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**

	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. **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.			
	PART-II Strings			
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'	**1**	**1, 2**	**2,3,4**
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 **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. Sample string : "The quick brown fox jumps over the lazy dog." **In the above string replace all the fox with cat.**	**1**	**1, 2**	**2,3**
9	Given a string, return a string where for every char in the original, there are two chars. double_char('The') → 'TThhee' double_char('AAbb') → 'AAAAbbbb' double_char('Hi-There') → 'HHii--TThheerree'	**1**	**1, 2**	**2**
10	Perform following functionalities of the string: ● Find Length of the String ● Lowercase of the String ● Uppercase of the String ● Reverse String	**1**	**1, 2**	**2,3,4**

	Sort the string			
11	Perform following Functionalities of the string: “CHARUSAT UNIVERSITY” ● 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**

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**

	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. **Supplementary Experiment:** 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]			
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.	**2**	**2,3**	**3**
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.	**1**	**2,3**	**3**
22	Write a java that implements an interface AdvancedArithmetic which contains amethod signature int divisor_sum(int n). You need to write a class calledMyCalculator 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. **Supplementary Experiment:** 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,	**2**	**2,3**	**2,3**

	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 radiusand 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**

	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**

	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: III
Subject Code: CSE201
Academic year: 2024-25

Part – I

No.1	**Installation of java**
	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. **Download Java Development Kit (JDK)** • **Visit the official Oracle JDK download page.** • **Choose the appropriate version for your operating system (Windows, macOS, or Linux).** **Install JDK** • **Run the installer and follow the installation instructions.** • **Set the environment variables:** ○ **Windows: Add JAVA_HOME variable with the path to the JDK installation directory (e.g., C:\Program Files\Java\jdk-11.0.10).** ○ **Add bin directory of JDK to the Path variable (e.g., C:\Program Files\Java\jdk-11.0.10\bin).** **CONCLUSION:** *Here with we installed java in our system.*

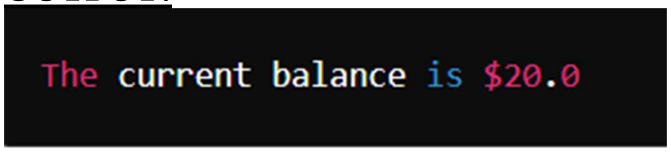
No.2	Currency Converter

Imagine you are developing a simple banking application where you need to display the current balance of a user account. For simplicity, let's say the current balance is $20. Write a java program to store this balance in a variable and then display it to the user.

PROGRAM CODE:

```
public class BankAccount {
    public static void main(String[] args) {
        double currentBalance = 20.00;

        System.out.println("The current balance is $" + currentBalance);
    }
}
```

OUTPUT:



```
The current balance is $20.0
```

CONCLUSION:

We made simple java program.

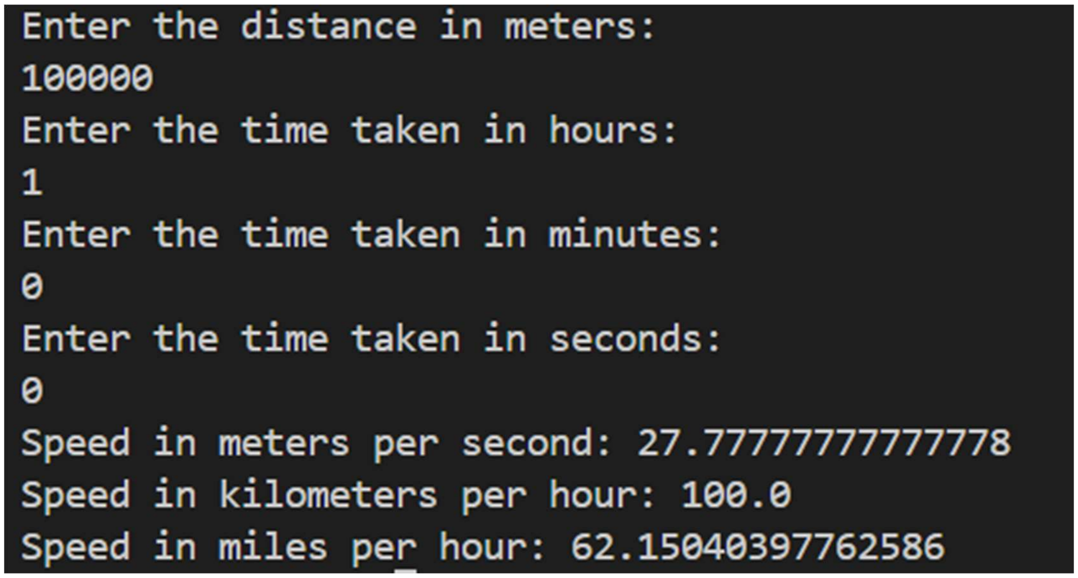
No.3	Speed Calculator

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 p3 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the distance in meters:");
        double distanceInMeters = sc.nextDouble();

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

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

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

        double timeInSeconds = hours * 3600 + minutes * 60 + seconds;

        double speedInMetersPerSecond = distanceInMeters / timeInSeconds;
        double speedInKilometersPerHour = (distanceInMeters / 1000) / (timeInSeconds / 3600);
        double speedInMilesPerHour = (distanceInMeters / 1609) / (timeInSeconds / 3600);

        System.out.println("Speed in meters per second: " + speedInMetersPerSecond);
        System.out.println("Speed in kilometers per hour: " + speedInKilometersPerHour);
        System.out.println("Speed in miles per hour: " + speedInMilesPerHour);
    }
}
```

OUTPUT:



```
Enter the distance in meters:
100000
Enter the time taken in hours:
1
Enter the time taken in minutes:
0
Enter the time taken in seconds:
0
Speed in meters per second: 27.77777777777778
Speed in kilometers per hour: 100.0
Speed in miles per hour: 62.15040397762586
```

CONCLUSION:
In this way we can implement conversions and make speed converter calculator.

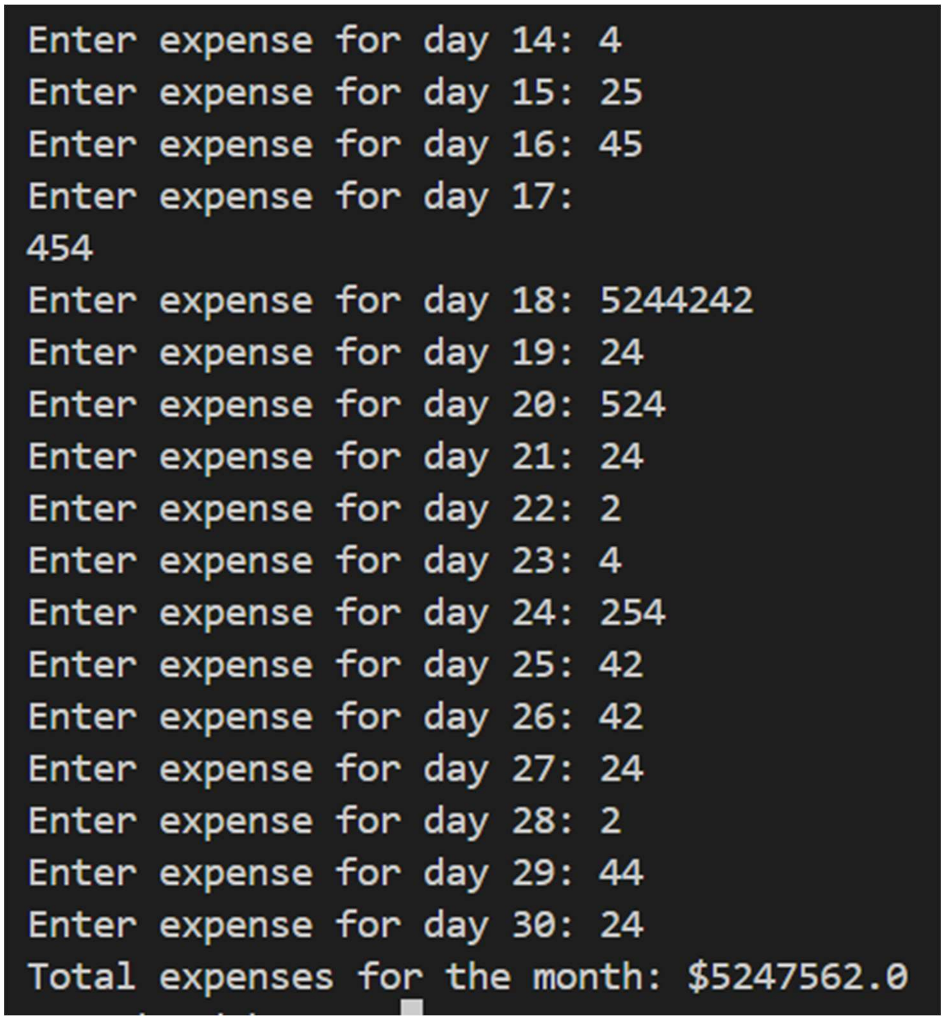
No.4	Budget tracking application

Imagine you are developing a budget tracking application. You need to calculate the total expenses for the month. Users will input their daily expenses, and the program should compute the sum of these expenses. Write a Java program to calculate the sum of elements in an array representing daily expenses.

PROGRAM CODE:

```
import java.util.Scanner;

public class p4 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double[] dailyExpenses = new double[30];
        for (int i = 0; i < dailyExpenses.length; i++) {
            System.out.print("Enter expense for day " + (i + 1) + ": ");
            dailyExpenses[i] = scanner.nextDouble();
        }
        double totalExpenses = 0;
        for (double expense : dailyExpenses) {
            totalExpenses += expense;
        }
        System.out.println("Total expenses for the month: $" + totalExpenses);
    }
}
```

OUTPUT:



```
Enter expense for day 14: 4
Enter expense for day 15: 25
Enter expense for day 16: 45
Enter expense for day 17:
454
Enter expense for day 18: 5244242
Enter expense for day 19: 24
Enter expense for day 20: 524
Enter expense for day 21: 24
Enter expense for day 22: 2
Enter expense for day 23: 4
Enter expense for day 24: 254
Enter expense for day 25: 42
Enter expense for day 26: 42
Enter expense for day 27: 24
Enter expense for day 28: 2
Enter expense for day 29: 44
Enter expense for day 30: 24
Total expenses for the month: $5247562.0
```

CONCLUSION:

We can use application of array in it and then summation of all elements can give us the answer for our application.

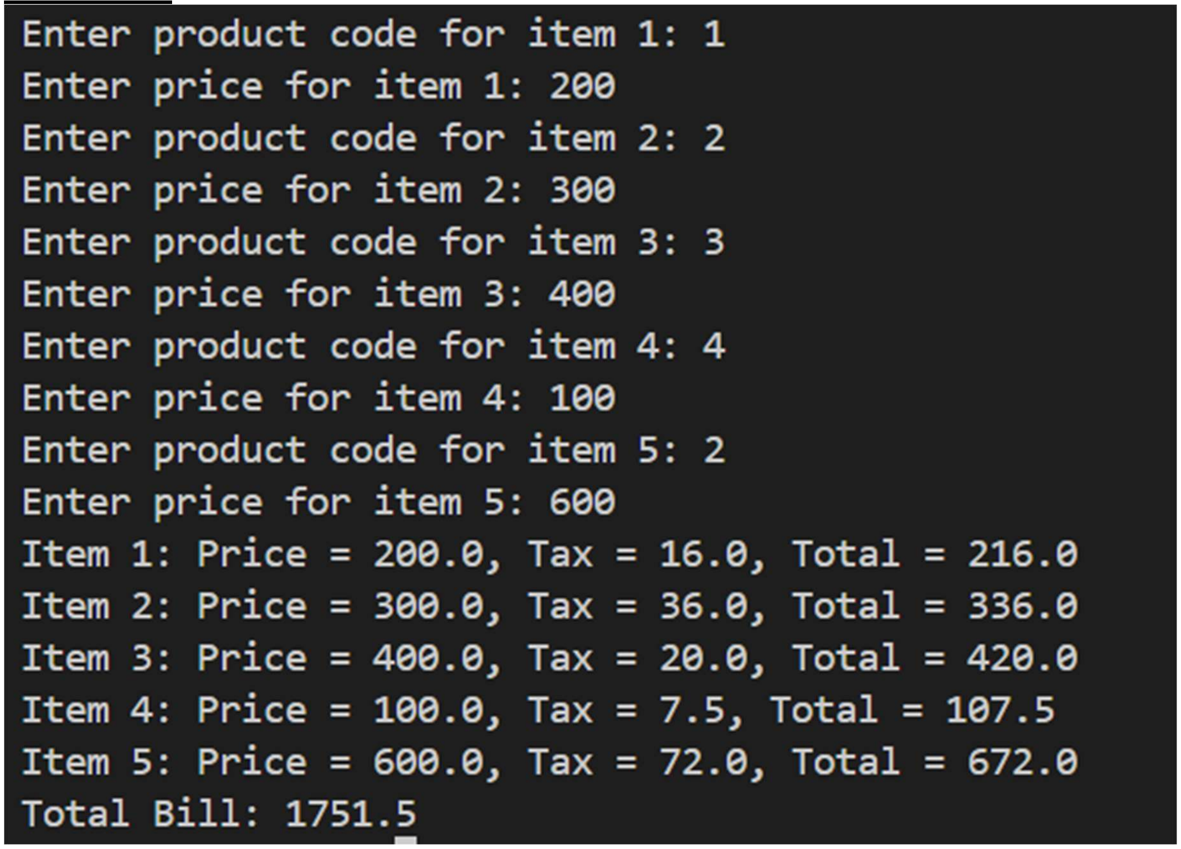
No.5 Electric shop product price calculator as per category

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 p5 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Initialize arrays for product codes and prices
        int[] productCodes = new int[5];
        double[] prices = new double[5];

        // Prompt the user to input product codes and prices
        for (int i = 0; i < productCodes.length; i++) {
            System.out.print("Enter product code for item " + (i + 1) + ": ");
            productCodes[i] = scanner.nextInt();
            System.out.print("Enter price for item " + (i + 1) + ": ");
            prices[i] = scanner.nextDouble();
        }

        // Prepare the bill
        double totalBill = 0.0;
        for (int i = 0; i < productCodes.length; i++) {
            double taxRate = getTaxRate(productCodes[i]);
            double taxAmount = prices[i] * (taxRate / 100);
            double totalAmount = prices[i] + taxAmount;
            System.out.println(
                    "Item " + (i + 1) + ": Price = " + prices[i] + ", Tax = " + taxAmount + ", Total = " +
totalAmount);
            totalBill += totalAmount;
        }

        System.out.println("Total Bill: " + totalBill);
    }
    public static double getTaxRate(int productCode) {
        double taxRate = 0.0;
        switch (productCode) {
            case 1:
                taxRate = 8.0;
                break;
            case 2:
                taxRate = 12.0;
                break;
            case 3:
                taxRate = 5.0;
                break;
            case 4:
                taxRate = 7.5;
                break;
            default:
                taxRate = 3.0;
                break;
        }
        return taxRate;
    }
}
```

OUTPUT:



```
Enter product code for item 1: 1
Enter price for item 1: 200
Enter product code for item 2: 2
Enter price for item 2: 300
Enter product code for item 3: 3
Enter price for item 3: 400
Enter product code for item 4: 4
Enter price for item 4: 100
Enter product code for item 5: 2
Enter price for item 5: 600
Item 1: Price = 200.0, Tax = 16.0, Total = 216.0
Item 2: Price = 300.0, Tax = 36.0, Total = 336.0
Item 3: Price = 400.0, Tax = 20.0, Total = 420.0
Item 4: Price = 100.0, Tax = 7.5, Total = 107.5
Item 5: Price = 600.0, Tax = 72.0, Total = 672.0
Total Bill: 1751.5
```

CONCLUSION:

We can divide 3 category of products and we can apply tax on basis of that 3 categories.

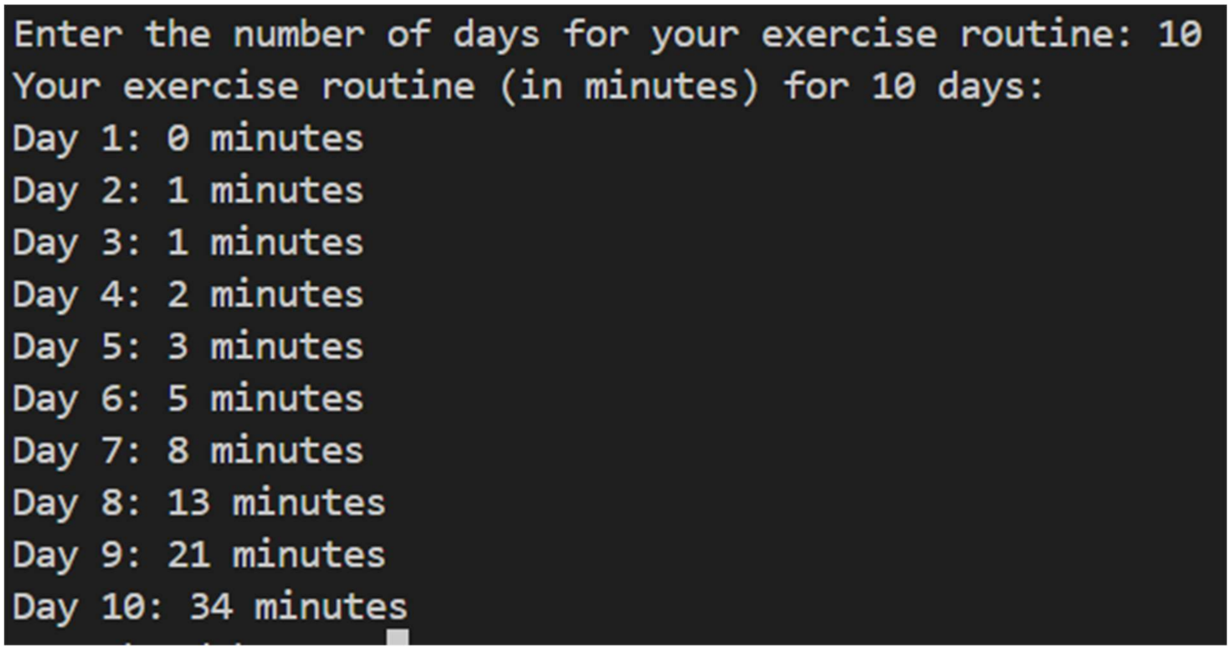
No.6 Daily exercise planning using Fibonacci

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

        // Prompt the user to enter the number of days
        System.out.print("Enter the number of days for your exercise routine: ");
        int n = scanner.nextInt();

        // Generate the Fibonacci series for the given number of days
        int[] fibonacciSeries = generateFibonacciSeries(n);

        // Display the exercise routine
        System.out.println("Your exercise routine (in minutes) for " + n + " days:");
        for (int i = 0; i < n; i++) {
            System.out.println("Day " + (i + 1) + ": " + fibonacciSeries[i] + " minutes");
        }


    // Method to generate the first n terms of the Fibonacci series
    public static int[] generateFibonacciSeries(int n) {
        int[] series = new int[n];
        if (n > 0) series[0] = 0; // First term
        if (n > 1) series[1] = 1; // Second term

        for (int i = 2; i < n; i++) {
            series[i] = series[i - 1] + series[i - 2];
        }

        return series;
    }
}
```

OUTPUT:



```
Enter the number of days for your exercise routine: 10
Your exercise routine (in minutes) for 10 days:
Day 1: 0 minutes
Day 2: 1 minutes
Day 3: 1 minutes
Day 4: 2 minutes
Day 5: 3 minutes
Day 6: 5 minutes
Day 7: 8 minutes
Day 8: 13 minutes
Day 9: 21 minutes
Day 10: 34 minutes
```

CONCLUSION:

Here we are using simple application of fibonacci series expression.

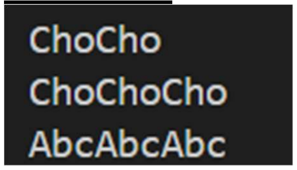
Part-II

No.7	**Operations on the String**
	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:

```java
public class Main {
    public static void main(String[] args) {
        System.out.println(frontTimes("Chocolate", 2)); // "ChoCho"
        System.out.println(frontTimes("Chocolate", 3)); // "ChoChoCho"
        System.out.println(frontTimes("Abc", 3)); // "AbcAbcAbc"
    }

    public static String frontTimes(String str, int n) {
        String front = str.substring(0, Math.min(3, str.length()));
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < n; i++) {
            result.append(front);
        }
        return result.toString();
    }
}
```

OUTPUT:



```
ChoCho
ChoChoCho
AbcAbcAbc
```

CONCLUSION:

With using of string functions we can implement this program.

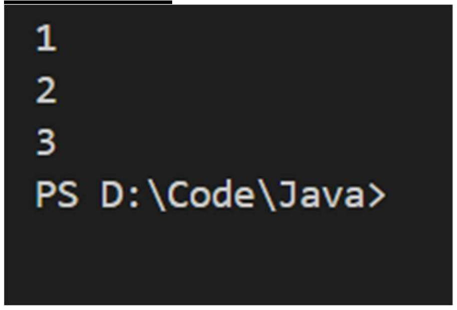
No.8	Operations in the array.

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 Main {
   public static void main(String[] args) {
      int[] nums1 = {1, 2, 9};
      int[] nums2 = {1, 9, 9};
      int[] nums3 = {1, 9, 9, 3, 9};

      System.out.println(array_count9(nums1)); // 1
      System.out.println(array_count9(nums2)); // 2
      System.out.println(array_count9(nums3)); // 3
   }

   public static int array_count9(int[] nums) {
      int count = 0;
      for (int num : nums) {
         if (num == 9) {
            count++;
         }
      }
      return count;
   }
}
```

OUTPUT:



```
1
2
3
PS D:\Code\Java>
```

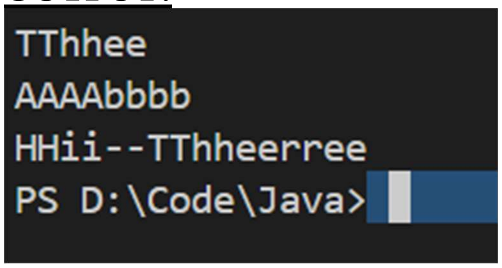
No.9	Changing the String

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

PROGRAM CODE:

```
public class Main {
   public static void main(String[] args) {
      System.out.println(double_char("The")); // "TThhee"
      System.out.println(double_char("AAbb")); // "AAAAbbbb"
      System.out.println(double_char("Hi-There")); // "HHii--TThheerree"
   }

   public static String double_char(String str) {
      StringBuilder result = new StringBuilder();
      for (char c : str.toCharArray()) {
         result.append(c).append(c);
      }
      return result.toString();
   }
}
```

OUTPUT:



```
TThhee
AAAAbbbb
HHii--TThheerree
PS D:\Code\Java>
```

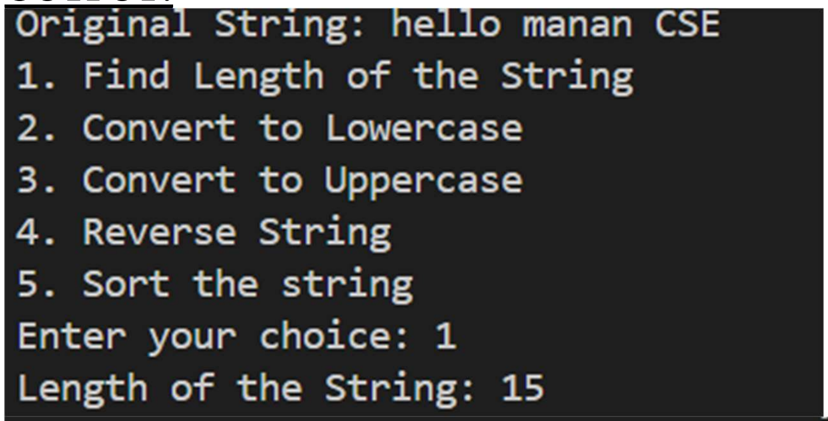
No.10	String operations

Perform following functionalities of the string: ● Find Length of the String ● Lowercase of the String ● Uppercase of the String ● Reverse String

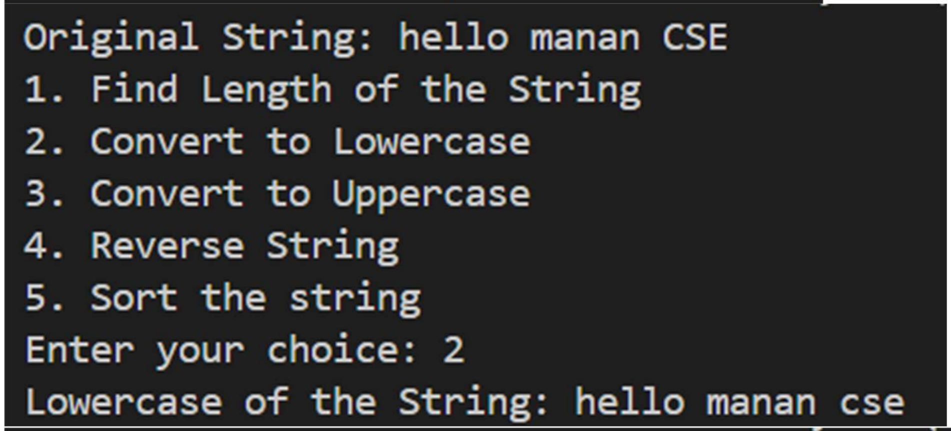
PROGRAM CODE:

```
public class Main {
    public static void main(String[] args) {
        String str = "hello manan CSE";
        int choice;

        System.out.println("Original String: " + str);
        System.out.println("1. Find Length of the String");
        System.out.println("2. Convert to Lowercase");
        System.out.println("3. Convert to Uppercase");
        System.out.println("4. Reverse String");
        System.out.println("5. Sort the string");
        System.out.print("Enter your choice: ");
        choice = Integer.parseInt(System.console().readLine());

        switch (choice) {
            case 1:
                System.out.println("Length of the String: " + str.length());
                break;
            case 2:
                System.out.println("Lowercase of the String: " + str.toLowerCase());
                break;
            case 3:
                System.out.println("Uppercase of the String: " + str.toUpperCase());
                break;
            case 4:
                System.out.println("Reverse String: " + reverseString(str));
                break;
            case 5:
                System.out.println("Sorted String: " + sortString(str));
                break;
            default:
                System.out.println("Invalid choice");
        }
    }

    public static String reverseString(String str) {
        StringBuilder sb = new StringBuilder(str);
        return sb.reverse().toString();
    }
    public static String sortString(String str) {
        char[] chars = str.toLowerCase().toCharArray();
        java.util.Arrays.sort(chars);
        return new String(chars);
    }
}
```

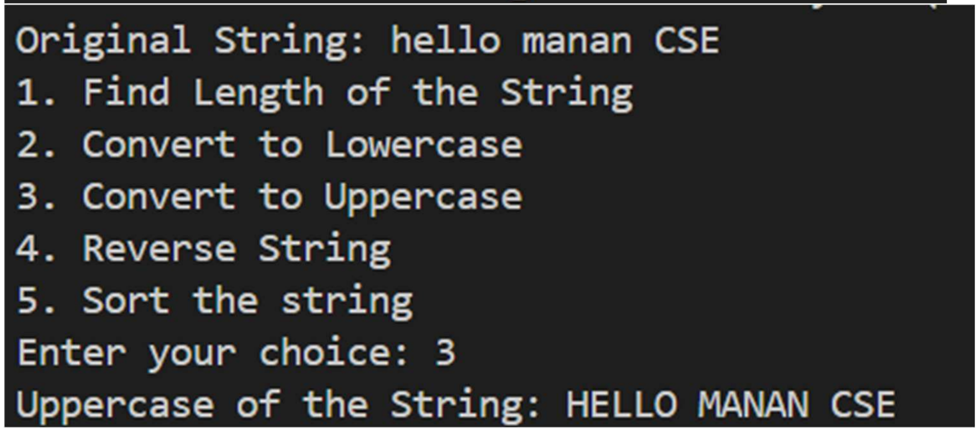
OUTPUT:



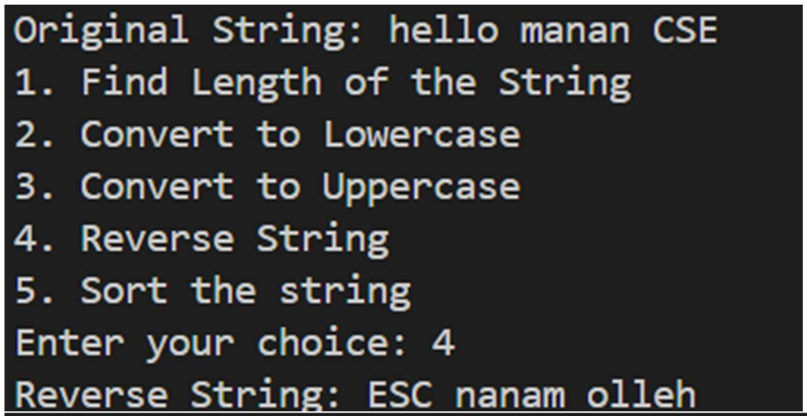
```
Original String: hello manan CSE
1. Find Length of the String
2. Convert to Lowercase
3. Convert to Uppercase
4. Reverse String
5. Sort the string
Enter your choice: 1
Length of the String: 15
```



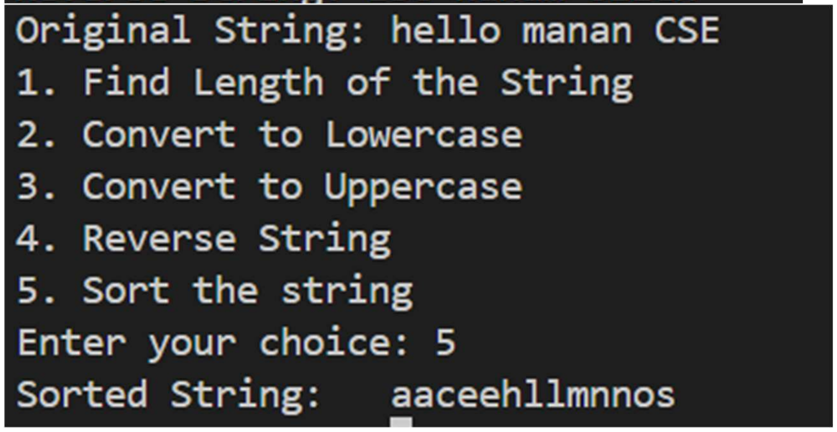
```
Original String: hello manan CSE
1. Find Length of the String
2. Convert to Lowercase
3. Convert to Uppercase
4. Reverse String
5. Sort the string
Enter your choice: 2
Lowercase of the String: hello manan cse
```



```
Original String: hello manan CSE
1. Find Length of the String
2. Convert to Lowercase
3. Convert to Uppercase
4. Reverse String
5. Sort the string
Enter your choice: 3
Uppercase of the String: HELLO MANAN CSE
```



```
Original String: hello manan CSE
1. Find Length of the String
2. Convert to Lowercase
3. Convert to Uppercase
4. Reverse String
5. Sort the string
Enter your choice: 4
Reverse String: ESC nanam olleh
```



```
Original String: hello manan CSE
1. Find Length of the String
2. Convert to Lowercase
3. Convert to Uppercase
4. Reverse String
5. Sort the string
Enter your choice: 5
Sorted String:   aaceehllmnnos
```

CONCLUSION:
We can perform string operations like above.

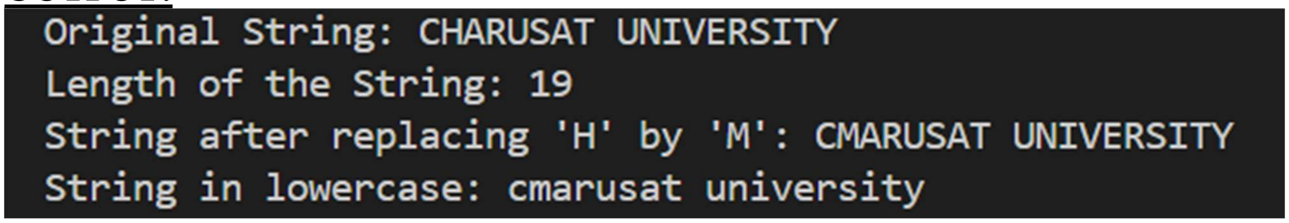
No.11	CHARUSAT String

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 Main {
   public static void main(String[] args) {
      String str = "CHARUSAT UNIVERSITY";

      System.out.println("Original String: " + str);

      // Find length
      System.out.println("Length of the String: " + str.length());

      // Replace 'H' by 'FIRST LETTER OF YOUR NAME'
      String firstName = "MANAN"; // Replace with your first name
      char firstLetter = firstName.charAt(0);
      String replacedStr = str.replace('H', firstLetter);
      System.out.println("String after replacing 'H' by '" + firstLetter + "': " + replacedStr);

      // Convert all characters in lowercase
      String lowerCaseStr = replacedStr.toLowerCase();
      System.out.println("String in lowercase: " + lowerCaseStr);
   }
}
```

OUTPUT:



```
Original String: CHARUSAT UNIVERSITY
Length of the String: 19
String after replacing 'H' by 'M': CMARUSAT UNIVERSITY
String in lowercase: cmarusat university
```

Part-III

No.12	CurrencyConverter

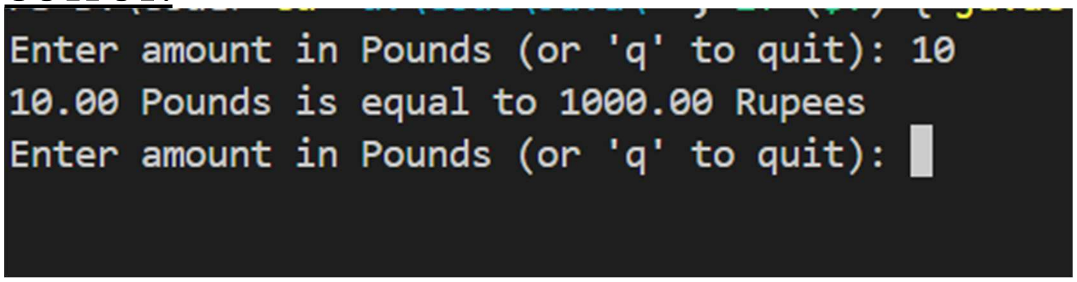
Imagine you are developing a currency conversion tool for a travel agency. This tool should be able to convert an amount in Pounds to Rupees. For simplicity, we assume the conversion rate is fixed: 1 Pound = 100 Rupees. The tool should be able to take input both from command-line arguments and interactively from the user.

PROGRAM CODE:

```
import java.util.Scanner;

public class CurrencyConverter {

   // Fixed conversion rate (1 Pound = 100 Rupees)
   private static final double CONVERSION_RATE = 100.0;

   public static void main(String[] args) {
      // Check if command-line argument is provided
      if (args.length > 0) {
         try {
            // Convert command-line argument to double
            double pounds = Double.parseDouble(args[0]);
            double rupees = convertPoundsToRupees(pounds);
            System.out.printf("%.2f Pounds is equal to %.2f Rupees%n", pounds, rupees);
         } catch (NumberFormatException e) {
            System.out.println("Invalid input. Please enter a valid number.");
         }
      } else {
         // Interactive mode
         Scanner scanner = new Scanner(System.in);
         while (true) {
            System.out.print("Enter amount in Pounds (or 'q' to quit): ");
            String input = scanner.next();
            if (input.equalsIgnoreCase("q")) {
               break;
            }
            try {
               double pounds = Double.parseDouble(input);
               double rupees = convertPoundsToRupees(pounds);
               System.out.printf("%.2f Pounds is equal to %.2f Rupees%n", pounds, rupees);
            } catch (NumberFormatException e) {
               System.out.println("Invalid input. Please enter a valid number.");
            }
         }
```

```
            scanner.close();
        }
    }

    private static double convertPoundsToRupees(double pounds) {
        return pounds * CONVERSION_RATE;
    }
}
```

OUTPUT:



```
Enter amount in Pounds (or 'q' to quit): 10
10.00 Pounds is equal to 1000.00 Rupees
Enter amount in Pounds (or 'q' to quit):
```

CONCLUSION:

In this way we can implement the currency converter calculator.

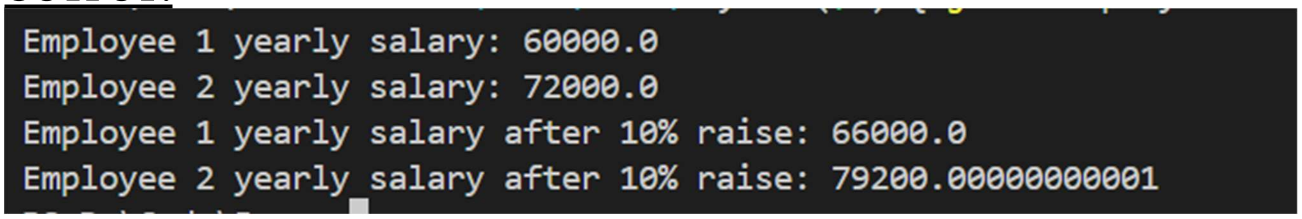
No.13	**Instance Variables**

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:

```
public class Employee {
   private String firstName;
   private String lastName;
   private double monthlySalary;

   public Employee(String firstName, String lastName, double monthlySalary) {
      this.firstName = firstName;
      this.lastName = lastName;
      if (monthlySalary >= 0) {
         this.monthlySalary = monthlySalary;
      } else {
         this.monthlySalary = 0.0;
      }
   }

   public void setFirstName(String firstName) {
      this.firstName = firstName;
   }

   public String getFirstName() {
      return firstName;
   }

   public void setLastName(String lastName) {
      this.lastName = lastName;
   }

   public String getLastName() {
      return lastName;
   }

   public void setMonthlySalary(double monthlySalary) {
      if (monthlySalary >= 0) {
            this.monthlySalary = monthlySalary;
        } else {
            this.monthlySalary = 0.0;
        }
    }

    public double getMonthlySalary() {
        return monthlySalary;
    }

    public double getYearlySalary() {
        return monthlySalary * 12;
    }
}

public class EmployeeTest {
    public static void main(String[] args) {
        Employee joe = new Employee("Joe", "Rogan", 9696);
        Employee tod = new Employee("Tod", "Howard", 8595);
        System.out.println(joe.getFirstName() + "'s yearly salary is " + joe.getYearlySalary());
        System.out.println(tod.getFirstName() + "'s yearly salary is " + tod.getYearlySalary());
        joe.setMonthlySalary(joe.getMonthlySalary() * 1.1);
        tod.setMonthlySalary(tod.getMonthlySalary() * 1.1);
        System.out.println("Salary after 10% boost");
        System.out.println(joe.getFirstName() + "'s yearly salary is " + joe.getYearlySalary());
        System.out.println(tod.getFirstName() + "'s yearly salary is " + tod.getYearlySalary());
    }
}
```

OUTPUT:



```
Employee 1 yearly salary: 60000.0
Employee 2 yearly salary: 72000.0
Employee 1 yearly salary after 10% raise: 66000.0
Employee 2 yearly salary after 10% raise: 79200.00000000001
```

CONCLUSION:

By performing this practical we can learn about instance variables and we can also learn that how to implement it in the real code.

No.14 Instance Variables

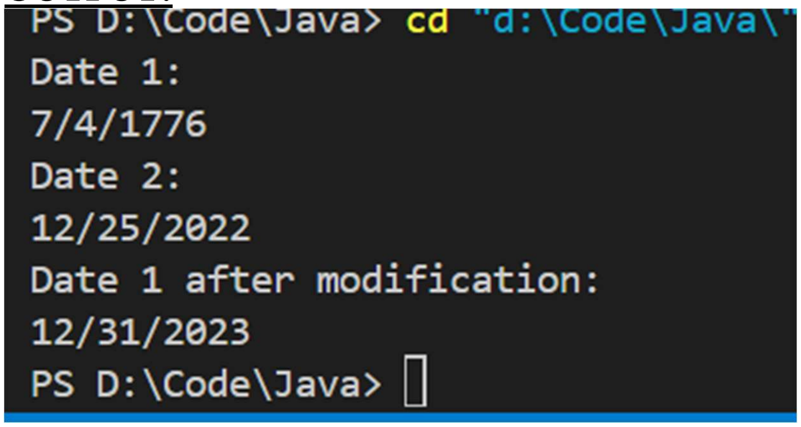
Create a class called Date that includes three pieces of information as instance variables—a month (type int), a day (type int) and a year (type int). Your class should have a constructor that initializes the three instance variables and assumes that the values provided are correct. Provide a set and 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:

```
public class Date {
   private int month;
   private int day;
   private int year;

   public Date(int month, int day, int year) {
      this.month = month;
      this.day = day;
      this.year = year;
   }

   public void setMonth(int month) {
      this.month = month;
   }

   public int getMonth() {
      return month;
   }

   public void setDay(int day) {
      this.day = day;
   }

   public int getDay() {
      return day;
   }

   public void setYear(int year) {
      this.year = year;
   }

   public int getYear() {
      return year;
   }
```

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

public class DateTest {
    public static void main(String[] args) {
        Date date1 = new Date(7, 4, 1776);
        System.out.println("Date 1:");
        date1.displayDate();

        Date date2 = new Date(12, 25, 2022);
        System.out.println("Date 2:");
        date2.displayDate();

        date1.setMonth(12);
        date1.setDay(31);
        date1.setYear(2023);
        System.out.println("Date 1 after modification:");
        date1.displayDate();
    }
}
```

OUTPUT:



```
PS D:\Code\Java> cd "d:\Code\Java\"
Date 1:
7/4/1776
Date 2:
12/25/2022
Date 1 after modification:
12/31/2023
PS D:\Code\Java> 
```

CONCLUSION:

By performing this type of practical we are able to deal with date & time related coding problems we can deal with them in a better way.

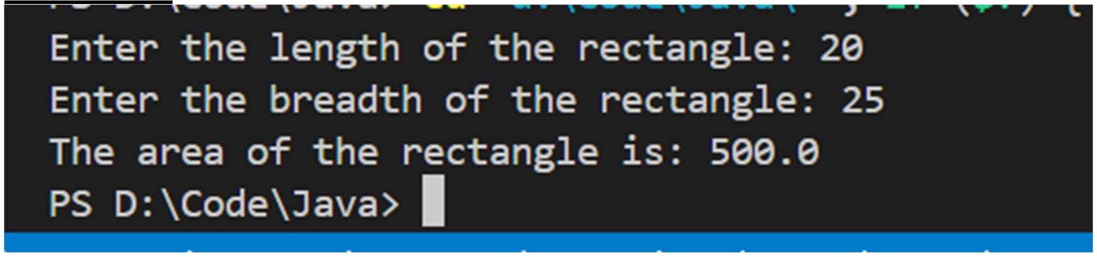
No.15 Area of Rectangle

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.Scanner;

class Area {
   private double length;
   private double breadth;

   public Area(double length, double breadth) {
      this.length = length;
      this.breadth = breadth;
   }

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

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

      System.out.print("Enter the length of the rectangle: ");
      double length = scanner.nextDouble();

      System.out.print("Enter the breadth of the rectangle: ");
      double breadth = scanner.nextDouble();

      Area rectangle = new Area(length, breadth);

      System.out.println("The area of the rectangle is: " + rectangle.returnArea());
   }
}
```

OUTPUT:



```
Enter the length of the rectangle: 20
Enter the breadth of the rectangle: 25
The area of the rectangle is: 500.0
PS D:\Code\Java>
```

CONCLUSION:

In this practical we can implement the code of calculating the area of the rectangle using a class named as 'Area'. And by using its object method we can implement the practical.

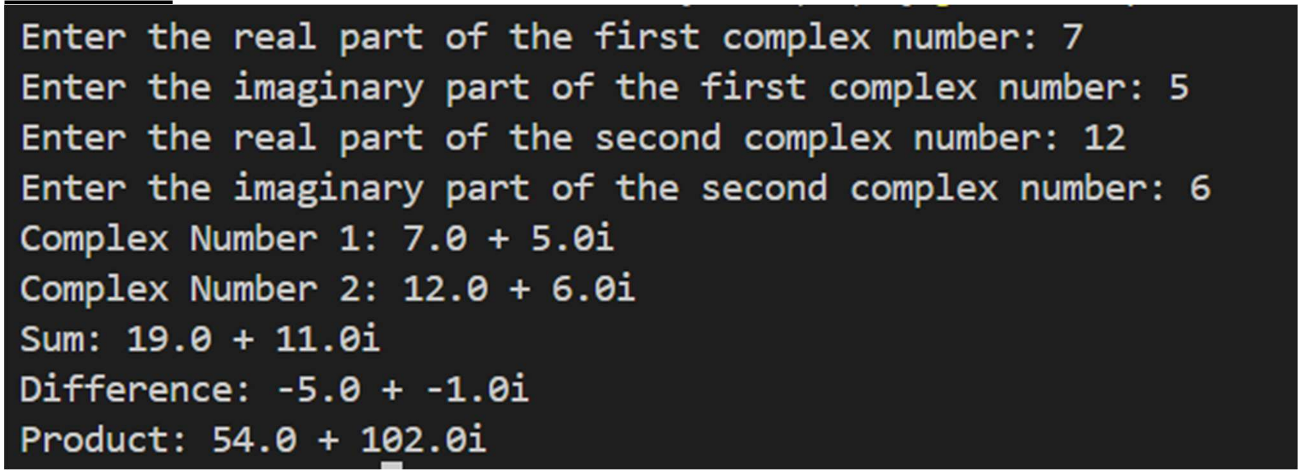
No.16	Operation on Complex Numbers

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 {
    private double real;
    private double imaginary;

    public Complex(double real, double imaginary) {
        this.real = real;
        this.imaginary = imaginary;
    }

    public Complex add(Complex other) {
        double newReal = this.real + other.real;
        double newImaginary = this.imaginary + other.imaginary;
        return new Complex(newReal, newImaginary);
    }

    public Complex subtract(Complex other) {
        double newReal = this.real - other.real;
        double newImaginary = this.imaginary - other.imaginary;
        return new Complex(newReal, newImaginary);
    }

    public Complex multiply(Complex other) {
        double newReal = this.real * other.real - this.imaginary * other.imaginary;
        double newImaginary = this.real * other.imaginary + this.imaginary * other.real;
        return new Complex(newReal, newImaginary);
    }

    @Override
    public String toString() {
        return real + " + " + imaginary + "i";
    }
}

public class ComplexNumberOperations {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the real part of the first complex number: ");
        double real1 = scanner.nextDouble();

        System.out.print("Enter the imaginary part of the first complex number: ");
        double imaginary1 = scanner.nextDouble();

        Complex complex1 = new Complex(real1, imaginary1);

        System.out.print("Enter the real part of the second complex number: ");
        double real2 = scanner.nextDouble();

        System.out.print("Enter the imaginary part of the second complex number: ");
        double imaginary2 = scanner.nextDouble();

        Complex complex2 = new Complex(real2, imaginary2);

        Complex sum = complex1.add(complex2);
        Complex difference = complex1.subtract(complex2);
        Complex product = complex1.multiply(complex2);

        System.out.println("Complex Number 1: " + complex1);
        System.out.println("Complex Number 2: " + complex2);
        System.out.println("Sum: " + sum);
        System.out.println("Difference: " + difference);
        System.out.println("Product: " + product);
    }
}
```

OUTPUT:



```
Enter the real part of the first complex number: 7
Enter the imaginary part of the first complex number: 5
Enter the real part of the second complex number: 12
Enter the imaginary part of the second complex number: 6
Complex Number 1: 7.0 + 5.0i
Complex Number 2: 12.0 + 6.0i
Sum: 19.0 + 11.0i
Difference: -5.0 + -1.0i
Product: 54.0 + 102.0i
```

CONCLUSION:

By making the class of complex we can implement this practical.

Part-IV

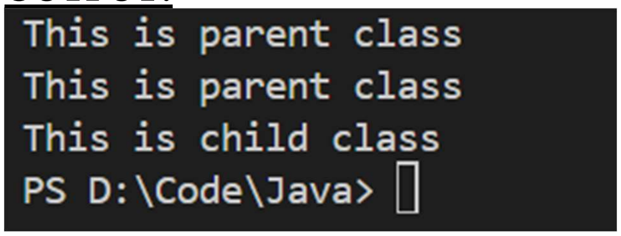
No.17	Class Calling

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

PROGRAM CODE:

```
class Parent {
   void printtextParent() {
      System.out.println("this is parent class");
   }
}

class child extends Parent {
   void printtextChild() {
      System.out.println("this is child class");
   }
}

class grandchild extends child {
   void printtextGrandchild() {
      System.out.println("this is grandchild class");
   }
}

public class practical17 {
   public static void main(String[] args) {
      {
         Parent p1 = new Parent();
         child c = new child();
         grandchild gc = new grandchild();

         p1.printtextParent();
         c.printtextParent();
         c.printtextChild();
         gc.printtextParent();
         gc.printtextChild();
         gc.printtextGrandchild();

      }
   }
}
```

OUTPUT:



```
This is parent class
This is parent class
This is child class
PS D:\Code\Java> 
```

CONCLUSION:

In this practical we can learn about the class and how to call a parent class from the child class . we can learn about the concepts of inheritance of the class.

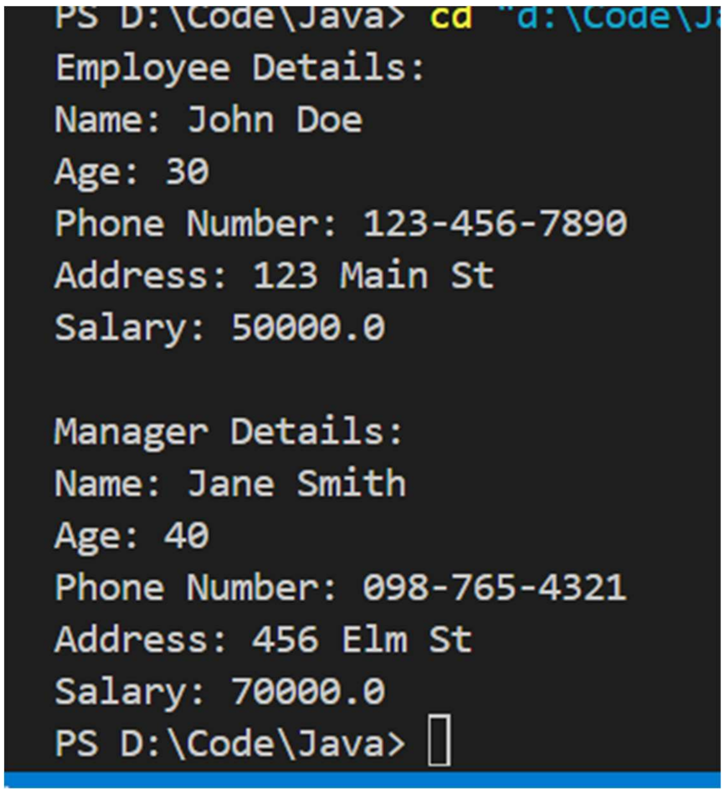
No.18 Employee & Manager class

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
{
   public
   int age;
   String name;
   long phoneNumber;
   String address;
   int salary;
   // method printSalary
   public void printSalary()
   {
      System.out.println("Salary: " + salary);
   }
   // assign age,name,phoneNumber,address,salary
   void insert(int age,String name,long phoneNumber,String address,int salary)
   {
      this.age=age;
      this.name=name;
      this.phoneNumber=phoneNumber;
      this.address=address;
      this.salary=salary;

   }
   void printBio()
   {
      // print all the details

      System.out.println("name is "+name);
      System.out.println("age is"+age);
      System.out.println("phone number is "+phoneNumber);
      System.out.println("address is"+address);
      System.out.println("salary is "+salary);
   }

}

class  Employee extends member
{public
   String specializatioString;
   String departmentString;
}

class Manager extends member{
   public
   String specializatioString;
   String departmentString;
}

public class prectical18 {
   public static void main(String[] args) {
      Employee e1Employee=new Employee();
      Manager m1Manager=new Manager();
      e1Employee.insert(25,"John",1234567890,"New York",50000);
      m1Manager.insert(30,"Mike",987654321,"Los Angeles",60000);
      e1Employee.printBio();
      m1Manager.printBio();

   }
```

OUTPUT:



```
PS D:\Code\Java> cd "d:\Code\J
Employee Details:
Name: John Doe
Age: 30
Phone Number: 123-456-7890
Address: 123 Main St
Salary: 50000.0

Manager Details:
Name: Jane Smith
Age: 40
Phone Number: 098-765-4321
Address: 456 Elm St
Salary: 70000.0
PS D:\Code\Java>
```

CONCLUSION:
we can learn about the concepts of inheritance of the class.

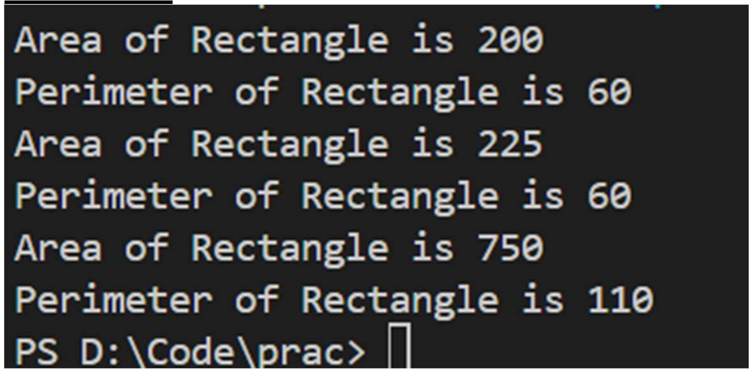
No.19	**Area + perimeter of the Rectangle and Square**

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:
```
class Rectangle {
   private int length;
   private int breadth;

   public Rectangle(int length, int breadth) {
      this.length = length;
```

```
        this.breadth = breadth;
    }

    void printArea() {
        System.out.println("Area of Rectangle is " + length * breadth);
    }

    void printPerimeter() {
        System.out.println("Perimeter of Rectangle is " + 2 * (length + breadth));
    }
}

class Square extends Rectangle {
    private int side;

    public Square(int side) {
        super(side, side);
    }
}

public class practical19{
    public static void main(String[] args) {
        Rectangle[] rectangles = new Rectangle[3];
        rectangles[0] = new Rectangle(10, 20);
        rectangles[1] = new Square(15);
        rectangles[2] = new Rectangle(25, 30);
        for (Rectangle rectangle : rectangles) {
            rectangle.printArea();
            rectangle.printPerimeter();

        }}}
```

OUTPUT:



```
Area of Rectangle is 200
Perimeter of Rectangle is 60
Area of Rectangle is 225
Perimeter of Rectangle is 60
Area of Rectangle is 750
Perimeter of Rectangle is 110
PS D:\Code\prac> 
```

CONCLUSION:

we can learn about the concepts of inheritance of the class.

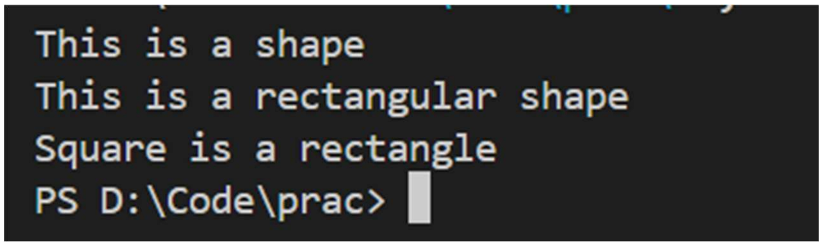
No.20	**Shapes class inheritance**

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 {
   public void printShape() {
      System.out.println("This is a shape");
   }
}

 class Rectangle extends Shape {
   public void printRectangle() {
      System.out.println("This is a rectangular shape");
   }
}

 class Circle extends Shape {
   public void printCircle() {
      System.out.println("This is a circular shape");
   }
}

 class Square extends Rectangle {
   public void printSquare() {
      System.out.println("Square is a rectangle");
   }
}

// Main
public class prectical20{
   public static void main(String[] args) {
      Square square = new Square();

      square.printShape();
      square.printRectangle();
      square.printSquare();
   }
}
```

OUTPUT:



```
This is a shape
This is a rectangular shape
Square is a rectangle
PS D:\Code\prac>
```

CONCLUSION:

we can learn about the concepts of inheritance of the class.

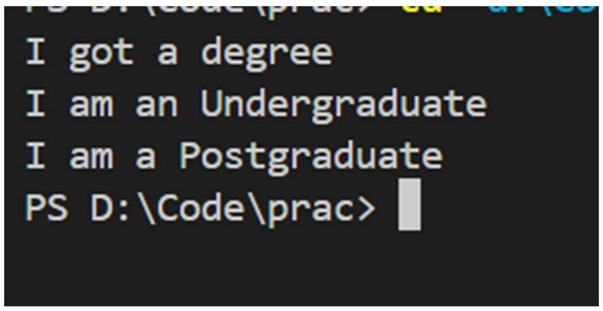
No.21	Making of subclass

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 prectical21 {
   public  static void main(String[] args) {
      Degree degree = new Degree();
      degree.getDegree(); // Output: I got a degree

      Undergraduate undergraduate = new Undergraduate();
      undergraduate.getDegree(); // Output: I am an Undergraduate

      Postgraduate postgraduate = new Postgraduate();
      postgraduate.getDegree(); // Output: I am a Postgraduate
    }
}
```

OUTPUT:

```
I got a degree
I am an Undergraduate
I am a Postgraduate
PS D:\Code\prac>
```

CONCLUSION:

we can learn about the concepts of inheritance of the class.

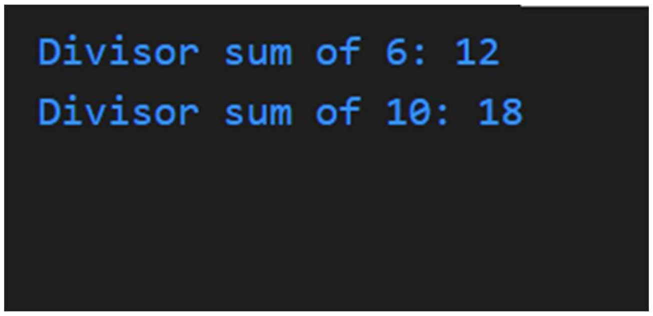
No.22	My Calculator

Write a java that implements an interface AdvancedArithmetic which contains amethod signature int divisor_sum(int n). You need to write a class calledMyCalculator 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.Scanner;

interface AdvancedArithmetic {
   int divisorSum(int n);
}

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

   public static void main(String[] args) {
      MyCalculator calculator = new MyCalculator();
      System.out.print("enter the number for the find the sum of the divisiors : ");
      Scanner sc = new Scanner(System.in);
      int number = sc.nextInt();
      System.out.println("Sum of divisors of " + number + " is " + calculator.divisorSum(number));

   }
}
```

OUTPUT:



```
Divisor sum of 6: 12
Divisor sum of 10: 18
```

CONCLUSION:
we can learn about the concepts of interface in java programming and how it will be used to implement the class .

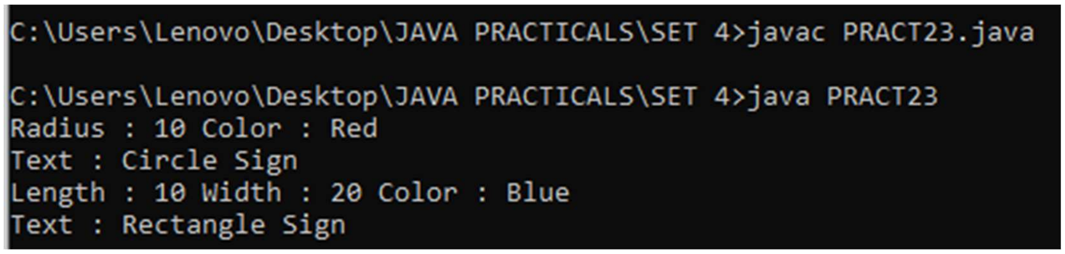
No.23	A program using interface concept
	Assume you want to capture shapes, which can be either circles (with a radiusand a color) or rectangles (with a length, width, and color). You also want to be able to create signs (to post in the campus center, for example), each of which has a shape (for the 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:

```
interface Shape{
void print();
}
class Circle implements Shape{
int radius;
String color;
Circle(int radius, String color){
this.radius = radius;
this.color = color;
}
public void print(){
System.out.println("Radius : "+radius+" Color : "+color);
}
}
class Rectangle implements Shape{
int length;
int width;
String color;
Rectangle(int length, int width, String color){
this.length = length;
this.width = width;
this.color = color;
}
public void print(){
System.out.println("Length : "+length+" Width : "+width+" Color : "+color);
}}
class Sign{
Shape s;
String text;
Sign(Shape s, String text){
this.s = s;
this.text = text;
}
void print(){
s.print();
System.out.println("Text : "+text);
}
}

public class PRACT23 {
public static void main(String[] args) {
Circle c = new Circle(10, "Red");
Rectangle r = new Rectangle(10, 20, "Blue");
Sign s = new Sign(c, "Circle Sign");
s.print();
Sign s1 = new Sign(r, "Rectangle Sign");
s1.print();
}
```

}

OUTPUT:



```
C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 4>javac PRACT23.java

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 4>java PRACT23
Radius : 10 Color : Red
Text : Circle Sign
Length : 10 Width : 20 Color : Blue
Text : Rectangle Sign
```

CONCLUSION:

The code defines an interface Shape with a default and an abstract method, and two classes Circle and Rectangle implement it. The Sign class associates a shape with text, and in the main() method, it prints the details of both a circle and a rectangle along with their corresponding signs.

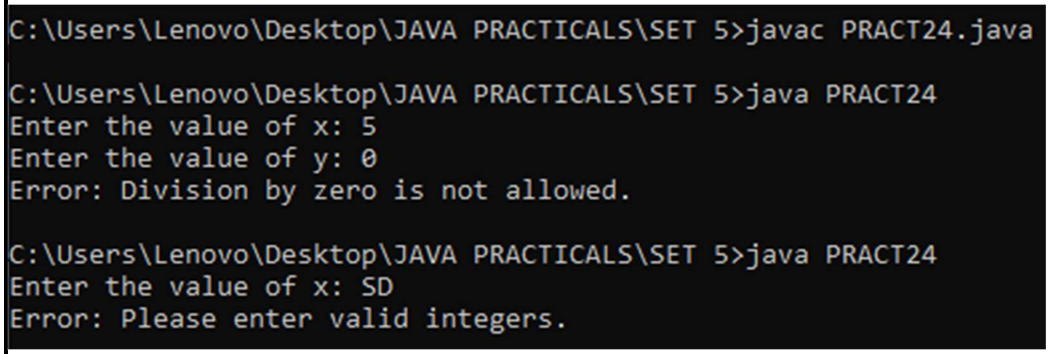
Part - 5

No.	Aim of the Practical
24.	Write a java program which takes two integers x & y as input, you have to compute x/y. If x and y are not integers or if y is zero, exception will occur and you have to report it.

PROGRAM CODE :

```
import java.util.*;
public class PRACT24 {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
try {
System.out.print("Enter the value of x: ");
int x = scanner.nextInt();
System.out.print("Enter the value of y: ");
int y = scanner.nextInt();
int result = x / y;
System.out.println("Result: " + result);
} catch (InputMismatchException e) {
System.out.println("Error:   Please   enter   valid
integers.");
} catch (ArithmeticException e) {
System.out.println("Error:Division by zero is not allowed");
}
scanner.close();
}}
```

OUTPUT:



```
C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 5>javac PRACT24.java

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 5>java PRACT24
Enter the value of x: 5
Enter the value of y: 0
Error: Division by zero is not allowed.

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 5>java PRACT24
Enter the value of x: SD
Error: Please enter valid integers.
```

CONCLUSION:

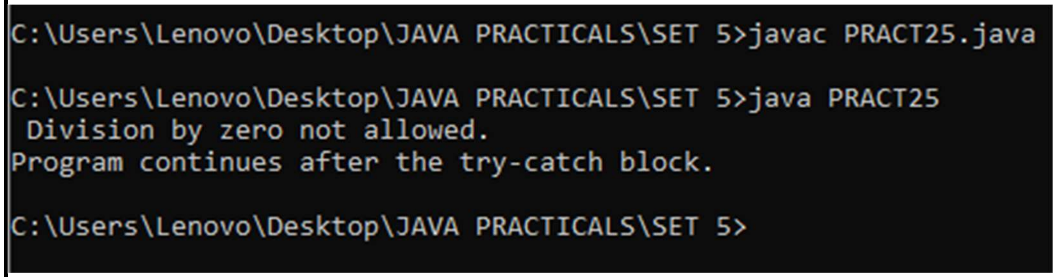
This Java program takes two integer inputs from the user and performs division, handling exceptions for invalid input and division by zero. It ensures the program doesn't crash by providing appropriate error messages for these cases.

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

PROGRAM CODE:

```
public class PRACT25 {

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

int number = 10;
int result = number / 0;
System.out.println("The result is: " + result);
} catch (ArithmeticException e) {

System.out.println(" Division by zero not allowed.");
}

System.out.println("Program continues after the try-catch block.");
}
}
```

OUTPUT:



```
C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 5>javac PRACT25.java

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 5>java PRACT25
 Division by zero not allowed.
Program continues after the try-catch block.

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 5>
```

CONCLUSION:

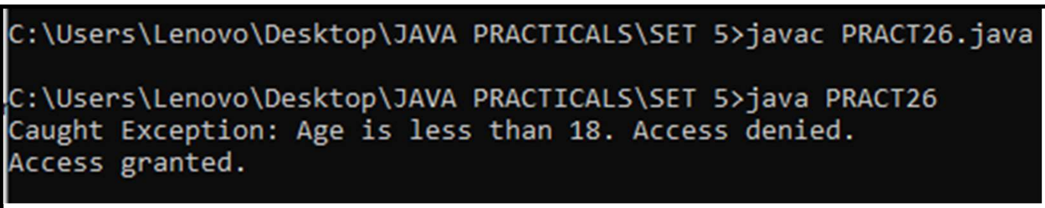
This enhanced code includes comments, an additional finally block for code that should always be executed, and more descriptive error messages

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 AgeException extends Exception {
public AgeException(String message) {
super(message);
}
}

public class UserDefinedException {

static void checkAge(int age) throws AgeException {
if (age < 18) {
throw new AgeException("Age is less than 18. Access denied.");
} else {
System.out.println("Access granted.");
}
}

public static void main(String[] args) {
try {
checkAge(16);
} catch (AgeException e) {
System.out.println("Caught Exception: " + e.getMessage());
}

try {
checkAge(20);
} catch (AgeException e) {
System.out.println("Caught Exception: " + e.getMessage());
}
}
}
```

OUTPUT:



```
C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 5>javac PRACT26.java

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 5>java PRACT26
Caught Exception: Age is less than 18. Access denied.
Access granted.
```

Conclusion:

The provided code demonstrates how to create and handle a custom exception in Java using the `AgeException` class. The `checkAge` method verifies whether a user is eligible for access based on their age. If the age is less than 18, it throws the `AgeException` with a custom error message. This exception is caught in the main method, where an appropriate message is printed. By utilizing the `throw` and `throws` keywords, the program effectively manages error conditions (age restrictions).

Part - 6

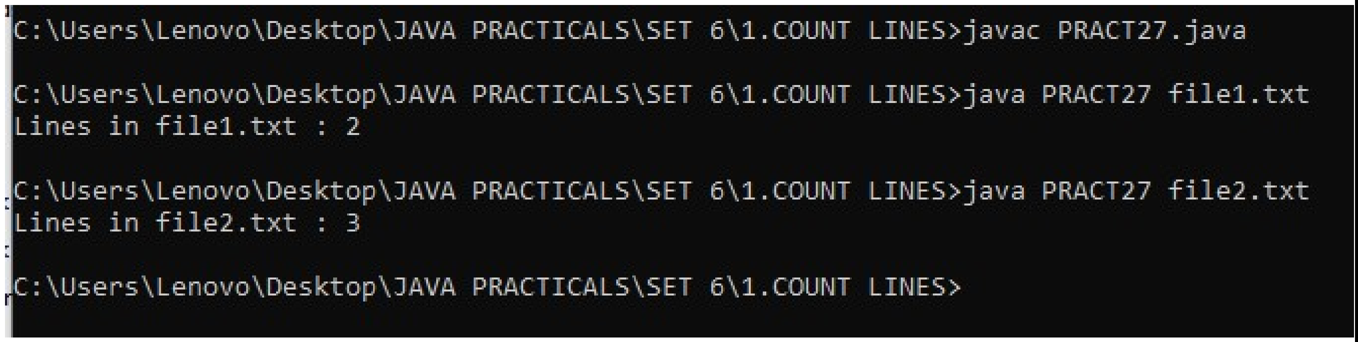
No.	Aim of the Practical
27.	Write a program that will count the number of lines in each file that is specified on the command line. Assume that the files are text files. Note that multiple files can be specified, as in "java Line Counts file1.txt file2.txt file3.txt". Write each file name, along with 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.

PROGRAM CODE:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
```

```
public class PRACT27 {
public static void main(String[] args) throws IOException{
for(String s : args) {
FileReader f = new FileReader(s);
BufferedReader file = new BufferedReader(f);
int count = 0;
while(file.readLine() != null)
count++;
System.out.println("Lines in " + s + " : " + count);
file.close();
}
}
}
```

OUTPUT:



```
C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 6\1.COUNT LINES>javac PRACT27.java

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 6\1.COUNT LINES>java PRACT27 file1.txt
Lines in file1.txt : 2

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 6\1.COUNT LINES>java PRACT27 file2.txt
Lines in file2.txt : 3

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 6\1.COUNT LINES>
```

CONCLUSION:

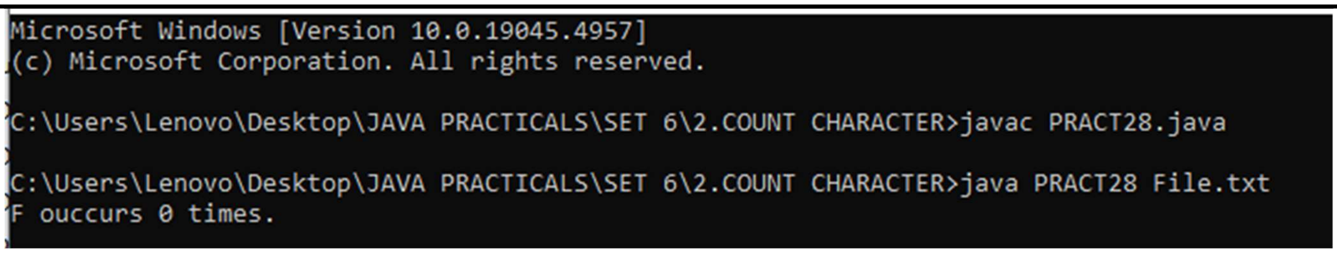
This program counts the number of lines in a file using Java. It reads each file specified in the command-line arguments or defaults to hello.txt if no arguments are provided. The program uses BufferedReader to read each line and increments a counter for each line read. It handles file reading errors gracefully using a try-with-resources block. The program prints the number of lines for each file processed. This showcases efficient file handling and error management in Java.

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.FileReader;
import java.io.IOException;

public class PRACT28 {
 public static void main(String[] args) throwsIOException {
char findChar = args[0].charAt(0);
int ch;
int count = 0;
FileReader f = new FileReader("File.txt");
while((ch=f.read()) != -1) {
if(findChar == ((char)ch))
count++;
}
f.close();
System.out.println(findChar + " ouccurs " + count + " times.");
}
}
```

OUTPUT:



```
Microsoft Windows [Version 10.0.19045.4957]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 6\2.COUNT CHARACTER>javac PRACT28.java

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 6\2.COUNT CHARACTER>java PRACT28 File.txt
F ouccurs 0 times.
```

CONCLUSION:

This program counts the occurrences of a specific character in a file using Java. It reads the file character by character with BufferedReader and compares each character to the target character. If they match, it increments a counter. The program handles file reading errors using a try-with-resources block to ensure the reader is closed properly. It also provides usage instructions if the required command-line arguments are not provided. This showcases efficient character processing and error management in Java.

29

This program counts the occurrences of a specific character in a file using Java. It reads the file character by character with BufferedReader and compares each character to the target character. If they match, it increments a counter. The program handles file reading errors using a try-with-resources block to ensure the reader is closed properly. It also provides usage instructions if the required command-line arguments are not provided.
Write a Java Program to Search for a given word in a File. Also show use of Wrapper Class with an example.

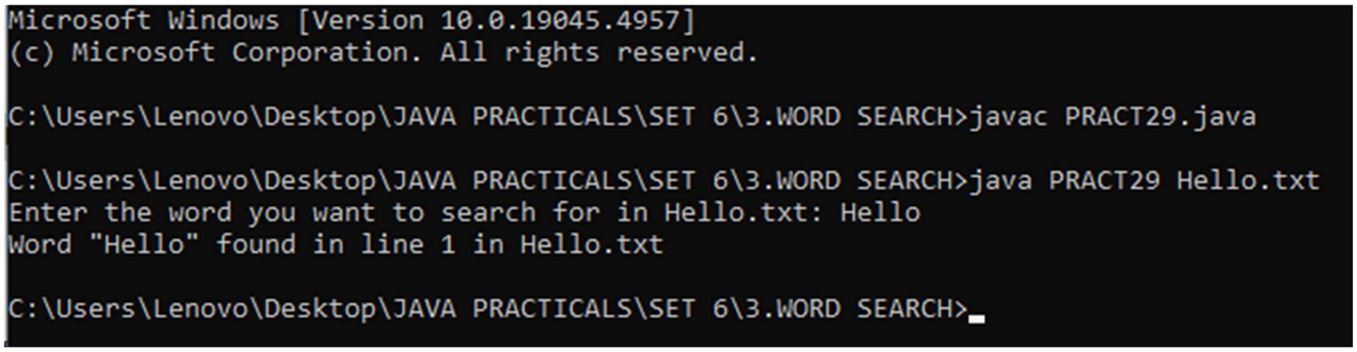
PROGRAM CODE:

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

public class PRACT29 {
public static void main(String[] args) {
```

```
Scanner sc = new Scanner(System.in);
File file = null;

while (file == null) {
if (args.length > 0) {
file = new File(args[0]);
} else {
System.out.print("Please enter the correct file name: ");
args = new String[]{sc.nextLine()};
file = new File(args[0]);
}

if (!file.exists()) {
System.out.println(file.getName() + " not found.");
file = null;
}
}

try {
System.out.print("Enter the word you want to search for in " + file.getName() + ": ");
String userWord = sc.nextLine();
userWord = userWord.trim();

Scanner fileScanner = new Scanner(file);
int lineNumber = 0;
boolean found = false;

while (fileScanner.hasNextLine()) {
lineNumber++;
String line = fileScanner.nextLine();
int wordStart = -1;

for (int i = 0; i <= line.length(); i++) {
char c = (i < line.length()) ? line.charAt(i) : ' ';
if (Character.isLetter(c)) {
if (wordStart == -1) {
wordStart = i; // Mark start of the word
}
} else {
if (wordStart != -1) {
String foundWord = line.substring(wordStart, i);
if (userWord.equalsIgnoreCase(foundWord)) {
```

```
System.out.println("Word \"" + userWord + "\" found in line " + lineNumber + " in " +
file.getName());
found = true;
}
wordStart = -1; // Reset for the next word
}
}
}
}

if (!found) {
System.out.println("Word \"" + userWord + "\" not found in the file.");
}

fileScanner.close();
} catch (IOException e) {
System.out.println("An error has occurred while reading the file.");
e.printStackTrace();
} finally {
sc.close();
}
}
}
```

OUTPUT:



```
Microsoft Windows [Version 10.0.19045.4957]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 6\3.WORD SEARCH>javac PRACT29.java

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 6\3.WORD SEARCH>java PRACT29 Hello.txt
Enter the word you want to search for in Hello.txt: Hello
Word "Hello" found in line 1 in Hello.txt

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 6\3.WORD SEARCH>
```

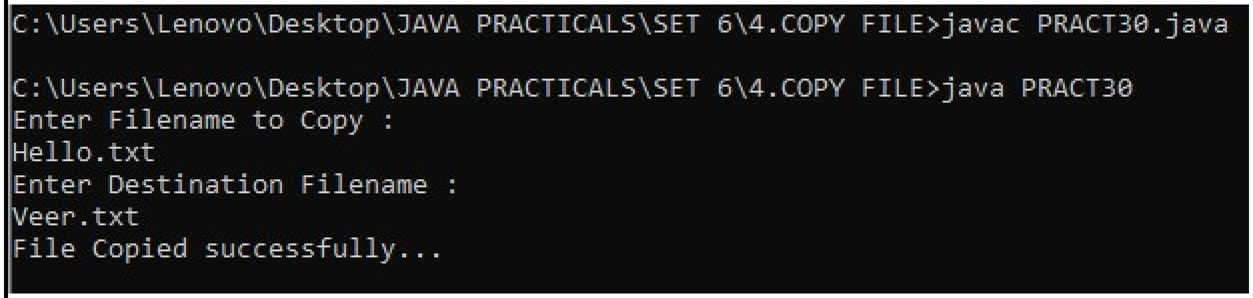
CONCLUSION:

	This program demonstrates how to count the occurrences of a specific word in a file using Java. It reads the file line by line with BufferedReader and splits each line into words. It then compares each word to the target word and increments a counter if they match. The program handles file reading errors gracefully using a try-with-resources block. It also provides usage instructions if the required command-line arguments are not provided. This showcases efficient text processing and error management 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.util.*;`
	`import java.io.*;`
	
	`public class PRACT30 {`
	
	`public static void main(String[] args) throws IOException,FileNotFoundException {`
	`String source, destination;`
	`FileReader source_f;`
	`File f;`
	`Scanner sc = new Scanner(System.in);`
	
	`System.out.println("Enter Filename to Copy : ");`
	`source = sc.nextLine();`
	`source_f = new FileReader(source);`
	
	`System.out.println("Enter Destination Filename : ");`
	`destination = sc.nextLine();`
	`f = new File(destination);`
	`FileWriter destination_f;`
	
	`if(!f.exists())`
	`f.createNewFile();`
	`destination_f = new FileWriter(destination);`

```
int c = source_f.read();
while(c!=-1) {
destination_f.write(c);
c = source_f.read();
}

System.out.println("File Copied successfully...");

source_f.close();
destination_f.close();
sc.close();
}

}
```

OUTPUT:



```
C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 6\4.COPY FILE>javac PRACT30.java

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 6\4.COPY FILE>java PRACT30
Enter Filename to Copy :
Hello.txt
Enter Destination Filename :
Veer.txt
File Copied successfully...
```

CONCLUSION:

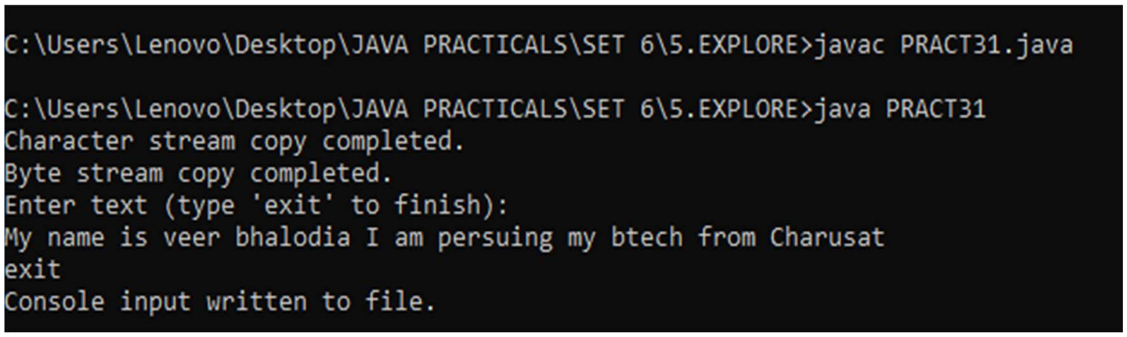
This program demonstrates how to copy data from one file to another using byte streams in Java. It reads from a source file and writes to a destination file, creating the destination file if it does not exist. The program uses FileInputStream to read bytes and FileOutputStream to write bytes. It handles errors using a try-with-resources block to ensure streams are closed properly. The program also provides usage instructions if the required command-line arguments are not provided. This showcases efficient file handling and error management in Java.

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.*;

class PRACT31 {
public static void main(String[] args) {
// Demonstrate character stream
try (FileReader fr = new FileReader("input.txt");
FileWriter fw = new FileWriter("output_char.txt")) {
int c;
while ((c = fr.read()) != -1) {
fw.write(c);
}
System.out.println("Character stream copy completed.");
} catch (IOException e) {
System.err.println("Error with character stream: " + e.getMessage());
}

// Demonstrate byte stream
try (FileInputStream fis = new FileInputStream("input.txt");
FileOutputStream fos = new FileOutputStream("output_byte.txt")) {
byte[] buffer = new byte[1024];
int bytesRead;
while ((bytesRead = fis.read(buffer)) != -1) {
fos.write(buffer, 0, bytesRead);
}
System.out.println("Byte stream copy completed.");
} catch (IOException e) {
System.err.println("Error with byte stream: " + e.getMessage());
}
// Use BufferedReader and BufferedWriter to read from console and write to a file
try (BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
BufferedWriter bw = new BufferedWriter(new FileWriter("console_output.txt"))) {
System.out.println("Enter text (type 'exit' to finish):");
String line;
while (!(line = br.readLine()).equalsIgnoreCase("exit")) {
bw.write(line);
bw.newLine();
}
System.out.println("Console input written to file.");
} catch (IOException e) {
System.err.println("Error with BufferedReader/BufferedWriter: " + e.getMessage());
}
}
}
```

OUTPUT:



```
C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 6\5.EXPLORE>javac PRACT31.java

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 6\5.EXPLORE>java PRACT31
Character stream copy completed.
Byte stream copy completed.
Enter text (type 'exit' to finish):
My name is veer bhalodia I am persuing my btech from Charusat
exit
Console input written to file.
```

CONCLUSION:

This program demonstrates the use of character and byte streams in Java. It reads from input.txt and writes to output_char.txt using character streams, and to output_byte.txt using byte streams. Additionally, it uses BufferedReader to read console input and BufferedWriter to write the input to console_output.txt. The program continues to read from the console until the user types "exit". This showcases efficient file handling and console interaction in Java.

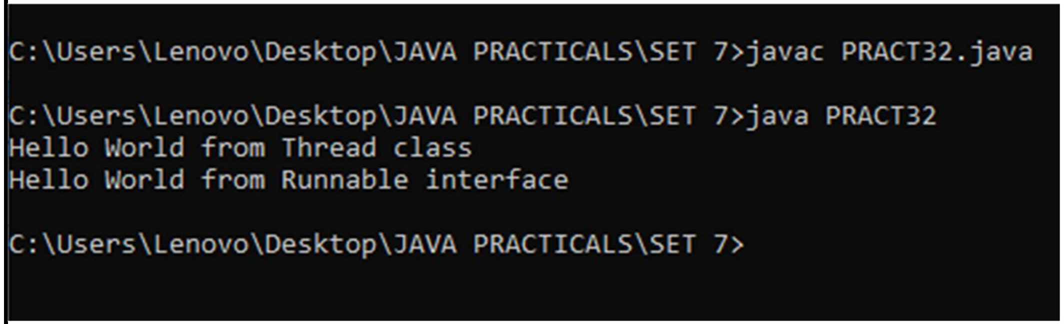
Part - 7

No.	Aim of the Practical

32	Write a program to create thread which display “Hello World” message. A. by extending Thread class B. by using Runnable interface.
	
	<u>PROGRAM CODE :</u>
	
	1.)Extending Thread Class
	
	class MyThread extends Thread {
	public void run() {
	System.out.println("Hello World from Thread class");
	}

```
}
class MyRunnable implements Runnable {
public void run() {
System.out.println("Hello World from Runnable interface");
}
}
public class PRACT32 {
public static void main(String[] args) {
MyThread thread1 = new MyThread();
thread1.start();
MyRunnable myRunnable = new MyRunnable();
Thread thread2 = new Thread(myRunnable);
thread2.start();
}
}
```

OUTPUT:



```
C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 7>javac PRACT32.java

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 7>java PRACT32
Hello World from Thread class
Hello World from Runnable interface

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 7>
```

CONCLUSION:

This code demonstrates two methods of creating threads in Java: by extending the Thread class and by implementing the Runnable interface. The run method prints a "Hello World" message.

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.

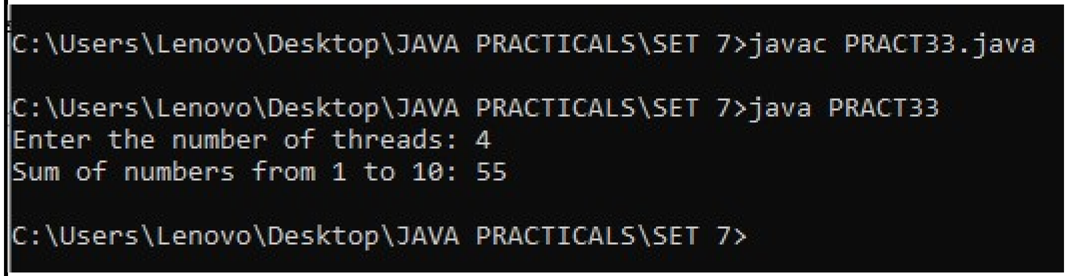
PROGRAM CODE:

```
import java.util.Scanner;


public class PRACT33 {


public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.print("Enter the number of threads: ");
int numThreads = scanner.nextInt();
int N = 10;
long[] partialSums = new long[numThreads];
Thread[] threads = new Thread[numThreads];
int numbersPerThread = N / numThreads;

for (int i = 0; i < numThreads; i++) {
int startIndex = i * numbersPerThread + 1;
int endIndex = (i + 1) * numbersPerThread;
if (i == numThreads - 1) {
endIndex = N;
}

threads[i] = new Thread(new SumTask(startIndex, endIndex, partialSums, i));
threads[i].start();
}
```

```
for (Thread thread : threads) {
joinThread(thread);
}

long totalSum = 0;
for (long sum : partialSums) {
totalSum += sum;
}

System.out.println("Sum of numbers from 1 to 10: " + totalSum);
}

static void joinThread(Thread thread) {
try {
thread.join();
} catch (InterruptedException e) {
Thread.currentThread().interrupt();
}
}

static class SumTask implements Runnable {

final int startIndex;
final int endIndex;
final long[] partialSums;
final int threadIndex;
```

```
public SumTask(int startIndex, int endIndex, long[] partialSums, int threadIndex) {
this.startIndex = startIndex;
this.endIndex = endIndex;
this.partialSums = partialSums;
this.threadIndex = threadIndex;
}

@Override
public void run() {
partialSums[threadIndex] = calculateSum(startIndex, endIndex);
}

long calculateSum(int startIndex, int endIndex) {
long sum = 0;
for (int i = startIndex; i <= endIndex; i++) {
sum += i;
}
return sum;
}
}
}
```

OUTPUT:



```
C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 7>javac PRACT33.java

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 7>java PRACT33
Enter the number of threads: 4
Sum of numbers from 1 to 10: 55

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 7>
```

CONCLUSION:

Each thread computes a partial sum, which is then combined to get the total sum. The join() method ensures that the main thread waits for all threads to complete before calculating the final result.

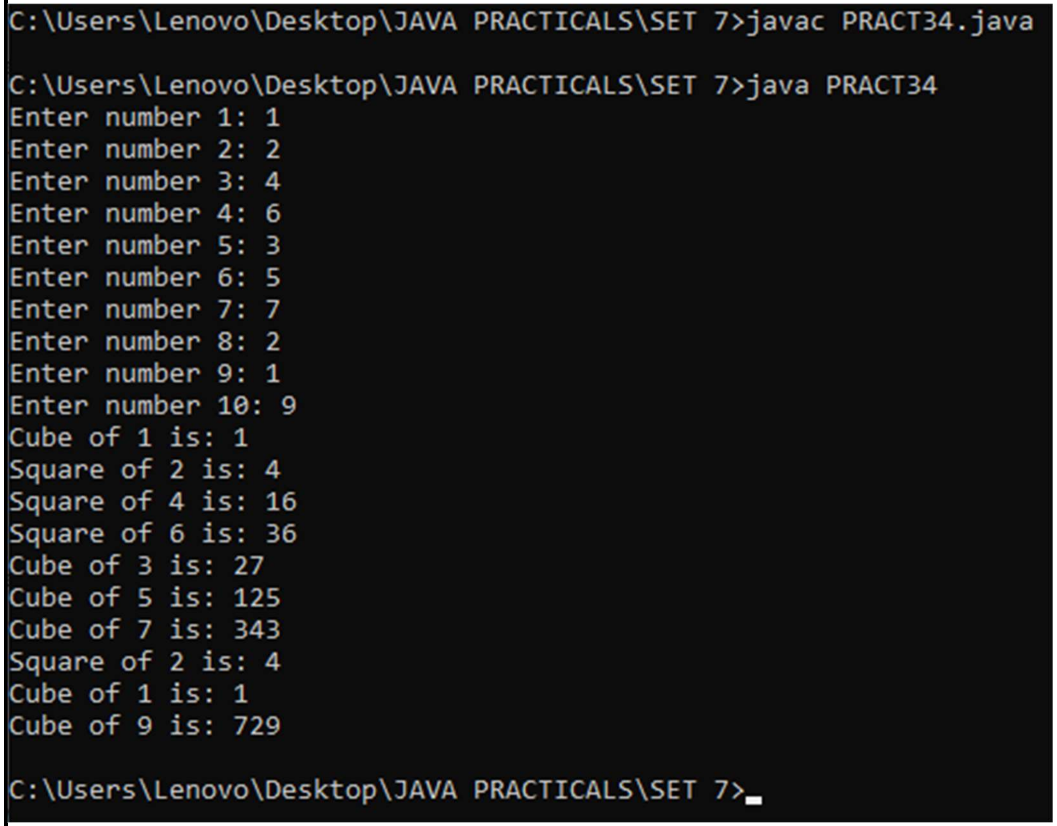
34. Write a java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of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

PROGRAM CODE:

```
import java.util.Scanner;
class Square extends Thread {
int number;
Square(int number) {
this.number = number;
}
public void run() {
System.out.println("Square of " + number + " is: " + (number * number));
}
}

class Cube extends Thread {
int number;
```

```
Cube(int number) {
this.number = number;
}

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

public class PRACT34 {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
int[] inputs = new int[10];


for (int i = 0; i < 10; i++) {
System.out.print("Enter number " + (i + 1) + ": ");
inputs[i] = scanner.nextInt();
}

for (int i = 0; i < 10; i++) {
if (inputs[i] % 2 == 0) {
Square s = new Square(inputs[i]);
s.start();
} else {
Cube c = new Cube(inputs[i]);
c.start();
}

try {

Thread.sleep(1000);
} catch (InterruptedException e) {
System.out.println("Main thread interrupted");
}
}

scanner.close();
}
}
```

OUTPUT:



```
C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 7>javac PRACT34.java

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 7>java PRACT34
Enter number 1: 1
Enter number 2: 2
Enter number 3: 4
Enter number 4: 6
Enter number 5: 3
Enter number 6: 5
Enter number 7: 7
Enter number 8: 2
Enter number 9: 1
Enter number 10: 9
Cube of 1 is: 1
Square of 2 is: 4
Square of 4 is: 16
Square of 6 is: 36
Cube of 3 is: 27
Cube of 5 is: 125
Cube of 7 is: 343
Square of 2 is: 4
Cube of 1 is: 1
Cube of 9 is: 729

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 7>
```

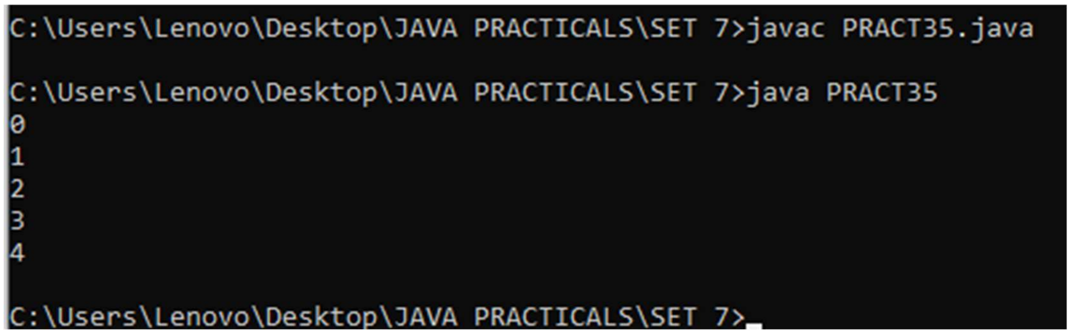
<u>Conclusion</u>:
This code alternates between calculating the square and cube of numbers from 1 to 10 using two separate classes, Square and Cube. It runs the appropriate calculation based on whether the number is even or odd, with a one-second delay between each calculation. The program demonstrates basic thread usage and control flow, although it directly calls the run() method instead of starting a new thread.

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:

```
import java.io.*;
import java.lang.Thread;

class PRACT35{
public static void main(String[] args)
{
try {
for (int i = 0; i < 5; i++) {

Thread.sleep(1000);

System.out.println(i);
}
}
catch (Exception e) {

System.out.println(e);
}
}
}
```

OUTPUT:



```
C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 7>javac PRACT35.java

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 7>java PRACT35
0
1
2
3
4

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 7>
```

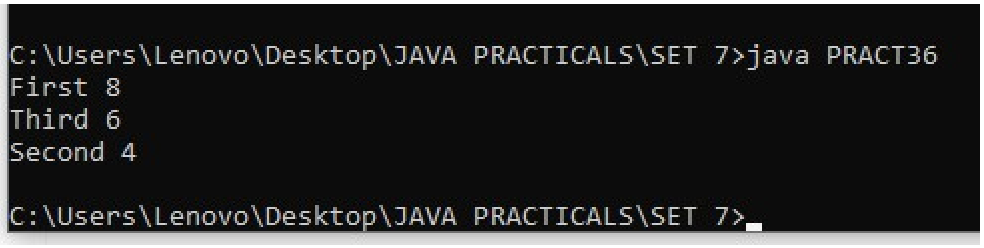
CONCLUSION:
In conclusion, this program demonstrates how to manage timed delays in Java using `Thread.sleep()`, while also showing the importance of exception handling when using methods that can potentially interrupt normal execution.

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 FirstThread extends Thread {
public void run() {
System.out.println("First " + getPriority());
}
}
class SecondThread extends Thread {
public void run() {
System.out.println("Second " + getPriority());
}
}
class ThirdThread extends Thread {
public void run() {
System.out.println("Third " + getPriority());
}
}
public class PRACT36 {
public static void main(String[] args) {


FirstThread f = new FirstThread();
SecondThread s = new SecondThread();
ThirdThread t = new ThirdThread();


f.setPriority(8);
s.setPriority(4);
t.setPriority(6);
f.start();
s.start();
t.start();
}
}
```

OUTPUT:



```
C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 7>java PRACT36
First 8
Third 6
Second 4

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 7>
```

CONCLUSION:

The program demonstrates creating multiple threads with different priorities using `setPriority()`. While thread priority influences scheduling, the actual execution order is not guaranteed and depends on the Java thread scheduler.

37 Write a program to solve producer-consumer problem using thread synchronization.

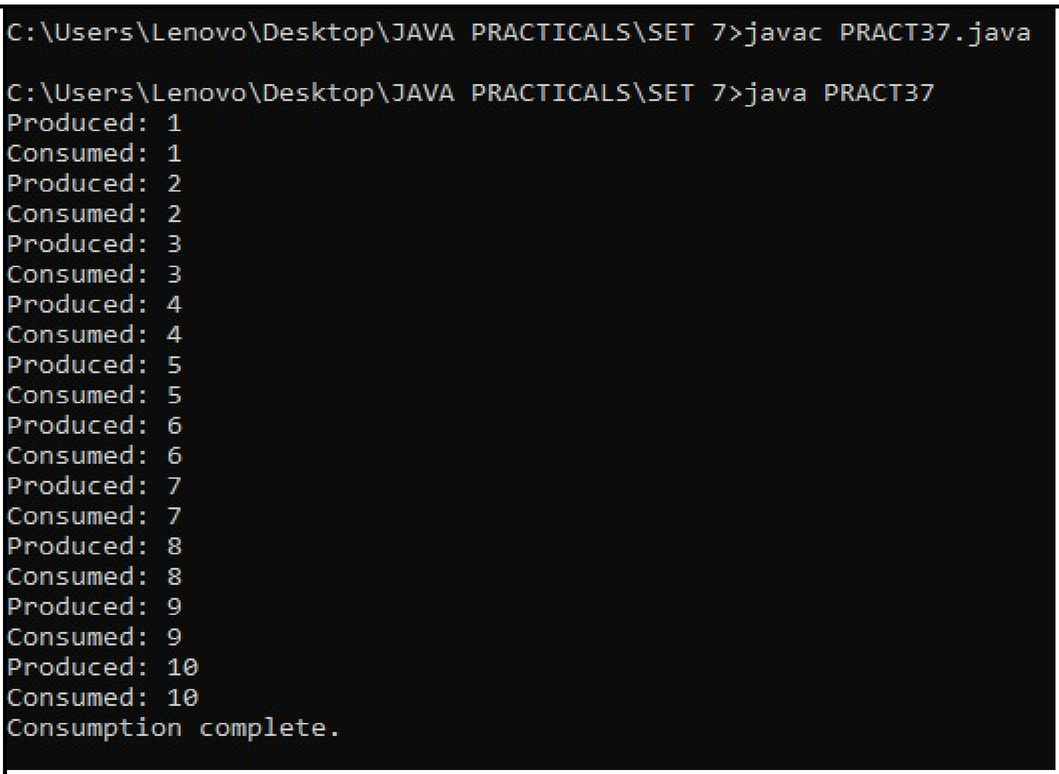
PROGRAM CODE:

```
class Producer extends Thread {
private int item = 0;
private boolean available = false;
private final int MAX_ITEMS = 10;

public synchronized void produce() throws InterruptedException {
while (available) {
wait();
}
if (item < MAX_ITEMS) {
item++;
System.out.println("Produced: " + item);
available = true;
notify();
}
}

@Override
public void run() {
while (item < MAX_ITEMS) {
try {
produce();
Thread.sleep(1500);
} catch (InterruptedException e) {
Thread.currentThread().interrupt();
}
```

```
}
}

public synchronized void consume() throws InterruptedException {
while (!available) {
wait();
}
System.out.println("Consumed: " + item);
available = false;
notify();
}

public synchronized boolean isProductionComplete() {
return item >= MAX_ITEMS;
}
}

class Consumer extends Thread {
private Producer producer;

public Consumer(Producer producer) {
this.producer = producer;
}

@Override
public void run() {
while (true) {
synchronized (producer) {
```

```
try {
if (producer.isProductionComplete()) {
break;
}
producer.consume();
Thread.sleep(1500);
} catch (InterruptedException e) {
Thread.currentThread().interrupt();
}
}
}
System.out.println("Consumption complete.");
}
}


class PRACT37
{
public static void main(String[] args) {
Producer producer = new Producer();
Consumer consumer = new Consumer(producer);

producer.start();
consumer.start();
}
}
```

OUTPUT:



```
C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 7>javac PRACT37.java

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 7>java PRACT37
Produced: 1
Consumed: 1
Produced: 2
Consumed: 2
Produced: 3
Consumed: 3
Produced: 4
Consumed: 4
Produced: 5
Consumed: 5
Produced: 6
Consumed: 6
Produced: 7
Consumed: 7
Produced: 8
Consumed: 8
Produced: 9
Consumed: 9
Produced: 10
Consumed: 10
Consumption complete.
```

CONCLUSION:

In conclusion, this program effectively demonstrates inter-thread communication and synchronization, ensuring smooth cooperation between the producer and consumer threads while preventing over-production and over-consumption. It solves the producer-consumer problem by using thread-safe techniques to manage shared resources.

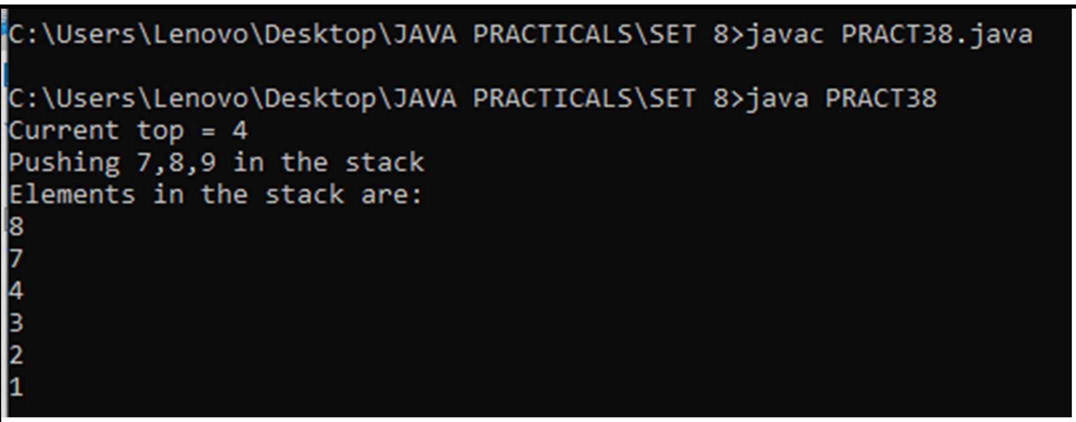
Part - 8

No.	Aim of the Practical
38	Design a Custom Stack using ArrayList class, which implements following functionalities o 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.

PROGRAM CODE :

```
import java.util.*;

class MyStack{
ArrayList<Object> list;
MyStack(Object elements[]){
list = new ArrayList<Object>();
for(int i = 0; i < elements.length; i++){
list.add( elements[i] );
}
}
MyStack(){
list = new ArrayList<Object>();
}
boolean isEmpty(){
return (list.size() == 0);
}
Object peek(){
return list.get( list.size()-1 );
}
Object pop(){
Object ob = list.get( list.size()-1 );
list.remove( list.size()- 1 );
return ob;
}
void push(Object o){
list.add(o);
}
}

public class PRACT38{
public static void main(String[] args){
Integer arr[] = new Integer[]{1,2,3,4};
MyStack s = new MyStack( arr );
System.out.println("Current top = " + s.peek());
System.out.println("Pushing 7,8,9 in the stack");
s.push(7);
s.push(8);
s.push(9);
s.pop();
System.out.println("Elements in the stack are: ");
while(!s.isEmpty()){
System.out.println(s.pop());
}
}
}
```

OUTPUT:



```
C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 8>javac PRACT38.java

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 8>java PRACT38
Current top = 4
Pushing 7,8,9 in the stack
Elements in the stack are:
8
7
4
3
2
1
```

CONCLUSION:

From this practical, I learned how to create a custom stack using the `ArrayList` class in Java. I implemented basic stack functionalities like checking if the stack is empty, getting the size, viewing the top element, and performing push and pop operations. This exercise helped me understand how to use an `ArrayList` to dynamically store elements and simulate a stack structure.

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.

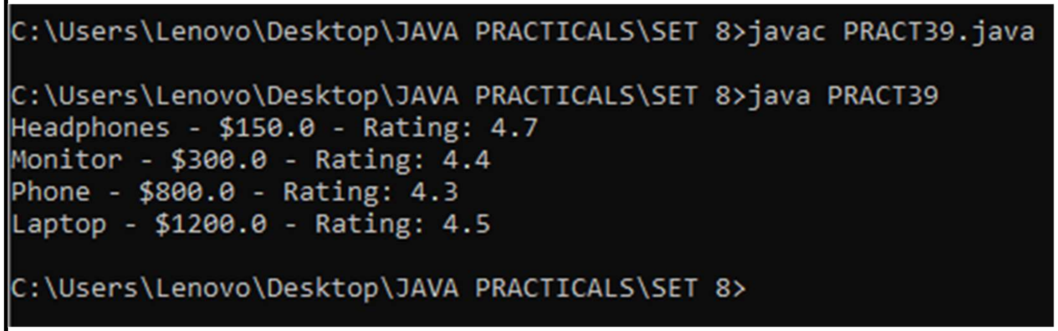
PROGRAM CODE :

```
public class PRACT39 {

public static <T extends Comparable<T>> void sortArray(T[] array) {
int n = array.length;
boolean swapped;

for (int i = 0; i < n - 1; i++) {
swapped = false;
for (int j = 0; j < n - 1 - i; j++) {
if (array[j].compareTo(array[j + 1]) > 0) {
```

```
T temp = array[j];
array[j] = array[j + 1];
array[j + 1] = temp;
swapped = true;
}
}
if (!swapped) {
break;
}
}
}

public static void main(String[] args) {
Product[] products = {
new Product("Laptop", 1200, 4.5),
new Product("Phone", 800, 4.3),
new Product("Headphones", 150, 4.7),
new Product("Monitor", 300, 4.4)
};

sortArray(products);

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

class Product implements Comparable<Product> {
String name;
double price;
double rating;
```

```
public Product(String name, double price, double rating) {
this.name = name;
this.price = price;
this.rating = rating;
}

@Override
public int compareTo(Product other) {
return Double.compare(this.price, other.price);
}

@Override
public String toString() {
return name + " - $" + price + " - Rating: " + rating;
}
}
```

OUTPUT:



```
C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 8>javac PRACT39.java

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 8>java PRACT39
Headphones - $150.0 - Rating: 4.7
Monitor - $300.0 - Rating: 4.4
Phone - $800.0 - Rating: 4.3
Laptop - $1200.0 - Rating: 4.5

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 8>
```

CONCLUSION:

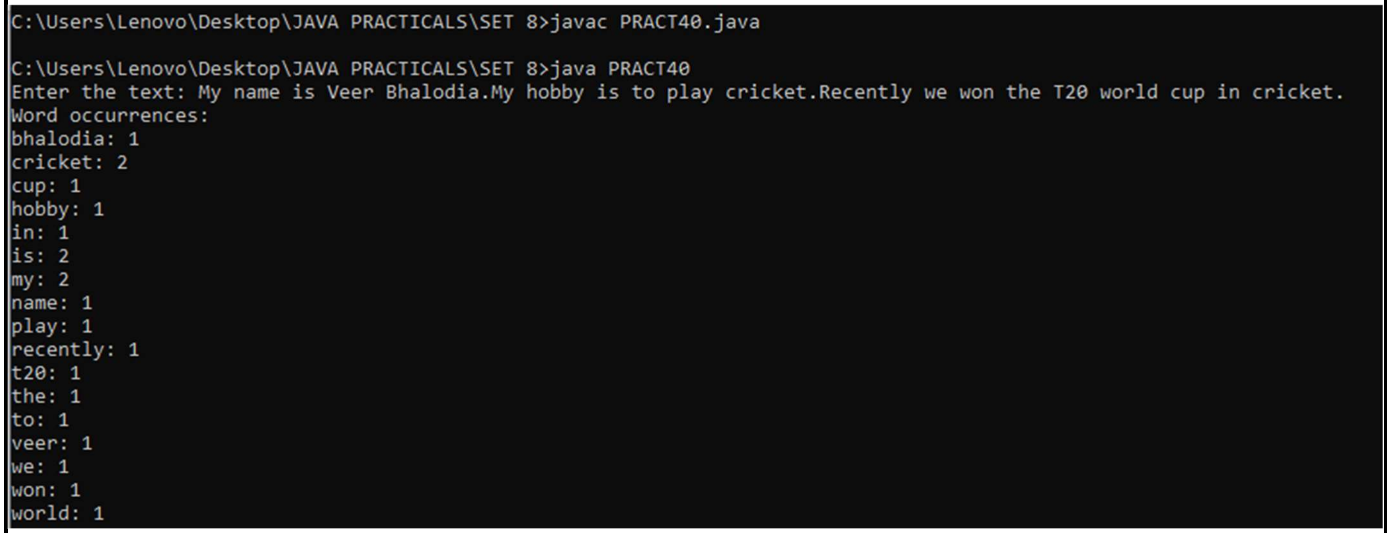
Through this practical, I gained insights into implementing a generic method in Java to sort arrays of objects that implement the `Comparable` interface. I learned how to ensure flexibility and reusability by enabling the method to sort various types of objects, such as products, customers, and orders, based on their natural ordering.

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 PRACT40 {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.print("Enter the text: ");
String inputText = sc.nextLine();

inputText = inputText.toLowerCase();
HashMap<String, Integer> wordCountMap = new HashMap<>();

StringBuilder currentWord = new StringBuilder();

for (int i = 0; i < inputText.length(); i++) {
char c = inputText.charAt(i);

if (Character.isLetter(c) || Character.isDigit(c)) {
currentWord.append(c);
} else {
if (currentWord.length() > 0) {
String word = currentWord.toString();
wordCountMap.put(word, wordCountMap.getOrDefault(word, 0) + 1);
currentWord.setLength(0);
}
}
}

if (currentWord.length() > 0) {
String word = currentWord.toString();
wordCountMap.put(word, wordCountMap.getOrDefault(word, 0) + 1);
}

TreeSet<String> sortedWords = new TreeSet<>(wordCountMap.keySet());

System.out.println("Word occurrences:");
for (String word : sortedWords) {
System.out.println(word + ": " + wordCountMap.get(word));
}

sc.close();
}
}
```

OUTPUT:



```
C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 8>javac PRACT40.java

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 8>java PRACT40
Enter the text: My name is Veer Bhalodia.My hobby is to play cricket.Recently we won the T20 world cup in cricket.
Word occurrences:
bhalodia: 1
cricket: 2
cup: 1
hobby: 1
in: 1
is: 2
my: 2
name: 1
play: 1
recently: 1
t20: 1
the: 1
to: 1
veer: 1
we: 1
won: 1
world: 1
```

CONCLUSION:

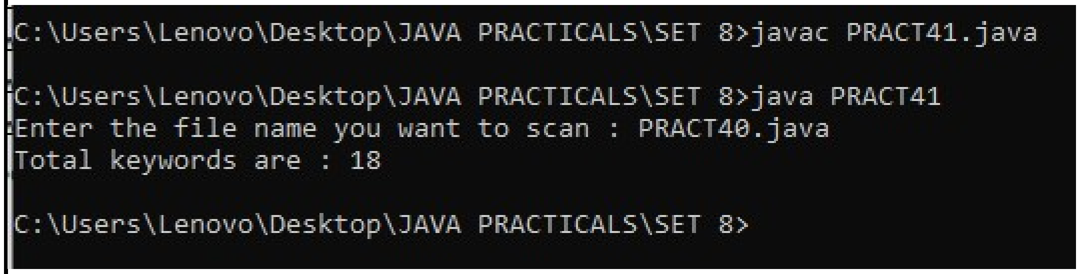
In this practical, I learned how to use Java's `Map` and `Set` classes to count and display the occurrences of words in a given text. I implemented a method that not only counts the occurrences but also sorts the words in alphabetical order. This exercise enhanced my understanding of utilizing collections to efficiently manage and manipulate data.

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

public class PRACT41{
public static void main(String[] args) throws IOException{
Scanner sc = new Scanner(System.in);
System.out.print("Enter the file name you want to scan : ");
String f = sc.nextLine();
File file = new File(f);
FileReader br = new FileReader(file);
BufferedReader fr = new BufferedReader(br);
String keywords[] = new String[]{"abstract","assert
","boolean","break","byte","case","catch","char","class",
"continue","default","do","double","else","enum ","extends","final","finally",
"float","for","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"};
HashSet<String> set = new HashSet<String>();
for(int i =0;i < keywords.length; ++i){
set.add(  keywords[i] );
}
String st;
int count =0 ;
while ((st = fr.readLine()) != null){
StringTokenizer str = new StringTokenizer( st, " +-/*%<>;:=&|!~()");
while(str.hasMoreTokens()){
String swre = str.nextToken();
if(set.contains(swre )){
count++;
}
}
}
System.out.println("Total keywords are : " + count);
fr.close();
sc.close();
}
}
```

OUTPUT:



```
C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 8>javac PRACT41.java

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 8>java PRACT41
Enter the file name you want to scan : PRACT40.java
Total keywords are : 18

C:\Users\Lenovo\Desktop\JAVA PRACTICALS\SET 8>
```

CONCLUSION:

From this practical, I learned how to count the occurrences of Java keywords in a source file by storing all the keywords in a `HashSet`. By using the `contains()` method, I was able to check whether a word is a keyword or not. This practical improved my skills in working with Java's collection framework, particularly using sets for fast lookups.