1.1 Computer Processing

SR 1.1

The hardware of a computer system consists of its physical components such as a circuit board, monitor, or keyboard. Computer software consists of the programs that are executed by the hardware and the data that those programs use. Hardware is tangible, whereas software is intangible.

SR 1.2

The operating system provides a user interface and efficiently coordinates the use of resources such as main memory and the CPU.

SR 1.3

It takes 7,200,000 numbers for a three-minute song (40,000 × 60 × 3) and 144,000,000 numbers for one hour of music (40,000 × 60 × 60).

SR 1.4

The information is broken into pieces, and those pieces are represented as numbers.

SR 1.5

In general, N bits can represent 2 unique items. Therefore:

a. 2 bits can represent 4 items because 2 = 4.

b. 4 bits can represent 16 items because 2 = 16.

N

2

4

5

c. 5 bits can represent 32 items because 2 = 32.

d. 7 bits can represent 128 items because 2 = 128.

SR 1.6

It would take 6 bits to represent each of the 50 states. Five bits is not enough because 2 = 32 but six bits would be enough because 2 = 64.

1.2 Hardware Components

SR 1.7

A kilobyte (KB) is 2 = 1,024 bytes, a megabyte (MB) is 2 = 1,048,576 bytes, and a gigabyte (GB) is 2 = 1,073,741,824 bytes.

Therefore:

a. 3 KB = 3 \* 1,024 bytes = 3,072 bytes = approximately 3 thousand bytes

b. 2 MB = 2 \* 1,048,576 bytes = 2,097,152 bytes = approximately 2.1 million bytes

c. 4 GB = 4 \* 1,073,741,824 bytes = 4,294,967,296 bytes = approximately 4.3 billion bytes

SR 1.8

There are eight bits in a byte. Therefore:

a. 8 bytes = 8 \* 8 bits = 64 bits

b. 2 KB = 2 \* 1,024 bytes = 2,048 bytes = 2,048 \* 8 bits = 16,384 bits

5

7

5

6

10

20

30

c. 4 MB = 4 \* 1,048,576 bytes = 4,194,304 bytes = 4,194,304 \* 8 bits = 33,554,432 bits

SR 1.9

Under the stated conditions, one hour of music would require 288,000,000 bytes (40,000 × 60 × 60 × 2). Dividing this number by the number of bytes in a megabyte (1,048,576 bytes) gives approximately 275 MB. Note that a typical audio CD has a capacity of about 650 MB and can store about 70 minutes of music. This coincides with an actual sampling rate of 41,000 measurements per second, two bytes of storage space per measurement, and the need to store two streams of music to produce a stereo effect.

SR 1.10

The two primary hardware components are main memory and the CPU. Main memory holds the currently active programs and data. The CPU retrieves individual program instructions from main memory, one at a time, and executes them.

SR 1.11

A memory address is a number that uniquely identifies a particular memory location in which a value is stored.

SR 1.12

Main memory is volatile, which means the information that is stored in it will be lost if the power supply to the computer is turned off. Secondary memory devices are nonvolatile; therefore, the information that is stored on them is retained even if the power goes off.

SR 1.13

The word that best matches is

a. peripheral

b. controller

c. modem

d. main or RAM

e. secondary or ROM

f. RAM

g. CPU

1.3 Networks

SR 1.14

A file server is a network computer that is dedicated to storing and providing programs and data that are needed by many network users.

SR 1.15

Counting the number of unique connections in Figure

1.16

, there are 10 communication lines needed to fully connect a point-to-point network of five computers. Adding a sixth computer to the network will require that it be connected to the original five, bringing the total to 15 communication lines.

SR 1.16

Having computers on a network share a communication line is cost effective because it cuts down on the number of connections needed and it also makes it easier to add a new computer to the network. Sharing lines, however, can mean delays in communication if the network is busy.

SR 1.17

The word Internet comes from the word internetworking, a concept related to wide-area networks (WANs). An internetwork connects one network to another. The Internet is a WAN.

SR 1.18

TCP stands for Transmission Control Protocol. IP stands for Internet Protocol. A protocol is a set of rules that govern how two things communicate.

SR 1.19

Breaking down the parts of each URL:

a. duke is the name of a computer within the csc subdomain (the Department of Computing Sciences) of the villanova.edu domain, which represents Villanova University. The edu top-level domain indicates that it is an educational organization. This URL is requesting a file called examples.html from within a subdirectory called jss.

b. java is the name of a computer (Web server) at the sun.com domain, which represents Sun Microsystems, Inc. The com top-level domain indicates that it is a commercial business. This URL is requesting a file called index.html from within a subdirectory called products.

1.4 The Java Programming

Language

SR 1.20

The Java programming language was developed in the early 1990s by James Gosling at Sun Microsystems. It was introduced to the public in 1995.

SR 1.21

The processing of a Java application begins with the main method.

SR 1.22

The characters “Hello” will be printed on the computer screen.

SR 1.23

The entire line of code is a comment, so there is no result.

SR 1.24

All of the identifiers shown are valid except 12345 (since an identifier cannot begin with a digit) and black&white (since an identifier cannot contain the character &). The identifiers RESULT and result are both valid, but should not be used together in a program because they differ only by case. The underscore character (as in answer\_7) is a valid part of an identifier.

SR 1.25

Although any of the listed names could be used as the required identifier, the only “good” choice is scoreSum. The identifier x is not descriptive and is meaningless, the identifier sumOfTheTestScoresOfTheStudents is unnecessarily long, and the identifier smTstScr is unclear.

SR 1.26

White space is a term that refers to the spaces, tabs, and newline characters that separate words and symbols in a program. The compiler ignores extra white space; therefore, it doesn’t affect execution. However, it is crucial to use white space appropriately to make a program readable to humans.

1.5 Program Development

SR 1.27

At the lowest level, a computer’s instructions perform only simple tasks, such as copying a value or comparing two numbers. However, by putting together millions of these simple instructions every second, a computer can perform complex tasks.

SR 1.28

High-level languages allow a programmer to express a series of program instructions in English-like terms that are relatively easy to read and use. However, in order to execute, a program must be expressed in a particular computer’s machine language, which consists of a series of bits that are basically unreadable by humans. A high-level language program must be translated into machine language before it can be run.

SR 1.29

Java bytecode is a low-level representation of a Java source code program. The Java compiler translates the source code into bytecode, which can then be executed using the Java interpreter. The bytecode might be transported across the Web before being executed by a Java interpreter that is part of a Web browser.

SR 1.30

The word that best matches is

a. machine

b. assembly

c. high-level

d. high-level

e. compiler

f. interpreter

SR 1.31

Syntax rules define how the symbols and words of a programming language can be put together. The semantics of a programming language instruction determine what will happen when that instruction is executed.

SR 1.32

a. Compile-time error

b. Run-time error (you cannot divide by zero)

c. Logical error

1.6 Object-Oriented Programming

SR 1.33

1. Understand the problem.

2. Design a solution.

3. Consider alternatives and refinements to the solution.

4. Implement the solution.

5. Test the solution.

SR 1.34

The first solution to a problem that we think of may not be a good one. By considering alternative solutions before expending too much energy implementing our first idea, we can often save overall time and effort.

SR 1.35

The primary elements that support object-oriented programming are objects, classes, encapsulation, and inheritance. An object is defined by a class, which contains methods that define the operations on those objects (the services that they perform). Objects are encapsulated such that they store and manage their own data. Inheritance is a reuse technique in which one class can be derived from another.

Chapter 2 Data and Expressions

2.1 Character Strings

SR 2.1

A string literal is a sequence of characters delimited by double quotes.

SR 2.2

Both the print and println methods of the System.out object write a string of characters to the monitor screen. The difference is that, after printing the characters, the println performs a carriage return so that whatever’s printed next appears on the next line. The print method allows subsequent output to appear on the same line.

SR 2.3

A parameter is data that is passed into a method when it is invoked. The method usually uses that data to accomplish the service that it provides. For example, the parameter to the println method indicates what characters should be printed.

SR 2.4

The output produced by the code fragment is

One

Two Three

SR 2.5

The output produced by the code fragment is

Ready

Set

Go

SR 2.6

The output produced by the statement is

It is good to be 10

The + operator in the sub-expression (5 + 5) represents integer addition, since both of its operands are integers. If the inner parentheses are removed, the + operators represent string concatenation and the output produced is

It is good to be 55

SR 2.7

An escape sequence is a series of characters that begins with the backslash ( \) and that implies that the following characters should be treated in some special way. Examples: \n represents the newline character, \t represents the tab character, and \" represents the quotation character (as opposed to using it to terminate a string).

SR 2.8

System.out.println("\"I made this letter longer than " + "usual because I lack the time to\nmake it short.\"" + "\n\tBlaise Pascal");

2.2 Variables and Assignment

SR 2.9

A variable declaration establishes the name of a variable and the type of data that it can contain. A declaration may also have an optional initialization, which gives the variable an initial value.

SR 2.10

Given those variable declarations, the answers are:

a. Five variables are declared: count, value, total, MAX\_VALUE, and myValue.

b. They are all of type int.

c. count, MAX\_VALUE, and myValue are each given an initial value.

d. Yes, it is valid. myValue is a variable of type int and 100 is an int literal.

e. No, it is not valid. MAX\_VALUE is declared as a final variable and therefore it cannot be assigned a value other than its initial value.

SR 2.11

The variable name you choose should reflect the purpose of the variable. For example:

int numCDs = 0;

SR 2.12

The variable name you choose should reflect the purpose of the variable. Since the number of feet in a mile will not change, it is a good idea to declare a constant. For example:

final int FT\_PER\_MILE = 5280;

SR 2.13

First, by carefully choosing the name of the constant, you can make your program more understandable than if you just use the literal value. Second, using a constant ensures that the literal value represented by the variable will not be inadvertently changed somewhere in the program. Third, if you ever do have to rewrite the program using a different literal value, you will only need to change that value once, as the initial value of the constant, rather than many places throughout the program.

2.3 Primitive Data Types

SR 2.14

Primitive data are basic values such as numbers or characters. Objects are more complex entities that usually contain primitive data that help define them.

SR 2.15

An integer variable can store only one value at a time. When a new value is assigned to it, the old one is overwritten and lost.

SR 2.16

The four integer data types in Java are byte, short, int, and long. They differ in how much memory space is allocated for each and therefore how large a number they can hold.

SR 2.17

Java automatically assigns an integer literal the data type int. If you append an L or an l on the end of an integer literal, for example 1234L, Java will assign it the type long.

SR 2.18

Java automatically assigns a floating point literal the data type double. If you append an F or an f on the end of a floating point literal, for example 12.34f, Java will assign it the type float.

SR 2.19

A character set is a list of characters in a particular order. A character set defines the valid characters that a particular type of computer or programming language will support. Java uses the Unicode character set.

SR 2.20

The original ASCII character set supports 27 = 128 characters, the extended ASCII character set supports 28 = 256 characters, and the UNICODE character set supports 216 = 65,536 characters.

2.4 Expressions

SR 2.21

The result of 19%5 in a Java expression is 4. The remainder operator % returns the remainder after dividing the second operand into the first. The remainder when dividing 19 by 5 is 4.

7

8

16

SR 2.22

The result of 13/4 in a Java expression is 3 (not 3.25). The result is an integer because both operands are integers. Therefore, the / operator performs integer division, and the fractional part of the result is truncated.

SR 2.23

After executing the statement, diameter holds the value 20. First, the current value of diameter (5) is multiplied by 4, and then the result is stored back in diameter.

SR 2.24

Operator precedence is the set of rules that dictates the order in which operators are evaluated in an expression.

SR 2.25

The evaluations of the expressions are

a. 15 + 7 \* 3 = 15 + 21 = 36

b. (15 + 7) \* 3 = 22 \* 3 = 66

c. 3 \* 6 + 10 / 5 + 5 = 18 + 2 + 5 = 25

d. 27 % 5 + 7 % 3 = 2 + 1 = 3

e. 100 / 2 / 2 / 2 = 50 / 2 / 2 = 25 / 2 = 12

f. 100 / ( 2 / 2) / 2 = 100 / 1 / 2 = 100 / 2 = 50

SR 2.26

Expression a is valid. Expression b is invalid because there are two open parentheses but only one close parenthesis. Similarly with expression c, where there are two open parentheses but no close parenthesis. Expression d might be a valid algebraic expression in an algebra book, but it is not a valid expression in Java. There is no operator between the operands 2 and ( 4 ).

SR 2.27

After the sequence of statements, the value in result is 8.

SR 2.28

After the sequence of statements, the value in result is 8. Note that even though result was set to base + 3, changing the value of base to 7 does not retroactively change the value of result.

SR 2.29

An assignment operator combines an operation with assignment. For example, the + = operator performs an addition, then stores the value back into the variable on the left-hand side.

SR 2.30

After executing the statement, weight holds the value 83. The assignment operator −= modifies weight by first subtracting 17 from the current value (100), then storing the result back into weight.

2.5 Data Conversion

SR 2.31

A widening conversion tends to go from a small data value, in terms of the amount of space used to store it, to a larger one. A narrowing conversion does the opposite. Information is more likely to be lost in a narrowing conversion, which is why narrowing conversions are considered to be less safe than widening ones.

SR 2.32

The conversions are:

a. widening

b. narrowing

c. widening

d. widening

e. widening

SR 2.33

During the execution of the statement, the value stored in value is read and transformed into a float as it is being copied into the memory location represented by result. But the value variable itself is not changed, so value will remain an int variable after the assignment statement.

SR 2.34

During the execution of the statement, the value stored in result is read and then transformed into an int as it is being copied into the memory location represented by value. But the result variable itself is not changed, so it remains equal to 27.32, whereas value becomes 27.

SR 2.35

The results stored are

a. 3 integer division is used since both operands are integers.

b. 3.0 integer division is used since both operands are integers, but then assignment conversion converts the result of 3 to 3.0.

c. 2.4 floating point division is used since one of the operands is a floating point.

d. 3.4 num1 is first cast as a double; therefore, floating point division is used since one of the operands is a floating point.

e. 2 val1 is first cast as an int; therefore, integer division is used since both operands are integers.

2.6 Interactive Programs

SR 2.36

The corresponding lines of the GasMileage program are

a. import java.util.Scanner;

b. Scanner scan = new Scanner(System.in);

c. Scanner scan = new Scanner(System.in);

d. miles = scan.nextInt();

SR 2.37

Under the stated assumptions, the following code will ask users to enter their age and store their response in value.

System.out.print("Enter your age in years: ");

value = myScanner.nextInt();

Chapter 3 Using Classes and Objects

3.1 Creating Objects

SR 3.1

A null reference is a reference that does not refer to any object. The reserved word null can be used to check for null references before following them.

SR 3.2

The new operator creates a new instance (an object) of the specified class. The constructor of the class is then invoked to help set up the newly created object.

SR 3.3

The following declaration creates a String variable called author and initializes it:

String author = new String("Fred Brooks");

For strings, this declaration could have been abbreviated as follows:

String author = "Fred Brooks";

This object reference variable and its value can be depicted as follows:

author “Fred Brooks”

SR 3.4

To set an integer variable size to the length of a String object called name, you code:

size = name.length();

SR 3.5

Two references are aliases of each other if they refer to the same object. Changing the state of the object through one reference changes it for the other because there is actually only one object. An object is marked for garbage collection only when there are no valid references to it.

3.2 The String Class

SR 3.6

Strings are immutable. The only way to change the value of a String variable is to reassign it a new object. Therefore, the variables changed by the statements are:

a. none

b. s1

c. none

d. s3.

SR 3.7

The output produced is:

o

Found

11

5

SR 3.8

The following statement prints the value of a String object in all uppercase letters:

System.out.println(title.toUpperCase());

SR 3.9

The following declaration creates a String object and sets it equal to the first 10 characters of the String description;

String front = description.substring(0, 10);

3.3 Packages

SR 3.10

A Java package is a collection of related classes. The Java standard class library is a group of packages that support common programming tasks.

SR 3.11

Each package contains a set of classes that support particular programming activities. The classes in the java.net package support network communication and the classes in the javafx.scene.shape package represent shapes such as circles and rectangles.

SR 3.12

The Scanner class and the Random class are part of the java.util package. The String and Math classes are part of the java.lang package.

SR 3.13

The Point class, according to the online Java API documentation, represents a location with coordinates (x, y) in two-dimensional space.

SR 3.14

An import statement establishes the fact that a program uses a particular class, specifying what package that class is a part of. This allows the programmer to use the class name (such as Random) without having to fully qualify the reference (such as java.util.Random) every time.

SR 3.15

The String class is part of the java.lang package, which is automatically imported into any Java program. Therefore, no separate import declaration is needed.

3.4 The Random Class

SR 3.16

A call to the nextInt method of a Random object returns a random integer in the range of all possible int values, both positive and negative.

SR 3.17

Passing a positive integer parameter x to the nextInt method of a Random object returns a random number in the range of 0 to x -1. So a call to nextInt(20) will return a random number in the range 0 to 19, inclusive.

SR 3.18

The ranges of the expressions are:

a. From 0 to 49

b. From 10 to 14

c. From 5 to 14

d. From -25 to 24

SR 3.19

The expressions to generate the given ranges are:

a. generator.nextInt(31); // range is 0 to 30

b. generator.nextInt(10) + 10; // range is 10 to 19

c. generator.nextInt(11) - 5; // range is -5 to 5

3.5 The Math Class

SR 3.20

A class or static method can be invoked through the name of the class that contains it, such as Math.abs. If a method is not static, it can be executed only through an instance (an object) of the class.

SR 3.21

The values of the expressions are:

a. 20

b. 16.0

c. 16.0

d. 243.0

e. 125.0

f. 4.0

SR 3.22

The following statement prints the sine of an angle measuring 1.23 radians:

System.out.println(Math.sin(1.23));

SR 3.23

The following declaration creates a double variable and initializes it to 5 raised to the power 2.5:

double result = Math.pow(5, 2.5);

SR 3.24

Examples of methods that are not listed in Figure 3.5 include:

static int min(int a, int b)

static float max(long a, long b)

static long round(double a)

3.6 Formatting Output

SR 3.25

To obtain a NumberFormat object for use within a program, you request an object using one of the static methods provided by the NumberFormat class. The method you invoke depends upon your intended use of the object. For example, if you intend to use it for formatting percentages, you might code:

NumberFormat fmt = NumberFormat.getPercentInstance();

SR 3.26

a. The statement is:

NumberFormat moneyFormat = NumberFormat.getCurrencyInstance();

Do not forget, you also must import java.text.NumberFormat into your program.

b. The statement is:

System.out.println(moneyFormat.format(cost));

c. If the locale is the United States, the output will be $54.89. If the locale is the United Kingdom, the output will be £54.89.

SR 3.27

To output a floating point value as a percentage, you first obtain a NumberFormat object using a call to the static method getPercentageInstance of the NumberFormat class. Then, you pass the value to be formatted to the format method of the formatter object, which returns a properly formatted string. For example:

NumberFormat fmt = NumberFormat.getPercentageInstance();

System.out.println(fmt.format(value));

SR 3.28

The following code will prompt for and read in a double value from the user and then print the result of taking the square root of the absolute value of the input value to two decimal places:

Scanner scan = new Scanner(System.in);

DecimalFormat fmt = new DecimalFormat("0.00");

double value, result;

System.out.print("Enter a double value: ");

value = scan.nextDouble();

result = Math.sqrt(Math.abs(value));

System.out.println(fmt.format(result));

3.7 Enumerated Types

SR 3.29

The following is a declaration of an enumerated type for movie ratings:

enum Ratings {G, PG, PG13, R, NC17}

SR 3.30

Under the listed assumptions, the output is:

clubs

hearts

0

2

SR 3.31

By using an enumerated type, you guarantee that variables of that type will only take on the enumerated values.

3.8 Wrapper Classes

SR 3.32

A wrapper class is defined in the Java standard class library for each primitive type. In situations where objects are called for, an object created from a wrapper class may suffice.

SR 3.33

The corresponding wrapper classes are Byte, Integer, Double, Character, and Boolean.

SR 3.34

One approach is to use the constructor of Integer, as follows:

holdNumber = new Integer(number);

Another approach is to take advantage of autoboxing, as follows:

holdNumber = number;

SR 3.35

The following statement uses the MAX\_VALUE constant of the Integer class to print the largest possible int value:

System.out.println(Integer.MAX\_VALUE);

3.9 Introduction to JavaFX

SR 3.36

If the IDE you’re using will automatically launch a JavaFX application, no main method is needed. Otherwise, a one-line main method calling the launch method is required.

SR 3.37

A JavaFX stage is a window in which a scene is displayed. The primary stage of a JavaFX application is automatically created and passed into the start method.

SR 3.38

The root node of a scene contains all nodes displayed in the scene.

SR 3.39

The point (20, 50) is 20 pixels from the left edge of the scene and 50 pixels down from the top edge of the scene. All visible pixels in the Java coordinate system have positive coordinate values.

3.10 Basic Shapes

SR 3.40

Shapes are drawn in the order they are added to a container such as a group or pane. So, to make one shape appear in front of another, it should be added after it.

SR 3.41

The following declarations create a Rectangle that is 100 pixels wide, 200 pixels high, with its upper-left corner positioned at point (30, 20):

Rectangle myRect = new Rectangle(30, 20, 100, 200);

SR 3.42

The last two parameters to the Ellipse constructor specify the horizontal radius and the vertical radius, respectively. So that ellipse is taller than it is wide.

SR 3.43

If the value null is passed to the setFill method of a Circle object, its fill color will be fully transparent. Any shapes behind the circle will be visible through the circle.

SR 3.44

By grouping a set of nodes in a scene, you can apply transformations such as rotations and position shifts (translation) to the entire group at once.

3.11 Representing Colors

SR 3.45

An RGB value is a set of three integer values that represent a color by specifying the relative contributions of the colors red, green, and blue.

SR 3.46

The following statement creates a Color object equivalent to Color.PINK:

Color myPink = Color.rgb(255, 165, 0);

SR 3.47

The following statement creates a Color object equivalent to Color.YELLOW, which is defined by a full contribution of red and green, and no contribution of blue:

Color myYellow = Color.color(1.0, 1.0, 0.0);

Chapter 4 Writing Classes

4.1 Classes and Objects Revisited

SR 4.1

An attribute is a data value stored in an object and defines a particular characteristic of that object. For example, one attribute of a Student object might be that student’s current grade point average. Collectively, the values of an object’s attributes determine that object’s current state.

SR 4.2

An operation is a function that can be done to or done by an object. For example, one operation of a Student object might be to compute that student’s current grade point average. Collectively, an object’s operations are referred to as the object’s behaviors.

SR 4.3

Some attributes and operations that might be defined for a class called Book that represents a book in a library are:

Attributes Operations

idNumber checkOut

onShelfStatus checkIn

readingLevel isAvailable

dueDate placeOnHold

setStatus

SR 4.4

The answers are:

a. False—Identifying classes to help us solve a problem is a key step in object-oriented programming. In addition to identifying classes that already exist, we also identify, design, and implement new classes, as needed.

b. True—We call such operations mutators.

c. True—The result of many operations depends on the current state of the object on which they are operating.

d. False—In Java, the state of an object is represented by its instance data.

4.2 Anatomy of a Class

SR 4.5

A class is the blueprint of an object. It defines the variables and methods that will be a part of every object that is instantiated from it. But a class reserves no memory space for variables. Each object has its own data space and, therefore, its own state.

SR 4.6

The instance data of the Die class are MAX, an integer constant equal to 6 that represents the number of faces on the die and therefore the maximum value of the die, and faceValue, an integer variable that represents the current “up” or face value of the die.

SR 4.7

The methods defined for the Die class that can change the state of a Die object are roll and setFaceValue.

SR 4.8

When you pass an object to a print or println method, the toString method of the object is called automatically to obtain a string description of the object. If no toString method is defined for the object, then a default string is used. Therefore, it is usually a good idea to define a toString method when defining classes.

SR 4.9

The scope of a variable is the area within a program in which the variable can be referenced. An instance variable, declared at the class level, can be referenced in any method of the class. Local variables, including the formal parameters, declared within a particular method, can be referenced only in that method.

SR 4.10

A UML diagram helps us visualize the entities (classes and objects) in a program as well as the relationships among them. UML diagrams are tools that help us capture the design of a program prior to writing it.

4.3 Encapsulation

SR 4.11

A self-governing object is one that controls the values of its own data. Encapsulated objects, which don’t allow an external client to reach in and change its data, are self-governing.

SR 4.12

An object’s interface is the set of public operations (methods) defined on it. That is, the interface establishes the set of services the object will perform for the rest of the system.

SR 4.13

A modifier is a Java reserved word that can be used in the definition of a variable or method and that specifically defines certain characteristics of its use. For example, by declaring a variable with private visibility, the variable cannot be directly accessed outside of the object in which it is defined.

SR 4.14

A constant can be declared with public visibility because that would not violate encapsulation. Since the value of a constant cannot be changed, it is not generally a problem for another object to access it directly.

SR 4.15

The modifiers affect the methods and variables in the following ways:

a. A public method is called a service method for an object because it defines a service that the object provides.

b. A private method is called a support method because it cannot be invoked from outside the object and is used to support the activities of other methods in the class.

c. A public variable is a variable that can be directly accessed and modified by a client. This explicitly violates the principle of encapsulation and therefore should be avoided.

d. A private variable is a variable that can be accessed and modified only from within the class. Variables almost always are declared with private visibility.

4.4 Anatomy of a Method

SR 4.16

Although a method is defined in a class, it is invoked through a particular object to indicate which object of that class is being affected. For example, the Student class may define the operation that computes the grade point average (GPA) of a student, but the operation is invoked through a particular Student object to compute the GPA for that student. The exception to this rule is the invocation of a static method (see Chapter 3), which is executed through the class name and does not affect any particular object.

SR 4.17

An invoked method may return a value, which means it computes a value and provides that value to the calling method. The calling method usually uses the invocation and thus its return value, as part of a larger expression.

SR 4.18

An explicit return statement is used to specify the value that is returned from a method. The type of the return value must match the return type specified in the method definition.

SR 4.19

A return statement is required in methods that have a return type other than void. A method that does not return a value could use a return statement without an expression, but it is not necessary.

SR 4.20

An actual parameter is a value sent to a method when it is invoked. A formal parameter is the corresponding variable in the header of the method declaration; it takes on the value of the actual parameter so that it can be used inside the method.

SR 4.21

The following code implements the requested getFaceDown method:

//--------------------------------------------------

// Face down value accessor.

//--------------------------------------------------

public int getFaceDown()

{

return (MAX + 1) - faceValue;

}

SR 4.22

In the Transactions program

a. Three Account objects are created.

b. Two arguments (actual parameters) are passed to the withdraw method when it is invoked on the acct2 object.

c. No arguments (actual parameters) are passed to the addInterest method when it is invoked on the acct3 object.

SR 4.23

The method getBalance is a classic accessor method. One can also classify the toString method as an accessor, since it returns information about the object. The deposit, withdraw, and addInterest methods all provide both mutator and accessor capabilities, because they can be used to change the account balance and also return the value of the balance after the change is made. All of the methods mentioned above are service methods––they all have public visibility and provide a service to the client.

4.5 Constructors Revisited

SR 4.24

Constructors are special methods in an object that are used to initialize the object when it is instantiated.

SR 4.25

A constructor has the same name as its class, and it does not return a value.

4.6 Arcs

SR 4.26

An arc is defined as a portion of an ellipse using a start angle (where the arc begins) and a length angle (how long the arc is along the ellipse edge).

SR 4.27

ArcType.ROUND specifies an arc that includes the portion between the rounded ellipse edge and the center point of the ellipse, forming a “pie” shape. ArcType.OPEN specifies an arc that is only made up of the curve along the ellipse edge.

SR 4.28

A start angle of 180 degrees and an arc length of 180 degrees specify the entire bottom half of the underlying ellipse. The same arc could be defined using a start angle of 0 degree and an arc length of −180 degrees.

4.7 Images

SR 4.29

An Image object represents the image itself. An ImageView object is a JavaFX node that allows an Image to be displayed.

SR 4.30

A layout pane is a JavaFX node that organizes the visual presentation of its contents according to particular rules.

SR 4.31

Style properties for JavaFX nodes can be set using a call to the setStyle method, passing in a string that contains property name/value pairs in Cascading Style Sheet (CSS) notation.

4.8 Graphical User Interfaces

SR 4.32

A GUI control generates an event, typically when the user interacts with the control. A programmer sets up an event handler to execute certain code when an event occurs.

SR 4.33

A Button object generates an action event when the user pushes it.

SR 4.34

A JavaFX event handler can be defined using (1) a method reference, which specifies which method will be invoked when the event occurs, (2) a full class that implements the appropriate event handler interface, or (3) a lambda expression that defines the code to be executed right in the call to set the event handler. These are all notational variations of the same approach.

SR 4.35

A FlowPane is a layout pane that organizes its nodes into a row or column that wraps when it reaches a pane boundary.

4.9 Text Fields

SR 4.36

When the user presses Return while the cursor is in a text field, the text field generates an action event (which may be processed by an event handler).

SR 4.37

The rows and columns of a GridPane are numbered starting at 0. So, the upper left cell in a GridPane is at column 0 and row 0. Likewise, the cell that is three over and two down from the upper left corner is at column 2 and row 1.