EXEPTION HANDLING

IT IS ONE OF THE POWERFUL MECHANISM TO HANDLE RUNTIME ERROR SO THAT NORMAL FLOW OF APPLICATION CAN BE MAINTAINED.

1)TRY CATCH BLOCK:

try {

// code

}

catch(Exception e) {

// code

}

public class errorchecking

{

public static void main (String []args)

try

{

int x=10;

int y= 0;

int z=x/y;

System.out.println(z);

}

catch ( Exeption object )

{

System.out.println(err.get message());

}

}

class Main {

public static void main(String[] args) {

try {

// code that generate exception

int divideByZero = 5 / 0;

System.out.println("Rest of code in try block");

}

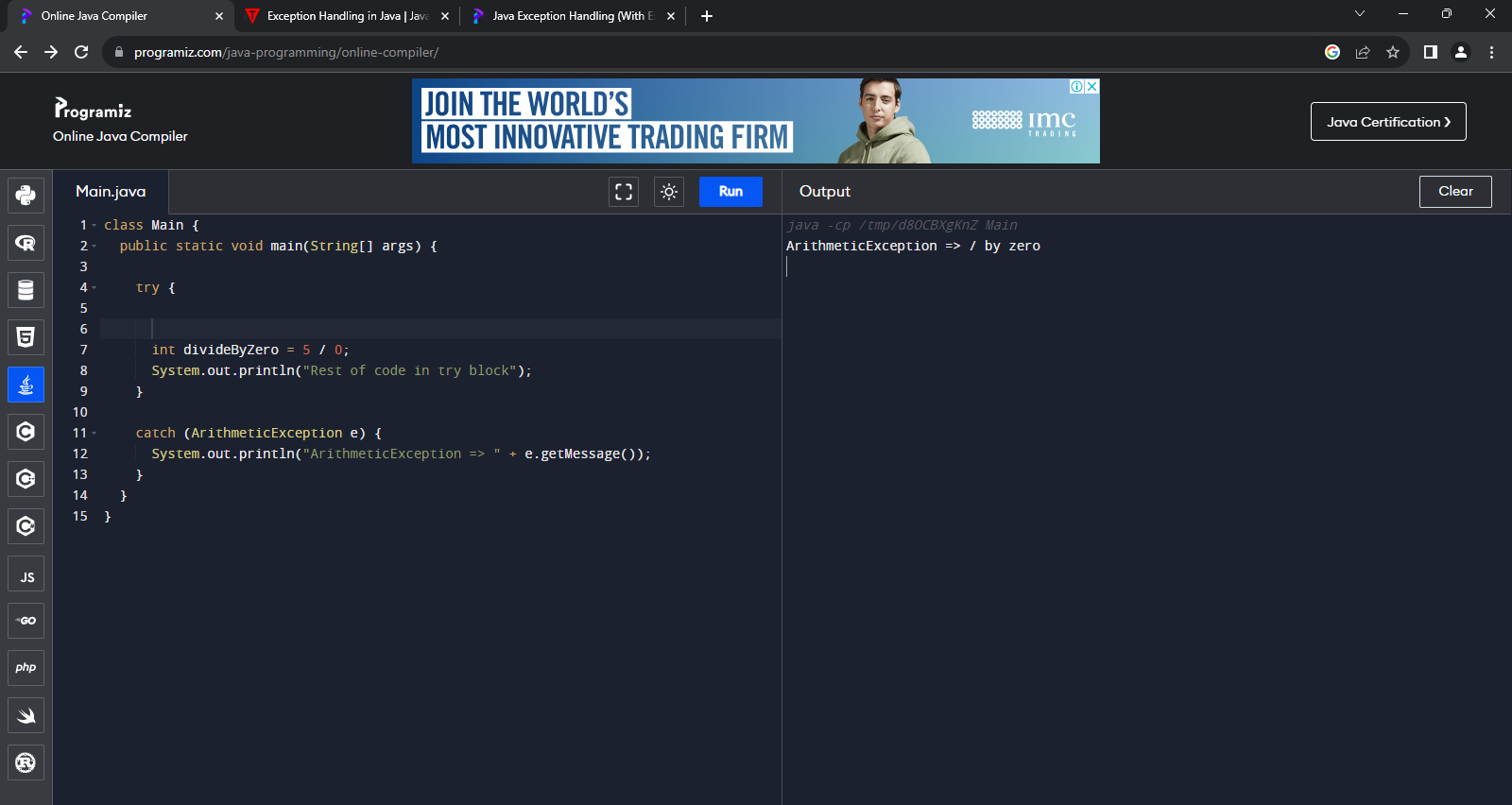
catch (ArithmeticException e) {

System.out.println("ArithmeticException => " + e.getMessage());

}

}

}



Multiple catch .

import java.util.Scanner;

public class Test

{

public static void main(String args[])

{

Scanner scn = new Scanner(System.in);

try

{

int n = Integer.parseInt(scn.nextLine());

if (99%n == 0)

System.out.println(n + " is a factor of 99");

}

catch (ArithmeticException ex)

{

System.out.println("Arithmetic " + ex);

}

catch (NumberFormatException ex)

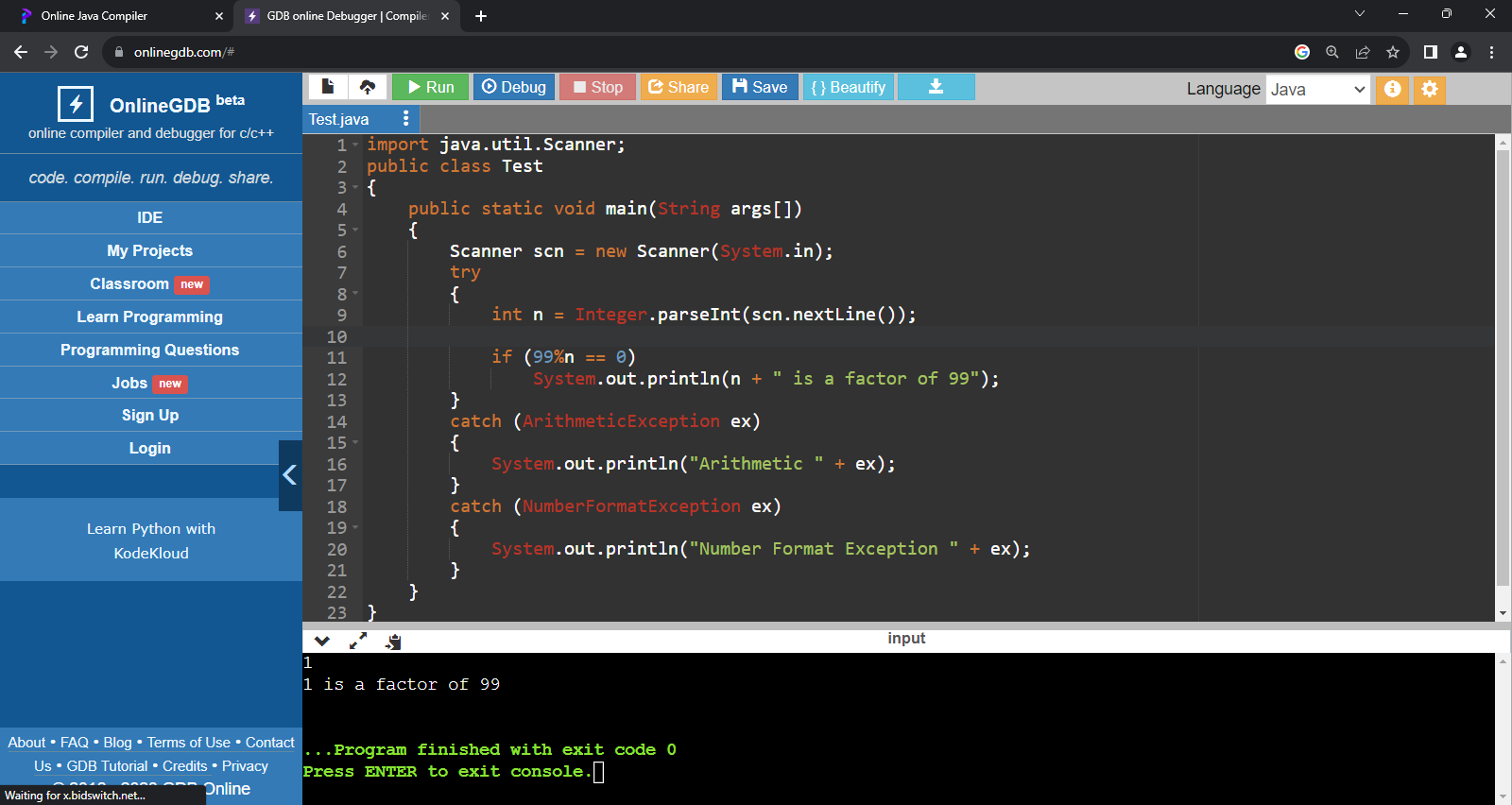
{

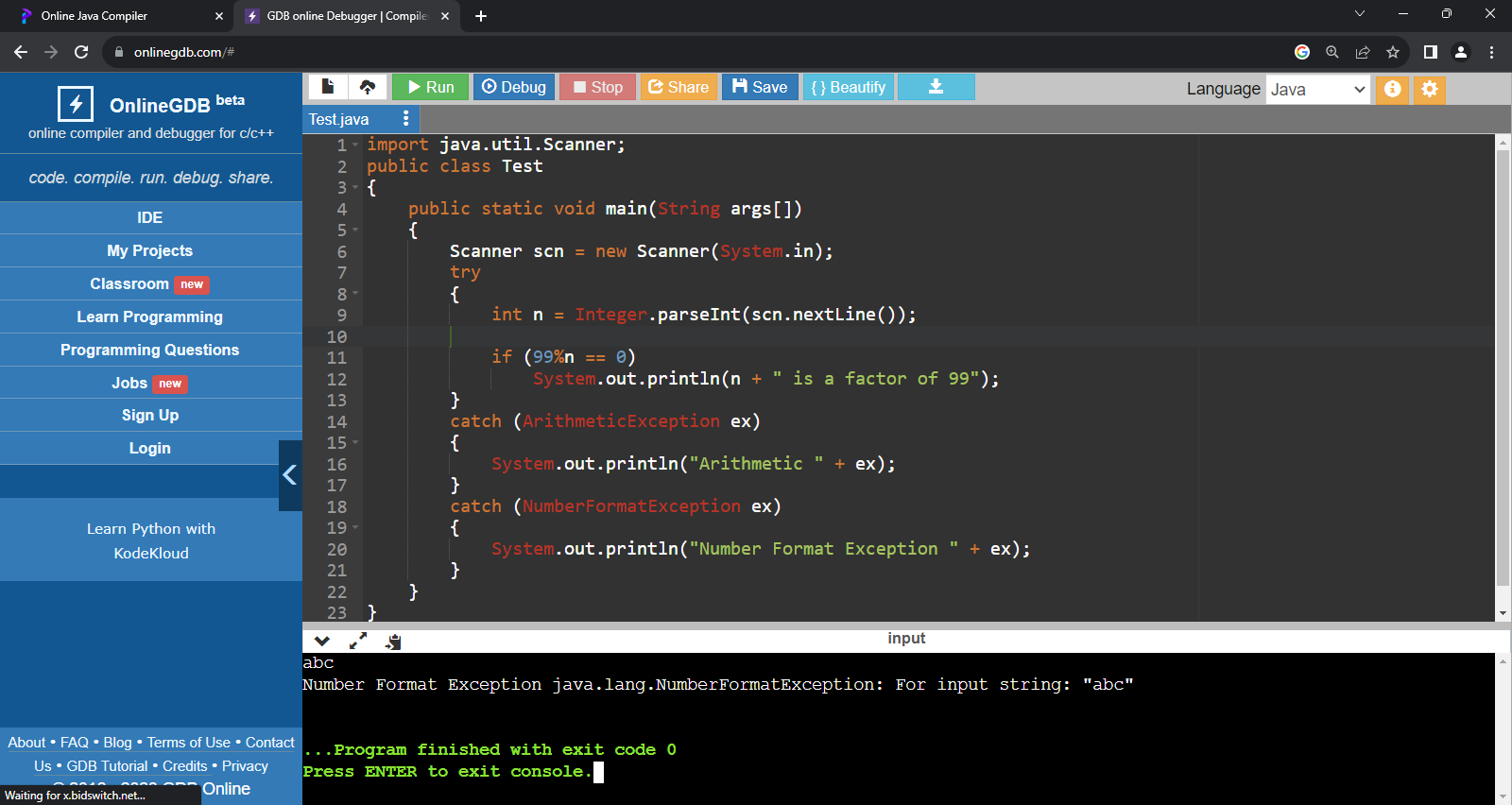
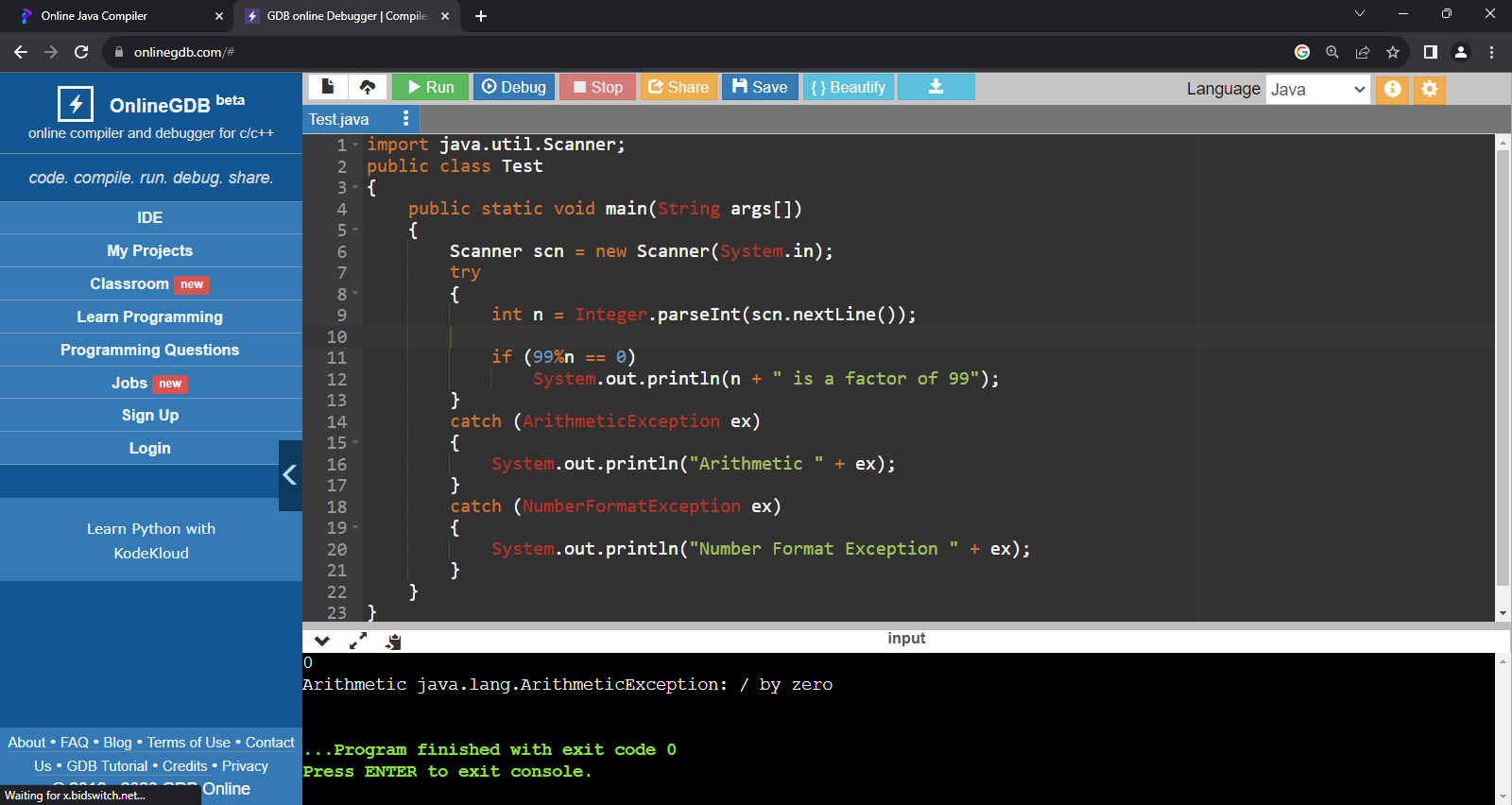
System.out.println("Number Format Exception " + ex);

}

}

}





FINALLY BLOCK.

class Main {

public static void main(String[] args) {

try {

int divideByZero = 5 / 2;

}

catch (ArithmeticException e) {

System.out.println("ArithmeticException => " + e.getMessage());

}

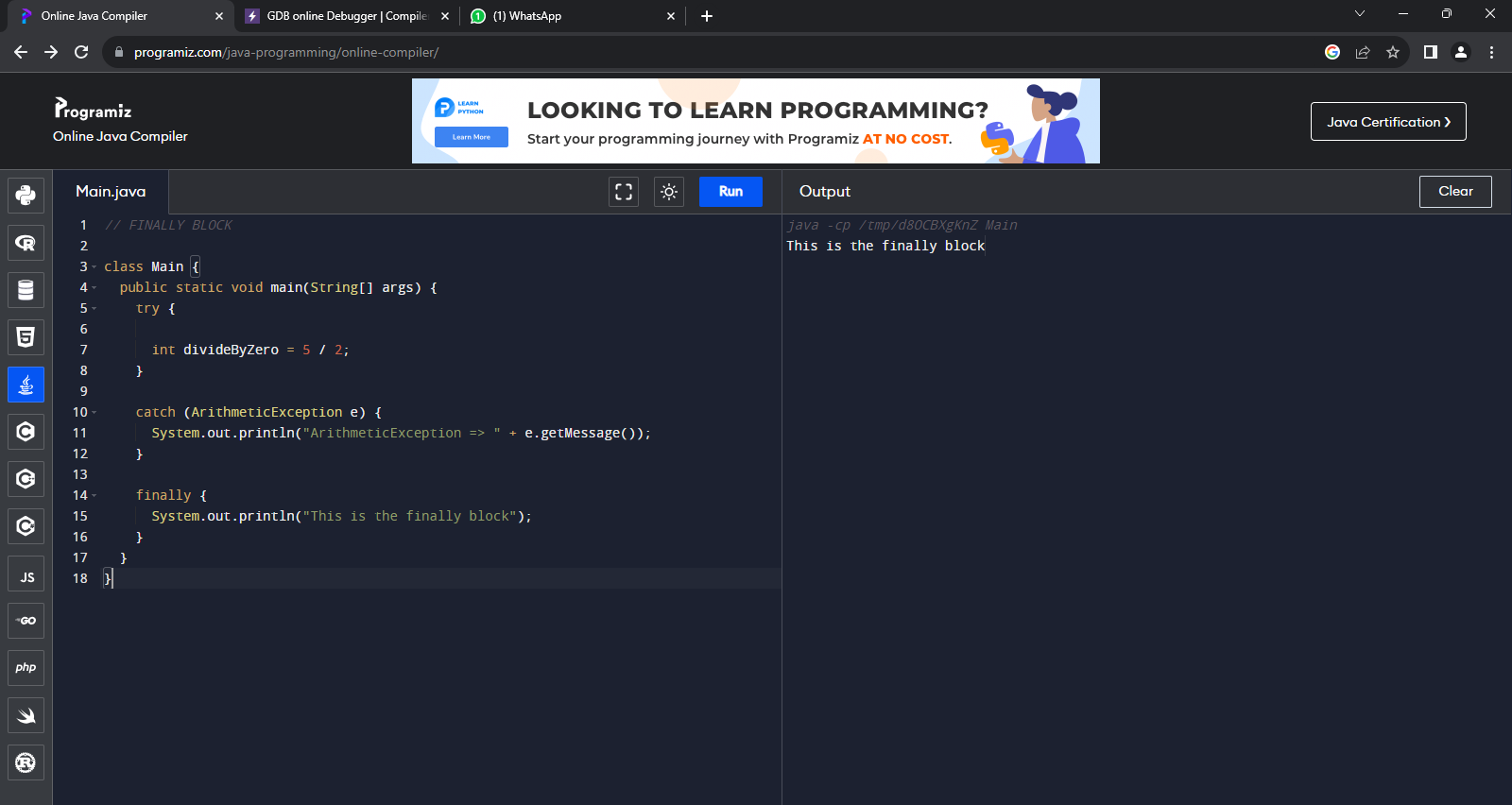
finally {

System.out.println("This is the finally block");

}

}

}



Null pointer exeption

public class NassMultiCatch {

public static void main(String[] args) {

String s=null;

int[] arr = {10, 20, 30};

try {

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

System.out.println(arr[2]);

} catch (ArrayIndexOutOfBoundsException el) {

System.out.println("Exception Index Out of Bounds: " + el);

} catch (NullPointerException e2) {

System.out.println("Exception Null Pointer: " + e2);

}

}}

