Exp 7. Exception Handling in java

1. **Write a program that reads an integer from the user and divides it by another integer. If a DivideByZeroException occurs, handle it and print an appropriate message.**

package ExecptionHandling;

import java.util.Scanner;

public class DivideByZero {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

System.out.println("Enter the dividend and divisor");

int a = in.nextInt();

int b = in.nextInt();

try {

int val = a/b;

System.out.println("The quotient is "+ val);

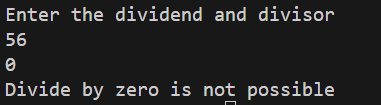
} catch (ArithmeticException ae) {

System.out.println("Divide by zero is not possible");

}

}

}



1. **Create a program that accepts two integers from the user and performs division. If the second integer is zero, catch the ArithmeticException and display a message "Cannot divide by zero."**

package ExecptionHandling;

import java.util.Scanner;

public class DivideByZero {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

System.out.println("Enter the dividend and divisor");

int a = in.nextInt();

int b = in.nextInt();

try {

int val = a/b;

System.out.println("The quotient is "+ val);

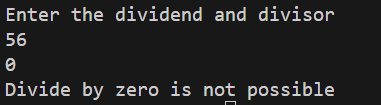
} catch (ArithmeticException ae) {

System.out.println("Divide by zero is not possible");

}

}

}



1. **Write a program that handles an invalid array index scenario. Declare an array of size 5 and attempt to access an index that is out of bounds. Catch the ArrayIndexOutOfBoundsException and display a custom error message.**

package ExecptionHandling;

public class ArrayOutofBound {

public static void main(String[] args) {

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

try {

for (int i = 0; i < 6; i++) {

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

}

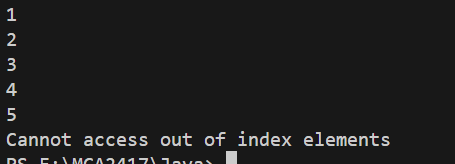
} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Cannot access out of index elements");

}

}

}



1. **Create a program where the user is asked to enter a string. If the user enters a non-numeric value when the program expects an integer, catch the NumberFormatException and print a message indicating that the input is invalid.**

package ExecptionHandling;

import java.util.Scanner;

public class NonNumericException {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

try {

int numeric = Integer.parseInt(in.next());

System.out.println("Entered value is: "+ numeric);

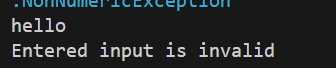
} catch (NumberFormatException e) {

System.out.println("Entered input is invalid");

}

}

}



1. **Write a program that reads data from a file. If the file doesn't exist, catch the FileNotFoundException and print an error message stating that the file is missing.**

package ExceptionHandling;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

public class FileNotFound {

public static void main(String[] args) throws IOException{

FileInputStream f;

try {

f = new FileInputStream("ExceptionHandling/sample.txt");

int c;

do{

c = f.read();

if(c!=-1){

System.out.print((char)c);

}

}while(c!=-1);

} catch (FileNotFoundException e) {

System.out.println("File not found");

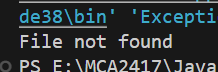
return;

}

f.close();

}

}



1. **Create a method that calculates the square root of a number. If the number is negative, throw an IllegalArgumentException with a message stating that the input cannot be negative. Handle this exception properly in the main method.**

package ExceptionHandling;

import java.util.Scanner;

public class NegativeSquareRoot {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int val = in.nextInt();

try {

if(val<1){

throw new IllegalArgumentException();

}

double sqrt = Math.sqrt(val);

System.out.println("Square root of "+val+" is " +sqrt);

} catch (IllegalArgumentException e) {

System.out.println("Input value cannot be negative");

}

}

}



1. **Write a program that catches multiple exceptions (e.g., ArithmeticException, NullPointerException, and ArrayIndexOutOfBoundsException) using multiple catch blocks within a single try block.**

package ExceptionHandling;

import java.util.Scanner;

public class MultipleException {

public static void main(String[] args) {

try {

Scanner in = new Scanner(System.in);

String str = in.nextLine();

if(str.isEmpty()){

str = null;

}

System.out.println("Enterd string length"+str.length());

int val = Integer.parseInt(str);

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

System.out.println(arr[val]/val);

}catch (ArithmeticException e) {

System.out.println("Divide by zero is not possible");

}catch (NullPointerException e){

System.out.println("Null pointer exception");

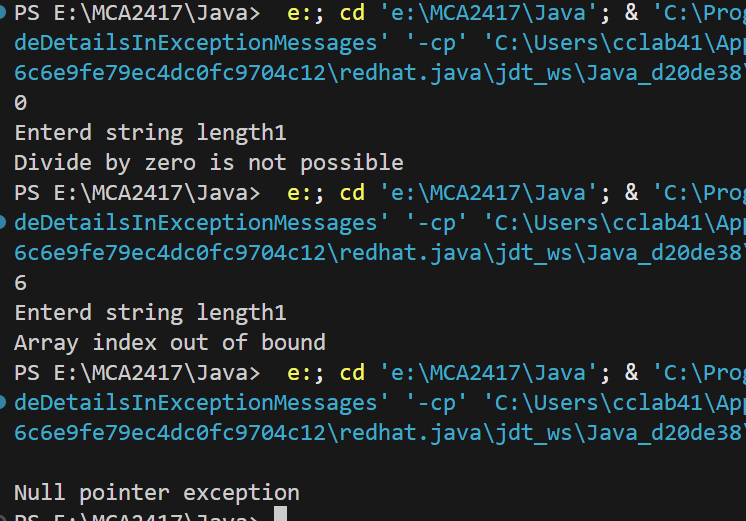
}catch (ArrayIndexOutOfBoundsException e){

System.out.println("Array index out of bound");

}

}

}



1. **Write a program to handle exceptions using a try-catch-finally block. In the finally block, print a message saying "Execution completed," whether or not an exception occurred in the try block.**
2. **Create a program where a method throws an exception if a number is less than zero. In the main method, call this method inside a try-catch block and handle the thrown exception appropriately.**
3. **Write a program that reads an integer from the user and throws a custom exception if the number is negative. The custom exception should extend the Exception class, and the main method should catch and handle it.**