



Devang Patel Institute of Advance Technology and Research

(A Constituent Institute of CHARUSAT)

Certificate

This is to certify that

Mr./Mrs. Valand Vishva P.

of 3CSE2 *Class,*

ID. No. 23DCS140 *has satisfactorily completed*

his/ her term work in Java Programming *for*

the ending in NOV. 2024/2025

Date : 16/10/24


Sign. of Faculty


Head of Department

**CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY (CHARUSAT)
DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY AND RESEARCH
DEPSTAR**

Subject : JAVA PROGRAMMING

Semester: 3

Subject Code: CSE201

Academic Year :2024-25

Course Outcome (COs):

At the end of the course, the students will be able to:

CO1	Comprehend Java Virtual Machine architecture and Java Programming Fundamentals.
CO2	Demonstrate basic problem-solving skills: analyzing problems, modelling a problem as a system of objects, creating algorithms, and implementing models and algorithms in an object-oriented computer language (classes, objects, methods with parameters)
CO3	Design applications involving Object Oriented Programming concepts such as inheritance, polymorphism, abstract classes and interfaces.
CO4	Build and test program using exception handling
CO5	Design and build multi-threaded Java Applications.
CO6	Build software using concepts such as files and collection frameworks.

Bloom's Taxonomy:

Level 1- Remembering

Level 2- Understanding

Level 3- Applying

Level 4- Analyzing

Level 5- Evaluating

Level 6- Creating

**CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY (CHARUSAT)
DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY AND RESEARCH
DEPSTAR**

Practical List

Sr No.	AIM	Hrs.	CO	Bloom's Taxonomy
PART-I Data Types, Variables, String, Control Statements, Operators, Arrays				
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.	2	1	1
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.	1	1	2,3,4
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).	1	1	2,3,4
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. 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.	1	1, 2	2,3
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.	1	1, 2	2
6	Create a Java program that prompts the user to enter the	1	1, 2	2,3,4

CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY (CHARUSAT)
DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY AND RESEARCH
DEPSTAR

	<p>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.</p> <p>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.</p>			
PART-II Strings				
7	<p>Given a string and a non-negative int n, we'll say that the front of the string is the first 3 chars, or whatever is there if the string is less than length 3. Return n copies of the front;</p> <p>front_times('Chocolate', 2) → 'ChoCho'</p> <p>front_times('Chocolate', 3) → 'ChoChoCho'</p> <p>front_times('Abc', 3) → 'AbcAbcAbc'</p>	1	1, 2	2,3,4
8	<p>Given an array of ints, return the number of 9's in the array. array_count9([1, 2, 9]) → 1</p> <p>array_count9([1, 9, 9]) → 2</p> <p>array_count9([1, 9, 9, 3, 9]) → 3</p> <p>Supplementary Experiment: 1. Write a Java program to replace each substring of a given string that matches the given regular expression with the given replacement.</p> <p>Sample string : "The quick brown fox jumps over the lazy dog."</p> <p>In the above string replace all the fox with cat.</p>	1	1, 2	2,3
9	<p>Given a string, return a string where for every char in the original, there are two chars.</p> <p>double_char('The') → 'TThhee'</p> <p>double_char('AAbb') → 'AAAAbbbb'</p> <p>double_char('Hi-There') → 'HHii--TThheerree'</p>	1	1, 2	2
10	<p>Perform following functionalities of the string:</p> <ul style="list-style-type: none"> ● Find Length of the String ● Lowercase of the String ● Uppercase of the String ● Reverse String 	1	1, 2	2,3,4

**CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY (CHARUSAT)
 DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY AND RESEARCH
 DEPSTAR**

	Sort the string			
11	Perform following Functionalities of the string: “CHARUSAT UNIVERSITY” <ul style="list-style-type: none"> ● Find length ● Replace ‘H’ by ‘FIRST LATTER OF YOUR NAME’ ● Convert all character in lowercase Supplementary Experiment: 1. Write a Java program to count and print all duplicates in the input string. Sample Output: The given string is: resource The duplicate characters and counts are: e appears 2 times r appears 2 times	1	1, 2	4
PART-III Object Oriented Programming: Classes, Methods, Constructors				
12	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.	1	2	3
13	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 again.	2	1, 2	3
14	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.	2	1, 2	3

**CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY (CHARUSAT)
 DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY AND RESEARCH
 DEPSTAR**

15	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. Supplementary Experiment: 1. Write a Java program to create a class called "Airplane" with a flight number, destination, and departure time attributes, and methods to check flight status and delay. [L:M]	1	1, 2	3
16	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.	1	1, 2	2,3
PART-IV Inheritance, Interface, Package				
17	Create a class with a method that prints "This is parent class" and its subclass with another method that prints "This is child class". Now, create an object for each of the class and call 1 - method of parent class by object of parent	1	1, 2, 3	3
18	Create a class named 'Member' having the following members: Data members 1 - Name 2 - Age 3 - Phone number 4 - Address 5 - Salary It also has a method named 'printSalary' which prints the salary of the members. Two classes 'Employee' and 'Manager' inherits the 'Member' class. The 'Employee' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an employee and a manager by making an object of both of these classes and print the same.	2	1, 2, 3	3
19	Create a class named 'Rectangle' with two data members 'length' and 'breadth' and two methods to print the area and perimeter of the rectangle respectively. Its constructor having parameters for length and breadth is used to initialize length and breadth of the rectangle. Let class 'Square' inherit the 'Rectangle' class with its constructor having a parameter for its side (suppose s) calling the	1	2,3	3

**CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY (CHARUSAT)
DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY AND RESEARCH
DEPSTAR**

	<p>constructor of its parent class as 'super(s,s)'. Print the area and perimeter of a rectangle and a square. Also use array of objects.</p> <p>Supplementary Experiment:</p> <p>1. Write a Java program to create a vehicle class hierarchy. The base class should be Vehicle, with subclasses Truck, Car and Motorcycle. Each subclass should have properties such as make, model, year, and fuel type. Implement methods for calculating fuel efficiency, distance traveled, and maximum speed. [L:A]</p>			
20	<p>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.</p>	2	2,3	3
21	<p>Create a class 'Degree' having a method 'getDegree' that prints "I got a degree". It has two subclasses namely 'Undergraduate' and 'Postgraduate' each having a method with the same name that prints "I am an Undergraduate" and "I am a Postgraduate" respectively. Call the method by creating an object of each of the three classes.</p>	1	2,3	3
22	<p>Write a java that implements an interface AdvancedArithmetic which contains amethod signature int divisor_sum(int n). You need to write a class called MyCalculator which implements the interface. divisorSum function just takes an integer as input and return the sum of all its divisors.</p> <p>For example, divisors of 6 are 1, 2, 3 and 6, so divisor_sum should return 12. The value of n will be at most 1000.</p> <p>Supplementary Experiment:</p> <p>1. Write a Java programming to create a banking system with three classes - Bank, Account, SavingsAccount, and CurrentAccount. The bank should have a list of accounts and methods for adding them. Accounts should be an interface with methods to deposit, withdraw,</p>	2	2,3	2,3

**CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY (CHARUSAT)
 DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY AND RESEARCH
 DEPSTAR**

	calculate interest, and view balances. SavingsAccount and CurrentAccount should implement the Account interface and have their own unique methods. [L:A]			
23	Assume you want to capture shapes, which can be either circles (with a radius and a color) or rectangles (with a length, width, and color). You also want to be able to create signs (to post in the campus center, for example), each of which has a shape (for the background of the sign) and the text (a String) to put on the sign. Create classes and interfaces for circles, rectangles, shapes, and signs. Write a program that illustrates the significance of interface default method.	2	2,3	6
PART-V Exception Handling				
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.	1	4	3
25	Write a Java program that throws an exception and catch it using a try-catch block.	1	4	3
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). Supplementary Experiment: 1. Write a Java program that reads a list of integers from the user and throws an exception if any numbers are duplicates. [L:M]	2	4	2,3
PART-VI File Handling & Streams				
27	Write a program that will count the number of lines in each file that is specified on the command line. Assume that the files are text files. Note that multiple files can be specified, as in "java Line Counts file1.txt file2.txt file3.txt". Write each file name, along with the number of lines in that file, to standard output. If an error occurs while trying to read from one of the files, you should print an error message for that file, but you should still process all the remaining files.	1	4,6	3
28	Write an example that counts the number of times a	1	4,6	3

**CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY (CHARUSAT)
DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY AND RESEARCH
DEPSTAR**

	particular character, such as e, appears in a file. The character can be specified at the command line. You can use xanadu.txt as the input file.			
29	Write a Java Program to Search for a given word in a File. Also show use of Wrapper Class with an example.	2	4,6	3
30	Write a program to copy data from one file to another file. If the destination file does not exist, it is created automatically. Supplementary Experiment: 1. Write a Java program to sort a list of strings in alphabetical order, ascending and descending using streams.	2	4,6	3
31	Write a program to show use of character and byte stream. Also show use of BufferedReader/BufferedWriter to read console input and write them into a file.	2	4,6	2,3
PART-VII Multithreading				
32	Write a program to create thread which display “Hello World” message. A. by extending Thread class B. by using Runnable interface.	1	5,6	3
33	Write a program which takes N and number of threads as an argument. Program should distribute the task of summation of N numbers amongst number of threads and final result to be displayed on the console.	1	5,6	3
34	Write a java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.	2	5,6	3
35	Write a program to increment the value of one variable by one and display it after one second using thread using sleep() method.	2	5,6	2,3
36	Write a program to create three threads ‘FIRST’, ‘SECOND’, ‘THIRD’. Set the priority of the ‘FIRST’ thread to 3, the ‘SECOND’ thread to 5(default) and the ‘THIRD’ thread to 7.	2	5,6	2,3
37	Write a program to solve producer-consumer problem using thread synchronization.	2	5,6	3
PART-VIII Collection Framework and Generic				
38	Design a Custom Stack using ArrayList class, which	2	5	3

**CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY (CHARUSAT)
 DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY AND RESEARCH
 DEPSTAR**

	implements following functionalities of stack. My Stack -list ArrayList<Object>: A list to store elements. +isEmpty(): boolean: Returns true if this stack is empty. +getSize(): int: Returns number of elements in this stack. +peek(): Object: Returns top element in this stack without removing it. +pop(): Object: Returns and Removes the top elements in this stack. +push(o: object): Adds new element to the top of this stack.			
39	Imagine you are developing an e-commerce application. The platform needs to sort lists of products based on different criteria, such as price, rating, or name. Each product object implements the Comparable interface to define the natural ordering. To ensure flexibility and reusability, you need a generic method that can sort any array of Comparable objects. Create a generic method in Java that sorts an array of Comparable objects. This method should be versatile enough to sort arrays of different types of objects (such as products, customers, or orders) as long as they implement the Comparable interface.	2	5	6
40	Write a program that counts the occurrences of words in a text and displays the words and their occurrences in alphabetical order of the words. Using Map and Set Classes.	2	5	3
41	Write a code which counts the number of the keywords in a Java source file. Store all the keywords in a HashSet and use the contains () method to test if a word is in the keyword set.	2	5	2,3

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: CSE-201****Academic year: 2024 - 2025****Part - 1**

No.	Aim of the Practical
1.	<p>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.</p> <p><u>OUTPUT:</u></p> <p>Installation Steps of Java</p> <ol style="list-style-type: none">1. Download the JDK2. Install the JDK3. Set Up Environment Variables (Windows)4. Verify the Installation <p>Introduction to Object-Oriented Concepts</p> <ol style="list-style-type: none">1. Class and Object Class: A blueprint for creating objects. Object: An instance of a class.2. Inheritance - Mechanism where one class inherits the fields

and methods of another.

3. Polymorphism

- Ability of different classes to respond to the same method call in different ways.

4. Encapsulation

- Wrapping of data and methods into a single unit (class), and restricting access to some of the object's components.

5. Abstraction

- Hiding complex implementation details and showing only the necessary features.

:: Java vs. C++

- Memory Management: Java has automatic garbage collection, whereas C++ requires manual memory management.
- Platform Independence: Java is platform-independent at the source level (write once, run anywhere), while C++ is platform-dependent.
- Pointers: Java does not support pointers explicitly for security reasons, whereas C++ supports pointers.

:: Introduction to JDK, JRE, JVM, Javadoc,

JDK (Java Development Kit)

- A software development kit used to develop Java applications.

JRE (Java Runtime Environment)

- Provides the libraries, Java Virtual Machine (JVM), and other components to run

	<p>applications written in Java.</p> <p>JVM (Java Virtual Machine)</p> <ul style="list-style-type: none">- An abstract machine that enables your computer to run a Java program.- Converts bytecode into machine code, ensuring Java programs can run on any platform.
2.	<p>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.</p> <p><u>PROGRAM CODE :</u></p> <pre>class prac2 { public static void main(String []args){ float current_balance = 20; System.out.print("current balance is "); System.out.print("\$"); System.out.println(current_balance); System.out.println("23DCS140 Vishva"); } }</pre>

OUTPUT:

```
current balance is $20.0  
23DCS140 Vishva
```

CONCLUSION: This code teaches us how to use variables in java and how to store different types of values in variables.

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.Scanner;
```

```
public class prac3{
```

```
    public static void main(String[] args){  
        Scanner sc = new Scanner(System.in);
```

```
        System.out.println("Enter Distance (in meter)");  
        double distance=sc.nextDouble();
```

```
        System.out.println("Enter the time taken - hours: ");  
        int hours = sc.nextInt();
```

```
        System.out.println("Enter the time taken - minutes: ");  
        int minutes = sc.nextInt();
```



```
System.out.println("Enter the time taken - seconds: ");
int seconds = sc.nextInt();

int totaltime=(hours * 3600) + (minutes * 60) + seconds;

double speedMetersPerSecond = distance / totaltime;

double speedKilometersPerHour = (distance / 1000.0) / (totaltime / 3600.0);

double speedMilesPerHour = (distance / 1609.0) / (totaltime / 3600.0);

System.out.println("Speed in meters per second:"+ speedMetersPerSecond);
System.out.println("Speed in kilometers per hour: "+ speedKilometersPerHour);
System.out.println("Speed in miles per hour:"+ speedMilesPerHour);

System.out.println("23DCS140 Vishva ");

}
}
```

OUTPUT:

```
Enter the time taken - hours:
1
Enter the time taken - minutes:
30
Enter the time taken - seconds:
0
Speed in meters per second:1.8518518518518519
Speed in kilometers per hour: 6.666666666666667
Speed in miles per hour:4.143360265175057
23DCS140 Vishva
```

CONCLUSION: The Java program calculates speed from user-input distance and time in various units (meters per second, kilometers per hour, and miles per hour). It demonstrates basic input handling, arithmetic operations, and output formatting in Java.

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.*;

public class prac4 {
    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        int arr[]= new int[30];
        int sum=0;
```

```
for(int i=0;i<30;i++){  
    System.out.print("enter expenses of day "+(i+1) + ": ");  
    arr[i]=sc.nextInt();  
    sum=sum+arr[i];  
}  
  
System.out.println("total :"+ sum);  
System.out.println("23DCS140 Vishva");  
}  
}
```

OUTPUT:

```
enter expenses of day 1: 30  
enter expenses of day 2: 5  
enter expenses of day 3: 2  
enter expenses of day 4: 8  
enter expenses of day 5: 6  
enter expenses of day 6: 5  
enter expenses of day 7: 32  
enter expenses of day 8: 36  
enter expenses of day 9: 35  
enter expenses of day 10: 26  
enter expenses of day 11: 36  
enter expenses of day 12: 32  
enter expenses of day 13: 36  
enter expenses of day 14: 85  
enter expenses of day 15: 95  
enter expenses of day 16: 75  
enter expenses of day 17: 65  
enter expenses of day 18: 42  
enter expenses of day 19: 15  
enter expenses of day 20: 25  
enter expenses of day 21: 36  
enter expenses of day 22: 85  
enter expenses of day 23: 62  
enter expenses of day 24: 42  
enter expenses of day 25: 52  
enter expenses of day 26: 14  
enter expenses of day 27: 17  
enter expenses of day 28: 48  
enter expenses of day 29: 48  
enter expenses of day 30: 95  
total :1190  
23DCS140 Vishva
```

CONCLUSION : The Java program records daily expenses for a month and calculates the total monthly expenses based on user input. It uses an array to store daily expenses and a loop to iterate through each day's input. The program demonstrates basic array handling, looping, and accumulation of values in Java.

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 :

```
import java.util.Scanner;
```

```
public class prac5 {
```

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

```
    {
```

```
        Scanner sc= new Scanner(System.in);
```

```
        int code[]={ 1,2,3,4,5,6};
```

```
        double price[]=new double[5];
```

```
        double tax,total=0;
```

```
        int no;
```

```
        do{
```

```
            System.out.println("1.motor\n2.fan\n3.tube\n4.wires\n5.other items");
```

```
            no= sc.nextInt();
```

```
switch(code[no-1])
{
    case 1:
    {
        System.out.print("Enter Price of motor:");
        price[0]=sc.nextDouble();
        tax=price[0]*(8/100);
        total=total+(price[0]+tax);
    }
    break;
    case 2:
    {
        System.out.print("Enter Price of fan:");
        price[1]=sc.nextDouble();
        tax=price[1]*(12/100);
        total=total+(price[1]+tax);
    }
    break;
    case 3:
    {
        System.out.print("Enter Price of tube:");
        price[2]=sc.nextDouble();
        tax=price[2]*(5/100);
        total=total+(price[2]+tax);
    }
}
```

```
        break;
        case 4:
        {
            System.out.print("Enter Price of wires:");
            price[3]=sc.nextDouble();
            tax=price[3]*(7.5/100);
            total=total+(price[3]+tax);
        }
        break;
        case 5:
        {
            System.out.print("Enter Price of other:");
            price[4]=sc.nextDouble();
            tax=price[4]*(3/100);
            total=total+(price[4]+tax);

        }
        break;

        default:
            System.out.println("wrong");
            break;
    }

}while(no!=6);

System.out.println("TOTAL BILL PRICE:"+total);
System.out.println("23DCS140 Vishva");
```



```
}  
}
```

OUTPUT:

```
1.motor  
2.fan  
3.tube  
4.wires  
5.other items  
6.exit  
1  
Enter Price of motor:500  
1.motor  
2.fan  
3.tube  
4.wires  
5.other items  
6.exit  
2  
Enter Price of fan:500  
1.motor  
2.fan  
3.tube  
4.wires  
5.other items  
6.exit  
3  
Enter Price of tube:500  
1.motor  
2.fan  
3.tube  
4.wires  
5.other items  
6.exit  
4  
Enter Price of wires:500
```

```

Enter Price of wires:500
1.motor
2.fan
3.tube
4.wires
5.other items
6.exit
5
Enter Price of other:500
1.motor
2.fan
3.tube
4.wires
5.other items
6.exit
6
TOTAL BILL PRICE:2537.5
23DCS140 Vishva

```

CONCLUSION:

The Java program simulates an electric appliance shop transaction, allowing users to select items and calculate the total bill based on predefined prices and taxes. It utilizes arrays, loops, and switch-case statements for functionality and demonstrates formatted output for a detailed bill summary.

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 prac6
{

    public static void main(String[] args)
    {
        System.out.println("Enter Number:");
    }
}

```

```
Scanner sc=new Scanner(System.in);

int n=sc.nextInt();
    sc.close();

if (n <= 0) {
    System.out.println("Please enter a positive number.");
    return;
}

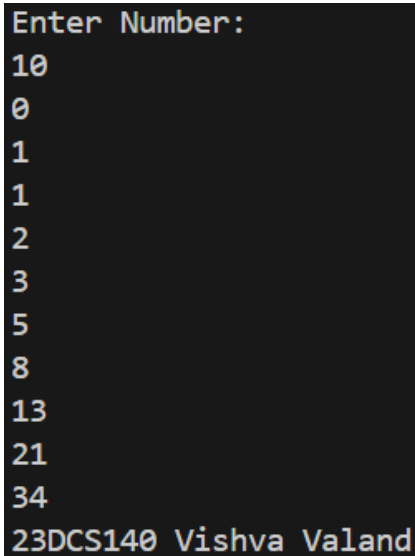
if(n==1)
{
    System.out.println("0");
}

else if(n==2)
{
    System.out.println("0\n1");
}

else{
    int first=0;
    int second=1;
    System.out.println(first);
    System.out.println(second);

    for(int i=3;i<=n;i++)
```

```
{  
    int next=first+second;  
  
    System.out.println(next);  
    first=second;  
    second=next;  
}  
}  
  
System.out.println("23DCS140 Vishva Valand");  
}  
}
```

OUTPUT:

```
Enter Number:  
10  
0  
1  
1  
2  
3  
5  
8  
13  
21  
34  
23DCS140 Vishva Valand
```

CONCLUSION: This Java program uses basic constructs like loops and variables to generate an exercise routine. Specifically, it calculates and prints the exercise duration for each day based on the Fibonacci sequence for a specified number of days.

7. Given a string and a non-negative int n, we'll say that the front of the string is the first 3 chars, or whatever is there if the string is less than length 3. Return n copies of the front;
- front_times('Chocolate', 2) → 'ChoCho'
- front_times('Chocolate', 3) → 'ChoChoCho'
- front_times('Abc', 3) → 'AbcAbcAbc'

PROGRAM CODE :

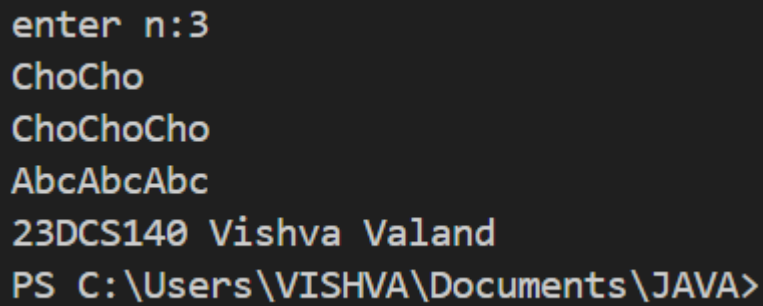
```
import java.util.*;

public class practical7 {
    public static void main(String[] args){
        String s1="Chocolate";
        String s2="Abc";
        Scanner sc=new Scanner(System.in);
        System.out.print("enter n:");
        int n=sc.nextInt();
        front_times(s1, 2);
        System.out.println("");
        front_times(s1, 3);
        System.out.println("");
        front_times(s2,n);

        System.out.println("23DCS140 Vishva Valand");

    }

    static void front_times(String s1,int n ){
        String s=s1.substring(0,3);
        for(int i=0;i<n;i++){
            System.out.print(s);
        }
    }
}
```

OUTPUT:

```
enter n:3
ChoCho
ChoChoCho
AbcAbcAbc
23DCS140 Vishva Valand
PS C:\Users\VISHVA\Documents\JAVA>
```

8. Given an array of ints, return the number of 9's in the array.
- array_count9([1, 2, 9]) → 1
 - array_count9([1, 9, 9]) → 2
 - array_count9([1, 9, 9, 3, 9]) → 3

PROGRAM CODE :

```
public class prac8 {
    public static void main(String[] args){
        int arr[]={1,2,9};
        int arr2[]={1,9,9};
        int arr3[]={1,9,9,3,9};
        array_count(arr);
        array_count(arr2);
        array_count(arr3);
        System.out.println("\n23DCS140 Vishva Valand");
    }

    static void array_count(int arr[]){
        int count=0;
        for(int i=0;i<arr.length;i++){
            if(arr[i]==9){
                count++;
            }
        }
    }
}
```



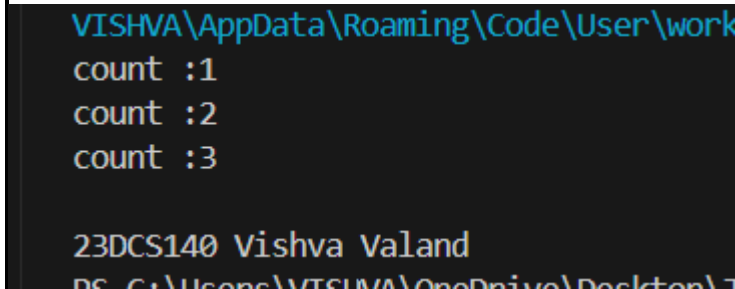
```

        System.out.println("count :"+count);
    }

}

```

OUTPUT:



```

VISHVA\AppData\Roaming\Code\User\work
count :1
count :2
count :3

23DCS140 Vishva Valand
PS C:\Users\VISHVA\OneDrive\Desktop>

```

CONCLUSION:

The Java program counts occurrences of the number 9 in an array of integers input by the user. It utilizes a method (`Array_count9`) to iterate through the array and count occurrences of the specified number. The program demonstrates basic array handling, method invocation, and input/output operations using Scanner.

9. Given a string, return a string where for every char in the original, there are two chars.
 double_char('The') → 'TThhee'
 double_char('AAAbb') → 'AAAAbbbb'
 double_char('Hi-There') → 'HHii--TThheerree'.

PROGRAM CODE :

```

import java.util.*;

class prac9 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.println("Enter String:");
    }
}

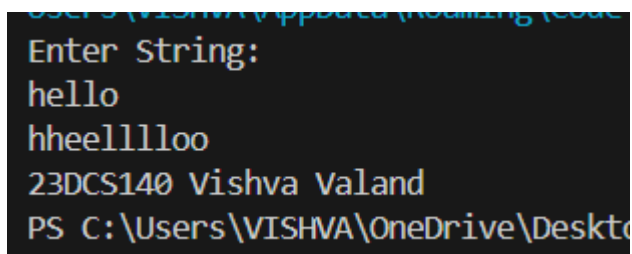
```

```
String s1 = sc.nextLine();

double_char(s1);
System.out.println("\n23DCS140 Vishva Valand");

}

static void double_char(String s1) {
    for (int i = 0; i < s1.length(); i++) {
        System.out.print(s1.charAt(i));
        System.out.print(s1.charAt(i));
    }
}
}
```

OUTPUT:A screenshot of a terminal window showing the execution of a Java program. The prompt 'Enter String:' is followed by the input 'hello'. The output shows 'hheelllloo' where each character of 'hello' is repeated twice. Below this, the program prints '\n23DCS140 Vishva Valand'. The terminal path is 'PS C:\Users\VISHVA\OneDrive\Desktop'.**CONCLUSION:**

This Java program takes a user-inputted string and calculate the length of the string, duplicates each character in that string, and then prints the double char of that string to the output.

10.

Perform following functionalities of the string:

- Find Length of the String
- Lowercase of the String
- Uppercase of the String
- Reverse String

PROGRAM CODE :

```
import java.util.Arrays;
import java.util.Scanner;
class prac10 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter string : ");
        String s = sc.nextLine();
        System.out.println("Length of String: " + s.length());
        System.out.println("Lowercase of the string: " + s.toLowerCase());
        System.out.println("Uppercase of the string: " + s.toUpperCase());
        System.out.print("String in reverse order :");
        for (int i = s.length() - 1; i >= 0; i--) {
            System.out.print(s.charAt(i));
        }
        System.out.println();
        char[] temp = s.toCharArray();
        Arrays.sort(temp);
        String sorted_string = new String(temp);
        System.out.print("Sorted String:");
        System.out.println(sorted_string);
        System.out.println("\n23DCS140 Vishva Valand");
    }
}
```

OUTPUT:

```
Enter string : vishva
Length of String: 6
Lowercase of the string: vishva
Uppercase of the string: VISHVA
String in reverse order :avhsiv
Sorted String:ahisvv
```

```
23DCS140 Vishva Valand
```

CONCLUSION:

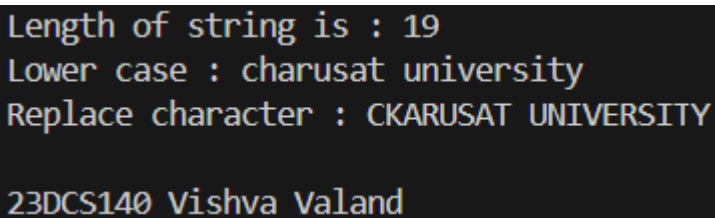
In this java program we learn and different types of String methods like for counting the length of string, to convert it to lower or uppercase ,etc.

11. Perform following Functionalities of the string:
“CHARUSAT UNIVERSITY”
- Find length
 - Replace ‘H’ by ‘FIRST LATTER OF YOUR NAME’
 - Convert all character in lowercase

PROGRAM CODE :

```
public class prac11 {  
    public static void main(String [] args){  
        String s1="CHARUSAT UNIVERSITY";  
        System.out.println("Length of string is : " + s1.length());  
        System.out.println("Lower case : " + s1.toLowerCase());  
        System.out.println("Replace character : " + s1.replace('H', 'K'));  
        System.out.println("\n23DCS140 Vishva Valand");  
    }  
}
```

OUTPUT:



```
Length of string is : 19  
Lower case : charusat university  
Replace character : CKARUSAT UNIVERSITY  
  
23DCS140 Vishva Valand
```

CONCLUSION:

In this java program we again uses the String method and how we can replace a char with another char in string using String method.

Part - 3

12.

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.

PROGRAM CODE :

```
import java.util.*;

class prac12{

static double convert(double p)
{
    double r=p/100.0;
    System.out.println("Enter Value of ruppe: "+r);
    return r;
}

public static void main(String[] args)
{

    // System.out.print("Enter Value of Pound: ");
    double po=Double.parseDouble(args[0]);
    convert(po);
    Scanner sc=new Scanner(System.in);
    System.out.print("Enter Value of Pound: ");
    double pound=sc.nextDouble();
    convert(pound);
    System.out.println("\n23DCS140 Vishva Valand");

}
}
```


OUTPUT:

```

PS C:\Users\VISHVA\OneDrive\Desktop\JAVA> cd set3
PS C:\Users\VISHVA\OneDrive\Desktop\JAVA\set3> javac prac12.java
PS C:\Users\VISHVA\OneDrive\Desktop\JAVA\set3> java prac12 100
Enter Value of ruppe: 1.0

23DCS140 Vishva Valand
PS C:\Users\VISHVA\OneDrive\Desktop\JAVA\set3>

```

CONCLUSION:

13. This Java program converts an input value in pounds to rupees using a fixed conversion rate of 100 rupees per pound. It demonstrates basic command-line argument handling and arithmetic operations in Java for currency conversion.

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 again.

PROGRAM CODE:

```

import java.util.*;
class Employee{
    double salary;
    String fname,lname;

    Employee(){
        fname = null;
        lname = null;
        salary = 0.0;
    }

    void get()
    {

```

```
Scanner sc = new Scanner(System.in);
System.out.print("Enter your first name:");
fname = sc.nextLine();
System.out.print("Enter your last name:");
lname = sc.nextLine();
System.out.print("Enter your salary:");
salary = sc.nextDouble();
}

void display(){
    System.out.println("First name :" + fname);
    System.out.println("Last name :" + lname);
    if(salary<0.0){
        salary = 0.0;
        System.out.println("Please enter valid salary.");
    }
    else{
        System.out.println("Yearly salary :" + salary * 12);
        System.out.println("Increased yearly salary is :" + (salary + (salary *12 * 0.1)));
    }
}

class Pract13{
    public static void main(String [] args){
        System.out.println("Employee 1:");
        Employee e1 = new Employee();
        e1.get();
        e1.display();
        System.out.println("\nEmployee 2:");
        Employee e2 = new Employee();
        e2.get();
        e2.display();

    }
}
```

OUTPUT:

```
Enter Fname:vishva
Enter LName:valand
Enter Salary:100000
First name: vishva
First name: valand
Yearly Salary: 1200000.0
Increased salary is :110000.0
PS C:\Users\VISHVA\OneDrive\Desktop\JAVA>
```

CONCLUSION:

The Java program efficiently manages employee details by capturing user input for name and salary, then calculating and displaying the yearly salary. It handles salary validation and applies a 10% increase for output.

14. 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.Scanner;

class Date{
Scanner sc=new Scanner(System.in);

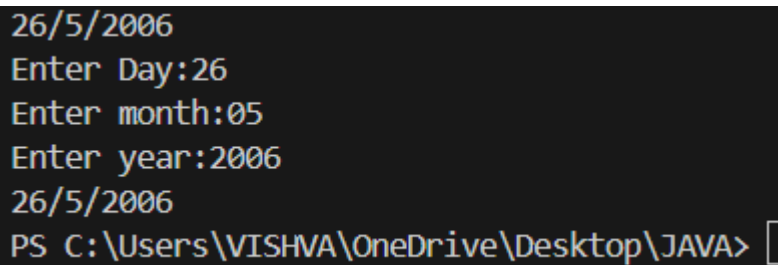
int m,d,y;

Date(int dd,int mo,int yy)
{
    m=mo;
    d=dd;
    y=yy;
}
void getdata()
{
    System.out.print("Enter Day:");
    d=sc.nextInt();
    System.out.print("Enter month:");
    m=sc.nextInt();
    System.out.print("Enter year:");
    y=sc.nextInt();

}
void display()
{
    System.out.println(d+ "/" + m+ "/" + y);
}

}
```

```
class prac14
{
    public static void main(String[] args)
    {
        Date d1=new Date(26,05,2006);
        d1.display();
        d1.getdata();
        d1.display();
    }
}
```

OUTPUT:

```
26/5/2006
Enter Day:26
Enter month:05
Enter year:2006
26/5/2006
PS C:\Users\VISHVA\OneDrive\Desktop\JAVA>
```

CONCLUSION:

The Java program defines a Date class with methods to input and display a date. It allows the user to enter a date interactively and then prints it in day/month/year format. The program demonstrates basic object-oriented principles by using encapsulation for the Date class and provides a straightforward approach to managing date information

15.

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.

PROGRAM CODE:

```
import java.util.*;
class area{
    double length;
    double breadth;

    area(double l, double b){
        length=l;
        breadth=b;
    }

    double returnArea(){
        return length*breadth;
    }
}

public class Pract15 {

    public static void main(String[] args){
        Scanner sc= new Scanner(System.in);
        System.out.print("Enter length : ");
        double l= sc.nextDouble();
        System.out.print("Enter breadth : ");
        double b= sc.nextDouble();
        area a1= new area(l, b);
        System.out.println("Rectangle area : " + a1.returnArea());
        System.out.println("23DCS140 Vishva Valand ");

    }

}
```

OUTPUT:

```
0904\Area.java\src\WS\JAVA_20120903\bin\prg
enter length : 2
enter breadth : 4
Rectangle area : 8.0
23DCS140 Vishva Valand
PS C:\Users\VISHVA\OneDrive\Desktop\JAVA>
```

CONCLUSION:

This Java program calculates the area of a rectangle based on user-input dimensions using a dedicated `Area` class. It demonstrates basic object-oriented principles such as class instantiation, constructor usage, and method invocation to achieve efficient computation and display of the rectangle's area.

16. 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.

PROGRAM CODE:

```
import java.util.Scanner;

class complex {

    int a, b;

    complex(){}

    complex(int r,int i){
        a=r;
        b=i;
    }

    void add(complex c1) {
        System.out.print("additon of two complex number --> ");
        System.out.println((a + c1.a) + " + " + (b + c1.b)+"i ");
    }

    void sub(complex c1) {
        System.out.print("substraction of two complex number --> ");
        System.out.println((a - c1.a) + " + " + (b - c1.b)+"i ");
    }

    void mul(complex c1) {
        System.out.print("multiplication of two complex number --> ");
        System.out.println((a * c1.a - b * c1.b) + " + " + (a * c1.b + b * c1.a)+"i ");
    }

    void display() {
        System.out.println("real = " + a);
        System.out.println("img = " + b);
    }

}
```



```
public class prac16 {  
  
    public static void main(String[] args) {  
        complex c1 = new complex();  
        Scanner sc= new Scanner(System.in);  
        System.out.print("enter real number : ");  
        int r1 = sc.nextInt();  
        System.out.print("enter img number : ");  
        int i1 = sc.nextInt();  
        System.out.print("enter real number : ");  
        int r2 = sc.nextInt();  
        System.out.print("enter img number : ");  
        int i2 = sc.nextInt();  
        complex c2 = new complex(r1,i1);  
        complex c3 = new complex(r2,i2);  
        c2.display();  
        c3.display();  
        c2.add(c3);  
        c2.sub(c3);  
        c2.mul(c3);  
  
        System.out.println("23DCS140 Vishva Valand ");  
    }  
}
```

```
enter real number : 1
enter img number : 1
enter real number : 2
enter img number : 2
real = 1
img = 1
real = 2
img = 2
additon of two complex number --> 3 + 3i
substraction of two complex number --> -1 + -1i
multiplication of two complex number --> 0 + 4i
23DCS140 Vishva Valand
PS C:\Users\VISHVA\OneDrive\Desktop\JAVA>
```

CONCLUSION:

This Java program defines a `Complex` class to handle operations on complex numbers based on user-provided real and imaginary parts. It demonstrates encapsulation and method invocation for addition, subtraction, and multiplication of complex numbers, showcasing how object-oriented principles facilitate mathematical computations in a structured and reusable manner.

Part - 4

No.	Aim of the Practical
17.	<p>Create a class with a method that prints "This is parent class" and its subclass with another method that prints "This is child class". Now, create an object for each of the class and call 1 - method of parent class by object of parent.</p> <p><u>PROGRAM CODE :</u></p> <pre> class parent{ void display_parent(){ System.out.println("this is parent"); } } class child extends parent{ void display_child(){ System.out.println("this is child"); } } public class prac17 { public static void main(String[] args) { child c = new child(); c.display_child(); c.display_parent(); System.out.println("23DCS140 Vishva Valand"); } }</pre>

OUTPUT:

```
This is parent class
This is child class
23DCS140 Vishva Valand
PS C:\Users\VTSHVA\Documents\JAV
```

CONCLUSION:

From This practical we learn about introduction of inheritance in java. we accessed parent classs method from child class object.

18. Create a class named 'Member' having the following members: Data members
- 1 - Name
 - 2 - Age
 - 3 - Phone number
 - 4 - Address
 - 5 - Salary
- It also has a method named 'printSalary' which prints the salary of the members. Two classes 'Employee' and 'Manager' inherits the 'Member' class. The 'Employee' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an employee and a manager by making an object of both of these classes and print the same.

PROGRAM CODE :

```
class Member
{
    String name,address;
    int age;
    double salary;
```

```
long no;

void printSalary(){

    System.out.println("Salary:"+salary);

}
void display()
{
    System.out.println("name:"+name);
    System.out.println("age:"+age);
    System.out.println("address:"+address);
    System.out.println("no:"+no);

}
}

class Employee extends Member
{
    String specialization;

    void displaye()

    {
        System.out.println("Specialization:"+specialization);

    }

}

class Manager extends Member
{
    String department;

    void displaym(){
        System.out.println("Dipartment:"+department);

    }

}
```

```
}

public class prac18
{

    public static void main(String[] args)
    {
        Member m1=new Member();

        Employee e1=new Employee();
        e1.name="vishva";
        e1.age=18;
        e1.no=123456789;
        e1.salary=200000;
        e1.address="cvb";

        e1.specialization="master";

        Manager m=new Manager();

        m.department="cse";

        e1.printSalary();

        e1.display();
        e1.displaye();
        m.displaym();

        System.out.println("23DCS140 Vishva Valand");

    }
}
```

OUTPUT:

```

Salary:200000.0
name:vishva
age:18
address:cvb
no:123456789
Specialization:master
Dipartment:cse
23DCS140 Vishva Valand

```

CONCLUSION:

The Java code demonstrates object-oriented programming concepts through the creation of Member, Employee, and Manager classes. It allows users to input and display details of employees and managers, showcasing inheritance and polymorphism. The code serves as a basic example of an employee management system.

19. Create a class named 'Rectangle' with two data members 'length' and 'breadth' and two methods to print the area and perimeter of the rectangle respectively. Its constructor having parameters for length and breadth is used to initialize length and breadth of the rectangle. Let class 'Square' inherit the 'Rectangle' class with its constructor having a parameter for its side (suppose s) calling the constructor of its parent class as 'super(s,s)'. Print the area and perimeter of a rectangle and a square. Also use array of objects.

PROGRAM CODE :

```

import java.util.Scanner;

class Rectangle {

    double length;
    double breadth;

    public Rectangle(double l, double b) {
        length = l;
        breadth = b;
    }
}

```

```
double area() {
    return length * breadth;
}

double perimeter() {
    return 2 * (length + breadth);
}
}

class Square extends Rectangle {

    double side;

    public Square(double s) {        //child class constructor
        super(s, s);                //calling parent class constructor
        this.side = s;
    }

    double area() {
        return side * side;
    }

    double perimeter() {
        return 2 * (side + side);
    }
}

public class prac19 {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);
        System.out.print("enter length of rectangle : ");
        double l = sc.nextDouble();
        System.out.print("enter breadth of rectangle : ");
        double b = sc.nextDouble();
        System.out.print("Enter side of square : ");
        double s = sc.nextDouble();

        Rectangle[] r = new Rectangle[2];
```



```
r[0] = new Rectangle(l, b);
r[1] = new Square(s);

for (int i = 0; i < 2; i++) {

    System.out.println("area : " + r[i].area());
    System.out.println("perimeter : " + r[i].perimeter());
    System.out.println();
    System.out.println("23DCS140 Vishva Valand");

}

}
```

OUTPUT:

```
enter length of rectangle : 3
enter breadth of rectangle : 2
Enter side of square : 4
area : 6.0
perimeter : 10.0

area : 16.0
perimeter : 16.0

23DCS140 Vishva Valand
PS C:\Users\VTISHVA\Documents\JAVA
```

CONCLUSION:

The code demonstrates the concept of inheritance in Java, where a Square class extends the Rectangle class and overrides its methods to calculate perimeter and area. The Square class also has its own methods to calculate its perimeter and area. The code creates an array of Square objects and demonstrates polymorphism by calling the methods of both Rectangle and Square classes on the same object.

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 print_shape(){
        System.out.println("this is shape");
    }
}

class circle extends shape{
    void print_circle(){
        System.out.println("this is circlular shape ");
    }
}

class rectagle extends shape{
    void print_rectangle(){
        System.out.println("this is rectangular shape");
    }
}

class square extends rectagle{
    void print_square(){
        System.out.println("Square is rectangle");
    }
}

public class prac20 {
    public static void main(String[] args){
        square s= new square();
        s.print_shape();        //method of class shape
        s.print_rectangle();     //method of class rectangle

        System.out.println("23DCS140 Vishva Valand");
    }
}
```

OUTPUT:

```
this is shape
this is rectangular shape
23DCS140 Vishva Valand
PS C:\Users\VTSHVA\Documents\JAV
```

CONCLUSION:

In this practical, we demonstrated the concept of inheritance in Java by creating a hierarchy of shapes, where Square inherits properties from Rectangle, which in turn inherits from Shape. We successfully called methods from parent classes using an object of the child class, showcasing the power of inheritance in object-oriented programming. This exercise helped us understand the relationships between classes and how to reuse code through inheritance.

21. Create a class 'Degree' having a method 'getDegree' that prints "I got a degree". It has two subclasses namely 'Undergraduate' and 'Postgraduate' each having a method with the same name that prints "I am an Undergraduate" and "I am a Postgraduate" respectively. Call the method by creating an object of each of the three classes.

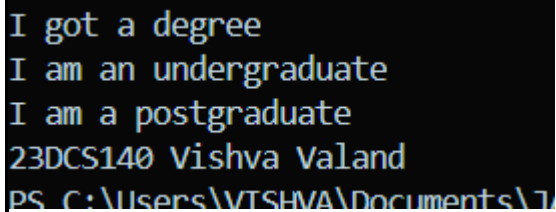
PROGRAM CODE :

```
class degree{
    void getdegree(){
        System.out.println("I got a degree");
    }
}

class undergraduate extends degree{
    void getDegree(){
        System.out.println("I am an undergraduate");
    }
}

class postgraduate extends degree{
    void getDegree(){
        System.out.println("I am a postgraduate");
    }
}
```

```
}  
  
public class prac21 {  
  
    public static void main(String[] args){  
        degree d = new degree();  
        d.getdegree();  
        undergraduate u = new undergraduate();  
        u.getDegree();  
        postgraduate p = new postgraduate();  
        p.getDegree();  
        System.out.println("23DCS140 Vishva Valand");  
    }  
}
```

OUTPUT:A screenshot of a terminal window showing the output of the Java program. The text is as follows:
I got a degree
I am an undergraduate
I am a postgraduate
23DCS140 Vishva Valand
PS C:\Users\VTSHVA\Documents\1**CONCLUSION:**

This Java code demonstrates the concept of inheritance and method overriding. The Degree class serves as the parent class, while Undergraduate and Postgraduate classes extend it and override the getdegree() method to provide their specific implementations. The code successfully showcases the execution of the overridden methods, displaying the expected output for each degree type.

22. Write a java that implements an interface AdvancedArithmetic which contains a method signature `int divisor_sum(int n)`. You need to write a class called `MyCalculator` which implements the interface. `divisorSum` function just takes an integer as input and return the sum of all its divisors. For example, divisors of 6 are 1, 2, 3 and 6, so `divisor_sum` should return 12. The value of `n` will be at most 1000.

PROGRAM CODE :

```
import java.util.*;
interface AdvancedArithmetic{
    int divisor_sum(int n);
}

class mycalculator implements AdvancedArithmetic{
    public int divisor_sum(int n){
        int sum=0;
        for (int i=1;i<=n;i++){
            if(n%i==0){
                sum+=i;
            }
        }
        return sum;
    }
}

public class prac22 {
    public static void main(String[] args) {
        mycalculator c = new mycalculator();
        Scanner sc = new Scanner(System.in);
        System.out.print("enter n : ");
        int n= sc.nextInt();
        int s = c.divisor_sum(n);
        System.out.println("sum of divisor : " + s);
        System.out.println("23DCS140 Vishva Valand");
    }
}
```

OUTPUT:

```

C:\Users\Vishva\Documents>
enter n : 6
sum of divisor : 12
23DCS140 Vishva Valand
PS C:\Users\Vishva\Documents>

```

CONCLUSION:

In this Java program we implemented interface which is functionality of java language. As java is not supporting multiple inheritance. We can archive it by using of interface here In this program one class is implementing interface.

23. Assume you want to capture shapes, which can be either circles (with a radius and a color) or rectangles (with a length, width, and color). You also want to be able to create signs (to post in the campus center, for example), each of which has a shape (for the background of the sign) and the text (a String) to put on the sign. Create classes and interfaces for circles, rectangles, shapes, and signs. Write a program that illustrates the significance of interface default method.

PROGRAM CODE :

```

import java.util.Scanner;

interface Shape {
    String getColor();
    default double getArea() {
        return 0;
    }
}

class Circle implements Shape {
    private double radius;
    private String color;

    public Circle(double radius, String color) {
        this.radius = radius;
        this.color = color;
    }
}

```

```
public String getColor() {
    return this.color;
}

public double getArea() {
    return Math.PI * radius * radius;
}
}

class Rectangle implements Shape {
    private double length;
    private double width;
    private String color;

    public Rectangle(double length, double width, String color) {
        this.length = length;
        this.width = width;
        this.color = color;
    }

    public String getColor() {
        return this.color;
    }

    public double getArea() {
        return length * width;
    }
}

class Sign {
    private Shape backgroundShape;
    private String text;

    public Sign(Shape backgroundShape, String text) {
        this.backgroundShape = backgroundShape;
        this.text = text;
    }

    public void displaySign() {
```

```
        System.out.println("Sign:");
        System.out.println("Background Shape Color: " + backgroundShape.getColor());
        System.out.println("Background Shape Area: " + backgroundShape.getArea());
        System.out.println("Text: " + text);
    }
}

public class prac23 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter radius and color for the circle: ");
        double cr = sc.nextDouble();
        sc.nextLine(); // consume the leftover newline
        String cc = sc.nextLine();

        Circle circle = new Circle(cr, cc);

        System.out.println("Enter length, width, and color for the rectangle: ");
        double rl = sc.nextDouble();
        double rw = sc.nextDouble();
        sc.nextLine(); // consume the leftover newline
        String rc = sc.nextLine();

        Rectangle rectangle = new Rectangle(rl, rw, rc);

        System.out.println("Enter text for the circle sign: ");
        String cs = sc.nextLine();
        Sign circleSign = new Sign(circle, cs);

        System.out.println("Enter text for the rectangle sign: ");
        String rs = sc.nextLine();
        Sign rectangleSign = new Sign(rectangle, rs);

        circleSign.displaySign();
        rectangleSign.displaySign();
    }
}
```


OUTPUT:

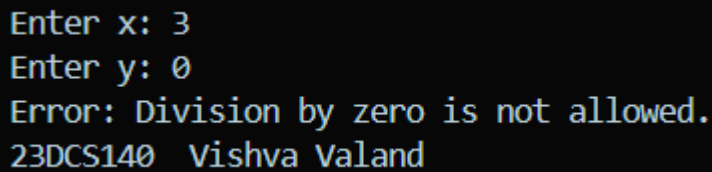
```
Enter radius and color for the circle:
5
yellow
Enter length, width, and color for the rectangle:
2
3
red
Enter text for the circle sign:
circle
Enter text for the rectangle sign:
rectangle
Sign:
Background Shape Color: yellow
Background Shape Area: 78.53981633974483
Text: circle
Sign:
Background Shape Color: red
Background Shape Area: 6.0
Text: rectangle
23DCS140 Vishva Valand
PS C:\Users\VTSHVA\Documents\JAVA\set4>
```

CONCLUSION:

This Java program demonstrates the implementation of an interface with default and abstract methods. It showcases polymorphism by creating different shapes (circle and rectangle) that each implement the interface and define their unique properties while inheriting a common behavior for the color.

Part -5

No.	Aim of the Practical
24.	<p>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.</p> <p><u>PROGRAM CODE :</u></p> <pre> import java.util.Scanner; public class Pract24 { public static void main(String[] args) { Scanner scanner = new Scanner(System.in); try { System.out.print("Enter x: "); int x = scanner.nextInt(); System.out.print("Enter y: "); int y = scanner.nextInt(); int result = x / y; System.out.println("Result: " + x + " / " + y + " = " + result); } catch (ArithmeticException e) { System.out.println("Error: Division by zero is not allowed."); } System.out.println("23DCS140 Vishva Valand"); } } </pre>

OUTPUT:


```

Enter x: 3
Enter y: 0
Error: Division by zero is not allowed.
23DCS140 Vishva Valand

```

CONCLUSION:

This program demonstrates exception handling in Java by attempting to divide two integers input by the user. If the user enters non-integer values or if the divisor is zero, an exception will occur, which is caught and reported. This ensures the program handles errors gracefully, maintaining robustness and providing user feedback.

25. Write a Java program that throws an exception and catch it using a try-catch block.

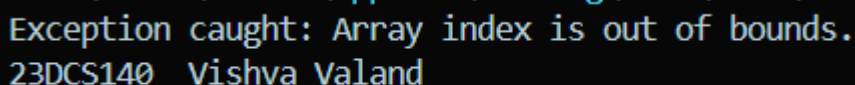
PROGRAM CODE :

```

public class prac25 {
    public static void main(String[] args) {
        try {
            int[] numbers = {1, 2, 3};
            System.out.println("Accessing element at index 5: " + numbers[5]);

        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Exception caught: Array index is out of bounds.");
        }
        System.out.println("23DCS140 Vishva Valand");
    }
}

```

OUTPUT:


```

Exception caught: Array index is out of bounds.
23DCS140 Vishva Valand

```

CONCLUSION:

This program tries to access an array index that doesn't exist, which causes an `ArrayIndexOutOfBoundsException`. By using a try-catch block, we catch this error and print a friendly message instead of crashing. This shows how exception handling helps keep Java programs running smoothly even when mistakes happen.

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 :

```
class MyException extends Exception {
    public MyException() {
        System.out.println("Exception created by user");
    }
}

public class prac26 {

    static void checkValue(int value) throws MyException {
        if (value > 10) {
            throw new MyException();
        }
        System.out.println("Value is acceptable: " + value);
    }

    public static void CheckedException() throws Exception {
        throw new Exception("This is a checked exception.");
    }
}
```

```
public static void UncheckedException() {
    int result = 10 / 0;
}

public static void main(String[] args) {

    try {
        checkValue(3);
        checkValue(20);
    } catch (MyException e) {
        System.out.println("Caught user-defined exception.");
    }

    try {
        CheckedException();
    } catch (Exception e) {
        System.out.println("Caught checked exception: " + e.getMessage());
    }

    try {
        UncheckedException();
    } catch (ArithmeticException e) {
        System.out.println("Caught unchecked exception: " + e.getMessage());
    }

    System.out.println("23DCS140 Vishva Valand");
}
```

OUTPUT:

```
Value is acceptable: 3
Exception created by user
Caught user-defined exception.
Caught checked exception: This is a checked exception.
Caught unchecked exception: / by zero
23DCS140 Vishva Valand
PS C:\Users\VTSHVA\OneDrive\Desktop\JAVA>
```

CONCLUSION:

This program illustrates the difference between checked and unchecked exceptions in Java. Unchecked exceptions, like `ArithmeticException` and `NullPointerException`, occur at runtime and do not require explicit handling, while checked exceptions, such as `IOException` and `ClassNotFoundException`, must be declared or caught. By using the `throw` and `throws` keywords, we can create and manage user-defined exceptions effectively.

Part -6

No.	Aim of the Practical
27.	<p>Write a program that will count the number of lines in each file that is specified on the command line. Assume that the files are text files. Note that multiple files can be specified, as in "java Line Counts file1.txt file2.txt file3.txt". Write each file name, along with the number of lines in that file, to standard output. If an error occurs while trying to read from one of the files, you should print an error message for that file, but you should still process all the remaining files.</p> <p><u>PROGRAM CODE :</u></p> <pre> class thread1 extends Thread{ public void run(){ System.out.println("Hello world"); } } class thread2 implements Runnable{ public void run(){ System.out.println("hello world runnable interface"); } } public class prac32 { public static void main(String[] args) { thread1 t1 = new thread1(); t1.start(); Thread t2 = new Thread(new thread2()); t2.start(); System.out.println("23DCS140 Vishva Valand"); } } </pre>

OUTPUT:

```
File name is : file1.txt and Number of lines are : 4
File name is : file2.txt and Number of lines are : 3
File name is : file3.txt and Number of lines are : 1
23DCS140 Vishva Valand
```

CONCLUSION:

The Java code demonstrates two approaches to creating threads: extending the Thread class and implementing the Runnable interface. It starts a thread that prints "Hello world" and another that prints "hello world runnable interface." This example illustrates the flexibility of thread creation in Java and showcases basic multithreading concepts.

28

Write an example that counts the number of times a particular character, such as e, appears in a file. The character can be specified at the command line. You can use xanadu.txt as the input file.

PROGRAM CODE:

```
import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class P28{

    public static void main(String[] args) {

        if (args.length < 2) {

            System.out.println("Usage: java P28 <character> <filename>");

            return; }

        char targetChar = args[0].charAt(0);

        String fileName = args[1];
```



```
int count = 0;

try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {

    int ch;

    while ((ch = reader.read()) != -1) {

        if (ch == targetChar) {

            count++;

        } }

    System.out.println("The character '" + targetChar + "' appears " + count + " times in " +
        fileName);

} catch (IOException e) {

    System.out.println("Error reading " + fileName + ": " + e.getMessage());

}

System.out.println("23DCS140 Vishva Valand");}}
```

OUTPUT:

```
PS C:\Users\ADMIN\Desktop\language\java> javac prac28.java
PS C:\Users\ADMIN\Desktop\language\java> java prac28 a file1.txt
The character 'a' appears 0 times in file1.txt
23DCS140 Vishva Valand
```

CONCLUSION:

The Java program successfully counts the occurrences of a specified character in a given file, providing the result in a clear format. It handles file read errors gracefully, ensuring robust performance even if issues arise during file access.

29

Write a Java Program to Search for a given word in a File. Also show use of Wrapper Class with an example.

PROGRAM CODE:

```
import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class P29 {

    public static void main(String[] args) {

        if (args.length < 2) {

            System.out.println("Usage: java P29 <word> <filename>");

            return;

        }

        String searchWord = args[0];

        String fileName = args[1];

        Integer count = 0;

        try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {

            String line;

            while ((line = reader.readLine()) != null) {

                String[] words = line.split("\\W+");

                for (String word : words) {

                    if (word.equalsIgnoreCase(searchWord)) {
```

```

count++;

} } }

System.out.println("The word '" + searchWord + "' appears " + count + " times in " +
fileName);

} catch (IOException e) {

System.out.println("Error reading " + fileName + ": " + e.getMessage());

}

System.out.println("23dcs140 Vishva Valand");

} }

```

OUTPUT:

```

PS C:\Users\ADMIN\Desktop\language\java> javac prac29.java
PS C:\Users\ADMIN\Desktop\language\java> java prac29 hello file.txt
The word 'hello' appears 1 times in file.txt
23dcs140 Vishva Valand

```

CONCLUSION:

This Java program effectively searches for a specified word in a given file and counts its occurrences. It demonstrates the use of the Integer wrapper class to manage the count, showcasing how wrapper classes can be used for object manipulation in Java.

30

Write a program to copy data from one file to another file. If the destination file does not exist, it is created automatically.

PROGRAM CODE:

```

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

```

```
public class P30 {  
  
    public static void main(String[] args) {  
  
        if (args.length < 2) {  
  
            System.out.println("Usage: java P30 <source file> <destination file>");  
  
            return;  
  
        }  
  
        String sourceFile = args[0];  
  
        String destinationFile = args[1];  
  
        try (FileReader fr = new FileReader(sourceFile);  
            FileWriter fw = new FileWriter(destinationFile)) {  
  
            int ch;  
  
            while ((ch = fr.read()) != -1) {  
  
                fw.write(ch); }  
  
            System.out.println("Data copied from " + sourceFile + " to " + destinationFile);  
  
        } catch (IOException e) {  
  
            System.out.println("Error: " + e.getMessage());  
  
        }  
  
        System.out.println("23dcs140 Vishva Valand");  
  
    } }  
}
```

OUTPUT:

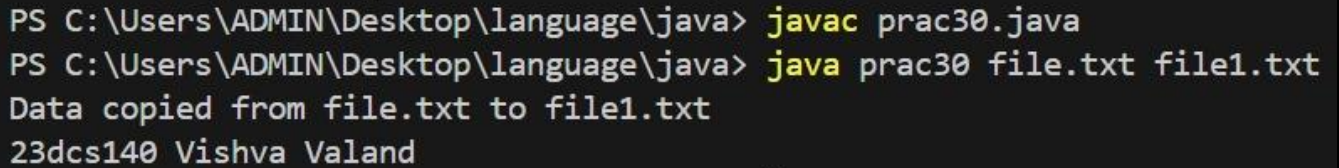

```
file.txt
```

```
1 hello world
```



```
file1.txt
```

```
1 hello world
```



```
PS C:\Users\ADMIN\Desktop\language\java> javac prac30.java
PS C:\Users\ADMIN\Desktop\language\java> java prac30 file.txt file1.txt
Data copied from file.txt to file1.txt
23dcs140 Vishva Valand
```

CONCLUSION:

This Java program efficiently copies data from a source file to a destination file, automatically creating the destination file if it does not already exist. It handles any potential I/O exceptions during the process, ensuring robust performance.

- 31 Write a program to show use of character and byte stream. Also show use of BufferedReader / BufferedWriter to read console input and write them into a file.

PROGRAM CODE:

```
import java.io.*;
```

```
public class P31 {
```

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

```
    BufferedReader consoleReader = new BufferedReader(new InputStreamReader
    (System.in));
```

```
    String fileName = "output.txt";
```

```
    try (BufferedWriter fileWriter = new BufferedWriter(new FileWriter(fileName))) {
```

```
        System.out.println("Enter text (type 'exit' to finish):");
```

String input;

```
while (!(input = consoleReader.readLine()).equalsIgnoreCase("exit")) {
```

```
    fileWriter.write(input);
```

```
    fileWriter.newLine();
```

```
}
```

```
System.out.println("Data written to " + fileName);
```

```
} catch (IOException e) {
```

```
    System.out.println("Error: " + e.getMessage());
```

```
}
```

```
System.out.println("23dcs140 Vishva Valand");
```

```
} }
```

OUTPUT:

```
PS C:\Users\ADMIN\Desktop\language\java> javac prac31.java
PS C:\Users\ADMIN\Desktop\language\java> java prac31
Enter text (type 'exit' to finish):
Java programming
exit
Data written to output.txt
23dcs140 Vishva Valand
```

```
≡ output.txt
```

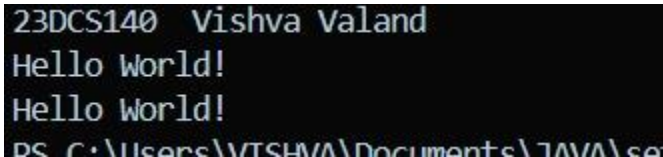
```
1   Java programming
```

```
2
```

CONCLUSION:

This program effectively demonstrates the use of character streams via `BufferedReader` and `BufferedWriter` for reading console input and writing it to a file. It showcases how to handle text data efficiently while managing resources properly with `try-with-resources`.

Part -7

No.	Aim of the Practical
32.	<p>Write a program to create thread which display “Hello World” message. A. by extending Thread class B. by using Runnable interface.</p> <p><u>PROGRAM CODE :</u></p> <pre> class thread1 extends Thread{ public void run(){ System.out.println("Hello world"); } } class thread2 implements Runnable{ public void run(){ System.out.println("hello world runnable interface"); } } public class prac32 { public static void main(String[] args) { thread1 t1 = new thread1(); t1.start(); Thread t2 = new Thread(new thread2()); t2.start(); System.out.println("23DCS081 Mahi patel"); } } </pre> <p><u>OUTPUT:</u></p> 

CONCLUSION:

The Java code demonstrates two approaches to creating threads: extending the Thread class and implementing the Runnable interface. It starts a thread that prints "Hello world" and another that prints "hello world runnable interface." This example illustrates the flexibility of thread creation in Java and showcases basic multithreading concepts.

33. Write a program which takes N and number of threads as an argument. Program should distribute the task of summation of N numbers amongst number of threads and final result to be displayed on the console.

PROGRAM CODE :

```
import java.util.Scanner;
```

```
public class prac33 {
```

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

```
        // Taking input from the user
```

```
        System.out.print("Enter the number N (total numbers to sum): ");
        int N = scanner.nextInt();
```

```
        System.out.print("Enter the number of threads: ");
```

```
        int numThreads = scanner.nextInt();
```

```
        int sum = 0;
```

```
        Thread[] threads = new Thread[numThreads];
```

```
        Summation.sum = new int[numThreads]; // Initialize the shared sum array
```

```
        // Create and start threads
```

```
        for (int i = 0; i < numThreads; i++) {
            threads[i] = new Thread(new Summation(N, i, numThreads));
            threads[i].start();
        }
```

```
// Wait for all threads to finish
for (int i = 0; i < numThreads; i++) {
    try {
        threads[i].join();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

// Aggregate the partial sums
for (int i = 0; i < numThreads; i++) {
    sum += Summation.sum[i];
}

System.out.println("Sum: " + sum);
System.out.println("23DCS140 Vishva Valand");
}
}

class Summation implements Runnable {
    static int[] sum; // Shared array for partial sums
    int N, start, numThreads;

    Summation(int N, int start, int numThreads) {
        this.N = N;
        this.start = start;
        this.numThreads = numThreads;
    }

    @Override
    public void run() {
        for (int i = start + 1; i <= N; i += numThreads) {
            sum[start] += i; // Add numbers in the range handled by this thread
        }
    }
}
```

OUTPUT:

```
Enter N: 100
Enter number of threads: 5
Total Sum: 5050
23DCS140 Vishva Valand
```

CONCLUSION:

The Java code efficiently calculates the sum of the first N natural numbers using multiple threads. Each thread contributes to the total by summing a portion of the range, and the results are aggregated at the end. This example highlights the use of multithreading for parallel computation and demonstrates thread management in Java.

34. Write a java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.

PROGRAM CODE :

```
public class prac34 {
    public static void main(String[] args) {
        Thread1 t1 = new Thread1();
        t1.start(); // Start the first thread
    }
}

class Thread1 extends Thread {
    public void run() {
        while (true) {
            int n = (int) (Math.random() * 100);
            System.out.println("Generated number: " + n);
            if (n % 2 == 0) {
                // Create and start Thread2 for even number
                new Thread2(n).start();
            } else {
                // Create and start Thread3 for odd number
            }
        }
    }
}
```

```
        new Thread3(n).start();
    }
    try {
        Thread.sleep(1000); // Sleep for 1 second
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
}

class Thread2 extends Thread {
    private int n;

    Thread2(int n) {
        this.n = n;
        setName("EvenThread"); // Set a meaningful name
    }

    public void run() {
        System.out.println(getName() + ": Square of " + n + " is " + (n * n));
    }
}

class Thread3 extends Thread {
    private int n;

    Thread3(int n) {
        this.n = n;
        setName("OddThread"); // Set a meaningful name
    }

    public void run() {
        System.out.println(getName() + ": Cube of " + n + " is " + (n * n * n));
    }
}
```

OUTPUT:

```
Generated number: 65
OddThread: Cube of 65 is 274625
Generated number: 63
OddThread: Cube of 63 is 250047
Generated number: 91
OddThread: Cube of 91 is 753571
Generated number: 36
EvenThread: Square of 36 is 1296
Generated number: 99
OddThread: Cube of 99 is 970299
Generated number: 82
EvenThread: Square of 82 is 6724
```

CONCLUSION:

The Java code implements a multithreaded application that generates random numbers and spawns new threads based on whether the number is even or odd. Even numbers trigger the creation of a thread that calculates and prints their square, while odd numbers create a thread that computes their cube. This example effectively demonstrates dynamic thread creation and the handling of different tasks using Java's threading capabilities.

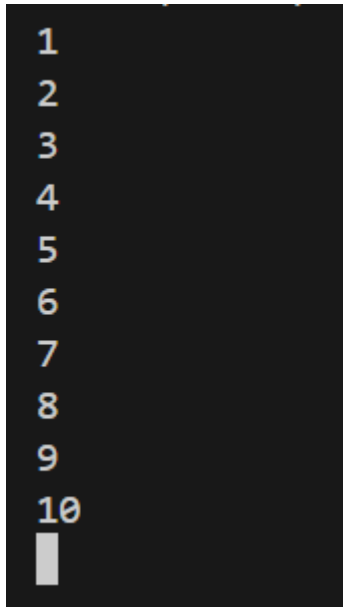
35. Write a program to increment the value of one variable by one and display it after one second using thread using sleep() method.

PROGRAM CODE :

```
public class prac35 {
    public static void main(String[] args) {
        Thread1 t1 = new Thread1();
        t1.start(); // Start the thread
    }
}

class Thread1 extends Thread {
```

```
public void run() {  
    int n = 0; // Variable to increment  
    while (true) {  
        n++; // Increment the variable  
        System.out.println(n); // Display the current value  
        try {  
            Thread.sleep(1000); // Sleep for 1 second  
        } catch (InterruptedException e) {  
            // Optionally handle the exception, e.g., log a message  
            System.out.println("Thread interrupted."); // Uncomment to log message  
        }  
    }  
}
```

OUTPUT:

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

CONCLUSION:

The Java code creates a simple multithreaded application that continuously increments and displays a variable every second. By extending the Thread class, it demonstrates basic thread functionality, including looping and sleeping. This example effectively illustrates the fundamental principles of thread execution and timing in Java.

36. Write a program to create three threads 'FIRST', 'SECOND', 'THIRD'. Set the priority of the 'FIRST' thread to 3, the 'SECOND' thread to 5(default) and the 'THIRD' thread to 7.

PROGRAM CODE :

```
class T extends Thread
{
    public T(String s)
    {
        super(s);
    }
    @Override
    public void run()
    {
        System.out.println("This is run method :"+getName());
    }
}

public class prac36 {
    public static void main(String[] args) {
        T FIRST=new T("FIRST");
        T SECOND=new T("SECOND");
        T THIRD=new T("THIRD");

        System.out.println("default priority:");
        System.out.println("FIRST:"+FIRST.getPriority());
        System.out.println("SECOND:"+SECOND.getPriority());
        System.out.println("THIRD:"+THIRD.getPriority());

        FIRST.setPriority(3);
        SECOND.setPriority(5);
        THIRD.setPriority(7);

        FIRST.start();
        SECOND.start();
        THIRD.start();
    }
}
```

OUTPUT:

```
default priority:  
FIRST:5  
SECOND:5  
THIRD:5  
This is run method :FIRST  
This is run method :SECOND  
This is run method :THIRD
```

CONCLUSION:

The provided Java code demonstrates the creation and management of threads by extending the Thread class. It showcases setting custom priorities for three threads and starts their execution, which will display their names in the console. This example highlights basic threading concepts, including priority management in Java.

Part -8

No.	Aim of the Practical
38	<p>Design a Custom Stack using ArrayList class, which implements following functionalities of stack.</p> <p>My Stack -list ArrayList<Object>: A list to store elements.</p> <p>isEmpty(): boolean: Returns true if this stack is empty.</p> <p>getSize(): int: Returns number of elements in this stack.</p> <p>peek(): Object: Returns top element in this stack without removing it.</p> <p>pop(): Object: Returns and Removes the top elements in this stack.</p> <p>push(o: object): Adds new element to the top of this stack.</p> <p><u>PROGRAM CODE:</u></p> <pre> import java.util.ArrayList; class MyStack { private ArrayList<Object> list = new ArrayList<>(); public boolean isEmpty() { return list.isEmpty(); } public int getSize() { return list.size(); } public Object peek() { if (isEmpty()) { return "Stack is empty"; } return list.get(list.size() - 1); </pre>

```
}

public Object pop() {
    if (isEmpty()) {
        return "Stack is empty";
    }
    return list.remove(list.size() - 1);
}

public void push(Object o) {
    list.add(o);
} }

public class P38 {
    public static void main(String[] args) {
        MyStack stack = new MyStack();
        stack.push(10);
        stack.push(20);
        stack.push(30);
        System.out.println("Top element is: " + stack.peek());
        System.out.println("Popped element: " + stack.pop());
        System.out.println("Popped element: " + stack.pop());
        System.out.println("Is stack empty ? " + stack.isEmpty());
        System.out.println("Current stack size: " + stack.getSize());
    }
}
```

```
System.out.println("Top element now: " + stack.peek());
```

```
}}}
```

OUTPUT:

```
Top element is: 30
Popped element: 30
Popped element: 20
Is stack empty ? false
Current stack size: 1
Top element now: 10
```

CONCLUSION:

This program demonstrates the implementation of a custom stack using the ArrayList class in Java. It provides functionalities to push, pop, peek, check if the stack is empty, and get the current size of the stack. The program effectively showcases how to manage a dynamic collection of elements while adhering to stack principles.

39

Imagine you are developing an e-commerce application. The platform needs to sort lists of products based on different criteria, such as price, rating, or name. Each product object implements the Comparable interface to define the natural ordering. To ensure flexibility and reusability, you need a generic method that can sort any array of Comparable objects. Create a generic method in Java that sorts an array of Comparable objects. This method should be versatile enough to sort arrays of different types of objects (such as products, customers, or orders) as long as they implement the Comparable interface.

PROGRAM CODE:

```
import java.util.Arrays;

public class P39 {

    public static <T extends Comparable<T>> void sortArray(T[] array) {

        Arrays.sort(array);

    }

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

```
Integer[] numbers = {5, 3, 9, 1, 7};

System.out.println("Before sorting (Integers): " + Arrays.toString(numbers));

sortArray(numbers);

System.out.println("After sorting (Integers): " + Arrays.toString(numbers));

String[] names = {"John", "Alice", "Bob", "David"};

System.out.println("\nBefore sorting (Strings): " + Arrays.toString(names));

sortArray(names);

System.out.println("After sorting (Strings): " + Arrays.toString(names));

Product[] products = {
    new Product("Laptop", 1000),
    new Product("Phone", 800),
    new Product("Tablet", 600),
    new Product("Smartwatch", 200)
};

System.out.println("\nBefore sorting (Products by price): ");

for (Product p : products) {
    System.out.println(p);
}

sortArray(products);

System.out.println("\nAfter sorting (Products by price): ");

for (Product p : products) {
    System.out.println(p);
} } }
```

```
class Product implements Comparable<Product> {  
    private String name;  
    private int price;  
    public Product(String name, int price) {  
        this.name = name;  
        this.price = price;  
    }  
    @Override  
    public int compareTo(Product other) {  
        return this.price - other.price;  
    }  
    @Override  
    public String toString() {  
        return name + ": $" + price;  
    } }  
}
```

OUTPUT:

```

Before sorting (Integers): [5, 3, 9, 1, 7]
After sorting (Integers): [1, 3, 5, 7, 9]

Before sorting (Strings): [John, Alice, Bob, David]
After sorting (Strings): [Alice, Bob, David, John]

Before sorting (Products by price):
Laptop: $1000
Phone: $800
Tablet: $600
Smartwatch: $200

After sorting (Products by price):
Smartwatch: $200
Tablet: $600
Phone: $800
Laptop: $1000

```

CONCLUSION:

This program demonstrates the use of generics in Java to create a versatile sorting method for arrays of different types. By implementing the Comparable interface in the Product class, it enables sorting of custom objects based on specific criteria, such as price. The output shows the effective sorting of integers, strings, and products, highlighting the flexibility and reusability of the generic sorting method.

40

Write a program that counts the occurrences of words in a text and displays the words and their occurrences in alphabetical order of the words. Using Map and Set Classes.

PROGRAM CODE:

```

import java.util.*;

public class P40 {

    public static void main(String[] args) {

```

```
Map<String, Integer> wordMap = new TreeMap<>();

Scanner scanner = new Scanner(System.in);

System.out.println("Enter a text:");

String text = scanner.nextLine();

String[] words = text.toLowerCase().split("\\W+");

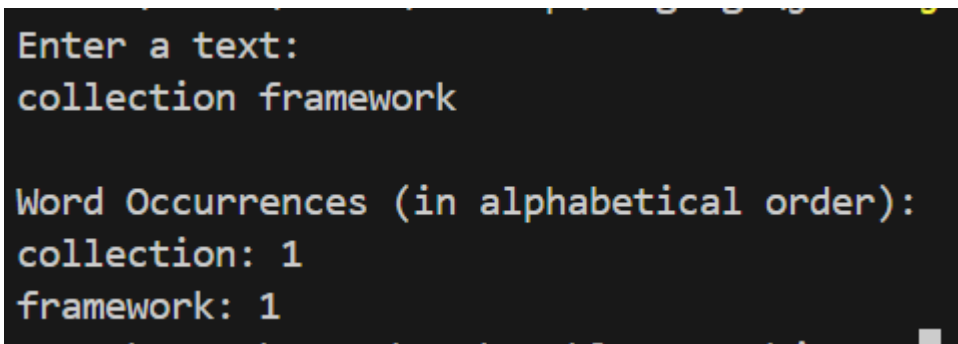
for (String word : words) {
    if (!word.isEmpty()) {
        wordMap.put(word, wordMap.getOrDefault(word, 0) + 1);
    }
}

System.out.println("\nWord Occurrences (in alphabetical order):");

Set<Map.Entry<String, Integer>> entrySet = wordMap.entrySet();

for (Map.Entry<String, Integer> entry : entrySet) {
    System.out.println(entry.getKey() + ": " + entry.getValue());
} } }
```

OUTPUT:

A screenshot of a terminal window with a dark background. It shows the program's output for the input 'collection framework'. The first line is 'Enter a text:' followed by 'collection framework' on the next line. Then, after a blank line, it says 'Word Occurrences (in alphabetical order):' followed by 'collection: 1' and 'framework: 1' on separate lines.

```
Enter a text:
collection framework

Word Occurrences (in alphabetical order):
collection: 1
framework: 1
```

CONCLUSION:

This program demonstrates how to count and display the occurrences of words in a given text using Java's Map and Set classes. The words are stored in a TreeMap, ensuring that they are presented in alphabetical order. The use of getOrDefault() simplifies the counting process, showcasing efficient word frequency analysis.

41

Write a code which counts the number of the keywords in a Java source file. Store all the keywords in a HashSet and use the contains () method to test if a word is in the keyword set.

PROGRAM CODE:

```
import java.io.*;
import java.util.*;

public class P41 {

    private static final HashSet<String> keywords = new HashSet<>();

    static {
        String[] keywordArray = {
            "abstract", "assert", "boolean", "break", "byte", "case", "catch", "char", "class",
            "const", "continue", "default", "do", "double", "else", "enum", "extends", "final",
            "finally", "float", "for", "goto", "if", "implements", "import", "instanceof", "int",
            "interface", "long", "native", "new", "package", "private", "protected", "public",
            "return", "short", "static", "strictfp", "super", "switch", "synchronized", "this",
            "throw", "throws", "transient", "try", "void", "volatile", "while"
        };

        for (String keyword : keywordArray) {
            keywords.add(keyword);
        }
    }

    public static void main(String[] args) {

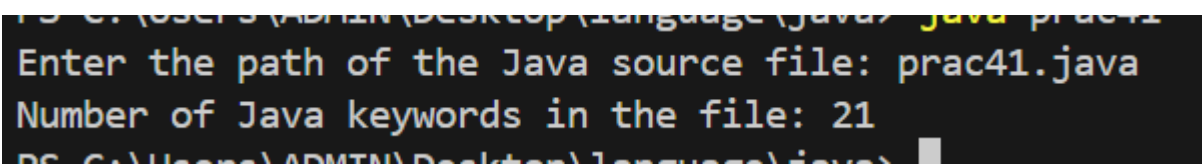
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the path of the Java source file: ");
```



```
String filePath = scanner.nextLine();

try {
    File file = new File(filePath);
    Scanner fileScanner = new Scanner(file);
    int keywordCount = 0;
    while (fileScanner.hasNext()) {
        String word = fileScanner.next();
        if (keywords.contains(word)) {
            keywordCount++;
        }
    }
    System.out.println("Number of Java keywords in the file: " + keywordCount);
    fileScanner.close();
} catch (FileNotFoundException e) {
    System.out.println("File not found: " + filePath);
} } }
```

OUTPUT:

```
PS C:\Users\ADMIN\Desktop\language\java> java prac41
Enter the path of the Java source file: prac41.java
Number of Java keywords in the file: 21
PS C:\Users\ADMIN\Desktop\language\java>
```

CONCLUSION:

This program demonstrates the use of a HashSet to efficiently count Java keywords in a source file. By reading each word from the file and checking for its presence in the set of keywords, it showcases how to utilize collections for rapid lookups. The result is the total number of keywords, providing a simple yet effective tool for analyzing Java code.