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:	**Deep Patel**
Roll No :	**8**

Title:

1. Write a program in java 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.

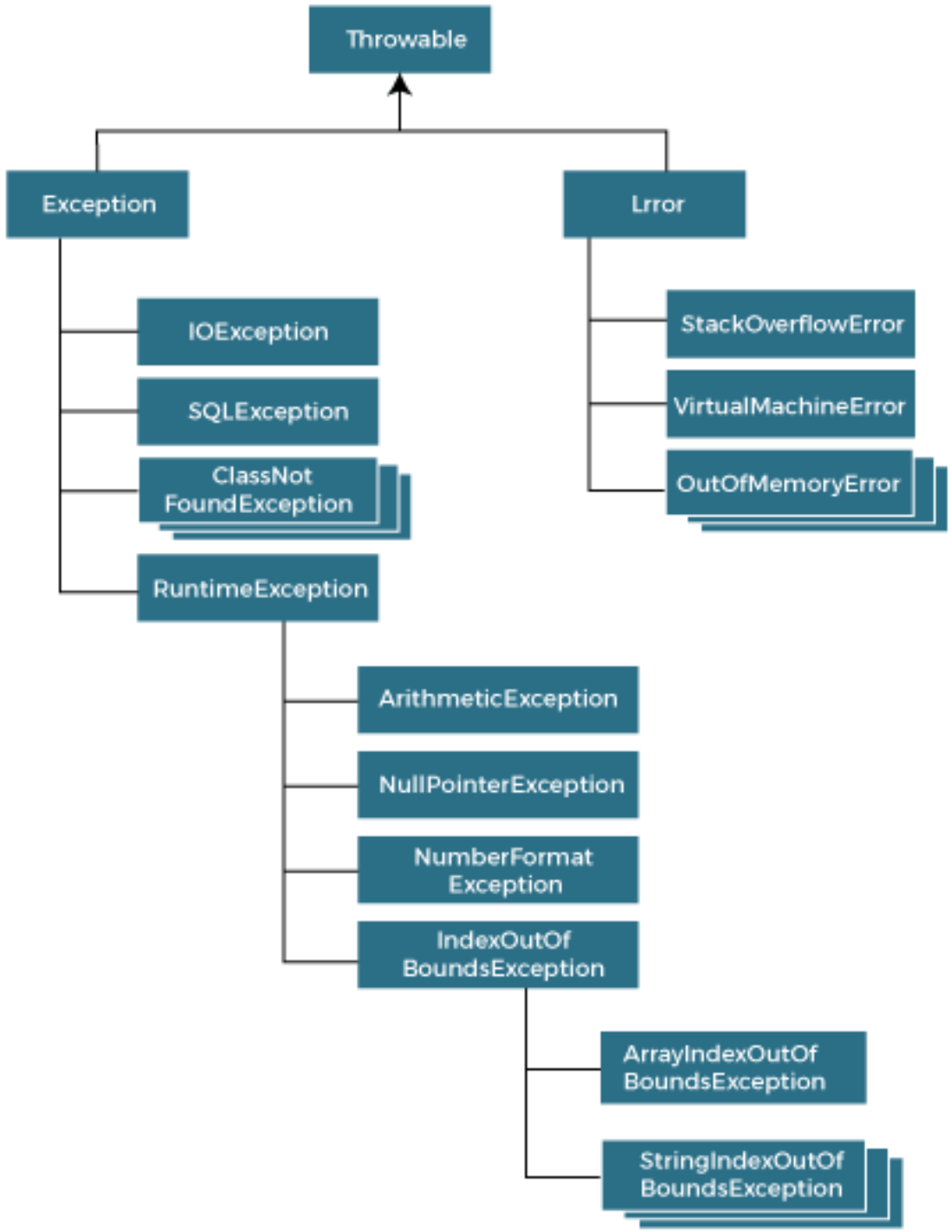
Theory:

1. What is exception handling and how is it achieved in JAVA?

- Exception is defined as a problem that arises during the execution of a program. When an Exception occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore these exceptions are to be handled.
- There is a difference between the terms error and exception. An Error indicates serious problem that a reasonable application should not try to catch. Exception indicates conditions that a reasonable application might try to catch.
- Java exception handling is managed with the help of the following keywords: try, catch, throw, throws, and finally. Program statements that we as programmers think can raise exceptions are contained within a try block. If an exception occurs within the try block, it is thrown. The code can catch this exception (using catch block) and handle it.
- Every try block should be immediately followed either by a class block or finally block. A catch statement involves declaring the type of exception you are trying to

catch. If an exception occurs in protected code, the catch block (or blocks) that follows the try is checked. If the type of exception that occurred is listed in a catch block, the exception is passed to the catch block much as an argument is passed into a method parameter.

- The java.lang.Throwable class is the root class of Java Exception hierarchy inherited by two subclasses: Exception and Error. The hierarchy of Java Exception classes is given below:



2. Explain user defined exceptions in java?

- Java user-defined exception is a custom exception created and throws that exception using a keyword 'throw'. It is done by extending a class 'Exception'. An exception is a problem that arises during the execution of the program. In Object-Oriented Programming language, Java provides a powerful mechanism to handle such exceptions. Java allows to create own exception class, which provides own exception class implementation. Such exceptions are called user-defined exceptions or custom exceptions.
- There are advantages in using these user-defined exceptions; it allows users to throw an exception which wants user wants to mean. Users can also reuse any existing application; any piece of code that catches exception deals with a possibility of actual exception was not thrown by code but by some other thirdparty code.
- User Defined Exception or custom exception is creating your own exception class and throws that exception using 'throw' keyword. ... There is no need to override any of the above methods available in the Exception class, in your derived class.
- We will consider an example of the user defined exceptions as follows:

```
class MyException extends Exception{
        String str1;
        MyException(String str2) {
        str1=str2;
        }
public String toString(){
        return ("MyException Occurred: "+str1);
        }
}
catch(MyException exp){
        System.out.println("Catch Block") ;
        System.out.println(exp) ;
}
```

Program 1:

Algorithm:

STEP 1: Start

STEP 2: Create class OutofRange and Exception extend OutofRange

STEP 3: Write attributes of class

STEP 4: If the number is >10, say out of range

STEP 5: If the number is <11, show its square at Output

STEP 6: Print Output

STEP 7: Stop

Code:

```
package extc1;
import java.util.*;

class OutOfRange extends Exception{
   int num;
   OutOfRange(int a){
      num = a;
   }

   public String toString()
   {
      return ("\nThe number is out of range: "+ num);
   }

}
```

```
class Main{


   void test(int num)
   {
      try{
         if(num<0||num>10)
            throw new OutOfRange(num);


         System.out.println();
         System.out.print("The number square is: ");
         System.out.println( num*num);
      }
      catch(OutOfRange u)
      {
         System.out.println("Out of range!");
         u.printStackTrace();
         System.out.println("This number is not eligible");
         System.exit(0);
      }
      System.out.println("This number is eligible ");
   }


   public static void main(String args[])
   {
      int num;
```

```
        Scanner sc = new Scanner(System.in);


        System.out.print("Enter the Number : ");
        num = sc.nextInt();


        Main e = new Main();
        e.test(num);
    }
}
```

Input Data: 5, 12

Output Data:

```
Enter the Number : 5
The number square is: 25
This number is eligible
```

```
Enter the Number : 12
Out of range!
The number is out of range: 12
at Main.test(Main.java:22)
    at Main.main(Main.java:48)
This number is not eligible
```

Program 2:

Algorithm:

STEP 1: Start

STEP 2: Create class IsNum and extend it to exception

STEP 3: If string is given at IP, say the number should be an integer

STEP 4: In main class, use try and catch for dividing the number with the other number.

STEP 5: If exception is created, display cannot be divided by 0

STEP 6: Create main class, write the code for throw and catch as valid/ invalid nubers, make its objects

STEP 7: Display the OP

STEP 8: Stop

Code:

```
import java.io.*;
import java.util.Scanner;

class IsNum extends Exception {
   public String toString() {
      return ("The 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("The number is not entered");
        System.exit(0);
      }

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

Input Data: 132, 12

Output Data:

```
Enter the Numbers : 132
12
The numbers are entered

 num1/num2 is: 11
```