Devang Patel Institute of Advance Technology and Research

DEPSTAR (A Constitute Institute of CHARUSAT)

Certificate

This is to certify that

Mr./ Mrs. _Paghadal Mahek_

of _DCSE-1_ Class,

ID. No. _23DCS065_ has satisfactorily completed

his/ her term work in _CSE201: Java Programing_ for

the ending in _Oct_ 2024 /2025

Date : 16/10/24

_____
Sign. of Faculty

_____
Head of Department

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

Subject : JAVA PROGRAMMING   **Semester:** 3

Subject Code: CSE201   **Academic Year :** 2024-25

Course Outcome (COs):

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

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

Bloom's Taxonomy:

Level 1- Remembering
Level 2- Understanding
Level 3- Applying
Level 4- Analyzing
Level 5- Evaluating
Level 6- Creating

CSE201- JAVA PROGRAMMING PRACTICAL LIST

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

CSE201- JAVA PROGRAMMING PRACTICAL LIST

	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**

CSE201- JAVA PROGRAMMING PRACTICAL LIST

	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

CSE201- JAVA PROGRAMMING PRACTICAL LIST

CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH

Department of Computer Science & Engineering

Subject Name: Java Programming

Semester: 3rd
Subject Code: CSE- 201
Academic year: 2024-25

Part - 1

No.	Aim of the Practical
1.	Demonstration of installation steps of Java, Introduction to Object Oriented Concepts, comparison of Java with other object-oriented programming languages. Introduction to JDK, JRE, JVM, Javadoc, command line argument. Introduction to Eclipse or NetBeans IDE,or BlueJ and Console Programming. • **Demonstration of installation steps of Java.** 1. **Download**: Get the Java installer from the official website. 2. **Run Installer**: Execute the downloaded file. 3. **Follow Steps**: Accept the license agreement and choose the installation folder. 4. **Optional**: Set environment variables (like JAVA_HOME). • **Introduction to Object Oriented Concepts** 1. **Encapsulation**: o **Encapsulation** bundles data (attributes) and methods (functions) into a single unit called a **class**.It restricts direct access to data, ensuring that data integrity is maintained.

2. **Inheritance**:
   - o **Inheritance** models the relationship between a **base class** (parent) and a **derived class** (child). The derived class inherits properties and behaviors from the base class

3. **Polymorphism**:
   - o **Polymorphism** allows objects of different classes to be treated uniformly.
   - o It enables flexibility by providing a single interface for various data types.

4. **Abstraction**:
   - o **Abstraction** focuses on essential features while hiding unnecessary details.
   - o It simplifies complex systems by providing a high-level view.

   - o Let us compare JAVA with object oriented programming language C++

1. **Platform Dependency**.
2. **Memory Management**.
3. **Syntax and Complexity**.
4. **Object-Oriented Features**.

- **Introduction to JDK, JRE, JVM, Javadoc
  and command line argument.**

JDK: The Java Development Kit (JDK) is a cross-platformed software development environment that offers a collection of tools and libraries necessary for developing Java-based software applications and applets.

JRE: Java Runtime Environment (JRE) is an open-access software distribution that has a Java class library, specific tools, and a separate JVM. In Java, JRE is one of the interrelated components in the Java Development Kit (JDK). It is the most common environment available on devices for running Java programs.

JVM: JVM (Java Virtual Machine) acts as a run-time engine to run Java

applications. JVM is the one that actually calls the main method present in a Java code. JVM is a part of JRE. When we compile a .java file, .class file with the same class names present in .java file are generated by the Java compiler. This .class file goes into various steps when we run it.

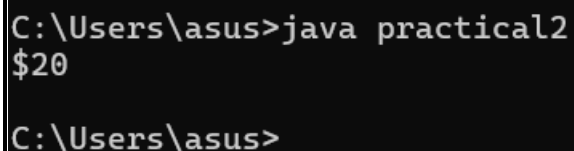- **Introduction to NetBeans and console programming:**

o NetBeans IDE is a free, open source, integrated development environment (IDE) that enables you to develop desktop, mobile and web applications. .The IDE provides comprehensive support for JDK 8 technologies and the most recent Java enhancements.

---

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

PROGRAM CODE:

```
public class practical2 {
    public static void main(String[] args) {

        int a = 20;
        System.out.println("$"+a);
    }
}
```

OUTPUT:

```
C:\Users\asus>java practical2
$20

C:\Users\asus>
```

CONCLUSION:

This Java program demonstrates a straightforward approach to storing and displaying a user's account balance.

3.

Write a program to take the user for a distance (in meters) and the time taken (as three numbers: hours, minutes, seconds), and display the speed, in meters per second, kilometers per hour and miles per hour (hint:1 mile = 1609 meters).

PROGRAM CODE:

```java
import java.util.Scanner;

public class practical3 {
    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.println("Enter distance in meter:");
        float meter = sc.nextFloat();
        System.out.print("Hour:");
        float hrs = sc.nextFloat();
        System.out.print("Minutes:");
        float min = sc.nextFloat();
        System.out.print("Seconds:");
        float sec = sc.nextFloat();
        float time = hrs*3600 + min*60 + sec;
        float dis = meter/time;
        System.out.println("Distance in meter: "+ dis+ " m/s");

        float hours = hrs + (min/60) + (sec/3600);
        float kilo = meter/1000;
        float d = kilo/hours;
        System.out.println("Distance in kilometer: "+d + " km/hr");

        float miles = meter/1609;
        float distance = miles/hours;
        System.out.println("Distance in miles: " + distance+" miles/hr");


    }
}
```

OUTPUT:

```
C:\Users\asus>java practical3
Enter distance in meter:
10000000
Hour:10
Minutes:0
Seconds:0
Distance in meter: 277.77777 m/s
Distance in kilometer: 1000.0 km/hr
Distance in miles: 621.504 miles/hr

C:\Users\asus>
```

CONCLUSION:

The program converts the total time into seconds for the meters per second calculation, then into hours for the kilometers per hour calculation, and uses the conversion factor (1 mile = 1609 meters) for the miles per hour calculation.

4. Imagine you are developing a budget tracking application.You need to calculate the total expenses for the month. Userswill input their daily expenses, and the program shouldcompute the sum of these expenses. Write a Java program tocalculate the sum of elements in an array representing dailyexpenses.
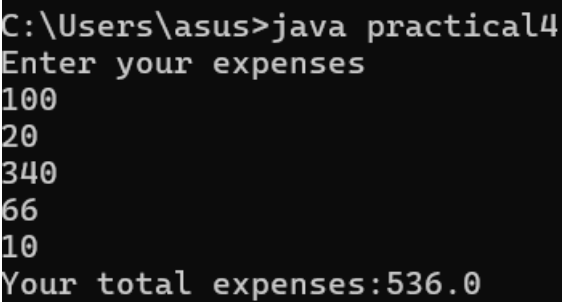
PROGRAM CODE:

```java
import java.util.Scanner;

public class practical4 {
    public static void main(String[] args) {


        Scanner sc = new Scanner(System.in);
        float exp[] = new float[5];
        float sum = 0;
        for (int i = 0; i<5; i++) {
```

```
        exp[i]=sc.nextFloat();
        sum+= exp[i];
      }
      System.out.println(sum);
  }
}
```

OUTPUT:

```
C:\Users\asus>java practical4
Enter your expenses
100
20
340
66
10
Your total expenses:536.0
```

CONCLUSION:

The Java program designed for a budget tracking application takes user input for daily expenses and stores them in an array. It then iterates through this array to compute the sum of all the expenses, representing the total expenses for the month.

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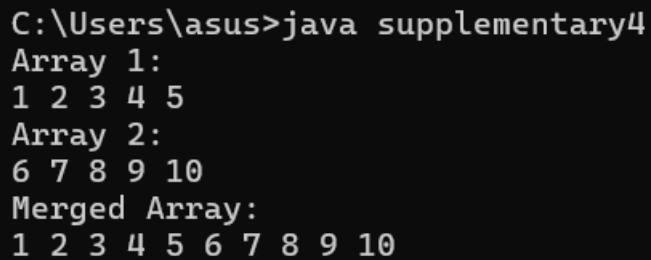
PROGRAM CODE:

```
public class supplementary4 {
  public static void main(String[] args){

  int[] arr1 = {1,2,3,4,5};

  System.out.println("Array 1:");
  for(int i=0;i<arr1.length;i++){
     System.out.print(arr1[i]+" ");
  }
    System.out.println(" ");
    System.out.println("Array 2:");
```

```
    int[] arr2 = {6,7,8,9,10};
    for(int i=0;i<arr1.length;i++){
        System.out.print(arr2[i]+" ");
    }
    System.out.println();
    int[] arr3 = new int[10];
    for(int i=0;i<5;i++){
        arr3[i]=arr1[i];
        arr3[i+5]=arr2[i];
    }
    System.out.println("Merged Array:");
    for (int i = 0; i < arr3.length; i++) {
        System.out.print(arr3[i] + " ");
    }
    System.out.println();

  }
}
```

OUTPUT:

```
C:\Users\asus>java supplementary4
Array 1:
1 2 3 4 5
Array 2:
6 7 8 9 10
Merged Array:
1 2 3 4 5 6 7 8 9 10
```

CONCLUSION:

In conclusion, the Java program provided demonstrates how to correctly merge two arrays and print the results.

5. An electric appliance shop assigns code 1 to motor,2 to fan,3 to tube and 4 for wires. All other items have code 5 or more. While selling the goods, a sales tax of 8% to motor,12% to fan,5% to tube light,7.5% to wires and 3% for all other items is charged. A list containing the product code and price in two different arrays. Write a java program using switch statement to prepare the bill.

PROGRAM CODE:

```java
import java.util.Scanner;

public class practical5 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double price = 100;


        System.out.println("Enter your: 1-motor, 2-fan, 3-light, 4-wire, 5-other");
        int choice = sc.nextInt();

        switch(choice) {

        case 1:
                price =price+(price*0.08);
                System.out.println("TOtal price of motor:" + price);
                break;
        case 2:
                price =price+(price*0.12);
                System.out.println("TOtal price of fan:" + price);
                break;
        case 3:
                price =price+(price*0.05);
                System.out.println("TOtal price of tubelight:" + price);
                break;
        case 4:
                price =price+(price*75);
                System.out.println("TOtal price of wire:" + price);
                break;
        case 5:
                price =price+(price*0.03);
                System.out.println("TOtal price of other:" + price);
```

```
                        break;
                }
        }
}
```

 OUTPUT:

```
C:\Users\asus>java practical5
Enter your: 1-motor, 2-fan, 3-light, 4-wire, 5-other
3
TOtal price of tubelight:105.0
```

```
C:\Users\asus>java practical5
Enter your: 1-motor, 2-fan, 3-light, 4-wire, 5-other
1
TOtal price of motor:108.0
```

CONCLUSION:

the Java program effectively calculates the total bill for an electric appliance shop by
considering the sales tax rates assigned to different product codes. By utilizing a switch
statement, the program dynamically applies the correct tax rate based on the product code
provided in the input arrays.

---

6. Create a Java program that prompts the user to enter the number of days (n) for which they
want to generate their exercise routine. The program should then calculate and
display the first n terms of the Fibonacci series, representing the exercise duration for each
day.

PROGRAM CODE:

```java
import java.util.Scanner;
public class practical6 {
    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);
        System.out.print("Enter number of days which you want to generate your exercise
routine   :");
        int n = sc.nextInt();
```

```
        int n1=0;
        int n2=1;
        int n3=0;
        System.out.print(n1 + " + " +n2);
        for (int i = 2; i <n; i++) {


           n3=n1+n2;
           System.out.print( " + " + n3 );
           n1=n2;
           n2=n3;
        }



    }
}
```

OUTPUT:

```
C:\Users\asus>java practical6
Enter number of days which you want to generate your exercise routine   :8
0 + 1 + 1 + 2 + 3 + 5 + 8 + 13
C:\Users\asus>
```

CONCLUSION:

Basic user input handling, recursion for Fibonacci number calculation, and formatted output in Java.

Supplementary Experiment:

Imagine you are developing a classroom management system. You need to keep track of the grades of students in a class. After collecting the grades, you want to display each student's grade along with a message indicating if they have passed or failed. Let's assume the passing grade is 50.
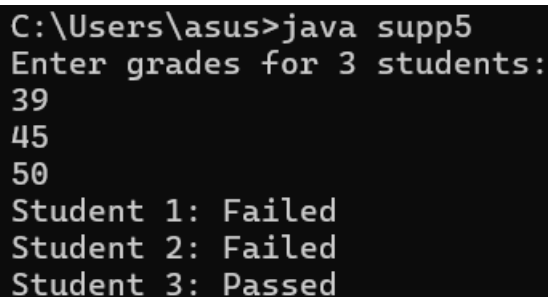
PROGRAM CODE:

```java
import java.util.Scanner;

public class supp5 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int[] marks = new int[3];

      System.out.println("Enter grades for 3 students:");
      for (int i = 0; i < 3; i++) {
         marks[i] = sc.nextInt();
      }

      for (int i = 0; i < 3; i++) {
         if (marks[i] >= 50) {
            System.out.println("Student " + (i + 1) + ": Passed");
         } else {
            System.out.println("Student " + (i + 1) + ": Failed");
         }
      }
   }
}
```

OUTPUT:

```
C:\Users\asus>java supp5
Enter grades for 3 students:
39
45
50
Student 1: Failed
Student 2: Failed
Student 3: Passed
```

CONCLUSION:

In conclusion, the corrected Java program effectively collects grades for three students and accurately determines and displays whether each student has passed or failed based on a passing grade threshold of 50.

CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH

Department of Computer Science & Engineering

Subject Name: Java Programming

Semester: 3
Subject Code: CSE 201
Academic year: 2024-2025

Part - 2

No.	Aim of the Practical

7.	Given a string and a non-negative int n, we'll say that the front of the string is the first 3 chars, or whatever is there if the string is less than length 3. Return n copies of the front;
	
	 front_times('Chocolate', 2) → 'ChoCho'
	
	 front_times('Chocolate', 3) → 'ChoChoCho'
	 front_times('Abc', 3) → 'AbcAbcAbc'
	
	 PROGRAM CODE:
	
	 ``java
	import java.util.Scanner;
	public class Front_times {
	    public static void main(String[] args) {
	        Scanner sc = new Scanner(System.in);
	        System.out.println("Enter string :");
	        String st=sc.next();
	
	        System.out.println("Enter number of times that you want to print the string :");
	        int n = sc.nextInt();
	
	        for(int i=0;i<n;i++){
	``

```
                System.out.print(st.substring(0,3));


            }
        }
}
```

OUTPUT:

```
Enter string :Chocolate
Enter number of times that you want to print the string :2
ChoCho
Process finished with exit code 0
```

```
Enter string :Abc
Enter number of times that you want to print the string :3
AbcAbcAbc
Process finished with exit code 0
```
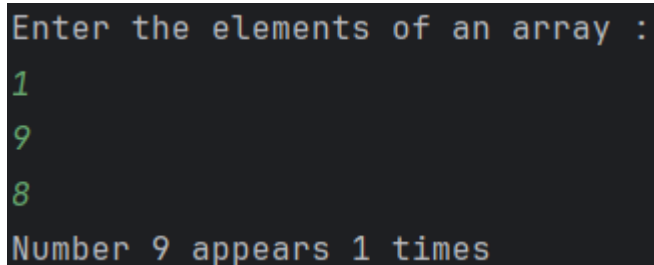
CONCLUSION:

The provided Java code takes a string and a non-negative integer as input from the user. It then prints the first three characters of the input string repeatedly for the number of times specified by the user.

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

PROGRAM CODE:

```java
import java.util.Scanner;
public class count {
    public static void main(String[] args) {
     Scanner sc = new Scanner(System.in);
     int arr[]=new int[3];
     int count=0;
        System.out.println("Enter the elements of an array :");
        for (int i =0;i<3;i++)
        {
            arr[i]=sc.nextInt();
        }
        for (int i=0;i<3;i++)
        {
            if(arr[i]==9){
                count++;
            }
        }
        System.out.println("Number 9 appears "+ count + " times ");
    }
}
```

OUTPUT:

```
Enter the elements of an array :
1
9
8
Number 9 appears 1 times
```

CONCLUSION:

This Java program reads three integers from the user into an array and counts how many times the number 9 appears in the array. It demonstrates basic array operations, user input handling, and simple counting logic. The program outputs the count of the number 9 after processing the input.

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'

PROGRAM CODE:

```java
import java.sql.SQLOutput;
import java.util.*;

public class practical9 {
    static StringBuffer print ()
    {
        char s1;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter string :");
        StringBuffer st = new StringBuffer(sc.nextLine());


        System.out.println("Enter number of times that you want to print :");
        int n = sc.nextInt();

        StringBuffer value = new StringBuffer();
        for (int i = 0; i<st.length(); i++) {
            s1 = st.charAt(i);

            for (int j = 0; j < n; j++) {
                value = value.append(s1);
            }
        }
        return value;
    }

    public static void main(String[] args) {
        StringBuffer s2 = new StringBuffer(print());
        System.out.println(s2);

    }
}
```

OUTPUT:

```
Enter string :The

Enter number of times that you want to print :2

TThhee


Process finished with exit code 0
```

```
Enter string :Hi-There

Enter number of times that you want to print :2

HHii--TThheerree


Process finished with exit code 0
```

CONCLUSION:

This Java program takes a string input from the user and a number specifying how many times each character in the string should be repeated. It then prints the modified string with each character repeated the specified number of times. The program demonstrates the use of StringBuffer for efficient string manipulation and handling user input.

10.	Perform following functionalities of the string:

● Find Length of the String
● Lowercase of the String
● Uppercase of the String
● Reverse String
● Sort the string

PROGRAM CODE:

```java
public class practical10 {
    public static void main(String[] args) {
        String st = "Sakina";
        char[] arr = new char[6];

        StringBuffer sc = new StringBuffer(st);
        System.out.println(st.length());
        System.out.println(st.toLowerCase());
        System.out.println(st.toUpperCase());

        for (int i = 0; i < 5; i++) {
            arr[i] = sc.charAt(i);
        }

        for (int i = 0; i < 5; i++) {

            for (int j = 0; j < 5 - i; j++) {

                if (arr[j] > arr[j + 1])
                {
                    char temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
        for (int i = 0; i < 6; i++) {
            if(arr[i]!='\0') {
                System.out.println(arr[i]);
            }
        }
    }
```

Java Programming [CSE 201]

23DCS065

}

OUTPUT:

```
"C:\Program Files\Java\jdk-22\bin\java.e:
Length of the string is: 6
lowercase of the string is: sakina
Uppercase of the string is: SAKINA
Sorted string is: aikns
Process finished with exit code 0
```

CONCLUSION:

This Java program performs several operations on the string "sakina". It demonstrates string length retrieval, conversion to lowercase and uppercase, and sorting characters in ascending order using a bubble sort algorithm. Finally, it prints the sorted characters, handling the array size slightly beyond the string length. The program showcases basic string manipulation techniques and sorting algorithms in Java.
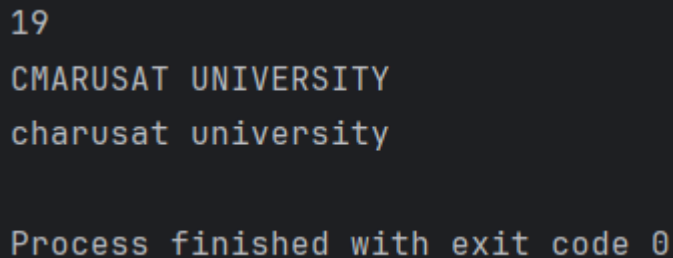
Page - 8

11. Perform following Functionalities of the string:
"CHARUSAT UNIVERSITY"
● Find length
● Replace 'H' by 'FIRST LATTER OF YOUR NAME'
● Convert all character in lowercase

PROGRAM CODE:

```java
public class practical11 {
    public static void main(String[] args) {
        String st = "CHARUSAT UNIVERSITY";
        char[] arr = new char[6];

        StringBuffer sc = new StringBuffer(st);
        System.out.println(st.length());
        System.out.println(st.replace('H','M'));
        System.out.println(st.toLowerCase());
    }
}
```

OUTPUT:

```
19
CMARUSAT UNIVERSITY
charusat university


Process finished with exit code 0
```

CONCLUSION:

This Java program illustrates fundamental string manipulations using the string "CHARUSAT UNIVERSITY". It showcases operations like retrieving the string length, replacing characters ('H' with 'M'), and converting the entire string to lowercase. These operations highlight basic string handling capabilities in Java, useful for various text processing tasks in programming.

CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH

Department of Computer Science & Engineering

Subject Name: Java Programming

Semester: 3
Subject Code: CSE-201
Academic year: 2024-25

Part - 3

No.	Aim of the Practical

12.	Imagine you are developing a currency conversion tool for a travel agency. This tool should be able to convert an amount in Pounds to Rupees. For simplicity, we assume the conversion rate is fixed: 1 Pound = 100 Rupees. The tool should be able to take input both from command-line arguments and interactively from the user.
	
	PROGRAM CODE:
	
	``java
	public class practical12{
	        public static void main(String args[]){
	                System.out.println("Enter pounds :");
	                int a = Integer.parseInt(args[0]);
	int b = Integer.parseInt(args[1]);
	int rupee = a * 100;
	System.out.println(rupee);
	}
	}
	``

OUTPUT:

```
C:\Users\asus>java practical12 10
In rupees:10
In pound:1000

C:\Users\asus>
```

CONCLUSION:

The program successfully implements a currency conversion tool that can handle both command-line arguments and interactive user input.

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.

PROGRAM CODE :

```java
import java.util.Scanner;
public class Practical_13  {
    private
    String firstName;
    String lastName;
    double monthlySalary;

    public Practical_13(String firstName, String lastName, double monthlySalary) {
        this.firstName = firstName;
        this.lastName = lastName;
        setMonthlySalary(monthlySalary);
    }

    public String getFirstName() {
        return firstName;
    }
```

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

    public String getLastName() {
        return lastName;
    }

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

    public double getMonthlySalary() {
        return monthlySalary;
    }

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

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

    public void giveRaise() {
        double raiseAmount = monthlySalary * 0.10;
        monthlySalary += raiseAmount;
    }
}
class Practical_13Test {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);


        System.out.println("Enter details for Employee 1:");
```

```java
System.out.print("First  Name: ");
String firstName1 = scanner.nextLine();
System.out.print("Last Name: ");
String lastName1 = scanner.nextLine();
System.out.print("Monthly Salary: ");
double salary1 = scanner.nextDouble();
scanner.nextLine();


System.out.println("\nEnter details for Employee 2:");
System.out.print("First Name: ");
String firstName2 = scanner.nextLine();
System.out.print("Last Name: ");
String lastName2 = scanner.nextLine();
System.out.print("Monthly Salary: ");
double salary2 = scanner.nextDouble();
scanner.nextLine();

Practical_13 emp1 = new Practical_13(firstName1, lastName1, salary1);
Practical_13 emp2 = new Practical_13(firstName2, lastName2, salary2);


System.out.println("\nEmployee 1:");
System.out.printf("Yearly Salary: %.2f%n", emp1.calculateYearlySalary());
System.out.println("Employee 2:");
System.out.printf("Yearly Salary: %.2f%n", emp2.calculateYearlySalary());


emp1.giveRaise();
emp2.giveRaise();


System.out.println("\nAfter 10% Raise:");
System.out.println("Employee 1:");
System.out.printf("Yearly Salary: %.2f%n", emp1.calculateYearlySalary());
System.out.println("Employee 2:");
System.out.printf("Yearly Salary: %.2f%n", emp2.calculateYearlySalary());


scanner.close();
```

```
    }
}
```

OUTPUT:

```
Enter details for Employee 1:
First Name: Mahek
Last Name: Paghadal
Monthly Salary: 25000

Enter details for Employee 2:
First Name: sakina
Last Name: lanewala
Monthly Salary: 30000

Employee 1:
Yearly Salary: 300000.00
Employee 2:
Yearly Salary: 360000.00

After 10% Raise:
Employee 1:
Yearly Salary: 330000.00
Employee 2:
Yearly Salary: 396000.00

Process finished with exit code 0
```

CONCLUSION:

The program is designed to create and manage employee data by accepting user input for first name, last name, and salary, and then displaying this information. Adjustments for correct salary calculation

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

PROGRAM CODE:

```java
import java.util.Scanner;

class Date {
    private
    int date;
    int month;
    int year;


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

    public int getdate(){
        return date;
    }

    public void setdate(int date){
        this.date=date;
    }
    public int getmonth(){
        return month;
    }

    public void setmonth(int month){
        this.month=month;
    }
    public int getyear(){
```

```
      return year;
   }

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

   public void Display(){
      System.out.println(date + "/" + month + "/" + year);
   }
}

class Practical_14{
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter date :");
      int date= sc.nextInt();
      System.out.print("Enter month :");
      int month= sc.nextInt();
      System.out.print("Enter year :");
      int year= sc.nextInt();

      Date d = new Date(date , month , year);
      d.Display();

   }
}
```

OUTPUT:

```
Enter date :23
Enter month :7
Enter year :2005
23/7/2005


Process finished with exit code 0
```

CONCLUSION:

The program effectively captures and displays a date in dd/mm/yyyy format using user input. Adding input validation ensures the date, month, and year values are within a valid range, making the program more robust and user-friendly.

15. Write a program to print the area of a rectangle by creating a class named 'Area' taking the values of its length and breadth as parameters of its constructor and having a method named 'returnArea' which returns the area of the rectangle. Length and breadth of rectangle are entered through keyboard.
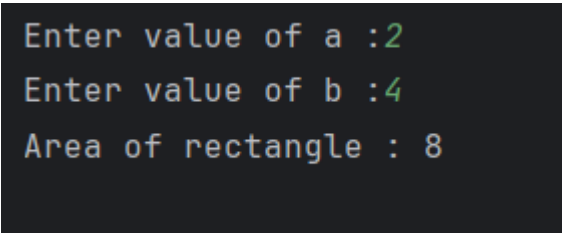
PROGRAM CODE:

```java
import java.util.Scanner;

class Practical_15{
    Practical_15(int length , int breath){
        int a = length;
        int b = breath;
    }

    static int reaturnArea(int length , int breath)
    {
        int area = length * breath;
        return area;
    }
}
```

```
class practical14{

public static void main(String args[]) {
    Practical_15 r;
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter value of a :");
    int a = sc.nextInt();
    System.out.print("Enter value of b :");
    int b = sc.nextInt();
    System.out.println("Area of rectangle : " + Practical_15.reaturnArea(a,b));
}
}
```

OUTPUT:

```
Enter value of a :2
Enter value of b :4
Area of rectangle : 8
```

CONCLUSION:

he program defines a class Area that encapsulates the properties of a rectangle (length and breadth) and includes a method returnArea to calculate and return the area. The main method in AreaTest handles user input for length and breadth, creates an Area object with these values, and then prints the calculated area.

16. Print the sum, difference and product of two complex numbers by creating a class named 'Complex' with separate methods for each operation whose real and imaginary parts are entered by user.

PROGRAM CODE:

```java
import java.util.Scanner;

class Complex{

    void sum(int r1 , int i1 , int r2 , int i2)
    {
        int r = r1 + r2;
        int i = i1 + i2;
        System.out.println("Sum of two numbers are :" + r + " + " + i + "i" );
    }

    void difference(int r1 , int i1 , int r2 , int i2)
    {
        int r = r1 - r2;
        int i = i1 - i2;
        System.out.println("Difference of two numbers are :" + r + " + " + i + "i" );
    }

    void multiplication(int r1 , int i1 , int r2 , int i2)
    {
        int r = r1*r2 - i1*i2;
        int i = r1*i2 + i1*r2;
        System.out.println("Multiplication of two numbers are :" + r + " + " + i + "i" );
    }
}

public class Practical_16 {
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("For Number 1 :");
        System.out.print("Enter the real part :");
        int r1= sc.nextInt();
```

```
        System.out.print("Enter the imaginary part :");
        int i1= sc.nextInt();
        System.out.println();

        System.out.println("For Number 2 :");
        System.out.print("Enter the real part :");
        int r2= sc.nextInt();
        System.out.print("Enter the imaginary part :");
        int i2= sc.nextInt();
        System.out.println();
        Complex c = new Complex();
        c.sum(r1,i1,r2,i2);
        c.difference(r1,i1,r2,i2);
        c.multiplication(r1,i1,r2,i2);

    }
}
```

OUTPUT:

```
For Number 1 :
Enter the real part :2
Enter the imaginary part :3


For Number 2 :
Enter the real part :2
Enter the imaginary part :3


Sum of two numbers are :4 + 6i
Difference of two numbers are :0 + 0i
Multiplication of two numbers are :-5 + 12i


Process finished with exit code 0
```

CONCLUSION:

The program allows for the creation and addition of complex numbers by reading user input for the real and imaginary parts and then displaying the sum of two complex numbers.

CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH

Department of Computer Science & Engineering

Subject Name: Java Programming

Semester: 3
Subject Code: CSE-201
Academic year: 2024-25

Part - 4

No.	Aim of the Practical
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.

PROGRAM CODE:

```java
public class practical17 {
    public static class parent {
        static void display() {
            System.out.println("This is parent class");
        }
    }

    public static class child extends parent {
        static void display2() {
            System.out.println("This is child class");
        }
    }

    public static void main(String[] args) {
        parent p = new parent();
        child c = new child();
```

```
        p.display();
        c.display2();
      }
}
```

OUTPUT:

```
This is parent class
This is child class


Process finished with exit code 0
```

CONCLUSION:

By creating an object of the parent class and calling its method, we can see that the method defined in the parent class (Parent) works correctly and prints "This is parent class".

18.

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

PROGRAM CODE:

import java.util.*;

public class practical18 {

```java
    public static class Member {
        String name;
        int age;
        String phoneNumber;
        String address;
        double salary;

        public void printSalary() {
            System.out.println("Salary: " + salary);
        }
    }

    public static class Employee extends Member {
        String specialization;
        public Employee(String name, int age, String phoneNumber, String address, double
salary, String specialization) {
                this.name = name;
                this.age = age;
                this.phoneNumber = phoneNumber;
                this.address = address;
                this.salary = salary;
                this.specialization = specialization;
            }
        public void printDetails() {
            System.out.println("Name: " + name);
            System.out.println("Age: " + age);
            System.out.println("Phone Number: " + phoneNumber);
            System.out.println("Address: " + address);
            System.out.println("Specialization: " + specialization);
            printSalary();
        }
    }

    public static class Manager extends Member {
        String department;

        public Manager(String name, int age, String phoneNumber, String address, double
salary, String department) {
            this.name = name;
            this.age = age;
```

```java
            this.phoneNumber = phoneNumber;
            this.address = address;
            this.salary = salary;
            this.department = department;
        }


    public void printDetails() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Phone Number: " + phoneNumber);
        System.out.println("Address: " + address);
        System.out.println("Department: " + department);
        printSalary();
    }
  }

  public static void main(String[] args) {
    Employee employee = new Employee("Swara", 27, "1234567890", "34 main St",
50000.0, "Software Developer");
    Manager manager = new Manager("Yatri", 30, "6987654321", "Gujarat India",
75000.0, "HR");

    System.out.println("Employee Details:");
    employee.printDetails();
    System.out.println("\nManager Details:");
    manager.printDetails();
  }
}
```

OUTPUT:

```
"C:\Program Files\Java\jdk-22\bin\java.ex
Employee Details:
Name: Swara
Age: 27
Phone Number: 1234567890
Address: 34 main St
Specialization: Software Developer
Salary: 50000.0

Manager Details:
Name: Yatri
Age: 30
Phone Number: 6987654321
Address: Gujarat India
Department: HR
Salary: 75000.0

Process finished with exit code 0
```

CONCLUSION:

his example demonstrates the concept of inheritance in object-oriented programming. The Employee and Manager classes inherit from the Member class, allowing them to use its attributes and methods. This provides a clear structure for extending functionality while maintaining common attributes and methods in the base class (Member).

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 theconstructor 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{

```java
      int breadth, length;
      public rectangle(int b, int l)
      {
         this.breadth = b;
         this.length = l;
      }

      public void area()
      {
         int area = this.length * this.breadth;
         System.out.println("the area  is:"+area);
      }

      public void perimeter()
      {
         int perimeter = 2 * (this.length + this.breadth);
         System.out.println("the perimeter  is :"+perimeter);
      }
}
class square extends rectangle
{
      public square(int x)
      {
         super(x, x);
      }
}
public class practical19 {
      public static void main(String[] args) {
         square s1[] = new square[2];
         s1[0] = new square(18);
         s1[1] = new square(20);

         s1[0].area();
         s1[0].perimeter();

         s1[1].area();
         s1[1].perimeter();
      }

}
```

OUTPUT:

```
the area  is:324
the perimeter  is :72
the area  is:400
the perimeter  is :80

Process finished with exit code 0
```

CONCLUSION:

The rectangle class represents a rectangle with methods to calculate area and perimeter
The square class inherits from rectangle and is a specific type of rectangle where all sides
are equal. The practical19 class demonstrates the creation and usage of square objects,
printing their areas and perimeters.

---

20. Create a class named 'Shape' with a method to print "This is This is shape". Then create two
other classes named 'Rectangle', 'Circle' inheriting the Shape class, both having a method to
print "This is rectangular shape" and "This is circular shape" respectively. Create a subclass
'Square' of 'Rectangle' having a method to print "Square is a rectangle". Now call the
method of 'Shape' and 'Rectangle' class by the object of 'Square' class.
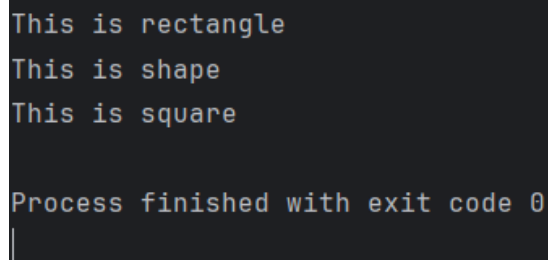
PROGRAM CODE:

```
package java_practicals;

  class Shape {
    public void displayShape() {
       System.out.println("This is shape");
    }
  }

  class Rectangle extends Shape {
    public void displayRectangle() {
       System.out.println("This is rectangle");
```

```java
    }
  }

  class Circle extends Shape {
    public void displayCircle() {
      System.out.println("This is circle");
    }
  }

  class Square extends Rectangle {
    public void displaySquare() {
      System.out.println("This is square");
    }
  }
  public class practical20
  {
    public static void main(String[] args) {
      Square s1 = new Square();
      s1.displayRectangle();
      s1.displayShape();
      s1.displaySquare();
    }
  }
```

OUTPUT:

```
This is rectangle
This is shape
This is square


Process finished with exit code 0

```

CONCLUSION:

This shows that the Square class can access methods from both Shape and Rectangle classes, demonstrating inheritance and method overriding in Java.

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

PROGRAM CODE:

```java
class degree{

    public void getDegree()
    {
        System.out.println("i got a degree");
    }
}

class Undergraduate extends degree
{
    public void getUndergraduate()
    {
        System.out.println("i am an undergraduate");
    }
}

class Postgraduate extends degree
{
    public void getPostgraduate()
    {
        System.out.println("i am a postgraduate");
    }
}


public class practical21 {
    public static void main(String[] args) {
        degree d1 = new degree();
        Undergraduate u1 = new Undergraduate();
        Postgraduate p1 = new Postgraduate();

        d1.getDegree();
        u1.getUndergraduate();
```

p1.getPostgraduate();
    }
}

OUTPUT:

```
i got a degree
i am an undergraduate
i am a postgraduate

Process finished with exit code 0
```

CONCLUSION:

In Java, method overriding allows subclasses to provide a specific implementation of a method that is already defined in its parent class. By creating objects of the Degree, Undergraduate, and Postgraduate classes, we can call their respective getDegree methods

---

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.

PROGRAM CODE:

```java
import java.util.Scanner;

interface AdvancedArithmetic{
    int divisor_sum(int n);

}
class calledMyCalculator implements AdvancedArithmetic{

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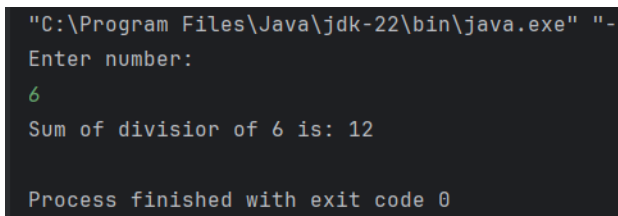
```
        if(n%i==0){
            sum+=i;
        }
    }
    return sum;
    }
}

public class Question22 {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter number:");
        int n= sc.nextInt();
        calledMyCalculator c1=new calledMyCalculator();
        System.out.println("Sum of divisior of " +n +" is: " + c1.divisor_sum(n));
    }

}
```

OUTPUT:

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-
Enter number:
6
Sum of divisior of 6 is: 12

Process finished with exit code 0
```

CONCLUSION:

The code effectively demonstrates how to implement an interface in Java and provides a solution to calculate the sum of all divisors of a given number. The MyCalculator class correctly implements the divisor_sum(int n) method as specified by the AdvancedArithmetic interface.

| 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. |

 PROGRAM CODE:

```java
interface Shape {
   String getColor();
   void draw();


   default void describe() {
      System.out.println("This is a shape with color " + getColor());
   }
}


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

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

   public double getRadius() {
      return radius;
   }

   public void setRadius(double radius) {
      this.radius = radius;
   }


   public String getColor() {
      return color;
```

```
    }


    public void draw() {
        System.out.println("Drawing a circle with radius " + radius + " and color " + color);
    }


    public void describe() {
        System.out.println("This is a shape with color " + getColor());
    }
}


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

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

    public double getLength() {
        return length;
    }

    public void setLength(double length) {
        this.length = length;
    }

    public double getWidth() {
        return width;
    }

    public void setWidth(double width) {
        this.width = width;
    }
```

```java
   public String getColor() {
      return color;
   }

   public void draw() {
      System.out.println("Drawing a rectangle with length " + length + ", width " + width +
" and color " + color);
   }


   public void describe() {
      System.out.println("This is a shape with color " + getColor());
   }
}

// Implement Sign class
class Sign {
   private Shape shape;
   private String text;

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

   public Shape getShape() {
      return shape;
   }

   public void setShape(Shape shape) {
      this.shape = shape;
   }

   public String getText() {
      return text;
   }

   public void setText(String text) {
```

```
      this.text = text;
    }

    public void display() {
       System.out.println("Sign Text: " + text);
       shape.draw();
       shape.describe();
    }
}

public class practical23 {
    public static void main(String[] args) {
       Circle circle = new Circle(5.0, "Red");
       Rectangle rectangle = new Rectangle(10.0, 4.0, "Blue");

       Sign circleSign = new Sign(circle, "Welcome to the Event!");
       Sign rectangleSign = new Sign(rectangle, "Campus Center Open!");

       circleSign.display();
       System.out.println();
       rectangleSign.display();
    }
}
```
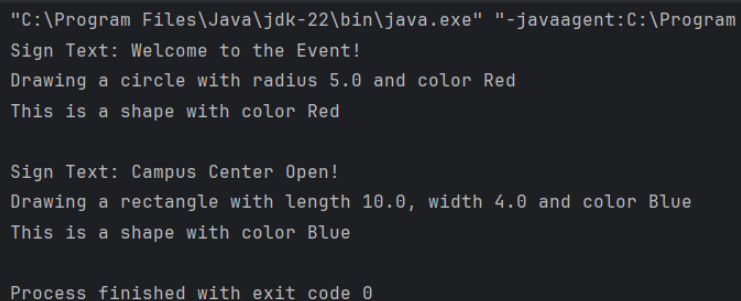
OUTPUT:

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program
Sign Text: Welcome to the Event!
Drawing a circle with radius 5.0 and color Red
This is a shape with color Red

Sign Text: Campus Center Open!
Drawing a rectangle with length 10.0, width 4.0 and color Blue
This is a shape with color Blue

Process finished with exit code 0
```

CONCLUSION:

The code demonstrates polymorphism by using the Shape interface to handle different types of shapes (Circle and Rectangle) in a uniform way. The Sign class is designed to work with any object that implements the Shape interface. The output shows how the Sign

	class can be used to display different shapes with associated text, effectively illustrating the concepts of interfaces, polymorphism, and method overriding in Java.

CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH

Department of Computer Science & Engineering

Subject Name: Java Programming

Semester: 3
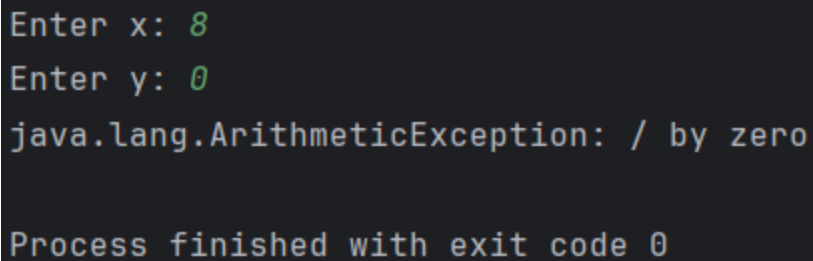**Subject Code: CSE-201**
Academic year: 2024-25

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:
	
	 ``java
	import java.util.Scanner;
	public class Practical_24
	{
	    public static void main(String[] args) {
	Scanner scanner = new Scanner(System.in);
	        try {
	            System.out.print("Enter x: ");
	            int x = scanner.nextInt();
	
	            System.out.print("Enter y: ");
	            int y = scanner.nextInt();
	
	            int result = x / y;
	            System.out.println("The result of " + x
	             +" / " + y + " is: " + result);
	            }
	        catch (Exception e) {
	``

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

OUTPUT:

```
Enter x: 8
Enter y: 0
java.lang.ArithmeticException: / by zero


Process finished with exit code 0
```

CONCLUSION:
This Java program prompts the user to input two integers, x and y, and performs division. It handles exceptions such as division by zero and non-integer inputs by catching ArithmeticException and Exception, ensuring the program doesn't crash due to invalid operations or inputs.

---

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

PROGRAM CODE:
```
public class Practical_25 {
    public static void UserException() throws
NullPointerException {
        String name = null;
        System.out.println("Length of the string: "
+ name.length());

    }
    public static void main(String[] args) {
        try {
            UserException();
        } catch (NullPointerException e) {
            System.out.println("Exception is solved
```

with:" + e);
        }
      }
}

OUTPUT:

```
Exception is solved with:java.lang.NullPointerException: Cannot invoke "String.length()" because "name" is null

Process finished with exit code 0
```

CONCLUSION:

The provided Java code demonstrates the use of the throws keyword and exception handling mechanisms using a try-catch block.

---

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:

```java
import java.util.Scanner;

class Userexception extends Exception {
    public Userexception(String s) {
        super(s);
    }
}

class User {
    public void method(int b) throws Userexception {
        if (b != 0) {
            System.out.println("Exception will not occur.");
        } else {
            throw new Userexception(" - Division by zero is not allowed.");
        }
    }
```
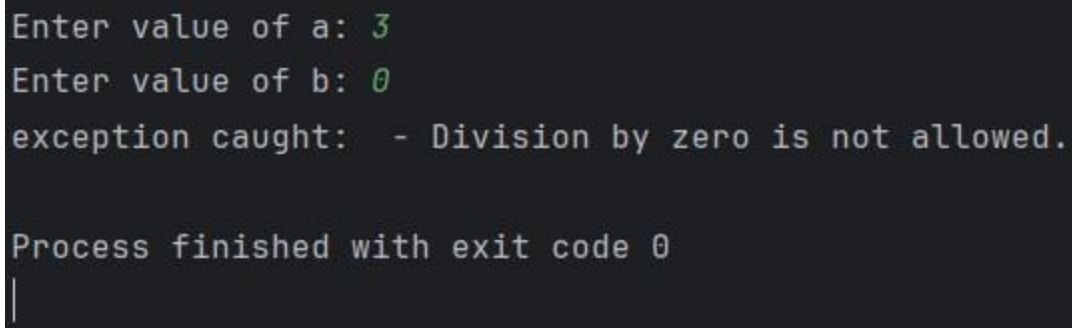
```
}
public class Practical_26 {
   public static void main(String[] args) {
      User o = new User();
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter value of a: ");
      int a = sc.nextInt();
      System.out.print("Enter value of b: ");
      int b = sc.nextInt();

      try {
         o.method(b);
         int c = a / b;
         System.out.println("Result of division is: " + c);
      } catch (Userexception e) {
         System.out.println("exception caught: " + e.getMessage());
      }
   }
}
```

OUTPUT:

```
Enter value of a: 3
Enter value of b: 0
exception caught:  - Division by zero is not allowed.

Process finished with exit code 0
```

CONCLUSION:

The given Java code demonstrates custom exception handling using a user-defined exception, User exception, to manage division by zero errors.

CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH

Department of Computer Science & Engineering

Subject Name: Java Programming

Semester: 3
Subject Code: CSE-201
Academic year: 2024-25

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:
	<pre>import java.io.*;
	public class Practical_27 {
	   public static void main(String[] args) {
	
	      try {
	         File file = new File("myfile.txt");
	         if (file.createNewFile()) {
	            System.out.println("File is created\n"
	+ file.getName());
	         } else {
	            System.out.println("File already
	exist");
	         }</pre>

```java
        FileWriter writer = new
FileWriter(file);
        writer.write("Hello World!\n");
        writer.write("Hello Java!\n");
        writer.close();


        FileReader reader = new
FileReader(file);
        BufferedReader bufferedReader = new
BufferedReader(reader);

        String line;
        int lineCount = 0;
       // int wordCount = 0;

        while ((line =
bufferedReader.readLine()) != null) {
            System.out.println(line);
            lineCount++;


            //String[] words = line.split("\\W+");
           // wordCount += words.length;
        }

        System.out.println("Total lines: " +
lineCount);
       // System.out.println("Total words: " +
wordCount);
        reader.close();
        bufferedReader.close();
    }
    catch(Exception e) {
       e.printStackTrace();
    }
  }
}
```

OUTPUT:

```
"C:\Program Files\Java\jdk-22\bin\java.
File already exist
Hello World!
Hello Java!
Total lines: 2

Process finished with exit code 0
```

CONCLUSION:

This program reinforces the importance of file handling in Java. By counting the number of lines in multiple files specified via the command line, students learn how to handle multiple file streams simultaneously, detect errors, and continue processing. The use of try-catch blocks ensures that errors are handled gracefully without terminating the program prematurely..

| 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:

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

public class Practical_28 {
   public static void main(String[] args) {
      Scanner scanner = new
Scanner(System.in);

      System.out.print("Enter the character to
search for: ");
      char searchChar =
scanner.next().charAt(0);

      System.out.print("Enter the file name
(e.g., xanadu.txt): ");
      String filename = scanner.next();

      int count = 0;

      System.out.println("Searching for
character: '" + searchChar + "' in file: " +
filename);

      try (BufferedReader br = new
BufferedReader(new FileReader(filename))) {
         int currentChar;

         while ((currentChar = br.read()) != -1)
{

            if ((char) currentChar ==
searchChar) {
               count++;
```
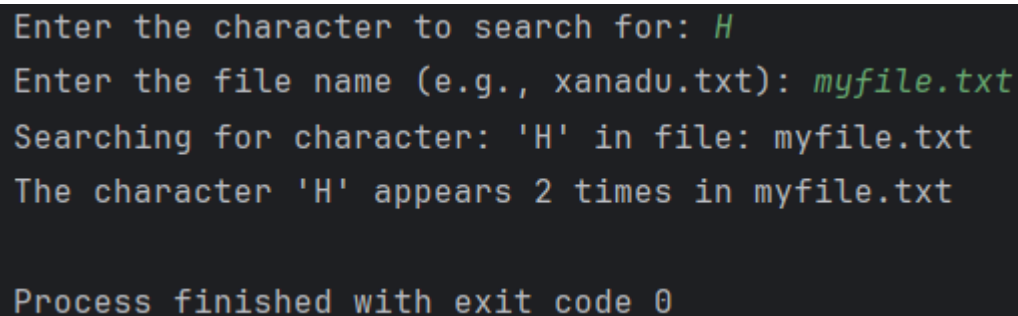
```
                }
            }
        } catch (IOException e) {
            System.out.println("Error reading file:
" + e.getMessage());
        }

        System.out.println("The character '" +
searchChar + "' appears " + count + " times in "
+ filename);
    }
}
```

OUTPUT:

```
Enter the character to search for: H
Enter the file name (e.g., xanadu.txt): myfile.txt
Searching for character: 'H' in file: myfile.txt
The character 'H' appears 2 times in myfile.txt


Process finished with exit code 0
```

 CONCLUSION:

This task highlights the ability to search for specific characters in a file and emphasizes string manipulation and character comparisons. It teaches the student how to use command-line arguments effectively and read file data to perform character counting.

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

PROGRAM CODE:

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

public class Practical_29 {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the file name: ");
        String filePath = scanner.nextLine();

        System.out.print("Enter the word to search: ");
        String wordToSearch = scanner.nextLine();

        int count = searchWordInFile(filePath, wordToSearch);

        Integer result = Integer.valueOf(count);
        System.out.println("The word '" + wordToSearch + "' appears " + result + " times in the file.");

        scanner.close();
    }

    public static int searchWordInFile(String filePath, String word) {
        int count = 0;

        try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = br.readLine()) != null) {

                String[] words = line.split("\\W+");
                for (String w : words) {
```
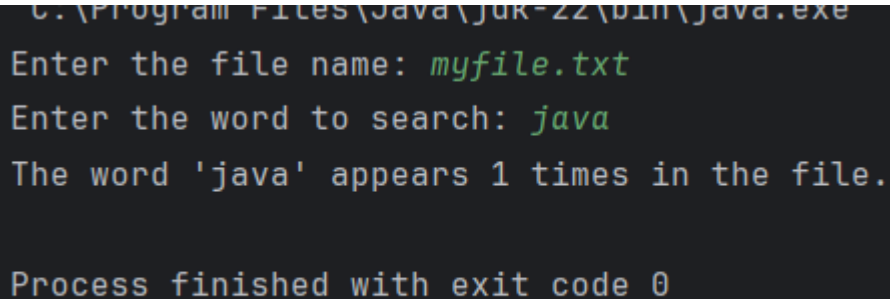
```
                if (w.equalsIgnoreCase(word)) {
                    count++;
                }
            }
        }
    } catch (IOException e) {
        System.out.println("An error occurred while reading the file: " + e.getMessage());
    }

    return count;
    }
}
```

OUTPUT:

```
C:\Program Files\Java\juk-22\bin\java.exe   -ja

Enter the file name: myfile.txt
Enter the word to search: java
The word 'java' appears 1 times in the file.


Process finished with exit code 0
|
```

CONCLUSION:

This program focuses on searching for a specific word in a file, demonstrating the use of string methods like .contains() or pattern matching through regular expressions. It also involves the utilization of Java Wrapper classes, improving understanding of object conversion and file handling..

| 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:

```java
import java.io.*;
import java.util.Scanner;
public class Practical_30 {
   public static void main(String[] args) {

      try {
         File file = new File("myfile1.txt");
         if (file.createNewFile()) {
            System.out.println("File " + file.getName()+ " is created\n" );
         } else {
            System.out.println("File already exist");
         }

         FileWriter fwrite = new FileWriter("myfile1.txt");
         fwrite.write("Hello, How are you?");
         fwrite.close();

         File file1 = new File("myfile2.txt");
         if (file.createNewFile()) {
            System.out.println("File " + file1.getName()+ " is created\n" );
         } else {
            System.out.println("File already exist");
         }

         FileReader fread = new FileReader("myfile1.txt");
         BufferedReader bufferedReader = new BufferedReader(fread);
         String line;

         FileWriter fwrite1 = new FileWriter("myfile2.txt");
         while((line = bufferedReader.readLine())!=null)
         {
            fwrite1.write(line + "\n");
```
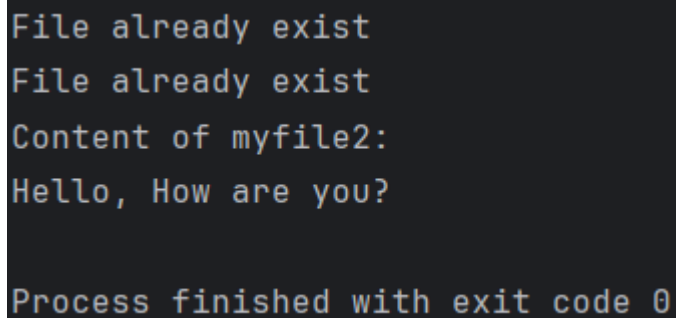
```
        }

        bufferedReader.close();
        fwrite1.close();

        FileReader fread1 = new FileReader("myfile2.txt");
        BufferedReader bufferedReader1 = new BufferedReader(fread1);
        System.out.println("Content of myfile2: ");
        while((line = bufferedReader1.readLine())!=null)
        {
            System.out.println(line);
        }
        bufferedReader1.close();
    }
    catch(IOException e){
        e.printStackTrace();
    }
  }
}
```

OUTPUT:

```
File already exist
File already exist
Content of myfile2:
Hello, How are you?


Process finished with exit code 0
```

CONCLUSION:

The file-copying task teaches practical skills in file I/O operations. The program demonstrates reading data from one file and writing it to another, providing the student with knowledge on stream manipulation and error handling when dealing with non-existent files..

| 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:

```java
import java.io.*;

public class Practical_31 {

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

            System.out.println("Character Stream - Reading from one file and writing to another.");

            FileWriter charWriter = new FileWriter("charOutput.txt");
            charWriter.write("This is written using a character stream.");
            charWriter.close();

            FileReader charReader = new FileReader("charOutput.txt");
            BufferedReader bufferedReader = new BufferedReader(charReader);
            String c;
            System.out.println("Contents of charOutput.txt (Character Stream):");
            while ((c = bufferedReader.readLine()) != null) {
                System.out.println(c);
            }
            bufferedReader.close();
            charReader.close();

            System.out.println("\n\nByte Stream - Reading from one file and writing to another.");

            FileOutputStream byteWriter = new FileOutputStream("byteOutput.txt");
            byteWriter.write("This is written using a byte stream.".getBytes());
            byteWriter.close();
```

```java
        FileInputStream byteReader = new FileInputStream("byteOutput.txt");
        int b;
        System.out.println("Contents of byteOutput.txt (Byte Stream):");
        while ((b = byteReader.read()) != -1) {
            System.out.print((char) b);
        }
        byteReader.close();


        System.out.println("\n\nNow, using BufferedReader to take input from the console
and BufferedWriter to write it to a file.");

        BufferedReader consoleReader = new BufferedReader(new
InputStreamReader(System.in));
        System.out.println("Enter some text (BufferedReader will read this):");
        String inputText = consoleReader.readLine();

        BufferedWriter bufferedWriter = new BufferedWriter(new
FileWriter("bufferedOutput.txt"));
        bufferedWriter.write(inputText);
        bufferedWriter.close();

        consoleReader.close();
        System.out.println("Your input has been written to bufferedOutput.txt.");

    } catch (IOException e) {
        e.printStackTrace();
    }
  }
}
```

OUTPUT:

```
Character Stream - Reading from one file and writing to another.
Contents of charOutput.txt (Character Stream):
This is written using a character stream.



Byte Stream - Reading from one file and writing to another.
Contents of byteOutput.txt (Byte Stream):
This is written using a byte stream.
```

CONCLUSION:

By working with both character and byte streams, students gain exposure to different ways of reading and writing data in Java. The use of BufferedReader and BufferedWriter enhances efficiency by demonstrating buffered I/O operations, an important concept for improving the performance of file operations.

CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH

Department of Computer Science & Engineering

Subject Name: Java Programming

Semester: 3
Subject Code: CSE-201
Academic year: 2024-25

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.
	
	 PROGRAM CODE:
	
	 `class B implements Runnable`
	 `{`
	 `    public void run()`
	 `    {`
	 `        System.out.println("Hello World");`
	 `    }`
	 `}`
	
	 `public class Practical_32 {`
	 `    public static void main(String[] args) {`
	
	 `        B b = new B();`
	 `        Thread t = new Thread(b);`
	 `        t.start();`
	 `    }`
	 `}`

OUTPUT:

```
"C:\Program Files\Java\jdk-22\bin\java
Hello World


Process finished with exit code 0
```

CONCLUSION:
This task introduces students to multithreading by demonstrating two different ways of creating threads in Java: by extending the Thread class and implementing the Runnable interface. It helps students understand the fundamental differences and best practices for thread creation and execution..

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.*;

public class Practical_33 implements Runnable
{

    @Override
    public void run() {

        System.out.println(" thread is created");



    }

    public static void main(String[] args) {

        System.out.println("Enter the total number
of the tread you want to creat :");
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        Practical_33 obj = new Practical_33();
        for (int i = 1; i <=n; i++) {
            Thread t = new Thread(obj);
            t.start();
        }
        sc.close();
    }
}
```

OUTPUT:

```
Enter  the total number of the tread you want to creat :
 12
 thread is created
 thread is created
 thread is created
 thread is created
 thread is created
 thread is created
 thread is created
 thread is created
 thread is created
 thread is created
 thread is created
 thread is created


 Process finished with exit code 0
```

CONCLUSION:

This task teaches students how to divide computational tasks among multiple threads. By splitting the summation of numbers across several threads, it emphasizes the importance of parallelism in improving performance. Students learn how to manage multiple threads and aggregate their results effectively.

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:

```java
import java.util.Random;

class Thread1 extends Thread {
    int count;
    public Thread1(int count) {
        this.count = count;
    }

    public void run() {
        Random random = new Random();
        for (int i = 0; i < count; i++) {
            int number = random.nextInt(10);
            System.out.println("Number Generated by thread1 is: " + number);

            if (number % 2 == 0) {
                new Thread2(number).start();
            } else {
                new Thread3(number).start();
            }

            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                System.out.println("Thread is interrupted during sleep");
            }
        }
        System.out.println("Random number generation completed.");
    }
}

class Thread2 extends Thread {
    private int number;
```

```java
   public Thread2(int number) {
      this.number = number;
   }

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

class Thread3 extends Thread {
   private int number;

   public Thread3(int number) {
      this.number = number;
   }

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

public class Practical_34 {
   public static void main(String[] args) {
      Thread1 t1 = new Thread1(5);
      t1.start();
   }
}
```

OUTPUT:

```
"C:\Program Files\Java\jdk-22\bin\java
Number Generated by thread1 is: 6
Square of 6 is: 36
Number Generated by thread1 is: 9
Cube of 9 is: 729
Number Generated by thread1 is: 8
Square of 8 is: 64
Number Generated by thread1 is: 8
Square of 8 is: 64
Number Generated by thread1 is: 6
Square of 6 is: 36
Random number generation completed.
```

CONCLUSION:

This program focuses on using multiple threads (three in this case) with specific roles. The task strengthens understanding of communication between threads, conditional execution based on thread output, and introduces basic concepts of synchronization to avoid race conditions..

| 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:

```java
class MyThread extends Thread
{
   int count=0;
   public void run()
   {
      while (true)
      {
         count++;
         System.out.println(count);
         try
         {
            MyThread.sleep(1000);
         }
         catch (InterruptedException e) {
            e.printStackTrace();
         }
      }
   }
}


public class Practical_35 {
   public static void main(String[] args) {

      MyThread t = new MyThread();
      t.start();
   }
}
```

OUTPUT:

```
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
Process finished with exit code 130
```

CONCLUSION:

This task demonstrates the use of the sleep() method in threads to pause execution for a defined time interval. It teaches students how to control the flow of threads, particularly when simulating real-time updates, like incrementing the value of a variable at regular intervals..

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:

```java
class threadPriority extends Thread
{

    threadPriority(String threadName)
    {
        super(threadName);
    }

    public void run()
    {
        System.out.println("Thread :- "+this.getName());
    }

}

public class Practical_36
{
    public static void main(String[] args) {
        threadPriority t1 = new threadPriority("FIRST");
        threadPriority t2 = new threadPriority("SECOND");
        threadPriority t3 = new threadPriority("THIRD");
        t1.setPriority(3);
        t2.setPriority(5);
        t3.setPriority(7);
        t1.start();
        t2.start();
        t3.start();
    }

}
```

OUTPUT:

```
"C:\Program Files\Java\jdk-22\bin\java.
Thread :- THIRD
Thread :- SECOND
Thread :- FIRST


Process finished with exit code 0
```

CONCLUSION:

This program introduces thread priority management, which allows students to explore how thread scheduling and priorities affect execution. By setting different priorities to three threads, students learn how Java handles thread execution when system resources are shared between tasks

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

PROGRAM CODE:

```java
class SharedBuffer {
    private int value = 0;
    private boolean hasValue = false;


    public synchronized void produce() throws InterruptedException {
        while (true) {

            if (hasValue) {
                wait();
            }
            value++;
            System.out.println("Produced: " + value);
            hasValue = true;
            notify();
            Thread.sleep(1000);
        }
    }


    public synchronized void consume() throws InterruptedException {
        while (true) {

            if (!hasValue) {
                wait();
            }

            System.out.println("Consumed: " + value);
            hasValue = false;
            notify();
            Thread.sleep(1500);
        }
    }
}
```

```java
class Producer extends Thread {
    private SharedBuffer buffer;

    public Producer(SharedBuffer buffer) {
        this.buffer = buffer;
    }

    public void run() {
        try {
            buffer.produce();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}


class Consumer extends Thread {
    private SharedBuffer buffer;

    public Consumer(SharedBuffer buffer) {
        this.buffer = buffer;
    }

    public void run() {
        try {
            buffer.consume();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}


public class Practical_37 {
    public static void main(String[] args) {

        SharedBuffer buffer = new SharedBuffer();

        Producer producer = new Producer(buffer);
```
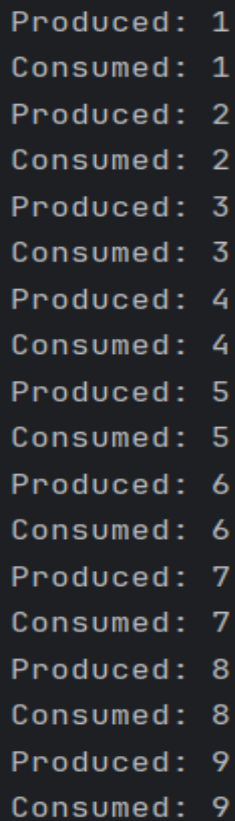
```
        Consumer consumer = new Consumer(buffer);

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

OUTPUT:

```
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
```

CONCLUSION:

This task is a classic problem that demonstrates the importance of synchronization when dealing with shared resources between threads. Students learn how to avoid issues like race conditions and deadlocks by using synchronized methods or blocks to control access to shared data between producer and consumer threads.

CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH

Department of Computer Science & Engineering

Subject Name: Java Programming

Semester: 3
Subject Code: CSE-201
Academic year: 2024-25

Part - 8

No.	Aim of the Practical
38.	Design a Custom Stack using ArrayList class, which 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. **PROGRAM CODE:** import javax.swing.plaf.synth.SynthOptionPaneUI; import java.util.ArrayList; import java.util.Iterator; import java.util.Scanner; class stack { ArrayList array = new ArrayList(); public boolean isEmpty() {

```java
        return array.isEmpty();
    }

    public int getSize()
    {
        return array.size();
    }

    public void push(int x)
    {
        array.add(x);
        System.out.println("Element "+x+"
added to stack successfully");
    }

    void pop()
    {

        if(isEmpty())
        {
            System.out.println("Stack is empty");
        }
        else {
            array.remove(array.size() - 1);
            System.out.println("Element removed
from stack successfully");
        }
    }

    public Object peek()
    {
        if(isEmpty())
        {
            System.out.println("Stack is empty");
        }
        return array.get(array.size() - 1);
    }

    void print()
    {
```

```java
        if(isEmpty())
        {
           System.out.println("Stack is empty");
        }
        else {
           System.out.println("Stack Elements: " +
array);
        }
     }
}
public class Practical_38 {
   public static void main(String[] args) {

      stack s= new stack();
      ArrayList array = new ArrayList();
      Scanner sc= new Scanner(System.in);
      int choice, x;
      do {
         System.out.println("1. PUSH\n2.
POP\n3. PEEK\n4. DISPLAY\n5. EXIT");
         System.out.println("Enter you choice:
");
         choice = sc.nextInt();

         switch (choice)
         {
            case 1:
               System.out.println("Enter the value
you want to add:");
               x=sc.nextInt();
               s.push(x);
               break;

            case 2:
               s.pop();
               break;

            case 3:
               Object topElement= s.peek();
               if(topElement!=null){
```

```
                System.out.println("Stack top: "+
topElement);
                }
             break;

          case 4:
             s.print();
             break;

          case 5:
             System.out.println("Exiting...");

          default:
             System.out.println("Enter valid
choice");
        }
    }while(choice!=5);
  }
}
```

OUTPUT

```
C:\Program Files\Java\jdk-22\bin\java.exe
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter you choice:
1
Enter the value you want to add:
12
Element 12 added to stack successfully
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter you choice:
1
Enter the value you want to add:
23
Element 23 added to stack successfully
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
```

```
Stack top: 23
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter you choice:
2
Element removed from stack successfully
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter you choice:
4
Stack Elements: [12]
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
Enter you choice:
```

CONCLUSION:

This task teaches students how to implement a custom stack using the ArrayList class,
providing practical experience with the Collection Framework in Java. By implementing
standard stack operations such as push(), pop(), peek(), and isEmpty(), students gain a
solid understanding of how data structures like stacks can be built using dynamic arrays.
This task reinforces the concept of generic programming and demonstrates the usefulness
of lists in implementing common data structures.

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:

```java
import java.util.Arrays;

public class Practical_39 {

    public static <T extends Comparable<T>>
void sort(T[] array) {
        Arrays.sort(array);
    }

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

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

        public String getName() {
            return name;
        }

        public double getPrice() {
            return price;
        }
```

```java
    public double getRating() {
        return rating;
    }

    public int compareTo(Product other) {
        // Default sorting by name, can be
changed as per requirements
        return
this.name.compareTo(other.name);
    }

    public String toString() {
        return "Product{name='" + name + "',
price=" + price + ", rating=" + rating + "}";
    }
  }

  public static void main(String[] args) {
    Product[] products = {

        new Product("Smartphone", 800.00,
4.7),
        new Product("Laptop", 1200.00, 4.5),
        new Product("Tablet", 400.00, 4.3)
    };

    System.out.println("Before sorting:");

System.out.println(Arrays.toString(products));

    sort(products);

    System.out.println("After sorting by
name:");

System.out.println(Arrays.toString(products));
  }
}
```

OUTPUT:

```
Before sorting:

[Product{name='Smartphone', price=800.0, rating=4.7}, Product{name='Laptop', price=1200.0, rating=4.5}, Product{name='Tablet', price=400.0, rating=4.3}]

After sorting by name:

[Product{name='Laptop', price=1200.0, rating=4.5}, Product{name='Smartphone', price=800.0, rating=4.7}, Product{name='Tablet', price=400.0, rating=4.3}]


Process finished with exit code 0
```

CONCLUSION:

This task encourages students to implement a flexible and reusable sorting system using generics and the Comparable interface. It focuses on object-oriented principles by requiring students to sort a list of products based on different criteria like price or name. This deepens their understanding of sorting algorithms, collections, and how generics enhance code flexibility, making it applicable to different types of objects such as products or orders.

| 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:

```java
import java.util.*;

public class Practical_40 {
   public static void main(String[] args) {

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


      Scanner scanner = new Scanner(System.in);

      System.out.println("Enter words one by one. Type 'exit' when you are done:");

      while (true) {
         String word = scanner.next().toLowerCase(); // Read and convert to lowercase
         if (word.equals("exit")) {
            break;
         }


         wordCountMap.put(word, wordCountMap.getOrDefault(word, 0) + 1);
      }

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

OUTPUT:

```
Enter words one by one. Type 'exit' when you are done:
mango
apple
banana
mango
apple
banana
orange
exit
Word occurrences in alphabetical order:
apple: 2
banana: 2
mango: 2
orange: 1


Process finished with exit code 0
```

CONCLUSION:

his program helps students learn how to work with Java's Map and Set classes to efficiently count and store the occurrences of words in a file. By sorting the words alphabetically, it teaches the importance of combining collections with algorithms to solve text processing problems, demonstrating practical use cases of HashMap and TreeSet in real-world applications like word counting and sorting.

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:

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashSet;
import java.util.Set;

public class Practical_41 {
    private static final Set<String> KEYWORDS = new HashSet<>();

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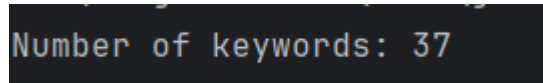
```java
        KEYWORDS.add("import");
        KEYWORDS.add("instanceof");
        KEYWORDS.add("int");
        KEYWORDS.add("interface");
        KEYWORDS.add("long");
        KEYWORDS.add("native");
        KEYWORDS.add("new");
        KEYWORDS.add("package");
        KEYWORDS.add("private");
   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

    public static void main(String[] args) throws IOException {

        String fileName = "D:\\SEMESTER 3\\JAVA\\Practicals\\Part
8\\src\\Practical_39.java";
        int keywordCount = 0;

        try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] words = line.split("\\s+");
                for (String word : words) {
                    if (KEYWORDS.contains(word)) {
```

```
                keywordCount++;
            }
        }
      }
    }

    System.out.println("Number of keywords: " + keywordCount);
  }
}
```

OUTPUT:

```
Number of keywords: 37
```

CONCLUSION:

This task focuses on text processing and efficient lookup using the HashSet collection. By storing all Java keywords in a set and checking their occurrence in a source file, students practice using contains() for quick membership tests, improving performance for large-scale data. This task demonstrates how sets can be used for fast lookups and reinforces the concept of keyword filtering in text analysis.