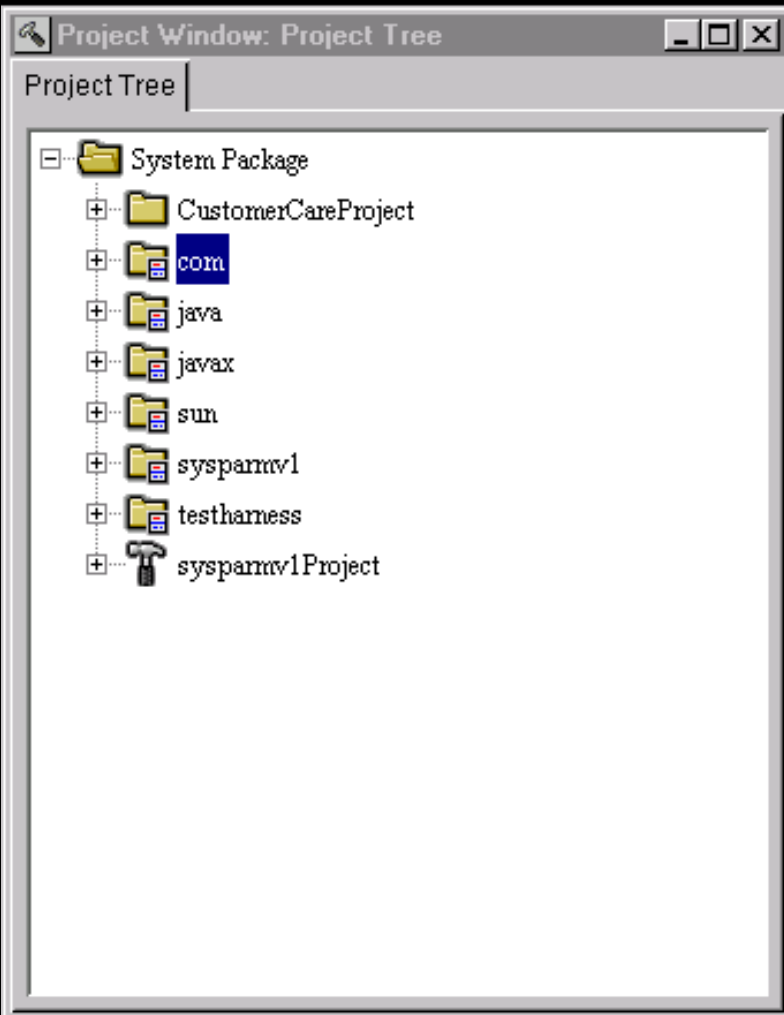
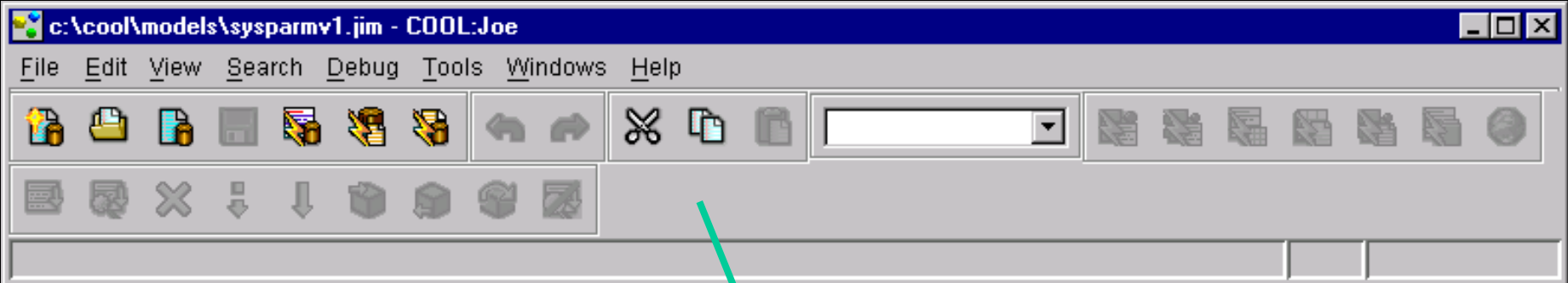
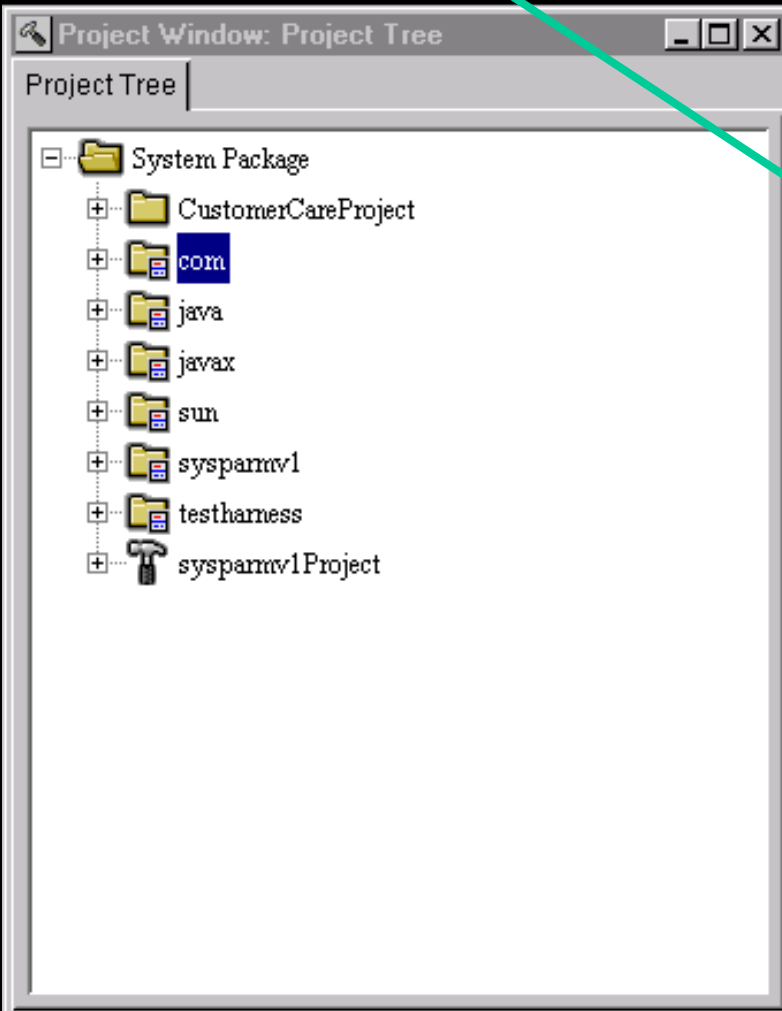
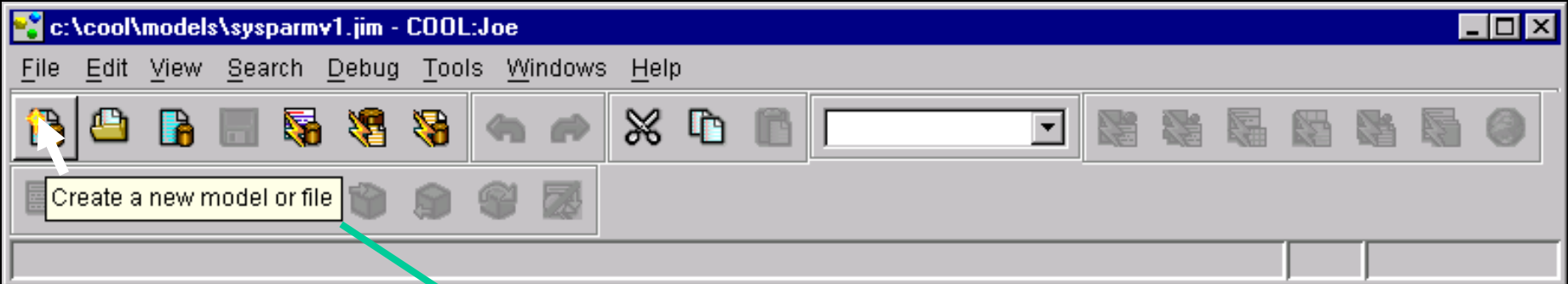


COOL:Joe 1.0

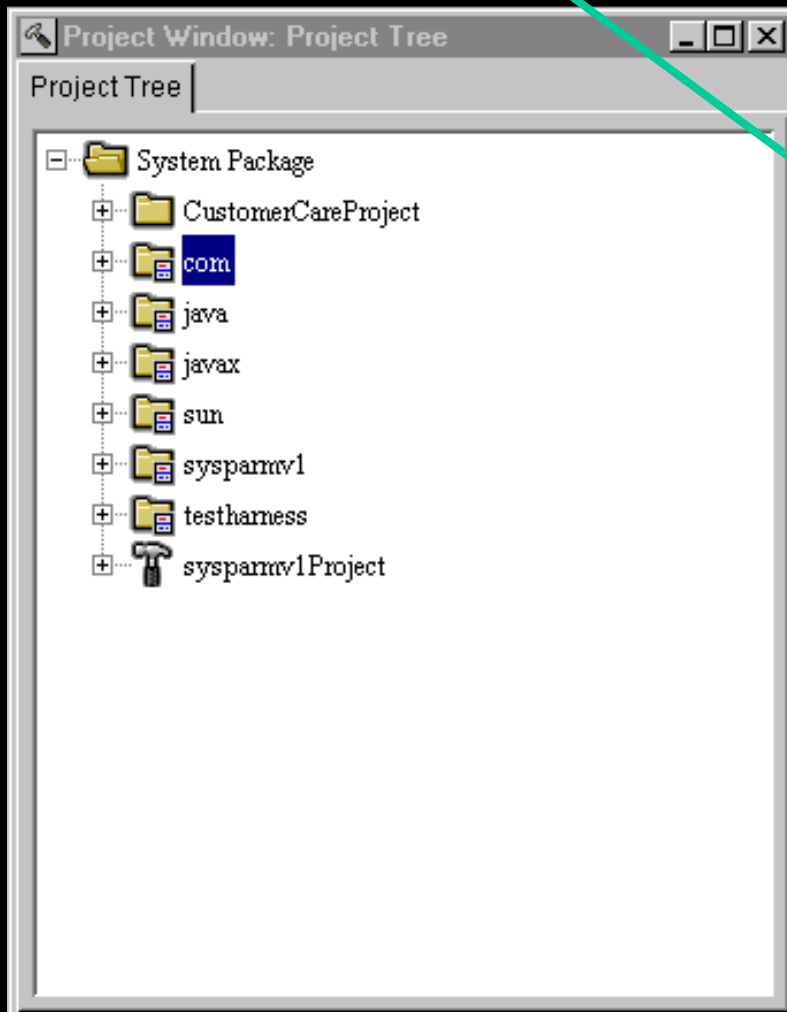
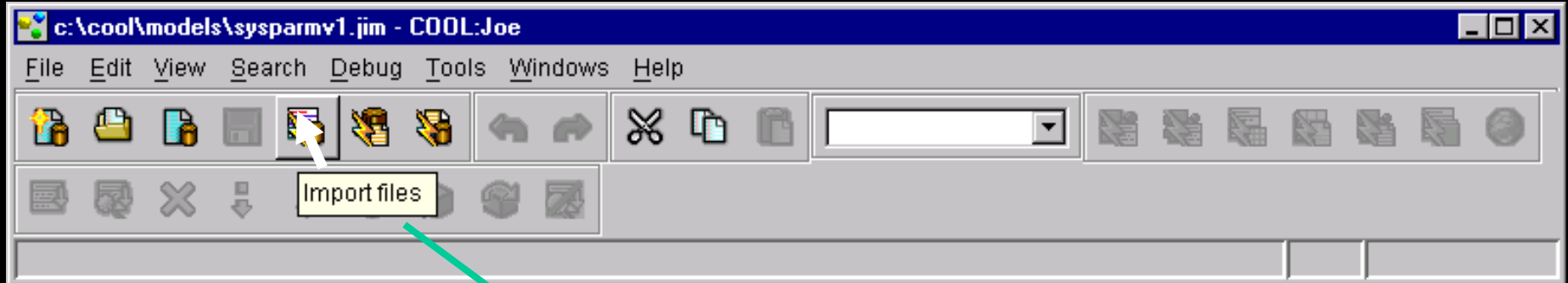
A Quick Tour



This is the COOL:Joe tool set framework. From here the COOL:Joe developer controls the entire architecture, design, implementation, assembly, and deployment process.



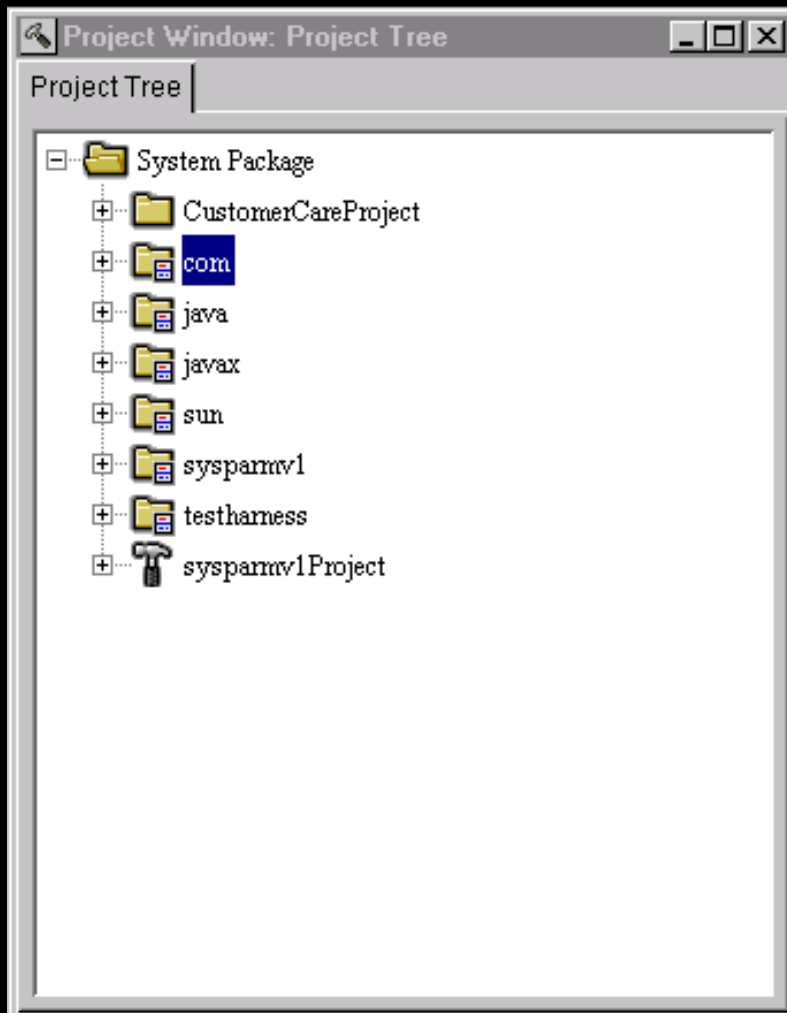
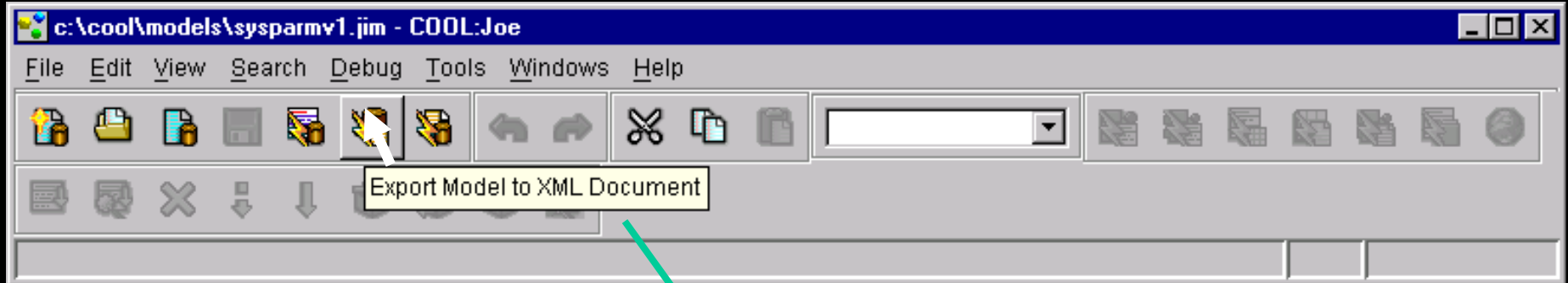
To help organize your projects you can manage each component in its own separate model supporting a federated team development approach.



COOL:Joe allows you to keep all of your Java project in the COOL:Joe repository. You can import any file and save it in the COOL:Joe project tree.

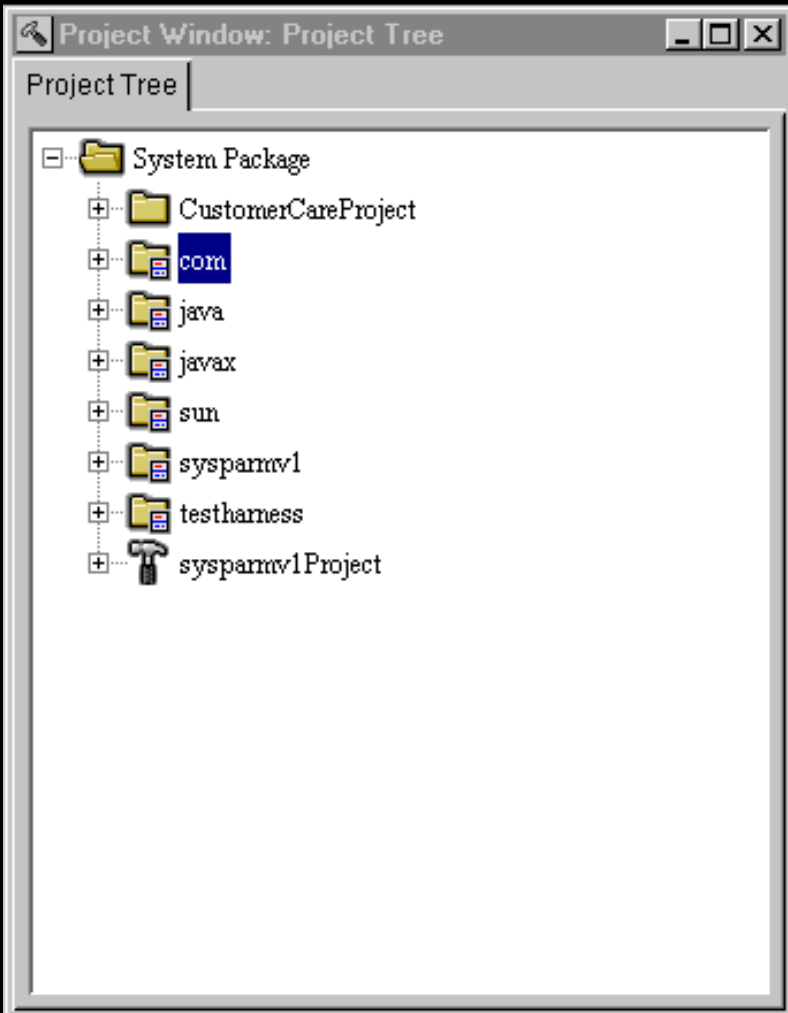
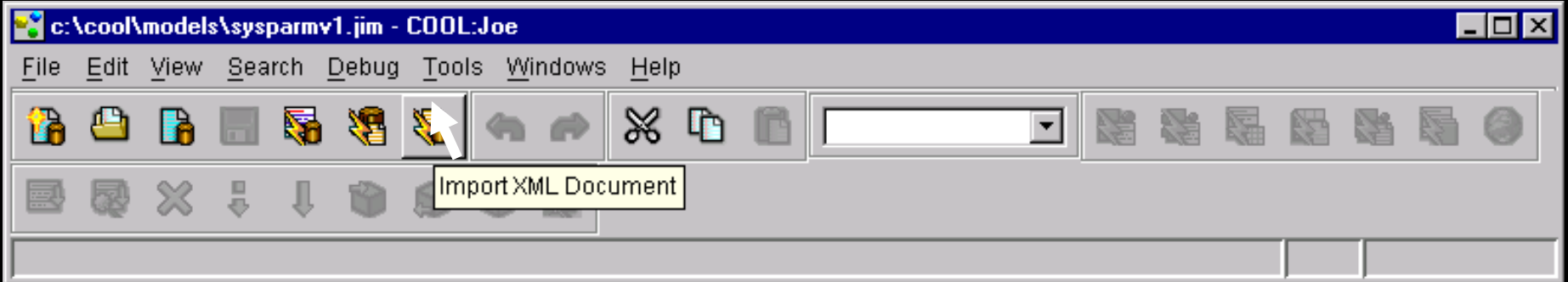
This allows you to manage all aspects of your development from one place.

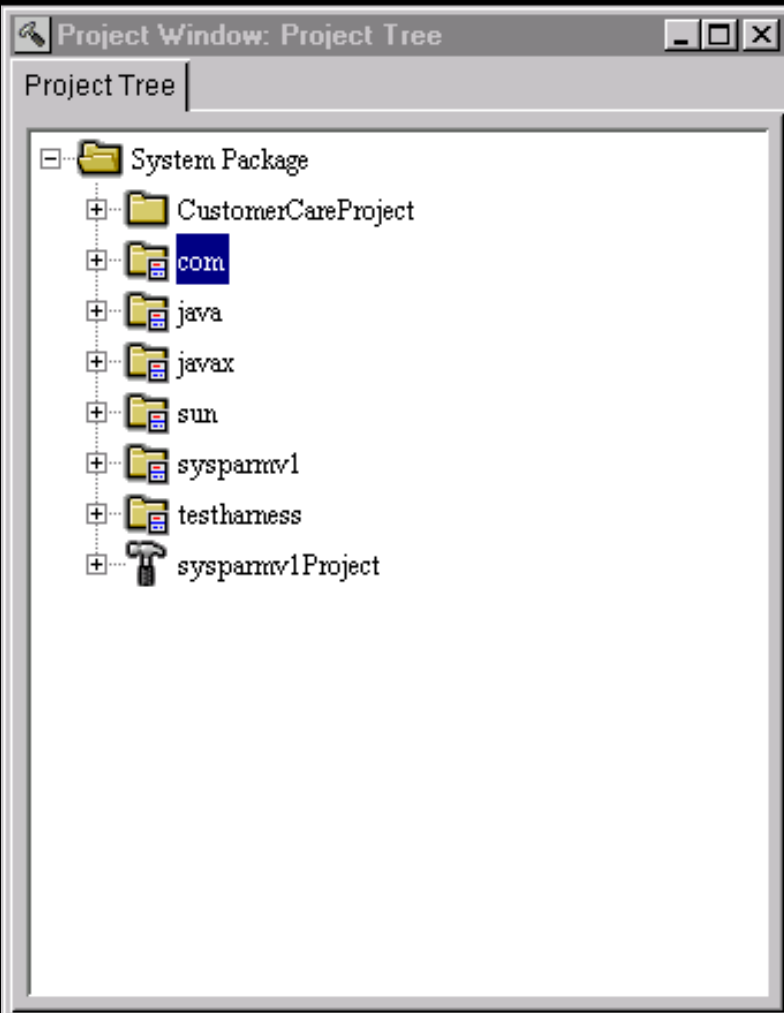
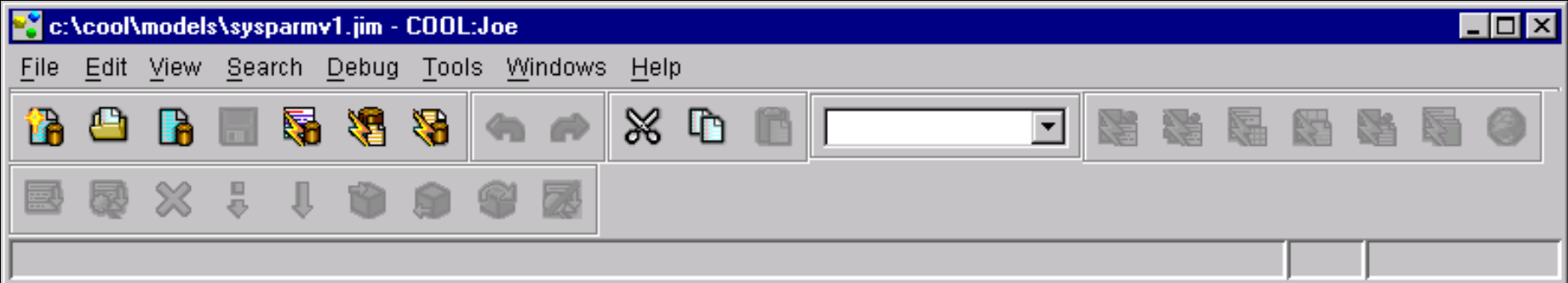
COOL:Joe!



As your project grows and changes over time you will want to create versions, share model elements with other team members, and interact with other tools.

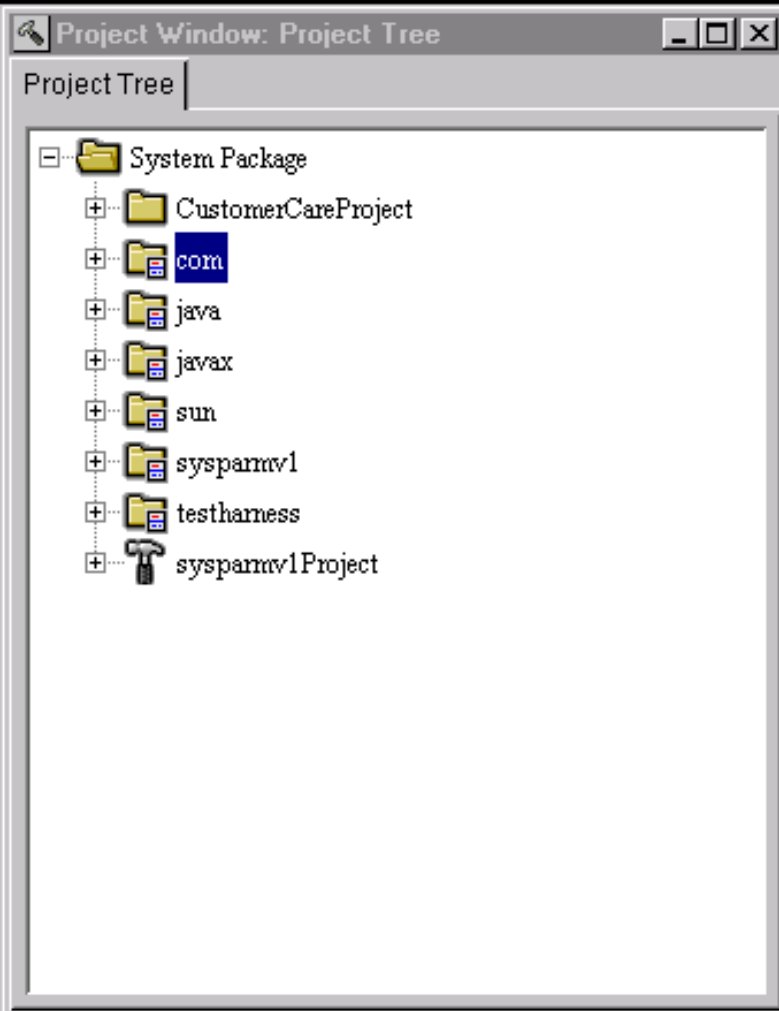
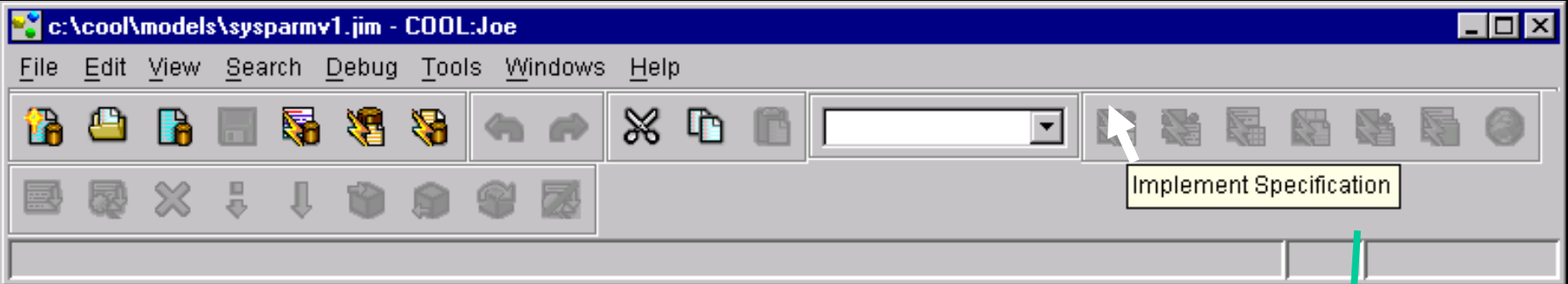
COOL:Joe uses XML to make this happen.



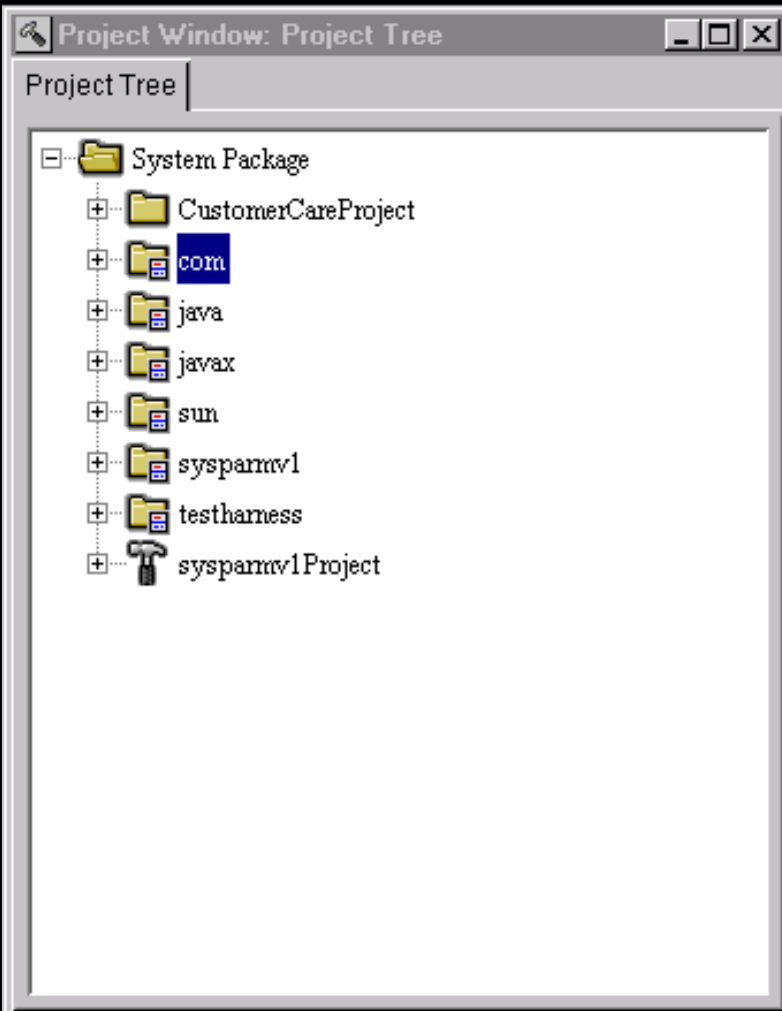
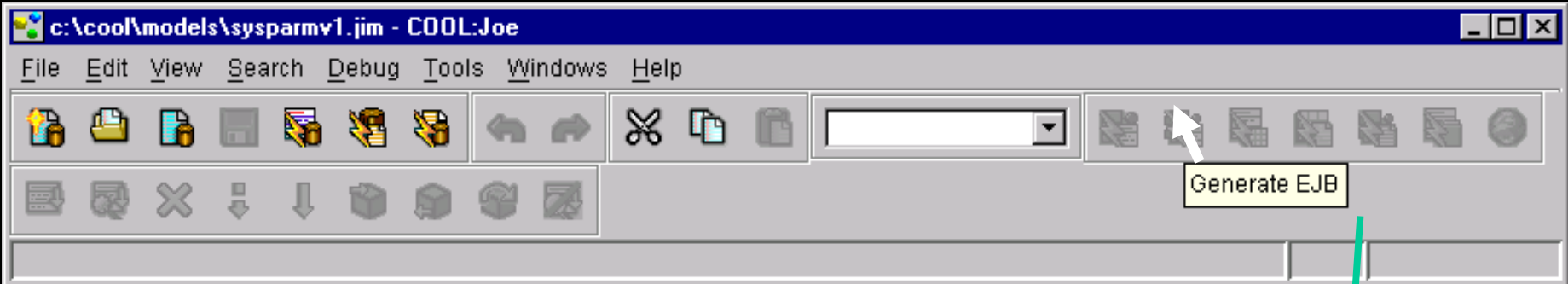


Now lets take a look at how COOL:Joe can help you be more productive when building server side Java applications following the EJB specification from Sun Microsystems.

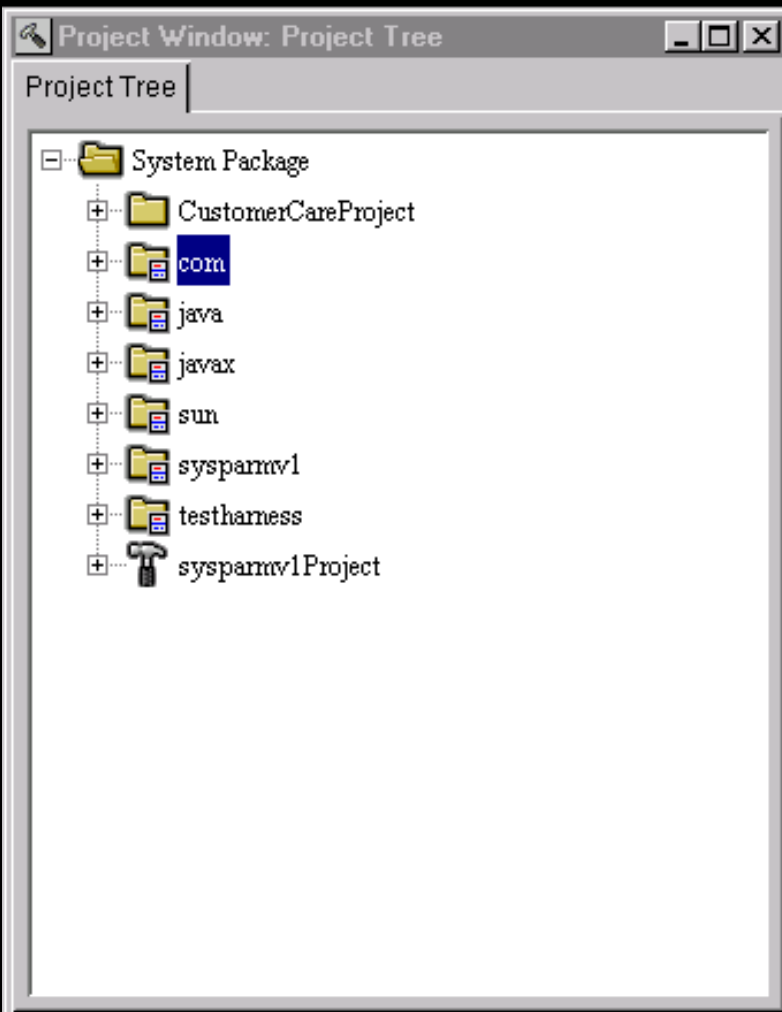
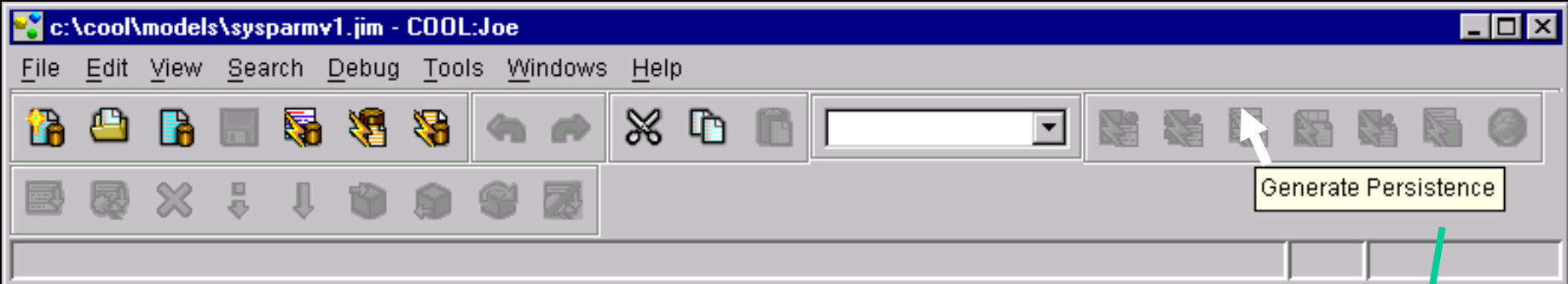
You will find COOL:Joe's wizards simplify the entire development process.



COOL:Joe allows you to work at different levels of abstraction. A component specification level and an actual Java class level. The Implement Specification wizard generates your first cut Java implementation directly from the specification allowing you to work smarter, not harder.



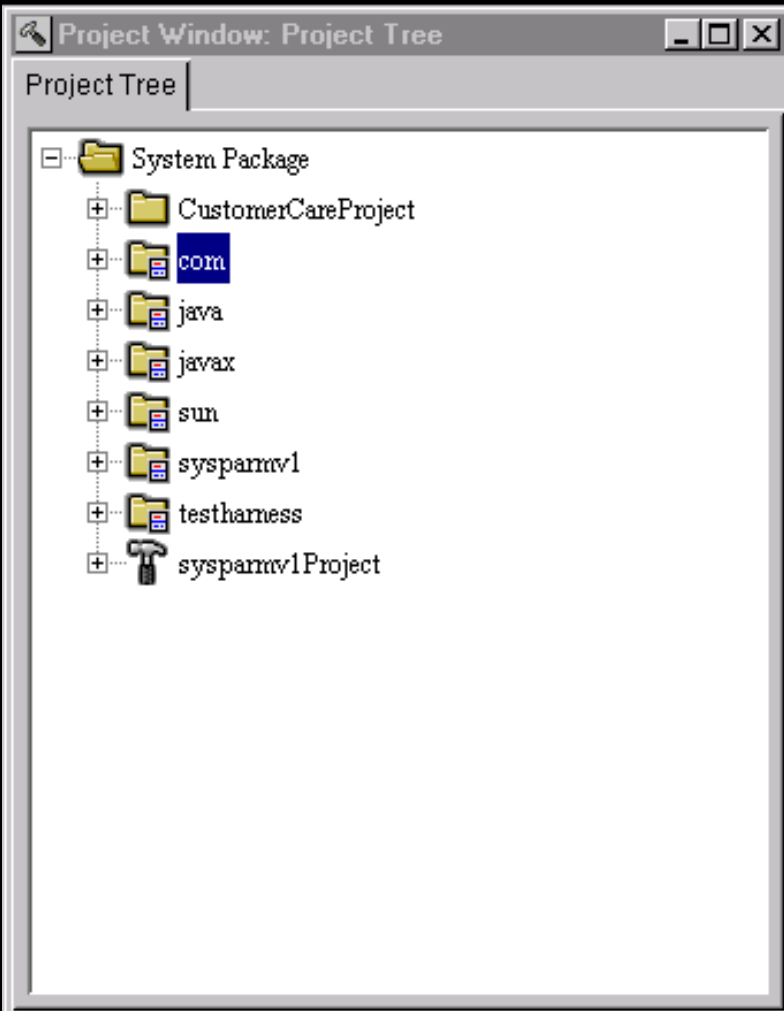
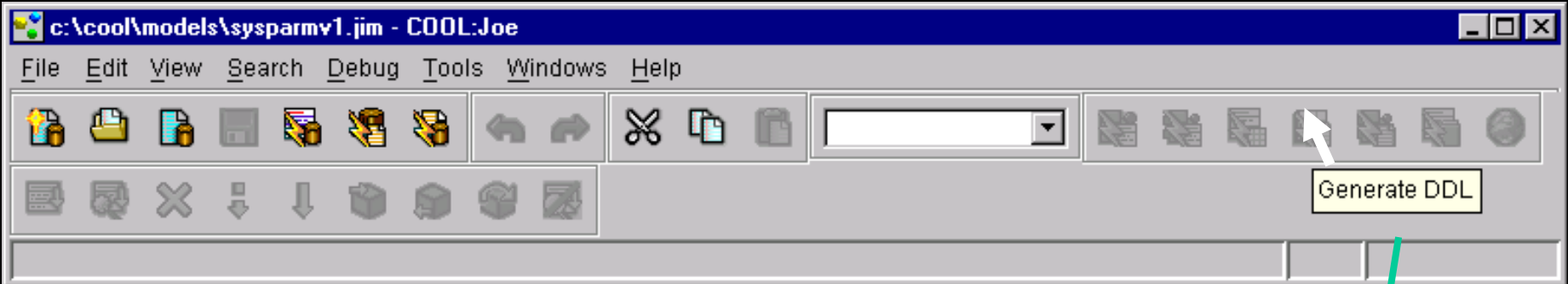
The Generate EJB wizard takes your business component and wraps it with the necessary classes to deliver a fully compliant EJB.



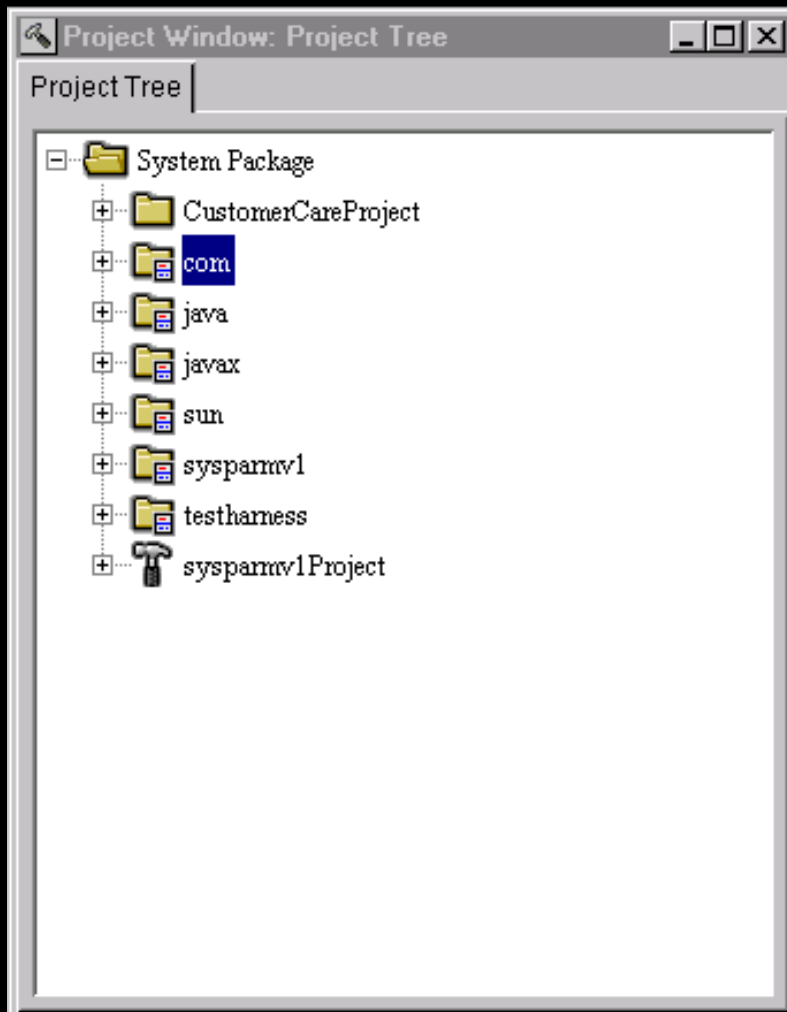
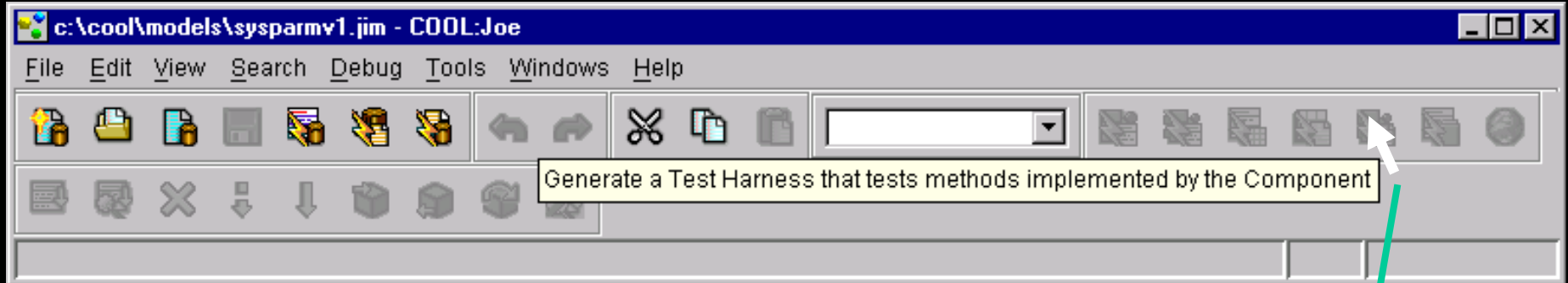
The Generate Persistence wizard allows you to forward engineer a new data base and supporting object-to-relational mapping code or to reverse engineer existing data structures to be used with this new component.

You no longer have to be an expert in SQL, object to relational mapping, etc.

COOL:Joe does it for you!



The Generate DDL wizard assists you with creating new database structures for your component.

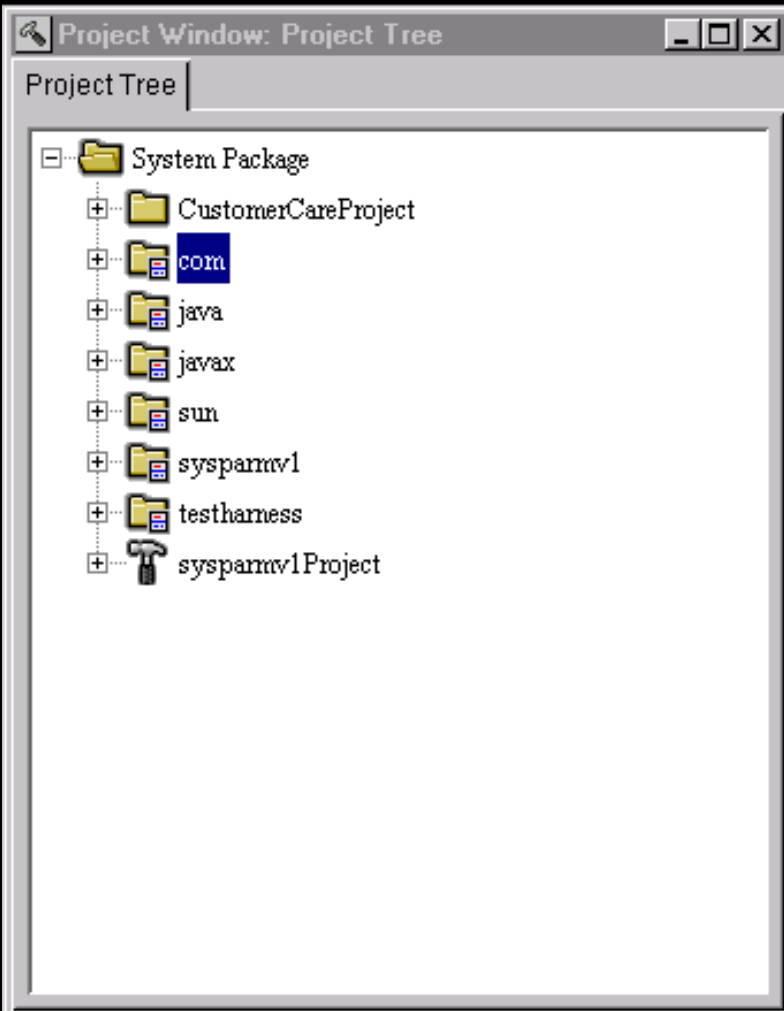
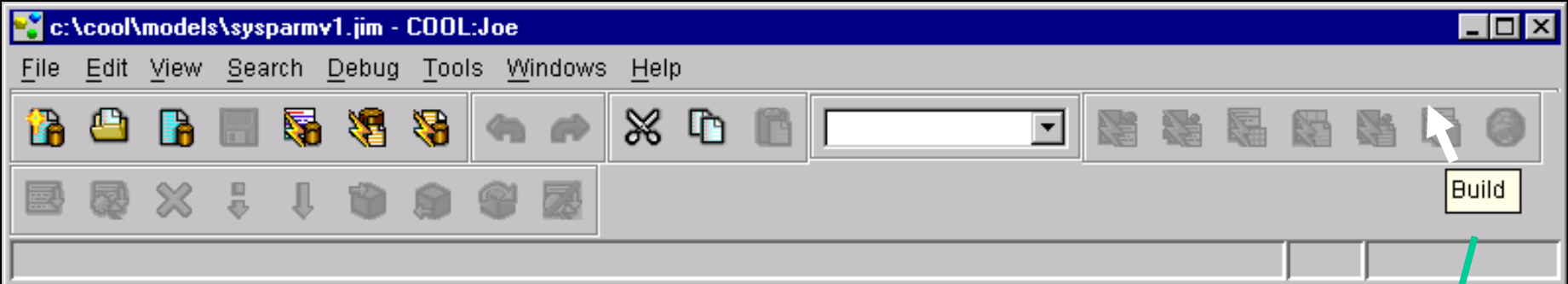


When its time to test your server side logic COOL:Joe becomes your best friend.

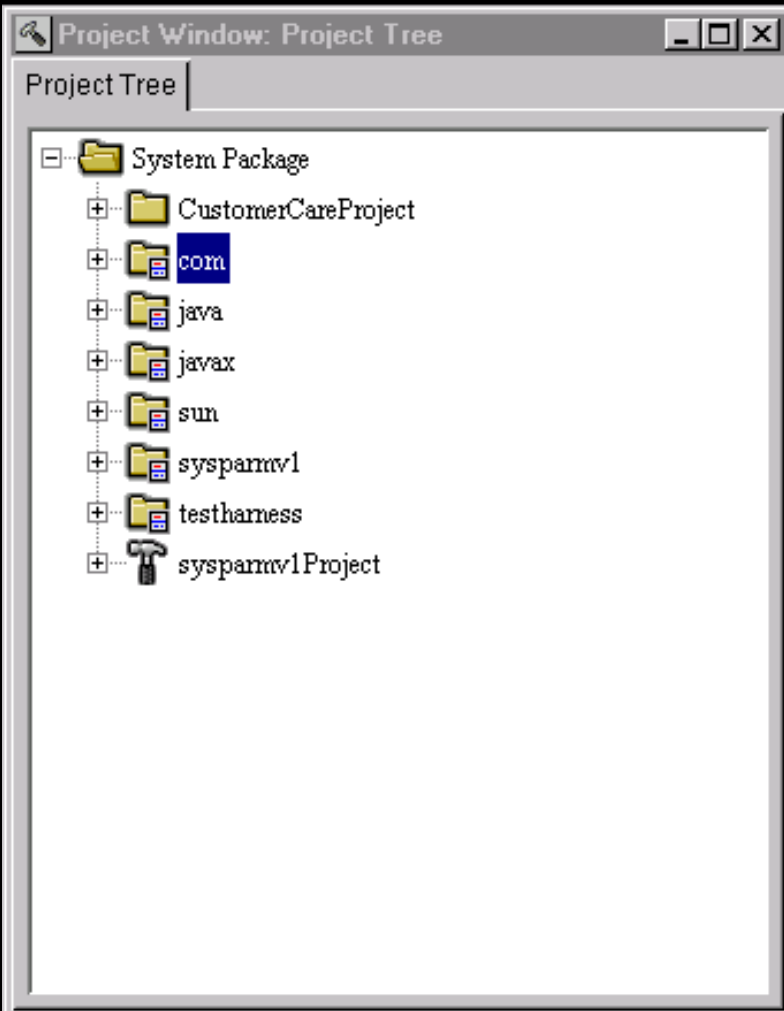
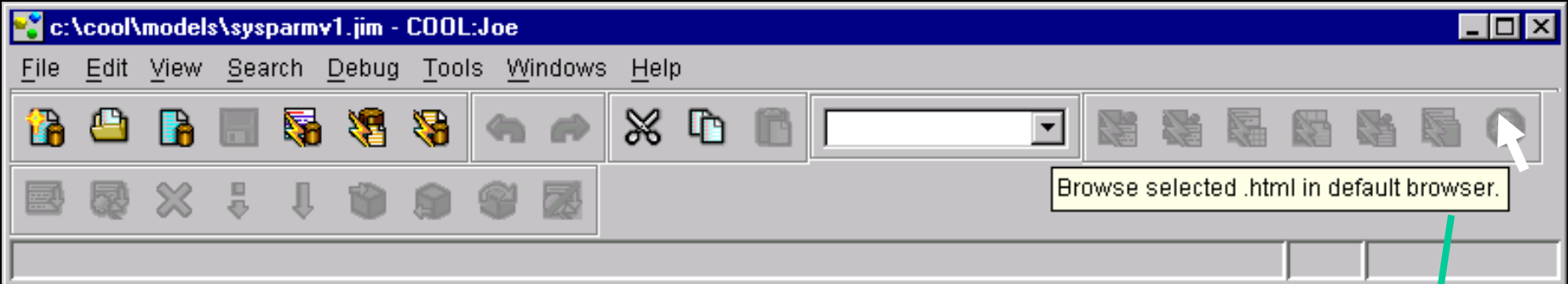
No longer will you have to code (and debug) 'dummy' windows to drive your code.

COOL:Joe generates this for you.

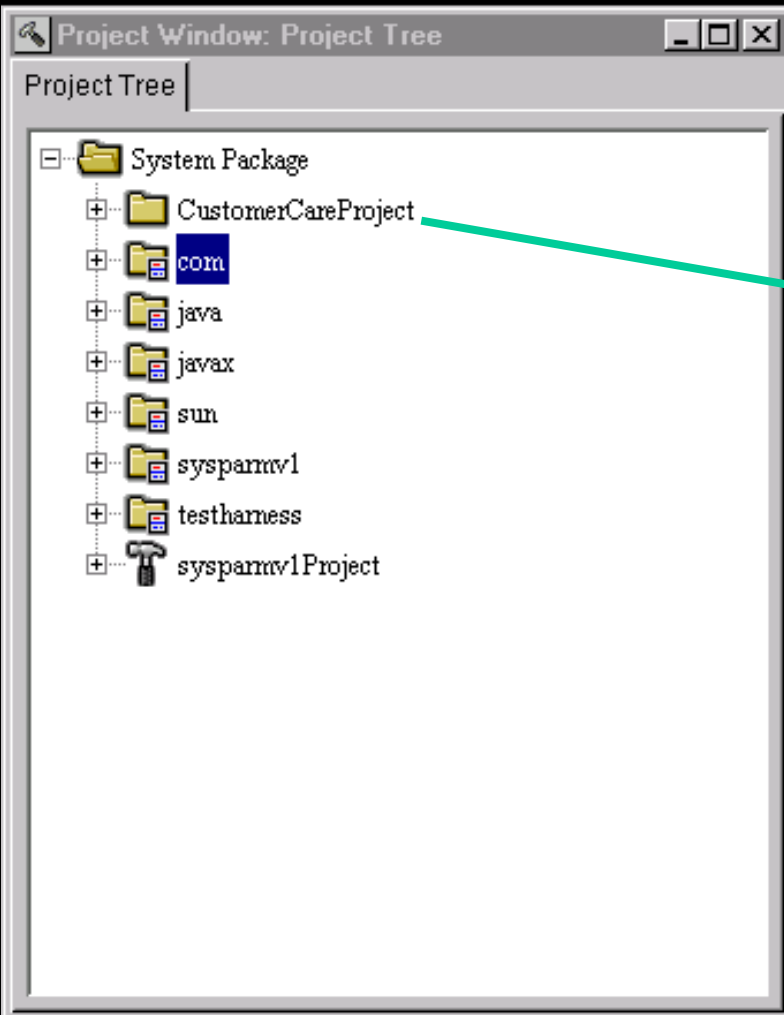
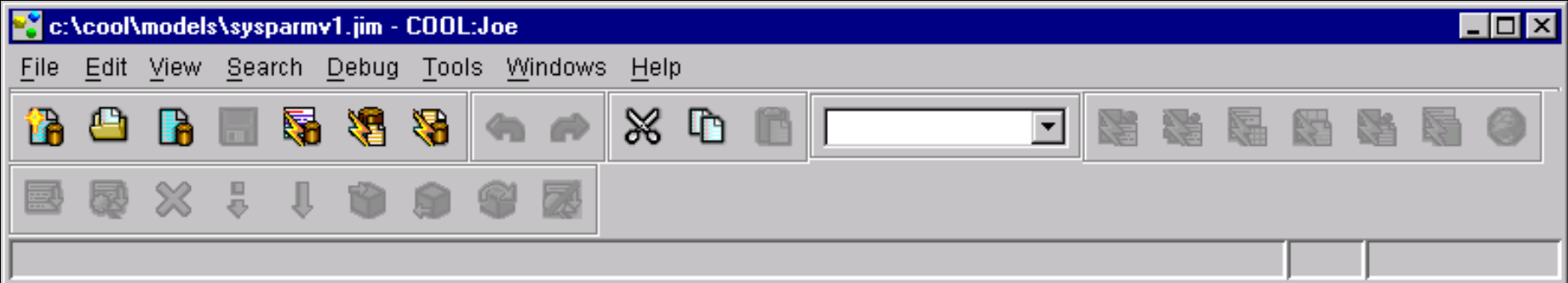
With a few clicks of a wizard you will be testing and debugging!



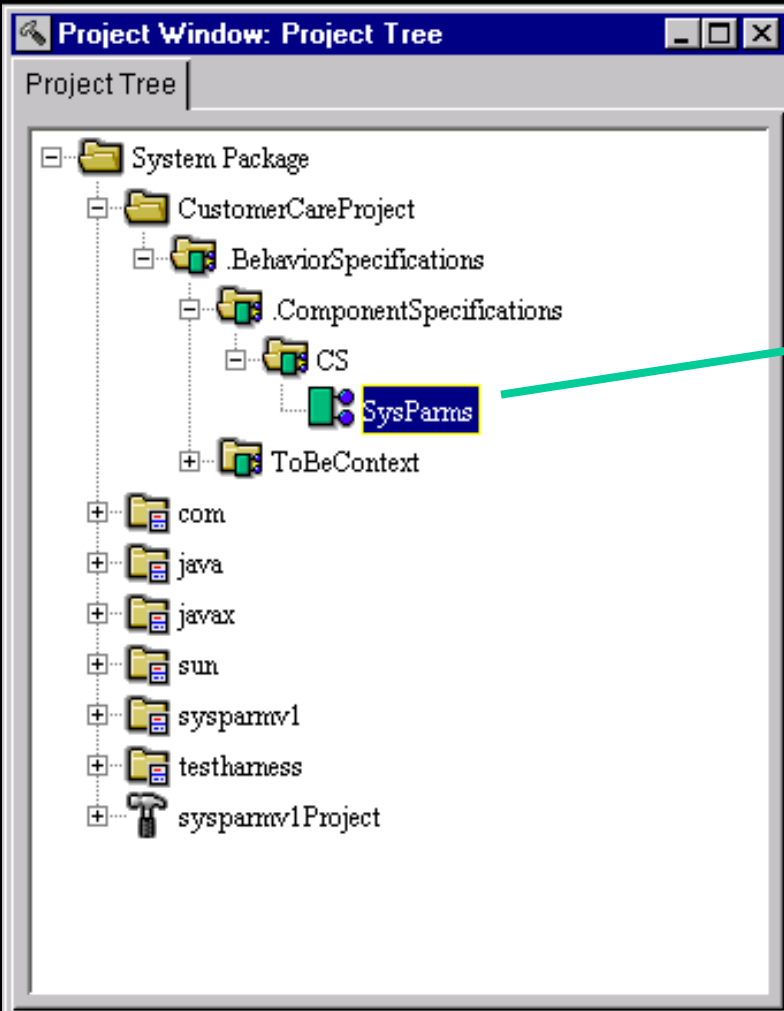
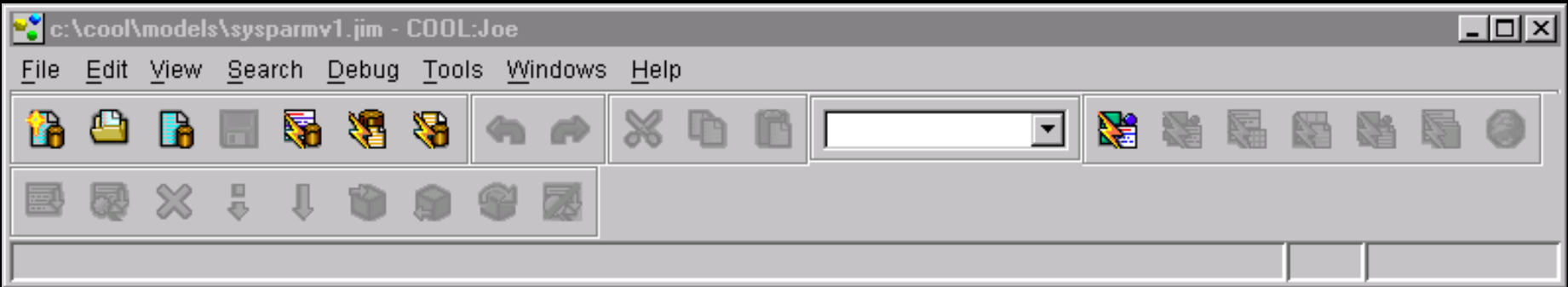
When it comes time to compile your code COOL:Joe is there to help. The Build Wizard takes your project and makes it ready to go.



Many COOL:Joe customers keep all of their code in the COOL:Joe repository. The Browse Wizard allows you to view your HTML quickly and effortlessly.



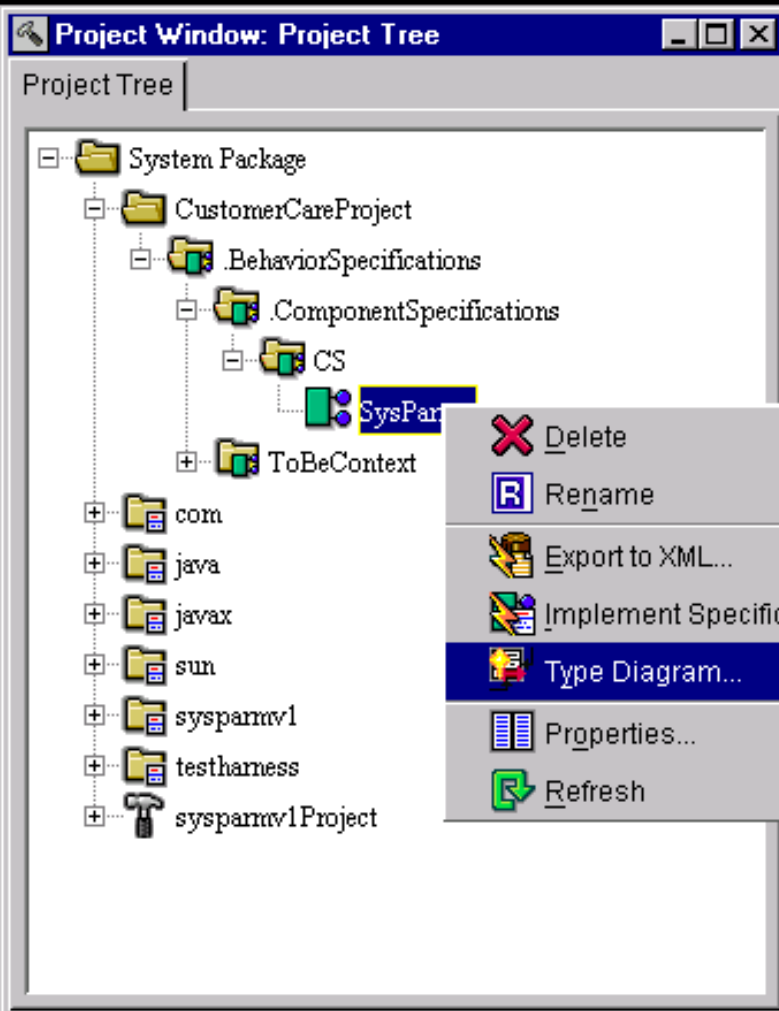
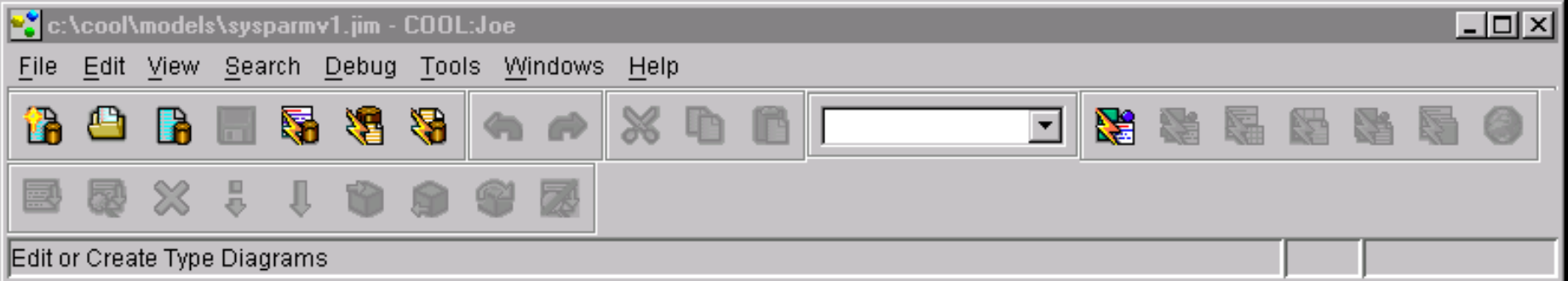
Now lets go look at COOL:Joe in action. We will start first by looking at a high level component specification. Lets start by expanding the COOL:Joe System Package and looking into the CustomerCareProject.

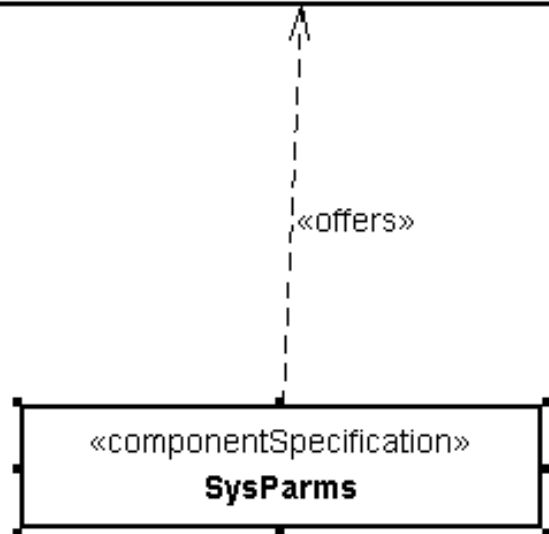
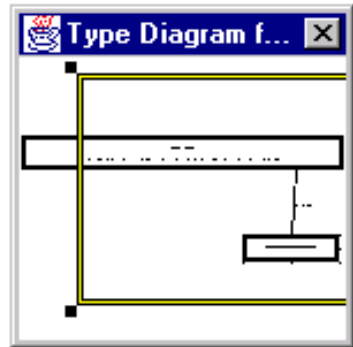
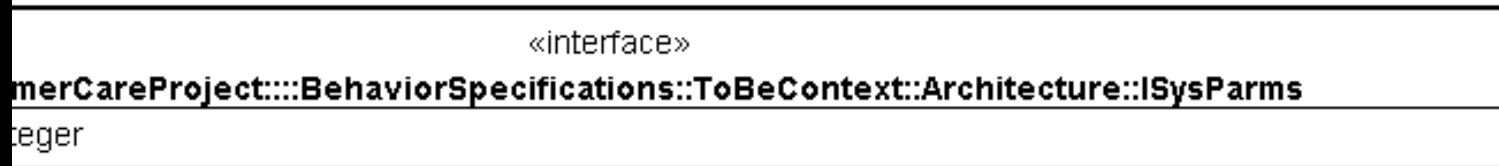


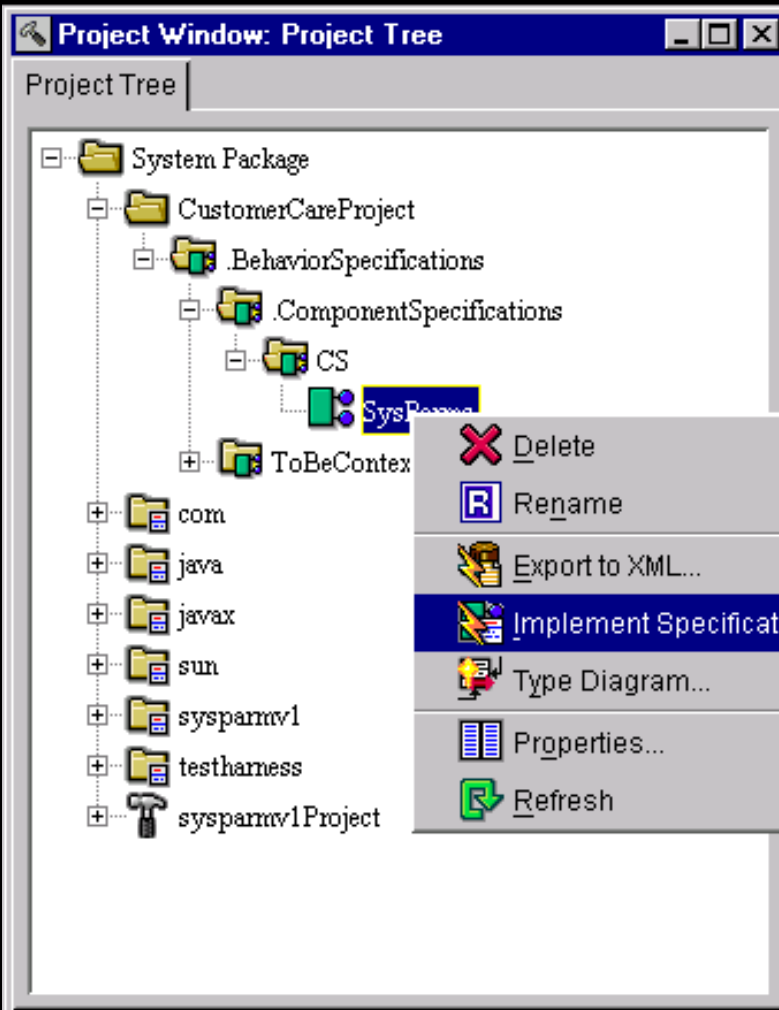
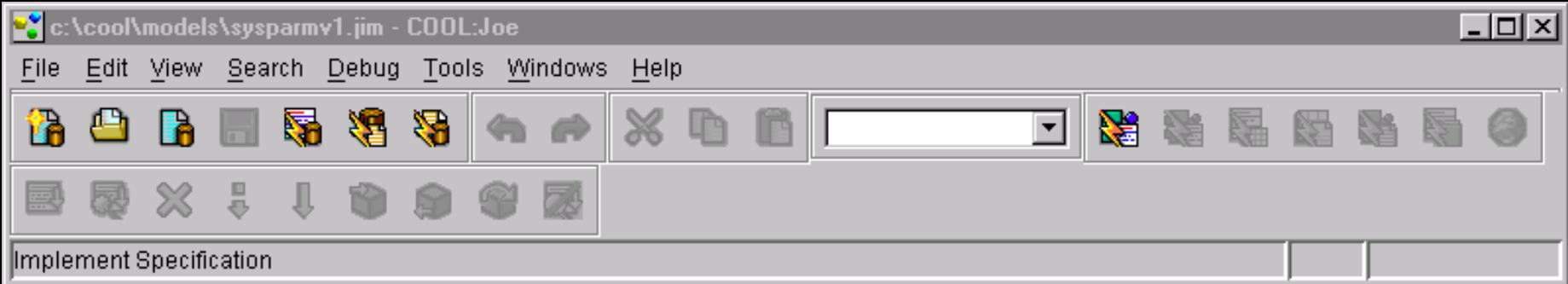
The SysParm component must be implemented as an EJB.

The first thing you might want to do is visualize the component specification.

COOL:Joe can help!

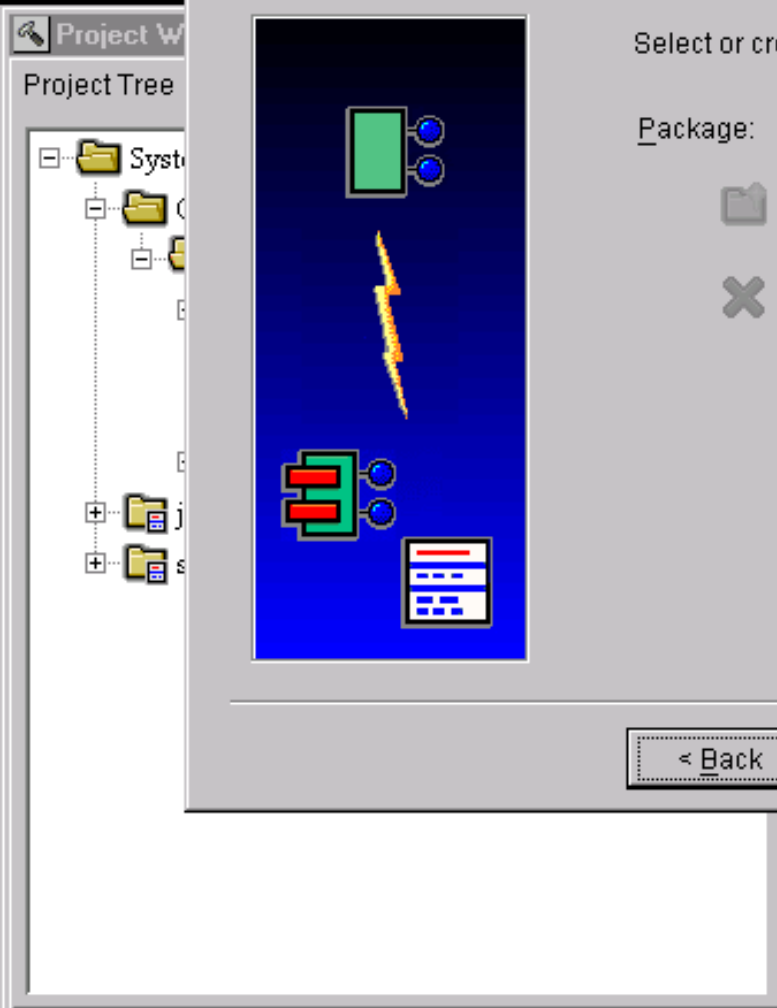
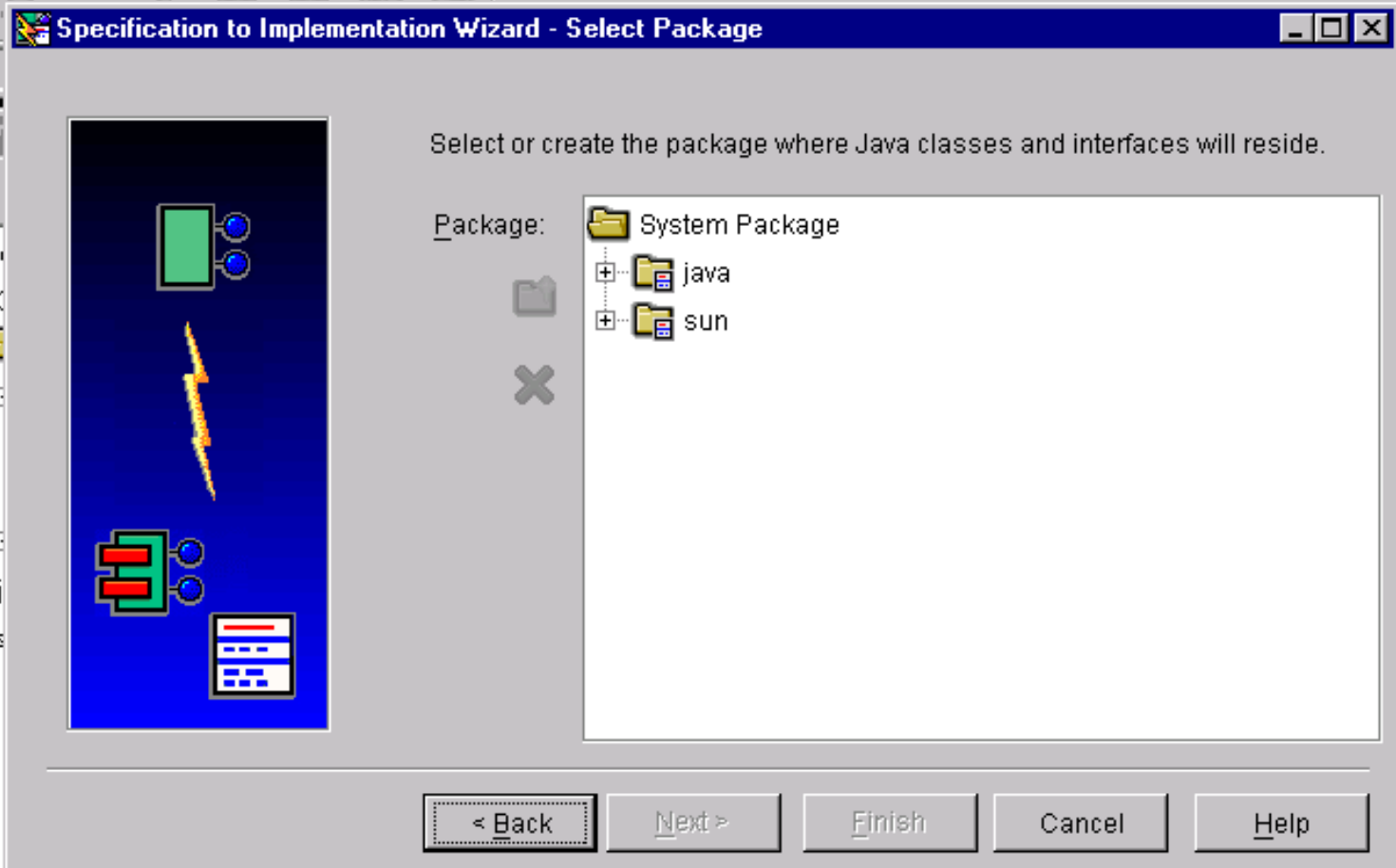
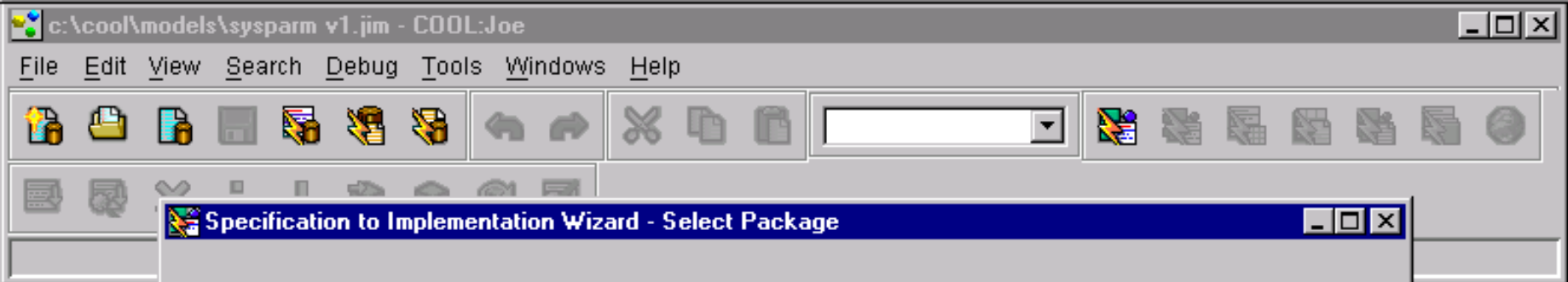


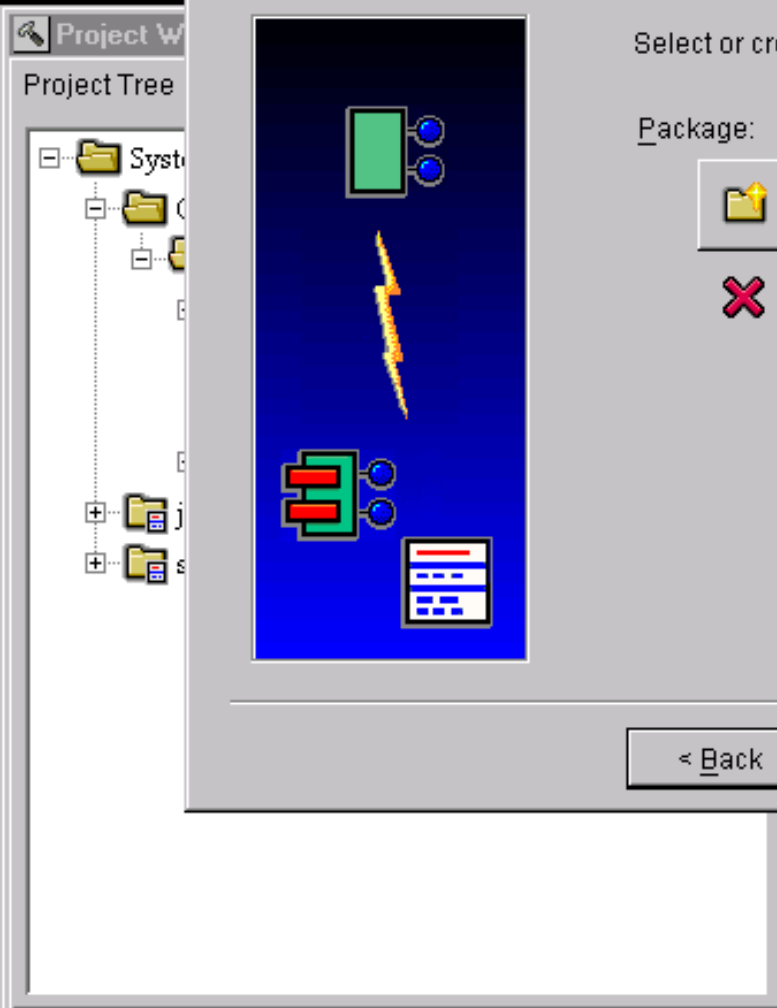
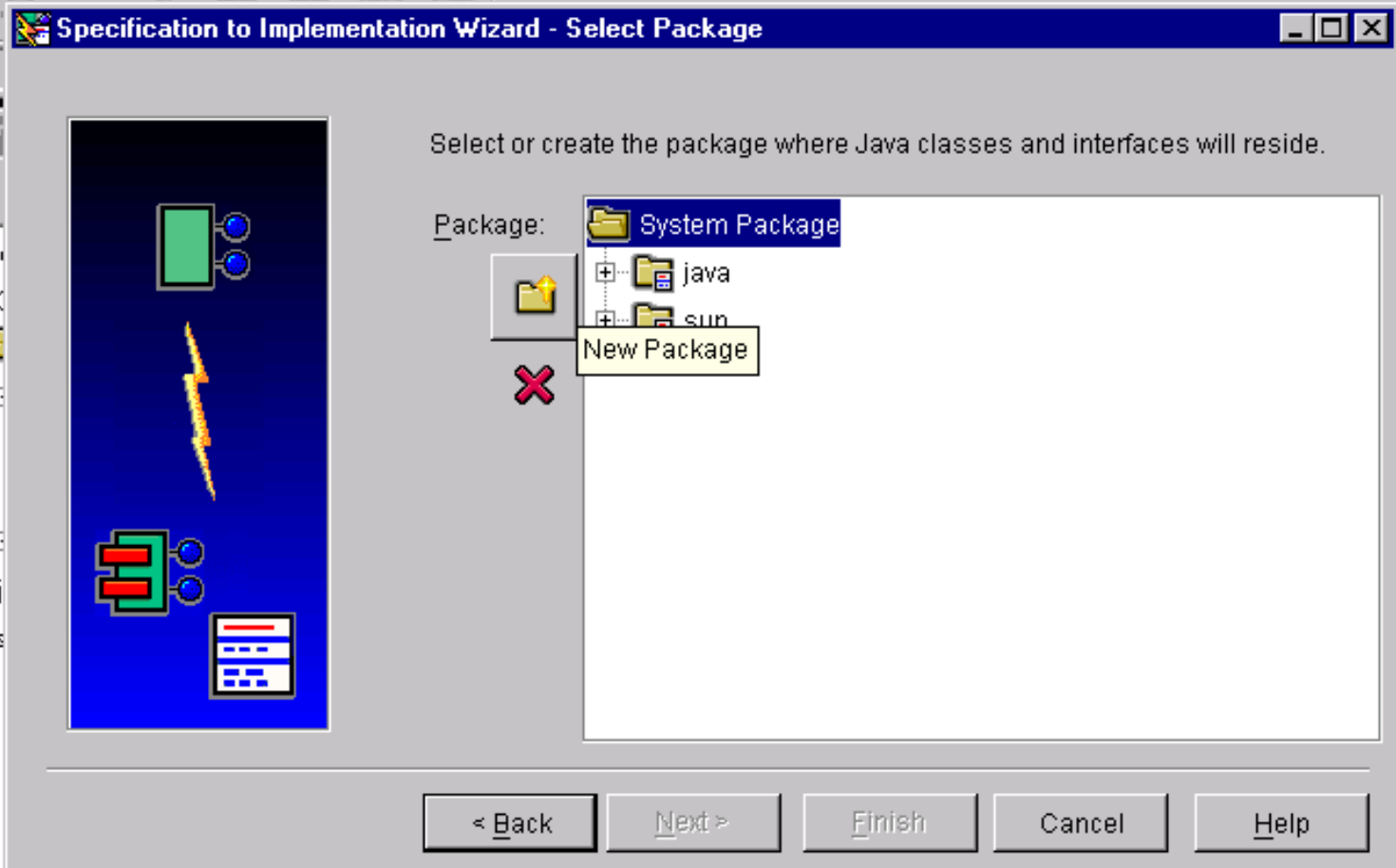
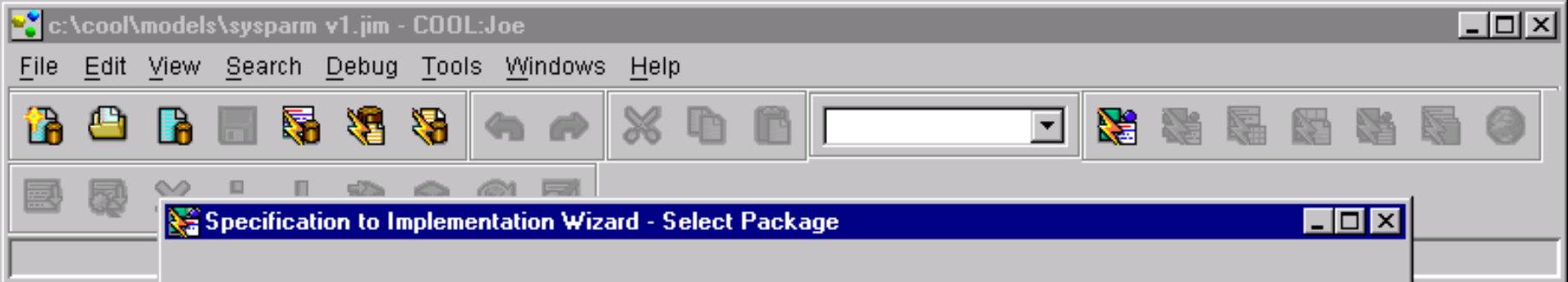


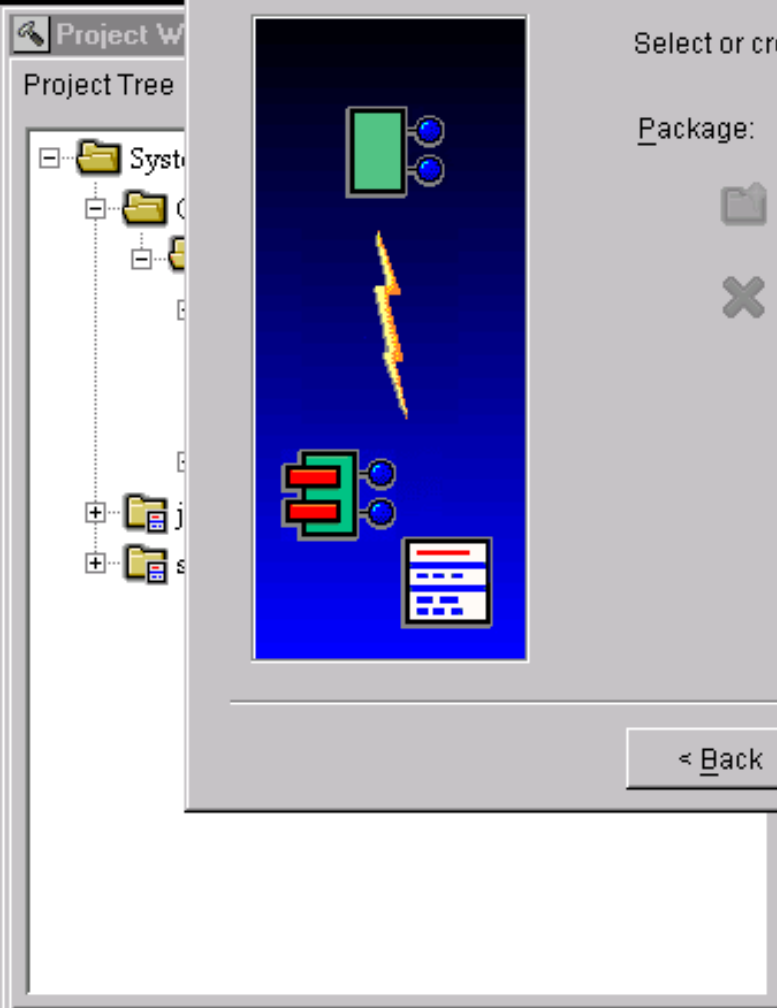
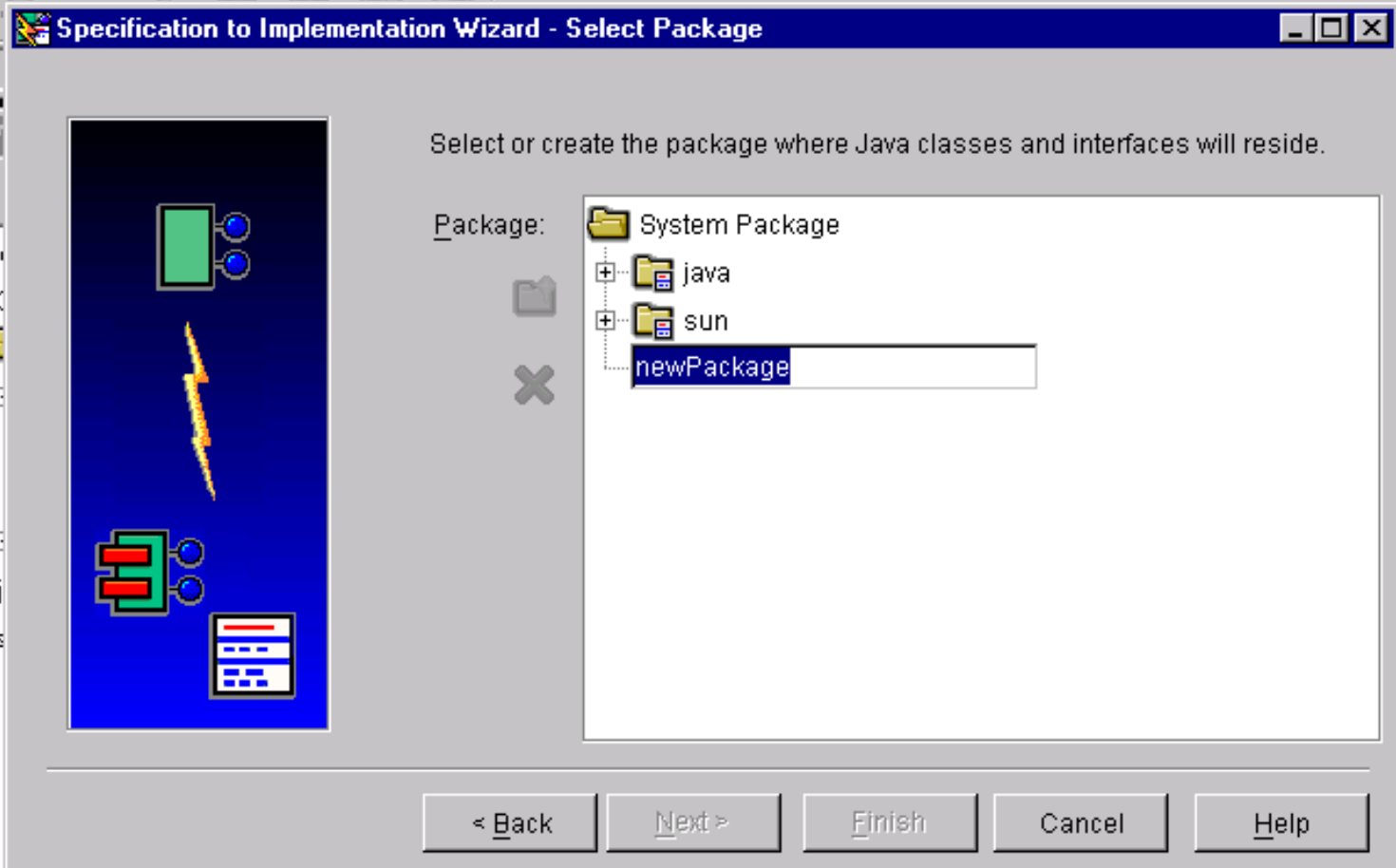
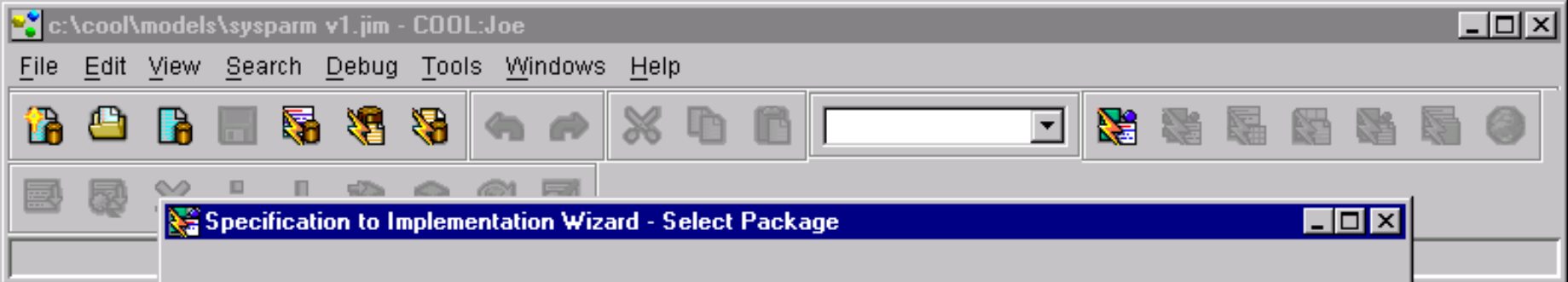


Now that you understand the component specification (yes we know it is a simple problem) you are ready to create the appropriate Java classes to implement.

COOL:Joe helps you do this with its Implement Specification wizard.








c:\cool\models\sysparm v1.jim - COOL:Joe

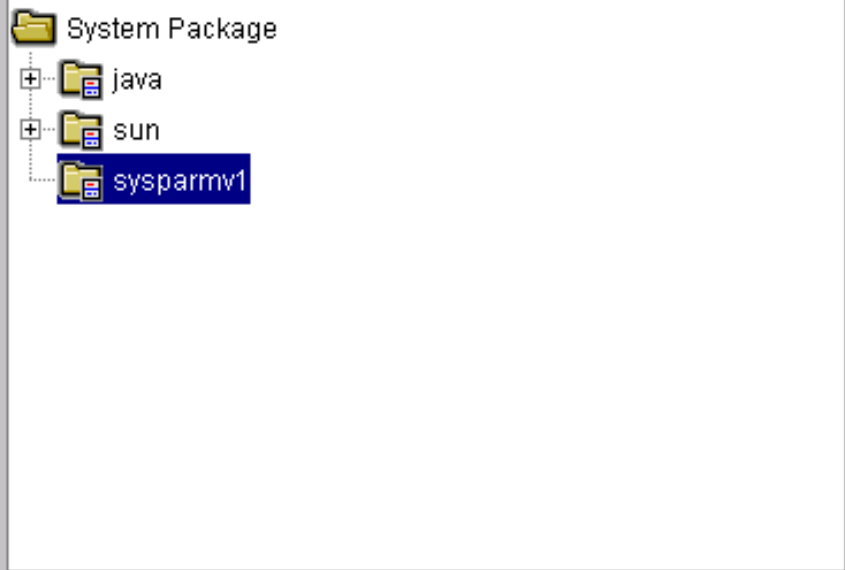
File Edit View Search Debug Tools Windows Help



Specification to Implementation Wizard - Select Package

Select or create the package where Java classes and interfaces will reside.

Package:



- System Package
 - java
 - sun
 - sysparmv1



< Back Next > Finish Cancel Help

Project W
Project Tree



- System Package
 - ...
 - ...
 - ...



c:\cool\models\sysparm v1.jim - COOL:Joe

File Edit View Search Debug Tools Windows Help

Specification to Implementation Wizard - Status: In Progress

Starting transformation.

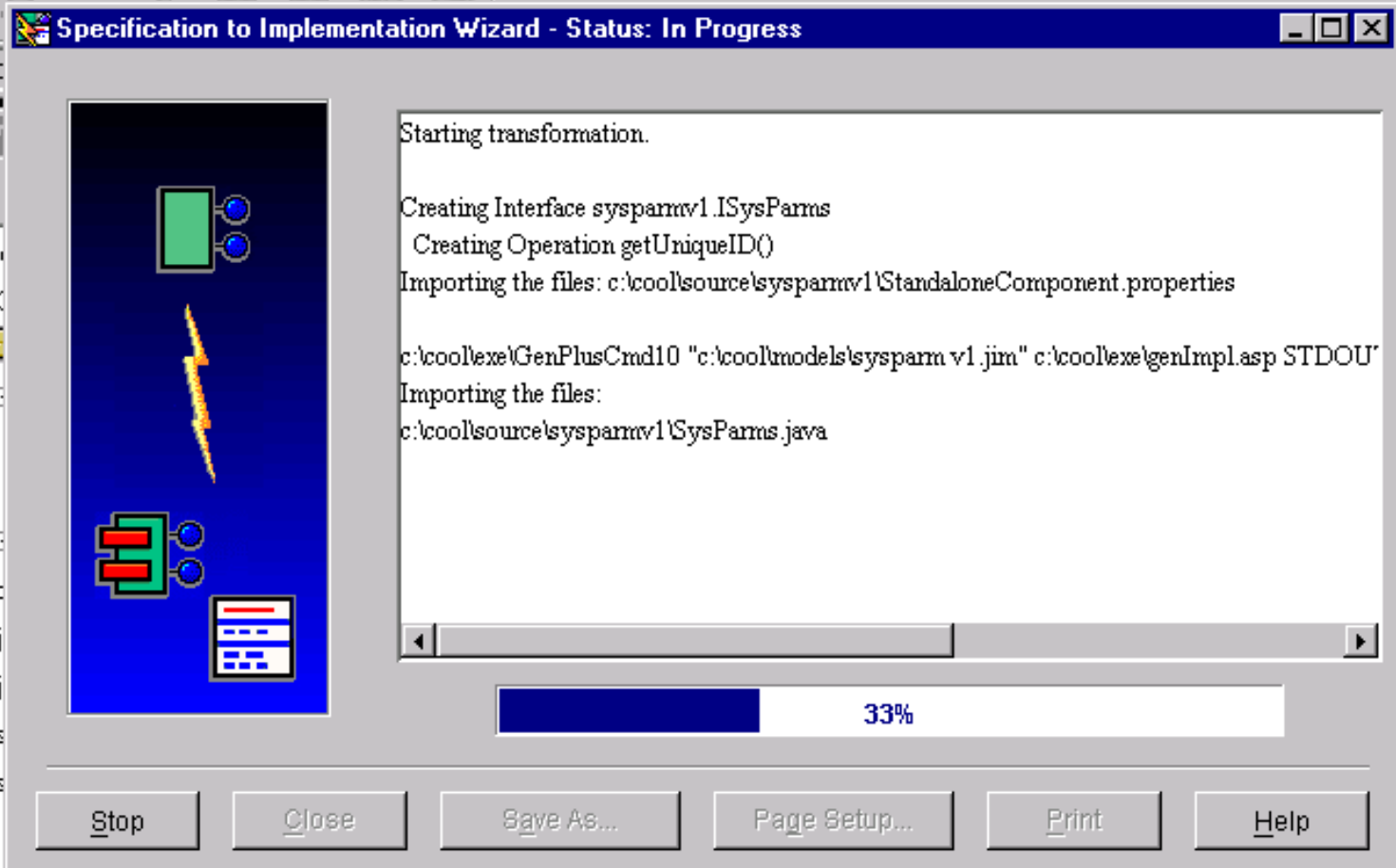
Creating Interface sysparmv1.ISysParams
Creating Operation getUniqueID()
Importing the files: c:\cool\source\sysparmv1\StandaloneComponent.properties

```
c:\cool\exe\GenPlusCmd10 "c:\cool\models\sysparm v1.jim" c:\cool\exe\genImpl.asp STDOUT
```

Importing the files:
c:\cool\source\sysparmv1\SysParams.java

33%

Stop Close Save As... Page Setup... Print Help



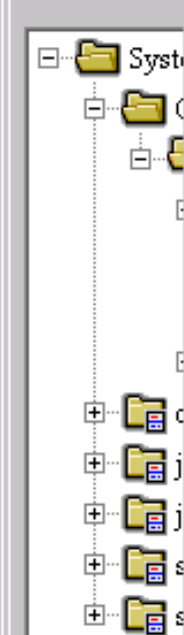
The image shows a software development environment window titled 'c:\cool\models\sysparm v1.jim - COOL:Joe'. A 'Specification to Implementation Wizard' dialog box is open, displaying the progress of a transformation. The wizard's status is 'In Progress' and the progress bar is at 33%. The main text area shows the following steps: 'Starting transformation.', 'Creating Interface sysparmv1.ISysParams', 'Creating Operation getUniqueID()', and 'Importing the files: c:\cool\source\sysparmv1\StandaloneComponent.properties'. A command line is shown: 'c:\cool\exe\GenPlusCmd10 "c:\cool\models\sysparm v1.jim" c:\cool\exe\genImpl.asp STDOUT'. Below the command line, it says 'Importing the files: c:\cool\source\sysparmv1\SysParams.java'. The wizard has a blue background with a lightning bolt icon and a diagram of a component. At the bottom, there are buttons for 'Stop', 'Close', 'Save As...', 'Page Setup...', 'Print', and 'Help'. In the background, a 'Project Tree' is visible on the left side of the main window.



Implementati

Project W

Project Tree

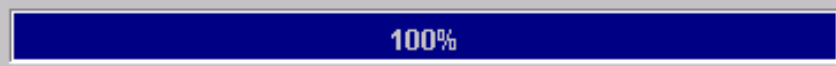


Starting transformation.

Creating Interface sysparmv1.ISysParms
Creating Operation getUniqueID()
Importing the files: c:\cool\source\sysparmv1\StandaloneComponent.properties

c:\cool\exe\GenPlusCmd10 "c:\cool\models\sysparm v1.jim" c:\cool\exe\genImpl.asp STDC
Importing the files:
c:\cool\source\sysparmv1\SysParms.java
c:\cool\source\sysparmv1\ISysParms.java
Created Component Implementation sysparmv1.SysParms

Implementation transformation completed.



Stop

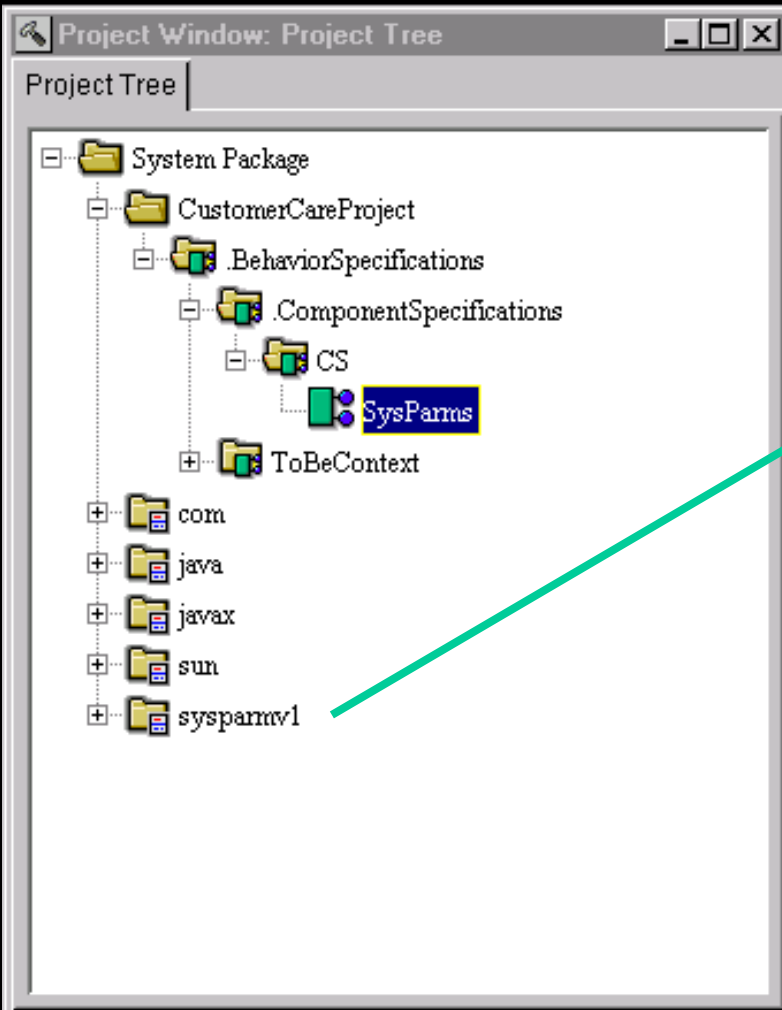
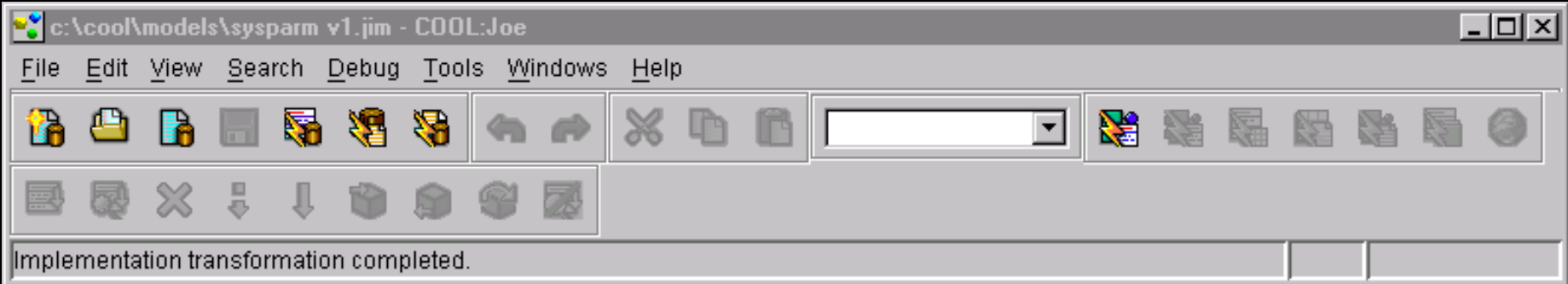
Close

Save As...

Page Setup...

Print

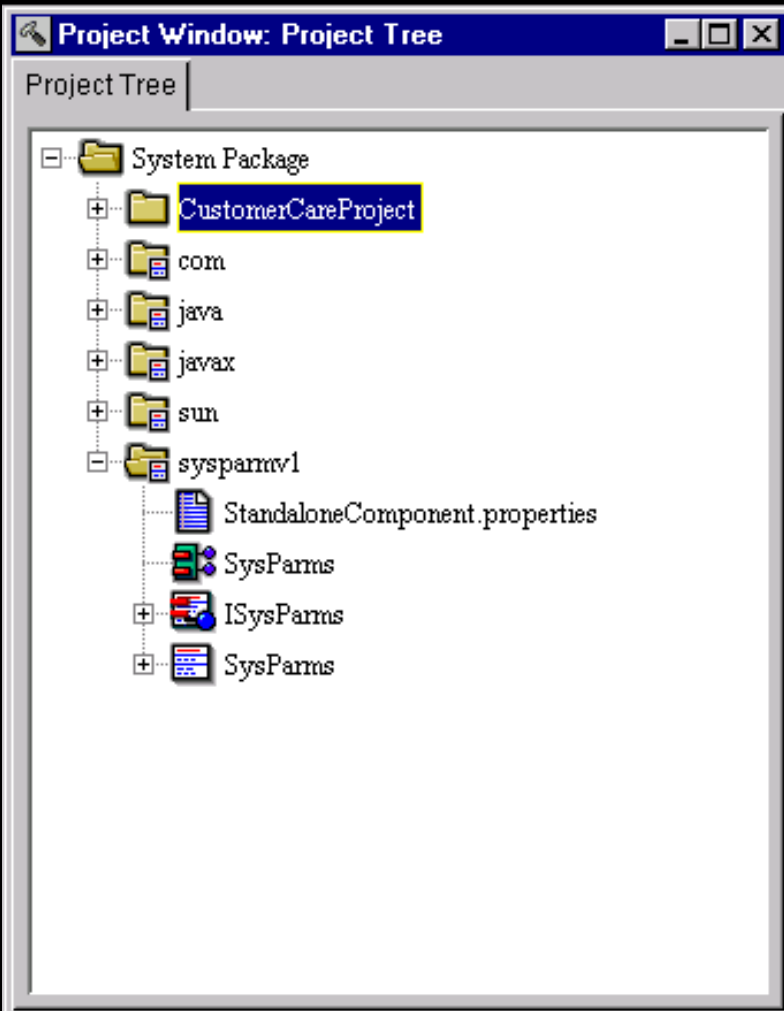
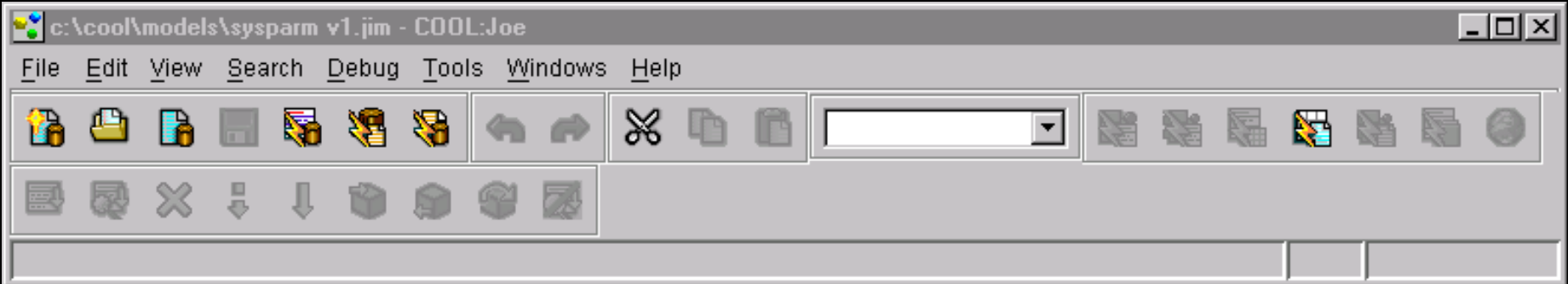
Help

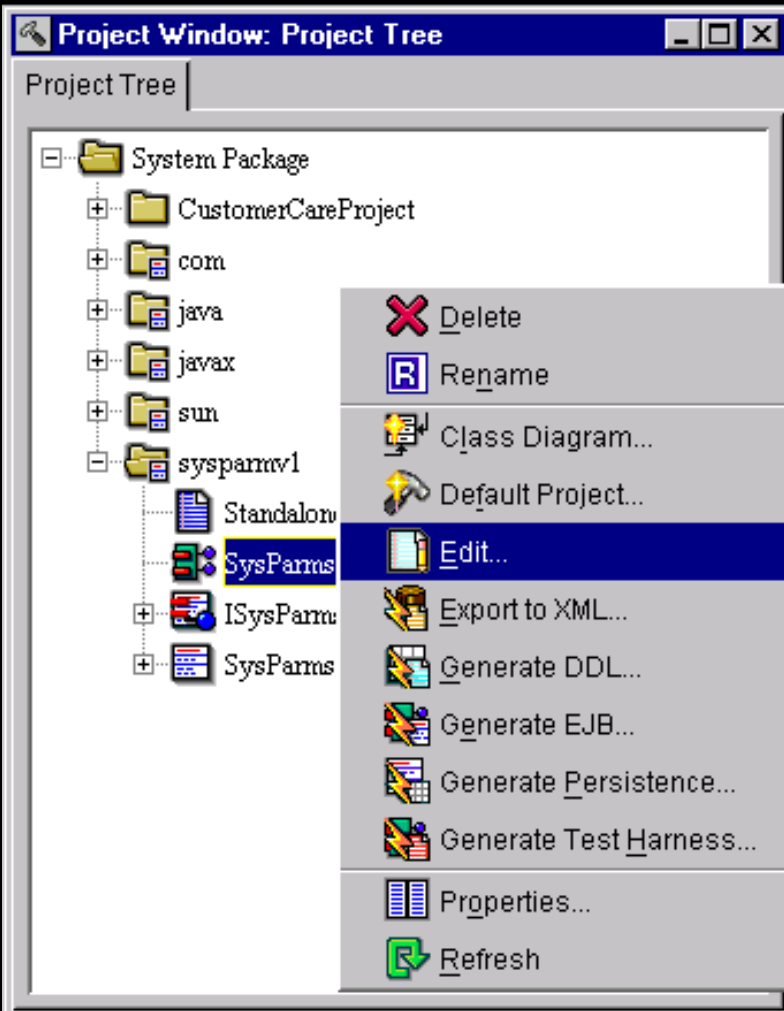
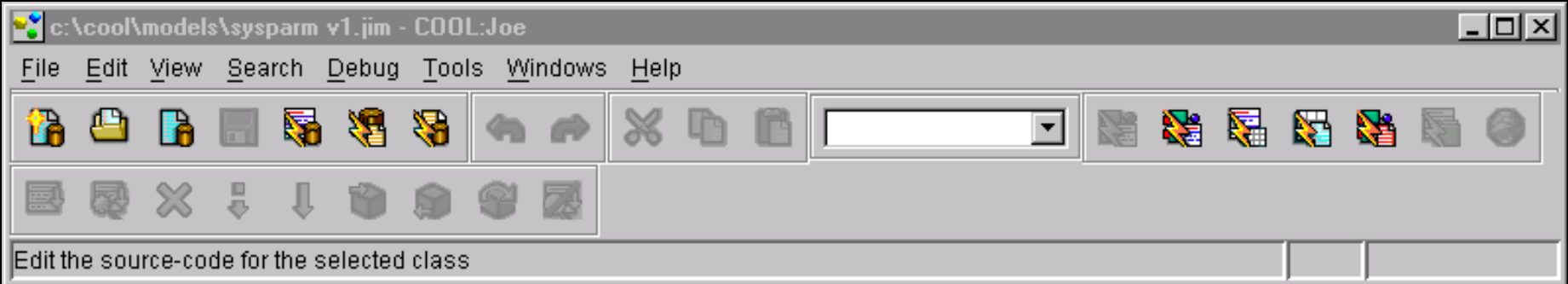


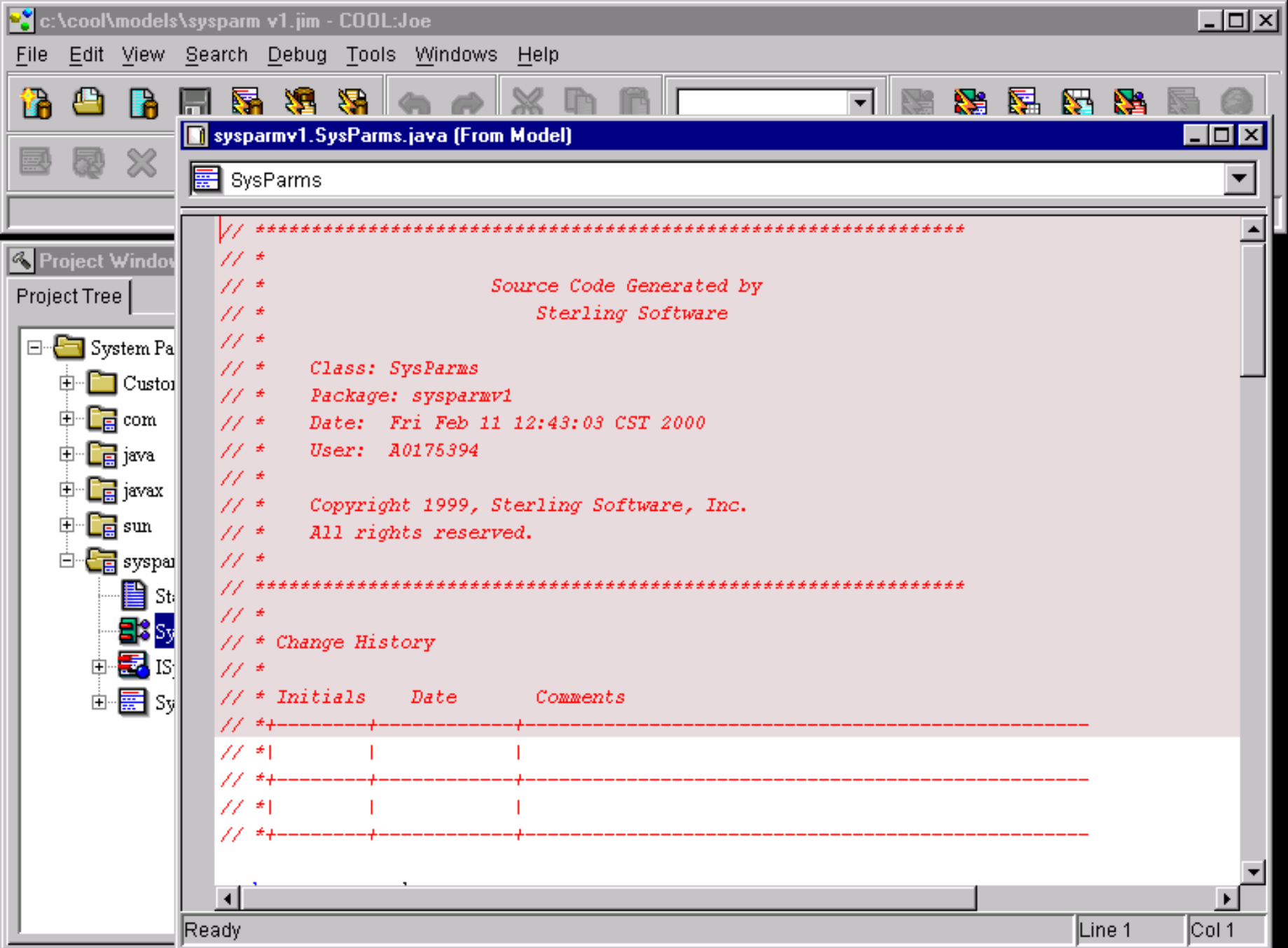
You now have a set of Java classes which represent your component specification.

An exact interpretation of the requirements!

Lets go see how Joe helps you implement your business logic. We will launch the COOL:Joe editor and look at the generated code for our one operation, GetUniqueID.







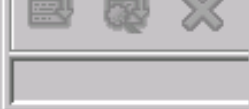
Ready

Line 1

Col 1



sysparmv1.SysParms.java (From Model)



Project Window

Project Tree

- System Pa
- + Custom
- + com
- + java
- + javax
- + sun
- syspar
 - St
 - Sy
 - IS
 - Sy

- SysParms
- SysParms
 - componentContext
 - commitIfNotEJB()
 - getComponentContext()
 - getUniqueID()
 - rollbackIfNotEJB()
 - setComponentContext(IComponentContext)
 - SysParms()

```

// *
// *   Copyright 1999, Sterling Software, Inc.
// *   All rights reserved.
// *
// *****
// *
// * Change History
// *
// * Initials   Date       Comments
// *+-----+-----+-----+
// *|       |       |       |
// *+-----+-----+-----+
// *|       |       |       |
// *+-----+-----+-----+

```


c:\cool\models\sysparm v1.jim - COOL:Joe

File Edit View Search Debug Tools Windows Help

sysparmv1.SysParms.java (From Model)

getUniqueID()

```
private transient IComponentContext componentContext = null;

/**
 * Successful
 * PreCondition:
 * Invoked
 * PostCondition:
 * Will return a unique integer
 */
public long getUniqueID ()
{
    return 0;
}

public SysParms()
{
    /*Default constructor*/
}

public void commitIfNotEJB()
```

Project Window

Project Tree

- System Pa
 - Custom
 - com
 - java
 - javax
 - sun
 - syspar
 - St
 - Sy
 - IS
 - Sy

Ready

Line 42 Col 24



sysparmv1.SysParms.java (From Model) *

getUniqueID()

Project Window

Project Tree

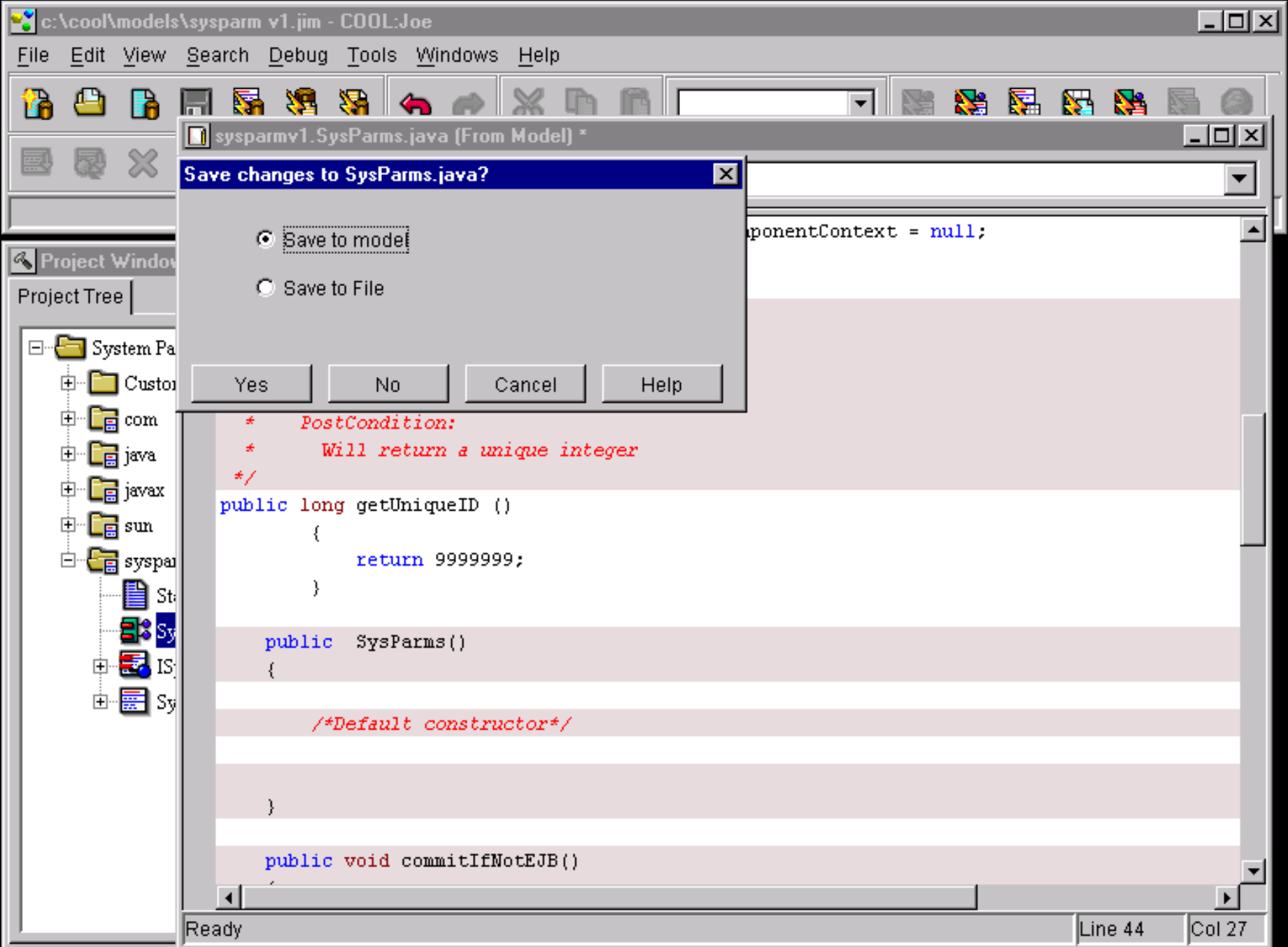
- System Pa
- + Custom
- + com
- + java
- + javax
- + sun
- syspar
 - St
 - Sy
 - + IS
 - + Sy

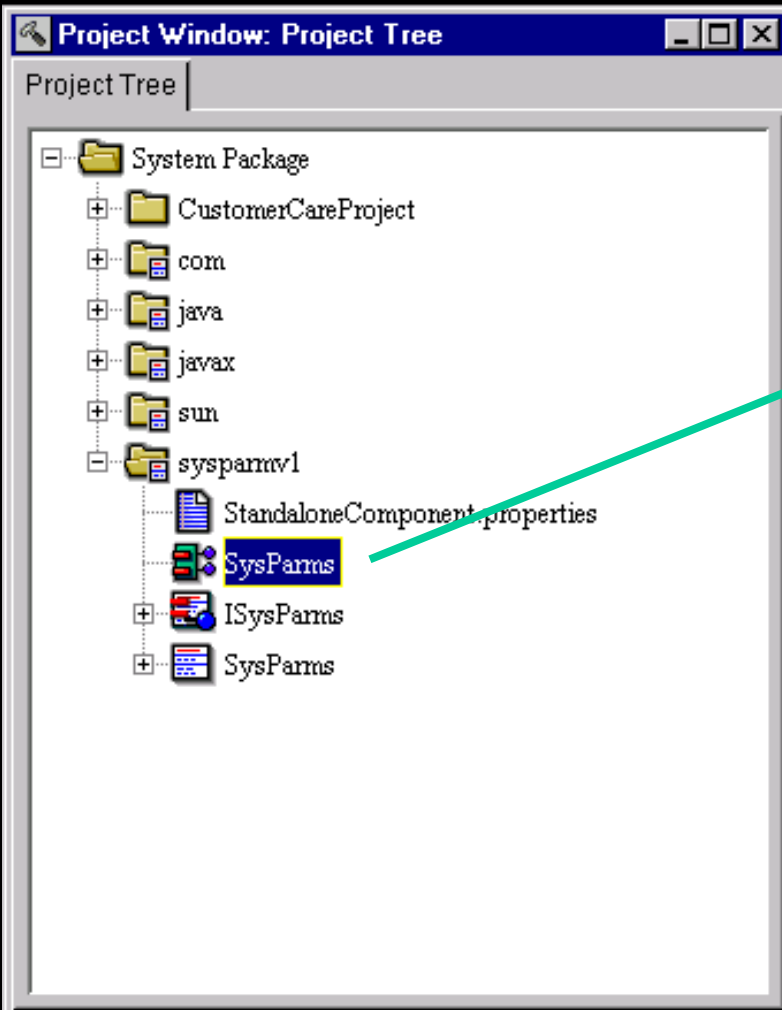
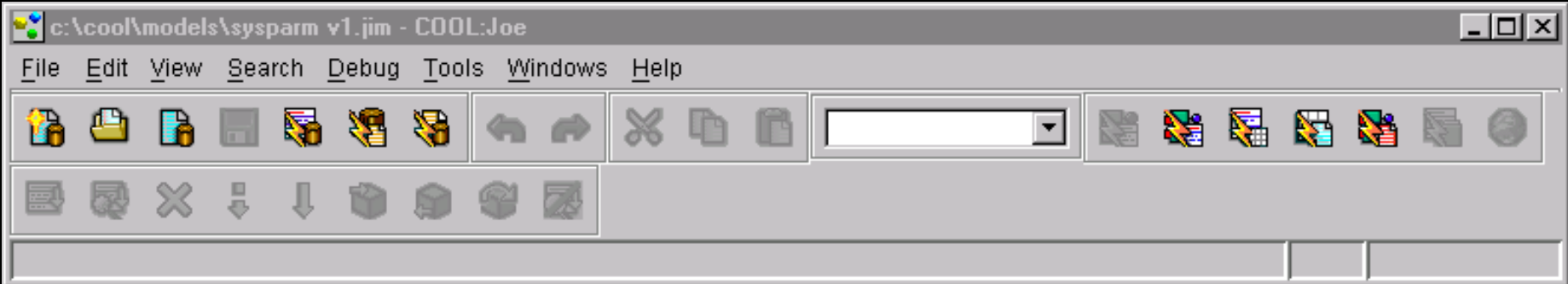
```
private transient IComponentContext componentContext = null;

/**
 * Successful
 * PreCondition:
 * Invoked
 * PostCondition:
 * Will return a unique integer
 */
public long getUniqueID ()
{
    return 9999999;
}

public SysParms()
{
    /*Default constructor*/
}

public void commitIfNotEJB()
```





Ok, you have written your business logic and saved it to the COOL:Joe repository. Now you need to think about how to test your code to make sure all is 'Golden'.

COOL:Joe helps by providing you with the ability to generate a simple User Interface to drive your server code.

Lets take a look at the Test Harness Wizard.

c:\cool\models\sysparm v1.jim - COOL:Joe

File Edit View Search Debug Tools Windows Help

Generate a Test Harness that tests methods implemented by the Component

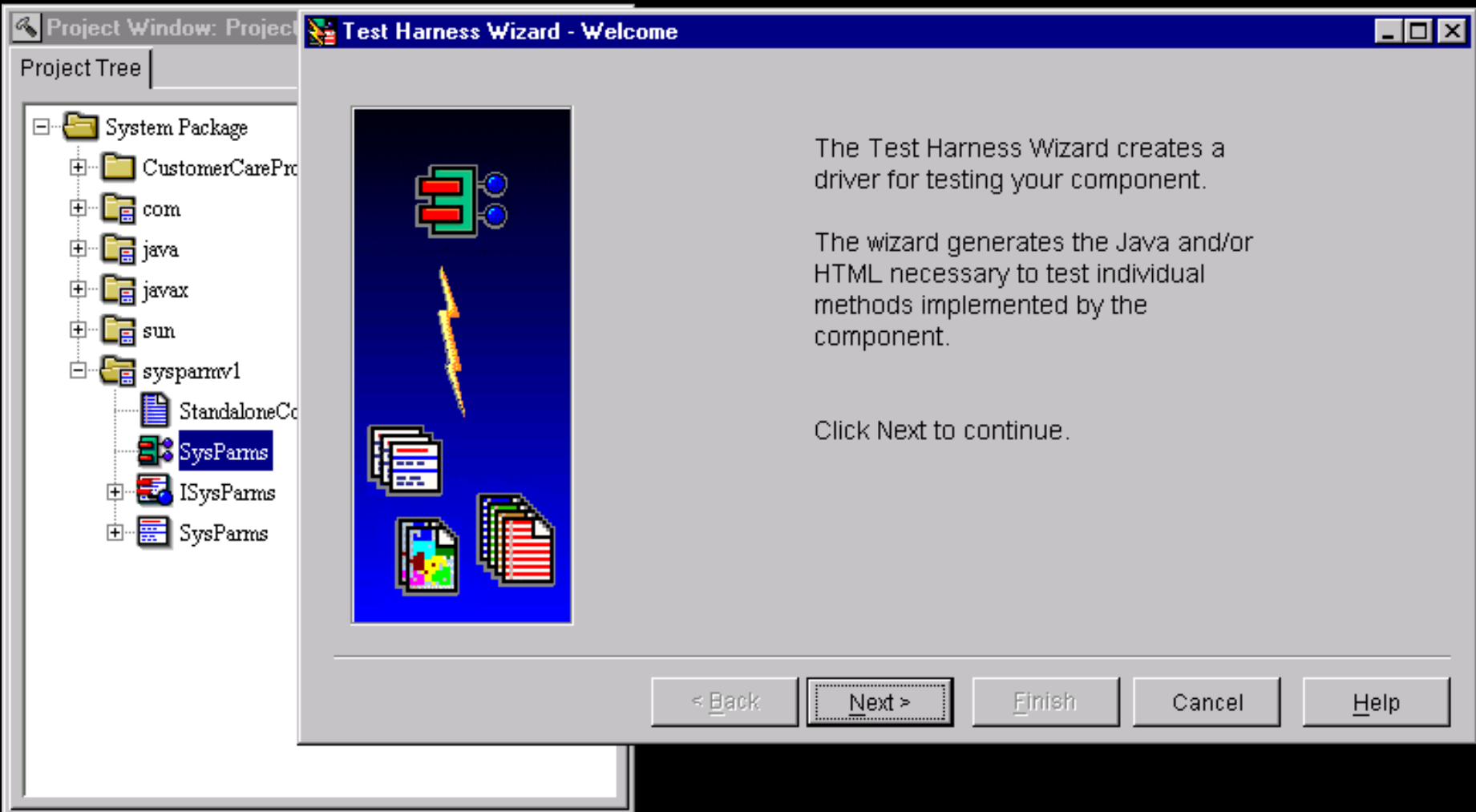
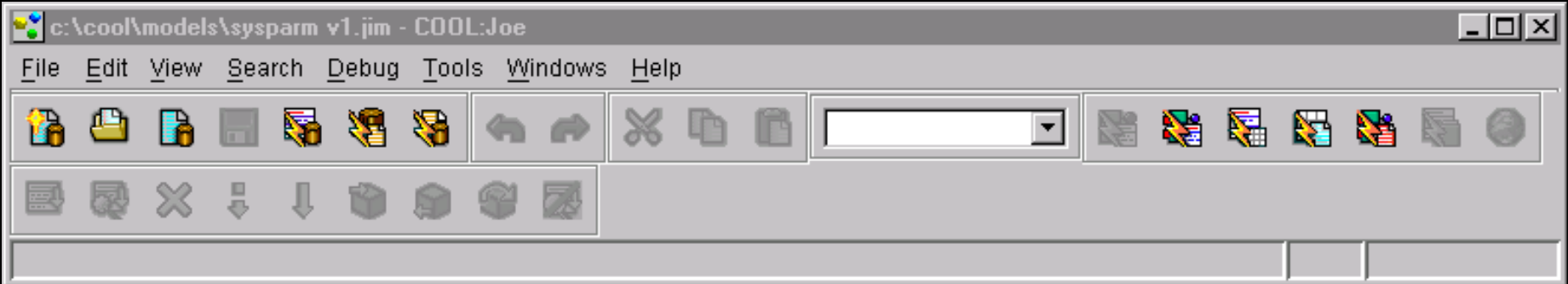
Project Window: Project Tree

Project Tree

- System Package
 - CustomerCareProject
 - com
 - java
 - javax
 - sun
 - sysparmv
 - Stand
 - SysP
 - ISysF
 - SysP

Context Menu:

- Delete
- Rename
- Class Diagram...
- Default Project...
- Edit...
- Export to XML...
- Generate DDL...
- Generate EJB...
- Generate Persistence...
- Generate Test Harness...**
- Properties...
- Refresh



The Test Harness Wizard creates a driver for testing your component.

The wizard generates the Java and/or HTML necessary to test individual methods implemented by the component.

Click Next to continue.

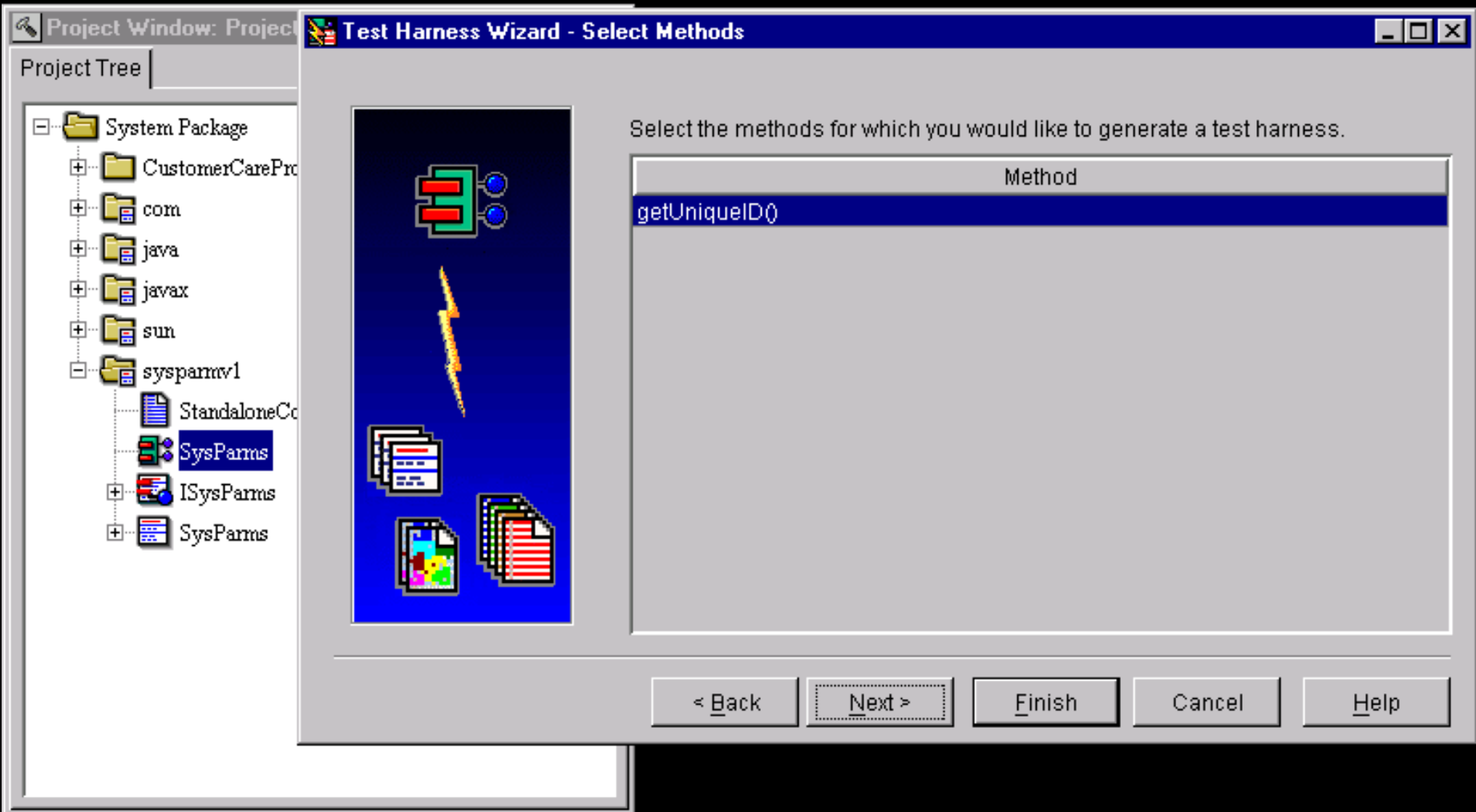
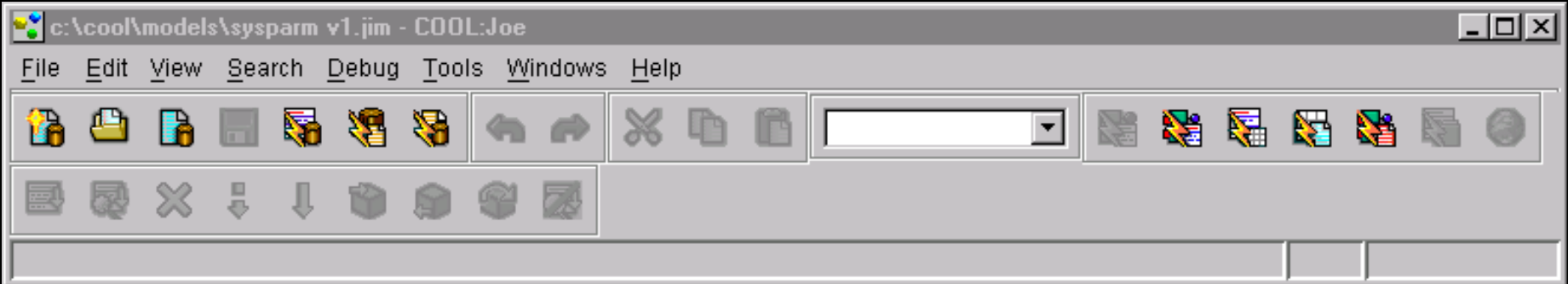
< Back

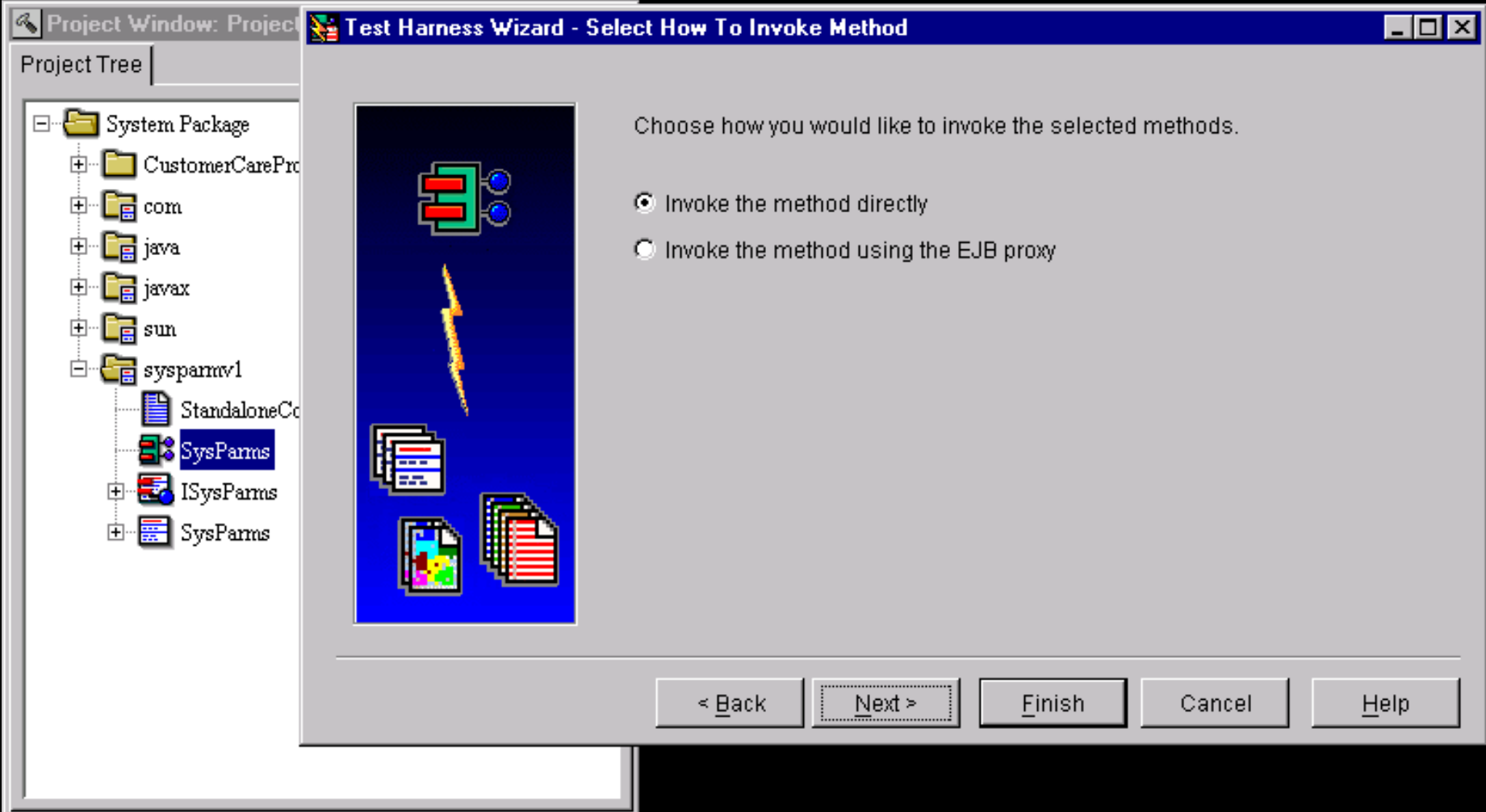
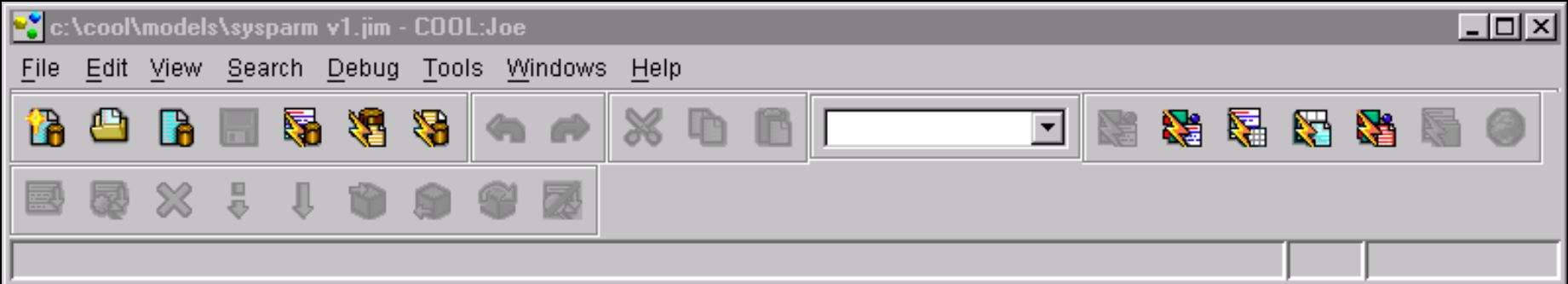
Next >

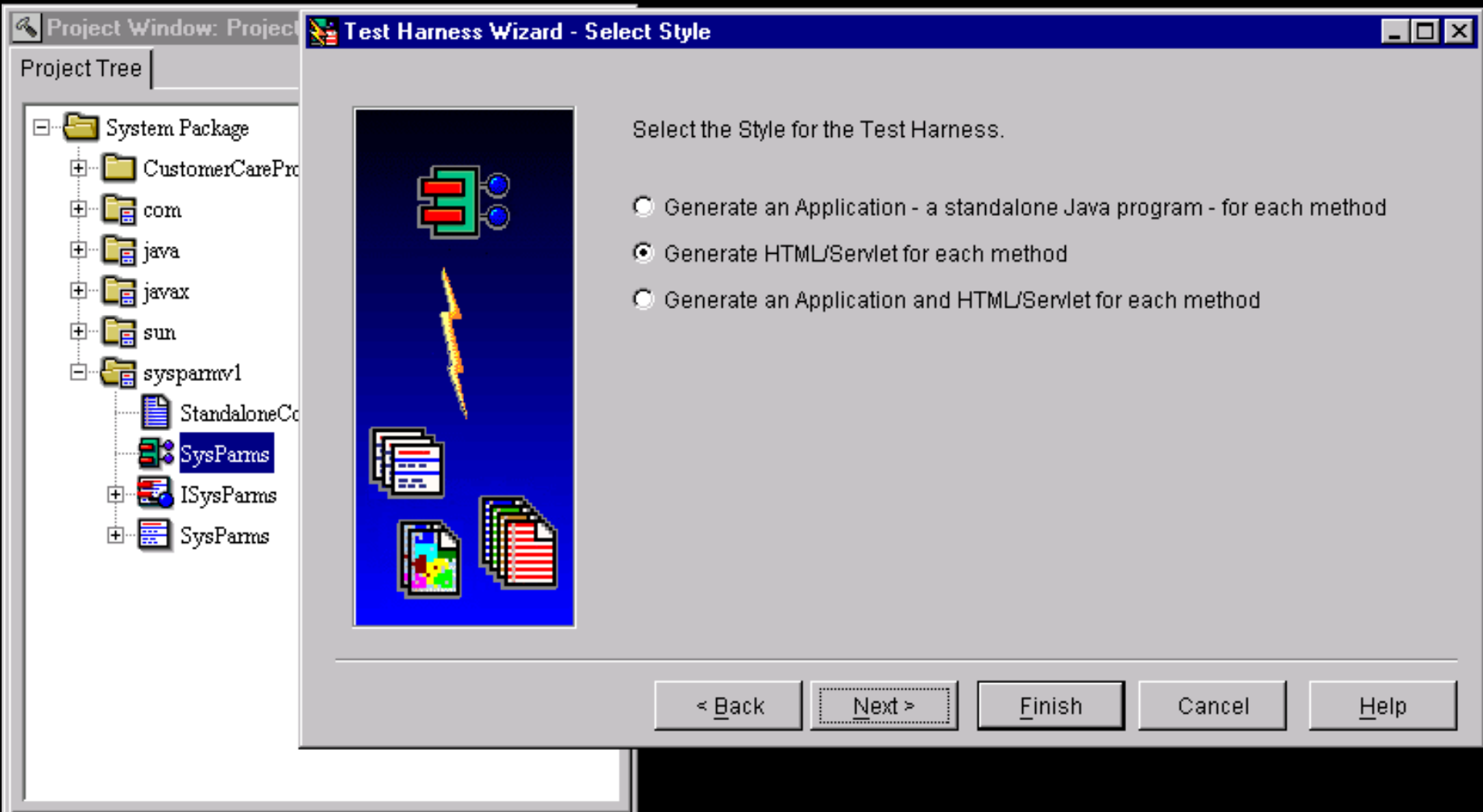
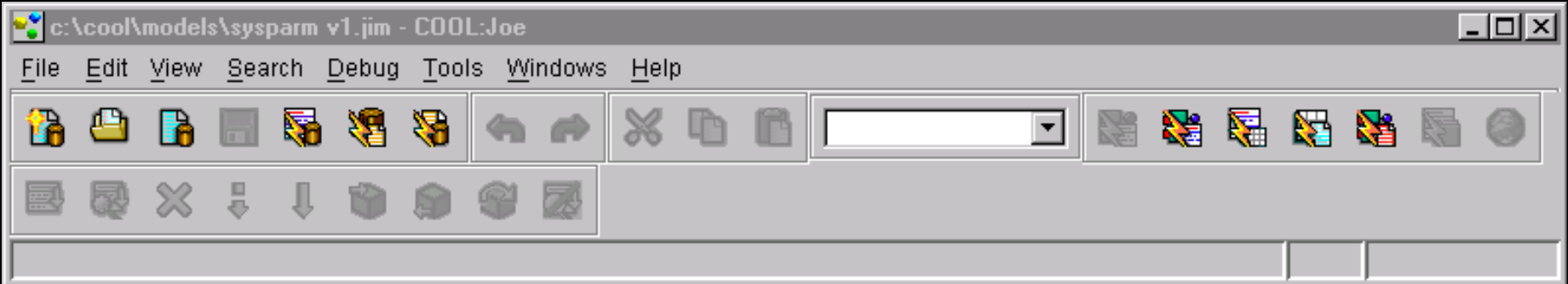
Finish

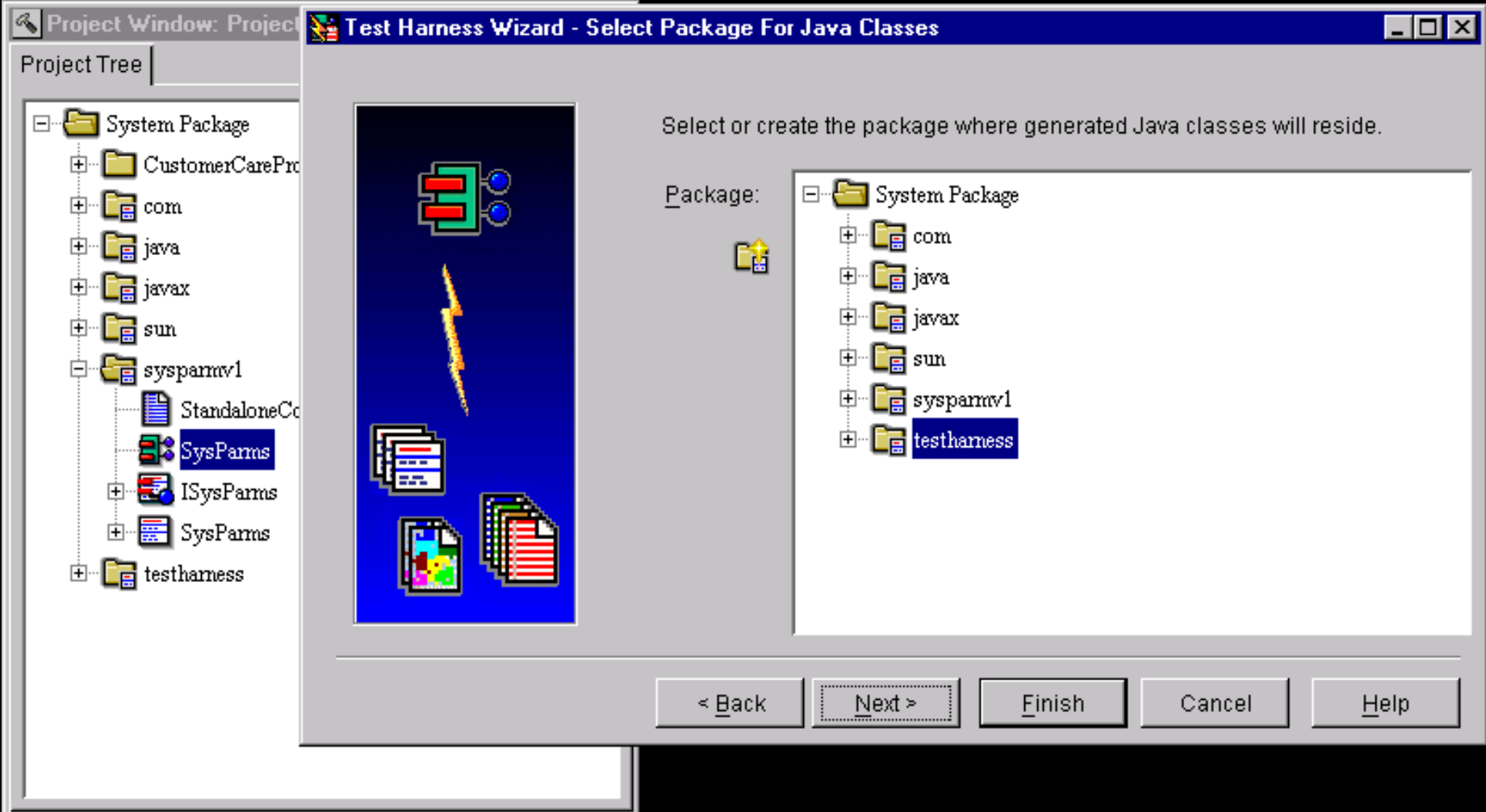
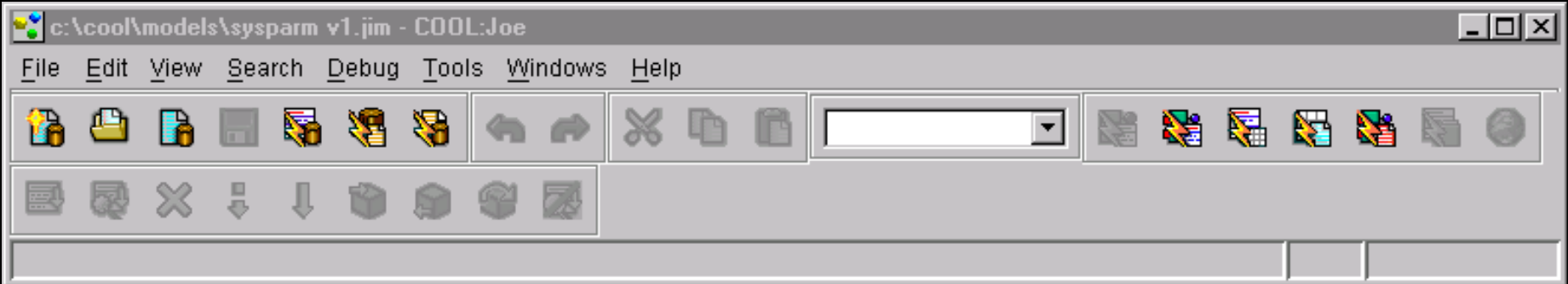
Cancel

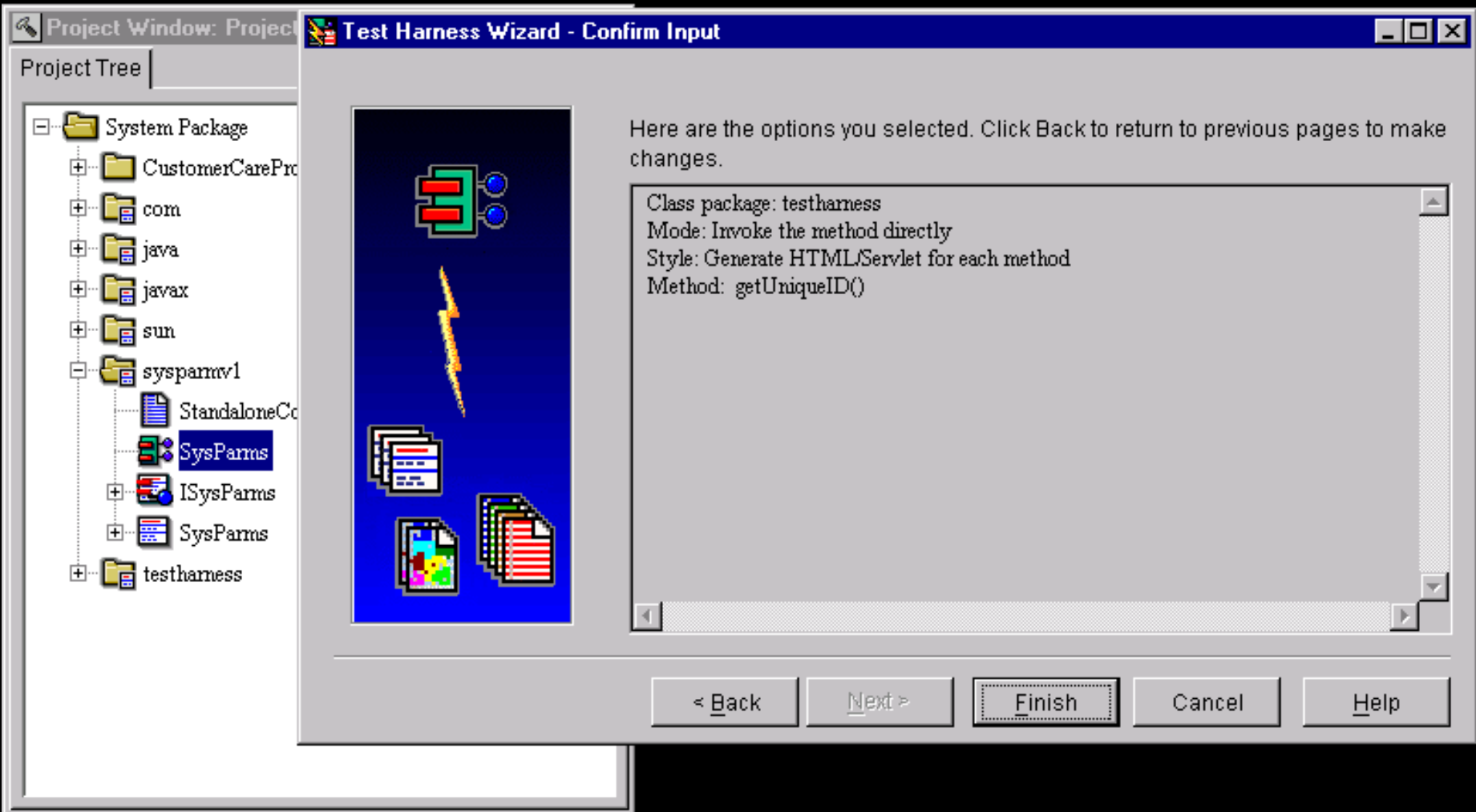
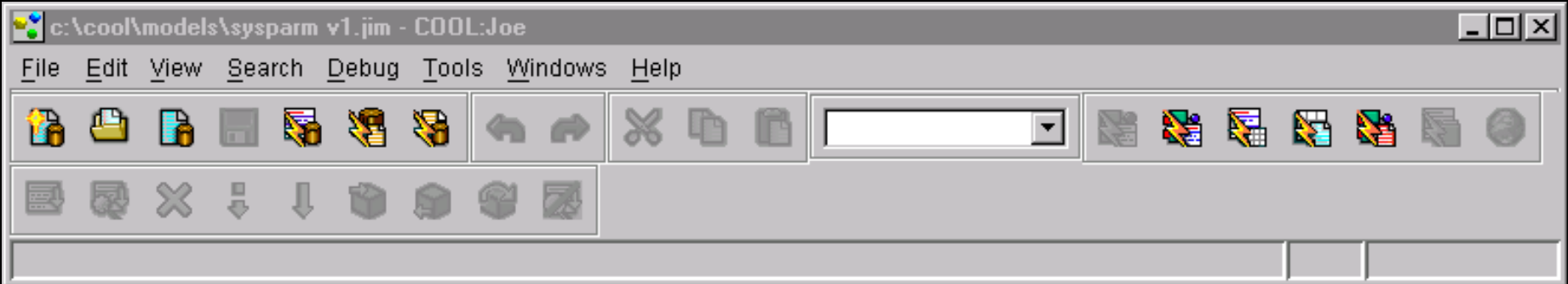
Help












c:\cool\models\sysparm v1.jim - COOL:Joe

File Edit View Search Debug Tools Windows Help




Project Window: Project

Project Tree

- System Package
 - CustomerCarePro
 - com
 - java
 - javax
 - sun
 - sysparmv1
 - StandaloneCo
 - SysParms**
 - ISysParms
 - SysParms
 - testharness

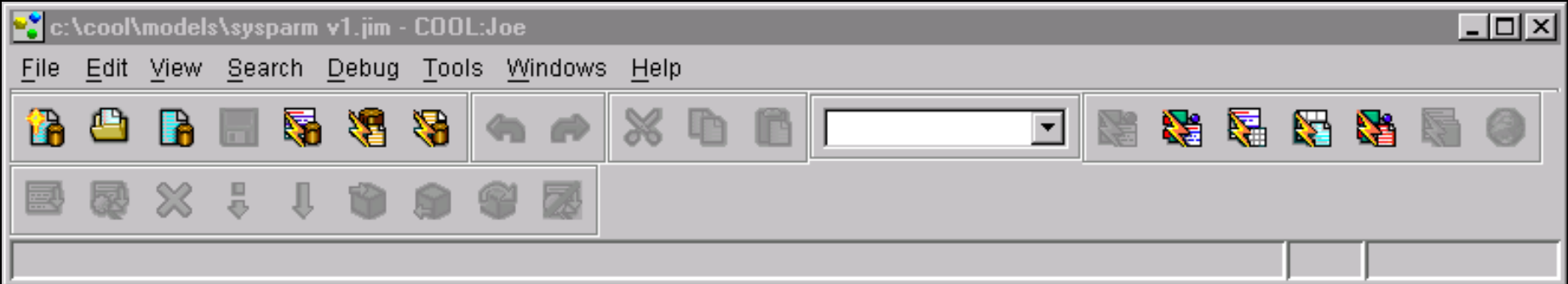
Test Harness Wizard - Status: In Progress



Starting UI Object Creation.
Creating UI objects for getUniqueID()
UI object creation complete for getUniqueID()
UI Object Creation Complete.
Creation of XML file complete
Creation of Java file(s) complete. Import is under way.
c:\cool\source\ConvertType.java
c:\cool\source\ArrayWrapper.java
c:\cool\source\VectorWrapper.java
c:\cool\source\EnumerationWrapper.java
c:\cool\source\Wrapper.java
c:\cool\source\CheckForData.java
6 Java files Imported
17 Miscellaneous files Imported
Committing changes in Model.

80%

Stop Close Save As... Page Setup... Print Help



Project Window: Project

Project Tree

- System Package
 - CustomerCareProc
 - com
 - java
 - javax
 - sun
 - sysparmv1
 - StandaloneCo
 - SysParms**
 - ISysParms
 - SysParms
 - testharness

Test Harness Wizard - Status: Finished



UI Object Creation Complete.
Creation of XML file complete
Creation of Java file(s) complete. Import is under way.
c:\cool\source\ConvertType.java
c:\cool\source\ArrayWrapper.java
c:\cool\source\VectorWrapper.java
c:\cool\source\EnumerationWrapper.java
c:\cool\source\Wrapper.java
c:\cool\source\CheckForData.java
6 Java files Imported
17 Miscellaneous files Imported
Committing changes in Model.
Model changes due to Generation stored in the model.
Removing UI objects created for generation.
UI object removal complete.

100%

Stop

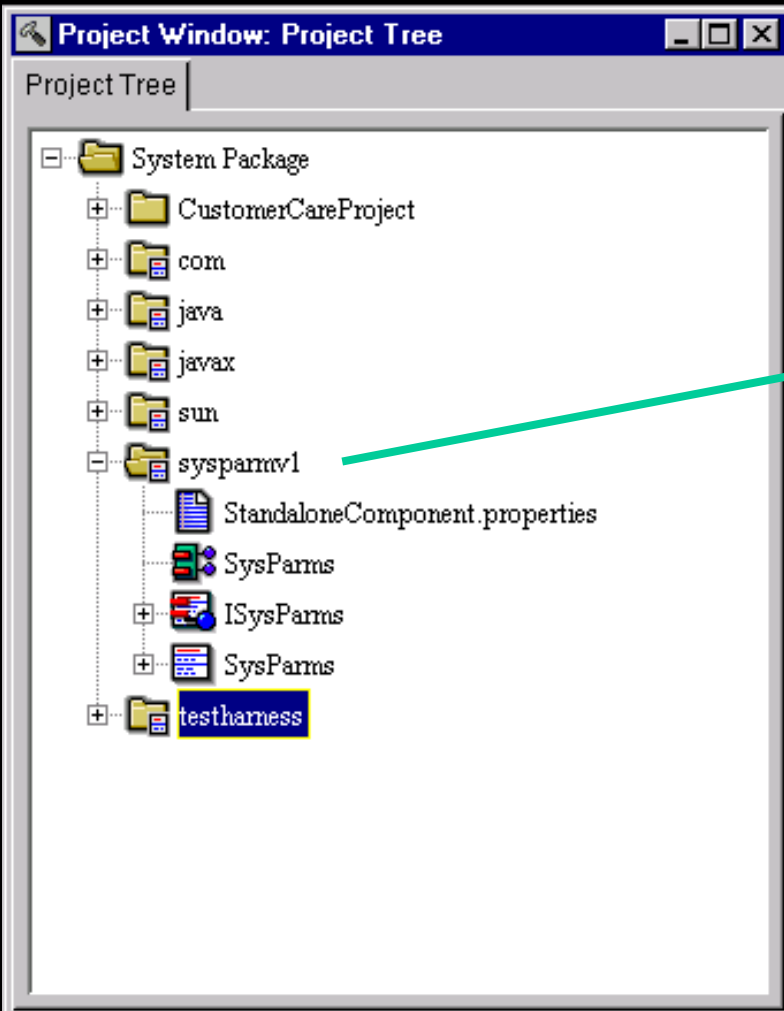
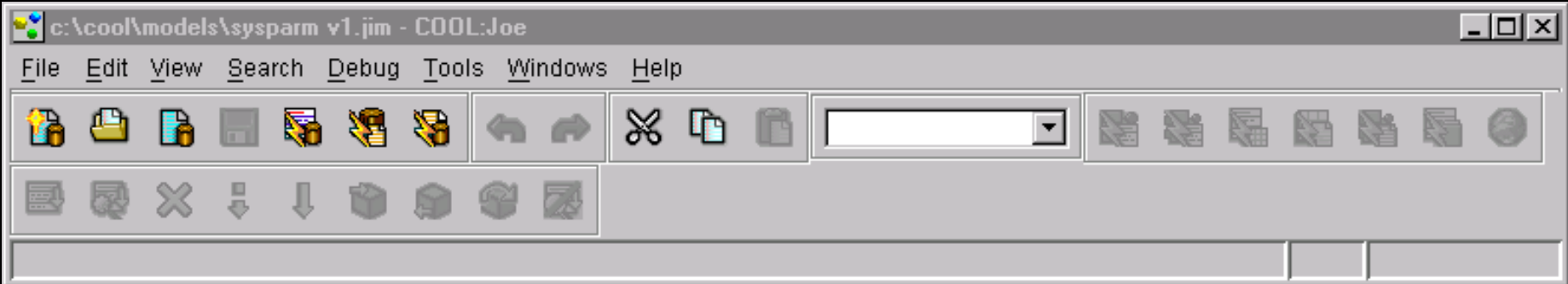
Close

Save As...

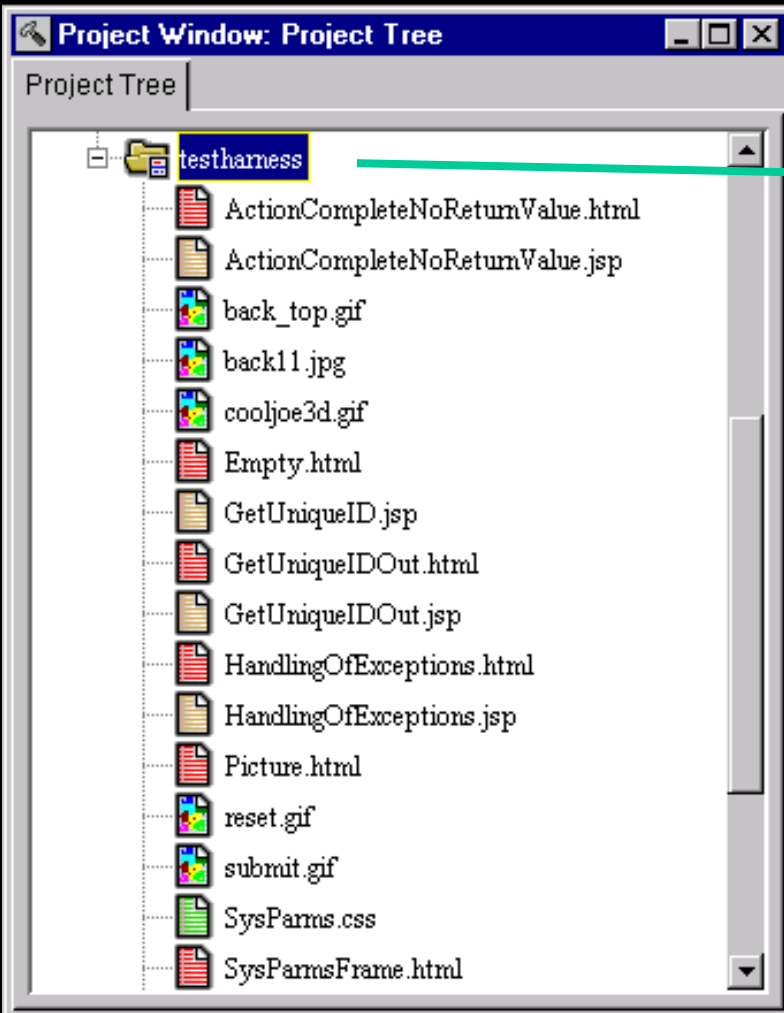
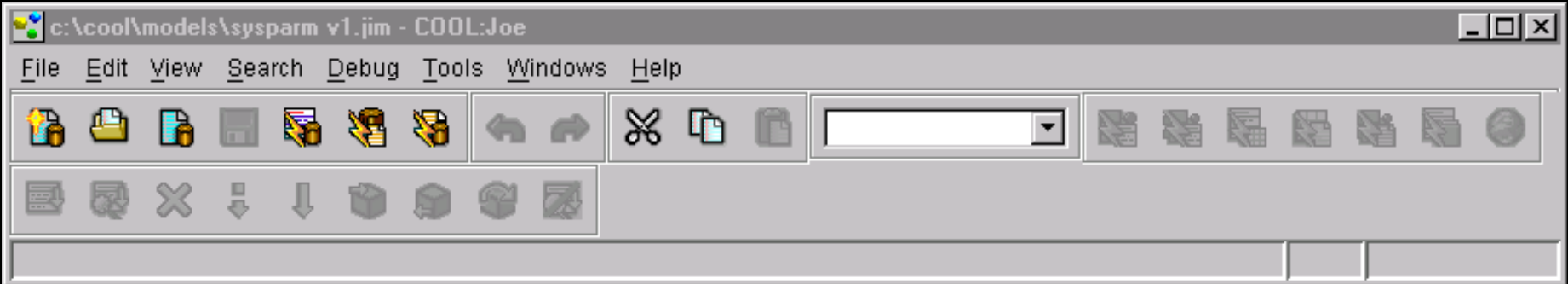
Page Setup...

Print

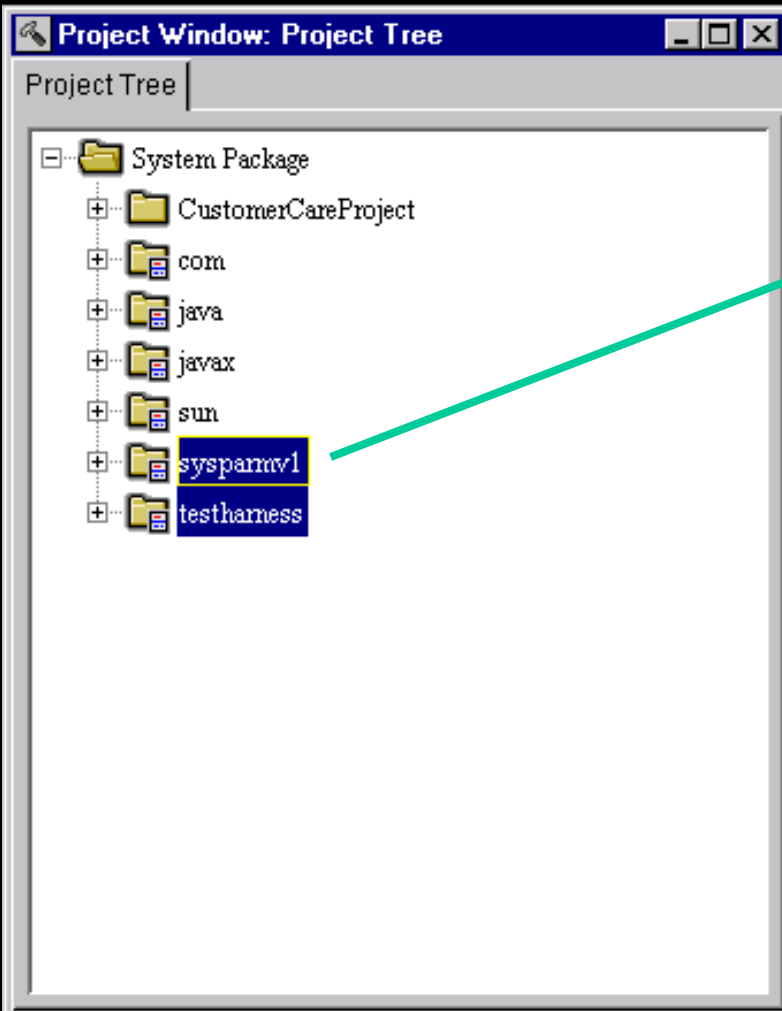
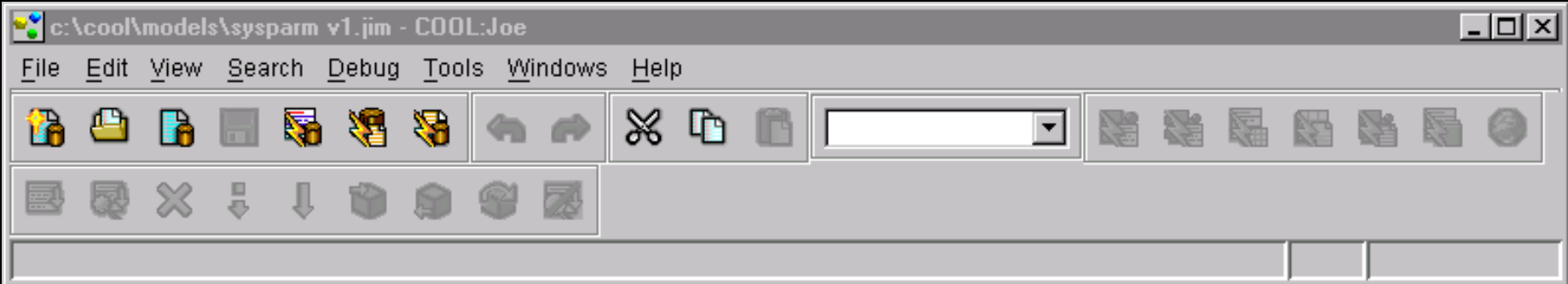
Help



Ok, you now have a User Interface made up of HTML and JSP code to drive your business component.



Here is what COOL:Joe generated for you.




Your next step is to get all of your code compiled, jared up and ready to test.

You guessed it. . . COOL:Joe makes this easy too!

Lets run the Default Project wizard and get our code ready to be compiled and tested.

c:\cool\models\sysparm v1.jim - COOL:Joe

File Edit View Search Debug Tools Windows Help



DefaultProject for the selected object

Project Window: Project Tree

Project Tree

- System Package
 - CustomerCareProject
 - com
 - java
 - javax
 - sun
 - sysparm**
 - testharr

- Create
- Cut
- Copy
- Delete
- Rename
- Class Diagram...
- Default Project...**
- Export to XML...
- Properties...
- Refresh

Name Project

sysparmv1 Project

Enter a new name for the project above or use the default name provided. A new project will be created and default packaging will be performed for the object that you selected.

OK Cancel

Project

- +
- +
- +
- +
- +
- +
- +
- +
- +
- +

com

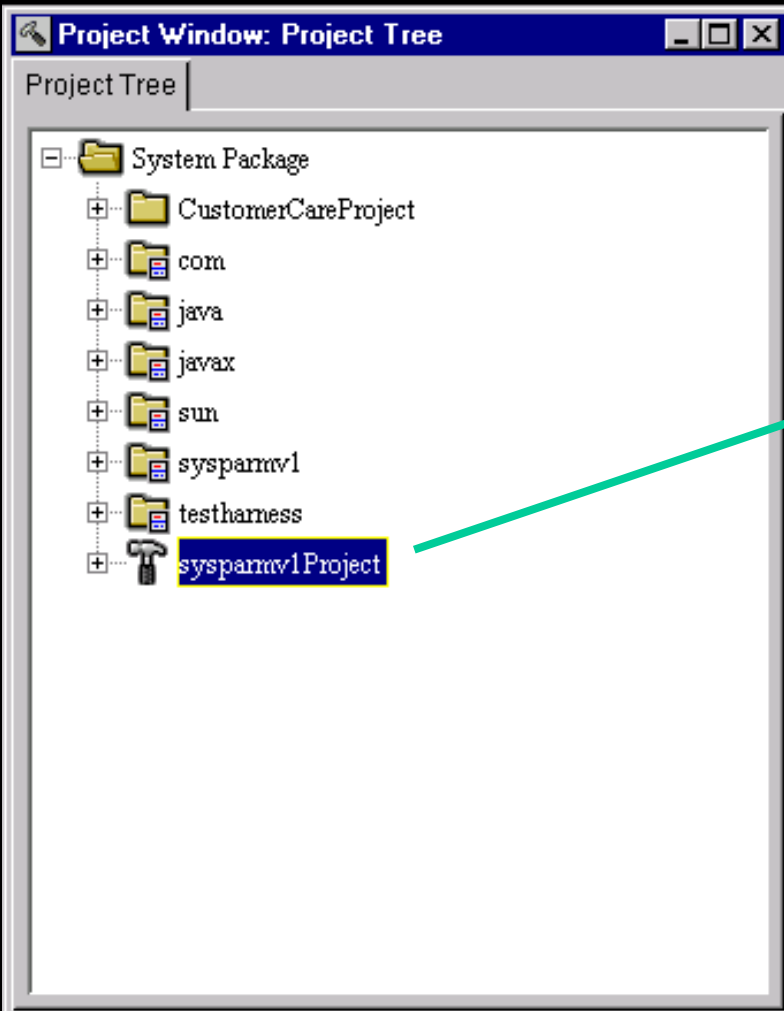
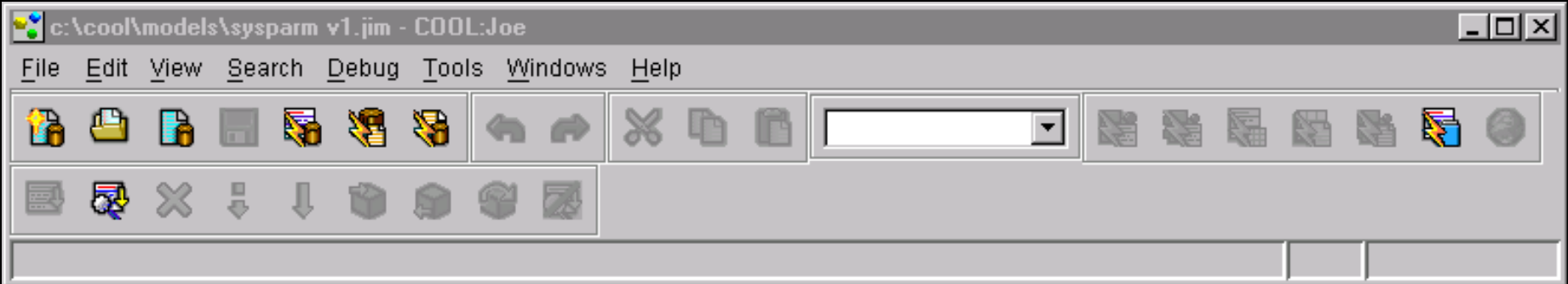
java

javax

sun

sysparmv1

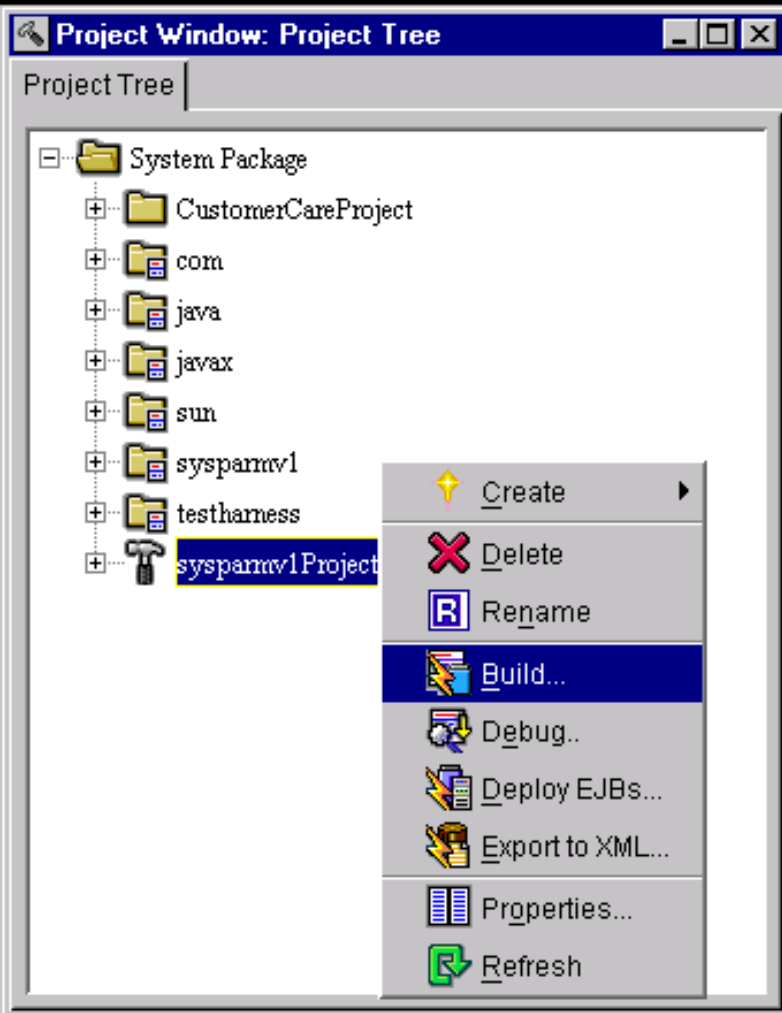
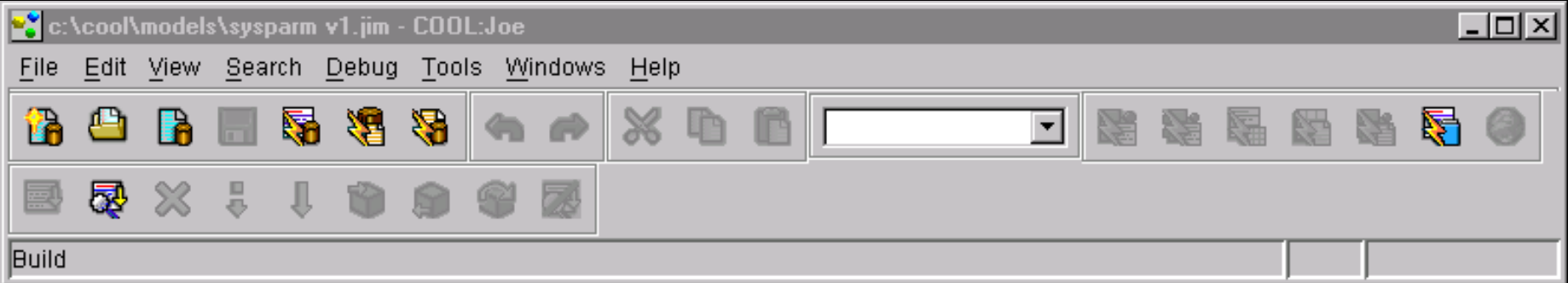
testharness

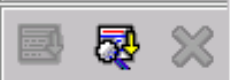


COOL:Joe uses a special project folder to help you keep track of what code you want to compile and test.

By running the Build wizard you don't need to know anything about how to compile, and jar your project, COOL:Joe does it for you.

Simple!





Build Wizard - Welcome



The Build Wizard allows you to control the build process, including the export and compilation of Java files and the creation of Jar files.

Click Next to continue, or click Finish to use the Project properties and default build options (export only changed files, compile all changed files, and create a Jar).

Project Window

Project Tree

- System P
- + Custo
- + com
- + java
- + javax
- + sun
- + syspe
- + testhe
- + syspe

< Back

Next >

Finish

Cancel

Help



Build Wizard - Specify Build Options

Project Window

Project Tree

- System P
- + Custo
- + com
- + java
- + javax
- + sun
- + syspe
- + testhe
- + syspa



Select the options you want to use for this build.

Export

- Export Java files only if changed
- Export Java files even if unchanged

Compile

- Compile only changed Java source files
- Clean output directories and compile all Java Source files

Compiler options:

Jar

- Create Jar file(s)

< Back Next > Finish Cancel Help



Build Wizard - Confirm Input



Here are the options you selected. Click Back to return to previous pages to make changes.

Build the following:
Project sysparmv1Project
Export Java files only if changed
Compile only changed Java source files
Compiler options: <none>
Create Jar files:
sysparmv1.jar

< Back Next > **Finish** Cancel Help

Project Window

Project Tree

- System P
- + Custo
- + com
- + java
- + javax
- + sun
- + syspe
- + testhe
- + syspe



Build Wizard - Status: In Progress

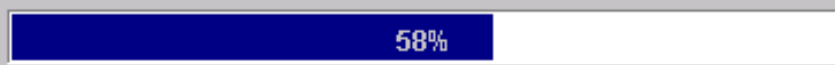
Project Window

Project Tree

- System P
- + Custom
- + com
- + java
- + javax
- + sun
- + syspe
- + testh
- + syspa



```
Exporting non-Java file c:\cool\output\sysparmv1Project\sysparmv1_jar\testharness\Pictu
Exporting non-Java file c:\cool\output\sysparmv1Project\sysparmv1_jar\testharness\SysP
Exporting non-Java file c:\cool\output\sysparmv1Project\sysparmv1_jar\testharness\SysP
Exporting non-Java file c:\cool\output\sysparmv1Project\sysparmv1_jar\testharness\SysP
Exporting non-Java file c:\cool\output\sysparmv1Project\sysparmv1_jar\testharness\back:
Exporting non-Java file c:\cool\output\sysparmv1Project\sysparmv1_jar\testharness\coolj
Exporting non-Java file c:\cool\output\sysparmv1Project\sysparmv1_jar\testharness\subr
Exporting non-Java file c:\cool\output\sysparmv1Project\sysparmv1_jar\testharness\reset
Exporting non-Java file c:\cool\output\sysparmv1Project\sysparmv1_jar\testharness\back:
Compiling c:\cool\source\sysparmv1Project\sysparmv1_jar\sysparmv1\SysParams.java
Compiling c:\cool\source\sysparmv1Project\sysparmv1_jar\testharness\ConvertType.java
Note: c:\cool\source\sysparmv1Project\sysparmv1_jar\testharness\ConvertType.java uses
1 warning
Compiling c:\cool\source\sysparmv1Project\sysparmv1_jar\testharness\ArrayWrapper.jav
Compiling c:\cool\source\sysparmv1Project\sysparmv1_jar\testharness\VectorWrapper.jav
Compiling c:\cool\source\sysparmv1Project\sysparmv1_jar\testharness\EnumerationWrap
```



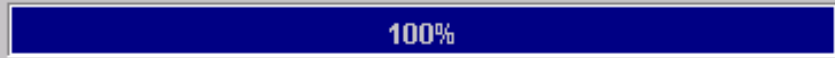
- Stop
- Close
- Save As...
- Page Setup...
- Print
- Help



Build Wizard - Status: Finished



```
Exporting non-Java file c:\cool\output\sysparmv1Project\sysparmv1_jar\testharness\back
Exporting non-Java file c:\cool\output\sysparmv1Project\sysparmv1_jar\testharness\coolj
Exporting non-Java file c:\cool\output\sysparmv1Project\sysparmv1_jar\testharness\subr
Exporting non-Java file c:\cool\output\sysparmv1Project\sysparmv1_jar\testharness\reset
Exporting non-Java file c:\cool\output\sysparmv1Project\sysparmv1_jar\testharness\back
Compiling c:\cool\source\sysparmv1Project\sysparmv1_jar\sysparmv1\SysParams.java
Compiling c:\cool\source\sysparmv1Project\sysparmv1_jar\testharness\ConvertType.java
Note: c:\cool\source\sysparmv1Project\sysparmv1_jar\testharness\ConvertType.java uses
1 warning
Compiling c:\cool\source\sysparmv1Project\sysparmv1_jar\testharness\ArrayWrapper.jav
Compiling c:\cool\source\sysparmv1Project\sysparmv1_jar\testharness\VectorWrapper.jar
Compiling c:\cool\source\sysparmv1Project\sysparmv1_jar\testharness\EnumerationWrap
Compiling c:\cool\source\sysparmv1Project\sysparmv1_jar\testharness\Wrapper.java
Compiling c:\cool\source\sysparmv1Project\sysparmv1_jar\testharness\CheckForData.jav
Creating Jar File c:\cool\output\sysparmv1Project\sysparmv1.jar
Build Completed
```

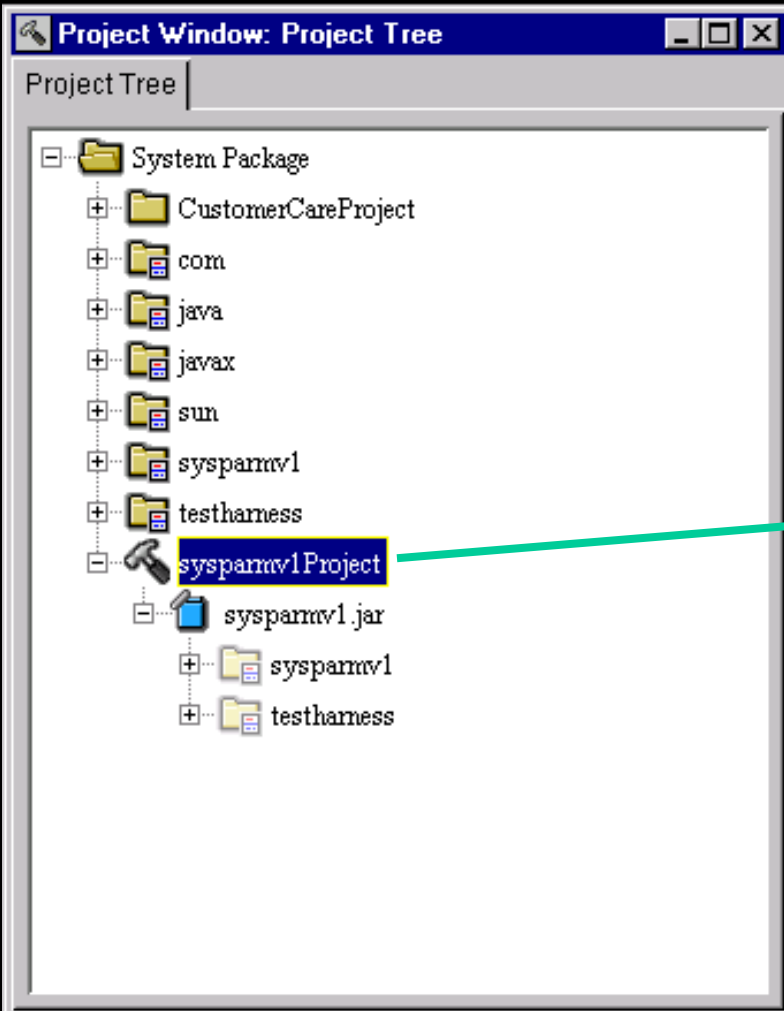
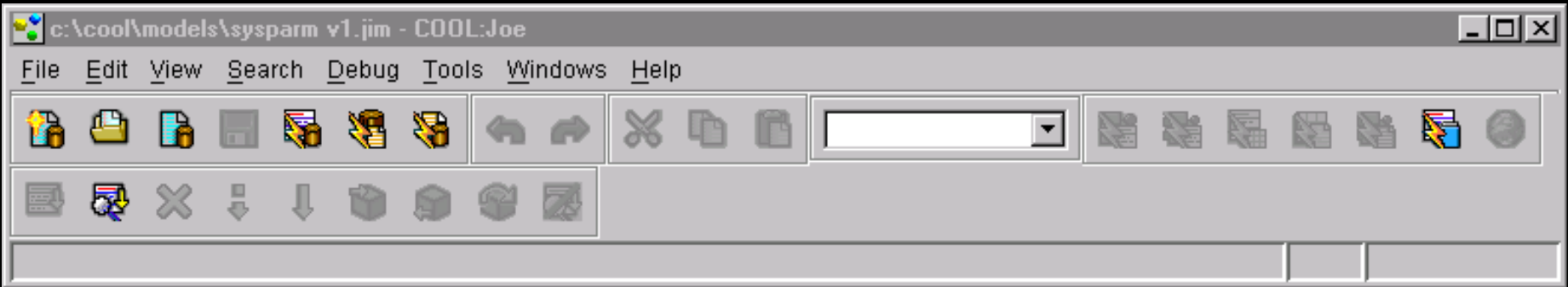


Stop Close Save As... Page Setup... Print Help

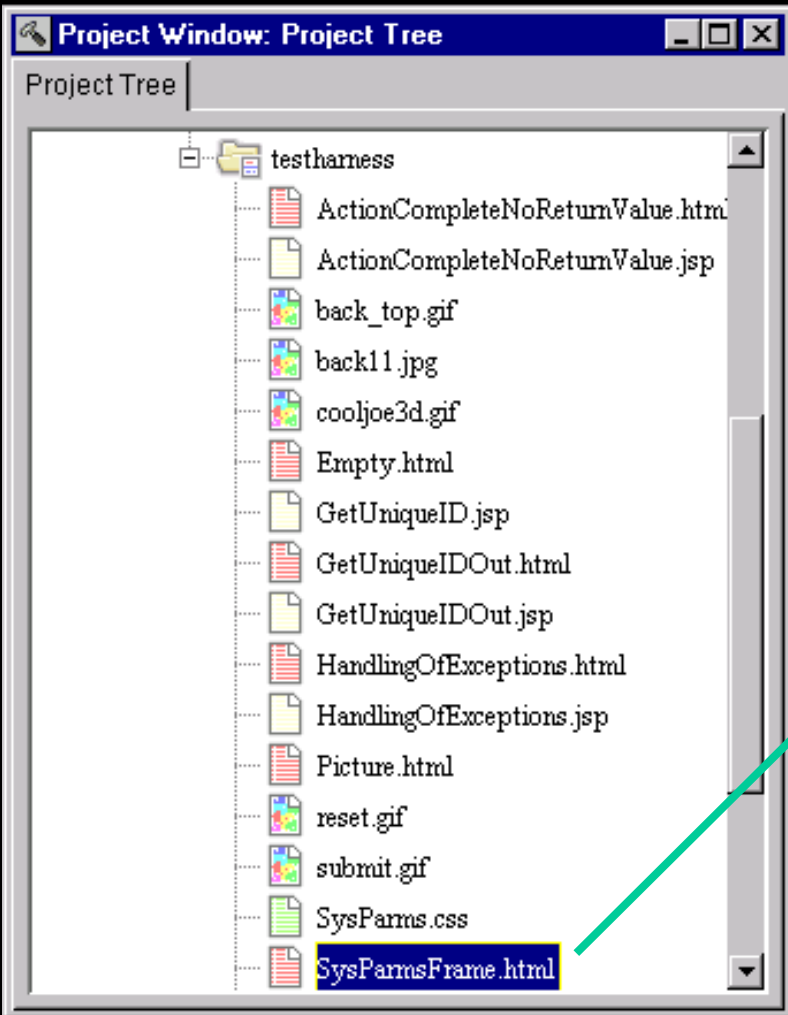
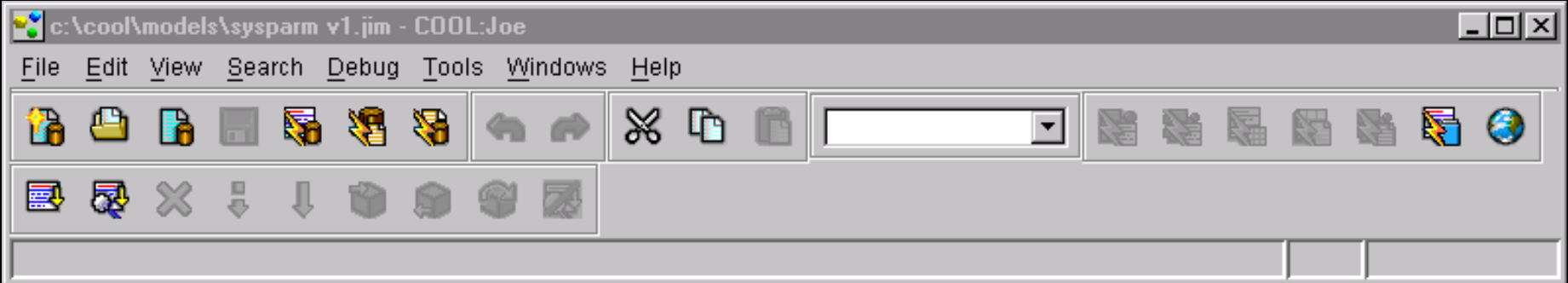
Project Window

Project Tree

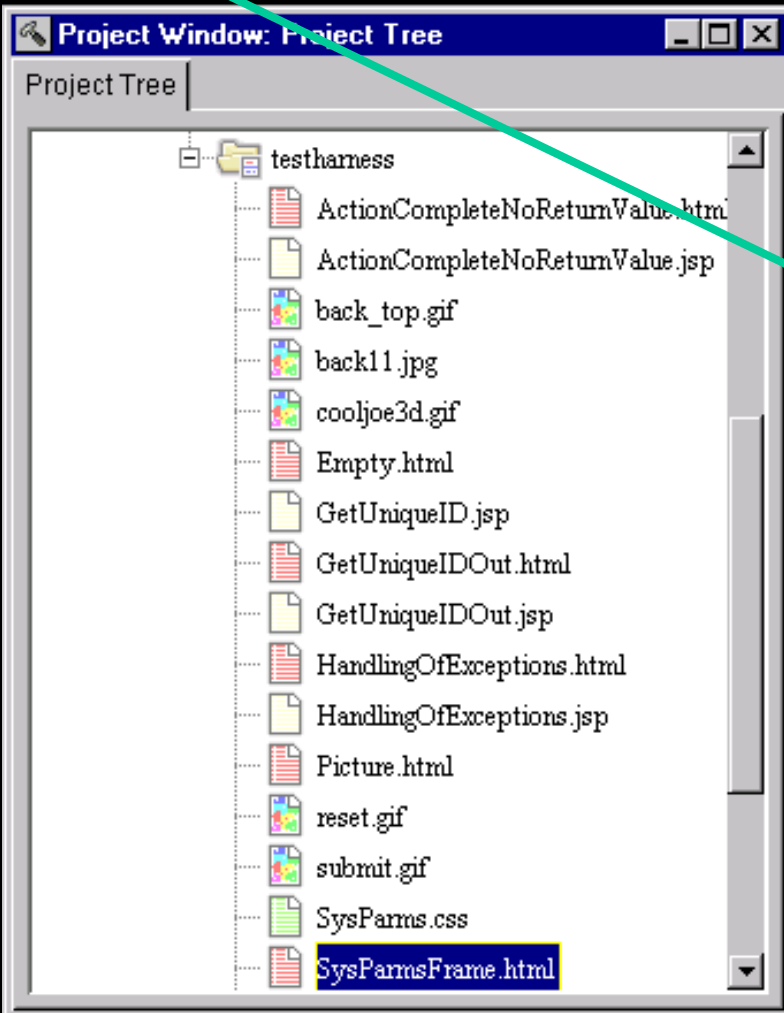
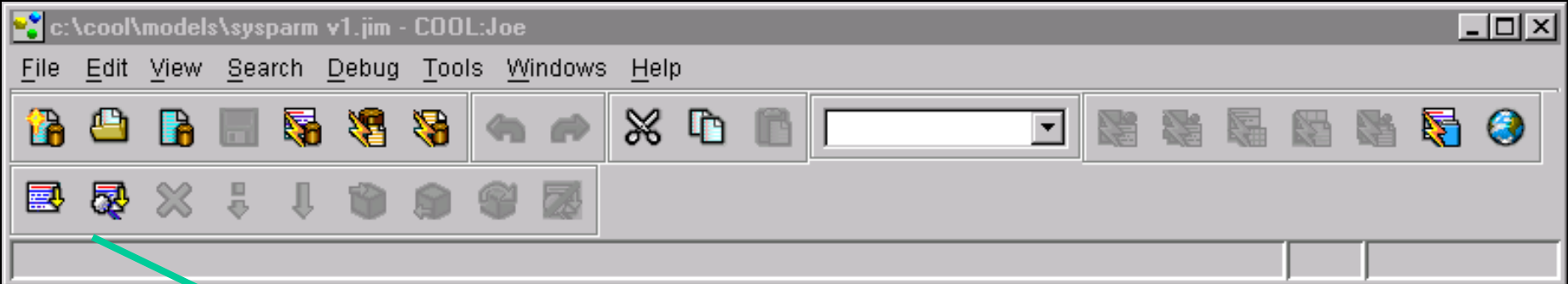
- System P
- + Custom
- + com
- + java
- + javax
- + sun
- + syspe
- + testh
- + syspa



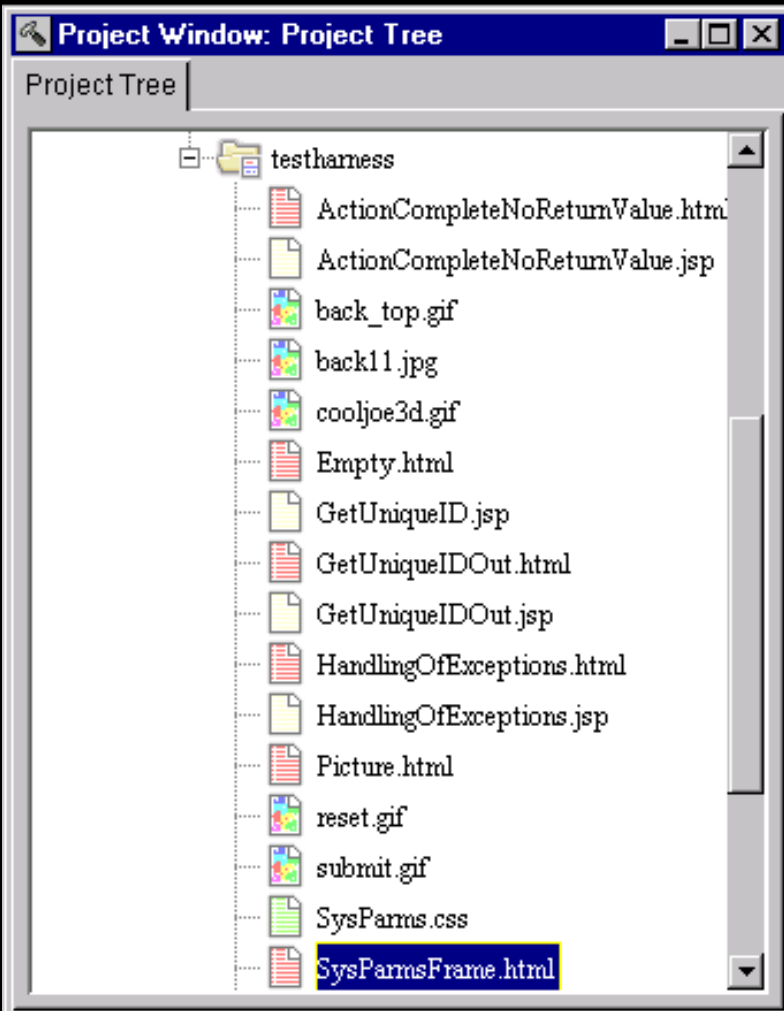
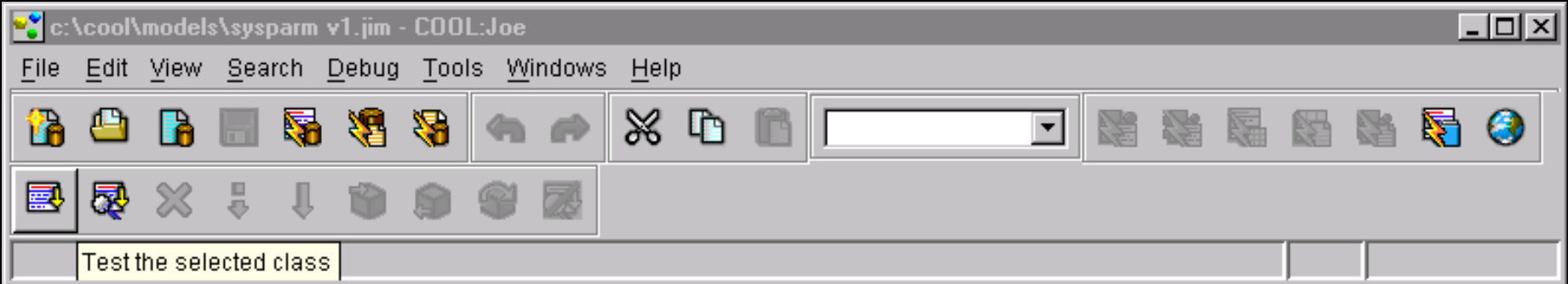
Almost done! Lets test our logic and see if all is good.



Since we generated our test harness as HTML and JSP we will pick the HTML class to launch our test.



Notice the Test and Debug buttons are now active. COOL:Joe supports full debugging of your code. Either locally on your desktop or remotely. Your choice.



c:\cool\models\sysparm v1.jim - COOL:Joe

File Edit View Search Debug Tools Windows Help



http://localhost:8080/testharness/SysParamsFrame.html - Microsoft Internet Explorer

File Edit View Favorites Tools Help



Address <http://localhost:8080/testharness/SysParamsFrame.html> Go Links

Project Window: Pr

Project Tree



GetUniqueID

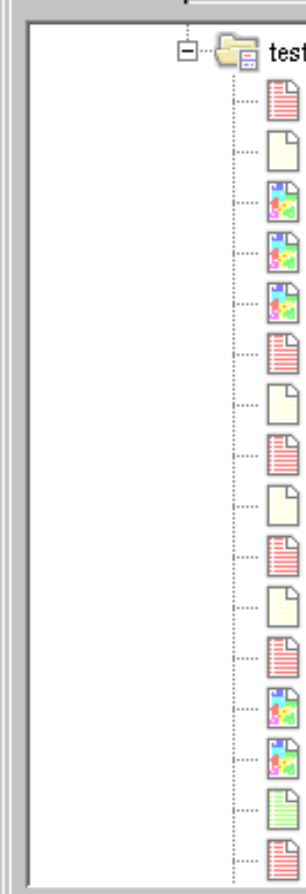


Done Local intranet

SysParamsFrame.html

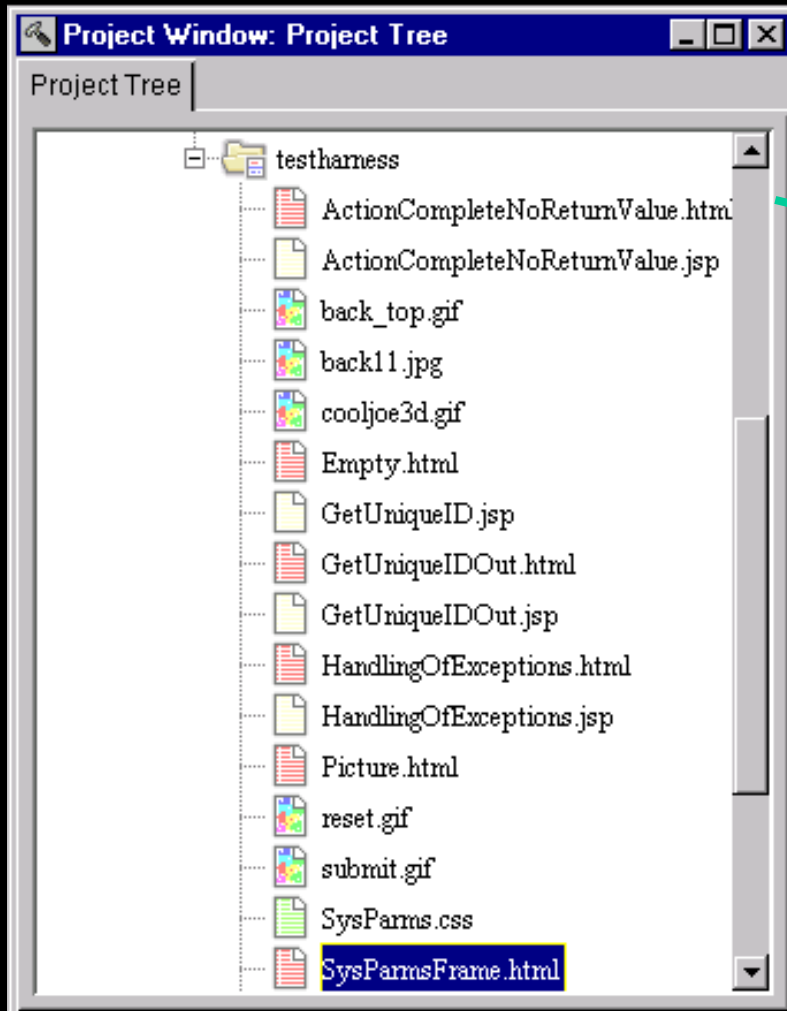
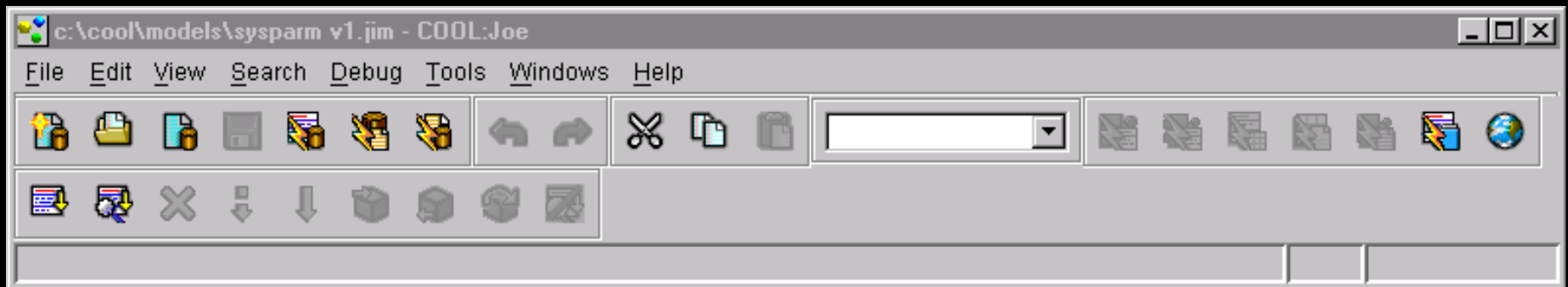


Project Window: Pr
Project Tree



GetUniqueID

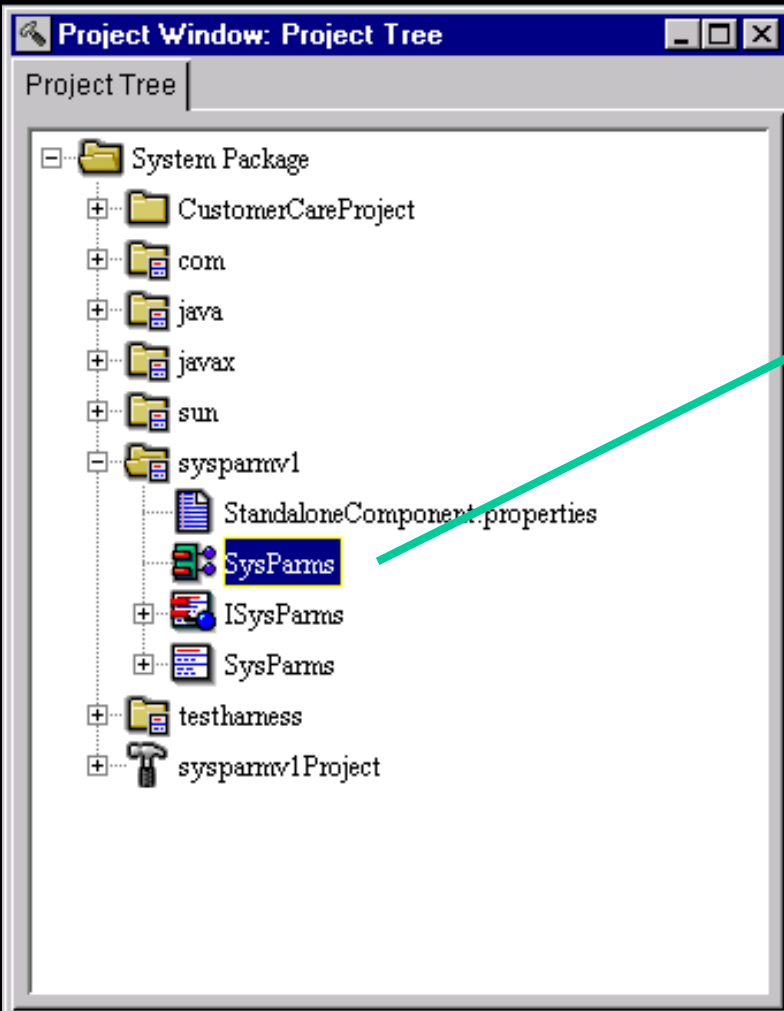
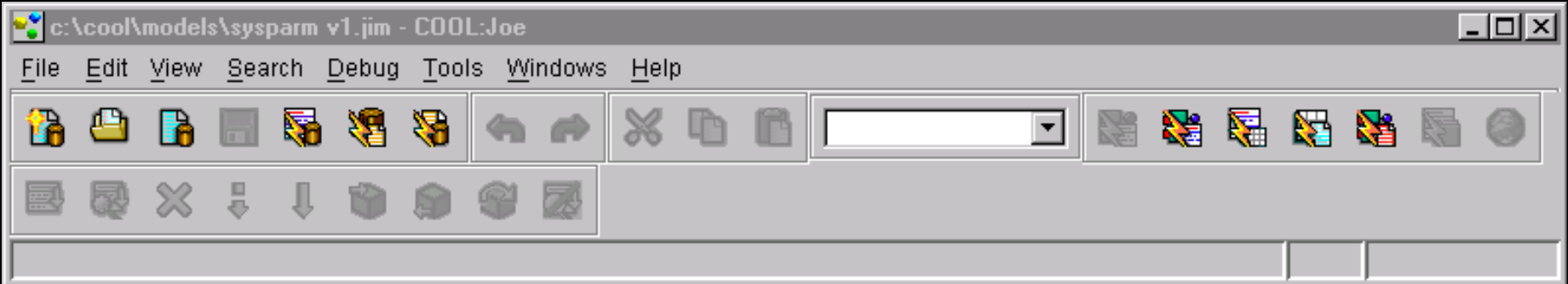
Long: 9,999,999



OK, Our code is working. Now we need to worry about delivering an EJB.

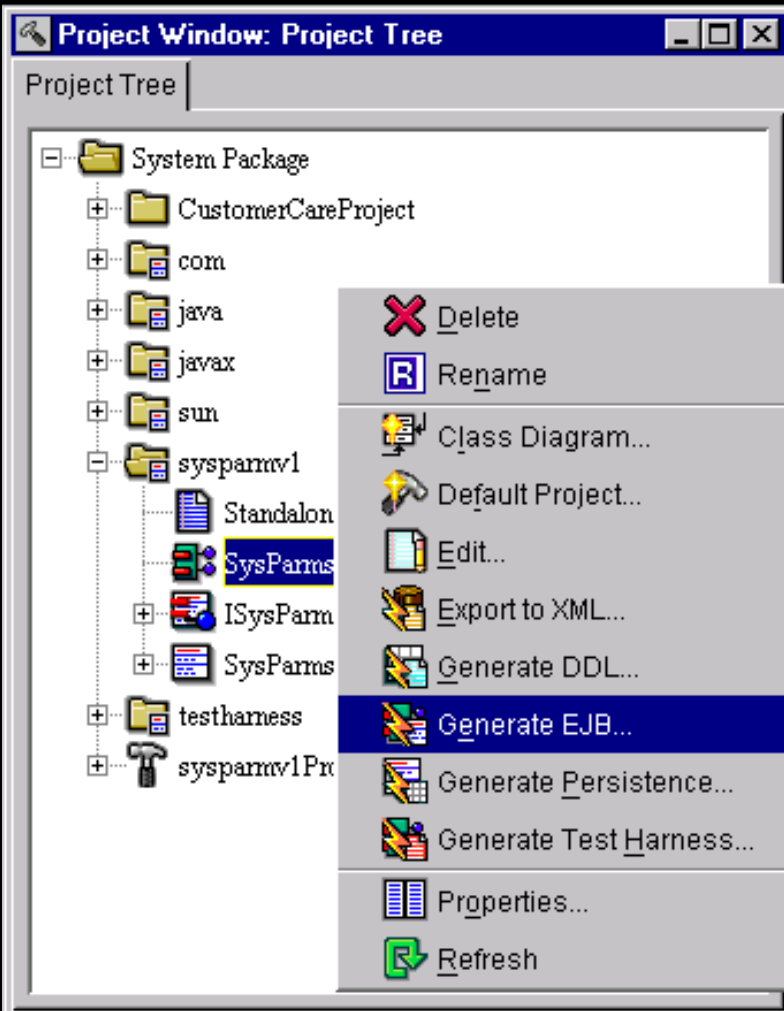
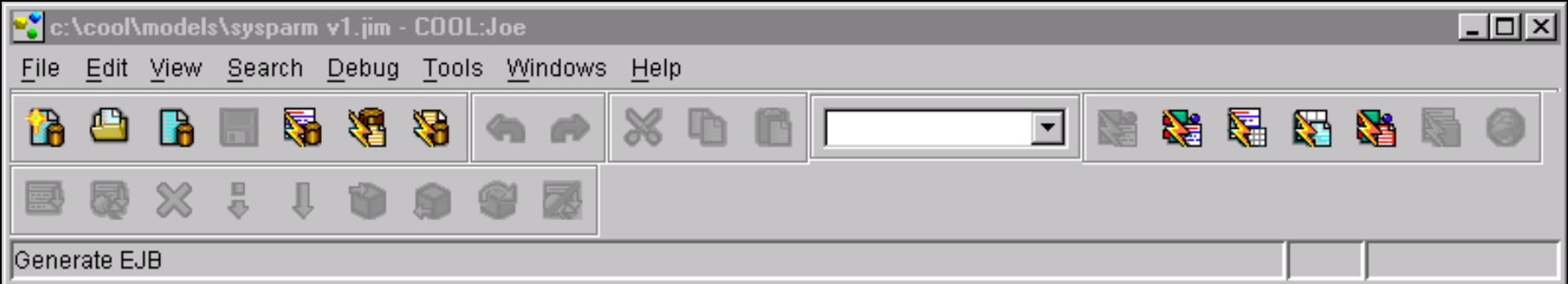
Once again, COOL:Joe makes this easy.

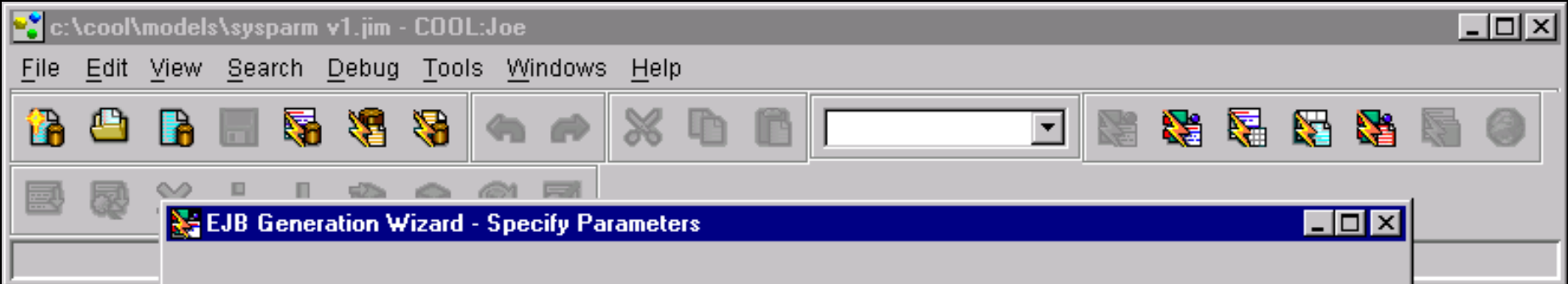
We will return to our sysParm business component to create the necessary EJB classes.



From here we will run the Generate EJB wizard.

Lets see if COOL:Joe can make this easy?





EJB Generation Wizard - Specify Parameters

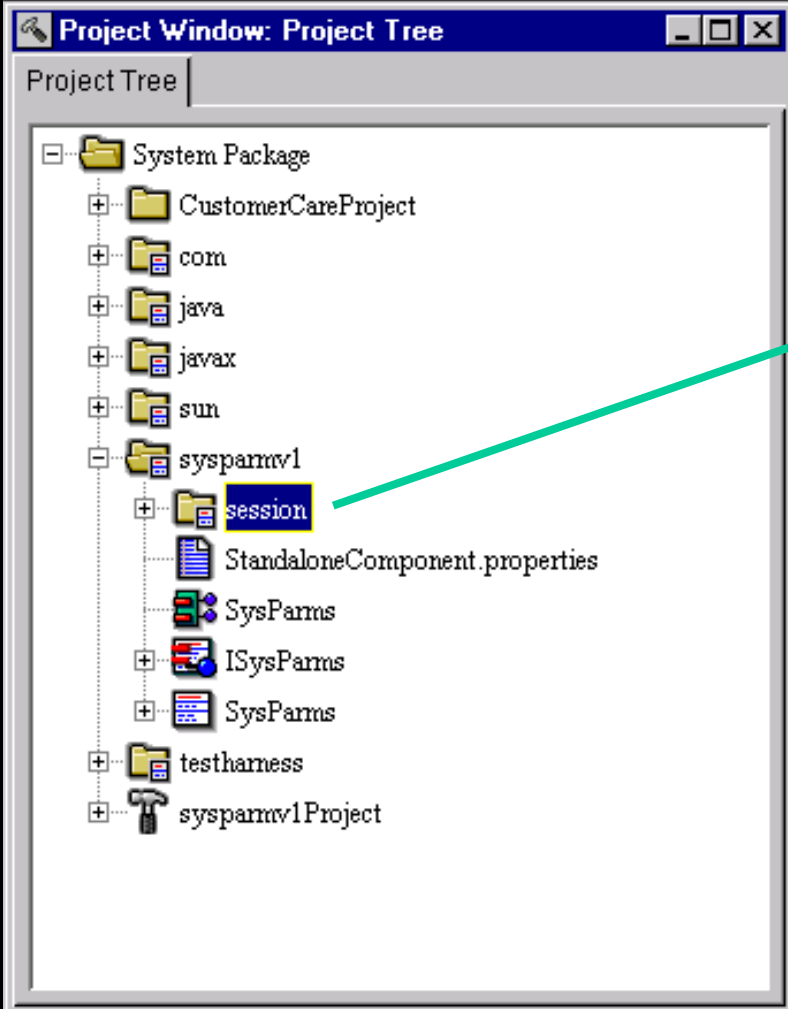
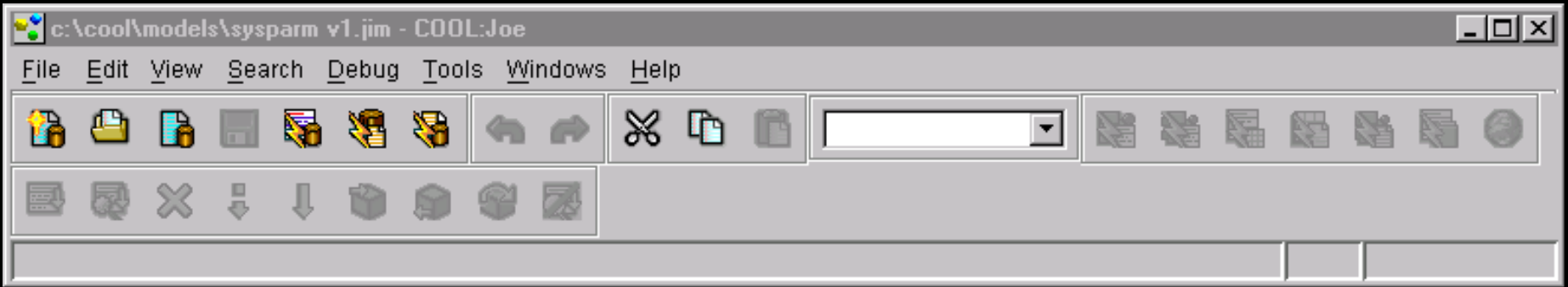


The fields below display the default EJB generation parameters. You can edit each field if you choose to change the default settings.

Session Bean Name:	<input type="text" value="SysParmsSessionBean"/>
Home Interface Name:	<input type="text" value="ISysParmsEJBHome"/>
Remote Interface Name:	<input type="text" value="ISysParmsEJBObject"/>
Home Interface JNDI Name:	<input type="text" value="SysParms"/>

Click Finish to generate EJB.

<input type="button" value=" < Back"/>	<input type="button" value=" Next >"/>	<input type="button" value=" Finish"/>	<input type="button" value=" Cancel"/>	<input type="button" value=" Help"/>
---	---	--	--	--------------------------------------



Lets see if COOL:Joe can make this easy?

You bet! You now have a session package and all of the EJB classes you need to deliver your business component as an EJB.

Life is good.

COOL:Joe