
# Introduction to Java™ Programming

## *Brief Version*

TENTH EDITION

Y. Daniel Liang

ALWAYS LEARNING

PEARSON

# ONLINE ACCESS

Thank you for purchasing a new copy of ***Introduction to Java Programming, Brief Version,* Tenth Edition**. Your textbook includes six months of prepaid access to the book's Companion Website. Your textbook includes six months of prepaid access to the book's Companion Website. This prepaid subscription provides you with full access to the following student support areas:

- Video Notes are step-by-step video tutorials specifically designed to enhance the programming concepts presented in this textbook

**Use a coin to scratch off the coating and reveal your student access code.**
**Do not use a knife or other sharp object as it may damage the code.**

To access the ***Introduction to Java Programming, Brief Version,* Tenth Edition**, Companion Website for the first time, you will need to register online using a computer with an Internet connection and a web browser. The process takes just a couple of minutes and only needs to be completed once.

1. Go to **www.pearsonglobaleditions.com/Liang**
2. Click on **Companion Website**.
3. Click on the **Register** button.
4. On the registration page, enter your student access code* found beneath the scratch-off panel. Do not type the dashes. You can use lower - or uppercase.
5. Follow the on-screen instructions. If you need help at any time during the online registration process, simply click the **Need Help?** icon.
6. Once your personal Login Name and Password are confirmed, you can begin using the *Introduction to Introduction to Java Programming, Brief Version* Companion Website!

### To log in after you have registered:

You only need to register for this Companion Website once. After that, you can log in any time at **www.pearsonglobaleditions.com/Liang** by providing your Login Name and Password when prompted.

*Important: The access code can only be used once. This subscription is valid for six months upon activation and is not transferable. If this access code has already been revealed, it may no longer be valid. If this is the case, you can purchase a subscription by going to **www.pearsonglobaleditions.com/Liang** and following the on-screen instructions.

# INTRODUCTION TO JAVA®

# PROGRAMMING

## BRIEF VERSION

Tenth Edition

Global Edition


**Y. Daniel Liang**

*Armstrong Atlantic State University*

Global Edition contributions by

**Ming-Jyh Tsai**

*Fu Jen Catholic University*



PEARSON

Boston Columbus Indianapolis New York San Francisco Upper Saddle River
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montreal Toronto
Delhi Mexico City Sao Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

*To Samantha, Michael, and Michelle*

Editorial Director, ECS: Marcia Horton
Head of Learning Assets Acquisition, Global Editions: Laura Dent
Executive Editor: Tracy Johnson (Dunkelberger)
Editorial Assistant: Jenah Blitz-Stoehr
Project Director, Global Editions: Shona Mullen
Assistant Project Editor, Global Editions: Paromita Banerjee
Director of Marketing: Christy Lesko
Marketing Manager: Yez Alayan
Marketing Assistant: Jon Bryant
Director of Program Management: Erin Gregg
Senior Manufacturing Controller, Global Editions: Trudy Kimber
Program Management-Team Lead: Scott Disanno
Program Manager: Carole Snyder
Project Management-Team Lead: Laura Burgess
Project Manager: Robert Engelhardt
Procurement Specialist: Linda Sager
Cover Designer: Lumina Datamatics Ltd
Permissions Supervisor: Michael Joyce
Permissions Administrator: Jenell Forschler
Director, Image Asset Services: Annie Atherton
Manager, Visual Research: Karen Sanatar
Cover Art: © Ints Vikmanis/Shutterstock
Media Project Manager: Renata Butera
Media Producer, Global Editions: Pallavi Pandit
Full-Service Project Management: Laserwords Private Ltd.



---

*Pearson Education Limited*
Edinburgh Gate
Harlow
Essex CM20 2JE
England

and Associated Companies throughout the world

*Visit us on the World Wide Web at:*
www.pearsonglobaleditions.com



*Authorized adaptation from the United States edition, entitled to Introduction to Java Programming, Brief Version, 10th Edition, ISBN 978-0-13-359220-7 by Y. Daniel Liang, published by Pearson Education © 2015.*



ISBN 10: 1-292-07856-1
ISBN 13: 978-1-292-07856-4

**British Library Cataloguing-in-Publication Data**
A catalogue record for this book is available from the British Library

10 9 8 7 6 5 4 3 2 1
15 14 13 12 11

Typeset in 10/12 Times LT Std by Laserwords Private Ltd
Printed and bound by Courier Kendallville in United States of America

# PREFACE

Dear Reader,

Many of you have provided feedback on earlier editions of this book, and your comments and suggestions have greatly improved the book. This edition has been substantially enhanced in presentation, organization, examples, exercises, and supplements. The new edition:

what is new?

- Replaces Swing with JavaFX. JavaFX is a new framework for developing Java GUI programs. JavaFX greatly simplifies GUI programming and is easier to learn than Swing.
- Introduces exception handling, abstract classes, and interfaces before GUI programming to enable the GUI chapters to be skipped completely if the instructor chooses not to cover GUI.
- Covers introductions to objects and strings earlier in Chapter 4 to enable students to use objects and strings to develop interesting programs early.
- Includes many new interesting examples and exercises to stimulate student interests. More than 100 additional programming exercises are provided to instructors only on the Companion Website.

Please visit www.pearsonglobaleditions.com/Liang for a complete list of new features as well as correlations to the previous edition.

fundamentals-first

The book is fundamentals first by introducing basic programming concepts and techniques before designing custom classes. The fundamental concepts and techniques of selection statements, loops, methods, and arrays are the foundation for programming. Building this strong foundation prepares students to learn object-oriented programming and advanced Java programming.

problem-driven

This book teaches programming in a problem-driven way that focuses on problem solving rather than syntax. We make introductory programming interesting by using thought-provoking problems in a broad context. The central thread of early chapters is on problem solving. Appropriate syntax and library are introduced to enable readers to write programs for solving the problems. To support the teaching of programming in a problem-driven way, the book provides a wide variety of problems at various levels of difficulty to motivate students. To appeal to students in all majors, the problems cover many application areas, including math, science, business, financial, gaming, animation, and multimedia.

brief version
comprehensive version

The book is widely used in the introductory programming courses in the universities around the world. The book is a *brief version* of Introduction to Java Programming, *Comprehensive Version*, Tenth Edition, Global Edition. This version is designed for an introductory programming course, commonly known as CS1. It contains the first eighteen chapters in the comprehensive version and covers fundamentals of programming, object-oriented programming, GUI programming, exception handling, I/O, and recursion. The comprehensive version has additional twenty-four chapters that cover data structures, algorithms, concurrency, parallel programming, networking, internationalization, advanced GUI, database, and Web programming. The first thirteen chapters of this book are appropriate for preparing the AP Computer Science exam.

AP Computer Science

examples and exercises

The best way to teach programming is *by example*, and the only way to learn programming is *by doing*. Basic concepts are explained by example and a large number of exercises with various levels of difficulty are provided for students to practice. For our programming courses, we assign programming exercises after each lecture.

Our goal is to produce a text that teaches problem solving and programming in a broad context using a wide variety of interesting examples. If you have any comments on and suggestions for improving the book, please email me.

Sincerely,

Y. Daniel Liang
**y.daniel.liang@gmail.com**
www.cs.armstrong.edu/liang
**www.pearsonglobaleditions.com/Liang**

## ACM/IEEE Curricular 2013 and ABET Course Assessment

The new ACM/IEEE Computer Science Curricular 2013 defines the Body of Knowledge organized into 18 Knowledge Areas. To help instructors design the courses based on this book, we provide sample syllabi to identify the Knowledge Areas and Knowledge Units. The sample syllabi are for a three semester course sequence and serve as an example for institutional customization. The sample syllabi are available to instructors at www.pearsonglobaleditions.com/Liang.

Many of our users are from the ABET-accredited programs. A key component of the ABET accreditation is to identify the weakness through continuous course assessment against the course outcomes. We provide sample course outcomes for the courses and sample exams for measuring course outcomes on the instructor Website accessible from www.pearsonglobaleditions.com/Liang.

## What's New in This Edition?

This edition is completely revised in every detail to enhance clarity, presentation, content, examples, and exercises. The major improvements are as follows:

- Updated to Java 8.
- Since Swing is replaced by JavaFX, all GUI examples and exercises are revised using JavaFX.
- Lambda expressions are used to simplify coding in JavaFX and threads.
- More than 100 additional programming exercises with solutions are provided to the instructor on the Companion Website. These exercises are not printed in the text.
- Math methods are introduced earlier in Chapter 4 to enable students to write code using math functions.
- Strings are introduced earlier in Chapter 4 to enable students to use objects and strings to develop interesting programs early.
- The GUI chapters are moved to after abstract classes and interfaces so that these chapters can be easily skipped if the instructor chooses not to cover GUI.
- Chapters 4, 14, 15, and 16 are brand new chapters.

# Pedagogical Features

The book uses the following elements to help students get the most from the material:

- The **Objectives** at the beginning of each chapter list what students should learn from the chapter. This will help them determine whether they have met the objectives after completing the chapter.
- The **Introduction** opens the discussion with representative problems to give the reader an overview of what to expect from the chapter.
- **Key Points** highlight the important concepts covered in each section.
- **Check Points** provide review questions to help students track their progress as they read through the chapter and evaluate their learning.
- **Problems and Case Studies**, carefully chosen and presented in an easy-to-follow style, teach problem solving and programming concepts. The book uses many small, simple, and stimulating examples to demonstrate important ideas.
- The **Chapter Summary** reviews the important subjects that students should understand and remember. It helps them reinforce the key concepts they have learned in the chapter.
- **Quizzes** are accessible online, grouped by sections, for students to do self-test on programming concepts and techniques.
- **Programming Exercises** are grouped by sections to provide students with opportunities to apply the new skills they have learned on their own. The level of difficulty is rated as easy (no asterisk), moderate (*), hard (**), or challenging (***). The trick of learning programming is practice, practice, and practice. To that end, the book provides a great many exercises. Additionally, more than 100 programming exercises with solutions are provided to the instructors on the Companion Website. These exercises are not printed in the text.
- **Notes**, **Tips**, **Cautions**, and **Design Guides** are inserted throughout the text to offer valuable advice and insight on important aspects of program development.

**Note**
Provides additional information on the subject and reinforces important concepts.

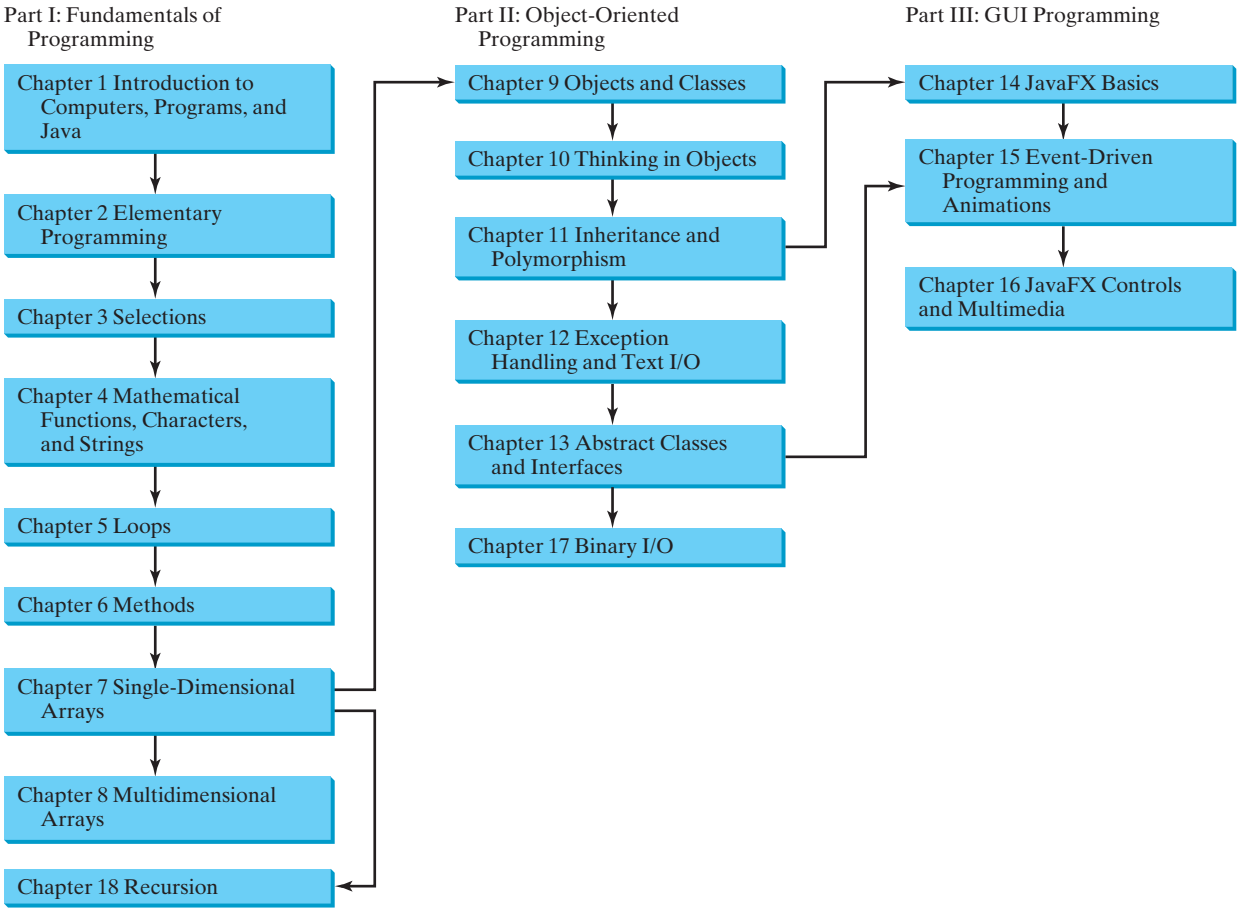
**Tip**
Teaches good programming style and practice.

**Caution**
Helps students steer away from the pitfalls of programming errors.

**Design Guide**
Provides guidelines for designing programs.

# Flexible Chapter Orderings

The book is designed to provide flexible chapter orderings to enable GUI, exception handling, and recursion to be covered earlier or later. The diagram on the next page shows the chapter dependencies.




# Organization of the Book

The chapters in this brief version can be grouped into three parts that, taken together, form a solid introduction to Java programming. Because knowledge is cumulative, the early chapters provide the conceptual basis for understanding programming and guide students through simple examples and exercises; subsequent chapters progressively present Java programming in detail, culminating with the development of comprehensive Java applications. The appendixes contain a mixed bag of topics, including an introduction to number systems, bitwise operations, regular expressions, and enumerated types.

**Part I: Fundamentals of Programming (Chapters 1–8, 18)**

The first part of the book is a stepping stone, preparing you to embark on the journey of learning Java. You will begin to learn about Java (Chapter 1) and fundamental programming techniques with primitive data types, variables, constants, assignments, expressions, and operators (Chapter 2), selection statements (Chapter 3), mathematical functions, characters, and strings (Chapter 4), loops (Chapter 5), methods (Chapter 6), and arrays (Chapters 7–8). After Chapter 7, you can jump to Chapter 18 to learn how to write recursive methods for solving inherently recursive problems.

**Part II: Object-Oriented Programming (Chapters 9–13, and 17)**

This part introduces object-oriented programming. Java is an object-oriented programming language that uses abstraction, encapsulation, inheritance, and polymorphism to provide great flexibility, modularity, and reusability in developing software. You will learn programming with objects and classes (Chapters 9–10), class inheritance (Chapter 11), polymorphism

(Chapter 11), exception handling (Chapter 12), abstract classes (Chapter 13), and interfaces (Chapter 13). Text I/O is introduced in Chapter 12 and binary I/O is discussed in Chapter 17.

**Part III: GUI Programming (Chapters 14–16)**

JavaFX is a new framework for developing Java GUI programs. It is not only useful for developing GUI programs, but also an excellent pedagogical tool for learning object-oriented programming. This part introduces Java GUI programming using JavaFX in Chapters 14–16. Major topics include GUI basics (Chapter 14), container panes (Chapter 14), drawing shapes (Chapter 14), event-driven programming (Chapter 15), animations (Chapter 15), and GUI controls (Chapter 16), and playing audio and video (Chapter 16). You will learn the architecture of JavaFX GUI programming and use the controls, shapes, panes, image, and video to develop useful applications.

**Appendixes**

This part of the book covers a mixed bag of topics. Appendix A lists Java keywords. Appendix B gives tables of ASCII characters and their associated codes in decimal and in hex. Appendix C shows the operator precedence. Appendix D summarizes Java modifiers and their usage. Appendix E discusses special floating-point values. Appendix F introduces number systems and conversions among binary, decimal, and hex numbers. Finally, Appendix G introduces bitwise operations. Appendix H introduces regular expressions. Appendix I covers enumerated types.

# Java Development Tools

You can use a text editor, such as the Windows Notepad or WordPad, to create Java programs and to compile and run the programs from the command window. You can also use a Java development tool, such as NetBeans or Eclipse. These tools support an integrated development environment (IDE) for developing Java programs quickly. Editing, compiling, building, executing, and debugging programs are integrated in one graphical user interface. Using these tools effectively can greatly increase your programming productivity. NetBeans and Eclipse are easy to use if you follow the tutorials. Tutorials on NetBeans and Eclipse can be found under Tutorials on the Student Companion Website at www.pearsonglobaleditions.com/Liang.

IDE tutorials

# Student Resource Website

The Student Resource Website www.pearsonglobaleditions.com/Liang provides access to some of the following resources. Other resources are available using the student access code printed on the inside front cover of this book. (For students with a used copy of this book, you can purchase access to the premium student resources through www.pearsonglobaleditions.com/Liang.)

- Answers to review questions
- Solutions to even-numbered programming exercises
- Source code for the examples in the book
- Interactive quiz (organized by sections for each chapter)
- Supplements
- Debugging tips
- Algorithm animations
- Errata

# Instructor Resource Website

The Instructor Resource Website, accessible from www.pearsonglobaleditions.com/Liang, provides access to the following resources:

- Microsoft PowerPoint slides with interactive buttons to view full-color, syntax-highlighted source code and to run programs without leaving the slides.

- Solutions to all programming exercises. Students will have access to the solutions of even-numbered programming exercises.
- More than 100 additional programming exercises organized by chapters. These exercises are available only to the instructors. Solutions to these exercises are provided.
- Web-based quiz generator. (Instructors can choose chapters to generate quizzes from a large database of more than two thousand questions.)
- Sample exams. Most exams have four parts:
  - Multiple-choice questions or short-answer questions
  - Correct programming errors
  - Trace programs
  - Write programs
- ACM/IEEE Curricula 2013. The new ACM/IEEE Computer Science Curricula 2013 defines the Body of Knowledge organized into 18 Knowledge Areas. To help instructors design the courses based on this book, we provide sample syllabi to identify the Knowledge Areas and Knowledge Units. The sample syllabi are for a three semester course sequence and serve as an example for institutional customization. Instructors can access the syllabi at www.pearsonglobaleditions.com/Liang.
- Sample exams with ABET course assessment.
- Projects. In general, each project gives a description and asks students to analyze, design, and implement the project.

Some readers have requested the materials from the Instructor Resource Website. Please understand that these are for instructors only. Such requests will not be answered.


VideoNote

## VideoNotes

We are excited about the new VideoNotes feature that is found in this new edition. These videos provide additional help by presenting examples of key topics and showing how to solve problems completely, from design through coding. VideoNotes are available from www.pearsonglobaleditions.com/Liang.


Animation

## Algorithm Animations

We have provided numerous animations for algorithms. These are valuable pedagogical tools to demonstrate how algorithms work. Algorithm animations can be accessed from the Companion Website.

# Acknowledgments

I would like to thank Armstrong Atlantic State University for enabling me to teach what I write and for supporting me in writing what I teach. Teaching is the source of inspiration for continuing to improve the book. I am grateful to the instructors and students who have offered comments, suggestions, bug reports, and praise.

This book has been greatly enhanced thanks to outstanding reviews for this and previous editions. The reviewers are: Elizabeth Adams (James Madison University), Syed Ahmed (North Georgia College and State University), Omar Aldawud (Illinois Institute of Technology), Stefan Andrei (Lamar University), Yang Ang (University of Wollongong, Australia), Kevin Bierre (Rochester Institute of Technology), David Champion (DeVry Institute), James Chegwidden (Tarrant County College), Anup Dargar (University of North Dakota), Charles Dierbach (Towson University), Frank Ducrest (University of Louisiana at Lafayette), Erica Eddy (University of Wisconsin at Parkside), Deena Engel (New York University), Henry A. Etlinger (Rochester Institute of Technology), James Ten Eyck (Marist College), Myers Foreman (Lamar University), Olac Fuentes (University of Texas at El Paso), Edward F. Gehringer (North Carolina State University), Harold Grossman (Clemson University), Barbara Guillot (Louisiana State University), Stuart Hansen (University of Wisconsin, Parkside), Dan Harvey (Southern Oregon University), Ron Hofman (Red River College, Canada), Stephen Hughes (Roanoke College), Vladan Jovanovic (Georgia Southern University), Edwin Kay (Lehigh University), Larry King (University of Texas at Dallas), Nana Kofi (Langara College, Canada), George Koutsogiannakis (Illinois Institute of Technology), Roger Kraft (Purdue University at Calumet), Norman Krumpe (Miami University), Hong Lin (DeVry Institute), Dan Lipsa (Armstrong Atlantic State University), James Madison (Rensselaer Polytechnic Institute), Frank Malinowski (Darton College), Tim Margush (University of Akron), Debbie Masada (Sun Microsystems), Blayne Mayfield (Oklahoma State University), John McGrath (J.P. McGrath Consulting), Hugh McGuire (Grand Valley State), Shyamal Mitra (University of Texas at Austin), Michel Mitri (James Madison University), Kenrick Mock (University of Alaska Anchorage), Frank Murgolo (California State University, Long Beach), Jun Ni (University of Iowa), Benjamin Nystuen (University of Colorado at Colorado Springs), Maureen Opkins (CA State University, Long Beach), Gavin Osborne (University of Saskatchewan), Kevin Parker (Idaho State University), Dale Parson (Kutztown University), Mark Pendergast (Florida Gulf Coast University), Richard Povinelli (Marquette University), Roger Priebe (University of Texas at Austin), Mary Ann Pumphrey (De Anza Junior College), Pat Roth (Southern Polytechnic State University), Amr Sabry (Indiana University), Ben Setzer (Kennesaw State University), Carolyn Schauble (Colorado State University), David Scuse (University of Manitoba), Ashraf Shirani (San Jose State University), Daniel Spiegel (Kutztown University), Joslyn A. Smith (Florida Atlantic University) , Lixin Tao (Pace University), Ronald F. Taylor (Wright State University), Russ Tront (Simon Fraser University), Deborah Trytten (University of Oklahoma), Michael Verdicchio (Citadel), Kent Vidrine (George Washington University), and Bahram Zartoshty (California State University at Northridge).

It is a great pleasure, honor, and privilege to work with Pearson. I would like to thank Tracy Johnson and her colleagues Marcia Horton, Yez Alayan, Carole Snyder, Scott Disanno, Bob Engelhardt, Haseen Khan, and their colleagues for organizing, producing, and promoting this project.

As always, I am indebted to my wife, Samantha, for her love, support, and encouragement.

Pearson would like to thank and acknowledge Lee Yee Lien (Multimedia University) and Vincent Chung Shen Hung (Wawasan Open University) for reviewing the Global Edition.

# BRIEF CONTENTS

# CONTENTS

# VideoNotes

Locations of **VideoNotes**
http://www.pearsonglobaleditions.com/Liang

CHAPTER

1

# INTRODUCTION TO COMPUTERS, PROGRAMS, AND JAVA

## Objectives

- To understand computer basics, programs, and operating systems (§§1.2–1.4).
- To describe the relationship between Java and the World Wide Web (§1.5).
- To understand the meaning of Java language specification, API, JDK, and IDE (§1.6).
- To write a simple Java program (§1.7).
- To display output on the console (§1.7).
- To explain the basic syntax of a Java program (§1.7).
- To create, compile, and run Java programs (§1.8).
- To use sound Java programming style and document programs properly (§1.9).
- To explain the differences between syntax errors, runtime errors, and logic errors (§1.10).
- To develop Java programs using NetBeans (§1.11).
- To develop Java programs using Eclipse (§1.12).

## 1.1 Introduction


*The central theme of this book is to learn how to solve problems by writing a program.*

what is programming?
programming
program

This book is about programming. So, what is programming? The term *programming* means to create (or develop) software, which is also called a *program.* In basic terms, software contains the instructions that tell a computer—or a computerized device—what to do.

Software is all around you, even in devices that you might not think would need it. Of course, you expect to find and use software on a personal computer, but software also plays a role in running airplanes, cars, cell phones, and even toasters. On a personal computer, you use word processors to write documents, Web browsers to explore the Internet, and e-mail programs to send and receive messages. These programs are all examples of software. Software developers create software with the help of powerful tools called *programming languages.*

This book teaches you how to create programs by using the Java programming language. There are many programming languages, some of which are decades old. Each language was invented for a specific purpose—to build on the strengths of a previous language, for example, or to give the programmer a new and unique set of tools. Knowing that there are so many programming languages available, it would be natural for you to wonder which one is best. But, in truth, there is no "best" language. Each one has its own strengths and weaknesses. Experienced programmers know that one language might work well in some situations, whereas a different language may be more appropriate in other situations. For this reason, seasoned programmers try to master as many different programming languages as they can, giving them access to a vast arsenal of software-development tools.

If you learn to program using one language, you should find it easy to pick up other languages. The key is to learn how to solve problems using a programming approach. That is the main theme of this book.

You are about to begin an exciting journey: learning how to program. At the outset, it is helpful to review computer basics, programs, and operating systems. If you are already familiar with such terms as CPU, memory, disks, operating systems, and programming languages, you may skip Sections 1.2–1.4.

## 1.2 What Is a Computer?


*A computer is an electronic device that stores and processes data.*

hardware
software

A computer includes both *hardware* and *software.* In general, hardware comprises the visible, physical elements of the computer, and software provides the invisible instructions that control the hardware and make it perform specific tasks. Knowing computer hardware isn't essential to learning a programming language, but it can help you better understand the effects that a program's instructions have on the computer and its components. This section introduces computer hardware components and their functions.

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

- A central processing unit (CPU)
- Memory (main memory)
- Storage devices (such as disks and CDs)
- Input devices (such as the mouse and keyboard)
- Output devices (such as monitors and printers)
- Communication devices (such as modems and network interface cards)

bus

A computer's components are interconnected by a subsystem called a *bus.* You can think of a bus as a sort of system of roads running among the computer's components; data and power travel along the bus from one part of the computer to another. In personal computers,

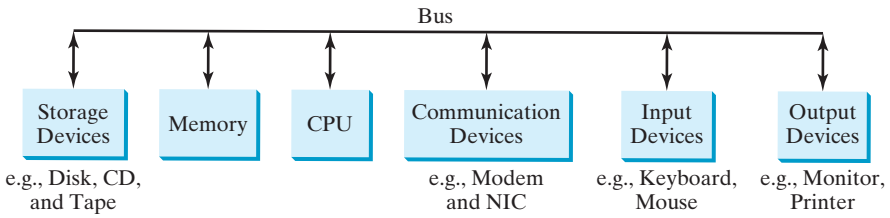



**FIGURE 1.1** A computer consists of a CPU, memory, storage devices, input devices, output devices, and communication devices.

the bus is built into the computer's *motherboard*, which is a circuit case that connects all of the parts of a computer together.

motherboard

## 1.2.1 Central Processing Unit

The *central processing unit (CPU)* is the computer's brain. It retrieves instructions from memory and executes them. The CPU usually has two components: a *control unit* and an *arithmetic/logic unit.* The control unit controls and coordinates the actions of the other components. The arithmetic/logic unit performs numeric operations (addition, subtraction, multiplication, division) and logical operations (comparisons).

CPU

Today's CPUs are built on small silicon semiconductor chips that contain millions of tiny electric switches, called *transistors*, for processing information.

Every computer has an internal clock, which emits electronic pulses at a constant rate. These pulses are used to control and synchronize the pace of operations. A higher clock *speed* enables more instructions to be executed in a given period of time. The unit of measurement of clock speed is the *hertz* (*Hz*), with 1 hertz equaling 1 pulse per second. In the 1990s, computers measured clocked speed in *megahertz* (*MHz*), but CPU speed has been improving continuously; the clock speed of a computer is now usually stated in *gigahertz (GHz)*. Intel's newest processors run at about 3 GHz.

speed

hertz

megahertz

gigahertz

CPUs were originally developed with only one core. The *core* is the part of the processor that performs the reading and executing of instructions. In order to increase CPU processing power, chip manufacturers are now producing CPUs that contain multiple cores. A multicore CPU is a single component with two or more independent cores. Today's consumer computers typically have two, three, and even four separate cores. Soon, CPUs with dozens or even hundreds of cores will be affordable.

core

## 1.2.2 Bits and Bytes

Before we discuss memory, let's look at how information (data and programs) are stored in a computer.

A computer is really nothing more than a series of switches. Each switch exists in two states: on or off. Storing information in a computer is simply a matter of setting a sequence of switches on or off. If the switch is on, its value is 1. If the switch is off, its value is 0. These 0s and 1s are interpreted as digits in the binary number system and are called *bits* (binary digits).

bits

The minimum storage unit in a computer is a *byte.* A byte is composed of eight bits. A small number such as **3** can be stored as a single byte. To store a number that cannot fit into a single byte, the computer uses several bytes.

byte

Data of various kinds, such as numbers and characters, are encoded as a series of bytes. As a programmer, you don't need to worry about the encoding and decoding of data, which the computer system performs automatically, based on the encoding scheme. An *encoding scheme* is a set of rules that govern how a computer translates characters, numbers, and symbols into data the computer can actually work with. Most schemes translate each character

encoding scheme

into a predetermined string of bits. In the popular ASCII encoding scheme, for example, the character **C** is represented as **01000011** in one byte.

A computer's storage capacity is measured in bytes and multiples of the byte, as follows:

kilobyte (KB)

- A *kilobyte (KB)* is about 1,000 bytes.

megabyte (MB)

- A *megabyte (MB)* is about 1 million bytes.

gigabyte (GB)

- A *gigabyte (GB)* is about 1 billion bytes.

terabyte (TB)

- A *terabyte (TB)* is about 1 trillion bytes.

A typical one-page word document might take 20 KB. Therefore, 1 MB can store 50 pages of documents and 1 GB can store 50,000 pages of documents. A typical two-hour high-resolution movie might take 8 GB, so it would require 160 GB to store 20 movies.

## 1.2.3 Memory

memory

A computer's *memory* consists of an ordered sequence of bytes for storing programs as well as data that the program is working with. You can think of memory as the computer's work area for executing a program. A program and its data must be moved into the computer's memory before they can be executed by the CPU.

unique address

Every byte in the memory has a *unique address*, as shown in Figure 1.2. The address is used to locate the byte for storing and retrieving the data. Since the bytes in the memory can be accessed in any order, the memory is also referred to as *random-access memory (RAM)*.

RAM

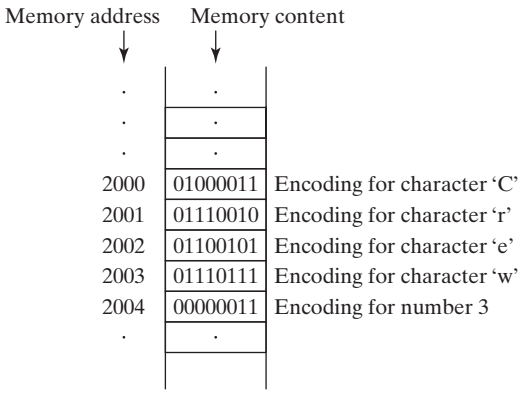



**FIGURE 1.2** Memory stores data and program instructions in uniquely addressed memory locations.

Today's personal computers usually have at least 4 gigabyte of RAM, but they more commonly have 6 to 8 GB installed. Generally speaking, the more RAM a computer has, the faster it can operate, but there are limits to this simple rule of thumb.

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

Like the CPU, memory is built on silicon semiconductor chips that have millions of transistors embedded on their surface. Compared to CPU chips, memory chips are less complicated, slower, and less expensive.

## 1.2.4 Storage Devices

A computer's memory (RAM) is a volatile form of data storage: any information that has been stored in memory (i.e., saved) is lost when the system's power is turned off. Programs and data are permanently stored on *storage devices* and are moved, when the computer

storage devices

actually uses them, to memory, which operates at much faster speeds than permanent storage devices can.

There are three main types of storage devices:

- Magnetic disk drives
- Optical disc drives (CD and DVD)
- USB flash drives

*Drives* are devices for operating a medium, such as disks and CDs. A storage medium physically stores data and program instructions. The drive reads data from the medium and writes data onto the medium.

drive

### Disks

A computer usually has at least one hard disk drive. *Hard disks* are used for permanently storing data and programs. Newer computers have hard disks that can store from 500 gigabytes to 1 terabytes of data. Hard disk drives are usually encased inside the computer, but removable hard disks are also available.

hard disk

### CDs and DVDs

*CD* stands for compact disc. There are two types of CD drives: CD-R and CD-RW. A *CD-R* is for read-only permanent storage; the user cannot modify its contents once they are recorded. A *CD-RW* can be used like a hard disk; that is, you can write data onto the disc, and then overwrite that data with new data. A single CD can hold up to 700 MB. Most new PCs are equipped with a CD-RW drive that can work with both CD-R and CD-RW discs.

CD-R

CD-RW

*DVD* stands for digital versatile disc or digital video disc. DVDs and CDs look alike, and you can use either to store data. A DVD can hold more information than a CD; a standard DVD's storage capacity is 4.7 GB. Like CDs, there are two types of DVDs: DVD-R (read-only) and DVD-RW (rewritable).

DVD

### USB Flash Drives

*Universal serial bus (USB)* connectors allow the user to attach many kinds of peripheral devices to the computer. You can use a USB to connect a printer, digital camera, mouse, external hard disk drive, and other devices to the computer.

A USB *flash drive* is a device for storing and transporting data. A flash drive is small—about the size of a pack of gum. It acts like a portable hard drive that can be plugged into your computer's USB port. USB flash drives are currently available with up to 256 GB storage capacity.

## 1.2.5 Input and Output Devices

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

### The Keyboard

A keyboard is a device for entering input. Compact keyboards are available without a numeric keypad.

*Function keys* are located across the top of the keyboard and are prefaced with the letter *F*. Their functions depend on the software currently being used.

function key

A *modifier key* is a special key (such as the *Shift*, *Alt*, and *Ctrl* keys) that modifies the normal action of another key when the two are pressed simultaneously.

modifier key

The *numeric keypad*, located on the right side of most keyboards, is a separate set of keys styled like a calculator to use for entering numbers quickly.

numeric keypad

*Arrow keys,* located between the main keypad and the numeric keypad, are used to move the mouse pointer up, down, left, and right on the screen in many kinds of programs.

arrow keys

Insert key
Delete key
Page Up key
Page Down key

The *Insert*, *Delete*, *Page Up*, and *Page Down keys* are used in word processing and other programs for inserting text and objects, deleting text and objects, and moving up or down through a document one screen at a time.

### The Mouse

A *mouse* is a pointing device. It is used to move a graphical pointer (usually in the shape of an arrow) called a *cursor* around the screen or to click on-screen objects (such as a button) to trigger them to perform an action.

### The Monitor

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

screen resolution
pixels

The *screen resolution* specifies the number of pixels in horizontal and vertical dimensions of the display device. *Pixels* (short for "picture elements") are tiny dots that form an image on the screen. A common resolution for a 17-inch screen, for example, is 1,024 pixels wide and 768 pixels high. The resolution can be set manually. The higher the resolution, the sharper and clearer the image is.

dot pitch

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

## 1.2.6 Communication Devices

Computers can be networked through communication devices, such as a dial-up modem (*mo*dulator/*demo*dulator), a DSL or cable modem, a wired network interface card, or a wireless adapter.

dial-up modem

- A *dial-up modem* uses a phone line and can transfer data at a speed up to 56,000 bps (bits per second).

digital subscriber line (DSL)

- A *digital subscriber line (DSL)* connection also uses a standard phone line, but it can transfer data 20 times faster than a standard dial-up modem.

cable modem

- A *cable modem* uses the cable TV line maintained by the cable company and is generally faster than DSL.

network interface card (NIC)
local area network (LAN)
million bits per second (mbps)

- A *network interface card (NIC)* is a device that connects a computer to a *local area network (LAN)*. LANs are commonly used in universities, businesses, and government agencies. A high-speed NIC called *1000BaseT* can transfer data at 1,000 million bits per second (mbps).
- Wireless networking is now extremely popular in homes, businesses, and schools. Every laptop computer sold today is equipped with a wireless adapter that enables the computer to connect to a local area network and the Internet.

**Note**

Answers to checkpoint questions are on the Companion Website.


**1.1** What are hardware and software?

**1.2** List five major hardware components of a computer.

**1.3** What does the acronym "CPU" stand for?

**1.4** What unit is used to measure CPU speed?

**1.5** What is a bit? What is a byte?

**1.6** What is memory for? What does RAM stand for? Why is memory called RAM?

**1.7** What unit is used to measure memory size?

**1.8** What unit is used to measure disk size?

**1.9** What is the primary difference between memory and a storage device?

## 1.3 Programming Languages

*Computer programs, known as software, are instructions that tell a computer what to do.*


Computers do not understand human languages, so programs must be written in a language a computer can use. There are hundreds of programming languages, and they were developed to make the programming process easier for people. However, all programs must be converted into the instructions the computer can execute.

### 1.3.1 Machine Language

machine language

A computer's native language, which differs among different types of computers, is its *machine language*—a set of built-in primitive instructions. These instructions are in the form of binary code, so if you want to give a computer an instruction in its native language, you have to enter the instruction as binary code. For example, to add two numbers, you might have to write an instruction in binary code, like this:

```
1101101010011010
```

### 1.3.2 Assembly Language

assembly language

Programming in machine language is a tedious process. Moreover, programs written in machine language are very difficult to read and modify. For this reason, *assembly language* was created in the early days of computing as an alternative to machine languages. Assembly language uses a short descriptive word, known as a *mnemonic*, to represent each of the machine-language instructions. For example, the mnemonic **add** typically means to add numbers and **sub** means to subtract numbers. To add the numbers **2** and **3** and get the result, you might write an instruction in assembly code like this:

```
add 2, 3, result
```

assembler

Assembly languages were developed to make programming easier. However, because the computer cannot execute assembly language, another program—called an *assembler*—is used to translate assembly-language programs into machine code, as shown in Figure 1.3.

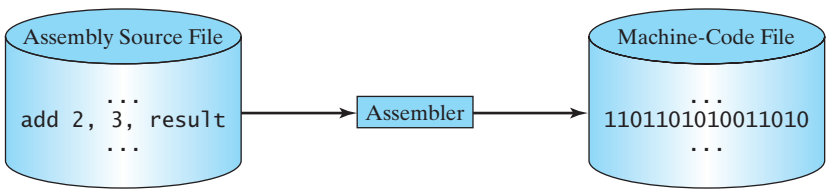



**FIGURE 1.3** An assembler translates assembly-language instructions into machine code.

low-level language

Writing code in assembly language is easier than in machine language. However, it is still tedious to write code in assembly language. An instruction in assembly language essentially corresponds to an instruction in machine code. Writing in assembly requires that you know how the CPU works. Assembly language is referred to as a *low-level language*, because assembly language is close in nature to machine language and is machine dependent.

### 1.3.3 High-Level Language

high-level language

In the 1950s, a new generation of programming languages known as *high-level languages* emerged. They are platform independent, which means that you can write a program in a high-level language and run it in different types of machines. High-level languages are English-like and easy to learn and use. The instructions in a high-level programming language are called *statements*. Here, for example, is a high-level language statement that computes the area of a circle with a radius of 5:

statement

```
area = 5 * 5 * 3.14159;
```

There are many high-level programming languages, and each was designed for a specific purpose. Table 1.1 lists some popular ones.

**TABLE 1.1** Popular High-Level Programming Languages

| *Language* | *Description* |
|---|---|
| Ada | Named for Ada Lovelace, who worked on mechanical general-purpose computers. The Ada language was developed for the Department of Defense and is used mainly in defense projects. |
| BASIC | Beginner's All-purpose Symbolic Instruction Code. It was designed to be learned and used easily by beginners. |
| C | Developed at Bell Laboratories. C combines the power of an assembly language with the ease of use and portability of a high-level language. |
| C++ | C++ is an object-oriented language, based on C. |
| C# | Pronounced "C Sharp." It is a hybrid of Java and C++ and was developed by Microsoft. |
| COBOL | COmmon Business Oriented Language. Used for business applications. |
| FORTRAN | FORmula TRANslation. Popular for scientific and mathematical applications. |
| Java | Developed by Sun Microsystems, now part of Oracle. It is widely used for developing platform-independent Internet applications. |
| Pascal | Named for Blaise Pascal, who pioneered calculating machines in the seventeenth century. It is a simple, structured, general-purpose language primarily for teaching programming. |
| Python | A simple general-purpose scripting language good for writing short programs. |
| Visual Basic | Visual Basic was developed by Microsoft and it enables the programmers to rapidly develop graphical user interfaces. |

source program
source code
interpreter
compiler

A program written in a high-level language is called a *source program* or *source code*. Because a computer cannot execute a source program, a source program must be translated into machine code for execution. The translation can be done using another programming tool called an *interpreter* or a *compiler*.

- An interpreter reads one statement from the source code, translates it to the machine code or virtual machine code, and then executes it right away, as shown in Figure 1.4a. Note that a statement from the source code may be translated into several machine instructions.
- A compiler translates the entire source code into a machine-code file, and the machine-code file is then executed, as shown in Figure 1.4b.


**1.10** What language does the CPU understand?

**1.11** What is an assembly language?

**1.12** What is an assembler?

**1.13** What is a high-level programming language?

**1.14** What is a source program?

**1.15** What is an interpreter?

**1.16** What is a compiler?

**1.17** What is the difference between an interpreted language and a compiled language?

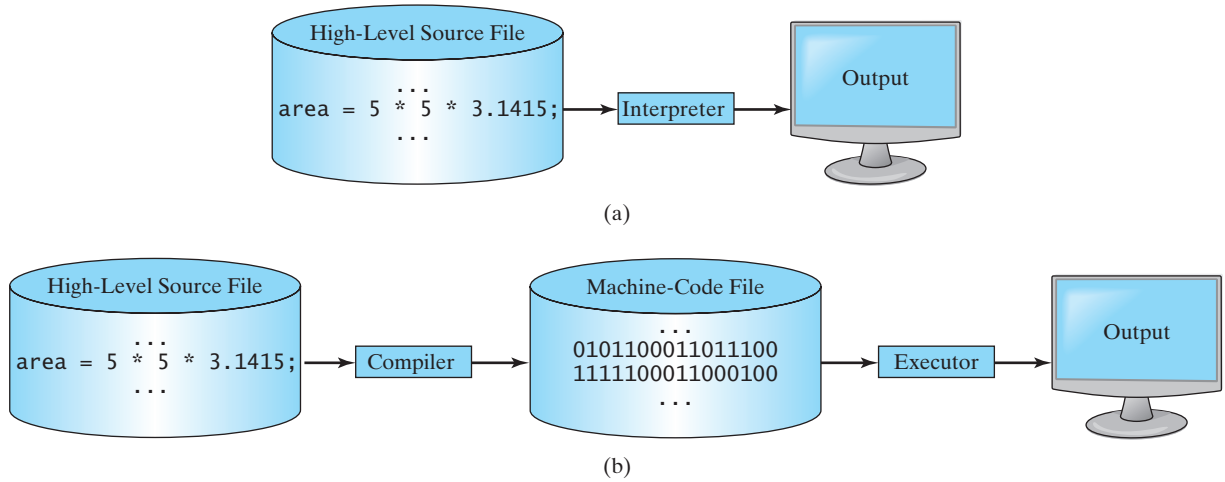



**FIGURE 1.4** (a) An interpreter translates and executes a program one statement at a time. (b) A compiler translates the entire source program into a machine-language file for execution.

## 1.4 Operating Systems

*The* operating system (OS) *is the most important program that runs on a computer. The OS manages and controls a computer's activities.*

Key Point

operating system (OS)

The popular *operating systems* for general-purpose computers are Microsoft Windows, Mac OS, and Linux. Application programs, such as a Web browser or a word processor, cannot run unless an operating system is installed and running on the computer. Figure 1.5 shows the interrelationship of hardware, operating system, application software, and the user.

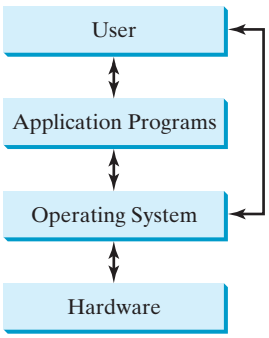



**FIGURE 1.5** Users and applications access the computer's hardware via the operating system.

The major tasks of an operating system are as follows:

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

### 1.4.1 Controlling and Monitoring System Activities

Operating systems perform basic tasks, such as recognizing input from the keyboard, sending output to the monitor, keeping track of files and folders on storage devices, and controlling peripheral devices, such as disk drives and printers. An operating system must also ensure that different programs and users working at the same time do not interfere with each other. In addition, the OS is responsible for security, ensuring that unauthorized users and programs are not allowed to access the system.

### 1.4.2 Allocating and Assigning System Resources

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

### 1.4.3 Scheduling Operations

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

multiprogramming

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

multithreading

*Multithreading* allows a single program to execute multiple tasks at the same time. For instance, a word-processing program allows users to simultaneously edit text and save it to a disk. In this example, editing and saving are two tasks within the same application. These two tasks may run concurrently.

multiprocessing

*Multiprocessing*, or *parallel processing*, uses two or more processors together to perform subtasks concurrently and then combine solutions of the subtasks to obtain a solution for the entire task. It is like a surgical operation where several doctors work together on one patient.


**1.18** What is an operating system? List some popular operating systems.

**1.19** What are the major responsibilities of an operating system?

**1.20** What are multiprogramming, multithreading, and multiprocessing?

## 1.5 Java, the World Wide Web, and Beyond


*Java is a powerful and versatile programming language for developing software running on mobile devices, desktop computers, and servers.*

This book introduces Java programming. Java was developed by a team led by James Gosling at Sun Microsystems. Sun Microsystems was purchased by Oracle in 2010. Originally called *Oak,* Java was designed in 1991 for use in embedded chips in consumer electronic appliances. In 1995, renamed *Java*, it was redesigned for developing Web applications. For the history of Java, see www.java.com/en/javahistory/index.jsp.

Java has become enormously popular. Its rapid rise and wide acceptance can be traced to its design characteristics, particularly its promise that you can write a program once and run it anywhere. As stated by its designer, Java is *simple*, *object oriented*, *distributed*,

*interpreted*, *robust*, *secure*, *architecture neutral*, *portable*, *high performance*, *multithreaded*, and *dynamic*. For the anatomy of Java characteristics, see www.cs.armstrong.edu/liang/JavaCharacteristics.pdf.

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

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

Java initially became attractive because Java programs can be run from a Web browser. Such programs are called *applets*. Applets employ a modern graphical interface with buttons, text fields, text areas, radio buttons, and so on, to interact with users on the Web and process their requests. Applets make the Web responsive, interactive, and fun to use. Applets are embedded in an HTML file. *HTML (Hypertext Markup Language)* is a simple scripting language for laying out documents, linking documents on the Internet, and bringing images, sound, and video alive on the Web. Today, you can use Java to develop rich Internet applications. A rich Internet application (RIA) is a Web application designed to deliver the same features and functions normally associated with deskop applications.

Java is now very popular for developing applications on Web servers. These applications process data, perform computations, and generate dynamic Web pages. Many commercial Websites are developed using Java on the backend.

Java is a versatile programming language: you can use it to develop applications for desktop computers, servers, and small handheld devices. The software for Android cell phones is developed using Java.


**1.21** Who invented Java? Which company owns Java now?

**1.22** What is a Java applet?

**1.23** What programming language does Android use?

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


*Java syntax is defined in the Java language specification, and the Java library is defined in the Java API. The JDK is the software for developing and running Java programs. An IDE is an integrated development environment for rapidly developing programs.*

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

Java language specification

The *Java language specification* is a technical definition of the Java programming language's syntax and semantics. You can find the complete Java language specification at http://docs.oracle.com/javase/specs/.

API
library

The *application program interface (API)*, also known as *library*, contains predefined classes and interfaces for developing Java programs. The API is still expanding. You can view and download the latest version of the Java API at http://download.java.net/jdk8/docs/api/.

Java SE, EE, and ME

Java is a full-fledged and powerful language that can be used in many ways. It comes in three editions:

- Java *Standard Edition (Java SE)* to develop client-side applications. The applications can run standalone or as applets running from a Web browser.
- *Java Enterprise Edition (Java EE)* to develop server-side applications, such as Java servlets, JavaServer Pages (JSP), and JavaServer Faces (JSF).
- *Java Micro Edition (Java ME)* to develop applications for mobile devices, such as cell phones.

Java Development Toolkit (JDK)

JDK 1.8 = JDK 8

This book uses Java SE to introduce Java programming. Java SE is the foundation upon which all other Java technology is based. There are many versions of Java SE. The latest, Java SE 8, is used in this book. Oracle releases each version with a *Java Development Toolkit (JDK)*. For Java SE 8, the Java Development Toolkit is called *JDK 1.8* (also known as *Java 8* or *JDK 8*).

Integrated development environment

The JDK consists of a set of separate programs, each invoked from a command line, for developing and testing Java programs. Instead of using the JDK, you can use a Java development tool (e.g., NetBeans, Eclipse, and TextPad)—software that provides an *integrated development environment (IDE)* for developing Java programs quickly. Editing, compiling, building, debugging, and online help are integrated in one graphical user interface. You simply enter source code in one window or open an existing file in a window, and then click a button or menu item or press a function key to compile and run the program.


**1.24** What is the Java language specification?

**1.25** What does JDK stand for?

**1.26** What does IDE stand for?

**1.27** Are tools like NetBeans and Eclipse different languages from Java, or are they dialects or extensions of Java?

## 1.7 A Simple Java Program


*A Java program is executed from the* `main` *method in the class.*

what is a console?
console input
console output

Let's begin with a simple Java program that displays the message `Welcome to Java!` on the console. (The word *console* is an old computer term that refers to the text entry and display device of a computer. *Console input* means to receive input from the keyboard, and *console output* means to display output on the monitor.) The program is shown in Listing 1.1.

### LISTING 1.1 Welcome.java

class
`main` method
display message

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

**VideoNote**
Your first Java program

```
Welcome to Java!
```

line numbers

Note that the line numbers are for reference purposes only; they are not part of the program. So, don't type line numbers in your program.