Don Bosco Institute of Technology, Kurla(W)
Department of Electronics and Tele-Communication Engineering
ECL304 - Skill Lab: C++ and Java Programming
Sem III
2021-22

Lab Number:	**11**
Student Name:	**Shruti Gokhale**
Roll No :	**40**

Title:

1. Write a program injava if a number is less than 0 and greater than 10 it generates the user-defined exception "out of range". Else it displays the square of the number.
2. Write a program in java to enter the number. If the first and second number is not entered it will generate the exception. Also, divide the first number with the second number and generate the arithmetic exception.

Learning Objective:

Students will be able to implement user-defined exceptions

Learning Outcome:

Understanding the exception handling concept and making the programming interface error-free.

Course Outcome:

ECL304.3	Articulate exception handling methods.

Theory:

- What is exception handling and how is it achieved in JAVA?

Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc. Let's start with the basics of exception handling in Java before we move to more advanced topics. The try-catch is the simplest method of handling exceptions. Put the code you want to run in the try block, and any Java exceptions that the code throws are caught by one or more catch blocks.

This method will catch any type of Java exceptions that get thrown. This is the simplest mechanism for handling exceptions.

You can also specify specific exceptions you would like to catch. This allows you to have dedicated code to recover from those errors. For example, some Java errors returned from a REST API are recoverable and others are not. This allows you to treat those conditions separately. A try block can be followed by one or more catch blocks, each specifying a different type. The first catch block that handles the exception class or one of its superclasses will be executed. So, make sure to catch the most specific class first.

- Explain user defined exceptions in java?

User Defined Exception or custom exception is creating your own exception class and throws that exception using 'throw' keyword. This can be done by extending the class Exception. These predefined exceptions are used to handle errors occurring in the program. But sometimes the programmer wants to create his own customized exception as per the requirements of the application which is called userdefined exception or custom exception. Custom exceptions in Java are those exceptions that are created by a programmer to meet the specific requirements of the application. That's why it is also known as user-defined exception in Java.

Program 1:	import java.io.*;
	import java.util.Scanner;
	class IsNum extends Exception{
	
	public String toString()
	{
	return ("number is not valid it should be an integer : ");
	}
	
	}
	
	class Main{
	
	void test(int num1,int num2)
	{
	try{
	
	int res=num1/num2;
	System.out.println();
	System.out.print(" num1/num2 is: ");
	System.out.println(res);
	}

```
    catch(ArithmeticException e)
    {
        System.out.println(" can't divide by zero "+e);
    }

}

public static void main(String args[])
{
    int num1=0,num2=0;

    Scanner sc = new Scanner(System.in);

    System.out.print("ENTER THE NUMBERS : ");
    try
    {
        if(sc.hasNextInt())
        {
            num1=sc.nextInt();
        }
        else
        {
            throw new IsNum();
        }
        if(sc.hasNextInt())
        {
            num2=sc.nextInt();
        }
```

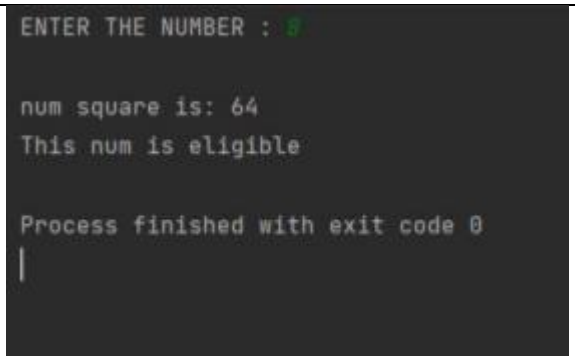
```
            else
            {
               throw new IsNum();
            }
        }

        catch(IsNum u)
        {
            System.out.println(" INVALID ");
            u.printStackTrace();
            System.out.println("his number is not entered");
            System.exit(0);
        }

        System.out.println("numbers are entered ");
        Main e = new Main();
        e.test(num1,num2);
    }
}
```

Input given: **Number= 8**

Output Screenshot:



Program 2:

```
import java.io.*;
import java.util.Scanner;
class IsNum extends Exception{

    public String toString()
    {
        return ("number is not valid it should be an
integer : ");
    }

}

class Main{

    void test(int num1,int num2)
    {
        try{

            int res=num1/num2;
            System.out.println();
            System.out.print("        num1/num2 is:
");
            System.out.println(res);
        }
        catch(ArithmeticException e)
        {
            System.out.println(" can't divide by zero
"+e);
        }

    }

    public static void main(String args[])
    {
        int num1=0,num2=0;

        Scanner sc = new Scanner(System.in);

        System.out.print("ENTER THE NUMBERS : ");
        try
        {
            if(sc.hasNextInt())
            {
```

```
                        num1=sc.nextInt();
                    }
                    else
                    {
                        throw new IsNum();
                    }
                    if(sc.hasNextInt())
                    {
                        num2=sc.nextInt();
                    }
                    else
                    {
                        throw new IsNum();
                    }
                }

                catch(IsNum u)
                {
                    System.out.println(" INVALID ");
                    u.printStackTrace();
                    System.out.println("his number is not
        entered");
                    System.exit(0);
                }

                System.out.println("numbers are entered ");
                Main e = new Main();
                e.test(num1,num2);
            }
        }
```

Output screenshot:

Input given: **Numbers: 0 & 0**