DATE:27/09/2024 REGNO:2022503045

JAVA ASSIGNMENT 1

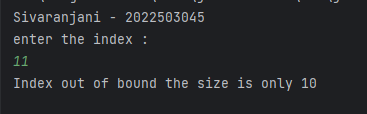
1. Write a program to perform unchecked exception. Use appropriate try-catch blocks to handle these exceptions and provide meaningful error messages

CODE:

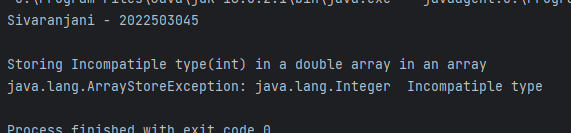
import java.sql.SQLOutput;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
import java.util.Scanner;  
  
public class Exceptions\_3045 {  
 int size\_of\_array=10;  
 int[] array=new int[size\_of\_array];  
 Object[] a=new Double[10];  
 public void display(){  
 System.out.println("From exceptions class..");  
 }  
 public int getElement(int index){  
 try{  
 int element=array[index];  
 return element;  
 }  
 catch(ArrayIndexOutOfBoundsException e){  
 System.out.println("Index out of bound the size is only "+size\_of\_array);  
 }  
 return -1;  
 }  
 public void IncompatipleType(){  
 try {  
 a[4] = 8;  
 }  
 catch(ArrayStoreException e){  
 System.out.println(e+" Incompatiple type");  
 }  
 }  
 public void classCasting(){  
 Exceptions\_3045 exceptionObj=new Exceptions\_3045();  
  
 try{  
 sample sampleobj=(sample)exceptionObj;  
 sampleobj.display();  
  
 }  
 catch(ClassCastException e){  
 System.out.println("Invalid Casting");  
 }  
  
 }  
 public void indexOutofbound(List<Integer> list,int index) {  
 try {  
 System.out.println("the element is " + list.get(index));  
 }  
 catch(IndexOutOfBoundsException e){  
 System.out.println("the list size is only "+list.size());  
 }  
 }  
 public void declareNegativeArray(){  
 try{  
 array=new int[-2];  
 }  
 catch(NegativeArraySizeException e){  
 System.out.println("You declared a array with negative size "+e);  
 }  
 }  
 public void comparingWithNull(String str){  
 String compare=null;  
 try {  
 if (compare.compareTo(str) == 0) {  
 System.out.println("hii");  
 } else {  
 System.out.println("bye");  
 }  
 }  
 catch(NullPointerException e){  
 System.out.println("The compared string is null "+e);  
 }  
 }  
 public void illegalArgument(String name,String collegeName){  
 if(name==null||collegeName==null){  
 throw new IllegalArgumentException("Sorry! Both name and collegeName necessary!");  
 }  
 System.out.println("Name :"+name +"College Name : "+collegeName);  
 }  
 public void convertToInt(String str){  
 try{  
 int word=Integer.parseInt(str);  
 System.out.println(word);  
 }  
 catch(NumberFormatException e){  
 System.out.println("String to convert is invalid "+e);  
 }  
 }  
 public void getcharat(String str,int i){  
 try{  
 System.out.println(str.charAt(i));  
 }  
 catch(StringIndexOutOfBoundsException e){  
 System.out.println("The string length is only "+str.length()+" "+e);  
 }  
 }  
 public void convertoList(String str){  
 try{  
 String[] arr=str.split(" ");  
 List<String> list= Arrays.asList(arr);  
 list.add("added");  
  
 }  
 catch(UnsupportedOperationException e){  
 System.out.println("Unsupported exception caught "+ e);  
 }  
 }  
 public static void main(String[] args){  
 System.out.println("Sivaranjani - 2022503045");  
 Scanner sc=new Scanner(System.in);  
 System.out.println("enter the index : ");  
 int index=sc.nextInt();  
 Exceptions\_3045 obj=new Exceptions\_3045();  
 if(obj.getElement(index)!=-1){  
 System.out.println(obj.getElement(index));  
 }  
 System.out.println();  
 System.out.println("Storing Incompatiple type(int) in a double array in an array ");  
 obj.IncompatipleType();  
 System.out.println();  
 System.out.println("Casting a parent object to a child object class ... ");  
 obj.classCasting();  
 System.out.println();  
 System.out.println("Providing name or collegeName as null in the method for exception like this:");  
 System.out.println("obj.illegalArgument(\"sivaranjani\",null);\n" +  
 " obj.illegalArgument(null,null);");  
 try {  
 obj.illegalArgument("sivaranjani", null);  
 obj.illegalArgument(null, null);  
 }  
 catch(IllegalArgumentException e){  
 System.out.println(e.getMessage());  
 }  
 List<Integer> list=new ArrayList<>(10);  
 System.out.println("enter the index for a list : ");  
 index=sc.nextInt();  
 obj.indexOutofbound(list,index);  
 System.out.println("Declaring an array with negative size of -2");  
 System.out.println();  
 obj.declareNegativeArray();  
 String str;  
 System.out.println("enter a string : ");  
 str=sc.nextLine();  
 obj.comparingWithNull(str);  
 System.out.println();  
 obj.convertToInt(str);  
 System.out.println();  
 obj.getcharat(str,9);  
 System.out.println();  
 obj.convertoList(str);  
  
 }  
}  
class sample extends Exceptions\_3045{  
 public void display(){  
 System.out.println("from sample class..");  
 }  
}

OUTPUT :

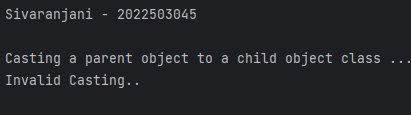
ArrayIndexOutOfBoundsException



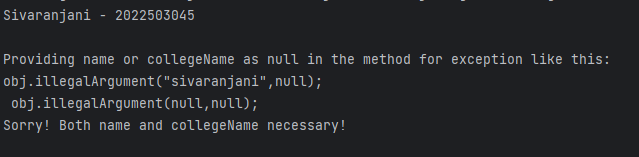
ArrayStoreException



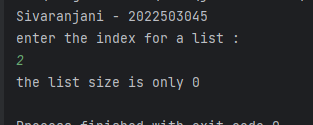
ClassCastingException



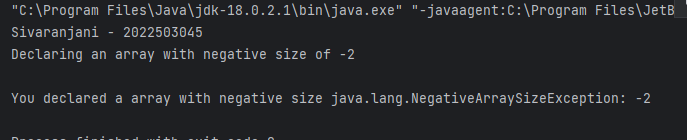
IllegalArgumentException



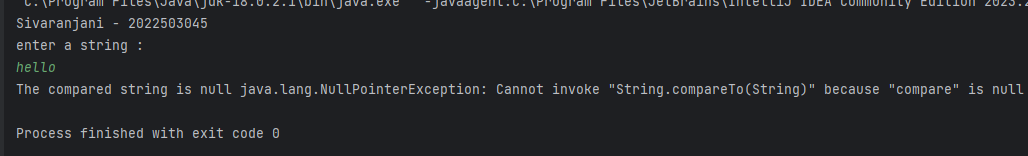
IndexOutOfBoundException



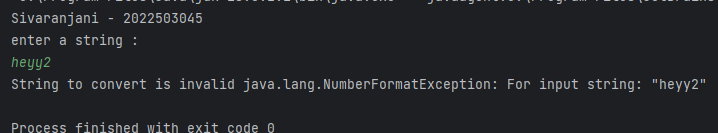
NegativeArraySizeException



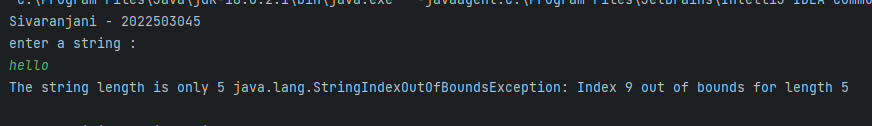
NullPointerException



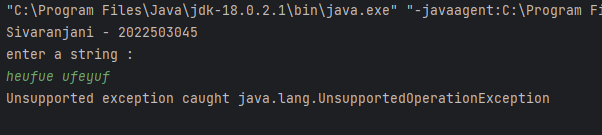
NumberFormateException



StringIndexOutOfBoundsExceptions



UnSupportedoperatorException



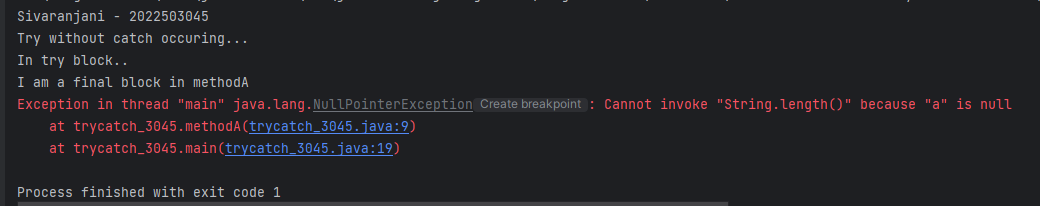
2. Write a program that demonstrates different try-catch-finally block combinations

CODE:

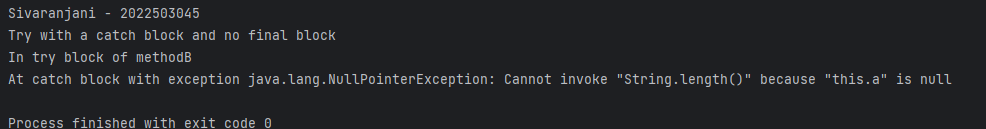
import java.io.IOException;  
import java.util.Scanner;  
import java.io.FileInputStream;  
public class trycatch\_3045 {  
 String a=null;  
 public void methodA(){  
  
 System.out.println("Try without catch occuring...");  
 try {  
 System.out.println("In try block..");  
 System.out.println(a.length());  
 }  
 finally {  
 System.out.println("I am a final block in methodA");  
 }  
 }  
 public void methodB(){  
 System.out.println("Try with a catch block and no final block");  
 try{  
 System.out.println("In try block of methodB");  
 System.out.println(a.length());  
 }  
 catch(Exception e){  
 System.out.println("At catch block with exception "+e);  
 }  
 }  
 public void methodC(){  
 System.out.println("Try with catch and finall block");  
 try{  
 System.out.println("In try block of methodC");  
 System.out.println(a.length());  
 }  
 catch(Exception e){  
 System.out.println("At catch block with exception "+e);  
 }  
 finally{  
 System.out.println("At finally block of methodC");  
 }  
 }  
 public void methodD(){  
 System.out.println("Try with multiple catches");  
 try{  
 System.out.println("In try block of methodD");  
 System.out.println(a.length());  
 System.out.println(1/0);  
 }  
 catch(NullPointerException e){  
 System.out.println("1ST catch block with exception "+e);  
 }  
 catch(Exception e){  
 System.out.println("2nd exception is "+e);  
 }  
  
 }  
 public void methodE(){  
 System.out.println("Try with Nested catches and finally block");  
 try{  
 System.out.println("In try block of methodE");  
 a="hello";  
 System.out.println(a.length());  
 try {  
 System.out.println(1 / 0);  
 }  
 catch(ArithmeticException e){  
 System.out.println("Nested exception divide by zero");  
 }  
 finally {  
 System.out.println("Final block in nested try");  
 }  
 }  
 catch(NullPointerException e){  
 System.out.println("1ST catch block with exception "+e);  
 }  
 finally{  
 System.out.println("final block in methodE outside");  
 }  
 }  
 public void methodF(){  
 System.out.println("Try with resources");  
 try(FileInputStream fp=new FileInputStream("./example.txt")){  
 System.out.println("In try block of methodF");  
 int c;  
 while((c=fp.read())!=-1){  
 System.out.println(c);  
 }  
  
 }  
 catch(IOException e){  
 System.out.println("FILE NOT OPENED "+e);  
 }  
  
 }  
 public static void main(String[] args) {  
 System.out.println("Sivaranjani - 2022503045");  
 Scanner sc = new Scanner(System.in);  
 trycatch\_3045 obj=new trycatch\_3045();  
 obj.methodA();  
 obj.methodB();  
 obj.methodC();  
 obj.methodD();  
 obj.methodE();

obj.methodF();  
 }  
}

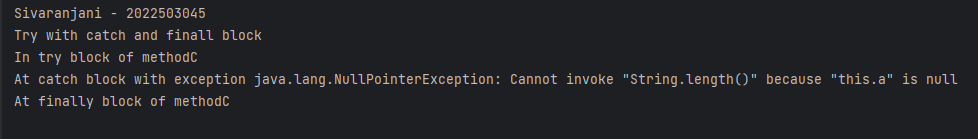
a. Try without catch block



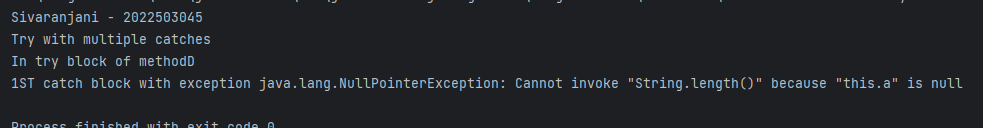
b. Try without finally block



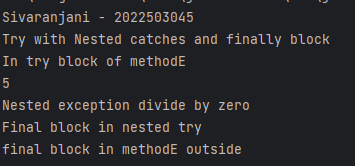
c. Try with catch and finally block



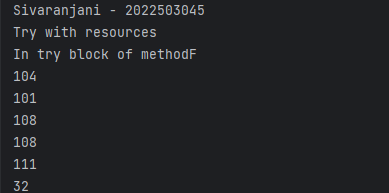
d. Try with multiple catch block



e. Nested try catch finally block



f. Try with resources



3. Create a custom exception class called InvalidMarkException that extends Exception. Then, write a Student class with a method to set marks that throws this custom exception if the mark is out of range (e.g., less than 0 or greater than 100).

CODE:

import java.util.Scanner;

class InvalidMarkException extends Exception{

int mark;

public InvalidMarkException(int mark){

this.mark=mark;

}

public String toString(){

return "Invalid mark of Student "+mark;

}

}

public class Student\_3045 {

String name;

int mark;

Student\_3045(String name){

this.name=name;

}

public void setMark(int mark) throws InvalidMarkException{

if(mark<0||mark>100){

throw new InvalidMarkException(mark);

}

this.mark=mark;

}

public static void main(String[] args){

System.out.println("Sivaranjani - 2022503045");

Scanner sc=new Scanner(System.in);

System.out.println("Enter the no of students:");

int n=sc.nextInt();

Student\_3045[] students=new Student\_3045[n];

System.out.println("enter the names of students :");

for(int i=0;i<n;i++){

String s=sc.next();

Student\_3045 student=new Student\_3045(s);

students[i]=student;

}

System.out.println("enter the marks of each student :");

for(int i=0;i<n;i++){

Student\_3045 student=students[i];

System.out.print(student.name+": ");

int mark=sc.nextInt();

try{

student.setMark(mark);

}

catch(Exception e){

System.out.println(e);

System.out.println("enter the correct mark again ");

i=i-1;

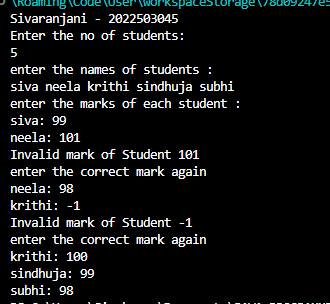
}

}

}

}

OUTPUT:



4. Write a program to illustrate the propagation of checked and unchecked exception.

CODE:

import java.io.\*;

public class ExceptionPropagation\_3045 {

public static void main(String[] args) {

ExceptionPropagation\_3045 ep = new ExceptionPropagation\_3045();

System.out.println("Sivaranjani -2022503045");

try {

ep.uncheckedExceptionExample();

} catch (ArithmeticException e) {

System.out.println("Caught unchecked exception: " + e.getMessage());

}

try {

ep.checkedExceptionExample();

} catch (IOException e) {

System.out.println("Caught checked exception: " + e.getMessage());

}

}

void uncheckedExceptionExample() {

level1();

}

void level1() {

level2();

}

void level2() {

// This will throw an ArithmeticException (division by zero)

int result = 10 / 0;

}

void checkedExceptionExample() throws IOException {

levelA();

}

void levelA() throws IOException {

levelB();

}

void levelB() throws IOException {

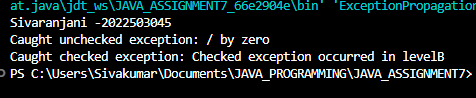
// Throwing a checked exception

throw new IOException("Checked exception occurred in levelB");

}

}

OUTPUT:



5. Write a program to illustrate the method overloading in exception handling mechanism for checked and unchecked exception.

import java.io.IOException;

public class overLoading\_3045 {

public static void main(String[] args) {

CheckedException obj=new CheckedException();

UncheckedException obj2=new UncheckedException();

System.out.println("Sivaranjani -2022503045");

System.out.println("Checked Exception with method overloading : ");

try{

obj.method1(5);

}

catch(IOException e){

System.out.println(e.getMessage());

}

finally{

obj.method1("parameter");

}

System.out.println();

System.out.println("Unchecked exceptions with method overloading : ");

try{

obj2.method1(null);

}

catch(NullPointerException e){

System.out.println("Null string as a parameter in unchecked exception");

}

try{

obj2.method1(4);

}

catch(ArithmeticException e){

System.out.println("Unchecked exception of same method with divide bye zero "+e);

}

}

}

class CheckedException{

public void method1(int a) throws IOException{

System.out.println(a);

throw new IOException("exception from method 1 of checked Exception with int as parameter");

}

public void method1(String str){

System.out.println("method1 with str as parameter");

}

}

class UncheckedException{

public void method1(int a) throws ArithmeticException{

int d=a/0;

}

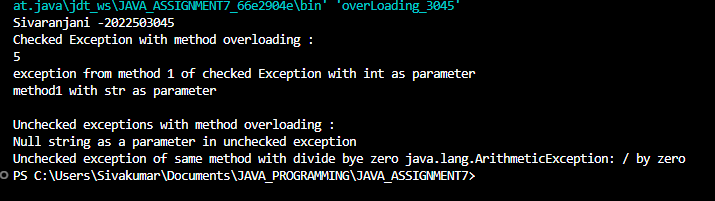
public void method1(String str) throws NullPointerException{

int length=str.length();

}

}

OUTPUT:



6. Implement a base class and a derived class to demonstrate exception handling in method overriding:

CODE:

import java.io.FileNotFoundException;

import java.io.IOException;

public class overRiding\_3045 {

public static void main(String[] args){

System.out.println("Sivaranjani -2022503045");

checkedOverloading co=new checkedOverloading();

try{

co.checkedException1();

co.checkedException2();

}

catch(FileNotFoundException e){

System.out.println(e.getMessage());

}

UncheckedOverriding uo=new UncheckedOverriding();

try{

uo.uncheckedException1();

uo.uncheckedException2();

}

catch(ArithmeticException e){

System.out.println("Unchecked Exception overrided by a different exception in derived class"+e);

}

}

}

class baseClass{

public void checkedException1() throws IOException{

throw new IOException("checked exception1 from base class");

}

public void checkedException2() throws IOException{

throw new IOException("checked exception1 from base class");

}

public void uncheckedException1() throws NullPointerException{

String str=null;

int length=str.length();

}

public void uncheckedException2() throws NullPointerException{

String str=null;

int length=str.length();

}

}

class checkedOverloading extends baseClass{

public void checkedException1(){

System.out.println("checkedException1 overRided by a derived class without throwing any exception");

}

public void checkedException2() throws FileNotFoundException{

throw new FileNotFoundException("throws filenotfoundexception overrided the IOexception of checkedException2");

}

}

class UncheckedOverriding extends baseClass{

public void uncheckedException1(){

System.out.println("UncheckException 1 with no exception");

}

public void uncheckedException2() throws ArithmeticException{

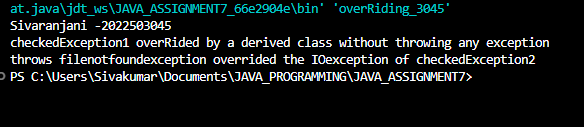
int a=7/0;

}

}

OUTPUT:

a) Overriding a method that throws an unchecked exception



b) Overriding a method that throws a checked exception

