AIM:

To Explain the concept of Object-Oriented Programming (OOP) and its benefits. To explain inheritance and Provide a code example in Java.

ANSWER:

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which can contain data and code: data in the form of fields and code, in the form of procedures.

Benefits:

-> Code Reusability

-> Security

-> Maintenance of Code

-> Easily upgradable and scalable

-> Flexibility

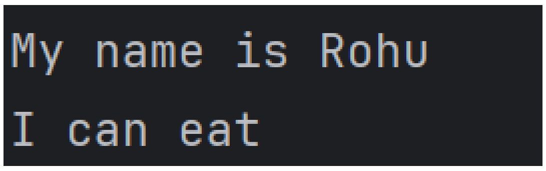
Inheritance: It is the process by which one object acquires the properties(fields and methods) of another object. In Addition, The new class can also add its own methods and fields. The top most class is called the SuperClass and the class inherited from subclass is called Subclass.

CODE:

```
 class Animal {
        String name;
         public void eat() {
           System.out.println("I can eat");
         }
}
class Dog extends Animal {
         public void display() {
           System.out.println("My name is " + name);
         }
}

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

```
        Dog labrador = new Dog();
        labrador.name = "Rohu";
        labrador.display();
        labrador.eat();

        }
}
```

OUTPUT:



```
My name is Rohu
I can eat
```

AIM:

To write a Java program to calculate the factorial of a number.

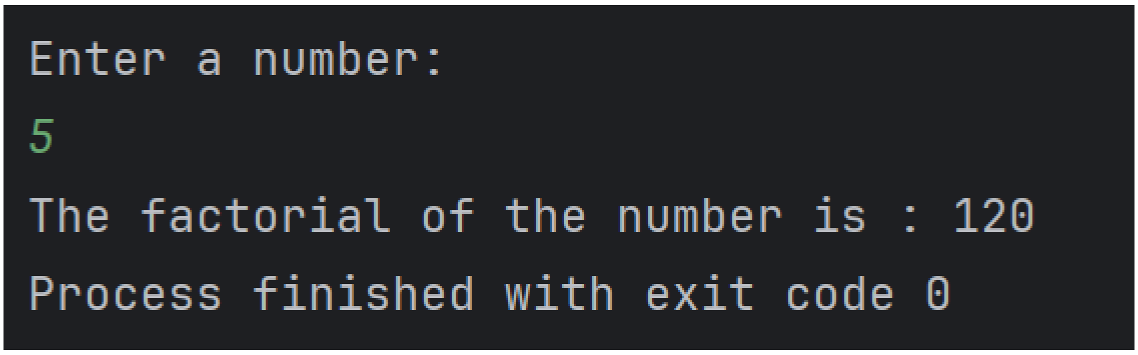
ALGORITHM:

1. Input the number and call the factorial function.
2. Initialise a variable result to 1.
3. Iterate from 1 to the given number n.
4. Multiply the result by the current number in each iteration.
5. After the loop ends, the result will contain the factorial of the given number.
6. Print the resultant variable , which is the factorial.

CODE:

```
import java.util.Scanner;

public class Main {


    public static void factorial(int num){

        int factorial=1;

        if (num<0){

            System.out.print("The factorial of the number does not exist!");

        }

        else if(num==0 || num==1){

            System.out.printf("The factorial of the number is : %d",1);

        }

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

            factorial*=i;

        }

        System.out.printf("The factorial of the number is : %d",factorial);
        }
        public static void main(String[] args) {
            Scanner input=new Scanner(System.in);
            System.out.println("Enter a number:");
            int number=input.nextInt();
            factorial(number);
        }
    }
```

OUTPUT:



```
Enter a number:
5
The factorial of the number is : 120
Process finished with exit code 0
```

AIM:

To write a Java program to implement a simple calculator.

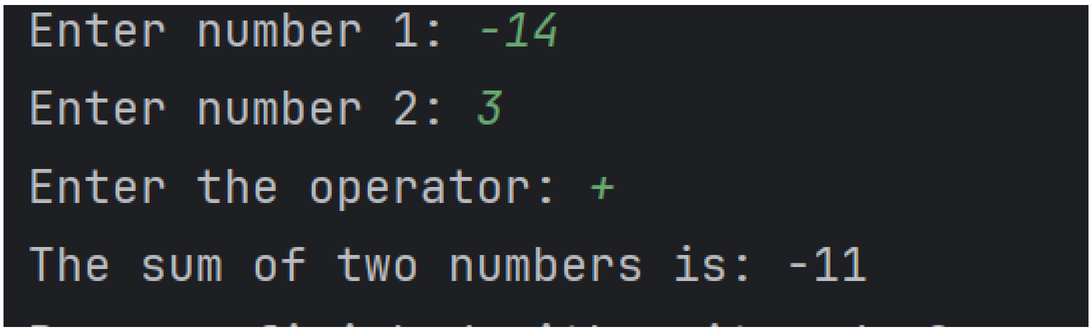
ALGORITHM:

1. Input two numbers and an operator.
2. Call the calculator function with the two numbers and the operator.
3. In the calculator function, use a switch statement to determine the operation:
   - Add the numbers if the operator is '+' and print the result.
   - Subtract the numbers if the operator is '-' and print the result.
   - Multiply the numbers if the operator is '*' and print the result.
   - Divide the numbers if the operator is '/' and print the result.
   - Print an error message if the operator is not recognized.

CODE:

```
import  java.util.Scanner;
public class Calculator {
   public static void calculator(int num1,int num2,char operator){
      switch (operator) {
         case '+':
            System.out.printf("The sum of two numbers is: %d", num1 + num2);
            break;
         case '-':
            System.out.printf("The difference of two numbers is: %d", num1 - num2);
            break;
         case '*':
            System.out.printf("The product of two numbers is: %d", num1 * num2);
            break;
         case '/' :
            System.out.printf("The division of two numbers is: %d%n", num1 / num2);
            break;
          default:
            System.out.println("No such operation exists.");
        }
      }
      public static void main(String[] args){
        Scanner input=new Scanner(System.in);
        System.out.print("Enter number 1: ");
        int num1=input.nextInt();
        System.out.print("Enter number 2: ");
        int num2=input.nextInt();
        System.out.print("Enter the operator: ");
        char operator=input.next().charAt(0);
        calculator(num1,num2,operator);
      }
    }
```

OUTPUT:



```
Enter number 1: -14
Enter number 2: 3
Enter the operator: +
The sum of two numbers is: -11
```

AIM:

To write a Java program to perform the multiplication of two matrices.

ALGORITHM:

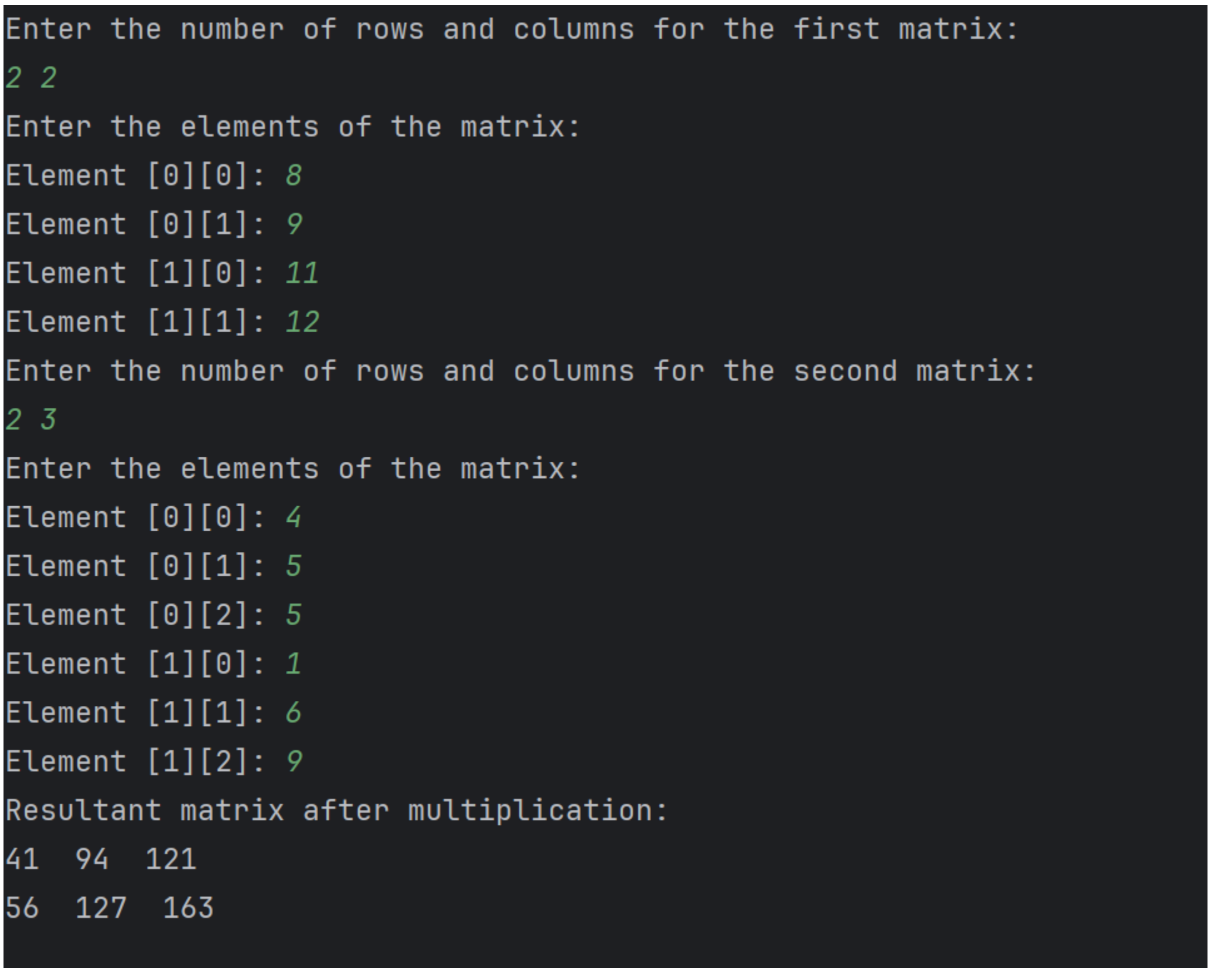
1. Input the dimensions of matrix and create the respective matrix.
2. Ensure the number of columns in the first matrix equals the number of rows in the second matrix
3. Perform Multiplication using nested for loops and compute the sum of products of corresponding elements from the first matrix row and second matrix column.
4. Return the resultant matrix and print it.

CODE:

```
import java.util.Scanner;


public class MatrixMultiplication {
   public static int[][] createMatrix(Scanner input, int rows, int cols) {
      int[][] matrix = new int[rows][cols];
      System.out.println("Enter the elements of the matrix:");
      for (int i = 0; i < rows; i++) {
         for (int j = 0; j < cols; j++) {
            System.out.printf("Element [%d][%d]: ", i, j);
            matrix[i][j] = input.nextInt();
         }
      }
      return matrix;
   }
   public static void readMatrix(int[][] matrix) {
      for (int[] row : matrix) {
         for (int elem : row) {
            System.out.printf("%d  ", elem);
         }
```

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

    public static int[][] multiplyMatrix(int[][] m1, int[][] m2, int rows1, int cols1, int cols2) {
        int[][] result = new int[rows1][cols2];
        for (int i = 0; i < rows1; i++) {
            for (int j = 0; j < cols2; j++) {
                result[i][j] = 0;
                for (int k = 0; k < cols1; k++) {
                    result[i][j] += m1[i][k] * m2[k][j];
                }
            }
        }
        return result;
    }

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

        System.out.println("Enter the number of rows and columns for the first matrix:");
        int rows1 = input.nextInt();
        int cols1 = input.nextInt();
        int[][] m1 = createMatrix(input, rows1, cols1);

        System.out.println("Enter the number of rows and columns for the second matrix:");
        int rows2 = input.nextInt();
        int cols2 = input.nextInt();

        if (cols1 != rows2) {
            System.out.println("Matrix multiplication not possible: number of columns in the first
matrix must equal number of rows in the second matrix.");
            return;
        }
        int[][] m2 = createMatrix(input, rows2, cols2);
        int[][] result = multiplyMatrix(m1, m2, rows1, cols1, cols2);
        System.out.println("Resultant matrix after multiplication:");
        readMatrix(result);
    }
}
```

OUTPUT:



```
Enter the number of rows and columns for the first matrix:
2 2
Enter the elements of the matrix:
Element [0][0]: 8
Element [0][1]: 9
Element [1][0]: 11
Element [1][1]: 12
Enter the number of rows and columns for the second matrix:
2 3
Enter the elements of the matrix:
Element [0][0]: 4
Element [0][1]: 5
Element [0][2]: 5
Element [1][0]: 1
Element [1][1]: 6
Element [1][2]: 9
Resultant matrix after multiplication:
41  94  121
56  127  163
```

AIM:

To write a Java method to compute the determinant of an $N \times N$ matrix using recursion

ALGORITHM:

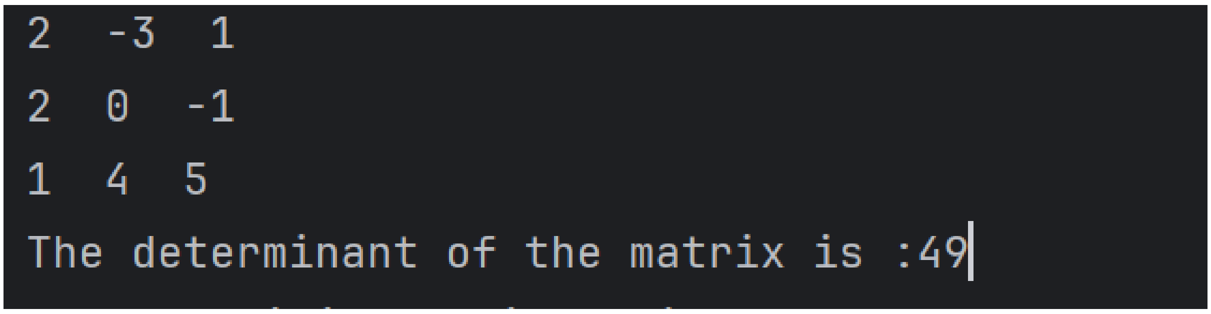
1. **Base cases:** If the matrix is 1x1, return the single element. If the matrix is 2x2, return the difference of the products of its diagonals.
2. **Recursion:** For each element in the first row, calculate the determinant of the corresponding cofactor matrix and compute the results with alternating signs.
3. **Cofactor matrix:** Create a cofactor matrix by excluding the current row and column.
4. Display the matrix and print the calculated determinant.

CODE:

```
public class Determinant {
   public static int determinant(int[][] mat){
      int n=mat.length;
      if (n==1){
         return mat[0][0];
      }
      else if (n==2){
         return mat[0][0]*mat[1][1]-mat[0][1]*mat[1][0];
      }
      int det=0;
      for (int i=0;i<n;i++){
         int[][] cofactor=cofactor(mat,0,i);
         int sign=(i%2==0)? 1 : -1;
         det+=sign*mat[0][i]*determinant(cofactor);
      }
      return det;
   }
   public static int[][] cofactor(int[][] mat,int row, int col){
      int n=mat.length;
      int r=0;
      int[][] cofactor=new int[n-1][n-1];
```

```
        for(int i=0;i<n;i++){
            if(i==row) continue;
            int c=0;
            for(int j=0;j<n;j++){
                if (j==col) continue;
                cofactor[r][c++]=mat[i][j];
            }
            r++;
        }
        return cofactor;
    }
    public static void display(int[][] mat, int row, int col)
    {
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++)
                System.out.printf("%d  ",mat[i][j]);

            System.out.print("\n");
        }
    }
    public static void main(String[] args){
        int[][] matrix = {
                {2, -3, 1},
                {2, 0, -1},
                {1, 4, 5}
        };
        display(matrix,3,3);
        int det=determinant(matrix);
        System.out.printf("The determinant of the matrix is :%d",det);
    }
}
```

OUTPUT



```
2  -3  1
2  0  -1
1  4  5
The determinant of the matrix is :49
```

AIM:

To Write a Java program to solve a system of linear equations using Cramer's rule

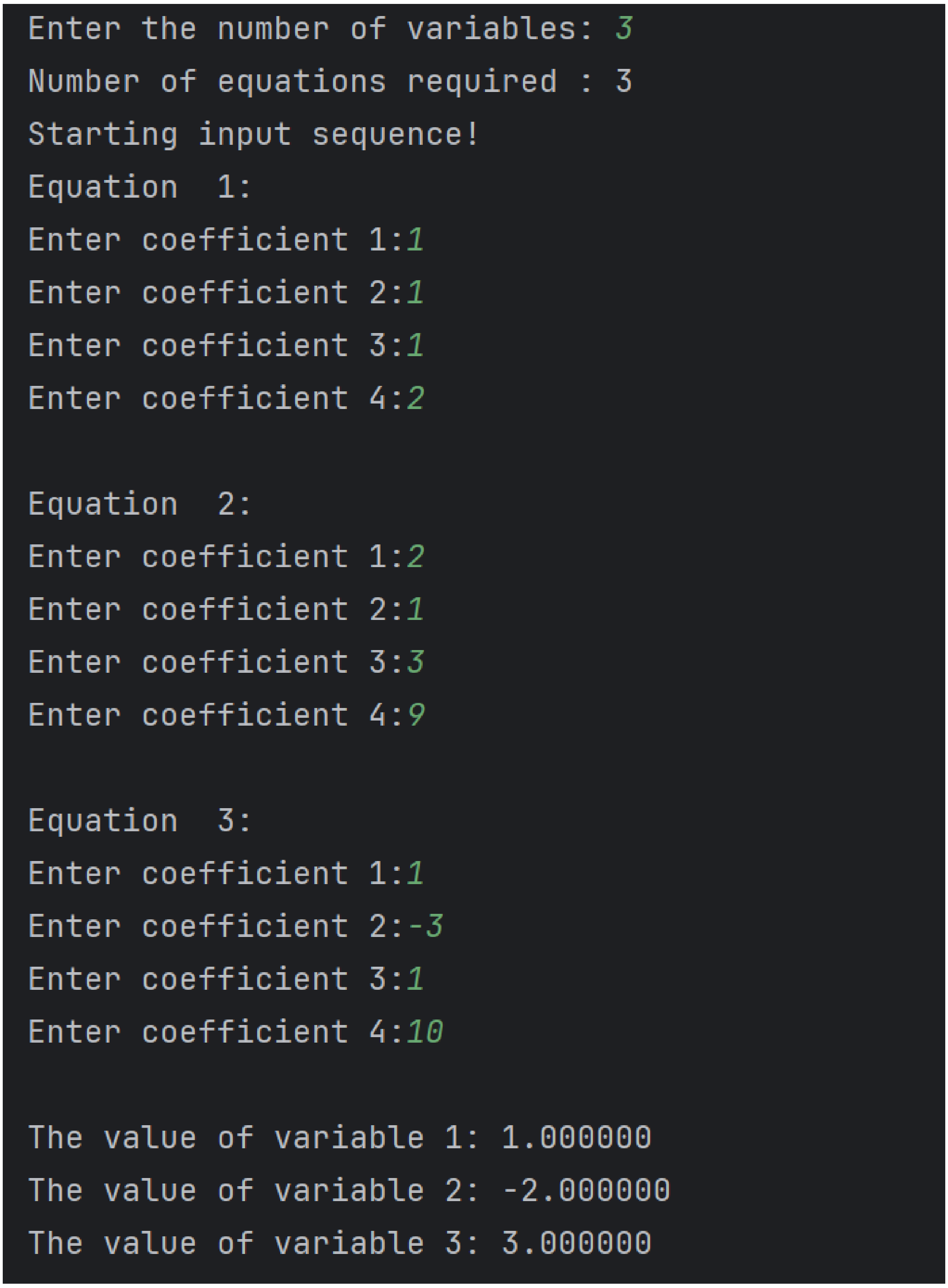
ALGORITHM:

1. Read the number of variables ‘n’ and then input the coefficients of each equation into matrix.
2. Use the determinant function to calculate the determinant of the matrix.
3. Create a temporary matrix by replacing the respective column and compute the determinant of this matrix.
4. Divide each partial determinant by the determinant of the original matrix.
5. Print the result of each variable.

CODE:

```
import java.util.Scanner;

public class CramersRule {

   static int  col=0;

   public static int determinant(int[][] mat){

      int n=mat.length;

      if (n==1){

         return mat[0][0];

      }

      else if (n==2){

         return mat[0][0]*mat[1][1]-mat[0][1]*mat[1][0];

      }

      int det=0;

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

         int[][] cofactor=cofactor(mat,0,i);

         int sign=(i%2==0)? 1 : -1;

         det+=sign*mat[0][i]*determinant(cofactor);

      }

      return det;

   }

   public static int[][] cofactor(int[][] mat,int row, int col){

      int n=mat.length;

      int r=0;

      int[][] cofactor=new int[n-1][n-1];

      for(int i=0;i<n;i++){
            if(i==row) continue;
            int c=0;
            for(int j=0;j<n;j++){
                if (j==col) continue;
                cofactor[r][c++]=mat[i][j];
            }
            r++;
        }
        return cofactor;
    }
    public static int partial(int[][]m,int[] cons,int n){
        int [][] temp= new int[n][n];
        //copying each element from m to temp
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                temp[i][j] = m[i][j];
            }
        }
        for(int i=0;i<n;i++){
            temp[i][col]=cons[i];
        }
        col+=1;
        int partial_det=determinant(temp);
        return partial_det;
    }
    public static void main(String[] args){
        Scanner input=new Scanner(System.in);
        System.out.print("Enter the number of variables: ");
        int n=input.nextInt();
        System.out.printf("Number of equations required : %d",n);
        System.out.println();
        System.out.println("Starting input sequence!");
        int[][] m=new int[n][n];
        int [] cons=new int[n];

        for(int i=0;i<n;i++){
            System.out.printf("Equation  %d:",i+1);
            System.out.println();
            for (int j=0;j<=n;j++){
                System.out.printf("Enter coefficient %d:",j+1);
                if (j == n) {
                    cons[i] = input.nextInt();
                } else {
                    m[i][j] = input.nextInt();
                }
            }
            System.out.println();
        }

        int det= determinant(m);
        for(int i=0;i<n;i++){
            int partial_det=partial(m,cons,n);
            double value=(double)partial_det/det;
            System.out.printf("The value of variable %d: %f ",i+1,value);
            System.out.println();
        }
    }
}
```

OUTPUT:



```
Enter the number of variables: 3
Number of equations required : 3
Starting input sequence!
Equation  1:
Enter coefficient 1:1
Enter coefficient 2:1
Enter coefficient 3:1
Enter coefficient 4:2

Equation  2:
Enter coefficient 1:2
Enter coefficient 2:1
Enter coefficient 3:3
Enter coefficient 4:9

Equation  3:
Enter coefficient 1:1
Enter coefficient 2:-3
Enter coefficient 3:1
Enter coefficient 4:10

The value of variable 1: 1.000000
The value of variable 2: -2.000000
The value of variable 3: 3.000000
```

AIM:

To write a Java program to find the eigenvalues of a 2x2 matrix and verify the determinant as the product of eigenvalues.

ALGORITHM:

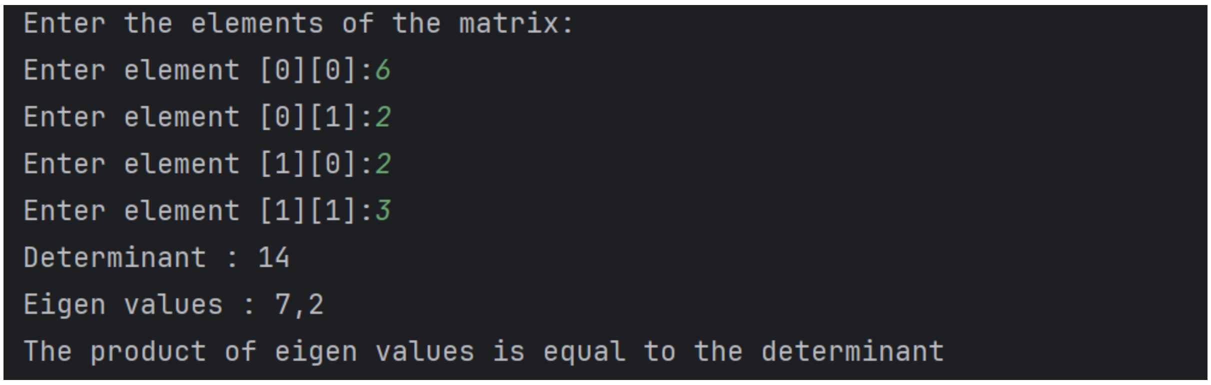
1. **Input the matrix:** Read the elements of a 2x2 matrix from user input.
2. Compute the sum of diagonal elements(S1) of the matrix.
3. Compute the determinant of the matrix(S2).
4. Use the quadratic formula to solve the characteristic equation: $\lambda^2 - S1 \cdot \lambda + S2 = 0$ for eigenvalues.
5. **Output the eigenvalues:** Print the calculated eigenvalues.
6. **Verify the result:** Check if the product of the eigenvalues equals the determinant and print the result.

CODE:

```
import java.util.Scanner;

import java.lang.Math;

public class EigenValues {


  public static int sum(int[][] m,int n){

    int sum=0;

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

      sum+=m[i][i];

    }

    return sum;

  }

  public static int determinant(int[][] mat){

    int n=mat.length;

    if (n==1){

      return mat[0][0];

    }

    else if (n==2){

      return mat[0][0]*mat[1][1]-mat[0][1]*mat[1][0];

    }

    int det=0;

    for (int i=0;i<n;i++){
        int[][] cofactor=cofactor(mat,0,i);
        int sign=(i%2==0)? 1 : -1;
        det+=sign*mat[0][i]*determinant(cofactor);
    }
    return det;
}
public static int[][] cofactor(int[][] mat,int row, int col){
    int n=mat.length;
    int r=0;
    int[][] cofactor=new int[n-1][n-1];
    for(int i=0;i<n;i++){
        if(i==row) continue;
        int c=0;
        for(int j=0;j<n;j++){
            if (j==col) continue;
            cofactor[r][c++]=mat[i][j];
        }
        r++;
    }
    return cofactor;
}
public static double[] solve(int sum,int det){
    //lambda^2-sum*lambda + determinant
    //a=1,b=sum,c=det
    int a=1;
    int b=-sum;
    int c=det;
    double value1=(-(b)+(Math.sqrt(b*b-4*a*c)))/2*a;
    double value2=(-(b)-(Math.sqrt(b*b-4*a*c)))/2*a;
    double[] eigen_values={value1,value2};
    return eigen_values;
}
```

```
    public static void main(String[] args){
        int n=2;
        Scanner input=new Scanner(System.in);
        System.out.println("Enter the elements of the matrix:");
        int[][] m= new int [n][n];
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                System.out.printf("Enter element [%d][%d]:",i,j);
                m[i][j]=input.nextInt();
            }
        }
        int sum=sum(m,n);
        int det=determinant(m);
        System.out.printf("Determinant : %d",det);
        System.out.println();
        double[] eigen_values=solve(sum,det);
        System.out.printf("Eigen values : %d,%d",(int)eigen_values[0],(int)eigen_values[1]);
        System.out.println();
        if (det == eigen_values[0] * eigen_values[1]) System.out.print("The product of eigen
values is equal to the determinant");
        else System.out.print("The product of eigen values is not equal to the determinant");


    }
}
```

OUTPUT:



```
Enter the elements of the matrix:
Enter element [0][0]:6
Enter element [0][1]:2
Enter element [1][0]:2
Enter element [1][1]:3
Determinant : 14
Eigen values : 7,2
The product of eigen values is equal to the determinant
```