Umm Al-Qura university
Computer & Information Systems college
Instructor: Dr. Aymen M. Akremi

Lab 07: Object Oriented Programming

Objectives:

- This lab will give you practical implementation of different types of **Exceptions** and **Assertions** in java.

Exercise 1

```
import java.util.Scanner;
public class Exception1 {
  public static void main(String[] args)
    {
        Scanner scan = new Scanner(System.in);
        int num = 0;
        do {
              System.out.println("Enter a number between 1 and 10");
              num = scan.nextInt();

              if (num < 1 || num > 10)

              System.out.println("\nIllegal value, " + num + " entered.
Please try again.");
           }  while (num < 1 || num > 10);

              System.out.println("\nValue correctly entered! Thank you.");
      }
 }
```

Questions:

1. Type the program above and compile. Run and enter an integer between 1 and 10.
2. The program is requesting a number between 1 and 10. Run the program again and enter 5.5. Although this number is between 1 and 10, the program will abort. Examine the error message. You should see the word Exception, the method where the exception occurred (main), the class name of the exception (InputMismatchException), as well as the call stack listing the method calls.
3. Add a try/catch block to catch and handle the InputMismatchException exception. Identify the statements that cause the error as well as the portions of the program that depend upon these statements. Enclose these statements within the try block. Follow the try block with the catch block given below. Note, the InputMismatchException class is defined in java.util and must be imported. Also, when the Scanner throws an InputMismatchException, the input token will remain in the buffer so that it can be examined by the program. In our case, we will not be examining the token, but will simply clear out of the buffer to start over.

```
catch (InputMismatchException ime) {

    System.out.println("Enter whole numbers only, with no spaces or other
characters");
    scan.next();        // clear the scanner buffer
 }
```

4. Compile and run the program again, testing with a variety of input (integers, floats, characters) The program should not abort when floats or character data is given.
5. Complete the security checklist for Program 1.

Exercise 2

Capture all exceptions in the following program, printing out error messages that describe the type of error that occurred.

```
import java.io.*;
public class Exercise2 {
        public static void main(String[] args) {
                int n=10; int[] v = new int[n];
                FileReader f = new FileReader("dati.txt");
                BufferedReader in = new BufferedReader(f);
                int i=0;
                String linea = in.readLine();
                while (linea!=null) {
                        v[i] = Integer.parseInt(linea);
                        linea = in.readLine();
                        i++;
                }
                f.close();
        }
}
```