Devang Patel Institute of Advance Technology and Research

DEPSTAR (A Constitute Institute of CHARUSAT)

Certificate

This is to certify that

Mr./~~Mrs.~~ Meet Kalpeshbhai Patel

of 3CSE-2 *Class,*

ID. No. 23DCS085 *has satisfactorily completed*

his/~~her~~ term work in CSE201:JAVA PROGRAMING *for*

the ending in NOV *2024/2025*

Date : 17/10/24

Sign. of Faculty

Head of Department

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

Subject : JAVA PROGRAMMING **Semester:** 3

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

Course Outcome (COs):

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

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

Bloom's Taxonomy:

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

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

Practical List

Sr No.	AIM	Hrs.	CO	Bloom's Taxonomy
PART-I Data Types, Variables, String, Control Statements, Operators, Arrays				
1	Demonstration of installation steps of Java, Introduction to Object Oriented Concepts, comparison of Java with other object-oriented programming languages. Introduction to JDK, JRE, JVM, Javadoc, command line argument. Introduction to Eclipse or NetBeans IDE,or BlueJ and Console Programming.	**2**	**1**	**1**
2	Imagine you are developing a simple banking application where you need to display the current balance of a user account. For simplicity, let's say the current balance is $20. Write a java program to store this balance in a variable and then display it to the user.	**1**	**1**	**2,3,4**
3	Write a program to take the user for a distance (in meters) and the time taken (as three numbers: hours, minutes, seconds), and display the speed, in meters per second, kilometers per hour and miles per hour (hint:1 mile = 1609 meters).	**1**	**1**	**2,3,4**
4	Imagine you are developing a budget tracking application. You need to calculate the total expenses for the month. Users will input their daily expenses, and the program should compute the sum of these expenses. Write a Java program to calculate the sum of elements in an array representing daily expenses. **Supplementary Experiment:** You are creating a library management system. The library has two separate lists of books for fiction and non-fiction. The system should merge these lists into a single list for inventory purposes. Write a Java program to merge two arrays.	**1**	**1, 2**	**2,3**
5	An electric appliance shop assigns code 1 to motor,2 to fan,3 to tube and 4 for wires. All other items have code 5 or more. While selling the goods, a sales tax of 8% to motor,12% to fan,5% to tube light,7.5% to wires and 3% for all other items is charged. A list containing the product code and price in two different arrays. Write a java program using switch statement to prepare the bill.	**1**	**1, 2**	**2**
6	Create a Java program that prompts the user to enter the	**1**	**1, 2**	**2,3,4**

	number of days (n) for which they want to generate their exercise routine. The program should then calculate and display the first n terms of the Fibonacci series, representing the exercise duration for each day. **Supplementary Experiment:** Imagine you are developing a classroom management system. You need to keep track of the grades of students in a class. After collecting the grades, you want to display each student's grade along with a message indicating if they have passed or failed. Let's assume the passing grade is 50.			
	PART-II Strings			
7	Given a string and a non-negative int n, we'll say that the front of the string is the first 3 chars, or whatever is there if the string is less than length 3. Return n copies of the front; front_times('Chocolate', 2) → 'ChoCho' front_times('Chocolate', 3) → 'ChoChoCho' front_times('Abc', 3) → 'AbcAbcAbc'	**1**	**1, 2**	**2,3,4**
8	Given an array of ints, return the number of 9's in the array. array_count9([1, 2, 9]) → 1 array_count9([1, 9, 9]) → 2 array_count9([1, 9, 9, 3, 9]) → 3 **Supplementary Experiment:** 1. Write a Java program to replace each substring of a given string that matches the given regular expression with the given replacement. Sample string : "The quick brown fox jumps over the lazy dog." **In the above string replace all the fox with cat.**	**1**	**1, 2**	**2,3**
9	Given a string, return a string where for every char in the original, there are two chars. double_char('The') → 'TThhee' double_char('AAbb') → 'AAAAbbbb' double_char('Hi-There') → 'HHii--TThheerree'	**1**	**1, 2**	**2**
10	Perform following functionalities of the string: ● Find Length of the String ● Lowercase of the String ● Uppercase of the String ● Reverse String	**1**	**1, 2**	**2,3,4**

	Sort the string			
11	Perform following Functionalities of the string: "CHARUSAT UNIVERSITY" ● Find length ● Replace 'H' by 'FIRST LATTER OF YOUR NAME' ● Convert all character in lowercase **Supplementary Experiment:** 1. Write a Java program to count and print all duplicates in the input string. Sample Output: The given string is: resource The duplicate characters and counts are: e appears 2 times r appears 2 times	**1**	**1, 2**	**4**
PART-III Object Oriented Programming: Classes, Methods, Constructors				
12	Imagine you are developing a currency conversion tool for a travel agency. This tool should be able to convert an amount in Pounds to Rupees. For simplicity, we assume the conversion rate is fixed: 1 Pound = 100 Rupees. The tool should be able to take input both from command-line arguments and interactively from the user.	**1**	**2**	**3**
13	Create a class called Employee that includes three pieces of information as instance variables—a first name (type String), a last name (type String) and a monthly salary (double). Your class should have a constructor that initializes the three instance variables. Provide a set and a get method for each instance variable. If the monthly salary is not positive, set it to 0.0. Write a test application named EmployeeTest that demonstrates class Employee's capabilities. Create two Employee objects and display each object's yearly salary. Then give each Employee a 10% raise and display each Employee's yearly salary again.	**2**	**1, 2**	**3**
14	Create a class called Date that includes three pieces of information as instance variables—a month (type int), a day (type int) and a year (type int). Your class should have a constructor that initializes the three instance variables and assumes that the values provided are correct. Provide a set and a get method for each instance variable. Provide a method displayDate that displays the month, day and year separated by forward slashes (/). Write a test application named DateTest that demonstrates class Date's capabilities.	**2**	**1, 2**	**3**

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

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

	calculate interest, and view balances. SavingsAccount and CurrentAccount should implement the Account interface and have their own unique methods. [L:A]			
23	Assume you want to capture shapes, which can be either circles (with a radiusand a color) or rectangles (with a length, width, and color). You also want to be able to create signs (to post in the campus center, for example), each of which has a shape (for the background of the sign) and the text (a String) to put on the sign. Create classes and interfaces for circles, rectangles, shapes, and signs. Write a program that illustrates the significance of interface default method.	**2**	**2,3**	**6**
	PART-V Exception Handling			
24	Write a java program which takes two integers x & y as input, you have to compute x/y. If x and y are not integers or if y is zero, exception will occur and you have to report it.	**1**	**4**	**3**
25	Write a Java program that throws an exception and catch it using a try-catch block.	**1**	**4**	**3**
26	Write a java program to generate user defined exception using “throw” and “throws” keyword. Also Write a java that differentiates checked and unchecked exceptions. (Mention at least two checked and two unchecked exceptions in program). **Supplementary Experiment:** 1.Write a Java program that reads a list of integers from the user and throws an exception if any numbers are duplicates. [L:M]	**2**	**4**	**2,3**
	PART-VI File Handling & Streams			
27	Write a program that will count the number of lines in each file that is specified on the command line. Assume that the files are text files. Note that multiple files can be specified, as in "java Line Counts file1.txt file2.txt file3.txt". Write each file name, along with the number of lines in that file, to standard output. If an error occurs while trying to read from one of the files, you should print an error message for that file, but you should still process all the remaining files.	**1**	**4,6**	**3**
28	Write an example that counts the number of times a	**1**	**4,6**	**3**

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

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

CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH

Department of Computer Science & Engineering

Subject Name: Java Programming

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

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:- 1. Installation of Java Steps to install Java Development Kit (JDK): □ Download JDK: - Go to the Oracle JDK download page: [Oracle JDK Downloads] (https://www.oracle.com/java/technologies/javase-downloads.html). - Select the appropriate JDK version for your operating system (Windows, macOS,Linux). - Download the installer package (.exe for Windows, .dmg for macOS, .tar.gz for Linux). □ Install JDK: - Windows: Double-click the downloaded .exe file and follow the installation instructions.

- macOS: Double-click the downloaded .dmg file, then drag and drop the JDK package icon to the Applications folder.
- Linux: Extract the downloaded .tar.gz file to a directory and follow the instructions in the README file for installation.

- Set JAVA_HOME (Optional):
- Windows: Set the JAVA_HOME environment variable to the JDK installation directory.
- macOS/Linux: Add the JDK bin directory to your PATH and set JAVA_HOME in your shell profile (e.g., ~/.bash_profile, ~/.bashrc).

- Verify Installation:
- Open a terminal or command prompt.
- Type java -version and javac -version to verify that Java runtime and compiler are installed correctly.

2. Introduction to Object-Oriented Concepts

Object-oriented programming (OOP) revolves around the concept of objects, which are instances of classes. Key principles include:

- Classes and Objects: Classes define the blueprint for objects.
- Encapsulation: Bundling data (attributes) and methods (functions) that operate on the data within a single unit (class).
- Inheritance: Mechanism where a new class (derived or child class) is created from an existing class (base or parent class).
- Polymorphism: Ability of different objects to be treated as instances of the same class through method overriding and overloading.

3. Comparison of Java with Other Object-Oriented Programming Languages

Java is often compared with languages like C++, C#, and Python in terms of syntax, features, and application domains. Key points of comparison include:

- Syntax: Java has a C-style syntax with similarities to C++.
- Memory Management: Java uses automatic garbage collection, unlike C++ which requires

manual memory management.
- Platform Independence: Java programs are compiled into bytecode, which can run on any JVM, making it platform-independent.

- Libraries: Java has a rich standard library (Java API) comparable to those in C++ and C#.
- Community and Ecosystem: Java has a large developer community and extensive third-party libraries and frameworks.

4. Introduction to JDK, JRE, JVM, Javadoc, Command Line Arguments

- JDK (Java Development Kit): Includes tools for developing and running Java programs, including JRE and development tools such as javac (Java compiler).
- JRE (Java Runtime Environment): Includes JVM (Java Virtual Machine) and libraries required to run Java applications, but does not include development tools.
- JVM (Java Virtual Machine): Executes Java bytecode and provides a runtime environment for Java programs.
- Javadoc: Tool for generating API documentation from Java source code comments.
- Command Line Arguments: Parameters passed to a Java program when it is invoked from the command line.

5. Introduction to Eclipse or NetBeans IDE (Integrated Development Environment)
- Eclipse : A widely used open-source IDE for Java development, also supports other programming languages through plugins. Features include code editing, debugging, and version control integration.
- NetBeans: Another popular open-source IDE primarily for Java development, with features similar to Eclipse.

6. Introduction to BlueJ and Console Programming

- BlueJ : A lightweight IDE specifically designed for teaching and learning Java programming, providing a simplified interface and visualization tools for object-oriented concepts.
- Console Programming : Refers to writing Java programs that interact with users via text-based input and output through the console (command line interface).Command Line Argument : -

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 :

```
class JavaPractical02
{
   public static void main(String[] args)
   {
      int Current_Balance;

      Current_Balance=20;

      System.out.print("Current Balance : ");
      System.out.print("$");
      System.out.print(Current_Balance);

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

      System.out.println("23DCS085 Meet K. Patel");
 }
}
```

OUTPUT:

```
Current Balance : $20

23DCS085 Meet K. Patel
```

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

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

PROGRAM CODE :

```
import java.util.*;
class JavaPractical03
{
   public static void main(String[] args)
   {

       float Distance_in_meters;
       float Time_in_hour;
       float Time_in_minute;
       float Time_in_seconds;

       Scanner sc =  new Scanner(System.in);

       System.out.print("Enter Distance in meters
: ");
       Distance_in_meters = sc.nextFloat();

       System.out.println(" ");

       System.out.print("Enter Time taken for the
distance in hours : ");
       Time_in_hour = sc.nextFloat();

       System.out.println(" ");

       System.out.print("Enter Time taken for the
distance in minutes : ");
       Time_in_minute = sc.nextFloat();
```

```
        System.out.println(" ");

        System.out.print("Enter Time taken for the
distance in seconds : ");
        Time_in_seconds = sc.nextFloat();

        float Time_1;
        float Time_2;
        float Distance_2;
        float Distance_3;
        float Speed;
        float Speed_2;
        float Speed_3;

        Distance_2 = Distance_in_meters/1000;
        Distance_3 = Distance_in_meters/1609;

        Time_1 =
((Time_in_hour*60*60)+(Time_in_minute*60)
+(Time_in_seconds));
        Time_2 =
Time_in_hour+(Time_in_minute/60)+(Time_in
_seconds/3600);

        Speed = Distance_in_meters/Time_1;
        Speed_2 = Distance_2/Time_2;
        Speed_3 = Distance_3/Time_2;

        System.out.println(" ");

        System.out.print("Speed : " + Speed + "
M/S");

        System.out.println(" ");
```

```
        System.out.print("Speed : " + Speed_2 + "
KM/H");

        System.out.println(" ");

        System.out.print("Speed : " + Speed_3 + "
Mile/H");

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

        System.out.println("23DCS085 Meet K.
Patel");

        sc.close();

    }
}
```

OUTPUT:

```
Enter Distance in meters : 20000

Enter Time taken for the distance in hours : 1

Enter Time taken for the distance in minutes : 30

Enter Time taken for the distance in seconds : 0

Speed : 3.7037036 M/S
Speed : 13.333333 KM/H
Speed : 8.28672 Mile/H

23DCS085 Meet K. Patel
```

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

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

PROGRAM CODE :

```
import java.util.*;
class JavaPractical04
{
   public static void main(String[] args)
   {

      Scanner sc = new Scanner(System.in);

      int i;
      int[] Exp = new int[30];
      int Total_Expenses = 0;

      System.out.println("---: Enter your monthly expenses here :---");

      for(i=0;i<30;i++)
      {

         System.out.print("Enter Expenses for Day - " + (i+1) + " : ");
```

```
            Exp[i] = sc.nextInt();

            Total_Expenses += Exp[i];

        }

        System.out.println(" ");

        System.out.println("Total Monthly Expenses is : " + Total_Expenses);

        System.out.println(" ");

        System.out.println("23DCS085 Meet K. Patel");

        sc.close();

    }

}
```

OUTPUT:

```
---: Enter your monthly expenses here :---
Enter Expenses for Day - 1 : 1
Enter Expenses for Day - 2 : 1
Enter Expenses for Day - 3 : 1
Enter Expenses for Day - 4 : 1
Enter Expenses for Day - 5 : 1
Enter Expenses for Day - 6 : 1
Enter Expenses for Day - 7 : 1
Enter Expenses for Day - 8 : 1
Enter Expenses for Day - 9 : 1
Enter Expenses for Day - 10 : 1
Enter Expenses for Day - 11 : 1
Enter Expenses for Day - 12 : 1
Enter Expenses for Day - 13 : 1
Enter Expenses for Day - 14 : 1
Enter Expenses for Day - 15 : 1
Enter Expenses for Day - 16 : 1
Enter Expenses for Day - 17 : 1
Enter Expenses for Day - 18 : 1
Enter Expenses for Day - 19 : 1
Enter Expenses for Day - 20 : 1
Enter Expenses for Day - 21 : 1
Enter Expenses for Day - 22 : 1
Enter Expenses for Day - 23 : 1
Enter Expenses for Day - 24 : 1
Enter Expenses for Day - 25 : 1
Enter Expenses for Day - 26 : 1
Enter Expenses for Day - 27 : 1
Enter Expenses for Day - 28 : 1
Enter Expenses for Day - 29 : 1
Enter Expenses for Day - 30 : 1

Total Monthly Expenses is : 30

23DCS085 Meet K. Patel
```

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

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

PROGRAM CODE :

```
import java.util.*;
class JavaPractical05
{
   public static void main(String[] args) {

      Scanner sc = new Scanner(System.in);

      int n;

      int[] Product_code = {1,2,3,4,5};
      float[] Product_price = {100,50,30,20,10};

      float MOTOR_TAX = 0.08f;
      float FAN_TAX = 0.12f;
      float TUBELIGHT_TAX = 0.05f;
      float WIRES_TAX = 0.075f;
      float OTHER_TAX = 0.03f;

      int Motor_count = 0;
      int Fan_count = 0;
      int Tubelight_count = 0;
      int Wires_count = 0;
      int Other_count = 0;

      float Motor_price = Product_price[0] + Product_price[0] * MOTOR_TAX;
      float Fan_price = Product_price[1] + Product_price[1] * FAN_TAX;
      float Tubelight_price = Product_price[2] + Product_price[2] * TUBELIGHT_TAX;
      float Wires_price = Product_price[3] + Product_price[3] * WIRES_TAX;
```

```
float Other_price = Product_price[4] + Product_price[4] * OTHER_TAX;

float Total_bill_ammount = 0;

System.out.println("+-------------------------------------+");
System.out.println("|       Electric Appliance Shop      |");
System.out.println("+-------------------------------------+");
System.out.println("|                                    | ");
System.out.println("| -> Press \"1\" to buy a Motor        |");
System.out.println("| -> Press \"2\" to buy a Fan          |");
System.out.println("| -> Press \"3\" to buy a Tube Light    |");
System.out.println("| -> Press \"4\" to buy a Wires        |");
System.out.println("| -> Press \"5\" to buy all other items |");
System.out.println("|                                    | ");
System.out.println("+-------------------------------------+");

System.out.println(" ");

System.out.print("How many items that you want to buy : ");
n = sc.nextInt();

int i;
int choice;

for(i=0;i<n;i++)
{

    System.out.println(" ");
    System.out.print("Select item - " + (i+1) + " that you want to buy : ");
    choice = sc.nextInt();

    switch(choice)
    {
        case 1:
        {
```

```
                Motor_count++;
                Total_bill_ammount += Motor_price;
                break;
            }

            case 2:
            {
                Fan_count++;
                Total_bill_ammount += Fan_price;
                break;
            }

            case 3:
            {
                Tubelight_count++;
                Total_bill_ammount += Tubelight_price;
                break;
            }

            case 4:
            {
                Wires_count++;
                Total_bill_ammount += Wires_price;
                break;
            }

            case 5:
            {
                Other_count++;
                Total_bill_ammount += Other_price;
                break;
            }
        }
    }
```

```
        System.out.println(" ");

        System.out.println("+-----------------------------------------------------------------+");
        System.out.println("|                    Electric Appliance Shop Bill                    |");
        System.out.println("+-----------------------------------------------------------------+");
        System.out.println("|  SR NO.  |  Product Code  |  Product Name  |  Quantity  |   Price
|");
        System.out.println("+-----------------------------------------------------------------+");
        System.out.println("|          |                |                |            |          | ");
        System.out.println("|    1.    |       " + Product_code[0] + "       | MOTOR          |     " +
Motor_count + "      | " + Motor_price + "     |");
        System.out.println("|    2.    |       " + Product_code[1] + "       | FAN            |     " +
Fan_count + "      | " + Fan_price + "      |");
        System.out.println("|    3.    |       " + Product_code[2] + "       | TUBELIGHT      |     "
+ Tubelight_count + "      | " + Tubelight_price + "      |");
        System.out.println("|    4.    |       " + Product_code[3] + "       | WIRES          |     " +
Wires_count + "      | " + Wires_price + "      |");
        System.out.println("|    5.    |       " + Product_code[4] + "       | OTHER          |     " +
Other_count + "      | " + Other_price + "      |");
        System.out.println("|          |                |                |            |          | ");
        System.out.println("|          |                |                |            |          | ");
        System.out.println("+-----------------------------------------------------------------+");
        System.out.println("|  Total Ammount                                |         " +
Total_bill_ammount + "         |");
        System.out.println("+-----------------------------------------------------------------+");

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

        System.out.println("23DCS085 Meet K. Patel");

        sc.close();
    }
}
```


OUTPUT:

```
+-------------------------------------+
|        Electric Appliance Shop      |
+-------------------------------------+
|                                     |
| -> Press "1" to buy a Motor         |
| -> Press "2" to buy a Fan           |
| -> Press "3" to buy a Tube Light    |
| -> Press "4" to buy a Wires         |
| -> Press "5" to buy all other items |
|                                     |
+-------------------------------------+

How many items that you want to buy : 5

Select item - 1 that you want to buy : 1

Select item - 2 that you want to buy : 2

Select item - 3 that you want to buy : 3

Select item - 4 that you want to buy : 4

Select item - 5 that you want to buy : 5
```

```
+-----------------------------------------------------------------------+
|                     Electric Appliance Shop Bill                      |
+-----------------------------------------------------------------------+
|  SR NO.  |  Product Code  |  Product Name  |  Quantity  |   Price    |
+-----------------------------------------------------------------------+
|          |                |                |            |            |
|    1.    |        1       | MOTOR          |     1      | 108.0      |
|    2.    |        2       | FAN            |     1      | 56.0       |
|    3.    |        3       | TUBELIGHT      |     1      | 31.5       |
|    4.    |        4       | WIRES          |     1      | 21.5       |
|    5.    |        5       | OTHER          |     1      | 10.3       |
|          |                |                |            |            |
|          |                |                |            |            |
+-----------------------------------------------------------------------+
|  Total Ammount                            |          227.3           |
+-----------------------------------------------------------------------+


23DCS085 Meet K. Patel
```

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

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

PROGRAM CODE :

```
import java.util.*;
public class JavaPractical06
{
   public static void main(String[] args)
   {

      Scanner sc = new Scanner(System.in);

      int i;
      int Days;

      int a = 0;
      int b = 1;
      int c;

      System.out.println(" ");

      System.out.print("Enter Number of Days : ");
      Days = sc.nextInt();
```

```
        System.out.println(" ");

        for(i=0;i<Days;i++)
        {
            System.out.println("Day " + (i+1) + " : " + a);

            c = a + b;
            a = b;
            b = c;
        }
        System.out.println(" ");
        System.out.println(" ");
        System.out.println("23DCS085 Meet K. Patel");
        sc.close();
    }
}
```

OUTPUT:

```
Enter Number of Days : 10

Day 1 : 0
Day 2 : 1
Day 3 : 1
Day 4 : 2
Day 5 : 3
Day 6 : 5
Day 7 : 8
Day 8 : 13
Day 9 : 21
Day 10 : 34


23DCS085 Meet K. Patel
```

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

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 :

```
import java.util.*;
public class JavaPractical07
{
   public static void main(String[] args)
   {

      Scanner sc = new Scanner(System.in);

      int n,i;
      String s1;

      System.out.println(" ");

      System.out.print("Enter Any String : ");
      s1 = sc.nextLine();

      System.out.println(" ");

      System.out.print("How many times do you want to print the substring : ");
      n = sc.nextInt();

      System.out.println(" ");
```

```
        for(i=0;i<n;++i)
        {
            System.out.print(front_times(s1));
        }

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

        System.out.println("23DCS085 Meet K. Patel");

        sc.close();

    }

    public static String front_times(String s2)
    {

        int str_len = s2.length();

        if(str_len >= 3)
        {
            return s2.substring(0,3);
        }
        else
        {
            return  s2.substring(0,str_len);
        }
    }
}
```

OUTPUT:

```
Enter Any String : Chocolate

How many times do you want to print the substring : 3

ChoChoCho

23DCS085 Meet K. Patel
```

CONCLUSION: The Java program repeatedly prints the first three characters of a user-entered string, a number of times specified by the user. It demonstrates the use of methods (`front_times`) for substring extraction and basic input/output operations using Scanner.

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 :

```
import java.util.*;
public class JavaPractical08
{
   public static void main(String[] args)
   {

      Scanner sc = new Scanner(System.in);

      int i;
      int MAX_SIZE;
      int count9;

      System.out.println(" ");

      System.out.print("Enter Size of Array : ");
      MAX_SIZE = sc.nextInt();
```

```
        int[] Array = new int[MAX_SIZE];

        System.out.println(" ");

        System.out.println("---: Enter Elements of
Array :---");

        System.out.println(" ");

        for(i=0;i<MAX_SIZE;++i)
        {
            Array[i] = sc.nextInt();
        }

        count9 = Array_count9(Array);

        System.out.println(" ");

        System.out.println("The Number of times
9 appears in the given Array is : " + count9);

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

        System.out.println("23DCS085 Meet K.
Patel");

    }

    public static int Array_count9(int[] arr)
    {
        int j,count = 0;

        for(j=0;j<arr.length;++j)
        {
```

```
            if(arr[j] == 9)
            {
                count++;
            }
        }

        return count;
    }
}
```

OUTPUT:

```
Enter Size of Array : 5

---: Enter Elements of Array :---

1
9
9
2
9

The Number of times 9 appears in the given Array is : 3


23DCS085 Meet K. Patel
```

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

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

PROGRAM CODE :

```
import java.util.*;
public class JavaPractical09
{
   public static void main(String[] args)
   {

      Scanner sc = new Scanner(System.in);

      String s1;

      System.out.println(" ");

      System.out.print("Enter Any String : ");
      s1 = sc.nextLine();

      System.out.println(" ");

      System.out.println("---: After doubling every char in string the output string is :---");

      System.out.println(" ");

      Double_char(s1);

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

      System.out.println("23DCS085 Meet K. Patel");

      sc.close();
```

```
    }

    public static boolean Double_char(String s2)
    {

        int i = 0,count = 0;
        int str_len = s2.length();

        char[] arr = s2.toCharArray();

        while(i<str_len)
        {
            if(count>1)
            {
                count = 0;
                i++;

                continue;
            }

            System.out.print(arr[i]);
            count++;
        }

        return true;
    }
}
```

OUTPUT:

```
Enter Any String : Hi-There

---: After doubling every char in string the output string is :---

HHii--TThheerree

23DCS085 Meet K. Patel
```

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

10. Perform following functionalities of the string:

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

PROGRAM CODE :

```
import java.util.*;
public class JavaPractical10
{
   public static void main(String[] args)
   {

      Scanner sc = new Scanner(System.in);

      int i;
      String s1;

      System.out.println(" ");

      System.out.print("Enter Any String : ");
      s1 = sc.nextLine();
```

```
        System.out.println(" ");

        System.out.println("Length of a given
string : " + s1.length());

        System.out.println("Lowercase form of a
given string : " + s1.toLowerCase());

        System.out.println("Uppercase form of a
given string : " + s1.toUpperCase());

        System.out.print("Reverse form of a given
string : ");

        char[] arr = s1.toCharArray();

        for(i=s1.length()-1;i>=0;i--)
        {
            System.out.print(arr[i]);
        }

        Arrays.sort(arr);

        System.out.println(" ");

        System.out.print("Sorted form of a given
string : ");

        for(i=0;i<s1.length();i++)
        {
            System.out.print(arr[i]);
        }

        System.out.println(" ");
```

```
        System.out.println(" ");

        System.out.println("23DCS085 Meet K.
Patel");


        sc.close();

    }
}
```

OUTPUT:

```
Enter Any String : MeetPatel

Length of a given string : 9
Lowercase form of a given string : meetpatel
Uppercase form of a given string : MEETPATEL
Reverse form of a given string : letaPteeM
Sorted form of a given string : MPaeeeltt

23DCS085 Meet K. Patel
```

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

11. Perform following Functionalities of the string:

"CHARUSAT UNIVERSITY"

- Find length
- Replace 'H' by 'FIRST LATTER OF YOUR NAME'
- Convert all character in lowercase

PROGRAM CODE :

```
import java.util.*;
public class JavaPractical11
{
   public static void main(String[] args)
   {

       Scanner sc = new Scanner(System.in);

       String s1;

       System.out.println(" ");

       System.out.print("Enter Any String : ");
       s1 = sc.nextLine();

       System.out.println(" ");

       System.out.println("Length of a given
string : " + s1.length());

       System.out.println(" ");

       System.out.println("Replacing \"H\" by
first letter of your name : " +
s1.replace("H","M"));

       System.out.println(" ");
```

```
        System.out.println("Lowercase form of a
given string : " + s1.toLowerCase());

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

        System.out.println("23DCS085 Meet K.
Patel");

        sc.close();
    }
}
```

OUTPUT:

```
Enter Any String : CHARUSAT UNIVERSITY

Length of a given string : 19

Replacing "H" by first letter of your name : CMARUSAT UNIVERSITY

Lowercase form of a given string : charusat university


23DCS085 Meet K. Patel
```

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

Part – 3

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 :

```
import java.util.*;
class Practical12
{
   public static void main(String[] args)
   {

      System.out.print("Enter ammount in pound
: " + args[0]);

      int temp = Integer.parseInt(args[0]);
      int pound;
      int rupees;

      pound = temp;

      rupees = pound*100;

      System.out.println(" ");

      System.out.println("Ammount in rupees : "
+ rupees);
```

```
        System.out.println(" ");
        System.out.println(" ");
        System.out.println("23DCS085 Meet K.
Patel");

    }
}
```

OUTPUT:

```
D:\Meet Patel SY\Java File Work\Java Practical Set - 03>javac Practical12.java

D:\Meet Patel SY\Java File Work\Java Practical Set - 03>java Practical12 10
Enter ammount in pound : 10
Ammount in rupees : 1000


23DCS085 Meet K. Patel
```

CONCLUSION: In this practical we learned about how take inputs,compile java program and get output with the use of command line argument in java programming.This type of programming also known as “Console programming”.

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 :

```
import java.util.*;

class Employee
{
   private String First_name;
   private String Last_name;
   private double Monthly_salary;
   private double Yearly_salary;

   Scanner sc = new Scanner(System.in);

   public Employee()
   {
      First_name = "Employee_Name";
      Last_name = "Employee_surname";
      Monthly_salary = 0.0;
   }

   public void get_Data()
   {

      System.out.println(" ");

      System.out.print("Enter Employee's First
Name : ");
      First_name = sc.nextLine();

      System.out.print("Enter Employee's Last
Name : ");
      Last_name = sc.nextLine();

      System.out.print("Enter Employee's
Monthly Salary : ");
```

```
        Monthly_salary = sc.nextInt();

        Yearly_salary = Monthly_salary*12;
    }

    public void put_Data()
    {

        if(Monthly_salary < 0)
        {
            Monthly_salary = 0.0;
        }

        System.out.println(" ");

        System.out.println("---: Employee's
Detail's :---");

        System.out.println(" ");

        System.out.println("Employee's First
Name : " + First_name);
        System.out.println("Employee's Last Name
: " + Last_name);
        System.out.println("Employee's Monthly
Salary : " + Monthly_salary);
        System.out.println("Employee's Yearly
Salary : " + Yearly_salary);
    }

    public void Employee_salary_increament()
    {
        double increament_ration;
        double incremented_salary;
        System.out.println(" ");
```

```
        System.out.print("Enter Increament Ratio
of Salary : ");
        increament_ration = sc.nextInt();
        incremented_salary =
(increament_ration/100)*Yearly_salary;
        Yearly_salary += incremented_salary;
    }
}

class Practical13
{
    public static void main(String[] args)
    {
        Employee E1 = new Employee();
        E1.put_Data();

        Employee E2 = new Employee();
        E2.get_Data();
        E2.put_Data();

        E2.Employee_salary_increament();
        E2.put_Data();

        System.out.println(" ");
        System.out.println(" ");
        System.out.println("23DCS085 Meet K.
Patel");
    }
}
```

OUTPUT:

```
---: Employee's Detail's :---

Employee's First Name : Employee_Name
Employee's Last Name : Employee_surname
Employee's Monthly Salary : 0.0
Employee's Yearly Salary : 0.0

Enter Employee's First Name : Meet
Enter Employee's Last Name : Patel
Enter Employee's Monthly Salary : 100000

---: Employee's Detail's :---

Employee's First Name : Meet
Employee's Last Name : Patel
Employee's Monthly Salary : 100000.0
Employee's Yearly Salary : 1200000.0

Enter Increament Ratio of Salary : 10

---: Employee's Detail's :---

Employee's First Name : Meet
Employee's Last Name : Patel
Employee's Monthly Salary : 100000.0
Employee's Yearly Salary : 1320000.0


23DCS085 Meet K. Patel
```

CONCLUSION: From this java program we learned about class , Default constructor of class , class methods , class objects , how to declare class objects , how to access class methods using class objects etc.

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

PROGRAM CODE :

```
import java.util.*;

class Date
{
   private int Day;
   private int Month;
   private int Year;

   Scanner sc = new Scanner(System.in);

   public Date()
   {
      Day = 0;
      Month = 0;
      Year = 0;
   }
```

```
public void set_Data(int D,int M,int Y)
{
   Day = D;
   Month = M;
   Year = Y;
}

public void get_Data()
{
   System.out.println(" ");

   for(int i=0;i<1;i++)
   {

   System.out.print("Enter Date : ");
   Day = sc.nextInt();

   if(Day<=0 || Day > 31)
   {
      System.out.println(" ");

      System.out.println("Entered Invalid Date");

      System.out.println(" ");

      i--;
   }

   }

   for(int i=0;i<1;i++)
   {

   System.out.print("Enter Month : ");
   Month = sc.nextInt();

   if(Month <= 0 || Month > 12)
   {
      System.out.println(" ");

      System.out.println("Entered Invalid Month");
```

```
            System.out.println(" ");

            i--;
        }
        }

        for(int i=0;i<1;i++)
        {

        System.out.print("Enter Year : ");
        Year = sc.nextInt();

        if(Year <= 0)
        {
            System.out.println(" ");

            System.out.println("Entered Invalid Year");

            System.out.println(" ");

            i--;
        }
        }

    }

    public void Display_Data()
    {
        System.out.println(" ");

        System.out.println("Date : " + Day + "/" + Month + "/" + Year);
    }

}

class Practical14
{
    public static void main(String[] args)
    {
    Date D1 = new Date();
```

```
    Date D2 = new Date();

    D2.get_Data();

    D1.Display_Data();
    D2.Display_Data();

    System.out.println(" ");
    System.out.println(" ");
    System.out.println("23DCS085 Meet K. Patel");

    }

}
```

OUTPUT:

\

```
Enter Date : 50

Entered Invalid Date

Enter Date : 22
Enter Month : 15

Entered Invalid Month

Enter Month : 07
Enter Year : 2024

Date : 0/0/0

Date : 22/7/2024


23DCS085 Meet K. Patel
```

CONCLUSION: This Java program teaches us about getter – setter methods to take input from the user and display output on the output screen in java programming.

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

PROGRAM CODE :

```
import java.util.*;
class Area
{

   private float Length;
   private float Breadth;
   private float rec_Area;

   public Area()
   {
      Length = 0;
      Breadth = 0;
   }

   public Area(float X,float Y)
   {
      Length = X;
```

```
        Breadth = Y;
    }

    public float returnArea()
    {
        rec_Area = Length*Breadth;

        return rec_Area;
    }

}

class Practical15
{
    public static void main(String[] args)
    {

        Scanner sc = new Scanner(System.in);

        float A,B,area1,area2;

        Area A1 = new Area();

        System.out.println(" ");

        System.out.print("Enter the value of
Length for Rectangle : ");
        A = sc.nextFloat();

        System.out.print("Enter the value of
Breadth for Rectangle : ");
        B = sc.nextFloat();

        Area A2 = new Area(A,B);
```

```
        area1 = A1.returnArea();
        area2 = A2.returnArea();

        System.out.println(" ");

        System.out.println("Area of Rectangle : " +
area1);

        System.out.println(" ");

        System.out.println("Area of Rectangle : " +
area2);

        System.out.println(" ");
        System.out.println(" ");
        System.out.println("23DCS085 Meet K.
Patel");

        sc.close();

    }

}
```

OUTPUT:

```
Enter the value of Length for Rectangle : 2
Enter the value of Breadth for Rectangle : 2.5

Area of Rectangle : 0.0

Area of Rectangle : 5.0


23DCS085 Meet K. Patel
```

CONCLUSION: From this practical we learned about constructor overloading in java programming, we used parameterized constructor to initialize the variables of class.

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

PROGRAM CODE :

```
import java.util.*;

class Complex
{

   private int Real;
   private int Img;

   Complex()
   {
      Real = 0;
      Img = 0;
   }
```

```
public void get_Data()
{

    Scanner sc = new Scanner(System.in);

    System.out.println(" ");

    System.out.print("Enter Real Part : ");
    Real = sc.nextInt();

    System.out.print("Enter Imaginary Part : ");
    Img = sc.nextInt();

}

public void Display_Data()
{

    System.out.println(" ");

    System.out.println("Entered Complex Number : " + Real + " + " + Img + "i");

}

public void Addition(Complex C)
{
    System.out.println(" ");
```

```
    System.out.println("---: Addition of Complex Numbers :---");

    System.out.println(" ");

    System.out.println("Sum = " + (Real + C.Real) + " + (" + (Img + C.Img) + ")i");

  }

  public void Difference(Complex C)
  {
    System.out.println(" ");

    System.out.println("---: Difference of Complex Numbers :---");

    System.out.println(" ");

    System.out.println("Difference = " + (Real + C.Real) + " + (" + (Img + C.Img) + ")i");

  }

  public void Product(Complex C)
  {
    System.out.println(" ");

    System.out.println("---: Product of Complex Numbers :---");

    System.out.println(" ");
```

```
        System.out.println("Product = " + ((Real * C.Real) - (Img * C.Img)) + " + (" + ((Real
* C.Img) + (Img * C.Real)) + ")i");

    }

}

class Practical16
{

    public static void main(String[] args)
    {

        Complex C1 = new Complex();

        C1.Display_Data();

        Complex C2 = new Complex();
        C2.get_Data();

        Complex C3 = new Complex();
        C3.get_Data();

        C2.Display_Data();
        C3.Display_Data();

        C2.Addition(C3);
        C2.Difference(C3);
```

```
        C2.Product(C3);

        System.out.println(" ");
        System.out.println(" ");
        System.out.println("23DCS085 Meet K. Patel");

    }
}
```

OUTPUT:

```
Entered Complex Number : 0 + (0)i

Enter Real Part : 1
Enter Imaginary Part : -2

Enter Real Part : 1
Enter Imaginary Part : -2

Entered Complex Number : 1 + (-2)i

Entered Complex Number : 1 + (-2)i

---: Addition of Complex Numbers :---

Sum = 2 + (-4)i

---: Difference of Complex Numbers :---

Difference = 2 + (-4)i

---: Product of Complex Numbers :---

Product = -3 + (-4)i


23DCS085 Meet K. Patel
```

	CONCLUSION: In this practical we learned about how to define class methods using class reference copy also in this program we perform binary operations like Addition,Subtraction and Multiplication of Complex Numbers.

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 :
	
	class Parent
	{
	public void Print_Parent()
	{
	System.out.println("\nThis is Parent
	Class");
	}
	}
	
	class Child extends Parent
	{
	public void Print_Child()
	{
	System.out.println("\nThis is Child
	Class");
	}
	}
	
	class Practical17
	{
	public static void main(String[] args)
	{
	@SuppressWarnings("unused")

```
        Parent P1 = new Parent();
        Child C1 = new Child();

        C1.Print_Parent();
        C1.Print_Child();

        System.out.println(" ");
        System.out.println(" ");
        System.out.println("23DCS085 Meet K.
Patel");

    }
}
```

OUTPUT:

```
This is Parent Class

This is Child Class


23DCS085 Meet K. Patel
```

CONCLUSION: From this practical we learned about Single Inheritance with his syntax.

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.*;
class Member
{

    Scanner sc = new Scanner(System.in);

    protected String Name;
    protected int Age;
    protected long Phone_Number;
    protected String Address;
    protected int Salary;

}

class Manager extends Member
{
    private String Department;

    void get_Data()
    {
        System.out.println("---: Enter Manager Detail's :---");

        System.out.print("\nEnter Name : ");
        Name = sc.nextLine();

        System.out.print("Enter Age : ");
```

```
        Age = sc.nextInt();

        System.out.print("Enter Phone Number : ");
        Phone_Number = sc.nextLong();
        sc.nextLine();

        System.out.print("Enter Address : ");
        Address = sc.nextLine();

        System.out.print("Enter Department : ");
        Department = sc.nextLine();

        System.out.print("Enter Salary : ");
        Salary = sc.nextInt();
    }

    void Display_Data()
    {
        System.out.println("\n---: Manager Detail's :---");

        System.out.println("\nName : " + Name);
        System.out.println("Age : " + Age);
        System.out.println("Phone Number : " + Phone_Number);
        System.out.println("Address : " + Address);
        System.out.println("Department : " + Department);
    }

    void Print_Salary()
    {
        System.out.println("\nManager Salary : " + Salary);
    }
}

class Employee extends Member
{
```

```
private String Specialization;

void get_Data2()
{
    System.out.println("---: Enter Employee Detail's :---");

    System.out.print("\nEnter Name : ");
    Name = sc.nextLine();

    System.out.print("Enter Age : ");
    Age = sc.nextInt();

    System.out.print("Enter Phone Number : ");
    Phone_Number = sc.nextLong();
    sc.nextLine();

    System.out.print("Enter Address : ");
    Address = sc.nextLine();

    System.out.print("Enter Specialization : ");
    Specialization = sc.nextLine();

    System.out.print("Enter Salary : ");
    Salary = sc.nextInt();
}

void Display_Data2()
{
    System.out.println("\n\n---: Employee Detail's :---");

    System.out.println("\nName : " + Name);
    System.out.println("Age : " + Age);
    System.out.println("Phone Number : " + Phone_Number);
    System.out.println("Address : " + Address);
    System.out.println("Specialization : " + Specialization);
```

```
    }

    void Print_Salary2()
    {
        System.out.println("\nEmployee Salary : " + Salary);
    }
}

class Practical18
{
    public static void main(String[] args)
    {

        Manager M1 = new Manager();
        Employee E1 = new Employee();

        M1.get_Data();
        E1.get_Data2();

        M1.Display_Data();
        E1.Display_Data2();

        System.out.println(" ");

        M1.Print_Salary();
        E1.Print_Salary2();

        System.out.println(" ");
        System.out.println(" ");
        System.out.println("23DCS085 Meet K. Patel");
    }
}
```

OUTPUT:

```
---: Enter Manager Detail's :---

Enter Name : Meet Patel
Enter Age : 19
Enter Phone Number : 8320635225
Enter Address : Surat
Enter Department : Development
Enter Salary : 1200000

---: Enter Employee Detail's :---

Enter Name : ABCD
Enter Age : 18
Enter Phone Number : 1234567890
Enter Address : Ahemedabad
Enter Specialization : AI/ML
Enter Salary : 700000

---: Manager Detail's :---

Name : Meet Patel
Age : 19
Phone Number : 8320635225
Address : Surat
Department : Development


---: Employee Detail's :---

Name : ABCD
Age : 18
Phone Number : 1234567890
Address : Ahemedabad
Specialization : AI/ML
```

```
Manager Salary : 1200000

Employee Salary : 700000


23DCS085 Meet K. Patel
```

CONCLUSION: From this practical we learned about Hierarchical Inheritance with example code.

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

PROGRAM CODE :

```
import java.util.*;
class Rectangle
{
   float Length;
   float Breadth;

   public Rectangle()
   {
      Length = 0;
      Breadth = 0;
   }
```

```
    public Rectangle(float Length,float Breadth)
    {
        this.Length = Length;
        this.Breadth = Breadth;
    }

    void Perimeter()
    {
        System.out.println("\nPerimeter of Rectangle is : " + 2*(Length+Breadth));
        System.out.println("Perimeter of Square is : " +  4*Length);
    }

    void Area()
    {
        System.out.println("\nArea of Rectangle is : " + Length*Breadth);
        System.out.println("Area of Square is : " + Length*Length);
    }
}

class Square extends Rectangle
{
    public Square(float S)
    {
        super(S,S);
    }
}

class Practical19
{
    public static void main(String[] args)
    {

        Scanner sc = new Scanner(System.in);

        int no_of_object;
```

```
        float L;

        System.out.println(" ");

        System.out.print("Enter the number of Object's that you want to create : ");
        no_of_object = sc.nextInt();

        Square[] S = new Square[no_of_object];

        for(int i=0;i<no_of_object;i++)
        {
            System.out.print("\nEnter the Value of Length : ");
            L = sc.nextFloat();

            S[i] = new Square(L);

            S[i].Perimeter();
            S[i].Area();
        }

        sc.close();

        System.out.println(" ");
        System.out.println(" ");
        System.out.println("23DCS085 Meet K. Patel");

    }
}
```

OUTPUT:

```
Enter the number of Object's that you want to create : 2

Enter the Value of Length : 5

Perimeter of Rectangle is : 20.0
Perimeter of Square is : 20.0

Area of Rectangle is : 25.0
Area of Square is : 25.0

Enter the Value of Length : 3.2

Perimeter of Rectangle is : 12.8
Perimeter of Square is : 12.8

Area of Rectangle is : 10.240001
Area of Square is : 10.240001


23DCS085 Meet K. Patel
```

CONCLUSION: In this practical we make use of super keyword which is used to call the constructor of parent class from child class,super keyword must written first.

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

PROGRAM CODE :

```
class Shape
{
   void Print_Shape()
   {
      System.out.println("\nThis is Shape");
   }
}

class Rectangle extends Shape
{
   void Print_Rectangle()
   {
      System.out.println("\nThis is Rectangular Shape");
   }
}

class Circle extends Shape
{
   void Print_Circle()
   {
      System.out.println("\nThis is Circular Shape");
   }
}

class Square extends Rectangle
{
   void Print_Square()
   {
      System.out.println("\nSquare is Rectangle");
   }
}
```

```
class Practical20
{
   public static void main(String[] args)
   {

      Square S1 = new Square();

      S1.Print_Shape();
      S1.Print_Rectangle();
      S1.Print_Square();

      System.out.println(" ");
      System.out.println(" ");
      System.out.println("23DCS085 Meet K. Patel");

   }
}
```

OUTPUT:

```
This is Shape

This is Rectangular Shape

Square is Rectangle


23DCS085 Meet K. Patel
```

CONCLUSION: From this practical we learned about Multi Level Inheritance with Example code.

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

PROGRAM CODE :

```
class Degree
{
   public void get_Degree()
   {
      System.out.println("I got a Degree");
   }
}

class Undergraduate extends Degree
{
   public void Print()
   {
      System.out.println("I am Undergraduate");
   }
}

class Postgraduate extends Degree
{
   public void Print()
   {
      System.out.println("I am Postgraduate");
   }
}

class Practical21
{
   public static void main(String[] args)
   {
```

```
        Degree D = new Degree();
        Undergraduate U = new Undergraduate();
        Postgraduate P = new Postgraduate();

        D.get_Degree();
        U.Print();
        P.Print();

        System.out.println(" ");
        System.out.println(" ");
        System.out.println("23DCS085 Meet K. Patel");

    }
}
```

OUTPUT:

```
I got a Degree
I am Undergraduate
I am Postgraduate


23DCS085 Meet K. Patel
```

CONCLUSION: From this practical we learned about method overloading in inheritance. In method overloading we have same name and same parameter method in both child and parent class we have to call this method using their respective class name.

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 :

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

class calledMyCalculator implements AdvancedArithmatic
{
   public int divisor_sum(int n)
   {

      int temp = 0;

      for(int i=1;i<=n;i++)
      {
         if(n % i == 0)
         {
            temp += i;
         }
      }

      return temp;
   }
}

class Practical22
{
   public static void main(String[] args)
   {

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

```
        calledMyCalculator C1 = new calledMyCalculator();

        int A,B;

        System.out.println(" ");

        System.out.print("Enter Any Number : ");
        A = sc.nextInt();

        B = C1.divisor_sum(A);

        System.out.println(" ");

        System.out.println("Sum of all Divisors of " + A + " is : " + B);

        System.out.println(" ");
        System.out.println(" ");
        System.out.println("23DCS085 Meet K. Patel");

        sc.close();
    }
}
```

OUTPUT:

```
Enter Any Number : 6

Sum of all Divisors of 6 is : 12


23DCS085 Meet K. Patel
```

CONCLUSION: From this practical we learned about interface and apply it with class with example code.

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 :

```
import java.util.Scanner;

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

// Circle class implementing Shape interface
class Circle implements Shape {
    private double radius;
    private String color;

    public Circle(double rad, String col) {
        radius = rad;
        color = col;
    }

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

    @Override
    public double getArea() {
        return (3.14 * radius * radius);
    }
}

// Rectangle class implementing Shape interface
```

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

    public Rectangle(double len, double wid, String col) {
        length = len;
        width = wid;
        color = col;
    }

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

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

// Sign class
class Sign {
    private Shape backgroundShape;
    private String text;

    public Sign(Shape BShape, String tex) {
        backgroundShape = BShape;
        text = tex;
    }

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

public class Practical23 {
```

```
    public static void main(String[] args) {
        // Create a Circle
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter radius of circle :");
        int x =sc.nextInt();
        sc.nextLine();
        System.out.print("Enter color of circle :");
        String y = sc.nextLine();
        Circle circle = new Circle(x, y);

        // Create a Rectangle
        System.out.print("Enter length:");
        int a =sc.nextInt();
        System.out.print("Enter width:");
        int b =sc.nextInt();
        sc.nextLine();
        System.out.print("Enter color:");
        String c =sc.nextLine();
        Rectangle rectangle = new Rectangle(a,b,c);

        // Create signs using the shapes


        Sign circleSign = new Sign(circle, "Welcome to the Campus!");
        Sign rectangleSign = new Sign(rectangle, "Library ->");

        // Display the signs
        circleSign.displaySign();
        rectangleSign.displaySign();

        System.out.println(" ");
        System.out.println(" ");
        System.out.println("23DCS085 Meet K. Patel");

        sc.close();
    }
}
```

OUTPUT:

```
Enter radius of circle :5
Enter color of circle :red
Enter length:5
Enter width:5
Enter color:blue
Sign:
Background Shape Color: red
Background Shape Area: 78.5
Text: Welcome to the Campus!
Sign:
Background Shape Color: blue
Background Shape Area: 25.0
Text: Library ->


23DCS085 Meet K. Patel
```

CONCLUSION: This program shows how interfaces work. The Shape interface is used by Circle and Rectangle classes to provide color and area. The Sign class uses these shapes to display signs with their details. It shows how interfaces help share common methods between different classes.

Part – 5

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

PROGRAM CODE :

```
import java.util.*;
class Practical24
{
   public static void main(String[] args)
   {

      Scanner sc = new Scanner(System.in);

      int x,y;

      try{
         System.out.println(" ");

         System.out.print("Enter Value of X : ");
         x = sc.nextInt();

         System.out.println(" ");

         System.out.print("Enter Value of Y : ");
         y = sc.nextInt();

         System.out.println(" ");
         System.out.println(x + "/" + y + " = " +
x/y);
```

```
        }
        catch(InputMismatchException e)
        {
            System.out.println(" ");
            System.out.println("Entered Numbers
are not an Integers");
        }
        catch(ArithmeticException e)
        {
            System.out.println(" ");
            System.out.println("Number is not
Divide by Zero");
        }

        System.out.println(" ");
        System.out.println(" ");
        System.out.println("23DCS085 Meet K.
Patel");

        sc.close();
    }
}
```

OUTPUT:

```
Enter Value of X : 2.5

Entered Numbers are not an Integers


23DCS085 Meet K. Patel
```

```
Enter Value of X : 5

Enter Value of Y : 0


Number is not Divide by Zero


23DCS085 Meet K. Patel
```

```
Enter Value of X : 5

Enter Value of Y : 5

5/5 = 1


23DCS085 Meet K. Patel
```

CONCLUSION: From this practical we learned about how to handle the exception using try-catch to maintain normal flow of code.

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

PROGRAM CODE :

```
import java.util.*;
class Practical25
{
   public static void main(String[] args)
   {

      Scanner sc = new Scanner(System.in);

      int x,y;

      try
      {
      System.out.println(" ");

      System.out.print("Enter Value of X : ");
      x = sc.nextInt();

      System.out.print("Enter Value of Y : ");
      y = sc.nextInt();

      if(y == 0)
      {
         throw new ArithmeticException("Number is Not Divided By Zero");
      }

      System.out.println(" ");
      System.out.println(x + "/" + y + " = " + x/y);

      }
      catch(InputMismatchException e)
      {
```

```
            System.out.println(" ");
            System.out.println("Exception : " + e);
        }
        catch(ArithmeticException e)
        {
            System.out.println(" ");
            System.out.println("Exception : " + e);
        }

        System.out.println(" ");
        System.out.println(" ");
        System.out.println("23DCS085 Meet K. Patel");
    }
}
```

OUTPUT:

```
Enter Value of X : 1.5

Exception : java.util.InputMismatchException


23DCS085 Meet K. Patel
```

```
Enter Value of X : 5
Enter Value of Y : 0

Exception : java.lang.ArithmeticException: Number is Not Divided By Zero


23DCS085 Meet K. Patel
```

```
Enter Value of X : 5
Enter Value of Y : 5

5/5 = 1


23DCS085 Meet K. Patel
```

CONCLUSION: From this practical we learned about how to throw exception using throw keyword and also learned to manage that exception using try-catch to maintain the normal flow of code.

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 :

```
import java.io.IOException;
import java.util.*;
class Practical26
{

    int balance = 5000;
    int withDrawlAmount;

    Scanner sc = new Scanner(System.in);

    void checkInteger()throws IOException
    {
        System.out.print("Enter Withdrawl Ammount that you want to Withdraw : ");
```

```
        if(sc.hasNextInt())
        {
            withDrawlAmount = sc.nextInt();
        }
        else
        {
            throw new IOException("Enter Amount is Not an Integer Value");
        }
    }

    void checkNegative()throws IOException
    {
        if(withDrawlAmount < 0)
        {
            throw new IOException("Entered Amount is Negative Value");
        }
    }

    void checkInsufficientBalance()throws ArithmeticException
    {
        if(balance < withDrawlAmount)
        {
            throw new ArithmeticException("Insufficient Balance");
        }
    }

    void checkwithDrawlAmount()throws ArithmeticException
    {
        if(!(withDrawlAmount % 100 == 0 || withDrawlAmount % 50 == 0 || withDrawlAmount
% 200 == 0 || withDrawlAmount % 500 == 0 || withDrawlAmount % 10 == 0 ||
withDrawlAmount % 20 == 0))
        {
            throw new ArithmeticException("Entered WithDrawl Amount is not a Multiple of
10,20,50,100,200 OR 500");
        }
```

```
    }

    void printCurrentBalance()
    {
        int currentBalance = balance - withDrawlAmount;

        if((balance != currentBalance) && (withDrawlAmount > 0) && (withDrawlAmount %
100 == 0 || withDrawlAmount % 50 == 0 || withDrawlAmount % 200 == 0 ||
withDrawlAmount % 500 == 0 || withDrawlAmount % 10 == 0 || withDrawlAmount % 20
== 0))
        {
            System.out.println(" ");
            System.out.println("Amount is Withdrawed Successfully");
            System.out.println(" ");
            System.out.println("Current Balance : " + currentBalance);
        }
        else
        {
            System.out.println(" ");
            System.out.println("Invalid WithDrawl Amount Entered");
        }
    }

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

        System.out.println(" ");

        Practical26 P1 = new Practical26();

        try
        {
            P1.checkInteger();
        }
        catch(IOException e)
```

```
{
    System.out.println(" ");
    System.out.println("Exception : " + e);
}

try
{
    P1.checkNegative();
}
catch(IOException e)
{
    System.out.println(" ");
    System.out.println("Exception : " + e);
}

try
{
    P1.checkInsufficientBalance();
}
catch(ArithmeticException e)
{
    System.out.println(" ");
    System.out.println("Exception : " + e);
}

try
{
    P1.checkwithDrawlAmount();
}
catch(ArithmeticException e)
{
    System.out.println(" ");
    System.out.println("Exception : " + e);
}
```

```
        P1.printCurrentBalance();

        System.out.println(" ");
        System.out.println(" ");
        System.out.println("23DCS085 Meet K. Patel");

    }
}
```

OUTPUT:

```
Enter Withdrawl Ammount that you want to Withdraw : 2.5

Exception : java.io.IOException: Enter Amount is Not an Integer Value

Amount is Withdrawed Successfully

Current Balance : 5000


23DCS085 Meet K. Patel
```

```
Enter Withdrawl Ammount that you want to Withdraw : -2000

Exception : java.io.IOException: Entered Amount is Negative Value
```

```
Enter Withdrawl Ammount that you want to Withdraw : 23

Exception : java.lang.ArithmeticException: Entered WithDrawl Amount is not a Multiple of 10,20,50,100,200 OR 500
```

```
Enter Withdrawl Ammount that you want to Withdraw : 6000

Exception : java.lang.ArithmeticException: Insufficient Balance
```

```
Enter Withdrawl Ammount that you want to Withdraw : 3000

Amount is Withdrawed Successfully

Current Balance : 2000


23DCS085 Meet K. Patel
```

CONCLUSION: From this practical we learned about throws keyword and also we perform a code to handle checked and unchecked exception using throws and throw keyword.

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 A extends Thread
{
    public void run()
    {
        System.out.println("Hello World");
    }
}

class B implements Runnable
{
    public void run()
    {
        System.out.println("Hello World");
    }
}

public class Practical32
{
    public static void main(String[] args)
    {

        A t1 = new A();

        t1.start();
```

```
        B obj = new B();
        Thread t2 = new Thread(obj);

        t2.start();

        System.out.println(" ");
        System.out.println("23DCS085 Meet K.
Patel");
    }
}
```

OUTPUT:

```
Hello World

23DCS085 Meet K. Patel
Hello World
```

CONCLUSION: From this practical we learned about how to make use of thread and run method with implementing runnable interface and extending Thread class.

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 :

```
class SumTask implements Runnable {
    private int start;
    private int end;
    private int[] result;
    private int index;

    public SumTask(int start, int end, int[] result, int index) {
        this.start = start;
        this.end = end;
        this.result = result;
        this.index = index;
    }

    @Override
    public void run() {
        int sum = 0;
        for (int i = start; i <= end; i++) {
            sum += i;
        }
        result[index] = sum;
    }
}

public class Practical33 {
    public static void main(String[] args) {
        if (args.length < 2) {
            System.out.println("Please provide two arguments: N and the number of threads.");
            return;
        }
```

```
int N = Integer.parseInt(args[0]);
int numThreads = Integer.parseInt(args[1]);

int[] result = new int[numThreads];

int range = N / numThreads;
int remainder = N % numThreads;

Thread[] threads = new Thread[numThreads];

int start = 1;
for (int i = 0; i < numThreads; i++) {
    int end = start + range - 1;

    if (i == numThreads - 1) {
        end += remainder;
    }

    threads[i] = new Thread(new SumTask(start, end, result, i));
    threads[i].start();

    start = end + 1;
}

try {
    for (Thread thread : threads) {
        thread.join();
    }
} catch (InterruptedException e) {
    System.out.println("Thread interrupted: " + e.getMessage());
}

int finalSum = 0;
for (int sum : result) {
```

```
            finalSum += sum;
        }

        System.out.println("The sum of the first " + N + " numbers is: " + finalSum);

        System.out.println(" ");
        System.out.println("23DCS085 Meet K. Patel");
    }
}
```

OUTPUT:

```
PS D:\Meet Patel SY\Java File Work\Java Practical Set - 07> javac Practical33.java
PS D:\Meet Patel SY\Java File Work\Java Practical Set - 07> java Practical33.java
Please provide two arguments: N and the number of threads.
PS D:\Meet Patel SY\Java File Work\Java Practical Set - 07> java Practical33 100 4
The sum of the first 100 numbers is: 5050

23DCS085 Meet K. Patel
```

CONCLUSION: From this practical we learned about how to divide a task of calculating sum of N number in N number of threads

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

PROGRAM CODE :

```
import java.util.Random;

class RandomNumberGenerator extends Thread {
    public static int number;
```

```
    public void run() {
        Random rand = new Random();
        while (true) {
            number = rand.nextInt(100); // Generates a random number between 0 and 99
            System.out.println("Generated Number: " + number);
            try {
                Thread.sleep(1000); // Pauses for 1 second
            } catch (InterruptedException e) {
                System.out.println("RandomNumberGenerator interrupted.");
            }
        }
    }
}

class SquareCalculator extends Thread {
    public void run() {
        while (true) {
            if (RandomNumberGenerator.number % 2 == 0) {
                int square = RandomNumberGenerator.number *
RandomNumberGenerator.number;
                System.out.println("Square of " + RandomNumberGenerator.number + " is: " +
square);
            }
            try {
                Thread.sleep(1000); // Pauses for 1 second
            } catch (InterruptedException e) {
                System.out.println("SquareCalculator interrupted.");
            }
        }
    }
}

class CubeCalculator extends Thread {
    public void run() {
```

```
        while (true) {
            if (RandomNumberGenerator.number % 2 != 0) {
                int cube = RandomNumberGenerator.number * RandomNumberGenerator.number
* RandomNumberGenerator.number;
                    System.out.println("Cube of " + RandomNumberGenerator.number + " is: " +
cube);
            }
            try {
                Thread.sleep(1000); // Pauses for 1 second
            } catch (InterruptedException e) {
                System.out.println("CubeCalculator interrupted.");
            }
        }
    }
}

public class Practical34 {
    public static void main(String[] args) {
                            RandomNumberGenerator randomNumberGenerator = new
RandomNumberGenerator();
        SquareCalculator squareCalculator = new SquareCalculator();
        CubeCalculator cubeCalculator = new CubeCalculator();

        randomNumberGenerator.start();
        squareCalculator.start();
        cubeCalculator.start();

        System.out.println(" ");
        System.out.println("23DCS085 Meet K. Patel");
    }
}
```

OUTPUT:

```
23DCS085 Meet K. Patel
Generated Number: 89
Square of 0 is: 0
Generated Number: 6
Cube of 6 is: 704969
Generated Number: 83
Square of 6 is: 36
Cube of 83 is: 571787
Generated Number: 72
Square of 72 is: 5184
Generated Number: 55
Square of 72 is: 5184
Cube of 55 is: 166375
Generated Number: 34
Square of 34 is: 1156
Generated Number: 87
Cube of 87 is: 658503
Generated Number: 10
Square of 10 is: 100
Generated Number: 41
Square of 10 is: 100
Cube of 41 is: 68921
Generated Number: 28
Generated Number: 52
Square of 52 is: 2704
Generated Number: 40
```

CONCLUSION: From this practical we learned about throws keyword and also we perform a code to handle checked and unchecked exception using throws and throw keyword.

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

PROGRAM CODE :

```
import java.util.*;
public class Practical35
{
    public static void main(String[] args)
    {

        Scanner sc = new Scanner(System.in);

        int n;

        System.out.println(" ");

        System.out.print("Enter the Value of n : ");
        n = sc.nextInt();

        System.out.println(" ");

        try
        {
            for(int i=1;i<=n;i++)
            {
                Thread.sleep(1000);

                System.out.println(i);
            }
        }
        catch(Exception e)
        {
            System.out.println(" ");
            System.out.println("Exception : " + e);
```

```
        }

        System.out.println(" ");
        System.out.println("23DCS085 Meet K. Patel");

        sc.close();
    }
}
```

OUTPUT:

```
Enter the Value of n : 5

1
2
3
4
5

23DCS085 Meet K. Patel
```

CONCLUSION: From this practical we learned about sleep method of Thread Class also we make use of that method is above code and print a number after 1 Second Sleep.

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

PROGRAM CODE :

```
class A extends Thread
{
    public void run()
    {

    }
}
```

```
public class Practical36
{
   public static void main(String[] args)
   {
      A First = new A();
      A Second = new A();
      A Third = new A();

      System.out.println(" ");

      System.out.println("---: Before Setting Name And Priority of Thread :---");

      System.out.println(" ");

       System.out.println("[1.] Name : " + First.getName() + " , Priority of Thread : " +
First.getPriority());
       System.out.println("[2.] Name : " + Second.getName() + " , Priority of Thread : " +
Second.getPriority());
       System.out.println("[3.] Name : " + Third.getName() + " , Priority of Thread : " +
Third.getPriority());

      System.out.println(" ");

      First.setName("FIRST");
      Second.setName("SECOND");
      Third.setName("THIRD");

      First.setPriority(3);
      Second.setPriority(5);
      Third.setPriority(7);

      System.out.println("---: After Setting Name And Priority of Thread :---");

      System.out.println(" ");
```

```
        System.out.println("[1.] Name : " + First.getName() + " , Priority of Thread : " + First.getPriority());
        System.out.println("[2.] Name : " + Second.getName() + " , Priority of Thread : " + Second.getPriority());
        System.out.println("[3.] Name : " + Third.getName() + " , Priority of Thread : " + Third.getPriority());

      System.out.println(" ");
      System.out.println("23DCS085 Meet K. Patel");
   }
}
```

OUTPUT:

```
---: Before Setting Name And Priority of Thread :---

[1.] Name : Thread-0 , Priority of Thread : 5
[2.] Name : Thread-1 , Priority of Thread : 5
[3.] Name : Thread-2 , Priority of Thread : 5

---: After Setting Name And Priority of Thread :---

[1.] Name : FIRST , Priority of Thread : 3
[2.] Name : SECOND , Priority of Thread : 5
[3.] Name : THIRD , Priority of Thread : 7

23DCS085 Meet K. Patel
```

CONCLUSION: From this practical we learned about setName and setPriority methods of Thread class also we make use of that methods in above given code.

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

PROGRAM CODE :

```
class SharedBuffer {
    int item; // A shared place for the item
    boolean isProduced = false; // Whether the item is produced or not

    public synchronized void produce() throws InterruptedException {
        if (isProduced) {
            return;  // If an item is already produced, do nothing
        }
        item = (int) (Math.random() * 100); // Produce a random item
        System.out.println("Produced: " + item);
        isProduced = true; // Mark the item as produced
        notify(); // Notify the consumer that the item is ready
    }

    public synchronized void consume() throws InterruptedException {
        if (!isProduced) {
            return;  // If no item is produced, do nothing
        }
        System.out.println("Consumed: " + item); // Consume the item
        isProduced = false; // Mark that the item has been consumed
        notify(); // Notify the producer that the buffer is now empty
    }
}

class Producer extends Thread {
    SharedBuffer buffer;

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

```
    @Override
    public void run() {
        try {
            for (int i = 0; i < 10; i++) {
                buffer.produce(); // Produce an item
                Thread.sleep(1000); // Simulate some delay
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

class Consumer extends Thread {
    SharedBuffer buffer;

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

    @Override
    public void run() {
        try {
            for (int i = 0; i < 10; i++) {
                buffer.consume(); // Consume an item
                Thread.sleep(1000); // Simulate some delay
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

public class Practical37 {
    public static void main(String[] args) throws InterruptedException {
```

```
        SharedBuffer buffer = new SharedBuffer(); // Shared buffer

        // Create producer and consumer threads by extending Thread
        Producer producerThread = new Producer(buffer);
        Consumer consumerThread = new Consumer(buffer);

        // Start the threads
        producerThread.start();
        consumerThread.start();

        // Wait for both threads to complete
        producerThread.join();
        consumerThread.join();

        System.out.println("Producer and Consumer have finished execution.");
        System.out.println(" ");
        System.out.println("23DCS085 Meet K. Patel");
    }
}
```

OUTPUT:

```
Produced: 58
Consumed: 58
Produced: 39
Consumed: 39
Produced: 10
Consumed: 10
Produced: 35
Consumed: 35
Produced: 93
Consumed: 93
Produced: 3
Consumed: 3
Produced: 60
Consumed: 60
Produced: 8
Consumed: 8
Produced: 88
Consumed: 88
Producer and Consumer have finished execution.

23DCS085 Meet K. Patel
```

CONCLUSION: From this practical we learned about Thread Synchronization and also we make use of that concept in above code we use that concept to apply Producer and Consumer Logic.

Part – 8

No.	Aim of the Practical

38.	Design a Custom Stack using ArrayList class, which 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
	
	<u>PROGRAM CODE :</u>
	
	import java.util.*;
	import java.io.*;
	
	class StackClass
	{
	public int MAX_SIZE;
	public int top;
	public ArrayList<Integer> Stack;
	
	public StackClass(int MAX_SIZE)
	{
	this.MAX_SIZE = MAX_SIZE;
	this.Stack = new ArrayList<>(MAX_SIZE);
	this.top = -1;
	}

```
public int getSize()
{
    int countStack = Stack.size();

    return countStack;
}

public boolean isEmpty()
{
    if(Stack.isEmpty())
    {
        return true;
    }
    else
    {
        return false;
    }
}

public int peek()
{
    if(top == -1)
    {
        return 0;
    }
    else
    {
        return Stack.get(top);
    }
}

public void push(int Element)
{
    if(top == (MAX_SIZE - 1))
```

```
        {
            System.out.println("\nStack is Full");
        }
        else
        {
            top = top + 1;
            Stack.add(top,Element);

            System.out.println("\nElement " +
Element + " is Successfully Entered in Stack");
        }
    }

    public void pop()
    {
        int popElement = 0;

        if(top == -1)
        {
            System.out.println("\nStack is Empty");
        }
        else
        {
            popElement = Stack.get(top);
            Stack.remove(top);
            top = top - 1;

            System.out.println("\nElement " +
popElement + " is Successfully Removed From
the Stack");
        }
    }

    public void Display_Stack()
    {
```

```
        System.out.println("\n---: Printing Stack :--
-");

        System.out.println(" ");

        System.out.println(Stack);
    }

}

class Practical38
{
    public static void main(String[] args)
    {

        Scanner sc = new Scanner(System.in);

        int Stack_Size;

        System.out.print("\nEnter Size of Stack :
");
        Stack_Size = sc.nextInt();

        StackClass S = new
StackClass(Stack_Size);

        int choice;

        System.out.println("\nPress \"1\" to
Perform isEmpty Operation");
        System.out.println("Press \"2\" to Perform
getSize Operation");
        System.out.println("Press \"3\" to Perform
push Operation");
        System.out.println("Press \"4\" to Perform
```

```
pop Operation");
        System.out.println("Press \"5\" to Perform
peek Operation");
        System.out.println("Press \"6\" to Display
Stack");
        System.out.println("Press \"7\" to Exit");

        for(int i=0;i<1;i++)
        {

        System.out.print("\nEnter Your Choice
Here : ");
        choice = sc.nextInt();

        char checkChoice;

        switch (choice)
        {
            case 1:
            {
                boolean result1 = S.isEmpty();

                System.out.println("Result of isEmpty
: " + result1);

                System.out.print("\nYou want to
Continue(Y/N) : ");
                checkChoice = sc.next().charAt(0);

                if(checkChoice == 'Y' || checkChoice
== 'y')
                {
                    i--;
                }
                break;
```

```
            }
            case 2:
            {
                System.out.println("Size of Stack : " +
S.getSize());

                System.out.print("\nYou want to
Continue(Y/N) : ");
                checkChoice = sc.next().charAt(0);

                if(checkChoice == 'Y' || checkChoice
== 'y')
                {
                    i--;
                }
                break;
            }
            case 3:
            {
                int pushElement;

                System.out.print("\nEnter Element
that you want to Push in Stack : ");
                pushElement = sc.nextInt();

                S.push(pushElement);

                System.out.print("\nYou want to
Continue(Y/N) : ");
                checkChoice = sc.next().charAt(0);

                if(checkChoice == 'Y' || checkChoice
== 'y')
                {
                    i--;
```

```
            }
            break;
        }
        case 4:
        {
            S.pop();

            System.out.print("\nYou want to
Continue(Y/N) : ");
            checkChoice = sc.next().charAt(0);

            if(checkChoice == 'Y' || checkChoice
== 'y')
            {
                i--;
            }
            break;
        }
        case 5:
        {
           if(S.peek() != 0)
           {
           System.out.println("\nPeek Element of
Stack is : " + S.peek());
           }
           else
           {
            System.out.println("\nStack is
Empty");
           }

           System.out.print("\nYou want to
Continue(Y/N) : ");
            checkChoice = sc.next().charAt(0);
```

```
            if(checkChoice == 'Y' || checkChoice
== 'y')
            {
               i--;
            }
            break;
         }
         case 6:
         {
            if(S.isEmpty())
            {
               System.out.println("Stack is
Empty");
            }
            else
            {
               S.Display_Stack();
            }

            System.out.print("\nYou want to
Continue(Y/N) : ");
            checkChoice = sc.next().charAt(0);

            if(checkChoice == 'Y' || checkChoice
== 'y')
            {
               i--;
            }
            break;
         }
         case 7:
         {
            break;
         }
```

```
            default:
            {
                System.out.println("Invalid Choice");

                i--;

                break;
            }
        }

        }
    }
}
```

OUTPUT:

```
Enter Size of Stack : 3

Press "1" to Perform isEmpty Operation
Press "2" to Perform getSize Operation
Press "3" to Perform push Operation
Press "4" to Perform pop Operation
Press "5" to Perform peek Operation
Press "6" to Display Stack
Press "7" to Exit

Enter Your Choice Here : 3

Enter Element that you want to Push in Stack : 5

Element 5 is Successfully Entered in Stack

You want to Continue(Y/N) : Y

Enter Your Choice Here : 6

---: Printing Stack :---

[5]

You want to Continue(Y/N) : Y
```

.

```
Enter Your Choice Here : 6

---: Printing Stack :---

[5]

You want to Continue(Y/N) : Y

Enter Your Choice Here : 3

Enter Element that you want to Push in Stack : 6

Element 6 is Successfully Entered in Stack

You want to Continue(Y/N) : Y

Enter Your Choice Here : 6

---: Printing Stack :---

[5, 6]

You want to Continue(Y/N) : Y

Enter Your Choice Here : 4

Element 6 is Successfully Removed From the Stack

You want to Continue(Y/N) : Y

Enter Your Choice Here : 6

---: Printing Stack :---

[5]
```

```
You want to Continue(Y/N) : N
23DCS085 Meet K. Patel
```

Conclusion :- From this practical we learned about collection frameword – ArrayList and also implemented stack with the use of ArrayList.

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

PROGRAM CODE :

```
import java.util.Arrays;

class Product implements Comparable<Product> {

    private String name;

    private int price;

 public Product(String name, int price) {

        this.name = name;

        this.price = price;

    }

@Override
```

```
    public int compareTo(Product other) {
        return this.price - other.price;
    }
  @Override
    public String toString() {
        return name + ": $" + price;
    }
}
public class prac39 {
    public static <T extends Comparable<T>> void sortArray(T[] array) {
        Arrays.sort(array);
    }
  public static void main(String[] args) {
        Integer[] numbers = { 8, 3, 19, 13, 7 ,2};
        System.out.println("Before sorting (Integers): " + Arrays.toString(numbers));
        sortArray(numbers);
        System.out.println("After sorting (Integers): " + Arrays.toString(numbers));
        String[] names = { "Cristiano", "Alice", "Marco", "Messi" };
        System.out.println("\nBefore sorting (Strings): " + Arrays.toString(names));
        sortArray(names);
        System.out.println("After sorting (Strings): " + Arrays.toString(names));
```

```
Product[] products = {
        new Product("Laptop", 700),
        new Product("Phone", 550),
        new Product("Tablet", 540),
        new Product("Smartwatch", 200)
};
System.out.println("\nBefore sorting (Products by price): ");
for (Product p : products) {
    System.out.println(p);
}
sortArray(products);
System.out.println("\nAfter sorting (Products by price): ");
for (Product p : products) {
    System.out.println(p);
}
```

OUTPUT:

```
Before sorting (Integers): [8, 3, 19, 13, 7, 2]
After sorting (Integers): [2, 3, 7, 8, 13, 19]

Before sorting (Strings): [Cristiano, Alice, Marco, Messi]
After sorting (Strings): [Alice, Cristiano, Marco, Messi]

Before sorting (Products by price):
Laptop: $700
Phone: $550
Tablet: $540
Smartwatch: $200

After sorting (Products by price):
Smartwatch: $200
Tablet: $540
Phone: $550
Laptop: $700
```

CONCLUSION: This program demonstrates generic sorting by using Java's Comparable interface. It sorts arrays of integers, strings, and custom Product objects based on price in ascending order. By leveraging the Arrays.sort() method, it efficiently arranges elements and displays the sorted results. It provides a versatile approach to sorting different types of objects.

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

PROGRAM CODE :

```
import java.util.*;

public class prac40 {

public static void main(String[] args) {

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

```
Scanner scanner = new Scanner(System.in);
System.out.println("Enter a text:");
String text = scanner.nextLine();
String[] words = text.toLowerCase().split("\\W+");
for (String word : words) {
if (!word.isEmpty()) {
wordMap.put(word, wordMap.getOrDefault(word, 0) + 1);
} }
System.out.println("\nWord Occurrences (in alphabetical order):");
Set<Map.Entry<String, Integer>> entrySet = wordMap.entrySet();
for (Map.Entry<String, Integer> entry : entrySet) {
System.out.println(entry.getKey() + ": " + entry.getValue());
} } }
```

OUTPUT:

```
Enter a text:
hello my name is marco

Word Occurrences (in alphabetical order):
hello: 1
is: 1
marco: 1
my: 1
name: 1
```

CONCLUSION: This program takes a text input from the user, counts the occurrences of each word, and displays the results in alphabetical order. It uses a TreeMap to store words, ensuring automatic sorting by key. The program efficiently processes text by splitting it into words and counting their frequency.

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

PROGRAM CODE :

```
import java.io.*;

import java.util.*;

public class P41 {

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

static {

String[] keywordArray = {

"abstract", "assert", "boolean", "break", "byte", "case", "catch", "char", "class",

"const", "continue", "default", "do", "double", "else", "enum", "extends", "final",

"finally", "float", "for", "goto", "if", "implements", "import", "instanceof", "int",

"interface", "long", "native", "new", "package", "private", "protected", "public",

"return", "short", "static", "strictfp", "super", "switch",  "this",

"throw", "throws", "transient", "try", "void", "volatile", "while"

};
```

```
for (String keyword : keywordArray) {
keywords.add(keyword);
} }
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.print("Enter the path of the Java source file: ");
String filePath = scanner.nextLine();
try {
File file = new File(filePath);
Scanner fileScanner = new Scanner(file);
int keywordCount = 0;
while (fileScanner.hasNext()) {
String word = fileScanner.next();
if (keywords.contains(word)) {
keywordCount++;
} }
System.out.println("Number of Java keywords in the file: " + keywordCount);
fileScanner.close();
} catch (FileNotFoundException e) {
System.out.println("File not found: " + filePath);
}
} }
```

OUTPUT:

```
Enter the path of the Java source file: prac41.java
Number of Java keywords in the file: 21
```

CONCLUSION:

This program reads a Java source file and counts the number of Java keywords it contains. By utilizing a predefined set of keywords, it efficiently scans through the file and outputs the total count. The program also handles file not found errors gracefully.

Part – 6

No.	Aim of the Practical
27.	Write a program that will count the number of lines in each file that is specified on the command line. Assume that the files are text files. Note that multiple files can be specified, as in "java Line Counts file1.txt file2.txt file3.txt". Write each file name, along with the number of lines in that file, to standard output. If an error occurs while trying to read from one of the files, you should print an error message for that file, but you should still process all the remaining files.

PROGRAM CODE :

```
import java.io.*;

public class prac27 {

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

        if (args.length == 0) {

            System.out.println("No file Found!");

        } else {

            for (int i = 0; i < args.length; i++) {

                try {

                    BufferedReader f = new
BufferedReader(new FileReader(args[i]));

                    String j;
```

```
                int count = 0;

                while ((j = f.readLine()) != null) {

                    count++;

            }

            System.out.println("File name is : " +
args[i] + " and Number of lines are : " + count);

            } catch (Exception e) {

                System.out.println(e);

            }

        }

    }

  }

  }
```

OUTPUT:

```
PS C:\Users\markp\OneDrive\Documents\Desktop\practials\pracroll83\java_college_pracs> javac prac27.java
PS C:\Users\markp\OneDrive\Documents\Desktop\practials\pracroll83\java_college_pracs> java prac27
No file Found!
```

```
PS C:\Users\markp\OneDrive\Documents\Desktop\practials\pracroll83\java_college_pracs> java prac27 pqr.txt xyz.txt
File name is : pqr.txt and Number of lines are : 4
File name is : xyz.txt and Number of lines are : 2
```

Conclusion :- This Java program reads several files named by the command line arguments and counts the number of lines in each. If no files are provided as command-line arguments, it will print out the appropriate message. Exception handling ensures graceful error management during file reading, thus a stable program.

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

PROGRAM CODE :

```
import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class prac28{

public static void main(String[] args) {

if (args.length < 2) {

System.out.println("Usage: java prac28 <character> <filename>");

return; }

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

String fileName = args[1];
```

```
int count = 0;

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

int ch;

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

if (ch == targetChar) {

count++;

} }

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

} catch (IOException e) {

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

}

}}
```

OUTPUT:

```
PS C:\Users\markp\OneDrive\Documents\Desktop\practials\pracroll83\java_college_pracs> javac prac28.java
PS C:\Users\markp\OneDrive\Documents\Desktop\practials\pracroll83\java_college_pracs> java prac28 d  pqr.txt
The character 'd' appears 4 times in pqr.txt
```

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

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

PROGRAM CODE :

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
public class prac29 {
public static void main(String[] args) {
if (args.length < 2) {
System.out.println("Usage: java prac29 <word> <filename>");
return;
}
String searchWord = args[0];
String fileName = args[1];
```

```
Integer count = 0;
try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
String line;
while ((line = reader.readLine()) != null) {
String[] words = line.split("\\W+");
for (String word : words) {
if (word.equalsIgnoreCase(searchWord)) {
count++;
} } }
System.out.println("The word '" + searchWord + "' appears " + count + " times in " + fileName);
} catch (IOException e) {
System.out.println("Error reading " + fileName + ": " + e.getMessage());
}
} }
```

OUTPUT:

```
PS C:\Users\markp\OneDrive\Documents\Desktop\practials\pracroll83\java_college_pracs> javac prac29.java
PS C:\Users\markp\OneDrive\Documents\Desktop\practials\pracroll83\java_college_pracs> java prac29 am xyz.txt
The word 'am' appears 2 times in xyz.txt
```

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

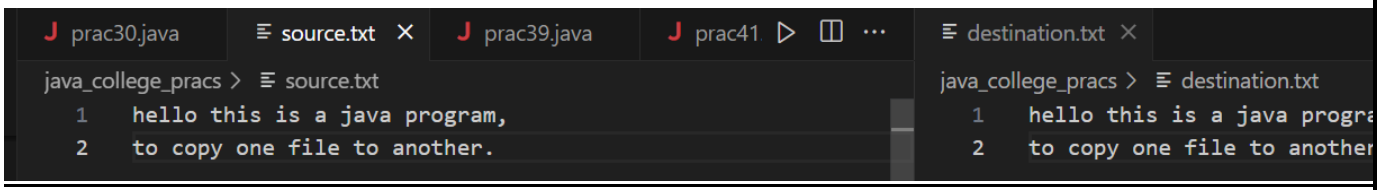
30. Write a program to copy data from one file to another file. If the destination file does not exist, it is created automatically.

PROGRAM CODE :

```
Import java.util.*;
public class prac30 {

    public static void main(String[] args) {

        // Specify the source and destination file paths

        String sourceFilePath = "source.txt";

        String destinationFilePath = "destination.txt";



        // Use try-with-resources to ensure resources are closed automatically

        try (

            FileInputStream fis = new FileInputStream(sourceFilePath);

            FileOutputStream fos = new FileOutputStream(destinationFilePath)

        ) {
```

```
            int byteContent;
            // Read from source and write to destination file byte by byte
            while ((byteContent = fis.read()) != -1) {
                fos.write(byteContent);
            }
            System.out.println("File copied successfully.");
        } catch (FileNotFoundException e) {
            System.out.println("File not found: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("Error occurred while copying the file: " + e.getMessage());
        }
    }
}
```

OUTPUT:




```
PS C:\Users\markp\OneDrive\Documents\Desktop\practials\pracroll83\java_college_pracs> javac p
PS C:\Users\markp\OneDrive\Documents\Desktop\practials\pracroll83\java_college_pracs> java pr
File copied successfully.
```

CONCLUSION:

This program efficiently copies data from a source file to a destination file in Java, creating the destination file automatically if it doesn't exist. It uses file input and output streams to handle byte-by-byte reading and writing, ensuring proper resource management with try-with-resources.

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

PROGRAM CODE:

```
import java.io.*;

public class prac31 {

   public static void main(String[] args) {

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

      String fileName = "output.txt";

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

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

         String input;

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

            fileWriter.write(input);

            fileWriter.newLine();
```

```
            }
        System.out.println("Data written to " + fileName);
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

OUTPUT:

```
PS C:\Users\markp\OneDrive\Documents\Desktop\practials\pracroll83\java_college_pracs> javac prac31.java
PS C:\Users\markp\OneDrive\Documents\Desktop\practials\pracroll83\java_college_pracs> java prac31
Enter text (type 'exit' to finish):
hello my name is marco
exit
Data written to output.txt
```

CONCLUSION:

The program reads user input and writes it to a file called "output.txt." It uses BufferedReader and BufferedWriter for efficient input and output handling. The process stops when the user types "exit." This demonstrates simple file handling in Java