Study Unit 2: Java Programming basics

Introduction

This study unit you'll learn all about the Java language and how to use it to create applets, as well as how to create stand-alone Java applications that you can use for just about anything.

An applet is a dynamic and interactive program that can run inside a Web page displayed by a Java-capable browser such as HotJava or Netscape 2.0. The HotJava browser is a World Wide Web browser used to view Web pages, follow links, and submit forms. It can also download and play applets on the reader's system.

Learning Outcomes of Study Unit 2

Upon completion of this study unit, you should be able to know

2.1 What exactly Java is, and its current status

2.2 Why you should learn Java-its various features and advantages over other programming languages

2.3 How to get started programming in Java-what you'll need in terms of software and background, as well as some basic terminology

2.4 How to create your first Java programs

2.1 Java Programming

2.1.1 What Is Java?

Based on the enormous amount of press Java is getting and the amount of excitement it has generated, you may get the impression that Java will save the world-or at least solve all the problems of the Internet. Not so. Java's hype has run far ahead of its capabilities, and while Java is indeed interesting, it really is another programming language with which you write programs that run on the Internet. In this respect, Java is closer to popular programming languages such as C, C++, Visual Basic, or Pascal, than it is to a page description language such as HTML, or a very simple scripting language such as JavaScript or PHP.

More specifically, Java is an object-oriented programming language developed by Sun Microsystems, a company best known for its high-end UNIX workstations. Modeled after C++, the Java language was designed to be small, simple, and portable across platforms and operating systems, both at the source and at the binary level, which means that Java programs (applets and applications) can run on any machine that has the Java virtual machine installed (you'll learn more about this later).

Java is usually mentioned in the context of the World Wide Web, where browsers such as Netscape's Navigator and Microsoft's Internet Explorer claim to be "Java enabled." *Java enabled* means that the browser in question can download and play Java programs, called *applets*, on the reader's system. Applets appear in a Web page much the same way as images do, but unlike images, applets are dynamic and interactive. Applets can be used to create animation, figures, forms that immediately respond to input from the reader, games, or other interactive effects on the same Web pages among the text and graphics.

Box 2.1: Definition of applets

Applets are programs that are downloaded from the World Wide Web by a Web browser and run inside an HTML Web page. You'll need a Java-enabled browser such as Netscape Navigator or Microsoft's Internet Explorer to run applets.

To create an applet, you write it in the Java language, compile it using a Java compiler, and refer to that applet in your HTML Web pages. You put the resulting HTML and Java files on a Web site in the same way that you make ordinary HTML and image files available. Then, when someone using a Java-enabled browser views your page with the embedded applet, that browser downloads the applet to the local system and executes it, allowing your reader to view and interact with your applet in all its glory. (Readers using other browsers that are not Java enabled may see text, a static graphic, or nothing.)

While applets are probably the most popular use of Java, the important thing to understand about Java is that you can do so much more with it than create and use applets. Java was written as a full-fledged general-purpose programming language in which you can accomplish the same sorts

of tasks and solve the same sorts of problems that you can in other programming languages, such as C or C++.

Java's Past, Present and Future

The Java language was developed at Sun Microsystems in 1991 as part of a research project to develop software for consumer electronics devices-television sets, VCRs, toasters, and the other sorts of machines you can buy at any department store. Java's goals at that time were to be small, fast, efficient, and easily portable to a wide range of hardware devices. Those same goals made Java an ideal language for distributing executable programs via the World Wide Web and also a general-purpose programming language for developing programs that are easily usable and portable across different platforms.

The Java language was used in several projects within Sun (under the name Oak), but did not get very much commercial attention until it was paired with HotJava. HotJava, an experimental World Wide Web browser, was written in 1994 in a matter of months, both as a vehicle for downloading and running applets and also as an example of the sort of complex application that can be written in Java. Although HotJava got a lot of attention in the Web community, it wasn't until Netscape incorporated HotJava's ability to play applets into its own browser that Java really took off and started to generate the excitement that it has both on and off the World Wide Web. Java has generated so much excitement, in fact, that inside Sun the Java group spun off into its own subsidiary called JavaSoft.

Currently, to program in Java, you'll need a Java development environment of some sort for your platform. Sun's JDK(Java Development Kit) works just fine for this purpose and includes tools for compiling and testing Java applets and applications. In addition, a wide variety of excellent Java development environments have been developed, including Sun's own Java Workshop, Symantec's Café, Microsoft's Visual J++ (which is indeed a Java tool, despite its name), and Natural Intelligence's Roaster, Borland’s JBuilder, JCreator, DrJava, with more development tools appearing all the time.

To run and view Java applets, you'll need a Java-enabled browser or other tool. As mentioned before, recent versions of Netscape Navigator (2.0 and higher) and Internet Explorer (3.0) can both

run Java applets. (Note that for Windows you'll need the 32-bit version of Netscape, and for Macintosh you'll need Netscape 3.0.) You can also use Sun's own HotJava browser to view applets, as long as you have the 1.0 pre-beta version (older versions are not compatible with newer applets, and vice versa). Even if you don't have a Java-enabled browser, many development tools provide simple viewers with which you can run your applets. The JDK comes with one of these; it's called the `appletviewer`.

2.2 Why Learn Java

2.2.1 Why Learn Java?

Java as a programming language has significant advantages over other languages and other environments that make it suitable for just about any programming task.

2.2.2 Java Is Platform Independent

Platform independence-that is, the ability of a program to move easily from one computer system to another-is one of the most significant advantages that Java has over other programming languages, particularly if your software needs to run on many different platforms. If you're writing software for the World Wide Web, being able to run the same program on many different systems is crucial to that program's success. Java is platform independent at both the source and the binary level.

Box 2.2: Definition Platform independence

Platform independence means that a program can run on any computer system. Java programs can run on any system for which a Java virtual machine has been installed.

At the source level, Java's primitive data types have consistent sizes across all development platforms. Java's foundation class libraries make it easy to write code that can be moved from platform to platform without the need to rewrite it to work with that platform. When you write a program in Java, you don't need to rely on features of that particular operating system to

accomplish basic tasks. Platform independence at the source level means that you can move Java source files from system to system and have them compile and run cleanly on any system.

Platform independence in Java doesn't stop at the source level, however. Java compiled binary files are also platform independent and can run on multiple platforms (if they have a Java virtual machine available) without the need to recompile the source.

Normally, when you compile a program written in C or in most other languages, the compiler translates your program into machine code or processor instructions. Those instructions are specific to the processor your computer is running-so, for example, if you compile your code on an Intel-based system, the resulting program will run only on other Intel-based systems. If you want to use the same program on another system, you have to go back to your original source code, get a compiler for that system, and recompile your code so that you have a program specific to that system.

Things are different when you write code in Java. The Java development environment actually has two parts: a Java compiler and a Java interpreter. The Java compiler takes your Java program and, instead of generating machine codes from your source files, it generates bytecodes. Bytecodes are instructions that look a lot like machine code, but are not specific to any one processor.

To execute a Java program, you run a program called a bytecode interpreter, which in turn reads the bytecodes and executes your Java program. The Java bytecode interpreter is often also called the Java virtual machine or the Java runtime.

Box 2.2: Definition Java bytecodes

Java bytecodes are a special set of machine instructions that are not specific to any one processor or computer system. A platform-specific bytecode interpreter executes the Java bytecodes. The bytecode interpreter is also called the Java virtual machine or the Java runtime interpreter.

Where do you get the bytecode interpreter? For applets, the bytecode interpreter is built into every Java-enabled browser, so you don't have to worry about it-Java applets just automatically run. For

more general Java applications, you'll need to have the interpreter installed on your system in order to run that Java program. Right now, you can get the Java interpreter as part of your development environment, or if you buy a Java program, you'll get it with that package. In the future, however, the Java bytecode interpreter will most likely come with every new operating system-buy a Windows machine, and you'll get Java for free.

Why go through all the trouble of adding this extra layer of the bytecode interpreter? Having your Java programs in bytecode form means that instead of being specific to any one system, your programs can be run on any platform and any operating or window system as long as the Java interpreter is available. This capability of a single binary file to be executable across platforms is crucial to what makes applets work because the World Wide Web itself is also platform independent. Just as HTML files can be read on any platform, so can applets be executed on any platform that has a Java-enabled browser.

The disadvantage of using bytecodes is in execution speed. Because system-specific programs run directly on the hardware for which they are compiled, they run significantly faster than Java bytecodes, which must be processed by the interpreter. For many basic Java programs, speed may not be an issue. If you write programs that require more execution speed than the Java interpreter can provide, you have several solutions available to you, including being able to link native code into your Java program or using special tools (called just-in-time compilers) to convert your Java bytecodes into native code and speed up their execution. Note that by using any of these solutions, you lose the portability that Java bytecodes provide.

2.2.3 Java Is Object Oriented

To some, the object-oriented programming (OOP) technique is merely a way of organizing programs, and it can be accomplished using any language. Working with a real object-oriented language and programming environment, however, enables you to take full advantage of object-oriented methodology and its capabilities for creating flexible, modular programs and reusing code.

Many of Java's object-oriented concepts are inherited from C++, the language on which it is based, but it borrows many concepts from other object-oriented languages as well. Like most object-

oriented programming languages, Java includes a set of class libraries that provide basic data types, system input and output capabilities, and other utility functions. These basic libraries are part of the standard Java environment, which also includes simple libraries, form networking, common Internet protocols, and user interface toolkit functions. Because these class libraries are written in Java, they are portable across platforms as all Java applications are.

You'll learn more about object-oriented programming and Java later.

2.2.4 Java Is Easy to Learn

In addition to its portability and object orientation, one of Java's initial design goals was to be small and simple, and therefore easier to write, easier to compile, easier to debug, and, best of all, easy to learn. Keeping the language small also makes it more robust because there are fewer chances for programmers to make mistakes that are difficult to fix. Despite its size and simple design, however, Java still has a great deal of power and flexibility.

Java is modeled after C and C++, and much of the syntax and object-oriented structure is borrowed from the latter. If you are familiar with C++, learning Java will be particularly easy for you because you have most of the foundation already.

Although Java looks similar to C and C++, most of the more complex parts of those languages have been excluded from Java, making the language simpler without sacrificing much of its power. There are no pointers in Java, nor is there pointer arithmetic. Strings and arrays are real objects in Java. Memory management is automatic. To an experienced programmer, these omissions may be difficult to get used to, but to beginners or programmers who have worked in other languages, they make the Java language far easier to learn.

However, while Java's design makes it easier to learn than other programming languages, working with a programming language is still a great deal more complicated than, say, working in HTML. If you have no programming language background at all, you may find Java difficult to understand and to grasp. But don't be discouraged! Learning programming is a valuable skill for the Web and for computers in general, and Java is a terrific language to start out with.

2.3 Getting started with Programming in Java

2.3.1 Getting Started Programming in Java

With the above background, we are now ready to create our first two Java programs: a standalone Java application and an applet that you can view in a Java-enabled browser. Although both these programs are extremely simple, they will give you an idea of what a Java program looks like and how to compile and run it.

2.3.2 Getting a Java Development Environment

In order to write Java programs, you will, of course, need a Java development environment. A number of development environments called Integrated Development Environments (IDEs) are available for you to develop your Java programs. Some of these IDEs are commercial but with trial versions while others are free.

2.3.3 Creating a Java Application

Now let's actually get to work. We'll start by creating a simple Java application: the classic Hello World example that many programming language books use to begin.

Java applications are different from Java applets. Applets, as you have learned, are Java programs that are downloaded over the World Wide Web and executed by a Web browser on the reader's machine. Applets depend on a Java-enabled browser in order to run.

Box 2.1: Definition Java applications

Java applications, however, are more general programs written in the Java language. Java applications don't require a browser to run; in fact, Java can be used to create all the kinds of applications that you would normally use a more conventional programming language to create.

Java applications are standalone Java programs that do not require a Web browser to run. Java applications are more general-purpose programs such as you'd find on any computer.

A single Java program can be an applet or an application, or both, depending on how you write that program and the capabilities that program uses. Throughout the first chapters of the course as you learn the Java language, you'll be writing mostly applications; then you'll apply what you've learned to write applets later. If you're eager to get started with applets, be patient. Everything that you learn while you're creating simple Java applications will apply to creating applets, and it's easier to start with the basics before moving onto the hard stuff. You'll be creating plenty of applets later in the course.

2.4 Creating java code

2.4.1 Creating the Source File

As with all programming languages, your Java source files are created in a plain text editor, or in an editor that can save files in plain ASCII without any formatting characters. On UNIX, `emacs`, `pico`, kate and `vi` will work; on Windows, Notepad or DOS Edit are both text editors that will work. If you're using a development environment like Café or Roaster, it'll have its own built-in text editor you can use.

Note

If you're using Windows to do your Java development, you may have to make sure Windows understands the `.java` file extension before you start; otherwise, your text editor may insist on giving all your files a `.txt` extension.

Fire up your editor of choice and enter the Java program shown in Listing 1.1. Type this program, as shown, in your text editor. Be careful that all the parentheses, braces, and quotes are there, and that you've used all the correct upper- and lowercase letters.

Listing 2.1. Your first Java application.

```
1: class HelloWorld {
2:     public static void main (String args[]) {
3:         System.out.println("Hello World!");
4:     }
5: }
```

Warning

The number before each line is part of the listing and not part of the program; the numbers are there so I can refer to specific line numbers when I explain what's going on in the program. Do not include them in your own file.

After you've finished typing in the program, save the file somewhere on your disk with the name `HelloWorld.java`. This is very important. Java source files must have the same name as the class they define (including the same upper- and lowercase letters), and they must have the extension `.java`. Here, the class definition has the name `HelloWorld`, so the filename must be `HelloWorld.java`. If you name your file something else (even something like `helloworld.java` or `Helloworld.java`), you won't be able to compile it. Make absolutely certain the name is `HelloWorld.java`.

You can save your Java files anywhere you like on your disk, but I like to have a central directory or folder to keep them all in.

2.4.2 Compiling and Running the Source File

Now it's time to compile the file. If your JDK is well configured, you can compile your Java program at the command prompt by typing the command *javac* followed by your Java file. For example, suppose your Java file called MyFile.java is located on your C drive, and you want to compile it at the command prompt, you would type and execute the following command.

C:\> javac MyFile.java

If your program has no errors, then you will automatically see a file called MyFile.class where you had stored your java file MyFile.java. The presence of the class file indicates that you have the Java bytecode to give to the Java interpreter. Remember the Java interpreter is used to execute our Java compiled programs (i.e. the .class files). Therefore, to execute our bytecode in MyFile.class, we type and execute the following command.

C:\>java MyFile

Note that this time we are calling the interpreter called java and we are not putting the file extension (.class) on our command.

If you're using a graphical development environment like JCreator or DrJava, there will most likely be a button or option to compile the file (check with the documentation that comes with such a program). What is import here is to ensure that your development environment points properly to the JDK. You should follow the instructions in the documentation of the environment you are using on how to configure it to point to the JDK.

Now for the case of our Java example HelloWorld.java above, we will type

C:\> javac HelloWorld.java

and then we type

C:\>java HelloWorld

This will automatically print the word “Hello World!” on the screen. Note that when you have finished compiling, you can run the program any number of times you want without recompiling. However, if you make any changes in your Java source file, you must compile again and run to see the changes.

Note

Remember, the Java compiler and the Java interpreter are different things. You use the Java compiler (`javac`) for your Java source files to create `.class` files, and you use the Java interpreter (`java`) to actually run your class files.

Tip

Putting an alias for Java Compiler on the desktop makes it easy to drag and drop Java source files.

If you get any errors, go back to your original source file and make sure you typed it exactly as it appears in Listing 1.1, with the same upper- and lowercase. Also make sure the filename has exactly the same upper- and lowercase as the name of the class (that is, both should be `HelloWorld`).

2.4.3 Creating a Java Applet

Creating applets is different from creating a simple application. Java applets run and are displayed inside a Web page with other page elements, and therefore have special rules for how they behave. Because of these special rules for applets, creating an applet may in many cases be more complex than creating an application.

For example, to create a simple Hello World applet, instead of merely being able to print a message as a set of characters, you have to make space for your message on the Web pages and then use special font and graphics operations to paint the message to the screen.

Note
Actually, you can run a plain Java application as an applet, but the `Hello World` message will print to a special window or to a log file, depending on how the browser has its output set up. You'll learn more about this later.

Creating the Source File for the Applet

In this example, you'll create a simple Hello World applet, place it inside a Web page, and view the result. As with the Hello World application, you'll first create the source file in a plain text editor. Listing 1.2 shows the code for the example.

Listing 2.2. The Hello World applet.

```
1: import java.awt.Graphics;
2:
3: public class HelloWorldApplet extends java.applet.Applet {
4:
```

```
5:     public void paint(Graphics g) {
6:        g.drawString("Hello world!", 5, 25);
7:    }
8:}
```

Save that file just as you did the Hello World application, with the filename exactly the same as the name of the class. In this case the class name is `HelloWorldApplet`, so the filename you save it to would be `HelloWorldApplet.java`.

2.4.4 Compiling the Source File

The next step is to compile the Java applet file. Despite the fact that this is an applet, you compile the file exactly the same way you did the Java application, using the following command.

```
C:\> javac HelloWorldApplet.java
```

Remember this is an applet; therefore you don't call the Java interpreter. Instead, the bytecode (the .class file) is the one to include in the Web page where your applet should run from.

2.5 Troubleshooting a java code

2.5.1 Troubleshooting

If you've run into any problems with the previous examples, this section can help. Here are some of the most common problems and how to fix them:

a. **Bad command or filename or Command not found**

   These errors result when you do not have the JDK's `bin` directory in your execution path, or the path to that directory is wrong. On Windows, double-check your `autoexec.bat` file; on UNIX, check the system file with your path commands in it (`.cshrc`, `.login`, `.profile`, or some similar file).

b. **javac: invalid argument**

   Make sure the name of the file you're giving to the `javac` command is exactly the same

name as the file. In particular, in the DOS shell you want to use the Windows filename with a `.java` extension, not the DOS equivalent (`HELLOW~1.jav`, for example).

c. **Warning: public class HelloWorldApplet must be defined in a file called HelloWorldApplet.java**

This error most often happens if there is a mismatch between the name of the class as defined in the Java file itself (the name following the word class) and the name of the `java` source file. Both the filenames must match, including upper- and lowercase letters (this particular error implies that the filename had lowercase letters). Rename either the filename or the class name, and this error will go away.

d. **Insufficient-memory errors**

The JDK is not the most efficient user of memory. If you're getting errors about memory, consider closing larger programs before running the Java compiler, turn on virtual memory, or install more RAM.

e. **Other code errors**

If you're unable to compile the Java source files because of other errors I haven't mentioned here, be sure that you've typed them in exactly as they appear, including all upper- and lowercase letters. Java is case sensitive, meaning that upper- and lowercase letters are treated differently, so you will need to make sure that everything is capitalized correctly.

2.5.2 Summary

In this unit, you've gotten a basic introduction to the Java language and its goals and features. Java is a programming language, similar to C or C++, in which you can develop a wide range of programs. The most common use of Java at the moment is in creating applets for HotJava, an advanced World Wide Web browser also written in Java. Applets are Java programs that are downloaded and run as part of a Web page. Applets can create animation, games, interactive programs, and other multimedia effects on Web pages.

Java's strengths lie in its portability-both at the source and at the binary level, in its object-oriented design-and in its simplicity. Each of these features helps make applets possible, but they also make Java an excellent language for writing more general-purpose programs that do not require a Java-enabled browser to run. These general-purpose Java programs are called applications.

To end this study unit, you experimented with an example of an applet and an example of an application, getting a feel for the differences between the two and how to create, compile, and run Java programs-or, in the case of applets, how to include them in Web pages.