|  |
| --- |
|  |
| Tic-Tac-Toe  Documentation  Instructor: Dr Sami Naji  Group members: |
| |  |  |  | | --- | --- | --- | | Mohammad Sameer Naeemi | Ahmad Roein Saba | Bushra Naeemi | |

Table of Content:

1. Brief introduction of the project…………. Page 2
2. The output of the code………………………… Page 3
3. Pseudo-code…………………………………………. Page 4
4. Flow Chart……………………………………………. Page 7
5. Code……………………………………………………… Page 13
6. Brief Introduction of the Project:

Tic-Tac-Toe is a two-player game which the players take turns to mark on a 3x3x3 board. The goal of winning is to be the first to get 3-in-a-row, including the 3-in-a-row on any of the three levels, or between levels.

The code consists of many functions, to make it understandable, reusable, and clear.

(a) A function board () which draws the 3x3x3 board, with different marks are represented by different characters, and placed on the board according to the current stage of the game.

(b) A function get-move () which receives the coordinate of mark to be placed by the player at each move. Make proper validations on the input to prohibit illegal moves, such as placing on a square which has a mark, and placing out of the 3x3x3 board.

(c) A function check\_win () which checks whether the game has reached the end.

according to the rule, and determines the winner of the game

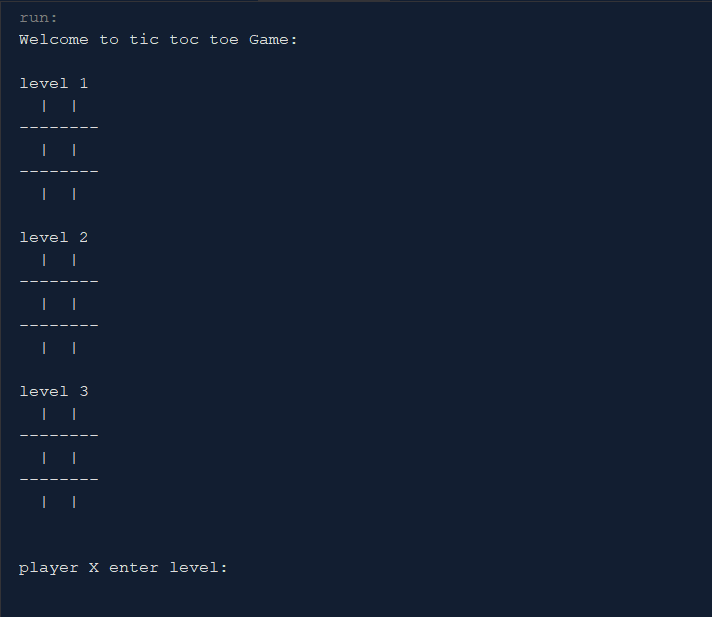
(d) A main function which controls the turn of the players and calls the functions above

according to a proper flow.

1. The output of the code:

There are player X and O.

The players should choose the level first, then the row and column. Players switch turns every time any of the player make a move If any of the players get 3-in-a-row in any of the three levels, it will win the game. Or if both of the players fill all the positions in a board but none of them got 3-in-a-row the game will draw.



1. Pseudo Code:

Pseudocode is an informal high-level description of the operating principle of a computer program or other algorithm. It uses the structural conventions of a normal programming language, but is intended for human reading rather than machine reading.

Pseudo Code for the methods used in the game:

**Pseudocode:**

Class tictactoeMain()

Main();

Output “Welcome to tic toc toe Game: ”

Output board;

Repeat until project.active() is false under condition (count > 9)

Count the turn of the player;

Output the project.ask;

Output board;

Check the checkWin();

If (count >10)

Output “the game is draw”;

Class functionsOfGame ();

functionsOfGame ();

Output ‘ ‘ (empty space) in every level, row and column positions

board();

Repeat the number of levels according to length of the array

Output number of levels;

Output an empty line;

Repeat the number of rows according to length of the array

If row==0 or row==1

Output “--------";

Repeat the number of columns according to length of the array

Output “ “(empty space);

If col==0 or col==1

Output “|”;

active();

Continue the game;

Ask();

Repeat the following until the position is not valid

Output “player %s(player turn) enter level: “

Input level;

Output “player %s(player turn) enter row: “

Input row;

Output “player %s(player turn) enter column: “

Input column;

Output notValid();

notValid();

if level, row and column is bigger than 3 or less than 1 or not empty

output false;

isEmpty();

if level, row and column is ‘ ‘

Output true;

else

Output “this position is taken”;

Output false;

checkWin();

Repeat and increment the level

Repeat and increment the row

If any row has the same value and is not empty

Output “the winner is X/O”;

Repeat and increment the column

If any column has the same value and is not empty

Output “the winner is X/O”;

If any of the diagonal row has the same value and is not empty

Output “the winner is X/O”;

getMove();

if row and column is between 0 and 2

if the position is not empty

Output false;

else

Output player;

Output true;

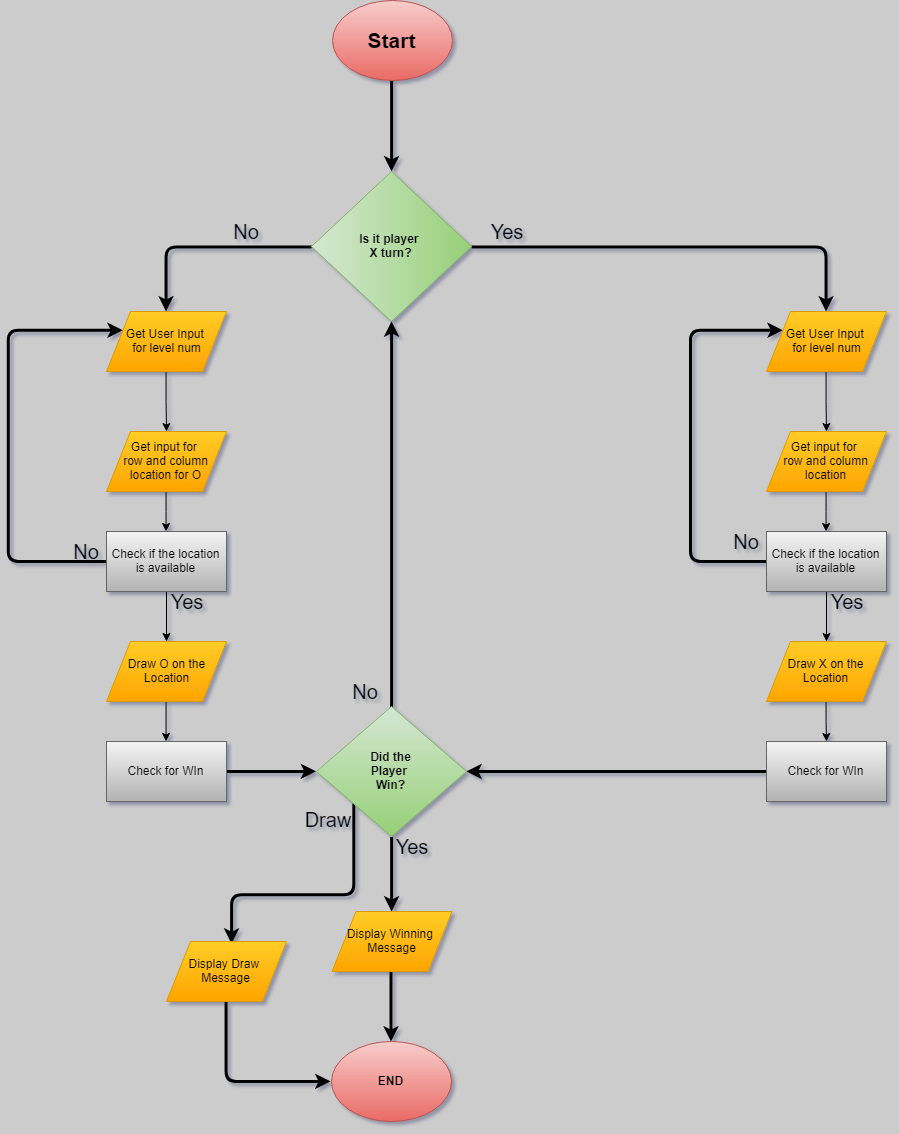
else

Output false;

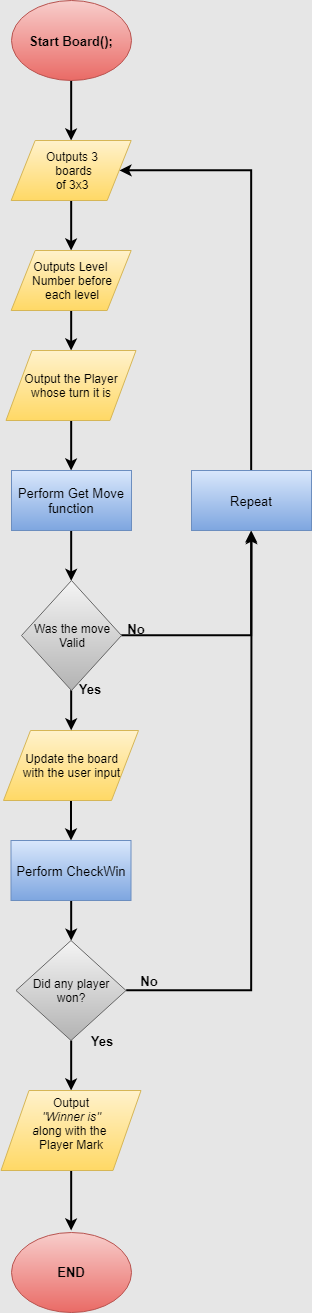
1. Flowchart:

A flowchart is a type of diagram that represents an algorithm, workflow or process. Flowchart can also be defined as a diagrammatic representation of an algorithm. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.

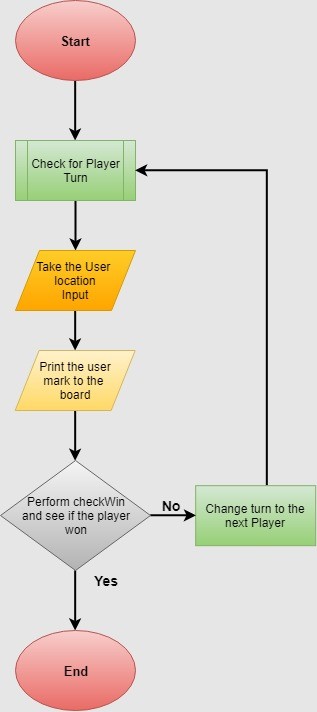
Flowchart of the Overall Tic Tac Toe Game:



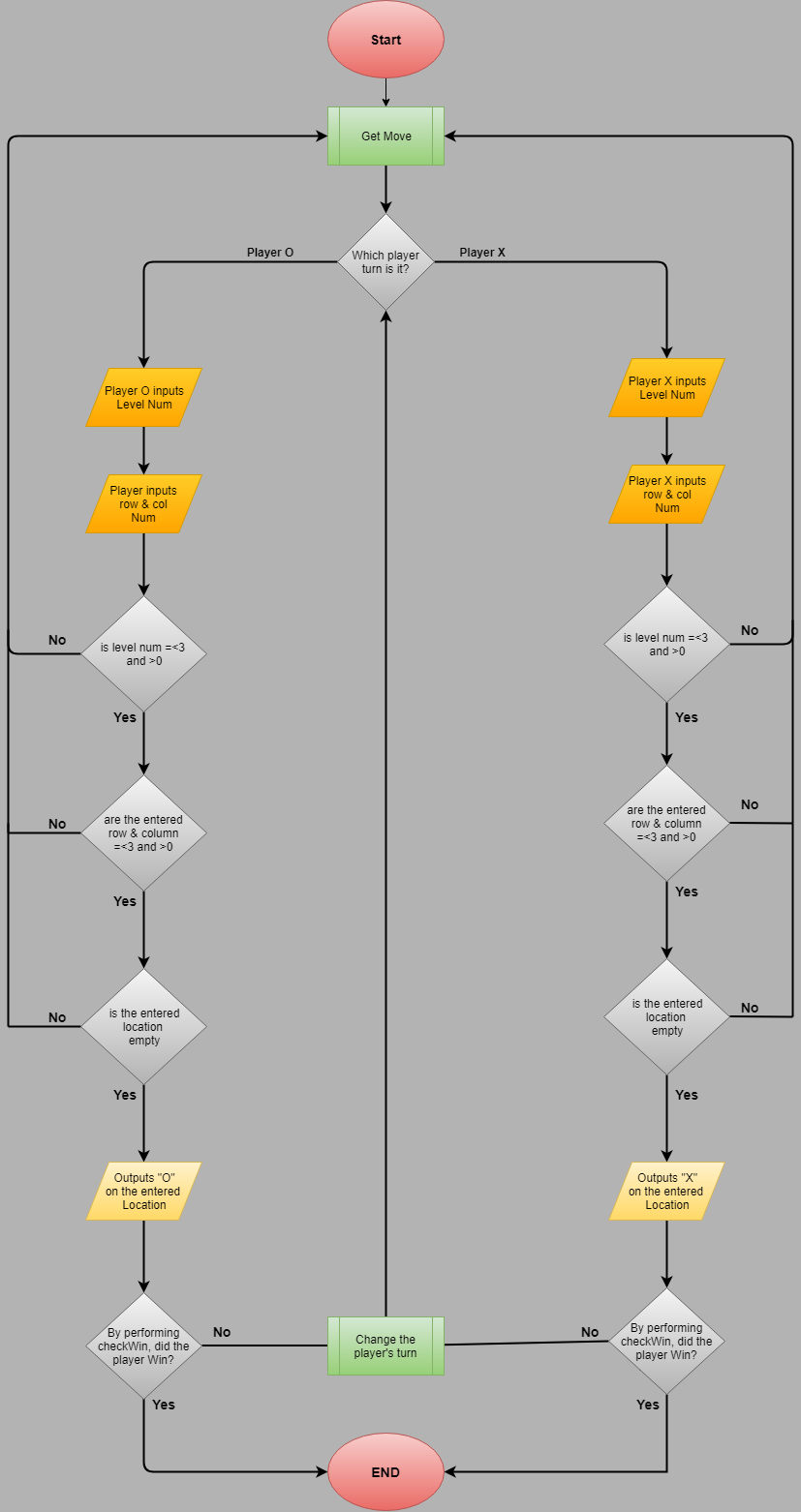
Flowchart of the Function Board:



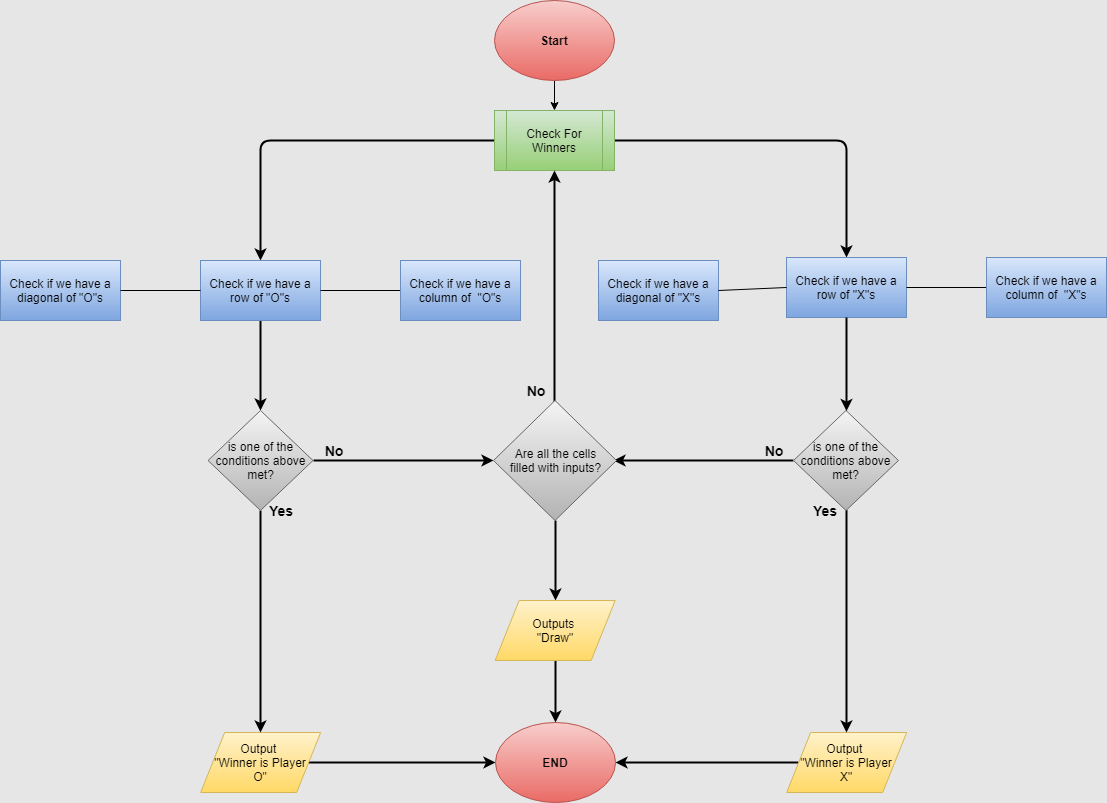
Flowchart of the Function Player Turn:



Flowchart of the Function Get Move:



Flowchart of the Function Check Win:



1. Code for the Main Method:

package tictactoe;

public class tictactoeMain {

public static void main(String[] args) {

functionsOfGame project = new functionsOfGame();

System.out.println("Welcome to tic toc toe Game: ");

System.out.println();

project.board();

int count = 1;

while(project.active())

{

if (count % 2 == 0){

project.ask('O');

}

else

{

project.ask('X');

}

count++;

project.board();

project.checkWin();

if(count == 10)

{

System.out.println("the game is draw \n");

project.gameon = false;

}

}

}

Functions of the Game in Another Class:

package tictactoe;

import java.util.Arrays;

import java.util.Scanner;

public class functionsOfGame {

char [][][] board ;

boolean gameon = true;

public functionsOfGame()

{

board = new char[3][3][3];

for (int num = 0; num<board.length; num++){

for (int row = 0; row<board.length;row++)

{

Arrays.fill(board[num][row],' ');

}

}

}

//Method below is used for creating a 3x3 tictactoe game

public void board()

{

for(int num = 0; num<board.length;num++){

int an = num+1;

System.out.println("level "+ an);

for(int row=0;row<board.length;row++){

for(int col=0; col<board.length;col++)

{

System.out.print(" "+board[num][row][col]);

if(col==0 || col==1)

System.out.print("|");

}

if(row==0 || row ==1)

System.out.print("\n--------\n");

}

System.out.println("\n");

}

System.out.println();

}

public boolean active()

{

return gameon;

}

public void ask(char player)

{

Scanner key = new Scanner(System.in);

int num;

int row, col;

do

{

System.out.printf("player %s enter level: ", player);

num = key.nextInt();

System.out.printf("player %s enter available row: ", player);

row = key.nextInt();

System.out.printf("player %s enter available column: ", player);

col = key.nextInt();

} while(notValid(num,row,col));

getMove(player,num-1,row-1,col-1);

}

public boolean notValid(int num , int row , int col)

{

if(num>3 || num<1 || row > 3 || row < 1 || col>3 || col<1 || !isEmpty(num, row, col))

return true;

else{

return false;}

}

public boolean isEmpty(int num , int row , int col)

{

if (board[num-1][row-1][col-1] == ' ')

return true;

else

{

System.out.println("this position is taken.\n");

return false;

}

}

public boolean checkWin()

{

for(int num = 0 ; num<board.length; num++){

for(int row=0; row<board.length;row++)

{

if(board[num][row][0] == board[num][row][1] && board[num][row][1] == board[num][row][2] && board[num][row][0] != ' ' )

{

System.out.println("The winner is player "+ board[num][row][0]);

gameon = false;

}

}

for(int col=0; col<board[0].length;col++)

{

if(board[num][0][col] == board[num][1][col] && board[num][1][col] == board[num][2][col] && board[num][0][col] != ' ')

{

System.out.println("The winner is player "+ board[num][0][col]);

gameon = false;

}

}

if(board[num][0][0] == board[num][1][1] && board[num][1][1] == board[num][2][2] && board[num][0][0] != ' ')

{

System.out.println("The winner is player "+ board[num][0][0]);

gameon = false;

}

if(board[num][2][0] == board[num][1][1] && board[num][1][1] == board[num][0][2] && board[num][2][0] != ' ')

{

System.out.println("The winner is player "+ board[num][2][0]);

gameon = false;

}

}

return false;

}

public boolean getMove(char player, int num, int row, int col)

{

if(row >=0 && row<=2 && col <=2 && col>=0)

{

if (board[num][row][col] != ' ')

return false;

else

{

board[num][row][col] = player;

return true;

}

}

else

return false;

}

}