# AI Lab-2

## Harsh Parmar 9763 Batch D

### 1. Tic tac toe by Magic Force Method:

```
import java.util.Scanner;

public class TicTacToeMagicSquare {
   private static final char EMPTY = '-';
   private static final char X = 'X';
   private static final char O = 'O';

   private char[][] board;
   private char currentPlayer;

   public TicTacToeMagicSquare() {
      board = new char[3][3];
      currentPlayer = X;
      initializeBoard();
   }

   private void initializeBoard() {
      for (int i = 0; i < 3; i++) {
         for (int j = 0; j < 3; j++) {
            board[i][j] = EMPTY;
         }
      }
   }

   public void printBoard() {
      for (int i = 0; i < 3; i++) {
         for (int j = 0; j < 3; j++) {
            System.out.print(board[i][j] + " ");
         }
         System.out.println();
      }
   }

   public boolean makeMove(int row, int col) {
      if (row < 0 || row >= 3 || col < 0 || col >= 3 || board[row][col] != EMPTY) {
         return false;
      }
      board[row][col] = currentPlayer;
      return true;
   }
```

```
  public char checkWinner() {
    // Check rows and columns
    for (int i = 0; i < 3; i++) {
      if (board[i][0] != EMPTY && board[i][0] == board[i][1] && board[i][0] ==
board[i][2]) {
        return board[i][0];
      }
      if (board[0][i] != EMPTY && board[0][i] == board[1][i] && board[0][i] ==
board[2][i]) {
        return board[0][i];
      }
    }
    // Check diagonals
    if (board[0][0] != EMPTY && board[0][0] == board[1][1] && board[0][0] ==
board[2][2]) {
      return board[0][0];
    }
    if (board[0][2] != EMPTY && board[0][2] == board[1][1] && board[0][2] ==
board[2][0]) {
      return board[0][2];
    }
    // Check for a draw
    if (isBoardFull()) {
      return ' ';
    }
    return EMPTY;
  }

  public boolean isBoardFull() {
    for (int i = 0; i < 3; i++) {
      for (int j = 0; j < 3; j++) {
        if (board[i][j] == EMPTY) {
          return false;
        }
      }
    }
    return true;
  }

  public boolean isGameOver() {
    return checkWinner() != EMPTY || isBoardFull();
  }

  public void play() {
    Scanner scanner = new Scanner(System.in);

    while (!isGameOver()) {
      System.out.println("Current Board:");
```

```
        printBoard();
        System.out.println("Player " + currentPlayer + "'s turn.");
        int row, col;
        if (currentPlayer == X) {
            System.out.print("Enter row (0-2): ");
            row = scanner.nextInt();
            System.out.print("Enter column (0-2): ");
            col = scanner.nextInt();
            if (!makeMove(row, col)) {
                System.out.println("Invalid move. Try again.");
            }
        } else {
            makeAIMove();
        }
        currentPlayer = (currentPlayer == X) ? O : X;
    }

    System.out.println("Final Board:");
    printBoard();
    char winner = checkWinner();
    if (winner == EMPTY) {
        System.out.println("It's a draw!");
    } else {
        System.out.println("Player " + winner + " wins!");
    }
}

public void makeAIMove() {
    // Magic square strategy: place O in the center if available, else place in any corner
    if (board[1][1] == EMPTY) {
        makeMove(1, 1);
    } else {
        // Place O in a corner
        int[][] corners = {{0, 0}, {0, 2}, {2, 0}, {2, 2}};
        for (int[] corner : corners) {
            if (board[corner[0]][corner[1]] == EMPTY) {
                makeMove(corner[0], corner[1]);
                return;
            }
        }
        // If no corner available, place O in any empty cell
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                if (board[i][j] == EMPTY) {
                    makeMove(i, j);
                    return;
                }
            }
```

```
        }
      }
    }

    public static void main(String[] args) {
      TicTacToeMagicSquare game = new TicTacToeMagicSquare();
      game.play();
    }
}
```