Coding Java Under OS/2

BY MICHAEL NORTON

I doubt that many of you believe that IBM is going to lose the enterprise market; after all, the "B" in IBM stands for Business, making any question concerning the future of enterprise computing like asking what color a Smurf turns when it holds its breath. Nevertheless, IBM must respond to the competition; they must change. Increasingly, it seems IBM is adopting the tactics of rival Microsoft and absorbing rather than developing new technologies. Sometimes this involves acquiring the development company outright, as IBM did with Lotus. More often, however, it is a matter of IBM embracing outside technologies and merging them with their own. Such is the case with Java. Although Sun Microsystems developed Java, it has become central to IBM's larger strategy to maintain its position in the enterprise market. It, therefore, behooves all of us Smurfs to acquire at least a rudimentary knowledge of Java.

If you are really, really Blue, you run OS/2 and you have at your fingertips an offering from IBM to encourage you in this pursuit: the Java for OS/2 development package that ships with every copy of the operating system. Last month's column showed you how to install and experiment with the included sample applets. This month, we'll have even more fun and actually do some coding, although the intent is not to provide instructions on the Java language but rather to demonstrate how to use the tools provided with the OS/2 Java Development package.

OS/2: AN UNSURPASSED JAVA DEVELOPMENT PLATFORM

OS/2 is an excellent development platform for Java for many reasons; in my humble opinion, it has advantages that make it clearly superior to other platforms. One reason is the long file name support. A couple of columns ago, I noted how Windows 95 had butchered my filenames. The Java utilities (on all platforms) are rather insistent on "*.java" and "*.class" filename extensions and Java filenames themselves, since they are inextricably linked to the classes they contain and tend to use longer names. Of course, both NT and UNIX support long filenames but neither have the drag-and-drop capabilities of OS/2. Although I'm a self-confessed command line commando, it is exceptionally convenient to drag a source file object and drop it on the compiler object to perform compilations.

Since this is an OS/2 column, we'll use that particular methodology rather than the command line for our purposes.

1. To start, create a new file by clicking your right mouse button on the Data file template in the Templates folder and dragging the new object to your Desktop.

2. Rename the object "HelloWorld.java" by holding down the Alt key while clicking on the object label with the left mouse button, then overtyping the name. Note that renaming the object will associate it with the "Java for OS/2 Editor," which is really just EPM, and the icon will change accordingly.

3. Double click on the object, then type the code listed in Figure 1.

4. When you are finished, open the "Java for OS/2" folder in the "Programs" folder on your Desktop. Leave it open (you'll need it in a moment) and open the "Toolkit for Sun's Java Programming Environment" folder.

5. Drag the HelloWorld.java object from your Desktop to the "Compile Java Code" object. An OS/2 window will open and close, and no messages will be displayed if you have typed the code in correctly. If there are errors, the OS/2 window will remain open and the errors will be displayed. Correct the error(s) and try again.

6. When the code compiles successfully, a new object, HelloWorld.class, will be created on your Desktop. Double clicking on this icon will not yield the results you'd expect, i.e., executing the HelloWorld program; for some reason the default association is with the method's dump utility. Instead, drag the HelloWorld.class object to the "Run Java program" object in the "Java for OS/2" folder. Note that there are two "Run Java..." objects, including one for PM programs that must be used if any references are made to the Java AWT (i.e., if it is a GUI Java application). Since our Hello World program is a STDIO type of program, we use the other, "command prompt" version.

That's it. You are now a full-fledged Java developer. Well, almost. Besides the trivial matter of the language, there are a few tools you need to know about, all of which use the same drag-and-drop methodology you used to compile and run the Hello World sample. Java has an auto-documentation feature that allows you to document while you are coding through the specialized use of comment symbols (/** and */). To extract this documentation, drag and drop the Hello-

Figure 1: HelloWorld

```
/**************************************************************************
* HelloWorld.java
**************************************************************************/
public class HelloWorld {

    public static void main(String args[]){
        System.out.println("Hello World!");
        } // main method

    } // class
```

World.java file on to the "Generate documentation from Java code" icon in the "Toolkit for Sun's Java Programming Environment" folder. Three HTML files will be generated documenting packages used (packages.html), the class hierarchy (tree.html), and an index of all fields and methods (AllNames.html). You can also disassemble a class file or display all methods in a class by dragging and dropping the class file onto the appropriate icon in the "Toolkit for Sun's Java Programming Environment" folder.

MADE POSSIBLE BY THE WORKPLACE SHELL

The marvelous, and somewhat sad, aspect of the entire process I just described is that it is made possible by the Workplace Shell; indeed, this little example shows that with very little effort, the Workplace Shell can be molded into a development environment. That was the intent, of course, which is why OS/2 will continue to excel as a development platform. Not because of the Workplace Shell — as I've already noted; the new browser interfaces will more or less force the WPS into obsolescence. However, IBM is demonstrably committed to the object-oriented technologies including Java that will allow development to step forward into the next millennium. I'm frequently asked whether I think OS/2 will survive; I'm holding my breath.

GOODBYE, AND HELLO

This will be my last OS/2 column. Don't worry — I'm not going far; you'll find me in the new "Evolutions" column in ***Technical Support*** magazine.

Replacing me here will be Rick Byrley of SofTouch Systems. For several years, Rick has been admirably performing the same trench-level technical support work that taught me the OS/2 operating system. Rest assured, you'll still be receiving the most accurate technical information available. I think you'll enjoy his contributions in the upcoming issues. Just don't forget about me over in the "Evolutions" column! *ts*

Michael Norton is the network administrator at SofTouch Systems, Oklahoma City, Okla., which provides both mainframe and PC software solutions. He has written mainframe manuals in addition to articles for a number of publications. Michael can be contacted at norton@softouch.com.