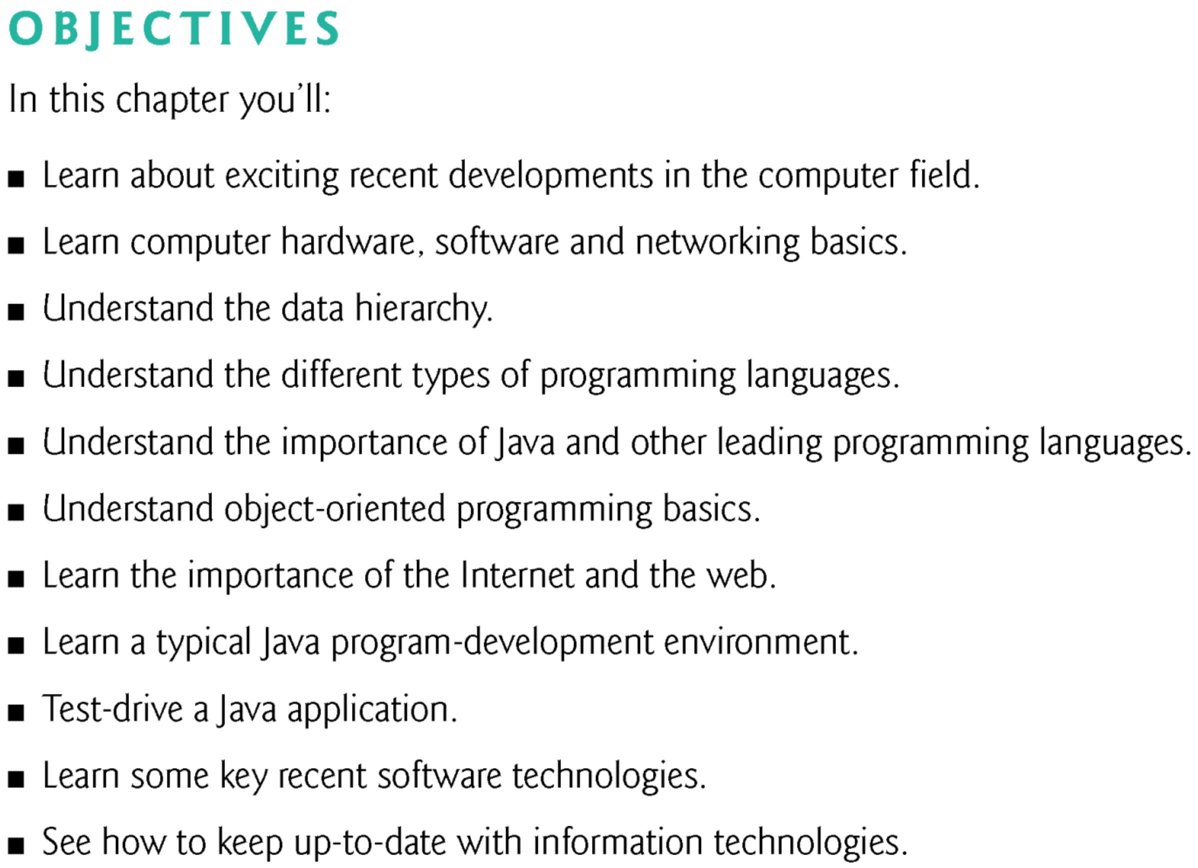
Chapter 1 Introduction to Computers, the Internet and Java

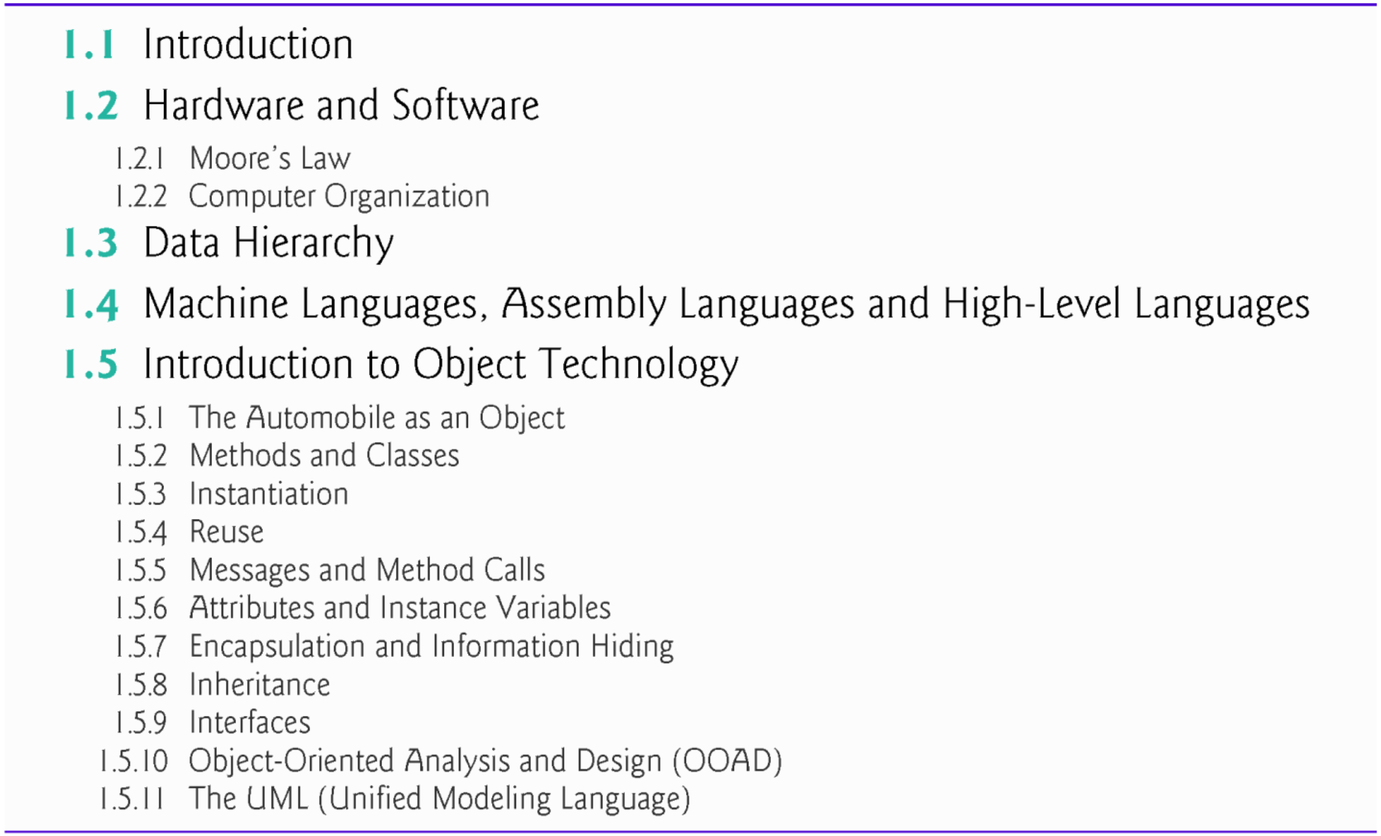
Java How to Program, 10/e



OBJECTIVES

In this chapter you'll:

- Learn about exciting recent developments in the computer field.
- Learn computer hardware, software and networking basics.
- Understand the data hierarchy.
- Understand the different types of programming languages.
- Understand the importance of Java and other leading programming languages.
- Understand object-oriented programming basics.
- Learn the importance of the Internet and the web.
- Learn a typical Java program-development environment.
- Test-drive a Java application.
- Learn some key recent software technologies.
- See how to keep up-to-date with information technologies.



1.1 Introduction

1.2 Hardware and Software

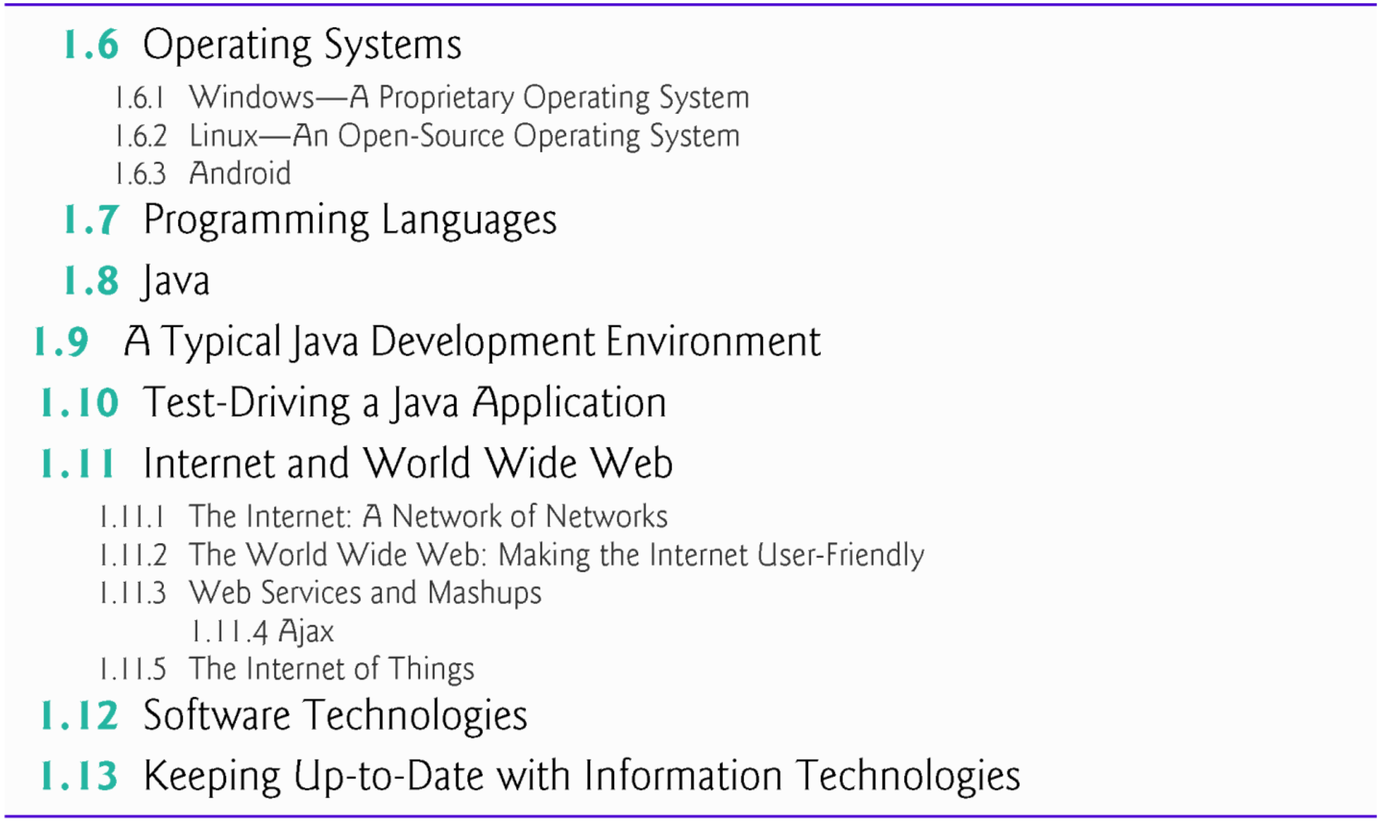
- 1.2.1 Moore's Law
- 1.2.2 Computer Organization

1.3 Data Hierarchy

1.4 Machine Languages, Assembly Languages and High-Level Languages

1.5 Introduction to Object Technology

- 1.5.1 The Automobile as an Object
- 1.5.2 Methods and Classes
- 1.5.3 Instantiation
- 1.5.4 Reuse
- 1.5.5 Messages and Method Calls
- 1.5.6 Attributes and Instance Variables
- 1.5.7 Encapsulation and Information Hiding
- 1.5.8 Inheritance
- 1.5.9 Interfaces
- 1.5.10 Object-Oriented Analysis and Design (OOAD)
- 1.5.11 The UML (Unified Modeling Language)



1.6 Operating Systems

1.6.1 Windows—A Proprietary Operating System
1.6.2 Linux—An Open-Source Operating System
1.6.3 Android

1.7 Programming Languages

1.8 Java

1.9 A Typical Java Development Environment

1.10 Test-Driving a Java Application

1.11 Internet and World Wide Web

1.11.1 The Internet: A Network of Networks
1.11.2 The World Wide Web: Making the Internet User-Friendly
1.11.3 Web Services and Mashups
1.11.4 Ajax
1.11.5 The Internet of Things

1.12 Software Technologies

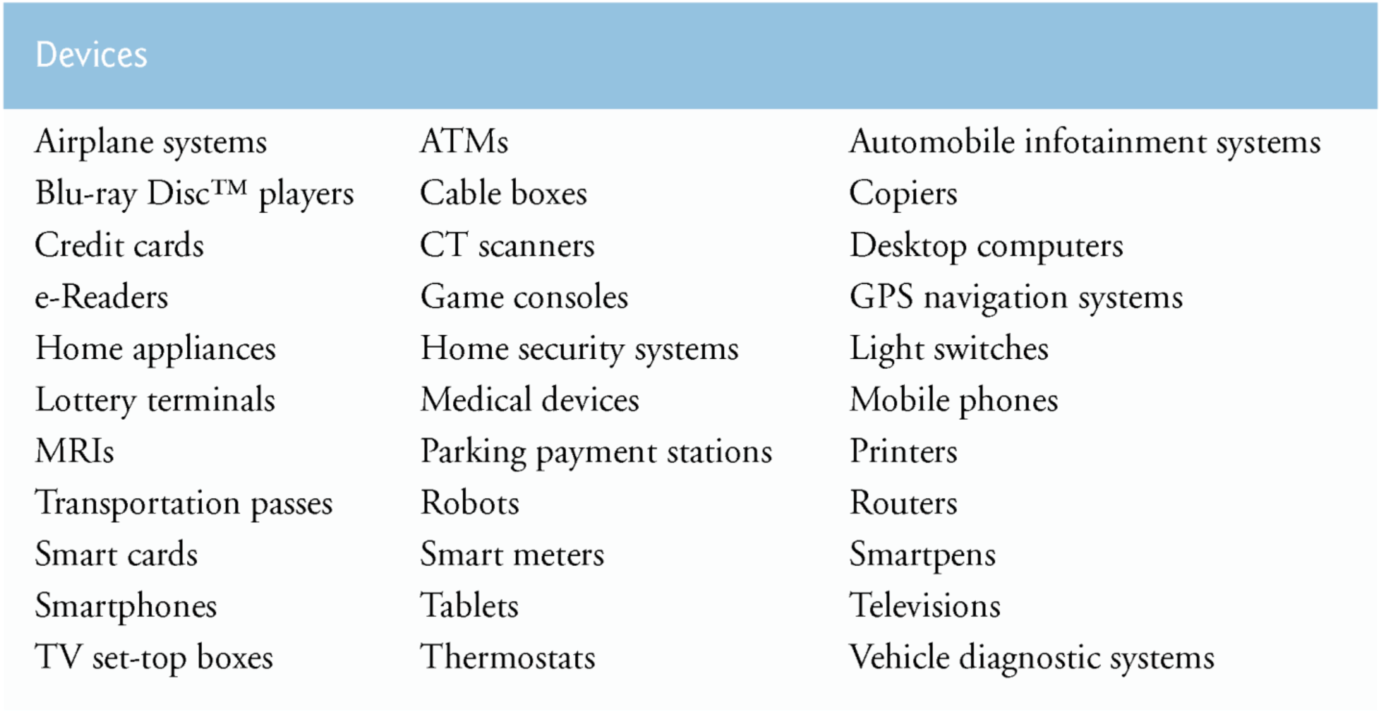
1.13 Keeping Up-to-Date with Information Technologies

1.1 Introduction

- Java is one of the world's most widely used computer programming languages.
- You'll learn to write instructions commanding computers to perform 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*.

1.1 Introduction (Cont.)

- 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.
- According to Oracle, 97% of enterprise desktops, 89% of PC desktops, three billion devices (Fig. 1.1) and 100% of all Blu-ray Disc™ players run Java, and there are over 9 million Java developers. (http://www.oracle.com/technetwork/articles/java/javaone12review-1863742.html)



Devices		
Airplane systems	ATMs	Automobile infotainment systems
Blu-ray Disc™ players	Cable boxes	Copiers
Credit cards	CT scanners	Desktop computers
e-Readers	Game consoles	GPS navigation systems
Home appliances	Home security systems	Light switches
Lottery terminals	Medical devices	Mobile phones
MRIs	Parking payment stations	Printers
Transportation passes	Robots	Routers
Smart cards	Smart meters	Smartpens
Smartphones	Tablets	Televisions
TV set-top boxes	Thermostats	Vehicle diagnostic systems

Fig. 1.1 | Some devices that use Java.

1.1 Introduction (Cont.)

Java Standard Edition

- Java How to Program, 10/e is based on Java Standard Edition 7 (Java SE 7) and Java Standard Edition 8 (Java SE 8)
- Java Standard Edition contains the capabilities needed to develop desktop and server applications.
- Prior to Java SE 8, Java supported three programming paradigms—procedural programming, object-oriented programming and generic programming. Java SE 8 adds functional programming.

1.1 Introduction (Cont.)

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.

1.1 Introduction (Cont.)

- Java Micro Edition (Java ME)
  - a subset of Java SE.
  - geared toward developing applications for resource-constrained embedded devices, such as
    - Smartwatches
    - MP3 players
    - television set-top boxes
    - smart meters (for monitoring electric energy usage)
    - and more.

1.2 Hardware and Software

- Computers can perform calculations and make logical decisions phenomenally faster than human beings *can*.
- 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!
- Computers process data under the control of sequences of instructions called computer programs.

1.2 Computers: Hardware and Software (Cont.)

- These software programs guide the computer through ordered actions specified by people called computer programmers.
- You’ll learn a key programming methodology that’s enhancing programmer productivity, thereby reducing software development costs—*object-oriented programming*.

1.2 Computers: Hardware and Software (Cont.)

- 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.
- Silicon-chip technology has made computing so economical that computers have become a commodity.

1.2 .1 Moore’s Law

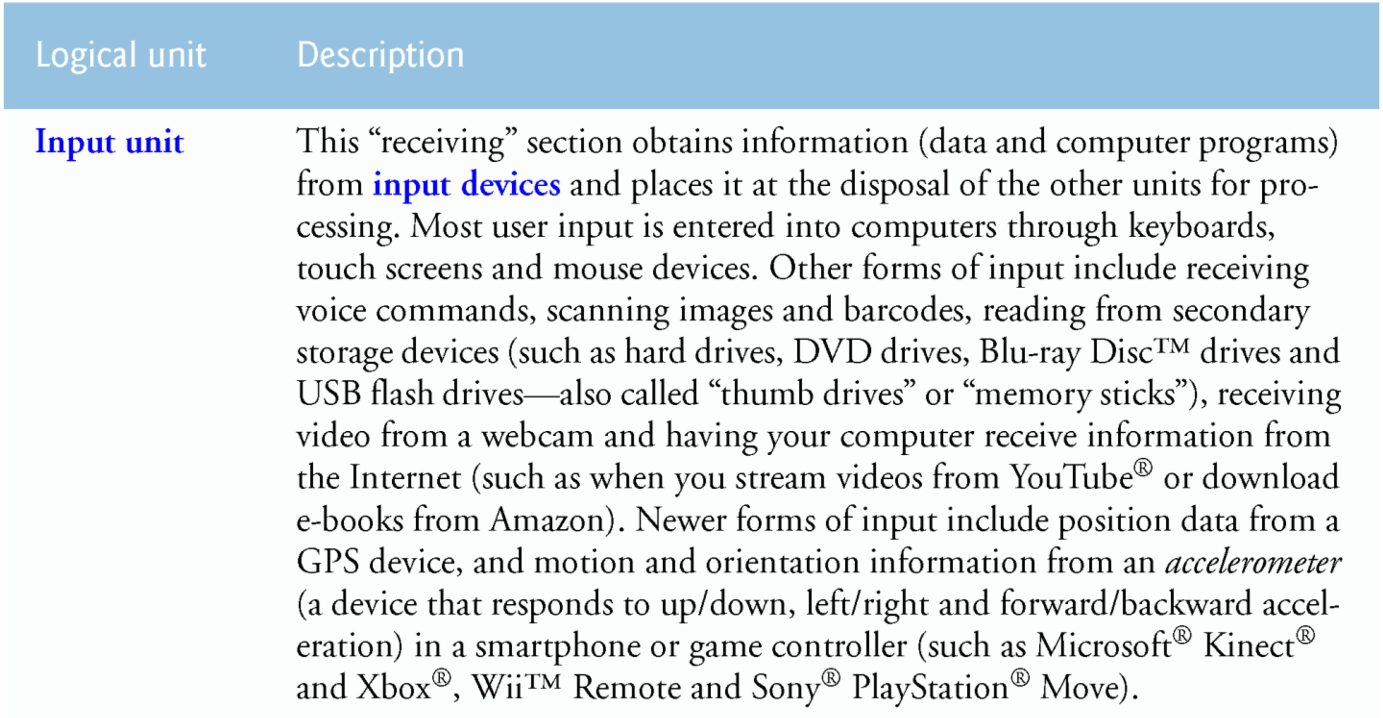
- Every year or two, the capacities of computers have approximately *doubled* inexpensively.
- This remarkable trend often is called Moore’s Law.
- Named for the person who identified the trend, Gordon Moore, co-founder of Intel.
- Moore’s Law and related observations apply especially to the amount of memory that computers have for programs, the amount of secondary storage (such as disk 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).

1.2 .1 Moore’s Law (Cont.)

- 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.
- Such phenomenal improvement is fostering the *Information Revolution.*

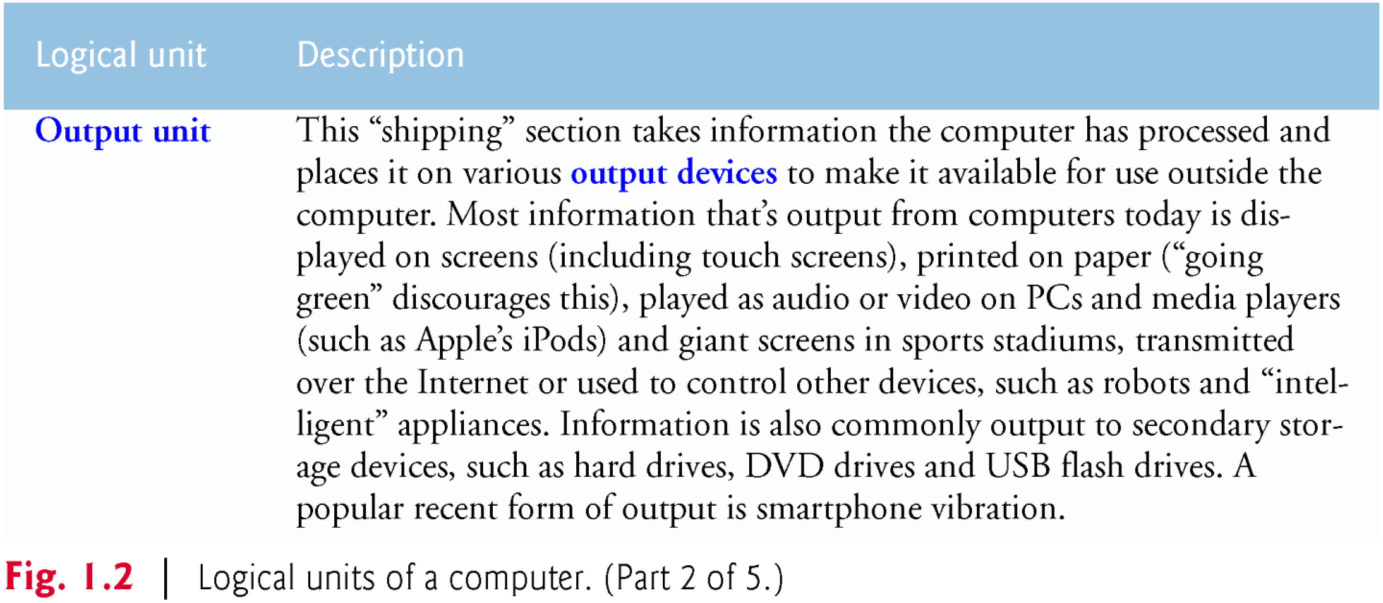
1.2.2 Computer Organization

- Computers can be envisioned as divided into various logical units or sections.



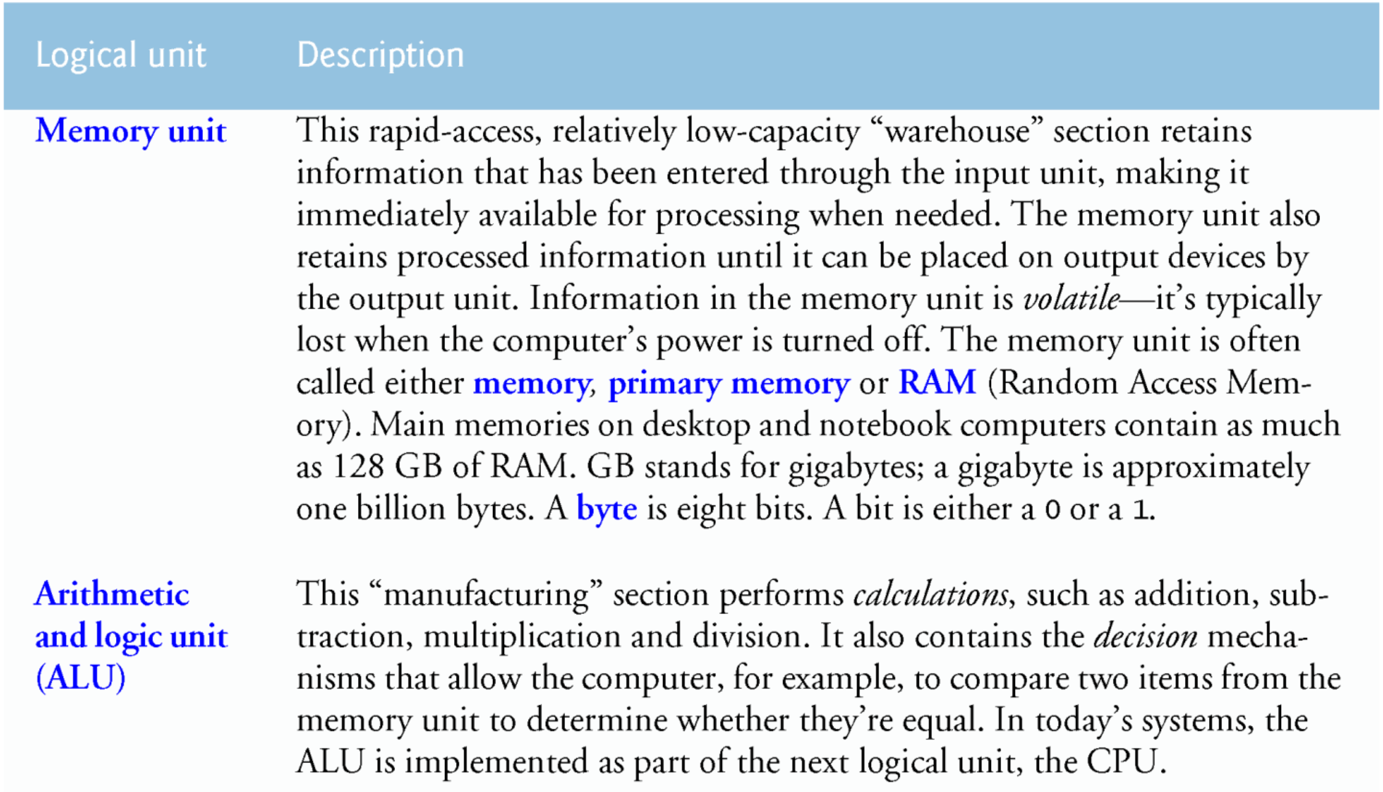
Logical unit	Description
Input unit	This “receiving” section obtains information (data and computer programs) from **input devices** and places it at the disposal of the other units for processing. Most user input is entered into computers through keyboards, touch screens and mouse devices. Other forms of input include receiving voice commands, scanning images and barcodes, 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, and 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® and Xbox®, Wii™ Remote and Sony® PlayStation® Move).

Fig. 1.2 | Logical units of a computer. (Part 1 of 5.)



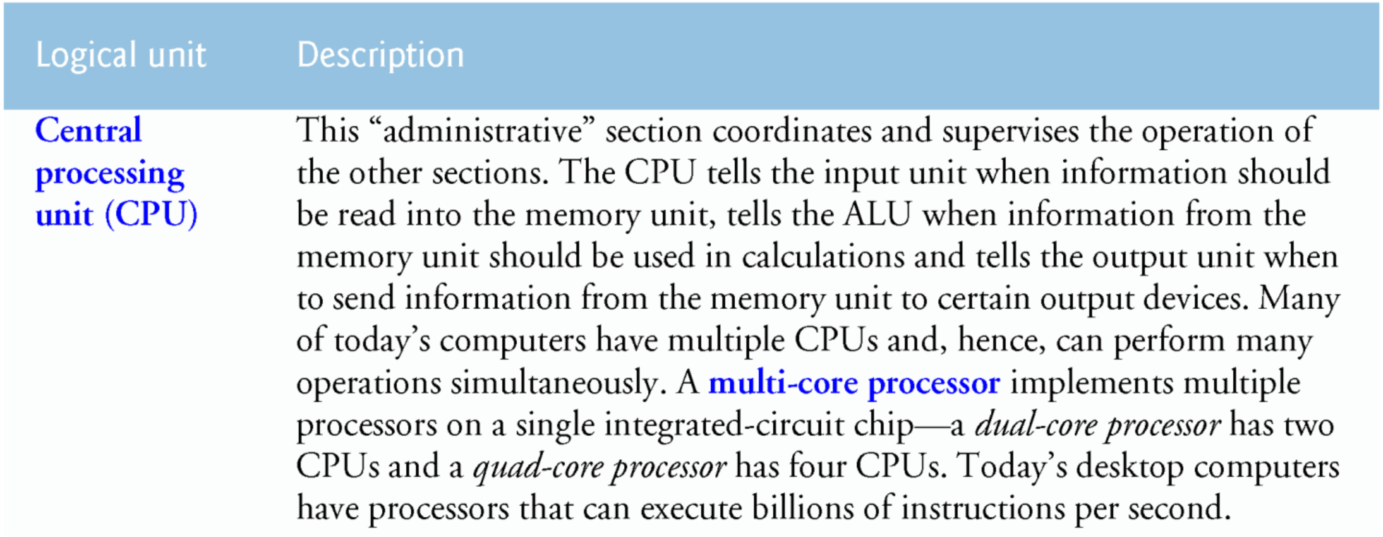
Logical unit	Description
Output unit	This “shipping” section takes information the computer has processed and places it on various **output devices** to make it available for use outside the computer. Most information that’s output from computers today is displayed on screens (including touch screens), 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 hard drives, DVD drives and USB flash drives. A popular recent form of output is smartphone vibration.

Fig. 1.2 | Logical units of a computer. (Part 2 of 5.)



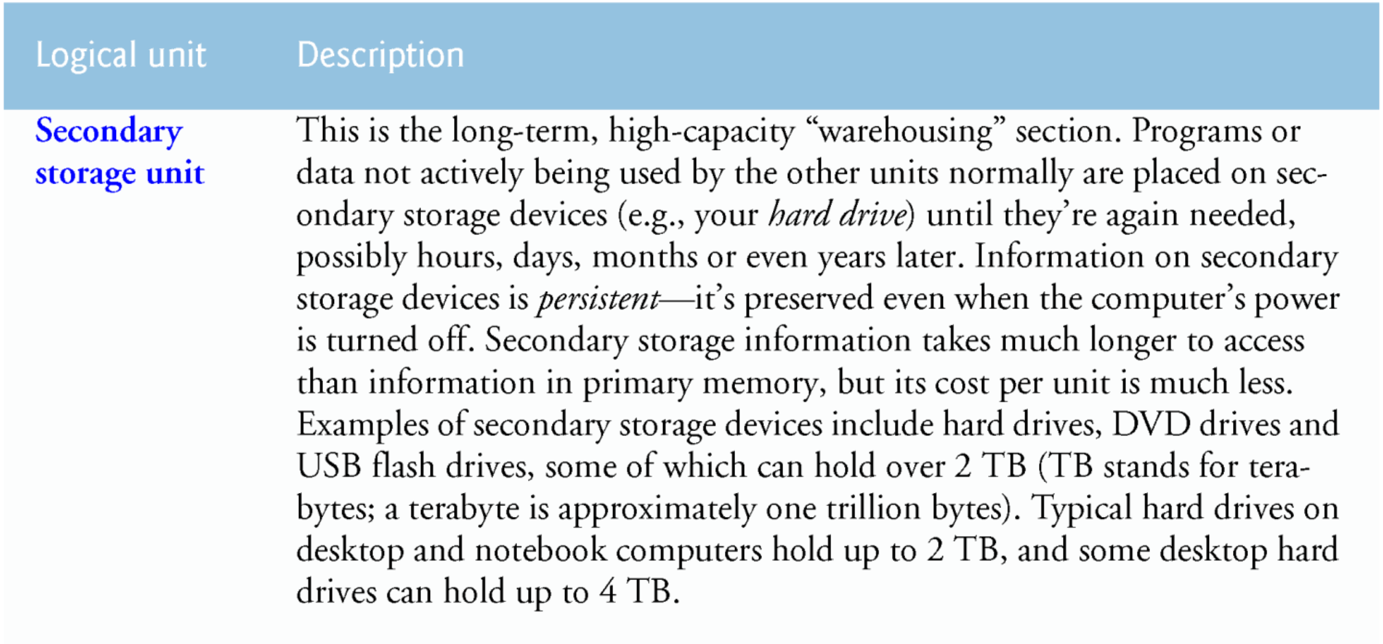
Logical unit	Description
Memory unit	This rapid-access, relatively low-capacity "warehouse" section retains information that has been entered through the input unit, making it immediately available for processing when needed. The memory unit also retains processed information until it can be placed on 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. 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.

Fig. 1.2 | Logical units of a computer. (Part 3 of 5.)



Logical unit	Description
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 **multi-core processor** implements multiple processors on a single integrated-circuit chip—a *dual-core processor* has two CPUs and a *quad-core processor* has four CPUs. Today's desktop computers have processors that can execute billions of instructions per second.

Fig. 1.2 | Logical units of a computer. (Part 4 of 5.)

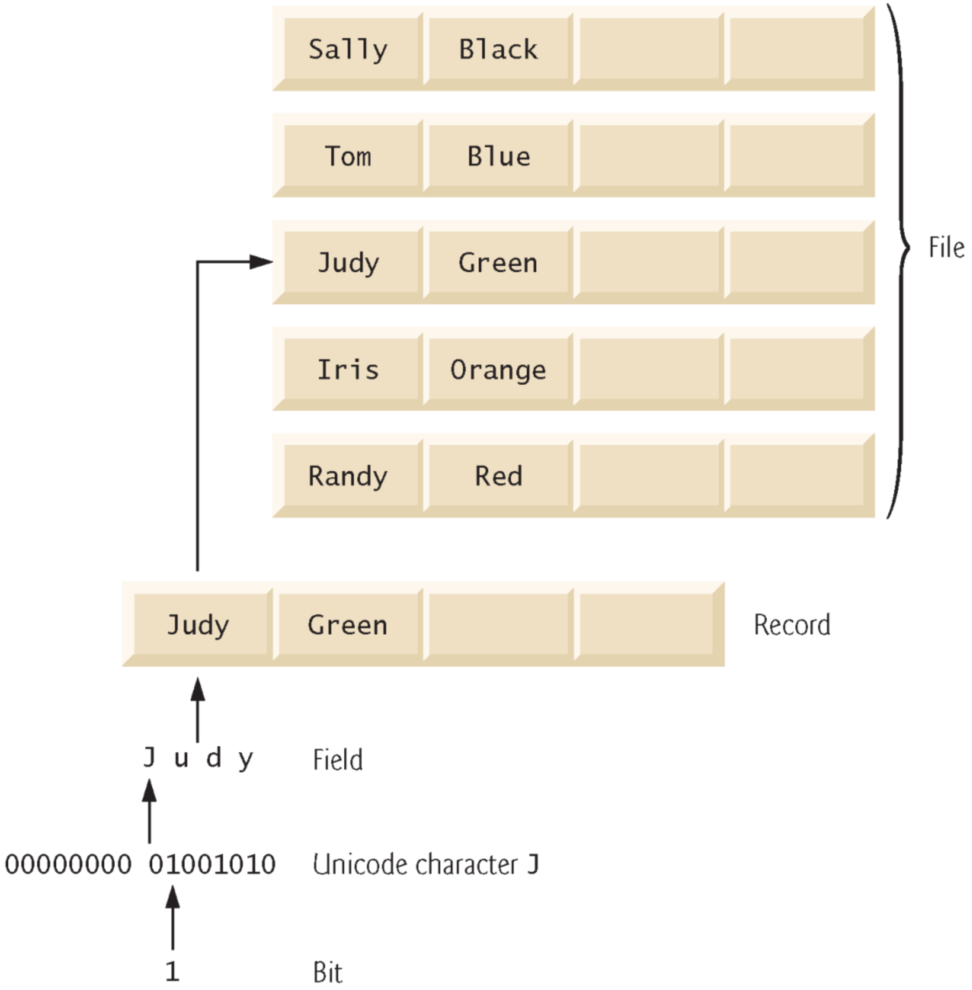


Logical unit	Description
Secondary storage unit	This is the long-term, high-capacity "warehousing" section. Programs or data not actively being used by the other units normally are placed on secondary storage devices (e.g., your *hard drive*) until they're again needed, possibly hours, days, months or 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 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 4 TB.

Fig. 1.2 | Logical units of a computer. (Part 5 of 5.)

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.


Sally
Black
Tom
Blue
Judy
Green
Iris
Orange
Randy
Red
File
Judy
Green
Record
J u d y
Field
00000000 01001010
Unicode character J
1
Bit

1.3 Data Hierarchy (Cont.)

Bits

- The smallest data item in a computer can assume the value 0 or the value 1.
- Such a data item is called a bit (short for “binary digit”—a digit that can assume either 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).

1.3 Data Hierarchy (Cont.)

Characters

- We prefer to work with *decimal digits* (0–9), *uppercase letters* (A–Z), *lowercase letters* (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 on that device.
- Computers process only 1s and 0s, so every character is represented 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).

1.3 Data Hierarchy (Cont.)

- Unicode contains characters for many of the world’s languages.
- 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 in the English alphabet, digits and some common special characters.

1.3 Data Hierarchy (Cont.)

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 could be used to represent a person’s name, and a field consisting of decimal digits could represent a person’s age.

1.3 Data Hierarchy (Cont.)

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.

1.3 Data Hierarchy (Cont.)

Files

- A file is a group of related records.
- 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.

1.3 Data Hierarchy (Cont.)

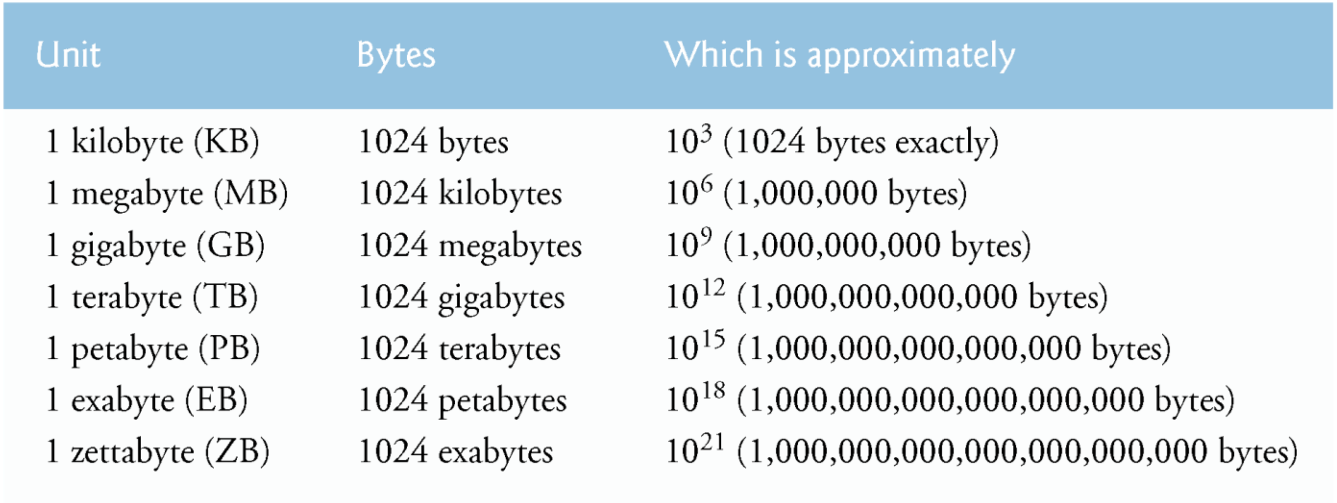
Database

- A database is a collection of data that’s organized for easy access and manipulation.
- The most popular database 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.

1.3 Data Hierarchy (Cont.)

Big Data

- The amount of data being produced worldwide is enormous and growing explosively.
- According to IBM, approximately 2.5 quintillion bytes (2.5 exabytes) of data are created daily and 90% of the world's data was created in just the past two years! (www-01.ibm.com/software/data/bigdata/)
- Big data applications deal with such massive amounts of data and this field is growing quickly, creating lots of opportunity for software developers.



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*.
- These may be divided into three general types:
  - Machine languages
  - Assembly languages
  - High-level languages

1.4 Machine Languages, Assembly Languages and High-Level Languages (Cont.)

Machine Languages

- Any computer can directly understand only its own machine language, *defined by its hardware design.*
  - 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 dependent—a particular ma-chine language can be used on only one type of computer.*

Assembly Languages and Assemblers

- English-like abbreviations that represent elementary operations formed the basis of assembly languages.
- *Translator programs* called assemblers convert early assembly-language programs to machine language.

1.4 Machine Languages, Assembly Languages and High-Level Languages (Cont.)

High-Level Languages and Compilers

- High-level languages
  - Single statements accomplish substantial tasks.
  - Compilers convert high-level language programs into machine language.
  - 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`

Interpreters

- Compiling a high-level language program into machine language can take considerable computer time.
- Interpreter programs, developer to execute high-level language programs directly, avoid the delay or compilation, although they run slower than compiled programs.

1.5 Introduction to Object Technology

- Objects, or more precisely, the *classes* objects come from, are essentially *reusable* software components.
  - There are date objects, time objects, audio objects, video objects, automobile objects, people objects, etc.
  - Almost any *noun* can be reasonably represented as a software object in terms of *attributes* (e.g., name, color and size) and *behaviors* (e.g., calculating, moving and communicating).
- Software development groups can use a modular, object-oriented design-and-implementation approach to be much more productive than with earlier popular techniques like "structured programming"—object-oriented programs are often easier to understand, correct and modify.

1.5.1 The Automobile as an Object

- The Automobile as an Object
  - Let’s begin with a simple analogy.
  - Suppose you want to *drive a car and make it go faster by pressing its accelerator pedal*.
  - Before you can drive a car, someone has to *design* it.
  - A car typically begins as engineering drawings, similar to the *blueprints* that describe the design of a house.
  - Drawings include the design for an accelerator pedal.
  - Pedal *hides* from the driver the complex mechanisms that actually make the car go faster, just as the brake pedal hides the mechanisms that slow the car, and the steering wheel “hides” the mechanisms that turn the car.

1.5.1 The Automobile as an Object (Cont.)

- Enables people with little or no knowledge of how engines, braking and steering mechanisms work to drive a car easily.
- Before you can drive a car, it must be *built* from the engineering drawings that describe it.
- A completed car has an *actual* accelerator pedal to make it go faster, but even that's not enough—the car won't accelerate on its own (hopefully!), so the driver must press the pedal to accelerate the car.

1.5.2 Methods and Classes

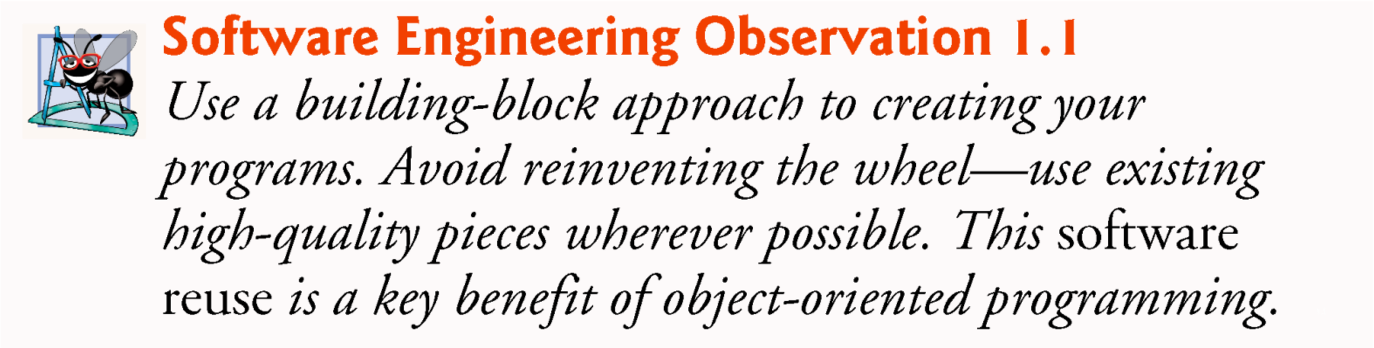
- Performing a task in a program requires a method.
- The method houses the program statements that actually perform its tasks.
- Hides these statements from its user, just as the accelerator pedal of a car hides from the driver the mechanisms of making the car go faster.
- In Java, we create a program unit called a class to house the set of methods that perform the class's tasks.
- A class is similar in concept to a car's engineering drawings, which house the design of an accelerator pedal, steering wheel, and so on.

1.5.3 Instantiation

- Just as someone has to *build* a car from its engineering drawings before you can actually drive a car, you must *build an object* of a class before a program can perform the tasks that the class’s methods define.
- An object is then referred to as an instance of its class.

1.5.4 Reuse

- Just as a car’s engineering drawings can be *reused* many times to build many cars, you can reuse a class many times to build many objects.
- Reuse of existing classes when building new classes and programs saves time and effort.
- Reuse also helps you build more reliable and effective systems, because existing classes and components often have undergone extensive *testing*, *debugging* and *performance* tuning.
- Just as the notion of *interchangeable parts* was crucial to the Industrial Revolution, reusable classes are crucial to the software revolution that has been spurred by object technology.



Software Engineering Observation 1.1

Use a building-block approach to creating your programs. Avoid reinventing the wheel—use existing high-quality pieces wherever possible. This software reuse *is a key benefit of object-oriented programming.*

1.5.5 Messages and Method Calls

- When you drive a car, pressing its gas pedal sends a *message* to the car to perform a task—that is, to go faster.
- Similarly, you *send messages to an object*.
- Each message is implemented as a method call that tells a method of the object to perform its task.

1.5.6 Attributes and Instance Variables

- A car has *attributes*
- Color, its number of doors, the amount of gas in its tank, its current speed and its record of total miles driven (i.e., its odometer reading).
- The car’s attributes are represented as part of its design in its engineering diagrams.
- Every car maintains its *own* attributes.
- Each car knows how much gas is in its own gas tank, but *not* how much is in the tanks of *other* cars.

1.5.6 Attributes and Instance Variables (Cont.)

- An object, has attributes that it carries along as it’s used in a program.
- Specified as part of the object’s class.
- A bank-account object has a *balance attribute* that represents the amount of money in the account.
- Each bank-account object knows the balance in the account it represents, but *not* the balances of the *other* accounts in the bank.
- Attributes are specified by the class’s instance variables.

1.5.7 Encapsulation

- Classes (and their objects) encapsulate, i.e., encase, their attributes and methods.
- Objects may communicate with one another, but they're normally not allowed to know how other objects are implemented—implementation details are *hidden* within the objects themselves.
- Information hiding, as we'll see, is crucial to good software engineering.

1.5.8 Inheritance

- A new class of objects can be created conveniently by inheritance—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.
- In our car analogy, an object of class “convertible” certainly *is an* object of the more *general* class “automobile,” but more *specifically*, the roof can be raised or lowered.

1.5.9 Interfaces

- Interfaces are collections of related methods that typically enable you to tell objects *what* to do, but not *how* to do it (we’ll see an exception to this in Java SE 8).
- In the car analogy, a “basic-driving-capabilities” interface consisting of a steering wheel, an accelerator pedal and a brake pedal would enable a driver to tell the car what to do.
- Once you know how to use this interface for turning, accelerating and braking, you can drive many types of cars, even though manufacturers may implement these systems differently.

1.5.9 Interfaces (Cont.)

- A class implements zero or more interfaces, each of which can have one or more methods, just as a car implements separate interfaces for basic driving functions, controlling the radio, controlling the heating and air conditioning systems, and the like.
- Just as car manufacturers implement capabilities *differently*, classes may implement an interface's methods *differently*.

1.5.10 Object-Oriented Analysis and Design (OOAD)

- How will you create the code (i.e., the program instructions) for your programs?
- Follow a detailed analysis process for determining your project's requirements (i.e., defining *what* the system is supposed to do)
- Develop a design that satisfies them (i.e., specifying *how* the system should do it).
- Carefully review the design (and have your design reviewed by other software professionals) before writing any code.

1.5.10 Object-Oriented Analysis and Design (OOAD) (Cont.)

- Analyzing and designing your system from an object-oriented point of view is called an object-oriented-analysis-and-design (OOAD) process.
- Languages like Java are object oriented.
- Object-oriented programming (OOP) allows you to implement an object-oriented design as a working system.

1.5.11 The UML (Unified Modeling Language)

- The Unified Modeling Language (UML) is the most widely used graphical scheme for modeling object-oriented systems.

1.6 Operating Systems

- Software systems that make using computers more convenient.
- Provide services that allow each application to execute safely, efficiently and *concurrently* (i.e., in parallel) with other applications.
- The software that contains the core components of the operating system is called the kernel.
- Popular desktop operating systems include Linux, Windows 7 and Mac OS X.
- Popular mobile operating systems used in smartphones and tablets include Google's Android, Apple's iOS (for its iPhone, iPad and iPod Touch devices), Windows Phone and Blackberry OS.

1.6.1 Windows—A Proprietary Operating System

- Mid-1980s: Microsoft developed the Windows operating system, consisting of a graphical user interface built on top of DOS—an enormously popular personal-computer operating system that users interacted with by typing commands.
- Windows borrowed many concepts (such as icons, menus and windows) popularized by early Apple Macintosh operating systems and originally developed by Xerox PARC.

1.6.1 Windows—A Proprietary Operating System (Cont.)

- Windows 8, Microsoft’s latest operating system, features include PC and tablet, a tiles-based user interface, security enhancements, touch-screen and multi-touch support, and more.
- Windows is a *proprietary* operating system—it’s controlled by Microsoft exclusively.
- It’s by far the world’s most widely used operating system.

1.6.2 Linux—An Open Source Operating System

- Open-source software
  - A software development style that departs from the *proprietary* development that dominated software's early years.
  - Individuals and companies—often worldwide—contribute their efforts in developing, maintaining and evolving software in exchange for the right to use that software for their own purposes, typically at no charge.

1.6.2 Linux—An Open Source Operating System (Cont.)

- Some organizations in the open-source community
  - *Eclipse Foundation* (the *Eclipse Integrated Development Environment* helps Java programmers conveniently develop software)
  - *Mozilla Foundation* (creators of the *Firefox web browser*)
  - *Apache Software Foundation* (creators of the *Apache web server*)
  - *GitHub* and *SourceForge* (which provide the tools for managing open source projects)

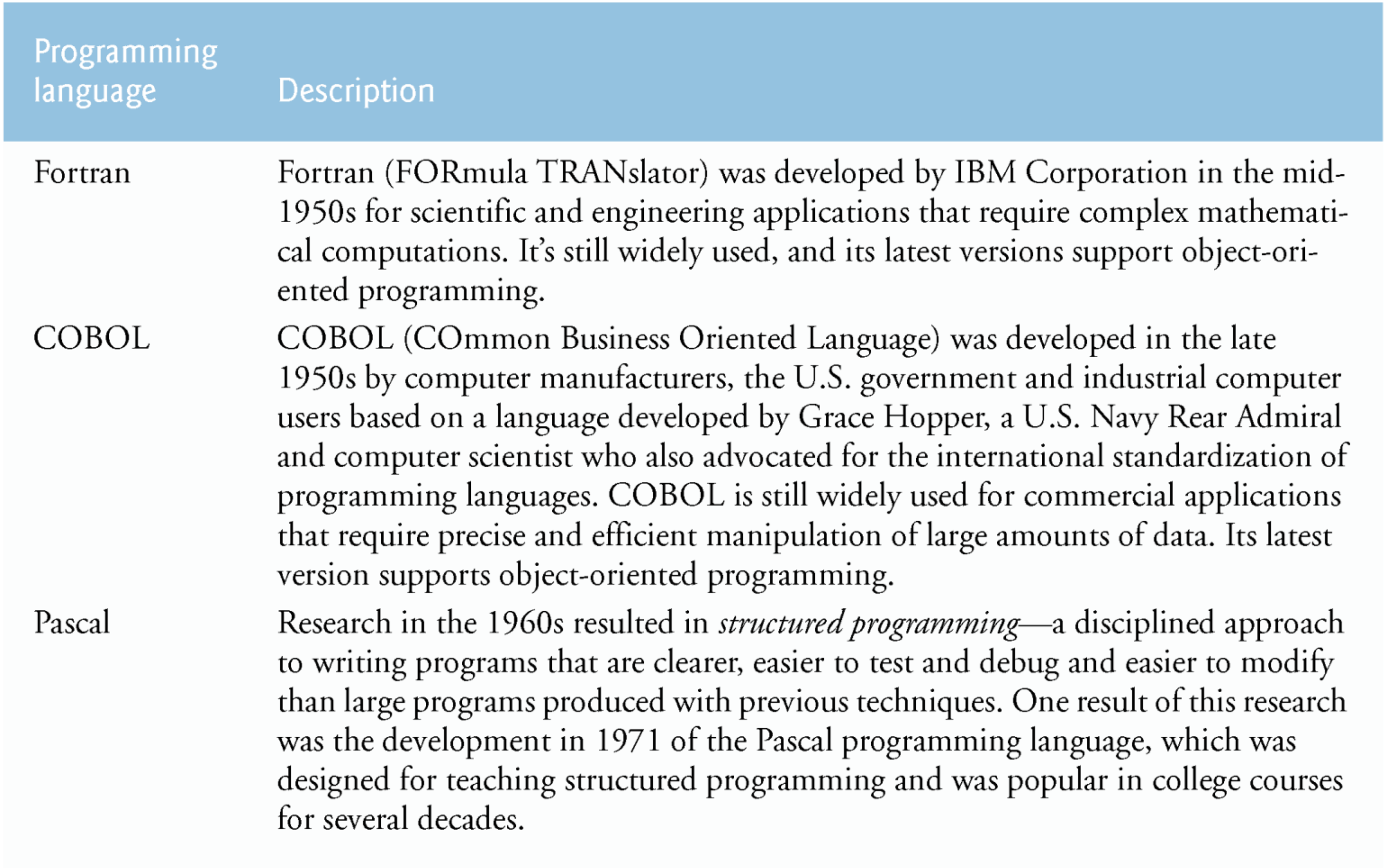
1.6.3 Android

- Fastest-growing mobile and smartphone operating system
- Based on the Linux kernel and uses Java.
- Open source and free.
- Developed by Android, Inc., which was acquired by Google in 2005.
- As of April 2013, more than 1.5 million Android devices (smartphones, tablets, etc.) were being activated daily. (www.technobuffalo.com/2013/04/16/google-daily-android-activations-1-5-million/)

1.6.3 Android (Cont.)

- Android devices now include smartphones, tablets, e-readers, robots, jet engines, NASA satellites, game consoles, refrigerators, televisions, cameras, health-care devices, smartwatches, automobile in-vehicle infotainment systems (for controlling the radio, GPS, phone calls, thermostat, etc.) and more.
- Android smartphones include the functionality of a mobile phone, Internet client (for web browsing and Internet communication), MP3 player, gaming console, digital camera and more.
- These handheld devices feature full-color *multitouch* screens—screens which allow you to control the device with gestures involving one touch or multiple simultaneous touches.

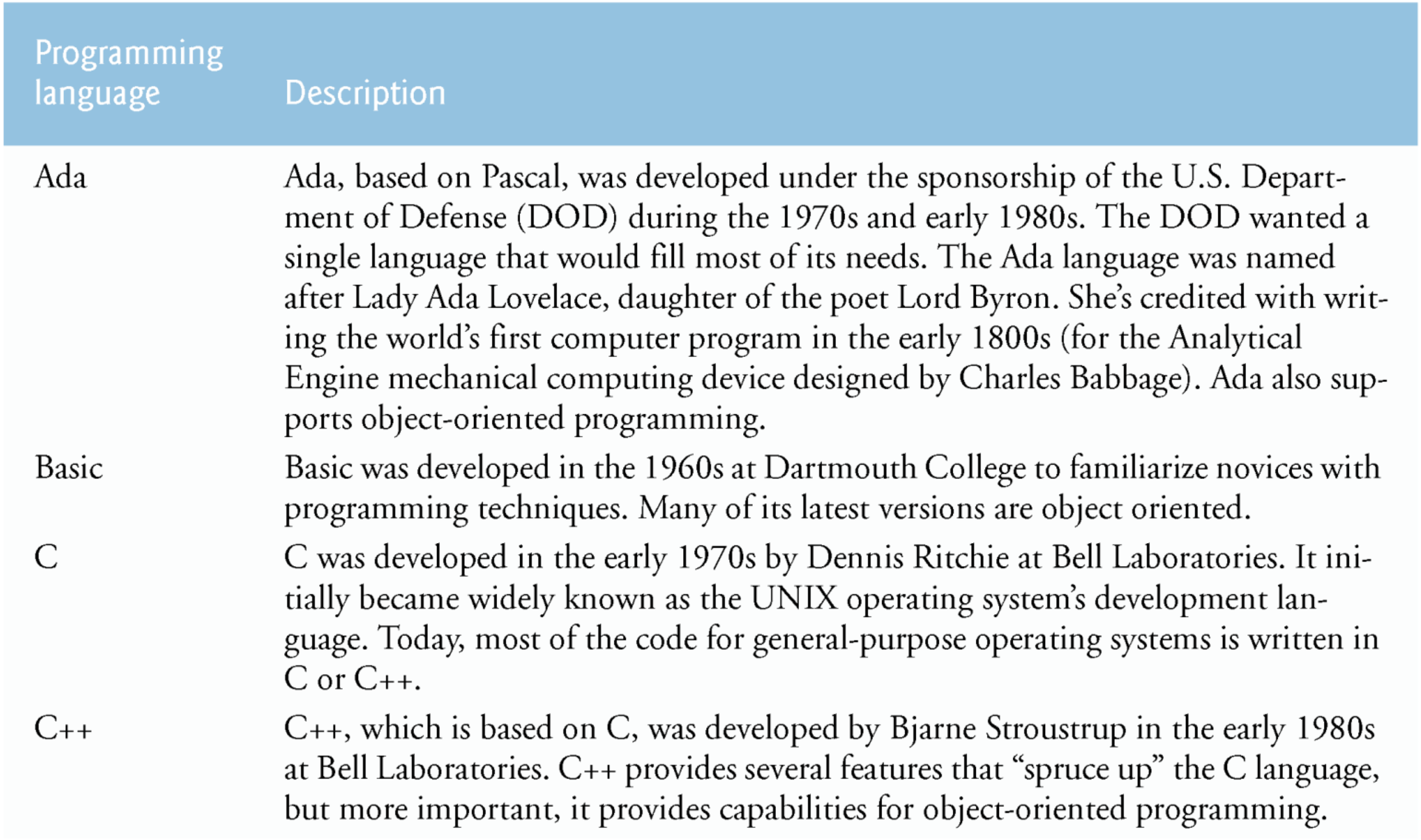
1.7 Programming Languages



Programming language	Description
Fortran	Fortran (FORmula TRANslator) was developed by IBM Corporation in the mid-1950s for scientific and engineering applications that require complex mathematical computations. It's still widely used, and its latest versions support object-oriented programming.
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 U.S. Navy Rear Admiral and computer scientist who also advocated for the international standardization of programming languages. 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.
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 large programs produced with previous techniques. One result of this research was the development in 1971 of the Pascal programming language, which was designed for teaching structured programming and was popular in college courses for several decades.

Fig. 1.5 | Some other programming languages. (Part 1 of 4.)

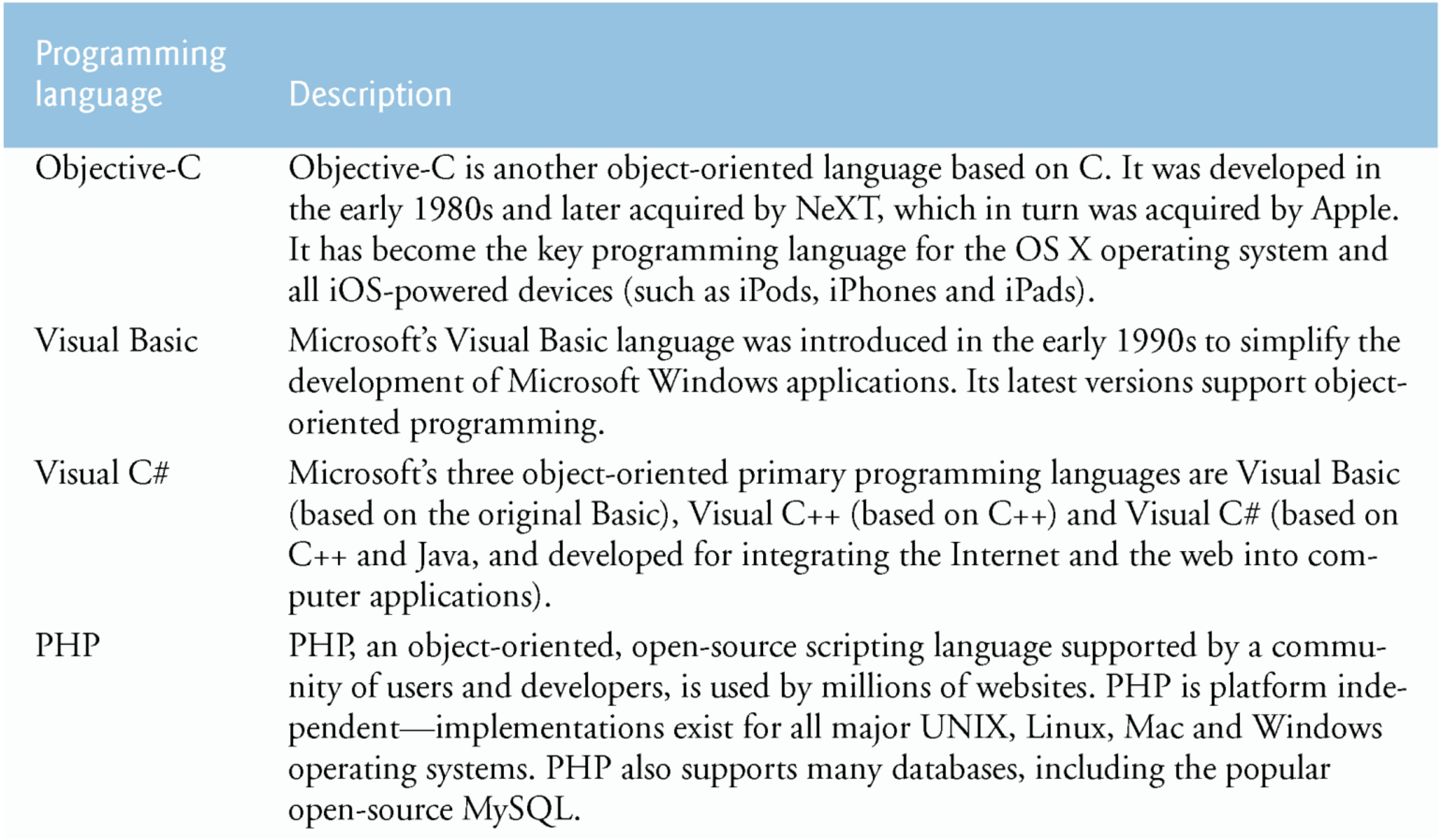
1.7 Programming Languages



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 Ada 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 of the 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.

Fig. 1.5 | Some other programming languages. (Part 2 of 4.)

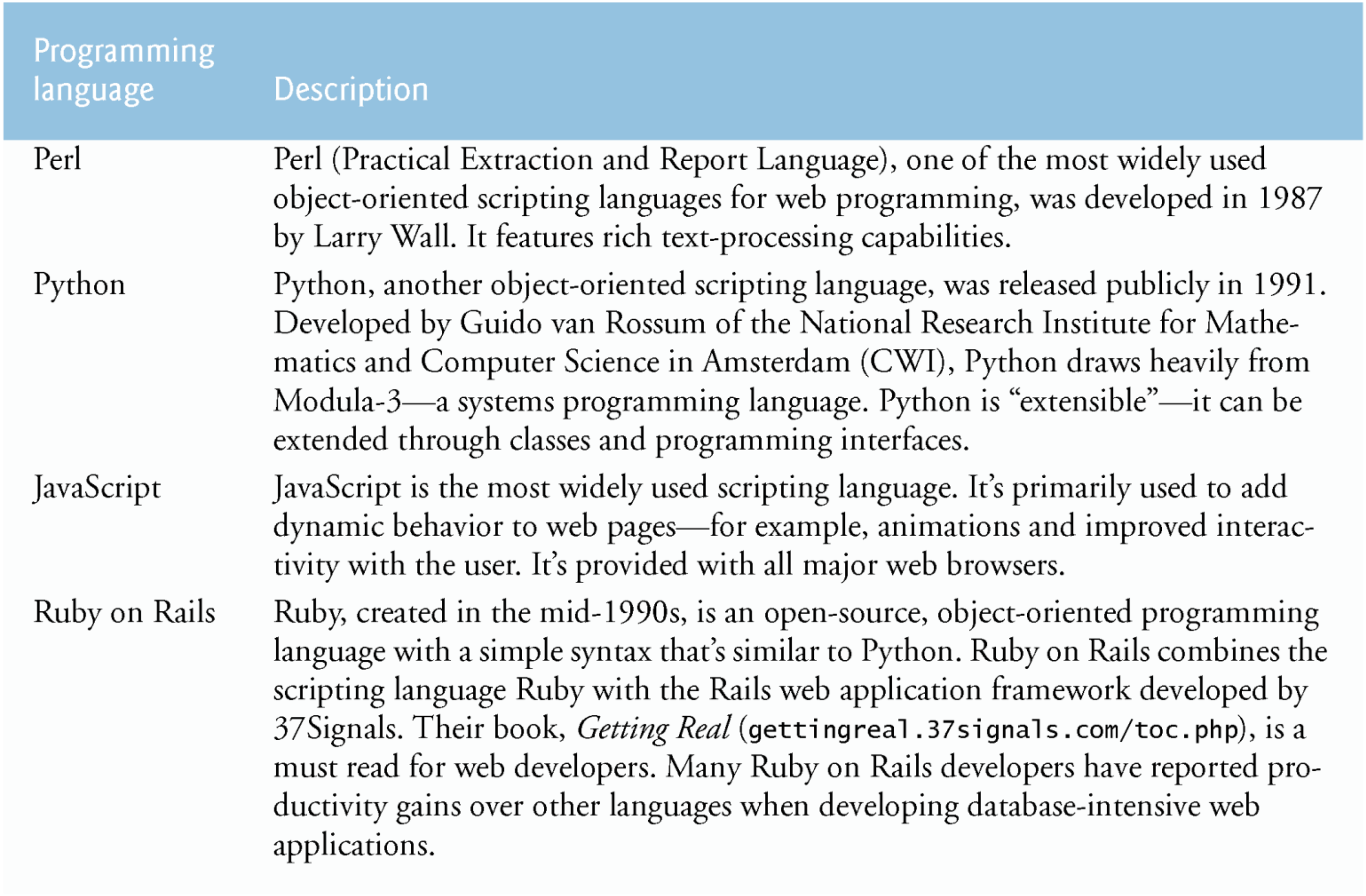
1.7 Programming Languages



Programming language	Description
Objective-C	Objective-C is another 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 has become the key programming language for the OS X operating system and all iOS-powered devices (such as iPods, iPhones and iPads).
Visual Basic	Microsoft's Visual Basic language was introduced in the early 1990s to simplify the development of Microsoft Windows applications. Its latest versions support object-oriented programming.
Visual C#	Microsoft's three object-oriented primary programming languages are Visual Basic (based on the original Basic), Visual C++ (based on C++) and Visual C# (based on C++ and Java, and developed for integrating the Internet and the web into computer applications).
PHP	PHP, an object-oriented, open-source scripting language supported by a community of users and developers, is used by millions of websites. PHP is platform independent—implementations exist for all major UNIX, Linux, Mac and Windows operating systems. PHP also supports many databases, including the popular open-source MySQL.

Fig. 1.5 | Some other programming languages. (Part 3 of 4.)

1.7 Programming Languages



Programming language	Description
Perl	Perl (Practical Extraction and Report Language), one of the most widely used object-oriented scripting languages for web programming, was developed in 1987 by Larry Wall. It features rich text-processing capabilities.
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 (CWI), Python draws heavily from Modula-3—a systems programming language. Python is "extensible"—it can be extended through classes and programming interfaces.
JavaScript	JavaScript is the most widely used scripting language. It's primarily used to add dynamic behavior to web pages—for example, animations and improved interactivity with the user. It's provided with all major web browsers.
Ruby on Rails	Ruby, created in the mid-1990s, 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 37Signals. Their book, *Getting Real* (`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 4 of 4.)

1.8 Java

- Microprocessors have had a profound impact in intelligent consumer-electronic devices.
- 1991
  - Recognizing this, Sun Microsystems funded an internal corporate research project led by James Gosling, which resulted in a C++-based object-oriented programming language that Sun called Java.
  - Key goal of Java is to be able to write programs that will run on a great variety of computer systems and computer-controlled devices.
  - This is sometimes called "write once, run anywhere."

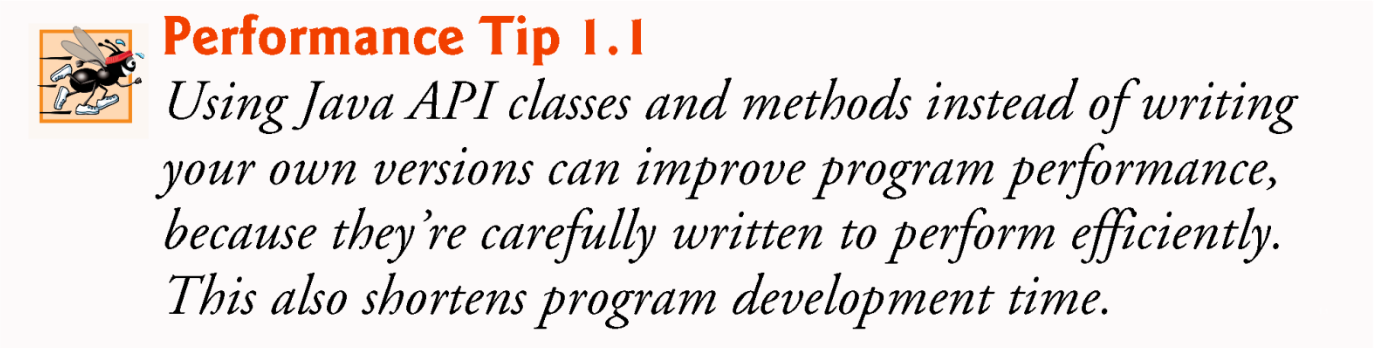
1.8 Java (Cont.)

- 1993
  - The web exploded in popularity
  - Sun saw the potential of using Java to add *dynamic content* to web pages.
- Java drew the attention of the business community because of the phenomenal interest in the web.
- Java is 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.

1.8 Java (Cont.)

Java Class Libraries

- Rich collections of existing classes and methods
- 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

- Normally there are five phases
  - edit
  - compile
  - load
  - verify
  - execute.
- See the Before You Begin section for information on downloading and installing the JDK on Windows, Linux and OS X.

1.9 A Typical Java Development Environment (Cont.)

- Phase 1 consists of editing a file with an *editor program*
  - Using the editor, you type a Java program (source code).
  - Make any necessary corrections.
  - Save the program.
  - Java source code files are given a name ending with the .java extension indicating that the file contains Java source code.

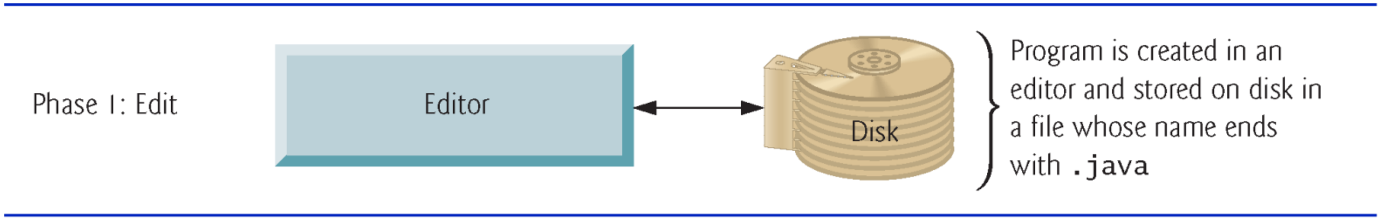

Phase 1: Edit
Editor
Disk
Program is created in an editor and stored on disk in a file whose name ends with .java

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

1.9 A Typical Java Development Environment (Cont.)

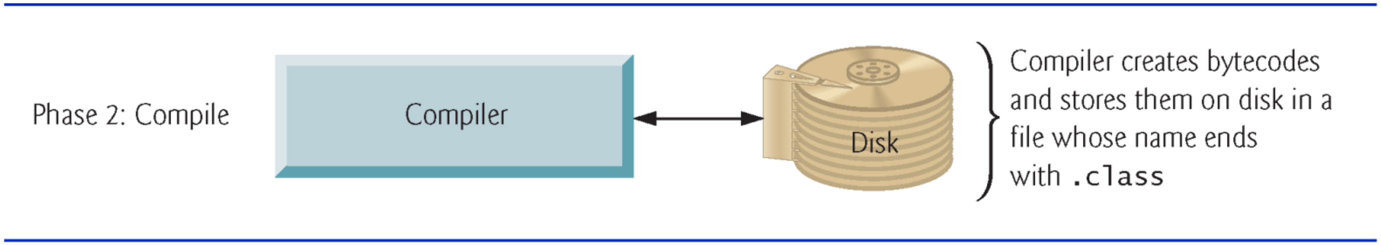
- Linux editors: `vi` and `emacs`.
- Windows provides Notepad.
- OSX provides TextEdit.
- Many freeware and shareware editors available online:
  - Notepad++ (`notepad-plus-plus.org`)
  - EditPlus (`www.editplus.com`)
  - TextPad (`www.textpad.com`)
  - jEdit (`www.jedit.org`).
- Integrated development environments (IDEs)
  - Provide tools that support the software development process, such as editors, debuggers for locating logic errors (errors that cause programs to execute incorrectly) and more.

1.9 Java and a Typical Java Development Environment (Cont.)

- Popular Java IDEs
  - Eclipse (`www.eclipse.org`)
  - NetBeans (`www.netbeans.org`)
  - IntelliJ IDEA (`www.jetbrains.com`)
- On the book's website at `www.deitel.com/books/jhtp10`
  - Dive-Into® 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.

1.9 A Typical Java Development Environment (Cont.)

- Phase 2: Compiling a Java Program into Bytecodes
  - Use the command javac (the Java compiler) to compile a program. For example, to compile a program called `Welcome.java`, you'd type
    - `javac Welcome.java`
  - If the program compiles, the compiler produces a .class file called `Welcome.class` that contains the compiled version.



Compiler creates bytecodes and stores them on disk in a file whose name ends with `.class`

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

1.9 A Typical Java Development Environment (Cont.)

- Java compiler translates Java source code into bytecodes that represent the tasks to execute.
- The Java Virtual Machine (JVM)—a part of the JDK and the foundation of the Java platform—executes bytecodes.
- Virtual machine (VM)—a software application that simulates a computer
  - 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.

1.9 A Typical Java Development Environment (Cont.)

- Bytecode instructions are *platform independent*
- Bytecodes are portable
  - The same bytecode instructions can execute on any platform containing a JVM that understands the version of Java in which the bytecode instructions 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`

1.9 A Typical Java Development Environment (Cont.)

- Phase 3: Loading a Program into Memory
  - The JVM places the program in memory to execute it—this is known as loading.
  - Class loader takes the `.class` files containing the program's bytecodes and transfers them to primary memory.
  - 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.

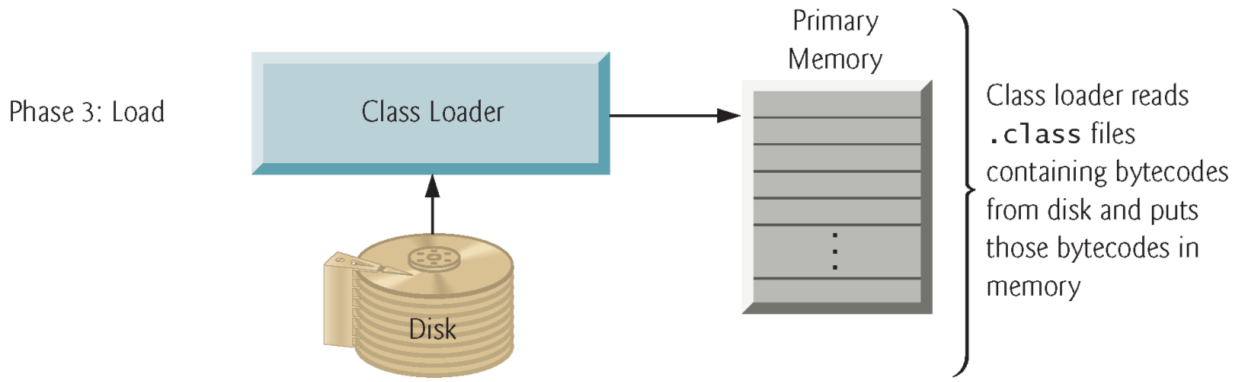

Phase 3: Load
Class Loader
Disk
Primary Memory
Class loader reads .class files containing bytecodes from disk and puts those bytecodes in memory

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

1.9 A Typical Java Development Environment (Cont.)

- Phase 4: Bytecode Verification
  - As the classes are loaded, the bytecode verifier examines their bytecodes
  - Ensures that they’re valid and do not violate Java’s security restrictions.
- 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).

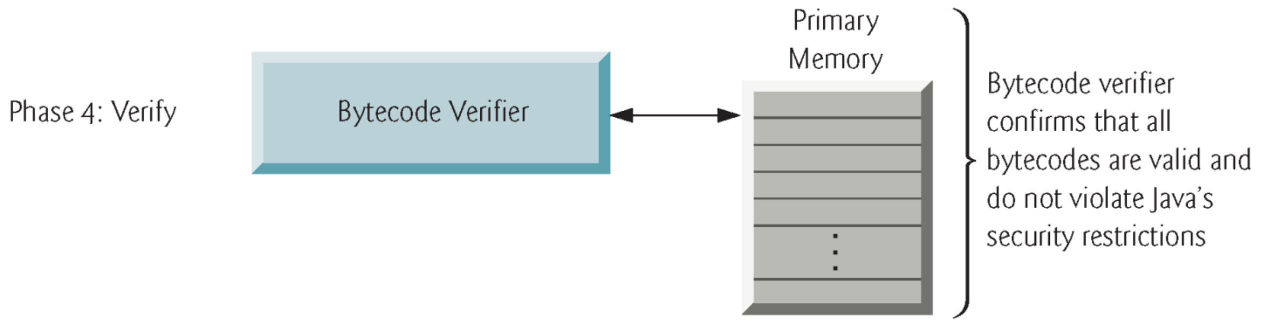

Phase 4: Verify
Bytecode Verifier
Primary Memory
Bytecode verifier confirms that all bytecodes are valid and do not violate Java's security restrictions

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

1.9 A Typical Java Development Environment (Cont.)

- Phase 5: Execution
  - The JVM executes the program’s bytecodes.
  - JVMs typically execute bytecodes using a combination of interpretation and so-called just-in-time (JIT) compilation.
  - Analyzes the bytecodes as they’re interpreted
  - A just-in-time (JIT) compiler—such as Oracle’s Java HotSpot™ compiler—translates the bytecodes into the underlying computer’s machine language.

1.9 A Typical Java Development Environment (Cont.)

- When the JVM encounters these compiled parts again, the faster machine-language code executes.
- Java programs go through *two* compilation phases
- One in which source 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 actual computer on which the program executes.

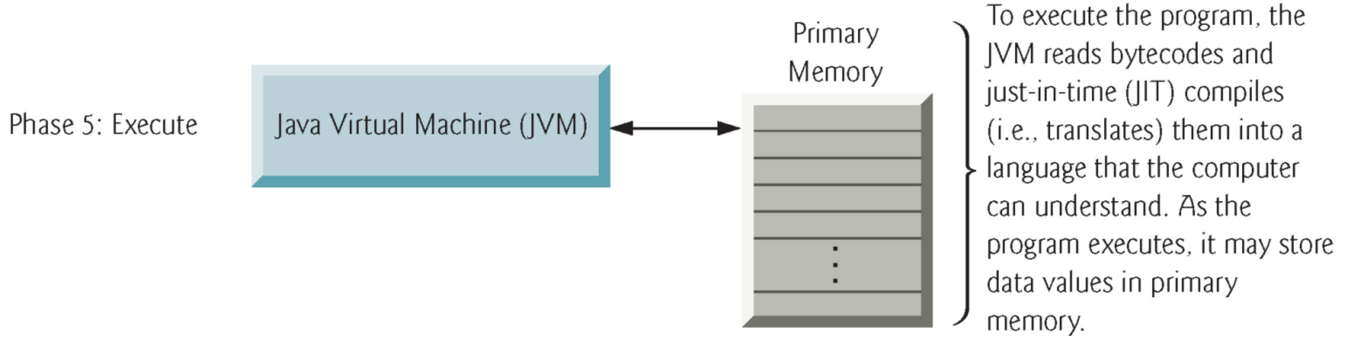
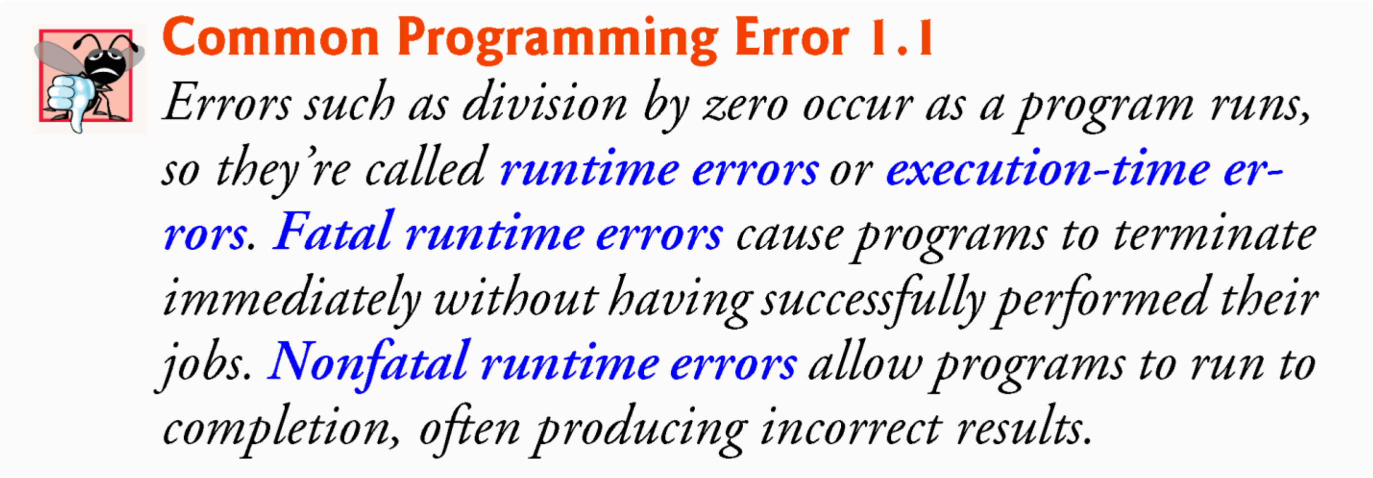

Phase 5: Execute
Java Virtual Machine (JVM)
Primary Memory
To execute the program, the JVM reads bytecodes and just-in-time (JIT) compiles (i.e., translates) them into a language that the computer can understand. As the program executes, it may store data values in primary memory.

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

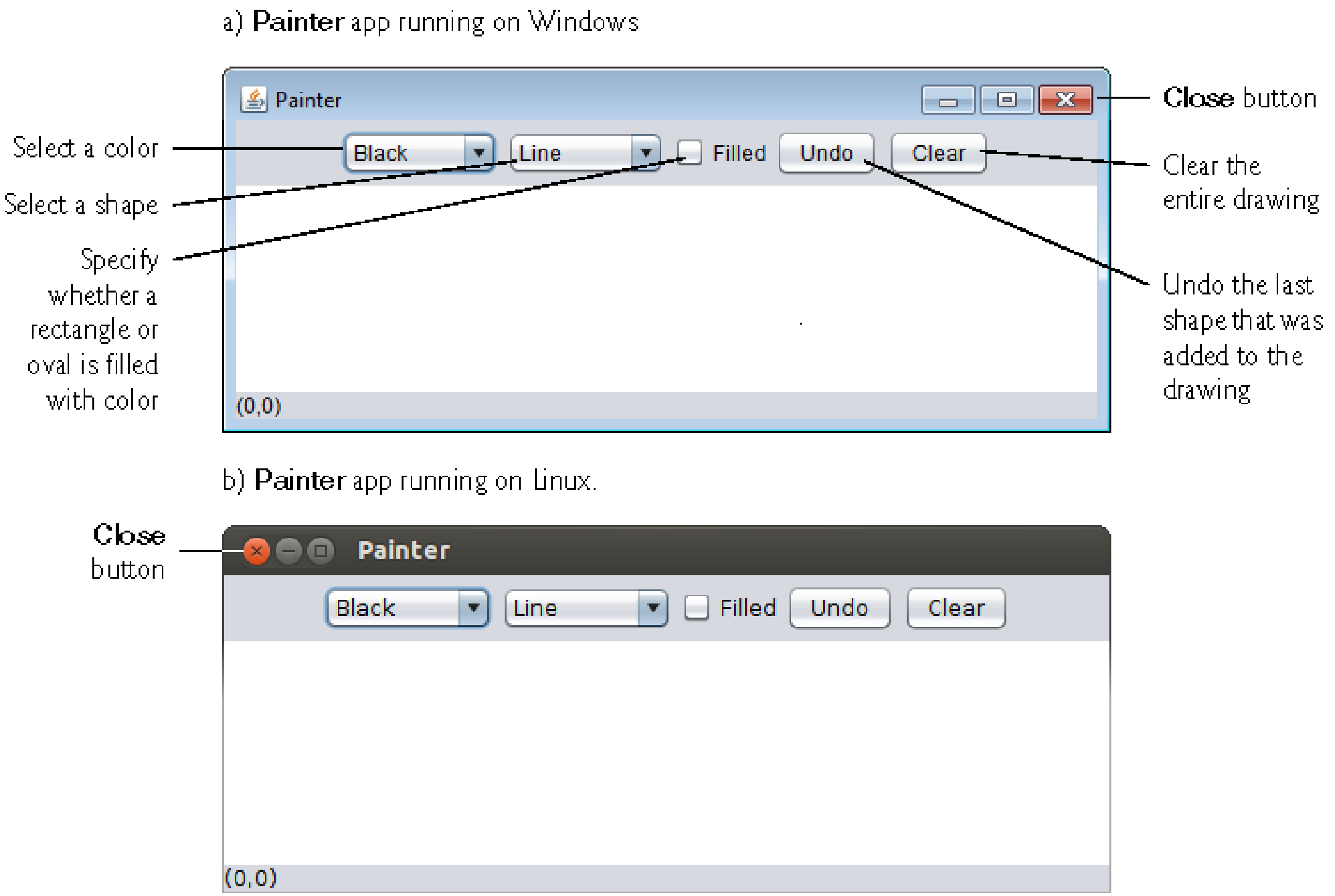


Common Programming Error 1.1

*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.*

1.10 Test-Driving a Java Application

- ***Checking your setup.*** Read the Before You Begin section to confirm that you've set up Java properly on your computer, that you've copied the book's examples to your hard drive and that you know how to open a command window on your system.



a) **Painter** app running on Windows

Select a color

Select a shape

Specify whether a rectangle or oval is filled with color

Close button

Clear the entire drawing

Undo the last shape that was added to the drawing

b) **Painter** app running on Linux.

Close button

Fig. 1.11 | **Painter** app executing in Windows 7, Linux and OS X. (Part 1 of 2.)

c) **Painter** app running on OS X.

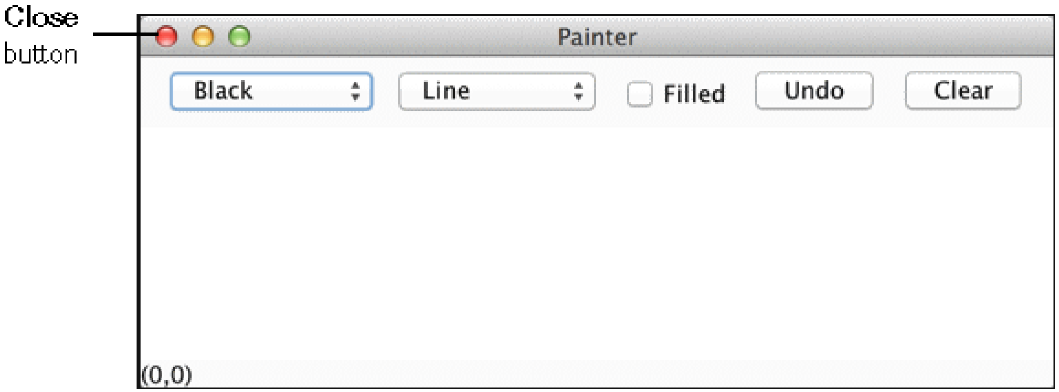


Fig. 1.11 | **Painter** app executing in Windows 7, Linux and OS X. (Part 2 of 2.)

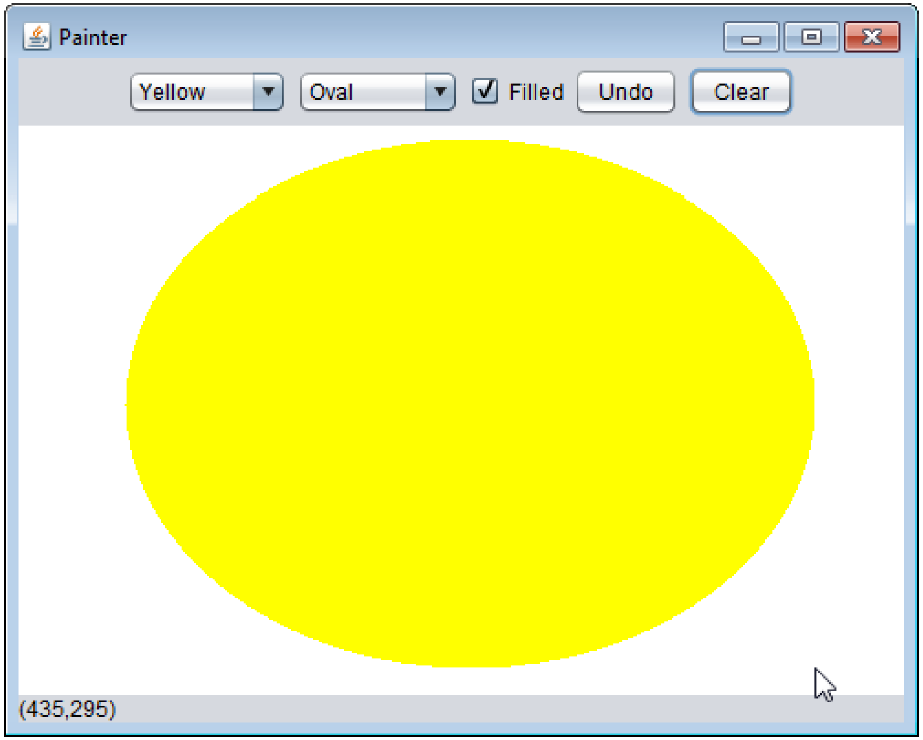
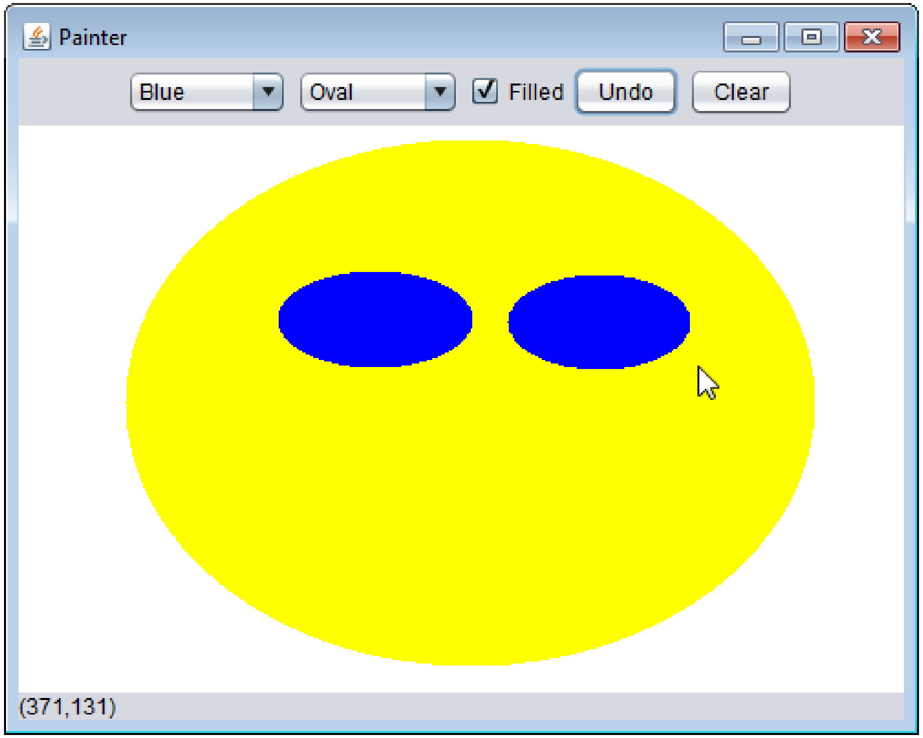


Fig. 1.12 | Drawing a filled yellow oval for the face.



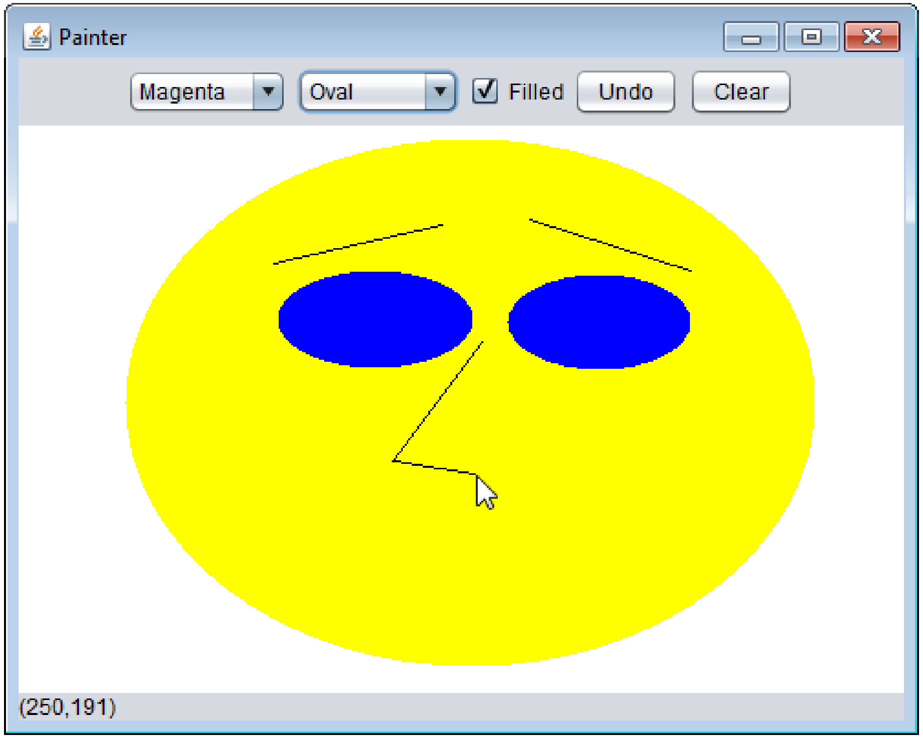
Blue

Oval

☑ Filled

(371,131)

Fig. 1.13 | Drawing blue eyes.



(250,191)

Fig. 1.14 | Drawing black eyebrows and a nose.

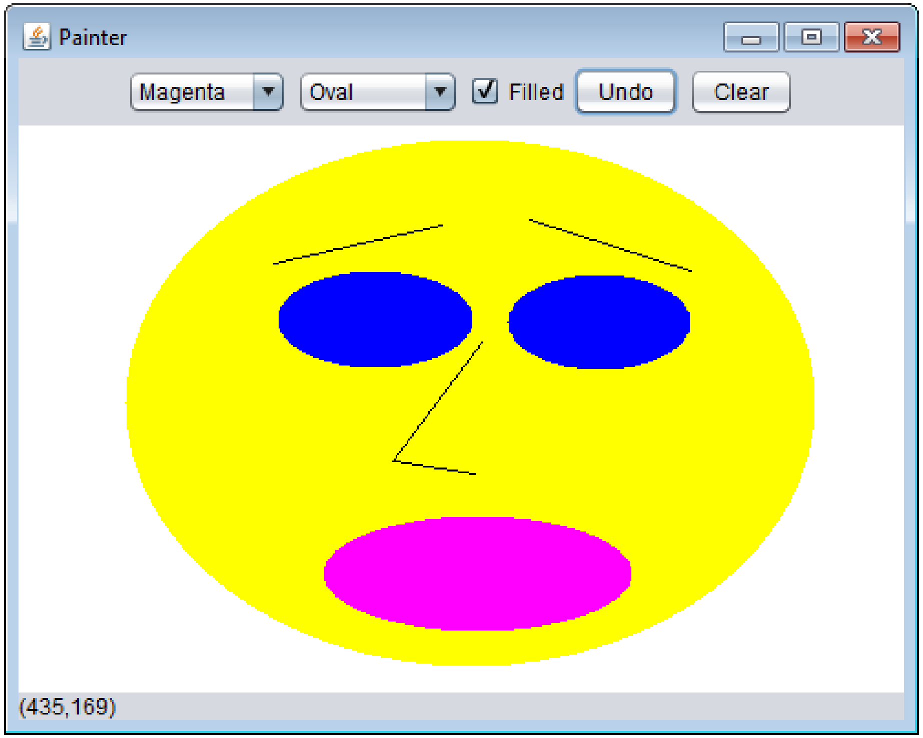


Fig. 1.15 | Drawing a magenta mouth.

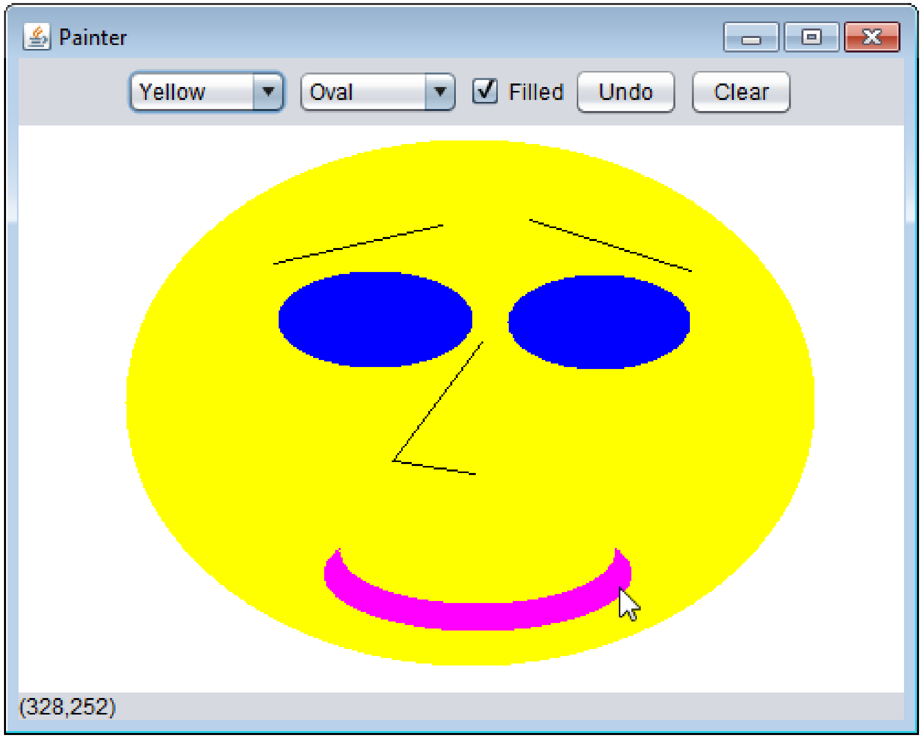


Fig. 1.16 | Drawing a yellow oval on the mouth to make a smile.

1.11 Internet and the World Wide Web

- In the late 1960s, ARPA—the Advanced Research Projects Agency of the Department of Defense—rolled out plans to network the main computer systems of approximately a dozen ARPA-funded universities and research institutions.
- ARPA implemented what quickly became known as the ARPAnet, the precursor of today's Internet.
- Its main benefit proved to be the capability for quick and easy communication via 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.

1.11 Internet and the World Wide Web (Cont.)

- 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 The 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.
- One challenge was to enable these different networks to communicate with each other.
- The Internet Protocol (IP) created a true "network of networks," the current architecture of the Internet.
- The combined set of protocols is now called TCP/IP.

1.11.1 The Internet: A Network of Networks (Cont.)

- Businesses rapidly realized that by using the Internet, they could improve their operations and offer new and better services to their clients.
- 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 The 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 multimedia-based documents (documents with various combinations of text, graphics, animations, audios and videos) on almost any subject.

1.11.2 The World Wide Web: Making the Internet User-Friendly (Cont.)

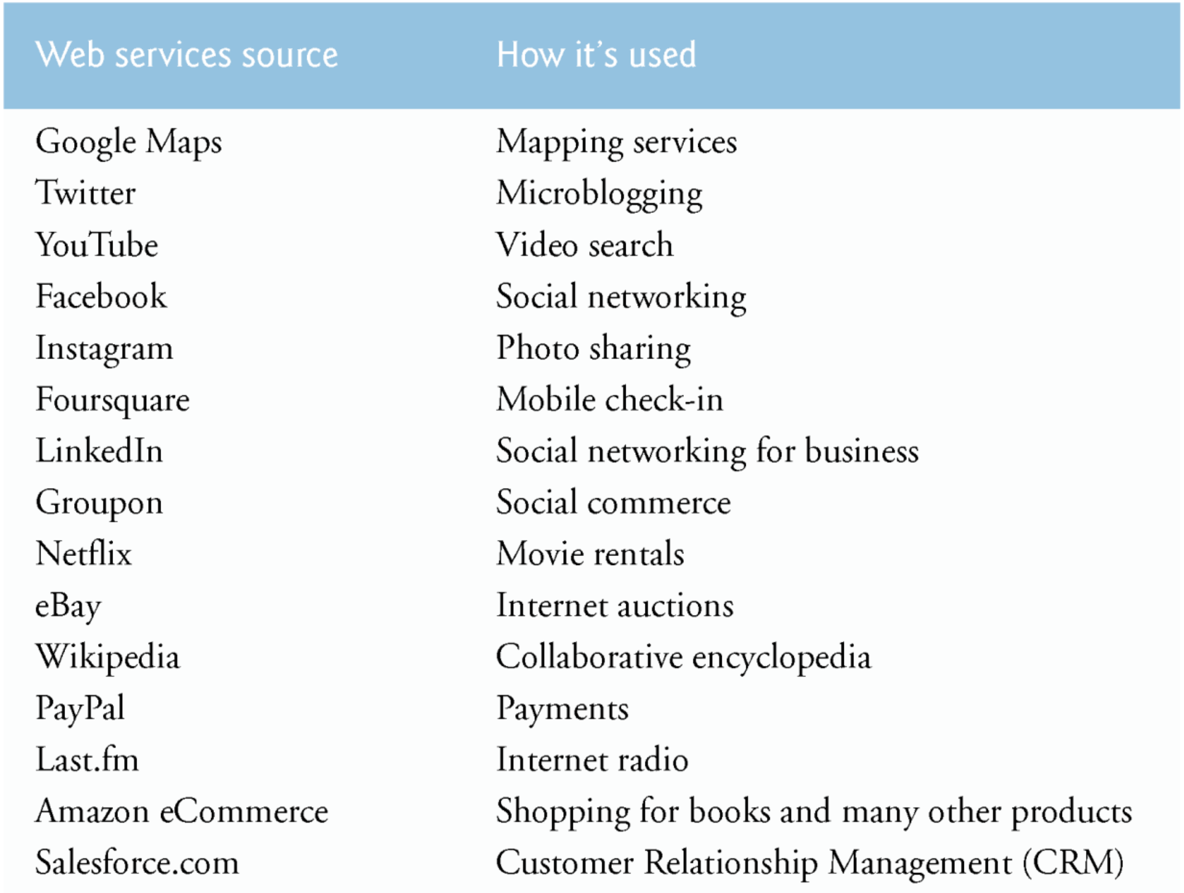
- In 1989, Tim Berners-Lee of CERN (the European Organization for Nuclear Research) began to develop a technology for sharing information via “hyperlinked” text documents.
- Berners-Lee called his invention the HyperText Markup Language (HTML).
- He also wrote communication protocols such as HyperText Transfer Protocol (HTTP) to form the backbone of his new hypertext information system, which he referred to as the World Wide Web.

1.11.2 The World Wide Web: Making the Internet User-Friendly (Cont.)

- In 1994, Berners-Lee founded the World Wide Web Consortium (W3C), devoted to developing web technologies.
- One of the W3C's goals is to make the web accessible to everyone regardless of disabilities, language or culture.

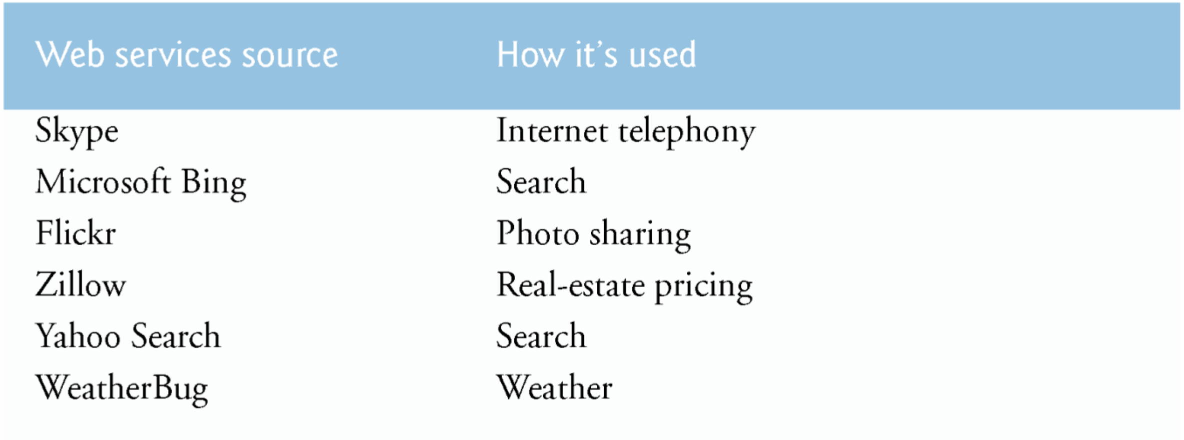
1.11.3 Web Services and Mashups

- Mashup is an applications-development methodology in which you can rapidly develop powerful software applications by combining (often free) complementary web services and other forms of information feeds.
-



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)

Fig. 1.17 | Some popular web services (`www.programmableweb.com/apis/directory/1?sort=mashups`). (Part 1 of 2.)



Web services source	How it's used
Skype	Internet telephony
Microsoft Bing	Search
Flickr	Photo sharing
Zillow	Real-estate pricing
Yahoo Search	Search
WeatherBug	Weather

Fig. 1.17 | Some popular web services (`www.programmableweb.com/apis/directory/1?sort=mashups`). (Part 2 of 2.)

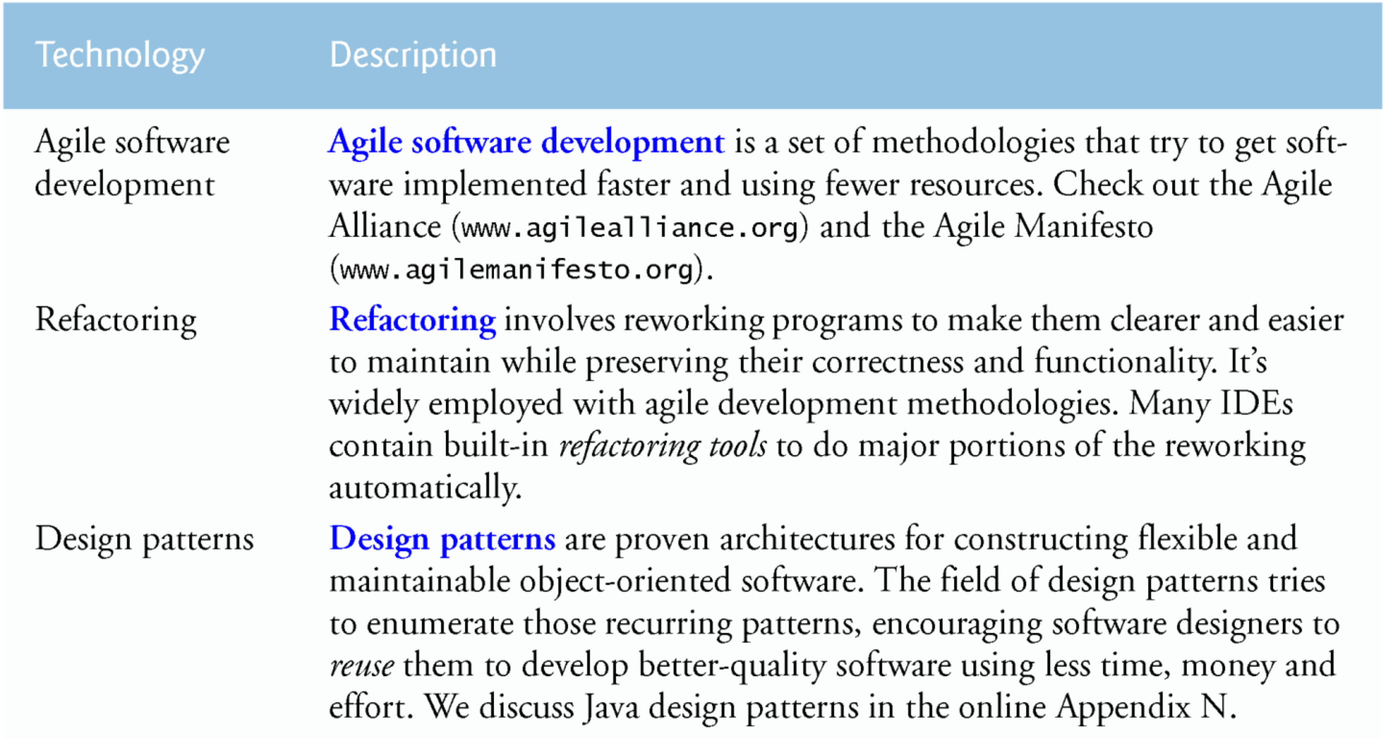
1.11.4 Ajax

- Ajax helps Internet-based applications perform like desktop applications
- A difficult task, given that such applications suffer transmission delays as data is shuttled back and forth between your computer and server computers on the Internet.
- Applications like Google Maps have used Ajax to achieve excellent performance and approach the look-and-feel of desktop applications.

1.11.5 The Internet of Things

- The Internet is no longer just a network of computers—it's an Internet of Things.
- A *thing* is any object with an IP address and the ability to send data automatically over a network:
  - a car with a transponder for paying tolls
  - a heart monitor implanted in a human
  - a smart meter that reports energy usage
  - mobile apps that can track your movement and location
  - smart thermostats that adjust room temperatures based on weather forecasts and activity in the home.

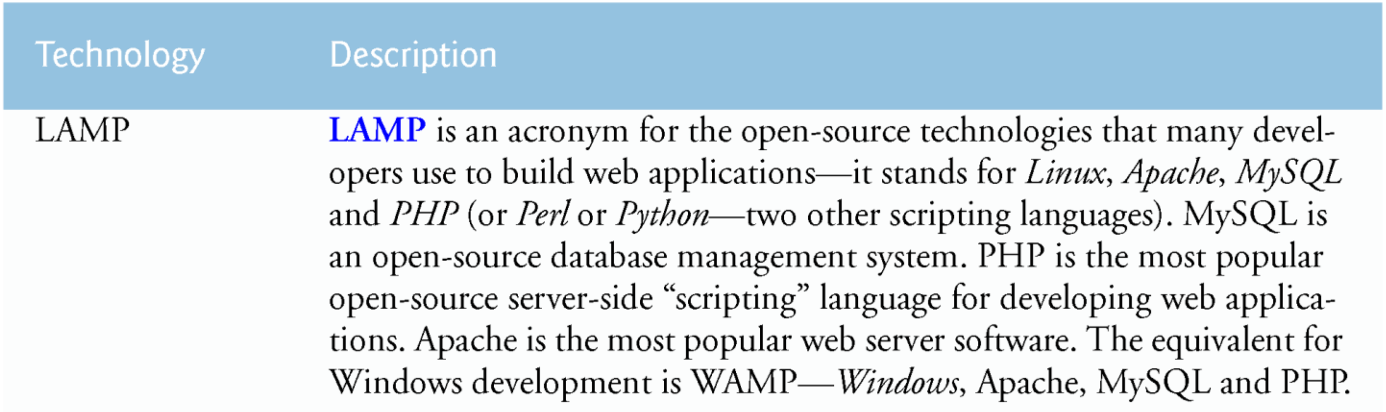
1.12 Software Technologies



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. We discuss Java design patterns in the online Appendix N.

Fig. 1.18 | Software technologies. (Part 1 of 4.)

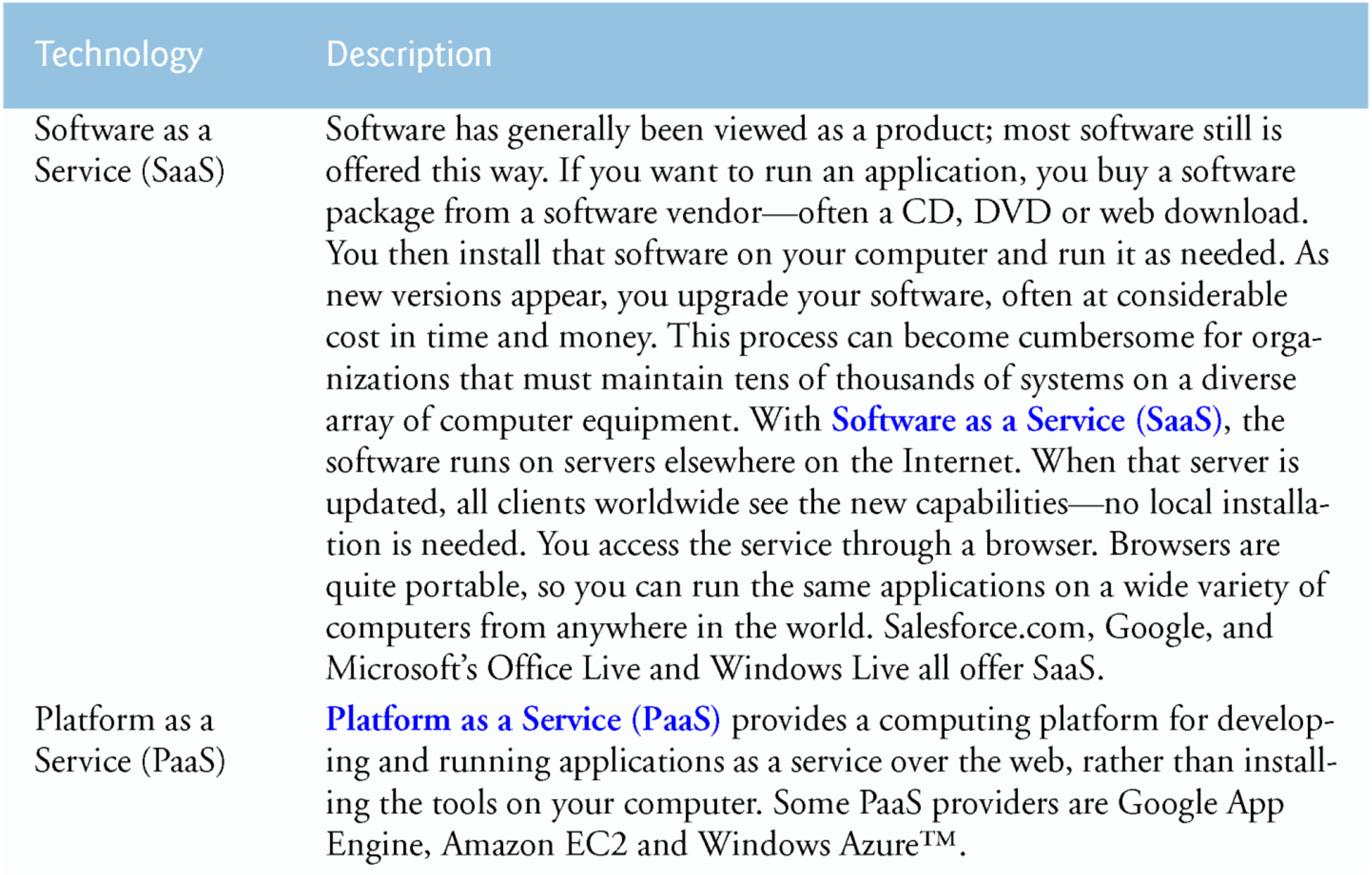
1.12 Software Technologies



Technology	Description
LAMP	LAMP is an acronym for the open-source technologies that many developers use to build web applications—it stands for *Linux*, *Apache*, *MySQL* and *PHP* (or *Perl* or *Python*—two other scripting languages). MySQL is an open-source database management system. PHP is the most 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.

Fig. 1.18 | Software technologies. (Part 2 of 4.)

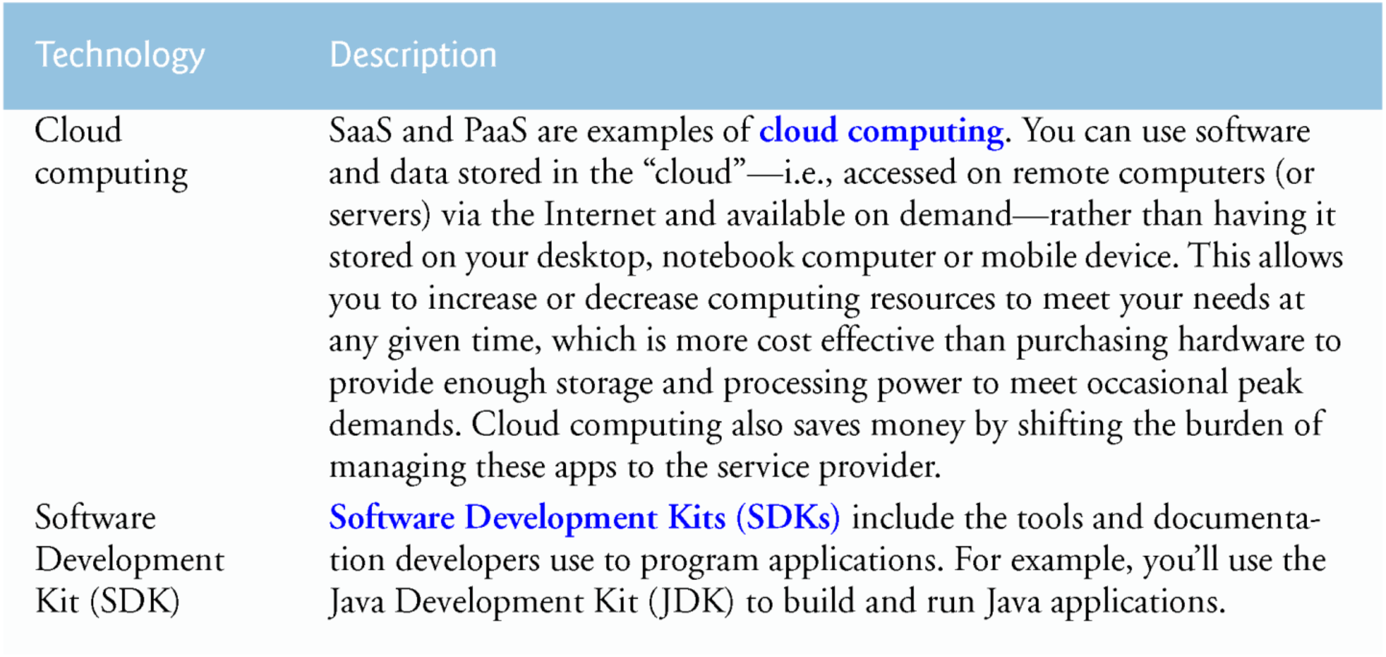
1.12 Software Technologies



Technology	Description
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, and Microsoft's Office Live and Windows Live all 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™.

Fig. 1.18 | Software technologies. (Part 3 of 4.)

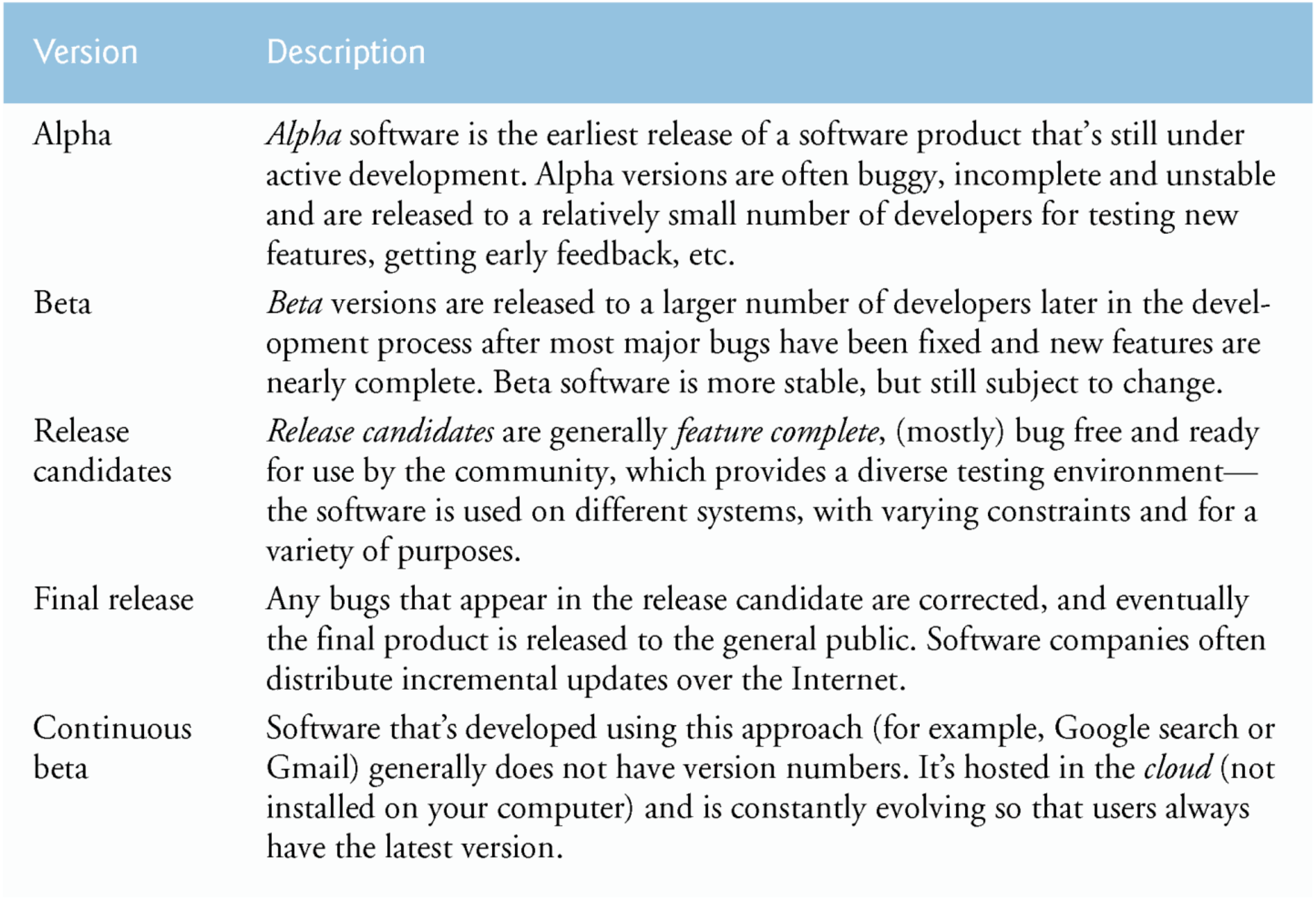
1.12 Software Technologies



Technology	Description
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 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 the burden of managing these apps to the service provider.
Software Development Kit (SDK)	**Software Development Kits (SDKs)** include the tools and documentation developers use to program applications. For example, you’ll use the Java Development Kit (JDK) to build and run Java applications.

Fig. 1.18 | Software technologies. (Part 4 of 4.)

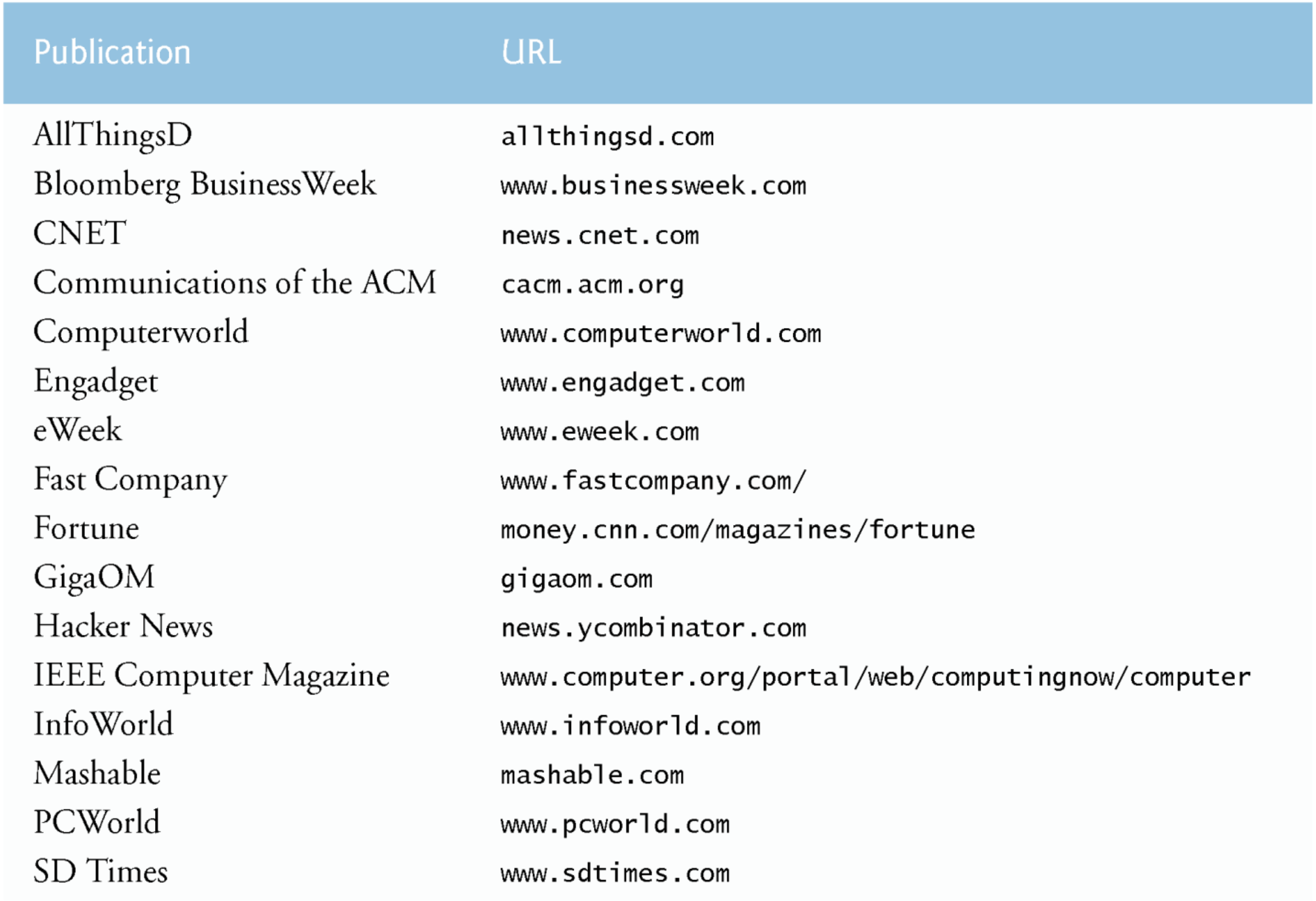
1.12 Software Technologies



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.
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. 1.19 | Software product-release terminology.

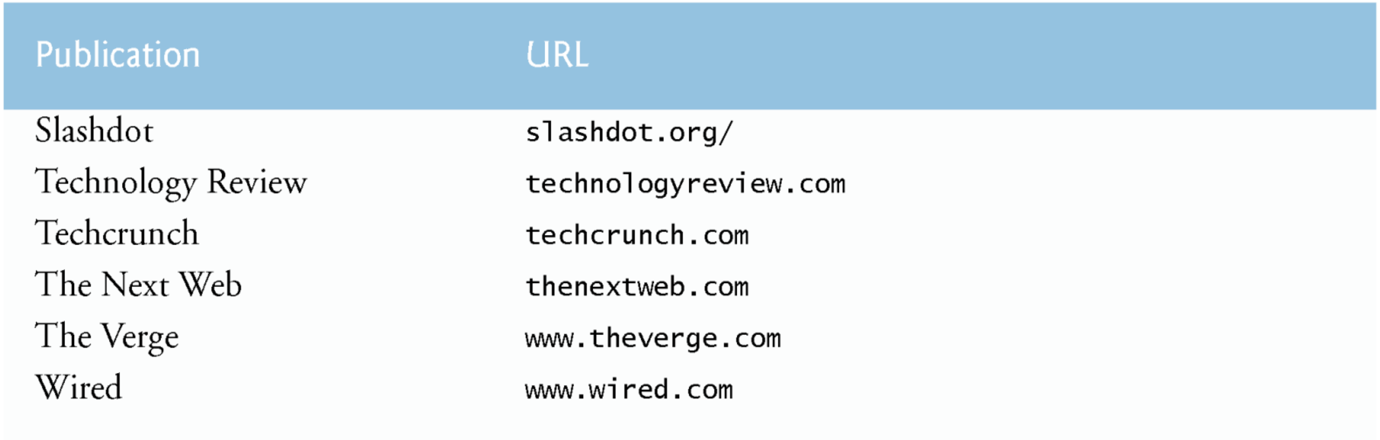
1.13 Keeping Up-to-Date with Information Technologies



Publication	URL
AllThingsD	allthingsd.com
Bloomberg BusinessWeek	www.businessweek.com
CNET	news.cnet.com
Communications of the ACM	cacm.acm.org
Computerworld	www.computerworld.com
Engadget	www.engadget.com
eWeek	www.eweek.com
Fast Company	www.fastcompany.com/
Fortune	money.cnn.com/magazines/fortune
GigaOM	gigaom.com
Hacker News	news.ycombinator.com
IEEE Computer Magazine	www.computer.org/portal/web/computingnow/computer
InfoWorld	www.infoworld.com
Mashable	mashable.com
PCWorld	www.pcworld.com
SD Times	www.sdtimes.com

Fig. 1.20 | Technical and business publications. (Part 1 of 2.)

1.13 Keeping Up-to-Date with Information Technologies



Publication	URL
Slashdot	slashdot.org/
Technology Review	technologyreview.com
Techcrunch	techcrunch.com
The Next Web	thenextweb.com
The Verge	www.theverge.com
Wired	www.wired.com

Fig. 1.20 | Technical and business publications. (Part 2 of 2.)