Outline



## 1.1 Introduction

Welcome to Java—one of the world's most widely used computer programming languages and, according to the TIOBE Index, the world's most popular.[1] You're probably familiar with the powerful tasks computers perform. Using this textbook, you'll write instructions in the Java programming language commanding computers to perform those tasks. **Software** (i.e., the instructions you write) controls **hardware** (i.e., computers).

You'll learn *object-oriented programming*—today's key programming methodology. You'll create and work with many *software objects*.

For many organizations, the preferred language for meeting their enterprise programming needs is Java. Java is also widely used for implementing Internet-based applications and software for devices that communicate over a network.

There are billions of personal computers in use and an even larger number of mobile devices with computers at their core. According to Oracle's 2016 JavaOne conference keynote presentation,[2] there are now 10 million Java developers worldwide and Java runs on 15 billion devices (Fig. 1.1), including two billion vehicles and 350 million medical devices. In addition, the explosive growth of mobile phones, tablets and other devices is creating significant opportunities for programming mobile apps.

1. `http://www.tiobe.com/tiobe-index/`
2. `http://bit.ly/JavaOne2016Keynote`

| Devices | | |
|---|---|---|
| Access control systems | Airplane systems | ATMs |
| Automobiles | Blu-ray Disc™ players | Building controls |
| Cable boxes | Copiers | Credit cards |
| CT scanners | Desktop computers | e-Readers |
| Game consoles | GPS navigation systems | Home appliances |
| Home security systems | Internet-of-Things gateways | Light switches |
| Logic controllers | Lottery systems | Medical devices |
| Mobile phones | MRIs | Network switches |
| Optical sensors | Parking meters | Personal computers |
| Point-of-sale terminals | Printers | Robots |
| Routers | Servers | Smart cards |
| Smart meters | Smartpens | Smartphones |
| Tablets | Televisions | Thermostats |
| Transportation passes | TV set-top boxes | Vehicle diagnostic systems |

**Fig. 1.1** | Some devices that use Java.

### *Java Standard Edition*

Java has evolved so rapidly that this eleventh edition of *Java How to Program*—based on **Java Standard Edition 8 (Java SE 8)** and the new **Java Standard Edition 9 (Java SE 9)**—was published just 21 years after the first edition. Java Standard Edition contains the capabilities needed to develop desktop and server applications. The book can be used conveniently with *either* Java SE 8 *or* Java SE 9 (released just after this book was published). For instructors and professionals who want to stay with Java 8 for a while, the Java SE 9 features are discussed in modular, easy-to-include-or-omit sections throughout this book and its Companion Website.

Prior to Java SE 8, Java supported three programming paradigms:

- *procedural programming*,
- *object-oriented programming* and
- *generic programming*.

Java SE 8 added the beginnings of *functional programming with lambdas and streams*. In Chapter 17, we'll show how to use lambdas and streams to write programs faster, more concisely, with fewer bugs and that are easier to *parallelize* (i.e., perform multiple calculations simultaneously) to take advantage of today's *multi-core* hardware architectures to enhance application performance.

### *Java Enterprise Edition*

Java is used in such a broad spectrum of applications that it has two other editions. The **Java Enterprise Edition (Java EE)** is geared toward developing large-scale, distributed networking applications and web-based applications. In the past, most computer applications ran on "standalone" computers (that is, not networked together). Today's applications can

be written with the aim of communicating among the world's computers via the Internet and the web. Later in this book we discuss how to build such web-based applications with Java.

*Java Micro Edition*

The **Java Micro Edition (Java ME)**—a subset of Java SE—is geared toward developing applications for resource-constrained embedded devices, such as smartwatches, television set-top boxes, smart meters (for monitoring electric energy usage) and more. Many of the devices in Fig. 1.1 use Java ME.

## 1.2 Hardware and Software

Computers can perform calculations and make logical decisions phenomenally faster than human beings can. Many of today's personal computers can perform billions of calculations in one second—more than a human can perform in a lifetime. *Supercomputers* are already performing *thousands of trillions (quadrillions)* of instructions per second! China's National Research Center of Parallel Computer Engineering & Technology (NRCPC) has developed the Sunway TaihuLight supercomputer can perform over 93 quadrillion calculations per second (93 *petaflops*)![3] To put that in perspective, *the Sunway TaihuLight supercomputer can perform in one second over 12 million calculations for every person on the planet!* And supercomputing upper limits are growing quickly.

Computers process data under the control of sequences of instructions called **computer programs**. These software programs guide the computer through ordered actions specified by people called computer **programmers**. In this book, you'll learn key programming methodologies that are enhancing programmer productivity, thereby reducing software development costs.

A computer consists of various devices referred to as hardware (e.g., the keyboard, screen, mouse, hard disks, memory, DVD drives and processing units). Computing costs are *dropping dramatically*, owing to rapid developments in hardware and software technologies. Computers that might have filled large rooms and cost millions of dollars decades ago are now inscribed on silicon chips smaller than a fingernail, costing perhaps a few dollars each. Ironically, silicon is one of the most abundant materials on Earth—it's an ingredient in common sand. Silicon-chip technology has made computing so economical that computers have become a commodity.

### 1.2.1 Moore's Law

Every year, you probably expect to pay at least a little more for most products and services. The opposite has been the case in the computer and communications fields, especially with regard to the hardware supporting these technologies. For many decades, hardware costs have fallen rapidly.

Every year or two, the capacities of computers have approximately *doubled* inexpensively. This remarkable trend often is called **Moore's Law**, named for the person who identified it in the 1960s, Gordon Moore, co-founder of Intel—the leading manufacturer of the processors in today's computers and embedded systems. Moore's Law *and related observations* apply especially to the amount of memory that computers have for programs,

3. https://www.top500.org/lists/2016/06/

the amount of secondary storage (such as solid-state drive storage) they have to hold programs and data over longer periods of time, and their processor speeds—the speeds at which they *execute* their programs (i.e., do their work).

Similar growth has occurred in the communications field—costs have plummeted as enormous demand for communications *bandwidth* (i.e., information-carrying capacity) has attracted intense competition. We know of no other fields in which technology improves so quickly and costs fall so rapidly. Such phenomenal improvement is truly fostering the *Information Revolution*.

### 1.2.2 Computer Organization

Regardless of differences in *physical* appearance, computers can be envisioned as divided into various **logical units** or sections (Fig. 1.2).

| Logical unit | Description |
|---|---|
| **Input unit** | This "receiving" section obtains information (data and computer programs) from **input devices** and places it at the disposal of the other units for processing. Most user input is entered into computers through keyboards, touch screens and mouse devices. Other forms of input include receiving voice commands, scanning images and bar codes, reading from secondary storage devices (such as hard drives, DVD drives, Blu-ray Disc™ drives and USB flash drives—also called "thumb drives" or "memory sticks"), receiving video from a webcam and having your computer receive information from the Internet (such as when you stream videos from YouTube® or download e-books from Amazon). Newer forms of input include position data from a GPS device, motion and orientation information from an *accelerometer* (a device that responds to up/down, left/right and forward/backward acceleration) in a smartphone or game controller (such as Microsoft® Kinect® for Xbox®, Wii™ Remote and Sony® PlayStation® Move) and voice input from intelligent assistants like Amazon Echo and Google Home. |
| **Output unit** | This "shipping" section takes information the computer has processed and places it on various **output devices** to make it available for use outside the computer. Most information that's output from computers today is displayed on screens (including touch screens), printed on paper ("going green" discourages this), played as audio or video on PCs and media players (such as Apple's iPods) and giant screens in sports stadiums, transmitted over the Internet or used to control other devices, such as robots and "intelligent" appliances. Information is also commonly output to secondary storage devices, such as solid-state drives (SSDs), hard drives, DVD drives and USB flash drives. Popular recent forms of output are smartphone and game-controller vibration, virtual reality devices like Oculus Rift® and Google Cardboard™ and mixed reality devices like Microsoft's HoloLens™. |

**Fig. 1.2** | Logical units of a computer. (Part 1 of 2.)

| Logical unit | Description |
|---|---|
| **Memory unit** | This rapid-access, relatively low-capacity "warehouse" section retains information that has been entered through the input unit, making it immediately available for processing when needed. The memory unit also retains processed information until it can be placed on output devices by the output unit. Information in the memory unit is *volatile*—it's typically lost when the computer's power is turned off. The memory unit is often called either **memory**, **primary memory** or **RAM** (Random Access Memory). Main memories on desktop and notebook computers contain as much as 128 GB of RAM, though 2 to 16 GB is most common. GB stands for gigabytes; a gigabyte is approximately one billion bytes. A **byte** is eight bits. A bit is either a 0 or a 1. |
| **Arithmetic and logic unit (ALU)** | This "manufacturing" section performs *calculations*, such as addition, subtraction, multiplication and division. It also contains the *decision* mechanisms that allow the computer, for example, to compare two items from the memory unit to determine whether they're equal. In today's systems, the ALU is implemented as part of the next logical unit, the CPU. |
| **Central processing unit (CPU)** | This "administrative" section coordinates and supervises the operation of the other sections. The CPU tells the input unit when information should be read into the memory unit, tells the ALU when information from the memory unit should be used in calculations and tells the output unit when to send information from the memory unit to certain output devices. Many of today's computers have multiple CPUs and, hence, can perform many operations simultaneously. A **multicore processor** implements multiple processors on a single integrated-circuit chip—a *dual-core processor* has two CPUs, a *quad-core processor* has four and an *octa-core processor* has eight. Intel has some processors with up to 72 cores. Today's desktop computers have processors that can execute billions of instructions per second. Chapter 23 explores how to write apps that can take full advantage of multicore architecture. |
| **Secondary storage unit** | This is the long-term, high-capacity "warehousing" section. Programs or data not actively being used by the other units normally are placed on secondary storage devices (e.g., your *hard drive*) until they're again needed, possibly hours, days, months or even years later. Information on secondary storage devices is *persistent*—it's preserved even when the computer's power is turned off. Secondary storage information takes much longer to access than information in primary memory, but its cost per unit is much less. Examples of secondary storage devices include solid-state drives (SSDs), hard drives, DVD drives and USB flash drives, some of which can hold over 2 TB (TB stands for terabytes; a terabyte is approximately one trillion bytes). Typical hard drives on desktop and notebook computers hold up to 2 TB, and some desktop hard drives can hold up to 10 TB. |

**Fig. 1.2** | Logical units of a computer. (Part 2 of 2.)

## 1.3 Data Hierarchy

Data items processed by computers form a **data hierarchy** that becomes larger and more complex in structure as we progress from the simplest data items (called "bits") to richer ones, such as characters and fields. Figure 1.3 illustrates a portion of the data hierarchy.

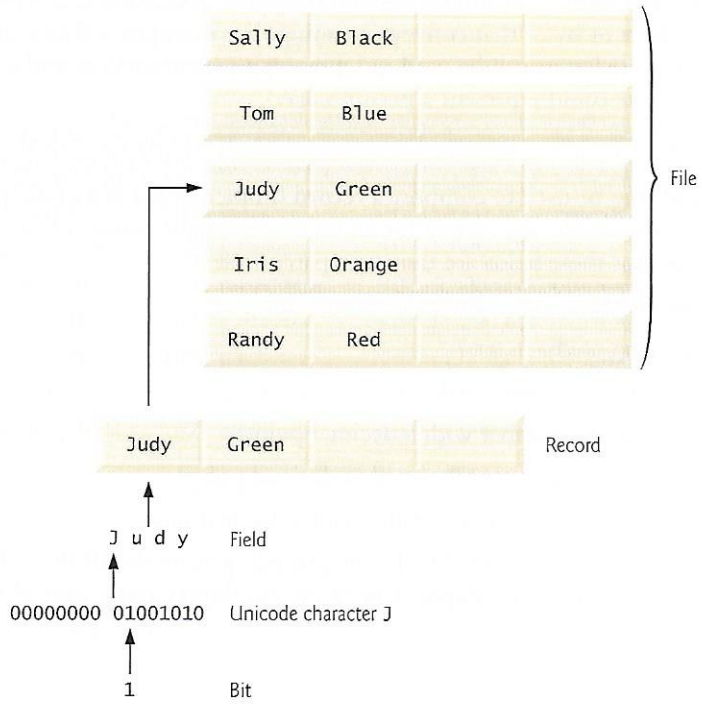


**Fig. 1.3** | Data hierarchy.

*Bits*

The smallest data item in a computer can assume the value 0 or the value 1. It's called a **bit** (short for "binary digit"—a digit that can assume one of *two* values). Remarkably, the impressive functions performed by computers involve only the simplest manipulations of 0s and 1s—*examining a bit's value*, *setting a bit's value* and *reversing a bit's value* (from 1 to 0 or from 0 to 1).

*Characters*

It's tedious for people to work with data in the low-level form of bits. Instead, they prefer to work with *decimal digits* (0–9), *letters* (A–Z and a–z), and *special symbols* (e.g., $, @, %, &, *, (, ), –, +, ", :, ? and /). Digits, letters and special symbols are known as **characters**. The computer's **character set** is the set of all the characters used to write programs and represent data items. Computers process only 1s and 0s, so a computer's character set represents every character as a pattern of 1s and 0s. Java uses **Unicode**® characters that are composed of one, two or four bytes (8, 16 or 32 bits). Unicode contains characters for

many of the world's languages. See Appendix H for more information on Unicode. See Appendix B for more information on the **ASCII (American Standard Code for Information Interchange)** character set—the popular subset of Unicode that represents uppercase and lowercase letters, digits and some common special characters.

*Fields*

Just as characters are composed of bits, **fields** are composed of characters or bytes. A field is a group of characters or bytes that conveys meaning. For example, a field consisting of uppercase and lowercase letters can be used to represent a person's name, and a field consisting of decimal digits could represent a person's age.

*Records*

Several related fields can be used to compose a **record** (implemented as a `class` in Java). In a payroll system, for example, the record for an employee might consist of the following fields (possible types for these fields are shown in parentheses):

- Employee identification number (a whole number)
- Name (a string of characters)
- Address (a string of characters)
- Hourly pay rate (a number with a decimal point)
- Year-to-date earnings (a number with a decimal point)
- Amount of taxes withheld (a number with a decimal point)

Thus, a record is a group of related fields. In the preceding example, all the fields belong to the *same* employee. A company might have many employees and a payroll record for each.

*Files*

A **file** is a group of related records. [*Note:* More generally, a file contains arbitrary data in arbitrary formats. In some operating systems, a file is viewed simply as a *sequence of bytes*—any organization of the bytes in a file, such as organizing the data into records, is a view created by the application programmer. You'll see how to do that in Chapter 15.] It's not unusual for an organization to have many files, some containing billions, or even trillions, of characters of information.

*Database*

A **database** is a collection of data organized for easy access and manipulation. The most popular model is the *relational database*, in which data is stored in simple *tables*. A table includes *records* and *fields*. For example, a table of students might include first name, last name, major, year, student ID number and grade point average fields. The data for each student is a record, and the individual pieces of information in each record are the fields. You can *search*, *sort* and otherwise manipulate the data based on its relationship to multiple tables or databases. For example, a university might use data from the student database in combination with data from databases of courses, on-campus housing, meal plans, etc. We discuss databases in Chapter 24, Accessing Databases with JDBC and online Chapter 29, Java Persistence API (JPA).

*Big Data*

The amount of data being produced worldwide is enormous and growing quickly. According to IBM, approximately 2.5 quintillion bytes (2.5 *exabytes*) of data are created daily,[4] and according to Salesforce.com, as of October 2015 90% of the world's data was created in just the prior 12 months![5] According to an IDC study, the global data supply will reach 40 *zettabytes* (equal to 40 trillion gigabytes) annually by 2020.[6] Figure 1.4 shows some common byte measurements. **Big data** applications deal with massive amounts of data and this field is growing quickly, creating lots of opportunity for software developers. Millions of IT jobs globally already are supporting big data applications.

| Unit | Bytes | Which is approximately |
|---|---|---|
| 1 kilobyte (KB) | 1024 bytes | $10^3$ (1024) bytes exactly |
| 1 megabyte (MB) | 1024 kilobytes | $10^6$ (1,000,000) bytes |
| 1 gigabyte (GB) | 1024 megabytes | $10^9$ (1,000,000,000) bytes |
| 1 terabyte (TB) | 1024 gigabytes | $10^{12}$ (1,000,000,000,000) bytes |
| 1 petabyte (PB) | 1024 terabytes | $10^{15}$ (1,000,000,000,000,000) bytes |
| 1 exabyte (EB) | 1024 petabytes | $10^{18}$ (1,000,000,000,000,000,000) bytes |
| 1 zettabyte (ZB) | 1024 exabytes | $10^{21}$ (1,000,000,000,000,000,000,000) bytes |

**Fig. 1.4** | Byte measurements.

## 1.4 Machine Languages, Assembly Languages and High-Level Languages

Programmers write instructions in various programming languages, some directly understandable by computers and others requiring intermediate *translation* steps. Hundreds of such languages are in use today. These may be divided into three general types:

1. Machine languages
2. Assembly languages
3. High-level languages

*Machine Languages*

Any computer can directly understand only its own **machine language**, defined by its hardware design. Machine languages generally consist of strings of numbers (ultimately reduced to 1s and 0s) that instruct computers to perform their most elementary operations one at a time. Machine languages are *machine dependent* (a particular machine language can be used on only one type of computer). Such languages are cumbersome for humans.

4. http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html
5. https://www.salesforce.com/blog/2015/10/salesforce-channel-ifttt.html
6. http://recode.net/2014/01/10/stuffed-why-data-storage-is-hot-again-really/

For example, here's a section of an early machine-language payroll program that adds overtime pay to base pay and stores the result in gross pay:

```
+1300042774
+1400593419
+1200274027
```

### *Assembly Languages and Assemblers*

Programming in machine language was simply too slow and tedious for most programmers. Instead of using the strings of numbers that computers could directly understand, programmers began using English-like abbreviations to represent elementary operations. These abbreviations formed the basis of **assembly languages**. *Translator programs* called **assemblers** were developed to convert early assembly-language programs to machine language at computer speeds. The following section of an assembly-language payroll program also adds overtime pay to base pay and stores the result in gross pay:

```
load    basepay
add     overpay
store   grosspay
```

Although such code is clearer to humans, it's incomprehensible to computers until translated to machine language.

### *High-Level Languages and Compilers*

With the advent of assembly languages, computer usage increased rapidly, but programmers still had to use numerous instructions to accomplish even the simplest tasks. To speed the programming process, **high-level languages** were developed in which single statements could be written to accomplish substantial tasks. Translator programs called **compilers** convert high-level language programs into machine language. High-level languages allow you to write instructions that look almost like everyday English and contain commonly used mathematical notations. A payroll program written in a high-level language might contain a *single* statement such as

```
grossPay = basePay + overTimePay
```

From the programmer's standpoint, high-level languages are preferable to machine and assembly languages. Java is the world's most widely used high-level programming language.

### *Interpreters*

Compiling a large high-level language program into machine language can take considerable computer time. *Interpreter* programs, developed to execute high-level language programs directly, avoid the delay of compilation, although they run slower than compiled programs. We'll say more about interpreters in Section 1.9, where you'll learn that Java uses a clever performance-tuned mixture of compilation and interpretation to run programs.

was formed to develop, maintain and evolve Android, driving innovation in mobile technology and improving the user experience while reducing costs. According to Statista.com, as of Q3 2016, Android had 87.8% of the global smartphone market share, compared to 11.5% for Apple.[7] The Android operating system is used in numerous smartphones, e-reader devices, tablets, in-store touch-screen kiosks, cars, robots, multimedia players and more.

We present an introduction to Android app development in our textbook, *Android How to Program, Third Edition,* and in our professional book, *Android 6 for Programmers: An App-Driven Approach, Third Edition.* After you learn Java, you'll find it straightforward to begin developing and running Android apps. You can place your apps on Google Play (`play.google.com`), and if they're successful, you may even be able to launch a business. Just remember—Facebook, Microsoft and Dell were all launched from college dorm rooms.

## 1.7 Programming Languages

Figure 1.6 provides brief comments on several popular programming languages. In the next section, we introduce Java.

| Programming language | Description |
|---|---|
| Ada | Ada, based on Pascal, was developed under the sponsorship of the U.S. Department of Defense (DOD) during the 1970s and early 1980s. The DOD wanted a single language that would fill most of its needs. The Pascal-based language was named after Lady Ada Lovelace, daughter of the poet Lord Byron. She's credited with writing the world's first computer program in the early 1800s (for the Analytical Engine mechanical computing device designed by Charles Babbage). Ada also supports object-oriented programming. |
| Basic | Basic was developed in the 1960s at Dartmouth College to familiarize novices with programming techniques. Many of its latest versions are object oriented. |
| C | C was developed in the early 1970s by Dennis Ritchie at Bell Laboratories. It initially became widely known as the UNIX operating system's development language. Today, most code for general-purpose operating systems is written in C or C++. |
| C++ | C++, which is based on C, was developed by Bjarne Stroustrup in the early 1980s at Bell Laboratories. C++ provides several features that "spruce up" the C language, but more important, it provides capabilities for object-oriented programming. |
| C# | Microsoft's three primary object-oriented programming languages are C# (based on C++ and Java), Visual C++ (based on C++) and Visual Basic (based on the original Basic). C# was developed to integrate the web into computer applications, and is now widely used to develop enterprise applications and for mobile application development. |

**Fig. 1.5** | Some other programming languages. (Part 1 of 3.)

7. `https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems`

| Programming language | Description |
|---|---|
| COBOL | COBOL (COmmon Business Oriented Language) was developed in the late 1950s by computer manufacturers, the U.S. government and industrial computer users, based on a language developed by Grace Hopper, a career U.S. Navy officer and computer scientist. (She was posthumously awarded the Presidential Medal of Freedom in November of 2016.) COBOL is still widely used for commercial applications that require precise and efficient manipulation of large amounts of data. Its latest version supports object-oriented programming. |
| Fortran | Fortran (FORmula TRANslator) was developed by IBM Corporation in the mid-1950s to be used for scientific and engineering applications that require complex mathematical computations. It's still widely used, and its latest versions support object-oriented programming. |
| JavaScript | JavaScript is the most widely used scripting language. It's primarily used to add programmability to web pages—for example, animations and interactivity with the user. All major web browsers support it. |
| Objective-C | Objective-C is an object-oriented language based on C. It was developed in the early 1980s and later acquired by NeXT, which in turn was acquired by Apple. It became the key programming language for the OS X operating system and all iOS-powered devices (such as iPods, iPhones and iPads). |
| Pascal | Research in the 1960s resulted in *structured programming*—a disciplined approach to writing programs that are clearer, easier to test and debug and easier to modify than programs produced with previous techniques. The Pascal language, developed by Professor Niklaus Wirth in 1971, grew out of this research. It was popular for teaching structured programming for several decades. |
| PHP | PHP is an object-oriented, *open-source* "scripting" language supported by a community of developers and used by numerous websites. PHP is platform independent—implementations exist for all major UNIX, Linux, Mac and Windows operating systems. |
| Python | Python, another object-oriented scripting language, was released publicly in 1991. Developed by Guido van Rossum of the National Research Institute for Mathematics and Computer Science in Amsterdam, Python draws heavily from Modula-3—a systems programming language. Python is "extensible"—it can be extended through classes and programming interfaces. |
| Ruby on Rails | Ruby—created in the mid-1990s by Yukihiro Matsumoto—is an open-source, object-oriented programming language with a simple syntax that's similar to Python. Ruby on Rails combines the scripting language Ruby with the Rails web-application framework developed by the company 37Signals. Their book, *Getting Real* (free at `http://gettingreal.37signals.com/toc.php`), is a must-read for web developers. Many Ruby on Rails developers have reported productivity gains over other languages when developing database-intensive web applications. |

**Fig. 1.5** | Some other programming languages. (Part 2 of 3.)

| Programming language | Description |
|---|---|
| Scala | Scala (`http://www.scala-lang.org/what-is-scala.html`)—short for "scalable language"—was designed by Martin Odersky, a professor at École Polytechnique Fédérale de Lausanne (EPFL) in Switzerland. Released in 2003, Scala uses both the object-oriented programming and functional programming paradigms and is designed to integrate with Java. Programming in Scala can reduce the amount of code in your applications significantly. |
| Swift | Swift, which was introduced in 2014, is Apple's programming language of the future for developing iOS and OS X applications (apps). Swift is a contemporary language that includes popular programming-language features from languages such as Objective-C, Java, C#, Ruby, Python and others. According to the Tiobe Index, Swift has already become one of the most popular programming languages. Swift is now *open source*, so it can be used on non-Apple platforms as well. |
| Visual Basic | Microsoft's Visual Basic language was introduced in the early 1990s to simplify the development of Microsoft Windows applications. Its features are comparable to those of C#. |

**Fig. 1.5** | Some other programming languages. (Part 3 of 3.)

## 1.8 Java

The microprocessor revolution's most important contribution to date is that it enabled the development of personal computers. Microprocessors also have had a profound impact in intelligent consumer-electronic devices, including the recent explosion in the "Internet of Things." Recognizing this early on, Sun Microsystems in 1991 funded an internal corporate research project led by James Gosling, which resulted in a C++-based object-oriented programming language that Sun called Java. Using Java, you can write programs that will run on a great variety of computer systems and computer-controlled devices. This is sometimes called "write once, run anywhere."

Java drew the attention of the business community because of the phenomenal interest in the Internet. It's now used to develop large-scale enterprise applications, to enhance the functionality of web servers (the computers that provide the content we see in our web browsers), to provide applications for consumer devices (cell phones, smartphones, television set-top boxes and more), to develop robotics software and for many other purposes. It's also the key language for developing Android smartphone and tablet apps. Sun Microsystems was acquired by Oracle in 2010.

Java has become the most widely used general-purpose programming language with more than 10 million developers. In this textbook, you'll learn the two most recent versions of Java—Java Standard Edition 8 (Java SE 8) and Java Standard Edition 9 (Java SE 9).

### *Java Class Libraries*

You can create each class and method you need to form your programs. However, most Java programmers take advantage of the rich collections of existing classes and methods in the **Java class libraries**, also known as the **Java APIs (Application Programming Interfaces)**.

**Performance Tip 1.1**

*Using Java API classes and methods instead of writing your own versions can improve program performance, because they're carefully written to perform efficiently. This also shortens program development time.*

## 1.9 A Typical Java Development Environment

We now explain the steps to create and execute a Java application. Normally there are five phases—edit, compile, load, verify and execute. We discuss them in the context of the Java SE 8 Development Kit (JDK). See the *Before You Begin* section *for information on downloading and installing the JDK on Windows, Linux and macOS.*

### *Phase 1: Creating a Program*

Phase 1 consists of editing a file with an *editor program*, normally known simply as an *editor* (Fig. 1.6). Using the editor, you type a Java program (typically referred to as **source code**), make any necessary corrections and save it on a secondary storage device, such as your hard drive. Java source code files are given a name ending with the **`.java` extension**, indicating that the file contains Java source code.

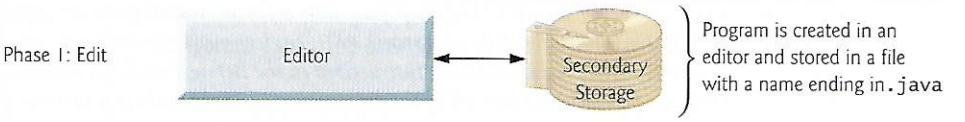


**Fig. 1.6** | Typical Java development environment—editing phase.

Two editors widely used on Linux systems are `vi` and `emacs` (). Windows provides **Notepad**. macOS provides **TextEdit**. Many freeware and shareware editors are also available online, including Notepad++ (`http://notepad-plus-plus.org`), EditPlus (`http://www.editplus.com`), TextPad (`http://www.textpad.com`), jEdit (`http://www.jedit.org`) and more.

**Integrated development environments (IDEs)** provide tools that support the software development process, such as editors, debuggers for locating **logic errors** that cause programs to execute incorrectly and more. The most popular Java IDEs are:

- Eclipse (`http://www.eclipse.org`)
- IntelliJ IDEA (`http://www.jetbrains.com`)
- NetBeans (`http://www.netbeans.org`)

On the book's website at

```
http://www.deitel.com/books/jhtp11
```

we provide videos that show you how to execute this book's Java applications and how to develop new Java applications with Eclipse, NetBeans and IntelliJ IDEA.

### *Phase 2: Compiling a Java Program into Bytecodes*

In Phase 2, you use the command **`javac`** (the **Java compiler**) to **compile** a program (Fig. 1.7). For example, to compile a program called `Welcome.java`, you'd type

```
javac Welcome.java
```

in your system's command window (i.e., the **Command Prompt** in Windows, the **Terminal** application in macOS) or a Linux shell (also called **Terminal** in some Linux versions). If the program compiles, the compiler produces a `.class` file called `Welcome.class`. IDEs typically provide a menu item, such as **Build** or **Make**, that invokes the `javac` command for you. If the compiler detects errors, you'll need to go back to Phase 1 and correct them. In Chapter 2, we'll say more about the kinds of errors the compiler can detect.

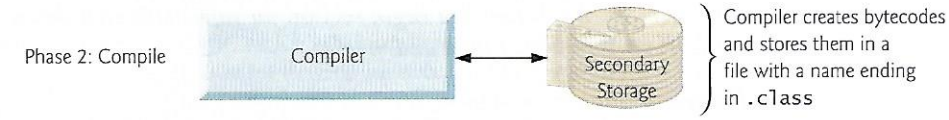


**Fig. 1.7** | Typical Java development environment—compilation phase.

### Common Programming Error 1.1

*When using `javac`, if you receive a message such as "`bad command or filename`," "`javac: command not found`" or "`'javac' is not recognized as an internal or external command, operable program or batch file`," then your Java software installation was not completed properly. This indicates that the system's `PATH` environment variable was not set properly. Carefully review the installation instructions in the Before You Begin section of this book. On some systems, after correcting the `PATH`, you may need to reboot your computer or open a new command window for these settings to take effect.*

The Java compiler translates Java source code into **bytecodes** that represent the tasks to execute in the execution phase (Phase 5). The **Java Virtual Machine** (**JVM**)—a part of the JDK and the foundation of the Java platform—executes bytecodes. A **virtual machine** (**VM**) is a software application that simulates a computer but hides the underlying operating system and hardware from the programs that interact with it. If the same VM is implemented on many computer platforms, applications written for that type of VM can be used on all those platforms. The JVM is one of the most widely used virtual machines. Microsoft's .NET uses a similar virtual-machine architecture.

Unlike machine-language instructions, which are *platform dependent* (that is, dependent on specific computer hardware), bytecode instructions are *platform independent*. So, Java's bytecodes are **portable**—without recompiling the source code, the same bytecode instructions can execute on any platform containing a JVM that understands the version of Java in which the bytecodes were compiled. The JVM is invoked by the **java** command. For example, to execute a Java application called `Welcome`, you'd type the command

```
java Welcome
```

in a command window to invoke the JVM, which would then initiate the steps necessary to execute the application. This begins Phase 3. IDEs typically provide a menu item, such as **Run**, that invokes the `java` command for you.

### *Phase 3: Loading a Program into Memory*

In Phase 3, the JVM places the program in memory to execute it—this is known as **loading** (Fig. 1.8).The JVM's **class loader** takes the `.class` files containing the program's byte-

codes and transfers them to primary memory. It also loads any of the `.class` files provided by Java that your program uses. The `.class` files can be loaded from a disk on your system or over a network (e.g., your local college or company network, or the Internet).

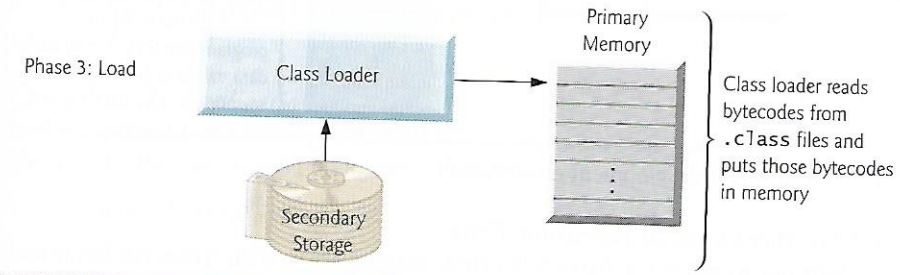


**Fig. 1.8** | Typical Java development environment—loading phase.

*Phase 4: Bytecode Verification*

In Phase 4, as the classes are loaded, the **bytecode verifier** examines their bytecodes to ensure that they're valid and do not violate Java's security restrictions (Fig. 1.9). Java enforces strong security to make sure that Java programs arriving over the network do not damage your files or your system (as computer viruses and worms might).

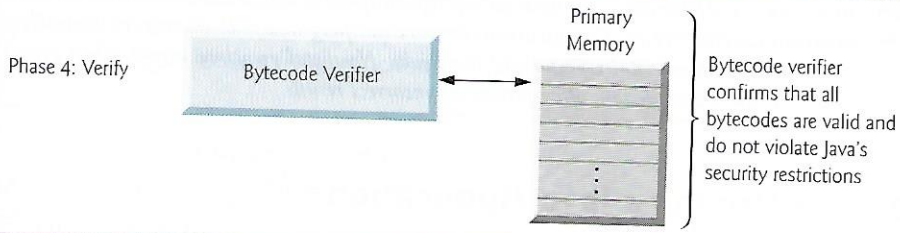


**Fig. 1.9** | Typical Java development environment—verification phase.

*Phase 5: Execution*

In Phase 5, the JVM **executes** the bytecodes to perform the program's specified actions (Fig. 1.10). In early Java versions, the JVM was simply a Java-bytecode *interpreter*. Most programs would execute slowly, because the JVM would interpret and execute one bytecode at a time. Some modern computer architectures can execute several instructions in parallel. Today's JVMs typically execute bytecodes using a combination of interpretation and **just-in-time (JIT) compilation**. In this process, the JVM analyzes the bytecodes as they're interpreted, searching for *hot spots*—bytecodes that execute frequently. For these parts, a **just-in-time (JIT) compiler**, such as Oracle's **Java HotSpot™ compiler**, translates the bytecodes into the computer's machine language. When the JVM encounters these compiled parts again, the faster machine-language code executes. Thus programs actually go through *two* compilation phases—one in which Java code is translated into bytecodes (for portability across JVMs on different computer platforms) and a second in which, during execution, the bytecodes are translated into *machine language* for the computer on which the program executes.

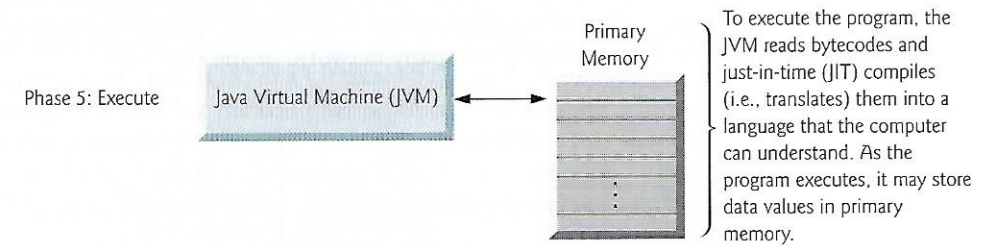


**Fig. 1.10** | Typical Java development environment—execution phase.

### *Problems That May Occur at Execution Time*

Programs might not work on the first try. Each of the preceding phases can fail because of various errors that we'll discuss throughout this book. For example, an executing program might try to divide by zero (an illegal operation for whole-number arithmetic in Java). This would cause the Java program to display an error message. If this occurred, you'd return to the edit phase, make the necessary corrections and proceed through the remaining phases again to determine whether the corrections fixed the problem(s). [*Note:* Most programs in Java input or output data. When we say that a program displays a message, we normally mean that it displays that message on your computer's screen.]

**Common Programming Error 1.2**

*Errors such as division by zero occur as a program runs, so they're called* ***runtime errors*** *or* ***execution-time errors****. Fatal runtime errors cause programs to terminate immediately without having successfully performed their jobs. Nonfatal runtime errors allow programs to run to completion, often producing incorrect results.*

## 1.11 Internet and World Wide Web

In the late 1960s, ARPA—the Advanced Research Projects Agency of the United States Department of Defense—rolled out plans for networking the main computer systems of approximately a dozen ARPA-funded universities and research institutions. The computers were to be connected with communications lines operating at speeds on the order of 50,000 bits per second, a stunning rate at a time when most people (of the few who even had networking access) were connecting over telephone lines to computers at a rate of 110 bits per second. Academic research was about to take a giant leap forward. ARPA proceeded to implement what quickly became known as the ARPANET, the precursor to today's **Internet**. Today's fastest Internet speeds are on the order of billions of bits per second with trillion-bits-per-second speeds on the horizon!

Things worked out differently from the original plan. Although the ARPANET enabled researchers to network their computers, its main benefit proved to be the capability for quick and easy communication via what came to be known as electronic mail (e-mail). This is true even on today's Internet, with e-mail, instant messaging, file transfer and social media such as Facebook and Twitter enabling billions of people worldwide to communicate quickly and easily.

The protocol (set of rules) for communicating over the ARPANET became known as the **Transmission Control Protocol (TCP)**. TCP ensured that messages, consisting of sequentially numbered pieces called *packets*, were properly routed from sender to receiver, arrived intact and were assembled in the correct order.

### 1.11.1 Internet: A Network of Networks

In parallel with the early evolution of the Internet, organizations worldwide were implementing their own networks for both intraorganization (that is, within an organization) and interorganization (that is, between organizations) communication. A huge variety of networking hardware and software appeared. One challenge was to enable these different networks to communicate with each other. ARPA accomplished this by developing the **Internet Protocol (IP)**, which created a true "network of networks," the current architecture of the Internet. The combined set of protocols is now called **TCP/IP**. Each Internet-connected device has an **IP address**—a unique numerical identifier used by devices communicating via TCP/IP to locate one another on the Internet.

Businesses rapidly realized that by using the Internet, they could improve their operations and offer new and better services to their clients. Companies started spending large amounts of money to develop and enhance their Internet presence. This generated fierce competition among communications carriers and hardware and software suppliers to meet the increased infrastructure demand. As a result, **bandwidth**—the information-carrying capacity of communications lines—on the Internet has increased tremendously, while hardware costs have plummeted.

### 1.11.2 World Wide Web: Making the Internet User-Friendly

The **World Wide Web** (simply called "the web") is a collection of hardware and software associated with the Internet that allows computer users to locate and view documents (with various combinations of text, graphics, animations, audios and videos) on almost any subject. In 1989, Tim Berners-Lee of CERN (the European Organization for Nuclear Research) began developing **HyperText Markup Language (HTML)**—the technology for sharing information via "hyperlinked" text documents. He also wrote communication protocols such as **HyperText Transfer Protocol (HTTP)** to form the backbone of his new hypertext information system, which he referred to as the World Wide Web.

In 1994, Berners-Lee founded the **World Wide Web Consortium** (W3C, `http://www.w3.org`), devoted to developing web technologies. One of the W3C's primary goals is to make the web universally accessible to everyone regardless of disabilities, language or culture.

### 1.11.3 Web Services and Mashups

In online Chapter 32, we implement web services (Fig. 1.15). The applications-development methodology of *mashups* enables you to rapidly develop powerful software applications by combining (often free) complementary web services and other forms of information feeds. One of the first mashups combined the real-estate listings provided by `http://www.craigslist.org` with the mapping capabilities of *Google Maps* to offer maps

that showed the locations of homes for sale or rent in a given area. ProgrammableWeb (`http://www.programmableweb.com/`) provides a directory of over 16,500 APIs and 6,300 mashups. Their API University (`https://www.programmableweb.com/api-university`) includes how-to guides and sample code for working with APIs and creating your own mashups. According to their website, some of the most widely used APIs are Facebook, Google Maps, Twitter and YouTube.

| Web services source | How it's used |
|---|---|
| Google Maps | Mapping services |
| Twitter | Microblogging |
| YouTube | Video search |
| Facebook | Social networking |
| Instagram | Photo sharing |
| Foursquare | Mobile check-in |
| LinkedIn | Social networking for business |
| Groupon | Social commerce |
| Netflix | Movie rentals |
| eBay | Internet auctions |
| Wikipedia | Collaborative encyclopedia |
| PayPal | Payments |
| Last.fm | Internet radio |
| Amazon eCommerce | Shopping for books and many other products |
| Salesforce.com | Customer Relationship Management (CRM) |
| Skype | Internet telephony |
| Microsoft Bing | Search |
| Flickr | Photo sharing |
| Zillow | Real-estate pricing |
| Yahoo Search | Search |
| WeatherBug | Weather |

**Fig. 1.15** | Some popular web services (`https://www.programmableweb.com/category/all/apis`).

### 1.11.4 Internet of Things

The Internet is no longer just a network of computers—it's an **Internet of Things (IoT)**. A *thing* is any object with an IP address and the ability to send data automatically over the Internet. Such things include:

- a car with a transponder for paying tolls,
- monitors for parking-space availability in a garage,

- a heart monitor implanted in a human,
- monitors for drinkable water quality,
- a smart meter that reports energy usage,
- radiation detectors,
- item trackers in a warehouse,
- mobile apps that can track your movement and location,
- smart thermostats that adjust room temperatures based on weather forecasts and activity in the home
- intelligent home appliances
- and many more.

According to `statista.com`, there are already over 22 billion IoT devices in use today and there are expected to be over 50 billion IoT devices in 2020.[8]

## 1.12 Software Technologies

Figure 1.16 lists a number of buzzwords that you'll hear in the software development community. We've created Resource Centers on most of these topics, with more on the way.

| Technology | Description |
|---|---|
| Agile software development | **Agile software development** is a set of methodologies that try to get software implemented faster and using fewer resources. Check out the Agile Alliance (`www.agilealliance.org`) and the Agile Manifesto (`www.agilemanifesto.org`). |
| Refactoring | **Refactoring** involves reworking programs to make them clearer and easier to maintain while preserving their correctness and functionality. It's widely employed with agile development methodologies. Many IDEs contain built-in *refactoring tools* to do major portions of the reworking automatically. |
| Design patterns | **Design patterns** are proven architectures for constructing flexible and maintainable object-oriented software. The field of design patterns tries to enumerate those recurring patterns, encouraging software designers to *reuse* them to develop better-quality software using less time, money and effort (see online Appendix N, Design Patterns). |

**Fig. 1.16** | Software technologies. (Part 1 of 2.)

8. `https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/`

| Technology | Description |
|---|---|
| LAMP | **LAMP** is an acronym for the open-source technologies that many developers use to build web applications inexpensively—it stands for *Linux*, *Apache*, *MySQL* and *PHP* (or *Perl* or *Python*—two other popular scripting languages). MySQL is an open-source database-management system. PHP is a popular open-source server-side "scripting" language for developing web applications. Apache is the most popular web server software. The equivalent for Windows development is WAMP—*Windows*, *Apache*, *MySQL* and *PHP*. |
| Software as a Service (SaaS) | Software has generally been viewed as a product; most software still is offered this way. If you want to run an application, you buy a software package from a software vendor—often a CD, DVD or web download. You then install that software on your computer and run it as needed. As new versions appear, you upgrade your software, often at considerable cost in time and money. This process can become cumbersome for organizations that must maintain tens of thousands of systems on a diverse array of computer equipment. With **Software as a Service (SaaS)**, the software runs on servers elsewhere on the Internet. When that server is updated, all clients worldwide see the new capabilities—no local installation is needed. You access the service through a browser. Browsers are quite portable, so you can run the same applications on a wide variety of computers from anywhere in the world. Salesforce.com, Google, Microsoft and many other companies offer SaaS. |
| Platform as a Service (PaaS) | **Platform as a Service (PaaS)** provides a computing platform for developing and running applications as a service over the web, rather than installing the tools on your computer. Some PaaS providers are Google App Engine, Amazon EC2 and Windows Azure™. |
| Cloud computing | SaaS and PaaS are examples of cloud computing. You can use software and data stored in the "cloud"—i.e., accessed on remote computers (or servers) via the Internet and available on demand—rather than having it stored locally on your desktop, notebook computer or mobile device. This allows you to increase or decrease computing resources to meet your needs at any given time, which is more cost effective than purchasing hardware to provide enough storage and processing power to meet occasional peak demands. Cloud computing also saves money by shifting to the service provider the burden of managing these apps (such as installing and upgrading the software, security, backups and disaster recovery). |
| Software Development Kit (SDK) | **Software Development Kits (SDKs)** include the tools and documentation developers use to program applications. |

**Fig. 1.16** | Software technologies. (Part 2 of 2.)

Software is complex. Large, real-world software applications can take many months or even years to design and implement. When large software products are under development, they typically are made available to the user communities as a series of releases, each more complete and polished than the last (Fig. 1.17).

| Version | Description |
|---|---|
| Alpha | *Alpha* software is the earliest release of a software product that's still under active development. Alpha versions are often buggy, incomplete and unstable and are released to a relatively small number of developers for testing new features, getting early feedback, etc. Alpha software also is commonly called *early access* software. |
| Beta | *Beta* versions are released to a larger number of developers later in the development process after most major bugs have been fixed and new features are nearly complete. Beta software is more stable, but still subject to change. |
| Release candidates | *Release candidates* are generally *feature complete*, (mostly) bug free and ready for use by the community, which provides a diverse testing environment—the software is used on different systems, with varying constraints and for a variety of purposes. |
| Final release | Any bugs that appear in the release candidate are corrected, and eventually the final product is released to the general public. Software companies often distribute incremental updates over the Internet. |
| Continuous beta | Software that's developed using this approach (for example, Google search or Gmail) generally does not have version numbers. It's hosted in the *cloud* (not installed on your computer) and is constantly evolving so that users always have the latest version. |

**Fig. 1.17** | Software product-release terminology.

## 1.13 Getting Your Questions Answered

There are many online forums in which you can get your Java questions answered and interact with other Java programmers. Some popular Java and general programming forums include:

- `StackOverflow.com`
- `Coderanch.com`
- The Oracle Java Forum—`https://community.oracle.com/community/java`
- `</dream.in.code>`—`http://www.dreamincode.net/forums/forum/32-java/`

## Self-Review Exercises

**1.1** Fill in the blanks in each of the following statements:

a) Computers process data under the control of sets of instructions called _____.

b) The key logical units of the computer are the _____, _____, _____, _____, _____ and _____.

c) The three types of languages discussed in the chapter are _____, _____ and _____.

d) The programs that translate high-level language programs into machine language are called _____.

e) _____ is an operating system for mobile devices based on the Linux kernel and Java.
f) _____ software is generally feature complete, (supposedly) bug free and ready for use by the community.
g) The Wii Remote, as well as many smartphones, use a(n) _____ which allows the device to respond to motion.

**1.2** Fill in the blanks in each of the following sentences about the Java environment:
a) The _____ command from the JDK executes a Java application.
b) The _____ command from the JDK compiles a Java program.
c) A Java source code file must end with the _____ file extension.
d) When a Java program is compiled, the file produced by the compiler ends with the _____ file extension.
e) The file produced by the Java compiler contains _____ that are executed by the Java Virtual Machine.

**1.3** Fill in the blanks in each of the following statements (based on Section 1.5):
a) Objects enable the design practice of _____—although they may know how to communicate with one another across well-defined interfaces, they normally are not allowed to know how other objects are implemented.
b) Java programmers concentrate on creating _____, which contain fields and the set of methods that manipulate those fields and provide services to clients.
c) The process of analyzing and designing a system from an object-oriented point of view is called _____.
d) A new class of objects can be created conveniently by _____—the new class (called the subclass) starts with the characteristics of an existing class (called the superclass), possibly customizing them and adding unique characteristics of its own.
e) _____ is a graphical language that allows people who design software systems to use an industry-standard notation to represent them.
f) The size, shape, color and weight of an object are considered _____ of the object's class.

## Answers to Self-Review Exercises

**1.1** a) programs. b) input unit, output unit, memory unit, central processing unit, arithmetic and logic unit, secondary storage unit. c) machine languages, assembly languages, high-level languages. d) compilers. e) Android. f) Release candidate. g) accelerometer.

**1.2** a) `java`. b) `javac`. c) `.java`. d) `.class`. e) bytecodes.

**1.3** a) information hiding. b) classes. c) object-oriented analysis and design (OOAD). d) Inheritance. e) The Unified Modeling Language (UML). f) attributes.

## Exercises

**1.4** Fill in the blanks in each of the following statements:
a) The logical unit that receives information from outside the computer for use by the computer is the _____.
b) The process of instructing the computer to solve a problem is called _____.
c) _____ is a type of computer language that uses Englishlike abbreviations for machine-language instructions.
d) _____ is a logical unit that sends information which has already been processed by the computer to various devices so that it may be used outside the computer.
e) _____ and _____ are logical units of the computer that retain information.

f) _____ is a logical unit of the computer that performs calculations.
g) _____ is a logical unit of the computer that makes logical decisions.
h) _____ languages are most convenient to the programmer for writing programs quickly and easily.
i) The only language a computer can directly understand is that computer's _____.
j) _____ is a logical unit of the computer that coordinates the activities of all the other logical units.

**1.5** Fill in the blanks in each of the following statements:
a) The _____ programming language is now used to develop large-scale enterprise applications, to enhance the functionality of web servers, to provide applications for consumer devices and for many other purposes.
b) _____ initially became widely known as the development language of the UNIX operating system.
c) The _____ ensures that messages, consisting of sequentially numbered pieces called bytes, were properly routed from sender to receiver, arrived intact and were assembled in the correct order.
d) The _____ programming language was developed by Bjarne Stroustrup in the early 1980s at Bell Laboratories.

**1.6** Fill in the blanks in each of the following statements:
a) Java programs normally go through five phases—_____, _____, _____, _____ and _____.
b) A(n) _____ provides many tools that support the software development process, such as editors for writing and editing programs, debuggers for locating logic errors in programs, and many other features.
c) The command `java` invokes the _____, which executes Java programs.
d) A(n) _____ is a software application that simulates a computer, but hides the underlying operating system and hardware from the programs that interact with it.
e) The _____ takes the `.class` files containing the program's bytecodes and transfers them to primary memory.
f) The _____ examines bytecodes to ensure that they're valid.

**1.7** Explain the two compilation phases of Java programs.

**1.8** One of the world's most common objects is a wrist watch. Discuss how each of the following terms and concepts applies to the notion of a watch: object, attributes, behaviors, class, inheritance (consider, for example, an alarm clock), modeling, messages, encapsulation, interface and information hiding.

## Making a Difference

The Making-a-Difference exercises will ask you to work on problems that really matter to individuals, communities, countries and the world.

**1.9** *(Test Drive: Carbon Footprint Calculator)* Some scientists believe that carbon emissions, especially from the burning of fossil fuels, contribute significantly to global warming and that this can be combated if individuals take steps to limit their use of carbon-based fuels. Various organizations and individuals are increasingly concerned about their "carbon footprints." Websites such as TerraPass

```
http://www.terrapass.com/carbon-footprint-calculator-2/
```

and Carbon Footprint

```
http://www.carbonfootprint.com/calculator.aspx
```

provide carbon-footprint calculators. Test drive these calculators to determine your carbon footprint. Exercises in later chapters will ask you to program your own carbon-footprint calculator. To prepare for this, research the formulas for calculating carbon footprints.

**1.10** *(Test Drive: Body-Mass-Index Calculator)* By recent estimates, two-thirds of the people in the United States are overweight and about half of those are obese. This causes significant increases in illnesses such as diabetes and heart disease. To determine whether a person is overweight or obese, you can use a measure called the body mass index (BMI). The United States Department of Health and Human Services provides a BMI calculator at `http://www.nhlbi.nih.gov/guidelines/obesity/BMI/bmicalc.htm`. Use it to calculate your own BMI. An exercise in Chapter 3 will ask you to program your own BMI calculator. To prepare for this, research the formulas for calculating BMI.

**1.11** *(Attributes of Hybrid Vehicles)* In this chapter you learned the basics of classes. Now you'll begin "fleshing out" aspects of a class called "Hybrid Vehicle." Hybrid vehicles are becoming increasingly popular, because they often get much better mileage than purely gasoline-powered vehicles. Browse the web and study the features of four or five of today's popular hybrid cars, then list as many of their hybrid-related attributes as you can. For example, common attributes include city-miles-per-gallon and highway-miles-per-gallon. Also list the attributes of the batteries (type, weight, etc.).

**1.12** *(Gender Neutrality)* Some people want to eliminate sexism in all forms of communication. You've been asked to create a program that can process a paragraph of text and replace gender-specific words with gender-neutral ones. Assuming that you've been given a list of gender-specific words and their gender-neutral replacements (e.g., replace "wife" with "spouse," "man" with "person," "daughter" with "child" and so on), explain the procedure you'd use to read through a paragraph of text and manually perform these replacements. How might your procedure generate a strange term like "woperchild?" In Chapter 4, you'll learn that a more formal term for "procedure" is "algorithm," and that an algorithm specifies the steps to be performed and the order in which to perform them.

**1.13** *(Intelligent Assistants)* Developments in the field of artificial intelligence have been accelerating in recent years. Many companies now offer computerized intelligent assistants, such as IBM's Watson, Amazon's Alexa, Apple's Siri, Google's Google Now and Microsoft's Cortana. Research these and others and list uses that can improve people's lives.

**1.14** *(Big Data)* Research the rapidly growing field of big data. List applications that hold great promise in fields such as healthcare and scientific research.

**1.15** *(Internet of Things)* It's now possible to have a microprocessor at the heart of just about any device and to connect those devices to the Internet. This has led to the notion of the Internet of Things (IoT), which already interconnects tens of billions of devices. Research the IoT and indicate the many ways it's improving people's lives.