**Assignment No 4**

**PL-2**

**1. Write a program in java to handle below exceptions**

a. Divide by Zero

public class Q1\_1 {

    public static void main(String[] args){

        int a=5;

        int b=0;

        try{

            System.out.println(a/b);

        }

        catch (ArithmeticException e){

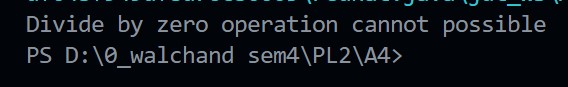
            System.out.println("Divide by zero operation cannot possible");

        }

    }

}

**Output:**

****

b. Array Index Out Of Bound

public class Q1\_2 {

    public static void main(String[] args)

    {

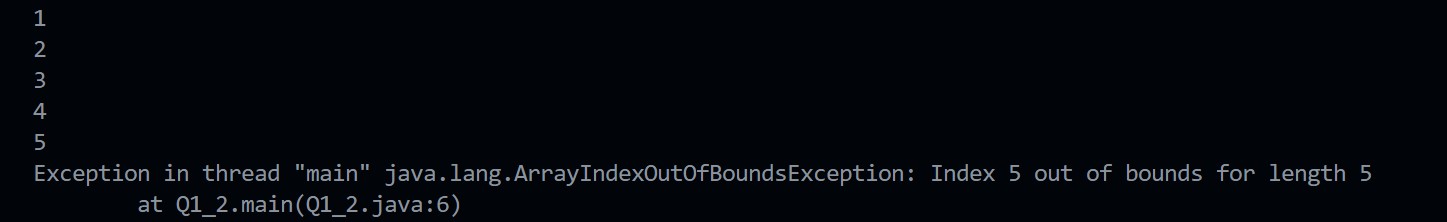
        int ar[] = { 1, 2, 3, 4, 5 };

        for (int i = 0; i <= ar.length; i++)

            System.out.println(ar[i]);

    }

}



c. Number Format

public class Q1\_3 {

    public static void main(String args[])

    {

        String a="50a";

        String b="60";

        try{

            int x= Integer.parseInt(a);

            int y= Integer.parseInt(b);

            int z=x+y;

            System.out.println("sum = "+z);

        }

        catch(NumberFormatException e)

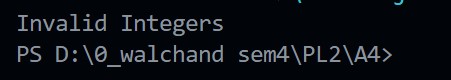
        {

            System.out.println("Invalid Integers");

        }

    }

}



d. Null Pointer

import java.util.Scanner;

public class Q1\_4 {

 public static void main(String args[]) {

 String name = null;

 try {

 System.out.println(name.length());

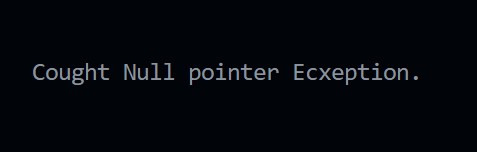
 } catch (NullPointerException e) {

 System.out.println("\n\nCought Null pointer Ecxeption.\n\n");

 }

 }

}



**2. Write a program in java to handle custom exception with single try block and multiple catch block.**

public class Q2 {

    public static void main(String[] args) {

           try{

                String s=null;

                System.out.println(s.length());

               }

               catch(ArithmeticException e)

                  {

                   System.out.println("Arithmetic Exception occurs");

                  }

               catch(ArrayIndexOutOfBoundsException e)

                  {

                   System.out.println("ArrayIndexOutOfBounds Exception occurs");

                  }

               catch(Exception e)

                  {

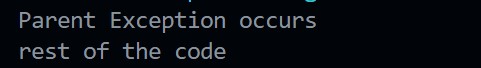
                   System.out.println("Parent Exception occurs");

                  }

               System.out.println("rest of the code");

    }

}



**3. Write a program in java to show the use of finally keyword.**

class FinallyBlock {

    public static void main(String args[]){

    try{

  //below code do not throw any exception

     int data=25/5;

     System.out.println(data);

    }

  //catch won't be executed

    catch(NullPointerException e){

  System.out.println(e);

  }

  //executed regardless of exception occurred or not

   finally {

  System.out.println("This block is always executed");

  }

    }

  }



**4. Write a program in java for handling exceptions with nested try block.**

public class Q4{

    public static void main(String args[]){

     try{

       try{

        System.out.println("going to divide by 0");

        int b =39/0;

      }

       catch(ArithmeticException e)

       {

         System.out.println(e);

       }

       try{

       int a[]=new int[5];

        a[5]=4;

        }

       catch(ArrayIndexOutOfBoundsException e)

       {

          System.out.println(e);

       }

       System.out.println("other statement");

     }

     catch(Exception e)

     {

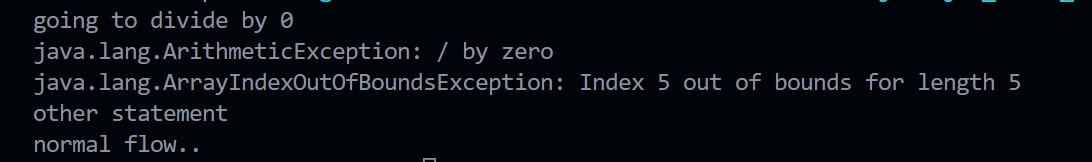
       System.out.println("handled the exception (outer catch)");

     }

     System.out.println("normal flow..");

    }

   }



**5. Write a program in java for custom exception to check speed of car on highway, if speed exceeds 120Km/hr then throw a ‘Speed Limit Exceeded’ exception. (use throw)**

import java.util.Scanner;

public class Q5 {

    static void checkspeed(int speed) {

        if (speed > 120) {

            throw new ArithmeticException("\n\nSpeed limit Exceeded !\n\n");

        } else {

            System.out.println("\n\nYour speed is in limit.\n\n");

        }

    }

    public static void main(String[] args) {

        int speed;

        System.out.println("\n\nEnter the speed of the car : ");

        Scanner s = new Scanner(System.in);

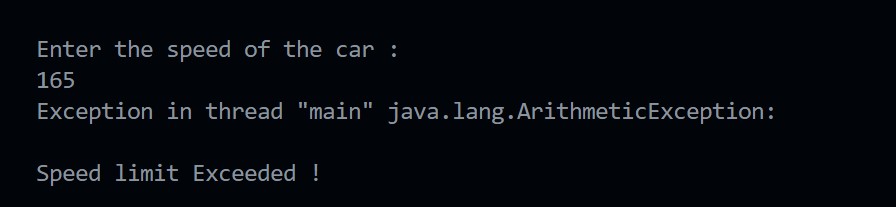
        speed = s.nextInt();

        checkspeed(speed);

    }

}

Output



**6. Differentiate in between throw and throws keyword.**

**Throw keyword:**

The throw keyword is used inside a function. It is used when it is required to throw an Exception logically. The throw keyword is used to throw an exception explicitly. It can throw only one exception at a time. Syntax of throw keyword includes the instance of the Exception to be thrown. Syntax wise throw keyword is followed by the instance variable. throw keyword cannot propagate checked exceptions. It is only used to propagate the unchecked Exceptions that are not checked using the throws keyword.

**Throws keyword:**

The throws keyword is used in the function signature. It is used when the function has some statements that can lead to exceptions. The throws keyword can be used to declare multiple exceptions, separated by a comma. Whichever exception occurs, if matched with the declared ones, is thrown automatically then. Syntax of throws keyword includes the class names of the Exceptions to be thrown. Syntax wise throws keyword is followed by exception class names. throws keyword is used to propagate the checked Exceptions only.

**7. Explain exception handling mechanism.**

The core advantage of exception handling is to maintain the normal flow of the application. An exception normally disrupts the normal flow of the application; that is why we need to handle exceptions. Let's consider a scenario,

Suppose there are 10 statements in a Java program and an exception occurs at statement 5; the rest of the code will not be executed, i.e., statements 6 to 10 will not be executed. However, when we perform exception handling, the rest of the statements will be executed. That is why we use exception handling in Java.

**8. Write a program in java for handling checked exceptions using throws keyword.**

import java.io.\*;

class Q8 {

    public static void findFile() throws IOException {

        File newFile = new File("test.txt");

        FileInputStream stream = new FileInputStream(newFile);

    }

    public static void main(String[] args) {

        try {

            findFile();

        } catch (IOException e) {

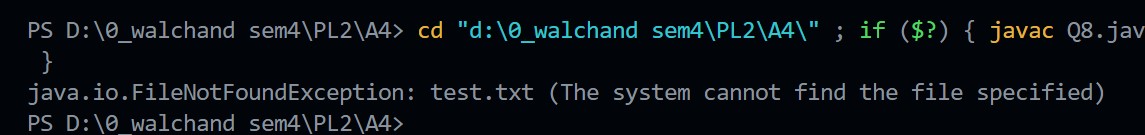
            System.out.println(e);

        }

    }

}

**Output:**

****