CHAROTAR UNIVERSITY OF SCIENCE TECHNOLOGY
DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH
Department of Computer Science & Engineering

Subject Name: Java Programming
Semester: 3rd
Subject Code: CSE201
Academic year: 2024

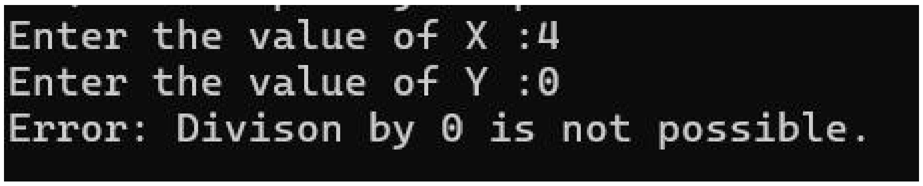
Part - 5

No.	Aim of the Practical
24	Write a java program which takes two integers x & y as input, you have to compute x/y. If x and y are not integers or if y is zero, exception will occur and you have to report it.

PROGRAM CODE:

```
import java.util.Scanner;

public class practical_24 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        try {
            System.out.print("Enter the value of X :");
            int x = sc.nextInt();
            System.out.print("Enter the value of Y :");
            int y = sc.nextInt();

            int result = x / y;
            System.out.println("Result : " + result);
        } catch (Exception e) {
            if (e instanceof ArithmeticException) {
                System.out.println("Error: Divison by 0 is not possible.");
            } else {
                System.out.println("Please Enter valid integer.");
            }
        }
    }
}
```

OUTPUT:



```
Enter the value of X :4
Enter the value of Y :0
Error: Divison by 0 is not possible.
```

CONCLUSION:

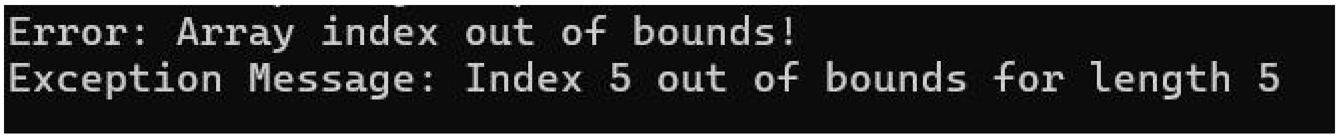
In conclusion, this Java program effectively handles the division of two integers while managing potential exceptions such as invalid input types and division by zero. By utilizing `InputMismatchException` and `ArithmeticException`, the program ensures that errors are caught and reported, preventing crashes and providing informative feedback to the user. This approach enhances the program's robustness and ensures smooth execution even in the face of unexpected input.

25 Write a Java program that throws an exception and catch it using a try-catch block.

PROGRAM CODE:

```
public class practical_25  {
   public static void main(String[] args) {
      try {
         int[] numbers = { 1, 2, 3, 4, 5 };

         System.out.println(numbers[5]);

      } catch (ArrayIndexOutOfBoundsException e) {
         System.out.println("Error: Array index out of bounds!");
         System.out.println("Exception Message: " + e.getMessage());
      }
   }
}
```

OUTPUT:



```
Error: Array index out of bounds!
Exception Message: Index 5 out of bounds for length 5
```

CONCLUSION:

This program demonstrates how to manually throw and handle exceptions in Java. By using a try-catch block, the program ensures that thrown exceptions are caught, preventing abnormal termination. This structured error handling mechanism allows developers to manage unexpected scenarios effectively, ensuring that the program continues to operate smoothly and provides meaningful error messages to users.

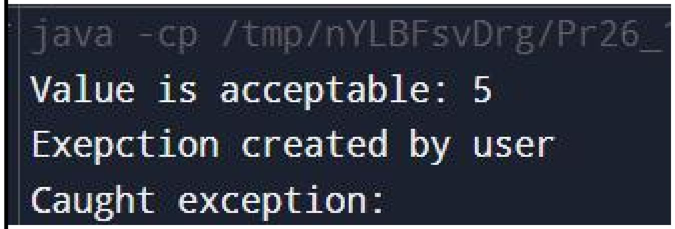
26

Write a java program to generate user defined exception using “throw” and “throws” keyword. Also Write a java that differentiates checked and unchecked exceptions. (Mention at least two checked and two unchecked exceptions in program).

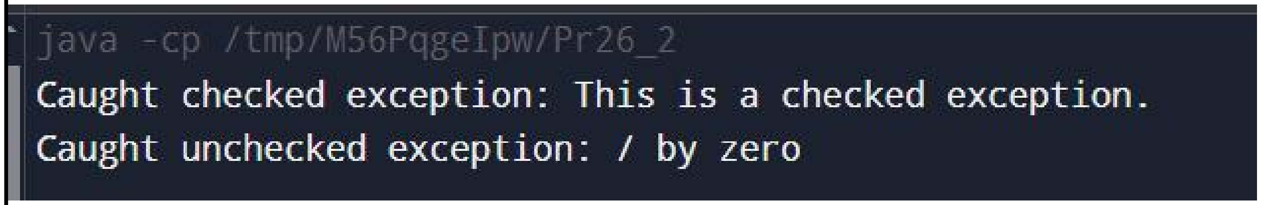
PROGRAM CODE:

```
class MyException extends Exception {
    public MyException() {
        System.out.println("Exepction created by user");
    }
}

public class Pr26_1 {

    static void checkValue(int value) throws MyException {
        if (value > 10) {
            throw new MyException();
        }
        System.out.println("Value is acceptable: " + value);
    }
    public static void main(String[] args) {
        try {
            checkValue(5);
            checkValue(15);

        } catch (MyException e) {
            System.out.println("Caught exception: ");
        }


    }
}
```

OUTPUT:



```
java -cp /tmp/nYLBFsvDrg/Pr26_
Value is acceptable: 5
Exepction created by user
Caught exception:
```

	 ```java -cp /tmp/M56PqgeIpw/Pr26_2``` ```Caught checked exception: This is a checked exception.``` ```Caught unchecked exception: / by zero``` This program highlights the difference between checked and unchecked exceptions in Java. Checked exceptions must be declared or handled at compile-time, ensuring potential errors are addressed, while unchecked exceptions occur at runtime and don't need to be explicitly handled, often arising from logical or programming errors.