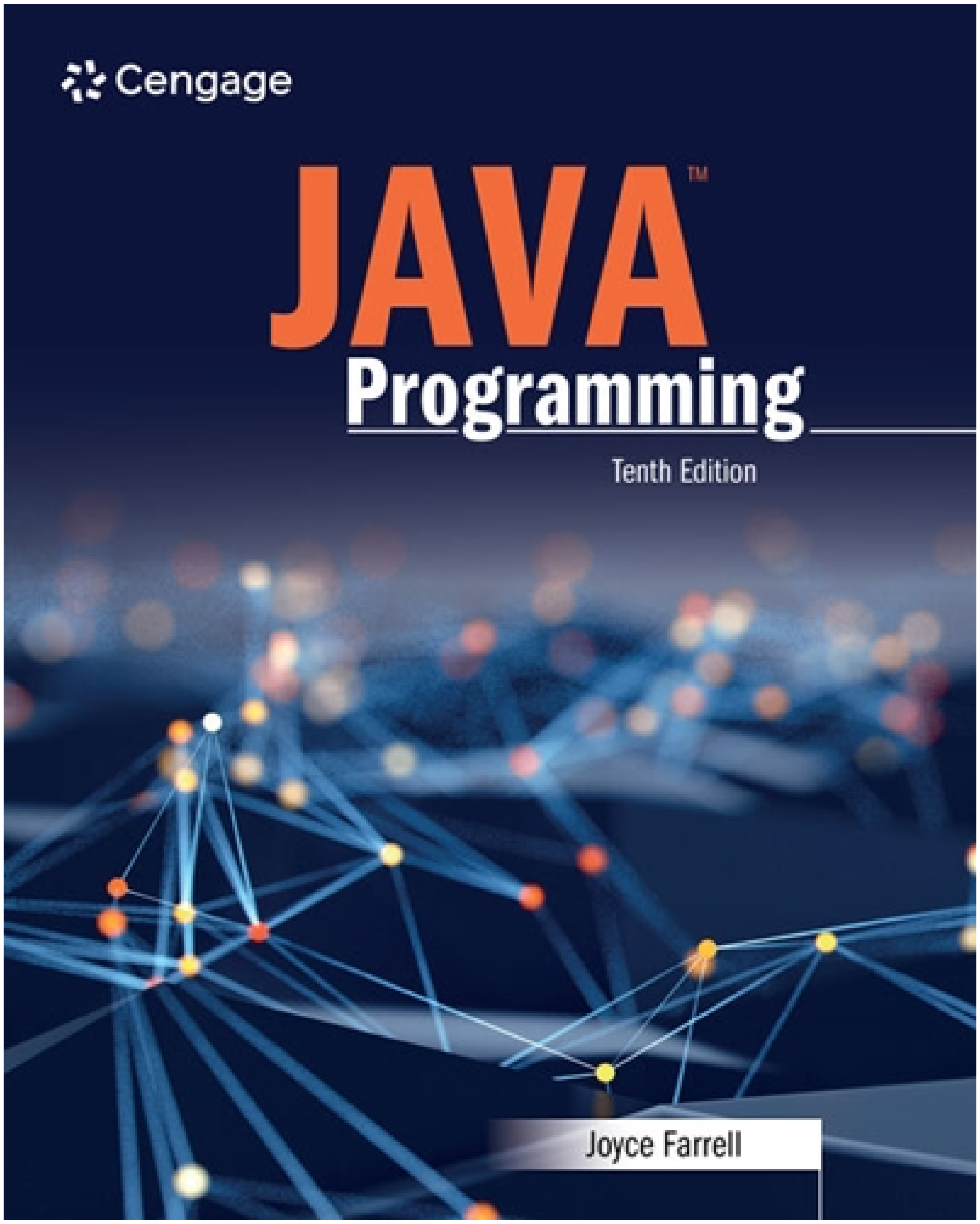
Cengage
JAVA™
Programming
Tenth Edition
Joyce Farrell

Solution and Answer Guide

Farrell, Java Programming 10e, [978-035-767-3423], Chapter 1: Creating Java Programs

Table of Contents

Review Questions Answers

1. The most basic circuitry-level computer language is ___________.

   a. machine language

   b. Java

   c. high-level language

   d. C++

   Answer: a

   Feedback:

   The most basic circuitry-level computer language is machine language. Machine language, or machine code, is the most basic set of instructions a computer can execute. Java and C++ are both high-level languages and are the opposite of circuitry-level computer language.

2. Languages that let you use an easily understood vocabulary of descriptive terms, such as *read*, *write*, or *add*, are known as ___________languages.

   a. procedural

   b. high-level

   c. machine

   d. object-oriented

Answer: b

Feedback:

High-level languages use English-like terms; Java is an example of a high-level language. Procedural languages are those that run by executing a series of procedures or methods. Machine-level languages do not use English-like terms; they use 1s and 0s. Object-oriented languages are run by declaring and using objects that contain data and methods.

3. The rules of a programming language constitute its ___________.

   a. syntax

   b. logic

   c. format

   d. objects

   Answer: a

   Feedback:

   The rules of a programming language constitute its syntax.

4. A ___________ translates high-level language statements into machine code.

   a. programmer

   b. syntax detector

   c. compiler

   d. decipherer

   Answer: c

   Feedback:

   A compiler translates high-level language statements into machine code. A programmer writes high-level language statements but does not translate them. “Syntax detector” and “decipherer” are not terms used in programming.

5. Named computer memory locations are called ___________.

   a. compilers

   b. variables

c. addresses

d. appellations

Answer: b

Feedback:

Named computer memory locations are variables. Compilers translate programming statements into machine language; they are not memory locations. Addresses are unnamed computer memory locations. “Appellations” is not a term used in programming.

6. The individual operations used in a computer program are often grouped into logical units called ___________.

   a. procedures

   b. variables

   c. constants

   d. logistics

   Answer: a

   Feedback:

   The individual operations used in a computer program are often grouped into logical units called procedures. Variables are named memory locations, and constants are values that do not change; they are not groups of logical operations. “Logistics” is not a term commonly used in programming.

7. Envisioning program components as objects that are similar to concrete objects in the real world is the hallmark of ___________.

   a. command-line operating systems

   b. procedural programming

   c. object-oriented programming

   d. machine languages

   Answer: c

   Feedback:

Envisioning program components as objects that are similar to concrete objects in the real world is the hallmark of object-oriented programming.

8. The values of an object's attributes are known as its ___________.

    a. state

    b. orientation

    c. methods

    d. condition

    Answer: a

    Feedback:

    The values of an object's attributes are known as its state.

9. An instance of a class is a(n) ___________.

    a. method

    b. procedure

    c. object

    d. case

    Answer: c

    Feedback:

    An instance of a class is an object.

10. Java is architecturally ___________.

    a. neutral

    b. oriented

    c. specific

    d. abstract

    Answer: a

    Feedback:

    Java is architecturally neutral.

11. You must compile classes written in Java into ___________.

    a. bytecode

    b. source code

    c. Javadoc statements

    d. object code

    Answer: a

    Feedback:

    You must compile classes written in Java into bytecode. Source code is the high-level programming statements. Javadoc statements are a type of comment used for documentation. Object code is the low-level code produced when a compiler translates high-level code.

12. All Java programming statements must end with a ___________.

    a. period

    b. comma

    c. closing parenthesis

    d. semicolon

    Answer: d

    Feedback:

    All Java programming statements must end with a semicolon.

13. Arguments to methods always appear within ___________.

    a. parentheses

    b. double quotation marks

    c. single quotation marks

    d. curly braces

    Answer: a

    Feedback:

Arguments to methods always appear within parentheses.

14. In a Java program, you must use ___________ to separate classes, objects, and methods.

    a. commas

    b. semicolons

    c. dots

    d. forward slashes

    Answer: c

    Feedback:

    In a Java program, you must use dots to separate classes, objects, and methods.

15. All Java applications must have a method named ___________.

    a. `method()`

    b. `main()`

    c. `java()`

    d. `Hello()`

    Answer: b

    Feedback:

    All Java applications must have a method named `main()`. Although an application could have a method named `method()`, `java()`, or `Hello()`, they are not required method names. `Hello()` also is not a conventional name because methods usually start with a lowercase letter.

16. Nonexecuting program statements that provide documentation are called ___________.

    a. classes

    b. notes

    c. comments

    d. commands

    Answer: c

    Feedback:

Nonexecuting program statements that provide documentation are called comments.

17. Java supports three types of comments: ___________, ___________, and Javadoc.

    a. line, block

    b. string, literal

    c. constant, variable

    d. single, multiple

    Answer: a

    Feedback:

    Java supports line, block, and Javadoc comments.

18. Which of the following is not necessary to do before you can run a Java program?

    a. coding

    b. compiling

    c. saving

    d. debugging

    Answer: d

    Feedback:

    Although you should debug a program, it can run with bugs—it simply will produce incorrect results. You must code, compile, and save a program before you can run it.

19. The command to execute a compiled Java application is ___________.

    a. `run`

    b. `execute`

    c. `javac`

    d. `java`

    Answer: d

    Feedback:

The command to execute a compiled Java application is `java`. The command to compile a program is `javac`.

20. You save text files containing Java source code using the file extension ____________.

    a. *.java*

    b. *.class*

    c. *.txt*

    d. *.src*

    Answer: a

    Feedback:

    You save text files containing Java source code using the file extension .java.

Programming Exercises Solutions

1. Is each of the following class identifiers (a) legal and conventional, (b) legal but unconventional, or (c) illegal?

    a. `myClass`

    b. `void`

    c. `Golden Retriever`

    d. `invoice#`

    e. `36542ZipCode`

    f. `Apartment`

    g. `Fruit`

    h. `8888`

    i. `displayTotal()`

    j. `Accounts_Receivable`

    Solution

    a. `myClass` (b) Unconventional because class names conventionally start with an uppercase letter

b. `void` (c) Illegal because `void` is a reserved word

c. `Golden Retriever` (c) Illegal because class names cannot contain a space

d. `invoice#` (c) Illegal because class names cannot contain #

e. `36542ZipCode` (c) Illegal because class names cannot start with a number

f. `Apartment` (a) Legal and conventional because it is one word, not a reserved word, contains no spaces or special characters, and starts with an uppercase letter

g. `Fruit` (a) Legal and conventional because it is one word, not a reserved word, contains no spaces or special characters, and starts with an uppercase letter

h. `8888` (c) Illegal because class names cannot start with a number

i. `displayTotal()` (c) Illegal because class names cannot contain parentheses

j. `Accounts_Receivable` (b) Legal but unconventional because class names do not usually contain an underscore

2. Is each of the following method identifiers (a) legal and conventional, (b) legal but unconventional, or (c) illegal?

   a. `associationRules()`

   b. `void()`

   c. `Golden Retriever()`

   d. `invoice#()`

e. `36542ZipCode()`

f. `PayrollApp()`

g. `getReady()`

h. `911()`

i. `displayTotal()`

j. `Accounts_Receivable()`

Solution

a. `associationRules()` (a) Legal and conventional because it is not a keyword, contains no spaces or special characters, and starts with a lowercase letter

b. `void()` (c) Illegal because `void` is a keyword

c. `Golden Retriever()` (c) Illegal because method names cannot contain a space

d. `invoice#()` (c) Illegal because method names cannot contain #

e. `36542ZipCode()` (c) Illegal because method names cannot start with a number

f. `PayrollApp()` (b) Legal but unconventional because method names usually start with a lowercase letter

g. `getReady()` (a) Legal and conventional because it is not a keyword, contains no spaces or special characters, and starts with a lowercase letter

h. `911()` (c) Illegal because method names cannot start with a number

i. `displayTotal()` (a) Legal and conventional because it is not a keyword, contains no spaces or special characters, and starts with a lowercase letter

j. `Accounts_Receivable()` (b) Legal but unconventional because method names usually start with a lowercase letter and do not contain underscores

3. Name at least three attributes that might be appropriate for each of the following classes:

   a. `RealEstateListing`

   b. `Vacation`

   c. `CreditCardBill`

   Solution

   a. `RealEstateListing`

   `streetAddress, numberOfBedrooms, price`

   b. `Vacation`

   `lengthInDays, destination, cost`

   c. `CreditCardBill`

   `creditCardNumber, nameOnAccount, amountDue`

4. Name at least three real-life objects that are instances of each of the following classes:

   a. `Song`

   b. `CollegeCourse`

   c. `Musician`

   Solution

   a. `Song`

   Happy Birthday, Star Spangled Banner, All You Need is Love

   b. `CollegeCourse`

   History 102, English 200, College Algebra

   c. `Musician`

   Yo-Yo Ma, Vladimir Horowitz, Eric Clapton

5. Name at least three classes to which each of these objects might belong:

   a. `myRedBicycle`

b. `friedEgg`

c. `cellPhone`

Solution

a. `myRedBicycle`

`Bicycle, Transportation, ExerciseDevice`

b. `friedEgg`

`BreakfastItem, EggDish, HighProteinFood`

c. `cellPhone`

`Telephone, CommunicationDevice, ExpensiveItem`

6. Write, compile, and test a class that uses four `println()` statements to display four lines of the lyrics of your favorite song. Save the class as **SongLyrics.java**.

Solution
```
class SongLyrics
{
   public static void main(String[] args)
   {
      System.out.println("Somewhere over the rainbow");
      System.out.println("Way up high");
      System.out.println("There's a land that I heard of");
      System.out.println("Once in a lullaby");
   }
}
```

7. Write, compile, and test a class that uses four `println()` statements to display, in order, your favorite movie quote, the movie it comes from, the character who said it, and the year of the movie. Save the class as **MovieQuoteInfo.java**.

Solution
```
class MovieQuoteInfo
{
   public static void main(String[] args)
   {
      System.out.println("Rosebud,");
      System.out.println("said by Charles Foster Kane");
      System.out.println("in the movie Citizen Kane");
      System.out.println("in 1941.");
   }
}
```

8. Write, compile, and test a class that displays the pattern shown in Figure 1-26. Save the class as

   TableAndChairs.java.

   Solution
   ```
   class TableAndChairs
   {
      public static void main(String[] args)
      {
         System.out.println(" ");
         System.out.println("X                      X");
         System.out.println("X                      X");
         System.out.println("X      XXXXXXXXXX      X");
         System.out.println("XXXXX  X        X  XXXXX");
         System.out.println("X   X  X        X  X   X");
         System.out.println("X   X  X        X  X   X");
      }
   }
   ```
9. Write, compile, and test a class that displays the pattern shown in Figure 1-27. Save the class as

   Triangle.java.

   Solution
   ```
   class Triangle
   {
      public static void main(String[] args)
      {
         System.out.println(" ");
         System.out.println("       T");
         System.out.println("      TTT");
         System.out.println("     TTTTT");
         System.out.println("    TTTTTTT");
         System.out.println("   TTTTTTTTT");
         System.out.println("  TTTTTTTTTTT");
         System.out.println(" TTTTTTTTTTTTT");
      }
   }
   ```

10. Write, compile, and test a class that displays the following statement about comments on two

    lines:

    Program comments are nonexecuting statements

    you add to a file for documentation.

    Also include the same statement in three different comments in the class; each comment should use one of the three different methods of including comments in a Java class. Save the class as **Comments.java**.

Solution

```
class Comments
{
   public static void main(String[] args)
   {
      System.out.println("Program comments are nonexecuting statements ");
      System.out.println("you add to a file for documentation.");
      // Program comments are nonexecuting statements
      // you add to a file for documentation.
      /* Program comments are nonexecuting statements
         you add to a file for documentation. */
      /** Program comments are nonexecuting statements
          you add to a file for documentation.  */
   }
```

11. From 1925 through 1963, Burma Shave advertising signs appeared next to highways across the United States. There were always four or five signs in a row containing pieces of a rhyme, followed by a final sign that read *Burma Shave*. For example, one set of signs that has been preserved by the Smithsonian Institution reads as follows:

    Shaving brushes

    You'll soon see 'em

    On a shelf

    In some museum

    Burma Shave

    Find a classic Burma Shave rhyme on the Web, making sure the fifth line is *Burma Shave*. Write, compile, and test a class that displays the slogan. Save the class as **BurmaShave.java**.

Solution

```
class BurmaShave
{
   public static void main(String[] args)
   {
      System.out.println("Shaving brushes");
      System.out.println("You'll soon see 'em");
```

```
        System.out.println("On a shelf");
        System.out.println("In some museum");
        System.out.println("Burma Shave");
    }
}
```

Debugging Exercises Solutions

1. Each of the following files in the Chapter01 folder in your downloadable student files has syntax and/or logic errors. In each case, determine the problem and fix the errors. After you correct the errors, save each file using the same filename preceded with Fix. For example, DebugOne1.java will become FixDebugOne1.java.

   a. **DebugOne1.java**

```
public class DebugOne1
{
    /* This program displays a greeting
    public void main(String[] args)
    {
        System.out.println("Hello")
    }
}
```

Solution

```
public class FixDebugOne1
{
    /* This program displays a greeting */
    public static void main(String[] args)
    {
        System.out.println("Hello");
    }
}
```

   b. **DebugOne2.java**

```
public class DebugOne2
{
    /* This program displays some output*/
    public static void Main(String[] args)
    {
        System.outprintln("Java programming is fun.");
        System.outprintln("Getting a program to work);
        System.outprintln(can be a challenge,");
        System.outprintln("but when everything works correctly,");
        System.outprintln(it's very satisfying");
    }
 }
```

Solution

```
public class FixDebugOne2
{
      /* This program displays some output*/
      public static void main(String[] args)
      {
            System.out.println("Java programming is fun.");
            System.out.println("Getting a program to work");
            System.out.println("can be a challenge,");
            System.out.println("but when everything works correctly,");
            System.out.println("it's very satisfying");
      }
 }
```

c. **DebugOne3.java**

```
public class DebugOne3
{
   public static void main(String args)
   {
      System.out.print1n("Over the river");
      System.out.pr1ntln("and through the woods");
      system.out.println("to Grandmother's house we go");
   }
}
```

Solution

```
public class FixDebugOne3
{
   public static void main(String[] args)
   {
      System.out.println("Over the river");
      System.out.println("and through the woods");
      System.out.println("to Grandmother's house we go");
   }
}
```

d. **DebugOne4.java**

```
public class DebugOne4
{
   public statoc void main(String[] args)
   {
      Systemout.println("This output ");
      Systemout.print("is on the same line as the last one.");
      Systemout.println("But this is on a new ine.");
   }
}
```

Solution

```
public class FixDebugOne4
{
   public static void main(String[] args)
   {
      System.out.print("This output ");
      System.out.println("is on the same line as the last one.");
      System.out.println("But this is on a new ine.");
   }
}
```

Game Zone Solutions

1. Computer games have been around for a long time:

   - In 1947, Thomas T. Goldsmith Jr. and Estle Ray Mann filed the first patent for a computer game.
   - In 1952, A. S. Douglas wrote his University of Cambridge Ph.D. dissertation on human–computer interaction, and created the first graphical computer game—a version of tic-tac-toe.
   - In 1962, the first computer game that could be played at multiple computers was developed at MIT. It was called *Spacewar!*
   - The first commercially available video game was *Pong*, introduced by Atari in 1973.
   - In 1980, Atari's *Asteroids* and *Lunar Lander* became the first video games to be registered in the U.S. Copyright Office.
   - Throughout the 1980s, players spent hours with games that now seem very simple and unglamorous, such as *Adventure*, *Oregon Trail*, *Where in the World Is Carmen Sandiego?*, and *Myst*.

   Today, commercial computer games are much more complex; they require many programmers, graphic artists, and testers to develop them, and large management and marketing staffs are needed to promote them. A game might cost many millions of dollars to develop and market, but a successful game might earn hundreds of millions of dollars. Obviously, with the brief introduction to programming you have had in this chapter, you cannot create a very sophisticated game. However, you can get started.

   For games to hold your interest, they almost always include some random, unpredictable behavior. For example, a game in which you shoot asteroids loses some of its fun if the asteroids follow the same, predictable path each time you play the game. Therefore, generating random values is a key component in creating most interesting computer games.

Appendix D contains information about generating random numbers. To fully understand the process, you must learn more about Java classes and methods. For now, however, you can copy the following statement to generate and use a dialog box that displays a random number between 1 and 10:

```
JOptionPane.showMessageDialog(null,"The number is "+
   (1 + (int)(Math.random() * 10)));
```

Write a Java application that displays two dialog boxes in sequence. The first asks you to think of a number between 1 and 10. The second displays a randomly generated number; the user can see whether the guess was accurate. (In future chapters, you will improve this game so that the user can enter a guess and the program can determine whether the user was correct. Also, as you gain skills in Java, you will be able to tell the user how far off the guess was, whether the guess was high or low, and provide a specific number of repeat attempts.) Save the file as **RandomGuess.java**.

Solution

```
import javax.swing.JOptionPane;
public class RandomGuess
{
   public static void main(String[] args)
   {
      JOptionPane.showMessageDialog(null,
         "Before pressing OK try to guess my number between 1 and 10");
      JOptionPane.showMessageDialog(null,
         "The number is " + (1 + (int)(Math.random() * 10)));
   }
}
```

Case Problems Solutions

1. a. Yummy Catering provides meals for parties and special events. Write a program that displays

Yummy Catering's motto, which is *Yummy makes the food that makes it a party.* Save the file as **YummyMotto.java**.

Solution

```
public class YummyMotto
{
   public static void main(String[] args)
   {
      System.out.println("Yummy makes the food that makes it a party.");
   }
}
```

b. Create a second program that displays the motto surrounded by a border composed of asterisks. Save the file as **YummyMotto2.java**.

Solution

```
public class YummyMotto2
{
   public static void main(String[] args)
   {
      System.out.println("****************************************************");
      System.out.println("*                                                  *");
      System.out.println("*   Yummy makes the food that makes it a party.    *");
      System.out.println("*                                                  *");
      System.out.println("****************************************************");
   }
}
```

2. Sunshine Seashore Supplies rents beach equipment such as kayaks, canoes, beach chairs, and umbrellas to tourists. Write a program that displays Sunshine's motto, which is *Sunshine Seashore makes it fun in the sun.* Save the file as **SunshineMotto.java**.

Solution

```
public class SunshineMotto
{
   public static void main(String[] args)
   {
      System.out.println("Sunshine Seashore makes it fun in the sun.");
   }
}
```

b. Create a second program that displays the motto surrounded by a border composed of repeated *S*s. Save the file as **SunshineMotto2.java**.

Solution

```
public class SunshineMotto2
{
   public static void main(String[] args)
   {
      System.out.println("SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS");
      System.out.println("S                                             S");
      System.out.println("S  Sunshine Seashore makes it fun in the sun. S");
      System.out.println("S                                             S");
      System.out.println("SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS");
   }
}
```

Instructor Manual

Farrell, Java Programming, 10e, ©2023, 978-035-767-3423; Chapter 1: Creating Java Programs

Table of Contents

Purpose and Perspective of the Chapter

The purpose of this chapter is to provide an introduction to programming. Students will learn basic programming terminology and apply this to the Java programming environment. They will also learn how to create a simple Java application that produces output to the console. The chapter covers the basic components of a Java application and how to compile and run a Java program. Students will also create a simple GUI application and learn about resources that they can use to develop their Java skills.

Cengage Supplements

The following product-level supplements provide additional information that may help you in preparing your course. They are available in the Instructor Resource Center.

- Sample Syllabus (provides a sample course outline)
- Test Bank | Cengage Testing, powered by Cognero® (contains assessment questions and problems)
- Solution and Answer Guide (offers solutions, answers, and feedback)
- PowerPoint (provides text-based lectures and presentations)
- Transition Guide (provides information about what's new from edition to edition)
- Complete List of Objectives (provides a comprehensive list of learning objectives)
- MindTap Educator Guide (contains a detailed outline of the MindTap course)
- MindTap Coding IDE Lab Solutions (provides solutions to the IDE labs)
- MindTap Coding IDE ReadMe (provides helpful information related to the IDE labs)

List of Student Downloads

Students should download the following items from the Student Companion Center to complete the activities and assignments related to this chapter:

- Data Files (provides files needed to complete the end of chapter Debugging Exercises)
- MindTap Coding IDE ReadMe (provides helpful information related to the IDE labs)
- MindTap Collaboration Lab Self-Reflection (prompts students with questions to answer after completing each of the Collaboration Labs)

Chapter Objectives

The following objectives are addressed in this chapter:

01.01 Define basic programming terminology

01.02 Compare procedural and object-oriented programming

01.03 Describe the features of the Java programming language

01.04 Analyze a Java application that produces console output

01.05 Compile a Java class and correct syntax errors

01.06 Run a Java application and correct logic errors

01.07 Add comments to a Java class

01.08 Create a Java application that produces GUI output

01.09 Identify and consult resources to help develop Java programming skills

Complete List of Chapter Activities and Assessments

For additional guidance refer to the Teaching Online Guide.

Chapter Objective	PPT slide	Activity/Assessment	Activity Duration
01.01	7	Activity 1.1: Knowledge Check	5 minutes
01.01, 01.04–01.06		Chapter 1/MindTap Debugging Exercises	30-40 minutes
01.01–01.07		Chapter 1/MindTap: Review Questions	10-20 minutes
01.02, 01.04–01.06		Chapter 1/MindTap: Programming Exercises	45-90 minutes per exercise
01.02, 01.07-01.09	48	Activity 1.4: Case Study	15-20 minutes
01.03	18	Activity 1.2: Knowledge Check	5 minutes
01.04–1.06	37	Activity 1.3: Discussion	20 minutes
01.04–01.06		Chapter 1/MindTap: Game Zone	45-90 minutes per question
01.04–01.06		Chapter 1/MindTap: Case Problems	45-90 minutes per problem

[return to top]

Key Terms

access specifier: defines the circumstances under which a class can be accessed and the other classes that have the right to use a class.

Allman style: the indent style in which curly braces are aligned and each occupies its own line; it is named for Eric Allman, a programmer who popularized the style.

application software: performs tasks for users. Also called *applications* or *apps*.

architecturally neutral: used to write a program that runs on any platform (operating system).

arguments: information passed to a method so it can perform its task.

at run time: is a phrase that describes the period of time during which a program executes.

attributes: the characteristics that define an object as part of a class.

block comments: start with a forward slash and an asterisk (/*) and end with an asterisk and a forward slash (*/). Block comments can appear on a line by themselves, on a line before executable code, or on a line after executable code. Block comments can also extend across as many lines as needed.

bug: Logic or syntax error in a program.

bytecode: consists of programming statements that have been compiled into binary format.

call a procedure: when a procedural program calls a series of procedures to input, manipulate, and output values.

class: describes a group or collection of objects with common properties.

class body: set of data and methods that are enclosed within the {} following the class definition.

class definition: describes what attributes its objects will have and what those objects will be able to do.

class header: contains the keyword class

clean build: created when you delete all previously compiled versions of a class before compiling again.

commands: program statements that are orders to a computer.

comment out: turn a statement into a comment so the compiler will not execute its command.

compile-time error: one in which the compiler detects a violation of language syntax rules and is unable to translate the source code to machine code.

compiler: a program that translates language statements into machine code.

computer program: a set of instructions that you write to tell a computer what to do.

computer simulations: attempt to mimic real-world activities.

console applications: support character output to a computer screen in a DOS window.

debugging: a process to remove errors from a program.

development environment: set of tools that help you write programs.

dialog box: a GUI object resembling a window in which you can place messages you want to display.

documentation comments: used by Javadoc to generate nicely formatted program documentation.

encapsulation: refers to the hiding of data and methods within an object.

executing: carrying out a statement.

FAQs: frequently asked questions.

graphical user interfaces (GUIs): a graphical environment in which users interact with a program; pronounced "gooeys."

hardware: the general term for computer equipment.

high-level programming language: allows you to use a vocabulary of reasonable terms, such as “read,” “write,” or “add,” instead of the sequences of on and off switches that perform these tasks.

identifier: a name of a program component such as a class, an object, or a variable.

`import` statement: used to access a built-in Java class that is contained in a package.

inheritance: the ability to create classes that share the attributes and methods of existing classes but with more specific features.

instance: an existing object of a class.

instantiation: to create a class variable.

interpreter: a program that translates language statements into machine code.

Java: a programming language developed by Sun Microsystems as an object-oriented language used both for general-purpose business applications and for interactive, Web-based Internet applications.

Java API: the application programming interface; a collection of information about how to use every prewritten Java class.

Java interpreter: a program that checks the bytecode and communicates with the operating system, executing the bytecode instructions line by line within the Java Virtual Machine.

Java Virtual Machine (JVM): a hypothetical (software-based) computer on which Java runs.

Javadoc: a special case of block comments. They begin with a forward slash and two asterisks (/**) and end with an asterisk and a forward slash (*/). You can use Javadoc comments to generate documentation with a program named Javadoc.

JDK: the Java Development Kit, a software development kit (SDK) that includes tools used by programmers.

K & R style: the indent style in which the opening brace follows the header line; it is named for Kernighan and Ritchie, who wrote the first book on the C programming language.

keywords: the vocabulary of a programming language.

line comments: start with two forward slashes (//) and continue to the end of the current line. Line comments can appear on a line by themselves or at the end of a line following executable code.

literal string: a series of characters that appear exactly as entered. Any literal string in Java appears between double quotation marks.

logic: executing the various statements and procedures in the correct order to produce the desired results.

logic error: occurs when a program compiles successfully but produces an error during execution.

low-level programming language: written to correspond closely to a computer processor's circuitry and is not easily read or understood.

machine code: a program written in circuitry-level language as a series of on and off switches.

machine language: a program written in circuitry-level language as a series of on and off switches.

method: a self-contained block of program code, similar to a procedure.

method header: the first line in a method; contains information about how other methods can interact with it.

object: an instance of a class; made up of attributes and methods.

object-oriented programs: involve creating classes, creating objects from those classes, and creating applications that use those objects. Thinking in an object-oriented manner involves envisioning program components as objects that are similar to concrete objects in the real world; then, you can manipulate the objects to achieve a desired result.

package: contains a group of built-in Java classes.

parsing: the process the compiler uses to divide source code into meaningful portions

Pascal casing: using an uppercase letter to begin an identifier and starting each new word in an identifier. Also called *upper camel casing.*

Passing arguments: sending arguments to a method.

polymorphism: describes the feature of languages that allows the same word to be interpreted correctly in different situations based on the context.

procedural programming: a style of programming in which sets of operations are executed one after another in sequence.

procedures: sets of operations performed by a computer program.

program: a set of instructions that you write to tell a computer what to do.

program comments: nonexecuting statements that you add to a Java file for the purpose of documentation.

program statements: similar to English sentences; they carry out the tasks that programs perform.

properties: another term for an attribute of a class.

`public`: a reserved keyword; means that the fields and methods are accessible to any other class.

runtime error: occurs when a program compiles successfully but does not execute.

SDK: a software development kit, or a set of tools useful to programmers.

semantic errors: occur when you use a correct word in the wrong context in program code.

software: a general term for computer programs.

source code: consists of programming statements written in a high-level programming language.

standard output device: normally the monitor.

state: the values of the attributes of an object.

`static`: a reserved keyword; means that a method is accessible and usable even though no objects of the class exist.

`String` class: a Java class that can be used to hold character strings.

syntax: the rules of the language.

syntax error: a programming error that occurs when you introduce typing errors into your program or use the programming language incorrectly. A program containing syntax errors will not compile.

system software: manages the computer.

Unicode: an international system of character representation.

upper camel casing: using an uppercase letter to begin an identifier and starting each new word in an identifier. Also called *Pascal casing.*

variable: a named computer memory location that holds values that might vary.

`void`: the keyword used in a method header that indicates that the method does not return any value when it is called.

whitespace: any combination of nonprinting characters; for example, spaces, tabs, and carriage returns (blank lines).

windowed applications: create a graphical user interface (GUI) with elements such as menus, toolbars, and dialog boxes.

Write once, run anywhere (WORA): a slogan developed by Sun Microsystems to describe the ability of one Java program version to work correctly on multiple platforms.

[return to top]

What's New in This Chapter

The following elements are improvements in this chapter from the previous edition:

- Addition of underscore as keyword
- Addition of Eclipse as an example of a development environment
- Instruction change on how to search for Java help because students have to be more flexible now that a new version of Java is released every six months

[return to top]

Chapter Outline

In the outline below, each element includes references (in parentheses) to related content. "CH.##" refers to the chapter objective; "PPT Slide #" refers to the slide number in the PowerPoint deck for this chapter (provided in the PowerPoints section of the Instructor Resource Center); and, as applicable for each discipline, accreditation or certification standards ("BL 1.3.3"). Introduce the chapter and use the Ice Breaker in the PPT if desired, and if one is provided for this chapter. Review chapter objectives for Chapter 1. (PPT Slide 3).

I. 1.1 Learning Programming Terminology (01.01, PPT Slides 4-6)

a. A computer system consists of hardware (equipment) and software (programs that tell the computer what to do).
b. Software can be divided into two categories: application (performs tasks for the user), and system (manages the computer).
c. The logic behind a program determines the order of instructions needed to produce the desired result.
d. Computer programs can be written in high- or low-level programming languages.
    - High-level languages (Java, Visual Basic, C++, and C#) use English-like terms such as read, write, and add.
    - Low-level languages correspond to a computer's circuitry, are not easily read or understood, and must be customized for every type of machine.
e. Machine language is the most basic set of instructions that a computer can execute.
    - All computer programs ultimately are converted to machine language, which is the lowest-level language.
f. Every program uses its own syntax, which are rules about how language elements, such as keywords, are combined to produce usable program statements.
g. The compiler or interpreter translates the statements into machine language.
    - A compiler translates an entire program before executing it.
    - An interpreter translates one statement at a time, and executes the statement as it is translated.
h. A syntax error is a misuse of the language. Debugging is the process of getting rid of bugs.
i. Logic errors allow a program to run but causes it to operate incorrectly.

II. 1.2 Comparing Procedural and Object-Oriented Programming Concepts (01.02, PPT Slides 9-13)

a. Procedural programming is a programming style in which operations are executed one after another in sequence.
    - Procedural programs use variables, which are named locations that can hold data.
    - The individual operations are grouped into logical units called procedures.
    - When a program calls a procedure, the logic is suspended so that the procedure's commands can execute.
b. Object-oriented programming (OOP) is an extension of procedural programming in which you create:
    - Classes, which are blueprints for objects
    - Objects, which are specific instances of classes

- Applications that use the objects

c. OOP was originally used for computer simulations and graphical user interfaces (GUIs).

d. A class is a group or collection of objects with common properties.
- A class definition describes the attributes of objects and what they will do.
- Attributes are the properties of an object, or characteristics that define it.
- An object is s specific instance of a class.
- Instantiation is the creation of an instance.
- The values of the properties of an object are the object's state.

e. A method is a self-contained block of program code that carries out some action.

f. In OOP, attributes and methods are encapsulated into objects.
- Encapsulation is the enclosure of data and methods within an object, which allows you to treat an object's methods and data as a single entity.
- Encapsulations also refers to the concealment of an object's data and methods from outside sources.

g. Inheritance is the ability to create classes that share attributes and methods of existing classes.
- Can build new classes and concentrate on the specialized features being added.

III. 1.3 Features of the Java Programming Language (01.03, PPT Slides 14-17)

a. Java was created by Sun Microsystems
- For general purpose and interactive, web-based applications
- Known for security features
- Architecturally neutral: can be run on any operating system

b. Runs on the JVM (Java Virtual Machine) a hypothetical computer composed only of software

c. Source code can be constructed using:
- Plain text editor (Notepad), *OR*
- Set of tools called a development environment (Eclipse, NetBeans, JDeveloper)

d. Three steps in writing a Java program:
- Source code statements you write are saved to a file
- Compiler converts to bytecode

e. JVM provides security because it is insulated from the hardware on which it runs.

f. Write once, run anywhere (WORA) is the ability to run on multiple platforms.

g. There are two types of Java applications:

- Console (character or text output)
- Windowed (GUI)

IV. 1.4 Analyzing a Java Application That Produces Console Output (01.04, PPT Slides 20-28)

a. Even the simplest Java application involves a fair amount of confusing syntax.
b. All Java statements end with a semicolon.
c. Most Java statements can be spread across multiple lines if you place line breaks in appropriate places.
d. A literal string is a series of characters:
    - Will appear in output exactly as entered
    - Written between double quotation marks
e. Arguments are Pieces of information passed to a method
    - A method requires information to perform its task
    - System class Refers to the standard output device for a system
f. Everything used within a Java program must be part of a class
    - Define a Java class using any name or identifier
g. Upper Camel casing (Pascal casing)
    - Each word of an identifier begins with uppercase letter
    - UnderGradStudent
    - InventoryItem
    - An access specifier defines how a class can be accessed
h. Identifiers must begin with one of the following:
    - Letter of the English alphabet
    - Non-English letter (such as α or π)
    - Underscore
    - Dollar sign
i. Identifiers can only contain:
    - Letters
    - Digits
    - Underscores
    - Dollar signs
j. Identifiers cannot:
    - be a Java reserved keyword
    - be true, false, or null
    - begin with a digit
    - Method header
k. The first line in a method Contains information about how other methods can interact with it
    - String class is a Java class used to hold character strings
l. `Static` is a reserved keyword

- Means the method is accessible and usable even though no objects of the class exist

m. void
- Use in the `main()` method header
- Does not indicate the `main()` method is empty
- Indicates the `main()` method does not return a value when called
- Does not mean that `main()` doesn't produce output

n. Indent Style
- Use whitespace to organize code and improve readability
- For every opening curly brace ({) in a Java program, there must be a corresponding closing curly brace
- Placement of the opening and closing curly braces is not important to the compiler
- Allman style used in text

o. To save a Java class
- Save the class in a file with exactly the same name and .java extension
- For public classes, class name and filename must match exactly

V. 1.5 Compiling a Java Class and Correcting Syntax errors (01.05, PPT Slides 29-32)

a. Compiling a Java class
- Compile the source code into bytecode
- Translate the bytecode into executable statements using a Java interpreter
- Type `javac First.java`

b. Compilation outcomes:
- `javac` is an unrecognized command
- Program language error messages
- No messages indicating successful completion

c. Reasons for error messages
- Misspelled the command `javac`
- A misspelled filename
- Not within the correct subfolder or subdirectory on the command line
- Improper installation of Java

d. The first line of the error message displays:
- The name of the file where the error was found
- The line number
- The nature of the error

e. Next lines identify:
- The symbol
- The location

f. Compile-time error:
- The compiler detects a violation of language rules

- Refuses to translate the class to machine code
- During parsing, the compiler divides source code into meaningful portions

VI. 1.6 Running a Java Application and Correcting Logical errors (01.06, PPT Slides 33-36)

a. Run the application from the command line
   - Type `java` First

b. Shows the application's output in the command window

c. The class is stored in a folder named Java on the C drive
   - Modify the text file that contains the existing class
   - Save the file with changes using the same filename
   - Compile the class with the `javac` command
   - Interpret the class bytecode and execute the class using the java command

d. Logic error
   - The syntax is correct but incorrect results were produced when executed

e. Run-time error
   - Not detected until execution
   - Often difficult to find and resolve

VII. 1.7 Adding Comments to a Java Class (01.07, PPT Slides 39-41)

a. Program comments
   - Nonexecuting statements added to a program for documentation
   - Use to leave notes for yourself or others
   - Include the author, date, and class's name or function

b. Comment out a statement
   - Turn it into a comment
   - The compiler does not translate, and the JVM does not execute its command

c. Line comments:
   - Start with two forward slashes (//)
   - Continue to the end of the current line
   - Do not require an ending symbol

d. Block comments:
   - Start with a forward slash and an asterisk (/*)
   - End with an asterisk and a forward slash (*/)

e. Javadoc comments:
   - A special case of block comments
   - Begin with a slash and two asterisks (/**)
   - End with an asterisk and a forward slash (*/)
   - Use to generate documentation

VIII. 1.8 Creating a Java Application that Produces GUI Output (01.08, PPT Slides 42-44)
   a. Dialog box
      - A GUI object resembling a window
      - Messages placed for display
      - JOptionPane class is used to produce
   b. import statement
      - Use to access a built-in Java class
   c. Package
      - A group of classes

IX. 1.9 Finding Help (01.09, PPT Slides 45-47)
   a. Java API
      - Also called the Java class library
      - Provides prewritten information about Java classes
   b. FAQs on the Java Web site
   c. Java Development Kit (JDK)
      - A software development kit (SDK) of programming tools
      - Free to download
   d. Don't forget
      - The file's name must match the class name
      - To end a block comment
      - Java is case sensitive
      - To end every statement with a semicolon
      - To recompile when making changes
   e. Don't confuse these terms: Parentheses, braces, brackets, curly braces, square brackets, and angle brackets
   f. Do not end class or method headers with a semicolon
   g. Don't panic when you see a lot of compiler error messages
   h. Don't assume your program is perfect when all compiler errors are eliminated

[return to top]

Discussion Questions

You can assign these questions several ways: in a discussion forum in your LMS; as whole-class discussions in person; or as a partner or group activity in class.

1. Discussion: (PPT Slide 37) Duration 20 minutes.
   a. Question 1 (01.05, 01.06)

    i. How do syntax errors and logic errors differ? Discuss what defines a logic error, list some examples, and explain why they are more difficult to detect.
    ii. Answer: A syntax error occurs when there is a misuse of language, such as a misspelled word. A logic error occurs when the syntax of a program is correct, but the results it produces are incorrect. Examples of logic errors include misspelling a word in a literal string or instructing to multiply two numbers instead of divide. The compiler doesn't find errors such as spelling errors in output, or incorrectly used mathematical operators. It is the responsibility of the program author to test programs and find logic errors.

b. Question 2 (01.04)
    i. Identify why each of the following class names are illegal in Java: Shoe Type, 2023inventory, private, Item#
    ii. Answer: Each of the following class names are illegal in Java:
        - **Shoe Type:** Includes a space
        - **2023inventory:** Starts with a number
        - **Private:** Uses a reserved keyword
        - **Item#:** Contains the number symbol

[return to top]

Suggested Usage for Lab Activities

1. Coding Labs, graded (in the MindTap Learning Path):
    a. Programming Exercise 1-6 (1.04 -1.06)
    b. Programming Exercise 1-7 (1.04 -1.06)
    c. Programming Exercise 1-8 (1.04 -1.06)
    d. Programming Exercise 1-9 (1.04 -1.06)
    e. Programming Exercise 1-10 (1.04 -1.06)
    f. Programming Exercise 1-11 (1.04 -1.06)
    g. Debugging Exercise 1-1 (1.01, 1.04, 1.05-1.06)
    h. Debugging Exercise 1-2 (1.01, 1.04, 1.05-1.06)
    i. Debugging Exercise 1-3 (1.01, 1.04, 1.05-1.06)
    j. Debugging Exercise 1-4 (1.01, 1.04, 1.05-1.06)
    k. Bonus Bug 1 (1.04, 1.06, 1.07)
2. Coding Snippets, ungraded (embedded in the MindTap Reader):
    a. Producing Console Output Using `print()` (1.04)
    b. Producing Console Output Using `println()` (1.04)
    c. Creating a Java Class (1.04)
    d. Creating Java Comments (1.07)

[return to top]

Additional Activities and Assignments

1. (01.03, 01.04, 01.06, 01.07, 01.08) You want to create a Java program that prints out your name to the console and displays it in a dialog box. The program should run on an Android device. Research what resources are needed to develop Android using Java. List the steps you will need to accomplish to save, compile, and execute your program. Include a list of classes and statements. What types of comments are necessary? What types of logic errors might you encounter, and how will you identify and solve them?

[return to top]

Additional Resources

Cengage Video Resources

- MindTap Videos:
    - What is Computer Programming? 1:18 min
    - A Java Program 4:00 min (1.04-1.06)

Internet Resources

- **The Java Tutorials: Understanding Class Members (webpage)** (01.01)
- **The Java Tutorials: Object-Oriented Programming Concepts (webpage)** (01.02)

[return to top]

Appendix

Generic Rubrics

Providing students with rubrics helps them understand expectations and components of assignments. Rubrics help students become more aware of their learning process and progress, and they improve students' work through timely and detailed feedback. Customize these rubrics as you wish.

Standard Discussion Rubric

Criteria	Meets Requirements	Needs Improvement	Incomplete
Participation	Submits or participates in discussion by the posted deadlines. Follows all assignment. instructions for initial post and responses. 5 points	Does not participate or submit discussion by the posted deadlines. Does not follow instructions for initial post and responses. 3 points	Does not participate in discussion. 0 points
Contribution Quality	Comments stay on task. Comments add value to discussion topic. Comments motivate other students to respond. 10 points	Comments may not stay on task. Comments may not add value to discussion topic. Comments may not motivate other students to respond. 5 points	Does not participate in discussion. 0 points
Etiquette	Maintains appropriate language. Offers criticism in a constructive manner. Provides both positive and negative feedback. 5 points	Does not always maintain appropriate language. Offers criticism in an offensive manner. Provides only negative feedback. 3 points	Does not participate in discussion. 0 points

Standard Collaboration Rubric

Criteria	Meets Requirements	Needs Improvement	Incomplete
Share knowledge	Consistently and actively contributes knowledge, opinions, and skills. 5 points	Contributes to the group only when prompted. 3 points	Does not contribute to the group. 0 points
Adjust to unforeseen circumstances	Seeks different solutions, approaches, and strategies in an effective, original, and/or creative way. 5 points	Seeks a single solution, approach, or strategy, but does not pursue better or original alternatives. 3 points	Relies on the first solution generated and does not offer other solutions. 0 points

Make decisions	Helps the group identify necessary changes and encourages group action for change. 10 points	Participates in needed changes only when prompted. 5 points	Does not participate in decision making. 0 points
Build consensus	Values, encourages, and acknowledges the work of other group members. Takes responsibility for the assignment that reflects minority and majority conclusions of the group. 10 points	Listens attentively to members of the group but contributes little to the group and assignment. 5 points	Does not participate in building consensus within the group. 0 points
Complete deliverables (only if applicable)	Completes all assigned components or deliverables with quality, creativity, and accuracy. Turns in all assigned components by agreed upon deadlines without reminders. 10 points	Completes all assigned components with bare minimum of quality and creativity. Requires reminders and follow-up from team members in order to complete deliverables. 5 points	Does not complete assigned components or deliverables or turns them in late. Assigned components are poorly executed, inaccurate, or not functioning. Does not respond to communication from team. 0 points

Programming Rubrics

Providing students with programming rubrics helps them understand expectations and components of writing code. Rubrics help students become more aware of their learning process and progress, and they improve students' work through timely and detailed feedback. Customize these rubrics as you wish.

Standard Programming Rubric (Simple)

Criteria	**Excellent**	**Good**	**Fair**	**Poor**	**None**
Function: The code works without errors and fulfills all project requirements. Test cases for in- and out-of-range inputs are successful. Error handling is used correctly, and where necessary and appropriate.	4	3	2	1	0

Formatting: The code is readable, formatted properly with indents, and line spacing. Variables are named appropriately and consistently. The code is well-commented.	4	3	2	1	0
Simplicity: The code efficiently fulfills the project requirements. Hardcoded values are minimized. Complex programming logic is distributed in short, single-purpose lines of code.	4	3	2	1	0
Modularity: Programming follows the Don't Repeat Yourself best practice. Each method/function performs one discrete task. Variables are strongly typed and used for singular, discrete purposes.	4	3	2	1	0
Comprehension: The code demonstrates the student understands the concepts required. The student can explain how the program works. The student can explain why alternative solutions are not as good as the one(s) used.	4	3	2	1	0

Standard Programming Rubric (Detailed)

Criteria	**4**	**3**	**2**	**1**	**0**	**Total**
Function	The code works without errors and fulfills all project requirements. Test cases for in- and out-of-range inputs are successful. Error handling is used correctly, where necessary and appropriate.	The code works but may generate some errors with edge cases. Most project requirements are met. Test cases for in-range inputs are successful. Error handling, if required, is present.	The code compiles/executes but does not fulfill all project requirements. Test case for at least one in-range input is successful. Error handling, if required, is present.	The code is mostly complete but does not compile/execute. Error handling, if required, is present.	The code is substantially incomplete or nonfunctional. No error handling is present.	

Formatting	The code is readable and formatted properly into control structures with indents and line spacing. Variables are named appropriately and consistently. The code is well-commented.	The code is readable with appropriate line spacing but not indented. Variables are named appropriately but not consistently. The code is mostly commented.	The code is readable but written in large blocks without indents or spacing. Variables are inconsistently named. The code is commented sporadically.	The code is written haphazardly and/or appears to be copied from external sources with minimal modification. Variables are randomly named or single characters. Minimal comments are present.	No attempt made to format the code. Variables are single characters. No comments are present.	
Simplicity	The code efficiently accomplishes what is in the project requirements. Hardcoded values are used only where appropriate. Complex programming logic is distributed in short, single-purpose lines of code.	The code includes some unnecessary extra steps but accomplishes the project requirements. Some hardcoded values that will require future code updates are present. Some multifunction or multistep lines of code are used.	Numerous extra blocks of code and/or unused variables are present. The code includes many extra unnecessary steps. Hardcoded values are used where variables should be. Lines of code combine numerous logical functions/steps.	Code complexity is great enough to challenge future modification or debugging. Extra/unnecessary steps in code throughout. Programming flow is difficult to follow. Hardcoded values are used instead of variables.	Code complexity is too great to allow future modifications or debugging. Programming flow does not exist. No logical order to programmatic steps. Variables are not used or used incorrectly.	

Modularity	The code follows the Don't Repeat Yourself best practice. Each method/function performs one discrete task. Variables are strongly typed and used for singular, discrete purposes.	The code mostly follows the Don't Repeat Yourself best practice. Some methods/functions perform more than one discrete task. Occasional variable repurposing or reuse exists.	Logical blocks of code are duplicated in different methods/functions. Methods/functions combine multiple logical operations and steps. Variables are untyped and/or occasionally reused.	Minimal effort to create logical blocks of code. Procedural coding practices used for logical flow without code reuse. Variables are untyped and/or repurposed.	No attempt made to create modular code. All code exists in one large block. Variables, if present, are untyped and/or repurposed.	
Comprehension	The code clearly demonstrates the student understands the concepts required. The student can explain how the entire program works. The student can explain why alternative solutions are not as good as the one(s) used.	The code shows the student understands most of the concepts required. The student can explain how most of the program works. The student can explain at least one alternative solution.	The code shows the student is aware of some of the concepts required. The student can explain how some parts of the program work. The student can discuss alternative solutions.	The code shows the student is aware of at least one of the concepts required. The student can explain how at least one part of the program works. The student can discuss at least one alternative solution.	The code does not show an awareness of the concepts required. The student cannot explain how the program should work. The student cannot discuss alternative solutions.	
					Total	