NAME: ARUN SINGH DHANADI

ROLL NO: 5

DEPARTMENT: BEIT

COLLEGE: EVEREST ENGINEERING COLLEGE

LAB 01

**Practical 1: Programming Warm-Up Through the Tic-Tac-Toe Game** *Instructor: Shailesh B. Pandey (Everest Engineering College) Course: Intelligent System (BEIT)* Course webpage: sites.google.com/view/eec-is

*1. Overview*

Tic-Tac-Toe is a two-player game that is usually played on a 3x3 board (grid). As you may know, if the game is played with an ideal strategy the game always ends in a draw. In our course we will study how we can create an “intelligent” Tic-Tac-Toe playing Bot (agent) that plays the game perfectly against a Human opponent i.e. it never loses. The task in this practical is rather non-challenging. We are going to code a console-based Tic-Tac-Toe game that is played between two Human players. The objective of this practical is for you to get your programming skills up to the level that is required in this course. Take this course as an opportunity to improve your knowledge on programming language basics, data structure (particularly, graphs and trees) and implementing AI algorithms.

*2. Input and Output*

There will be two players: PLAYER\_1 (‘X’) and PLAYER\_2 (‘O’). You will ask the players to choose the location in the grid where they want to put their mark. Tic-Tac-Toe can board can be stored in a 2D array. So, you will need to get a row and column number from the standard input.

The output should look like this:

| |

-----------

| |

-----------

| |

X's turn to put a mark. Type row and col:

1

1

| |

-----------

| X |

-----------

| |

O's turn to put a mark. Type row and col:

There are three results possible: X wins or O wins or there is a draw. At the end of the game a suitable message is displayed.

*3. Tasks*

I have provided a starter code to build the game. Study the TTTGame class. It is the class that you will run to play the game. Observe how the game starts, is played and ends i.e. program flow. Try to understand what each method is doing without worrying about how it is being done. There is more than one way to do the same thing. This class utilizes the TicTacToe class to handle the game board (grid). It declares a 2D char array (matrix) to store the board information. There are variables to determine the turn of the players and the symbol (‘X’ or ‘O’) the players will use.

Part A:

Prepared by: Shailesh B. Pandey Course: Intelligent System/AI (Edited: 16 April 2024) Draft: Please do not circulate without permission. This practical is created for EEC’s use.

• Your main task is to complete the code in the TicTacToe.java file. The comments give you information on what these methods are supposed to do. After you have implemented all the empty methods, you can execute TTTGame class to play the game.

• Add code to check row and column are within the range [0,2]. If you run the current code and input a negative number, the program will crash. This is because an array does not take negative index.

Part B (Extra Marks):

• The code only works on a 3x3 grid. Modify the code to handle a n x n board. Modify starter code to handle any size of board.

• There are no standard rules to play n x n Tic-Tac-Toe. You can then implement the following rule: the game continues until all cells are filled. The winner is the one who has the greatest number of three-in-a-row lines (horizontal, vertical and diagonal) i.e. count how many three lines with three consecutive X’s or O’s each player has.

*4. Submission*

You need to submit practical reports using the provided template. The template is also available from https://bit.ly/eecispractical

Please note that plagiarism is not tolerated. You can always ask for help from friends and me when you are stuck but you cannot ask for them to write your code or use their code. If you have not written the code yourself you will not get any marks. If you have copied from your friends both will lose marks. Submission only does not give you passing marks.

*5. Starter Code*

Create a Java Project in Eclipse or IDE of your choice. Add these two classes and complete the code.

Keywords: tic tac toe, ttt in java, two player game, 2d array

1. PROBLEM DESCRIPTION

In this practical , I have to make a Tic Tac Toe game using Java. The game should allow two players to take turns placing their marks on a grid until one player wins or the game ends in a draw. It needs to manage player input, check if moves are valid, and figure out who wins.

1. BACKGROUND

Tic Tac Toe is a game played on a grid of 3x3 squares. Players take turns putting their symbol, like "X" or "O," in an empty square. The goal is to get three of your symbols in a row, either straight across, up and down, or diagonally.

It's quite simple for a computer to win at Tic Tac Toe against a human. This is because the game has a limited number of moves and outcomes, so a computer can analyze them methodically to make the best choices.

1. PROGRAM DESCRIPTION

The Tic Tac Toe program acts like the classic game where two players take turns marking empty squares on a 3x3 grid, trying to get three of their symbols in a row, column, or diagonal. It helps players by asking them to make moves, checking if their choices are okay, and showing the updated grid after each move. It also looks to see if someone has won after every turn, announcing the winner if there's a line of three symbols. If nobody wins, and the grid fills up, it says the game is a draw.

1. IMPLEMENTATION

Class A: TTTGame

This class manages the overall Tic Tac Toe game.

Method 1: startGame()

Purpose: Initiates and controls the game flow. Steps:

* 1. Displays the initial empty game board.
  2. Initiates the game.
  3. Determines the winner.
  4. Prints the result of the game (whether X wins, O wins, or it's a draw).

Method 2: playGame()

Purpose: Manages the sequence of moves in the game. Steps:

1. Initializes variables for player turns, count of moves, row, and column.
2. Enters a loop until there's a winner or the board is full (max 9 moves).
3. Prompts the current player to enter their move (row and column).
4. Validates the move (if the cell is empty).
5. Updates the board with the player's mark.
6. Displays the updated board.
7. Increments the move count.

Method 3: printMessage()

Purpose: Prints the result of the game. Steps:

1. Checks the winner.
2. Prints a message indicating the winner (X, O) or declares a draw if there's no winner.

Class B: TicTacToe

This class represents the Tic Tac Toe board and contains the game logic.

Method 1: getWinner()

Purpose: Determines if there's a winner and returns the winning player's mark. Steps:

1. Checks for winning conditions in rows, columns, and diagonals.
2. Returns the mark of the winning player or a space if there's no winner.

Method 2: displayBoard()

Purpose: Displays the current state of the game board. Steps:

1. Iterates through the board array.
2. Prints each cell's mark enclosed in curly braces.
3. Moves to the next row after printing each row.

Method 3: whoseTurn()

Purpose: Determines whose turn it is to play. Steps:

1. Toggles between 'X' and 'O' for each call.
2. Returns the current player's mark.

Method 4: putMark(int row, int col)

Purpose: Places the current player's mark on the specified cell. Steps:

1. Updates the board array with the current player's mark at the specified row and column.

Method 5: getMark(int row, int col)

Purpose: Retrieves the mark at the specified cell. Steps:

1. Returns the mark stored in the board array at the specified row and column.
2. These methods work together to implement the game mechanics and manage the state of the Tic Tac Toe game.
3. TESTS AND RESULTS

Basic Gameplay Test:

Input: Simulate a complete game with valid moves by both players.

Expected Result: The game ends with either player X or player O winning, or it ends in a draw.

Winning Conditions Test:

Input: Place marks on the board to achieve winning conditions (three in a row, column, or diagonal) for both players.

Expected Result: The game correctly identifies the winning player (X or O) or declares a draw if no winning conditions are met.

Invalid Input Test:

Input: Enter invalid row and column values (e.g., negative numbers, out-of-range values, non- integer inputs).

Expected Result: The game prompts the player to enter valid inputs and does not progress until valid inputs are provided.

Occupied Cell Test:

Input: Attempt to place a mark in a cell that is already occupied.

Expected Result: The game prompts the player to choose another cell and does not allow the move until an unoccupied cell is selected.

Game State Display Test:

Input: Display the game board at various stages of the game.

Expected Result: The game board is displayed accurately, showing the current state of marks on the board.

Alternate Turn Test:

Input: Alternate turns between player X and player O without any winning conditions being met.

Expected Result: The game correctly switches turns between players after each valid move until the game ends in a win or draw.

Results:

Basic Gameplay Test Result:

→ If the game ends with a player winning: The game displays the winning player (X or O).

→ If the game ends in a draw: The game displays a message indicating a draw.

Winning Conditions Test Result:

→ If a winning condition is met: The game correctly identifies the winning player (X or O).

→ f no winning conditions are met: The game correctly declares a draw.

Invalid Input Test Result:

→ If invalid input is provided: The game prompts the player to enter valid input until it is provided.

Occupied Cell Test Result:

→ If a player attempts to mark an occupied cell: The game prompts the player to choose another cell until an unoccupied one is selected.

Game State Display Test Result:

→ The game board is displayed accurately at each stage of the game.

Alternate Turn Test Result:

→ The game correctly alternates turns between player X and player O until the game ends.

1. LESSONS LEARNT

Through the process of developing and testing the Tic Tac Toe game, several fundamental programming concepts and ideas can be learned and reinforced. The use of classes (TTTGame and TicTacToe) demonstrates the principles of encapsulation, inheritance, and polymorphism. Each class encapsulates related functionality and data, promoting modular and organized code.

PROGRAM LISTING

1. TicTacToe.java

public class TicTacToe { char[][] board;

// char p1 = 'X';

// char p2 = 'O'; char turn;

public TicTacToe() { turn = 'X';

board = new char[3][3];

for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) {

board[i][j] = ' ';

}

}

}

public char getWinner() {

// check row

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

if (board[i][0] == board[i][1] && board[i][0] == board[i][2]) return board[i][0];

}

}

// check col

for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) {

if (board[0][j] == board[1][j] && board[0][j] == board[2][j]) return board[0][j];

}

}

// check diagonal

if (board[0][0] == board[1][1] && board[1][1] == board[2][2]) return board[0][0];

if (board[0][2] == board[1][1] && board[1][1] == board[2][0]) return board[0][0];

return ' ';

}

public void displayBoard() {

for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) {

System.out.print("{"); System.out.print(board[i][j]); System.out.print("}");

}

System.out.println();

}

}

public char whoseTurn() { if (turn == 'X')

turn = 'O';

else if (turn == 'O') turn = 'X';

return turn;

}

public void putMark(int row, int col) { board[row][col] = turn;

}

public char getMark(int row, int col) { return board[row][col];

}

}

1. TTTGame.java

import java.util.Scanner; public class TTTGame {

TicTacToe game = new TicTacToe();

char winner;

public void startGame() { game.displayBoard(); playGame();

winner = game.getWinner(); printMessage();

}

public void playGame() {

Scanner in = new Scanner(System.in); int count = 0;

char turn; int row, col;

while (game.getWinner() == ' ' && count < 9) { turn = game.whoseTurn();

System.out.println(turn + "s turn. Type row and col: "); do {

row = in.nextInt(); col = in.nextInt();

} while (game.getMark(row, col) != ' ');

game.putMark(row, col); game.displayBoard(); count++;

}

in.close();

}

public void printMessage() { if (winner == 'X')

System.out.println("X has won"); else if (winner == '0')

System.out.println("O has won");

else

System.out.println("Draw");

}

public static void main(String[] args) { TTTGame ttt = new TTTGame(); ttt.startGame();

}

}