DEPSTAR

Devang Patel Institute of Advance Technology and Research

(A Constitute Institute of CHARUSAT)

Certificate

This is to certify that

Mr./~~Mrs~~. Malay Manishbhai Patel

of 3CSE -II *Class,*

ID. No. 23DCS082 *has satisfactorily completed*

his/ her term work in CSE-201 Java Programming *for*

the ending in November 2024 /2025

[illegible] 17/10/24

Date : 17/10/24

Sign. of Faculty

[illegible]

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

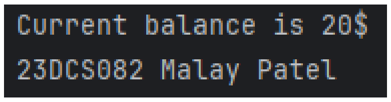
No.	Aim of the Practical
1.	Demonstration of installation steps of Java, Introduction to Object Oriented Concepts, comparison of Java with other object-oriented programming languages. Introduction to JDK, JRE, JVM, Javadoc, command line argument. Introduction to Eclipse or NetBeans IDE, or BlueJ and Console Programming. **CONCLUSION:** In this program we learn about the basics of java and know more about of JDK, JRE, JVM, Javadoc and etc. Where JDK “Java development kit” which contains tools and JRE to run the program and JRE “Java Runtime Environment” which is used to run the java applications. JVM “Java Virtual Machine” which is used as interpreter to the program.

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

PROGRAM CODE :

```
public class pr2 {
    public static void main(String args[])
    {
int balance = 20;
        System.out.printf("Current balance is " +
balance + "$");
        System.out.println(" ");
        System.out.println("23DCS082 Malay
Patel");
    }
}
```

OUTPUT:



```
Current balance is 20$
23DCS082 Malay Patel
```

CONCLUSION:

From this practical we learn about the variables and how we can print it.

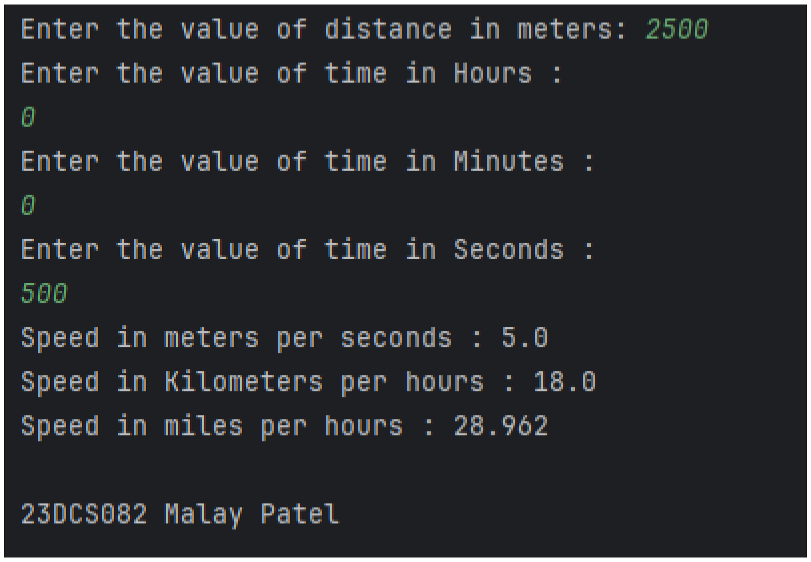
3. Write a program to take the user for a distance (in meters) and the time taken (as three numbers: hours, minutes, seconds), and display the speed, in meters per second, kilometers per hour and miles per hour (hint:1 mile = 1609 meters).

PROGRAM CODE :

```
import java.util.Scanner;

public class pr3 {
    public static void main(String args[])
    {
        Scanner obj = new Scanner(System.in);

        double distance,ms,kh,mh;
        int hours, minute, seconds;

        System.out.printf("Enter the value of
distance in meters: ");
        distance = obj.nextDouble();

        System.out.println("Enter the value of time
in Hours :");
        hours = obj.nextInt();

        System.out.println("Enter the value of time
in Minutes :");
        minute = obj.nextInt();

        System.out.println("Enter the value of time
in Seconds :");
        seconds = obj.nextInt();

        ms = distance/((hours * 3600) + (minute *
60) + (seconds));
        System.out.printf("Speed in meters per
seconds : " + ms);
```

```
        kh = (ms * 18)/ 5;
        System.out.printf("Speed in Kilometers per 
hours : " + kh);

        mh = kh*1.609;
        System.out.printf("Speed in miles per 
hours : " + mh);
        System.out.println(" ");
        System.out.println("23DCS082 Malay 
Patel");
    }
}
```

OUTPUT:



```
Enter the value of distance in meters: 2500
Enter the value of time in Hours :
0
Enter the value of time in Minutes :
0
Enter the value of time in Seconds :
500
Speed in meters per seconds : 5.0
Speed in Kilometers per hours : 18.0
Speed in miles per hours : 28.962

23DCS082 Malay Patel
```

CONCLUSION:

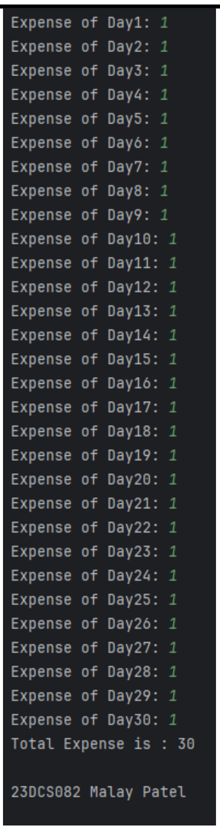
This Java program calculates and displays the speed of an object in meters per second (m/s), kilometers per hour (km/h), and miles per hour (mph) based on user input for distance in meters and time in hours, minutes, and seconds.

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.Scanner;

public class pr4 {]
    public static void main(String args[])
    {
        Scanner obj = new Scanner(System.in);
        int[] arr = new int[30];
        int total=0;

        for(int i=0 ; i<30 ; i++)
        {
            System.out.printf("Expense of Day" + (i+1) + ": ");
            arr[i]= obj.nextInt();
            total = total + arr[i];
        }
        System.out.println("Total Expense is : " + total);
        System.out.println(" ");
        System.out.println("23DCS082 Malay Patel");
    }
}
```

OUTPUT:



```
Expense of Day1: 1
Expense of Day2: 1
Expense of Day3: 1
Expense of Day4: 1
Expense of Day5: 1
Expense of Day6: 1
Expense of Day7: 1
Expense of Day8: 1
Expense of Day9: 1
Expense of Day10: 1
Expense of Day11: 1
Expense of Day12: 1
Expense of Day13: 1
Expense of Day14: 1
Expense of Day15: 1
Expense of Day16: 1
Expense of Day17: 1
Expense of Day18: 1
Expense of Day19: 1
Expense of Day20: 1
Expense of Day21: 1
Expense of Day22: 1
Expense of Day23: 1
Expense of Day24: 1
Expense of Day25: 1
Expense of Day26: 1
Expense of Day27: 1
Expense of Day28: 1
Expense of Day29: 1
Expense of Day30: 1
Total Expense is : 30

23DCS082 Malay Patel
```

CONCLUSION:

This Java program calculates the total monthly expenses by collecting daily expenses for 30 days from the user and summing them up.

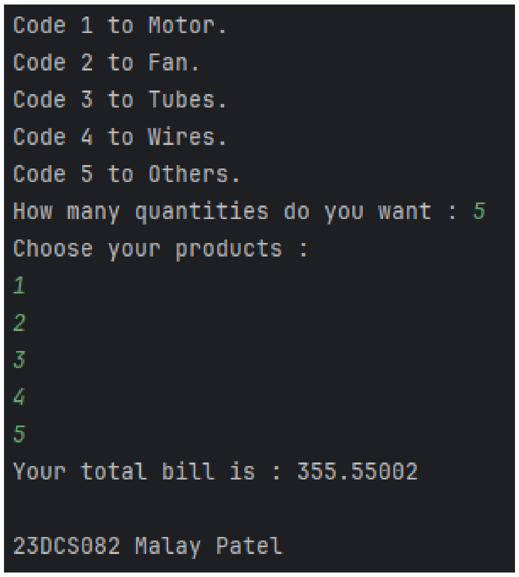
5. An electric appliance shop assigns code 1 to motor,2 to fan,3 to tube and 4 for wires. All other items have code 5 or more. While selling the goods, a sales tax of 8% to motor,12% to fan,5% to tube light,7.5% to wires and 3% for all other items is charged. A list containing the product code and price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

PROGRAM CODE :

```
import java.util.Scanner;

public class pr5 {
    public static void main(String args[])
    {
        Scanner obj = new Scanner(System.in);
        int[] code = {1,2,3,4,5};
        float[] price = {100,80,60,50,40};
        int n,p;
        float
motor=0,fan=0,tubes=0,wires=0,others=0,total;

        System.out.println("Code 1 to Motor."); //8
        System.out.println("Code 2 to Fan.");  //12
        System.out.println("Code 3 to Tubes.");
//5
        System.out.println("Code 4 to Wires.");
//7.5
        System.out.println("Code 5 to Others.");
//3

        System.out.printf("How many quantities
do you want : ");
        n= obj.nextInt();

        System.out.println("Choose your products
: ");
        for (int i=0 ; i<n ; i++)
        {
            p= obj.nextInt();
            switch(p)
            {
                case 1:
                {
                      motor = motor + price[0];
                      break;
                }
                case 2:
                {
                    fan = fan + price[1];
                    break;
                }
                case 3:
                {
                    tubes = tubes + price[2];
                    break;
                }
                case 4:
                {
                    wires = wires + price[3];
                    break;
                }
                case 5:
                {
                    others = others + price[4];
                    break;
                }
                default: {
                    System.out.println("Invalid
choice");
                    break;
                }
            }
        }
        motor = (motor * 0.08f) + motor;
        fan = (fan * 0.12f) + fan;
        tubes = (tubes *0.05f) + tubes;
        wires = (wires * 0.075f) + wires;
        others = (others * 0.03f) + others;
        total = motor + fan + wires + tubes +
others;

        System.out.println("Your total bill is : " +
total);
        System.out.println(" ");
        System.out.println("23DCS082 Malay
Patel");
    }
```

```
}
```

OUTPUT:



```
Code 1 to Motor.
Code 2 to Fan.
Code 3 to Tubes.
Code 4 to Wires.
Code 5 to Others.
How many quantities do you want : 5
Choose your products :
1
2
3
4
5
Your total bill is : 355.55002

23DCS082 Malay Patel
```

CONCLUSION:

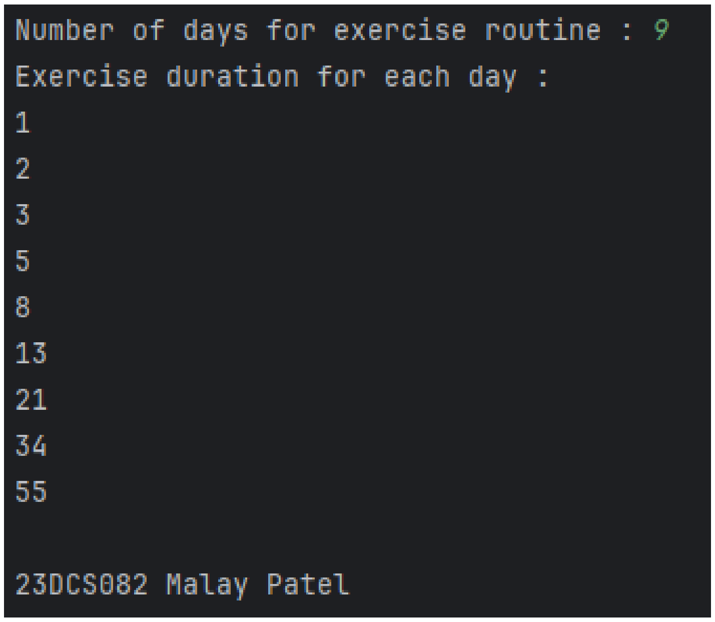
This Java program calculates the total bill for selected quantities of products, each with a specific code and price, by including applicable tax rates for each product category.

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

PROGRAM CODE :

```
import java.util.Scanner;
public class pr6 {
    public static void main(String args[])
    {
        Scanner obj = new Scanner(System.in);
        int n,a,b,c;
        System.out.print("Number of days for
exercise routine : ");
        n= obj.nextInt();
        a=0;
        b=1;
        System.out.println("Exercise duration for
each day : ");
        for (int i=0 ;i<n ; i++)
        {
            c=a+b;
            System.out.println(c);
            a=b;
            b=c; }
        System.out.println(" ");
        System.out.println("23DCS082 Malay
Patel");
    }
}
```

OUTPUT:



```
Number of days for exercise routine : 9
Exercise duration for each day :
1
2
3
5
8
13
21
34
55

23DCS082 Malay 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**

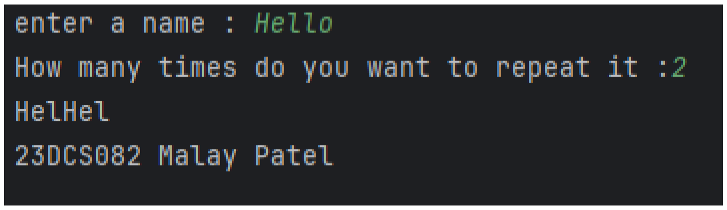
7. Given a string and a non-negative int n, we'll say that the front of the string is the first 3 chars, or whatever is there if the string is less than length 3. Return n copies of the front; front_times('Chocolate', 2) → 'ChoCho' front_times('Chocolate', 3) → 'ChoChoCho' front_times('Abc', 3) → 'AbcAbcAbc'.

PROGRAM CODE :

```
import java.util.*;
public class pr7 {
    public static void main(String args[]) {
        Scanner obj = new Scanner(System.in);
        System.out.printf("enter a name : ");
        String name = obj.nextLine();
        String ch;
        int t = name.length();

        if(t > 3)
        {
            ch = name.substring(0,3);
          //  System.out.printf(ch);
        }
        else
        {
            ch = name.substring(0,t);
           //System.out.printf(ch);
        }
        System.out.printf("How many times do
you want to repeat it :");
        int n = obj.nextInt();

        for(int i=0 ; i<n ; i++)
        {
           System.out.print(ch);
        }
       System.out.println("23DCS082 Malay
Patel");
    }
```

}

OUTPUT:



```
enter a name : Hello
How many times do you want to repeat it :2
HelHel
23DCS082 Malay Patel
```

CONCLUSION:

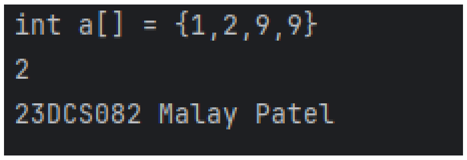
This Java program takes a user's name and repeats the first three characters (or the entire name if it's shorter than three characters) a specified number of times. It demonstrates string manipulation and user input handling in Java.

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

PROGRAM CODE :

```
import java.util.Scanner;


public class pr8 {
    public static void main(String args[])
    {
        int[] a = {1,2,9,9};
        int t = a.length;
        int count=0;
System.out.println("int a[] = {1,2,9,9}");
        for(int  i=0 ; i<t ; i++)
        {
            if(a[i]==9)
            {
                count++;
            }
        }
        System.out.print(count);
System.out.println("");
System.out.println("23DCS082 Malay Patel");
    }
}
```

OUTPUT:



```
int a[] = {1,2,9,9}
2
23DCS082 Malay Patel
```

CONCLUSION:

This Java program counts how many times the number appears in an array. It then prints the count.

9. Given a string, return a string where for every char in theoriginal, there are two chars.

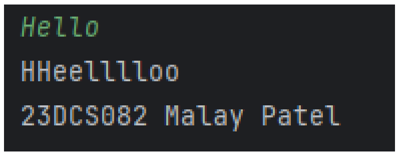
double_char('The') → 'TThhee' double_char('AAbb') → 'AAAAbbbb'

double_char('Hi-There') → 'HHii--TThheerree'.

PROGRAM CODE :

```
import java.util.Scanner;

public class pr9 {
    public static void main(String args[])
    {
        Scanner obj = new Scanner(System.in);
        String a ;
        a = obj.nextLine();
        int l = a.length();
        char ch;
        for(int i=0 ; i<l ; i++)
        {
            ch = a.charAt(i);
            for(int j=0 ; j<2 ; j++)
            {
                System.out.print(ch);
            }
        }
        System.out.println("");
        System.out.println("23DCS082 Malay
Patel");
    }
}
```

OUTPUT:



```
Hello
HHeellllloo
23DCS082 Malay 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.

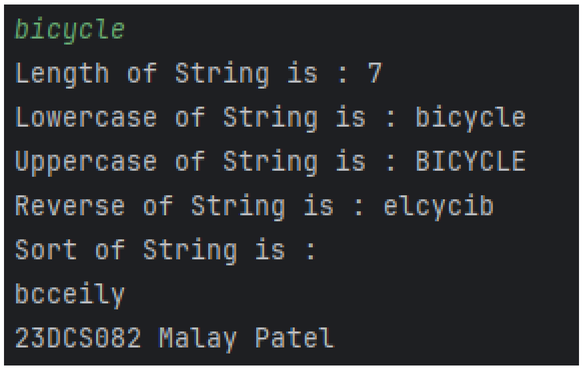
10. Perform following functionalities of the string:

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

PROGRAM CODE :

```
import java.util.Arrays;
import java.util.Scanner;
public class pr10 {
    public static void main(String args[])
    {
        Scanner obj = new Scanner(System.in);
        String a;
        int i;
        a = obj.nextLine();

        int l = a.length();
        System.out.println("Length of String is : " + l);

        String lower = a.toLowerCase();
        System.out.println("Lowercase of String is : " + lower);

        String upper = a.toUpperCase();
        System.out.println("Uppercase of String is : " + upper);

        System.out.print("Reverse of String is : ");
        for(i=l-1 ; i>=0 ; i--)
        {
           char ch = a.charAt(i);
           System.out.print(ch);
        }
        System.out.println(" ");
        char[] arr = a.toCharArray();
        Arrays.sort(arr);
        System.out.println("Sort of String is : ");
        System.out.print(String.valueOf(arr));
        System.out.println("");
        System.out.println("23DCS082 Malay Patel");
    }
}
```

OUTPUT:



```
bicycle
Length of String is : 7
Lowercase of String is : bicycle
Uppercase of String is : BICYCLE
Reverse of String is : elcycib
Sort of String is :
bcceily
23DCS082 Malay Patel
```

CONCLUSION:

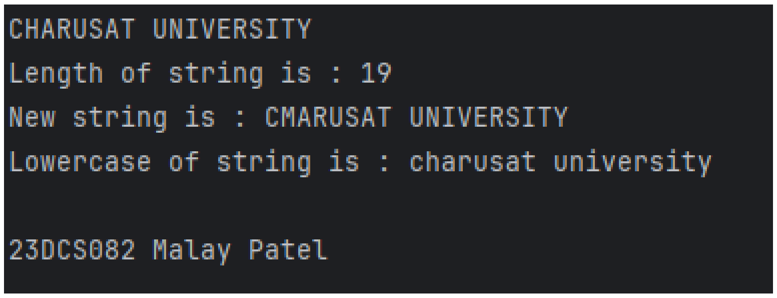
In this java program we learn and understand the String methods for counting the length of

	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
	
	<u>PROGRAM CODE :</u>
	
	public class pr11 {
	    public static void main(String args[])
	    {
	        String a = "CHARUSAT UNIVERSITY";
	
	        int l=a.length();
	        System.*out*.println("Length of string is : "
	+ l);
	
	        String n = a.replace("H", "M");
	        System.*out*.println("New string is : " + n);
	
	        System.*out*.println("Lowercase of string is
	: " + a.toLowerCase());
	
	        System.*out*.println("");

```
        System.out.println("23DCS082 Malay
Patel");
    }
}
```

OUTPUT:



```
CHARUSAT UNIVERSITY
Length of string is : 19
New string is : CMARUSAT UNIVERSITY
Lowercase of string is : charusat university

23DCS082 Malay 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.

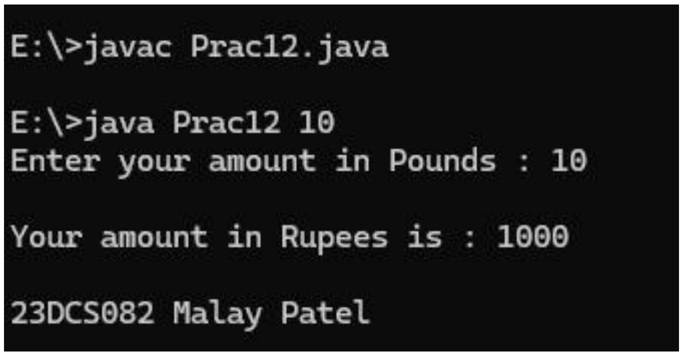
Part -3 Object Oriented Programming: Classes, Methods, Constructors

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.

PROGRAM CODE :

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

        System.out.print("Enter amount 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("Amount in rupees : " + rupees);

        System.out.println(" ");
        System.out.println(" ");
        System.out.println("23DCS082 Malay Patel");

    }
}
```

OUTPUT:



```
E:\>javac Prac12.java

E:\>java Prac12 10
Enter your amount in Pounds : 10

Your amount in Rupees is : 1000

23DCS082 Malay Patel
```

CONCLUSION:

In this java program we convert the pound currency to rupee using command-line argument and interact from the user.

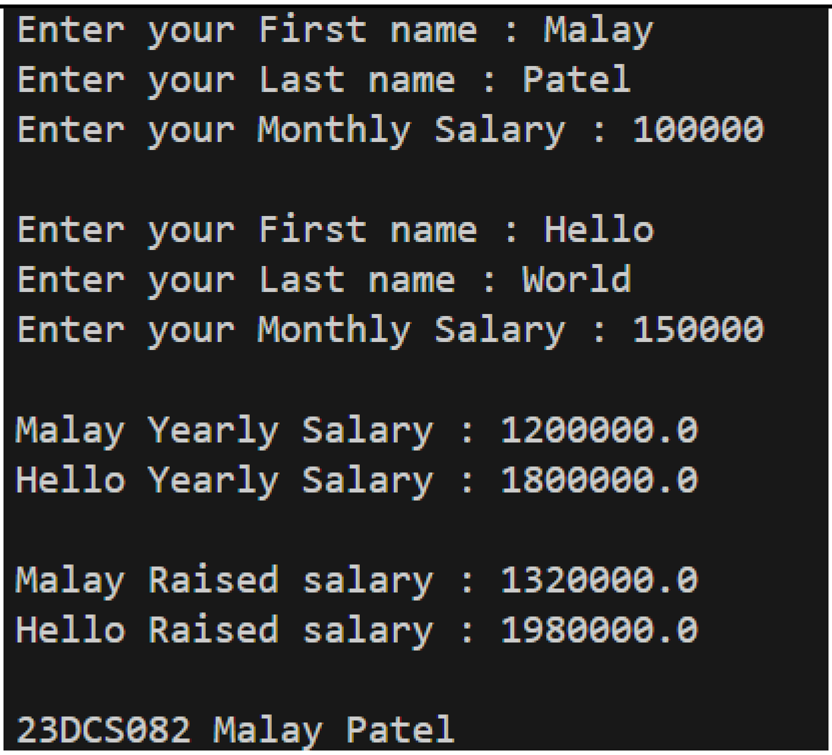
13. Create a class called Employee that includes three pieces of information as instance variables—a first name (type String), a last name (type String) and a monthly salary (double). Your class should have a constructor that initializes the three instance variables. Provide a set and a get method for 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.Scanner;

public class pr13 {

    public class Employee {
        String fname, lname;
        double salary;

        public Employee()
        {
            fname="NULL";
            lname="NULL";
            salary =0;
        }
        public void get()
        {
            Scanner s = new Scanner(System.in);

            System.out.print("Enter your First name
: ");
            fname=s.nextLine();
            System.out.print("Enter your Last name
: ");
            lname=s.nextLine();
            System.out.print("Enter your Monthly
Salary : ");
            salary=s.nextInt();

            if(salary<0)
            {
                salary = 0.0;
            }
            System.out.println("");
        }
        public void put()
        {
           System.out.println(fname + " Yearly
Salary : " + (salary*12));
        }
        public void display()
        {
           System.out.println(fname + " Raised
salary : " + (((salary*0.1)+salary)*12));
        }
    }
    public void main(String args[])
    {
    Employee o1 =  new Employee();
    Employee o2 =  new Employee();

    o1.get();
    o2.get();

    o1.put();
    o2.put();

    System.out.println("");

    o1.display();
    o2.display();

    System.out.println("");
    System.out.println("23DCS082 Malay
Patel");
    }
}
```

OUTPUT:



```
Enter your First name : Malay
Enter your Last name : Patel
Enter your Monthly Salary : 100000

Enter your First name : Hello
Enter your Last name : World
Enter your Monthly Salary : 150000

Malay Yearly Salary : 1200000.0
Hello Yearly Salary : 1800000.0

Malay Raised salary : 1320000.0
Hello Raised salary : 1980000.0

23DCS082 Malay Patel
```

CONCLUSION:

In this java program we learnt about constructor, in how we can we initialize the instance variable with it and using get & set method to take & print data from the user.

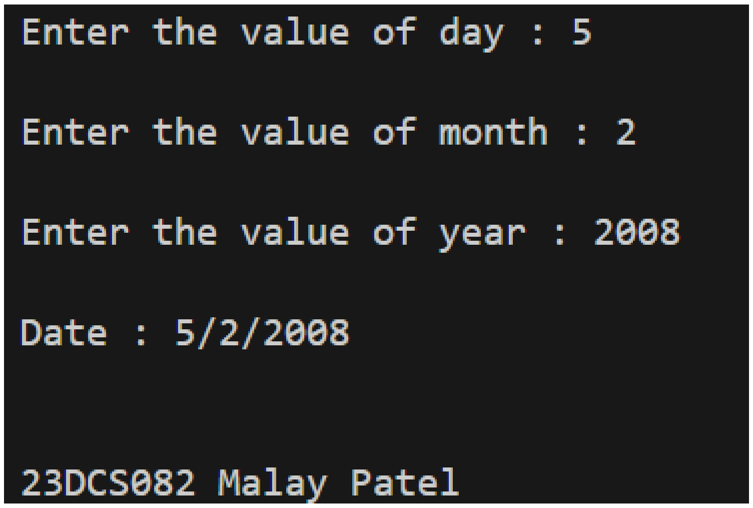
14. Create a class called Date that includes three pieces of information as instance variables—a month (type int), a day (type int) and a year (type int). Your class should have a constructor that initializes the three instance variables and assumes that the values provided are correct. Provide a set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

PROGRAM CODE:

```
import java.util.Scanner;

public class pr14 {
   public static void main(String args[])
   {
      Date o1 = new Date();

      o1.get();
      o1.display();
```

```
        System.out.println("");
        System.out.println("23DCS082 Malay Patel");
    }
}

class Date
{
    int day,month,year;

    Date()
    {
        day = 0;
        month = 0;
        year = 0;
    }
    public void get()
    {
        Scanner obj =  new Scanner(System.in);
        for(int i=0 ; i<1 ; i++)
        {
        System.out.print("Enter the value of day : ");
        day = obj.nextInt();
            if(day>=31)
            {
                System.out.println("Invalid input");
                i--;
            }
        }
        System.out.println("");
        for(int i=0 ; i<1 ; i++)
        {
        System.out.print("Enter the value of month : ");
        month = obj.nextInt();
        if(month>=12)
        {
            System.out.println("Invalid input");
            i--;
        }
        }
        System.out.println("");
        System.out.print("Enter the value of year : ");
        year = obj.nextInt();
    }
    public void display()
    {
        System.out.println("");
        System.out.println("Date : " + day + "/" + month + "/" + year);
        System.out.println("");
    }
}
```

OUTPUT:



```
Enter the value of day : 5

Enter the value of month : 2

Enter the value of year : 2008

Date : 5/2/2008


23DCS082 Malay Patel
```

CONCLUSION:

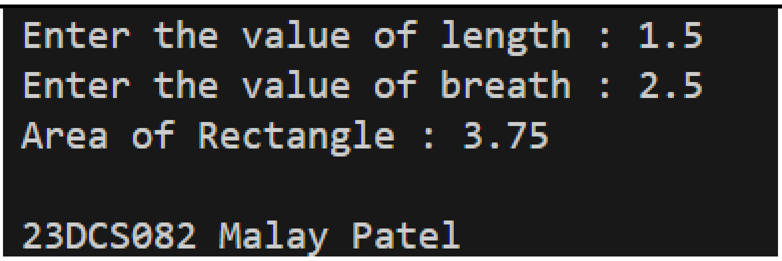
In this java program we learnt about how to initialize a instance variable using default constructor and take input from get method for date and print the date format using display method.

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.Scanner;

public class pr15 {
    public static void main(String args[])
    {
        float area;
        float L,B;
        Scanner obj = new Scanner(System.in);
        System.out.print("Enter the value of length : ");
        L = obj.nextFloat();
        System.out.print("Enter the value of breath : ");
        B = obj.nextFloat();
        Area o1 = new Area(L,B);

        area = o1.returnArea();

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

        System.out.println("");
        System.out.println("23DCS082 Malay Patel");
    }
}

class Area
{
    float length,breath,area;

    Area()
    {
        length = 0;
        breath = 0;
    }
    Area(float l , float b)
    {
        length=l;
        breath=b;
        area=length*breath;
    }
    public float returnArea()
    {
        return (area);
    }
}
```

OUTPUT:



```
Enter the value of length : 1.5
Enter the value of breath : 2.5
Area of Rectangle : 3.75

23DCS082 Malay Patel
```

CONCLUSION:

From above Java program we use default constructor to initialize the instance variable and parameterized constructor to set the value and return the area of rectangle using ‘returnArea’ method.

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

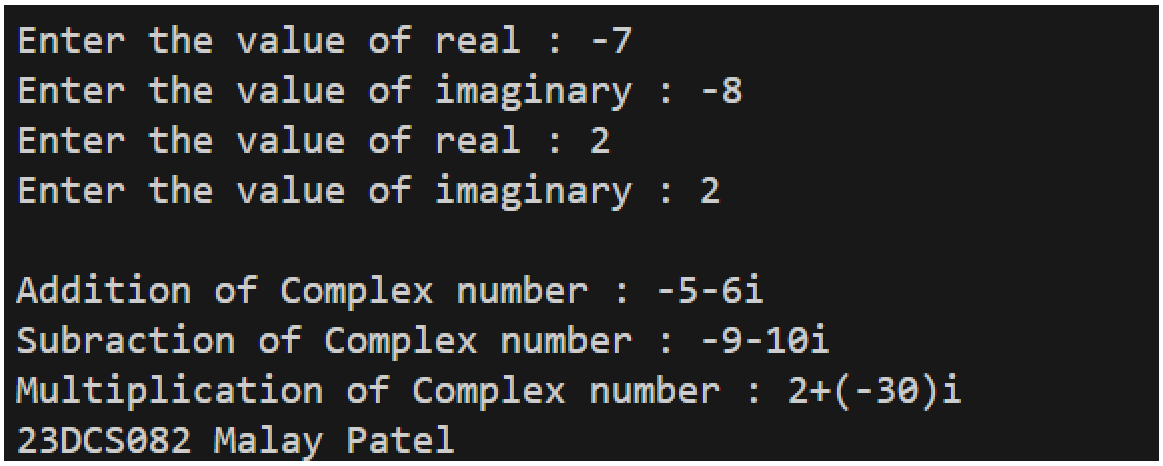
PROGRAM CODE:

```
 import java.util.Scanner;

public class pr16 {
    public static void main(String args[])
    {
        complex o1 =  new complex();
        complex o2 =  new complex();

        o1.get();
        o2.get();

        System.out.println("");

        o1.display(o2);

        System.out.println("");
        System.out.println("23DCS082 Malay Patel");
    }
}

class complex
{
    int real,imaginary;
```

```
complex()
{
   real = 0;
   imaginary = 0;
}

public void get()
{
   Scanner obj = new Scanner(System.in);
   System.out.print("Enter the value of real : ");
   real = obj.nextInt();
   System.out.print("Enter the value of imaginary : ");
   imaginary = obj.nextInt();
}

public void display(complex c)
{
   System.out.print("Addition of Complex number : ");
   if((imaginary + c.imaginary) > 0)
   {
      System.out.print((real + c.real) + "+" + (imaginary + c.imaginary) + "i");
   }
   else
   {
      System.out.print((real + c.real) + "" + (imaginary + c.imaginary) + "i");
   }
   System.out.println("");

   System.out.print("Subraction of Complex number : ");
   if((imaginary - c.imaginary) > 0)
   {
      System.out.print((real - c.real) + "+" + (imaginary - c.imaginary) + "i");
   }
   else
   {
      System.out.print((real - c.real) + "" + (imaginary - c.imaginary) + "i");
   }
   System.out.println("");

   System.out.print("Multiplication of Complex number : ");
   System.out.print(((real * c.real)-(imaginary * c.imaginary))+ "+"+ "(" +
((real*c.imaginary)+(imaginary*c.real))+ ")" +"i");
```

```
    }
}
```

OUTPUT:



```
Enter the value of real : -7
Enter the value of imaginary : -8
Enter the value of real : 2
Enter the value of imaginary : 2

Addition of Complex number : -5-6i
Subraction of Complex number : -9-10i
Multiplication of Complex number : 2+(-30)i
23DCS082 Malay Patel
```

CONCLUSION:

In this java program we use default constructor to initialize the value and then from get method we take input from user for two complex number for addition, subtraction, multiplication and then display it using display method.

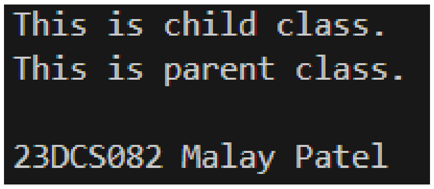
PART-IV Inheritance, Interface, Package

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.

PROGRAM CODE :

```
public class pr17 {
    public static void main(String args[])
    {
        child o = new child();

        o.ch();
        o.par();
        System.out.println("");
        System.out.println("23DCS082 Malay
Patel");
    }
}

class parent{
    public void par()
    {
        System.out.println("This is parent class.");
    }
}
class child extends parent{
    public void ch()
    {
        System.out.println("This is child class.");
    }
}
```

OUTPUT:


```
This is child class.
This is parent class.

23DCS082 Malay 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.

18. Create a class named 'Member' having the following members: Data members

1 - Name

2 - Age

3 - Phone number

4 - Address

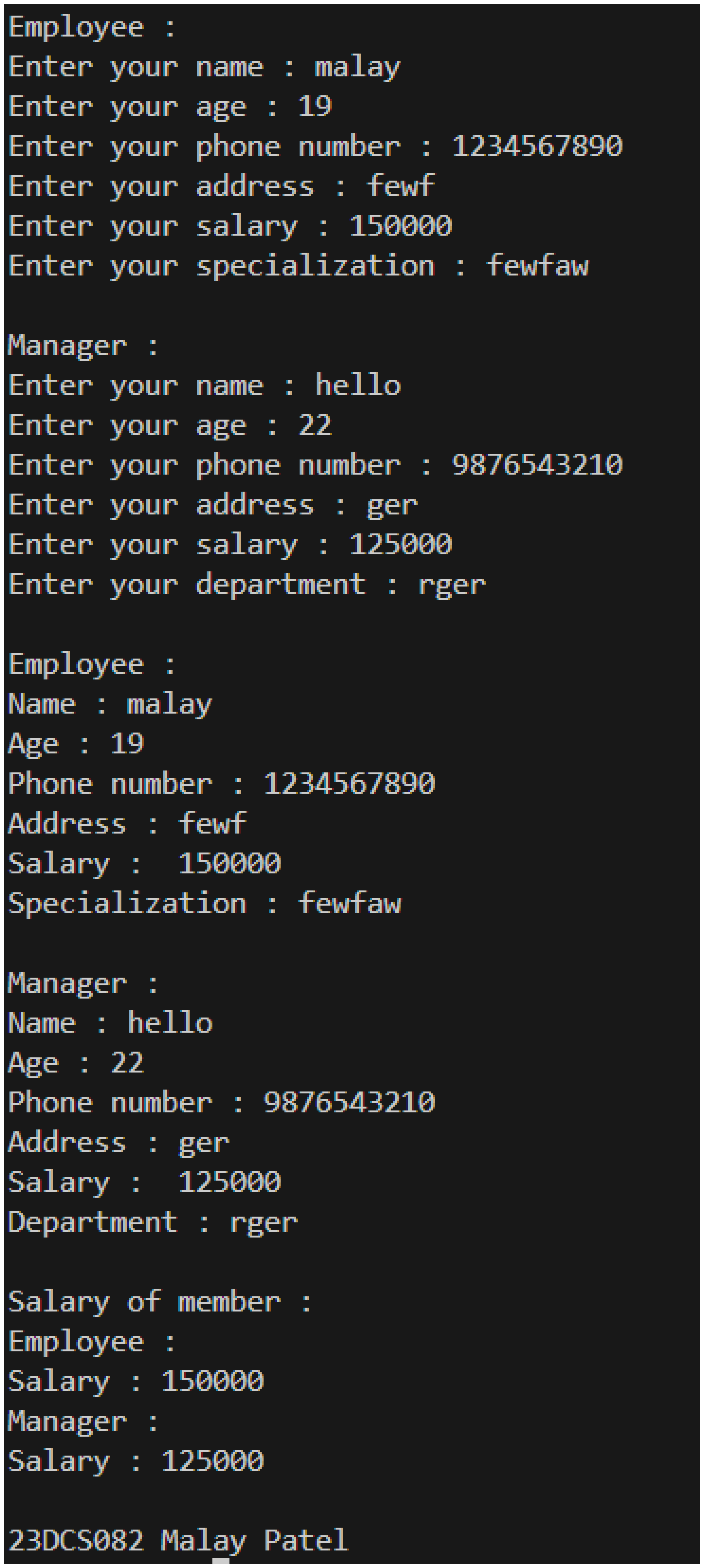
5 – Salary

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.Scanner;


public class pr18 {
    public static void main(String args[])
    {
        employee e = new employee();
        manager m = new manager();
        System.out.println("Employee : ");
        e.get();
        System.out.println("");
        System.out.println("Manager : " );
        m.get();
        System.out.println("");
        e.put();
        System.out.println("");
        m.put();
        System.out.println("");
        System.out.println("Salary of member : ");
        System.out.println("Employee : ");
        e.display();
        System.out.println("Manager : ");
        m.display();
        System.out.println("");
        System.out.println("23DCS082 Malay Patel");
    }
}

class Member{
    String name,add;
    int age, salary;
    long num;

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

class employee extends Member{
    String specialization;
    public void get()
    {
        Scanner obj = new Scanner(System.in);
        System.out.print("Enter your name : ");
        name = obj.nextLine();
        System.out.print("Enter your age : ");
        age = obj.nextInt();
        System.out.print("Enter your phone number : ");
        num = obj.nextLong();
        obj.nextLine();
        System.out.print("Enter your address : ");
        add = obj.nextLine();
        System.out.print("Enter your salary : ");
        salary = obj.nextInt();
        obj.nextLine();
        System.out.print("Enter your specialization : ");
        specialization = obj.nextLine();
        obj.close();
    }
    public void put()
    {
        System.out.println("Employee : ");
        System.out.println("Name : " + name);
        System.out.println("Age : "  + age);
        System.out.println("Phone number : " + num);
        System.out.println("Address : " + add);
        System.out.println("Salary :  " + salary);
        System.out.println("Specialization : " + specialization);
    }
}

class manager extends Member{
    String department;
    public void get()
    {
        Scanner obj = new Scanner(System.in);
        System.out.print("Enter your name : ");
        name = obj.nextLine();
        System.out.print("Enter your age : ");
        age = obj.nextInt();
        System.out.print("Enter your phone number : ");
        num = obj.nextLong();
        obj.nextLine();
        System.out.print("Enter your address : ");
        add = obj.nextLine();
        System.out.print("Enter your salary : ");
        salary = obj.nextInt();
        obj.nextLine();
        System.out.print("Enter your department : ");
        department = obj.nextLine();
        obj.close();
    }
    public void put()
    {
        System.out.println("Manager : ");
        System.out.println("Name : " + name);
        System.out.println("Age : "  + age);
        System.out.println("Phone number : " + num);
        System.out.println("Address : " + add);
        System.out.println("Salary :  " + salary);
        System.out.println("Department : " + department);
    }
}
```

OUTPUT:



```
Employee :
Enter your name : malay
Enter your age : 19
Enter your phone number : 1234567890
Enter your address : fewf
Enter your salary : 150000
Enter your specialization : fewfaw

Manager :
Enter your name : hello
Enter your age : 22
Enter your phone number : 9876543210
Enter your address : ger
Enter your salary : 125000
Enter your department : rger

Employee :
Name : malay
Age : 19
Phone number : 1234567890
Address : fewf
Salary :  150000
Specialization : fewfaw

Manager :
Name : hello
Age : 22
Phone number : 9876543210
Address : ger
Salary :  125000
Department : rger

Salary of member :
Employee :
Salary : 150000
Manager :
Salary : 125000

23DCS082 Malay Patel
```

CONCLUSION:

This program demonstrates **inheritance**. The `employee` and `manager` classes inherit from `Member`, sharing details like name and salary. Each class adds its own info (`specialization` for employees, `department` for managers). It collects and displays details, showing how inheritance helps reuse common features.

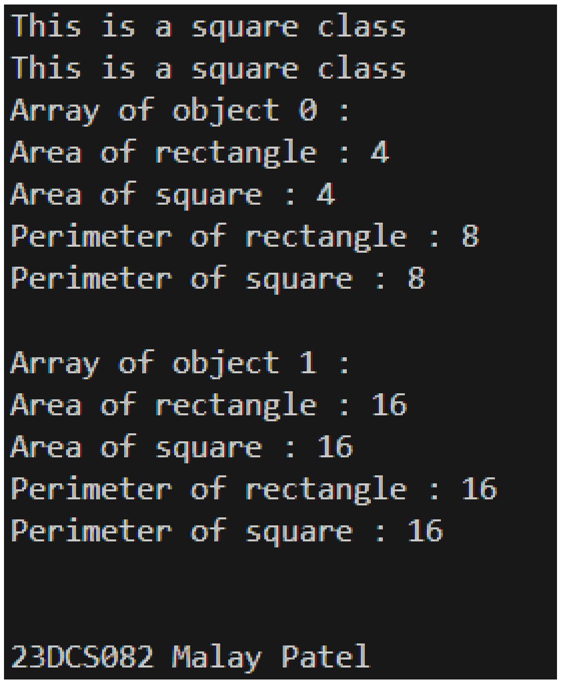
19. Create a class named 'Rectangle' with two data members 'length' and 'breadth' and two methods to print the area and perimeter of the rectangle respectively. Its constructor having parameters for length and breadth is used to initialize length and breadth of the rectangle. Let class 'Square' inherit 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:

```
public class pr19 {
    public static void main(String args[])
    {
        // square o = new square(2);
        // o.area();
        // o.perimeter();

        square[] arr = new square[2];
        arr[0] = new square(2);
        arr[1] = new square(4);

        for (int i = 0; i < 2 ; i++)
        {
            System.out.println("Array of object " + i + " : ");
            arr[i].area();
            arr[i].perimeter();
            System.out.println("");
        }
        System.out.println("");
        System.out.println("23DCS082 Malay Patel");
    }
}

class rectangle{
    int length, breath;

    rectangle()
    {
        length = 0;
        breath = 0;
    }
    rectangle(int l , int b)
    {
        length = l;
        breath = b;
    }
    public void area()
    {
        System.out.println("Area of rectangle : " + (length*breath));
        System.out.println("Area of square : " + (length*length));
    }
    public void perimeter()
    {
        System.out.println("Perimeter of rectangle : " + (2*(length+breath)));
        System.out.println("Perimeter of square : " + (4*length));
    }
```

```
}


class square extends rectangle{
    square(int s)
    {
        System.out.println("This is a square class");
       // there is a ERROR because super must be the first statement of the block but somehow it runs completely fine.
        super(s,s);
    }
}
```

OUTPUT:



```
This is a square class
This is a square class
Array of object 0 :
Area of rectangle : 4
Area of square : 4
Perimeter of rectangle : 8
Perimeter of square : 8

Array of object 1 :
Area of rectangle : 16
Area of square : 16
Perimeter of rectangle : 16
Perimeter of square : 16



23DCS082 Malay Patel
```

CONCLUSION:

This program shows how inheritance works in Java. The `square` class extends `rectangle` and reuses its methods to calculate area and perimeter. It creates an array of `square` objects and prints their details. Despite a comment about an error with `super()`, the program runs fine.

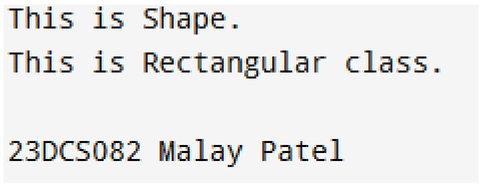
20. Create a class named 'Shape' with a method to print "This is This is shape". Then create two other classes named 'Rectangle', 'Circle' inheriting the Shape class, both having a method to print "This is rectangular shape" and "This is circular shape" respectively. Create a subclass 'Square' of 'Rectangle'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

```
public class pr20 {
    public static void main(String args[])
    {
        square o = new square();
        o.sha();
        o.rect();
        System.out.println("");
        System.out.println("23DCS082 Malay Patel");
    }
}

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

class rectangle extends shape{
    public void rect()
    {
        System.out.println("This is Rectangular class.");
    }
}
```

```
class circle extends shape{
    public void cir()
    {
        System.out.println("This is Circular class.");
    }
}


class square extends rectangle{
    public void squ()
    {
        System.out.println("Square is a Rectangle.");
    }
}
//Runs on IntellIJ Idea
```

OUTPUT:



```
This is Shape.
This is Rectangular class.

23DCS082 Malay Patel
```

CONCLUSION:

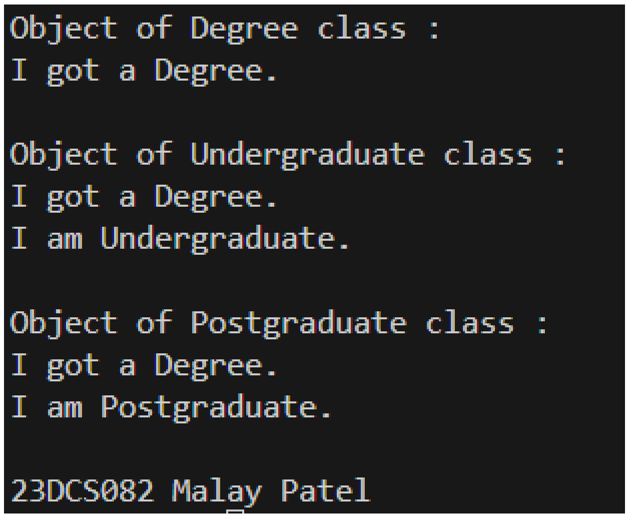
In this java program we learnt about how to use hierarchical inheritance which parent class “Shape” and child class of “circle” and “rectangle”.

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:

```
public class pr21 {
   public static void main(String args[])
   {
      degree o1 = new degree();
      undergraduate o2 = new undergraduate();
      postgraduate o3 = new postgraduate();

      System.out.println("Object of Degree class : ");
      o1.getDegree();
      System.out.println("");
      System.out.println("Object of Undergraduate class : ");
      o2.getDegree();
      o2.ug();
      System.out.println("");
      System.out.println("Object of Postgraduate class : ");
      o3.getDegree();
      o3.pg();
      System.out.println("");
      System.out.println("23DCS082 Malay Patel");
   }
}

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

class undergraduate extends degree
{
    public void ug()
    {
        System.out.println("I am Undergraduate.");
    }
}

class postgraduate extends degree
{
    public void pg()
    {
        System.out.println("I am Postgraduate.");
    }
}
```

OUTPUT:



```
Object of Degree class :
I got a Degree.

Object of Undergraduate class :
I got a Degree.
I am Undergraduate.

Object of Postgraduate class :
I got a Degree.
I am Postgraduate.

23DCS082 Malay Patel
```

CONCLUSION:

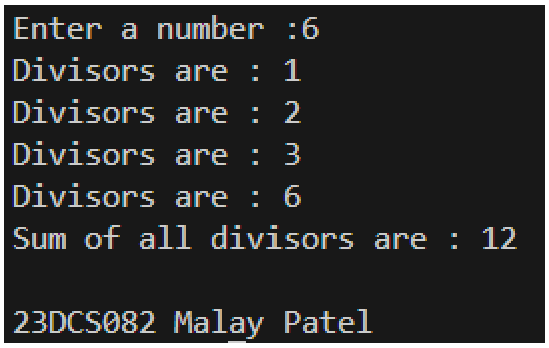
This program shows hierarchical inheritance. Both `undergraduate` and `postgraduate` classes inherit from the `degree` class, allowing them to use its `getDegree()` method. Each class adds its own method for specific details, showing how inheritance helps share features across related classes.

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.Scanner;


interface AdvancedArithmetic
{
    int divisor_sum(int n);
}
class calledMyCalculator implements AdvancedArithmetic
{
    public int divisor_sum(int n)
    {
        int d=n;
        int total =0 ;
        for (int i = 1; i <= d; i++)
        {
            if(d%i==0)
            {
                System.out.println("Divisors are : " + i);
                total = total + i;
            }
        }
        return total;
    }
}
public class pr22 {

    public static void main(String args[])
    {
        int sum,number;
        Scanner obj = new Scanner(System.in);

        calledMyCalculator o = new calledMyCalculator();

        System.out.print("Enter a number :");
        number = obj.nextInt();
        sum = o.divisor_sum(number);

        System.out.println("Sum of all divisors are : " + sum);
        System.out.println("");
        System.out.println("23DCS082 Malay Patel");
        obj.close();
    }
}
```

OUTPUT:



```
Enter a number :6
Divisors are : 1
Divisors are : 2
Divisors are : 3
Divisors are : 6
Sum of all divisors are : 12

23DCS082 Malay Patel
```

CONCLUSION:

This program shows how to use an interface in Java. The `AdvancedArithmetic` interface has a method `divisor_sum()`, which is implemented in the `calledMyCalculator` class. The program calculates and prints all divisors of a number and their sum. It demonstrates how interfaces define a contract that classes must follow.

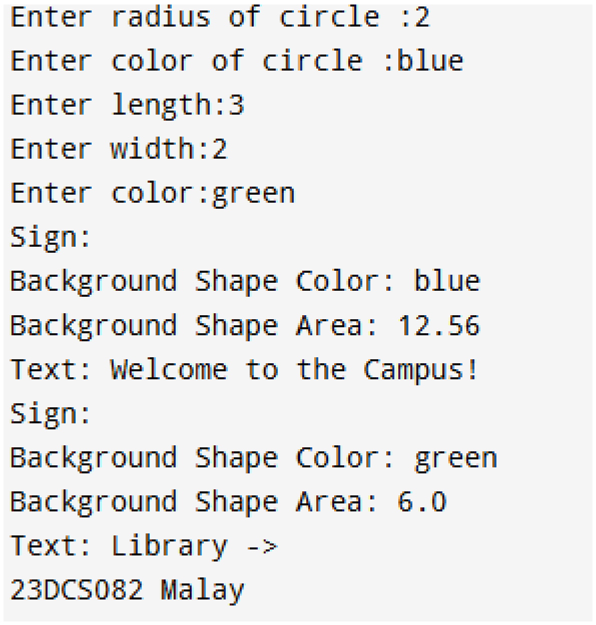
23. Assume you want to capture shapes, which can be either circles (with a radiusand a color) or rectangles (with a length, width, and color). You also want to be able to create signs (to post in the campus center, for example), each of which has a shape (for the background of the sign) and the text (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:

```
//not working vs
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 prac23 {
    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();
        sc.close();

        System.out.println("23DCS082 Malay");
    }
}
```

OUTPUT:



```
Enter radius of circle :2
Enter color of circle :blue
Enter length:3
Enter width:2
Enter color:green
Sign:
Background Shape Color: blue
Background Shape Area: 12.56
Text: Welcome to the Campus!
Sign:
Background Shape Color: green
Background Shape Area: 6.0
Text: Library ->
23DCS082 Malay
```

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.

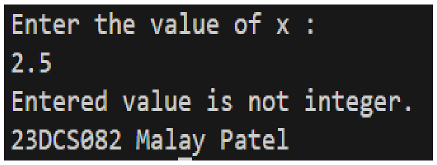
PART-V Exception Handling

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.

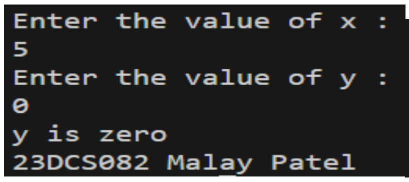
PROGRAM CODE :

```
import java.util.InputMismatchException;

import java.util.Scanner;

public class pr24 {

    public static void main(String args[])

    {

        Scanner obj = new Scanner(System.in);

        int x,y;


        try

        {

            System.out.println("Enter the value of x
: ");

            x = obj.nextInt();

            System.out.println("Enter the value of y:
");

            y = obj.nextInt();
            System.out.println("Division is " x/y);
        }
        catch (InputMismatchException e)
        {
            System.out.println("Entered value is not
integer.");
        }
        catch (ArithmeticException e)
        {
                System.out.println("y is zero");
        }
        obj.close();
        System.out.println("23DCS082 Malay
Patel");
    }
```

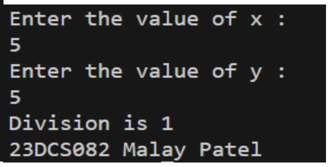
OUTPUT:



```
Enter the value of x :
2.5
Entered value is not integer.
23DCS082 Malay Patel
```



```
Enter the value of x :
5
Enter the value of y :
0
y is zero
23DCS082 Malay Patel
```



```
Enter the value of x :
5
Enter the value of y :
5
Division is 1
23DCS082 Malay Patel
```

CONCLUSION:

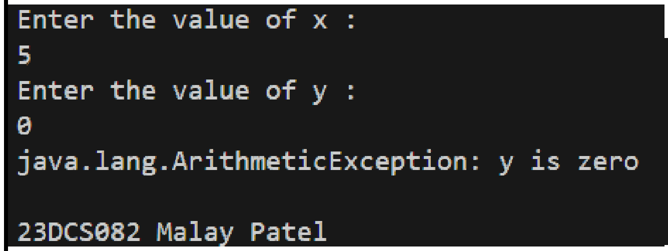
In this java program we take two intergers x and y , we compute x/y and it will give exception if x & y are not integers or y is zero and we handle the exception and report the exception.

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

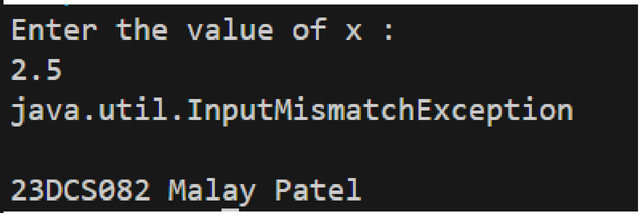
Program Code:

```
// import java.util.InputMismatchException;
import java.util.InputMismatchException;
import java.util.Scanner;

public class pr25 {
    public static void main(String args[])
    {
        Scanner obj = new Scanner(System.in);
        int x,y;
        try
        {
            System.out.println("Enter the value of x : ");
            x = obj.nextInt();
            System.out.println("Enter the value of y : ");
            y = obj.nextInt();
            obj.close();
            if (y==0)
            {
                throw new ArithmeticException("y is zero");
            }
            System.out.println("Division is " + x/y);
        }
        catch(InputMismatchException e)
        {
            System.out.println(e);
        }
        catch (ArithmeticException e)
        {
            System.out.println(e);
        }
        System.out.println("");
        System.out.println("23DCS082 Malay Patel");
    }
}
```

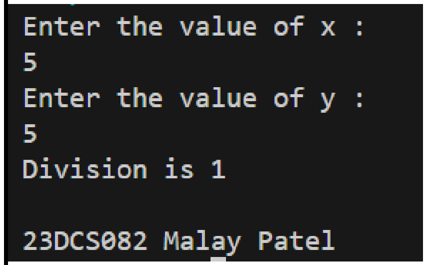
Output:



```
Enter the value of x :
5
Enter the value of y :
0
java.lang.ArithmeticException: y is zero

23DCS082 Malay Patel
```



```
Enter the value of x :
2.5
java.util.InputMismatchException

23DCS082 Malay Patel
```



```
Enter the value of x :
5
Enter the value of y :
5
Division is 1

23DCS082 Malay Patel
```

Conclusion:

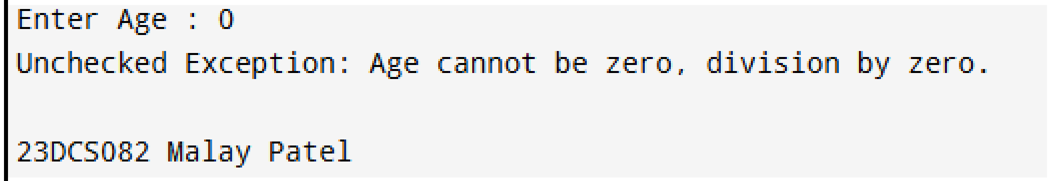
In this java program we take two intergers x & y and compute x/y, it will give expection if x & y are not intergers or y is zero and we handle the exception using try and catch block.

26. Write a java program to generate user defined exception using “throw” and “throws” keyword. Also Write a java that differentiates checked and unchecked exceptions. (Mention at least two checked and two unchecked exceptions in program).

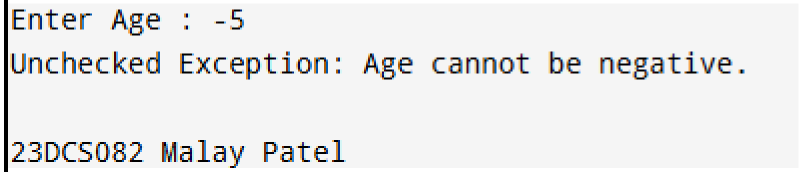
Program Code:

```
import java.util.Scanner;


class AgeNotValidException extends Exception {
    public AgeNotValidException(String message) {
        super(message);
    }
}

class AgeValidator {

    public void validate(int age) throws AgeNotValidException, java.io.IOException {

        // Unchecked Exception: ArithmeticException
        if (age == 0) {
            throw new ArithmeticException("Age cannot be zero, division by zero.");
        }

        // Unchecked Exception: NullPointerException
        if (age < 0) {
                throw new NullPointerException("Age cannot be negative, null string
encountered.");
        }

        // Checked Exception: User-defined exception
        if (age < 18) {
            throw new AgeNotValidException("Age must be 18 or older.");
        }

        // Checked Exception: IOException
        if (age > 100) {
            throw new java.io.IOException("Age cannot be greater than 100.");
        }
        System.out.println("Age is valid.");
    }
}

public class pr26 {
    public static void main(String[] args) {
        AgeValidator validator = new AgeValidator();

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter Age : ");
        int age = sc.nextInt();
        try {
            validator.validate(age);
        } catch (ArithmeticException e) {
            System.out.println("Unchecked Exception: " + e.getMessage());
        } catch (NullPointerException e) {
            System.out.println("Unchecked Exception: " + e.getMessage());
        } catch (AgeNotValidException e) {
            System.out.println("Checked Exception: " + e.getMessage());
        } catch (java.io.IOException e) {
            System.out.println("Checked Exception: " + e.getMessage());
        }
        sc.close();

        System.out.println("");
        System.out.println("23DCS082 Malay Patel");
    }
}
```

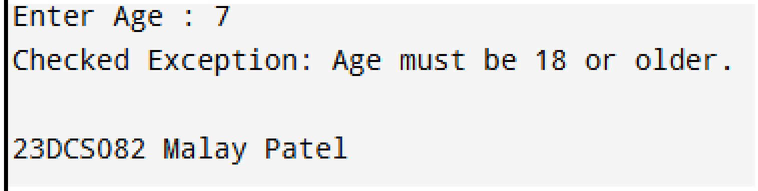
Output:



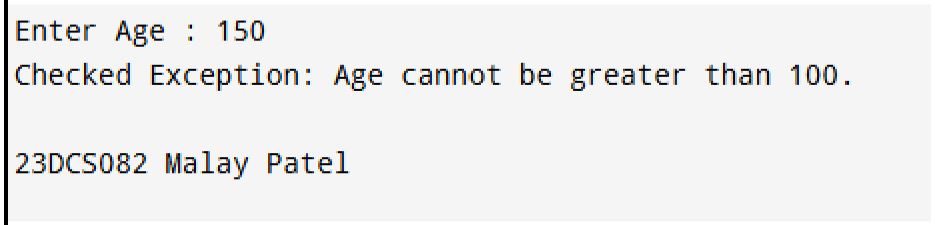
```
Enter Age : 0
Unchecked Exception: Age cannot be zero, division by zero.

23DCS082 Malay Patel
```



```
Enter Age : -5
Unchecked Exception: Age cannot be negative.

23DCS082 Malay Patel
```



```
Enter Age : 7
Checked Exception: Age must be 18 or older.

23DCS082 Malay Patel
```



```
Enter Age : 150
Checked Exception: Age cannot be greater than 100.

23DCS082 Malay Patel
```

Conclusion:

In this java program we learned the concept of throw and throws keyword and used them for checked and unchecked exception for kinds of two checked and two unchecked exception.

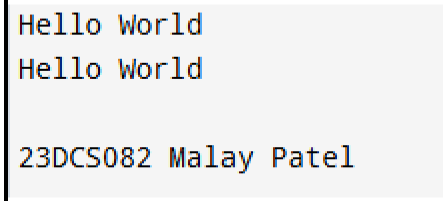
PART-VII Multithreading

No.	Aim of the Practical
32.	Write a program to create thread which display "Hello World" message. A. by Thread class B. by using Runnable interface

PROGRAM CODE

```
public class pr32 {
    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("23DCS082 Malay
Patel");
    }
}

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

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

OUTPUT:



```
Hello World
Hello World

23DCS082 Malay Patel
```

CONCLUSION:

In this java program we learned about to use thread and runnable class and print “Hello World”.

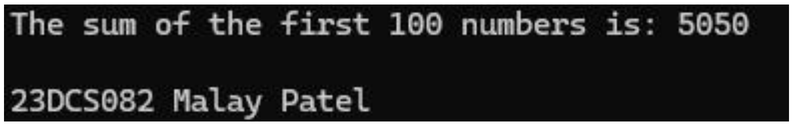
33. **Write a program which takes N and number of threads as an argument. Program should distribute the task of summation of N numbers amongst number of threads and final result to be displayed on the console.**

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; // Store the partial
sum in the result array
    }
}

public class SumWithThreads {
    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]);

        // Array to hold the partial results
        int[] result = new int[numThreads];

        // Calculate the range for each thread
        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; // Add the
remainder to the last thread's range
```

```
            }

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

            start = end + 1;
        }

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

        // Calculate the final sum
        int finalSum = 0;
        for (int sum : result) {
            finalSum += sum;
        }

        // Display the final result
        System.out.println("The sum of the first "
+ N + " numbers is: " + finalSum);
        System.out.println("23DCS082 Malay
Patel");
    }
}
```

OUTPUT:



```
The sum of the first 100 numbers is: 5050

23DCS082 Malay Patel
```

CONCLUSION:

In this Java program, we learned how to use threads and the `Runnable` interface to parallelize the summation of numbers. By dividing the range across multiple threads, we efficiently calculated the sum of the first N numbers and displayed the result.

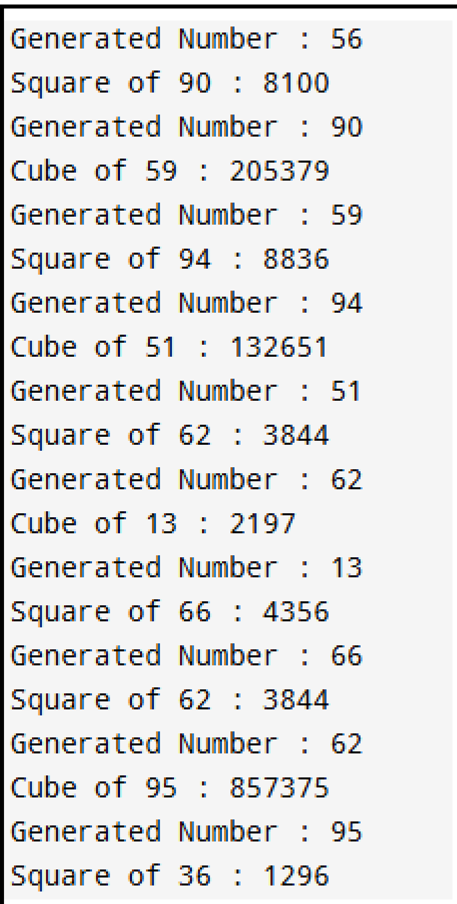
34. Write a java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of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

PROGRAM:

```
import java.util.Random;


public class pr34 {
    public static void main(String[] args) {


        generate t1 = new generate();
        Sq t2 = new Sq();
        Cube t3 = new Cube();


        t1.start();
        t2.start();
        t3.start();
    }
}
```

```
class generate extends Thread {
    static int num;
    Random random = new Random();

    public void run() {
        while (true)
        {
            num = random.nextInt(100);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            System.out.println("Generated Number : " + num);
        }
    }
}

class Sq extends Thread {
    public void run()
    {
        while (true)
        {
            if(generate.num % 2 == 0)
            {
                        System.out.println("Square   of   "   +   generate.num   +   "   :   "   +
(generate.num*generate.num));
            }
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

class Cube extends Thread {
    public void run()
    {
        while (true)
        {
            if(generate.num % 2 != 0)
            {
                        System.out.println("Cube of " + generate.num + " : " +
(generate.num*generate.num*generate.num));
            }
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

OUTPUT:



```
Generated Number : 56
Square of 90 : 8100
Generated Number : 90
Cube of 59 : 205379
Generated Number : 59
Square of 94 : 8836
Generated Number : 94
Cube of 51 : 132651
Generated Number : 51
Square of 62 : 3844
Generated Number : 62
Cube of 13 : 2197
Generated Number : 13
Square of 66 : 4356
Generated Number : 66
Square of 62 : 3844
Generated Number : 62
Cube of 95 : 857375
Generated Number : 95
Square of 36 : 1296
```

CONCLUSION:

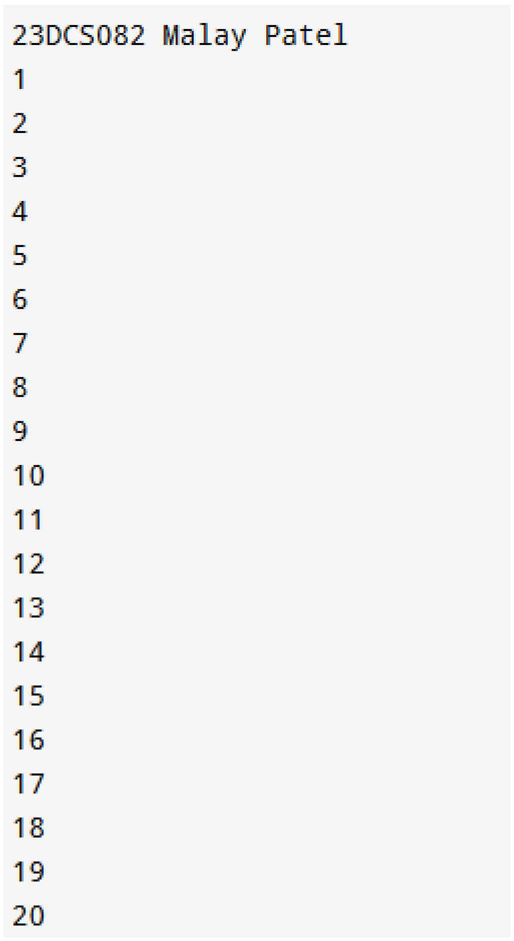
In this java program we learned about use thread and generate random numbers using thread and calculate square of that number if that number is even or cube if that number is odd.

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:

```
public class pr35 {
    public static void main(String[] args) {


        Inc t1 = new Inc();
        t1.start();
        System.out.println("");
        System.out.println("23DCS082 Malay Patel");


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

OUTPUT:



```
23DCS082 Malay Patel
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

CONCLUSION:

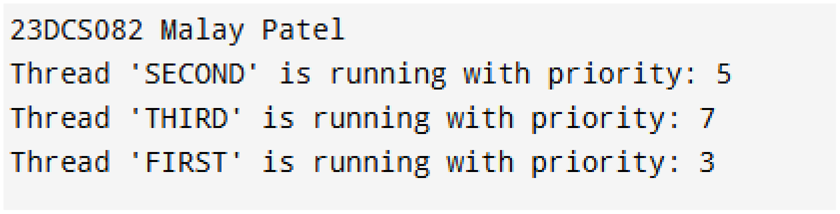
In this java program we learned the sleep method which helps to delay the following function, in this program we print number by incrementing it by 1 and gives delays of 1 second.

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:

```
class FirstThread extends Thread {
    public void run() {
        System.out.println("Thread 'FIRST' is running with priority: " + this.getPriority());
    }
}


class SecondThread extends Thread {
    public void run() {
        System.out.println("Thread 'SECOND' is running with priority: " + this.getPriority());
    }
}

class ThirdThread extends Thread {
    public void run() {
        System.out.println("Thread 'THIRD' is running with priority: " + this.getPriority());
    }
}

public class ThreadPriorityDemo {
    public static void main(String[] args) {
        // Create the thread objects
        FirstThread first = new FirstThread();
        SecondThread second = new SecondThread();
        ThirdThread third = new ThirdThread();

        // Set priorities
        first.setPriority(3); // Priority of FIRST thread set to 3
        second.setPriority(5); // Default priority 5
        third.setPriority(7); // Priority of THIRD thread set to 7

        // Start the threads
        first.start();
        second.start();
        third.start();

        System.out.println("");
        System.out.println("23DCS082 Malay Patel");
    }
}
```

OUTPUT:



```
23DCS082 Malay Patel
Thread 'SECOND' is running with priority: 5
Thread 'THIRD' is running with priority: 7
Thread 'FIRST' is running with priority: 3
```

CONCLUSION:

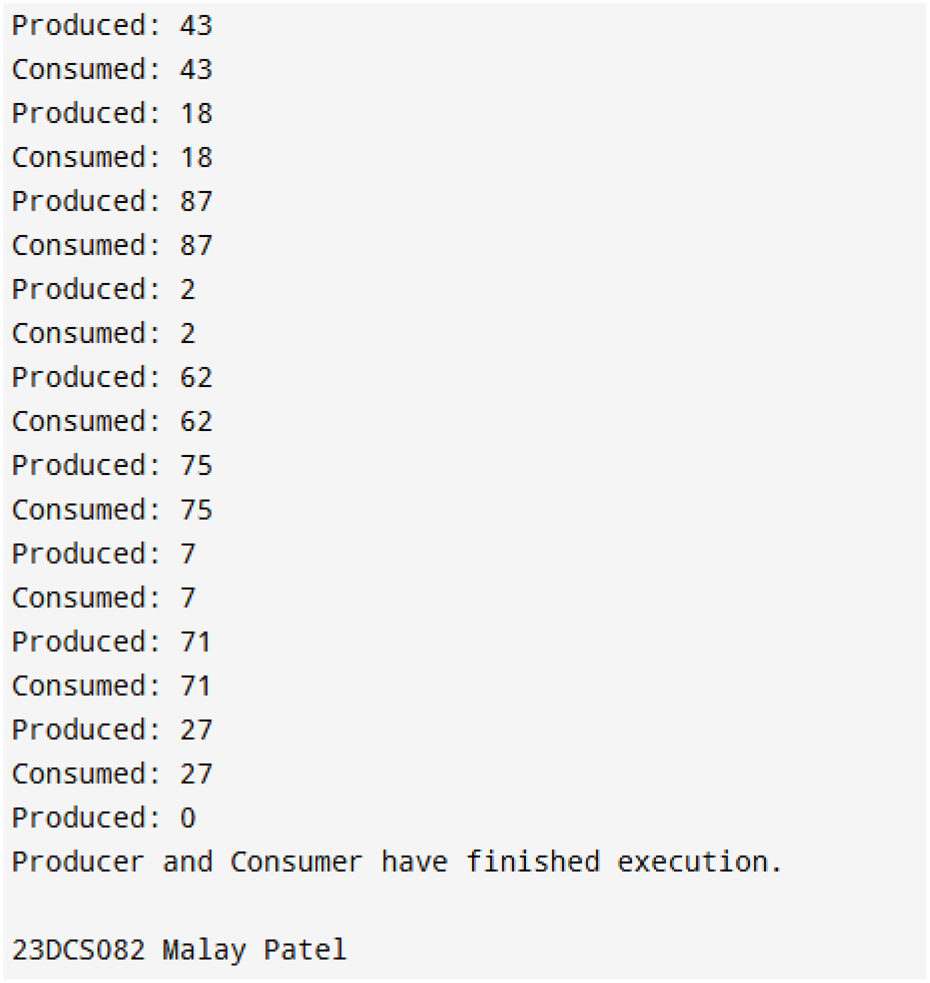
In this java program we learned how to set priorities to thread and display the priorities of different threads.

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

PROGRAM:

```
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 ProducerConsumerThreadExample {
    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("\n23DCS082 Malay Patel");
    }
}
```

OUTPUT:



```
Produced: 43
Consumed: 43
Produced: 18
Consumed: 18
Produced: 87
Consumed: 87
Produced: 2
Consumed: 2
Produced: 62
Consumed: 62
Produced: 75
Consumed: 75
Produced: 7
Consumed: 7
Produced: 71
Consumed: 71
Produced: 27
Consumed: 27
Produced: 0
Producer and Consumer have finished execution.

23DCS082 Malay Patel
```

CONCLUSION:

In this Java program, we learned how to implement the Producer-Consumer problem using threads and synchronization. The producer creates items, and the consumer consumes them from a shared buffer, ensuring thread coordination using `synchronized` methods and `notify`. This demonstrates how to manage shared resources between multiple threads safely.

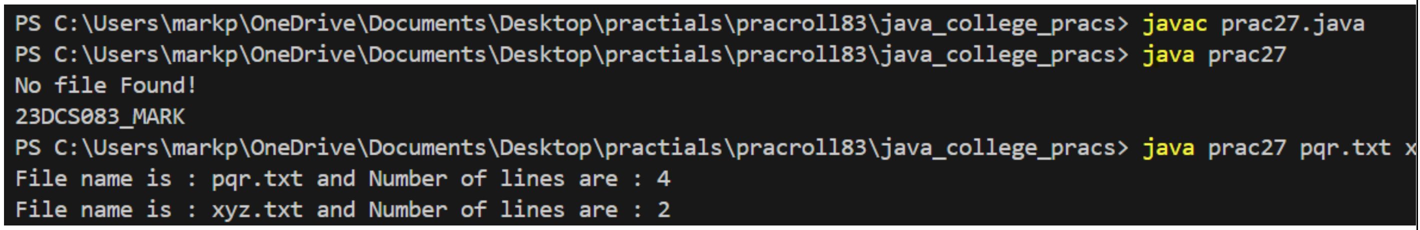
PART-VI File Handling & Streams

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);

                }

            }

        }

        System.out.println("23DCS082 Malay
Patel");

    }

}
```

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!
23DCS083_MARK
PS C:\Users\markp\OneDrive\Documents\Desktop\practials\pracroll83\java_college_pracs> java prac27 pqr.txt x
File name is : pqr.txt and Number of lines are : 4
File name is : xyz.txt and Number of lines are : 2
```

CONCLUSION:

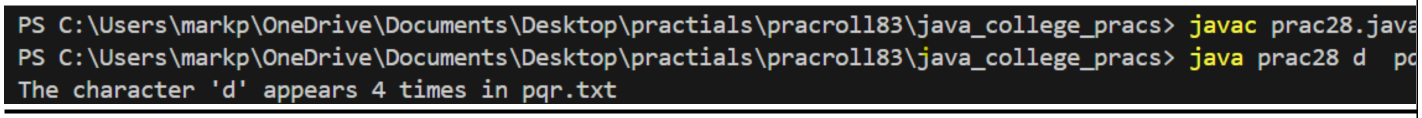
This Java program reads several files named by the command line arguments and counts the number of lines in each. If no files are provided as command-line arguments, it will print out the appropriate message. Exception handling ensures graceful error management during file reading, thus a stable program.

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());
}
System.out.println("23DCS082 Malay Patel”);
}}
```

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  pq
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.

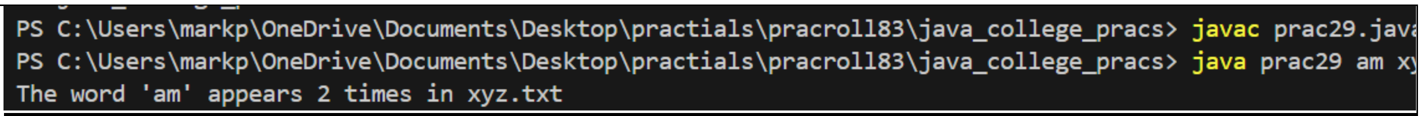
29 Write a Java Program to Search for a given word in a File. Also show use of Wrapper Class with an example.

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());

}

System.out.println("23DCS082 Malay Patel ");

} }
```

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 xy
The word 'am' appears 2 times in xyz.txt
```

CONCLUSION:

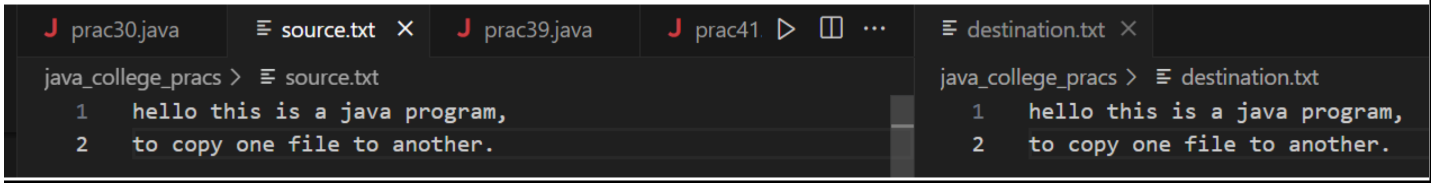
This Java program effectively searches for a specified word in a given file and counts its occurrences. It demonstrates the use of the Integer wrapper class to manage the count, showcasing how wrapper classes can be used for object manipulation in Java.

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:

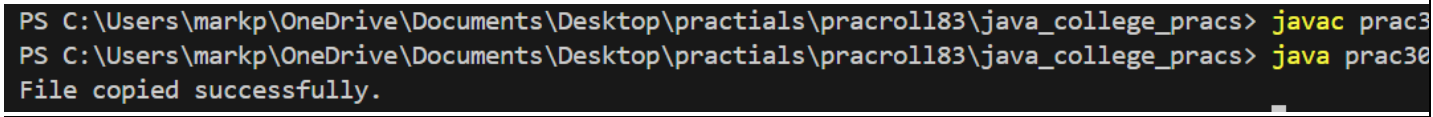
```
import java.io.*;

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:


prac30.java
source.txt
prac39.java
prac41
destination.txt
java_college_pracs > source.txt
java_college_pracs > destination.txt

```
hello this is a java program,
to copy one file to another.
```

```
hello this is a java program,
to copy one file to another.
```



```
PS C:\Users\markp\OneDrive\Documents\Desktop\practials\pracroll83\java_college_pracs> javac prac3
PS C:\Users\markp\OneDrive\Documents\Desktop\practials\pracroll83\java_college_pracs> java prac30
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.

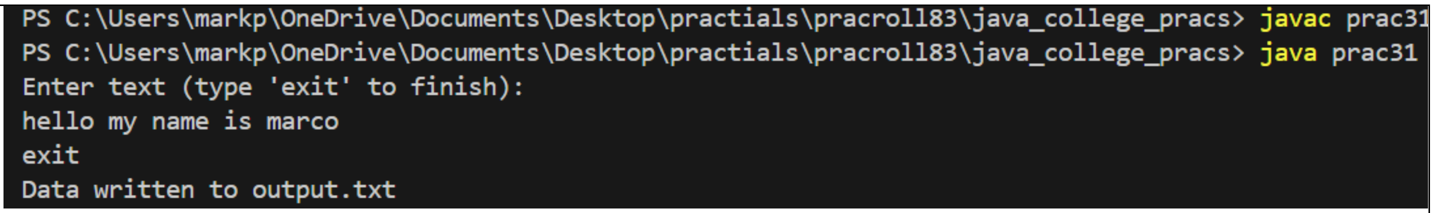
31. Write a program to show use of character and byte stream. Also show use of BufferedReader / BufferedWriter to read console input and write them into a file.

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());

    }

    System.out.println("23DCS082 Malay Patel ");

  }

}
```

OUTPUT:



```
PS C:\Users\markp\OneDrive\Documents\Desktop\practials\pracroll83\java_college_pracs> javac prac31
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

PART-VIII Collection Framework and Generic

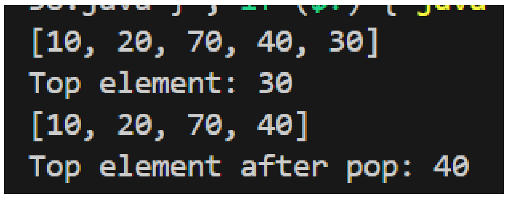
No.	Aim of the Practical

38.	Design a Custom Stack using ArrayList class, which implements following functionalities of stack. My Stack
	-list ArrayList<Object>: A list to store elements.
	+isEmpty: boolean: Returns true if this stack is empty.
	+getSize(): int: Returns number of elements in this stack.
	+peek(): Object: Returns top element in this stack without removing it.
	+pop(): Object: Returns and Removes the top elements in this stack.
	+push(o: object): Adds new element to the top of this stack.
	
	PROGRAM CODE:
	
	import java.util.*;
	
	class StackList{
	
	    private ArrayList<Integer> STL;
	
	    public StackList(){
	        STL = new ArrayList<>();
	    }

```
public boolean isEmpty() {
    return STL.isEmpty();
}

public void push(Integer value) {

    STL.add(value);
    //updateFile();
}

public void pop(){
    if(STL.isEmpty()){
        System.out.println("Stack is empty");
        //return null;
    }
    else{
         STL.remove(STL.size()-1);
    }
}

public Integer peek(){
    if(STL.isEmpty()){
        System.out.println("Stack is empty");
        return null;
    }
    else{

        return STL.get(STL.size()-1);
    }
}

public void display(){
    System.out.println(STL);
}

public void clr(){
    STL.clear();
}

public void size1(){
    System.out.println(STL.size());
}
```

```
    public void updatafile(){

    }

}
public class prac38 {
    public static void main(String[] args) {

        StackList stack = new StackList();
        stack.push(10);
        stack.push(20);
        stack.push(70);
        stack.push(40);
        stack.push(30);
        stack.display();
        System.out.println("Top element: " + stack.peek()); // Outputs: 30
        stack.pop();
        stack.display();


        // System.out.println("Popped element: " + stack.pop()); // Outputs: 30
        System.out.println("Top element after pop: " + stack.peek()); // Outputs: 20
        System.out.println("23DCS082 Malay Patel");
    }
}
```

OUTPUT:



```
[10, 20, 70, 40, 30]
Top element: 30
[10, 20, 70, 40]
Top element after pop: 40
```

CONCLUSION:

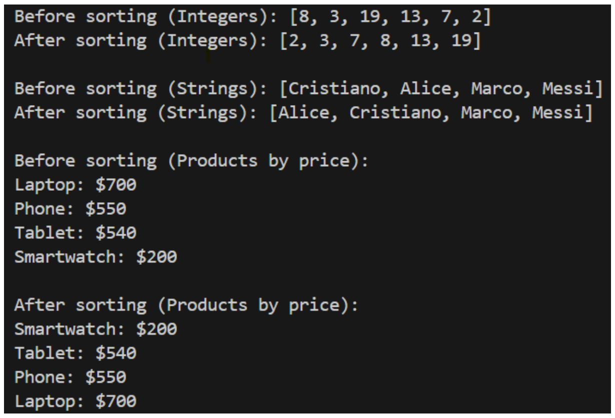
This program implements a stack using an ArrayList, providing basic stack operations like push, pop, peek, and clear. It allows adding and removing elements from the stack and displays the current stack contents. The program efficiently handles stack operations and checks for underflow when the stack is empty.

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);

        }

        System.out.println("23DCS082 Malay Patel ");

    }

}
```

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.

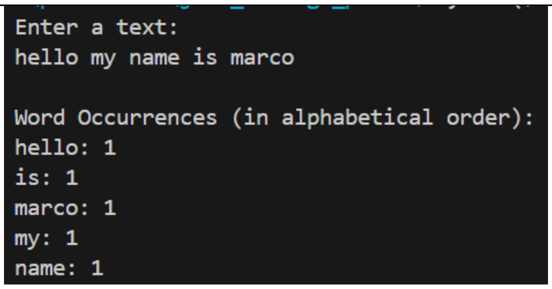
40.	Write a program that counts the occurrences of words in a text and displays the words

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());

 System.out.println("23DCS082 Malay Patel ");



} } }
```

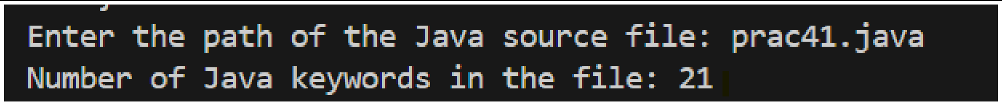
OUTPUT:

	
	```
	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);

}

System.out.println("23DCS082 Malay Patel ");

} }
```

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.