Chapter 4: Writing Classes
Solutions

Multiple Choice Solutions

1. b
2. c
3. d
4. b
5. e
6. d
7. a
8. e
9. a
10. b

True/False Solutions

1. T
2. F
3. T
4. F
5. T
6. T
7. F
8. F
9. F
10. F

Short Answer Solutions

4.1. Write a method header for a method named translate that takes an integer parameter and returns a double.

```
double translate (int param)
```

4.2. Write a method header for a method named `find` that takes a String and a double as parameters and returns an integer.

```
int find (String str, double num)
```

4.3. Write a method header for a method named printAnswer that takes three doubles as parameters and doesn't return anything.

```
void printAnswer (double d1, double d2, double d3)
```

4.4. Write the body of the method for the following header. The method should return a welcome message that includes the user's name and visitor number. For example, if the parameters were "Joe" and 5, the returned string would be "Welcome Joe! You are visitor #5."

```
String welcomeMessage (String name, int visitorNum)
{
   return "Welcome " + name + "! You are visitor #" + visitorNum;
}
```

4.5. Write a method called `powersOfTwo` that prints the first 10 powers of 2 (starting with 2). The method takes no parameters and doesn't return anything.

```
public void powersOfTwo()
{
   int base = 2;
   for (int power = 1; power <= 10; power++)
      System.out.println (Math.pow(base,power));
}
```

4.6. Write a method called `alarm` that prints the string `"Alarm!"` multiple times on separate lines. The method should accept an integer parameter that specifies how many times the string is printed. Print an error message if the parameter is less than 1.

```
public void alarm (int number)
{
   if (number < 1)
      System.out.println ("ERROR: Number is less than 1.");
   else
      for (int count = 1; count <= number; count++)
         System.out.println ("Alarm!");
}
```

4.7. Write a method called `sum100` that returns the sum of the integers from 1 to 100, inclusive.

```
public int sum100()
{
   int sum = 0;
   for (int count = 1; count <= 100; count++)
      sum += count;
   return sum;
}

or

public int sum100()
{
       return (101 * 100 / 2);
}
```

4.8. Write a method called `maxOfTwo` that accepts two integer parameters and returns the larger of the two.

```
public int maxOfTwo (int num1, int num2)
{
   int result = num1;

   if (num2 > num1)
      result = num2;

   return result;
}
```

4.9. Write a method called `sumRange` that accepts two integer parameters that represent a range. Issue an error message and return zero if the second parameter is less than the first. Otherwise, the method should return the sum of the integers in that range (inclusive).

```
public int sumRange (int start, int end)
{
   int sum = 0;

   if (end < start)
      System.out.println ("ERROR: Invalid Range”);
   else
      for (int num = start; num <= end; num++)
         sum += num;

   return sum;
}
```

4.10. Write a method called `larger` that accepts two floating-point parameters (of type double) and returns true if the first parameter is greater than the second, and false otherwise.

```
public boolean larger (double num1, double num2)
{
   return (num1 > num2);
}
```

4.11. Write a method called `countA` that accepts a `String` parameter and returns the number of times the character `'A'` is found in the string.

```
public int countA (String text)
{
   int count = 0;
   for (int position = 0; position < text.length(); position++)
         if (text.charAt(position) == 'A')
            count++;
   return count;
}
```

4.12. Write a method called `evenlyDivisible` that accepts two integer parameters and returns true if the first parameter is evenly divisible by the second, or vice versa, and false otherwise. Return false if either parameter is zero.

```
public boolean evenlyDivisible (int num1, int num2)
{
   boolean result = false;

   if (num1 != 0 && num2 != 0)
      if (num1 % num2 == 0 || num2 % num1 == 0)
         result = true;

   return result;
}
```

4.13. Write a method called `average` that accepts two integer parameters and returns their average as a floating point value.

```
public double average (int num1, int num2)
{
   return (num1 + num2) / 2.0;
}
```

4.14. Overload the `average` method of Exercise 4.13 such that if three integers are provided as parameters, the method returns the average of all three.

```
public double average (int num1, int num2, int num3)
{
   return (num1 + num2 + num3) / 3.0;
}
```

4.15. Overload the `average` method of Exercise 4.13 to accept four integer parameters and return their average.

```
public double average (int num1, int num2, int num3, int num4)
{
   return (num1 + num2 + num3 + num4) / 4.0;
}
```

4.16. Write a method called `multiConcat` that takes a `String` and an integer as parameters. Return a `String` that consists of the string parameter concatenated with itself `count` times, where `count` is the integer

parameter. For example, if the parameter values are "hi" and 4, the return value is "hihihihi". Return the original string if the integer parameter is less than 2.

```
public String multiConcat (String text, int repeats)
{
   String result = text;

   if (repeats > 1)
      for (int count = 2; count <= repeats; count++)
         result += text;

   return result;
}
```

4.17. Overload the multiConcat method from Exercise 4.12 such that if the integer parameter is not provided, the method returns the string concatenated with itself. For example, if the parameter is "test", the return value is "testtest"

```
public String multiConcat (String text)
{
   String result = text + text;
   return result;
}
```

4.18. Write a method called isAlpha that accepts a character parameter and returns true if that character is either an uppercase or lowercase alphabetic letter.

```
public boolean isAlpha (char ch)
{
   return ( (ch >= 'a' && ch <= 'z') ||
            (ch >= 'A' && ch <= 'Z') );
}
```

4.19. Write a method called floatEquals that accepts three floating-point values as parameters. The method should return true if the first two parameters are equal within the tolerance of the third parameter. *Hint*: See the discussion in Chapter 3 on comparing floating-point values for equality.

```
public boolean floatEquals (double float1, double float2,
                            double tolerance)
{
   return (Math.abs(float1 - float2) <= tolerance);
}
```

4.20. Write a method called reverse that accepts a String parameter and returns a string that contains the characters of the parameter in reverse order. Note that there is a method in the String class that performs this operation, but for the sake of this exercise, you are expected to write your own.

```
public String reverse (String text)
{
   String result = "";

   for (int place = text.length()-1; place >= 0; place--)
       result += text.charAt(place);

   return result;
}
```

4.21. Write a mutator method for faceValue in the Die class in Listing 4.7. The method should only allow faceValue to take on a valid value.

```
public void setFaceValue (int newValue)
{
   if ( (newValue >= 1) && (newValue <= numFaces) )
      faceValue = newValue;
}
```

4.22. Write a method called `isIsoceles` that accepts three integer parameters that represent the lengths of the sides of a triangle. The method returns true if the triangle is isosceles but not equilateral (meaning that exactly two of the sides have an equal length), and false otherwise.

```
public boolean isIsoceles (int side1, int side2, int side3)
{
   boolean result = false;

   if ( (side1 == side2) && side1 != side3) ||
        (side2 == side3) && side2 != side1) ||
        (side1 == side3) && side1 != side2) )
         result = true;

   return result;
}
```

4.23. Write a method called `randomInRange` that accepts two integer parameters representing a range. The method should return a random integer in the specified range (inclusive). Return zero if the first parameter is greater than the second.

```
// assumes java.util.Random is imported
public int randomInRange (int first, int second)
{
   int result = 0;
   Random generator = new Random();

   if (first <= second)
   {
      int range = second - first + 1;
      result = generator.nextInt(range) + first;
   }

   return result;
}
```

4.24. Write a method called `randomColor` that creates and returns a `Color` object that represents a random color. Recall that a `Color` object can be defined by three integer values between 0 and 255 representing the contributions of red, green, and blue (its RGB value).

```
final int MAX = 256;

// assumes java.util.Random and java.awt.Color are imported
public Color randomColor ()
{
   Random generator = new Random();

   int randRed = generator.nextInt(MAX);
   int randGreen = generator.nextInt(MAX);
   int randBlue = generator.nextInt(MAX);

   return new Color(randRed, randGreen, randBlue);
}
```

4.25. Write a method called `drawCircle` that draws a circle based on the method's parameters: a `Graphics` object through which to draw the circle, two integer values representing the (x, y) coordinates of the center of the circle, another integer that represents the circle's radius, and a `Color` object that defines the circle's color. The method does not return anything.

```
//  assumes java.awt.* is imported
public void drawCircle (Graphics page, int x, int y, int rad,
                        Color color)
{
   page.setColor (color);
   page.drawOval (x-rad, y-rad, rad*2, rad*2);
}
```

4.26. Overload the `drawCircle` method of Exercise 4.25 such that if the `Color` parameter is not provided, the circle's color will default to black.

```
//  assumes java.awt.* is imported
public void drawCircle (Graphics page, int x, int y, int rad)
{
   page.setColor (Color.black);
   page.drawOval (x-rad, y-rad, rad*2, rad*2);
}
```

Programming Project Solutions

4.8 StudentBody

```
//********************************************************************
//  StudentBody.java       Author: Lewis/Loftus/Cocking
//
//  Demonstrates the use of an aggregate class.
//********************************************************************

public class StudentBody
{
   //-----------------------------------------------------------------
   //  Creates some Address and Student objects and prints them.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      Address school = new Address ("800 Lancaster Ave.", "Villanova",
                                    "PA", 19085);

      Address jHome = new Address ("21 Jump Street", "Lynchburg",
                                   "VA", 24551);
      Student john = new Student ("John", "Gomez", jHome, school, 89, 74, 92);

      Address mHome = new Address ("123 Main Street", "Euclid", "OH",
                                   44132);
      Student marsha = new Student ("Marsha", "Jones", mHome, school);
      marsha.setTestScore(1, 96);
      marsha.setTestScore(2, 67);
      marsha.setTestScore(3, 80);

      System.out.println (john);
      System.out.println ();
      System.out.println (marsha);
```

```
   }
}
```

4.8 Student

```
//********************************************************************
//  Student.java       Author: Lewis/Loftus/Cocking
//
//  Represents a college student.
//********************************************************************

public class Student
{
   private String firstName, lastName;
   private Address homeAddress, schoolAddress;
   private int test1, test2, test3;

   //-----------------------------------------------------------------
   //  Sets up this Student object with the specified initial values.
   //-----------------------------------------------------------------
   public Student (String first, String last, Address home,
                   Address school, int score1, int score2, int score3)
   {
      firstName = first;
      lastName = last;
      homeAddress = home;
      schoolAddress = school;
      test1 = score1;
      test2 = score2;
      test3 = score3;
   }

   //-----------------------------------------------------------------
   //  Sets up this Student object with the specified initial values
   //  and test scores of 0.
   //-----------------------------------------------------------------
   public Student (String first, String last, Address home,
                   Address school)
   {
      firstName = first;
      lastName = last;
      homeAddress = home;
      schoolAddress = school;
      test1 = 0;
      test2 = 0;
      test3 = 0;
   }

   //-----------------------------------------------------------------
   //  Sets the given test to the given score. If the test number
   //  is invalid, does nothing.
   //-----------------------------------------------------------------
   public void setTestScore (int test, int score)
   {
      if (test == 1)
         test1 = score;
      else if (test == 2)
         test2 = score;
      else if (test == 3)
```

```
         test3 = score;
   }

   //-----------------------------------------------------------------
   //  Returns the test score for the given test, or 0 if the test
   //  number is invalid.
   //-----------------------------------------------------------------
   public int getTestScore (int test)
   {
      if (test == 1)
         return test1;
      else if (test == 2)
         return test2;
      else if (test == 3)
         return test3;
      else
         return 0;
   }

   //-----------------------------------------------------------------
   //  Computes and returns this Student's average test score.
   //-----------------------------------------------------------------
   public double average ()
   {
      double avg = (test1 + test2 + test3) / 3.0;
      return avg;
   }

   //-----------------------------------------------------------------
   //  Returns this Student object as a string.
   //-----------------------------------------------------------------
   public String toString()
   {
      String result;

      result = firstName + " " + lastName + "\n";
      result += "Home Address:\n" + homeAddress + "\n";
      result += "School Address:\n" + schoolAddress + "\n";
      result += "Test 1 Score: " + test1 + "\n";
      result += "Test 2 Score: " + test2 + "\n";
      result += "Test 3 Score: " + test3 + "\n";
      result += "Average Test Score: " + average();

      return result;
   }
}
```

4.8 Address

```
//********************************************************************
//  Address.java       Author: Lewis/Loftus/Cocking
//
//  Represents a street address.
//********************************************************************

public class Address
{
   private String streetAddress, city, state;
   private int zipCode;
```

```
   //-----------------------------------------------------------------
   //  Sets up this Address object with the specified data.
   //-----------------------------------------------------------------
   public Address (String street, String town, String st, int zip)
   {
      streetAddress = street;
      city = town;
      state = st;
      zipCode = zip;
   }

   //-----------------------------------------------------------------
   //  Returns this Address object as a string.
   //-----------------------------------------------------------------
   public String toString()
   {
      String result;

      result = streetAddress + "\n";
      result += city + ", " + state + "  " + zipCode;

      return result;
   }
}
```

AP-Style Multiple Choice Solutions

1. A
2. C
3. B
4. D
5. C
6. A

AP-Style Free Response Solution

4.1

a.

```
// instance variables
private String myCode;
private String origCode;
private String xString;

public Code (String code)
{
   myCode = code;
   origCode = code;
   xString = "";
   for (int i=0; i < code.length(); i++)
      xString += "X";
}
```

b.

```
public void hide (int p1, int p2)
{
   myCode = myCode.substring(0, p1)
            + xString.substring(p1, p2)
            + myCode.substring(p2, myCode.length());
}
```

c.

```
public void recover (int p1, int p2)
{
   myCode = myCode.substring(0, p1)
            + origCode.substring(p1, p2)
            + myCode.substring(p2, myCode.length());
}
```