package game;

//Abstract class demonstrating Abstraction

abstract class Game {

public abstract void start();

public abstract void play();

public abstract void end();

}

//Parent class encapsulating game logic

class Board {

private char[][] board; // Encapsulation

private final int size = 3;

public Board() {

board = new char[size][size];

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

for (int j = 0; j < size; j++) {

board[i][j] = '-';

}

}

}

public boolean makeMove(int row, int col, char player) {

if (row >= 0 && row < size && col >= 0 && col < size && board[row][col] == '-') {

board[row][col] = player;

return true;

}

return false;

}

public void displayBoard() {

for (char[] row : board) {

for (char cell : row) {

System.out.print(cell + " ");

}

System.out.println();

}

}

public boolean checkWin(char player) {

// Check rows, columns, and diagonals

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

if ((board[i][0] == player && board[i][1] == player && board[i][2] == player) ||

(board[0][i] == player && board[1][i] == player && board[2][i] == player)) {

return true;

}

}

return (board[0][0] == player && board[1][1] == player && board[2][2] == player) ||

(board[0][2] == player && board[1][1] == player && board[2][0] == player);

}

public boolean isFull() {

for (char[] row : board) {

for (char cell : row) {

if (cell == '-') {

return false;

}

}

}

return true;

}

}

//TicTacToe class inherits from Game

class TicTacToe extends Game {

private Board board;

private char currentPlayer;

public TicTacToe() {

board = new Board();

currentPlayer = 'X';

}

@Override

public void start() {

System.out.println("Welcome to Tic Tac Toe!");

board.displayBoard();

}

@Override

public void play() {

java.util.Scanner scanner = new java.util.Scanner(System.in);

while (true) {

System.out.println("Player " + currentPlayer + ", enter your move (row and column): ");

int row = scanner.nextInt();

int col = scanner.nextInt();

if (board.makeMove(row, col, currentPlayer)) {

board.displayBoard();

if (board.checkWin(currentPlayer)) {

System.out.println("Player " + currentPlayer + " wins!");

break;

}

if (board.isFull()) {

System.out.println("It's a draw!");

break;

}

currentPlayer = (currentPlayer == 'X') ? 'O' : 'X'; // Polymorphism through behavior change

} else {

System.out.println("Invalid move. Try again.");

}

}

}

@Override

public void end() {

System.out.println("Game Over!");

}

}

//Main class

public class Tic {

public static void main(String[] args) {

Game ticTacToe = new TicTacToe(); // Polymorphism: Game reference, TicTacToe object

ticTacToe.start();

ticTacToe.play();

ticTacToe.end();

 }

}