Questions and Exercises

These questions and exercises is an opportunity to see what you've learnt from the lecture as well as practice the new things we've been talking about. In other words, these questions and exercises are completely optional but it's recomended to do them. In the end of the document you will find the answers to the questions as well as possible solutions to the exercises, note that one can solve an exercise in different ways. There will also be some suggestions about what one could code if one want to continue with some more advanced things. These suggestions will not come with a possible solution and might include things that haven't been covered in the lecture.

Question 1
When using switch statements one quite often uses break statements. Why? And what happens if they aren't used?

Question 2
The default statement can be used when using a switch statement. Where in the switch statement do you place it? And what does it do?

Question 3
The following code is supposed to print out the names of people but something is wrong. What's wrong and how is it supposed to be?

```
public static void main(String[] args) {
   myMethod("Alice");
   myMethod("Greg");
}

static void myMethod(name) {
   System.out.println(name);
}
```

Question 4
One can use the keyword void when creating a method. What does it do? And what impact does it have on the method?

Question 5
What would the following code print out on the screen? Answer the question before you run the code.

```
public static void main(String[] args) {
        System.out.println(multByTwo(getSum(1, multByTwo(10))));
}
```

```
static int getSum(int number1, int number2) {
        return number1 + number2;
}

static int multByTwo(int number) {
        return number * 2;
}
```

Exercise 1
Write a program that gives the user multiple options. These options can be answered with an integer and is then controlled by a switch statement. When the code for the chosen option has been executed the options should be presented to the user again(unless the user chose to exit the program). The options the user have are the following

1. *Enter new string*
2. *Reverse string*
3. *Remove first character of string*
4. *Exit program*

Exercise 2
Write a method that is used to simulate five dice beeing thrown. The method should return the value of all the dice. Use this method in your program and use it to print out how many ones you have, how many twos you have etc.

Further explorations
Use Exercise 2 as your starting point and write a game of Yahtzee.

Answers and solutions

Answer to Question 1
Break statements are used to exit the switch statement. This is usually used so the code for only one case will be run. If the break statements aren't used the next case will be run, and the next after that and so on. However, sometimes this exactly what one wants to achieve, and therefore one can ignore the break statements.

Answer to Question 2
The default case is used like any other case, when it's used the code below it will be run if no other case is matched. It's quite common to use the default case at the end of the switch statement but it's not required. In the following code you can see an example of a switch statement using the default case in the beginning

```
switch(myVariable) {
        default:
                System.out.println("Value is something else");
```

```
            break
        case 0:
            System.out.println("Value is zero");
            break;
        case 1:
            System.out.println("Value is one");
            break;
        case 2:
            System.out.println("Value is two");
}
```

Answer to Question 3
When creating methods with parameters you will have to define the type of the parameters, just like when you declare a variable. The proper code would look like this

```
public static void main(String[] args) {
   myMethod("Alice");
   myMethod("Greg");
}

static void myMethod(String name) {
   System.out.println(name);
}
```

Answer to Question 4
The void keyword is the return type of the method. By using void instead of a proper type(like int or String) this method won't return any value and therefore a return statement is not required. An empty return statement can however still be used to exit the method.

Answer to Question 5
The given code will print 42.

The methods that are being used both return an integer. The getSum one requires two integer parameters wihle the multByTwo only requires one. Since all of the parameters and return values are integers it's very easy to chain the method calls together, which is exactly what has been done. First we double 10 to get 20. This is added to 1 which gives us 21 which finally is being multiplied by 2 to get 42.

Possible solution to Exercise 1

```
import java.util.Scanner;

public class Exercise1 {

   public static void main(String[] args) {
      Scanner myScanner = new Scanner(System.in);
      String str = "";
```

```
        while (true) {
                            //Print out the current String and all the options
            System.out.println("The current string is \"" + str + "\"");
            System.out.println("What would you like to do?");
            System.out.println("1. Enter a new string");
            System.out.println("2. Reverse string");
            System.out.println("3. Remove first character of string");
            System.out.println("4. Exit program");

            switch(myScanner.nextInt()) {
               case 1: //Enter a new String
                  System.out.println("Please enter your new string");
                  str = myScanner.next();
                  break;
               case 2: //Reverse the String
                  String temp = str;
                  str = "";
                  for (char c : temp.toCharArray()) {
                     str = c + str;
                  }

                  break;
               case 3: //Remove the first character if there is one
                  if (str.length() >= 1) {
                     str = str.substring(1);
                  }
                  break;
               case 4: //Exit the program
                  return;
               default:
                  System.out.println("Invalid option");

            }
        }

    }

}
```

Possible solution to Exercise 2

```
import java.util.Random;

public class Exercise2 {
    public static void main(String[] args) {
        Random myRandomGenerator = new Random();
```

```
        int[] dice = rollDice(myRandomGenerator, 5);
        int[] values = new int[6];

        //calculate the amounts
        for (int die : dice) {
            values[die - 1]++;
        }

        for (int i = 0; i < values.length; i++) {
            System.out.println(values[i] + " dice with number " + (i + 1));
        }
    }

    //Roll the decired amount of dice
    static int[] rollDice(Random random, int amount) {
        int[] dice = new int[amount];
        for (int i = 0; i < amount; i++) {
            dice[i] = rollDie(random);
        }
        return dice;
    }

    //Roll a single die
    static int rollDie(Random random) {
        return random.nextInt(6) + 1; //return a number between 1 and 6
    }
}
```