5COSC001W - Tutorial 2 Exercises

Objects, Classes and Methods

The String class provides methods that you can apply to String objects. One of them is the length method. The length method counts the number of characters in a string. For example, the sequence of statements

```
String greeting = "Hello, World!";
int n = greeting.length();
```

sets `n` to the number of characters in the string object `"Hello, World!"` (13).

Let us look at another method of the String class. When you apply the toUpperCase method to a String object, the method creates another String object that contains the characters of the original string, with lowercase letters converted to uppercase. For example, the sequence of statements

```
String river = "Mississippi";
String bigRiver = river.toUpperCase();
```

sets `bigRiver` to the String object `"MISSISSIPPI"`. Similarly, the `toLowerCase` method to a String object creates another String object that contains the characters of the original string, with uppercase letters converted to lowercase.

Write a program with the above code followed a line which constructs a new String object from `bigRiver`, but with every character converted to lowercase. Then, print the new string.

Do you obtain the original string `"Mississippi"` back?

Familiarisation with the Java API (libraries)

1. The API (Application Programming Interface) documentation lists the classes and methods of the Java library.

   Go to:

   `https://docs.oracle.com/en/java/javase/15/docs/api/allclasses-index.html`

   and find out what the method `concat` of the class `String` does. Describe it in your own words.

2. Complete the following program so that it computes a string with the contents "the quick brown fox jumps over the lazy dog" (using the `concat` method), and then prints that string and its length.

```
public class ConcatTester
{
    public static void main(String[] args)
    {
       String animal1 = "quick brown fox";
       String animal2 = "lazy dog";
       String article = "the";
       String action = "jumps over";

       /* Your work goes here */


    }
}
```

3. Do the same as in 2 above, but instead of using the `concat` method, use the `+` operator.

Object References

1. The following program creates a new `Rectangle` and prints its info:

```
import java.awt.Rectangle;

public class RectangleTester
{
    public static void main(String[] args)
    {
       Rectangle r1 = new Rectangle(0, 0, 100, 50);
       /* Your code goes here */
       System.out.println(r1);
       /* and here */
    }
}
```

   Add code to the program above to create a second rectangle with the same values (x, y, width and height) as the first Rectangle. Then, apply the `grow` method to the second rectangle (`grow(10, 20)`) and print both rectangles. For more info on the `grow` method, look at the API documentation.

   You can use the following Rectangle constructor to create the second rectangle:

```
public Rectangle(Rectangle r)
```

   The above constructor constructs a new `Rectangle`, initialized to match the values of the specified `Rectangle`.

   Compile and run your modified program. What is its output?

2. Modify your program and change the line where you create the second rectangle to:

```
Rectangle r2 = r1;
```

Compile and run your program. What is the output? Why?

3. Consider the following program:

```
public class NumberVariablesTester
{
    public static void main(String[] args)
    {
       double n1 = 150;
       double n2 = n1;

       n2 = n2 * 20; // grow n2

       System.out.println(n1);
       System.out.println(n2);
    }
}
```

Notice that this program is very similar to the program you created for the previous excercise, but it uses number variables instead of object references.

Compile and run the program. What is the output? Why? (In your answer, contrast the output of this program to that of the program you used in the previous exercise).

Designing and Implementing a Class

1. In this exercise, you will implement a *vending machine*. The vending machine holds cans of soda. To buy a can of soda, the customer needs to insert a token into the machine. When the token is inserted, a can drops from the can reservoir into the product delivery slot. The vending machine can be filled with more cans. The goal is to determine how many cans and tokens are in the machine at any given time.

   What methods would you supply for a `VendingMachine` class? Describe them informally.

2. Now translate those informal descriptions into Java method signatures, such as

```
public void fillUp(int cans)
```

   Give the names, parameters, and return types of the methods. Do not implement them yet.

3. What instance variables would you supply? *Hint*: You need to track the number of cans and tokens.

4. Consider what happens when a user inserts a token into the vending machine. The number of tokens is increased, and the number of cans is decreased. Implement a method:

```
public void insertToken()
{
   // instructions for updating the token and can counts
}
```

You need to use the instance fields that you defined in the previous problem.

Do not worry about the case where there are no more cans in the vending machine. You will learn how to program a decision "if can count is > 0" later in this course. For now, assume that the `insertToken` method will not be called if the vending machine is empty. What is the code of your method?

5. Now supply a method `fillUp(int cans)` to add more cans to the machine. Simply add the number of new cans to the can count. What is the code of your method?

6. Next, supply two methods `getCanCount` and `getTokenCount` that return the current values of the can and token counts. (You may want to look at the `getBalance` method of the `BankAccount` class for guidance.) What is the code of your methods?

7. Put the implementation of the various methods above and the fields together into a class, like this:

```
class VendingMachine
{
    public your first method
    public your second method
    . . .
    private your first instance field
    private your second instance field
}
```

What is the code for your complete class?

Testing a Class

Now test the implementation of your `VendingMachine` class in the previous exercise with the following test program.

```
public class VendingMachineTester
{
    public static void main(String[] args)
    {
       VendingMachine machine = new VendingMachine();
       machine.fillUp(10); // fill up with ten cans
       machine.insertToken();
       machine.insertToken();
       System.out.print("Token count = ");
       System.out.println(machine.getTokenCount());
       System.out.print("Can count = ");
       System.out.println(machine.getCanCount());
    }
}
```

What is the output of the test program?

Challenge: The Birthday Problem

Suppose that people enter an empty room until a pair of people share a birthday. On average, how many people will have to enter before there is a match? Write a program which calculates this. Assume that an year has 365 days.

Hint: Run experiments to estimate an approximate value of this.