14 Oct 2020 Rinoy Kuriyakose R3 56

**Experiment : 9**

**Aim:**

Write a Java program that shows the usage of try, catch, throws and finally.

(a) Try-Finally example

(b) Multiple catch example (3 catches for a single try)

(c) Nested Try (3 levels of nesting)

(d) Throw an exception when there are no sufficient arguments passed

into command line as input for adding two numbers.

(e) Throws example (for handling two exceptions in a method)

**Concept Used:**

Exception Handling.

**Algorithm:**

Algorithm (a)

1. try block –Print a/0

2. finally block –Print message

Algorithm (b)

1. try block –Print arr[size+1], arr[index]/0.

2. catch block 1 – Print IndexOutOfBoundsException object

3. catch block 2 – Print ArithmeticException object

4. catch block 3 – Print Exception object

Algorithm (c)

1. try –

2. Print arr[index]

3. try –

4. Print arr[index]/10

5. try –

6. Print arr[index]/0

7. catch –Print ArithematicException object

8. catch –Print Exception object

9. catch –Print ArithematicException object

10. catch –Print ArrayIndexOutOfBoundsException object

Algorithm (d)

1. if(input.length != 2)

2. throw RuntimeException(“Enter two numbers”)

3. else

4. Print Integer.parseInt(input[0])+Integer.parseInt(input[1])

5. endif

Algorithm (e)

1. if(n<10)

2. throw ArithematicException

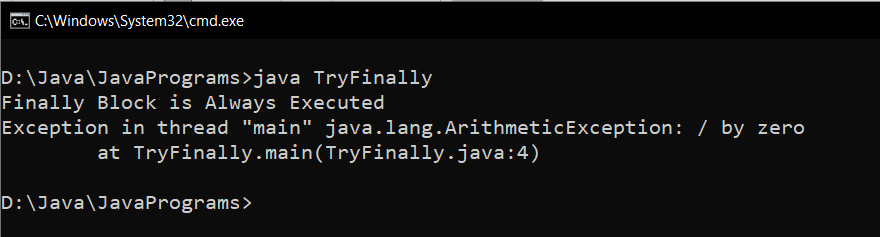
3. else

4. print Number Greater than 10

**Program (a): Try-Finally example**

public class TryFinally{  
 public static void main(String args[]){  
 try{  
 int a=5/0;  
 System.out.println(a);  
 }  
 finally{  
 System.out.println("Finally Block is Always Executed");  
 }  
 }  
}

**Output (a):**

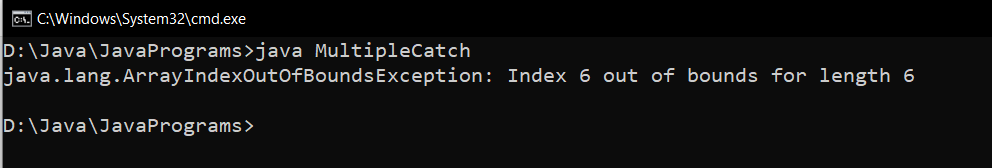


**Program (b): Multiple catch example**

public class MultipleCatch {  
 public static void main(String args[]) {  
 int[] arr = {67, 20, 39, 2, 10, 18};  
 try {  
 int res = arr[6];  
 int div = arr[2]/0;  
 System.out.println(res);  
 }  
 catch(ArrayIndexOutOfBoundsException exp1){  
 System.out.println(exp1);  
  
 }  
 catch(ArithmeticException exp2) {  
 System.out.println(exp2);  
 }

catch(Exception exp3) {  
 System.out.println(exp3);  
 }  
  
 }  
}

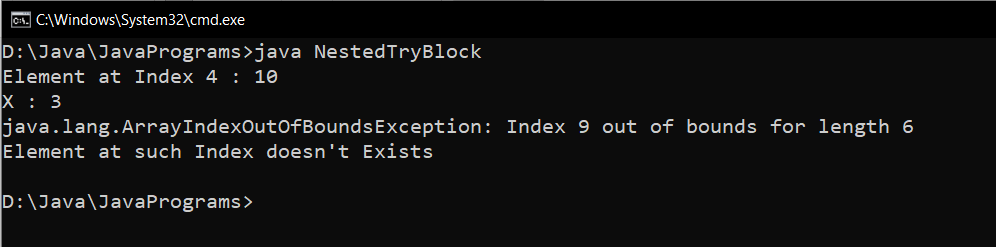
**Output (b):**



**Program (c): Nested Try Block**

class NestedTryBlock{  
  
 public static void main(String args[]){  
  
 try {  
  
 int[] arr = {67, 20, 39, 2, 10, 18};  
 System.out.println("Element at Index 4 : " +arr[4]);  
 try {  
  
 int x = arr[2]/ 10;  
 System.out.println("X : " +x);  
 try {  
  
 int y = arr[9]/ 0;  
 System.out.println("Y : " +y);  
 }  
 catch (ArrayIndexOutOfBoundsException e3) {  
 System.out.println(e3);  
 System.out.println("Element at such Index doesn't Exists");  
 }  
 catch (Exception e4) {  
 System.out.println(e4);  
 }  
 }  
 catch (ArithmeticException e2) {  
 System.out.println(e2);  
 }  
 }  
 catch (ArrayIndexOutOfBoundsException e1) {  
 System.out.println(e1);  
 System.out.println("Element at such Index doesn't Exists");  
 }  
 }  
}

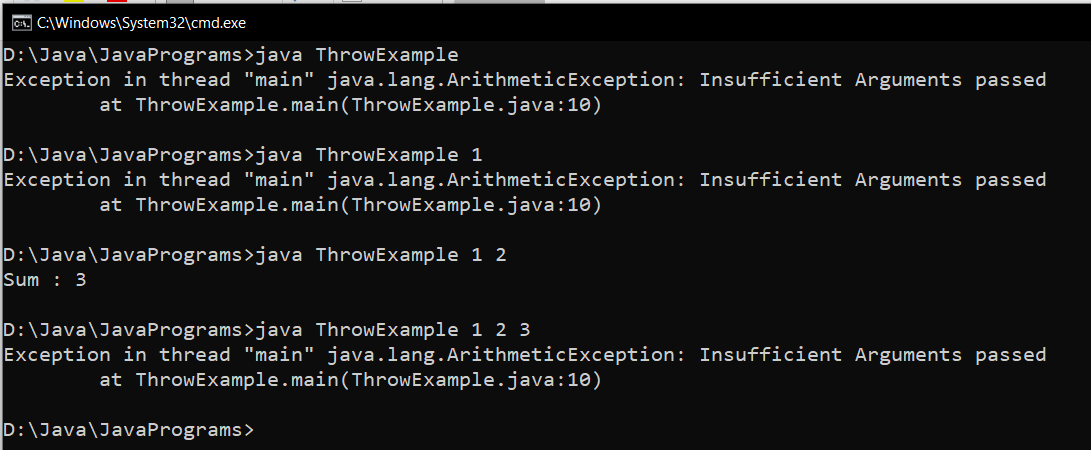
**Output (c):**



**Program (d): Throw an exception when there are no sufficient arguments passed into command line as input for adding two numbers**

public class ThrowExample {  
  
 public static void main(String[] args) {  
 int x=0 ;  
 for (String arg:args) {  
 x++;  
 }  
  
 if (x!=2) {  
 throw new ArithmeticException("Insufficient Arguments passed");  
 }  
 else {  
 int sum = Integer.parseInt(args[0])+Integer.parseInt(args[1]);  
 System.out.println("Sum : "+ sum);  
 }  
 }  
}

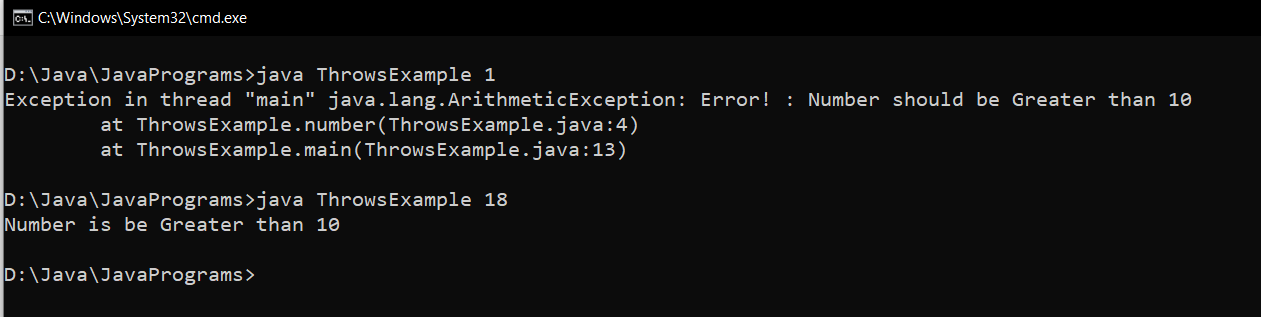
**Output (d):**



**Program (e): Throws example**

public class ThrowsExample {  
 static void number(int x) throws ArithmeticException {  
 if (x < 10) {  
 throw new ArithmeticException("Error! : Number should be Greater than 10");  
 }  
 else {  
 System.out.println("Number is be Greater than 10");  
 }  
 }  
  
 public static void main(String[] args) {  
 int y=Integer.parseInt(args[0]);  
 number(y);  
 }  
}

**Output (e):**



**Result:**

Exception handling is carried out in a Java program.