**Department of Computer Applications**

**Exception Handling Worksheet 9**

**Lab Practice Program 1**

Write a Java code to complete by writing down the handlers for exceptions thrown by the code. The exceptions the code may throw along with the handler message are listed below:

**Division by zero**: Print "Invalid division".  
**String parsed to a numeric variable** : Print "Format mismatch".  
**Accessing an invalid index in string** : Print "Index is invalid".  
**Accessing an invalid index in array** : Print "Array index is invalid".

**MyException** : This is a user defined Exception which you need to create. It takes a parameter

When an exception of this class is encountered, the handler should print "MyException[param]", here is the parameter passed to the exception class.

**Exceptions other than mentioned above**: Any other exception except the above ones fall in this category. Print "Exception encountered".

Finally, **after the exception is handled, print "Exception Handling Completed"**.

Example: For an exception of MyException class if the parameter value is 5, the message will look like MyException[5].

**Input Format**:  
The code handles all the input itself.

**Output Format**:  
If any exception is encountered in the code, print the respective handler code.  
The last line of output should be "Exception Handling Completed".

Sample Input

10

10

9

8

7

6

5

4

3

2

1

exceptionhandlingtime

Sample Output

a

MyException[117]

Exception Handling Completed

**Lab Practice Program 2**

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.");

}

}

**Lab 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.