M.Sc. (Five Year Integrated) in Computer Science (Artificial Intelligence & Data Science)

Second Semester

Laboratory Record

23-813-0207: JAVA PROGRAMMING LAB

Submitted in partial fulfillment
of the requirements for the award of degree in
Master of Science (Five Year Integrated)
in Computer Science (Artificial Intelligence & Data Science) of
Cochin University of Science and Technology (CUSAT)
Kochi

Submitted by

MUHSINA BEEGUM
(81323016)

DEPARTMENT OF COMPUTER SCIENCE
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY (CUSAT)
KOCHI-682022

APRIL 2024

DEPARTMENT OF COMPUTER SCIENCE

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY (CUSAT)

KOCHI, KERALA-682022

This is to certify that the software laboratory record for ***23-813-0207: JAVA PROGRAMMING Lab*** *is a record of work carried out by* ***MUHSINA BEEGUM(81323016)****, in partial fulfillment of the requirements for the award of degree in* ***Master of Science (Five Year Integrated)*** *in* ***Computer Science (Artificial Intelligence & Data Science)*** *of Cochin University of Science and Technology (CUSAT), Kochi. The lab record has been approved as it satisfies the academic requirements in respect of the second semester laboratory prescribed for the Master of Science (Five Year Integrated) in Computer Science degree.*

Faculty Member in-charge

Dr. Jeena K
Assistant Professor
Department of Computer Science
CUSAT

Dr. Madhu S Nair
Professor and Head
Department of Computer Science
CUSAT

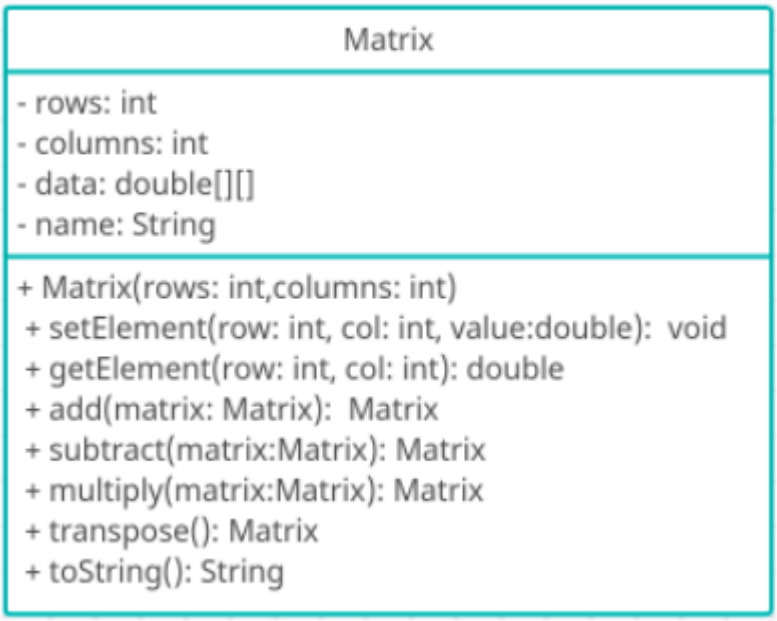
Contents

1.MATRIX-CLASS METHODS AND OBJECTS

AIM

Create a Java class "Matrix" with data fields and methods as shown in the UML Class diagram. Write a main function that reads the number of rows and columns from the user to create an object of the class and demonstrate each of its methods.

UML



PROGRAM

```
import java.util.Scanner;
public class Matrix {
    private int rows;
    private int coloumns;
    private double data[][];
    private String name;
    public Matrix(int r , int c){
        this.rows=r;
        this.coloumns=c;
        this.data=new double[rows][coloumns];

    }
    public void SetElement(int row ,int coloumn ,double value){
        if(row>=0 && row<rows && coloumn>=0 && coloumn<coloumns){
            data[row][coloumn]=value;
```

```
        }
        else{
            System.out.println("Invalid Position");
        }
    }

    public double getElement(int row, int coloumn){
        if(row>=0 && row<rows && coloumn>=0 && coloumn<coloumns){
            return data[row][coloumn];

        }
        else{
            System.out.println("invalid position");
            return -1;

        }
    }
    public void display(){
        System.out.println("Matrix : ");
        for(int i=0;i<rows;i++){
            for(int j=0;j<coloumns;j++){
                System.out.print(data[i][j]+" ");
            }
            System.out.println();
        }
    }
    public Matrix add (Matrix mat1){
        if(this.rows!=mat1.rows || this.coloumns!=mat1.coloumns){
            System.out.println("This Matrices cannot be added");
            return null;
        }
        else{
            Matrix sum = new Matrix (this.rows,this.coloumns);
            for(int i=0;i<this.rows;i++){
                for(int j=0;j<this.coloumns;j++){
                    double value=this.getElement(i,j)+mat1.getElement(i,j);
                    sum.SetElement(i,j, value);

                }
            }
             return sum;
```

```
            }
        }
    public Matrix sub (Matrix mat2){
        if(this.rows!=mat2.rows || this.coloumns!=mat2.coloumns){
            System.out.println("This matrices cannot be substracted");
            return null;
        }
        else{
            Matrix diff= new Matrix(this.rows,this.coloumns);
            for(int i=0;i<this.rows;i++){
                for(int j=0;j<this.coloumns;j++){
                    double value=this.getElement(i,j)-mat2.getElement(i,j);
                    diff.SetElement(i,j, value);
                }
            }
            return diff;
        }
    }
    public Matrix mul (Matrix mat3){
        if(this.coloumns!=mat3.rows){
            System.out.println("This matrices cannot be multiplied");
            return null;
        }
        else{
            Matrix prod=new Matrix (this.rows,this.coloumns);
            for(int i=0;i<this.rows;i++){
                for(int j=0;j<this.coloumns;j++){
                    double value=0;
                    for(int k=0;k<this.coloumns;k++){
                        value+=this.getElement(i,k)*mat3.getElement(k,j);
                    }
                    prod.SetElement(i,j, value);
                }
            }
            return prod;
        }
    }
    public Matrix transpose(){
        Matrix transpose=new Matrix(this.coloumns,this.rows);
        for(int i=0;i<this.rows;i++){
            for(int j=0;j<this.coloumns;j++){
```

```
                transpose.SetElement(j,i,this.getElement(i,j));
            }
        }
        return transpose;
    }
    public String toString(){
        StringBuilder string= new StringBuilder();
        for(int i=0;i<rows;i++){
            for(int j=0;j<coloumns;j++){
                string.append(getElement(i,j)).append(" ");
            }
        }
        return string.toString();
    }


    public static void main(String[] args){
        Scanner scanner=new Scanner(System.in);
        System.out.println("First matrix :");
        System.out.println("Number of rows of the matrix = ");
        int row=scanner.nextInt();
        System.out.println("Enter the number of coloumns of the matrix = ");
        int cols=scanner.nextInt();

        Matrix mat1=new Matrix(row,cols);
        System.out.println("Enter the elements of the matrix : ");
        for(int i=0;i<row;i++){
            for(int j=0;j<cols;j++){
                int val=scanner.nextInt();
                mat1.SetElement(i,j,val);
            }
            }
        mat1.display();

        System.out.println();
        System.out.println("Second matrix : ");
        System.out.println("Number of rows of matrix : ");
        int r=scanner.nextInt();
        System.out.println("Number of coloumns of matrix : ");
        int c=scanner.nextInt();
```

```
Matrix mat2=new Matrix(r,c);
System.out.println("Enter the elements of matrix : ");
for(int i=0;i<r;i++){
    for(int j=0;j<r;j++){
        int val=scanner.nextInt();
        mat2.SetElement(i,j, val);
    }
}
mat2.display();
System.out.println();

System.out.println("Addition of two matrices : ");
Matrix mat3=mat1.add(mat2);
if(mat3!=null){
    mat3.display();
}
System.out.println();

System.out.println("Difference of two matrices ");
Matrix mat4=mat1.sub(mat2);
if(mat4!=null){
    mat4.display();
}
System.out.println();

 System.out.println("Multiplication of two matrices : ");
 Matrix mat5=mat1.mul(mat2);
 if(mat5!=null){
     mat5.display();
 }
 System.out.println();

 System.out.println("Transpose of first matrix is : ");
 Matrix mat6=mat1.transpose();
 mat6.display();
 System.out.println();
 System.out.println();

 System.out.println("String form of the matrix is : ");
 System.out.println(mat1.toString());
```

```
    }
}
```

SAMPLE INPUT-OUTPUT

```
hp@hp-virtual-machine:~/java/record/cycle1/Qn1$ java Matrix
First matrix :
Number of rows of the matrix =
3
Enter the number of coloumns of the matrix =
2
Enter the elements of the matrix :
5
4
2
6
1
2
Matrix :
5.0 4.0
2.0 6.0
1.0 2.0

Second matrix :
Number of rows of matrix :
2
Number of coloumns of matrix :
4
Enter the elements of matrix :
2
3
1
5
Matrix :
2.0 3.0 0.0 0.0
1.0 5.0 0.0 0.0

Difference of two matrices
This matrices cannot be substracted

Multiplication of two matrices :
Matrix :
14.0 35.0
10.0 36.0
4.0 13.0

Transpose of first matrix is :
Matrix :
5.0 2.0 1.0
4.0 6.0 2.0


String form of the matrix is :
5.0 4.0 2.0 6.0 1.0 2.0
```

SAMPLE TEST CASE

SL NO	DESCRIPTION	INPUT	OUTPUT
1	Addition of 2 matrices	M1:- 5 4 2 6 1 2 M2:- 2 3 1 5	This matrices cannot be added.
2	Multiplication of 2 matrices	M1:- 5 4 2 6 1 2 M2:- 2 3 1 5	14 35 10 36 4 13
3	Difference of 2 matrices	M1:- 5 4 2 6 1 2 M2:- 2 3 1 5	This matrices cannot be substracted
4	Transpose of matrix	M1:- 1 2 3 4	1 3 2 4

2.MATRIX- input as command-line arguments.

AIM

Write a program to create a Matrix object by reading input as command-line arguments. The input shall be space-separated values as no. of rows and no. of columns followed by all elements in row-major format.

PROGRAM

```
public class Matrix2 {
    private int row;
    private int col;
    private int data[][];
    public Matrix2(int r,int c,int[] element){
        this.row=r;
        this.col=c;
        this.data=new int[row][col];
        int index=0;
        for(int i=0;i<row;i++){
            for(int j=0;j<col;j++){
                this.data[i][j]=element[index];
                index++;
            }
        }
    }
    public void display(){
        System.out.println("Matrix : ");
        for(int i=0;i<row;i++){
            for(int j=0;j<col;j++){
                System.out.print(data[i][j]+" ");
            }
            System.out.println();
        }
    }

    public static void main(String[] args){
        try{
            int rows = Integer.parseInt(args[0]);
            int cols = Integer.parseInt(args[1]);
            int[] elements = new int[args.length-2];
            for(int i=0;i<rows*cols;i++){
```

```
                elements[i] = Integer.parseInt(args[i+2]);
            }
            Matrix2 matrix2 = new Matrix2(rows, cols, elements);
            matrix2.display();
        }
        catch(NumberFormatException e){
                System.out.println("Invalid input. Please provide integers only");
        }
        catch(ArrayIndexOutOfBoundsException e){
            System.out.println("Insufficient arguments provided.");
        }
    }
}
```

SAMPLE INPUT-OUTPUT

```
hp@hp-virtual-machine:~/java/record/cycle1/Qn2$ java Matrix2 2 3 1 2 3 4 5 6
Matrix :
1 2 3
4 5 6
```

SAMPLE TEST CASE

SL NO		DESCRIPTION INPUT	OUTPUT	PASS/FAIL
1	Command Line argument	2 3 1 2 3 4 5 6	1 2 3 4 5 6	PASS

3.MATRIX- Menu Driven

AIM

Write a menu-driven program that repeatedly reads and carries out the following operations on a matrix object.

1. Create an object of the Matrix class by prompting the user for input, and storing it in the variable matx . You may replace the previous object with the new one if one already exists.
2. Print the values of the matrix in a proper format.
3. Print the column sum for any given column.
4. Print the row sum for any given row.
5. Print the average of all the values in the matrix
6. Check if the matrix is diagonal and Print YES or NO.
7. Exit the program after printing "Thank You!"

PROGRAM

```
import java.util.Scanner;

class Matrix3 {

    private int rows;
    private int col;
    private int data[][];

    public Matrix3(int rows, int cols, int[] Data) {
        this.rows = rows;
        this.col = cols;
        this.data = new int[rows][cols];
        int k = 0;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                this.data[i][j] = Data[k++];
            }
        }
    }


    public void display(){
```

```
        for(int i=0;i<rows;i++){
            for(int j=0;j<col;j++){
                System.out.print(data[i][j]+" ");
            }
            System.out.println();
        }
    }
    public int column_sum(int col_no){
        if(col_no>=0 && col_no<col ){
            int col_sum=0;
            for(int i=0;i<rows;i++){
                col_sum+=data[i][col_no];

            }
            return col_sum;
        }
        else{
            System.out.println("Incorrect column number");
            return -1;
        }
    }
    public int row_sum(int row_no){
        if(row_no>=0 && row_no<rows){
            int row_sum=0;
            for(int i=0;i<col;i++){
                row_sum+=data[row_no][i];
            }
            return row_sum;
        }
        else{
            System.out.println("Incorrect row number");
            return -1;
        }
    }
    public void average (){
        double sum=0;
        for(int i=0;i<this.rows;i++){
            for(int j=0;j<this.col;j++){
                sum+=this.data[i][j];
            }
        }
```

```
        double avg=sum/(this.rows*this.col);
        System.out.println("Average of all elements of matrix = "+avg);
    }
    public boolean isDiagonal() {
        if (rows != col) {
            return false;
        }
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < col; j++) {
                if (i != j && data[i][j] != 0) {
                    return false;
                }
            }
        }
        return true;
    }
}

public class MatrixMenu {
    private static Scanner scanner = new Scanner(System.in);
    private static Matrix3 matx;

    public static void main(String[] args) {
        char choice;
        do {
            System.out.println("\nMenu:");
            System.out.println("a. Create/Update matrix");
            System.out.println("b. Print matrix");
            System.out.println("c. Print Column Sum");
            System.out.println("d. Print Row Sum");
            System.out.println("e. Print Average of matrix");
            System.out.println("f. Check if matrix is Diagonal");
            System.out.println("g. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.next().charAt(0);

            switch (choice) {
                case 'a':
                    createOrUpdateMatrix();
                    break;
                case 'b':
```

```
                printMatrix();
                break;
            case 'c':
                printColumnSum();
                break;
            case 'd':
                printRowSum();
                break;
            case 'e':
                printAverage();
                break;
            case 'f':
                checkDiagonal();
                break;
            case 'g':
                System.out.println("Thank you!");
                break;
            default:
                System.out.println("Invalid choice. Please try again.");
        }
    } while (choice != 'g');
}

public static void createOrUpdateMatrix() {
    System.out.print("Enter number of rows: ");
    int rows = scanner.nextInt();
    System.out.print("Enter number of columns: ");
    int cols = scanner.nextInt();
    System.out.println("Enter matrix elements row-wise:");
    int[] elements = new int[rows * cols];
    for (int i = 0; i < rows * cols; i++) {
        elements[i] = scanner.nextInt();
    }
    matx = new Matrix3(rows, cols, elements);
    System.out.println("Matrix created/updated successfully.");
}

public static void printMatrix() {
    if (matx != null) {
        System.out.println("Matrix:");
        matx.display();
```

```
        }
        else {
            System.out.println("No matrix exists. Please create one first.");
        }
    }

    public static void printColumnSum() {
        if (matx != null) {
            System.out.print("Enter column number: ");
            int col = scanner.nextInt();
            int sum = matx.column_sum(col);
            System.out.println("Sum of column " + col + ": " + sum);
        } else {
            System.out.println("No matrix exists. Please create one first.");
        }
    }

    public static void printRowSum() {
        if (matx != null) {
            System.out.print("Enter row number: ");
            int row = scanner.nextInt();
            int sum = matx.row_sum(row);
            System.out.println("Sum of row " + row + ": " + sum);
        } else {
            System.out.println("No matrix exists. Please create one first.");
        }
    }

    public static void printAverage() {
        if (matx != null) {
             matx.average();
        } else {
            System.out.println("No matrix exists. Please create one first.");
        }
    }

    public static void checkDiagonal() {
        if (matx != null) {
            String result = matx.isDiagonal() ? "It is a diagonal matrix" :
            "It is not a diagonal matrix";
            System.out.println(result);
```

```
        } else {
            System.out.println("No matrix exists. Please create one first.");
        }
    }
}
```

SAMPLE INPUT-OUTPUT

```
Menu:
a. Create/Update matrix
b. Print matrix
c. Print Column Sum
d. Print Row Sum
e. Print Average of matrix
f. Check if matrix is Diagonal
g. Exit
Enter your choice: a
Enter number of rows: 2
Enter number of columns: 2
Enter matrix elements row-wise:
1
2
3
4
Matrix created/updated successfully.

Menu:
a. Create/Update matrix
b. Print matrix
c. Print Column Sum
d. Print Row Sum
e. Print Average of matrix
f. Check if matrix is Diagonal
g. Exit
Enter your choice: b
Matrix:
1 2
3 4
```

```
Menu:
a. Create/Update matrix
b. Print matrix
c. Print Column Sum
d. Print Row Sum
e. Print Average of matrix
f. Check if matrix is Diagonal
g. Exit
Enter your choice: c
Enter column number: 0
Sum of column 0: 4

Menu:
a. Create/Update matrix
b. Print matrix
c. Print Column Sum
d. Print Row Sum
e. Print Average of matrix
f. Check if matrix is Diagonal
g. Exit
Enter your choice: d
Enter row number: 1
Sum of row 1: 7

Menu:
a. Create/Update matrix
b. Print matrix
c. Print Column Sum
d. Print Row Sum
e. Print Average of matrix
f. Check if matrix is Diagonal
g. Exit
Enter your choice: d
Enter row number: 5
Incorrect row number
Sum of row 5: -1
```

```
Menu:
a. Create/Update matrix
b. Print matrix
c. Print Column Sum
d. Print Row Sum
e. Print Average of matrix
f. Check if matrix is Diagonal
g. Exit
Enter your choice: e
Average of all elements of matrix = 2.5

Menu:
a. Create/Update matrix
b. Print matrix
c. Print Column Sum
d. Print Row Sum
e. Print Average of matrix
f. Check if matrix is Diagonal
g. Exit
Enter your choice: f
It is not a diagonal matrix

Menu:
a. Create/Update matrix
b. Print matrix
c. Print Column Sum
d. Print Row Sum
e. Print Average of matrix
f. Check if matrix is Diagonal
g. Exit
Enter your choice: g
Thank you!
hp@hp-virtual-machine:~/java/record/cycle1/Qn3$
```

SAMPLE TEST CASES

SL NO	DESCRIPTION	INPUT	OUTPUT	PASS/FAIL
1	Print matrix	1 2 3 4	1 2 3 4	PASS
2	Column matrix sum	0	4	PASS
3	row sum	1	7	PASS
4	Average of matrix	1 2 3 4	2.5	PASS
5	If matrix is diagonal	1 2 3 4	NO	PASS

4.MATRIX - Random Matrix Generation

AIM

Add a no-argument constructor Matrix() that creates a matrix of random size (between 1 and 10) and populated with random values between 1 to 1000. Demonstrate.

PROGRAM

```
import java.util.Random;

public class MatrixRandom {
    private int[][] matrix;
    private int rows; // Added instance variable for rows
    private int cols; // Added instance variable for columns

    public MatrixRandom() {
        Random random = new Random();
        rows = random.nextInt(11); // Initialize rows
        cols = random.nextInt(11); // Initialize columns
        matrix = new int[rows][cols];

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                matrix[i][j] = random.nextInt(1000) + 1;
            }
        }
    }

    public void displayMatrix() {
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                System.out.print(matrix[i][j] + "\t");
            }
            System.out.println();
        }
    }

    public static void main(String[] args) {
        MatrixRandom randomMatrix = new MatrixRandom();
```

```
        System.out.println("Random Matrix:");
        System.out.println("Rows: " + randomMatrix.rows); // Display rows
        System.out.println("Columns: " + randomMatrix.cols); // Display columns
        randomMatrix.displayMatrix();
    }
}
```

SAMPLE INPUT-OUTPUT

```
hp@hp-virtual-machine:~/java/record/cycle1/Qn4$ java MatrixRandom
Random Matrix:
Rows: 3
Columns: 1
310
749
631
hp@hp-virtual-machine:~/java/record/cycle1/Qn4$ java MatrixRandom
Random Matrix:
Rows: 4
Columns: 5
996     258     801     558     421
791     331     729     529     667
409     216     776     498     603
851     523     117     65      190
```

SAMPLE TEST CASE

SL NO	DESCRIPTION	OUTPUT	PASS/FAIL
1	Random Matrix	310 749 631	PASS
2	Random Matrix	996 258 801 558 421 791 331 729 529 667 409 216 776 498 603 851 523 117 65 190	PASS

5.MATRIX- unique id

AIM

Add data field date UniqueID that stores a unique value for each object created. Hint: Keep count of no. of objects created and use it to generate the UniqueID.

PROGRAM

```
import java.util.Date;
import java.util.Scanner;

class Matrix3 {
    private int rows;
    private int col;
    private int data[][];

    public Matrix3(int rows, int cols, int[] Data) {
        this.rows = rows;
        this.col = cols;
        this.data = new int[rows][cols];
        int k = 0;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                this.data[i][j] = Data[k++];
            }
        }
    }

    public void display(){
        for(int i=0;i<rows;i++){
            for(int j=0;j<col;j++){
                System.out.print(data[i][j]+" ");
            }
            System.out.println();
        }
    }

    public int column_sum(int col_no){
        if(col_no>=0 && col_no<col ){
            int col_sum=0;
```

```
            for(int i=0;i<rows;i++){
                col_sum+=data[i][col_no];
            }
            return col_sum;
        }
        else{
            System.out.println("Incorrect column number");
            return -1;
        }
    }

    public int row_sum(int row_no){
        if(row_no>=0 && row_no<rows){
            int row_sum=0;
            for(int i=0;i<col;i++){
                row_sum+=data[row_no][i];
            }
            return row_sum;
        }
        else{
            System.out.println("Incorrect row number");
            return -1;
        }
    }

    public void average (){
        double sum=0;
        for(int i=0;i<this.rows;i++){
            for(int j=0;j<this.col;j++){
                sum+=this.data[i][j];
            }
        }
        double avg=sum/(this.rows*this.col);
        System.out.println("Average of all elements of matrix = "+avg);
    }

    public boolean isDiagonal() {
        if (rows != col) {
            return false;
        }
        for (int i = 0; i < rows; i++) {
```

```
            for (int j = 0; j < col; j++) {
                if (i != j && data[i][j] != 0) {
                    return false;
                }
            }
        }
        return true;
    }
}

public class UniqueId {
    private static Scanner in = new Scanner(System.in);
    private static Matrix3 matx;
    private static int ObjectCount = 0;
    private static Date date;
    private static int UniqueID;

    public UniqueId(){
        ObjectCount++;
        date = new Date();
        UniqueID = ObjectCount;
    }

    public void displayData(){
        System.out.println("Date : "+date);
        System.out.println("Unique Id : "+UniqueID);
    }

    public static void main(String[] args){


        char choice;
        do {
            System.out.println("\nMenu:");
            System.out.println("a. Create/Update matrix");
            System.out.println("b. Print matrix");
            System.out.println("c. Print Column Sum");
            System.out.println("d. Print Row Sum");
            System.out.println("e. Print Average of matrix");
            System.out.println("f. Check if matrix is Diagonal");
            System.out.println("g. Display Unique ID and Date");
```

```
            System.out.println("h. Exit");
            System.out.print("Enter your choice: ");
            choice = in.next().charAt(0);

            switch (choice) {
                case 'a':
                    createOrUpdateMatrix();
                    break;
                case 'b':
                    printMatrix();
                    break;
                case 'c':
                    printColumnSum();
                    break;
                case 'd':
                    printRowSum();
                    break;
                case 'e':
                    printAverage();
                    break;
                case 'f':
                    checkDiagonal();
                    break;
                case 'g':
                    displayUniqueIdAndDate();
                    break;
                case 'h':
                    System.out.println("Thank you!");
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        } while (choice != 'h');
    }

    public static void createOrUpdateMatrix() {
        System.out.print("Enter number of rows: ");
        int rows = in.nextInt();
        System.out.print("Enter number of columns: ");
        int cols = in.nextInt();
        System.out.println("Enter matrix elements row-wise:");
```

```
        int[] elements = new int[rows * cols];
        for (int i = 0; i < rows * cols; i++) {
            elements[i] = in.nextInt();
        }
        matx = new Matrix3(rows, cols, elements);
        ObjectCount++;
        date= new Date();
        UniqueID = ObjectCount;
        System.out.println("Matrix created/updated successfully.");
    }

    public static void printMatrix() {
        if (matx != null) {
            System.out.println("Matrix:");
            matx.display();
        }
        else {
            System.out.println("No matrix exists. Please create one first.");
        }
    }

    public static void printColumnSum() {
        if (matx != null) {
            System.out.print("Enter column number: ");
            int col = in.nextInt();
            int sum = matx.column_sum(col);
            System.out.println("Sum of column " + col + ": " + sum);
        } else {
            System.out.println("No matrix exists. Please create one first.");
        }
    }

    public static void printRowSum() {
        if (matx != null) {
            System.out.print("Enter row number: ");
            int row = in.nextInt();
            int sum = matx.row_sum(row);
            System.out.println("Sum of row " + row + ": " + sum);
        } else {
            System.out.println("No matrix exists. Please create one first.");
        }
```

```
    }

    public static void printAverage() {
        if (matx != null) {
             matx.average();
        } else {
            System.out.println("No matrix exists. Please create one first.");
        }
    }

    public static void checkDiagonal() {
        if (matx != null) {
            String result = matx.isDiagonal() ? "It is a diagonal matrix" :
            "It is not a diagonal matrix";
            System.out.println(result);
        } else {
            System.out.println("No matrix exists. Please create one first.");
        }
    }

    public static void displayUniqueIdAndDate() {

        System.out.println("Date : " + date);
        System.out.println("Unique Id : " + UniqueID);
    }
}
```

SAMPLE INPUT-OUTPUT

```
hp@hp-virtual-machine:~/java/record/cycle1/Qn5$ java UniqueId

Menu:
a. Create/Update matrix
b. Print matrix
c. Print Column Sum
d. Print Row Sum
e. Print Average of matrix
f. Check if matrix is Diagonal
g. Display Unique ID and Date
h. Exit
Enter your choice: a
Enter number of rows: 2
Enter number of columns: 2
Enter matrix elements row-wise:
1
1
1
1
Matrix created/updated successfully.

Menu:
a. Create/Update matrix
b. Print matrix
c. Print Column Sum
d. Print Row Sum
e. Print Average of matrix
f. Check if matrix is Diagonal
g. Display Unique ID and Date
h. Exit
Enter your choice: g
Date : Wed Apr 24 15:38:28 IST 2024
Unique Id : 1
```

SAMPLE TEST CASE

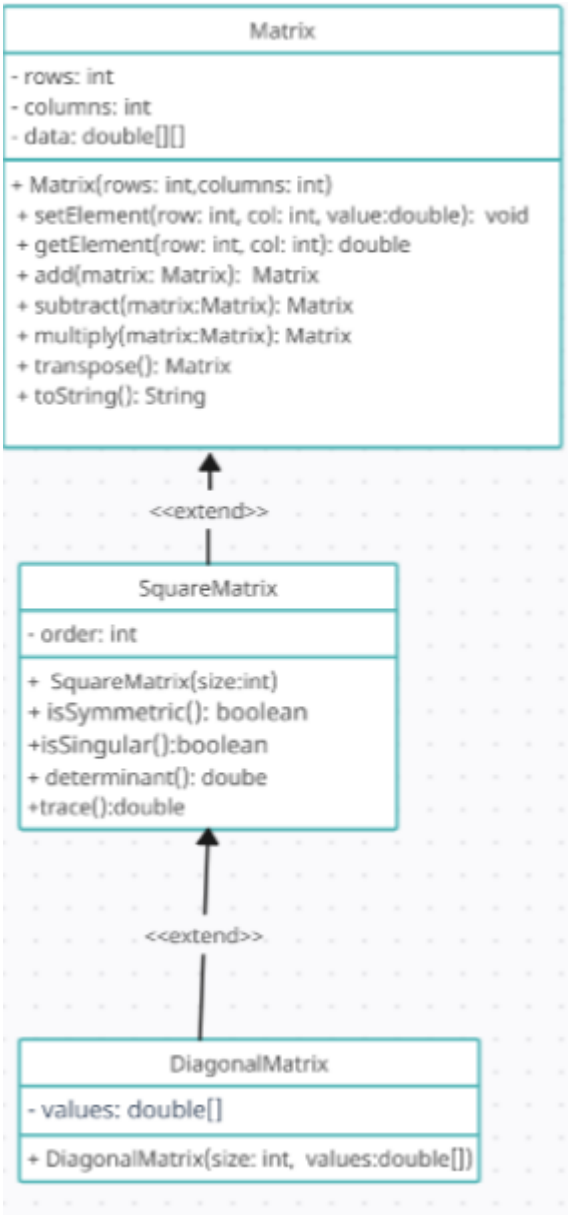
SL NO	DESCRIPTION	INPUT	OUTPUT	PASS/FAIL
1	enter your choice:	a	1 1 1 1	PASS
2	enter your choice :	g	1	PASS

6.MATRIX- INHERITANCE

AIM

Implement the subclasses SquareMatrix and Diagonal Matrix as shown in the UML diagram above . Demonstrate all methods with suitable test cases.

UML



PROGRAM

```
import java.util.Scanner;
public class Matrix {
    private int rows;
    private int coloumns;
```

```
private double data[][];
private String name;
public Matrix(int r , int c){
    this.rows=r;
    this.coloumns=c;
    this.data=new double[rows][coloumns];

}
public void SetElement(int row ,int coloumn ,double value){
    if(row>=0 && row<rows && coloumn>=0 && coloumn<coloumns){
        data[row][coloumn]=value;

    }
    else{
        System.out.println("Invalid Position");
    }
}

public double getElement(int row, int coloumn){
    if(row>=0 && row<rows && coloumn>=0 && coloumn<coloumns){
        return data[row][coloumn];

    }
    else{
        System.out.println("invalid position");
        return -1;

    }
}
public void display(){
    System.out.println("Matrix : ");
    for(int i=0;i<rows;i++){
        for(int j=0;j<coloumns;j++){
            System.out.print(data[i][j]+" ");
        }
        System.out.println();
    }
}
public Matrix add (Matrix mat1){
    if(this.rows!=mat1.rows || this.coloumns!=mat1.coloumns){
        System.out.println("This Matrices cannot be added");
```

```
            return null;
        }
        else{
            Matrix sum = new Matrix (this.rows,this.coloumns);
            for(int i=0;i<this.rows;i++){
                for(int j=0;j<this.coloumns;j++){
                    double value=this.getElement(i,j)+mat1.getElement(i,j);
                    sum.SetElement(i,j, value);

                }
            }
             return sum;
            }
        }
    public Matrix sub (Matrix mat2){
        if(this.rows!=mat2.rows || this.coloumns!=mat2.coloumns){
            System.out.println("This matrices cannot be substracted");
            return null;
        }
        else{
            Matrix diff= new Matrix(this.rows,this.coloumns);
            for(int i=0;i<this.rows;i++){
                for(int j=0;j<this.coloumns;j++){
                    double value=this.getElement(i,j)-mat2.getElement(i,j);
                    diff.SetElement(i,j, value);
                }
            }
            return diff;
        }
    }
    public Matrix mul (Matrix mat3){
        if(this.coloumns!=mat3.rows){
            System.out.println("This matrices cannot be multiplied");
            return null;
        }
        else{
            Matrix prod=new Matrix (this.rows,this.coloumns);
            for(int i=0;i<this.rows;i++){
                for(int j=0;j<this.coloumns;j++){
                    double value=0;
                    for(int k=0;k<this.coloumns;k++){
```

```
                    value+=this.getElement(i,k)*mat3.getElement(k,j);
                }
                prod.SetElement(i,j, value);
            }
        }
        return prod;
    }
}
public Matrix transpose(){
    Matrix transpose=new Matrix(this.coloumns,this.rows);
    for(int i=0;i<this.rows;i++){
        for(int j=0;j<this.coloumns;j++){
            transpose.SetElement(j,i,this.getElement(i,j));
        }
    }
    return transpose;
}
public String toString(){
    StringBuilder string= new StringBuilder();
    for(int i=0;i<rows;i++){
        for(int j=0;j<coloumns;j++){
            string.append(getElement(i,j)).append(" ");
        }
    }
    return string.toString();
}

public class SquareMatrix extends Matrix{
    public int order;

    public SquareMatrix(int order){
        super(order,order);
        this.order=order;
    }



    public double determinant(){
        if(this.order==1){
            return this.getElement(0,0);
        }
        else if(this.order==2){
```

```
                return (this.getElement(0, 0)
                * this.getElement(1, 1)) -
                (this.getElement(0, 1) * this.getElement(1, 0));
        } else {
            double det = 0;
            for (int i = 0; i < this.order; i++) {
                det+= Math.pow(-1, i) *
                this.getElement(0, i) * this.minor(0, i).determinant();
            }
            return det;
            }
        }


    private SquareMatrix minor(int row,int col){
           SquareMatrix minor= new SquareMatrix (this.order-1);
           int m=0;
           for(int i=0;i<this.order;i++){
               if(i==row)continue;
               int n=0;
               for(int j=0;j<this.order;j++){
                   if(j==col)continue;
                   minor.SetElement(m,n,this.getElement(i,j));
                   n++;
               }
               m++;
       }
           return minor;
    }


    public boolean isSymmetric(){
        return this.equals(this.transpose());
    }

    public boolean isSingular() {
        return this.determinant() == 0;
    }

    public double trace() {
        double trace = 0;
```

```
        for (int i = 0; i < this.order; i++) {
            trace += this.getElement(i, i);
        }
        return trace;
    }
}

class DiagonalMatrix extends SquareMatrix {
    public DiagonalMatrix(int size, double values[]) {
        super(size);
        for (int i = 0; i < size; i++) {
            this.SetElement(i, i, values[i]);
        }
    }
}
    public static void main(String[] args){
        Scanner scanner=new Scanner(System.in);
        System.out.println("First matrix :");
        System.out.println("Number of rows of the matrix = ");
        int row=scanner.nextInt();
        System.out.println("Enter the number of coloumns of the matrix = ");
        int cols=scanner.nextInt();

        Matrix mat1=new Matrix(row,cols);
        System.out.println("Enter the elements of the matrix : ");
        for(int i=0;i<row;i++){
            for(int j=0;j<cols;j++){
                int val=scanner.nextInt();
                mat1.SetElement(i,j,val);
            }
            }
        mat1.display();

        System.out.println();
        System.out.println("Second matrix : ");
        System.out.println("Number of rows of matrix : ");
        int r=scanner.nextInt();
        System.out.println("Number of coloumns of matrix : ");
        int c=scanner.nextInt();

        Matrix mat2=new Matrix(r,c);
```

```
System.out.println("Enter the elements of matrix : ");
for(int i=0;i<r;i++){
    for(int j=0;j<r;j++){
        int val=scanner.nextInt();
        mat2.SetElement(i,j, val);
    }
}
mat2.display();
System.out.println();

System.out.println("Addition of two matrices : ");
Matrix mat3=mat1.add(mat2);
if(mat3!=null){
    mat3.display();
}
System.out.println();

System.out.println("Difference of two matrices ");
Matrix mat4=mat1.sub(mat2);
if(mat4!=null){
    mat4.display();
}
System.out.println();

 System.out.println("Multiplication of two matrices : ");
 Matrix mat5=mat1.mul(mat2);
 if(mat5!=null){
     mat5.display();
 }
 System.out.println();

 System.out.println("Transpose of first matrix is : ");
 Matrix mat6=mat1.transpose();
 mat6.display();
 System.out.println();
 System.out.println();

 System.out.println(mat1.toString());

 System.out.println("Enter the order of square matrix :");
 int order=scanner.nextInt();
```

```
        Matrix matrixInst = new Matrix(0,0);
        SquareMatrix squareMatrix = matrixInst.new SquareMatrix(order);

        System.out.println("Enter the Elements for the Square matrix :");
        for(int i=0;i<order;i++){
            for(int j=0;j<order;j++){
                squareMatrix.SetElement(i,j,scanner.nextDouble());
            }
        }
        System.out.println("\n Square Matrix :");
        System.out.println(squareMatrix);

        System.out.println("Determinant of the square matrix = ");
        System.out.println(squareMatrix.determinant());

        System.out.println("Is the Matrix Symmetric : ");
        System.out.println(squareMatrix.isSymmetric());

        System.out.println("Is the Matrix Singular : ");
        System.out.println(squareMatrix.isSingular());

        System.out.println("Trace of the Square Matrix :");
        System.out.println(squareMatrix.trace());
        System.out.println();

       System.out.println("Enter the order for the diagonal Matrix :");
       int diaOrder = scanner.nextInt();
       double[] diaValues= new double [diaOrder];

       System.out.println("Enter the Diagonal Elements : ");
       for(int i=0;i<diaOrder;i++){
           diaValues[i]=scanner.nextDouble();
       }

       DiagonalMatrix diaMatrix = matrixInst.new DiagonalMatrix (diaOrder,diaValues);
       System.out.println("Diagonal Matrix :\n");
       diaMatrix.display();
    }
}
```

SAMPLE INPUT-OUTPUT

```
First matrix :
Number of rows of the matrix =
6
Enter the number of coloumns of the matrix =
2
Enter the elements of the matrix :
1
2
3
4
5
6
7
8
9
10
11
12
Matrix :
1.0 2.0
3.0 4.0
5.0 6.0
7.0 8.0
9.0 10.0
11.0 12.0

Second matrix :
Number of rows of matrix :
2
Number of coloumns of matrix :
2
Enter the elements of matrix :
3
2
4
```

```
Matrix :
3.0 2.0
4.0 1.0

Addition of two matrices :
This Matrices cannot be added

Difference of two matrices
This matrices cannot be substracted

Multiplication of two matrices :
Matrix :
11.0 4.0
25.0 10.0
39.0 16.0
53.0 22.0
67.0 28.0
81.0 34.0

Transpose of first matrix is :
Matrix :
1.0 3.0 5.0 7.0 9.0 11.0
2.0 4.0 6.0 8.0 10.0 12.0


1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0
Enter the order of square matrix :
2
Enter the Elements for the Square matrix :
5
4
8
9
```

```
 Square Matrix :
5.0 4.0 8.0 9.0
Determinant of the square matrix =
13.0
Is the Matrix Symmetric :
false
Is the Matrix Singular :
false
Trace of the Square Matrix :
14.0

Enter the order for the diagonal Matrix :
3
Enter the Diagonal Elements :
3
2
1
Diagonal Matrix :

Matrix :
3.0 0.0 0.0
0.0 2.0 0.0
0.0 0.0 1.0
hp@hp-virtual-machine:~/java/record/cycle2/Qn1$
```

SAMPLE TEST CASE

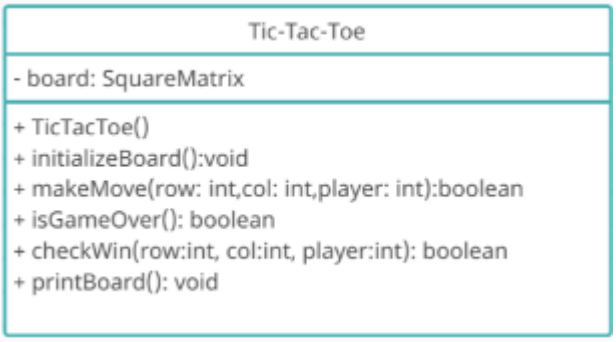
SL NO	DESCRIPTION	INPUT	OUTPUT	PASS
1	Matrix creation	Matrix1:- 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 Matrix2:- 3.0 2.0 4.0 1.0	1.0 2.0 3.0 4.0 5.0 6.0 3.0 2.0 7.0 8.0 4.0 1.0 9.0 10.0 11.0 12.0	PASS
2	Addition of matrix	”	This matrices cannot be added	PASS
3	Difference of matrix	”	This matrices cannot be substracted	PASS
4	Product of matrix	”	11.0 4.0 25.0 10.0 39.0 16.0 53.0 22.0 67.0 28.0 81.0 34.0	
5	Transpose	”	3.0 4.0 2.0 1.0	PASS
6	Determinant	5.0 4.0 8.0 9.0	13.0	PASS
7	Symmetric	”	False	PASS
8	Singular	”	False	PASS
9	Trace	”	14.0	PASS
10	Diagonal Matrix	3 2 1	3.0 0.0 0.0 0.0 2.0 0.0 0.0 0.0 1.0	PASS

7.TIC-TAC-TOE

AIM

Write a Java program to implement the tic-tac-toe game, as shown in the UML design above . The program should alternatively read row and column numbers from the two users and check if the move is valid. If the move is invalid, alert the user and ask for a new move. If a valid move, update the board, print it, and check for a winner. If no winner yet, check if the game is over. Otherwise, continue. The explanation of each method is given below. board: A private instance variable representing the game board as a SquareMatrix. TicTacToe(): Constructor for initializing a new game of Tic-Tac-Toe. startingBoard(): Method to initialize the game board with all zeroes. placeMark(row: int, col: int, player: int): boolean: Method to make a move on the game board for the specified player (-1 for player X, 1 for player 0). Returns true if the move is valid, false otherwise. isGameOver(): boolean: Method to check if the game is over (either a player wins or the board is full). If there are no zeroes, board is full. checkWin(row: int, col: int, player: char):boolean : Method to check the player has won after the new move. printBoard(): void: Method to print the current state of the game board. Remember to use X and O (not 1 and -1) while printing the board.

UML



PROGRAM

```
import java.util.*;
public class TicTacToe{
    static String[] board;
    static String turn;
    static String checkWinner(){
        for(int a=0;a<8;a++){
            String line=null;
            switch(a){
                case 0:
                    line=board[0]+board[1]+board[2];
                    break;
```

```
            case 1:
                line=board[3]+board[4]+board[5];
                break;
            case 2:
                line=board[6]+board[7]+board[8];
                break;
            case 3:
                line=board[0]+board[3]+board[6];
                break;
            case 4:
                line=board[1]+board[4]+board[7];
                break;
            case 5:
                line=board[2]+board[5]+board[8];
                break;
            case 6:
                line=board[0]+board[4]+board[8];
                break;
            case 7:
                line=board[2]+board[4]+board[6];
                break;

        }
        if (line.endsWith("XXX")){
            return "X";
        }
        else if(line.endsWith("OOO")){
            return "O";
        }
    }

    for (int a=0;a<9;a++){
        if(Arrays.asList(board).contains(String.valueOf(a+1))){
            break;
        }
        else if(a==8){
            return "draw";
        }
    }
    System.out.println(turn+" 's turn : enter a slot number to place "+turn+" in : ");
    return null;
```

```
    }

    //to print the board
    static void printBoard(){

        System.out.println("|-------------|");
        System.out.println("|  "+board[0]+" | "+board[1]+" | "+board[2]+"  |");
        System.out.println("|-------------|");
        System.out.println("|  "+board[3]+" | "+board[4]+" | "+board[5]+"  |");
        System.out.println("|-------------|");
        System.out.println("|  "+board[6]+" | "+board[7]+" | "+board[8]+"  |");
        System.out.println("|-------------|");

    }

    public static void main(String[] args) {
        Scanner in= new Scanner(System.in);
        board=new String[9];
        turn="X";
        String winner=null;
        for(int a=0;a<9;a++){
            board[a]=String.valueOf(a+1);
        }
        System.out.println("Welcome to 3*3 Tic Tac Toe.");
        printBoard();
        System.out.println("X will play first.Enter a slot number to place X in : ");
        while (winner==null){
            int numInput;
            try{
                numInput=in.nextInt();
                if(!(numInput > 0 && numInput <= 9)){
                    System.out.println("Invalid input : re-enter slot number :");
                    continue;
                }
            }
            catch(InputMismatchException e ){
                System.out.println("Invalid input : re-enter slot number : ");
                continue;
            }
            if (board[numInput-1].equals(String.valueOf(numInput))){
                board[numInput-1]=turn;
```

```
                if(turn.equals("X")){
                    turn="O";
                }
                else{
                    turn="X";
                }
                printBoard();
                winner=checkWinner();
            }
            else{
                System.out.println("Slot already taken : re- enter slot number : ");
            }
        }
        if(winner.equalsIgnoreCase("draw")){
            System.out.println("It's a draw! Thanks for playing .");

        }
        else{
            System.out.println("Congratulations !"+winner+" 's have won!
            Thanks for playing .");
        }
        in.close();
    }

}
```

SAMPLE INPUT-OUTPUT

```
Welcome to 3*3 Tic Tac Toe.
|-------------|
|  1 | 2 | 3  |
|-------------|
|  4 | 5 | 6  |
|-------------|
|  7 | 8 | 9  |
|-------------|
X will play first.Enter a slot number to place X in :
5
|-------------|
|  1 | 2 | 3  |
|-------------|
|  4 | X | 6  |
|-------------|
|  7 | 8 | 9  |
|-------------|
O 's turn : enter a slot number to place O in :
2
|-------------|
|  1 | O | 3  |
|-------------|
|  4 | X | 6  |
|-------------|
|  7 | 8 | 9  |
|-------------|
X 's turn : enter a slot number to place X in :
9
|-------------|
|  1 | O | 3  |
|-------------|
|  4 | X | 6  |
|-------------|
|  7 | 8 | X  |
|-------------|
O 's turn : enter a slot number to place O in :
1
|-------------|
|  O | O | 3  |
|-------------|
|  4 | X | 6  |
|-------------|
|  7 | 8 | X  |
|-------------|
X 's turn : enter a slot number to place X in :
3
|-------------|
|  O | O | X  |
|-------------|
|  4 | X | 6  |
|-------------|
|  7 | 8 | X  |
|-------------|
O 's turn : enter a slot number to place O in :
7
|-------------|
|  O | O | X  |
|-------------|
|  4 | X | 6  |
|-------------|
|  O | 8 | X  |
|-------------|
X 's turn : enter a slot number to place X in :
4
|-------------|
|  O | O | X  |
|-------------|
|  X | X | 6  |
|-------------|
|  O | 8 | X  |
|-------------|
```

```
O 's turn : enter a slot number to place O in :
6
|-------------|
|  O | O | X  |
|-------------|
|  X | X | O  |
|-------------|
|  O | 8 | X  |
|-------------|
X 's turn : enter a slot number to place X in :
8
|-------------|
|  O | O | X  |
|-------------|
|  X | X | O  |
|-------------|
|  O | X | X  |
|-------------|
It's a draw! Thanks for playing .
hp@hp-virtual-machine:~/java/record/cycle2/Qn2$ javac TicTacToe.java
```

SAMPLE TEST CASE

SL NO	DESCRIPTION	INPUT	OUTPUT	PASS/FAIL
1	Player X	5	1 2 3 4 *X* 6 7 8 9	PASS
2	Player O	2	1 *O* 3 4 *X* 6 7 8 9	PASS
3	Player X	9	1 *O* 3 4 *X* 6 7 8 *X*	PASS
4	Player O	1	*O* *O* 3 4 *X* 6 7 8 *X*	PASS

8.TIC-TAC-TOE - Exceptions(In-Built)

AIM

Modify the makeMove() method in the Tic-Tac-Toc class to handle conditions where the values of rows and columns are not within the limit. If a player tries an invalid move outside the board boundaries, the method should throw an IllegalArgumentException with a clear error message. The Main class should catch this exception and handle it by printing an error message and continuing the game with the correct turn.

PROGRAM

```
public class TicTacToe {
    static SquareMatrix board;
    static String turn;

    public TicTacToe() {
        board = new SquareMatrix(3);
        turn= "X";
    }

    public boolean MakeMove(int row, int col, int turn) {
        if (row < 0 || row >= 3 || col < 0 || col >= 3) {
            throw new IllegalArgumentException
            ("INVALID MOVE: Row and column values must be between 0 and 2.");
        } else if (board.getElement(row, col) != 0) {
            return false;
        } else {
            board.SetElement(row, col, turn);
            return true;
        }
    }

    public boolean isGameOver() {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                if (board.getElement(i, j) == 0) {
                    return false;
                }
            }
        }
```

```
        return true;
    }

    public boolean checkWin(int row, int col, int turn) {
        int sumrow = 0;
        int sumcol = 0;
        int sumdiagonal1 = 0;
        int sumdiagonal2 = 0;

        for (int i = 0; i < 3; i++) {
            sumrow += board.getElement(row, i);
            sumcol += board.getElement(i, col);
        }

        if (row == col) {
            for (int i = 0; i < 3; i++) {
                sumdiagonal1 += board.getElement(i, i);
            }
        }
        if (row + col == 2) {
            for (int i = 0; i < 3; i++) {
                sumdiagonal2 += board.getElement(i, 2 - i);
            }
        }

        if (sumrow == 3 || sumrow == -3 || sumcol == 3 || sumcol == -3
                || sumdiagonal1 == 3 || sumdiagonal1 == -3 ||
                sumdiagonal2 == 3 || sumdiagonal2 == -3) {
            return true;
        }
        return false;
    }

    public void printBoard() {
        System.out.println("_____________________________");
        for (int i = 0; i < 3; i++) {
            System.out.print("|");
            for (int j = 0; j < 3; j++) {
                if (board.getElement(i, j) == -1) {
                    System.out.print("   X   ");
                } else if (board.getElement(i, j) == 1) {
```

```
                System.out.print("   O   ");
            } else {
                System.out.print("   _   ");
            }
            System.out.print("|");
        }
        System.out.println();
        System.out.println("_________________________");
    }
}

public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    TicTacToe game = new TicTacToe();
    turn = "X";
    boolean gameEnd = false;
    String winner = null;
    System.out.println("Welcome to 3*3 Tic Tac Toe.");
    game.printBoard();
    System.out.println("X will play first. Enter
    a slot number to place X in: ");

    try {
        while (!gameEnd) {
            int numInput;
            numInput = in.nextInt();
            if (!(numInput > 0 && numInput <= 9)) {
                System.out.println("Invalid input:
                re-enter slot number:");
                continue;
            }
            int row = (numInput - 1) / 3;
            int col = (numInput - 1) % 3;

            if (game.MakeMove(row, col, turn.equals("X") ? -1 : 1)) {
                game.printBoard();
                if (game.checkWin(row, col, turn.equals("X") ? -1 : 1)) {
                    winner = turn;
                    gameEnd = true;
                } else if (game.isGameOver()) {
                    winner = "draw";
```

```
                    gameEnd = true;
                } else {
                    turn = turn.equals("X") ? "O" : "X";
                    System.out.println(turn + " It's your turn! Enter a
                    slot number where " + turn + " in: ");
                }
            } else {
                System.out.println("Slot already taken or invalid move:
                re-enter slot number:");
            }
        }
    } catch (IllegalArgumentException e) {
        System.out.println(e.getMessage());
    }

    if (winner.equalsIgnoreCase("draw")) {
        System.out.println("It's a draw! Thanks for playing.");
    } else {
        System.out.println("Congratulations! " + winner + " has won!
        Thanks for playing.");
    }
    in.close();
  }
}
```

SAMPLE INPUT-OUTPUT

```
hp@hp-virtual-machine:~/java/record/cycle3/TicTacToe$ java TicTacToe
Welcome to 3*3 Tic Tac Toe.
 _____ _____ _____
|     |     |     |
|  _  |  _  |  _  |
|_____|_____|_____|
|     |     |     |
|  _  |  _  |  _  |
|_____|_____|_____|
|     |     |     |
|  _  |  _  |  _  |
|_____|_____|_____|
X will play first. Enter a slot number to place X in:
6
 _____ _____ _____
|     |     |     |
|  _  |  _  |  _  |
|_____|_____|_____|
|     |     |     |
|  _  |  _  |  X  |
|_____|_____|_____|
|     |     |     |
|  _  |  _  |  _  |
|_____|_____|_____|
O It's your turn! Enter a slot number where O in:
5
 _____ _____ _____
|     |     |     |
|  _  |  _  |  _  |
|_____|_____|_____|
|     |     |     |
|  _  |  O  |  X  |
|_____|_____|_____|
|     |     |     |
|  _  |  _  |  _  |
|_____|_____|_____|
X It's your turn! Enter a slot number where X in:
4
```

```
 _____ _____ _____
|     |     |     |
|  _  |  _  |  _  |
|_____|_____|_____|
|     |     |     |
|  X  |  O  |  X  |
|_____|_____|_____|
|     |     |     |
|  _  |  _  |  _  |
|_____|_____|_____|
O It's your turn! Enter a slot number where O in:
3
 _____ _____ _____
|     |     |     |
|  _  |  _  |  O  |
|_____|_____|_____|
|     |     |     |
|  X  |  O  |  X  |
|_____|_____|_____|
|     |     |     |
|  _  |  _  |  _  |
|_____|_____|_____|
X It's your turn! Enter a slot number where X in:
2
 _____ _____ _____
|     |     |     |
|  _  |  X  |  O  |
|_____|_____|_____|
|     |     |     |
|  X  |  O  |  X  |
|_____|_____|_____|
|     |     |     |
|  _  |  _  |  _  |
|_____|_____|_____|
O It's your turn! Enter a slot number where O in:
6
Slot already taken or invalid move: re-enter slot number:
7
 _____ _____ _____
|     |     |     |
|  _  |  X  |  O  |
|_____|_____|_____|
|     |     |     |
|  X  |  O  |  X  |
|_____|_____|_____|
|     |     |     |
|  O  |  _  |  _  |
|_____|_____|_____|
Congratulations! O has won! Thanks for playing.
hp@hp-virtual-machine:~/java/record/cycle3/TicTacToe$
```

SAMPLE TEST CASE

SL NO	DESCRIPTION	INPUT	OUTPUT	PASS/FAIL
1	Player 0	1 1	– – – – 0 – – – –	PASS
2	Player X	0 0	X – – – 0 – – – –	PASS
3	Player 0	2 2	X – – – 0 – – – 0	PASS
4	Player X	2 3	Invalid move	PASS

9.TIC-TAC-TOE - Exceptions(User Defined)

AIM

Write your own Exception class InvalidMoveException and use it instead of IllegalArgumentException in the above example.

PROGRAM

```
import java.util.Scanner;

public class TicTacToee {
    static SquareMatrix board;
    static String turn;

    public TicTacToee() {
        board = new SquareMatrix(3);
    }

    public boolean MakeMove(int row, int col, int turn) throws InvalidMoveException {
        if (row < 0 || row >= 3 || col < 0 || col >= 3) {
            throw new InvalidMoveException
            ("INVALID MOVE: Row and column values must be between 0 and 2.");
        } else if (board.getElement(row, col) != 0) {
            return false;
        } else {
            board.SetElement(row, col, turn);
            return true;
        }
    }

    public boolean isGameOver() {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                if (board.getElement(i, j) == 0) {
                    return false;
                }
            }
        }
        return true;
    }
```

```
public boolean checkWin(int row, int col, int turn) {
    int sumrow = 0;
    int sumcol = 0;
    int sumdiagonal1 = 0;
    int sumdiagonal2 = 0;

    for (int i = 0; i < 3; i++) {
        sumrow += board.getElement(row, i);
        sumcol += board.getElement(i, col);
    }

    if (row == col) {
        for (int i = 0; i < 3; i++) {
            sumdiagonal1 += board.getElement(i, i);
        }
    }
    if (row + col == 2) {
        for (int i = 0; i < 3; i++) {
            sumdiagonal2 += board.getElement(i, 2 - i);
        }
    }

    if (sumrow == 3 || sumrow == -3 || sumcol == 3 || sumcol == -3
            || sumdiagonal1 == 3 || sumdiagonal1 == -3 ||
            sumdiagonal2 == 3 || sumdiagonal2 == -3) {
        return true;
    }
    return false;
}

public void printBoard() {
    System.out.println("_________________________");
    for (int i = 0; i < 3; i++) {
        System.out.print("|");
        for (int j = 0; j < 3; j++) {
            if (board.getElement(i, j) == -1) {
                System.out.print("   X   ");
            } else if (board.getElement(i, j) == 1) {
                System.out.print("   O   ");
            } else {
                System.out.print("   _   ");
```

```
            }
            System.out.print("|");
        }
        System.out.println();
        System.out.println("-------------------------");
    }
}

public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    TicTacToee game = new TicTacToee();
    turn = "X";
    boolean gameEnd = false;
    String winner = null;
    System.out.println("Welcome to 3*3 Tic Tac Toe.");
    game.printBoard();
    System.out.println("X will play first. Enter a slot
    number to place X in: ");

    while (!gameEnd) {
        int numInput;
        try {
            numInput = in.nextInt();
            if (!(numInput > 0 && numInput <= 9)) {
                System.out.println("Invalid input:
                re-enter slot number:");
                continue;
            }
            int row = (numInput - 1) / 3;
            int col = (numInput - 1) % 3;

            if (game.MakeMove(row, col, turn.equals("X") ? -1 : 1)) {
                game.printBoard();
                if (game.checkWin(row, col, turn.equals("X") ? -1 : 1)) {
                    winner = turn;
                    gameEnd = true;
                } else if (game.isGameOver()) {
                    winner = "draw";
                    gameEnd = true;
                } else {
                    turn = turn.equals("X") ? "O" : "X";
```

```
                        System.out.println(turn + " It's your turn!
                        Enter a slot number where " + turn + " in: ");
                    }
                } else {
                    System.out.println("Slot already taken or invalid move:
                    re-enter slot number:");
                }
            } catch (InvalidMoveException e) {
                System.out.println(e.getMessage());
            }
        }

        if (winner.equalsIgnoreCase("draw")) {
            System.out.println("It's a draw! Thanks for playing.");
        } else {
            System.out.println("Congratulations! " + winner + " has won!
            Thanks for playing.");
        }
        in.close();
    }
}

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

SAMPLE INPUT-OUTPUT

```
hp@hp-virtual-machine:~/java/record/cycle3/TicTacToe$ java TicTacToee
Welcome to 3*3 Tic Tac Toe.
 _____ _____ _____
|     |     |     |
|  _  |  _  |  _  |
|_____|_____|_____|
|     |     |     |
|  _  |  _  |  _  |
|_____|_____|_____|
|     |     |     |
|  _  |  _  |  _  |
|_____|_____|_____|
X will play first. Enter a slot number to place X in:
2
 _____ _____ _____
|     |     |     |
|  _  |  X  |  _  |
|_____|_____|_____|
|     |     |     |
|  _  |  _  |  _  |
|_____|_____|_____|
|     |     |     |
|  _  |  _  |  _  |
|_____|_____|_____|
O It's your turn! Enter a slot number where O in:
4
 _____ _____ _____
|     |     |     |
|  _  |  X  |  _  |
|_____|_____|_____|
|     |     |     |
|  O  |  _  |  _  |
|_____|_____|_____|
|     |     |     |
|  _  |  _  |  _  |
|_____|_____|_____|
X It's your turn! Enter a slot number where X in:
6
 _____ _____ _____
|     |     |     |
|  _  |  X  |  _  |
|_____|_____|_____|
|     |     |     |
|  O  |  _  |  X  |
|_____|_____|_____|
|     |     |     |
|  _  |  _  |  _  |
|_____|_____|_____|
```

```
_____________________________
O It's your turn! Enter a slot number where O in:
8

_____________________________
|   _   |   X   |   _   |
_____________________________
|   O   |   _   |   X   |
_____________________________
|   _   |   O   |   _   |
_____________________________
X It's your turn! Enter a slot number where X in:
3

_____________________________
|   _   |   X   |   X   |
_____________________________
|   O   |   _   |   X   |
_____________________________
|   _   |   O   |   _   |
_____________________________
O It's your turn! Enter a slot number where O in:
1

_____________________________
|   O   |   X   |   X   |
_____________________________
|   O   |   _   |   X   |
_____________________________
|   _   |   O   |   _   |
_____________________________
X It's your turn! Enter a slot number where X in:
5
```

SAMPLE TEST CASE

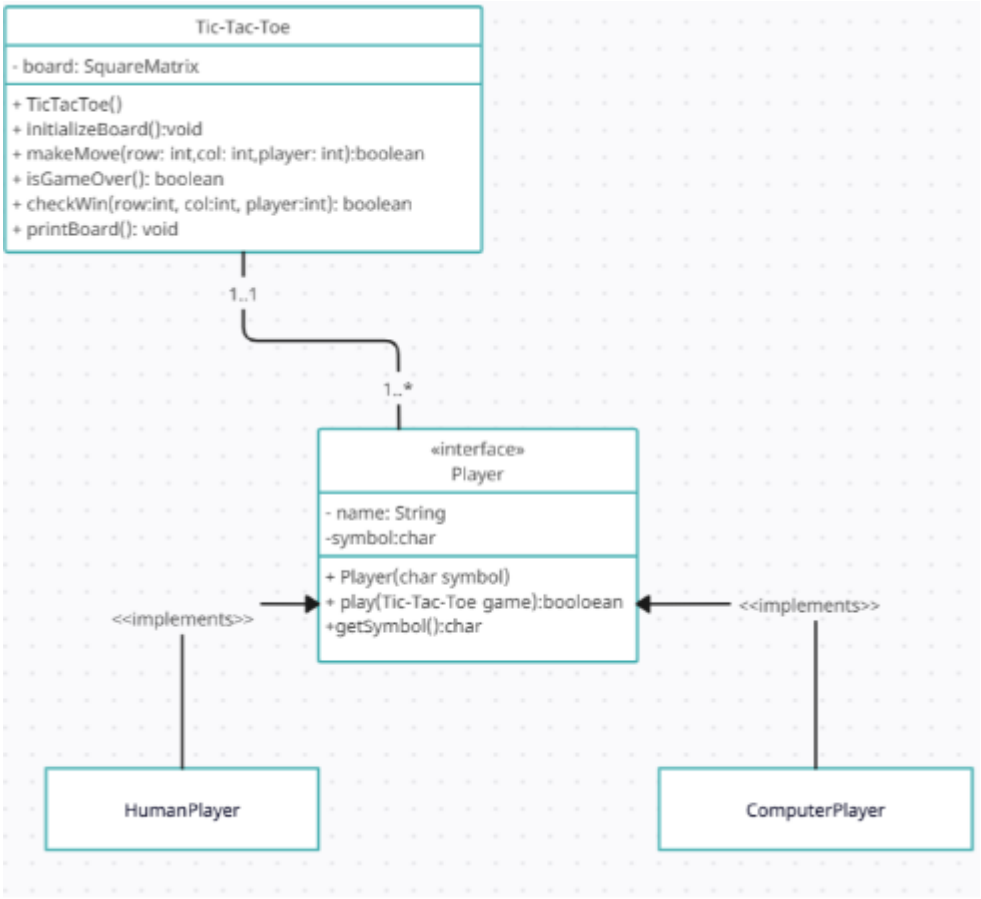
SL NO	DESCRIPTION	INPUT	OUTPUT	PASS/FAIL
1	Player 0	2 1	– – – – – – – 0 –	PASS
2	Player X	2 1	Invalid Move	PASS

10.TIC-TAC-TOE Human Vs Computer

AIM

Create an interface Player as shown in the figure. The play() method for HumanPlayer should display the board and allow the user to enter his move. The play() method in ComputerPlayer shall scan the board for empty cells and randomly choose one to play. The play() method should use the already existing placeMark() method to make a move.

UML



PROGRAM

```
import java.util.Scanner;
import java.util.Random;

interface Player {
    void play(TicTacToe game);
}

class HumanPlayer implements Player {
    @Override
```

```
    public void play(TicTacToe game) {
        Scanner scanner = new Scanner(System.in);
        game.printBoard();
        System.out.println("Player"+game.getCurrentPlayer()+" ,
        enter your move :");
        int numInput = scanner.nextInt();
        int row = (numInput - 1) / 3;
        int col = (numInput - 1) % 3;
        try {
            game.MakeMove(row, col, game.turn.equals("X") ? -1 : 1);
        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
        }
    }
}

class ComputerPlayer implements Player {
    @Override
    public void play(TicTacToe game) {
        Random random = new Random();
        int row, col;
        do {
         System.out.println("Player"+game.getCurrentPlayer()+" ,
         enter your move : ");
            row = random.nextInt(3);
            col = random.nextInt(3);
        } while (game.board.getElement(row, col) != 0);

        try {
            game.MakeMove(row, col, game.turn.equals("O") ? -1 : 1);
        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
        }
    }
}

public class TicTacToeAI{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        TicTacToe game = new TicTacToe();
        int row = 3;
```

```
        int col = 3;
        int currentPlayer =-1;
        Player player1 = new HumanPlayer();
        Player player2 = new ComputerPlayer();
        boolean gameEnd = false;
        String winner = null;

        System.out.println("Welcome to 3*3 Tic Tac Toe.");
        game.printBoard();
        System.out.println("X will play first.");

        while (!gameEnd) {
            player1.play(game);
            game.printBoard();
            if (game.checkWin(row,col,game.turn.equals("X")? -1 : 1 ) || game.isGameOver()
                break;
            }

            player2.play(game);
            game.printBoard();
            if (game.checkWin(row,col,game.turn.equals("O") ? -1 : 1) || game.isGameOver()
                break;
            }
        }

        if (game.isGameOver()) {
            System.out.println("It's a draw! Thanks for playing.");
        } else {
            winner = game.turn.equals("X") ? "O" : "X";
            System.out.println("Congratulations! " + winner + " has won!
            Thanks for playing.");
        }
        scanner.close();
    }
}
```

SAMPLE INPUT-OUTPUT

SAMPLE TEST CASE

SL NO	DESCRIPTION	INPUT	OUTPUT	PASS/FAIL
1	Computer Player		$-\ -\ -$ $-\ O\ -$ $-\ -\ -$	PASS
2	Human Player	9	$-\ -\ -$ $-\ O\ -$ $-\ -\ X$	PASS

11.ABSTRACT CLASS AND METHODS

AIM

Write a Java program to create an abstract class Employee with abstract methods calculateSalary() and displayInfo(). Create subclasses Manager and Programmer that extend the Employee class and implement the respective methods to calculate salary and display information for each role. 1. Demonstrate these methods by creating one object of each subclass. 2. Also, create a listEmployees(Employee[] emp) method in your Main class, that displays the Name and Salary of every employee as a table and pass an array containing both Managers and Programmers to this method.

PROGRAM

```
import java.util.Scanner;

abstract class Employee {
    Scanner in = new Scanner(System.in);
    private int empid;
    private String empname;
    private double salary;

    public Employee(int id, String name, double Sal) {
        empid = id;
        empname = name;
        salary = Sal;
    }

    abstract double CalculateSalary();

    abstract void DisplayInfo();

    abstract void enterInfo();

    public int getId() {
        return empid;
    }

    public void setId(int id) {
        empid = id;
    }
```

```
    public String getName() {
        return empname;
    }

    public void setName(String name) {
        empname = name;
    }

    public double getSal() {
        return salary;
    }

    public void setSal(double sal) {
        salary = sal;
    }
}

class Manager extends Employee {
    private double allowance = 0.5;

    Manager(int id, String name, double sal) {
        super(id, name, sal);
    }

    public double CalculateSalary() {
        double Totsal = allowance * super.getSal() + super.getSal();
        return Totsal;
    }

    public void DisplayInfo() {
        System.out.println("*********************************");
        System.out.println("Employee Post : Manager");
        System.out.println("Manager Id : " + super.getId());
        System.out.println("Manager Name : " + super.getName());
        double Tot = CalculateSalary();
        System.out.println("Manager Total salary " + Tot);
        System.out.println("*********************************");
    }

    public void enterInfo() {
        System.out.println("Enter the Employee Id : ");
```

```
        int id = in.nextInt();
        super.setId(id);
        System.out.println("Enter the Employee Name : ");
        String Name = in.next();
        super.setName(Name);
        System.out.println("Enter Employee  Salary : ");
        double Sal = in.nextDouble();
        super.setSal(Sal);
    }
}

class Programmer extends Employee {
    private double allowance;

    Programmer(int id, String name, double sal) {
        super(id, name, sal);
    }

    public double CalculateSalary() {
        allowance = 0.7;
        double Totsal = allowance * super.getSal() + super.getSal();
        return Totsal;
    }

    public void enterInfo() {
        System.out.println("Enter the Employee Id : ");
        int id = in.nextInt();
        super.setId(id);
        System.out.println("Enter the Employee Name : ");
        String Name = in.next();
        super.setName(Name);
        System.out.println("Enter Employee  Salary : ");
        double Sal = in.nextDouble();
        super.setSal(Sal);
    }

    public void DisplayInfo() {
        System.out.println("**********************************");
        System.out.println("Employee Post : Programmer");
        System.out.println("Programmer Id : " + super.getId());
        System.out.println("Programmer Name : " + super.getName());
```

```
        double Tots = CalculateSalary();
        System.out.println("Programmer Total Sal " + Tots);
        System.out.println("**********************************");
    }
}

public class Main {
    public static void EmployeeList(Employee[] emp) {
        System.out.println("NAME\t\tSALARY");
        int len = emp.length;
        for (int i = 0; i < len; i++) {
            System.out.println(emp[i].getName() + "\t\t" + emp[i].CalculateSalary());
        }
    }

    public static void main(String[] args) {
        Scanner obj = new Scanner(System.in);
        int limit;
        System.out.println("Enter the Number of employees :");
        limit = obj.nextInt();
        String choice;
        Employee[] e = new Employee[limit];
        for (int i = 0; i < limit; i++) {
            System.out.println("Enter your choice\n");
            System.out.println("M : Manager\nP: Programmer");
            choice = obj.next();
            if (choice.equals("M")) {
                Manager M1 = new Manager(0, "", 0);
                M1.enterInfo();
                e[i] = M1;
                M1.DisplayInfo();
            }

            if (choice.equals("P")) {
                Programmer P1 = new Programmer(0, "", 0);
                P1.enterInfo();
                e[i] = P1;
                P1.DisplayInfo();
            }
        }
        EmployeeList(e);
```

```
    }
}
```

SAMPLE INPUT-OUTPUT

```
hp@hp-virtual-machine:~/java/record/cycle3/Qn4$ java Main
Enter the Number of employees :
3
Enter your choice

M : Manager
P: Programmer
P
Enter the Employee Id :
1234
Enter the Employee Name :
Kiran
Enter Employee  Salary :
42000
**********************************
Employee Post : Programmer
Programmer Id : 1234
Programmer Name : Kiran
Programmer Total Sal 71400.0
**********************************
Enter your choice

M : Manager
P: Programmer
P
Enter the Employee Id :
2354
Enter the Employee Name :
Liya
Enter Employee  Salary :
85000
**********************************
Employee Post : Programmer
Programmer Id : 2354
Programmer Name : Liya
Programmer Total Sal 144500.0
```

```
*********************************
Employee Post : Programmer
Programmer Id : 2354
Programmer Name : Liya
Programmer Total Sal 144500.0
*********************************
Enter your choice

M : Manager
P: Programmer
M
Enter the Employee Id :
4563
Enter the Employee Name :
Kaithi
Enter Employee  Salary :
78200
*********************************
Employee Post : Manager
Manager Id : 4563
Manager Name : Kaithi
Manager Total salary 117300.0
*********************************
NAME            SALARY
Kiran           71400.0
Liya            144500.0
Kaithi          117300.0
```

SAMPLE TEST CASE

SL NO	DESCRIPTION	INPUT	OUTPUT	PASS/FAIL
1	No of Employee	3		PASS
2	Employee Information	$Programmer$ 1234 $Kiran$ 42000	$Employeetype : -Programmer$ $ProgrammerId : -1234 Name : -Kiran$ $TotalSalary : -71400.0$	PASS
3	”	$Programmer$ 2354 $Liya$ 850000	$Employeetype : -Programmer$ $Programmerid : -2354 Name : -Liya$ $TotalPay : -144500.0$	PASS
4	Employee List	$Manager$ 4563 $Kaithi$ 78200	$EmployeeType : -Manager$ $Managerid : -4563$ $Name : -Kaithi$ $TotalSalary : -117300.0$	PASS
5	”	$Programmer$ $AKHIL$ 70000	$Name : -AKHIL$ $Salary : -69825.00$	PASS

12.MULTIPLE INHERITANCE AND INTERFACE

AIM

Create an interface called PaymentMethod with abstract methods pay() and cancel(). Create classes CreditCard and PayPal that implement the PaymentMethod interface and provide their implementations of the pay() and cancel() methods. Demonstrate their working with appropriate data.

PROGRAM

```
import java.util.Scanner;

// PaymentMethod interface
interface PaymentMethod {
    void pay(double amount);
    void cancel(double amount);
}

// CreditCard class implementing PaymentMethod interface
class CreditCard implements PaymentMethod {
    private String cardNumber;
    private String expiryDate;
    private String cvv;

    public CreditCard(String cardNumber, String expiryDate, String cvv) {
        this.cardNumber = cardNumber;
        this.expiryDate = expiryDate;
        this.cvv = cvv;
    }

    public void pay(double amount) {
        // Implementation of paying with credit card
        System.out.println("Paid $" + amount + " with credit card " + cardNumber);
    }

    public void cancel(double amount) {
        // Implementation of canceling payment with credit card
        System.out.println("Cancelled payment of $" + amount +
        " with credit card " + cardNumber);
    }
}
```

```
// PayPal class implementing PaymentMethod interface
class PayPal implements PaymentMethod {
    private String email;
    private String password;

    public PayPal(String email, String password) {
        this.email = email;
        this.password = password;
    }

    public void pay(double amount) {
        // Implementation of paying with PayPal
        System.out.println("Paid $" + amount +
        " with PayPal account " + email);
    }

    public void cancel(double amount) {
        // Implementation of canceling payment with PayPal
        System.out.println("Cancelled payment of $" + amount +
        " with PayPal account " + email);
    }
}

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

        while (runAgain) {
            // Prompt user to choose payment method
            System.out.println("Choose payment method:");
            System.out.println("1. Credit Card");
            System.out.println("2. PayPal");
            int choice = scanner.nextInt();

            PaymentMethod paymentMethod;

            // Based on user choice, create instance of selected payment method
            if (choice == 1) {
                System.out.println("Enter Credit Card details:");
```

```
            System.out.print("Card Number: ");
            String cardNumber = scanner.next();
            System.out.print("Expiry Date: ");
            String expiryDate = scanner.next();
            System.out.print("CVV: ");
            String cvv = scanner.next();
            paymentMethod = new CreditCard(cardNumber, expiryDate, cvv);
        } else if (choice == 2) {
            System.out.println("Enter PayPal details:");
            System.out.print("Email: ");
            String email = scanner.next();
            System.out.print("Password: ");
            String password = scanner.next();
            paymentMethod = new PayPal(email, password);
        } else {
            System.out.println("Invalid choice. Exiting...");
            return;
        }

        // Prompt user to select pay or cancel
        System.out.println("Select action:");
        System.out.println("1. Pay");
        System.out.println("2. Cancel");
        int action = scanner.nextInt();

        // Perform pay or cancel based on user selection
        if (action == 1) {
            System.out.print("Enter amount to pay: ");
            double amount = scanner.nextDouble();
            paymentMethod.pay(amount);
        } else if (action == 2) {
            System.out.print("Enter amount to cancel: ");
            double amount = scanner.nextDouble();
            paymentMethod.cancel(amount);
        } else {
            System.out.println("Invalid action. Exiting...");
            return;
        }

        // Prompt user to go to next payment or exit
        System.out.println("Do you want to go to the next payment? (Y/N)");
```

```
            String nextPayment = scanner.next();
            if (!nextPayment.equalsIgnoreCase("Y")) {
                runAgain = false;
            }
        }

        scanner.close();
    }
}
```

SAMPLE INPUT-OUTPUT

```
hp@hp-virtual-machine:~/java/record/cycle3/Qn5$ java Main
Choose payment method:
1. Credit Card
2. PayPal
1
Enter Credit Card details:
Card Number: 123456789
Expiry Date: 01/04
CVV: 1984
Select action:
1. Pay
2. Cancel
1
Enter amount to pay: 10000
Paid $10000.0 with credit card 123456789
Do you want to go to the next payment? (Y/N)
Y
Choose payment method:
1. Credit Card
2. PayPal
2
Enter PayPal details:
Email: mhgicc@kla.com
Password: 564789
Select action:
1. Pay
2. Cancel
2
Enter amount to cancel: 450
Cancelled payment of $450.0 with PayPal account mhgicc@kla.com
Do you want to go to the next payment? (Y/N)
N
```

SAMPLE TEST CASE

SL NO	DESCRIPTION	INPUT	OUTPUT	PASS/FAIL
1	1.$CreditCard$ 2.$PayPal$	1 $cardno$: −123456789 $Expiry$: −01/04 cvv : −1984 $Amount$: −10000 Pay	Payment Successful	PASS
2	1.$CreditCard$ 2.$PayPal$	2 $email$: $-mhgicc@kla.com$ $password$: −564789 $amount$: −4501	Payment successful	PASS

13.MULTITHREADING

AIM

Write a program that demonstrates the problem of resource conflict. Suppose that you create and launch one hundred threads, each of which adds 1 rupee to an account. Assume that the account is initially empty. Also, demonstrate how this can be resolved using synchronized block or synchronized method. The main thread should wait for all the hundred threads to finish, and then print the final balance.

PROGRAM

```
   public class WithoutSynchronised {
    public static void main(String[] args) {
        // Create a shared bank account
        BankAccount account = new BankAccount();

        // Create and start 100 threads, each adding 1 rupee to the account
        for (int i = 0; i < 100; i++) {
            Thread thread = new Thread(() -> {
                account.addMoney(1);
            });
            thread.start();
        }

        // Wait for all threads to finish
        try {
            Thread.sleep(2000); // wait for 2 seconds to ensure all
            threads have finished
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        // Print the final balance
        System.out.println("Final balance: " + account.getBalance());
    }
}

class BankAccount {
    private int balance;

    public BankAccount() {
```

```
        this.balance = 0;
    }

    // Method to add money to the account
    public void addMoney(int amount) {
        // Simulate some processing time
        try {
            Thread.sleep(10);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        balance += amount;
    }

    // Method to get the current balance
    public int getBalance() {
        return balance;
    }
}

public class WithSynchronised1 {
    public static void main(String[] args) {
        // Create a shared bank account
        BankAccount account = new BankAccount();

        // Create and start 100 threads, each adding 1 rupee to the account
        for (int i = 0; i < 100; i++) {
            Thread thread = new Thread(() -> {
                account.addMoney(1);
            });
            thread.start();
        }

        // Wait for all threads to finish
        try {
            Thread.sleep(2000); // wait for 2 seconds to ensure all
            threads have finished
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
```

```
        // Print the final balance
        System.out.println("Final balance: " + account.getBalance());
    }
}

class BankAccount {
    private int balance;

    public BankAccount() {
        this.balance = 0;
    }

    // Synchronized method to add money to the account
    public synchronized void addMoney(int amount) {
        // Simulate some processing time
        try {
            Thread.sleep(10);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        balance += amount;
    }

    // Method to get the current balance
    public int getBalance() {
        return balance;
    }
}
public class WithSynchronised2 {
    public static void main(String[] args) {
        // Create a shared bank account
        BankAccount account = new BankAccount();

        // Create and start 100 threads, each adding 1 rupee to the account
        for (int i = 0; i < 100; i++) {
            Thread thread = new Thread(() -> {
                account.addMoney(1);
            });
            thread.start();
        }
```

```
        // Wait for all threads to finish
        try {
            Thread.sleep(2000); // wait for 2 seconds to ensure all
            threads have finished
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        // Print the final balance
        System.out.println("Final balance: " + account.getBalance());
    }
}

class BankAccount {
    private int balance;

    public BankAccount() {
        this.balance = 0;
    }

    // Method to add money to the account
    public void addMoney(int amount) {
        synchronized (this) {
            // Simulate some processing time
            try {
                Thread.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            balance += amount;
        }
    }

    // Method to get the current balance
    public int getBalance() {
        return balance;
    }
}
```

SAMPLE INPUT-OUTPUT

1.

```
hp@hp-virtual-machine:~/java/record/cycle4/Qn1$ java WithoutSynchronised
Final balance: 99
```

2.

```
hp@hp-virtual-machine:~/java/record/cycle4/Qn1$ java WithSynchronised1
Final balance: 100
```

SAMPLE TEST CASE

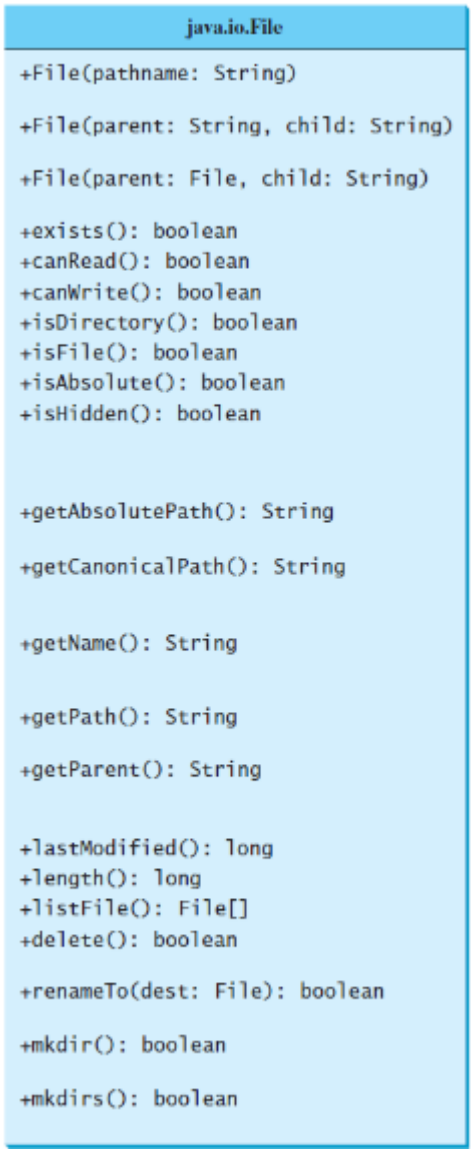
SL NO	DESCRIPTION	INPUT	OUTPUT	PASS
1	Balance		999	PASS
2	Final Balance		100	PASS

14.FILE CLASS AND METHODS

AIM

Write a Java program that demonstrates how to create files in a platform-independent way and use the methods in the File class to obtain their properties. Refer to File class shown in above figure.

UML



PROGRAM

```
import java.io.File;
import java.io.IOException;

public class FileDemo {
```

```
public static void main (String[] args ){
               // Create a new file
  File file = new File("test.txt");
  try {
  // Create the file (if it doesn't exist)
  if (file.createNewFile()) {
  System.out.println("File created: " +  file.getName());
  } else {
  System.out.println("File already exists.");
  }

  // Get file properties
  System.out.println("Absolute path: " + file.getAbsolutePath());
  System.out.println("Canonical path " + file.getCanonicalPath() );
  System.out.println("File name: " + file.getName());
  System.out.println("File path: " + file.getPath());
  System.out.println("Parent directory: " + file.getParent());
  System.out.println("File exists: " + file.exists());
  System.out.println("Is directory: " + file.isDirectory());
  System.out.println("Is file: " + file.isFile());
  System.out.println("File length: " + file.length() + " bytes");
  System.out.println("can Read " + file.canRead() );
  System.out.println("can Write " + file.canWrite() );
  System.out.println("is Absolute " + file.isAbsolute() );
  System.out.println("is Hidden " + file.isHidden() );
  } catch(IOException e){
  System.out.println("An error occurred while creating the file: " + e.getMessage());
  }
  // Delete the file
  if (file.delete()) {
  System.out.println("File deleted.");
  } else {
  System.out.println("Failed to delete the file.");
  }
      }
 }
```

SAMPLE INPUT-OUTPUT

```
hp@hp-virtual-machine:~/java/record/cycle4/Qn2$ java FileDemo
File created: test.txt
Absolute path: /home/hp/java/record/cycle4/Qn2/test.txt
Canonical path /home/hp/java/record/cycle4/Qn2/test.txt
File name: test.txt
File path: test.txt
Parent directory: null
File exists: true
Is directory: false
Is file: true
File length: 0 bytes
can Read true
can Write true
is Absolute false
is Hidden false
File deleted.
```

15.FILE - reading

AIM

Write a Java program to get filename of a text file as command-line argument and print its content on the screen.

PROGRAM

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
public class ReadTextFile {
 public static void main(String[] args) {
 // Check if the filename is provided as a command-line argument
 if (args.length != 1) {
     System.out.println("Usage: java ReadFile <filename>");
 return;
 }

// Get the filename from the command-line argument
 String filename = args[0];
 // Try to read the file and print its content
 try (Scanner scanner = new Scanner(new File(filename))) {
 System.out.println("Contents of " + filename + ":");
 while (scanner.hasNextLine()) {
 System.out.println(scanner.nextLine());
 }
 } catch (FileNotFoundException e) {
 System.out.println("File not found: " + filename);
 }
 }
}
```

SAMPLE INPUT-OUTPUT

```
hp@hp-virtual-machine:~/java/record/cycle4/Qn2$ javac ReadTextFile.java
hp@hp-virtual-machine:~/java/record/cycle4/Qn2$ java ReadTextFile myfile.txt
Contents of myfile.txt:
This is a sample Program
```

16.FILE- writing

AIM

Write a program that reads n lines from the user and stores it in a file "sentences.txt"

PROGRAM

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

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

        System.out.print("Enter the number of lines you want to write: ");
        int n = scanner.nextInt();
        scanner.nextLine(); // consume the newline character

        try (BufferedWriter writer = new BufferedWriter(new FileWriter("sentences.txt")))
            System.out.println("Enter " + n + " lines:");

            for (int i = 0; i < n; i++) {
                String line = scanner.nextLine();
                writer.write(line);
                writer.newLine();
            }

            System.out.println("Lines written to sentences.txt successfully.");
        } catch (IOException e) {
            System.out.println("An error occurred while writing to the file: "
            + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```

SAMPLE INPUT-OUTPUT

```
hp@hp-virtual-machine:~/java/record/cycle4/Qn2$ java WriteFile
Enter the number of lines you want to write: 3
Enter 3 lines:
can I
Write to
this File?
Lines written to sentences.txt successfully.
```

17.File-Overwriting

AIM

Modify the Tic-Tac-Toe program to log all the moves into a file "tic-tac-toe-dd-mm-yyyy.log" in the form "Marker row colum" or "X invalid move", where dd-mm-yyyy is the date, Marker can be either 0 or X. Note that, if a file already exists, you need to append to the existing content, else create a new file.

PROGRAM

```
import java.io.FileWriter;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;

public class TicTacToe {
    private SquareMatrix board;
    private FileWriter logFileWriter;

    public TicTacToe() {
        board = new SquareMatrix(3);
        try {
            SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");
            String fileName = "tic-tac-toe-" + dateFormat.format(new Date()) + ".log";
            logFileWriter = new FileWriter(fileName);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public boolean makeMove(int row, int col, int turn) {
        if (row < 0 || row >= 3 || col < 0 || col >= 3
        || board.getElement(row, col) != 0)
            return false;

        board.setElement(row, col, turn);
        logMove(turn, row, col, false);
        return true;
    }
```

```
    private void logMove(int turn, int row, int col, boolean invalid) {
        try {
            char marker = (turn == -1) ? 'X' : 'O';
            String move = invalid ? "invalid move" : "";
            String logEntry = marker + " " + row + " " + col + " " + move + "\n";
            logFileWriter.write(logEntry);
            logFileWriter.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public boolean isGameOver() {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                if (board.getElement(i, j) == 0) {
                    return false;
                }
            }
        }
        return true;
    }

    public boolean checkWinner(int row, int col, int turn) {
        int sumrow = 0;
        int sumcol = 0;
        int sumdiagonal1 = 0;
        int sumdiagonal2 = 0;
        for (int i = 0; i < 3; i++) {
            sumrow += board.getElement(row, i);
            sumcol += board.getElement(i, col);
        }
        if (row == col) {
            for (int i = 0; i < 3; i++) {
                sumdiagonal1 += board.getElement(i, i);
            }
        }
        if (row + col == 2) {
            for (int i = 0; i < 3; i++) {
                sumdiagonal2 += board.getElement(i, 2 - i);
            }
```

```
        }

        if (sumrow == 3 || sumrow == -3 || sumcol == 3 || sumcol == -3 ||
        sumdiagonal1 == 3 || sumdiagonal1 == -3 || sumdiagonal2 == 3 ||
        sumdiagonal2 == -3) {
            return true;
        }
        return false;
    }

    public void printBoard() {
        System.out.println();
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                if (board.getElement(i, j) == -1) {
                    System.out.print("X ");
                } else if (board.getElement(i, j) == 1) {
                    System.out.print("O ");
                } else {
                    System.out.print("_ ");
                }
            }
            System.out.println();
        }
    }

    public static void main(String[] args) {
        TicTacToe game = new TicTacToe();
        Scanner scanner = new Scanner(System.in);
        int currentPlayer = -1;
        boolean gameEnd = false;

        while (!gameEnd) {
            game.printBoard();
            System.out.println((currentPlayer == -1 ? "X" : "O") + "'s turn.");
            System.out.print("Enter row & column numbers (0-2): ");
            int row = scanner.nextInt();
            int col = scanner.nextInt();
            if (!game.makeMove(row, col, currentPlayer)) {
                System.out.println("Invalid move. Try again.");
                continue;
```

```
            }
            if (game.checkWinner(row, col, currentPlayer)) {
                System.out.println("Player " +
                (currentPlayer == -1 ? "X" : "O") + " wins!");
                gameEnd = true;
            } else if (game.isGameOver()) {
                System.out.println("It's a draw!");
                gameEnd = true;
            }

            currentPlayer *= -1;
        }

        scanner.close();
    }
}
```

SAMPLE INPUT-OUTPUT

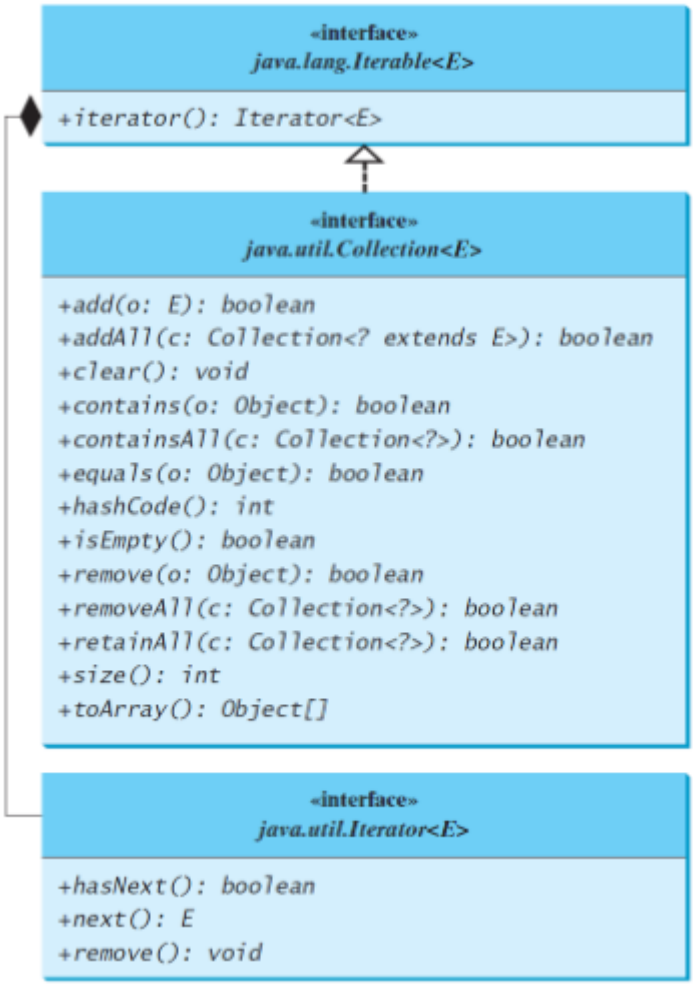
```
hp@hp-virtual-machine:~/java/record/cycle3/TicTacToe$ java TicTacToeFile

_ _ _
_ _ _
_ _ _
X's turn.
Enter row (0,1 or 2): 2
Enter column (0,1 or 2): 0

_ _ _
_ _ _
X _ _
O's turn.
Enter row (0,1 or 2): 1
Enter column (0,1 or 2): 1

_ _ _
_ O _
X _ _
X's turn.
Enter row (0,1 or 2): 0
Enter column (0,1 or 2): 0

X _ _
_ O _
X _ _
O's turn.
Enter row (0,1 or 2): 1
Enter column (0,1 or 2): 0

X _ _
O O _
X _ _
X's turn.
Enter row (0,1 or 2): 2
Enter column (0,1 or 2): 1

X _ _
O O _
X X _
O's turn.
Enter row (0,1 or 2): 1
Enter column (0,1 or 2): 2
Player O wins!
1X 2 0
2O 1 1
3X 0 0
4O 1 0
5X 2 1
6O 1 2
```

18.JAVA LIBRARY-Collections framework (HashSet)

AIM

Write a program demonstrating methods in Java's java.util.Collection Interface using a HashSet object..

UML



PROGRAM

```
import java.util.HashSet;
import java.util.Iterator;

public class CollectionDemo {
    public static void main(String[] args) {
        // Create a HashSet to demonstrate Collection methods
        HashSet<String> carSet1 = new HashSet<>();
        HashSet<String> carSet2 = new HashSet<>();
```

```
        // Adding elements to the HashSet
        carSet1.add("Toyota");
        carSet1.add("Honda");
        carSet1.add("BMW");
        carSet1.add("Ford");

        // Displaying elements in carSet1
        System.out.println("Elements in carSet1: " + carSet1);

        // Demonstrating remove() method
        carSet1.remove("BMW");
        System.out.println("After removal, carSet1: " + carSet1);

        // Demonstrating isEmpty() method
        System.out.println("Is carSet1 empty? " + carSet1.isEmpty());

        // Demonstrating elements in the HashSet
        System.out.println("Elements in carSet1:");
        Iterator<String> iterator1 = carSet1.iterator();
        while (iterator1.hasNext()) {
            System.out.println(iterator1.next());
        }

        // Example for contains method
        System.out.println("Does carSet1 contain 'Toyota'? " +
        carSet1.contains("Toyota"));

        // Demonstrating size() method
        System.out.println("Size of carSet1: " + carSet1.size());

        // Demonstrating clear() method
        carSet1.clear();
        System.out.println("After clearing, carSet1: " + carSet1);

        // Adding elements to carSet1 after clearing
        carSet1.add("Mercedes");
        carSet1.add("Audi");
        System.out.println("After adding, carSet1: " + carSet1);
    }
}
```

SAMPLE INPUT-OUTPUT

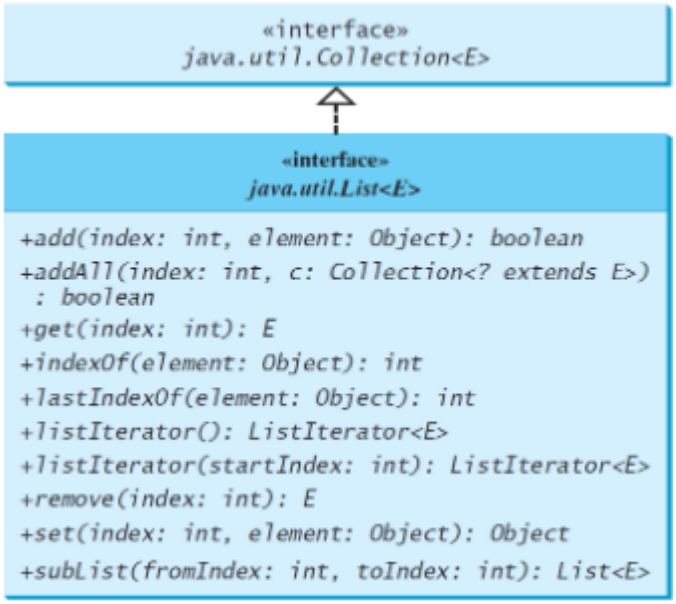
```
hp@hp-virtual-machine:~/java/record/cycle5/Qn1$ java CollectionDemo
Elements in carSet1: [Toyota, Ford, BMW, Honda]
After removal, carSet1: [Toyota, Ford, Honda]
Is carSet1 empty? false
Elements in carSet1:
Toyota
Ford
Honda
Does carSet1 contain 'Toyota'? true
Size of carSet1: 3
After clearing, carSet1: []
After adding, carSet1: [Audi, Mercedes]
```

19.JAVA LIBRARY-Collections framework (Arraylist)

AIM

Write a program demonstrating Java's java.util.List Interface methods using an ArrayList object. Also, print the elements of the ArrayList using a java.util.ListIterator.

UML



PROGRAM

```
import java.util.ArrayList;
import java.util.List;
import java.util.ListIterator;

public class ListExample {
    public static void main(String[] args) {
        // Creating an ArrayList
        List<String> arrayList = new ArrayList<>();

        // Adding elements to the ArrayList
        arrayList.add("Apple");
        arrayList.add("Banana");
        arrayList.add("Cherry");
        arrayList.add("Date");
        arrayList.add("Fig");

        // Printing the ArrayList
```

```
        System.out.println("ArrayList elements:");
        System.out.println(arrayList);

        // Demonstration of List interface methods
        System.out.println("\nList Interface Methods:");

        // 1. Using size() method
        System.out.println("Size of the ArrayList: " + arrayList.size());

        // 2. Using get() method
        System.out.println("Element at index 2: " + arrayList.get(2));

        // 3. Using set() method
        arrayList.set(1, "Blueberry");
        System.out.println("ArrayList after setting element at index 1 to 'Blueberry':");
        System.out.println(arrayList);

        // 4. Using remove() method
        arrayList.remove(3);
        System.out.println("ArrayList after removing element at index 3:");
        System.out.println(arrayList);

        // 5. Using contains() method
        System.out.println("Does ArrayList contain 'Apple'? " +
        arrayList.contains("Apple"));
        System.out.println("Does ArrayList contain 'Grape'? " +
        arrayList.contains("Grape"));

        // Printing elements using ListIterator
        System.out.println("\nPrinting elements using ListIterator:");
        ListIterator<String> listIterator = arrayList.listIterator();
        while (listIterator.hasNext()) {
            System.out.println(listIterator.next());
        }
    }
}
```

SAMPLE INPUT-OUTPUT

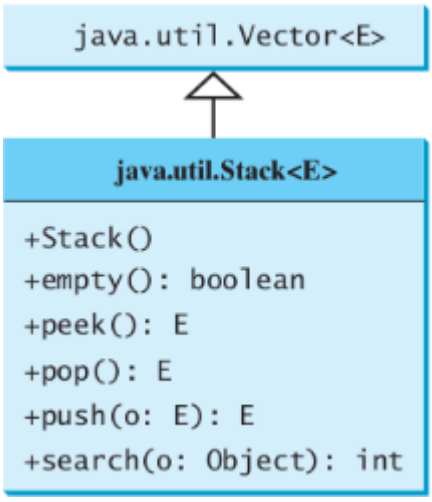
```
hp@hp-virtual-machine:~/java/record/cycle5/Qn2$ java ListExample
ArrayList elements:
[Apple, Banana, Cherry, Date, Fig]

List Interface Methods:
Size of the ArrayList: 5
Element at index 2: Cherry
ArrayList after setting element at index 1 to 'Blueberry':
[Apple, Blueberry, Cherry, Date, Fig]
ArrayList after removing element at index 3:
[Apple, Blueberry, Cherry, Fig]
Does ArrayList contain 'Apple'? true
Does ArrayList contain 'Grape'? false

Printing elements using ListIterator:
Apple
Blueberry
Cherry
Fig
```

20.JAVA LIBRARY-Collections framework (Stack)

AIM

Demonstrate the working of stack data structure using java.util.Stack class.

UML



PROGRAM

```
import java.util.Stack;

public class StackExample {
    public static void main(String[] args) {
        // Creating a stack
        Stack<Integer> stack = new Stack<>();

        // Pushing elements onto the stack
        stack.push(10);
        stack.push(20);
        stack.push(30);
        stack.push(40);
        stack.push(50);

        // Displaying the stack
        System.out.println("Stack elements: " + stack);

        // Popping elements from the stack
        System.out.println("Popped element: " + stack.pop());
        System.out.println("Popped element: " + stack.pop());
```

```
        // Displaying the stack after popping
        System.out.println("Stack elements after popping: " + stack);

        // Peeking at the top element of the stack
        System.out.println("Top element of the stack: " + stack.peek());

        // Checking if the stack is empty
        System.out.println("Is stack empty? " + stack.isEmpty());

        // Getting the size of the stack
        System.out.println("Size of the stack: " + stack.size());
    }
}
```

SAMPLE INPUT-OUTPUT

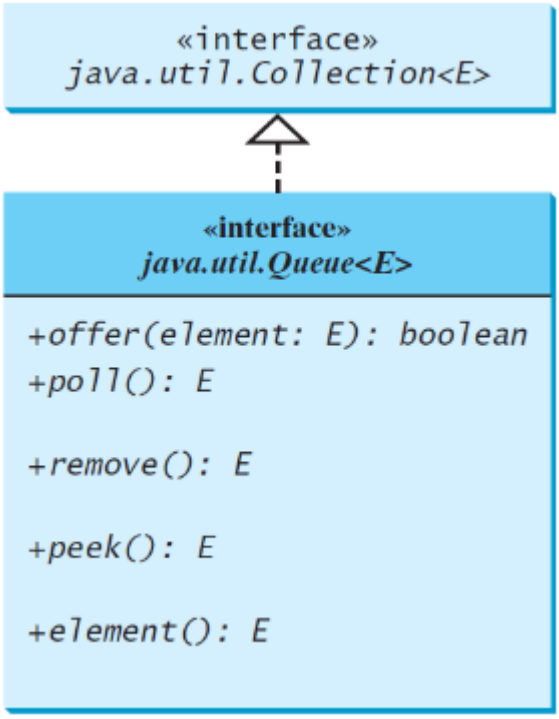
```
hp@hp-virtual-machine:~/java/record/cycle5/Qn3$ java StackExample
Stack elements: [10, 20, 30, 40, 50]
Popped element: 50
Popped element: 40
Stack elements after popping: [10, 20, 30]
Top element of the stack: 30
Is stack empty? false
Size of the stack: 3
```

21.JAVA LIBRARY-Collections framework (Queue)

AIM

Demonstrate the methods in java.util.Queue Interface using a LinkedList class.

UML



PROGRAM

```
import java.util.LinkedList;
import java.util.Queue;

public class QueueExample {
    public static void main(String[] args) {
        // Creating a Queue using LinkedList
        Queue<String> queue = new LinkedList<>();

        // Adding elements to the queue
        queue.add("Apple");
        queue.add("Banana");
        queue.add("Cherry");
        queue.add("Date");
        queue.add("Fig");
```

```
// Displaying the queue
System.out.println("Queue elements: " + queue);

// Demonstration of Queue interface methods
System.out.println("\nQueue Interface Methods:");

// 1. Using add() method
queue.add("Grape");
System.out.println("Queue after adding 'Grape': " + queue);

// 2. Using offer() method
boolean isOffered = queue.offer("Honeydew");
System.out.println("Is 'Honeydew' offered successfully? " + isOffered);
System.out.println("Queue after offering 'Honeydew': " + queue);

// 3. Using remove() method
String removedElement = queue.remove();
System.out.println("Removed element from the queue: " + removedElement);
System.out.println("Queue after removing an element: " + queue);

// 4. Using poll() method
String polledElement = queue.poll();
System.out.println("Polled element from the queue: " + polledElement);
System.out.println("Queue after polling an element: " + queue);

// 5. Using peek() method
String peekedElement = queue.peek();
System.out.println("Peeked element from the queue: " + peekedElement);
System.out.println("Queue after peeking an element: " + queue);

// 6. Using element() method
String firstElement = queue.element();
System.out.println("First element of the queue: " + firstElement);
System.out.println("Queue after accessing the first element: " + queue);

// 7. Using size() method
System.out.println("Size of the queue: " + queue.size());

// 8. Using isEmpty() method
System.out.println("Is queue empty? " + queue.isEmpty());
```

```
    }
}
```

SAMPLE INPUT-OUTPUT

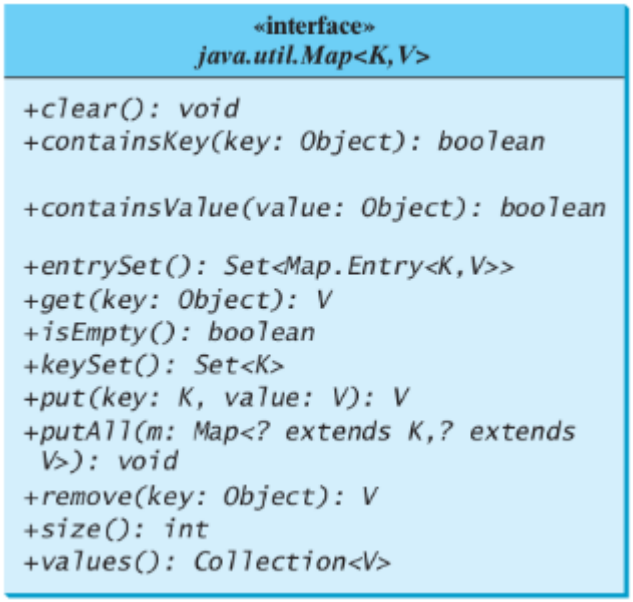
```
hp@hp-virtual-machine:~/java/record/cycle5/Qn4$ java QueueExample
Queue elements: [Apple, Banana, Cherry, Date, Fig]

Queue Interface Methods:
Queue after adding 'Grape': [Apple, Banana, Cherry, Date, Fig, Grape]
Is 'Honeydew' offered successfully? true
Queue after offering 'Honeydew': [Apple, Banana, Cherry, Date, Fig, Grape, Honeydew]
Removed element from the queue: Apple
Queue after removing an element: [Banana, Cherry, Date, Fig, Grape, Honeydew]
Polled element from the queue: Banana
Queue after polling an element: [Cherry, Date, Fig, Grape, Honeydew]
Peeked element from the queue: Cherry
Queue after peeking an element: [Cherry, Date, Fig, Grape, Honeydew]
First element of the queue: Cherry
Queue after accessing the first element: [Cherry, Date, Fig, Grape, Honeydew]
Size of the queue: 5
Is queue empty? false
```

22.JAVA LIBRARY-Collections framework (HashMap)

AIM

Demonstrate the working of a Map data structure using HashMap object.

UML



PROGRAM

```
import java.util.HashMap;
import java.util.Map;

public class MapExample {
    public static void main(String[] args) {
        // Creating a HashMap
        Map<String, Integer> hashMap = new HashMap<>();

        // Adding key-value pairs to the map
        hashMap.put("Apple", 10);
        hashMap.put("Banana", 20);
        hashMap.put("Cherry", 30);
        hashMap.put("Date", 40);
        hashMap.put("Fig", 50);

        // Displaying the map
        System.out.println("HashMap elements: " + hashMap);
```

```
        // Demonstration of Map interface methods
        System.out.println("\nMap Interface Methods:");

        // 1. Using get() method
        System.out.println("Value associated with key 'Banana': " +
        hashMap.get("Banana"));

        // 2. Using containsKey() method
        System.out.println("Does map contain key 'Cherry'? " +
        hashMap.containsKey("Cherry"));

        // 3. Using containsValue() method
        System.out.println("Does map contain value '60'? " +
        hashMap.containsValue(60));

        // 4. Using size() method
        System.out.println("Size of the map: " + hashMap.size());

        // 5. Using remove() method
        hashMap.remove("Date");
        System.out.println("HashMap after removing key 'Date': " +
        hashMap);

        // 6. Using keySet() method
        System.out.println("Keys in the map: " + hashMap.keySet());

        // 7. Using values() method
        System.out.println("Values in the map: " + hashMap.values());

        // 8. Using entrySet() method
        System.out.println("Key-Value pairs in the map: " +
        hashMap.entrySet());
    }
}
```

SAMPLE INPUT-OUTPUT

```
hp@hp-virtual-machine:~/java/record/cycle5/Qn5$ java MapExample
HashMap elements: {Apple=10, Cherry=30, Fig=50, Date=40, Banana=20}

Map Interface Methods:
Value associated with key 'Banana': 20
Does map contain key 'Cherry'? true
Does map contain value '60'? false
Size of the map: 5
HashMap after removing key 'Date': {Apple=10, Cherry=30, Fig=50, Banana=20}
Keys in the map: [Apple, Cherry, Fig, Banana]
Values in the map: [10, 30, 50, 20]
Key-Value pairs in the map: [Apple=10, Cherry=30, Fig=50, Banana=20]
```