Chapter 4: Writing Classes

Notes:

- *Every method, other than the constructor, has to have a return type. You get to keep in your mind about this piece.*
- *You have to understand at least those marked with a '*'.*
- If you are not sure about if your method works, *always* write a driver to test it out.

4.1. (*) 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.

A solution: Some of you put in 2^n to get the power, which won't work since Java does not recognize it. You have to use the `Math.pow` method to do it.

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

4.2. 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.

A solution:

```
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!");
}
```

4.3. (*) Write a method called sum100 that returns the sum of the integers from 1 to 100, inclusive.

A solution: Most of you got this right.

```
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);
}
```

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

A solution:

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

   if (num2 > num1)
      result = num2;

   return result;
}
```

4.5. (*) 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).

A solution: You get to understand the requirement first. For this one, it has to add up all the integers within the range, inclusively; but not just the sum of the two inputs. For example, if the two parameters are 1 and 3, respectively, it should output 6(=1+2+3), but not 4(=1+3).

```
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;
}
```

4.6. 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.

A solution:

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

4.7. (*) Write a method called countA that accepts a String parameter and returns the number of times the character 'A' is found in the string.

A solution: Most of you did this right.

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

4.8. 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.

A solution:

```
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;
}
```

4.9. (*) Write a method called average that accepts two integer parameters and returns their average as a floating point value.

A solution: The important thing for this and the next two is that you have to convert the result to a double, which could be done by using an automatic promotion as indicted by the 2.0 in the following solution, or like Eric did as follows:

```
return ((float) num1+num2)/2;
```

Now a sample solution:

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

4.10. (*) Overload the average method of Exercise 4.9 such that if three integers are provided as parameters, the method returns the average of all three.

A solution:

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

4.11. (*) Overload the average method of Exercise 4.9 to accept four integer parameters and return their average.

A solution:

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

4.12. (*) 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.

A solution: The mistake many of you made is that you have to use something, the variable `result` in the following example, to hold the concatenated string.

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

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

   return result;
}
```

4.13. (*) 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"

A solution: Most of you did this right.

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

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

A solution: Most of you did this right.

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

4.15. 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.

A solution:

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

4.16. (*) 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.

A solution: Two issues: one is again, you have to use a container, the variable `result` in the following example, to hold the concatenated string. The other is that you have to initialize it with the empty string, as we discussed in the class.

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

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

   return result;
}
```

4.17. 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.

A solution:

```
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;
   }
```

4.18. (*) 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.

A solution: Many of you made a mistake here and there for this one. If you still do not completely understand it, I would suggest you to have another look at the second part of the Pre-lab 2. The call to `generator.nextInt(range))` sends back a random value between 0 and `second-first,` thus, the variable `result` sends back what is required, with the offset.

```
// 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;
}
```

4.19. (*) 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).

A solution: Some of you used a value of 255 in the random number generator, which leads to a random value between 0 and 254.

```
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);
}
```

4.20. (*) 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.

A solution: All of you did this wrong. Two issues: one is that you forgot to put in the `Graphics page` parameter into the head, without which this circle won't show. The other is that in the normal signature of `drawOval(x, y, h, d,)` the pair `(x, y)` stands for the *to-pright* corner of the box in which the circle is drawn and the pair `h, d` stands for the height and the width of the box. But, in this problem and the next two, `(x, y)` stands for the center, and `r` stands for the radius.

If you think a little bit, you will see that the top-right corner of this box should be `x-r, y-r,` and both the height and the width should be `2*r.`

One more thing, one of you tried to use the method `drawCircle,` which Java won't understand.

```
//  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);
}
```

4.21. (*) Overload the drawCircle method of Exercise 4.20 such that if the Color parameter is not provided, the circle's color will default to black.

A solution:

```
//  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);
}
```

4.22. (*) Overload the drawCircle method of Exercise 4.20 such that if the radius is not provided, a random radius in the range 10 to 100 (inclusive) will be used.

A solution:

```
//  assumes java.awt.* and java.util.Random are imported
public void drawCircle (Graphics page, int x, int y, Color color)
{
   page.setColor (color);
   Random generator = new Random();
   int rad = generator.nextInt(91) + 10;
   page.drawOval (x-rad, y-rad, rad*2, rad*2);
}
```

4.23. (*) Overload the drawCircle method of Exercise 4.20 such that if both the color and the radius of the circle are not provided, the color will default to red and the radius will default to 40.

A solution:

```
//  assumes java.awt.* is imported
public void drawCircle (Graphics page, int x, int y)
{
   final int RAD = 40;
   page.setColor (Color.red);
   page.drawOval(x-RAD, y-RAD, RAD*2, RAD*2);
}
```