Chapter 1
Introduction to
Computers, the Internet and Java

Java How to Program, 11/e

Questions? E-mail paul.deitel@deitel.com

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.

OBJECTIVES(cont.)

In this chapter you'll:

- Understand object-oriented programming basics.
- Learn Internet and web basics.
- Learn a typical Java program-development environment.
- Test-drive a Java application.
- Learn some key recent software technologies.

OUTLINE

OUTLINE

1.5 Introduction to Object Technology

- 1.5.1 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)

OUTLINE

OUTLINE

1.1 Introduction

- Java is one of the world's most widely used computer programming languages.
- You'll learn to write instructions in the Java programming language
- 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.
- There are billions of personal computers in use and an even larger number of mobile devices with computers at their core
- According to Oracle's 2016 JavaOne conference keynote presentation (http://bit.ly/JavaOne2016Keynote), there are now 10 million Java developers worldwide and Java runs on 15 billion devices (Fig. 1.1), including two billion vehicles and 350 million medical devices.
- In addition, the explosive growth of mobile phones, tablets and other devices is creating significant opportunities for programming mobile apps.

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

Fig. 1.1 | Some devices that use Java.

1.1 Introduction (Cont.)

Java Standard Edition

- Java How to Program, 10/e is based on Java Standard Edition 8 (Java SE 8) and Java Standard Edition 9 (Java SE 9)
- 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
  - Generic programming
- Java SE 8 added the beginnings of functional programming with lambdas and streams

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)
  - 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.
  - Many of the devices in use Java ME.

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!

1.2 Computers: Hardware and Software (Cont.)

- Computers process data under the control of sequences of instructions called computer programs.
- These software programs guide the computer through ordered actions specified by people called computer programmers.
- 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 solid-state drive storage) they have to hold programs and data over longer periods of time, and their processor speeds—the speeds at which they *execute* their programs (i.e., do their work).

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 bar codes, reading from secondary storage devices (such as hard drives, DVD drives, Blu-ray Disc™ drives and USB flash drives—also called "thumb drives" or "memory sticks"), receiving video from a webcam and having your computer receive information from the Internet (such as when you stream videos from YouTube® or download e-books from Amazon). Newer forms of input include position data from a GPS device, motion and orientation information from an *accelerometer* (a device that responds to up/down, left/right and forward/backward acceleration) in a smartphone or game controller (such as Microsoft® Kinect® for Xbox®, Wii™ Remote and Sony® PlayStation® Move) and voice input from intelligent assistants like Amazon Echo and Google Home.

Fig. 1.2 | Logical units of a computer. (Part 1 of 6.)

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 solid-state drives (SSDs), hard drives, DVD drives and USB flash drives. Popular recent forms of output are smartphone and game-controller vibration, virtual reality devices like Oculus Rift® and Google Cardboard™ and mixed reality devices like Microsoft's HoloLens™.

Fig. 1.2 | Logical units of a computer. (Part 2 of 6.)

Logical unit	Description
Memory unit	This rapid-access, relatively low-capacity "warehouse" section retains information that has been entered through the input unit, making it immediately available for processing when needed. The memory unit also retains processed information until it can be placed on output devices by the output unit. Information in the memory unit is *volatile*—it's typically lost when the computer's power is turned off. The memory unit is often called either **memory**, **primary memory** or **RAM** (Random Access Memory). Main memories on desktop and notebook computers contain as much as 128 GB of RAM, though 2 to 16 GB is most common. GB stands for gigabytes; a gigabyte is approximately one billion bytes. A **byte** is eight bits. A bit is either a 0 or a 1.

Fig. 1.2 | Logical units of a computer. (Part 3 of 6.)

Logical unit	Description
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 4 of 6.)

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 **multicore processor** implements multiple processors on a single integrated-circuit chip—a *dual-core processor* has two CPUs, a *quad-core processor* has four and an *octa-core processor* has eight. Intel has some processors with up to 72 cores. Today's desktop computers have processors that can execute billions of instructions per second. Chapter 23 explores how to write apps that can take full advantage of multicore architecture.

Fig. 1.2 | Logical units of a computer. (Part 5 of 6.)

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 solid-state drives (SSDs), hard drives, DVD drives and USB flash drives, some of which can hold over 2 TB (TB stands for terabytes; a terabyte is approximately one trillion bytes). Typical hard drives on desktop and notebook computers hold up to 2 TB, and some desktop hard drives can hold up to 10 TB.

Fig. 1.2 | Logical units of a computer. (Part 6 of 6.)

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.

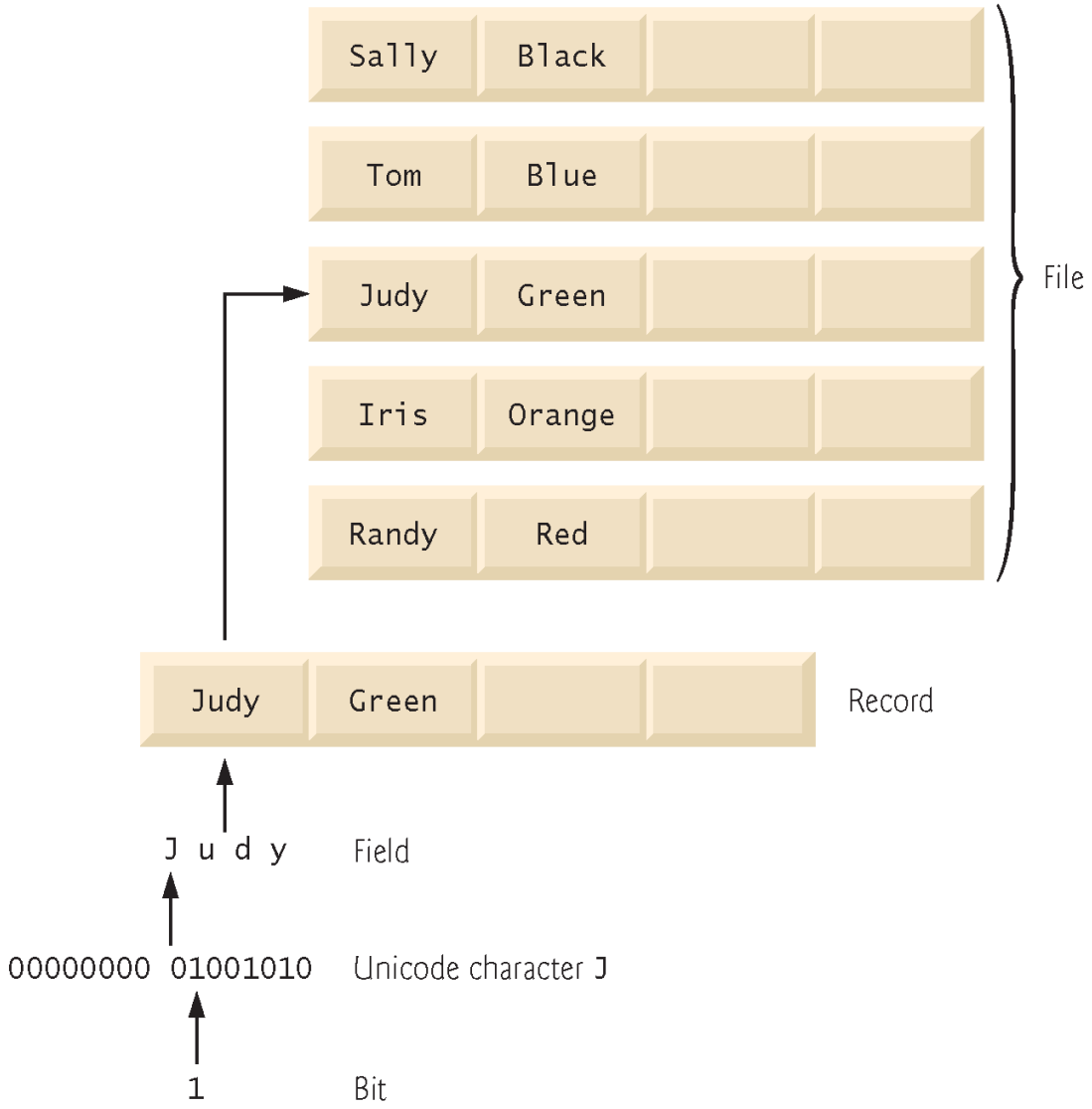


Fig. 1.3 | Data hierarchy.

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
- According to Salesforce.com, as of October 2015 90% of the world's data was created in just the prior 12 months
- 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

- 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

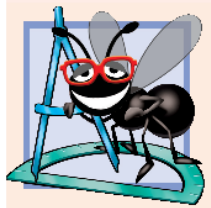
- Performing a task in a program requires a method.
- The method houses the program statements that actually perform its tasks.
- Hides these statements from its user, just as the accelerator pedal of a car hides from the driver the mechanisms of making the car go faster.
- In Java, we create a program 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 and Information Hiding

- 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 can be *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 exceptions to this in Java SE 8 and 9 when we discuss interfaces in Chapter 10).
- 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 macOS.
- The most popular mobile operating systems used in smartphones and tablets are Google's Android and Apple's iOS (for its iPhone, iPad and iPod Touch devices)

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 from many concepts (such as icons, menus and windows) developed by Xerox PARC and popularized by early Apple Macintosh operating systems.

1.6.1 Windows—A Proprietary Operating System (Cont.)

- Windows 10 is Microsoft's latest operating system
- Features include enhancements to the Start menu and user interface, Cortana personal assistant for voice interactions, Action Center for receiving notifications, Microsoft's new Edge web browser, and more
- Windows is a *proprietary* operating system—it's controlled by Microsoft exclusively.
- Windows is by far the world's most widely used operating system.

1.6.2 Linux—An Open Source Operating System

- The **Linux operating system** is perhaps the greatest success of the *open-source* movement
- **Open-source software** departs from the *proprietary* software development style that dominated software's early years
- With open-source development, individuals and companies *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.)

- Open-source code is often scrutinized by a much larger audience than proprietary software, so errors often get removed faster
- Open source also encourages innovation. Enterprise systems companies, such as IBM, Oracle and many others, have made significant investments in Linux open-source development

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

- Some key organizations in the open-source community are
  - the Eclipse Foundation (the Eclipse Integrated Development Environment helps programmers conveniently develop software)
  - the Mozilla Foundation (creators of the Firefox web browser)
  - the Apache Software Foundation (creators of the Apache web server used to develop web-based applications)
  - GitHub (which provides tools for managing open-source projects—it has millions of them under development).

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

- Rapid improvements to computing and communications, decreasing costs and open-source software have made it much easier and more economical to create a software-based business now than just a decade ago
- A great example is Facebook, which was launched from a college dorm room and built with open-source software

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

- The **Linux kernel** is the core of the most popular open-source, freely distributed, full-featured operating system
- Developed by a loosely organized team of volunteers and is popular in servers, personal computers and embedded systems (such as the computer systems at the heart of smartphones, smart TVs and automobile systems)

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

- Linux source code (the program code) is available to the public for examination and modification and is free to download and install
- Linux users benefit from a huge community of developers actively debugging and improving the kernel, and the ability to customize the operating system to meet specific needs.
- Linux has become extremely popular on servers and in embedded systems, such as Google's Android-based smartphones.

1.6.3 Apple's macOS and Apple's iOS for iPhone® , iPad® and iPod Touch® Devices

- Apple, founded in 1976 by Steve Jobs and Steve Wozniak, quickly became a leader in personal computing
- In 1979, Jobs and several Apple employees visited Xerox PARC (Palo Alto Research Center) to learn about Xerox's desktop computer that featured a graphical user interface (GUI)
- That GUI served as the inspiration for the Apple Macintosh, launched with much fanfare in a memorable Super Bowl ad in 1984

1.6.3 Apple's macOS and Apple's iOS for iPhone® , iPad® and iPod Touch® Devices (cont.)

- The Objective-C programming language, created by Brad Cox and Tom Love at Stepstone in the early 1980s, added capabilities for object-oriented programming (OOP) to the C programming language
- Steve Jobs left Apple in 1985 and founded NeXT Inc.
- In 1988, NeXT licensed Objective-C from StepStone and developed an Objective-C compiler and libraries which were used as the platform for the NeXTSTEP operating system's user interface, and Interface Builder—used to construct graphical user interfaces

1.6.3 Apple's macOS and Apple's iOS for iPhone® , iPad® and iPod Touch® Devices (cont.)

- Jobs returned to Apple in 1996 when Apple bought NeXT.
- Apple's macOS operating system is a descendant of NeXTSTEP.
- Apple's proprietary operating system, **iOS**, is derived from Apple's macOS and is used in the iPhone, iPad, iPod Touch, Apple Watch and Apple TV devices.
- In 2014, Apple introduced its new Swift programming language, which became open source in 2015.
- The iOS app-development community is shifting from Objective-C to Swift.

1.6.4 Google's Android

- **Android**—the fastest growing mobile and smartphone operating system—is based on the Linux kernel and Java.
- One benefit of developing Android apps is the openness of the platform—the operating system is open source and free.
- The Android operating system was developed by Android, Inc., which was acquired by Google in 2005.
- In 2007, the Open Handset Alliance was formed to develop, maintain and evolve Android, driving innovation in mobile technology and improving the user experience while reducing costs

1.6.4 Google's Android (cont.)

- According to Statista.com, as of Q3 2016, Android had 87.8% of the global smartphone market share, compared to 11.5% for Apple
- Android is used in smartphones, e-reader devices, tablets, in-store touch-screen kiosks, cars, robots, multimedia players and more.
- We present an introduction to Android app development in our textbook, *Android How to Program, Third Edition,* and in our professional book, *Android 6 for Programmers: An App-Driven Approach, Third Edition*.
- After you learn Java, you'll find it realtively straightforward to begin developing and running Android apps.

1.7 Programming Languages

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

Programming language	Description
Ada	Ada, based on Pascal, was developed under the sponsorship of the U.S. Department of Defense (DOD) during the 1970s and early 1980s. The DOD wanted a single language that would fill most of its needs. The Pascal-based language was named after Lady Ada Lovelace, daughter of the poet Lord Byron. She's credited with writing the world's first computer program in the early 1800s (for the Analytical Engine mechanical computing device designed by Charles Babbage). Ada also supports object-oriented programming.
Basic	Basic was developed in the 1960s at Dartmouth College to familiarize novices with programming techniques. Many of its latest versions are object oriented.

Fig. 1.5 | Some other programming languages. (Part 1 of 8.)

Programming language	Description
C	C was developed in the early 1970s by Dennis Ritchie at Bell Laboratories. It initially became widely known as the UNIX operating system's development language. Today, most code for general-purpose operating systems is written in C or C++.
C++	C++, which is based on C, was developed by Bjarne Stroustrup in the early 1980s at Bell Laboratories. C++ provides several features that "spruce up" the C language, but more important, it provides capabilities for object-oriented programming.
C#	Microsoft's three primary object-oriented programming languages are C# (based on C++ and Java), Visual C++ (based on C++) and Visual Basic (based on the original Basic). C# was developed to integrate the web into computer applications, and is now widely used to develop enterprise applications and for mobile application development.

Fig. 1.5 | Some other programming languages. (Part 2 of 8.)

Programming language	Description
COBOL	COBOL (COmmon Business Oriented Language) was developed in the late 1950s by computer manufacturers, the U.S. government and industrial computer users, based on a language developed by Grace Hopper, a career U.S. Navy officer and computer scientist. (She was posthumously awarded the Presidential Medal of Freedom in November of 2016.) COBOL is still widely used for commercial applications that require precise and efficient manipulation of large amounts of data. Its latest version supports object-oriented programming.
Fortran	Fortran (FORmula TRANslator) was developed by IBM Corporation in the mid-1950s to be used for scientific and engineering applications that require complex mathematical computations. It's still widely used, and its latest versions support object-oriented programming.

Fig. 1.5 | Some other programming languages. (Part 3 of 8.)

Programming language	Description
JavaScript	JavaScript is the most widely used scripting language. It's primarily used to add programmability to web pages—for example, animations and interactivity with the user. All major web browsers support it.
Objective-C	Objective-C is an object-oriented language based on C. It was developed in the early 1980s and later acquired by NeXT, which in turn was acquired by Apple. It became the key programming language for the OS X operating system and all iOS-powered devices (such as iPods, iPhones and iPads).
Pascal	Research in the 1960s resulted in *structured programming*—a disciplined approach to writing programs that are clearer, easier to test and debug and easier to modify than programs produced with previous techniques. The Pascal language, developed by Professor Niklaus Wirth in 1971, grew out of this research. It was popular for teaching structured programming for several decades.

Fig. 1.5 | Some other programming languages. (Part 4 of 8.)

Programming language	Description
PHP	PHP is an object-oriented, *open-source* "scripting" language supported by a community of developers and used by numerous websites. PHP is platform independent—implementations exist for all major UNIX, Linux, Mac and Windows operating systems.
Python	Python, another object-oriented scripting language, was released publicly in 1991. Developed by Guido van Rossum of the National Research Institute for Mathematics and Computer Science in Amsterdam, Python draws heavily from Modula-3—a systems programming language. Python is "extensible"—it can be extended through classes and programming interfaces.

Fig. 1.5 | Some other programming languages. (Part 5 of 8.)

Programming language	Description
Ruby on Rails	Ruby—created in the mid-1990s by Yukihiro Matsumoto—is an open-source, object-oriented programming language with a simple syntax that's similar to Python. Ruby on Rails combines the scripting language Ruby with the Rails web-application framework developed by the company 37Signals. Their book, *Getting Real* (free at `http://gettingreal.37signals.com/toc.php`), is a must-read for web developers. Many Ruby on Rails developers have reported productivity gains over other languages when developing database-intensive web applications.

Fig. 1.5 | Some other programming languages. (Part 6 of 8.)

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

Fig. 1.5 | Some other programming languages. (Part 7 of 8.)

Programming language	Description
Visual Basic	Microsoft's Visual Basic language was introduced in the early 1990s to simplify the development of Microsoft Windows applications. Its features are comparable to those of C#.

Fig. 1.5 | Some other programming languages. (Part 8 of 8.)

1.8 Java

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

1.8 Java (Cont.)

- Java drew the attention of the business community because of the phenomenal interest in the Internet.
- Now used to develop large-scale enterprise applications, to enhance the functionality of web servers, to provide applications for consumer devices, to develop robotics software and for many other purposes.
- Also the key language for developing Android smartphone and tablet apps.
- Java has become the most widely used general-purpose programming language with more than 10 million developers.

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 macOS.

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.

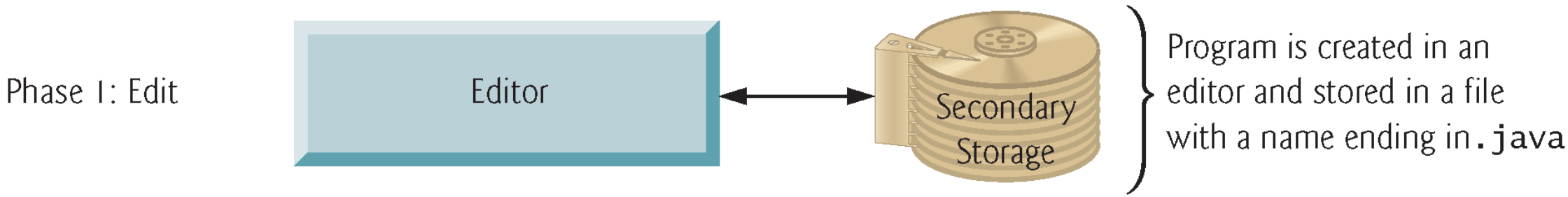


Fig. 1.6 | Typical Java development environment—editing phase.

1.9 A Typical Java Development Environment (Cont.)

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

1.9 A Typical Java Development Environment (Cont.)

- **Integrated development environments (IDEs)** provide tools that support the software development process, such as editors, debuggers for locating **logic errors** that cause programs to execute incorrectly and more.
- The most popular Java IDEs are:
  - Eclipse (http://www.eclipse.org)
  - IntelliJ IDEA (http://www.jetbrains.com)
  - NetBeans (http://www.netbeans.org)

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.

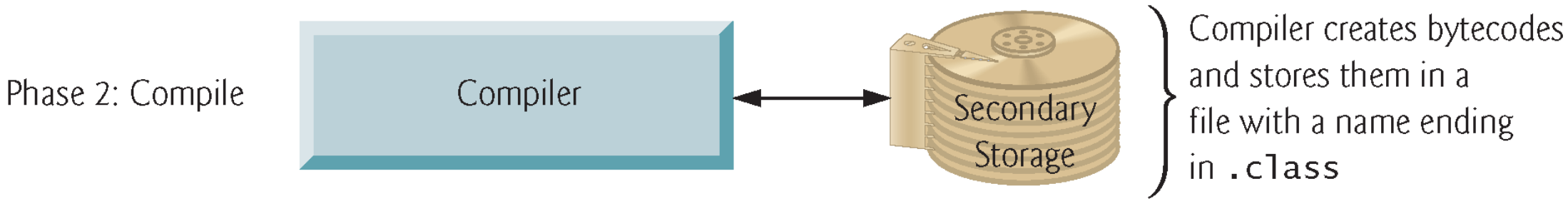


Fig. 1.7 | Typical Java development environment—compilation phase.

Common Programming Error 1.1

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

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.

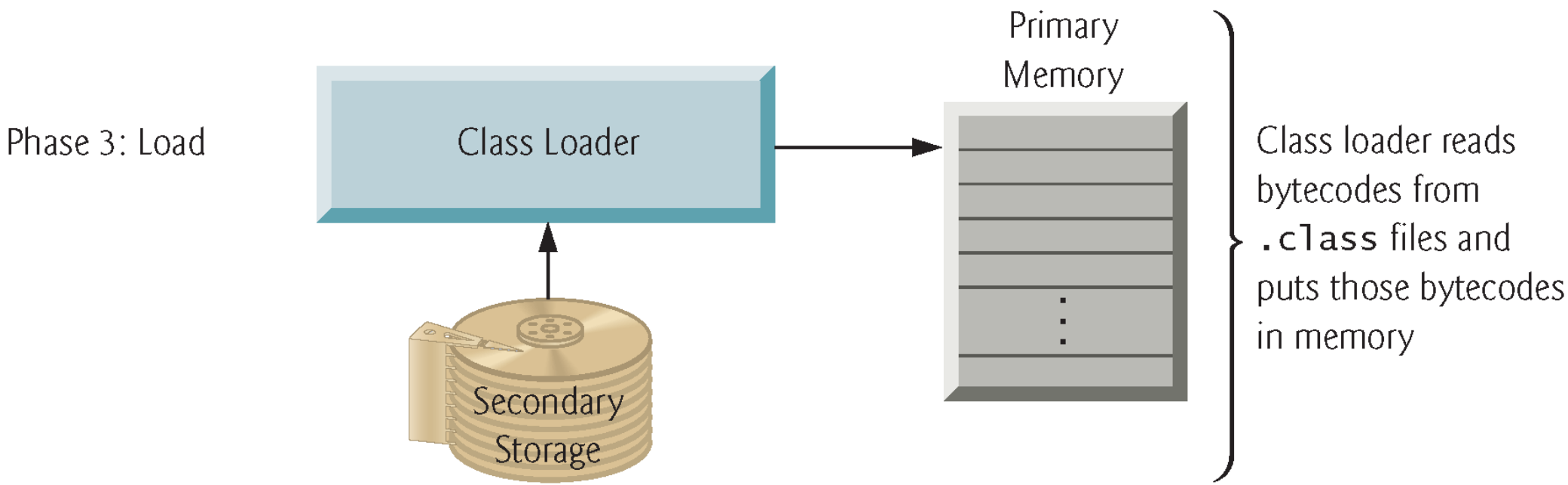


Fig. 1.8 | Typical Java development environment—loading phase.

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).

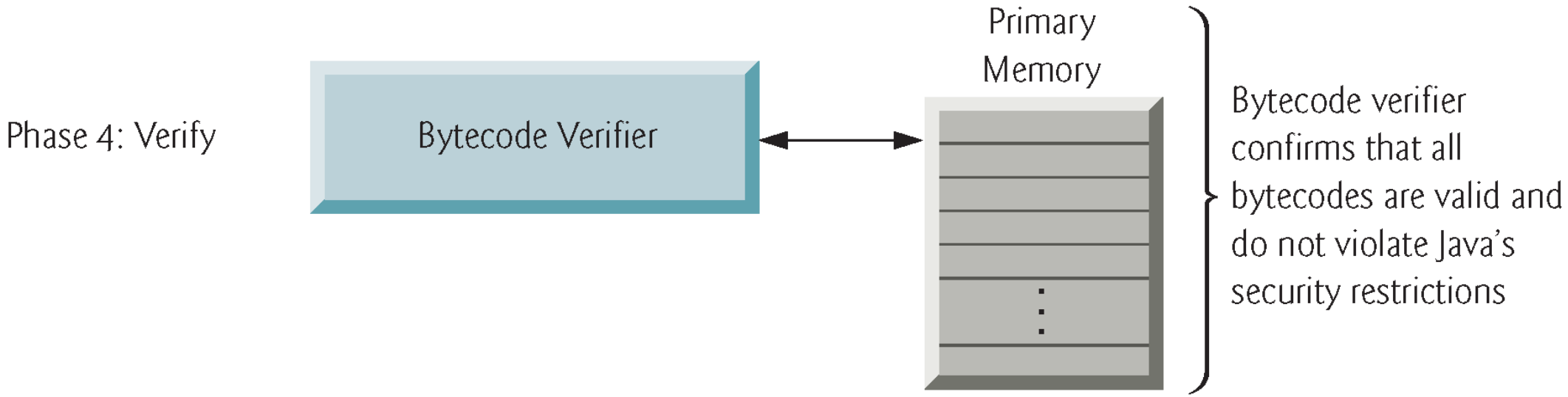


Fig. 1.9 | Typical Java development environment—verification phase.

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.

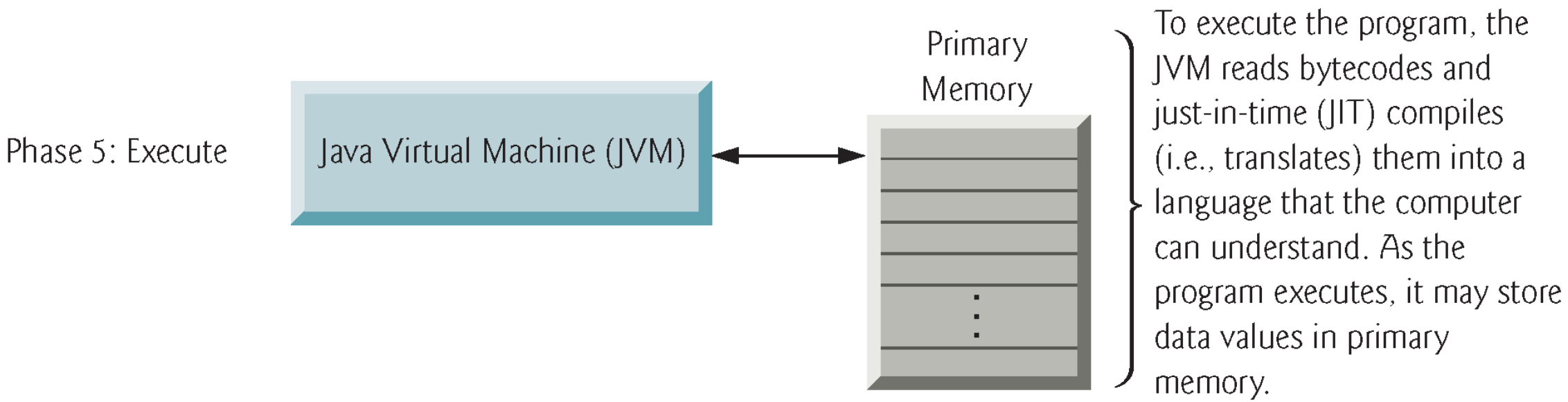


Fig. 1.10 | Typical Java development environment—execution phase.

Common Programming Error 1.2

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

1.10 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.

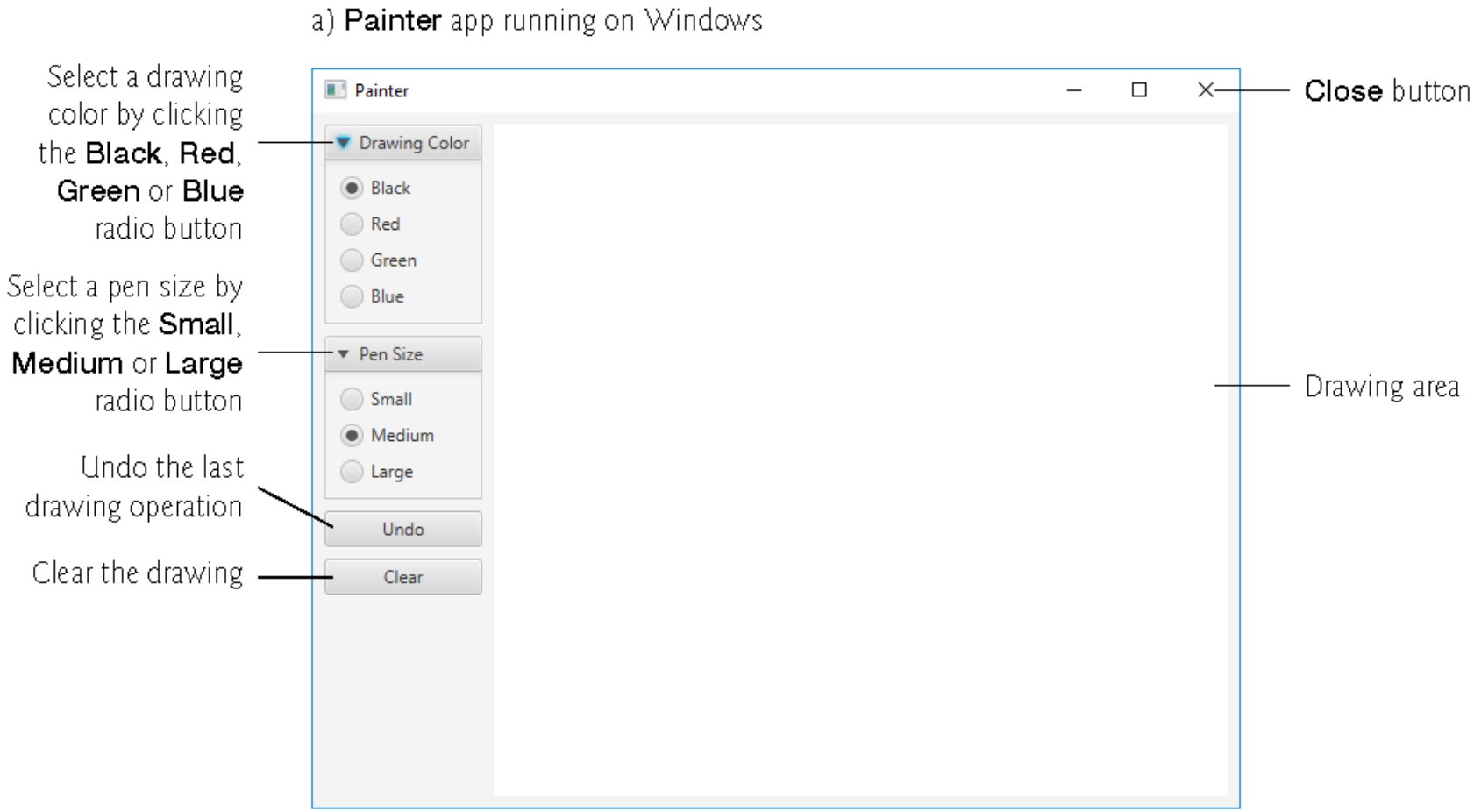


Fig. 1.11 | **Painter** app executing in Windows, Linux and macOS. (Part 1 of 3.)

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

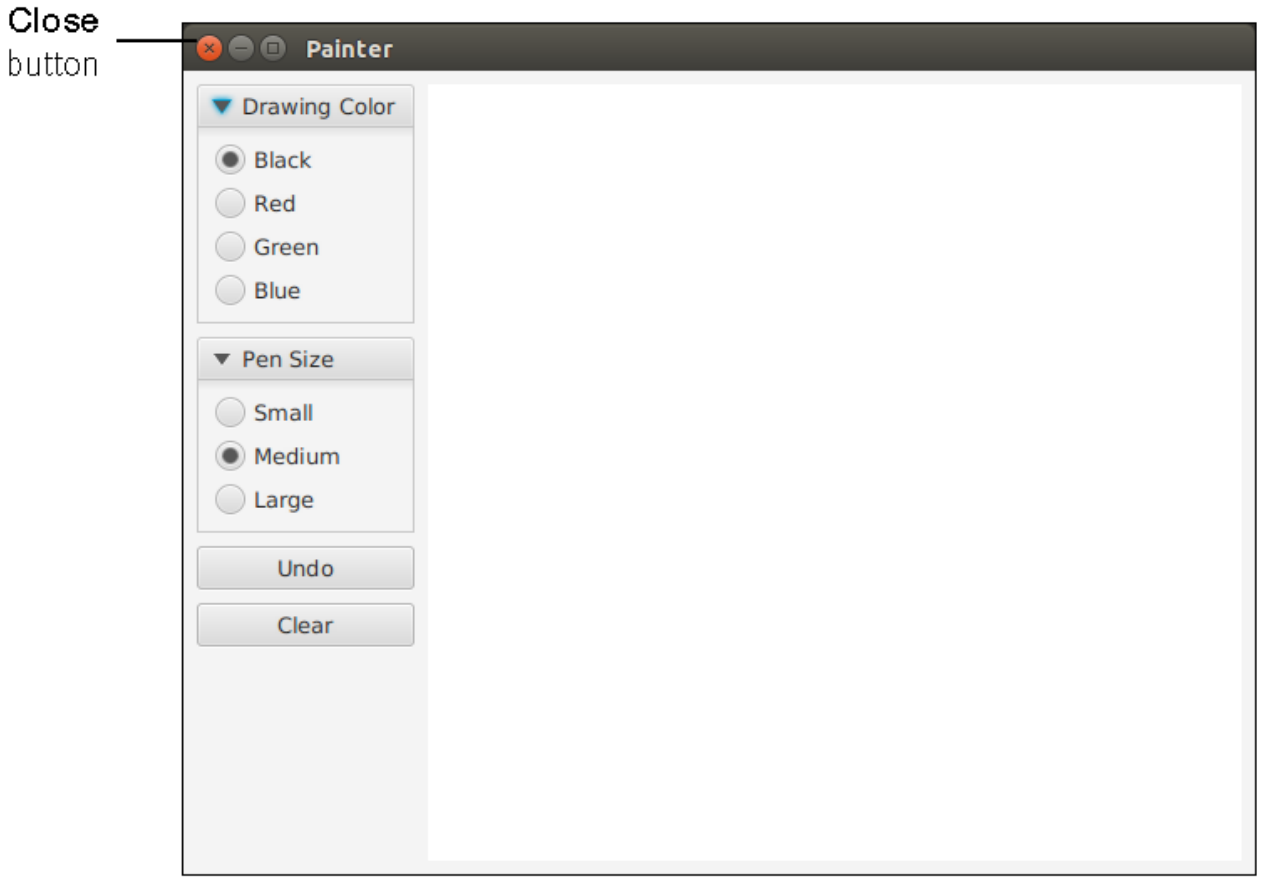


Fig. 1.11 | **Painter** app executing in Windows, Linux and macOS. (Part 2 of 3.)

c) **Painter** app running on macOS.

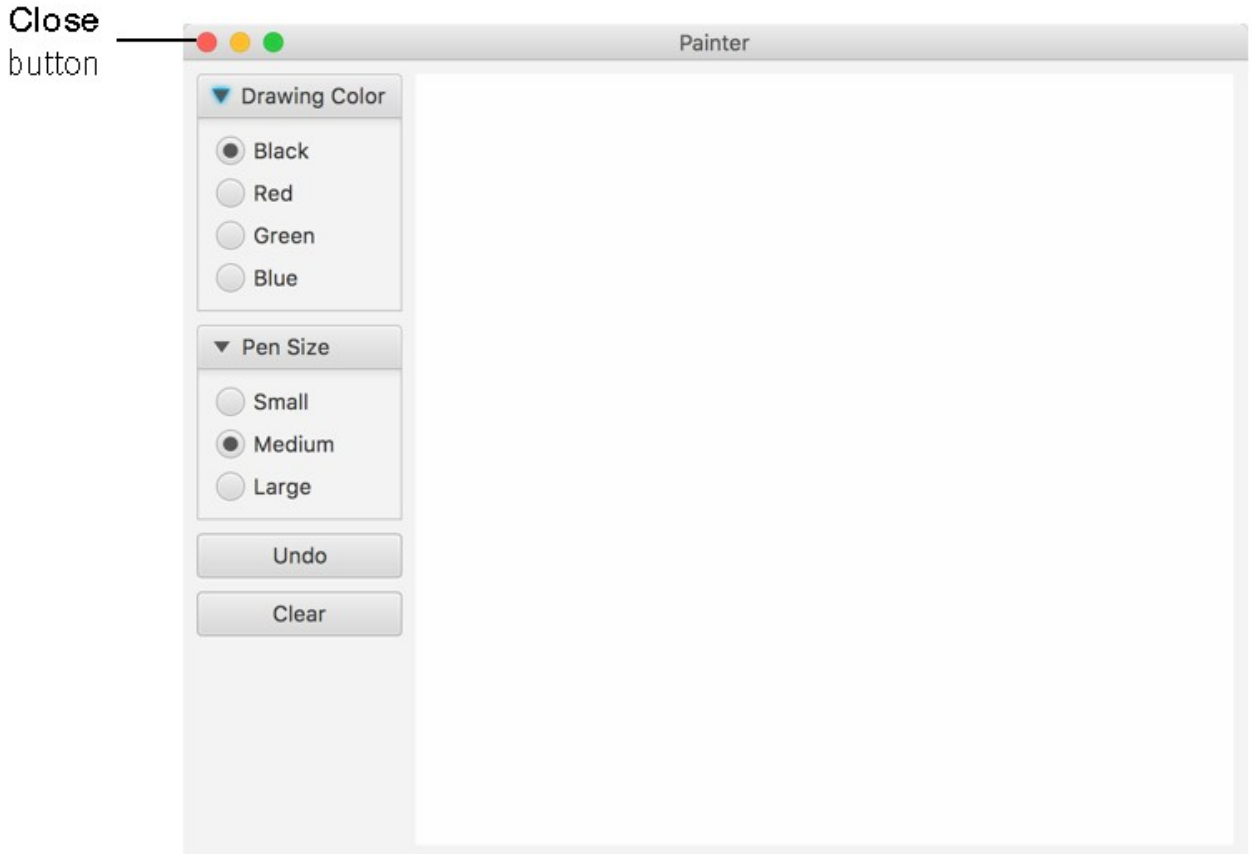


Fig. 1.11 | **Painter** app executing in Windows, Linux and macOS. (Part 3 of 3.)

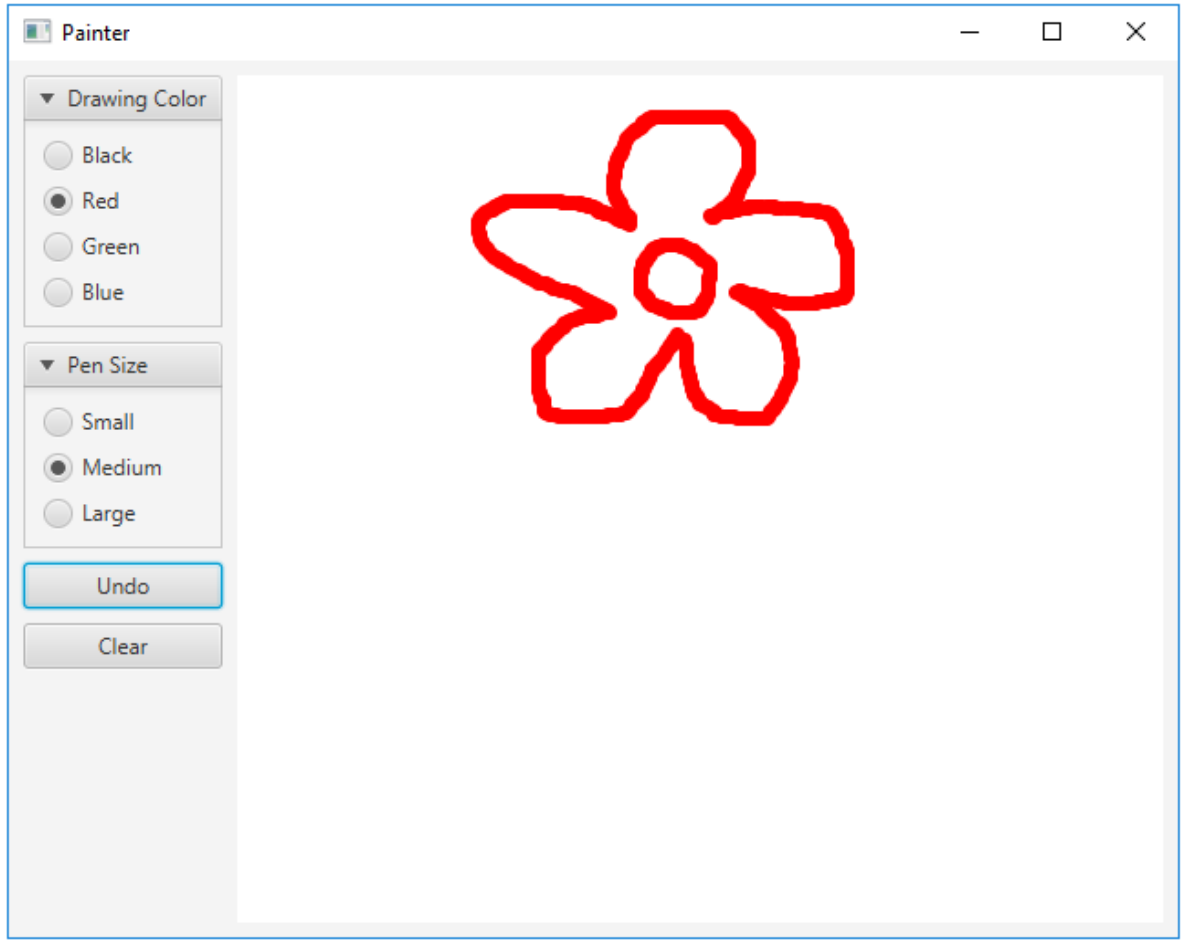


Fig. 1.12 | Drawing the flower petals.

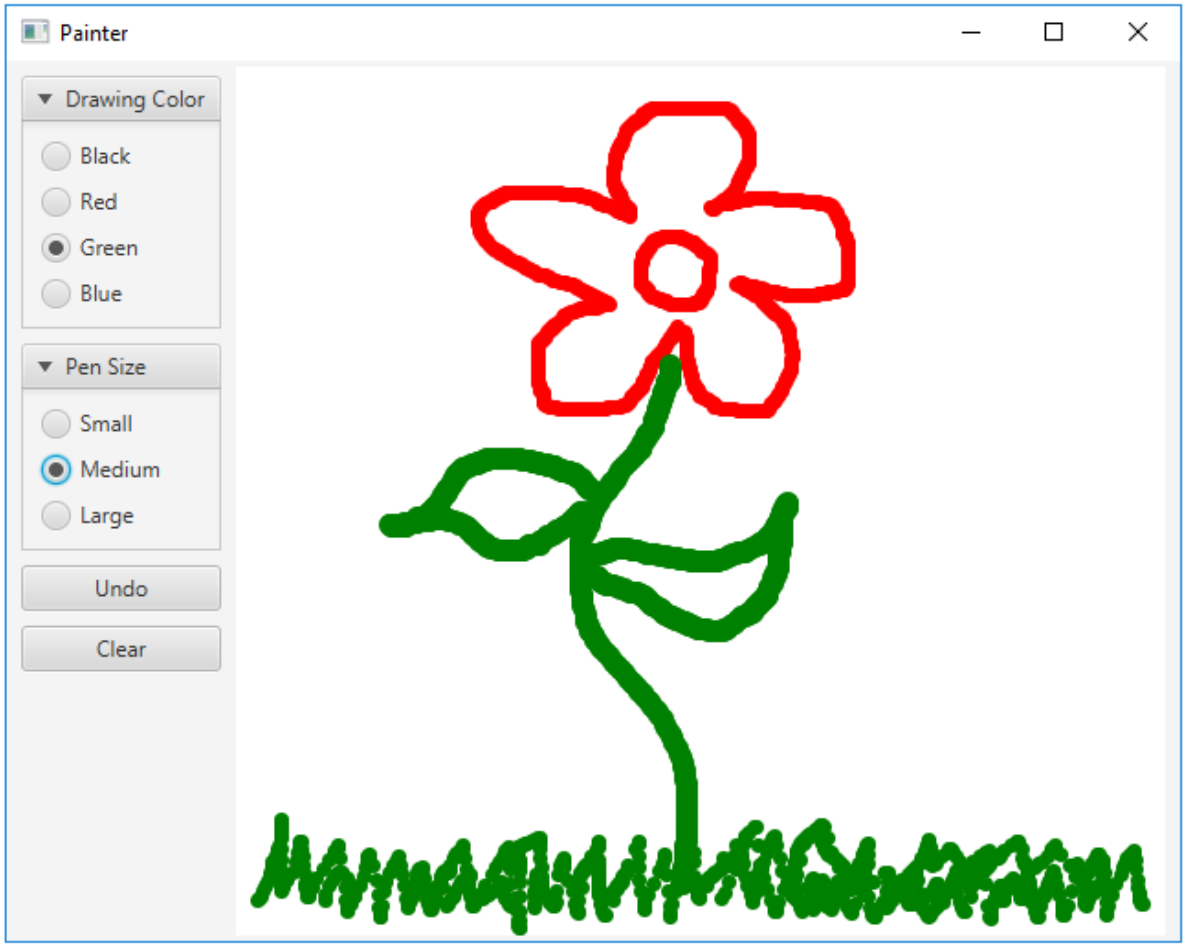


Fig. 1.13 | Drawing the stem and grass.

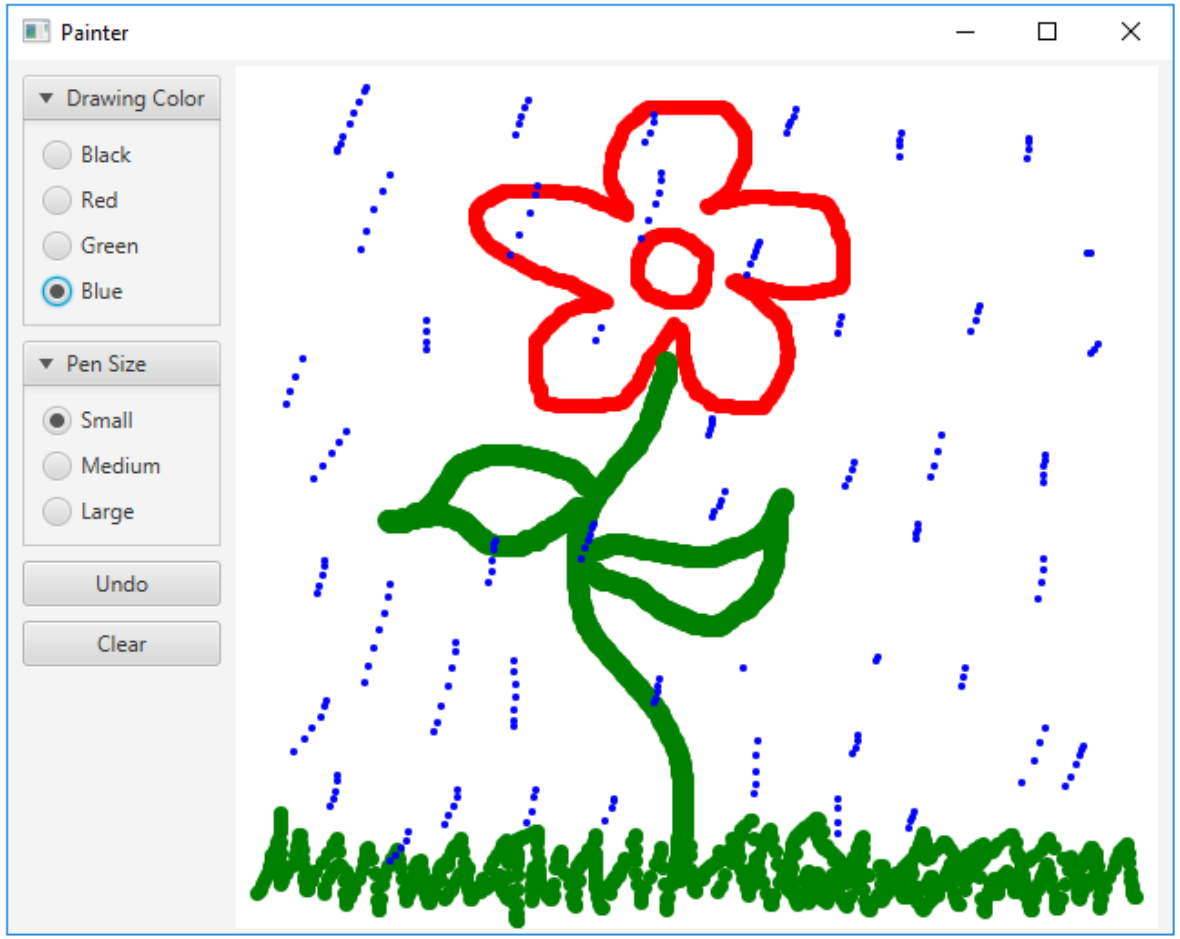


Fig. 1.14 | Drawing the rain.

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 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.
- Each Internet-connected- device has an **IP address**—a unique numerical identifier used by devices communicating via TCP/IP to locate one another on the Internet.

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 World Wide Web: Making the Internet User-Friendly

- The World Wide Web (simply called “the web”) is a collection of hardware and software associated with the Internet that allows computer users to locate and view documents with various combinations of text, graphics, animations, audios and videos on almost any subject.

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 developing **HyperText Markup Language (HTML)**—the technology for sharing information via "hyperlinked" text documents
- He also wrote communication protocols such as HyperText Transfer Protocol (HTTP) to form the backbone of his new hypertext information system, which he referred to as the World Wide Web.

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

- A 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

Fig. 1.15 | Some popular web services (`https://www.programmableweb.com/category/all/apis`). (Part 1 of 2.)

Web services source	How it's used
Wikipedia	Collaborative encyclopedia
PayPal	Payments
Last.fm	Internet radio
Amazon eCommerce	Shopping for books and many other products
Salesforce.com	Customer Relationship Management (CRM)
Skype	Internet telephony
Microsoft Bing	Search
Flickr	Photo sharing
Zillow	Real-estate pricing
Yahoo Search	Search
WeatherBug	Weather

Fig. 1.15 | Some popular web services (`https://www.programmableweb.com/category/all/apis`). (Part 2 of 2.)

1.11.4 Internet of Things

- The Internet is no longer just a network of computers—it's an Internet of Things (IoT)
- A *thing* is any object with an IP address and the ability to send data automatically over the Internet:
  - a car with a transponder for paying tolls,
  - monitors for parking-space availability in a garage,
  - a heart monitor implanted in a human,
  - monitors for drinkable water quality,
  - a smart meter that reports energy usage,
  - radiation detectors,
  - item trackers in a warehouse,
  - mobile apps that can track your movement and location,
  - smart thermostats that adjust room temperatures based on weather forecasts and activity in the home
  - intelligent home appliances
  - and many more.

1.11.4 Internet of Things (cont.)

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

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.

Fig. 1.16 | Software technologies. (Part 1 of 5.)

Technology	Description
Design patterns	**Design patterns** are proven architectures for constructing flexible and maintainable object-oriented software. The field of design patterns tries to enumerate those recurring patterns, encouraging software designers to *reuse* them to develop better-quality software using less time, money and effort (see online Appendix N, Design Patterns).
LAMP	**LAMP** is an acronym for the open-source technologies that many developers use to build web applications inexpensively—it stands for *Linux*, *Apache*, *MySQL* and *PHP* (or *Perl* or *Python*—two other popular scripting languages). MySQL is an open-source database-management system. PHP is a popular open-source server-side "scripting" language for developing web applications. Apache is the most popular web server software. The equivalent for Windows development is WAMP—*Windows*, *Apache*, *MySQL* and *PHP*.

Fig. 1.16 | Software technologies. (Part 2 of 5.)

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, Microsoft and many other companies offer SaaS.

Fig. 1.16 | Software technologies. (Part 3 of 5.)

Technology	Description
Platform as a Service (PaaS)	**Platform as a Service (PaaS)** provides a computing platform for developing and running applications as a service over the web, rather than installing the tools on your computer. Some PaaS providers are Google App Engine, Amazon EC2 and Windows Azure™.
Cloud computing	SaaS and PaaS are examples of cloud computing. You can use software and data stored in the "cloud"—i.e., accessed on remote computers (or servers) via the Internet and available on demand—rather than having it stored locally on your desktop, notebook computer or mobile device. This allows you to increase or decrease computing resources to meet your needs at any given time, which is more cost effective than purchasing hardware to provide enough storage and processing power to meet occasional peak demands. Cloud computing also saves money by shifting to the service provider the burden of managing these apps (such as installing and upgrading the software, security, backups and disaster recovery).

Fig. 1.16 | Software technologies. (Part 4 of 5.)

Technology	Description
Software Development Kit (SDK)	**Software Development Kits (SDKs)** include the tools and documentation developers use to program applications.

Fig. 1.16 | Software technologies. (Part 5 of 5.)

1.13 Keeping Up-to-Date with Information 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. Alpha software also is commonly called *early access* software.
Beta	*Beta* versions are released to a larger number of developers later in the development process after most major bugs have been fixed and new features are nearly complete. Beta software is more stable, but still subject to change.
Release candidates	*Release candidates* are generally *feature complete*, (mostly) bug free and ready for use by the community, which provides a diverse testing environment—the software is used on different systems, with varying constraints and for a variety of purposes.

Fig. 1.17 | Software product-release terminology. (Part 1 of 2.)

Version	Description
Final release	Any bugs that appear in the release candidate are corrected, and eventually the final product is released to the general public. Software companies often distribute incremental updates over the Internet.
Continuous beta	Software that's developed using this approach (for example, Google search or Gmail) generally does not have version numbers. It's hosted in the *cloud* (not installed on your computer) and is constantly evolving so that users always have the latest version.

Fig. 1.17 | Software product-release terminology. (Part 2 of 2.)

1.14 Getting Your Questions Answered

- There are many online forums in which you can get your Java questions answered and interact with other Java programmers.
- Some popular Java and general programming forums include:
  - StackOverflow.com
  - Coderanch.com
  - The Oracle Java Forum—https://community.oracle.com/community/java
  - </dream.in.code>—http://www.dreamincode.net/forums/forum/32-java/