Characteristics of Java (Optional)

Y. Daniel Liang
Supplement for Introduction to Java Programming

Java has become enormously popular. Java's rapid rise and wide acceptance can be traced to its design and programming features, particularly its promise that you can write a program once and run it anywhere. As stated in the Java language white paper by Sun, Java is *simple*, *object-oriented*, *distributed*, *interpreted*, *robust*, *secure*, *architecture-neutral*, *portable*, *high-performance*, *multithreaded*, and *dynamic*. Let's analyze these often-used buzzwords.

1 Java Is Simple

No language is simple, but Java is a bit easier than the popular object-oriented programming language C++, which was the dominant software-development language before Java. Java is partially modeled on C++, but greatly simplified and improved. For instance, pointers and multiple inheritance often make programming complicated. Java replaces the multiple inheritance in C++ with a simple language construct called an *interface*, and eliminates pointers.

Java uses automatic memory allocation and garbage collection, whereas C++ requires the programmer to allocate memory and collect garbage. Also, the number of language constructs is small for such a powerful language. The clean syntax makes Java programs easy to write and read. Some people refer to Java as "C++--" because it is like C++ but with more functionality and fewer negative aspects.

2 Java Is Object-Oriented

Java is inherently object-oriented. Although many object-oriented languages began strictly as procedural languages, Java was designed from the start to be object-oriented. Object-oriented programming (OOP) is a popular programming approach that is replacing traditional procedural programming techniques.

Software systems developed using procedural programming languages are based on the paradigm of procedures. Object-oriented programming models the real world in terms of

objects. Everything in the world can be modeled as an object. A circle is an object, a person is an object, and a Window icon is an object. Even a loan can be perceived as an object. A Java program is object-oriented because programming in Java is centered on creating objects, manipulating objects, and making objects work together.

Part I, “Fundamentals of Programming,” introduces primitive data types and operations, control statements, methods, and arrays. These are the fundamentals for all programming languages. You will learn object-oriented programming in Part II, “Object-Oriented Programming.”

One of the central issues in software development is how to reuse code. Object-oriented programming provides great flexibility, modularity, clarity, and reusability through encapsulation, inheritance, and polymorphism—all of which you will learn about in this book. For years, object-oriented technology was perceived as elitist, requiring a substantial investment in training and infrastructure. Java has helped object-oriented technology enter the mainstream of computing. Its simple, clean syntax makes programs easy to write and read. Java programs are quite *expressive* in terms of designing and developing applications.

3 Java Is Distributed

Distributed computing involves several computers working together on a network. Java is designed to make distributed computing easy. Since networking capability is inherently integrated into Java, writing network programs is like sending and receiving data to and from a file.

4 Java Is Interpreted

You need an interpreter to run Java programs. The programs are compiled into the Java Virtual Machine code called *bytecode*. The bytecode is machine-independent and can run on any machine that has a Java interpreter, which is part of the Java Virtual Machine (JVM).

Most compilers, including C++ compilers, translate programs in a high-level language to machine code. The code can only run on the native machine. If you run the program on other machines, it has to be recompiled on the native machine. For instance, if you compile a C++ program in Windows, the executable code generated by the compiler can only run on the Windows platform. With Java, you compile the source code once, and the bytecode generated by a Java compiler

can run on any platform with a Java interpreter. The Java interpreter translates the bytecode into the machine language of the target machine.

5 Java Is Robust

Robust means *reliable*. No programming language can ensure complete reliability. Java puts a lot of emphasis on early checking for possible errors, because Java compilers can detect many problems that would first show up at execution time in other languages. Java has eliminated certain types of error-prone programming constructs found in other languages. It does not support pointers, for example, thereby eliminating the possibility of overwriting memory and corrupting data.

Java has a runtime exception-handling feature to provide programming support for robustness. Java forces the programmer to write the code to deal with exceptions. Java can catch and respond to an exceptional situation so that the program can continue its normal execution and terminate gracefully when a runtime error occurs.

6 Java Is Secure

As an Internet programming language, Java is used in a networked and distributed environment. If you download a Java applet (a special kind of program) and run it on your computer, it will not damage your system because Java implements several security mechanisms to protect your system against harm caused by stray programs. The security is based on the premise that *nothing should be trusted*.

7 Java Is Architecture-Neutral

Java is interpreted. This feature enables Java to be *architecture-neutral*, or to use an alternative term, *platform-independent*. With a Java Virtual Machine (JVM), you can write one program that will run on any platform, as shown in Figure 1.5.

Java’s initial success stemmed from its Web-programming capability. You can run Java applets from a Web browser, but Java is for more than just writing Web applets. You can also run standalone Java applications directly from operating systems, using a Java interpreter. Today, software vendors usually develop multiple versions of the same product to run on different platforms (Windows, OS/2,

Macintosh, and various UNIX, IBM AS/400, and IBM mainframes). Using Java, developers need to write only one version that can run on every platform.

8 Java Is Portable

Because Java is architecture neutral, Java programs are portable. They can be run on any platform without being recompiled. Moreover, there are no platform-specific features in the Java language. In some languages, such as Ada, the largest integer varies on different platforms. But in Java, the range of the integer is the same on every platform, as is the behavior of arithmetic. The fixed range of the numbers makes the program portable.

The Java environment is portable to new hardware and operating systems. In fact, the Java compiler itself is written in Java.

9 Java's Performance

Java’s performance is sometimes criticized. The execution of the bytecode is never as fast as it would be with a compiled language, such as C++. Because Java is interpreted, the bytecode is not directly executed by the system, but is run through the interpreter. However, its speed is more than adequate for most interactive applications, where the CPU is often idle, waiting for input or for data from other sources.

CPU speed has increased dramatically in the past few years, and this trend will continue. There are many ways to improve performance. Users of the earlier Sun Java Virtual Machine certainly noticed that Java was slow. However, the new JVM is significantly faster. The new JVM uses the technology known as *just-in-time compilation*. It compiles bytecode into native machine code, stores the native code, and reinvokes the native code when its bytecode is executed. Sun recently developed the Java HotSpot Performance Engine, which includes a compiler for optimizing the frequently used code. The HotSpot Performance Engine can be plugged into a JVM to dramatically boost its performance.

10 Java Is Multithreaded

Multithreading is a program’s capability to perform several

tasks simultaneously. For example, downloading a video file while playing the video would be considered multithreading. Multithread programming is smoothly integrated in Java, whereas in other languages you have to call procedures specific to the operating system to enable multithreading.

Multithreading is particularly useful in graphical user interface (GUI) and network programming. In GUI programming, there are many things going on at the same time. A user can listen to an audio recording while surfing a Web page. In network programming, a server can serve multiple clients at the same time. Multithreading is a necessity in multimedia and network programming.

11 Java Is Dynamic

Java was designed to adapt to an evolving environment. New class can be loaded on the fly without recompilation. There is no need for developers to create, and for users to install, major new software versions. New features can be incorporated transparently as needed.