**Code-01**

1. Exceptions
2. Error vs Exceptions
3. Exception Handling
4. Try, Catch, and Finally block

Code:

Modification 1:

package labtest;

public class LabTest {

public static void main(String[] args) {

int num1=12;

int num2=0;

int[] a=new int[3];

//int a; //Redefiniton Error

try{

a[3]=10; //ArrayIndexOutOfBoundsException

int div=num1/num2; //ArithmeticException

}

catch(ArithmeticException ar){

System.out.println("Number cannot be devided by zero!"); //Exception handled

}

catch(ArrayIndexOutOfBoundsException ai){

System.out.println("Cannot access the array-index; out of bound!");//Exception handled

}

finally{

System.out.println("Exception handled!");

}

}

}

Modification 2:

package labtest;

public class LabTest {

public static void main(String[] args) {

int num1=12;

int num2=0;

int[] a=new int[3];

//int a; //Redefiniton Error

try{

// a[3]=10; //ArrayIndexOutOfBoundsException

int div=num1/num2; //ArithmeticException

}

catch(ArithmeticException ar){

System.out.println("Number cannot be devided by zero!"); //Exception handled

}

catch(ArrayIndexOutOfBoundsException ai){

System.out.println("Cannot access the array-index; out of bound!");//Exception handled

}

finally{

System.out.println("Exception handled!");

}

}

}

Modification 3:

package labtest;

public class LabTest {

public static void main(String[] args) {

int num1=12;

int num2=0;

int[] a=new int[3];

int a; //Redefiniton Error

try{

// a[3]=10; //ArrayIndexOutOfBoundsException

int div=num1/num2; //ArithmeticException

}

catch(ArithmeticException ar){

System.out.println("Number cannot be devided by zero!"); //Exception handled

}

catch(ArrayIndexOutOfBoundsException ai){

System.out.println("Cannot access the array-index; out of bound!");//Exception handled

}

finally{

System.out.println("Exception handled!");

}

}

}

Output:

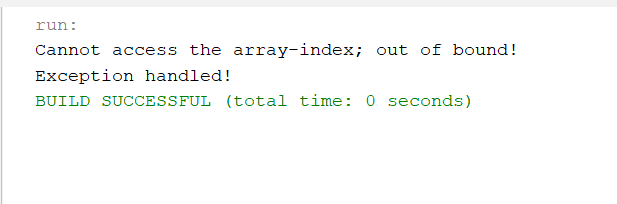


Fig 1.1: Output for modification 1.

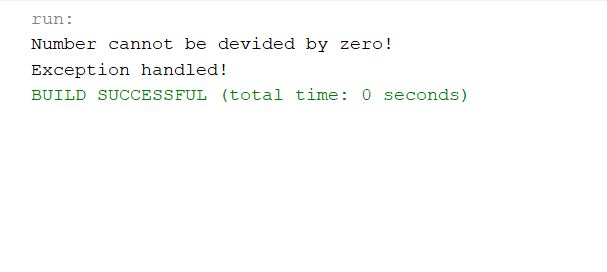
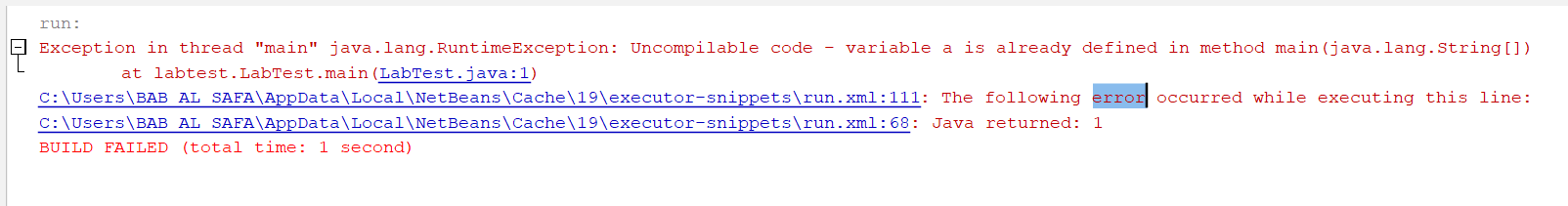


Fig 1.2: Output for modification 2.

  
Fig 1.3: Output for modification 3.

**Code-02**

1. Encapsulation
2. Polymrphism
3. Inheritance
4. Abstraction

Code:

package labtest;

import java.util.Scanner;

class Student{ //Encapsulation

private int totalMarks;

private int totalCourses;

private int avgMarks;

/\*

All the methods are exaples of Abstraction

\*/

public Student(int totalMarks, int totalCourses) {

this.totalMarks = totalMarks;

this.totalCourses = totalCourses;

}

public Student() { //Polymorphism: Constructor overloading

System.out.println("Give total obtained marks and number of courses : ");

Scanner input = new Scanner(System.in);

this.totalMarks = input.nextInt();

this.totalCourses = input.nextInt();

}

public void display(){

System.out.println("Average marks : " + (totalMarks/totalCourses));

}

}

class TimeDist extends Student{ //Inheritance

private int totalTime;

private int totalCourses;

public TimeDist(int totalCourses, int totalTime) {

this.totalTime = totalTime;

this.totalCourses = totalCourses;

System.out.println("Given number of total courses & total time: " + totalTime +" "+totalCourses);

}

public void display(){

System.out.println("Average time for each course : " + (totalTime/totalCourses)); //Polymorphism: Function overriding

}

}

public class LabTest {

public static void main(String[] args) {

Student s1 = new Student(140,10);

s1.display();

Student s2 = new Student();

s2.display();

TimeDist t1 = new TimeDist(230,5);

t1.display();

}

}

Output:

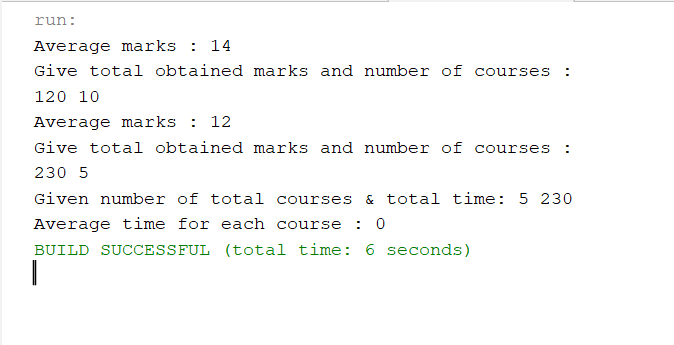


Fig 2.1: Output on console.