafe project options and change them as needed.
8. If the Visual J++ project was in a different directory than the dsw file, choose Save As from the File menu to save your new project to the directory where the project dsp file is. If you want, you can rename the project at the same time.

Tip: Remember that a Visual J++ project is given the same name as the Visual J++ workspace if the project file is in the same directory as the workspace file.

If you save the project to another directory, you can delete the Visual Cafe project files in the same directory as the workspace. Or you can just let them be overwritten when you import the next project.

To import a project through the dsp project file:

1. From the File menu, choose Open.

   The Open dialog box appears.
2. Choose Visual J++ Workspace Files in the Files of type field.
3. Locate the directory that contains the `dsp` file for the project you want to import.

   The `dsw` and `dsp` files appear.
4. Select the `dsp` file from the list, then click Open.
5. If you have more than two configurations defined for the project, or if you have two configurations and neither of them are named "Debug," a dialog box appears that asks what configuration you want to use for the Visual Cafe Debug and Final option sets. Select the configuration and click OK in each dialog box.

   The project is converted and opens in a Visual Cafe Project window.
6. Review the Visual Cafe project options and change them as needed.
7. Save your new project.

Considerations when importing Visual J++ projects

Your Visual J++ project is converted automatically when you open a workspace file (`dsw`) or project file (`dsp`) with Visual Cafe. While Visual Cafe has projects, Visual J++ has workspaces that contain projects. A workspace is made of a workspace file and an `opt` options file. Information about the projects a workspace contains is stored in a project file (one per project) and the options file. Because the name of the options file is not stored in the project file, it is usually better to import projects through the workspace file rather than the project file.

The following options are preserved from the Visual J++ opt file when you import a project through the `dsw` file or you import a `dsp` file that has the same name as the `opt` file and the `opt` file is in the same directory.

Visual Cafe project option	Visual J++ configuration that sets it
Project type field in Project tab	The Debug/Execute project under Browser and Stand-alone interpreter options determine the setting.
Start with Web page field in Project tab	The Use parameters from HTML page value is used; or the Enter parameters below option tells Visual Cafe to use Automatic.
Main class field in Project tab	The Class for debugging/executing value is used.
Runtime arguments field in Project tab	The Program arguments value is used.
Class Files list in Directories tab	The Class path directories setting is used. Note that in Visual Cafe the Append class path and Autogenerate class path options are selected by default.

The following options are preserved from the `dsp` project file, whether you open a workspace or project file.

Visual Cafe project option	Visual J++ configuration that sets it
Output directory field in Directories tab	The Output directory value is used.
Show compiler warnings option in Compiler tab	A Warning Level of None tells Visual Cafe to deselect the option; any other value means the option is selected.

Visual Cafe project option	Visual J++ configuration that sets it
Generate debug information option in Compiler tab	The Generate Debug info value is used.

Here are some additional considerations when importing Visual J++ projects:

- If you want to maintain your Visual J++ workspace and projects, you should copy the files to another directory before conversion.
- Converted projects cannot be saved back into Visual J++ format.
- Only Visual J++ version 1.1 workspace and project files can be imported. However, you can add Java source files created in another product to a Visual Cafe project. For instructions, see Chapter 3, "Working with projects and workspaces."
- Visual Cafe can read Visual J++ `java` files that are 100% pure Java, without proprietary Java extensions implemented by Microsoft.
- `class` files compiled with Visual Cafe might not be compatible with Visual J++, so you probably want to make sure your output directories are different for each product.
- You cannot import projects that use variable persistence, such as having the location of a file be stored as `d:\mysource\$(SRCDIR)\a.java`, where SRCDIR is an environment variable and a.java is the file. In Visual Cafe, you cannot use environment variables to control where Visual Cafe looks for a `java` file. If you want to import a project that uses variable persistence, you need to remove variable persistence before importing the project into Visual Cafe.
- While Visual J++ lets you specify a different project type, applet or application, for different configurations, Visual Cafe supports one project type per project. If the project type is different for the debug and final option sets, Visual Cafe uses the project type specified for debug.

III

Database Connectivity

CHAPTER 12

Developing a dbAWARE project

By using Symantec's Visual Cafe Database Development Environment, you can develop Java applets and applications which have components that interact with a database.

The Visual Cafe Database Development Environment has wizards that walk you through the setup of a dbAWARE project. Using these wizards you can quickly identify a data source for your applet or application and add buttons, which encapsulate database interactions, to your graphic interface. The Visual Cafe wizards automatically generate the necessary Java code for communicating with your server software while you use the wizards to create your Java applet or application.

An example of a dbAWARE applet might be an e-mail form for your Web page. An application might be a stock ticker application, which gets information from a database on the Web and stores it in a database on your local machine.

This chapter includes:

- An overview of the Visual Cafe Database Development Environment
- Steps for building a dbAWARE project

Note: The Symantec Visual Cafe Database Development Environment includes its own middleware software called dbANYWHERE. You must configure dbANYWHERE and have it running in the background before your applet or application can connect to a database. See Chapter 13, "Using dbANYWHERE."

Overview

The Visual Cafe Database Development Environment consists of all the Visual Cafe functionality plus the following database features:

- Database Environment Options settings
- The dbAWARE components
- The dbNAVIGATOR database browser
- Wizards to guide you through creating database projects
- Database functionality for navigation, state information, and so on

Setting the Database Environment Options

When you install the Database Development Edition of Visual Cafe, a Database tab is added to the Environment Options dialog box. You use this screen to create a list of the components to be associated with particular data types.

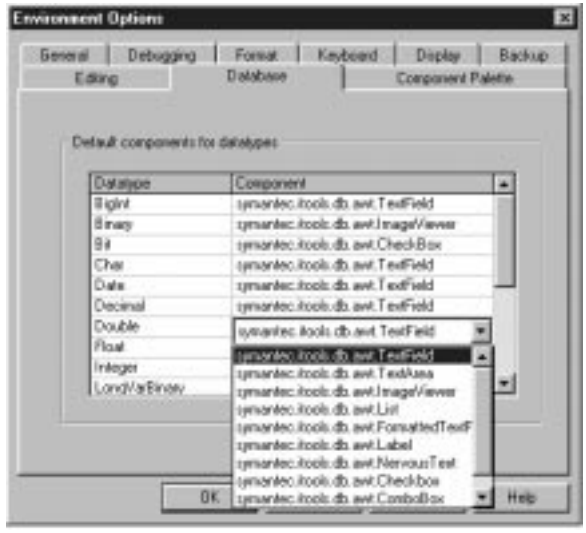
To use the Database Environment Options dialog box setting:

1. Choose Environment Options from the Tools menu.

   The Environment Options dialog box displays.

2. Click on the Database tab of the Environment Options.

   The Database Environment Options displays.



About the dbAWARE components

You use dbAWARE components to interact with database elements. These components provide the built-in functionality you need to display, communicate, and connect with a database, thus saving you extensive programming time and effort.

For example, you can set a dbAWARE TextField component to correspond to a table column. When you do this, the text reflects the value stored in that database table column.

Some of the dbAWARE components in the Visual Cafe Database Development Environment are:

- AutoDetail
- Checkbox
- ConnectionInfo
- DBTstamp
- FormattedTextField
- Grid
- ImageViewer
- Label
- List
- MultiView
- NervousText
- Projection
- RadioBox
- RadioButton
- RecordNumberLabel
- RecordStateLabel
- RelationView
- Session
- TextArea
- TextField

Some of the most important database components are documented in the following sections. For more information on any of these component's classes, see the dbANYWHERE API online HTML reference manual.

The AutoDetail Component

An AutoDetail object is an intermediate object that is used to define the properties for a Detail RelationView. A Detail RelationView typically manages a set of records that are related to the current record in another "parent" RelationView. For instance, a RelationView on a Department table may have a detail RelationView on a Employee table where there is a set of related records in the Employee table for every record in the Department table.

This relationship is usually defined by a common column in both tables. In this case, the data in the department id column for the Employee detail RelationView matches (is equal to) the data in the id column for the Department table. Whenever the current record changes in the Department table (via a `RelationView.next` call, for instance) the set of records available for the Employee detail RelationView automatically changes to maintain the id relationship.

You can set this kind of master/detail relationship by dragging a second set of database components including the database table with its RelationView, onto the Form Designer or into the dbNavigator. When you do this, code is automatically generated that sets the second RelationView to AutoDetail.

The most important properties you can set for an AutoDetail object are:

- ConnectionInfo object
- SQL select statement and join
- Cardinality
- Parent RelationView

ConnectionInfo object

This object holds the information needed to connect to a database. This information includes the database string, and optionally the username and password. AutoDetails are not restricted to connecting to the same database as the parent relationview.

SQL select statement and join

For AutoDetails, the SQL select statement must define a where clause that specifies the column that will be used to relate to the parent relationview. For instance, the Employee detail SQL string would be

```
"select * from Employee where deptid = ?"
```

In this SQL statement, `deptid` is the column in Employee table that will be used to define the relationship. The `?` will be automatically replaced by the data in the current record of the Department table.

Additionally, the related column in the Department table also needs to be defined. This is done by calling `AutoDetail.join` method.

```
AutoDetail.join (1,"id");
```

The first parameter in the join method indicates the position of the `?` in the SQL Select statement. Since there is only one `?` in the string, the number 1 is used. The second parameter is the name of the column in the parent Relationview.

Cardinality

The cardinality property defines how the sets of records in both the parent and detail relationviews are related. The dbANYWHERE server uses this property to determine the order of database inserts, updates, and deletes.

The following table includes all the possible values for the Cardinality property.

Values	Example
ONE_TO_ONE	One department record is related to only one employee record.
ONE_TO_MANY	One department record is related to many Employee records. This is the default setting.
MANY_TO_ONE	Many department records are related to only one employee record.
MANY_TO_MANY	Many department records are related to many employee records.

Parent RelationView

The parent RelationView is set during object construction. After the AutoDetail is constructed, other properties can be set. When `AutoDetail.executeRequest` is called, a new detail RelationView is created and joined to the parent RelationView. (Optionally, a new RelationView object can be constructed with the AutoDetail as the parameter.)

The ConnectionInfo component

The ConnectionInfo component is an invisible component which contains the data source definition. Basically, the data source is the information for creating a database connection. The ConnectionInfo component includes the name of the data source, its server location, and may also include a user name and password.

You can drag the ConnectionInfo component from the Component Palette or Library onto a form at any location, or into the Project window. If you are using the Project Wizard or the Add Table Wizard, Visual Cafe generates the necessary ConnectionInfo, Session, and RelationView components for you.

Though a project can contain more than one ConnectionInfo component, each ConnectionInfo component corresponds to only one database. The applet or application that contains the ConnectionInfo component must also include a Session component to enable it to connect to the dbANYWHERE server.

The Grid component

This component is for displaying multiple data records. The grid component provides an array, or spreadsheet-like display, of specified data. You can also specify predefined buttons that display at the bottom of the grid. You define these buttons in the Binding property of the grid.

You can adjust column widths at runtime but the column heading text can be customized in code only.

You need to define a RelationView component to manage result sets which you want to display in the grid on your form. (To find out more about the RelationView, see “The RelationView component” on page 12-9.) RelationView components are automatically added when you use the dbNAVIGATOR or the dbAWARE Project Wizard.

The MultiView component

The MultiView object serves as both a container of RelationViews and a controller for database transactions.

All RelationViews are contained within a MultiView. For those applications that need to see multiple views of data (from the same or different database), detail RelationViews can be created using the AutoDetail object.

These detail RelationViews become part of the same MultiView that the parent Relationview is contained in.

As a container, the MultiView manages the link between a parent RelationView and all of its detail RelationViews. Methods on the MultiView are applied to all the contained RelationViews. For example, closing the MultiView would close all its RelationViews.

The MultiView also controls database transactions. All requests for RelationView data is ultimately sent to the dbANYWHERE server via the MultiView object. The MultiView is also responsible for caching all changes to data. When MultiView is saved (via a MultiView.save() or a RelationView.saveMultiView()), all data changes are sent to the dbANYWHERE server where they are finally committed to the database. All data changes between calls to MultiView.save() are committed to the database within a single transaction. Any database errors that occur during save will rollback all data changes in the database. The MultiView's cache of data changes remains unchanged allowing the error to be corrected and the changes saved.

Typical applications don't need access to the MultiView object. For that reason, the construction of the MultiView is done within the constructor of the Relationview object. There are also, helper methods, those with MultiView in their name) which are part of the RelationView class. These helper methods include: `closeMultiView`, `getMultiView`, `saveMultiView`, and `restartMultiView`.

The Projection component

Visual Cafe comes with a suite of dbAWARE visual controls that can be bound to a RelationView in the Form Designer. With the Projection object, a developer can also bind components, that are invisible at runtime, to a RelationView column.

A common code scenario follows:

```
public class Customer {

    RelationView rv;
    public Projection   name = new Projection();
    public Projection   address = new Projection();
    public Projection   phone = new Projection();

public Customer (RelationView relview) throws SQLException
```

```
{
    rv = relview;
    //Bind projection to the corresponding columns in relview
    rv.bind(rv.findProjByName("name"),name) ;
    rv.bind(rv.findProjByName("address"),address) ;
    rv.bind(rv.findProjByName("phone"),phone) ;
}
public boolean nextCustomer () throws SQLException
{
    return rv.next();
    //Projection objects now have new data
    // which can be retrieved by using any of the methods
    // in the DataAccess interface.
}
    }
```

Another Sample usage is to bind a Projection to a column to automatically capitalize on the data entered to send a state change notification.

```
public class CapState extends Projection {

    public CapState ()
    {
    }

    public init (ProjBinder binder)
    {
    super.init(binder);
    }

    public notifyDataChange (ProjBinder binder)
    {
    //get the new data value, upper case it, and set it back
    //to the binder
    binder.setString(binder.getString().toUpperCase());
    }

    public notifySetData(ProjBinder binder)
        {
    //get the new data value, upper case it, and set it back
```

```
    //to the binder
    binder.setString(binder.getString().toUpperCase());

        }

}
```

The RelationView component

The RelationView component is where you define and maintain a result set. For instance, a result set might be a set of data rows, the contents of which are a set of data produced from all the rows of employees that have a department ID of 200. The RelationView maintains that result set, handling navigation and data manipulation operations on the data in the result set. The RelationView also makes available the methods for inserting, querying, deleting, updating and scrolling through the data rows of this result set.

The RelationView object handles complex operations such as maintaining master/detail relationships with other RelationViews, caching rows of data, obtaining data on an as-needed basis, enhancing SQL performance, and providing optimistic concurrency—ensuring that data changes only take effect if other users are not changing the same data at the same time.

The RelationView object provides messaging to dbAWARE objects which allow visualization and editing of the data. When data is changed in a dbAWARE textfield, for example, the RelationView provides the mechanism by which the data is bound to the result set. The RelationView also ensures that all dbAWARE objects bound to that data are informed of the change, for immediate updating.

The RelationView is the central controller for result sets. Interactions can be made via the interaction wizard to control all the operations available in the RelationView object. For example, if you wish to have a button which causes the result set to be saved, you simply create an interaction between a button and the RelationView. Select the interaction: Save Changes. Code is automatically generated such that when the button is pressed, the RelationView saves the current changes made in the result set. SQL is generated to update only what is needed, to ensure concurrency and integrity constraints are met, and to ensure that any related RelationViews' result sets are updated in the correct order.

Note that you should add a Session component prior to adding the RelationView component. Then use the name of this component as the value for the Session property of the RelationView.

The easiest way to add a Session component is by using the Project Wizard and Add Table Wizard. You access the Project wizard by selecting the dbAWARE Project Wizard from the window that displays when you choose the File menu item New Project. You access the Add Table Wizard by selecting Add Table Wizard from the Insert menu.

The Session component

The Session component is an invisible component at runtime. The Session component defines the connection to a dbANYWHERE server, via a TCP network connection, and provides access to the related dbANYWHERE classes.

At runtime, the Session component generates code that creates a *session service* via the dbANYWHERE server. A session service detects and discards duplicate information packets in a two-way exchange of information. Each session can only service one connection or uniform resource locator (URL). You can add multiple Session components to show connections via different servers.

Note: It is recommended that all connections to a single dbANYWHERE server utilize the same Session component. This helps to reduce the number of server connections.

About the database browser, dbNAVIGATOR

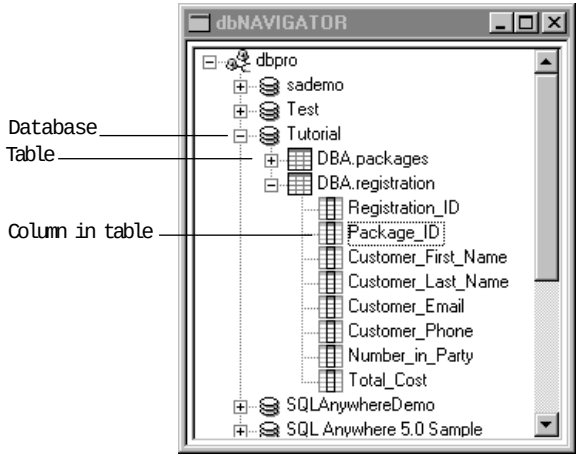
After connecting to dbANYWHERE and your database, you can use the window called the dbNAVIGATOR, to navigate across the dbANYWHERE servers to which you are connected and the databases managed by the servers.

Within dbNAVIGATOR window, you can expand and contract the data source listing and use drag-and-drop techniques to build your dbAWARE forms.

The dbNAVIGATOR window shows the databases to which you are connected. Using dbNAVIGATOR, you can view database contents in a hierarchical manner. The dbNAVIGATOR shows the databases that you are

connected to, tables within the databases, and the columns within the tables. This database information is commonly referred to as *metadata*.

In the following illustration, you can see the metadata for the Tutorial database. This database is used with the Tour in the *Getting Started with Visual Cafe, Database Development Edition*.



You can also use dbNAVIGATOR to drag database tables and/or fields onto your application forms. When you drop a database object onto your form, Visual Cafe creates the associated components and populates the component's property list with information collected from the database.

You can also use the database wizards, which step you through the process, to build dbAWARE applications.

The dbAWARE wizards

The Visual Cafe Database Development Environment wizards make it easy to quickly get to the heart of your dbAWARE application development.

You can use the dbAWARE Project Wizard to create a dbAWARE project. It provides screens where you are asked to make the choices that define your project. Using the wizard you define the data source, choose the table, specify the columns to use and add any new components and interactions. When you finish, the wizard displays the form you specified in the Visual Cafe form editor.

After you create your form, you can add more tables to it by using the Add Table Wizard.

Once you use the wizards to define the relationship between your data source and components on your applet or application, Visual Cafe automatically updates dbAWARE components in your form to match any changes in the data source.

Database manipulation functions

Visual Cafe provides a set of functions you can use to navigate and manipulate your database. Some of these are provided in buttons which you can add to forms. There are also several database methods which do not have a visible representation.

Visual Cafe includes the following database functionality:

- Next record: displays the contents of the next database record
- Previous record: displays the contents of the previous database record
- New record: creates a new record in the database
- First record: displays the first record
- Delete record: deletes the current record
- Undo record: undoes change made to a database record
- Save entire record set (save multiview): saves a set of records
- Rewind result set: go back to the beginning of the result sets
- Restart entire record set (restart multiview): starts over at the beginning of a set of records
- Goto record: goes to the record specified by an integer
- Get record state: reports the state (modified, unchanged, deleted, new) of the current record

Creating a dbAWARE project

When you are ready to develop your dbAWARE project, there are several steps you must follow before the application or applet will be able to read and store data.

- Start dbANYWHERE. See “Starting dbANYWHERE” on page 12-13 for details.

Note: There are several configurations which have to do with the location of your dbAWARE project, dbANYWHERE, and your database. Before you start the dbANYWHERE application, it must be configured to suit your needs. The configuration options are explained in Chapter 13, "Using dbANYWHERE."

- Create your database by using the tools shipped with Visual Cafe.
- A brief overview of how to do this is provided in "Creating the database for your project" on page 12-14.
- Identify the database you want to use for the project's data source by using the ODBC administrator tool. This tool is installed along with Visual Cafe and appears on the Windows Start menu.
- See "Defining your data source" on page 12-15 for more information.
- Create the dbAWARE project using the dbAWARE Project Wizard.
- See "Using the dbAWARE project wizard" on page 12-19 for details.
- Use the Add Table Wizard to add new tables from your database.
- See "Connecting to a server using dbNAVIGATOR" on page 12-30 for details.

Starting dbANYWHERE

Before your application or applet can communicate with your database, you must make sure that dbANYWHERE is running. The dbANYWHERE software, which is provided with Visual Cafe, is responsible for translating the Java database calls, which are made by your applet or application, into the appropriate database driver calls. It then sends these database requests on to your database.

If you have dbANYWHERE installed on your local machine, you can start it by using the Windows Start menu. If dbANYWHERE is installed as a Windows NT service you must start it from a command line.

To start dbANYWHERE from the Windows 95 platform:

From the Start menu, choose Programs > Symantec dbANYWHERE Server > Symantec dbANYWHERE Server.

To start dbANYWHERE from a command line:

1 Execute the a statement on your command line in the form:

```
dbaw [-options]
```

For example, the following command starts dbANYWHERE and specifies the default port: `dbaw -port 8889`

The options are:

Options	Description
`ipaddress` *a.b.c.d*	Specifies the IP Address for dbANYWHERE to use if the system has more than one.
`port` *number*	Listen on port #X instead of the default (8889)

Note: To see the dbANYWHERE command line options on the screen, invoke dbANYWHERE with a question mark:

```
dbaw -?
```

2 Determine the URL for connecting to the dbANYWHERE Server.

To see the dbANYWHERE Server's IP address and port number, look at the dbANYWHERE Server title bar.

Creating the database for your project

Before you can develop a dbAWARE project you must have a database in place for your project data. If you need to create a new database for your project, you can use:

- Proprietary database tools for the SQL Sybase, Informix, and Oracle databases.
- The Sybase SQL Anywhere 5.0 set of database creation tools, which are bundled with Visual Cafe.

The following optional procedure describes how to use the SQL Central tools to create a new database.

To create a database using the Create Database wizard:

1 From the Start menu, choose Programs > Sybase SQL Anywhere 5.0 > SQL Central.

A Navigator window opens.

2 Open the Database Utilities folder.

A list of wizards displays.

3 Double-click on Create Database.

A wizard screen opens, which is similar to this:

4 Click Browse to set up a location for your new database.

5 Click Next.

Follow the instructions in the wizard on each screen until your database is finished.

For more information, see the extensive set of online documentation provided with Sybase SQL Anywhere 5.0. In particular, the online chapter *Working with Database Object*s, presents information on creating new databases.

Defining your data source

A *data source* is what identifies a database to an ODBC-compliant database application. ODBC stands for Open Database Connectivity. It is the standard interface to database applications in the Windows 95 and Windows NT environments.

When you set a data source for your project, you are really packaging the information which enables your application to connect to the database you want to access. The data source includes details which are needed for connection, like the name of the database, its server, and its network location.

Before you can use your database in a Visual Cafe project, you must create its data source name. There are two ways to create a data source name for a Visual Cafe dbAWARE project.

If your database uses one of the “flat file” database systems such as, MS Access, SQL Anywhere, Watcom, or ODBC XBase as its driver, you must use the ODBC Administrator program to create its data source. These are the kinds of small databases you might use for a dbANYWHERE server running on your local machine.

If the database type uses any of the native dbANYWHERE drivers you must use the dbANYWHERE DataSource tool to create the data source. The native drivers for dbANYWHERE include: SQL Server, Sybase, Informix, and Oracle. These databases are more appropriate for a large system which is accessed across a network.

To create a data source using the dbANYWHERE DataSources tool:

1. If you do not have an existing database, see the previous section for information on creating a new database.
2. From the Start menu, choose Programs > Symantec dbANYWHERE Workgroup Server > Configure DataSources.

   This runs the `dsntool.bat` file. For information about where this file is on your disk, see the readme file.
3. Choose the appropriate database driver for your database.

   The necessary fields for that driver display.
4. Fill in the fields.

   For more information on the values needed for each field, see Chapter 13, “Using dbANYWHERE.”.

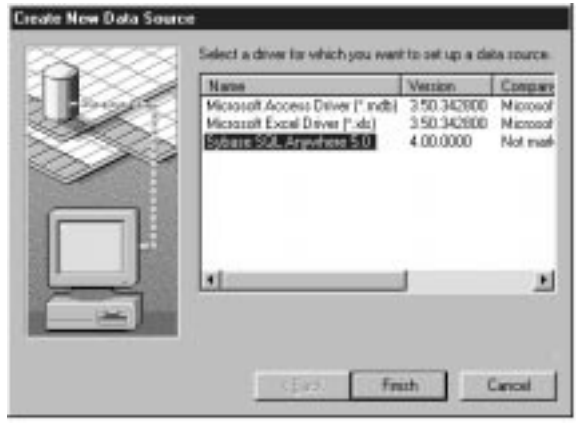
To add a data source using the ODBC Administrator:

1. From the Start menu, choose Programs > Sybase SQL Anywhere 5.0 > ODBC Administrator.

The Data Sources dialog box displays. The following illustration shows the registered Data Sources and drivers on your computer.

2 Click Add to create a new data source.

The Create New Data Source dialog box appears.

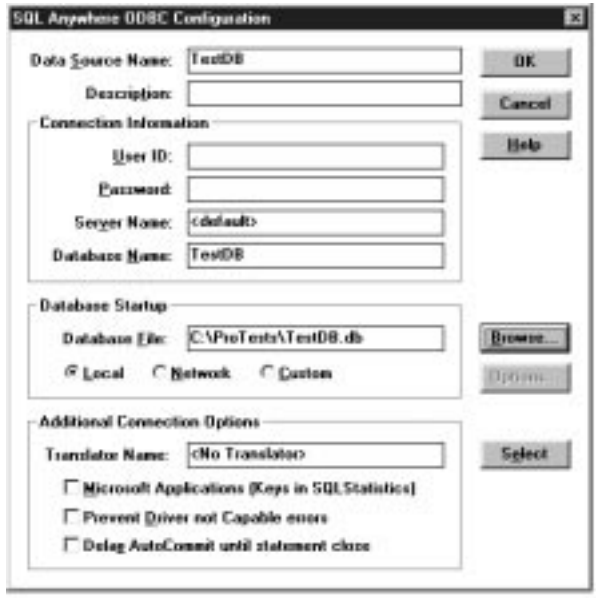


3 Select the appropriate database driver (for your type of database) from the list of Installed ODBC Drivers.

For example, if you have a Sybase database as your data source, select the Sybase SQL Anywhere 5.0 database driver.

4 Click Finish.

The appropriate ODBC Configuration dialog box appears. The SQL Anywhere ODBC Configuration dialog box is used in this example.



5 Click the Browse button to locate your database.

When you locate your database and click open, this fills in the Database File pathname, the Database Name field, and the Data Source Name.

Note: The file path is valid only for these “flat file” database types. Do not try to put a path into the dbANYWHERE DataSources tool. If you do, your data source will not work.

If your database is on your local machine, and your server software is running on your local machine, you do not need to change any other settings.

The User ID and Password are optional.

The ODBC Configuration settings are thoroughly documented in the Help file that comes with the tool. To see complete descriptions of the values needed in each field, Click the Help button.

Using the dbAWARE project wizard

The easiest way to create a database project is to use the dbAWARE Project wizard. It includes a series of interactions that allows you to identify database tables, variables in the tables, and the actions that are to be performed on that data. A button corresponds to each action and a field and label to each column in the table. The applet or application is constructed and displayed in the Form Designer.

To start the Project wizard:

1. Choose New Project from the File menu.

   A window opens and displays several wizard icons.

2. Double-click on the dbAWARE Project Wizard to open it.

   The initial dbAWARE Project Wizard screen opens.

3. Click Next. The Project Type wizard screen opens.

Setting up your project as an applet or application

The code used to generate a Java applet or application is added to your project when you set the project type using this wizard page.

To set your project as an application or applet:

1 Choose applet or application from the Project Type drop-down menu.

The Project Type wizard screen and the drop-down menu are shown in the following illustration.

2 Click Next.

The dbANYWHERE server screen displays.

Selecting the dbANYWHERE server for your project

You use this wizard page to identify your database server software. It also enables you to connect to the database you are going to use in your project.

To identify your server:

1. Type a name for your server or choose a name from the dbANYWHERE Server Name drop-down list.

   This list contains all the named dbANYWHERE servers to which you are connected. Since you don't need to name your server to connect to it, your server may not have a name. In this case, you should type in a new name for it.

2. Leave the Host Name or IP Address and the Port Number fields as they are.

3. Click Next.

   The Data Source wizard screen displays.

Identifying the data source

The Data Source wizard screen creates the data source for your project. It contains the information for connecting your applet or application to the database and server specified by it.

To identify the data source to use in your project:

1. Choose a name from the Data Source Name drop-down list.

   The list displays the registered data sources for the database server you selected previously.

2. Log on to your server, if necessary.

   If you still need to connect to your database server, the wizard screen opens the Logon dialog box that lets you logon to the database.

3. Click Next.

   If a user name and password are assigned for this data source, the User Authentication dialog box for these items displays, as shown in the following illustration.

4 Fill in the User name and Password fields. For an SQL Anywhere database the default User name is dba (for database administrator) and the corresponding password is sql.

   If the database has these values defined, you must do this before you can proceed with the dbAWARE Wizard.

5 If the database was not created by you, click on the checkbox for "Only show tables owned by this user" to deselect it. Otherwise, you are not going to be allowed to see any columns in the data source you selected.

Selecting tables

Databases can be thought of as a collection of tables. The tables in the database you selected earlier are displayed on this wizard screen. To make a table and its data available to your application or applet select the table(s) from the list and then click Next.

Choosing database columns for your application to use

Typically, each column of a database contains a single kind of information. For instance, once column of a customer database might contain customer identification numbers while another column contains a name, and yet another contains purchase information.

The Database Columns wizard page lets you choose the database columns that you want to access in your applet or application. The list displays columns from the previously selected database table.

To select a database column:

1 Click the column's check box.

2 Click Next.

The columns is selected and the Components and Labels wizard screen is displayed.

Selecting the components to apply to your table

The Components and Labels wizard page lets you select a component and specify a label for each database column. At runtime, the applet or application displays each column's data in the component assigned to it and displays the label for the component. The wizard displays a default component and label for each column.

Note: You can change the defaults. See "Setting the Database Environment Options" on page 12-2.

To change a column type:

1. Click on the component in the Component field.

   A drop-down menu list displays.

2. Choose a component from the drop-down list.

   The component name displays in the Component list.

To change a label:

1. Click on the text in the Label column.
2. Enter the new text.
3. If you don't want the component to have a label, click on the text for the label and delete it.

   Tip: You can change the component's label after it is built by modifying the component's Text property.

4. Click Next.

   The Actions Wizard screen opens.

Adding database function buttons to a form

The Actions wizard screen asks you to select the database functionality that your applet or application will provide. At runtime, the form displays a button for each action. The choices provided are listed for you in "Database manipulation functions" on page 12-12.

You can also create an action button outside of the wizard, by adding a button component to the form. With the button component selected,

choose the Add Interaction option from the right-mouse drop-down menu. Use the Interaction Wizard to specify the appropriate action for the button.

To select database buttons for your form:

1 Click the checkbox for each action you want enabled on the form.

If you want to change the text label of any button, click on the text in the Text column of the button you want to change and enter the text for the button.

2 Click Next.

The Finish Wizard screen displays.

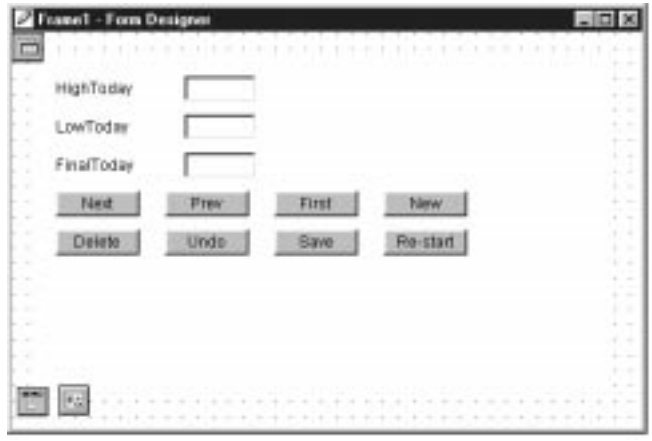
Tip: You can use the Property List to change properties of an action button after it is generated by the wizard.

Reviewing your project settings

The Finish wizard screen opens the choices you made while using the wizard to set up your applet or application.

If you need to change a selection, click the Back button to go back to the wizard page where you made that selection.

Click Finish when you're done reviewing your work. When you do this, the Form you specified displays in the Visual Cafe Form Designer.



Using the Add Table wizard

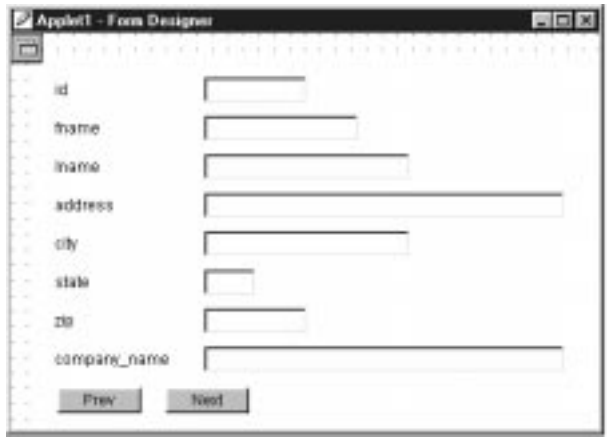
You will probably need to assign more than one table to a project. After you have a project open, you use the Add Table wizard to help you add new tables to your project. And, when your project already has a RelationView, you can use the Add Table wizard to create a master/detail relationship.

To add a new table using the Add Table wizard:

1. From the Insert menu, choose Add Table Wizard.

   The dbAWARE Project wizard displays.
2. From the dbANYWHERE Server drop-down menu, select the dbANYWHERE server you have running.
3. From the Data Source Name drop-down menu, select the SQL Anywhere 5.0 Sample
4. Click Next.
5. Choose a Database Table from the list displayed by selecting DBA.customer.
6. Click Next.
7. From the Database Columns list, click on the phone checkbox to deselect it.
8. Click Next.
9. Leave the default values in the Component and Label columns of the Choose Components and Labels dialog.
10. Click Next.
11. Select the Previous and Next actions in the “Choose Database Operations” dialog box.
12. Click Next.

The following form displays in the Form Designer.



13 From the Project menu, choose Execute.

The applet starts up and runs.

14 Click either the Prev or Next buttons to see the database records.

Using dbNAVIGATOR in form development

The dbNAVIGATOR lets you access dbANYWHERE Servers, and the databases that are managed by the servers. When you connect to a dbANYWHERE Server, dbNAVIGATOR displays a list of servers and Data Sources that are currently connected.

Using the dbNAVIGATOR window, you can quickly perform many database functions by dragging specific database fields from the dbNAVIGATOR window and dropping them onto your form in the Form Designer.

Visual Cafe then creates the appropriate dbAWARE component, according to the data type of the field. (You can change these default settings. See "Setting the Database Environment Options" on page 12-2, for more information.) Visual Cafe also generates the component properties by basing the settings on the database field's design. In addition, Visual Cafe generates the Java code to bind the component to the database field.

There are many database functions you can carry out by using the dbNAVIGATOR window. Some of the tasks are:

- Connecting to a dbANYWHERE Server

- Connecting to a data source
- Adding a RelationView to a form
- Disconnecting from a data source
- Updating the dbNAVIGATOR window

Connecting to a server using dbNAVIGATOR

In order to access a data source, you must first connect to a dbANYWHERE Server that has the data source registration.

To connect to the dbANYWHERE server using dbNAVIGATOR:

1. Choose Window > dbNAVIGATOR.

   The dbNAVIGATOR window displays.
2. If you display dbNAVIGATOR and it is empty, Visual Cafe opens the dbANYWHERE Server dialog box. See "Selecting the dbANYWHERE server for your project" on page 12-21 for information about these settings.
3. Use the dbANYWHERE Server dialog box to connect to a dbANYWHERE Server. Make sure that dbANYWHERE is running or this won't work.

Connecting to a database

Once you are connected to a server using dbNAVIGATOR, you can see all the data sources registered with that server.

To connect to a database:

1. Install the Sybase SQL Anywhere 5.0 product, if it is not already installed, so that you have access to the sample databases.
2. Start and connect to your dbANYWHERE server.
3. Click the plus sign next to the SQL Anywhere 5.0 Sample in the dbNAVIGATOR window to see its object list.

   The User Authentication dialog box displays.
4. Enter the default User name and Password values, dba and sql, respectively.
5. Click OK.

   The tables included in the SQL Anywhere 5.0 Sample displays in an indented list under the database name.

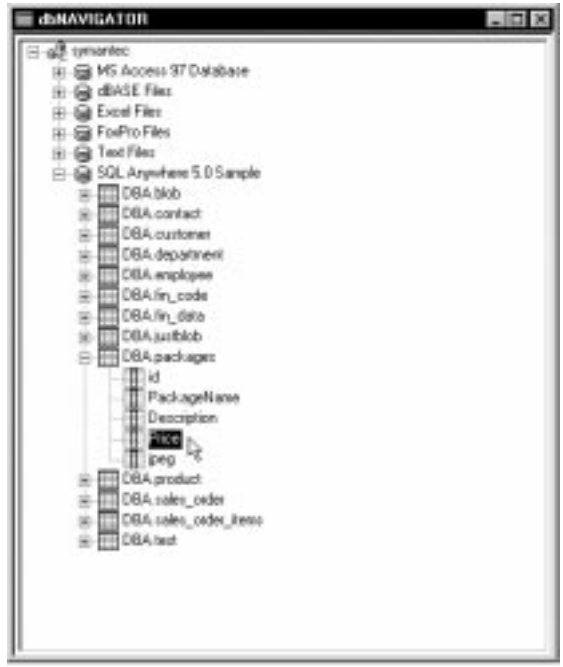
Adding a dbAWARE text field to a form

You can use the dbNAVIGATOR window to add a text field which reflects the contents of a database column to a pre-existing form.

To add a TextField, which is associated with a database column, to a form:

1. In dbNAVIGATOR, expand the desired server's data source object list.
2. Click the plus sign in the dbNAVIGATOR window to see the object list of the data source.
3. Expand the object list for the table that contains the desired column.

   This picture shows the expanded table object listing for the SQL Anywhere 5.0 Sample database.



4. Drag the appropriate column item to the Form Designer.

Visual Cafe Pro adds a TextField and Label component to the form. The label component uses the column name for the Text property. If the field is from a table where the necessary dbAWARE components (like the Session, ConnectionInfo, and RelationView components) are not yet created, then those supporting components are created for you.

5 Position and size the Text field and label as necessary.

Refreshing dbNAVIGATOR

Refreshing dbNAVIGATOR updates dbNAVIGATOR's display for a data source's tables and columns. Refreshing dbNAVIGATOR is useful for multiple concurrent users.

1 From the Window menu, choose dbNAVIGATOR.

2 Select a data source by clicking on a DataSource item.

3 Right-click to display the dbNAVIGATOR contextual pop-up menu.

4 Choose Refresh from the menu.

The screen updates itself and redisplays.

Adding a database grid component

The easiest way to add a grid component to a form is from the dbNAVIGATOR window. You can add your grid component, in a few steps, by using drag-and-drop functionality.

To use dbNAVIGATOR to add a grid component:

1 Drag a table name from the dbNAVIGATOR window onto your form.

When you do this, Visual Cafe automatically creates all the other components, which are needed to define the database connection, such as the RelationView, Session, and ConnectionInfo components.

The illustration below is a screen shot of the dbNAVIGATOR window which is showing the columns in the Visual Cafe Tutorial's

DBA.registation table. When the columns display in a Grid, they appear in the same order you see them in the dbNAVIGATOR window.

2 Select the Grid component from the Component Palette, drag it to the Form Designer, and drop it on the Form Designer.

The Grid display is empty. You do not see the actual data until runtime.

3 Size the Grid component, as needed.

If query columns do not fit within the Grid component, a scroll slider is provided.

4 Adjust the column widths by positioning the cursor on the column heading until a double-edge arrow displays. Then, click and drag the column border line left or right.

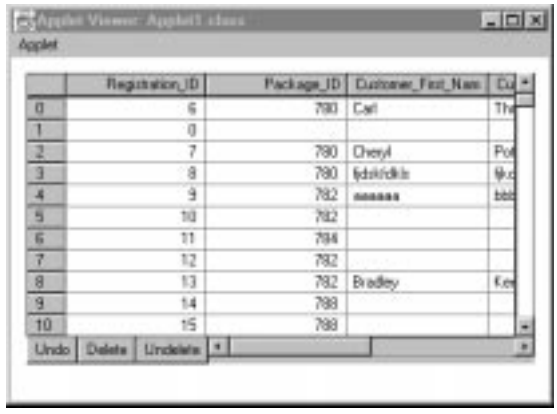
Note: Column heading text can be customized in code only.

---

5 Set the Grid component's RelationView property to the name of the RelationView component that was created automatically in step 1. The RelationView property is an element of the Binding property.

6 Run the form by choosing Execute from the Project menu.

The following illustration shows the sample Grid at runtime. Notice the placement of the Grid buttons. The Grid component uses the database column names as the Grid column headings.



Disconnecting from a database

When you finish your session you should disconnect from the database.

To disconnect from a data source:

1 Choose Window > dbNAVIGATOR.

2 Choose a data source by clicking on a DataSource item.

3 Right-click to display the dbNAVIGATOR pop-up menu.

4 Choose Disconnect.

Changing grid attributes

Once you have a grid in place on your form, you may decide to change its visual attributes. To do this you must modify the source code. The grid attributes you can change include:

- Foreground and background cell colors
- Cell fonts

- Protection status
- Column attributes

The following sections give examples of how to modify the grid attributes.

Important: You must place any custom source code after the `INIT_CONTROLS` code block.

Changing foreground and background cell colors

The Property List does not provide a property which controls the foreground and background colors of Grid cells. You must specify these attributes in code.

Below is a code sample that changes the cell background and foreground colors. The `setCellBgColor` and `setCellFgColor` methods take as arguments a row number, a column number, and a new color.

Note: Cell row and cell column numbers start with 1 (not 0).

```
grid1.setCellBgColor(1, 1, Color.blue);
grid1.setCellFgColor(1, 1, Color.white);
```

You can also change the color of an entire row or column of cells. The arguments for the methods that do this consist of a row (or column) number and a color.

```
grid1.setColBgColor(1, Color.blue);
grid1.setColFgColor(1, Color.white);
grid1.setRowBgColor(1, Color.red);
grid1.setRowFgColor(1, Color.white);
```

Changing cell fonts

You may want to change the font of a cell or range of cells for emphasis. Below are some code snippets that illustrate how to change cell, column and row fonts:

```
grid1.setCelllFont(2,2,new Font(“Helvetica”,Font.PLAIN,14));
grid1.setColFont(1, new Font (“TimesRoman”,Font.ITALIC, 18);
grid1.setRowFont(1, new Font (“Courier”, Font.BOLD, 16)
```

The arguments are the column number, a color as defined in the `java.awt.Color` class, and a font size.

Changing Grid column attributes

You can change attributes of Grid columns and protect sections of the grid. These attributes, which must also be changed in the code, include:

- Protecting sections of the grid
- Heading names
- Column alignments
- Heading colors and fonts

Protecting grid sections

You can protect cells, so that the user cannot modify the data, by enhancing the component's Java code. Like Grid cell colors, you can protect specific cells, a whole row of cells, or a whole column.

Below are some examples:

```
grid1.setCellEditable(1, 1, false);
grid1.setColEditable(2, false);
grid1.setRowEditable(2, false);
```

In the above syntax, the first argument is the column number and the second argument is a Boolean. The false Boolean value indicates a read-only row, column, or cell and the true value indicates a read-write row, column, or cell.

Changing Column Headings

You can change column headings by adding custom code, as follows:

```
grid1.setHeading (“Last Name”, 1, 15);
grid1.setHeading (“First Name”, 2, 15);
```

The third `setHeading` argument specifies the width of the column in average character widths.

Changing Column Alignment

You can also modify column alignment. The method arguments are the column number and the alignment type. The alignment can be specified by static final `int` values attached to the Grid class:

```
grid1.setColumnAlignment (1, grid1.LEFT);
grid1.setColumnAlignment (2, grid1.CENTER);
grid1.setColumnAlignment (3, grid1.RIGHT);
Changing Column Heading Colors And Font
```

You can modify column heading and font attributes. The `setHeadingColor` method changes the foreground and background colors of a column heading. The `setHeadingFont` method changes the font of a column heading:

```
grid1.setHeadingColors (1, Color.black, Color.white);
grid1.setHeadingFont (new Font("TimesRoman", Font.BOLD, 18), 1);
```

Defining automatic Grid row numbering

Automatic row numbering can be turned on or off. When activated, the starting row number can be set by using the `setupAutonumbering` method, for example,

```
grid1.setupAutonumbering (1);  //Begins row numbering at 1.
grid1.setupAutonumbering (5);  // Begins row numbering at 4.
grid1.setupAutonumbering (0);  //Turns off auto-row numbering.
```

Defining automatic Grid redraw

Each time a grid component is updated, it is automatically redrawn. If you are going to make several changes to your grid (column headings, cell color changes, and so forth) it is a good idea to turn off the automatic redraw capability. Once you have made all your changes, you can turn it back on again by using the following code snippet:

```
grid1.setAutoRedraw (false);  // Turns off auto-redraw
                             // Make changes to the grid here
grid1.setAutoRedraw (true);   // Turns on auto-redraw
```

Modifying the Grid toolbar

You can add additional functionality to a Grid toolbar by

- Creating a customized Grid event handler with the added functionality

- Informing the Grid about the new event handler

Each grid component has a `DefaultTvEventHandler` object that provides the standard functionality of the Grid: Insert, Go To, Undo, Restart, Delete, Undelete, and Save. Additional functions are made through a custom EventHandler object that extends from the `DefaultTvEventHandler`.

CHAPTER 13

Using dbANYWHERE

Java forms communicate with a data source through code calls to server software which generates the code for communicating with a database. The Symantec Visual Cafe Database Development Edition includes database deployment software (also known as middleware) called dbANYWHERE.

In this chapter you learn:

- About dbANYWHERE
- How to use the dbANYWHERE components

About dbANYWHERE

This section discusses how dbANYWHERE fits into your dbAWARE Java applet or application development scheme.

What is dbANYWHERE?

You use the dbANYWHERE Server software to drive your database connection. It comes with special data source connectivity components and wizards to help you configure the server connection. When you use the Visual Cafe Database Development Edition wizards and components, Visual Cafe automatically generates Java code that includes calls to the dbANYWHERE API.

The dbANYWHERE workgroup server database-deployment software manages transactions between a client's Java applet or application and one or more databases. Using dbANYWHERE, a Java applet or application from

a client Web browser transparently connects to the dbANYWHERE server, and the server in turn connects to the target databases.

The dbANYWHERE server then translates the information to the specific driver needed for you database to be able to interpret the calls from your applet or application. Conceptually dbANYWHERE sits between your applet/application and the database. It handles the necessary database drivers and as such, can be viewed as a database middleware server.

Since dbANYWHERE provides the database configuration data and other database-specific information, the Java applet or application which is the database client can be database-independent. When you use dbANYWHERE, the necessary database drivers are handled by dbANYWHERE. This means that you don't need to install any database libraries with the client application or applet.

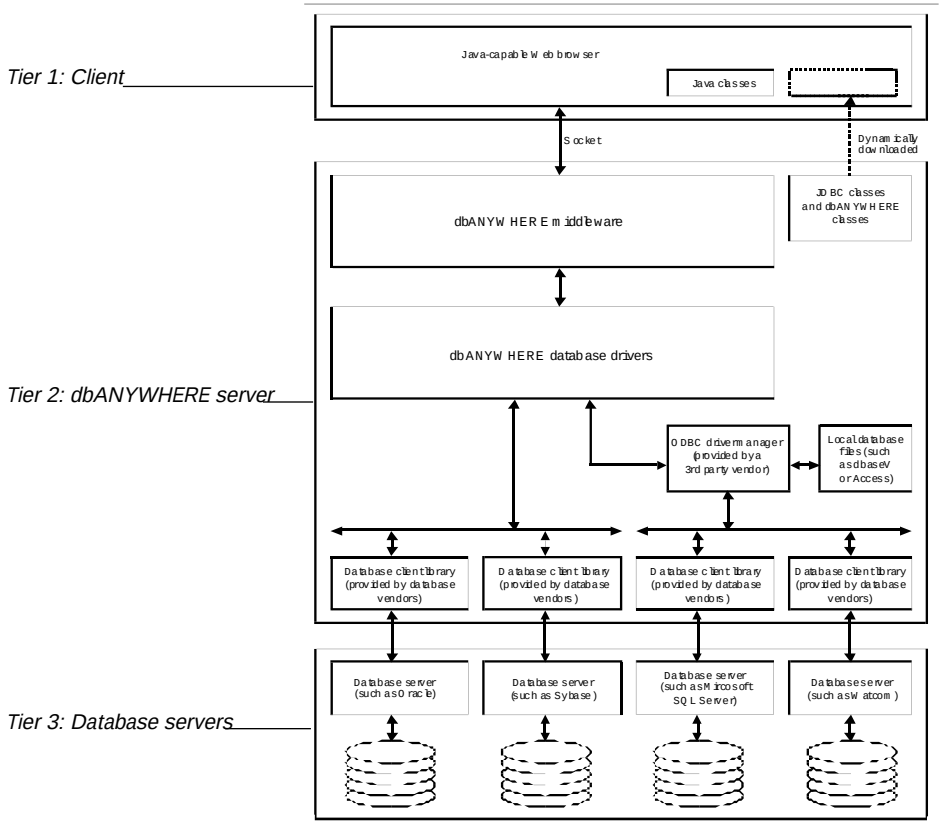
How the dbANYWHERE architecture works

The dbANYWHERE middleware software supports a 3-tier architecture for database access. This layered architecture refers to the database client, the dbANYWHERE server, and the database server. These tiers can exist together on one computer, or may be set up so that each one is on a separate machine. The benefits of having specific locations of each logical part of the server configuration is described in "Configuring dbANYWHERE" on page 13-3.

The three layers or tiers are:

- First tier: the client. A dbANYWHERE client consists simply of a TCP/IP network configuration. In order to run a Java applet and to download classes dynamically, a client also requires a standard Internet Web browser.
- Second tier: the dbANYWHERE server. This tier handles the database APIs and the ODBC configuration. The database client software and network software (TCP/IP) must also exist on this machine.
- Third tier: the database server(s). The Network software (TCP/IP) must also exist on this machine.

The following diagram illustrates the 3-tier architecture.



Configuring dbANYWHERE

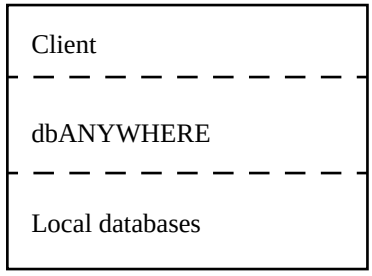
You can set up the three tiers on different computers, or combine any or all of the tiers on a single computer. For example, during development you may want to place all 3 tiers on one machine. However, when deploying large applications you will typically dedicate one or more machines to each tier. The dbANYWHERE runtime configuration is also tied to the number of users and the servers you support.

This section describes common ways to configure the dbANYWHERE tiers:

- One machine for local databases
- Two machines for remote databases
- Two machines for remote and local databases
- Three machines for remote databases and remote dbANYWHERE server.

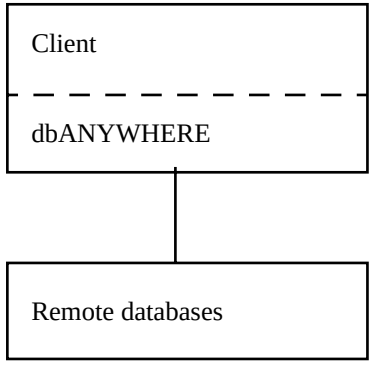
One machine for local databases

For local databases during development, you may want to use a single machine in which the client, dbANYWHERE, and the databases reside on one computer.



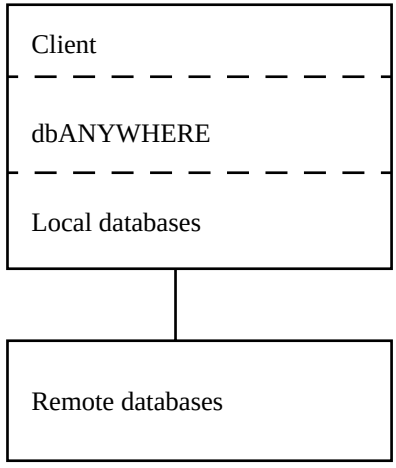
Two machine configuration for remote databases

For remote databases, you can use a two machine configuration in which the client and dbANYWHERE are on one machine and the databases are on other machines.



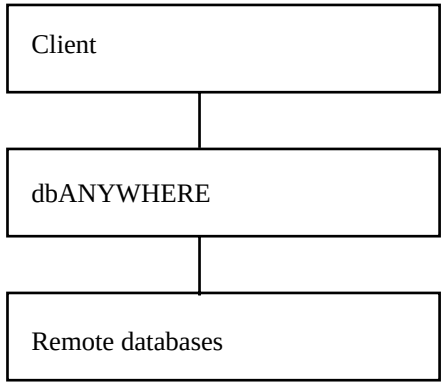
Two machine configuration for local and remote databases

For a combination of local and remote databases, you can use a two machine configuration in which the client, dbANYWHERE and the local databases reside on one machine, and the remote database reside on other machines.



Three machine configuration

You can place each tier on a separate machine. In this configuration, the client, dbANYWHERE, and the databases reside on separate machines. However, you are not limited to three machines. Additional databases can reside on other machines.



Using the dbANYWHERE packages

Visual Cafe Database Development Edition provides two Java packages which the Java client application or applet uses to "talk" to the dbANYWHERE middleware software. They are the `symjava` (JDBC) package and the `dbaw` (dbANYWHERE) package. The `symjava` package provides the standard Java classes which define the JDBC API. The `dbaw` (dbANYWHERE) package provides the classes that implement the JDBC API.

You can dynamically download the packages to the client or install the packages on the client to avoid the administration problems of installing the packages on each client. The client's Web browser handles this automatically by downloading the necessary classes as they are needed by the Java applet.

The following table lists the client programs and which package they use:

For client programs using:	Include:
JDBC API	`dbaw.zip`
dbANYWHERE API	`dbaw.zip` `dbaw_awt.zip`

Note: The sql.zip file which shipped with prior versions of Visual Cafe are no longer included. These files are now part of JDK 1.1 and are included in the corresponding class file.

The following is a list of the contents of each `.zip` file:

- `dbaw.zip`: Set of dbANYWHERE classes. These classes implement the JDBC classes that are only defined but not implemented in Sun's package.
- `dbaw.exe`: The dbANYWHERE executable.
- `dbawservice.exe`: The dbANYWHERE Service executable (NT only)
- `dbawut.dll`, `dbawdm.dll`, `dbawkrnl.dll`, `dbawserv.dll`, `dbawnbr.dll`, `dbawprops.dll`, `dbawlicense.dll`, `dbawdt.dll`: Support DLLs.

- ddaccess.dll, ddodbc.dll, ddsybany.dll, ddsybnt.dll, ddsybwin.dll, ddorant.dll, ddorawin.dll, ddmssql.dll, ddinfmx.dll, ddoracle.dll: Database driver DLLs.
- dbawlic.txt, ddinfmx.sup, ddmssql.sup, ddsybase.sup, dbaw_out.zip: Additional database files.
- dbawdrvr.ini: Configuration file.

Using the dbANYWHERE packages for development

Once you determine which dbANYWHERE packages you require for your applet or application you need to set them up on your development machine.

To set up the packages on your development system:

1. Copy the .zip files to the development machine.
2. Modify your CLASSPATH environment or application variable to point to the directory that contains the .zip files.
3. Modify the program's CLASSPATH statement in sc.ini. For example, if you put the .zip files in a directory called zipFiles:

   ```
   CLASSPATH=c:\zipFiles\dbaw.zip;c:\zipFiles\dbaw_awt.zip;
   ```

4. Add the pathnames for the dbANYWHERE files sql.zip and dbaw.zip files to CLASSPATH in c:\Visual Cafe\bin\sc.ini. This is important so that the Visual Cafe wizards will work. For example:

   ```
   CLASSPATH=...;c:\dbaw\sql.zip;c:\dbaw\dbaw.zip
   ```

Note: Visual Cafe Database Development Edition reads sc.ini only during launch. If you edit sc.ini while Visual Cafe Database Development Edition is running, restart the program for the changes to take effect.

Setting up the dbANYWHERE packages for a deployed application

We recommend providing the dbANYWHERE packages as unzipped files and setting up the files for automatic downloading. When you use this configuration you don't need to install packages onto the client. In addition, your applet or application downloads only the required classes.

To set up the dbANYWHERE packages as unzipped files for automatic downloading:

1. Copy the files to the directory of your applet or application. For example, if your applet is called `dbapplet.class` and is in `c:\client\dbapplet`, you would have the following files:

   ```
   c:\client\dbapplet\dbapplet.class
   c:\client\dbapplet\dbaw.zip
   c:\client\dbapplet\dbaw_awt.zip
   ```

2. Unzip the files and delete the `.zip` files. Retain the complete directory structure for the unzipped files.

   For example, if your applet is called `dbapplet.class` and is in `c:\client\dbapplet`, you would have the following files:

   ```
   c:\client\dbapplet\dbapplet.class
   c:\client\dbapplet\symantec\itools\db\jdbc\*.class
   c:\client\dbapplet\symantec\itools\db\net\*.class
   c:\client\dbapplet\symantec\itools\db\pro\*.class
   c:\client\dbapplet\symantec\itools\awt\image\*.class
   c:\client\dbapplet\symantec\itools\db\awt\*.class
   ```

Configuring dbANYWHERE as a Windows NT service (Windows NT only)

On Windows NT, you can run dbANYWHERE as a console or as a service. On Windows 95, you must run dbANYWHERE as a console. During setup, dbANYWHERE is configured to run in console mode.

There are several advantages to running dbANYWHERE as a service:

- You can configure dbANYWHERE to start automatically when the system boots.
- The System Event Log will track key actions.
- The Windows NT Performance Monitor can be used to monitor the dbANYWHERE Server.
- You can start and stop the dbANYWHERE service from remote machines.

To install the dbANYWHERE Windows NT service:

1. Start the dbANYWHERE Service Manager. To do this, double-click the icon in the dbANYWHERE Program group or type "`dbawmgr`" from the command line in the directory where the `dbawmgr.exe` file exists.
2. Go to the Options menu and select Install as Service.
3. Click OK.
4. The dbANYWHERE service is now installed.

To configure the dbANYWHERE Windows NT service:

1. If not already running, start the dbANYWHERE Service Manager.
2. From the Options menu, choose Properties.
3. If desired, select Autostart on System Boot. Otherwise, you must start the service manually.
4. Set the log-on values. For an ODBC-enabled database select "This account" and enter your Windows NT user account number and password. If you have selected automatic mode, Windows NT will automatically log in for you when the system boots.
5. Change other settings as needed. These are the same as the command line options. For example, if you want dbANYWHERE to use port 7864, select the Network tab and change the port number to 7864. Under the Logging tab, be sure to leave "Log to Event Log" enabled so that you can diagnose any startup problems.
6. Restart the server for the changes to take effect.

Running the service

Click the START button. Within a few seconds, the status should change to Running. At this point, you can try to connect from a client application.

If status changes briefly to "Start Pending", and then to "Stopped", check the Windows NT Event Viewer for problems.

Viewing messages

When running as a service, dbANYWHERE sends informational, warning, and error messages to the Windows NT Event Log.

To view these messages:

1. Start the Event Viewer. To do this, double-click Event Viewer in the Administrative Tools group or choose Options/Launch Event View in the dbANYWHERE Service Manager.
2. Choose the application log. When you do this, Event Viewer lists all the messages in the application log.
3. To get more information about a specific message, double-click on the message. When you do this, Event Viewer displays a dialog box that provides detailed information about the message.

Connecting to a data sources

You must set a data source for a database before it can be used by an application or applet. The data source contains the information which is necessary to locate the database. Once you have set a data source for your database you can set your application or applet to connect to the different kinds of databases, as described in the following sections.

Note: The URLs in this section are for JDBC and begin with: `jdbc:dbaw://`. The URL for the corresponding Visual Cafe Database Development Edition is the same except that the "`jdbc:`" is not included. The syntax is `dbaw://`.

Connecting to a Sybase SQL Anywhere and Watcom data source

This procedure describes how to connect to Sybase SQL Anywhere by using JDBC and the Visual Cafe Database Development Edition APIs as a direct ODBC Data Source using the ODBC driver that dbANYWHERE provides. To use SQL based ODBC or Xbase ODBC and a third-party ODBC driver, see:

- "Connecting to an ODBC SQL-based data source by means of JDBC" on page 13-12
- "Connecting to an ODBC Xbase data source by means of JDBC" on page 13-13

To connect to Sybase SQL Anywhere by means of JDBC:

1. Make sure you have access to Sybase SQL Anywhere or Watcom.

2. Make sure you have access to the Sybase SQL Anywhere or Watcom ODBC32 driver.
3. Use the Windows Control Panel ODBC32 Administrator to create and install the ODBC Data Source to the desired database on the dbANYWHERE Server machine.
4. Set the URL, as shown in the following examples.

   The Server and Data Source parameters in the Data Source URL need to be the same as the ODBC Data Source name defined on the dbANYWHERE Server machine. For example:

   To connect to a Sybase SQL Anywhere database with an ODBC Data Source named “Tutorial”:

   ```
   jdbc:dbaw://localhost:8889/Sybase_SQLANY/
    Tutorial/Tutorial
   ```

   To connect to a Sybase SQL Anywhere database with the default ODBC Data Source named SQL Anywhere 5.0 Sample:

   ```
   jdbc:dbaw://localhost:8889/Sybase_SQLANY/SQL Anywhere
    5.0 Sample/SQL Anywhere 5.0 Sample
   ```

   To connect to a Watcom 4.0 database with an ODBC Data Source named Sample:

   ```
   jdbc:dbaw://localhost:8889/Watcom/Sample/Sample
   ```

Connecting to an Informix data source

You can connect to an Informix data source by installing the Informix SQL Server 32-bit client software.

To connect to an Informix data source:

1. Install the Informix SQL Server 32-bit client software on the dbANYWHERE Server machine.
2. Set the URL.

Connecting to Microsoft Access data source by means of JDBC

This procedure describes how to connect to Sybase SQL Anywhere by means of JDBC and Visual Cafe Database Development Edition APIs as a direct ODBC Data Source using the ODBC driver that dbANYWHERE provides. To use SQL based ODBC or Xbase ODBC and a third-party ODBC driver, see:

- “Connecting to an ODBC SQL-based data source by means of JDBC” on page 13-12

- "Connecting to an ODBC Xbase data source by means of JDBC" on page 13-13

To connect to Microsoft Access data source by means of JDBC:

1. Make sure you have Microsoft Access and the Microsoft Access ODBC32 driver.
2. From the Start menu, choose Programs, then Sybase SQL Anywhere 5.0, then ODBC Administrator
3. Create and install an ODBC Data Source for the desired database on the dbANYWHERE server.

   If you need directions, use the online documentation provided with the product.
4. Set the URL, as shown in the following example.

   The Server and data source parameters in the data source URL must be the same as the ODBC data source name defined on the dbANYWHERE server. For example, the following URL connects to a Microsoft Access ODBC data source named NorthWind:

   ```
   jdbc:dbaw://localhost:8889/MS_Access/NorthWind/NorthWind
   ```

Connecting to an ODBC SQL-based data source by means of JDBC

This procedure describes how to connect to an ODBC SQL-based data source by means of JDBC and Visual Cafe Data Development Edition APIs by using a third-party ODBC driver.

To connect to an ODBC SQL-based data source:

1. Install the third-party 32-bit ODBC driver pack on the dbANYWHERE Server machine.
2. Install an ODBC Data Source on the dbANYWHERE Server machine. You can use the ODBC32 Administrator to create the ODBC Data Source.
3. Install client software on the dbANYWHERE Server machine.
4. Set the URL, as shown in the following example.

   The Server and Data Source parameters in the Data Source URL must be the same as the ODBC Data Source name defined on the dbANYWHERE server. For example, the following URL connects to a Microsoft SQL Server ODBC Data Source named SQLSrvDSN:

   ```
   jdbc:dbaw://localhost:8889/ODBC_DSN/SQLSrvDSN/SQLSrvDSN
   ```

Connecting to an ODBC Xbase data source by means of JDBC

This procedure describes how to connect to an ODBC Xbase Data Source by means of JDBC and Visual Cafe Database Development Edition APIs using a third-party ODBC driver.

To connect to an ODBC Xbase Data Source by means of JDBC using a third-party ODBC driver:

1. Install the third-party 32-bit ODBC driver pack on the dbANYWHERE Server machine.
2. Install an ODBC Data Source on the dbANYWHERE Server machine. You can use the ODBC32 Administrator to create the ODBC Data Source.
3. Install client software on the dbANYWHERE Server machine.
4. Set the URL, as shown in the example following.

   The Server and Data Source parameters in the Data Source URL need to be the same as the ODBC Data Source name defined on the dbANYWHERE Server machine. For example, the following URL connects to a dBase ODBC Xbase Data Source named DbaseFile:

   ```
   jdbc:dbaw://localhost:8889/ODBC_Xbase_DSN/DbaseFile/DbaseFile
   ```

Connecting to Microsoft SQL data source by means of JDBC

This procedure describes how to connect to a Microsoft SQL data source by means of JDBC.

To connect to a Microsoft SQL server:

1. Install the Microsoft SQL Server 32-bit client software on the dbANYWHERE Server machine.
2. Set the URL, as shown in the following example, by using the driver name: `SQL_Server`.

To connect to a Microsoft SQL server, the server parameter in a data source URL needs to correspond to the name of the SQL Server. For example, to connect to a Microsoft SQL Server named MySQLServer and a database named pubs, the Data Source URL server parameter is:

```
jdbc:dbaw://localhost:8889/SQL_Server/MySQLServer/pubs
```

Connecting to Sybase SQL data source by means of JDBC

To connect to a Sybase SQL server, the Server parameter in the data source URL needs to correspond to the name of the Sybase SQL server defined in the `sql.ini` file. You can use the vendor-supplied SQLEDIT tool to generate or view configuration files.

The Sybase SQL server ships two different sets of Open-Client/NetLibrary software to connect from Windows 95/NT clients.

To connect to a Sybase SQL server:

1. Depending on your client type, install the appropriate 32-bit client software on the dbANYWHERE Server machine.
2. Set the URL by using `Sybase_NT` or `Sybase_Win95` as the driver name.

   Use the data driver name that corresponds to your platform software, as shown in the following examples:

   To connect to a Sybase SQL server named "MySybServer" and a database named "sample" on Windows NT:

   ```
   jdbc:dbaw://localhost:8889/Sybase_NT/MySybServer/sample
   ```

   To connect to a Sybase SQL server named "MySybServer: and a database named "sample" on Windows 95:

   ```
   jdbc:dbaw://localhost:8889/Sybase_Win95/MySybServer/
    sample
   ```

Connecting to Oracle data source by means of JDBC

SQLNet is the Oracle client software for connecting to remote databases from client machines. Oracle Server ships different sets of SQLNet client software for Windows 95 and Windows NT.

To connect to an Oracle data source:

1. Install the appropriate client software on the dbANYWHERE Server machine.

   Note: dbANYWHERE supports several Oracle protocols. Refer to the Oracle Server documentation for more information about the different protocols and compatibility between them.

2. Set the URL according to the following examples.

The Server parameter in the Data Source URL needs to have the format and values described for the dbANYWHERE Data Source configuration file's server name parameter. For example:

To connect to an Oracle Server database named "ORCL" running at "oraserver.com" using SQLNET V1:

```
jdbc:dbaw://localhost:8889/Oracle_7/T:oraserver.com/
 ORCL  (TCP/IP)
jdbc:dbaw://localhost:8889/Oracle_7/P:oraserver.com/
 ORCL  (Named Pipes)
jdbc:dbaw://localhost:8889/Oracle_7/X:oraserver.com/
 ORCL  (SPX/IPX)
jdbc:dbaw://localhost:8889/Oracle_7/B:oraserver.com/
 ORCL  (NetBIOS)
```

To connect to an Oracle Server database named "ORCL" using SQLNET V2 with a service name "ORASERV.WORLD" specified in the file, `tnsnames.ora`:

```
jdbc:dbaw://localhost:8889/Oracle_7/TNS:ORASERV.WORLD/
 ORASERV.WORLD
```

To connect to a Personal Oracle7 database named "ORCL" running on the dbANYWHERE Server machine:

```
jdbc:dbaw://localhost:8889/Oracle_7/2:ORCL/ORCL
```

Testing a data source connection

There are three different ways you can use Visual Cafe to test your data source to see if it is going to be successful in connecting with the server and database it specifies.

To test a data source connection using the Test Data Sources command:

1. From the Help menu, choose Test Data Sources.
2. Select a Data Source from the list. The list consists of the Data Sources in the dbANYWHERE Data Source configuration file.
3. (Optional) Enter a user name and password. If you don't enter these values, the test uses the user name and password defined in the DSN section in the dbANYWHERE data source configuration file.
4. Click Test.
5. After a few moments, the test displays the test results.

To test a data source's connection using the DataSource tool:

1. Start the Data Source Tool. See "Using the DataSource tool" on page 13-16 for more information.
2. Select a Data Source from the Defined Data Sources list.
3. Click Test.

   This displays the Test dialog box.
4. Click PRO API Test or JDBC API Test.

To troubleshoot when a test fails:

1. Check the Data Source's user name and password.
2. Check the Data Source URL.
3. Check the Data Source's information in the dbANYWHERE Data Source configuration file. To modify this file, you can use a text editor or the Data Source Tool.

For more information, see "Testing a data source connection" on page 13-15.

Using the dbANYWHERE tools

You can use the dbANYWHERE tools to:

- Manage the dbANYWHERE Windows NT Service, covered in an earlier section
- Create and edit data sources
- View events
- Monitor server link performance
- Allow remote administration

Using the DataSource tool

The DataSource tool displays the Data Source names, (if there are any) which have been saved in the dbANYWHERE Data Source configuration file.

You can use the Data Source Tool to:

- Create new data sources
- Edit existing data sources

- Test the validity of the data source connection information

To use the DataSource tool, you must have the Java Virtual Machine installed on your system.

To start the Data Source tool:

From the Start menu, choose Programs, then Symantec dbANYWHERE Workgroup Server, then Configure DataSources.

This runs the `dsntool.bat` file. For information about where Setup installs this file, see the `readme` file.

Fields

Defined Data Sources	Data Sources listed in `dbawdsn.ini`.
Selected Data Source	The Data Source Tool displays the configuration information for the Data Source that is selected in the “Defined Data Sources” list.
Name	<label> value.
Description	<description> value.
Engine	Descriptive phrase for the <engine name> value.
Data Source	<Data Source name> value.
Username	<user name> value.
Password	<password> value.
dbANYWHERE Server URL	The dbANYWHERE Server’s URL.

Editing the data source

The DataSource Name tool lets you change an existing data source or add a new data source. Essentially, when you use this tool you are editing the `dbawdsn.ini` file, which is the dbANYWHERE Data Source configuration file.

To change data source values:

1. The `dsntool.bat` script must be running.
2. Select a Data Source from the Defined Data Sources list.

3 Enter or select a new value for each field you want to change.

4 Click Save.

To add a data source:

1 Click New.

2 Enter or select a value for each field.

To delete a Data Source

1 Select a Data Source from the Defined Data Sources list.

2 Click Delete.

Using the dbANYWHERE Admin tool

The dbANYWHERE Admin tool lets you conveniently check and test your dbANYWHERE system configuration.

Once you have started the tool, you can perform the following tasks with dbANYWHERE admin:

- Connecting to a dbANYWHERE Server machine
- Checking dbANYWHERE Server machine statistics
- Checking dbANYWHERE connections
- Testing a dbANYWHERE server

To start dbANYWHERE Admin tool:

1 Start dbANYWHERE.

2 From the Options menu, choose Properties, then Remote Admin.

The Remote Admin tab displays.

3 Select the Allow Remote Administration checkbox.

4 From the Start menu, choose Programs, then dbANYWHERE Server, then dbANYWHERE Admin.

Note: You can also start this tool from a command line by running the `dbawadmin.bat` file. It is in the dbANYWHERE `dbawAdmin` directory.

Connecting to a dbANYWHERE server

You need to connect to a dbANYWHERE server before you can use the dbANYWHERE Admin tool's features.

To connect dbANYWHERE Admin to a dbANYWHERE server:

1. Start dbANYWHERE Admin.
2. From the File menu, choose Connect.

   The dbANYWHERE Admin displays the Connect dialog box.
3. Enter the IP address and port number of the dbANYWHERE Server machine to which you want to connect.
4. Click Connect.

To disconnect dbANYWHERE Admin from a dbANYWHERE server:

From the File menu, choose Disconnect.

Checking dbANYWHERE server statistics

The dbANYWHERE Admin tool lets you check dbANYWHERE server machine statistics, like the dbANYWHERE version, amount of up time, and number of connections.

To check your server statistics:

1. Start dbANYWHERE Admin.
2. Connect to a dbANYWHERE Server machine.
3. From the View menu, choose Server Stats.
4. Click Refresh.

Fields

Version	dbANYWHERE version number
Up time	Length of time since dbANYWHERE started
# of accepted connections	Number of successful connections to dbANYWHERE since dbANYWHERE started
Current # of connections	Number of current connections to dbANYWHERE including the connection from dbANYWHERE Admin

Checking dbANYWHERE connections

The dbANYWHERE Admin tool lets you check the current connections to dbANYWHERE in terms of IP addresses, time connected, and transmission information.

To check current connections to your server:

1. Start the dbANYWHERE Admin.
2. Connect to a dbANYWHERE server.
3. From the View menu, choose Connections.
4. Click Refresh.

The format of the information is as follows:

```
<ID>, <IP address>, <time>, <not used>, <requests>, <bytes
 received>, <bytes sent>
```

Fields

<ID>	ID number for the connection. dbANYWHERE Admin assigns the ID numbers.
<IP address>	IP address of the machine that is connected to dbANYWHERE
<time>	Length of time since the connection was made
<not used>	Not used
<requests>	Number of requests the connected machine made to the dbANYWHERE Server machine
<bytes received>	Number of bytes dbANYWHERE received from the connected machine
<bytes sent>	Number of bytes dbANYWHERE sent to the connected machine

Testing link performance

The dbANYWHERE Admin tool lets you test a dbANYWHERE server's link performance.

To test link performance:

1. Start the dbANYWHERE Admin tool.
2. Connect to a dbANYWHERE server.
3. From the View menu, choose Ping.
4. Optional. Change the ping text, ping count, and repeat count.
5. Optional. Select the Results/Show checkbox.
6. Click Ping.

   The dbANYWHERE Admin tool pings the dbANYWHERE server. When the Admin tool receives the echo, it displays the round-trip time and the byte transfer rate.

Fields

Ping text	Text string to send in the ping message
Ping count	Number of copies of the text string to send in the ping message
Repeat count	Number of times to ping the dbANYWHERE Server machine
Results	Echo that dbANYWHERE Admin received back from the dbANYWHERE Server machine
Show	Indicates whether or not to display the results
Time stats	Length of time between sending the ping and receiving the echo
Clear	Clears the Time stats text area

Changing dbANYWHERE properties

You can use the Properties dialog box to specify the following:

- The IP address and port of the dbANYWHERE Server machine (Network tab)
- Message log configuration (Logging tab), SQL, Connection, Log to screen/file options.
- Memory list
- AutoDisconnect
- Idle disconnect
- Allocation of dbANYWHERE resources (Resources tab)
- Service configuration for dbANYWHERE on Windows NT (NT Service tab)

To change dbANYWHERE properties:

1. Start dbANYWHERE or the dbANYWHERE Service Manager.
2. From the Options menu, choose Properties.
3. For information about the properties, press F1. There is F1 help for each tab in the Properties dialog box.

Setting the dbANYWHERE server URL

The Network tab lets you set the dbANYWHERE Server machine's URL. The default settings work for typical systems.

To set the URL:

1. Display the Network tab by going to the Options menu and choosing Properties, then Network.
2. Specify the dbANYWHERE server's port number in the Port Number field.

   The default value for a server running on your local machine is 8889.
3. Specify the dbANYWHERE server's IP address in the IP Address field by either:
   - Using the default value (select the Use Default checkbox) if you have only one IP address defined for your dbANYWHERE server.
   - Entering a numeric value. Some servers are "multi-homed" and have more than one IP address. For such machines, we recommend that you specify the IP address.

Logging messages

The Logging tab lets you set up a message log. It specifies the information to log and the logging destination. A message log is useful for debugging during development. Since a message log can decrease performance, we recommend not using one at runtime.

Note: dbANYWHERE truncates each message printed in the console or Event Viewer at 8000 bytes.

To display the Logging tab:

From the Options menu, choose Properties, then Logging.

To log messages only when dbANYWHERE is maximized:

1. Select the Only When Visible checkbox.
2. Deselect the Always checkbox.

To log messages regardless of whether dbANYWHERE is maximized or minimized:

1. Deselect the Only When Visible checkbox.
2. Select the Always checkbox.

Fields

Log what: SQL statements	Indicates whether or not to log the SQL statements that are sent to the Data Source.
Log what: Connection stats	Indicates whether or not to log the connection statistics.
Log where: Log to screen	Indicates when to log messages to the dbANYWHERE window.
only when visible	Indicates whether or not to log messages when dbANYWHERE is maximized. This checkbox applies only when dbANYWHERE is in console mode. dbANYWHERE always selects this option and you cannot change it.
always	Indicates whether or not to always log messages regardless of whether dbANYWHERE is maximized or minimized.
Keep the most recent xxx messages.	Specifies the number of messages to keep in the log. The default value is 4096. To maximize the performance of dbANYWHERE, use a smaller number.
Log where: Log to file	Indicates whether or not to log messages to a file.
(checkbox)	Indicates whether or not to log messages to the file specified by the “Logfile path” value.
Let file grow to xxx megabytes.	Specifies the maximum file size. When the file reaches the maximum size, dbANYWHERE renames it to `filename.bak` and creates a new file for additional messages. If `filename.bak` already exists, dbANYWHERE deletes the old `filename.bak` before renaming the log file to `filename.bak`.
Logfile path	Specifies the pathname for the log file.

Log menu

The Log menu contains the following commands:

- Clear – deletes all log messages
- Copy – copies selected log messages to the clipboard
- Select All – selects all log messages

- Save As – saves all logged messages to the specified file

Allocating dbANYWHERE resources

The Resources tab lets you configure the connections of dbANYWHERE, and limit shared resources to minimize the impact of lazy or hostile clients.

To display the Resources tab:

From the Options menu, choose Properties, then Resources.

Limit connect time

These settings prevent a client from tying up resources indefinitely.

Disconnect client after xxx minutes	Indicates whether or not to disconnect a client after the client has been connected for the specified number of minutes even if the client is active.
or if idle for xxx minutes	Indicates whether or not to disconnect a client after the client has been idle for the specified number of minutes.

Limit simultaneous sessions

User defined limit	Specifies the maximum number of simultaneous sessions. This number cannot exceed the "Limit client sessions" values in the License dialog box. Limiting the number of simultaneous sessions can prevent too large a strain on your system. The best value for this limit depends on your system hardware and applications.

Limit system resource usage

Deny connections if available memory drops below xxx mbytes.	Specifies the minimum amount of system memory that must be available before making a new connection. dbANYWHERE denies new connections if available system memory drops below this level.

Allowing remote administration

The Remote Admin tab indicates whether or not to allow remote system administration which means that you can control dbANYWHERE from a computer other than the dbANYWHERE Server machine.

Note: Remote administration is currently limited to the features provided by dbANYWHERE Admin.

To display the Remote Admin tab:

From the Option menu, choose Properties, then Remote Admin.

License

The License dialog box displays your current license settings. These settings can only be changed by purchasing and upgrading your license. This dialog box includes the following options:

Customer Name

The customer or generic license name.

Customer ID

The customer or generic license ID.

Database drivers

The list of LICENSED drivers is displayed here. Licensed drivers have a different session limit than unlicensed drivers, as described below.

Limit Client Sessions

The maximum number of clients allowed to connect. There are separate settings for licensed and unlicensed drivers—unlicensed drivers are allowed a small number of connects for the purpose of evaluation only.

Note: dbANYWHERE uses the total number of connected clients when checking license limits. For example, if your unlicensed limit is 2, and 2 or more users are connected (to any driver), then you will not be able to connect to any unlicensed drivers.

IP Addresses

If your license restricts dbANYWHERE to specific a specific IP address or set of addresses, those IP addresses will be listed here. Otherwise, you'll see the "Any" box checked.

Expiration

Currently unsupported.

Release

These fields indicate which dbANYWHERE releases this license supports.

Allow NT Service Protocol

When this button is selected, this copy of dbANYWHERE may be configured and run as a NT Service.

Note: This feature is available only if you are running dbANYWHERE under Windows NT. For further information about installing an NT Service, see "Configuring dbANYWHERE as a Windows NT service (Windows NT only)" on page 13-8.

Allow Secure Protocol

Currently unsupported.

Allow Multiple Instances

Currently unsupported.

Upgrade button

Click this button to upgrade your license.

Note: You need one or more License Upgrade (`.luf`) files to upgrade. Use the file selection dialog box to find a `.luf` file and select it. The upgrade process displays a list of what was upgraded. Repeat for any additional license files. These changes are effective immediately -- there is no need to restart dbANYWHERE.

LiveUpdate

From the Options menu, select LiveUpdate to upgrade dbANYWHERE to the latest version. A wizard guides you through the update process. See "Updating Visual Cafe with LiveUpdate" on page 9-19.

I N D E X

Symbols

.ZIP file, contents, 13-6

A

accessor method, definition, 7-3
Actions wizard, 12-25
Add Table wizard, 12-28
adding
    code to Java source, 4-23
    components to forms, 5-9
    components to palette, 5-33
    DLL to a project, 10-19
    form to project, 5-8
    menu bar to frame or dialog box, 5-24
    menu to a form, 5-23
    packages to project, 4-37
Admin tool, dbANYWHERE, 13-18
administration, allowing remote (dbANYWHERE), 13-27
advanced search criteria, setting, 4-37
advanced Win32 compiler options, 10-8
applet
    adding to HTML page, 6-11
    advantages, 6-2
    creating, 2-6
    definition, 6-1
    determining required class files, 6-16
    ending remote debugging, 8-35
    including in Web page, 6-10
    limitations, 6-3
    passing parameters to, 6-12
    setting up project as, 12-20
    starting remote debugging, 8-34
Applet component, overview, 5-6
application
    considerations when creating native Win32, 10-15
    converting bytecode to Win32, 10-14
    creating, 2-7
    database columns, selecting, 12-23
    definition, 6-3
    deploying, 6-13
    determining required class files, 6-16
    ending remote debugging, 8-35
    setting up project as, 12-20
    starting remote debugging, 8-34
    tables, selecting, 12-23
application builder support service, JavaBeans, 7-8
architecture, three-tier, 13-2
arranging components on forms, 5-14
associating command keys and menu items, 5-27
AutoDetail component (dbAWARE), 12-4
automatic code generation, disabling, 9-27

B

backup options, specifying, 9-13
BeanInfo code, example, 7-17
Beans. *See* JavaBeans
binding code to a menu item, 5-28
binding of code
    to form or component, 4-26
    to menu command, 4-27
BorderLayout layout property, 5-16
bound property, definition, 7-4
    See also constrained property
breakpoint
    clearing, 8-20
    diamond indicator, 8-10, 8-17
    enabling and disabling, 8-20
    ignoring all, 8-21
    modifying conditional, 8-19
    setting
        conditional, 8-19
        on a line number, 8-17
        on a method name, 8-18
        on variable or expression, 8-19
        simple, 8-18
    toggling, 8-10
    viewing source code for, 8-21
    working with, 8-16–8-23
Breakpoints menu, Debug workspace, 8-38
Breakpoints window, overview, 8-3
browser limitations, 9-26
building Java programs, 6-18
bytecode, converting to native Win32, 10-14

C

Cafe project, importing into Visual Cafe, 11-2
Call Stack window
  overview, 8-5
  using, 8-26–8-28
  viewing method
    parameters, 8-27
    source code, 8-28
    variables, 8-27
Calls menu, Debug workspace, 8-39
Calls window. *See* Call Stack window
CardLayout layout property, 5-16
cell fonts, changing, 12-35
class
  adding, 4-7
  defining with Create Class Wizard, 4-20
  editing with Edit Class Wizard, 4-8, 4-21
  finding, 4-6
  grouping in Classes pane, 4-4
Class Browser
  adding a subclass from, 4-7
  adding methods from, 4-10
  configuring, 4-16
  editing event handler methods, 4-15
  editing events from, 4-41
  opening, 4-9
  overview, 4-1
  panes in, 4-5–4-16
  window, 4-3
Classes menu, using, 4-50
Classes pane
  navigating, 4-5
  Show All Classes option, using, 4-6
  Show Implements option, using, 4-6
CLASSPATH statement, 13-7
code
  adding to Java source, 4-23
  analyzing, 6-19
  binding to a menu command, 4-27
  binding to form or component, 4-26
  binding to menu item, 5-28
  compiling, 6-19
  correcting, 4-25, 6-7
  creating your own, 9-27
  disabling automatic generation, 9-27
  editing, 4-24
  editing options, specifying, 9-15
  enhancing, 4-25
  programming hot keys in, 4-27
  scrolling to specific line, 8-7
  stepping through, 8-8–8-13
  syntax errors, locating, 8-15
  viewing, 4-25
column label, changing in table, 12-25
column type, changing in table, 12-25
command keys
  associating with menu items, 5-27
  programming in code, 4-27
compiler options
  setting, 6-5
  setting advanced Win32, 10-8
compiling
  errors, about, 9-25
  errors, locating, 9-26
  projects, 6-4
  projects, including library files, 10-9
  source code, 6-19
  using JAVAC.EXE, 2-5
component
  adding an event, 4-39
  adding to palette, 5-33
  Applet, 5-6
  arranging
    in BorderLayout, 5-16
    in CardLayout, 5-16
    in FlowLayout, 5-17
    in GridBagLayout, 5-18
    in GridLayout, 5-18
    on form, 5-14
  attributes of
    events methods, 2-3
    interactions, 2-3
    properties, 2-2
    visual element, 2-2
  binding code to, 4-26
  changing, 5-12
  converting to JavaBeans, 7-11
  copying, 5-10
  definition, 3-3
  deleting
    from form, 5-12
    from palette, 5-35
  displaying invisible, 5-13
  form, overview, 2-3

interaction, creating, 5-20
InvisibleHTMLLink, using, 5-29
JavaBeans
adding to Component Library, 7-13
creating, 7-14
deleting from Component Library, 7-20
layouts, creating, 5-4
modifying properties, 5-19
moving between forms, 5-12
MultiList, using, 5-29
project, overview, 2-4
selecting for table, 12-24
top-level, overview, 5-6
TreeView, using, 5-32
using with events, 4-39
viewing Java source code, 5-37
workspace, overview, 2-4
Component Library
adding custom component, 5-36
deleting an object from, 5-37
JavaBeans, adding, 7-13
JavaBeans, deleting, 7-20
overview, 5-36
Component Palette, overview, 5-28
configuring
Class Browser, 4-16
dbANYWHERE, 13-3–13-5
on one machine (local), 13-4
with three machines, 13-5
with two machines (local and remote), 13-5
with two machines (remote), 13-4
Hierarchy Editor, 4-16
modem to use with LiveUpdate, 9-20
UNIX-based Web servers, 6-17
ConnectionDescriptor class, using, 7-16
ConnectionInfo component (dbAWARE), 12-6
considerations
differences in main class between bytecode and native Win32, 10-16
when creating a native Win32 application, 10-15
when linking a native Win32 application, 10-15
considerations when importing Visual J++ project, 11-5
constrained property, definition, 7-4
See also bound property
container
definition, 3-4
ScrollingPanel, using, 5-30
TabPanel, using, 5-31
container class, overview, 5-2–5-3
Continue to Cursor command, Debug workspace, 8-11
conventions used in manual, 1-11
converting Java applications, bytecode to native Win32, 10-14
copying
components between forms, 5-10
menus, 5-23
correcting source code, 4-25
corruption, detecting, 9-28
Create Class Wizard, defining new classes, 4-20
Create Database wizard, 12-14
creating
applets, 2-6
applications, 2-7
component interaction, 5-20
documents in Source Editor, 4-23
interactions, 4-44
menus in Form Designer, 5-21
Palette tab, 5-33
cross-platform development, 9-26
current statement arrow, Debug workspace, 8-18
custom component, adding to Component Library, 5-36
custom keywords, specifying, 9-7
custom palette, building, 5-33

D

data source
adding with ODBC Administrator, 12-16
connecting to, 13-10–13-14
creating with dbANYWHERE DataSources tool, 12-16
defining, 12-15
identifying for project, 12-22
testing connection, 13-15
troubleshooting failed connection test, 13-16
database
browser, about, 12-10
columns, selecting for application, 12-23

connecting to
data source, 13-10–13-14
Informix data source, 13-11
Microsoft Access data source, 13-11
Microsoft SQL data source, 13-13
ODBC SQL-based data source, 13-12
ODBC Xbase data source, 13-13
Oracle data source, 13-14
Sybase SQL Anywhere data source, 13-10
Sybase SQL data source, 13-14
Watcom data source, 13-10
connecting to dbNAVIGATOR, 12-30
creating for project, 12-14
disconnecting from, 12-34
environment options, 12-2
grid component, adding, 12-32
manipulation functions, about, 12-12
selecting server for project, 12-21
Database Development Edition (dbDE) overview, 1-6
Database Grid component (dbAWARE), 12-6
DataSource Name tool, dbANYWHERE, 13-17
DataSource tool, using, 13-16
dbANYWHERE
Admin tool, 13-18
allocating resources, 13-26
allowing remote administration, 13-27
architecture, 13-2
configuring, 13-3–13-5
as Windows NT service, 13-8
one machine (local), 13-4
three machines, 13-5
two machines (local and remote), 13-5
two machines (remote), 13-4
connections, checking, 13-20
DataSource Name tool, 13-17
DataSources tool, 12-16
definition, 13-1
installing Windows NT service, 13-9
logging messages, 13-23
package
setting up for development, 13-7
using, 13-6–13-8
properties, modifying, 13-22
resources
limiting connection time, 13-26
limiting simultaneous sessions, 13-27
limiting system resource usage, 13-27
Resources tab, displaying, 13-26
running the Windows NT service, 13-9
server
checking statistics, 13-19
connecting to, 13-19
disconnecting from, 13-19
setting URL, 13-23
service, viewing messages, 13-9
setting up package for automatic download, 13-7
starting from Windows 95, 12-13
starting from Windows NT, 12-13
testing link performance, 13-21
tools, using, 13-16
updating using LiveUpdate, 13-29
dbaw (dbANYWHERE) package, 13-6
dbAWARE
applet, creating, 2-7
application, creating, 2-7
component
AutoDetail, 12-4
ConnectionInfo, 12-6
Database Grid, 12-6
MultiView, 12-6
overview, 12-3
Projection, 12-7
RelationView, 12-9
Session, 12-10
Project wizard, 12-19
wizard, about, 12-11
dbNAVIGATOR
about, 12-10
connecting to database, 12-30
connecting with server, 12-30
refreshing, 12-32
table, using to add, 12-28
using in form development, 12-29
Debug menu, Debug workspace, 8-36
Debug workspace
Breakpoints menu, overview, 8-38
Breakpoints window, 8-3
Call Stack window, 8-5
Calls menu, overview, 8-39
Continue to Cursor command, 8-11
current statement arrow, 8-18

Debug menu, overview, 8-36
Debug toolbar, overview, 8-8
editing files in Source window, 8-8
Insert menu, overview, 8-37
menus, using, 8-35–8-42
Messages window, 8-5
printing from Source window, 8-7
Project menu, overview, 8-35
Set Breakpoint command, 8-18
Source menu, overview, 8-40
Source window, 8-6
Threads menu, overview, 8-39
Threads window, 8-4
using, 8-2–8-6
Variables menu, overview, 8-39
Variables window, 8-3
Watch window, 8-4
Window menu, overview, 8-41
debugging
ending a session, 8-28
ending remote, 8-35
incremental, 11-1
native Win32 Java applications, 10-13
overview, 2-4
remotely, 8-34–8-35
resuming a suspended thread, 8-31
resuming other suspended threads, 8-32
run-time editing, 9-4
setting up for remote, 8-34
single thread, 8-31
starting a session, 8-7
starting remote, 8-34
suspending a thread, 8-31
suspending other threads, 8-32
threads, 8-28
using Threads window, 8-29
ValueTips, setting, 9-5
default macro, saving, 4-53
defining, data source, 12-15
deleting
component from form, 5-12
inheritance relationships, 4-20
interactions, 4-47
object from Component Library, 5-37
palette components, 5-35
deploying
applications, 6-13
Java programs, 6-7
JavaBeans, 7-21
native Win32 applications, 10-13
description file
converting to JavaBeans, 7-11
definition, 7-11
design pattern, definition, 7-5
designing GUI with Visual Cafe, 5-7
dialog box, adding a menu bar, 5-24
disabling automatic code generation, 9-27
display font and color, customizing, 9-11
displaying inherited methods, 4-4
displaying invisible components, 5-13
DLL
adding to a project, 10-19
including in main project, 10-9
registering using SNJREG, 10-12
specifying a program for debugging, 10-5
specifying a program for running, 10-5
document, creating in Source Editor, 4-23
documentation
Visual Cafe
online help, 1-10
online tour, 1-10
user's guide, 1-11
Visual Page
online help, 1-4
online tour, 1-3
user's guide, 1-3
downloading latest updates, 9-19
drag-and-drop
in Package view, 4-38
in Project window, 3-24

E

Edit Class Wizard
editing classes, 4-8
editing new classes, 4-21
editing
event handler methods, 4-15
HTML files, 6-11
interactions, 4-46
menu and menu bar, 5-26
source code, 4-24
enhancing source code, 4-25
entry, definition, 3-2
environment options

for debugging, 9-3
setting, 9-1
errors
compiling, 9-25
locating compiling, 9-26
programming, 9-25
event
adding to components, 4-39
editing from Class Browser, 4-41
editing from Form Designer, 4-41
using with components, 4-39
working with, 4-39–4-41
event adapter, definition, 7-7
event handler method
editing from Class Browser, 4-15
editing from Form Designer, 4-15
event handling service, JavaBeans, 7-6
event listener, definition, 7-6
event state object, definition, 7-6
example
adding a DLL to a project, 10-19
creating a native Win32 executable, 10-17
creating an executable with a DLL, 10-18
implementing BeanInfo information, 7-17
exceptions
catching, 8-13
handling, 8-13
making all break, 8-15
making only unhandled break, 8-15
setting, 8-14
throwing, 8-13
exporting
native classes, 10-6
native packages, 10-6
expressions, using in Watch window, 8-24

F

features
Database Development Edition (dbDE), 1-6
Professional Development Edition (PDE), 1-6
support for JDK 1.1, 1-4
Visual Page, 1-2
Web Development Edition (WDE), 1-4
FlowLayout layout property, 5-17
fonts, conventions used in manual, 1-11
form
adding
a menu, 5-23
to project, 5-8
using drag-and-drop, 5-9
using the Insert menu, 5-9
binding code to, 4-26
dbNAVIGATOR, using to develop, 12-29
definition, 5-1
fields, tabbing between, 5-14
form component, overview, 2-3
Form Designer
creating menus, 5-21
editing event handler methods, 4-15
editing events from, 4-41
Frame component, overview, 5-7
frame, adding a menu bar, 5-24

G

getter method, definition, 7-3
graphics, displaying in Form Designer, 5-6
grid attributes, changing, 12-34–12-38
GridBagLayout layout property, 5-18
GridLayout layout property, 5-18
grouping
classes in Classes pane, 4-4
members in Members pane, 4-4
GUI, designing with Visual Cafe, 5-7
guidelines for adding code to source, 4-23

H

hardware limitations with Java, 9-24
Help file set, defining, 9-3
Hierarchy Editor
configuring, 4-16
deleting inheritance relationships, 4-20
right-click menu, using, 4-51
using, 4-17–4-20
hot keys
associating with menu items, 5-27
mapping Visual Cafe commands, 9-8
programming in code, 4-27
HTML file
adding an applet, 6-11
editing, 6-11
passing parameters to applets from, 6-12
viewing, 6-11

I

importing
- Cafe project, 11-2
- Visual J++ project through dsp project file, 11-4
- Visual J++ project through dsw workspace file, 11-3
- Visual J++project, overview, 11-3

incremental debugging, overview, 11-1
indexed properties, definition, 7-3
Informix data source, connecting to, 13-11
inheritance relationship
- changing, 4-20
- deleting, 4-20

inherited methods, displaying, 4-4
Insert Class Wizard, using, 4-20–4-22
Insert menu, Debug workspace, 8-37
interaction
- changing an existing, 4-46
- creating, 4-44
- creating component, 5-20
- deleting, 4-47
- working with, 4-41–4-50

interested party, definition, 7-4
interface
- defining with Create Class Wizard, 4-20
- editing with Edit Class Wizard, 4-8, 4-21

Internet connectivity requirements, 1-12
introspection service, JavaBeans, 7-4
invisible component, displaying, 5-13
InvisibleHTMLLink component, using, 5-29

J

JAR. *See* Java ARchive
Java Abstract Window ToolKit (AWT), overview, 5-4
Java ARchive (JAR)
- creating, 7-10
- JavaBeans storage, 7-8

Java language
- case sensitive limitations, 9-24
- cross-platform features, 9-26
- hardware limitations, 9-24

Java program, creating, 2-5
Java requirements for Visual Cafe, 1-15
Java source code
- adding code, 4-23
- correcting, 4-25
- editing, 4-24
- enhancing, 4-25
- viewing, 4-25, 5-37

JavaBeans
- adding to Component Library, 7-13
- adding Visual Cafe information, 7-15–7-18
- application builder support service, 7-8
- associated files, 7-22
- basic structure, 7-2
- converting components to, 7-11
- creating, 7-8, 7-14
- deleting from Component Library, 7-20
- design fundamentals, 7-9
- event handling service, 7-6
- features, 7-1
- introspection service, 7-4
- packaging, 7-21
- persistence service, 7-7
- properties, modifying, 7-20
- property management service, 7-3
- renaming, 7-23
- services, 7-2–7-8
- storage, 7-8
- terminology, 7-2
- testing, 7-9
- tools for building, 7-21

JAVAC.EXE compiler, 2-5

K

keyboard equivalents
- associating with menu items, 5-27
- mapping Visual Cafe commands, 9-8
- programming in code, 4-27

keywords, specifying custom, 9-7

L

layout manager, using None, 5-15
library files, including in main project, 10-9
License dialog box, 13-27
License Upgrade (.LUF) files, 13-29
limitations
- browser, 9-26
- case sensitivity in Java, 9-24
- hardware, 9-24

Java language, general, 9-24
linking a native Win32 application, considerations, 10-15
LiveUpdate
Internet, using over, 9-20
modem, using with, 9-20
troubleshooting, 9-23
uninstalling, 9-23
upgrading dbANYWHERE, 13-29
using with Visual Cafe, 9-19
LUCLEAN.EXE, using, 9-23

M

macro
playing the default, 4-52
recording the default, 4-52
saving the default, 4-53
using a recorded macro, 4-53
working with, 4-52–4-54
main project
setting the DLL directory, 10-10
setting the object file directory, 10-10
member
grouping in Members pane, 4-4
sorting in Members pane, 4-4
Member Attributes dialog box, using, 4-14
Members pane, navigating, 4-11
menu
adding menu items, 5-25
adding submenus, 5-25
adding to a form, 5-23
adding to menu bar, 5-24
Classes, using, 4-50
copying, 5-23
creating in Form Designer, 5-21
editing, 5-26
editing structure, 5-26
menu bar
adding to frame or dialog box, 5-24
editing, 5-26
menu command, binding code to, 4-27
Menu Designer, pop-up menu, 5-22
Messages window
build errors, 6-20
using, 6-19
Messages window, overview, 8-5
method
adding from Class Browser, 4-10
adding from Source pane, 4-15
displaying inherited, 4-4
setting breakpoint on, 8-18
stepping into, 8-9
stepping out of, 8-10
viewing parameters in Call Stack window, 8-27
viewing source code in Call Stack window, 8-28
viewing variables in Call Stack window, 8-27
Microsoft Access data source, connecting to, 13-11
Microsoft SQL data source, connecting to, 13-13
migrating
Visual Cafe 1.0 to 2.0, 3-22
modem
COM port, selecting, 9-21
configuring INIT string, 9-21
dialing parameters, selecting, 9-22
identifying for LiveUpdate, 9-20
modem, configuring for LiveUpdate, 9-20
modifying
JavaBean properties, 7-20
properties of components, 5-19
moving
component between forms, 5-12
palette components, 5-35
multicast event source, definition, 7-7
See also unicast event source
MultiList component, using, 5-29
MultiView component (dbAWARE), 12-6

N

native
applications
creating, 10-2
differences in main class, 10-16
exporting, 10-6
setting project options, 10-3
specifying the name, 10-3
classes, exporting, 10-6
DLLs, creating, 10-2
executables, creating, 10-2
packages, exporting, 10-6
Win32 executable, example, 10-17
Win32 executable, example with a DLL, 10-

18
new features
Class Browser and Editor, 1-9
contextual menus, 1-7
debugger, 1-9
display options, 1-9
drag-and-drop, 1-9
enhanced wizards, 1-9
LiveUpdate, 1-10
Open by Name, 1-7
project and environment management, 1-8
version compatibility, 1-8
ZIP archive and JAR support, 1-7
None layout manager, 5-15

O

ODBC Administrator, adding data source, 12-16
ODBC SQL-based data source, connecting to, 13-12
ODBC Xbase data source, connecting to, 13-13
online help
Visual Cafe, 1-10
Visual Page, 1-4
Oracle data source, connecting to, 13-14

P

package
dbANYWHERE
setting up for automatic download, 13-7
setting up for development, 13-7
dbaw (dbANYWHERE), 13-6
symjava (JDBC), 13-6
Package view
creating packages, 4-38
using drag-and-drop, 4-38
working with, 4-37
packaging JavaBeans, 7-21
palette
adding components, 5-33
building a custom, 5-33
deleting components, 5-35
moving components, 5-35
Palette tab, creating, 5-33
pane
Members (Class Browser), 4-11
Source (Class Browser), 4-15
parameter
passing to applet, 6-12
viewing in Call Stack window, 8-27
path, setting for source files, 9-3
persist, definition, 7-7
persistence service, JavaBeans, 7-7
playing the default macro, 4-52
prerequisites for Visual Cafe, 1-11
Professional Development Edition (PDE)
overview, 1-6
program
building, 6-18
deploying, 6-7
pausing to debug, 8-9
restarting, 8-12
resuming, 8-12
running
to cursor location, 8-12
to first line, 8-11
to the end, 8-12
stepping through when paused, 8-22
stopping to debug, 8-9
programming
errors, common, 9-25
project
adding files, 3-13
closing, 3-23
compiler options, setting, 3-45–3-55
components, working with, 3-29–3-32
concepts, 3-3
creating database for, 12-14
creating new, 3-15
definition, 3-1
file, definition, 3-2
folder, definition, 3-2
folder, included files, 3-12
identifying data source, 12-22
multiple, working with, 3-14
opening existing, 3-16
options, setting, 3-34–3-45
packages, working with, 3-33–3-34
path, definition, 3-6
selecting the database server, 12-21
setting path for source files, 9-3
setting the DLL directory, 10-10
setting the object file directory, 10-10

setting up as applet or application, 12-20
settings, reviewing, 12-27
templates, 3-8
viewing files in, 3-26
working with, 3-15–3-24
workspace, using, 3-55
project component, overview, 2-4
Project menu, Debug workspace, 8-35
Project window
Files view, 3-8
Objects view, 3-3
Packages view, 3-6
using, 3-24–3-29
Projection component (dbAWARE), 12-7
properties
definition, 7-3
modifying component, 5-19
modifying dbANYWHERE, 13-22
modifying JavaBean, 7-20
property editor, definition, 7-8
property management service, JavaBeans, 7-3
property sheet, definition, 7-8

Q

quick keys
associating with menu items, 5-27
mapping Visual Cafe commands, 9-8
programming in code, 4-27

R

recording the default macro, 4-52
reflection, definition, 7-5
registering DLLs using SNJREG, 10-12
RelationView component (dbAWARE), 12-9
remote debugging
ending, 8-35
setting up, 8-34
starting, 8-34
removing
component from Component Library, 5-37
component from form, 5-12
palette components, 5-35
right-click menu, using in Hierarchy Editor, 4-51
running projects, 6-4
run-time editing, 9-4

S

sample
adding a DLL to a project, 10-19
creating a native Win32 executable, 10-17
creating an executable with a DLL, 10-18
save options, specifying, 9-13
ScriptMaker dialog box
Macros option, 4-53
using, 4-53
scrolling in Source Editor, 8-7
ScrollingPanel container, using, 5-30
search criteria
setting advanced, 4-37
specifying, 4-34
search path, specifying, 4-35
server
checking statistics (dbANYWHERE), 13-19
connecting with dbANYWHERE, 13-19
connecting with dbNAVIGATOR, 12-30
disconnecting from dbANYWHERE, 13-19
setting dbANYWHERE URL, 13-23
service, dbANYWHERE
configuring on Windows NT, 13-8
installing on Windows NT, 13-9
running on Windows NT, 13-9
viewing messages, 13-9
Session component (dbAWARE), 12-10
Set Breakpoint command, Debug workspace, 8-18
setter method, definition, 7-3
settings, reviewing for project, 12-27
shortcut keys
associating with menu items, 5-27
mapping Visual Cafe commands, 9-8
programming in code, 4-27
SNJREG, registering DLLs, 10-12
sorting members in Members pane, 4-4
source code
adding code to, 4-23
analyzing, 6-19
correcting, 4-25, 6-7
creating your own, 9-27
editing, 4-24
editing options, specifying, 9-15
enhancing, 4-25
scrolling to specific line, 8-7
stepping through, 8-8–8-13

syntax errors, locating, 8-15
viewing, 4-25
viewing for thread, 8-33
Source Editor
creating new documents, 4-23
key editing options, specifying, 9-10
scrolling in, 8-7
setting typing mode, 4-39
using, 4-22–4-32
Source menu, Debug workspace, 8-40
Source pane, navigating, 4-15
Source window
editing files in, 8-8
overview, 8-6
printing files in, 8-7
startup mode, defining, 9-2
subclass, adding from Class Browser, 4-7
subprojects, working with, 3-58–3-60
Sybase SQL Anywhere data source, connecting to, 13-10
Sybase SQL data source, connecting to, 13-14
SymantecBeanDescriptor class, using, 7-16
symjava (JDBC) package, 13-6
syntax errors, 9-25
locating in source code, 8-15
system path, definition, 3-6
system requirements for Visual Cafe, 1-12

T

tab, creating Palette, 5-33
tabbing between form fields, 5-14
table
dbNAVIGATOR, using to add, 12-28
selecting for application, 12-23
TabPanel container, using, 5-31
target, definition, 3-1
templates for projects, 3-8
testing
data source connection, 13-15
link performance (dbANYWHERE), 13-21
text formatting, specifying, 9-5
thread
debugging, 8-28–8-34
debugging a single, 8-31
resuming other suspended, 8-32
resuming suspended, 8-31
suspending, 8-31
suspending other, 8-32
viewing
active variables, 8-33
call stack for, 8-33
source code for, 8-33
Threads menu, Debug workspace, 8-39
Threads window, overview, 8-4
three-tier architecture, 13-2
tool
Admin (dbANYWHERE), 13-18
DataSource (dbANYWHERE), 13-16
DataSource Name (dbANYWHERE), 13-17
for building JavaBeans, 7-21
toolbar, Debug workspace, 8-8
toolbar, position and visibility, 9-17
TreeView component, using, 5-32
troubleshooting
compiling errors, 9-25
data source connection, 13-16
detecting corruption, 9-28
LiveUpdate connection, 9-23
programming errors, 9-25
typing mode, setting in Source Editor, 4-39
typographic conventions used in manual, 1-11

U

unicast event source, definition, 7-7
See also multicast event source
uninstalling LiveUpdate, 9-23
UNIX-based Web server, configuring, 6-17
updating Visual Cafe, 9-19
URL, setting dbANYWHERE server, 13-23

V

ValueTips, setting, 9-5
variable
adding to Watch window, 8-25
deleting from Watch window, 8-26
modifying in Variables window, 8-24
modifying in Watch window, 8-25
setting breakpoint on, 8-19
viewing active in thread, 8-33
viewing in Call Stack window, 8-27
viewing the type of, 8-24
viewing the value of, 8-23
watching, 8-10

Variables menu, Debug workspace, 8-39
Variables window
overview, 8-3
using, 8-23–8-24
viewing
HTML files, 6-11
Java source code for a component, 5-37
source code, 4-25
Visual Cafe
Database Development Edition (dbDE), 1-1
editors overview, 1-15
Internet connectivity requirements, 1-12
LiveUpdate, using with, 9-19
migrating from 1.0 to 2.0, 3-22
new features, 1-7–1-10
Class Browser and Editor, 1-9
contextual menus, 1-7
debugger, 1-9
display options, 1-9
drag-and-drop, 1-9
enhanced wizards, 1-9
LiveUpdate, 1-10
Open by Name, 1-7
project and environment management, 1-8
version compatibility, 1-8
ZIP archive and JAR support, 1-7
prerequisites, 1-11
Professional Development Edition (PDE), 1-1
startup mode, defining, 9-2
system requirements, 1-12
text formatting, specifying, 9-5
toolbars overview, 1-14
tools for building JavaBeans, 7-21
Web Development Edition (WDE), 1-1
windows overview, 1-13
Visual J++ project, importing into Visual Cafe, 11-3

W

warnings for adding code to source, 4-23
Watch window
adding variables, 8-25
deleting variables, 8-26
modifying variables, 8-25
overview, 8-4
using expressions in, 8-24
watching a variable, 8-10
Watcom data source, connecting to, 13-10
Web Development Edition (WDE) overview, 1-4
Web page, including applet in, 6-10
Web server, configuring UNIX-based, 6-17
Win32 compiler options, setting advanced, 10-8
Window menu
Class Browser command, 4-3
Debug workspace, overview, 8-41
wizard
about dbAWARE, 12-11
Actions, 12-25
Add Table, 12-28
Create Database, 12-14
dbAWARE Project, 12-19
workspace component, overview, 2-4
workspaces, using with projects, 3-55

Z

ZIP file, contents, 13-6