

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

GOVERNMENT ENGINEERING COLLEGE

KOTTAYAM - 686 501



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CSL 203 – OBJECT ORIENTED PROGRAMMING LAB

2023-2024

Submitted by

JESSIN SUNNY(KTE22CS036)



**KERALA TECHNOLOGICAL UNIVERSITY
THIRUVANANTHAPURAM**

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

GOVERNMENT ENGINEERING COLLEGE



KOTTAYAM – 686 051

DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

LABORATORY RECORD

*This is to certify that this is a bonafide record of the work done by **JESSIN SUNNY** of 3rd Semester (**KTE22CS036**) in **CSL 203** – **OBJECT ORIENTED PROGRAMMING LAB** during the academic year **2023-2024**.*

Staff in Charge

Internal Examiner

External Examiner

DEPARTMENT VISION

TO BE A CENTER OF EXCELLENCE FOR NURTURING
THE YOUNG MINDS TO BECOME INNOVATIVE
COMPUTING PROFESSIONALS FOR THE
EMPOWERMENT OF SOCIETY.

DEPARTMENT MISSION

1. TO OFFER A SOLID FOUNDATION IN COMPUTING
AND TECHNOLOGY FOR CRAFTING COMPETENT
PROFESSIONALS.
2. TO PROMOTE INNOVATIVE AND ENTREPRENEURIAL
SKILLS OF STUDENTS BY EXPOSING THEM TO THE
FOREFRONT OF DEVELOPMENTS IN THE FIELD OF
COMPUTING.
3. TO INCULCATE STRONG ETHICAL VALUES IN THE
YOUNG MINDS TO WORK WITH COMMITMENT FOR THE
PROGRESS OF THE NATION.

COURSE OUTCOMES

At the end of the course, the student should be able to

CO1	Implement the Object-oriented concepts — constructors, inheritance, method overloading & overriding and polymorphism in Java (Cognitive Knowledge Level: Apply)
C02	Implement programs in Java which use datatypes, operators, control statements, built-in packages & interfaces, Input/Output streams and files (Cognitive Knowledge Level: Apply)
C03	Implement robust application programs in Java using exception handling (Cognitive Knowledge Level:
C04	Implement application programs in Java using Multithreading (Cognitive Knowledge Level: Apply)
C05	Implement Graphical User Interface based application programs by utilizing event handling features and Swing in Java (Cognitive Knowledge Level: Apply)

INDEX

EXP NO	EXPERIMENT NAME	PAGE NO
SESSION A(1)		
1	Palindrome	1
2	Frequency of Character in String	3
3	Matrix Multiplication	5
SESSION A(2)		
1	Palindrome	8
2	Frequency of Character in String	11
3	Matrix Multiplication	14
SESSION B		
4	Inheritance	17
5	Polymorphism	22
SESSION C		
7	File Handling Using Reader/Writer	25
8	Handling File Exceptions	27
9	String Tokenizer	30
SESSION D		
10	Exception Handling	32
11	Thread Generation	35
12	Thread Synchronization	41
SESSION E		

13	Calculator	46
14	Traffic Light	53
SESSION F		
16	Doubly Linked List	57
17	Quick Sort	62
18	Binary Search	65
SESSION P		
19	Package -1	68
20	Package -2	70

SESSION A(1)

Exp. NO: 01

Date: 07/10/2023

PALINDROME

Aim: To write a java program to check whether the input is palindrome or not.

Algorithm

Step 1: Start

Step 2: Declare string variables str and rev, then initialize rev = ""

Step 3: Read the string from the user and store it to the variable str

Step 4: Calculate the length of the string and store it in the variable len

Step 5: For the value of i = len - 1 to i >= 0 do a. rev = rev + str.charAt(i)

Step 6: If str and rev are equal a. Print the entered string is palindrome b.

Step 7: Else print the entered string is not palindrome

Step 8: Stop

Program Code

```
import java.util.Scanner;
public class String_palindrome
{
    public static void main(String[] args)
    {
        String str, rev = ""; System.out.print("Enter a string : ");
        Scanner s = new Scanner(System.in);
        str = s.nextLine(); int len = str.length();
        for (int i = len - 1; i >= 0; i--)
        {
            rev = rev + str.charAt(i);
        }
        if (str.equalsIgnoreCase(rev))
        {
            System.out.println(str + " is a palindrome");
        }
        else
        {
            System.out.println(str + " is not a palindrome");
        }
    }
}
```

```
}
```

Output

```
Enter a string : MALAYALAM  
MALAYALAM is a palindrome
```

```
Enter a string : hello  
hello is not a palindrome
```

Result

The algorithm was developed and the program was coded. The program was tested successfully.

Exp. NO: 02**Date: 07/10/2023**

FREQUENCY OF CHARACTER IN STRING

Aim: To write a Java Program to find the frequency of a given character in a string.

Algorithm

- Step 1: Start
- Step 2: Declare string variables str and str1
- Step 3: Declare integer variables i, len, count and initialize count as 0
- Step 4: Declare character variables ch and ch1
- Step 5: Read the string from the user and store it to str
- Step 6: Convert the string in str to lowercase and store in str1
- Step 7: Count the length of the string and store the value in len
- Step 8: Read the character to find the frequency from the user and store it to ch
- Step 9: Convert the character in ch to lowercase and store it in ch1
- Step 10: For the value of
 - i = 0 to i < len do
 - a. If(ch1 == str1.charAt(i))
 - i. Increase the value of count by 1.
- Step 11: Print the frequency of the character.
- Step 12: Stop

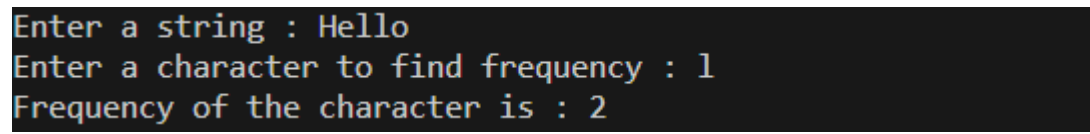
Program Code

```
import java.util.Scanner;

public class Test1
{
    public static void main(String[] args)
    {
        String str, str1;
        int i, len, count = 0;
        char ch, ch1;
        System.out.print("Enter a string : ");
        Scanner s = new Scanner(System.in);
        str = s.nextLine();
        str1 = str.toLowerCase();
        len = str.length();
        System.out.print("Enter a character to find frequency : ");
        Scanner c = new Scanner(System.in);
        ch = c.next().charAt(0);
        ch1 = Character.toLowerCase(ch);
```

```
    for (i = 0; i < len; i++)  
    {  
        if (ch1 == str1.charAt(i))  
        {  
            count++;  
        }  
    }  
    System.out.print("Frequency of the character is : " + count);  
}  
}
```

Output

A screenshot of a terminal window with a dark background. It shows three lines of text: "Enter a string : Hello", "Enter a character to find frequency : l", and "Frequency of the character is : 2".

```
Enter a string : Hello  
Enter a character to find frequency : l  
Frequency of the character is : 2
```

Result

The algorithm was developed and the program was coded. The program was tested successfully.

MATRIX MULTIPLICATION

Aim: To write a Java program to multiply two given matrices.

Algorithm

Step 1: Start

Step 2: Declare a class named Matrix with instance variables arr, m, and n.

- arr: a 2D array to store the matrix elements
- m: number of rows in the matrix
- n: number of columns in the matrix

Step 3: Define a parameterized constructor in the Matrix class to initialize the instance variables.

- Input: 2D array arr, and integers m and n

Step 4: Define a display() method in the Matrix class to display the matrix elements.

- Input: None
- Output: Display matrix elements row by row

Step 5: Declare a static method multiply() in the Matrix class to perform matrix multiplication.

- Input: Two Matrix objects m1 and m2
- Output: Return a new Matrix object representing the result of the multiplication

Step 6: In the multiply() method, initialize a 2D array 'arr' to store the result matrix.

Step 7: Use nested loops to perform matrix multiplication and store the result in the 'arr' array.

Step 8: Create a new Matrix object using the result array and return it.

Step 9: In the main method:

- a. Declare integers m, n, p, q representing the dimensions of the matrices.
- b. Declare 2D arrays 'a' and 'b' to store the matrix elements.
- c. Create Matrix objects m1 and m2 using the provided matrices and dimensions.
- d. Call the multiply() method to get the result matrix and store it in m3.
- e. Display the result using the display() method.

Step 10: Stop

Program Code

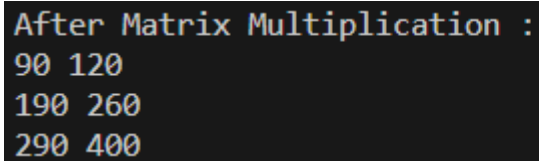
```
public class Matrix
{
    int[][] arr;    //instance variables
    int m,n;
```

OBJECT ORIENTED PROGRAMMING LAB

```
public Matrix(int[][] arr,int m,int n)    //Parameterised Constructor
{
    this.arr=arr;
    this.m=m;
    this.n=n;
}
public void display()                    //displaying after matrix multiplication
{
    int i,j;
    for(i=0;i<this.m;i++)
    {
        for (j=0;j<this.n;j++)
        {
            System.out.print(this.arr[i][j] + " ");
        }
        System.out.println();
    }
}
static Matrix multiply(Matrix m1,Matrix m2) //method having objects as arguments
{
    int i,j,k;
    int[][] arr=new int[m1.m][m2.n];    //accessing members
    for (i=0;i<m1.m;i++)
    {
        for (j=0;j<m2.n;j++)
        {
            arr[i][j]=0;
            for (k=0;k<m1.n;k++)
            {
                arr[i][j]+=m1.arr[i][k]*m2.arr[k][j];
            }
        }
    }
    return new Matrix(arr,m1.m,m2.n);
}
public static void main(String[] args)
{
    int m=3,n=2,p=2,q=2;    //User Input
    int[][] a={{1,2},
               {3,4},
               {5,6}};
    int[][] b={{10,20},
               {40,50}};
    Matrix m1=new Matrix(a,m,n);
    Matrix m2=new Matrix(b,p,q);
    Matrix m3=Matrix.multiply(m1,m2);
    System.out.println("After Matrix Multiplication : ");
}
```

```
m3.display();  
}  
}
```

Output



```
After Matrix Multiplication :  
90 120  
190 260  
290 400
```

Result

The algorithm was developed and the program was coded. The program was tested successfully.

SESSION A(2)

Exp. NO: 01

Date: 07/10/2023

PALINDROME

Aim: To write a java program to check whether the input is palindrome or not.

Algorithm

Step 1: Start

Step 2: Create a class named string1

- a. Declare a character array variable str as an instance variable
- b. Declare an integer variable i as an instance variable
- c. Define a method input(char a[]) to accept a character array as input

Step 3: In the constructor of string1 class, initialize str with the provided character array

Step 4: Define a method length() in string1 class

- a. Use a loop to find the length of the string (number of characters until '\0' is encountered)
- b. Return the length

Step 5: Define a method palindrome() in string1 class

- a. Initialize variables s to 0 and e to length - 1
- b. Use a loop to check if the string is a palindrome
 - i. If str[s] is not equal to str[e], return 0 (not a palindrome)
 - ii. Increment s and decrement e
- c. Return 1 (palindrome)

Step 6: In the main class Pgm1n

- a. Create a character array s with the string "MALAYALAM"
- b. Create an object s1 of the string1 class, passing the character array as an argument
- c. Call the length() method of s1 to get the length of the string
- d. Print the length of the string
- e. Check if the string is a palindrome using the palindrome() method of s1
- f. Print whether the string is a palindrome or not

Step 7: Stop

Program Code

```

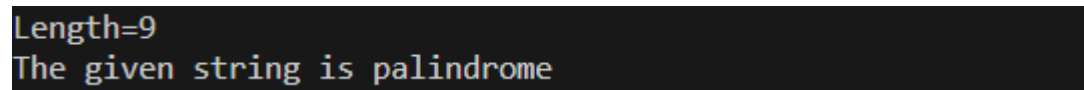
class string1
{
    char[] str;//instance variable
    int i;
    void input(char a[])
    {
        //a={'M','A','L','A','Y','L','A','M'};
    }
    string1(char a[])
    {
        str=a;
    }
    int length();//methods
    {
        for (i=0;str[i]!='\0';i++);
        return i;
    }
    int palindrome()
    {
        int s=0,e;
        e=i-1;
        while(e>s)
        {
            if (str[s]!=str[e])
            {
                return 0;
            }
            s=s+1;
            e=e-1;
        }
        return 1;
    }
}

public class Pgm1n
{
    public static void main(String[] args)
    {
        int len;
        char[] s={'M','A','L','A','Y','A','L','A','M','\0'};
        string1 s1;//defining an object
        s1=new string1(s);//instance creation
        len=s1.length();
        System.out.println("Length="+len);
    }
}

```

```
    if (s1.palindrome()==1)
    {
        System.out.println("The given string is palindrome");
    }
    else
    {
        System.out.println("The given string is NOT palindrome");
    }
}
```

Output



```
Length=9
The given string is palindrome
```

Result

The algorithm was developed and the program was coded. The program was tested successfully.

FREQUENCY OF CHARACTER IN STRING

Aim: To write a Java Program to find the frequency of a given character in a string.

Algorithm

Step 1: Start

Step 2: Declare a class named `string2` with instance variables `str`, `ch`, and `i`.

- `str`: a character array to store the string
- `ch`: a character to find the frequency of
- `i`: an integer variable

Step 3: Define a parameterized constructor in the `string2` class to initialize the instance variables.

- Input: a character array `a[]` and a character `c`

Step 4: Define a method named `length()` in the `string2` class to calculate the length of the string.

- Output: Return the length of the string

Step 5: In the `length()` method, use a loop to iterate through the characters in the string until the null character '\0' is encountered.

- Increment `i` in each iteration.

Step 6: Define a method named `frequency()` in the `string2` class to calculate the frequency of a given character `ch`.

- Output: Return the frequency of the character `ch` in the string.

Step 7: In the `frequency()` method, initialize variables `count` and `j` to 0.

- Use a while loop to iterate through the characters in the string until the null character '\0' is encountered.
- If the current character equals `ch`, increment `count`.
- Increment `j` in each iteration.

Step 8: Declare a class named `Pgm2`.

Step 9: In the `Pgm2` class:

- a. Declare a character array `s` to store the string.
- b. Create a `string2` object `s2` with the provided string `s` and a character `c`.
- c. Call the `length()` method on `s2` to calculate the length of the string.

- d. Call the `frequency()` method on `s2` to calculate the frequency of the character `c`.
- e. Print the calculated frequency.

Step 10: Stop

Program Code

```
class string2
{
    char[] str;
    char ch;
    int i;
    public string2(char a[],char c)    //Constructor
    {
        str=a;
        ch=c;
    }
    int length()
    {
        for (i=0;str[i]!='\0';i++);
        return i;
    }
    int frequency()
    {
        int count=0,j=0;
        while (str[j]!='\0')
        {
            if (str[j]==ch)
            {
                count=count+1;
            }
            j=j+1;
        }
        return count;
    }
}
public class Pgm2
{
    public static void main(String[] args)
    {
        char[] s={'M','A','L','A','Y','A','L','A','M','\0'};
        string2 s2;
        char c='A';        //object creation
        s2=new string2(s,c);    //instantion
        s2.length();
        int freq;
```

```
        freq=s2.frequency();  
        System.out.println("Frequency="+freq);  
    }  
}
```

Output

```
PS D:\My\RIT\S3\Object Oriented Programming using Java\LAB> javac Pgm2.java  
PS D:\My\RIT\S3\Object Oriented Programming using Java\LAB> java Pgm2  
Frequency=4
```

Result

The algorithm was developed and the program was coded. The program was tested successfully.

MATRIX MULTIPLICATION

Aim: To write a Java program to multiply two given matrices.

Algorithm

Step 1: Start

Step 2: Declare a class named Matrix with instance variables arr, m, and n.

1. arr: a 2D array to store the matrix elements
2. m: number of rows in the matrix
3. n: number of columns in the matrix

Step 3: Define a parameterized constructor in the Matrix class to initialize the instance variables.

-Input: 2D array arr, and integers m and n

Step 4: Define a display() method in the Matrix class to display the matrix elements.

-Input: None

-Output: Display matrix elements row by row

Step 5: Declare a static method multiply() in the Matrix class to perform matrix multiplication.

- Input: Two Matrix objects m1 and m2

- Output: Return a new Matrix object representing the result of the multiplication

Step 6: In the multiply() method, initialize a 2D array 'arr' to store the result matrix.

Step 7: Use nested loops to perform matrix multiplication and store the result in the 'arr' array.

Step 8: Create a new Matrix object using the result array and return it.

Step 9: In the main class Pgm3:

- a) Declare variables m, n, p, q, representing the dimensions of the matrices.
- b) Declare 2D arrays 'a' and 'b' to store the matrix elements.
- c) Create Matrix objects m1 and m2 using the provided matrices and dimensions.
- d) Call the multiply() method to get the result matrix and store it in m3.
- e) Display the result using the display() method.

Step 10: Stop

Program Code

```

class Matrix
{
    int[][] arr;    //instance variables
    int m,n;
    public Matrix(int[][] arr,int m,int n)    //Parameterised Constructor
    {
        this.arr=arr;
        this.m=m;
        this.n=n;
    }
    public void display()    //displaying after matrix multiplication
    {
        int i,j;
        for(i=0;i<this.m;i++)
        {
            for (j=0;j<this.n;j++)
            {
                System.out.print(this.arr[i][j] + " ");
            }
            System.out.println();
        }
    }
    static Matrix multiply(Matrix m1,Matrix m2) //method having objects as arguments
    {
        int i,j,k;
        int[][] arr=new int[m1.m][m2.n];    //accessing members
        for (i=0;i<m1.m;i++)
        {
            for (j=0;j<m2.n;j++)
            {
                arr[i][j]=0;
                for (k=0;k<m1.n;k++)
                {
                    arr[i][j]+=m1.arr[i][k]*m2.arr[k][j];
                }
            }
        }
        return new Matrix(arr,m1.m,m2.n);
    }
}

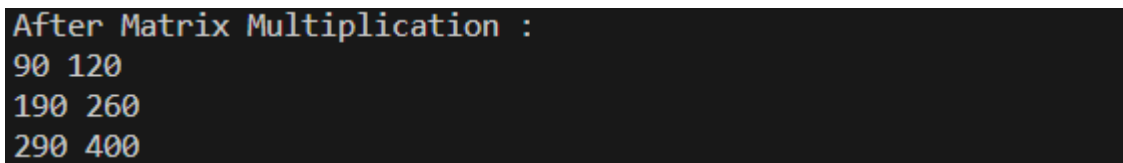
public class Pgm3
{
    public static void main(String[] args)

```

OBJECT ORIENTED PROGRAMMING LAB

```
{
    int m=3,n=2,p=2,q=2;    //User Input
    int[][] a={{1,2},
               {3,4},
               {5,6}};
    int[][] b={{10,20},
               {40,50}};
    //Matrix m1,m2,m3;      //defining object
    //m1=new Matrix();      //instanation
    Matrix m1=new Matrix(a,m,n);
    Matrix m2=new Matrix(b,p,q);
    Matrix m3=Matrix.multiply(m1,m2);
    System.out.println("After Matrix Multiplication : ");
    m3.display();
}
}
```

Output

A screenshot of a terminal window showing the output of the matrix multiplication program. The text is as follows:

```
After Matrix Multiplication :
90 120
190 260
290 400
```

Result

The algorithm was developed and the program was coded. The program was tested successfully.

SESSION B

Exp. NO: 04

Date: 16/10/2023

INHERITANCE

Aim: To write a Java program which creates a class named 'Employee' having the following members: Name, Age, Phone number, Address, Salary. It also has a method named 'printSalary()' which prints the salary of the Employee. Two classes 'Officer' and 'Manager' inherits the 'Employee' class. The 'Officer' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an officer and a manager by making an object of both of these classes and print the same.

Algorithm

- Step 1: Start
- Step 2: Creating employee class
- Step 3: Creating constructor for employee class
- Step 4: Method to print salary
- Step 5: Officer class inherits from employee
- Step 6: Constructor for officer class
- Step 7: Manager class inherits from employee
- Step 8: Constructor for manager class
- Step 9: Create main class
- Step 10: Get input for officer
- Step 11: Create officer object
- Step 12: Get input for manager
- Step 13: Create manager object
- Step 14: Print details and salary for officer
- Step 15: Print details and salary for manager
- Step 16: Close the scanner
- Step 17: Stop

Program Code

```
import java.util.Scanner;

class Employee
{
    String name;
    int age;
    String phoneNumber;
    String address;
    double salary;

    public Employee(String name, int age, String phoneNumber, String address, double salary)
    {
        this.name = name;
        this.age = age;
        this.phoneNumber = phoneNumber;
        this.address = address;
        this.salary = salary;
    }

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

class Officer extends Employee
{
    String specialization;

    public Officer(String name, int age, String phoneNumber, String address, double salary,
String specialization)
    {
        super(name, age, phoneNumber, address, salary);
        this.specialization = specialization;
    }
}

class Manager extends Employee
{
    String department;

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



```

    }
}

public class Main
{
    public static void main(String[] args)
    {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter Officer details:");
        System.out.print("Name: ");
        String officerName = scanner.nextLine();
        System.out.print("Age: ");
        int officerAge = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        System.out.print("Phone Number: ");
        String officerPhoneNumber = scanner.nextLine();
        System.out.print("Address: ");
        String officerAddress = scanner.nextLine();
        System.out.print("Salary: ");
        double officerSalary = scanner.nextDouble();
        scanner.nextLine(); // Consume newline
        System.out.print("Specialization: ");
        String officerSpecialization = scanner.nextLine();
        Officer officer = new Officer(officerName, officerAge, officerPhoneNumber,
        officerAddress, officerSalary, officerSpecialization);
        System.out.println("\nEnter Manager details:");
        System.out.print("Name: ");
        String managerName = scanner.nextLine();
        System.out.print("Age: ");
        int managerAge = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        System.out.print("Phone Number: ");
        String managerPhoneNumber = scanner.nextLine();
        System.out.print("Address: ");
        String managerAddress = scanner.nextLine();
        System.out.print("Salary: ");
        double managerSalary = scanner.nextDouble();
        scanner.nextLine(); // Consume newline
        System.out.print("Department: ");
        String managerDepartment = scanner.nextLine();
        Manager manager = new Manager(managerName, managerAge,
        managerPhoneNumber, managerAddress, managerSalary, managerDepartment);
        System.out.println("\nOfficer Details:");
        System.out.println("Name: " + officer.name);
        System.out.println("Age: " + officer.age);
        System.out.println("Phone Number: " + officer.phoneNumber);
        System.out.println("Address: " + officer.address);
    }
}

```

```
        officer.printSalary();
        System.out.println("Specialization: " + officer.specialization);
        System.out.println("\nManager Details:");
        System.out.println("Name: " + manager.name);
        System.out.println("Age: " + manager.age);
        System.out.println("Phone Number: " + manager.phoneNumber);
        System.out.println("Address: " + manager.address);
        manager.printSalary();
        System.out.println("Department: " + manager.department);
        scanner.close();
    }
}
```

Output

```
Enter Officer details:
Name: Mathew John
Age: 22
Phone Number: 9966442321
Address: New Residencies
Salary: 70000
Specialization: Programming
```

```
Enter Manager details:
Name: Rahul M
Age: 35
Phone Number: 8756889032
Address: New Villa
Salary: 90000
Department: Finance
```

```
Officer Details:
Name: Mathew John
Age: 22
Phone Number: 9966442321
Address: New Residencies
Salary: 70000.0
Specialization: Programming
```

```
Manager Details:
Name: Rahul M
Age: 35
Phone Number: 8756889032
Address: New Villa
Salary: 90000.0
Department: Finance
```

Result

The algorithm was developed and the program was coded. The program was tested successfully.

Exp. NO: 05**Date: 16/10/2023**

POLYMORPHISM

Aim: To write a java program to create an abstract class named Shape that contains an empty method named numberOfSides(). Provide three classes named Rectangle, Triangle and Hexagon such that each one of the classes extends the class Shape. Each one of the classes contains only the method numberOfSides() that shows the number of sides in the given geometrical structures.

Algorithm

Step 1: Start

Step 2: Declare an abstract class named Shape.

- a. Declare an abstract method named numberOfSides().

Step 3: Create a class named Rectangle that extends Shape.

- a. Declare a private variable sides with a value of 4.
- b. Implement the numberOfSides method by returning the value of sides.
- c. Define a method named display that prints "Sides of a Rectangle: sides".

Step 4: Create a class named Triangle that extends Shape.

- a. Declare a private variable sides with a value of 3.
- b. Implement the numberOfSides method by returning the value of sides.
- c. Define a method named display that prints "Sides of a Triangle: sides".

Step 5: Create a class named Hexagon that extends Shape.

- a. Declare a private variable sides with a value of 6.
- b. Implement the numberOfSides method by returning the value of sides.
- c. Define a method named display that prints "Sides of a Hexagon: sides".

Step 6: Create a class named Pgm5.

- a. Define the main method:
 - i. Create objects r (Rectangle), t (Triangle), and h (Hexagon).
 - ii. Call the display method for each object.

Step 7: Stop

Program Code

```
abstract class Shape
{
    abstract int numberOfSides();
}
class Rectangle extends Shape
{
    private int sides=4;
    int numberOfSides()
    {
        return sides;
    }
    void display()
    {
        System.out.println("Sides of a Rectangle : "+sides);
    }
}
class Triangle extends Shape
{
    private int sides=3;
    int numberOfSides()
    {
        return sides;
    }
    void display()
    {
        System.out.println("Sides of a Triangle : "+sides);
    }
}
class Hexagon extends Shape
{
    private int sides=6;
    int numberOfSides()
    {
        return sides;
    }

    void display()
    {
        System.out.println("Sides of a Hexagon : "+sides);
    }
}

public class Pgm5
{
    public static void main(String[] args)
```

```
{  
    Rectangle r=new Rectangle();  
    Triangle t=new Triangle();  
    Hexagon h=new Hexagon();  
    r.display();  
    t.display();  
    h.display();  
}
```

Output

```
Sides of a Rectangle : 4  
Sides of a Triangle : 3  
Sides of a Hexagon : 6
```

Result

The algorithm was developed and the program was coded. The program was tested successfully.

SESSION C

Exp. NO: 07

Date: 18/11/2023

FILE HANDLING USING READER/WRITER

Aim: To write a file handling program in Java with reader/writer.

Algorithm

Step 1: Start

Step 2: Main function is invoked

Step 3: In the try function do the following

- a. Accept the text from the user as str
- b. Create an object file for filewriter
- c. Write the string to the file
- d. Close the file

Step 4: If an exception occurred catch the exception

Step 5: In another try function do the following

- a. Declare a character array ch
- b. An object rd is created for filereader
- c. Print the contents in the file
- d. Close the file

Step 6: If an exception occurred catch the exception Step 7: Stop

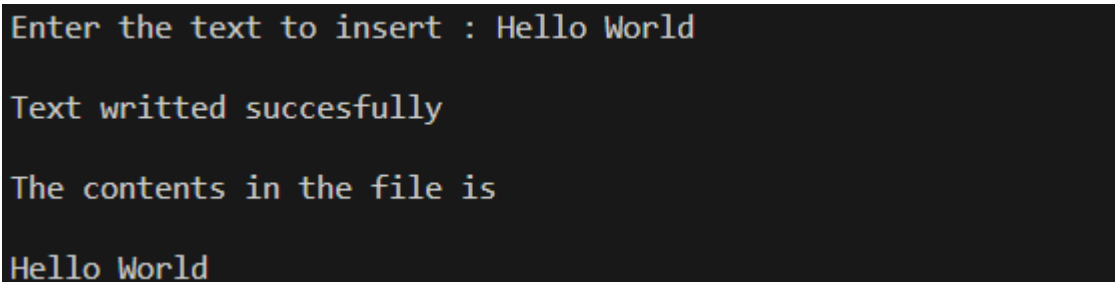
Program Code

```
import java.util.Scanner;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.FileNotFoundException;

public class Main
{
    public static void main(String args[])
    {
        try
        {
            System.out.print("Enter the text to insert : ");
            Scanner s = new Scanner(System.in);
            String str = s.nextLine();
            FileWriter file = new FileWriter("sample.txt");
            file.write(str);
```

```
        file.close();
        System.out.print("\nText writted succesfully");
    }
    catch (Exception e)
    {
        System.out.print("\nError occurred");
        e.printStackTrace();
    }
    try
    {
        char ch[] = new char[100];
        FileReader rd = new FileReader("sample.txt");
        rd.read(ch);
        System.out.println("\n\nThe contents in the file is \n");
        System.out.println(ch);
        rd.close();
    } catch (Exception e)
    {
        System.out.print("\n Error occurred");
        e.printStackTrace();
    }
}
```

Output

A screenshot of a terminal window with a black background and light blue/green text. It shows the output of a Java program: 'Enter the text to insert : Hello World', 'Text writted succesfully', 'The contents in the file is', and 'Hello World' on separate lines.

```
Enter the text to insert : Hello World
Text writted succesfully
The contents in the file is
Hello World
```

Result

The algorithm was developed and the program was coded. The program was tested successfully

Exp. NO: 08

Date: 18/11/2023

HANDLING FILE EXCEPTIONS

Aim: To Write a Java program that read from a file and write to file by handling all file related exceptions

Algorithm

Step 1: Start

Step 2: Main function is invoked

1. In the try function do the following
 - a. Accept file name from the user
 - b. Create a new file with the file name
 - c. If successfully created
 - a. Print file created with file name
 - d. Else
 - a. Print file already exists
2. If an exception occurred catch the exception 3. In another try function do the following
 - a. Accept the text from the user as str
 - b. Create an object file for filewriter
 - c. Write the string to the file
 - d. Close the file
 - e. If an exception occurred catch the exception
3. In another try function do the following
 - a. Declare a character array ch
 - b. An object rd is created for filereader
 - c. Print the contents in the file
 - d. Close the file

Step 3: If an exception occurred catch the exception

Step 4: Stop

Program Code

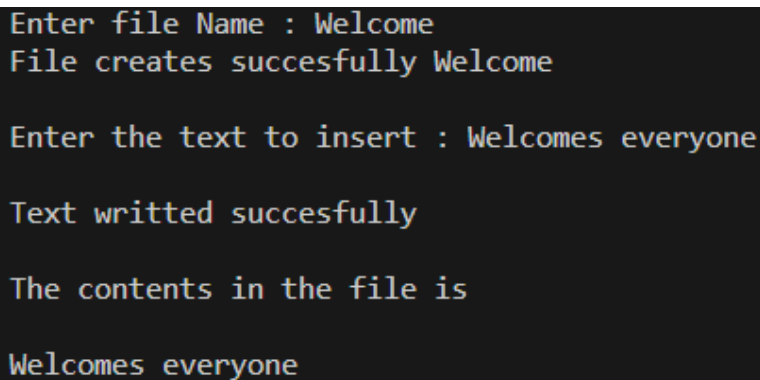
```
import java.util.Scanner;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.FileNotFoundException;

public class Main
{
    public static void main(String args[])
    {
        Scanner s = new Scanner(System.in);
        try
        {
            System.out.print("Enter file Name : ");
            String filename = s.nextLine();
            File create = new File(filename);
            if (create.createNewFile())
            {
                System.out.println("File creates succesfully " + create.getName());
            } else
            {
                System.out.println("File already Exist");
            }
        }
        catch (Exception e)
        {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
        try
        {
            System.out.print("\nEnter the text to insert : ");
            String str = s.nextLine();
            FileWriter file = new FileWriter("filename");
            file.write(str);
            file.close();
            System.out.print("\nText writted succesfully");
        }
        catch (Exception e)
        {
            System.out.print("\nError occurred");
            e.printStackTrace();
        }
        try
        {

```

```
char ch[] = new char[100];
FileReader rd = new FileReader("filename");
rd.read(ch);
System.out.println("\n\nThe contents in the file is \n");
System.out.println(ch);
rd.close();
}
catch (Exception e)
{
    System.out.print("\nError occurred");
    e.printStackTrace();
}
}
```

Output

A screenshot of a terminal window with a dark background and light-colored text. The output shows the program's execution steps: entering a file name, successful file creation, entering text to insert, successful text writing, and finally displaying the file's contents.

```
Enter file Name : Welcome
File creates succesfully Welcome

Enter the text to insert : Welcomes everyone

Text writted succesfully

The contents in the file is

Welcomes everyone
```

Result

The algorithm was developed and the program was coded. The program was tested successfully.

Exp. NO: 09**Date: 18/11/2023**

STRING TOKENIZER

Aim: Write a Java program that reads a line of integers, and then displays each integer, and the sum of all the integers

Algorithm

Step 1: Start

Step 2: Import the necessary Java utilities: Scanner and StringTokenizer.

Step 3: Declare a public class named Pgm9.

Step 4: Define the main method:

- a. Create a Scanner object named 'reader' to read input from the user.
- b. Declare variables n and sum to store integers and their sum.
- c. Print "Enter the integers : ".
- d. Read a line of input as a string using 'reader.nextLine()' and store it in the variables.
- e. Create a StringTokenizer 'st' with the input string 'S' and using "," as the delimiter.
- f. Use a while loop with 'hasMoreTokens()' to iterate through each token:
 - i. Parse the token to an integer and store it in the variable n.
 - ii. Print the integer value.
 - iii. Add the integer value to the sum.
 - iv. Move to the next token.
 - v. Print "Sum Of Integers : " followed by the value of 'sum'.
 - vi. Close the Scanner.

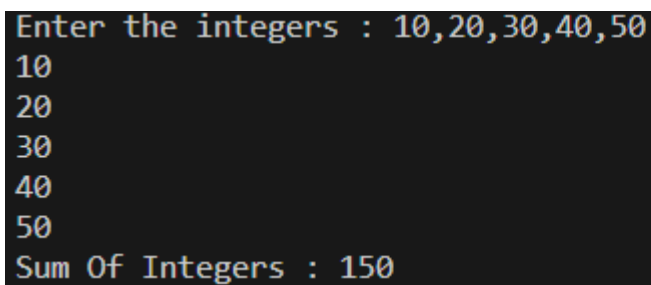
Step 5: Stop

Program Code

```
import java.util.Scanner;
import java.util.StringTokenizer;
public class Pgm9
{
    public static void main(String args[])
    {
        Scanner reader=new Scanner(System.in);
        int n,sum=0;
        System.out.print("Enter the integers : ");
        String S=reader.nextLine();
```

```
StringTokenizer st=new StringTokenizer(S,"");
while(st.hasMoreTokens())
{
    n=Integer.parseInt(st.nextToken());
    System.out.println(n);
    sum+=n;
}
System.out.println("Sum Of Integers : "+sum);
reader.close();
}
```

Output

A screenshot of a terminal window with a black background and light blue/green text. It shows the input sequence '10,20,30,40,50' and the resulting sum '150'.

```
Enter the integers : 10,20,30,40,50
10
20
30
40
50
Sum Of Integers : 150
```

Result

The algorithm was developed and the program was coded. The program was tested successfully.

SESSION D

Exp. NO: 10

Date: 18/11/2023

EXCEPTION HANDLING

Aim: Write a Java program that shows the usage of try, catch, throws and finally.

Algorithm

Algorithm: Pgm10

Step 1: Start

Step 2: Declare a class EvenException that extends Exception with the method toString() returning "Even Number"

Step 3: Declare a class OddException that extends Exception with the method toString() returning "Odd Number"

Step 4: Declare a class Pgm10

Step 5: Inside Pgm10, define the method checknum(int n) throws EvenException, OddException

- If n is even, throw an EvenException
- If n is odd, throw an OddException

Step 6: Inside Pgm10, define the method main(String[] args) throws IOException

- Declare an integer variable num
- Create a Scanner object reader to read input from the user
- Print "Enter a number : "
- Read an integer from the user and store it in the variable num
- Try the following:
 - Call Pgm10.checknum(num)
 - If an EvenException is caught, print "Exception : Even Number"
 - If an OddException is caught, print "Exception : Odd Number"
 - If a general Exception is caught, print "Exception!!"

Step 7: Stop

Program Code

```
import java.util.Scanner;
import java.io.*;
class EvenException extends Exception
{
    public String toString()
    {
        return "Even Number";
    }
}

class OddException extends Exception
{
    public String toString()
    {
        return "Odd Number";
    }
}


public class Pgm10
{
    public static void checknum(int n) throws EvenException,OddException
    {
        if (n%2==0)
        {
            throw new EvenException();
        }
        else
        {
            throw new OddException();
        }
    }

    public static void main(String[] args) throws IOException
    {
        int num;
        Scanner reader=new Scanner(System.in);
        System.out.print("Enter a number : ");
        num=reader.nextInt();
        try
        {
            Pgm10.checknum(num);
        }
        catch(EvenException even)
        {
            System.out.println("Exception : "+even);
        }
        catch(OddException odd)
```


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

}
}
```

Output



```
Enter a number : 33
Exception : Odd Number
```



```
Enter a number : 100
Exception : Even Number
```

Result

The algorithm was developed and the program was coded. The program was tested successfully.

THREAD GENERATION

Aim: Write a Java program that implements a multi-threaded program which has three threads. First thread generates a random integer every 1 second. 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.

Algorithm

Step 1: Start

Step 2: Declare a class named Data with integer variable num and three boolean variables nstatus, sstatus, and cstatus, initialized to false.

Step 3: Declare three synchronized methods in the Data class:

- a. ndisplay(int num):
 - i. Wait until nstatus is false.
 - ii. Set the value of num.
 - iii. Print "Number Generated: num".
 - iv. Set nstatus to true and sstatus, cstatus to false.
 - v. Notify other waiting threads.
- b. sdisplay(int num):
 - i. Wait until sstatus is false.
 - ii. If num is even, print "Square: num * num".
 - iii. Set sstatus to true and nstatus to false.
 - iv. Notify other waiting threads.
- c. cdisplay(int num):
 - i. Wait until cstatus is false.
 - ii. If num is odd, print "Cube: num * num * num".
 - iii. Set cstatus to true and nstatus to false.
 - iv. Notify other waiting threads.

Step 4: Declare a class named NumGenerator that extends Thread:

- a. Declare variables obj (of type Data), n (of type Random), and num.
- b. Define a constructor that takes a Data object as a parameter and assigns it to obj.
- c. Override the run method:
 - i. Initialize n as a new Random object.
 - ii. Enter an infinite loop:

1. Generate a random number (num) using n.
2. Call obj.ndisplay(num).
3. Sleep for 3000 milliseconds.

Step 5: Declare a class named Square that extends Thread:

- a. Declare a variable obj (of type Data).
- b. Define a constructor that takes a Data object as a parameter and assigns it to obj.
- c. Override the run method:
 - i. Enter an infinite loop:
 1. Call obj.sdisplay(obj.num).
 2. Sleep for 3000 milliseconds.

Step 6: Declare a class named Cube that extends Thread:

- a. Declare a variable obj (of type Data).
- b. Define a constructor that takes a Data object as a parameter and assigns it to obj.
- c. Override the run method:
 - i. Enter an infinite loop:
 1. Call obj.cdisplay(obj.num).
 2. Sleep for 3000 milliseconds.

Step 7: Declare a class named ThreadGenerator.

- a. Define the main method:
 - i. Create an object obj of type Data.
 - ii. Create objects number (of type NumGenerator), sq (of type Square), and cb (of type Cube) with obj as a parameter.
 - iii. Start the threads number, sq, and cb.

Step 8: Stop

Program Code

```
import java.util.Random;

class Data
{
    int num;
    boolean nstatus=false;
    boolean sstatus=false;
    boolean cstatus=false;
    synchronized void ndisplay(int num)
    {
        while(nstatus)
        {
            try
            {
                wait();
            }
        }
    }
}
```

```

        catch(InterruptedException ex)
        {
            System.out.println("Exception arises : "+ex);
        }
    }
    this.num=num;
    System.out.println("Number Generated : "+num);
    nstatus=true;
    sstatus=false;
    cstatus=false;
    notify();
}
synchronized void sdisplay(int num)
{
    while(sstatus)
    {
        try
        {
            wait();
        }
        catch(InterruptedException ex)
        {
            System.out.println("Exception arises : "+ex);
        }
    }
    if(num%2==0)
    {
        System.out.println("Square : "+(num*num));
        sstatus=true;
        nstatus=false;
        notify();
    }
}
synchronized void cdisplay(int num)
{
    while(cstatus)
    {
        try
        {
            wait();
        }
        catch(InterruptedException ex)
        {
            System.out.println("Exception arises : "+ex);
        }
    }
}

```

```
        if(num%2!=0)
        {
            System.out.println("Cube : "+(num*num*num));
            cstatus=true;
            nstatus=false;
            notify();
        }
    }
}
```

```
class NumGenerator extends Thread
{
    Data obj;
    Random n;
    int num;
    public NumGenerator(Data obj)
    {
        this.obj=obj;
    }
    public void run()
    {
        n=new Random();
        while(true)
        {
            num=n.nextInt(100);
            obj.ndisplay(num);
            try
            {
                Thread.sleep(3000);
            }
            catch(InterruptedException ex)
            {
                System.out.println("Exception arises : "+ex);
            }
        }
    }
}
```

```
class Square extends Thread
{
    Data obj;
    public Square(Data obj)
    {
        this.obj=obj;
    }
}
```

OBJECT ORIENTED PROGRAMMING LAB

```
public void run()
{
    while(true)
    {
        obj.sdisplay(obj.num);
        try
        {
            Thread.sleep(3000);
        }
        catch(InterruptedException ex)
        {
            System.out.println("Exception arises : "+ex);
        }
    }
}
```


```
class Cube extends Thread
{
    Data obj;
    public Cube(Data obj)
    {
        this.obj=obj;
    }
    public void run()
    {
        while(true)
        {
            obj.cdisplay(obj.num);
            try
            {
                Thread.sleep(3000);
            }
            catch(InterruptedException ex)
            {
                System.out.println("Exception arises : "+ex);
            }
        }
    }
}
```

```
public class ThreadGenerator
{
    public static void main(String args[])
    {
        Data obj=new Data();
```

```
NumGenerator number=new NumGenerator(obj);
Square sq=new Square(obj);
Cube cb=new Cube(obj);

number.start();
sq.start();
cb.start();
    }
}
```

Output



```
Number Generated : 48
Square : 2304
Number Generated : 21
Cube : 9261
Number Generated : 89
Cube : 704969
Number Generated : 8
Square : 64
Number Generated : 12
Square : 144
Number Generated : 19
Cube : 6859
```

Result

The algorithm was developed and the program was coded. The program was tested successfully.

THREAD SYNCHRONIZATION

Aim: Write a Java program that shows thread synchronization.

Algorithm

Step 1: Start

Step 2: Declare a class named Data with an integer variable num and two boolean variables estatus and ostatus, both initialized to false.

Step 3: Declare two synchronized methods in the Data class:

- a. edisplay(int num):
 - i. Wait until estatus is false.
 - ii. Print "Even Number: num".
 - iii. Set estatus to true and ostatus to false.
 - iv. Notify other waiting threads.
- b. odisplay(int num):
 - i. Wait until ostatus is false.
 - ii. Print "Odd Number: num".
 - iii. Set ostatus to true and estatus to false.
 - iv. Notify other waiting threads.

Step 4: Declare a class named EvenNumber that implements the Runnable interface:

- a. Declare a variable obj (of type Data).
- b. Define a constructor that takes a Data object as a parameter and assigns it to obj.
- c. Implement the run method:
 - i. Run a loop from 0 to 500000 with a step of 2:
 - i. Call obj.edisplay(i).
 - ii. Sleep for 3000 milliseconds.

Step 5: Declare a class named OddNumber that implements the Runnable interface:

- a. Declare a variable obj (of type Data).
- b. Define a constructor that takes a Data object as a parameter and assigns it to obj.
- c. Implement the run method:
 - i. Run a loop from 1 to 500000 with a step of 2:
 - i. Call obj.odisplay(i).
 - ii. Sleep for 3000 milliseconds.

Step 6: Declare a class named ThreadSynch.

- a. Define the main method:
 - ii. Create an object obj of type Data.

- iii. Create objects even (of type EvenNumber) and odd (of type OddNumber) with obj as a parameter.
- iv. Create threads t1 and t2 with even and odd as the Runnable targets.
- v. Start the threads t1 and t2.

Step 7: Stop

Program Code

```
class Data
{
    int num;
    boolean estatus=false;
    boolean ostatus=false;
    synchronized void edisplay(int num)
    {
        while(estatus)
        {
            try
            {
                wait();
            }
            catch(InterruptedException ex)
            {
                System.out.println("Exception Arrises : "+ex);
            }
        }
        System.out.println("Even Number : "+num);
        estatus=true;
        ostatus=false;
        notify();
    }
    synchronized void odisplay(int num)
    {
        while(ostatus)
        {
            try
            {
                wait();
            }
            catch(InterruptedException ex)
            {
                System.out.println("Exception Arrises : "+ex);
            }
        }
        System.out.println("Odd Number : "+num);
    }
}
```


OBJECT ORIENTED PROGRAMMING LAB

```
        ostatus=true;
        estatus=false;
        notify();
    }
}

class EvenNumber implements Runnable
{
    Data obj;
    public EvenNumber(Data obj)
    {
        this.obj=obj;
    }
    public void run()
    {
        for(int i=0;i<500000;i=i+2)
        {
            obj.edisplay(i);

            try
            {
                Thread.sleep(3000);
            }
            catch(InterruptedException ex)
            {
                System.out.println("Exception Arrises : "+ex);
            }
        }
    }
}
```

```
class OddNumber implements Runnable
{
    Data obj;
    public OddNumber(Data obj)
    {
        this.obj=obj;
    }
    public void run()
    {
        for(int i=1;i<500000;i=i+2)
        {
            obj.odisplay(i);

            try
            {
```

```
        Thread.sleep(3000);
    }
    catch(InterruptedException ex)
    {
        System.out.println("Exception Arrises : "+ex);
    }
}
}

public class ThreadSynch
{
    public static void main(String args[])
    {
        Data obj=new Data();

        EvenNumber even=new EvenNumber(obj);
        OddNumber odd=new OddNumber(obj);

        Thread t1=new Thread(even,"Even Number");
        Thread t2=new Thread(odd,"Odd Number");

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

Output

```
Even Number : 0
Odd Number : 1
Even Number : 2
Odd Number : 3
Even Number : 4
Odd Number : 5
Even Number : 6
Odd Number : 7
Even Number : 8
Odd Number : 9
Even Number : 10
Odd Number : 11
Even Number : 12
Odd Number : 13
Even Number : 14
Odd Number : 15
Even Number : 16
Odd Number : 17
Even Number : 18
Odd Number : 19
Even Number : 20
Odd Number : 21
Even Number : 22
Odd Number : 23
Even Number : 24
Odd Number : 25
Even Number : 26
```

Result

The algorithm was developed and the program was coded. The program was tested successfully.

SESSION E

Exp. NO: 13

Date: 20/12/2023

CALCULATOR

Aim: Write a Java program that works as a simple calculator. Arrange Buttons for digits and the + - * % operations properly. Add a text field to display the result. Handle any possible exceptions like divide by zero. Use Java Swing.

Algorithm

Step 1: Start

Step 2: Declare a class named calc implementing ActionListener.

Step 3: Declare instance variables: JFrame f, JTextField t, JButton objects b1, b2, ..., bdo, be, bc, double variables a, b, and r, and int variable op.

Step 4: Define a constructor for calc:

- a. Initialize JFrame f and set its properties.
- b. Initialize JTextField t.
- c. Initialize JButton objects b1, b2, ..., bdo, be, bc with labels.
- d. Set the layout of f to null.
- e. Set the bounds for t and add it to f.
- f. Set the bounds for each button and add them to f.
- g. Attach ActionListener (this) to each button.

Step 5: Implement the actionPerformed method:

- a. Check the source of the event using e.getSource() and perform corresponding actions.
- b. If a number button is clicked, append the button's label to the text field.
- c. If the decimal button is clicked, append a dot to the text field.
- d. If an operator button (+, -, *, /) is clicked, parse the current text to double, set op, and clear the text field.
- e. If the equals button is clicked, parse the current text to double, perform the operation based on op, and display the result.
- f. If the clear button is clicked, clear the text field.

Step 6: Declare a class named calcT.

- a. Define the main method:
 - i. Create an object c of type calc.

Step 7: Stop

Program Code

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
class calc implements ActionListener
{
    JFrame f;
    JButton b1,b2,b3,b4,b5,b6,b7,b8,b9,b0,ba,bs,bm,bdi,bdo,be,bc;
    JTextField t;
    double a,b,r;
    int op=0;
    calc()
    {
        f=new JFrame("Calculator");
        t=new JTextField();
        b1=new JButton("1");
        b2=new JButton("2");
        b3=new JButton("3");
        b4=new JButton("4");
        b5=new JButton("5");
        b6=new JButton("6");
        b7=new JButton("7");
        b8=new JButton("8");
        b9=new JButton("9");
        b0=new JButton("0");
        ba=new JButton("+");
        bs=new JButton("-");
        bm=new JButton("x");
        bdi=new JButton("/");
        be=new JButton("=");
        bc=new JButton("clear");
        bdo=new JButton(".");
        f.setLayout(null);
        f.setVisible(true);
        f.setBounds(50,50,350,500);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        t.setBounds(30,50,275,20);
        f.add(t);
        b7.setBounds(40,100,50,40);
        f.add(b7);
        b8.setBounds(110,100,50,40);
        f.add(b8);
```

```
b9.setBounds(180,100,50,40);
f.add(b9);
bdi.setBounds(250,100,50,40);
f.add(bdi);
b4.setBounds(40,170,50,40);
f.add(b4);
b5.setBounds(110,170,50,40);
f.add(b5);
b6.setBounds(180,170,50,40);
f.add(b6);
bm.setBounds(250,170,50,40);
f.add(bm);
b1.setBounds(40,240,50,40);
f.add(b1);
b2.setBounds(110,240,50,40);
f.add(b2);
b3.setBounds(180,240,50,40);
f.add(b3);
bs.setBounds(250,240,50,40);
f.add(bs);
b0.setBounds(40,310,50,40);
f.add(b0);
bdo.setBounds(110,310,50,40);
f.add(bdo);
be.setBounds(180,310,50,40);
f.add(be);
ba.setBounds(250,310,50,40);
f.add(ba);
bc.setBounds(100,380,100,40);
f.add(bc);
b1.addActionListener(this);
b2.addActionListener(this);
b3.addActionListener(this);
b4.addActionListener(this);
b5.addActionListener(this);
b6.addActionListener(this);
b7.addActionListener(this);
b8.addActionListener(this);
b9.addActionListener(this);
b0.addActionListener(this);
ba.addActionListener(this);
bs.addActionListener(this);
bm.addActionListener(this);
bdi.addActionListener(this);
bdo.addActionListener(this);
be.addActionListener(this);
bc.addActionListener(this);
```

```

}
public void actionPerformed(ActionEvent e)
{
    if(e.getSource()==b1)
    {
        t.setText(t.getText().concat("1"));
    }
    if(e.getSource()==b2)
    {
        t.setText(t.getText().concat("2"));
    }
    if(e.getSource()==b3)
    {
        t.setText(t.getText().concat("3"));
    }
    if(e.getSource()==b4)
    {
        t.setText(t.getText().concat("4"));
    }
    if(e.getSource()==b5)
    {
        t.setText(t.getText().concat("5"));
    }
    if(e.getSource()==b6)
    {
        t.setText(t.getText().concat("6"));
    }
    if(e.getSource()==b7)
    {
        t.setText(t.getText().concat("7"));
    }
    if(e.getSource()==b8)
    {
        t.setText(t.getText().concat("8"));
    }
    if(e.getSource()==b9)
    {
        t.setText(t.getText().concat("9"));
    }
    if(e.getSource()==b0)
    {
        t.setText(t.getText().concat("0"));
    }
    if(e.getSource()==bdo)
    {
        t.setText(t.getText().concat("."));
    }
}

```

```

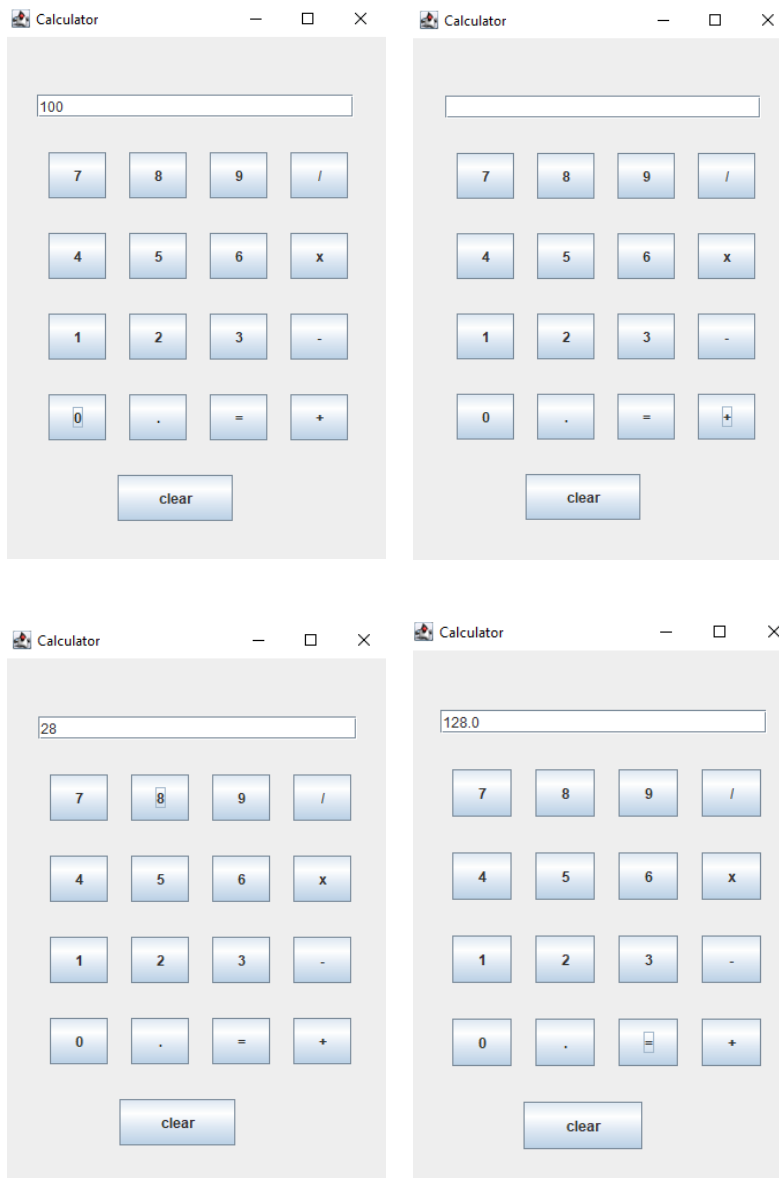
    }
    if(e.getSource()==ba)
    {
        a=Double.parseDouble(t.getText());
        op=1;
        t.setText(null);
    }
    if(e.getSource()==bs)
    {
        a=Double.parseDouble(t.getText());
        op=2;
        t.setText(null);
    }
    if(e.getSource()==bm)
    {
        a=Double.parseDouble(t.getText());
        op=3;
        t.setText(null);
    }
    if(e.getSource()==bdi)
    {
        a=Double.parseDouble(t.getText());
        op=4;
        t.setText(null);
    }
    if(e.getSource()==be)
    {
        b=Double.parseDouble(t.getText());
        switch(op)
        {
            case 1:r=a+b;
                break;
            case 2:r=a-b;
                break;
            case 3:r=a*b;
                break;
            case 4:r=a/b;
                break;
        }
        t.setText(""+r);
    }
    if(e.getSource()==bc)
    {
        t.setText(null);
    }
}
}

```


OBJECT ORIENTED PROGRAMMING LAB

```
public class calcT
{
    public static void main(String[] args)
    {
        calc c=new calc();
    }
}
```

Output



Result

The algorithm was developed and the program was coded. The program was tested successfully.

TRAFFIC LIGHT

Aim: Write a Java program that simulates a traffic light. The program lets the user select one of three lights: red, yellow, or green. When a radio button is selected, the light is turned on, and only one light can be on at a time. No light is on when the program starts.

Algorithm

Step 1: Start

Step 2: Declare a class named trafficlt that extends JPanel and implements ActionListener.

Step 3: Declare instance variables: JRadioButton objects r1, r2, r3, Color objects red_c, yelo_c, gr_c.

Step 4: Define the trafficlt constructor:

- a. Set the bounds for the panel (0, 0, 680, 450).
- b. Initialize JRadioButtons r1, r2, and r3 with labels "Red," "Yellow," and "Green."
- c. Add the radio buttons to the panel.
- d. Create a ButtonGroup g and add r1, r2, and r3 to it.
- e. Add an ActionListener (this) to each radio button.

Step 5: Implement the actionPerformed method:

- a. Check which radio button is selected using isSelected().
- b. Set the colors red_c, yelo_c, gr_c based on the selected radio button.
- c. Repaint the panel.

Step 6: Implement the paintComponent method:

- a. Draw three circles (representing traffic lights) with positions and sizes.
- b. Fill each circle with the corresponding color.

Step 7: Declare a class named traffic.

- a. Define the main method:
- b. Create a JFrame f titled "Traffic lights."
- c. Set the layout of f to null.
- d. Set the size of f to 680x450.
- e. Make f visible and set the default close operation.
- f. Create an object t of type trafficlt and add it to f.

Step 8: Stop

Program Code

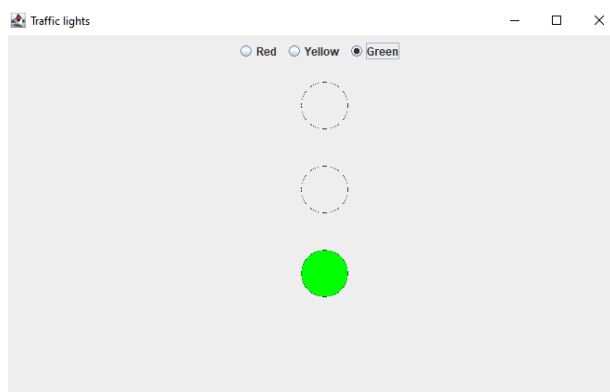
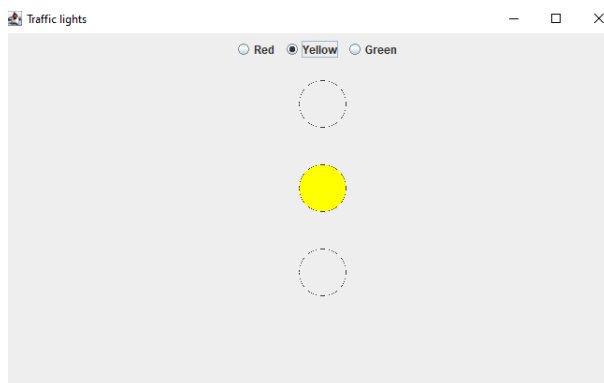
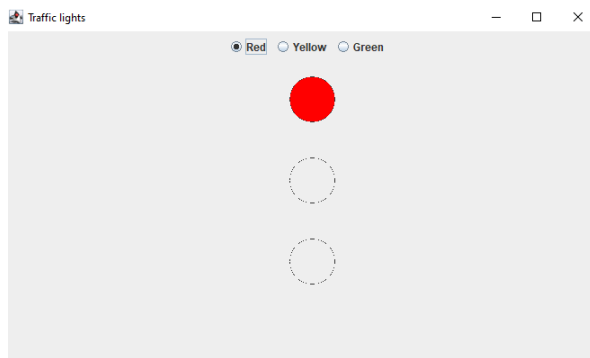
```

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
class trafficlt extends JPanel implements ActionListener
{
    JRadioButton r1,r2,r3;
    Color red_c,gr_c,yelo_c;
    trafficlt()
    {
        setBounds(0,0,680,450);
        r1=new JRadioButton("Red");
        r2=new JRadioButton("Yellow");
        r3=new JRadioButton("Green");
        add(r1);
        add(r2);
        add(r3);
        ButtonGroup g=new ButtonGroup();
        g.add(r1);
        g.add(r2);
        g.add(r3);
        r1.addActionListener(this);
        r2.addActionListener(this);
        r3.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e)
    {
        if(r1.isSelected()==true)
        {
            red_c=Color.red;
            yelo_c=getBackground();
            gr_c=getBackground();
        }
        else if(r2.isSelected()==true)
        {
            red_c=getBackground();
            yelo_c=Color.yellow;
            gr_c=getBackground();
        }
        else if(r3.isSelected()==true)
        {
            red_c=getBackground();
            yelo_c=getBackground();
            gr_c=Color.green;
        }
    }
}

```

```
    }
    repaint();
}
public void paintComponent(Graphics g)
{
    super.paintComponent(g);
    g.drawOval(320,50,50,50);
    g.drawOval(320,140,50,50);
    g.drawOval(320,230,50,50);
    g.setColor(red_c);
    g.fillOval(320,50,50,50);
    g.setColor(yelo_c);
    g.fillOval(320,140,50,50);
    g.setColor(gr_c);
    g.fillOval(320,230,50,50);
}
}
public class traffic
{
    public static void main(String[] args)
    {
        JFrame f=new JFrame("Traffic lights");
        f.setLayout(null);
        f.setVisible(true);
        f.setSize(680,450);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        traffict t=new traffict();
        f.add(t);
    }
}
```

Output



Result

The algorithm was developed and the program was coded. The program was tested successfully.

SESSION F

Exp. NO: 17

Date: 22/11/2023

DOUBLY LINKED LIST

Aim: Write a Java program for the following:

- 1) Create a doubly linked list of elements.
- 2) Delete a given element from the above list.
- 3) Display the contents of the list after deletion

Algorithm

Step 1: Start

Step 2: Initialize variables and create a DoubleLinkedList object (obj)

Step 3: Repeat while ch is 'y' or 'Y'

 Display menu options

 Input user choice (value)

 Switch on value

 Case 1:

 Input number of elements n

 Initialize a loop to input n elements and insert them into the double linked list using obj.insert(data)

 Case 2:

 Input data to be deleted

 Call obj.delete(data) to delete the specified data from the double linked list

 Case 3:

 Call obj.display() to display the elements in the double linked list

 Default:

 Print "Invalid Choice!!"

 Ask if the user wants to continue (input in ch)

Step 4: Stop

Program Code

```
import java.io.*;
import java.util.Scanner;
class Node
{
    int data;
    Node prev;
    Node next;
```

OBJECT ORIENTED PROGRAMMING LAB

```
Node(int data)
{
    this.data = data;
    prev = null;
    next = null;
}
}

class DoubleLinkedList
{
    Node head;

    void insert(int data)
    {
        Node newnode = new Node(data);
        if (head == null)
        {
            head = newnode;
        } else
        {
            Node temp;
            temp = head;
            while (temp.next != null)
            {
                temp = temp.next;
            }
            temp.next = newnode;
            newnode.prev = temp;
        }
    }

    void delete(int data)
    {
        Node temp, ptr;
        temp = head;
        while (temp != null) {
            if (temp.data == data) {
                if (temp.prev == null) // deleting node is head node
                {
                    head = temp.next;
                    if(temp.next!=null)
                    {
                        head.prev = null;
                    }
                } else if (temp.next == null) // deleting node is last node
                {
                    temp = temp.prev;
                }
            }
            temp = temp.next;
        }
    }
}
```



```

        temp.next = null;
    } else // deleting node is any node b/w head and last
    {
        ptr = temp.prev; // previous node is stored in ptr
        ptr.next = temp.next;
        temp = temp.next;
        temp.prev = ptr;
    }
    break;
}
temp = temp.next;
}
}

void display() {
    Node temp;
    temp = head;
    if (temp==null)
    {
        System.out.println("List is EMPTY");
    }
    else
    {
        while (temp!=null)
        {
            System.out.print(temp.data + " ");
            temp = temp.next;
        }
    }
}

public class Pgm16 {
    public static void main(String args[])
    {
        int value,n,data;
        char ch='y';
        Scanner reader=new Scanner(System.in);
        DoubleLinkedList obj=new DoubleLinkedList();
        while (ch=='y' || ch=='Y')
        {
            System.out.print("1 : CREATE\n2 : DELETE\n3 : DISPLAY\n");
            System.out.print("Enter your choice : ");
            value=reader.nextInt();
            switch(value)
            {
                case 1: System.out.print("How many elements ? : ");

```

```
        n=reader.nextInt();
        System.out.print("Enter the elements : ");
        for(int i=0;i<n;i++)
        {
            data=reader.nextInt();
            obj.insert(data);
        }
        break;

    case 2: System.out.print("Enter the data to be deleted : ");
            data=reader.nextInt();
            obj.delete(data);
            break;

    case 3: obj.display();
            System.out.println();
            break;

    default : System.out.println("Invalid Choice!!\n");
}
System.out.print("Do you want to continue?(y/n) : ");
ch=reader.next().charAt(0);
}
}
}
```

Output

```
1 : CREATE
2 : DELETE
3 : DISPLAY
Enter your choice : 1
How many elements ? : 5
Enter the elements : 10 20 30 40 50
Do you want to continue ?(y/n) : y
1 : CREATE
2 : DELETE
3 : DISPLAY
Enter your choice : 3
10 20 30 40 50
Do you want to continue ?(y/n) : y
1 : CREATE
2 : DELETE
3 : DISPLAY
Enter your choice : 2
Enter the data to be deleted : 20
Do you want to continue ?(y/n) : y
1 : CREATE
2 : DELETE
3 : DISPLAY
Enter your choice : 3
10 30 40 50
Do you want to continue ?(y/n) : n
```

Result

The algorithm was developed and the program was coded. The program was tested successfully.

Exp. NO: 17

Date: 22/11/2023

QUICK SORT

Aim: Write a Java program that implements Quick sort algorithm for sorting a list of names in ascending order.

Algorithm

Step 1: Start

Step 2: Declare a class QuickSort with the following methods:

- a. quicksort(int l, int h, String[] arr)
- b. partition(int l, int h, String[] arr)

Step 3: In the quicksort method:

- a. If $l < h$, then do the following:
 - i. Set partitionIndex to the result of calling partition(l, h, arr)
 - ii. Call quicksort(l, partitionIndex-1, arr)
 - iii. Call quicksort(partitionIndex+1, h, arr)

Step 4: In the partition method:

- a. Set pivot to arr[h]
- b. Initialize i to l - 1
- c. Loop through j from l to h-1:
 - i. If arr[j] is less than or equal to pivot, then do the following:
 - Increment i
 - Swap arr[i] and arr[j]
- d. Swap arr[i + 1] and arr[h] (pivot)
- e. Return i + 1 as the partition index

Step 5: In the Qsort class:

- a. Declare a main method
- b. Create an array names with the length of args
- c. Print "Before Quick Sort:" followed by the elements in args
- d. Create an instance of QuickSort called obj
- e. Call obj.quicksort(0, names.length-1, names)
- f. Print "After Quick Sort:" followed by the sorted elements in the names array

Step 6: Stop

Program Code

```

class QuickSort
{
    protected void quicksort(int l,int h,String[] arr)
    {
        if(l<h)
        {
            int partitionIndex=partition(l,h,arr);
            quicksort(l,partitionIndex-1,arr);
            quicksort(partitionIndex+1,h,arr);
        }
    }
    protected int partition(int l,int h,String[] arr)
    {
        String pivot = arr[h];
        int i = l - 1;

        for (int j = l; j < h; j++) {
            if (arr[j].compareToIgnoreCase(pivot) <= 0) {
                i++;

                // Swap arr[i] and arr[j]
                String temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }

        // Swap arr[i+1] and arr[high] (pivot)
        String temp = arr[i + 1];
        arr[i + 1] = arr[h];
        arr[h] = temp;

        return i + 1;
    }
}

public class Qsort
{
    public static void main(String args[])
    {
        String[] names=new String[args.length];
        System.out.print("Before Quick Sort : ");
        for(int i=0;i<args.length;i++)
        {
            names[i]=args[i];

```

```
        System.out.print(names[i]+" ");
    }
    System.out.println();
    QuickSort obj=new QuickSort();
    obj.quicksort(0,names.length-1,names);
    System.out.print("After Quick Sort : ");
    for(int i=0;i<names.length;i++)
    {
        System.out.print(names[i]+" ");
    }
    System.out.println();
}
}
```

Output

```
PS D:\My\RIT\S3\Object Oriented Programming using Java\LAB> javac Qsort.java
PS D:\My\RIT\S3\Object Oriented Programming using Java\LAB> java Qsort Emma Mathew Fathima Abhishek Aman
Before Quick Sort : Emma Mathew Fathima Abhishek Aman
After Quick Sort : Abhishek Aman Emma Fathima Mathew
```

```
PS D:\My\RIT\S3\Object Oriented Programming using Java\LAB> javac Qsort.java
PS D:\My\RIT\S3\Object Oriented Programming using Java\LAB> java Qsort Anwar John Abhijith
Before Quick Sort : Anwar John Abhijith
After Quick Sort : Abhijith Anwar John
```

Result

The algorithm was developed and the program was coded. The program was tested successfully.

Exp. NO: 18

Date: 22/11/2023

BINARY SEARCH

Aim: Write a Java program that implements the binary search algorithm

Algorithm

Step 1: Start

Step 2: Create a Search object

 Input array size n

 Initialize an array a of size 25

 Input elements into array a

 Input number to be searched sno

Step 3 : Create a Search object obj with array a, sno, and n

 Call obj.binarysearch() and store the result in 'result'

 If result is -1, then:

 Print sno + " Not Found"

 Else:

 Print sno + " is present at " + result

Step 4: Stop

Program Code

```
import java.util.Scanner;
class Search
{
    protected int low=0,high,mid,sno,n;
    private int arr[];
    Search(int a[],int sno,int n)
    {
        arr=a;
        this.sno=sno;
        this.n=n;
    }
    protected int binarysearch()
    {
        high=n-1;
        while(low<=high)
        {
            mid=(low+high)/2;
            if (arr[mid]==sno)
            {
```

```

        return mid+1;
    }
    if (arr[mid]>sno)
    {
        high=mid-1;
    }
    else
    {
        low=mid+1;
    }
}
return -1;
}

}

public class Pgm18
{
    public static void main(String args[])
    {
        int n,sno,result;
        Scanner reader=new Scanner(System.in);
        int a[]=new int[25];
        System.out.print("How many elements ? :");
        n=reader.nextInt();
        System.out.print("Enter the elements : ");
        for(int i=0;i<n;i++)
        {
            a[i]=reader.nextInt();
        }
        System.out.print("Enter the number to be searched : ");
        sno=reader.nextInt();
        Search obj=new Search(a,sno,n);
        result=obj.binarysearch();
        if (result!=-1)
        {
            System.out.println(sno+" Not Found");
        }
        else
        {
            System.out.println(sno+" is present at "+result);
        }
    }
}

```


Output

```
How many elements ? :5  
Enter the elements : 10 20 30 40 50  
Enter the number to be searched : 40  
40 is present at 4
```

```
How many elements ? :3  
Enter the elements : 18 -11 22  
Enter the number to be searched : 11  
11 Not Found
```

Result

The algorithm was developed and the program was coded. The program was tested successfully.

SESSION P

Exp. NO: 19**Date: 13/11/2023**

PACKAGE - 1

Aim: To write a Java program which contains three classes A, B and Main. The class A is in package 'packageone' and contains a method 'methodClassOne' that prints a message stating that it is printed from class A. Similarly, class B is in package 'packagetwo' and contains a method 'methodClassTwo' that prints a message stating that it is printed from class B. The Main class imports these classes, creates their instances and invokes their corresponding methods.

Algorithm

Step 1: Start
Step 2: Create a package packageone
Step 3: Create a class A in packageone
Step 4: Class A contain method methodclassone
Step 5: Create a package packagetwo
Step 6: Create a class B in packagetwo
Step 7: Class B contain method methodclasstwo
Step 8: Create main class
Step 9: Create instances of class A
Step 10: Create instances of class B
Step 11: Invokes methods from class A and class B
Step 12: Stop

Program Code


```
package packageone;
public class A
{
    public void methodClassOne()
    {
        System.out.println("I am from class A");
    }
}
```

OBJECT ORIENTED PROGRAMMING LAB

```
package packagetwo;
public class B
{
    public void methodClassTwo()
    {
        System.out.println("I am from class B");
    }
}
```

```
import packageone.A;
import packagetwo.B;
public class Main
{
    public static void main(String args[])
    {
        A obj1;
        B obj2;
        obj1=new A();
        obj2=new B();
        obj1.methodClassOne();
        obj2.methodClassTwo();
    }
}
```

Output



```
I am from class A
I am from class B
```

Result

The algorithm was developed and the program was coded. The program was tested successfully.

Exp. NO: 20**Date: 13/11/2023**

PACKAGE -2

Aim: Consider a scenario where an interface named 'Animal' is in a package named 'animalpackage' and it declares two methods 'get_cries' and 'get_eating_type' that are place holders for printing the cry and eating type of animals. The subpackages of 'animalpackage' are 'herbivorous' and 'carnivorous'. The 'herbivorous' package contains two classes 'Cow' and 'Elephant' that implements the 'Animal' interface. Similarly, the 'carnivorous' package contains 'Lion' and 'Bear' that too implement the interface. To Design a Main class that import these classes and invokes the corresponding methods.

Algorithm

Step 1: Start
Step 2: Create Animal interface in animalpackage
Step 3: Create classes in herbivores package implementing Animal interface
Step 4: Create class cow and elephant in herbivores package
Step 5: Create classes in carnivores package implementing Animal Interface
Step 6: Create class lion and bear in carnivores package
Step 7: Create class Main
Step 8: Create instances of herbivores animals
Step 9: Create instances of carnivores animals
Step 10: Invokes methods for herbivores animals
Step 11: Invokes methods for carnivores animals
Step 12: Stop

Program Code

```
package animalpackage;
public interface Animal
{
    void get_cries();
    void get_eating_type();
}

package animalpackage.Carnivores;
```

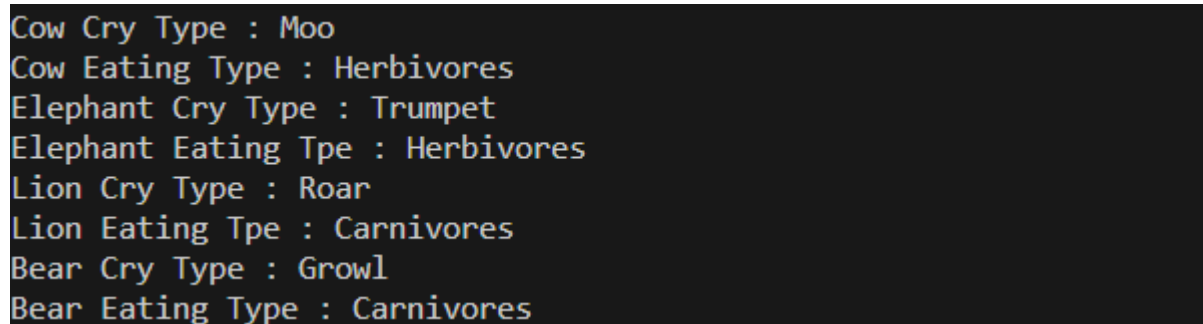
OBJECT ORIENTED PROGRAMMING LAB

```
import animalpackage.Animal;
public class Bear implements Animal
{
    public void get_cries()
    {
        System.out.println("Bear Cry Type : Growl");
    }
    public void get_eating_type()
    {
        System.out.println("Bear Eating Type : Carnivores");
    }
}
public class Lion implements Animal
{
    public void get_cries()
    {
        System.out.println("Lion Cry Type : Roar");
    }
    public void get_eating_type()
    {
        System.out.println("Lion Eating Type : Carnivores");
    }
}

package animalpackage.Herbivores;
import animalpackage.Animal;
public class Cow implements Animal
{
    public void get_cries()
    {
        System.out.println("Cow Cry Type : Moo");
    }
    public void get_eating_type()
    {
        System.out.println("Cow Eating Type : Herbivores");
    }
}
public class Elephant implements Animal
{
    public void get_cries()
    {
        System.out.println("Elephant Cry Type : Trumpet");
    }
    public void get_eating_type()
    {
        System.out.println("Elephant Eating Type : Herbivores");
    }
}
```

```
    }  
}  
  
import animalpackage.*;  
import animalpackage.Carnivores.*;  
import animalpackage.Herbivores.*;  
public class Main  
{  
    public static void main(String args[])  
    {  
        Animal obj;  
        obj=new Cow();  
        obj.get_cries();  
        obj.get_eating_type();  
        obj=new Elephant();  
        obj.get_cries();  
        obj.get_eating_type();  
        obj=new Lion();  
        obj.get_cries();  
        obj.get_eating_type();  
        obj=new Bear();  
        obj.get_cries();  
        obj.get_eating_type();  
    }  
}
```

Output



```
Cow Cry Type : Moo  
Cow Eating Type : Herbivores  
Elephant Cry Type : Trumpet  
Elephant Eating Tpe : Herbivores  
Lion Cry Type : Roar  
Lion Eating Tpe : Carnivores  
Bear Cry Type : Growl  
Bear Eating Type : Carnivores
```

Result

The algorithm was developed and the program was coded. The program was tested successfully.