Writing Methods (with solutions)

For each exercise below, write the method described. Give all of the methods public visibility. Assume all ranges are inclusive (include both end points).

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.

```
public void powersOfTwo ()
{
   int power = 1;
   for (int count=1; count <= 10; count++)
   {
      power *= 2;
      System.out.println (power);
   }
}
```

2. Write a method called **alarm** that prints the word "Alarm!" multiple times on separate lines. The method should accept an integer parameter that specifies how many times the output line is printed.

```
public void alarm (int num)
{
   for (int count=1; count <= num; count++)
      System.out.println ("Alarm!");
}
```

3. Write a method called **sum100** that returns the sum of the integers from 1 to 100.

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

4. Write a method called **sumRange** that accepts two integer parameters that represent a range. You may assume the first parameter is less than or equal to the second. The method should return the sum of the integers in that range.

```
public int sumRange (int low, int high)
{
   int sum = 0;
   for (int count=low; count <= high; count++)
      sum += count;
   return count;
}
```

5. 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;
}
```

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.

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

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

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

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 vise versa, and false otherwise. You may assume that neither parameter is zero.

```
public boolean evenlyDivisible (int num1, int num2)
{
   return ( (num1%num2)==0 || (num2%num1)==0 );
}
```

9. Write a method called **average** that accepts three integer parameters and returns their average as a floating point value.

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

10. Overload the **average** method of the previous exercise such that if four integers are provided as parameters, the method returns the average of all four.

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

11. Overload the **average** method once more to accept five integer parameters and return their average.

```
public double average (int num1, int num2, int num3,
                       int num4, int num5)
{
   return ((num1+num2+num3+num4+num5) / 5.0);
}
```

12. Write a method called **multiConcat** that takes a `String` and an integer as parameters, and returns a `String` that is the parameter string concatenated with itself n number of times (where n is the second parameter). For example, if the parameters are "hi" and 4, the return value is "hihihihi".

```
public String multiConcat (String str, int max)
{
   String result = "";
   for (count=1; count <= max; count++)
      result += str;
   return result;
}
```

13. Overload the **multiConcat** method from the previous example 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 str)
{
   return (str + str);
}
```

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.

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

15. Write a method called **validate** that accepts three integer parameters. The first two parameters represent a range, and the purpose of the method is to verify that the value of the third parameter is in that range. You may assume that the first parameter is less than or equal to the second. If the third parameter is not in the specified range, the method should prompt the user and read a new value. This new value should be tested for validity as well. The method should only return to the calling method once a valid value has been obtained, and it should return the valid value.

```
public int validate (int low, int high, int num)
{
   while (num < low || num > high)
   {
      System.out.print ("Invalid value. Please reenter: ");
      num = Keyboard.readInt();
   }
   return num;
}
```

16. Write a method called **floatEquals** that accepts three floating point values as parameters. The method should return true if the first two parameters are essentially equal, within the tolerance of the third parameter.

```
public boolean floatEquals (double num1, double num2,
                            double tolerance)
{
   return (Math.abs(num1-num2) < tolerance);
}
```

17. Write a method called **reverse** that accepts a `String` as a parameter and returns a `String` that contains the characters of the parameter in reverse order. Note: there is actually a method in the `String` class that performs this operation, but for the sake of this exercise you will write your own.

```
public String reverse (String str)
{
   String result = "";
   for (int index=str.length()-1; index >= 0; index--)
      result += str.charAt(index);
} return result;
```

18. Write a method called **isIsoceles** that accepts three integer parameters that represent the lengths of the sides of a triangle. The method should return 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 s1, int s2, int s3)
{
   return ( (s1 == s2 && s3 != s1) ||
            (s2 == s3 && s1 != s2) ||
            (s1 == s3 && s2 != s1) );
}
```

19. Write a method called **randomInRange** that accepts two integer parameters representing a range. You may assume that the first parameter is less than or equal to the second, and that both are positive. The method should return a random integer in the specified range.

```
public int randomInRange (int low, int high)
{
   return ((int) (Math.random()*(high-low+1)) + low);
}
```

20. Overload the **randomInRange** method of the previous exercise such that if only one parameter is provided, the range is assumed to be from 1 to that value. You may assume the parameter value is positive.

```
public int randomInRange (int high)
{
   return ((int) (Math.random()*high) + 1);
}
```

21. 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).

```
public Color randomColor ()
{
   int redPart = (int) (Math.random()*256);
   int greenPart = (int) (Math.random()*256);
   int bluePart = (int) (Math.random()*256);

   return (new Color(redPart, greenPart, bluePart));
}
```

22. Write a method called **darken** that accepts a `Color` object as a parameter and returns a new Color object that is "darker" than the parameter. Create the darker color using RGB values that are 10 percent less than the original.

```
public Color darken (Color current)
{
   int redPart = (int) (current.getRed() * 0.9);
   int greenPart = (int) (current.getGreen() * 0.9);
   int bluePart = (int) (current.getBlue() * 0.9);

   return (new Color(redPart, greenPart, bluePart));
}
```

23. 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 that define the (x, y) coordinate 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.

```
public void drawCircle (Graphics page, int x, int y,
                        int radius, Color shade)
{
   page.setColor (shade);
   page.drawOval (x-radius, y-radius, radius*2,
                  radius*2);
}
```

24. Overload the **drawCircle** method of the previous exercise such that if the `Color` parameter is not provided, the circle's color will default to black.

```
public void drawCircle (Graphics page, int x, int y,
                        int radius)
{
   page.setColor (Color.black);
   page.drawOval (x-radius, y-radius, radius*2,
                  radius*2);
}
```

25. Overload the **drawCircle** method again such that if the radius is not provided, a random radius in the range 10 to 100 will be used.

```
public void drawCircle (Graphics page, int x, int y,
                        Color shade)
{
   int radius = (int) (Math.random()*91) + 10;
   page.setColor (shade);
   page.drawOval (x-radius, y-radius, radius*2,
                  radius*2);
}
```

26. Overload the **drawCircle** method yet again such that if both the color and radius of the circle are not provided, the color will default to red and the radius will default to 40.

```
public void drawCircle (Graphics page, int x, int y)
{
   page.setColor (Color.red);
   page.drawOval (x-radius, y-radius, 80, 80);
}
```

In a complete program declare ***three*** objects of type coin; flip them simultaneously until one of them produces one hundred (100) *Heads*. **Output** the number of *Heads* of each coin.

Heads = 1
tails = 0

```
import cs1.Keyboard;
public class test
{
  public static void main (String [] args)
  {
    Coin c1 = new Coin();
    Coin c2 = new Coin();
    Coin c3 = new Coin();
    int count1=0, count2=0, count3=0;
    while ((count1!=100)&&(count2!=100)&&(count3!=100))
    {
      c1.flip();
      if (c1.getFace() == 1)
        count1++;
      c2.flip();
      if (c2.getFace() == 1)
        count2++;
      c3.flip();
      if (c2.getFace() == 1)
        count3++;
    }
    System.out.println("The number of Heads of C1 is:"+count1);
    System.out.println("The number of Heads of C2 is:"+count2);
    System.out.print("the number of Heads of C3:"+count3);
  }
}
```