5COSC001W - Solutions to Tutorial 2 Exercises

Objects, Classes and Methods

```
public class StringExample {
    public static void main(String[] args) {
        String river = "Mississippi";
        String bigRiver = river.toUpperCase();

        String lowRiver = bigRiver.toLowerCase();
        System.out.println(lowRiver);
    }
}
```

The output of the program is:

```
mississippi
```

therefore the original string is not obtained, since the first character of the original string has been converted to lowercase.

Familiarisation with the Java API (libraries)

2.
```
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";

       String total = article.concat(" ").concat(
                      animal1).concat(" ").concat(action).concat(
                      " the ").concat(animal2);
       System.out.println(total);
       System.out.println("Length of string: " + total.length());
    }
}
```

3. The only line which needs to be changed is the one which creates string `total`:

```
       String total = article + " " + animal1 + " " + action + " the " + animal2;
```

Object References

1. ```
   import java.awt.Rectangle;
   public class RectangleTester {
       public static void main(String[] args) {
           Rectangle r1 = new Rectangle(0, 0, 100, 50);
           Rectangle r2 = new Rectangle(r1); // or, new Rectangle(0, 0, 100, 50)
           r2.grow(10, 20);

           System.out.println(r1);
           System.out.println(r2);
       }
   }
   ```

   The output of the program is:

   ```
   java.awt.Rectangle[x=0,y=0,width=100,height=50]
   java.awt.Rectangle[x=-10,y=-20,width=120,height=90]
   ```

2. After the modification, the new output will be:

   ```
   java.awt.Rectangle[x=-10,y=-20,width=120,height=90]
   java.awt.Rectangle[x=-10,y=-20,width=120,height=90]
   ```

   This is because reference `r2` is made to point to the same memory location (object) that `r1` points to.

3. The output of the program is:

   ```
   150.0
   3000.0
   ```

   The difference between this program and the previous one is that objects are copied by reference while primitives by value. Thus, the assignment `n2 = n1` will copy the value of `n1` to `n2` but `n1` and `n2` remain allocated in different memory addresses. Changes to the value of one of them will not affect the other one.

Designing and Implementing a Class

```
class VendingMachine {
    private int numCans;
    private int tokens;


    public VendingMachine() {
        numCans = 10;
        tokens = 0;
    }
```
```

```
    public VendingMachine(int cans) {
        numCans = cans;
        tokens = 0;
    }

    public void fillUp(int cans) {
        numCans = numCans + cans;
    }

    public void insertToken() {
       numCans = numCans - 1;
       tokens = tokens + 1;
    }

    public int getCanCount() {
        return numCans;
    }

    public int getTokenCount() {
        return tokens;
    }
}
```

Testing a Class

1. Try and think what is the output without running your program.
2. Now run and code and verify your guess. If the output was different than what you thought it would be, figure out why.

Challenge: The Birthday Problem

For a single run of the problem, i.e. entering the room until someone else with the same birthday is already there, run the following manually a few times.

```
public class BirthdayProblem {
    public static void main(String[] args) {
        /* position i of the array contains true if someone with
           birthday on day i has already entered the room. Otherwise
           it contains false. All elements are initially set to false */
        boolean birthdayInRoom[] = new boolean[365];

        // how many people have entered the room so far
        int number_of_people = 0;

        while (true) {
            ++number_of_people;  // 1 more person has entered the room
```

```
            /* generate a random birthday in the range [0,364] simulating
               that someone with that birthday enters the room */
            int day = (int) (365*Math.random());  // you can use java.util.Random instead
            if (birthdayInRoom[day])  // if true
                break;
            else
                birthdayInRoom[day] = true;
        }

        System.out.println(number_of_people);
    }
}
```

You need to run a large number of simulations to estimate accurately the expected value for this (law of large numbers - see also `https://en.wikipedia.org/wiki/Law_of_large_numbers`).

```
public class BirthdayProblem2 {
    public static void main(String[] args) {
        // how many runs we will execute the simulation
        int number_of_runs = 10000;

        int total_people = 0;

        for (int i=1; i<=number_of_runs; i++) {
            int people = singleRun();
            total_people = total_people + people;
        }

        // the result is the average of the total
        System.out.println(total_people/number_of_runs);
    }


    public static int singleRun() {
        /* position i of the array contains true if someone with
           birthday on day i has already entered the room. Otherwise
           it contains false. All elements are initially set to false */
        boolean birthdayInRoom[] = new boolean[365];

        // how many people have entered the room so far
        int number_of_people = 0;

        while (true) {
            ++number_of_people;  // 1 more person has entered the room

            /* generate a random birthday in the range [0,364] simulating
               that someone with that birthday enters the room */
```

```
            int day = (int) (365*Math.random());  // you can use java.util.Random instead
            if (birthdayInRoom[day])  // if true
                break;
            else
                birthdayInRoom[day] = true;
        }

        return number_of_people;
    }
}
```