

# Tutorial 3

## 4COSC010C Software Development II

### Loops, Exceptions and Debugging

#### Aim:

- Get familiar with loops, data validation and debugging.
- Consolidate learning from lecture week 3.
- Get feedback.
- Submit tutorial answers (pdf file with source codes) to BB prior to the deadline provided in BB.
- You can use IntelliJ IDE (<https://www.jetbrains.com/idea/>) for all the tutorials from this week onwards.

#### Section 01 : Main Questions


##### Q1: Loops

Write a program that **both for loop and while loop** to print the following:

|      |       |        |
|------|-------|--------|
| i) 1 | ii) 0 | iii) 0 |
| 2    | 2     | 2      |
| 3    | 4     | 4      |
| 4    | 8     | 8      |
| 5    | 10    | 10     |
|      | 12    | 12     |
|      | 14    | 14     |

##### Q2: Nested Loops

Extend the previous program (exercise 3) to print the following patterns using nested for loops.

|       |          |   |   |
|-------|----------|---|---|
| 1     | 22       |  |  |
| 22    | 4444     |   |   |
| 333   | 666666   |   |   |
| 4444  | 88888888 |   |   |
| 55555 |          |   |   |

*Hint: You can use `System.out.print("\n")` or `System.out.println()` to print a new line.*

##### Q3: Positive Integers

Write a java program using a while loop to accept an positive integer  $n$  and a letter from the user to print the letter  $n$  times. Sample output is given below.

```
Enter a number : 5
Enter a letter : Y
output : YYYYY
```

**Q4: Factorial**

Design a Java program that calculates and prints the factorial of a given positive number.

Implement the solution using a for loop.

*(The factorial of a number is the multiplication of all positive integers smaller than the given number.)* Sample outputs are given below.

```
Factorial of 3 = 3 x 2 x 1 = 6.  
Factorial of 4 = 4 x 3 x 2 x 1 = 24.  
Factorial of 0 is 1.
```

**Q5: The Fibonacci series**

The Fibonacci series is as follows:

1, 1, 2, 3, 5, 8, 13, 21, 34

Can you see how each number is generated? After the first two numbers, each number is the sum of the previous two numbers. This series is unusual in that it can be used to describe natural phenomena, such as angular displacement of plant leaves for maximizing incident light. ([Read](#))

Write a program to compute this series to the  $n^{th}$  term.

**Q6. Data input validation – Practice for the lab based Assessment.**

In week 2, exercise 5, you had to implement a simple calculator. Add data input validation into your calculator. Think about which inputs need to be validated. **Use appropriate exception handling / input validation techniques.**

```
Scanner scanner = new Scanner(System.in);  
System.out.print("Enter the first number: ");  
double number_1 = scanner.nextDouble();  
System.out.print("Enter the second number: ");  
double number_2 = scanner.nextDouble();  
System.out.print("Enter the operator (+, -, *, /): ");  
String operator = scanner.next();
```

## **Section 02 (Challenging Questions)**

### **Q7: Passcode.**

Write a program that checks if a passcode is correct. The user has **four attempts** to input the passcode correctly. The passcode is 486251. To read an int, you can use `input.nextInt()`. If the user enters the passcode correctly should display a message saying "Correct passcode" and the program should end.

*Note: Think about which control structure you will use to allow the user inserting a maximum of 4 passcodes. Do not repeat your code 4 times. Will you use an if, a for loop, a while loop, a switch? Think about the most adequate control structure.*

### **Q8: Secret number.**

Write a program that generates a random number between 1 and 20 and asks the user to guess the number. The user should be able to enter a new number if the number is incorrect.

*Note: Use Random to generate a random number. To use Random, you will have to import it first: `import java.util.Random`.*

Read below source to learn more about random numbers

<https://www.javatpoint.com/how-to-generate-random-number-in-java>

### **Q9: Armstrong numbers.**

Write a program that prints all Armstrong numbers between 1 and 500. An Armstrong number is a number that the sum of cubes of each digit is equal to the number itself.

Example:

$153 = (1*1*1) + (5*5*5) + (3*3*3)$ .

*Hint: To get the remainder of a division, you can use the symbol %.*

### **Q10: Debugging**

See the program below.

Number to the power 'power' is `number * number * number ....` (power number of times) This program multiplies number by number while the counter is less than power.

Why doesn't it work? How do we debug?

```
import java.util.*;
import java.io.*;
public class Main {
    public static void main(String[] args)
    {

        int number, power, count;
        int total = 0;
        Scanner input = new Scanner(System.in);
        System.out.println("Enter number : ");
        number = input.nextInt();
        System.out.println("Enter power ");
        number = input.nextInt();
        count = 0;
        while (count <= power )
        {
            total = number * number;
        }
        System.out.println(" the answer is " + total);
    }
}
```

### **Section 03 : HackerRank Challenges**

Register in HackerRank (see tutorial notes from week 1) and solve the following exercises:

- Java Loops I
- Java Loops II
- Java output formatting