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 - 1

|  |  |
| --- | --- |
| **No.** | **Aim of the Practical** |
| 1. | Demonstration of installation steps of Java,Introduction Object Oriented Concepts, comparison of Java with other object-oriented programming languages. Introduction to  JDK, JRE, JVM, Javadoc, command line argument. Introduction to Eclipse or NetBeans IDE,or BlueJ and Console Programming.  1. Installation of Java  Steps to install Java Development Kit (JDK):   Download JDK:  - Go to the Oracle JDK download page: [Oracle JDK Downloads]  (https://www.oracle.com/java/technologies/javase-downloads.html).  - Select the appropriate JDK version for your operating system (Windows, macOS,Linux).  - Download the installer package (.exe for Windows, .dmg for macOS, .tar.gz for Linux).   Install JDK:  - Windows: Double-click the downloaded .exe file and follow the installation instructions.  - macOS: Double-click the downloaded .dmg file, then drag and drop the JDK package icon to the Applications folder.  - Linux: Extract the downloaded .tar.gz file to a directory and follow the instructions in the README file for installation.   Set JAVA\_HOME (Optional):  - Windows: Set the JAVA\_HOME environment variable to the JDK installation directory.  - macOS/Linux: Add the JDK bin directory to your PATH and set JAVA\_HOME in your shell profile (e.g., ~/.bash\_profile, ~/.bashrc).   Verify Installation:  - Open a terminal or command prompt.  - Type java -version and javac -version to verify that Java runtime and compiler are installed correctly.  2. Introduction to Object-Oriented Concepts  Object-oriented programming (OOP) revolves around the concept of objects, which are instances of classes. Key principles include:  - Classes and Objects: Classes define the blueprint for objects.  - Encapsulation: Bundling data (attributes) and methods (functions) that operate on the data within a single unit (class).  - Inheritance: Mechanism where a new class (derived or child class) is created from an existing class (base or parent class).  - Polymorphism: Ability of different objects to be treated as instances of the same class through method overriding and overloading.  3. Comparison of Java with Other Object-Oriented Programming Languages  Java is often compared with languages like C++, C#, and Python in terms of syntax, features, and application domains. Key points of comparison include:  - Syntax: Java has a C-style syntax with similarities to C++.  - Memory Management: Java uses automatic garbage collection, unlike C++ which requires manual memory management.  - Platform Independence: Java programs are compiled into bytecode, which can run on any JVM, making it platform-independent.  - Libraries: Java has a rich standard library (Java API) comparable to those in C++ and C#.  - Community and Ecosystem: Java has a large developer community and extensive third-party libraries and frameworks.  4. Introduction to JDK, JRE, JVM, Javadoc, Command Line Arguments  - JDK (Java Development Kit): Includes tools for developing and running Java programs, including JRE and development tools such as javac (Java compiler).  - JRE (Java Runtime Environment): Includes JVM (Java Virtual Machine) and libraries required to run Java applications, but does not include development tools.  - JVM (Java Virtual Machine): Executes Java bytecode and provides a runtime environment for Java programs.  - Javadoc: Tool for generating API documentation from Java source code comments.  - Command Line Arguments: Parameters passed to a Java program when it is invoked from the command line.  5. Introduction to Eclipse or NetBeans IDE (Integrated Development Environment)  - Eclipse : A widely used open-source IDE for Java development, also supports other programming languages through plugins. Features include code editing, debugging, and version control integration.  - NetBeans: Another popular open-source IDE primarily for Java development, with features similar to Eclipse.  6. Introduction to BlueJ and Console Programming  - BlueJ : A lightweight IDE specifically designed for teaching and learning Java programming, providing a simplified interface and visualization tools for object-oriented concepts.  - Console Programming : Refers to writing Java programs that interact with users via text-based input and output through the console (command line interface). |
| 2. | Imagine you are developing a simple banking application where you need to display the current balance of a user account. For simplicity, let's say the current balance is $20. Write a java program to store this balance in a variable and then display it to the user.  **PROGRAM CODE:**  public class JAVA\_P2 {  public static void main(String[] args) {  int balance = 20;  System.out.println("Your current balance is : $"+balance);  }  }  **OUTPUT:**    **CONCLUSION:**  This Java program sets a balance to $20 and displays it on the screen. When you run the program, it simply prints "Your current balance is: $20". |
| 3. | Write a program to take the user for a distance (in meters) and the time taken (as three numbers: hours, minutes, seconds), and display the speed, in meters per second, kilometers per hour and miles per hour (hint:1 mile = 1609 meters).  **PROGRAM CODE:**  import java.util.\*;  public class JAVA\_P3 {  public static void main(String[] args) {  float distance;  System.out.print("Enter the distance in metres : ");  Scanner sc = new Scanner(System.in);  distance = sc.nextFloat();  float second,minutes,hours,t\_hour,t\_minutes,t\_seconds;  System.out.print("Enter the time taken to cover "+distance+" metres in hour : ");  hours=sc.nextFloat();  System.out.print("Enter the time taken to cover "+distance+" metres in minutes : ");  minutes=sc.nextFloat();  System.out.print("Enter the time taken to cover "+distance+" metres in seconds : ");  second=sc.nextFloat();  t\_hour=hours+(minutes/60)+(second/3600);  t\_minutes=(hours\*60)+minutes+(second/60);  t\_seconds=(t\_minutes\*60);  float km=distance/1000;  float speed\_metre=distance/t\_seconds;  System.out.println("Speed is "+speed\_metre+" metre/sec");  float speed\_km\_=km/t\_hour;  System.out.println("Speed is "+speed\_km\_+" km/hours");  float mile = distance/1609;  float speed\_mile = mile/t\_hour;  System.out.println("Speed is "+speed\_mile+" mile/hour");  sc.close();  }  }  **OUTPUT:**    **CONCLUSION:**  This java program calculates and outputs the speed of an object based on the distance traveled and the time taken in hours, minutes, and seconds. It converts the distance into meters, kilometers, and miles, then calculates the speed in meters per second, kilometers per hour, and miles per hour accordingly. It uses basic input handling with Scanner and performs arithmetic operations to compute the required values. |
| 4. | Imagine you are developing a budget tracking application. You need to calculate the total expenses for the month. Users will input their daily expenses, and the program should  compute the sum of these expenses. Write a Java program to calculate the sum of elements in an array representing daily expenses.  **PROGRAM CODE:**  import java.util.Scanner;  class JAVA\_P4{  public static void main(String[] args) {  float []daily\_budge = new float[30];  int i;  Scanner sc = new Scanner(System.in);  float t\_expense\_of\_1\_month=0;  for(i=0;i<30;i++){  System.out.print("Day "+(i+1)+" expenses is :");  daily\_budge[i]=sc.nextFloat();  t\_expense\_of\_1\_month+=daily\_budge[i];  }  System.out.println("Total expense of this month is : "+t\_expense\_of\_1\_month);  }  }  **OUTPUT:**    **CONCLUSION:**  This Java program enables users to input their daily expenses over a month, computes the total sum of these expenses, and displays the result. By utilizing an array to store daily expenditures and iterating through each day, the program efficiently accumulates the total expenditure for the entire month. This straightforward approach empowers users to monitor and consolidate their monthly spending effectively, ensuring clarity and convenience in financial tracking. |
| 5. | An electric appliance shop assigns code 1 to motor,2 to fan,3 to tube and 4 for wires. All other items have code 5 or more. While selling the goods, a sales tax of 8% to motor,12% to fan,5% to tube light,7.5% to wires and 3% for all other items is charged. A list containing the product code and price in two different arrays. Write a java program using switch statement to prepare the bill.  **PROGRAM CODE:**  public class JAVA\_P5 {  public static void main(String[] args) {  int[] productCodes = {1, 2, 3, 4, 5};  double[] prices = {350.0, 950.0, 750.0, 450.0, 280.0};  double totalBill = 0;  System.out.printf("Product Code Price Tax Rate Total Price\n");  for (int i = 0; i < productCodes.length; i++) {  int productCode = productCodes[i];  double price = prices[i];  double taxRate;  switch (productCode) {  case 1:  taxRate = 0.08; // 8% tax for motor  break;  case 2:  taxRate = 0.12; // 12% tax for fan  break;  case 3:  taxRate = 0.05; // 5% tax for tube light  break;  case 4:  taxRate = 0.075; // 7.5% tax for wires  break;  default:  taxRate = 0.03; // 3% tax for all other items  break;  }  double taxAmount = price \* taxRate;  double totalPrice = price + taxAmount;  totalBill += totalPrice;  System.out.printf(" %d %.2f %.2f %.2f\n",productCode,price,taxRate  \*100,totalPrice);  }  System.out.println("-----------------------------------------------");  System.out.printf(" Total Bill: %.2f\n", totalBill);  }  }  **OUTPUT:**    **CONCLUSION:**  This Java program calculates the total bill for a predefined list of products, each with specific tax rates based on their product codes. It iterates through arrays of product codes and prices, applying respective tax rates using a switch-case statement. For each product, it computes the tax amount and total price, accumulating the overall bill. Finally, it prints a formatted table detailing each product's pricing breakdown and concludes with the total bill. |
| 6. | Create a Java program that prompts the user to enter the number of days (n) for which they want to generate their exercise routine. The program should then calculate and display the first n terms of the Fibonacci series, representing the exercise duration for each day.  **PROGRAM CODE:**  import java.util.Scanner;  public class JAVA\_P6 {  public static void main(String[] args) {  int i,n;  double a=0,b=1;  double c;  double sum=1;  System.out.print("Enter the number of terms : ");  Scanner sc = new Scanner(System.in);  n = sc.nextInt();  System.out.println("1)"+a);  System.out.println("2)"+b);  for (i=0;i<n-2;i++) {  c=a+b;  a=b;  b=c;  sum+=c;  System.out.println((i+3)+")"+c);  }  System.out.println("Total of fibonaci is = "+sum);  }  }    **OUTPUT:**    **CONCLUSION:**  This Java program computes and displays the Fibonacci sequence up to a specified number of terms entered by the user. It initializes the first two terms (`a=0` and `b=1`), calculates subsequent terms using a loop, and accumulates their sum. The program prints each Fibonacci number and concludes by displaying the total sum of the Fibonacci sequence up to the specified number of terms. This approach provides a clear demonstration of iterative Fibonacci sequence generation and sum calculation in Java, useful for mathematical and algorithmic learning purposes. |