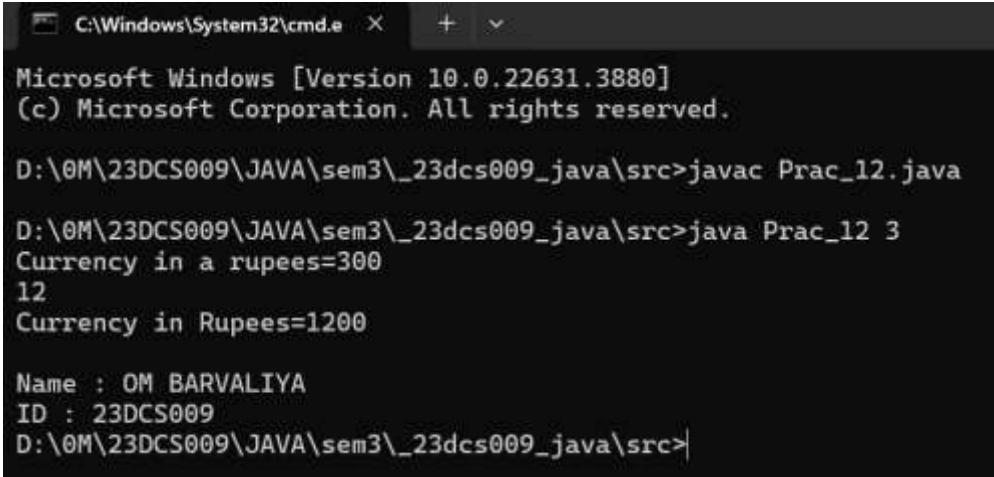


PART-III Object Oriented Programming: Classes, Methods, Constructors

No.	Aim of the Practical
12.	<p><u>AIM :</u> Imagine you are developing a currency conversion tool for a travel agency. This tool should be able to convert an amount in Pounds to Rupees. For simplicity, we assume the conversion rate is fixed: 1 Pound = 100 Rupees. The tool should be able to take input both from command-line arguments and interactively from the user.</p> <p><u>PROGRAM CODE :</u></p> <pre>import java.util.*; class Prac_12 { public static void main(String[] args) { int a= Integer.parseInt(args[0]); int c=a*100; System.out.println("Currency in a rupees=" + c); Scanner S1 = new Scanner(System.in); int R=S1.nextInt(); int P=R*100; System.out.println("Currency in Rupees=" + P); System.out.print("\nName : OM BARVALIYA \nID : 23DCS009 "); } }</pre> <p><u>OUTPUT:</u></p>  <pre>C:\Windows\System32\cmd.e X + v Microsoft Windows [Version 10.0.22631.3880] (c) Microsoft Corporation. All rights reserved. D:\0M\23DCS009\JAVA\sem3_23dcs009_java\src>javac Prac_12.java D:\0M\23DCS009\JAVA\sem3_23dcs009_java\src>java Prac_12 3 Currency in a rupees=300 12 Currency in Rupees=1200 Name : OM BARVALIYA ID : 23DCS009 D:\0M\23DCS009\JAVA\sem3_23dcs009_java\src></pre> <p style="text-align: center;"><u>OUTPUT: PRACTICAL-12</u></p>

	<p><u>CONCLUSION:</u></p> <p>This Java program demonstrates basic arithmetic operations and user input handling. It converts a given amount to a different currency unit by multiplying it with a fixed conversion rate (100). The program utilizes command-line arguments for initial input and the Scanner class for runtime input from the user. Key Java concepts used include parsing command-line arguments, arithmetic operations, and handling user inputs with the Scanner class.</p>
13.	<p><u>AIM :</u> Create a class called Employee that includes three pieces of information as instance variables—a first name (type String), a last name (type String) and a monthly salary (double). Your class should have a constructor that initializes the three instance variables. Provide a set and a get method for each instance variable. If the monthly salary is not positive, set it to 0.0. Write a test application named EmployeeTest that demonstrates class Employee’s capabilities. Create two Employee objects and display each object’s yearly salary. Then give each Employee a 10% raise and display each Employee’s yearly salary</p> <p><u>PROGRAM CODE :</u></p> <pre>import java.util.*; class Employee { private String fname; private String lname; private Double Sal; Employee(String fs, String ls, double sl) { fname = fs; lname = ls; Sal = sl; } void putfs() { System.out.println("" + fname); } void putls() { System.out.println("" + lname); } void putsal() { System.out.println("" + Sal); } public void setFname(String fname) {</pre>

	<pre> this.fname = fname; } public void setLname(String lname) { this.lname = lname; } public void setSal(Double sal) { this.Sal = sal; } public String getFname() { return fname; } public String getLname() { return lname; } public Double getSal() { return Sal; } } public class Prac_13 { public static void main(String[] args) { Employee emp1 = new Employee("John", "Doe", 3000.0); Employee emp2 = new Employee("Jane", "Doe", 4000.0); System.out.println(emp1.getFname() + " " + emp1.getLname() + "'s yearly salary: " + (emp1.getSal() * 12)); System.out.println(emp2.getFname() + " " + emp2.getLname() + "'s yearly salary: " + (emp2.getSal() * 12)); emp1.setSal(emp1.getSal() * 1.10); emp2.setSal(emp2.getSal() * 1.10); System.out.println(emp1.getFname() + " " + emp1.getLname() + "'s yearly salary after 10% raise: " + (emp1.getSal() * 12)); System.out.println(emp2.getFname() + " " + emp2.getLname() + "'s yearly salary after 10% raise: " + (emp2.getSal() * 12)); System.out.print("\nName : OM BARVALIYA \nID : 23DCS009 "); } } </pre>
--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

OUTPUT:

```
PS C:\Users\Om> cd "d:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src\" ;
John Doe's yearly salary: 36000.0
Jane Doe's yearly salary: 48000.0
John Doe's yearly salary after 10% raise: 39600.000000000001
Jane Doe's yearly salary after 10% raise: 52800.0

Name : OM BARVALIYA
ID : 23DCS009
PS D:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src> |
```

OUTPUT: PRACTICAL-13

CONCLUSION:

This code snippet defines an Employee class in Java, showcasing the use of encapsulation, one of the four fundamental OOP concepts. It provides a constructor for initializing objects with first name, last name, and salary. Methods for displaying these properties (putfs, putls, putsal) and setters (setFname, setLname) are included, demonstrating the use of class methods and data hiding by manipulating private data through public methods. This approach ensures controlled access to the class's fields, aligning with the encapsulation principle.

14. **AIM:** Create a class called Date that includes three pieces of information as instance variables—a month (type int), a day (type int) and a year (type int). Your class should have a constructor that initializes the three instance variables and assumes that the values provided are correct. Provide a set and a get method for each instance variable. Provide a method displayDate that displays the month, day and year separated by forward slashes (/). Write a test application named DateTest that demonstrates class Date's capabilities.

PROGRAM CODE :

```
import java.util.*;

class Date {
    int year;
    int month;
    int day;
    Scanner S1 = new Scanner(System.in);

    void getDate() {
        day = S1.nextInt();
```

```

    }

    void getMonth() {
        month = S1.nextInt();
    }

    void getYear() {
        year = S1.nextInt();
    }

    void displayDate() {
        System.out.println(day + "/" + month + "/" + year);
    }

    Date(int d, int m, int y) {
        day = d;
        month = m;
        year = y;
    }
}

class Prac_14 {
    public static void main(String[] args) {
        Date d1 = new Date(28, 07, 2005);
        d1.displayDate();
        System.out.print("\nName : OM BARVALIYA \nID : 23DCS009 ");
    }
}

```

OUTPUT:

```

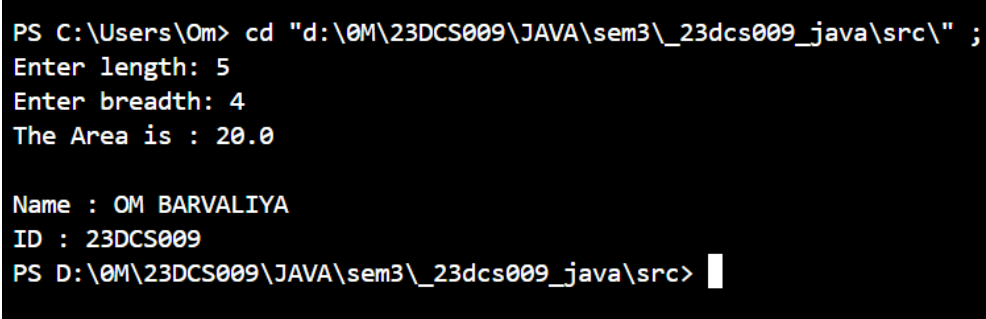
PS C:\Users\Om> cd "d:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src\" ;
28/7/2005

Name : OM BARVALIYA
ID : 23DCS009
PS D:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src>

```

OUTPUT: PRACTICAL-14

	<p><u>CONCLUSION:</u></p> <p>This Java code snippet demonstrates the use of classes and objects, fundamental concepts of Object-Oriented Programming (OOP). It defines a Date class with a constructor to initialize the date and a method to display it. The main method in the class creates an instance of the Date class and calls its displayDate method to print the date. This example illustrates encapsulation by keeping the date fields private and accessing them through a public method, and constructor overloading by providing a specific constructor for date initialization.</p>
15.	<p><u>AIM :</u> Write a program to print the area of a rectangle by creating a class named 'Area' taking the values of its length and breadth as parameters of its constructor and having a method named 'returnArea' which returns the area of the rectangle. Length and breadth of rectangle are entered through keyboard.</p> <p><u>PROGRAM CODE :</u></p> <pre>import java.util.Scanner; class Area { double length; double breadth; Area(double l, double b) { length = l; breadth = b; } double returnArea() { return length * breadth; } } class Prac_15 { public static void main(String[] args) { Scanner scanner = new Scanner(System.in); System.out.print("Enter length: "); double length = scanner.nextDouble(); System.out.print("Enter breadth: "); double breadth = scanner.nextDouble(); Area A = new Area(length, breadth); System.out.println("The Area is : "+A.returnArea()); System.out.print("\nName : OM BARVALIYA \nID : 23DCS009 "); } }</pre>

	<pre>} <u>OUTPUT:</u></pre>  <p style="text-align: center;"><u>OUTPUT: PRACTICAL-15</u></p> <p><u>CONCLUSION:</u></p> <p>This Java code snippet demonstrates the use of classes, objects, and methods, which are key principles of Object-Oriented Programming (OOP). It includes user input handling with the Scanner class to dynamically receive length and breadth values. An Area class instance is created with these values, and its method returnArea() calculates and returns the area of a rectangle. The program showcases encapsulation by using class methods to operate on private class fields. Additionally, it illustrates the practical application of constructors for initializing object states.</p>
16.	<p><u>AIM :</u> Print the sum, difference and product of two complex numbers by creating a class named ‘Complex’ with separate methods for each operation whose real and imaginary parts are entered by user.</p> <p><u>PROGRAM CODE :</u></p> <pre>import java.util.*; public class Complex_16 { private double real; private double imaginary; public Complex_16 () { } public Complex_16 (double real, double imaginary) { this.real = real; } }</pre>

	<pre> this.imaginary = imaginary; } public static Complex_16 getInput() { Scanner scanner = new Scanner(System.in); System.out.print("Enter real part: "); double real = scanner.nextDouble(); System.out.print("Enter imaginary part: "); double imaginary = scanner.nextDouble(); return new Complex_16 (real, imaginary); } public Complex_16 add(Complex_16 other) { return new Complex_16(real + other.real, imaginary + other.imaginary); } public Complex_16 subtract(Complex_16 other) { return new Complex_16(real - other.real, imaginary - other.imaginary); } public Complex_16 multiply(Complex_16 other) { double newReal = (real * other.real) - (imaginary * other.imaginary); double newImaginary = (real * other.imaginary) + (imaginary * other.real); return new Complex_16(newReal, newImaginary); } public String toString() { return String.format("%.2f + %.2fi", real, imaginary); } public static void main(String[] args) { System.out.println("Enter first complex number:"); Complex_16 number1 = getInput(); System.out.println("Enter second complex number:"); Complex_16 number2 = getInput(); Complex_16 sum = number1.add(number2); Complex_16 difference = number1.subtract(number2); Complex_16 product = number1.multiply(number2); System.out.println("Sum: " + sum); System.out.println("Difference: " + difference); System.out.println("Product: " + product); System.out.print("\nName : OM BARVALIYA \nID : 23DCS009 "); } } </pre>
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

OUTPUT:

```
PS C:\Users\Om> cd "d:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src\" ;
Enter first complex number:
Enter real part: 5
Enter imaginary part: 4
Enter second complex number:
Enter real part: 6
Enter imaginary part: 2
Sum: 11.00 + 6.00i
Difference: -1.00 + 2.00i
Product: 22.00 + 34.00i

Name : OM BARVALIYA
ID : 23DCS009
PS D:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src> |
```

OUTPUT: PRACTICAL-16

CONCLUSION:

This Java code demonstrates the use of object-oriented programming (OOP) principles to perform arithmetic operations (addition, subtraction, multiplication) on complex numbers. It utilizes classes, objects, and methods to encapsulate the behavior and state of complex numbers. The toString method showcases polymorphism by overriding the default method in the Object class for custom string representation. The code also exemplifies the use of static methods and user input handling in Java.