JAVA PROGRAMMING [CSE201] Enrolment No.: 23DCS052

# CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH

Department of Computer Science & Engineering

**Subject Name:** JAVA PROGRAMMING

# Semester: 3

**Subject Code:** CSE201

**Academic year:** 2024-25

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.\*;  public class PT\_24 {  public static void main(String[] args) { Scanner s = new Scanner(System.in); System.out.print("Enter X an Y : "); int x = s.nextInt();  int y = s.nextInt(); System.out.println("X / Y = " + x/y); s.close();  }  } |

|  |  |
| --- | --- |
|  | **OUTPUT:**    **CONCLUSION:**  **Structured exception handling (using try-catch) is generally the better approach in Java.** It is more maintainable, scalable, and aligns with Java's best practices for error handling. Exceptions also ensure that you handle unexpected situations effectively without cluttering the main logic. |
| 25 | Write a Java program that throws an exception and catch it using a try-catch block.  **PROGRAM CODE:**  import java.util.\*;  public class PT\_25 {  public static void main(String[] args) { try{  Scanner s = new Scanner(System.in); System.out.print("Enter X an Y : "); int x = s.nextInt();  int y = s.nextInt(); try{  System.out.println("X / Y = " + x/y);  }catch(Exception e){ System.out.println(e.toString() + "  Exception Handeled.");  System.out.println("Error: Division by zero is not allowed.");  } |

|  |  |
| --- | --- |
|  | s.close();  }catch(InputMismatchException i){ System.out.println(i.toString() + "  Exception Handeled.");  System.out.println("Error: Invalid input.  Please enter integer values.");  }  }  }  **OUTPUT:**    **CONCLUSION:**  This Java program illustrates how to explicitly throw an exception using the throw keyword  and catch it using a try-catch block. This ensures errors are handled gracefully, improving program reliability. |
| 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:**  import java.util.\*;  public class PT\_26 {  public static void main(String[] args) { Scanner sc= new Scanner(System.in); |

|  |  |
| --- | --- |
|  | try {  System.out.println("Enter two numbers : "); int a = sc.nextInt();  int b = sc.nextInt(); if (b == 0) {  throw new ArithmeticException("Division by zero is not allowed.");  }  int c = a/b; System.out.println("Result : " + c);  } catch (ArithmeticException e) { System.out.println("Arithmetic Exception: " + e);  }  sc.close();  }  }**OUTPUT :**    **CONCLUSION :**  This Java implementation demonstrates inheritance, code reuse, and polymorphism. Using arrays of objects allows handling multiple shapes efficiently, as both Rectangle and Square share common functionality through inheritance. |

|  |  |
| --- | --- |
| 20 | Create a class named 'Shape' with a method to print "This is This is shape". Then create two other classes named 'Rectangle', 'Circle' inheriting the Shape class, both having a method to print "This is rectangular shape" and "This is circular shape" respectively. Create a subclass 'Square' of 'Rectangle' having a method to print "Square is a rectangle". Now call the method of 'Shape' and 'Rectangle' class by the object of 'Square' class.  **PROGRAM CODE :**  class shape{ void print1(){  System.out.println("This is Shape");  }  }  class rectangle extends shape{  static class square extends rectangle{ void print4(){  System.out.println("Square is a Rectangle");  }  }  void print2(){  System.out.println("This is Rectangle Shape");  }  }  class circle extends shape{ void print3(){  System.out.println("This is Circuler Shape");  }  } |

|  |  |
| --- | --- |
|  | public class PT\_20 {  public static void main(String[] args) { rectangle.square s = new rectangle.square(); s.print4();  s.print1();  s.print2();  }  }  }  **OUTPUT :**    **CONCLUSION :**  In Java, **checked exceptions** (e.g., FileNotFoundException, InterruptedException) must be caught or declared with throws, while **unchecked exceptions** (e.g., ArithmeticException, ArrayIndexOutOfBoundsException) are runtime exceptions and need not be explicitly caught. This demonstrates the difference between both. |