SIEMENS

EasyCODE for JAVA

Technical Guideline

JAVA source files can be developed with **EasyCODE** with the ANSC technique.
ANSC technique means high-comfort development of JAVA source code with the help of a graphical editor. ANSCs (Advanced Nassi Shneiderman Charts = structure diagrams) with their three-dimensional structure offer substantial advantages over the one-dimensional representation of source code with conventional text editors. Structural errors are a thing of the past, the code is far easier to read, the structure diagram also fulfills documentation purposes and the source code has uniform format.

```
// This method displays the applet.
// The Graphics class is how you do all drawing in Java.
public void paint(Graphics g)
```

```
{
int i,j,x,y;
int m = 1;
int n = 1;
double k;
Color blue = new Color(0, 0, 255);
Color green = new Color(0, 255, 0);
```

```
Color red = new Color(255, 0, 0);
int BOLD = 1;
int ITALIC = 2;
Font ArialBold = new Font("Arial",BOLD,20);
Font ArialItalic= new Font ("ARIAL",ITALIC,20);
```

variables declaration

```
int i,j,x,y;
int m = 1;
int n = 1;
double k;
Color blue = new Color(0, 0, 255);
Color green = new Color(0, 255, 0);
Color red = new Color(255, 0, 0);
int BOLD = 1;
int ITALIC = 2;
Font ArialBold = new Font("Arial",BOLD,20);
Font ArialItalic= new Font ("ARIAL",ITALIC,20);
```

```
g.setFont(ArialBold);
do
  {
  j = 0;
  for (i = 1;  i < 4; i++)
    {
    x = (int)(Math.random()*300);
    y = (int)(Math.random()*200);
    switch (i)
      {
      case 1:
        g.setColor(red);
        break;
      case 2:
        g.setColor(blue);
        break;
      case 3:
        g.setColor(green);
        break;
      }
    if (m == 2)
      {
      g.drawString("EasyCODE for JAVA", x, y);
      m=1;
      }
    else
      {
      g.drawString("EasyCODE from Siemens", x, y);
      if (n == 2)
        {
        g.setFont(ArialBold);
        n=1;
        }
      else
        {
        g.setFont(ArialItalic);
        n=2;
        }
      m=2;
      }
    for (k=1; k<100000; k++)
      {
      //Do Nothing
      }
    if (i==2 || i==3)
      {
      g.clearRect(0, 0, 1000, 1000);
      }
    }
    g.clearRect(0, 0, 1000, 1000);
  }
 while (true);
}
```

But with **EasyCODE** you can not only develop new JAVA programs. You can also represent existing, possibly alien JAVA source files (such as JAVA demo programs from WWW or from other sources) in structure diagram form. The essential parts in an unknown source code can thus be identified more quickly and can more easily be reused in your own code.

EasyCODE for JAVA works directly with the JAVA code and does not administer redundant intermediate file formats. If you want to edit an existing JAVA source code in structure diagram form, the respective JAVA file is opened with **EasyCODE**. The **EasyCODE** JAVA language parser analyzes the source code for its flow structure and **EasyCODE** displays the structure in ANSC technique. No translation into another file format - no import or suchlike mechanism - is necessary to represent JAVA source code in structure diagram form. After editing the structure diagram with the methods of ANSC technique, the source code is saved with **EasyCODE**. And that's all. **EasyCODE** directly creates JAVA code which can be read by the JAVA compiler or other tools from the JAVA JDK (JAVA Development Kit).

EasyCODE with its open program linking architecture makes smooth **integration** of the Sun JDK (Java Development Kit) possible:

To integrate the Sun DK engineering tools into **EasyCODE for JAVA**, the respective entries must be made in the **EasyCODE** program linking (menu item "Options - Program linking"). With this, the JAVA compiler, the JAVA applet viewer and any other tools can be accessed in the **EasyCODE** "Execute" menu:

The ***JAVA Compiler*** can be called directly from the **EasyCODE** Execute menu. The current JAVA source file is then compiled. A possible error protocol of the compiler (STDOUT output by JAVAC.EXE) is then displayed directly in **EasyCODE**. Doubleclicking a line number in the error protocol leads you directly to the respective error in the structure diagram.

With the help of the ***JAVA Applet Viewer*** in the Execute menu, the result of the successful compilation can be displayed without leaving **EasyCODE**. The compiled JAVA applet will run in a separate window.

The entry "*C:\JAVA\BIN\JAVAC.EXE %DRIVE% %DIR%%PROMPT:Sourcefile:%.java*" in the "command line" entry contains the call of the JAVAC.EXE compiler from the respective JDK directory. The parameter transferred to the compiler must be the file name of the currently edited JAVA structure diagram in 32 bit notation. In this string it is composed from the symbolic **EasyCODE** names for the current directory and drive (%DIR% and %DRIVE%) and a subsequent interactive file name inquiry before the start of the compiler run (%PROMPT:Sourcefile:%.java).

This inquiry is necessary to pass the compiler the complete file name, which is administered by 32 bit Windows and absolutely required by the compiler, because the complete file name is not taken into account by **EasyCODE** in the current version. For this, the file name must be entered without the extension "java". The extension is automatically added to the prompted file name.

On the other hand this inquiry has the advantage that any file in the current directory of a JAVA project can be compiled from **EasyCODE** without opening the file.

To integrate the ***JAVA Applet Viewer*** in **EasyCODE**, the call of a currently generated applet must be entered in an HTML file (e.g. INDEX.HTM - see below) and this file must be passed to the applet viewer in JDK as parameter with the help of a short batch file (e.g. APPLETV.BAT - see below).

INDEX.HTM:

```
<html>
<applet code="FirstApplet.class" width=600
height=200>
</applet>
</html>
```

APPLETV.BAT:

```
cd c:\java\easycode
c:\java\java\bin\Appletviewer
index.htm
```

The batch file is inserted in the "Execute" menu with the **EasyCODE** program linking, and is addressed there, for example under the "Applet viewer" menu point. The batch file is necessary to make the change to the required directory before calling the applet viewer.

If you want to create several applets at a time, and you want to view all applets with the applet viewer, you can

1. either address all applets in the same HTML file,
2. or create several batch files which themselves access different applet-calling HTML files,
3. or parameterize the batch file with respect to the HTML file. By prompting this parameter in the EasyCODE program linking, the applet viewer can be passed the respective required HTML file and view the applet integrated there.

The symbolic names used in the previous examples for the **EasyCODE** program linking make numerous other applications possible. The **EasyCODE** program linking is described in detail in the **EasyCODE** online help.

Other possible entries in the EasyCODE for JAVA "Execute" menu:

- editors for quick and easy editing of HTML files containing call entries for the JAVA applets created with **EasyCODE**.
- internet browsers (e.g.: Netscape).
- other JDK tools such as JAVADOC.EXE for automatic generation of an engineering documentation for the current API in HTML format.
- links to particularly important URLs for the JAVA engineering, such as the Sun homepage (call Netscape with the URL as parameter: e.g.: http://java.sun.com).

From the Sun homepage you can download the JAVA Development Kit (JDK). In addition to this, the whole JAVA language syntax and JAVA API documentation is available from this Web page. The URLs for these JAVA references can also be entered as link in the "Execute" menu. **EasyCODE** offers a high-comfort, context-sensitive JAVA language syntax support.
The help chapter corresponding to a selected term in the **EasyCODE** structure diagram in one of the JAVA language references will be your permanent advisor.

This support is possible *firstly* by an **EasyCODE** mechanism allowing to combine any Windows HelpFiles with **EasyCODE** and to offer context-sensitive help for a selected word, in case the help file called contains such a keyword.

Secondly this support requires the existence of the JAVA reference in the Windows HelpFormat. These Windows outputs of the JAVA reference (otherwise only existing in HTML format) can be obtained via the WWW from a 3rd party producer (Bill Bercik (bill@dippybird.com)). You can access this Web page is possible from the Sun homepage.

Sun Homepage:

http://java.sun.com/java.sun.com/products/JDK/1.0.2/index.html

The integration of Windows HelpFiles in **EasyCODE** is possible with the following example entries in the [HelpFileList] section of the INI file of **EasyCODE for Java** (Default: EASY-JAV.INI in the Windows directory).

EASY-JAV.INI:

```
[HelpFileList]
HelpFile1=JAVA API Language Reference,c:\java\easycode\
api.hlp
HelpFile2=JAVA Language Reference,c:\java\easycode\
language.hlp
HelpFile3=JAVA Resources,c:\java\easycode\resource.hlp
HelpFile4=Microsoft C/C++ Language,c:\msvc\help\
mscxx.hlp
```

The exact syntax of the HelpFile entry can be seen from the file CUSTOM.DOC supplied in the **EasyCODE** package as a WinWord document. Because of the similarity between the JAVA and C/C++ languages it may be useful to integrate the Microsoft C/C++ Language Reference in **EasyCODE**.

Published by: Siemens AG Austria
Program and System Engineering PSE
Gudrunstraße 11, A-1100 Vienna.

Tel.: +43 1 1707/46522
Fax: +43 1 1707/57072
Mailbox: +43 1 1707/46496
CompuServe: 100141,2120
Internet: EasyCODE@siemens.at
WWW: