|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Content** | **Page No.** |
| 1 | Introduction | 1 |
| 2 | Tic Toe Game Rules | 2 |
| 3 | Methodology used | 3 |
| 4 | Implementation using Java Code | 4 |
| 5 | Output | 9 |
| 6 | Conclusion | 10 |

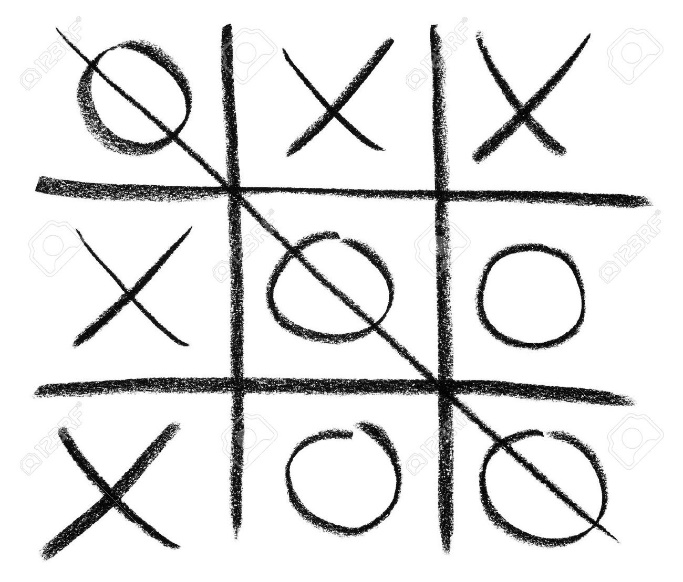
**INDEX**

**INTRODUCTION**

Tic Tac Toe is a well-known game played across all age groups. There is a 3x3 grid formed by two vertical and two horizontal lines. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game. The user can fill these nine places with either crosses X or noughts O. The aim of the game is to win by placing three similar marks in a horizontal, vertical, or diagonal row. In this paper, a simulation algorithm is presented to predict the win, or draw of a game by knowing the first move of a player. The game of tic-tac-toe is simulated using Min-max algorithm. The concept of combinational game theory is utilized to implement this game. This algorithm calculates only one step ahead using min-max algorithm. In an ideal scenario, a player must calculate all the possibilities to ensure the success. This simulation of tic tac toe needs input from QWERTY key here to book positions 0,1,2,3,4,5,6,7 and 8. It’s a small 3x3 game, so the state space tree generated will be short. Min-max is a decision-making algorithm which uses decision theory, game theory, statistics and philosophy to calculate the optimal move It is a two-player game. The mechanism evaluates minimum lose and maximum profit. In an ideal scenario, a player must calculate all the possibilities to ensure the success.

**TIC TAC TOE GAME RULES**

* The player who is playing "X" always goes first.
* Players alternate placing Xs and Os on the board
* This goes on until either one player has three in a row, horizontally, vertically, or diagonally; or all nine squares are filled.
* If a player is able to draw three of their Xs or three of their Os in a row, then that player wins.



**METHODOLOGY USED**

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

* A 3x3 grid is made.
* Numbers are assigned to each cell in the grid.
* A condition for winning is put into the function.
* Instructions are displayed for each player to choose the number in the grid where he wants to place X (or O for other player).
* Function recurs itself until the condition is met and a player wins or match is a draw.

**IMPLEMENTATION USING JAVA CODE:**

import java.util.Arrays;

import java.util.InputMismatchException;

import java.util.Scanner;

public class TicTacToe {

static Scanner in;

static String[] board;

static String turn;

public static void main(String[] args) {

in = new Scanner(System.in);

board = new String[9];

turn = "X";

String winner = null;

populateEmptyBoard();

System.out.println("Welcome to 2 Player Tic Tac Toe.");

System.out.println("--------------------------------");

printBoard();

System.out.println("X's 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:");

continue;

}

}

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.");

}

}

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.equals("XXX")) {

return "X";

} else if (line.equals("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;

}

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("/---|---|---\\");

}

static void populateEmptyBoard() {

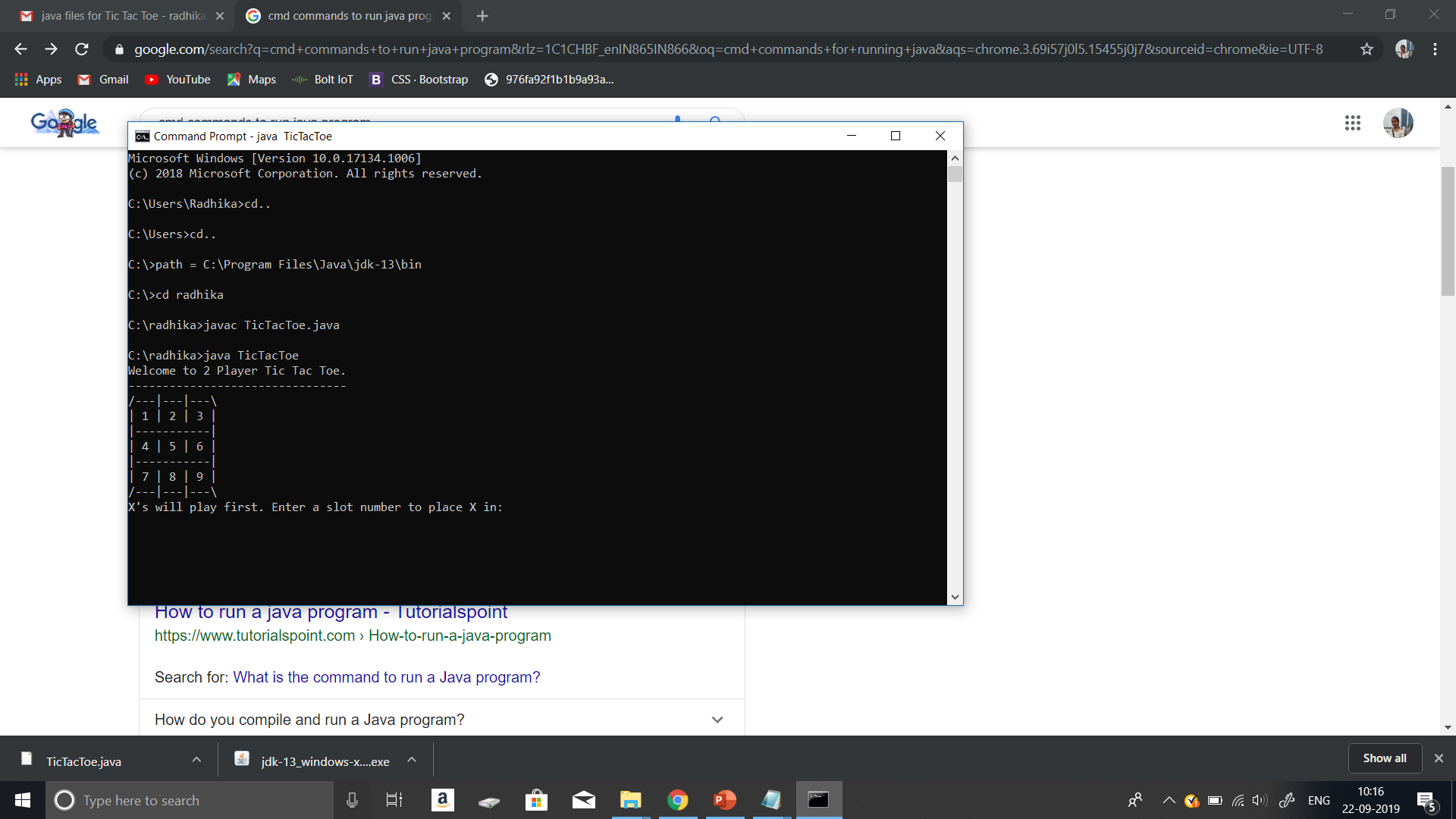
for (int a = 0; a < 9; a++) {

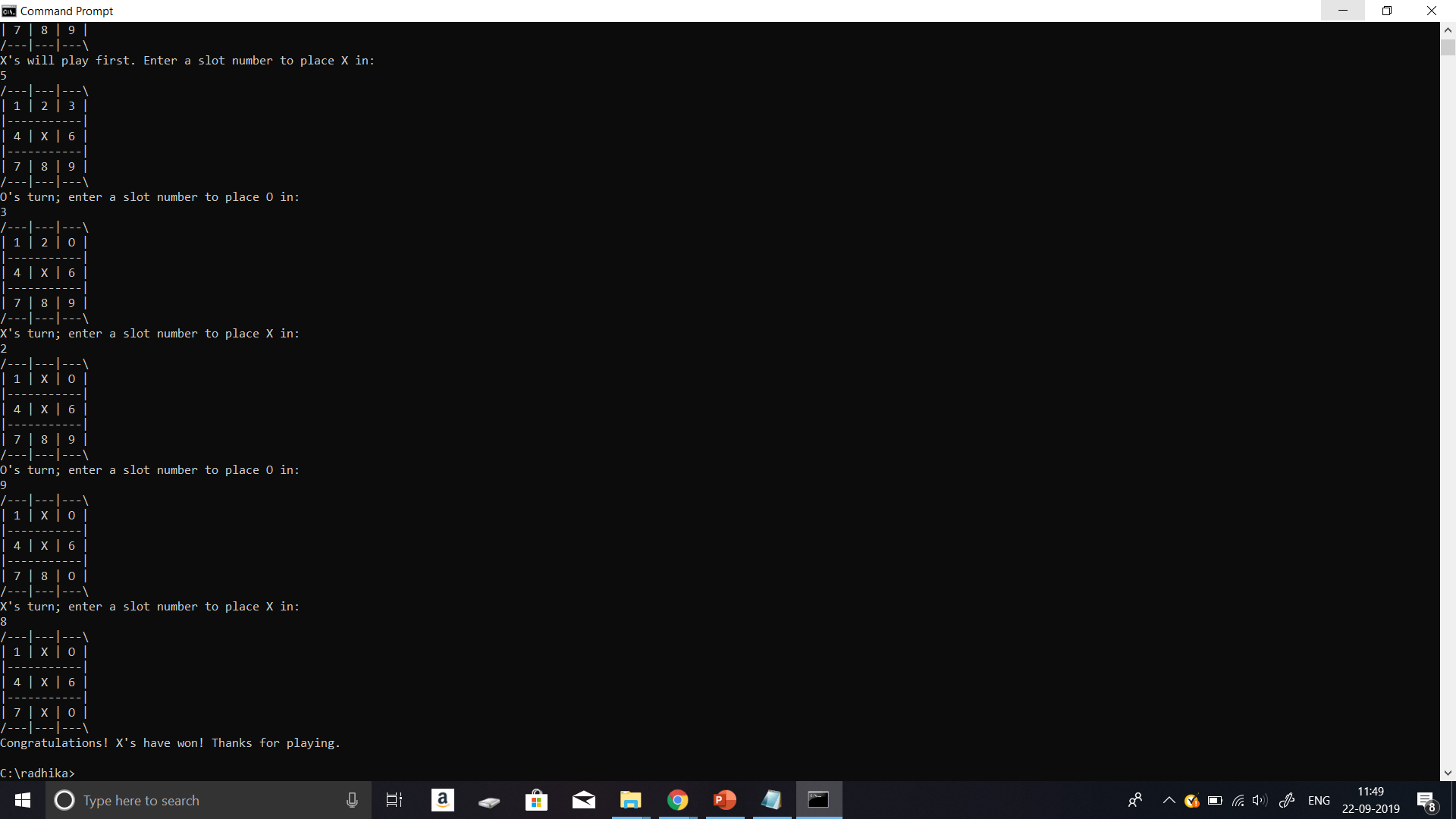
board[a] = String.valueOf(a+1);

}

}

}

**OUTPUT**



**FUTURE SCOPE**

The future scope we can see through this project is that, through java language using various functions and classes, much more complex games can be brought into existence. For example, snakes and ladders, dots, puzzles etc. By adding some more complex features like animations, styles and design we can develop games for android like tetris, car racing etc.

**CONCLUSION**

Thus, the famous game of Tic Tac Toe is successfully implemented using Java code.

**REFERENCES**

[1] Github : Available at *gist.github.com/xaviablaza-zz/3844825*

[2] CodePumpkin: Java Implementation of Tic Tac Toe. Available at: *codepumpkin.com/tic-tac-toe/*

[3] Information and rules about Tic Tac Toe: Wiki: Available at: *wikipedia.org*