CHAPTER 1

INTRODUCTION TO COMPUTERS, PROGRAMS, AND JAVA

Objectives

- To review computer basics, programs, and operating systems (§§1.2–1.4).
- To explore the relationship between Java and the World Wide Web (§1.5).
- To distinguish the terms API, IDE, and JDK (§1.6).
- To write a simple Java program (§1.7).
- To display output on the console (§1.7).
- To explain the basic syntax of a Java program (§1.7).
- To create, compile, and run Java programs (§1.8).
- (GUI) To display output using the `JOptionPane` output dialog boxes (§1.9).

1.1 Introduction

You use word processors to write documents, Web browsers to explore the Internet, and email programs to send email. These are all examples of software that runs on computers. Software is developed using programming languages. There are many programming languages—so *why Java*? The answer is that Java enables users to develop and deploy applications on the Internet for servers, desktop computers, and small hand-held devices. The future of computing is being profoundly influenced by the Internet, and Java promises to remain a big part of that future. Java is *the* Internet programming language.

why Java?

You are about to begin an exciting journey, learning a powerful programming language. At the outset, it is helpful to review computer basics, programs, and operating systems and to become familiar with number systems. If you are already familiar with such terms as CPU, memory, disks, operating systems, and programming languages, you may skip the review in §§1.2–1.4.

1.2 What Is a Computer?

A computer is an electronic device that stores and processes data. It includes both *hardware* and *software*. In general, hardware comprises the visible, physical elements of the computer, and software provides the invisible instructions that control the hardware and make it perform specific tasks. Writing instructions for computers to perform is called computer programming. Knowing computer hardware isn't essential to your learning a programming language, but it does help you understand better the effect of the program instructions. This section introduces computer hardware components and their functions.

hardware
software

A computer consists of the following major hardware components (Figure 1.1):

- Central processing unit (CPU)
- Memory (main memory)
- Storage devices (e.g., disks, CDs, tapes)
- Input and output devices (e.g., monitors, keyboards, mice, printers)
- Communication devices (e.g., modems and network interface cards (NICs))

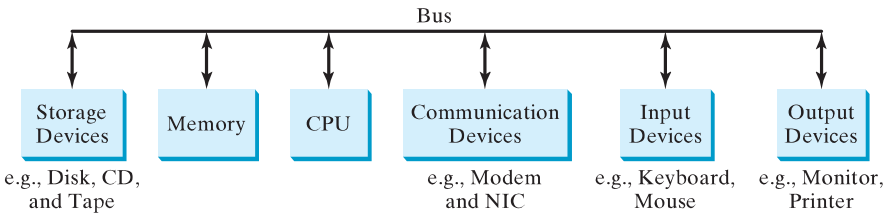


FIGURE 1.1 A computer consists of CPU, memory, storage devices, input devices, output devices, and communication devices.

bus

The components are connected through a subsystem called a *bus* that transfers data or power between them.

1.2.1 Central Processing Unit

CPU

The *central processing unit* (CPU) is the computer's brain. It retrieves instructions from memory and executes them. The CPU usually has two components: a *control unit* and an *arithmetic/logic unit*. The control unit controls and coordinates the actions of the other

components. The arithmetic/logic unit performs numeric operations (addition, subtraction, multiplication, division) and logical operations (comparisons).

Today's CPU is built on a small silicon semiconductor chip having millions of transistors. Every computer has an internal clock, which emits electronic pulses at a constant rate. These pulses are used to control and synchronize the pace of operations. The higher the clock speed, the more instructions are executed in a given period of time. The unit of measurement of clock speed is the *hertz* (*Hz*), with 1 hertz equaling 1 pulse per second. The clock speed of a computer is usually stated in megahertz (MHz) (1 MHz is 1 million Hz). CPU speed has been improved continuously. Intel's Pentium 3 Processor runs at about 500 megahertz and Pentium 4 Processor at about 3 gigahertz (GHz) (1 GHz is 1000 MHz).

speed
hertz
megahertz
gigahertz

1.2.2 Memory

To store and process information, computers use *off* and *on* electrical states, referred to by convention as *0* and *1*. These 0s and 1s are interpreted as digits in the binary number system and called *bits* (*b*inary dig*its*). Data of various kinds, such as numbers, characters, and strings, are encoded as series of bits. Data and program instructions for the CPU to execute are stored as groups of bits, or bytes, each byte composed of eight bits, in a computer's *memory*. A memory unit is an ordered sequence of *bytes*, as shown in Figure 1.2.

bit
byte

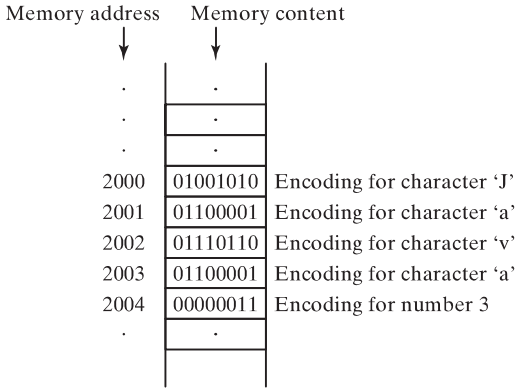


FIGURE 1.2 Memory stores data and program instructions.

The programmer need not be concerned about the encoding and decoding of data, which the system performs automatically, based on the encoding scheme. In the popular ASCII encoding scheme, for example, character `'J'` is represented by `01001010` in one byte.

A byte is the minimum storage unit. A small number such as `3` can be stored in a single byte. To store a number that cannot fit into a single byte, the computer uses several adjacent bytes. No two data items can share or split the same byte.

A memory byte is never empty, but its initial content may be meaningless to your program. The current content of a memory byte is lost whenever new information is placed in it.

A program and its data must be brought to memory before they can be executed.

Every byte has a unique address. The address is used to locate the byte for storing and retrieving data. Since bytes can be accessed in any order, the memory is also referred to as *random-access memory* (*RAM*). Today's personal computers usually have at least 1 gigabyte of RAM. Computer storage size is measured in bytes, kilobytes (KB), megabytes (MB), gigabytes (GB), and terabytes (TB). A *kilobyte* is $2^{10} = 1024$, about 1000 bytes, a *megabyte* is $2^{20} = 1048576$, about 1 million bytes, a *gigabyte* is about 1 billion bytes, and a terabyte is about 1000 gigabytes. Like the CPU, memory is built on silicon semiconductor chips having thousands of transistors embedded on their surface. Compared to CPU chips, memory chips are less complicated, slower, and less expensive.

RAM
megabyte

1.2.3 Storage Devices

Memory is volatile, because information is lost when the power is turned off. Programs and data are permanently stored on storage devices and are moved, when the computer actually uses them, to memory, which is much faster than storage devices.

There are four main types of storage devices:

- Disk drives
- CD drives (CD-R, CD-RW, and DVD)
- Tape drives
- USB flash drives

drive

Drives are devices for operating a medium, such as disks, CDs, and tapes.

Disks

hard disk

Each computer has at least one hard drive. *Hard disks* are for permanently storing data and programs. The hard disks of the latest PCs store from 80 to 250 gigabytes. Often disk drives are encased inside the computer. Removable hard disks are also available.

CDs and DVDs

CD-R

CD stands for compact disk. There are two types of CD drives: CD-R and CD-RW. A *CD-R* is for read-only permanent storage; the user cannot modify its contents once they are recorded.

CD-RW

A *CD-RW* can be used like a hard disk and can be both read and rewritten. A single CD can hold up to 700 MB. Most software is distributed through CD-ROMs. Most new PCs are equipped with a CD-RW drive that can work with both CD-R and CD-RW.

DVD stands for digital versatile disc or digital video disk. DVDs and CDs look alike, and you can use either to store data. A DVD can hold more information than a CD. A standard DVD's storage capacity is 4.7 GB.

Tapes

Tapes are mainly used for backup of data and programs. Unlike disks and CDs, tapes store information sequentially. The computer must retrieve information in the order it was stored. Tapes are very slow. It would take one to two hours to back up a 1-gigabyte hard disk. The new trend is to back up data using flash drives or external hard disks.

USB Flash Drives

USB flash drives are devices for storing and transporting data. A flash drive is small—about the size of a pack of gum. It acts like a portable hard drive that can be plugged into your computer's USB port. USB flash drives are currently available with up to 32 GB storage capacity.

1.2.4 Input and Output Devices

Input and output devices let the user communicate with the computer. The common input devices are *keyboards* and *mice*. The common output devices are *monitors* and *printers*.

The Keyboard

A computer *keyboard* resembles a typewriter keyboard with extra keys added for certain special functions.

function key

Function keys are located at the top of the keyboard and are numbered with prefix F. Their use depends on the software.

modifier key

A *modifier key* is a special key (e.g., *Shift*, *Alt*, *Ctrl*) that modifies the normal action of another key when the two are pressed in combination.

The *numeric keypad,* located on the right-hand corner of the keyboard, is a separate set of keys for quick input of numbers. numeric keypad

Arrow keys, located between the main keypad and the numeric keypad, are used to move the cursor up, down, left, and right.

The *Insert, Delete, Page Up, and Page Down keys*, located above the arrow keys, are used in word processing for performing insert, delete, page up, and page down.

The Mouse

A *mouse* is a pointing device. It is used to move an electronic pointer called a cursor around the screen or to click on an object on the screen to trigger it to respond.

The Monitor

The *monitor* displays information (text and graphics). The screen resolution and dot pitch determine the quality of the display.

The *screen resolution* specifies the number of pixels per square inch. Pixels (short for "picture elements") are tiny dots that form an image on the screen. A common resolution for a 17-inch screen, for example, is 1024 pixels wide and 768 pixels high. The resolution can be set manually. The higher the resolution, the sharper and clearer the image is. screen resolution

The *dot pitch* is the amount of space between pixels in millimeters. The smaller the dot pitch, the better the display. dot pitch

1.2.5 Communication Devices

Computers can be networked through communication devices, such as the dialup modem (*mo*dulator/*demo*dulator), DSL, cable modem, network interface card, and wireless. A dialup modem uses a phone line and can transfer data at a speed up to 56,000 bps (bits per second). A *DSL* (digital subscriber line) also uses a phone line and can transfer data twenty times faster. A cable modem uses the TV cable line maintained by the cable company and is as fast as a DSL. A network interface card (NIC) is a device that connects a computer to a local area network (LAN). The *LAN* is commonly used in universities and business and government organizations. A typical *NIC* called *10BaseT* can transfer data at 10 *mbps* (million bits per second). Wireless is becoming popular. Every laptop sold today is equipped with a wireless adapter that enables the computer to connect with the Internet. modem DSL NIC LAN mbps

1.3 Programs

Computer *programs*, known as *software*, are instructions to the computer, telling it what to do. Computers do not understand human languages, so you need to use computer languages in computer programs. *Programming* is the creation of a program that is executable by a computer and performs the required tasks. software programming

A computer's native language, which differs among different types of computers, is its *machine language*—a set of built-in primitive instructions. These instructions are in the form of binary code, so in telling the machine what to do, you have to enter binary code. Programming in machine language is a tedious process. Moreover, the programs are highly difficult to read and modify. For example, to add two numbers, you might have to write an instruction in binary like this: machine language

```
1101101010011010
```

Assembly language is a low-level programming language in which a mnemonic is used to represent each of the machine-language instructions. For example, to add two numbers, you might write an instruction in assembly code like this: assembly language

```
ADDF3 R1, R2, R3
```

assembler

Assembly languages were developed to make programming easy. However, since the computer cannot understand assembly language, a program called an *assembler* is used to convert assembly-language programs into machine code, as shown in Figure 1.3.

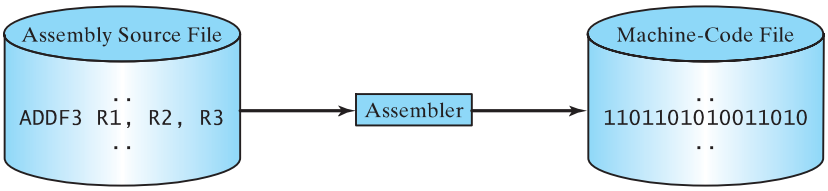


FIGURE 1.3 Assembler translates assembly-language instructions to machine code.

Assembly programs are written in terms of machine instructions with easy-to-remember mnemonic names. Since assembly language is machine dependent, an assembly program can be executed only on a particular kind of machine. The high-level languages were developed in order to transcend platform specificity and make programming easier.

high-level language

The *high-level languages* are English-like and easy to learn and program. Here, for example, is a high-level language statement that computes the area of a circle with radius 5:

```
area = 5 * 5 * 3.1415;
```

Among the more than one hundred high-level languages, the following are well known:

- COBOL (COmmon Business Oriented Language)
- FORTRAN (FORmula TRANslation)
- BASIC (Beginner's All-purpose Symbolic Instruction Code)
- Pascal (named for Blaise Pascal)
- Ada (named for Ada Lovelace)
- C (developed by the designer of B)
- Visual Basic (Basic-like visual language developed by Microsoft)
- Delphi (Pascal-like visual language developed by Borland)
- C++ (an object-oriented language, based on C)
- C# (a Java-like language developed by Microsoft)
- Java

Each of these languages was designed for a specific purpose. COBOL was designed for business applications and is used primarily for business data processing. FORTRAN was designed for mathematical computations and is used mainly for numeric computations. BASIC was designed to be learned and used easily. Ada was developed for the Department of Defense and is used mainly in defense projects. C combines the power of an assembly language with the ease of use and portability of a high-level language. Visual Basic and Delphi are used in developing graphical user interfaces and in rapid application development. C++ is popular for system software projects such as writing compilers and operating systems. The Microsoft Windows operating system was coded using C++. C# (pronounced C sharp) is a new language developed by Microsoft for developing applications based on the Microsoft .NET platform. Java, developed by Sun Microsystems, is widely used for developing platform-independent Internet applications.

A program written in a high-level language is called a *source program* or *source code*. Since a computer cannot understand a source program, a program called a *compiler* is used to translate it into a machine-language program. The machine-language program is then linked with other supporting library code to form an executable file, which can be run on the machine, as shown in Figure 1.4. On Windows, executable files have extension .exe.

source program
compiler

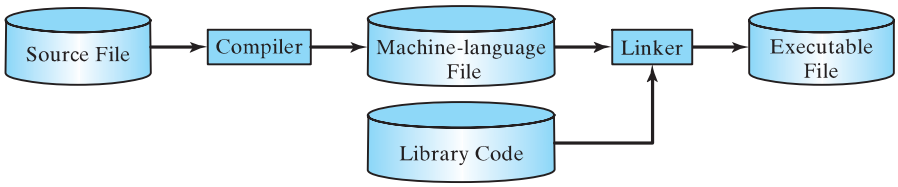


FIGURE 1.4 A source program is compiled into a machine-language file, which is then linked with the system library to form an executable file.

1.4 Operating Systems

The *operating system* (*OS*) is the most important program that runs on a computer, which manages and controls a computer's activities. The popular operating systems are Microsoft Windows, Mac OS, and Linux. Application programs, such as a Web browser or a word processor, cannot run without an operating system. The interrelationship of hardware, operating system, application software, and the user is shown in Figure 1.5.

OS

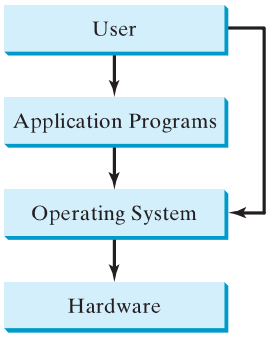


FIGURE 1.5 The operating system is the software that controls and manages the system.

The major tasks of an operating system are:

- Controlling and monitoring system activities
- Allocating and assigning system resources
- Scheduling operations

1.4.1 Controlling and Monitoring System Activities

Operating systems perform basic tasks, such as recognizing input from the keyboard, sending output to the monitor, keeping track of files and directories on the disk, and controlling peripheral devices, such as disk drives and printers. They also make sure that different programs and users running at the same time do not interfere with each other, and they are responsible for security, ensuring that unauthorized users do not access the system.

1.4.2 Allocating and Assigning System Resources

The operating system is responsible for determining what computer resources a program needs (e.g., CPU, memory, disks, input and output devices) and for allocating and assigning them to run the program.

1.4.3 Scheduling Operations

The OS is responsible for scheduling programs to make efficient use of system resources. Many of today's operating systems support such techniques as *multiprogramming*, *multithreading*, or *multiprocessing* to increase system performance.

multiprogramming

Multiprogramming allows multiple programs to run simultaneously by sharing the CPU. The CPU is much faster than the computer's other components. As a result, it is idle most of the time—for example, while waiting for data to be transferred from the disk or from other sources. A multiprogramming OS takes advantage of this situation by allowing multiple programs to use the CPU when it would otherwise be idle. For example, you may use a word processor to edit a file at the same time as the Web browser is downloading a file.

multithreading

Multithreading allows concurrency within a program, so that its subtasks can run at the same time. For example, a word-processing program allows users to simultaneously edit text and save it to a file. In this example, editing and saving are two tasks within the same application. These two tasks may run on separate threads concurrently.

multiprocessing

Multiprocessing, or parallel processing, uses two or more processors together to perform a task. It is like a surgical operation where several doctors work together on one patient.

1.5 Java, World Wide Web, and Beyond

This book introduces Java programming. Java was developed by a team led by James Gosling at Sun Microsystems. Originally called *Oak*, it was designed in 1991 for use in embedded chips in consumer electronic appliances. In 1995, renamed *Java*, it was redesigned for developing Internet applications. For the history of Java, see java.sun.com/features/1998/05/birthday.html.

Java has become enormously popular. Its rapid rise and wide acceptance can be traced to its design characteristics, particularly its promise that you can write a program once and run it anywhere. As stated by Sun, Java is *simple*, *object oriented*, *distributed*, *interpreted*, *robust*, *secure*, *architecture neutral*, *portable*, *high performance*, *multithreaded*, and *dynamic*. For the anatomy of Java characteristics, see www.cs.armstrong.edu/liang/JavaCharacteristics.pdf.

Java is a full-featured, general-purpose programming language that can be used to develop robust mission-critical applications. Today, it is employed not only for Web programming, but also for developing standalone applications across platforms on servers, desktops, and mobile devices. It was used to develop the code to communicate with and control the robotic rover on Mars. Many companies that once considered Java to be more hype than substance are now using it to create distributed applications accessed by customers and partners across the Internet. For every new project being developed today, companies are asking how they can use Java to make their work easier.

The World Wide Web is an electronic information repository that can be accessed on the Internet from anywhere in the world. The Internet, the Web's infrastructure, has been around for more than thirty years. The colorful World Wide Web and sophisticated Web browsers are the major reason for the Internet's popularity.

The primary authoring language for the Web is the Hypertext Markup Language (HTML). HTML is a simple language for laying out documents, linking documents on the Internet, and bringing images, sound, and video alive on the Web. However, it cannot interact with the user except through simple forms. Web pages in HTML are essentially static and flat.

Java initially became attractive because Java programs can be run from a Web browser. Such programs are called *applets*. Applets employ a modern graphical interface with buttons,

applet

text fields, text areas, radio buttons, and so on, to interact with users on the Web and process their requests. Applets make the Web responsive, interactive, and fun to use. Figure 1.6 shows an applet running from a Web browser for playing a Tic Tac Toe game.

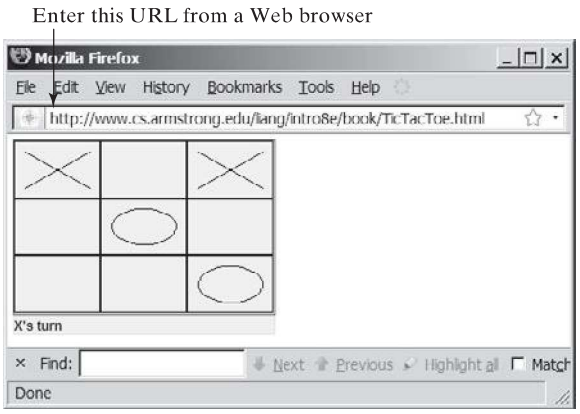


FIGURE 1.6 A Java applet for playing TicTacToe is embedded in an HTML page.

Tip

For a demonstration of Java applets, visit java.sun.com/applets. This site provides a rich Java resource as well as links to other cool applet demo sites. java.sun.com is the official Sun Java Website.

Java can also be used to develop applications on the server side. These applications can be run from a Web server to generate dynamic Web pages. The automatic grading system for this book, as shown in Figure 1.7, was developed using Java.

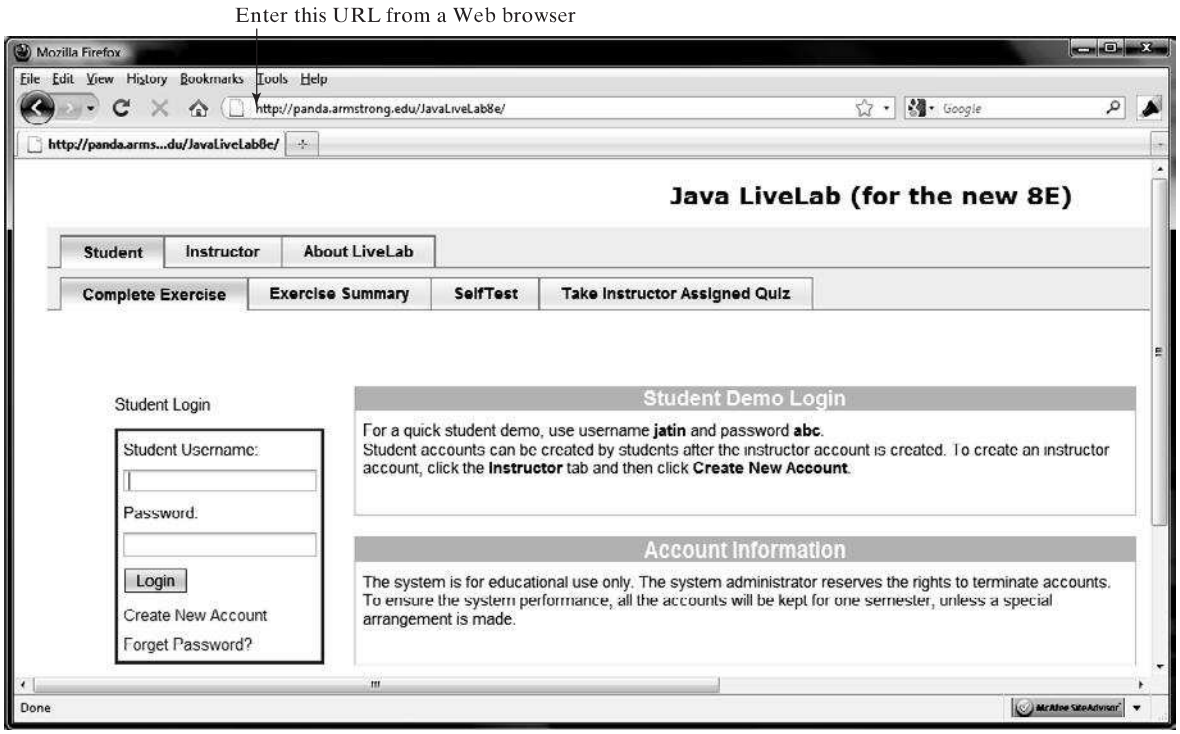


FIGURE 1.7 Java was used to develop an automatic grading system to accompany this book.

Java is a versatile programming language. You can use it to develop applications on your desktop and on the server. You can also use it to develop applications for small hand-held devices. Figure 1.8 shows a Java-programmed calendar displayed on a BlackBerry© and on a cell phone.

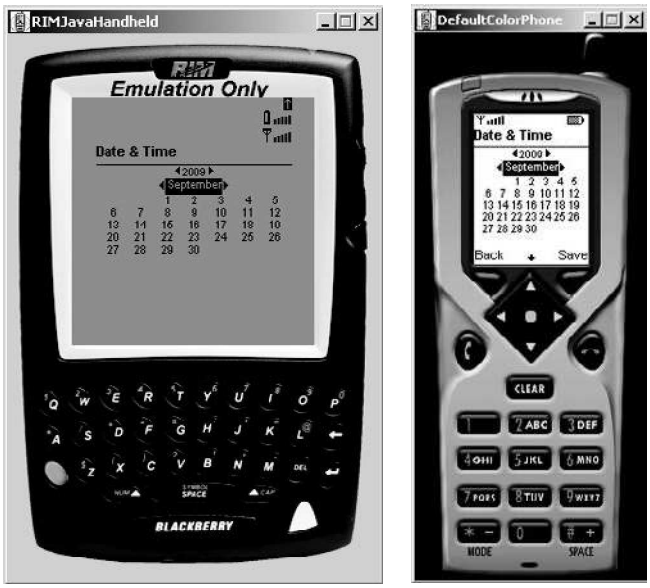


FIGURE 1.8 Java can be used to develop applications for hand-held and wireless devices, such as a BlackBerry© (left) and a cell phone (right).

1.6 The Java Language Specification, API, JDK, and IDE

Computer languages have strict rules of usage. If you do not follow the rules when writing a program, the computer will be unable to understand it. The Java language specification and Java API define the Java standard.

Java language specification

The *Java language specification* is a technical definition of the language that includes the syntax and semantics of the Java programming language. The complete Java language specification can be found at java.sun.com/docs/books/jls.

API

The *application program interface* (*API*) contains predefined classes and interfaces for developing Java programs. The Java language specification is stable, but the API is still expanding. At the Sun Java Website (java.sun.com), you can view and download the latest version of the Java API.

Java SE, EE, and ME

Java is a full-fledged and powerful language that can be used in many ways. It comes in three editions: *Java Standard Edition* (*Java SE*), *Java Enterprise Edition* (*Java EE*), and *Java Micro Edition* (*Java ME*). Java SE can be used to develop client-side standalone applications or applets. Java EE can be used to develop server-side applications, such as Java servlets and JavaServer Pages. Java ME can be used to develop applications for mobile devices, such as cell phones. This book uses Java SE to introduce Java programming.

JDK 1.6 = JDK 6

There are many versions of Java SE. The latest, Java SE 6, will be used in this book. Sun releases each version with a *Java Development Toolkit* (*JDK*). For Java SE 6, the Java Development Toolkit is called *JDK 1.6* (also known as *Java 6 or JDK 6*).

Java IDE

JDK consists of a set of separate programs, each invoked from a command line, for developing and testing Java programs. Besides JDK, you can use a Java development tool (e.g., NetBeans, Eclipse, and TextPad)—software that provides an *integrated development environment* (*IDE*) for rapidly developing Java programs. Editing, compiling, building, debugging, and

online help are integrated in one graphical user interface. Just enter source code in one window or open an existing file in a window, then click a button, menu item, or function key to compile and run the program.

1.7 A Simple Java Program

Let us begin with a simple Java program that displays the message "Welcome to Java!" on the console. *Console* refers to text entry and display device of a computer. The program is shown in Listing 1.1.

console

Video Note
First Java program

LISTING 1.1 Welcome.java

```
1 public class Welcome {                                    class
2   public static void main(String[] args) {                main method
3     // Display message Welcome to Java! to the console
4     System.out.println("Welcome to Java!");                display message
5   }
6 }
```

```
Welcome to Java!
```

The *line numbers* are displayed for reference purposes but are not part of the program. So, don't type line numbers in your program.

line numbers

Line 1 defines a class. Every Java program must have at least one class. Each class has a name. By convention, class names start with an uppercase letter. In this example, the *class name* is `Welcome`.

class name

Line 2 defines the *main method*. In order to run a class, the class must contain a method named `main`. The program is executed from the `main` method.

main method

A method is a construct that contains statements. The `main` method in this program contains the `System.out.println` statement. This statement prints a message `"Welcome to Java!"` to the console (line 4). Every statement in Java ends with a semicolon (`;`), known as the *statement terminator*.

statement terminator

Reserved words, or *keywords,* have a specific meaning to the compiler and cannot be used for other purposes in the program. For example, when the compiler sees the word `class`, it understands that the word after `class` is the name for the class. Other reserved words in this program are `public`, `static`, and `void`.

reserved word

Line 3 is a *comment* that documents what the program is and how it is constructed. Comments help programmers to communicate and understand the program. They are not programming statements and thus are ignored by the compiler. In Java, comments are preceded by two slashes (`//`) on a line, called a *line comment*, or enclosed between `/*` and `*/` on one or several lines, called a *block comment*. When the compiler sees `//`, it ignores all text after `//` on the same line. When it sees `/*`, it scans for the next `*/` and ignores any text between `/*` and `*/`. Here are examples of comments:

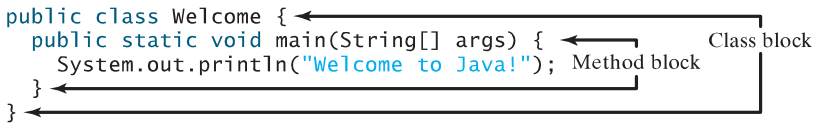
comment

```
// This application program prints Welcome to Java!
/* This application program prints Welcome to Java! */
/* This application program
   prints Welcome to Java! */
```

A pair of braces in a program forms a *block* that groups the program's components. In Java, each block begins with an opening brace (`{`) and ends with a closing brace (`}`). Every class has a *class block* that groups the data and methods of the class. Every method has a *method*

block

block that groups the statements in the method. Blocks can be *nested*, meaning that one block can be placed within another, as shown in the following code.



matching braces

Tip

An opening brace must be matched by a closing brace. Whenever you type an opening brace, immediately type a closing brace to prevent the missing-brace error. Most Java IDEs automatically insert the closing brace for each opening brace.

Note

You are probably wondering why the `main` method is declared this way and why `System.out.println(...)` is used to display a message to the console. For the time being, simply accept that this is how things are done. Your questions will be fully answered in subsequent chapters.

case sensitive

Caution

Java source programs are case sensitive. It would be wrong, for example, to replace `main` in the program with `Main`.

syntax rules

Note

Like any programming language, Java has its own syntax, and you need to write code that obeys the *syntax rules*. If your program violates the rules—for example if the semicolon is missing, a brace is missing, a quotation mark is missing, or `String` is misspelled—the Java compiler will report syntax errors. Try to compile the program with these errors and see what the compiler reports.

The program in Listing 1.1 displays one message. Once you understand the program, it is easy to extend it to display more messages. For example, you can rewrite the program to display three messages, as shown in Listing 1.2.

LISTING 1.2 Welcome1.java

class
main method
display message

```
1 public class Welcome1 {
2   public static void main(String[] args) {
3     System.out.println("Programming is fun!");
4     System.out.println("Fundamentals First");
5     System.out.println("Problem Driven");
6   }
7 }
```

```
Programming is fun!
Fundamentals First
Problem Driven
```

Further, you can perform mathematical computations and display the result to the console. Listing 1.3 gives an example of evaluating $\frac{10.5 + 2 \times 3}{45 - 3.5}$.

LISTING 1.3 ComputeExpression.java

```
1 public class ComputeExpression {                          class
2   public static void main(String[] args) {               main method
3     System.out.println((10.5 + 2 * 3) / (45 - 3.5));      compute expression
4   }
5 }
```

```
0.39759036144578314
```

The multiplication operator in Java is *. As you see, it is a straightforward process to translate an arithmetic expression to a Java expression. We will discuss Java expressions further in Chapter 2.

1.8 Creating, Compiling, and Executing a Java Program

You have to create your program and compile it before it can be executed. This process is repetitive, as shown in Figure 1.9. If your program has compilation errors, you have to modify the program to fix them, then recompile it. If your program has runtime errors or does not produce the correct result, you have to modify the program, recompile it, and execute it again.

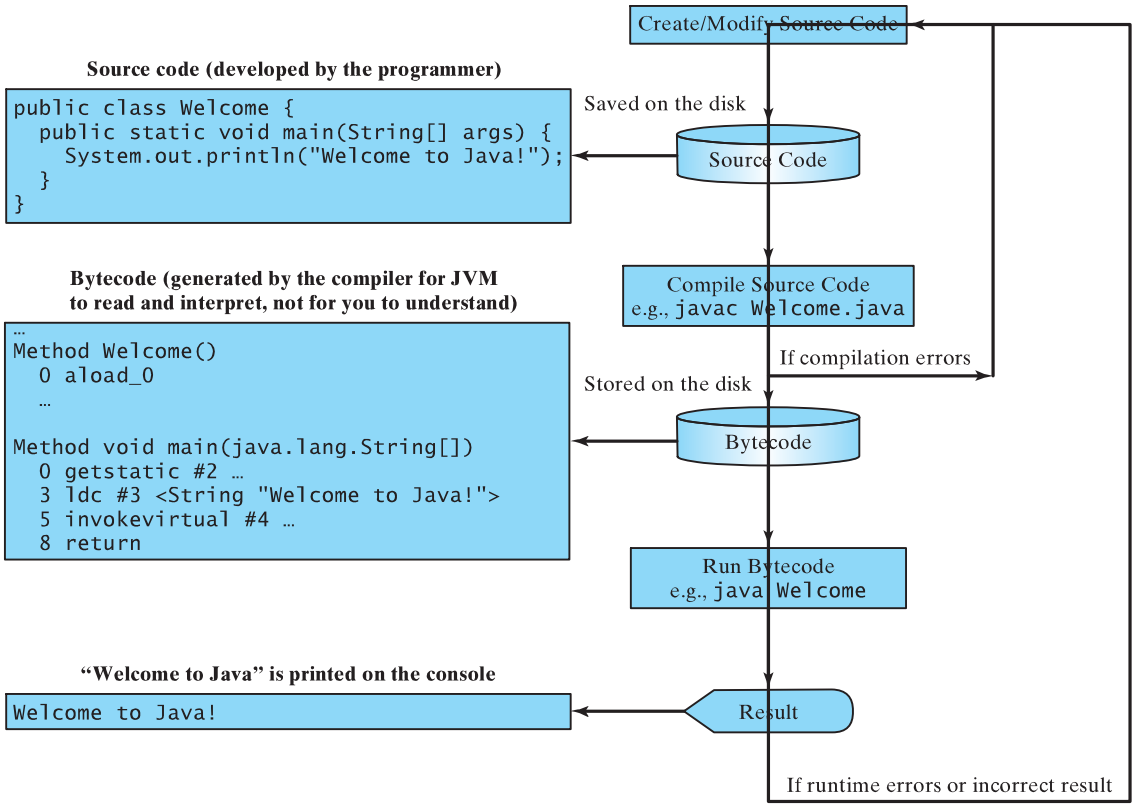


FIGURE 1.9 The Java program-development process consists of repeatedly creating/modifying source code, compiling, and executing programs.

editor

You can use any text *editor* or IDE to create and edit a Java source-code file. This section demonstrates how to create, compile, and run Java programs from a command window. If you wish to use an IDE such as Eclipse, NetBeans, or TextPad, please refer to Supplement II for tutorials. From the command window, you can use the NotePad to create the Java source code file, as shown in Figure 1.10.

Video Note
Brief Eclipse Tutorial

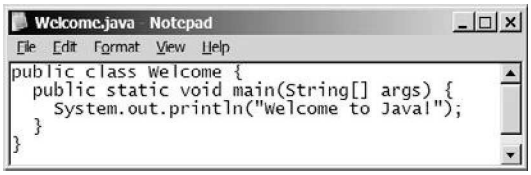


FIGURE 1.10 You can create the Java source file using Windows NotePad.

Note

file name

The source file must end with the extension .java and must have exactly the same name as the public class name. For example, the file for the source code in Listing 1.1 should be named Welcome.java, since the public class name is `Welcome`.

A Java compiler translates a Java source file into a Java bytecode file. The following command compiles Welcome.java:

compile

```
javac Welcome.java
```

Note

Supplement I.B

You must first install and configure JDK before compiling and running programs. See Supplement I.B, "Installing and Configuring JDK 6," on how to install JDK and set up the environment to compile and run Java programs. If you have trouble compiling and running programs, please see

Supplement I.C

Supplement I.C, "Compiling and Running Java from the Command Window." This supplement also explains how to use basic DOS commands and how to use Windows NotePad and WordPad to create and edit files. All the supplements are accessible from the Companion Website.

.class bytecode file

If there are no syntax errors, the *compiler* generates a bytecode file with a .class extension. So the preceding command generates a file named **Welcome. class**, as shown in Figure 1.11(a). The Java language is a high-level language while Java bytecode is a low-level language. The bytecode is similar to machine instructions but is architecture neutral and can run on any platform that has a Java Virtual Machine (JVM), as shown in Figure 1.11(b). Rather than a physical machine, the virtual machine is a program that interprets Java bytecode. This is one of Java's primary advantages: *Java bytecode can run on a variety of hardware platforms and operating systems.*

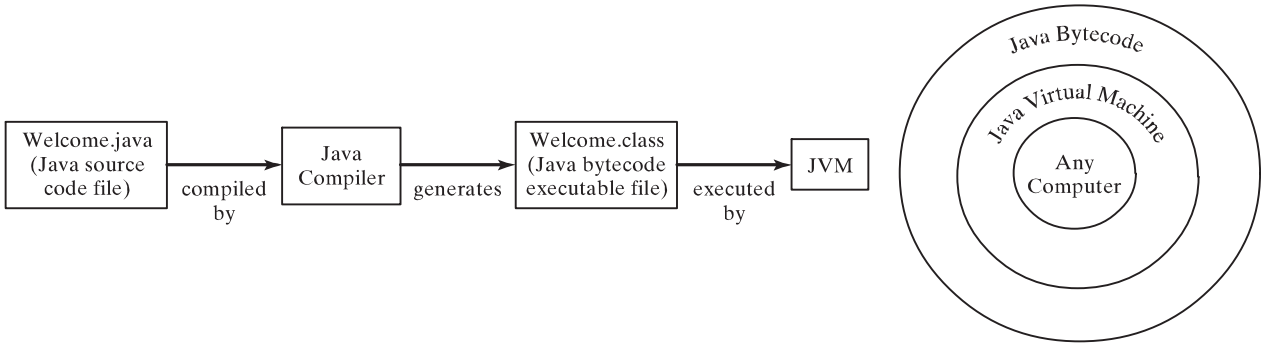


FIGURE 1.11 (a) Java source code is translated into bytecode. (b) Java bytecode can be executed on any computer with a Java Virtual Machine.

To execute a Java program is to run the program's bytecode. You can execute the bytecode on any platform with a JVM. Java bytecode is interpreted. Interpreting translates the individual steps in the bytecode into the target machine-language code one at a time rather than translating the whole program as a single unit. Each step is executed immediately after it is translated.

interpreting bytecode

The following command runs the bytecode:

```
java Welcome
```

run

Figure 1.12 shows the **javac** command for compiling Welcome.java. The compiler generated the Welcome.class file. This file is executed using the **java** command.

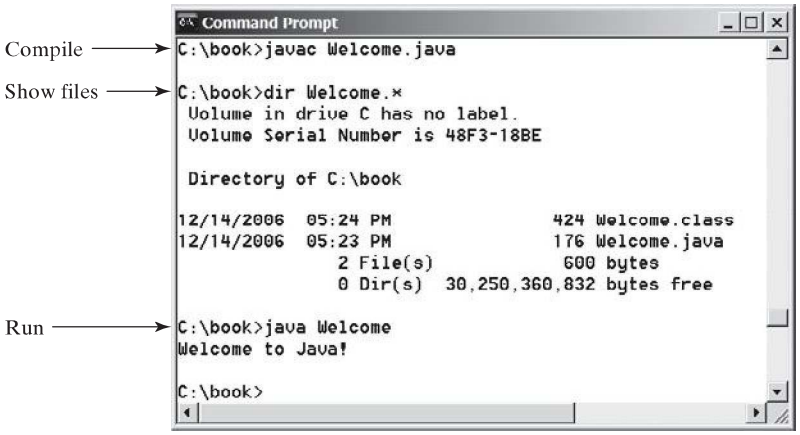


FIGURE 1.12 The output of Listing 1.1 displays the message "Welcome to Java!"

Video Note
Compile and run a Java program

Note

For simplicity and consistency, all source code and class files are placed under **c:\book** unless specified otherwise.

c:\book

Caution

Do not use the extension .class in the command line when executing the program. Use **java ClassName** to run the program. If you use **java ClassName.class** in the command line, the system will attempt to fetch **ClassName.class.class**.

java ClassName

Tip

If you execute a class file that does not exist, a `NoClassDefFoundError` will occur. If you execute a class file that does not have a `main` method or you mistype the `main` method (e.g., by typing `Main` instead of `main`), a `NoSuchMethodError` will occur.

`NoClassDefFoundError`

`NoSuchMethodError`

Note

When executing a Java program, the JVM first loads the bytecode of the class to memory using a program called the *class loader*. If your program uses other classes, the class loader dynamically loads them just before they are needed. After a class is loaded, the JVM uses a program called *bytecode verifier* to check the validity of the bytecode and ensure that the bytecode does not violate Java's security restrictions. Java enforces strict security to make sure that Java programs arriving from the network do not harm your computer.

class loader

bytecode verifier

Pedagogical Note

Instructors may require students to use packages for organizing programs. For example, you may place all programs in this chapter in a package named *chapter1*. For instructions on how to use packages, please see Supplement I.F, "Using Packages to Organize the Classes in the Text."

using package

1.9 (GUI) Displaying Text in a Message Dialog Box

The program in Listing 1.1 displays the text on the console, as shown in Figure 1.12. You can rewrite the program to display the text in a message dialog box. To do so, you need to use the `showMessageDialog` method in the `JOptionPane` class. `JOptionPane` is one of the many predefined classes in the Java system that you can reuse rather than "reinventing the wheel." You can use the `showMessageDialog` method to display any text in a message dialog box, as shown in Figure 1.13. The new program is given in Listing 1.4.

`JOptionPane`

`showMessageDialog`

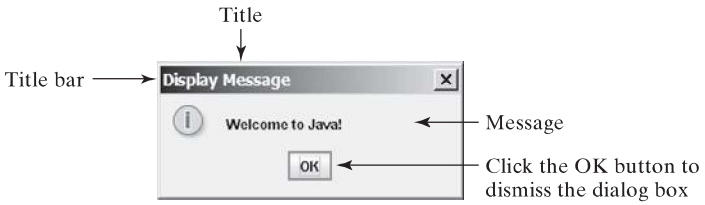


FIGURE 1.13 "Welcome to Java!" is displayed in a message box.

LISTING 1.4 WelcomeInMessageDialogBox.java

block comment

import

main method

display message

```
1 /*  This application program displays Welcome to Java!
2  *  in a message dialog box.
3  */
4 import javax.swing.JOptionPane;
5
6 public class WelcomeInMessageDialogBox {
7   public static void main(String[] args) {
8     // Display Welcome to Java! in a message dialog box
9     JOptionPane.showMessageDialog(null, "Welcome to Java!");
10  }
11 }
```

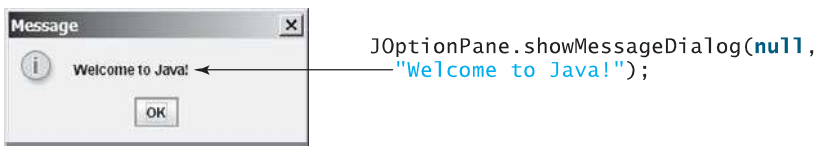
package

This program uses a Java class `JOptionPane` (line 9). Java's predefined classes are grouped into packages. `JOptionPane` is in the `javax.swing` package. `JOptionPane` is imported to the program using the `import` statement in line 4 so that the compiler can locate the class without the full name `javax.swing.JOptionPane`.

Note

If you replace `JOptionPane` on line 9 with `javax.swing.JOptionPane`, you don't need to import it in line 4. `javax.swing.JOptionPane` is the full name for the `JOptionPane` class.

The `showMessageDialog` method is a *static* method. Such a method should be invoked by using the class name followed by a dot operator (`.`) and the method name with arguments. Methods will be introduced in Chapter 5, "Methods." The `showMessageDialog` method can be invoked with two arguments, as shown below.



The first argument can always be `null`. `null` is a Java keyword that will be fully introduced in Chapter 8, "Objects and Classes." The second argument is a string for text to be displayed.

There are several ways to use the `showMessageDialog` method. For the time being, you need to know only two ways. One is to use a statement, as shown in the example:

two versions of `showMessageDialog`

```
JOptionPane.showMessageDialog(null, x);
```

where `x` is a string for the text to be displayed.

The other is to use a statement like this one:

```
JOptionPane.showMessageDialog(null, x,
  y, JOptionPane.INFORMATION_MESSAGE);
```

where `x` is a string for the text to be displayed, and `y` is a string for the title of the message box. The fourth argument can be `JOptionPane.INFORMATION_MESSAGE`, which causes the icon ( ) to be displayed in the message box, as shown in the following example.


```
JOptionPane.showMessageDialog(null,
"Welcome to Java!",
"Display Message",
JOptionPane.INFORMATION_MESSAGE);
```

Note

There are two types of `import` statements: *specific import* and *wildcard import*. The *specific import* specifies a single class in the import statement. For example, the following statement imports `JOptionPane` from package `javax.swing`.

specific import

```
import javax.swing.JOptionPane;
```

The *wildcard import* imports all the classes in a package. For example, the following statement imports all classes from package `javax.swing`.

wildcard import

```
import javax.swing.*;
```

The information for the classes in an imported package is not read in at compile time or runtime unless the class is used in the program. The import statement simply tells the compiler where to locate the classes. There is no performance difference between a specific import and a wildcard import declaration.

no performance difference

Note

Recall that you have used the `System` class in the statement `System.out.println("Welcome to Java");` in Listing 1.1. The `System` class is not imported because it is in the `java.lang` package. All the classes in the `java.lang` package are *implicitly* imported in every Java program.

`java.lang` implicitly imported

Key Terms

.class file 14
.java file 14
assembly language 5
bit 3
block 12
block comment 11
bus 2
byte 3
bytecode 14
bytecode verifier 15
cable modem 5
central processing unit (CPU) 2
class loader 15
comment 11

compiler 7
console 11
dot pitch 5
DSL (digital subscriber line) 5
hardware 2
high-level language 6
Integrated Development Environment (IDE) 10
java command 15
javac command 15
Java Development Toolkit (JDK) 10
Java Virtual Machine (JVM) 14
keyword (or reserved word) 11
line comment 11
machine language 5
main method 11
memory 3
modem 5
network interface card (NIC) 5
operating system (OS) 7
pixel 5
program 5
programming 5
resolution 5
software 5
source code 7
source file 14
specific import 17
storage devices 4
statement 11
wildcard import 17

Note

Supplement I.A

The above terms are defined in the present chapter. Supplement I.A, "Glossary," lists all the key terms and descriptions in the book, organized by chapters.

CHAPTER SUMMARY

1. A computer is an electronic device that stores and processes data.
2. A computer includes both *hardware* and *software*.
3. Hardware is the physical aspect of the computer that can be seen.
4. Computer *programs*, known as *software*, are the invisible instructions that control the hardware and make it perform tasks.
5. Computer programming is the writing of instructions (i.e., code) for computers to perform.
6. The central processing unit (CPU) is a computer's brain. It retrieves instructions from memory and executes them.
7. Computers use zeros and ones because digital devices have two stable states, referred to by convention as zero and one.
8. A bit is a binary digit 0 or 1.
9. A byte is a sequence of 8 bits.
10. A kilobyte is about 1000 bytes, a megabyte about 1 million bytes, a gigabyte about 1 billion bytes, and a terabyte about 1000 gigabytes.
11. Memory stores data and program instructions for the CPU to execute.
12. A memory unit is an ordered sequence of bytes.
13. Memory is volatile, because information is lost when the power is turned off.

14. Programs and data are permanently stored on storage devices and are moved to memory when the computer actually uses them.

15. The machine language is a set of primitive instructions built into every computer.

16. Assembly language is a low-level programming language in which a mnemonic is used to represent each machine-language instruction.

17. High-level languages are English-like and easy to learn and program.

18. A program written in a high-level language is called a source program.

19. A compiler is a software program that translates the source program into a machine-language program.

20. The operating system (OS) is a program that manages and controls a computer's activities.

21. Java is platform independent, meaning that you can write a program once and run it anywhere.

22. Java programs can be embedded in HTML pages and downloaded by Web browsers to bring live animation and interaction to Web clients.

23. Java source files end with the .java extension.

24. Every class is compiled into a separate bytecode file that has the same name as the class and ends with the .class extension.

25. To compile a Java source-code file from the command line, use the **javac** command.

26. To run a Java class from the command line, use the **java** command.

27. Every Java program is a set of class definitions. The keyword `class` introduces a class definition. The contents of the class are included in a block.

28. A block begins with an opening brace ({) and ends with a closing brace (}). Methods are contained in a class.

29. A Java program must have a `main` method. The `main` method is the entry point where the program starts when it is executed.

30. Every statement in Java ends with a semicolon (`;`), known as the *statement terminator*.

31. *Reserved words,* or *keywords,* have a specific meaning to the compiler and cannot be used for other purposes in the program.

32. In Java, comments are preceded by two slashes (`//`) on a line, called a *line comment*, or enclosed between `/*` and `*/` on one or several lines, called a block comment.

33. Java source programs are case sensitive.

34. There are two types of `import` statements: *specific import* and *wildcard import*. The *specific import* specifies a single class in the import statement. The *wildcard import* imports all the classes in a package.

Review Questions

Note

Answers to review questions are on the Companion Website.

Sections 1.2–1.4

1.1 Define hardware and software.

1.2 List the main components of the computer.

1.3 Define machine language, assembly language, and high-level programming language.

1.4 What is a source program? What is a compiler?

1.5 What is the JVM?

1.6 What is an operating system?

Sections 1.5–1.6

1.7 Describe the history of Java. Can Java run on any machine? What is needed to run Java on a computer?

1.8 What are the input and output of a Java compiler?

1.9 List some Java development tools. Are tools like NetBeans and Eclipse different languages from Java, or are they dialects or extensions of Java?

1.10 What is the relationship between Java and HTML?

Sections 1.7–1.9

1.11 Explain the Java keywords. List some Java keywords you learned in this chapter.

1.12 Is Java case sensitive? What is the case for Java keywords?

1.13 What is the Java source filename extension, and what is the Java bytecode filename extension?

1.14 What is a comment? Is the comment ignored by the compiler? How do you denote a comment line and a comment paragraph?

1.15 What is the statement to display a string on the console? What is the statement to display the message "Hello world" in a message dialog box?

1.16 The following program is wrong. Reorder the lines so that the program displays `morning` followed by `afternoon`.

```
  public static void main(String[] args) {
  }

public class Welcome {
    System.out.println("afternoon");
    System.out.println("morning");
}
```

1.17 Identify and fix the errors in the following code:

```
1 public class Welcome {
2   public void Main(String[] args) {
3     System.out.println('Welcome to Java!);
4   }
5 )
```

1.18 What is the command to compile a Java program? What is the command to run a Java program?

1.19 If a `NoClassDefFoundError` occurs when you run a program, what is the cause of the error?

1.20 If a `NoSuchMethodError` occurs when you run a program, what is the cause of the error?

1.21 Why does the `System` class not need to be imported?

1.22 Are there any performance differences between the following two `import` statements?

```
import javax.swing.JOptionPane;

import javax.swing.*;
```

1.23 Show the output of the following code:

```
public class Test {
  public static void main(String[] args) {
    System.out.println("3.5 * 4 / 2 - 2.5 is ");
    System.out.println(3.5 * 4 / 2 - 2.5);
  }
}
```

Programming Exercises

Note

Solutions to even-numbered exercises are on the Companion Website. Solutions to all exercises are on the Instructor Resource Website. The level of difficulty is rated easy (no star), moderate (*), hard (**), or challenging (***).

level of difficulty

1.1 (*Displaying three messages*) Write a program that displays `Welcome to Java`, `Welcome to Computer Science`, and `Programming is fun`.

1.2 (*Displaying five messages*) Write a program that displays `Welcome to Java` five times.

1.3* (*Displaying a pattern*) Write a program that displays the following pattern:

```
    J      A      V      V     A
    J     A A      V    V     A A
J   J    AAAAA      V V     AAAAA
 J J    A     A      V     A     A
```

1.4 (*Printing a table*) Write a program that displays the following table:

```
a       a^2     a^3
1       1       1
2       4       8
3       9       27
4       16      64
```

1.5 (*Computing expressions*) Write a program that displays the result of $\frac{9.5 \times 4.5 - 2.5 \times 3}{45.5 - 3.5}$.

1.6 (*Summation of a series*) Write a program that displays the result of $1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9$.

1.7 (*Approximating* π) π can be computed using the following formula:

$$\pi = 4 \times \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} + \cdots\right)$$

Write a program that displays the result of $4 \times \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13}\right)$. Use `1.0` instead of `1` in your program.