**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: 2024-2025**

**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.lang.\*;  import java.util.Scanner;  class prc24  {  public static void main(String args[])  {  System.out.println("Enter two numbers: ");  Scanner s=new Scanner(System.in);  int a=s.nextInt();  int b=s.nextInt();  if(b==0)  {  try  {  int c=a/b;  System.out.println(c);  }  catch(ArithmeticException ae)  {  System.out.println("Exception handled. "+ae.toString());  }  }  else  {  int c=a/b;  System.out.println(c);  }  }  }  **OUTPUT:**    **CONCLUSION:**  By this code, we can conclude that arithmetic exception is implicitly thrown by the statement and is handled by the try-catch block so that the program doesn’t terminate abruptly. |
| **25.** | Write a Java program that throws an exception and catch it using a try-catch block.  **PROGRAM CODE :**  public class prc25  {  public static void main(String[] args)  {  try  {  throwMethod();  }  catch(Exception e)  {  System.out.println(e);  }  }  static void throwMethod()  {  throw new ArithmeticException("User Thrown Exception");  }  }  **OUTPUT:**    **CONCLUSION:**  By this experiment, we learn to catch an exception using the try-catch block. |
| **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.io.\*;  class CheckedException extends Exception{  CheckedException(String Message){  super(Message);  }  CheckedException(){  System.out.println("Default Constructor of Checked Exception.");  }  }  class UncheckedException extends RuntimeException{  UncheckedException(String Message){  super(Message);  }  UncheckedException(){  System.out.println("Default Constructor of Unchecked Exception.");  }  }  public class prc26{  public static void main(String[] args)  {  try  {  int x,y,z;  x=y=0;  z=x/y;  }  catch(Exception e)  {  System.out.println(e);  }  try  {  throwCNFException();  }  catch (ClassNotFoundException e)  {  System.out.println(e);  }  try  {  int[] a={1,2,3};  System.out.println(a[10]);  }  catch(Exception e)  {  System.out.println(e);  }  try  {  throwFNFException();  }  catch (FileNotFoundException e)  {  System.out.println(e+"\n");  }  try  {  throwCheckedException("User-Defined Checked Exception - 1");  }  catch(Exception e)  {  System.out.println(e);  }  try  {  throwCheckedException("User-Defined Checked Exception - 2");  }  catch(Exception e)  {  System.out.println(e);  }  try  {  throw new CheckedException();  }  catch(Exception e)  {  e.getMessage();  }  try  {  throwUncheckedException("User-Defined Unchecked Exception - 1");  }  catch(Exception e)  {  System.out.println(e);  }  try  {  throwUncheckedException("User-Defined Unchecked Exception - 2");  }  catch(Exception e)  {  System.out.println(e);  }  }  static void throwCNFException() throws ClassNotFoundException{  throw new ClassNotFoundException("Pre-Defined Checked Exception : Class Not Found Exception");  }  static void throwFNFException() throws FileNotFoundException{  throw new FileNotFoundException("Pre-Defined Checked Exception : File Not Found Exception");  }  static void throwCheckedException(String message) throws CheckedException{  throw new CheckedException(message);  }  static void throwUncheckedException(String message) {  throw new UncheckedException(message);  }  }    **OUTPUT:**    **CONCLUSION:**  Hereby, we learnt about checked and unchecked exceptions.  **Supplementary experiment:**  **PROGRAM CODE:**  import java.util.Scanner;  class DuplicateNumberException extends Exception {  DuplicateNumberException(String message) {  super(message);  }  }  public class supp5 {  public static void checkForDuplicates(int[] numbers) throws DuplicateNumberException {  for (int i = 0; i < numbers.length; i++) {  for (int j = i + 1; j < numbers.length; j++) {  if (numbers[i] == numbers[j]) {  throw new DuplicateNumberException("Duplicate number found: " + numbers[i]);  }  }  }  }  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Enter the number of integers: ");  int count = scanner.nextInt();  int[] numbers = new int[count];  System.out.println("Enter the integers:");  for (int i = 0; i < count; i++) {  numbers[i] = scanner.nextInt();  }  try {  checkForDuplicates(numbers);  System.out.println("All numbers are unique.");  } catch (DuplicateNumberException e) {  System.out.println(e.getMessage());  }  }  }  **OUTPUT:** |
|  | **Extra codes:**   1. **PROGRAM CODE:**   import java.util.Scanner;  class InsufficientBalanceException extends Exception  {  InsufficientBalanceException(String message)  {  super(message);  }  }  public class extra1  {  public static void main(String[] args)  {  Scanner s = new Scanner(System.in);    double balance;  System.out.print("Enter your initial account balance: ");  balance = s.nextDouble();  System.out.print("Enter the amount to withdraw: ");  double amount = s.nextDouble();  try  {  if (balance < amount) {  throw new InsufficientBalanceException("Insufficient balance in your account");  }  balance -= amount;  System.out.println("Withdrawal successful. New balance: " + balance);    }  catch (InsufficientBalanceException e)  {  System.out.println("Error: " + e.getMessage());  }  }  }  **OUTPUT:**    **CONCLUSION:**  Here, we learnt about user defined exception and how we can declare and handle them.   1. **PROGRAM CODES:**   import java.util.\*;  class InvalidAgeException extends Exception  {  InvalidAgeException(String s)  {  super(s);  }  }  class user\_define\_extra3  {  public static void main(String []args)  {  Scanner sc=new Scanner(System.in);  System.out.println("Enter your age:");  int age=sc.nextInt();  if(age<18)  {  try  {  throw new InvalidAgeException("Not eligible for voting in 2024");  }  catch(InvalidAgeException e)  {  System.out.println(e.getMessage());  System.out.println("Exception caught successfully");  }  }  else  {  System.out.println("eligible for voting in 2024");  }  }  }  **OUTPUT:**    **CONCLUSION:** This is also a similar program handling user-defined exceptions. |