CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

**DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH**

Department of Computer Science & Engineering

Subject Name: Java Programming

Semester: III

Subject Code: CSE201

Academic year: 2024-25

Part - V

|  |  |
| --- | --- |
| **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 DivisionExample {      public static void main(String[] args) {          Scanner scanner = new Scanner(System.in);          try {              System.out.print("Enter the first integer (x): ");              int x = Integer.parseInt(scanner.nextLine());              System.out.print("Enter the second integer (y): ");              int y = Integer.parseInt(scanner.nextLine());                int result = x / y;              System.out.println("Result of " + x + " / " + y + " = " + result);          }            catch (NumberFormatException e) {              System.out.println("Error: Please enter valid integers.");          }            catch (ArithmeticException e) {              System.out.println("Division by zero is not allowed.");          }            catch (Exception e) {              System.out.println("An unexpected error occurred: " + e.getMessage());          }          // finally{          //     System.out.println("The End Of Code");          // }          scanner.close();      }  }  **OUTPUT:**    **CONCLUSION:**  I learned from this program how to handle exceptions when performing division, ensuring the program gracefully handles errors such as dividing by zero or invalid input types. It taught me the importance of exception handling for making programs robust. |
| 25. | Write a Java program that throws an exception and catch it using a try-catch block.  **PROGRAM CODE:**  public class JAVA\_25 {      public static void main(String[] args) {          int a[]=new int[5];          try{              a[5]=10;          }          catch(ArrayIndexOutOfBoundsException e){              System.out.println(e);          }      }  }  **OUTPUT:**    **CONCLUSION:**  I learned from this program how to handle exceptions like ArrayIndexOutOfBoundsException using a try-catch block, ensuring the program doesn't crash when attempting to access an invalid index. |
| 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:**  1) class AgeException extends Exception {  public AgeException(String message) {  super(message);  }  }  public class CustomExceptionExample {  public static void checkAge(int age) throws AgeException {  if (age < 18) {  throw new AgeException("Age is less than 18. Access Denied.");  } else {  System.out.println("Access Granted.");  }  }  public static void main(String[] args) {  try {  checkAge(16);  } catch (AgeException e) {  System.out.println("Exception caught: " + e.getMessage());  }  }  }    2) import java.io.\*;  public class CheckedUncheckedExample {  public static void readFile(String filePath) throws IOException {  FileReader file = new FileReader(filePath);  BufferedReader br = new BufferedReader(file);  System.out.println(br.readLine());  br.close();  }  public static void main(String[] args) {  try {  int a = 10 / 0;  } catch (ArithmeticException e) {  System.out.println("Unchecked Exception caught: " + e);  }  try {  String str = null;  System.out.println(str.length());  } catch (NullPointerException e) {  System.out.println("Unchecked Exception caught: " + e);  }  try {  readFile("nonexistentfile.txt");  } catch (IOException e) {  System.out.println("Checked Exception caught: " + e);  }  try {  Class.forName("UnknownClass");  } catch (ClassNotFoundException e) {  System.out.println("Checked Exception caught: " + e);  }  }  }  **OUTPUT:**  1)    2)  **CONCLUSION:**  I learned from these programs how to create and handle user-defined exceptions using the throw and throws keywords, and how to differentiate between checked (e.g., IOException, ClassNotFoundException) and unchecked exceptions (e.g., ArithmeticException, NullPointerException), enhancing my understanding of exception handling in Java. |