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: 2024-25

PART-I Data Types, Variables, String, Control Statements, Operators, Arrays

No.	Aim of the Practical
1.	Demonstration of installation steps of Java,Introduction to 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.Introduction to Object-Oriented Concepts**: • **Object-Oriented Programming (OOP)** is a paradigm based on the concept of objects. Objects contain data (fields or attributes) and code (methods or functions). Key OOP concepts include: • **Encapsulation**: Binding data and methods together. • **Inheritance**: Creating new classes based on existing ones. • **Polymorphism**: Using a single interface for different data types. • **Abstraction**: Hiding implementation details.

•

2. **Comparison of Java with Other Object-Oriented Languages**:

• Java shares OOP features with languages like C++, C#, and Python.
• Java's strengths include platform independence (thanks to the JVM), strong community support, and extensive libraries.
• Some differences:
• **C++**: Supports multiple inheritance, manual memory management, and operator overloading.
• **C#**: Primarily used for Windows development, integrates with .NET framework.
• **Python**: Dynamically typed, concise, and versatile.

3. **Introduction to JDK, JRE, and JVM**:

• **JDK (Java Development Kit)**:
• Essential for Java developers.
• Includes JRE (for running Java apps) and development tools (compiler, Javadoc, debugger).
• **JRE (Java Runtime Environment)**:
• Needed to run Java applications.
• Consists of JVM (Java Virtual Machine) and core libraries.
• No development tools included.
• **JVM (Java Virtual Machine)**:
• Abstract computing machine that executes Java bytecode.

4. **Introduction to IDEs (Integrated Development Environments)**:

• **Eclipse IDE**:
• Developed by IBM.
• Rich plugin ecosystem.
• Supports Java and other languages.
• **IntelliJ IDEA**:
• Developed by JetBrains.
• Java-based IDE.
• Requires JDK.
• Heavier but feature-rich.
• **NetBeans IDE**:
• Supports standalone, mobile, and web applications.
• **BlueJ**:
• Designed for teaching introductory Java

2.

AIM:

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 andthen display it to the user.

PROGRAM CODE:

```
import java.lang.*;

import java.util.Scanner;

class prac1bank

{

        public static void main(String []args)

        {

                Scanner s=new Scanner(System.in);

                System.out.println("Enter your balance:");

                int balance=s.nextInt();
```

```
        System.out.println("Your balance is:"+balance);

    }

}
```

OUTPUT:

```
C:\Users\saumy\OneDrive\Documents\JAVA\Practical 1>java prac1bank
Enter your balance:
2000
Your balance is:2000
```

CONCLUSION:

This Java program demonstrates a basic concept of storing and displaying data in a variable. In a real-world banking application, the current balance would likely be retrieved from a database or other data storage system, but for simplicity, we hard-coded the value in this example. The program uses the `System.out.println()` method to display the current balance to the user, including the dollar sign and the value of the `currentBalance` variable. This program lays the foundation for more complex banking applications that can perform various transactions and updates to the user's account.

AIM:

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:

3.

```
import java.lang.*;

import java.util.Scanner;

class prac1speed

{
```

```
        public static void main(String []args)
        {
        Scanner sc=new Scanner(System.in);
                System.out.println("Enter the distance(in metres):");
                float d=sc.nextFloat();
                System.out.println("Enter the time(in hour,minute and second
format):");
                float h=sc.nextFloat();
                float m=sc.nextFloat();
                float s=sc.nextFloat();

                float time=(h*60*60)+(m*60)+s;
                System.out.println("Time in sec is:"+time);

                float speed=d/time;
                System.out.println("Speed in m/s is:"+speed);

                float h1=time/3600;
                float d1=d/1000;
                float speed1=d1/h1;
                System.out.println("Speed in km/h is:"+speed1);

                float d2=d/1609;
                float speed2=d2/h1;
```

```
            System.out.println("Speed in miles/h is:"+speed2);

        }

}
```

OUTPUT:

```
C:\Users\saumy\OneDrive\Documents\JAVA\Practical 1>java prac1speed
Enter the distance(in metres):
2000
Enter the time(in hour,minute and second format):
1
2
3
Time in sec is:3723.0
Speed in m/s is:0.53720117
Speed in km/h is:1.9339242
Speed in miles/h is:1.2019417
```

CONCLUSION:

This program is a simple speed calculator that takes in a distance in meters and a time taken in hours, minutes, and seconds. It then calculates and displays the speed in three different units: meters per second, kilometers per hour, and miles per hour. The program uses basic arithmetic operations and unit conversions to perform the calculations. The user is prompted to enter the required input values, and the program outputs the calculated speeds in a clear and readable format. This program can be useful for anyone who needs to calculate speeds in different units, such as athletes, drivers, or cyclists.

4.

AIM:

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.lang.*;
```

```
import java.util.Scanner;
class prac1expense
{
        public static void main(String []args)
        {
        float []a=new float[100];
        int n,i;
        float sum=0;
        Scanner sc=new Scanner(System.in);
   System.out.println("Enter the number of elements in the array:");
        n=sc.nextInt();
        for(i=0;i<n;i++)
        {
         System.out.println("Array element["+(i+1)+"]");
         a[i]=sc.nextFloat();
        }

         for(i=0;i<n;i++)
         {
         sum=sum+a[i];
         }
         System.out.println("Your total expense is:"+sum);
```

```
        }

}
```

OUTPUT:

```
C:\Users\saumy\OneDrive\Documents\JAVA\Practical 1>java prac1expense
Enter the number of elements in the array:
5
Array element[1]
12
Array element[2]
21
Array element[3]
34
Array element[4]
43
Array element[5]
24
Your total expense is:134.0
```

CONCLUSION:

This Java program is a simple budget tracking application that calculates the total expenses for a month. The program prompts the user to input the number of days in the month and then enters a loop to input the daily expenses for each day. The daily expenses are stored in an array, and then the program iterates through the array to calculate the sum of the expenses. Finally, the program displays the total expenses for the month. This program demonstrates basic array manipulation and iteration in Java, as well as user input and output. It can be a useful starting point for building a more comprehensive budget tracking application.

SUPPLEMENTARY EXPERIMENT:

You are creating a library management system. The library has two separate lists of books for fiction and non-fiction. The system should merge these lists into a single list for inventory purposes. Write a Java program to merge two arrays.

PROGRAM CODE:

```
public class supplementary1 {

    public static void main(String[] args) {
```

```
        // Fiction books
        String[] fictionBooks = {"To Kill a Mockingbird", "1984", "The Great Gatsby"};


        // Non-fiction books
        String[] nonFictionBooks = {"The History of the World", "The Universe in a Nutshell", "A Brief History of Time"};


        // Create a new array with the combined length of both arrays
        String[] inventory = new String[fictionBooks.length + nonFictionBooks.length];


        // Copy the elements of the first array into the merged array
        for (int i = 0; i < fictionBooks.length; i++) {
            inventory[i] = fictionBooks[i];
        }


        // Copy the elements of the second array into the merged array
        for (int i = 0; i < nonFictionBooks.length; i++) {
            inventory[fictionBooks.length + i] = nonFictionBooks[i];
        }


        // Print the merged inventory
        System.out.println("Inventory:");
```

```
        for (String book : inventory) {

            System.out.println(book);

        }

    }

}
```

OUTPUT:

```
C:\Users\saumy\OneDrive\Documents\JAVA\Practical 1>java supplementary1
Inventory:
To Kill a Mockingbird
1984
The Great Gatsby
The History of the World
The Universe in a Nutshell
A Brief History of Time
```

CONCLUSION:

This Java program demonstrates how to merge two separate arrays into a single array, which is useful for inventory purposes in a library management system. The program defines two arrays, `fictionBooks` and `nonFictionBooks`, which represent the separate lists of fiction and non-fiction books. The `mergeArrays` method is used to combine these two arrays into a single array, `inventory`, which contains all the books. The program then prints out the merged array, displaying the entire inventory of books.

AIM:

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:

```
import java.lang.*;

import java.util.Scanner;
```

```
class prac1bill
{
    public static void main(String []args)
    {
    Scanner sc=new Scanner(System.in);
   System.out.println("Enter your choice(1 for motor,2 for fan,3 for tube and 4 for wires):");
    int n=sc.nextInt();
    switch(n)
    {
        case 1:
        System.out.println("Enter the price of motor:");
        float m=sc.nextFloat();
        System.out.println("Enter the quantity of motor:");
int q=sc.nextInt();
        float p=m*q;
        float price=p+(0.08f*p);
        System.out.println("Your total charge is:"+price);
        break;

        case 2:
        System.out.println("Enter the price of fan:");
        float m1=sc.nextFloat();
```

5.

```
System.out.println("Enter the quantity of fan:");
int q1=sc.nextInt();
float p1=m1*q1;
float price1=p1+(0.12f*p1);
System.out.println("Your total charge is:"+price1);
break;

case 3:
System.out.println("Enter the price of tube light:");
float m2=sc.nextFloat();
System.out.println("Enter the quantity of tube light:");
int q2=sc.nextInt();
float p2=m2*q2;
float price2=p2+(0.05f*p2);
System.out.println("Your total charge is:"+price2);
break;

case 4:
System.out.println("Enter the price of wires:");
float m3=sc.nextFloat();
System.out.println("Enter the quantity of wires:");
int q3=sc.nextInt();
```

```
                float p3=m3*q3;
                float price3=p3+(0.075f*p3);
                System.out.println("Your total charge is:"+price3);
                break;


                case 5:
                System.out.println("Enter the price of the product:");
                float m4=sc.nextFloat();
System.out.println("Enter the quantity of the product:");
                int q4=sc.nextInt();
                float p4=m4*q4;
                float price4=p4+(0.03f*p4);
                System.out.println("Your total charge is:"+price4);
                break;
        }
        }
}
```

OUTPUT:

```
C:\Users\saumy\OneDrive\Documents\JAVA\Practical 1>java prac1bill
Enter your choice(1 for motor,2 for fan,3 for tube and 4 for wires):
1
Enter the price of motor:
1000
Enter the quantity of motor:
12
Your total charge is:12960.0
```

CONCLUSION:

6.

This Java program demonstrates how to use a switch statement to prepare a bill for an electric appliance shop. The program prompts the user to input the product code and price for each item, and then calculates the tax amount and bill amount for each item based on the product code. The tax rate is determined using a switch statement, which returns the appropriate tax rate based on the product code. The program then prints out the bill amount for each item and the total

AIM:

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.lang.*;

import java.util.Scanner;

class prac1fibonacci

{

public static void main(String args[])

{

int n;

 int n1=0,n2=1,n3,i;

System.out.println("Enter the number of days you exercise(in a week):");

Scanner sc=new Scanner(System.in);

n=sc.nextInt();
```

```
System.out.print(n1+" "+n2);//printing 0 and 1


for(i=2;i<n;++i)
{
 n3=n1+n2;
 System.out.print(" "+n3);
 n1=n2;
 n2=n3;
 }
}
}
```

OUTPUT:

```
C:\Users\saumy\OneDrive\Documents\JAVA\Practical 1>java prac1fibonacci
Enter the number of days you exercise(in a week):
5
0 1 1 2 3
```

CONCLUSION:

This Java program generates an exercise routine for a user-specified number of days, using the Fibonacci series to determine the exercise duration for each day. The program prompts the user to enter the number of days for their exercise routine, and then calculates and displays the first n terms of the Fibonacci series, where each term represents the exercise duration for each day.

SUPPLEMENTARY EXPERIMENT:

Imagine you are developing a classroom management system. You need to keep track of the grades of students in a class. After collecting the grades, you want to display each student's grade along with a message indicating if they have passed or failed. Let's assume the passing grade

is 50.

PROGRAM CODE:

```
import java.util.Scanner;


public class supplementary2 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);


        System.out.print("Enter the number of students: ");
        int numStudents = scanner.nextInt();


        String[] students = new String[numStudents];
        int[] grades = new int[numStudents];


        // Collect student names and grades from user input
        for (int i = 0; i < numStudents; i++) {
            System.out.print("Enter student " + (i + 1) + "'s name: ");
            students[i] = scanner.next();


            System.out.print("Enter student " + (i + 1) + "'s grade: ");
            grades[i] = scanner.nextInt();
        }
```

```
        // Display each student's grade along with a message indicating if they have passed or failed

        for (int i = 0; i < numStudents; i++) {

            System.out.print("Student: " + students[i] + " - Grade: " + grades[i]);



            if (grades[i] >= 50) {

                System.out.println(" - Passed");

            } else {

                System.out.println(" - Failed");

            }

        }

    }

}
```

OUTPUT:

```
C:\Users\saumy\OneDrive\Documents\JAVA\Practical 1>java supplementary2
Enter the number of students: 3
Enter student 1's name: saumya
Enter student 1's grade: 90
Enter student 2's name: khushi
Enter student 2's grade: 85
Enter student 3's name: maniya
Enter student 3's grade: 30
Student: saumya - Grade: 90 - Passed
Student: khushi - Grade: 85 - Passed
Student: maniya - Grade: 30 - Failed
```

CONCLUSION:

This Java program demonstrates a simple classroom management system that keeps track of student grades and displays each student's grade along with a message indicating whether they have passed or failed. The program prompts the user to enter the number of students, and then collects the student names and grades. The program then uses a loop to iterate through the student data and displays each student's grade

	along with a message indicating whether they have passed or failed, based on a passing grade of 50.