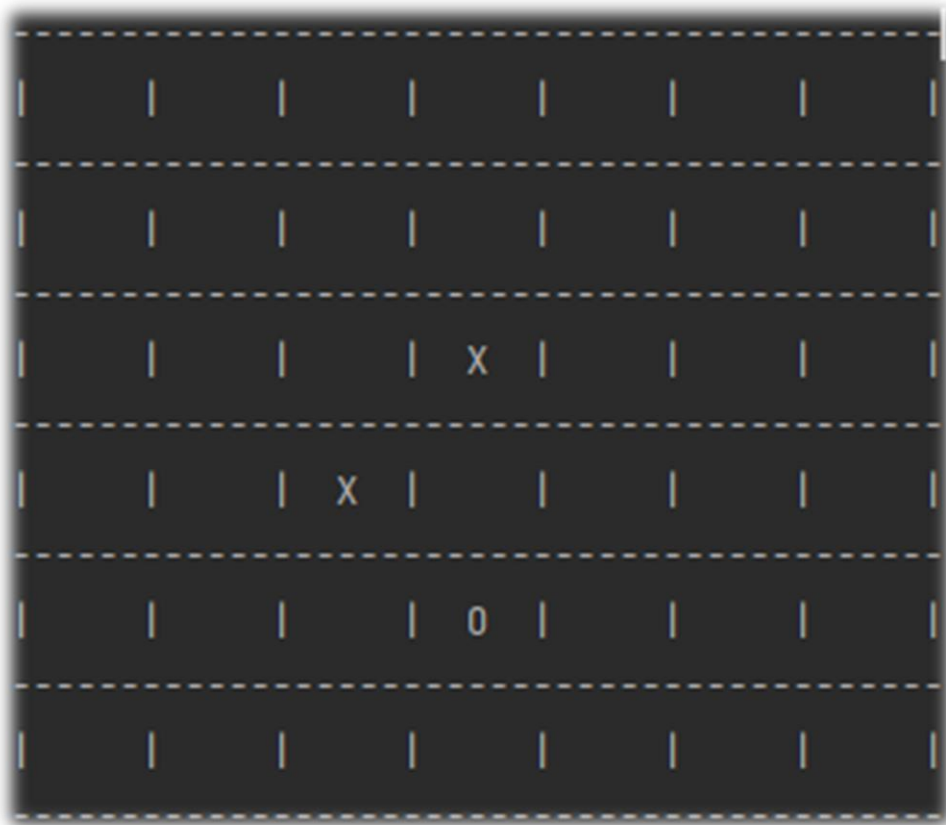


Tic Tac Toe Game




By: Mohamed Abdou Mabrouk

The Game



How to play:

When we start the game, the console will ask for the first and second players names:

A screenshot of a Java Swing dialog box titled "Input". The dialog box has a standard title bar with a close button (X) in the top right corner. The main content area has a light gray background. On the left side, there is a green square icon with a white question mark. To the right of the icon, the text "First Player name:" is displayed. Below the text is a white text input field with a thin gray border. At the bottom of the dialog, there are two buttons: "OK" and "Cancel", both with blue gradients and white text.



Then it will ask from the first player to choose the place to insert the letter:

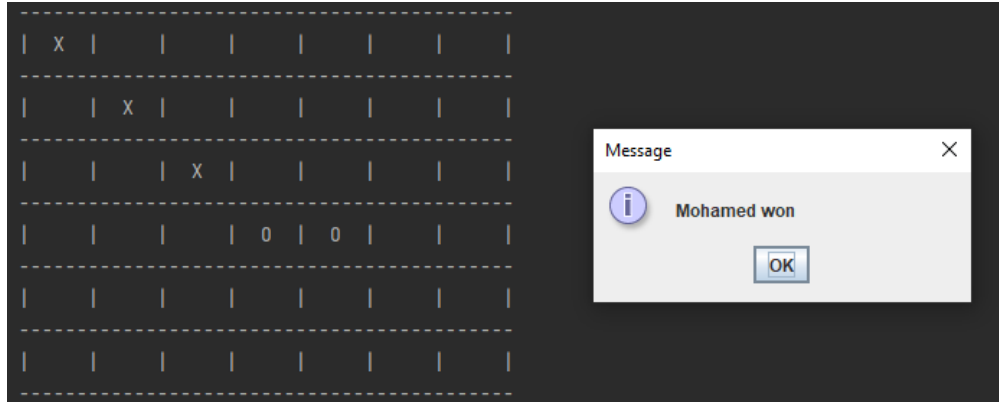


The letter will be inserted in the board:

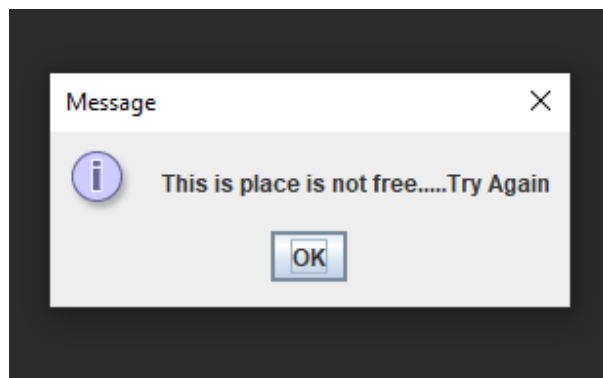
[illegible]

Rules:

- Who get three letters in row first win's:



- You cannot play in the same place:

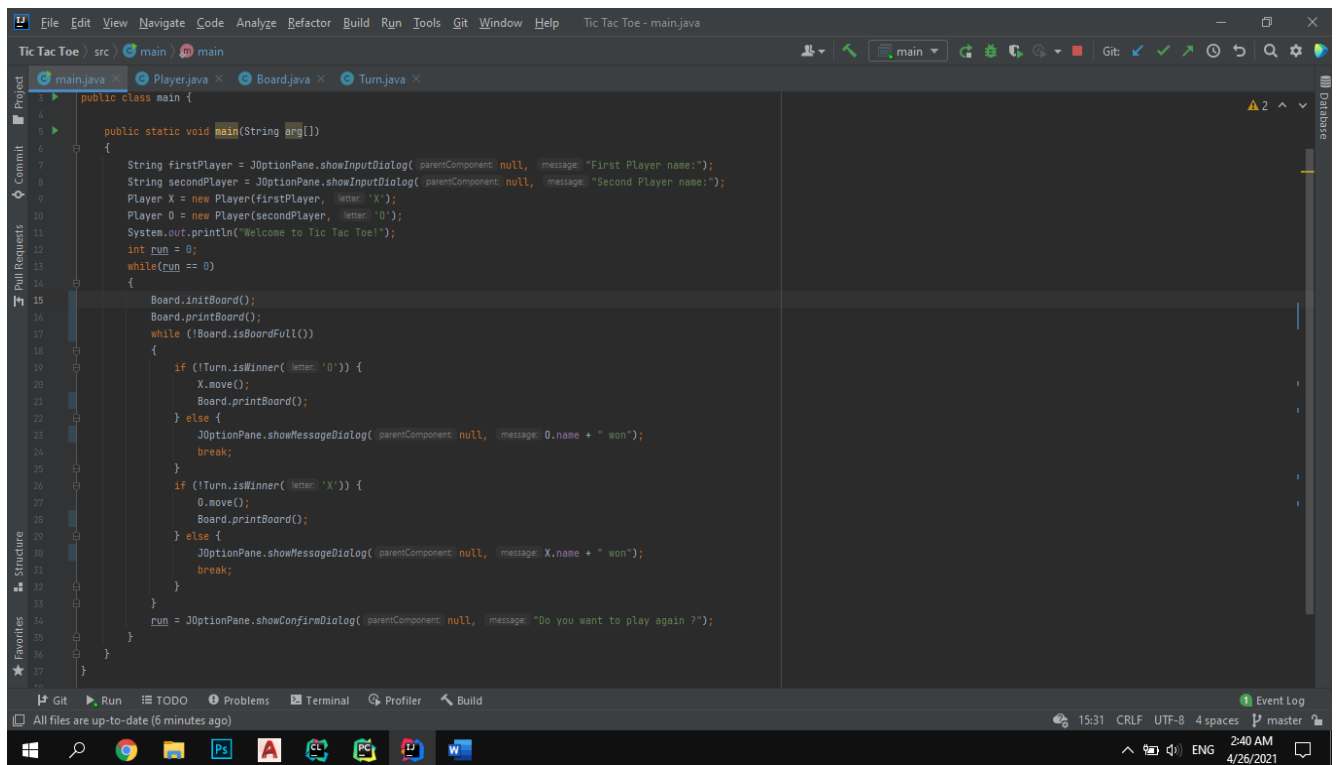


Syntax

The program has 4 classes: -

- Main
- Player
- Board
- Turn

1- Main:



```
1 public class main {
2
3     public static void main(String arg[])
4     {
5         String firstPlayer = JOptionPane.showInputDialog( parentComponent: null, message: "First Player name:");
6         String secondPlayer = JOptionPane.showInputDialog( parentComponent: null, message: "Second Player name:");
7         Player X = new Player(firstPlayer, letter: 'X');
8         Player O = new Player(secondPlayer, letter: 'O');
9         System.out.println("Welcome to Tic Tac Toe!");
10        int run = 0;
11        while(run == 0)
12        {
13            Board.initBoard();
14            Board.printBoard();
15            while (!Board.isBoardFull())
16            {
17                if (!Turn.isWinner( letter: 'O')) {
18                    X.move();
19                    Board.printBoard();
20                } else {
21                    JOptionPane.showMessageDialog( parentComponent: null, message: O.name + " won");
22                    break;
23                }
24                if (!Turn.isWinner( letter: 'X')) {
25                    O.move();
26                    Board.printBoard();
27                } else {
28                    JOptionPane.showMessageDialog( parentComponent: null, message: X.name + " won");
29                    break;
30                }
31            }
32            run = JOptionPane.showConfirmDialog( parentComponent: null, message: "Do you want to play again ?");
33        }
34    }
35 }
36
37 }
```

In this class we run the main and keep the game running by while loop until the board is full or any of players wins.

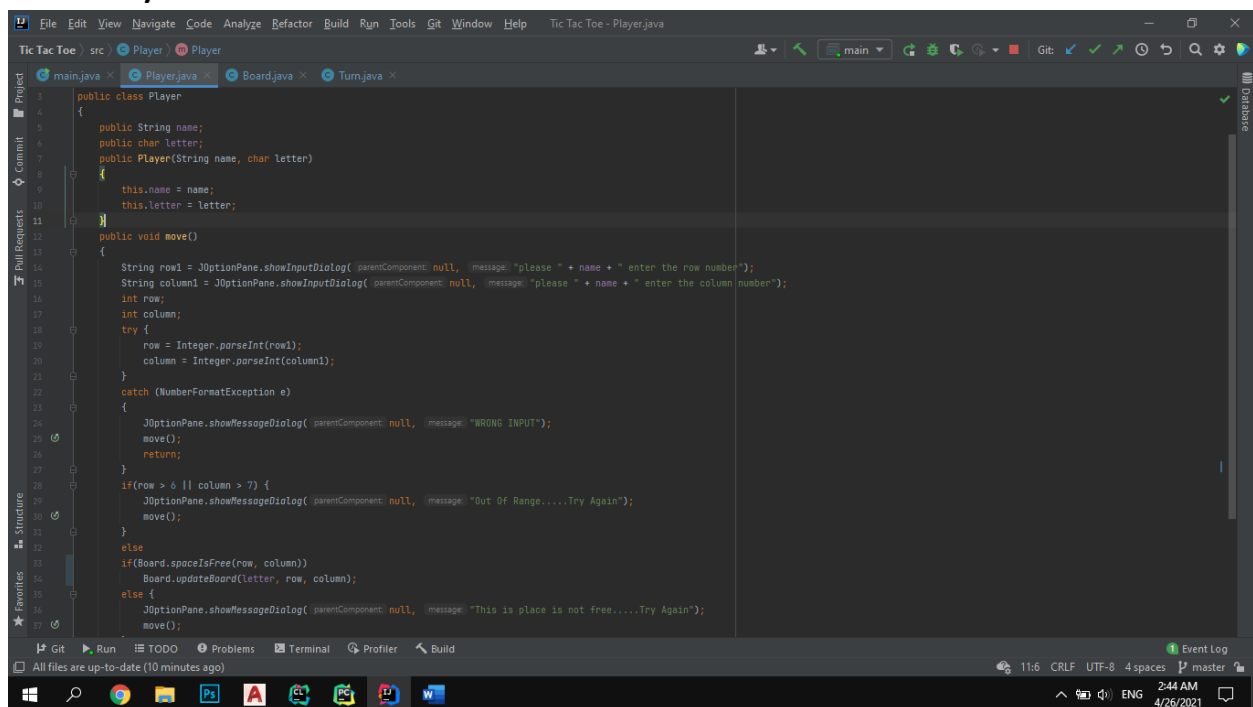
First, the program takes the players name from by the console.

Then create two objects from the player class.

Second, in the outer loop the program initiates the board and print it, in the inner loop condition the program checks if the board is full or not, if it's not full checks if the 'O' player won or not, if he/she did not win, the 'X' player plays after that checks if the 'X' player won, if he/she did not, the program will keep repeat the previous steps in the while loop.

Back to the outer loop the program will ask if the players want to play again.

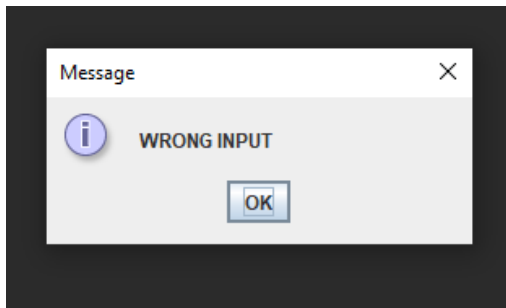
2- Player:



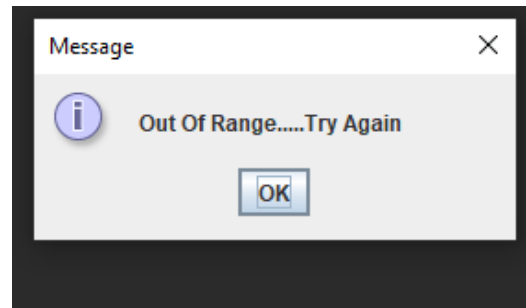
```
1 public class Player
2 {
3     public String name;
4     public char letter;
5     public Player(String name, char letter)
6     {
7         this.name = name;
8         this.letter = letter;
9     }
10
11
12     public void move()
13     {
14         String row1 = JOptionPane.showInputDialog( parentComponent: null, message: "please " + name + " enter the row number");
15         String column1 = JOptionPane.showInputDialog( parentComponent: null, message: "please " + name + " enter the column number");
16         int row;
17         int column;
18         try {
19             row = Integer.parseInt(row1);
20             column = Integer.parseInt(column1);
21         }
22         catch (NumberFormatException e)
23         {
24             JOptionPane.showMessageDialog( parentComponent: null, message: "WRONG INPUT");
25             move();
26             return;
27         }
28         if(row > 6 || column > 7) {
29             JOptionPane.showMessageDialog( parentComponent: null, message: "Out Of Range.....Try Again");
30             move();
31         }
32         else
33             if(Board.spaceIsFree(row, column))
34                 Board.updateBoard(letter, row, column);
35         else {
36             JOptionPane.showMessageDialog( parentComponent: null, message: "This place is not free.....Try Again");
37             move();
38         }
39     }
40 }
```

In this class constructor takes two parameters (name, letter) and set them.

Move method takes rows and column and set them in integers, then by try and catch the program check that if the input is not out of the range, if it not set them in them in the board if the place is free.



OR



3- Board:

```
1 public class Board
2 {
3     public static char board[][] = new char[6][7];
4
5     public static void printBoard()
6     {
7         System.out.println("-----");
8         for(int i = 0; i < 6; i++){
9             System.out.println(" | " + board[i][0] + " | " + board[i][1] + " | " + board[i][2] + " | " + board[i][3] + " | " + board[i][4] + " | " + board[i][5] + " | " + board[i][6] + " | ");
10            System.out.println("-----");
11        }
12    }
13
14    // Reset Board
15    public static void initBoard()
16    {
17        for(int i = 0; i < 6; i++){
18            for (int j = 0; j < 7; j++){
19                board[i][j] = ' ';
20            }
21        }
22    }
23
24    // Insert letter in the board
25    public static void updateBoard(char sig, int row, int col){ board[row - 1][col - 1] = sig; }
26
27    // Check if the place is free to insert letter
28    public static boolean spaceIsFree(int row, int col) {
29        return board[row - 1][col - 1] == ' ';
30    }
31
32    // check if the board is complet
33    public static Boolean isBoardFull()
34    {
35        for(int i = 0; i < 6; i++){
36            for(int j = 0; j < 7; j++){
37                if(board[i][j] == ' '){
38                    return false;
39                }
40            }
41        }
42        return true;
43    }
44 }
```

This class has the following methods:

- 1- Print the board.
- 2- Initiate the board and reset it by set all its elements by spaces.
- 3- Update the board by setting the letter in the element.
- 4- Check if space is free.
- 5- Check if the board is full.

4- Turn:

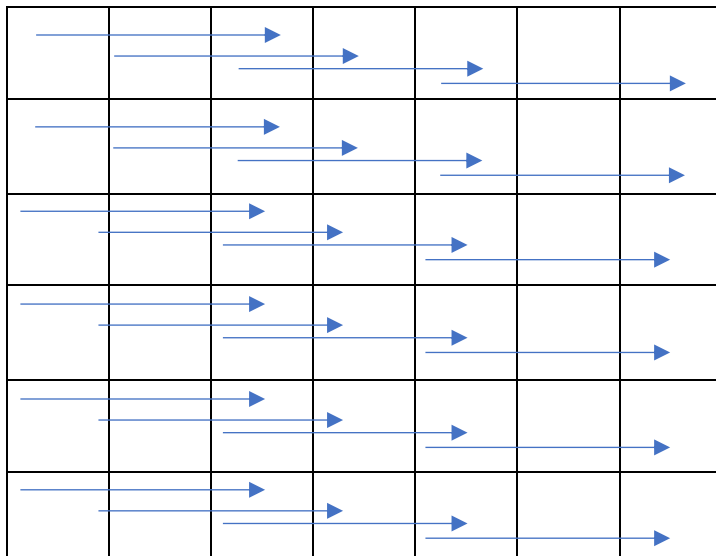
The screenshot shows an IDE with the following components:

- Menu Bar:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, Git, Window, Help.
- Toolbar:** Includes icons for file operations, search, and Git.
- Project Explorer:** Shows the project structure with files: main.java, Player.java, Board.java, and Turn.java. The Turn.java file is selected.
- Code Editor:** Displays the implementation of the `isWinner` method in the `Turn` class. The code checks for wins in rows, columns, and diagonals.

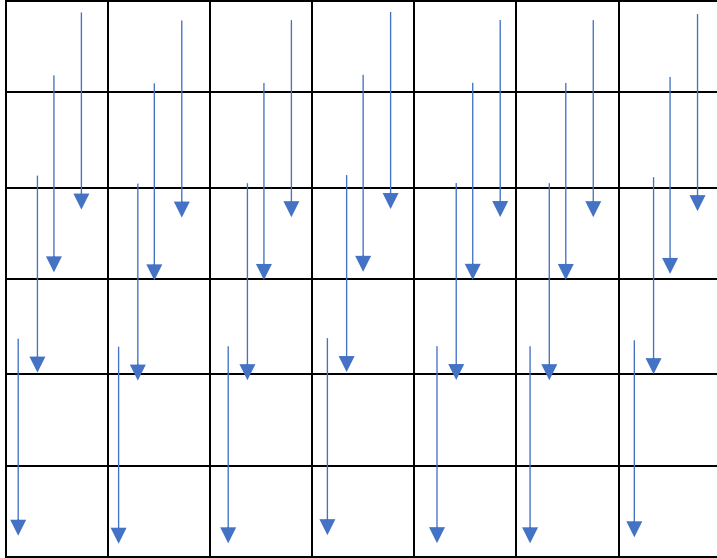
```
1 public class Turn
2 {
3     public static boolean isWinner(char letter) {
4         // check if rows win "-"
5         for (int i = 0; i < 6; i++)
6             for (int j = 0; j < 5; j++)
7                 if (Board.board[i][j] == letter && Board.board[i][j + 1] == letter && Board.board[i][j + 2] == letter)
8                     return true;
9
10        // check if columns win "|"
11        for (int i = 0; i < 7; i++)
12            for (int j = 0; j < 4; j++)
13                if (Board.board[j][i] == letter && Board.board[j + 1][i] == letter && Board.board[j + 2][i] == letter)
14                    return true;
15
16        // check if diagonal win "\"
17        for (int i = 0; i < 5; i++)
18            for (int j = 0; j < 4; j++)
19                if (Board.board[i][j] == letter && Board.board[i + 1][j + 1] == letter && Board.board[i + 2][j + 2] == letter)
20                    return true;
21
22        // check if diagonal win "/"
23        for (int i = 2; i < 7; i++)
24            for (int j = 0; j < 4; j++)
25                if (Board.board[i][j] == letter && Board.board[i - 1][j - 1] == letter && Board.board[i - 2][j - 2] == letter)
26                    return true;
27        return false;
28    }
29 }
```
- Bottom Bar:** Includes tabs for Git, Run, TODO, Problems, Terminal, Profiler, and Build. The status bar shows "All files are up-to-date (12 minutes ago)".
- System Tray:** Shows the date and time as 3:02 AM on 4/26/2021.

This class checks if the player had won or not by:

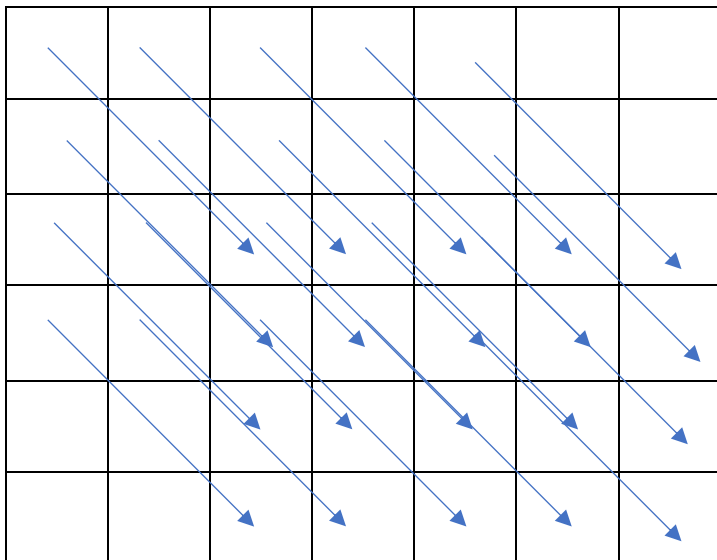
1- Checking Rows:



2- Checking Columns:



3- Checking First Diagonal:



4- Checking Second Diagonal:

