Chapter 1 Introduction to Computers, Programs, and Java™

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

2. A computer includes both hardware and software.

3. Hardware is the physical aspect of the computer that can be touched.

4. Computer programs, known as software, are the invisible instructions that control the hardware and make it perform tasks.

5. Computer programming is the writing of instructions (i.e., code) for computers to perform.

6. The central processing unit (CPU) is a computer’s brain. It retrieves instructions from memory and executes them.

7. Computers use zeros and ones because digital devices have two stable states, referred to by convention as zero and one.

8. A bit is a binary digit 0 or 1.

9. A byte is a sequence of 8 bits.

10. A kilobyte is about 1,000 bytes, a megabyte about 1 million bytes, a gigabyte about 1 billion bytes, and a terabyte about 1,000 gigabytes.

11. Memory stores data and program instructions for the CPU to execute.

12. A memory unit is an ordered sequence of bytes.

13. Memory is volatile, because information is lost when the power is turned off.

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

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

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

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

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

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

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

21. Java is platform independent, meaning you can write a program once and run it on any computer.

22. The Java source file name must match the public class name in the program. Java source-code files must end with the .java extension.

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

24. To compile a Java source-code file from the command line, use the javac command. 25. To run a Java class from the command line, use the java command.

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

27. A block begins with an opening brace ({) and ends with a closing brace (}).

28. Methods are contained in a class. To run a Java program, the program must have a main method. The main method is the entry point where the program starts when it is executed.

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

30. Keywords have a specific meaning to the compiler and cannot be used for other purposes in the program.

31. In Java, comments are preceded by two slashes (//) on a line, called a line comment, or enclosed between /\* and \*/ on one or several lines, called a block comment or paragraph comment. Comments are ignored by the compiler.

32. Java source programs are case sensitive.

33. Programming errors can be categorized into three types: syntax errors, runtime

errors, and logic errors. Errors reported by a compiler are called syntax errors

or compile errors. Runtime errors are errors that cause a program to terminate abnormally. Logic errors occur when a program does not perform the way it was intended to.

Chapter 2 Elementary Programming

1. Identifiers are names for naming elements such as variables, constants, methods, classes, and packages in a program.

2. An identifier is a sequence of characters that consists of letters, digits, underscores (\_), and dollar signs ($). An identifier must start with a letter or an underscore. It cannot start with a digit. An identifier cannot be a reserved word. An identifier can be of any length.

3. Variables are used to store data in a program. To declare a variable is to tell the compiler what type of data a variable can hold.

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

5. In Java, the equal sign (=) is used as the assignment operator.

6. A variable declared in a method must be assigned a value before it can be used.

7. A named constant (or simply a constant) represents permanent data that never changes.

8. A named constant is declared by using the keyword final.

9. Java provides four integer types (byte, short, int, and long) that represent integers of four different sizes.

10. Java provides two floating-point types (float and double) that represent floating-point numbers of two different precisions.

11. Java provides operators that perform numeric operations: + (addition), – (subtraction), \* (multiplication), / (division), and % (remainder).

12. Integer arithmetic (/) yields an integer result.

13. The numeric operators in a Java expression are applied the same way as in an arithmetic expression.

14. Java provides the augmented assignment operators += (addition assignment), –= (subtraction assignment), \*= (multiplication assignment), /= (division assignment), and %= (remainder assignment).

15. The increment operator (++) and the decrement operator (––) increment or decrement a variable by 1.

16. When evaluating an expression with values of mixed types, Java automatically converts the operands to appropriate types.

17. You can explicitly convert a value from one type to another using the (type)value notation.

18. Casting a variable of a type with a small range to a type with a larger range is known as widening a type.

19. Casting a variable of a type with a large range to a type with a smaller range is known as narrowing a type.

20. Widening a type can be performed automatically without explicit casting. Narrowing a type must be performed explicitly.

21. In computer science, midnight of January 1, 1970, is known as the UNIX epoch.

Chapter 3 Selections

1. A boolean-type variable can store a true or false value.

2. The relational operators (<, <=, ==, !=, >, and >=) yield a Boolean value.

3. Selection statements are used for programming with alternative courses of actions. There are several types of selection statements: one-way if statements, two-way ifelse statements, nested if statements, multi-way if-else statements, switch statements, and conditional operators.

4. The various if statements all make control decisions based on a Boolean expression. Based on the true or false evaluation of the expression, these statements take one of the two possible courses.

5. The Boolean operators &&, ||, !, and ^ operate with Boolean values and variables. 6. When evaluating p1 && p2, Java first evaluates p1 then evaluates p2 if p1 is true; if p1 is false, it does not evaluate p2. When evaluating p1 || p2, Java first evaluates p1 then evaluates p2 if p1 is false; if p1 is true, it does not evaluate p2. Therefore, && is referred to as the short-circuit or lazy AND operator, and || is referred to as the short-circuit or lazy OR operator.

7. The switch statement makes control decisions based on a switch expression of type char, byte, short, int, or String.

8. The keyword break is optional in a switch statement, but it is normally used at the end of each case in order to skip the remainder of the switch statement. If the break statement is not present, the next case statement will be executed.

9. The operators in expressions are evaluated in the order determined by the rules of parentheses, operator precedence, and operator associativity.

10. Parentheses can be used to force the order of evaluation to occur in any sequence. 11. Operators with higher precedence are evaluated earlier. For operators of the same precedence, their associativity determines the order of evaluation.

12. All binary operators except assignment operators are left associative; assignment operators are right associative.

Chapter 4 Mathematical Functions, Characters, and String

1. Java provides the mathematical methods sin, cos, tan, asin, acos, atan, toRadians, toDegrees, exp, log, log10, pow, sqrt, ceil, floor, rint, round, min, max, abs, and random in the Math class for performing mathematical functions.

2. The character type char represents a single character.

3. An escape sequence consists of a backslash (\) followed by a character or a combination of digits.

4. The character \ is called the escape character.

5. The characters ' ', \t, \f, \r, and \n are known as the whitespace characters.

6. Characters can be compared based on their Unicode using the relational operators.

7. The Character class contains the methods isDigit, isLetter, isLetterOrDigit, isLowerCase, and isUpperCase for testing whether a character is a digit, letter, lowercase, or uppercase. It also contains the toLowerCase and toUpperCase methods for returning a lowercase or uppercase letter.

8. A string is a sequence of characters. A string value is enclosed in matching double quotes ("). A character value is enclosed in matching single quotes (').

9. Strings are objects in Java. A method that can only be invoked from a specific object is called an instance method. A noninstance method is called a static method, which can be invoked without using an object.

10. You can get the length of a string by invoking its length() method, retrieve a character at the specified index in the string using the charAt(index) method, and use the indexOf and lastIndexOf methods to find a character or a substring in a string.

11. You can use the concat method to concatenate two strings or the plus (+) operator to concatenate two or more strings.

12. You can use the substring method to obtain a substring from the string.

13. You can use the equals and compareTo methods to compare strings. The equals method returns true if two strings are equal, and false if they are not equal. The compareTo method returns 0, a positive integer, or a negative integer, depending on whether one string is equal to, greater than, or less than the other string.

14. The printf method can be used to display a formatted output using format specifiers.

Chapter 5 Loops

1. There are three types of repetition statements: the while loop, the do-while loop, and the for loop.

2. The part of the loop that contains the statements to be repeated is called the loop body.

3. A one-time execution of a loop body is referred to as an iteration of the loop.

4. An infinite loop is a loop statement that executes infinitely.

5. In designing loops, you need to consider both the loop control structure and the loop body.

6. The while loop checks the loop-continuation-condition first. If the condition is true, the loop body is executed; if it is false, the loop terminates.

7. The do-while loop is similar to the while loop, except the do-while loop executes the loop body first then checks the loop-continuation-condition to decide whether to continue or to terminate.

8. The while loop and the do-while loop often are used when the number of repetitions is not predetermined.

9. A sentinel value is a special value that signifies the end of the loop.

10. The for loop generally is used to execute a loop body a fixed number of times.

11. The for loop control has three parts. The first part is an initial action that often initializes a control variable. The second part, the loop-continuation-condition, determines whether the loop body is to be executed. The third part is executed after each iteration and is often used to adjust the control variable. Usually, the loop control variables are initialized and changed in the control structure.

12. The while loop and for loop are called pretest loops because the continuation condition is checked before the loop body is executed.

13. The do-while loop is called a posttest loop because the condition is checked after the loop body is executed.

14. Two keywords break and continue can be used in a loop.

15. The break keyword immediately ends the innermost loop, which contains the break. 16. The continue keyword only ends the current iteration.

Chapter 6 Methods

1. Making programs modular and reusable is one of the central goals in software engineering. Java provides many powerful constructs that help to achieve this goal. Methods are one such construct.

2. The method header specifies the modifiers, return value type, method name, and parameters of the method. The static modifier is used for all the methods in this chapter.

3. A method may return a value. The returnValueType is the data type of the value the method returns. If the method does not return a value, the returnValueType is the keyword void.

4. The parameter list refers to the type, order, and number of a method’s parameters. The method name and the parameter list together constitute the method signature. Parameters are optional; that is, a method doesn’t need to contain any parameters.

5. A return statement can also be used in a void method for terminating the method and returning to the method’s caller. This is useful occasionally for circumventing the normal flow of control in a method.

6. The arguments that are passed to a method should have the same number, type, and order as the parameters in the method signature.

7. When a program calls a method, program control is transferred to the called method. A called method returns control to the caller when its return statement is executed, or when its method-ending closing brace is reached.

8. A value-returning method can also be invoked as a statement in Java. In this case, the caller simply ignores the return value.

9. A method can be overloaded. This means that two methods can have the same name, as long as their method parameter lists differ.

10. A variable declared in a method is called a local variable. The scope of a local variable starts from its declaration and continues to the end of the block that contains the variable. A local variable must be declared and initialized before it is used.

11. Method abstraction is achieved by separating the use of a method from its implementation. The client can use a method without knowing how it is implemented. The details of the implementation are encapsulated in the method and hidden from the client who invokes the method. This is known as information hiding or encapsulation.

12. Method abstraction modularizes programs in a neat, hierarchical manner. Programs written as collections of concise methods are easier to write, debug, maintain, and modify than would otherwise be the case. This writing style also promotes method reusability.

13. When implementing a large program, use the top-down and/or bottom-up coding approach. Do not write the entire program at once. This approach may seem to take more time for coding (because you are repeatedly compiling and running the program), but it actually saves time and makes debugging easier.

Chapter 7 Single-Dimensional Arrays

1. A variable is declared as an array type using the syntax elementType[] arrayRefVar or elementType arrayRefVar[]. The style elementType[] arrayRefVar is preferred, although elementType arrayRefVar[] is legal.

2. Unlike declarations for primitive data type variables, the declaration of an array variable does not allocate any space in memory for the array. An array variable is not a primitive data type variable. An array variable contains a reference to an array.

3. You cannot assign elements to an array unless it has already been created. You can create an array by using the new operator with the following syntax: new elementType[arraySize].

4. Each element in the array is represented using the syntax arrayRefVar[index]. An index must be an integer or an integer expression.

5. After an array is created, its size becomes permanent and can be obtained using arrayRefVar.length. Since the index of an array always begins with 0, the last index is always arrayRefVar.length − 1. An out-of-bounds error will occur if you attempt to reference elements beyond the bounds of an array.

6. Programmers often mistakenly reference the first element in an array with index 1, but it should be 0. This is called the index off-by-one error.

7. When an array is created, its elements are assigned the default value of 0 for the numeric primitive data types, \u0000 for char types, and false for boolean types.

8. Java has a shorthand notation, known as the array initializer, which combines declaring an array, creating an array, and initializing an array in one statement, using the syntax elementType[] arrayRefVar = {value0, value1, . . . , valuek}.

9. When you pass an array argument to a method, you are actually passing the reference of the array; that is, the called method can modify the elements in the caller’s original array.

10. If an array is sorted, binary search is more efficient than linear search for finding an element in the array.

11. Selection sort finds the smallest number in the list and swaps it with the first element. It then finds the smallest number remaining and swaps it with the first element in the remaining list, and so on, until only a single number remains.

Chapter 8 Multidimensional Arrays

1. A two-dimensional array can be used to store a table.

2. A variable for two-dimensional arrays can be declared using the syntax: elementType[][]

arrayVar.

3. A two-dimensional array can be created using the syntax: new elementType[ROW\_SIZE] [COLUMN\_SIZE].

4. Each element in a two-dimensional array is represented using the syntax: arrayVar[rowIndex][columnIndex].

5. You can create and initialize a two-dimensional array using an array initializer with the syntax: elementType[][] arrayVar = {{row values}, . . ., {row values}}.

6. You can use arrays of arrays to form multidimensional arrays. For example, a variable for three-dimensional arrays can be declared as elementType[][][] arrayVar and a three-dimensional array can be created using new elementType[size1][size2] [size3].

Chapter 9 Objects and Classes

1. A class is a template for objects. It defines the properties of objects and provides constructors for creating objects and methods for manipulating them.

2. A class is also a data type. You can use it to declare object reference variables. An object reference variable that appears to hold an object actually contains a reference to that object. Strictly speaking, an object reference variable and an object are different, but most of the time the distinction can be ignored.

3. An object is an instance of a class. You use the new operator to create an object and the dot operator (.) to access members of that object through its reference variable.

4. An instance variable or method belongs to an instance of a class. Its use is associated with individual instances. A static variable is a variable shared by all instances of the same class. A static method is a method that can be invoked without using instances.

5. Every instance of a class can access the class’s static variables and methods. For clarity, however, it is better to invoke static variables and methods using ClassName . variable and ClassName.method.

6. Visibility modifiers specify how the class, method, and data are accessed. A public class, method, or data is accessible to all clients. A private method or data is accessible only inside the class.

7. You can provide a getter (accessor) method or a setter (mutator) method to enable clients to see or modify the data.

8. A getter method has the signature public returnType getPropertyName(). If the returnType is boolean, the getter method should be defined as public boolean isPropertyName(). A setter method has the signature public void setPropertyName(dataType propertyValue).

9. All parameters are passed to methods using pass-by-value. For a parameter of a primitive type, the actual value is passed; for a parameter of a reference type, the reference for the object is passed.

10. A Java array is an object that can contain primitive-type values or object-type values. When an array of objects is created, its elements are assigned the default value of null.

11. Once it is created, an immutable object cannot be modified. To prevent users from modifying an object, you can define immutable classes.

12. The scope of instance and static variables is the entire class, regardless of where the variables are declared. Instance and static variables can be declared anywhere in the class. For consistency, they are declared at the beginning of the class in this book.

13. The keyword this can be used to refer to the calling object. It can also be used inside a constructor to invoke another constructor of the same class.

Chapter 10 Object-Oriented Thinking

1. The procedural paradigm focuses on designing methods. The object-oriented paradigm couples data and methods together into objects. Software design using the object-oriented paradigm focuses on objects and operations on objects. The object-oriented approach combines the power of the procedural paradigm with an added dimension that integrates data with operations into objects.

2. Many Java methods require the use of objects as arguments. Java offers a convenient way to incorporate, or wrap, a primitive data type into an object (e.g., wrapping int into the Integer class, and wrapping double into the Double class).

3. Java can automatically convert a primitive-type value to its corresponding wrapper object in the context and vice versa.

4. The BigInteger class is useful for computing and processing integers of any size. The BigDecimal class can be used to compute and process floating-point numbers with any arbitrary precision.

5. A String object is immutable; its contents cannot be changed. To improve efficiency and save memory, the JVM stores two literal strings that have the same character sequence in a unique object. This unique object is called an interned string object.

6. A regular expression (abbreviated regex) is a string that describes a pattern for matching a set of strings. You can match, replace, or split a string by specifying a pattern.

7. The StringBuilder and StringBuffer classes can be used to replace the String class. The String object is immutable, but you can add, insert, or append new contents into StringBuilder and StringBuffer objects. Use String if the string contents do not require any change and use StringBuilder or StringBuffer if they might change.

Chapter 11 Inheritance and Polymorphism

1. You can define a new class from an existing class. This is known as class inheritance. The new class is called a subclass, child class, or extended class. The existing class is called a superclass, parent class, or base class.

2. A constructor is used to construct an instance of a class. Unlike properties and methods, the constructors of a superclass are not inherited in the subclass. They can be invoked only from the constructors of the subclasses, using the keyword super.

3. A constructor may invoke an overloaded constructor or its superclass’s constructor. The call must be the first statement in the constructor. If none of them is invoked explicitly, the compiler puts super() as the first statement in the constructor, which invokes the superclass’s no-arg constructor.

4. To override a method, the method must be defined in the subclass using the same signature and the same or compatible return type as in its superclass.

5. An instance method can be overridden only if it is accessible. Thus, a private method cannot be overridden because it is not accessible outside its own class. If a method defined in a subclass is private in its superclass, the two methods are completely unrelated.

6. Like an instance method, a static method can be inherited. However, a static method cannot be overridden. If a static method defined in the superclass is redefined in a subclass, the method defined in the superclass is hidden.

7. Every class in Java is descended from the java.lang.Object class. If no superclass is specified when a class is defined, its superclass is Object.

8. If a method’s parameter type is a superclass (e.g., Object), you may pass an object to this method of any of the parameter’s subclasses (e.g., Circle or String). This is known as polymorphism.

9. It is always possible to cast an instance of a subclass to a variable of a superclass because an instance of a subclass is always an instance of its superclass. When casting an instance of a superclass to a variable of its subclass, explicit casting must be used to confirm your intention to the compiler with the (SubclassName) cast notation.

10. A class defines a type. A type defined by a subclass is called a subtype, and a type defined by its superclass is called a supertype.

11. When invoking an instance method from a reference variable, the actual type of the variable decides which implementation of the method is used at runtime. This is known as dynamic binding.

12. You can use obj instanceof AClass to test whether an object is an instance of a class.

13. You can use the ArrayList class to create an object to store a list of objects.

14. You can use the protected modifier to prevent the data and methods from being accessed by nonsubclasses from a different package.

15. You can use the final modifier to indicate a class is final and cannot be extended and to indicate a method is final and cannot be overridden.

Chapter 12 Exception Handling and Text I/O

1. Exception handling enables a method to throw an exception to its caller.

2. A Java exception is an instance of a class derived from java.lang.Throwable. Java provides a number of predefined exception classes, such as Error, Exception, RuntimeException, ClassNotFoundException, NullPointerException, and ArithmeticException. You can also define your own exception class by extending Exception.

3. Exceptions occur during the execution of a method. RuntimeException and Error are unchecked exceptions; all other exceptions are checked.

4. When declaring a method, you have to declare a checked exception if the method might throw it, thus telling the compiler what can go wrong.

5. The keyword for declaring an exception is throws, and the keyword for throwing an exception is throw.

6. To invoke the method that declares checked exceptions, enclose it in a try statement. When an exception occurs during the execution of the method, the catch block catches and handles the exception.

7. If an exception is not caught in the current method, it is passed to its caller. The process is repeated until the exception is caught or passed to the main method.

8. Various exception classes can be derived from a common superclass. If a catch block catches the exception objects of a superclass, it can also catch all the exception objects of the subclasses of that superclass.

9. The order in which exceptions are specified in a catch block is important. A compile error will result if you specify an exception object of a class after an exception object of the superclass of that class.

10. When an exception occurs in a method, the method exits immediately if it does not catch the exception. If the method is required to perform some task before exiting, you can catch the exception in the method and then rethrow it to its caller.

11. The code in the finally block is executed under all circumstances, regardless of whether an exception occurs in the try block, or whether an exception is caught if it occurs.

12. Exception handling separates error-handling code from normal programming tasks, thus making programs easier to read and to modify.

13. Exception handling should not be used to replace simple tests. You should perform simple test using if statements whenever possible and reserve exception handling for dealing with situations that cannot be handled with if statements.

14. The File class is used to obtain file properties and manipulate files. It does not contain the methods for creating a file or for reading/writing data from/to a file.

15. You can use Scanner to read string and primitive data values from a text file and use PrintWriter to create a file and write data to a text file. 16. You can read from a file on the Web using the URL class

Chapter 13 Abstract Classes and Interfaces

1. Abstract classes are like regular classes with data and methods, but you cannot create instances of abstract classes using the new operator.

2. An abstract method cannot be contained in a nonabstract class. If a subclass of an abstract superclass does not implement all the inherited abstract methods of the superclass, the subclass must be defined as abstract.

3. A class that contains abstract methods must be abstract. However, it is possible to define an abstract class that doesn’t contain any abstract methods.

4. A subclass can be abstract even if its superclass is concrete.

5. An interface is a class-like construct that contains only constants, abstract methods, default methods, and static methods. In many ways, an interface is similar to an abstract class, but an abstract class can contain data fields.

6. An interface is treated like a special class in Java. Each interface is compiled into a separate bytecode file, just like a regular class.

7. The java.lang.Comparable interface defines the compareTo method. Many classes in the Java library implement Comparable.

8. The java.lang.Cloneable interface is a marker interface. An object of the class that implements the Cloneable interface is cloneable.

9. A class can extend only one superclass but can implement one or more interfaces.

10. An interface can extend one or more interfaces.

Chapter 14 JavaFX Basics

1. JavaFX is the new framework for developing rich GUI applications. JavaFX completely replaces Swing and AWT.

2. A main JavaFX class must extend javafx.application.Application and implement the start method. The primary stage is automatically created by the JVM and passed to the start method.

3. A stage is a window for displaying a scene. You can add nodes to a scene. Panes, groups, controls, and shapes are nodes. Panes can be used as the containers for nodes.

4. A binding property can be bound to an observable source object. A change in the source object will be automatically reflected in the binding property. A binding property has a value getter method, value setter method, and property getter method.

5. The Node class defines many properties that are common to all nodes. You can apply these properties to panes, groups, controls, and shapes.

6. You can create a Color object with the specified red, green, blue components, and opacity value.

7. You can create a Font object and set its name, size, weight, and posture. 8. The javafx.scene.image.Image class can be used to load an image, and this image can be displayed in an ImageView object.

9. JavaFX provides many types of panes for automatically laying out nodes in a desired location and size. The Pane is the base class for all panes. It contains the getChildren() method to return an ObservableList. You can use

ObservableList’s add(node) and addAll(node1, node2,...) methods for adding nodes into a pane.

10. A FlowPane arranges the nodes in the pane horizontally from left to right or vertically from top to bottom, in the order in which they were added. A GridPane arranges nodes

in a grid (matrix) formation. The nodes are placed in the specified column and row indices. A BorderPane can place nodes in five regions: top, bottom, left, right, and center. An HBox lays out its children in a single horizontal row. A VBox lays out its children in a single vertical column.

11. JavaFX provides many shape classes for drawing texts, lines, circles, rectangles, ellipses, arcs, polygons, and polylines

Chapter 15 Event-Driven Programming and Animations

1. The root class of the JavaFX event classes is javafx.event.Event, which is a subclass of java.util.EventObject. The subclasses of Event deal with special types of events, such as action events, window events, mouse events, and key events. If a node can fire an event, any subclass of the node can fire the same type of event.

2. The handler object’s class must implement the corresponding event–handler interface. JavaFX provides a handler interface EventHandler<T extends Event> for every event class T. The handler interface contains the handle(T e) method for handling event e.

3. The handler object must be registered by the source object. Registration methods depend on the event type. For an action event, the method is setOnAction. For a mouse-pressed event, the method is setOnMousePressed. For a key-pressed event, the method is setOnKeyPressed.

4. An inner class, or nested class, is defined within the scope of another class. An inner class can reference the data and methods defined in the outer class in which it nests, so you need not pass the reference of the outer class to the constructor of the inner class.

5. An anonymous inner class can be used to shorten the code for event handling.

Furthermore, a lambda expression can be used to greatly simplify the event-handling code for functional interface handlers.

6. A functional interface is an interface with exactly one abstract method. This is also known as a single abstract method (SAM) interface.

7. A MouseEvent is fired whenever a mouse button is pressed, released, clicked, moved, or dragged on a node or a scene. The getButton() method can be used to detect which mouse button is pressed for the event.

8. A KeyEvent is fired whenever a key is pressed, released, or typed on a node or a scene. The getCode() method can be used to return the code value for the key.

9. An instance of Observable is known as an observable object, which contains the add-Listener(InvalidationListener listener) method for adding a listener. Once the value is changed in the property, a listener is notified. The listener class should implement the InvalidationListener interface, which uses the invalidated method to handle the property value change.

10. The abstract Animation class provides the core functionalities for animations in JavaFX. PathTransition, FadeTransition, and Timeline are specialized classes for implementing animations

Chapter 16 JavaFX UI Controls and Multimedia

1. The abstract Labeled class is the base class for Label, Button, CheckBox, and RadioButton. It defines properties alignment, contentDisplay, text, graphic, graphicTextGap, textFill, underline, and wrapText.

2. The abstract ButtonBase class is the base class for Button, CheckBox, and

RadioButton. It defines the onAction property for specifying a handler for action events.

3. The abstract TextInputContorl class is the base class for TextField and TextArea. It defines the properties text and editable.

4. A TextField fires an action event when clicking the Enter key with the text field focused. A TextArea is often used for editing a multiline text.

5. ComboBox<T> and ListView<T> are generic classes for storing elements of type T. The elements in a combo box or a list view are stored in an observable list.

6. A ComboBox fires an action event when a new item is selected. 7. You can set a single item or multiple items selection for a ListView and add a listener for processing selected items.

8. You can use a ScrollBar or Slider to select a range of values and add a listener to the value property to respond to the change of the value.

9. JavaFX provides the Media class for loading a media, the MediaPlayer class for controlling a media, and the MediaView for displaying a media.

Chapter 17 Binary I/O

1. I/O can be classified into text I/O and binary I/O. Text I/O interprets data in sequences of characters. Binary I/O interprets data as raw binary values. How text is stored in a file depends on the encoding scheme for the file. Java automatically performs encoding and decoding for text I/O.

2. The InputStream and OutputStream classes are the roots of all binary I/O classes. FileInputStream/FileOutputStream associates a file for input/output. Buffered InputStream/BufferedOutputStream can be used to wrap any binary I/O stream to improve performance. DataInputStream/DataOutputStream can be used to read/ write primitive values and strings.

3. ObjectInputStream/ObjectOutputStream can be used to read/write objects in addition to primitive values and strings. To enable object serialization, the object’s defining class must implement the java.io.Serializable marker interface.

4. The RandomAccessFile class enables you to read and write data to a file. You can open a file with the r mode to indicate that it is read-only, or with the rw mode to indicate that it is updateable. Since the RandomAccessFile class implements DataInput and DataOutput interfaces, many methods in RandomAccessFile are the same as those in DataInputStream and DataOutputStream.

Chapter 18 Recursion

1. A recursive method is one that directly or indirectly invokes itself. For a recursive method to terminate, there must be one or more base cases.

2. Recursion is an alternative form of program control. It is essentially repetition without a loop control. It can be used to write simple, clear solutions for inherently recursive problems that would otherwise be difficult to solve.

3. Sometimes the original method needs to be modified to receive additional parameters in order to be invoked recursively. A recursive helper method can be defined for this purpose.

4. Recursion bears substantial overhead. Each time the program calls a method, the system must allocate memory for all of the method’s local variables and parameters. This can consume considerable memory and requires extra time to manage the memory.

5. A recursive method is said to be tail recursive if there are no pending operations to be performed on return from a recursive call. Some compilers can optimize tail recursion to reduce stack size.

Chapter 19 Generics

1. Generics give you the capability to parameterize types. You can define a class or a method with generic types, which are substituted with concrete types.

2. The key benefit of generics is to enable errors to be detected at compile time rather than at runtime.

3. A generic class or method permits you to specify allowable types of objects that the class or method can work with. If you attempt to use a class or method with an incompatible object, the compiler will detect the error.

4. A generic type defined in a class, interface, or a static method is called a formal generic type, which can be replaced later with an actual concrete type. Replacing a generic type is called a generic instantiation.

5. A generic class such as ArrayList used without a type parameter is called a raw type. Use of raw types allows for backward compatibility with the earlier versions of Java.

6. A wildcard generic type has three forms:

? , ? extends T , and ? super T ,

where T is a generic type. The first form, ?, called an unbounded wildcard, is the same as ? extends Object. The second form, ? extends T, called a bounded wildcard, represents T or a subtype of T. The third form, ? super T, called a lower bound wildcard, denotes T or a supertype of T.

7. Generics are implemented using an approach called type erasure. The compiler uses the generic-type information to compile the code but erases it afterward, so the generic information is not available at runtime. This approach enables the generic code to be backward compatible with the legacy code that uses raw types.

8. You cannot create an instance using a generic-type parameter such as new E().

9. You cannot create an array using a generic-type parameter such as new E[10].

10. You cannot use a generic-type parameter of a class in a static context.

11. Generic-type parameters cannot be used in exception classes.

Chapter 20 Lists, Stacks, Queues, and Priority Queues

1. The Collection interface defines the common operations for lists, vectors, stacks, queues, priority queues, and sets.

2. Each collection is Iterable. You can obtain its Iterator object to traverse all the elements in the collection.

3. All the concrete classes except PriorityQueue in the Java Collections Framework implement the Cloneable and Serializable interfaces. Thus, their instances can be cloned and serialized.

4. A list stores an ordered collection of elements. To allow duplicate elements to be stored in a collection, you need to use a list. A list not only can store duplicate elements but also allows the user to specify where they are stored. The user can access elements by an index.

5. Two types of lists are supported: ArrayList and LinkedList. ArrayList is a

resizable-array implementation of the List interface. All the methods in ArrayList are defined in List. LinkedList is a linked-list implementation of the List interface. In addition to implementing the List interface, this class provides the methods for retrieving, inserting, and removing elements from both ends of the list.

6. Comparator can be used to compare the objects of a class that doesn’t implement Comparable.

7. The Vector class extends the AbstractList class. Starting with Java 2, Vector has been the same as ArrayList, except that the methods for accessing and modifying the vector are synchronized. The Stack class extends the Vector class and provides several methods for manipulating the stack.

8. The Queue interface represents a queue. The PriorityQueue class implements Queue for a priority queue.

Chapter 21 Sets and Maps

1. A set stores nonduplicate elements. To allow duplicate elements to be stored in a

collection, you need to use a list.

2. A map stores key/value pairs. It provides a quick lookup for a value using a key.

3. Three types of sets are supported: HashSet, LinkedHashSet, and TreeSet. HashSet stores elements in an unpredictable order. LinkedHashSet stores elements in the order they were inserted. TreeSet stores elements sorted. HashSet, LinkedHashSet, and TreeSet are subtypes of Collection.

4. The Map interface maps keys to the elements. The keys are like indexes. In List, the indexes are integers. In Map, the keys can be any objects. A map cannot contain duplicate keys. Each key can map to at most one value. The Map interface provides the methods for querying, updating, and obtaining a collection of values and a set of keys.

5. Three types of maps are supported: HashMap, LinkedHashMap, and TreeMap.

HashMap is efficient for locating a value, inserting an entry, and deleting an entry.

LinkedHashMap supports ordering of the entries in the map. The entries in a HashMap are not ordered, but the entries in a LinkedHashMap can be retrieved either in the order in which they were inserted into the map (known as the insertion order) or in the order in which they were last accessed, from least recently accessed to most recently (access order).

TreeMap is efficient for traversing the keys in a sorted order. The keys can be sorted using the Comparable interface or the Comparator interface.

Chapter 22 Developing Efficient Algorithms

1. The Big O notation is a theoretical approach for analyzing the performance of an

algorithm. It estimates how fast an algorithm’s execution time increases as the input size increases, which enables you to compare two algorithms by examining their growth rates.

2. An input that results in the shortest execution time is called the best-case input, and one that results in the longest execution time is called the worst-case input. Best- and worst-case analyses are not representative, but worst-case analysis is very useful. You can be assured that the algorithm will never be slower than the worst case.

3. An average-case analysis attempts to determine the average amount of time among all possible input of the same size. Average-case analysis is ideal, but difficult to perform because for many problems, it is hard to determine the relative probabilities and distributions of various input instances.

4. If the time is not related to the input size, the algorithm is said to take constant time with the notation O(1).

5. Linear search takes O(n) time. An algorithm with the O(n) time complexity is called a linear algorithm and it exhibits a linear growth rate. Binary search takes O(logn) time. An algorithm with the O(log n) time complexity is called a logarithmic algorithm and it exhibits a logarithmic growth rate.

6. The worst-time complexity for selection sort is O(n2). An algorithm with the O(n2) time complexity is called a quadratic algorithm and it exhibits a quadratic growth rate.

7. The time complexity for the Tower of Hanoi problem is O(2n). An algorithm with the O(2n) time complexity is called an exponential algorithm, and it exhibits an exponential growth rate.

8. A Fibonacci number at a given index can be found in O(n) time using dynamic programming approach.

9. Dynamic programming is the process of solving subproblems, then combining the solutions of the subproblems to obtain an overall solution. The key idea behind dynamic programming is to solve each subproblem only once and store the results for subproblems for later use to avoid redundant computing of the subproblems.

10. Euclid’s GCD algorithm takes O(logn) time.

11. All prime numbers less than or equal to n can be found in O¢n2n logn≤ time.

12. The closest pair can be found in O(n logn) time using the divide-and-conquer approach.

13. The divide-and-conquer approach divides the problem into subproblems, solves the subproblems, and then combines the solutions of the subproblems to obtain the solution for the entire problem. Unlike the dynamic programming approach, the subproblems in the divide-and-conquer approach don’t overlap. A subproblem is like the original problem with a smaller size, so you can apply recursion to solve the problem.

14. The Eight Queens problem can be solved using backtracking.

15. The backtracking approach searches for a candidate solution incrementally, abandoning that option as soon as it determines the candidate cannot possibly be a valid solution, then looks for a new candidate.

16. A convex hull for a set of points can be found in O(n2) time using the gift-wrapping algorithm, and in O(n logn) time using the Graham’s algorithm.

17. The brute force and Boyer-Moore string matching algorithms take O(nm) time and the KMP string matching algorithm takes O(n + m) time.

Chapter 23 Sorting

1. The worst-case complexity for a selection sort, insertion sort, bubble sort, and quick sort is O(n2).

2. The average- and worst-case complexity for a merge sort is O(n logn). The average time for a quick sort is also O(n logn).

3. Heaps are a useful data structure for designing efficient algorithms such as sorting. You learned how to define and implement a heap class, and how to insert and delete elements to/from a heap.

4. The time complexity for a heap sort is O(n logn).

5. Bucket and radix sorts are specialized sorting algorithms for integer keys. These algorithms sort keys using buckets rather than by comparing keys. They are more efficient than general sorting algorithms.

6. A variation of the merge sort—called an external sort—can be applied to sort large amounts of data from external files.

Chapter 24 Implementing Lists, Stacks, Queues, and Priority Queues

1. You learned how to implement array lists, linked lists, stacks, and queues.

2. To define a data structure is essentially to define a class. The class for a data structure should use data fields to store data and provide methods to support operations such as insertion and deletion.

3. To create a data structure is to create an instance from the class. You can then apply the methods on the instance to manipulate the data structure, such as inserting an element into the data structure or deleting an element from the data structure.

4. You learned how to implement a priority queue using a heap

Chapter 25 Binary Search Trees

1. A binary search tree (BST) is a hierarchical data structure. You learned how to define and implement a BST class, how to insert and delete elements into/from a BST, and how to traverse a BST using inorder, postorder, preorder, depth-first, and breadth-first searches.

2. An iterator is an object that provides a uniform way of traversing the elements in a container, such as a set, a list, or a binary tree. You learned how to define and implement iterator classes for traversing the elements in a binary tree.

3. Huffman coding is a scheme for compressing data by using fewer bits to encode characters that occur more frequently. The codes for characters are constructed based on the occurrence of characters in the text using a binary tree, called the Huffman coding tree.

Chapter 26 AVL Trees

1. An AVL tree is a well-balanced binary tree. In an AVL tree, the difference between the heights of two subtrees for every node is 0 or 1.

2. The process for inserting or deleting an element in an AVL tree is the same as in a binary search tree. The difference is that you may have to rebalance the tree after an insertion or deletion operation.

3. Imbalances in the tree caused by insertions and deletions are rebalanced through subtree rotations at the node of the imbalance.

4. The process of rebalancing a node is called a rotation. There are four possible rotations: LL rotation, LR rotation, RR rotation, and RL rotation.

5. The height of an AVL tree is O(log n). Therefore, the time complexities for the search, insert, and delete methods are O(log n).