Tic-Tac-Toe

OOP Implementation

Prepared by:

Name
Youssef Samuel Nachaat Labib
Youssef Amr Ismail Othman

Classes Description

Our tic-tac-toe game is implemented by the use of OOP using 3 classes: **Board** , **Player** and **Game**. We will discuss the state and the behaviour for each class individually and how they interact with each other.

1) Board Class

The **Board** class is used to encapsulate all the attributes and methods that would be used by the board. It uses 4 constants that will specify the size and properties of the board.

The constants used are:

- *ROWS* which is set to 6 stating the row size.
- *COLUMNS* which is set to 7 stating the column size.
- *ARRAY_EXTENSION* that is set to 4 (Its use will be fully elaborated in the isWinner method).
- *NUM_BLOCKS* that states the total number of cells in the board. It is the result of the multiplication of *ROWS* and *COLUMNS*.

A character array "board" will be used as the grid. It is declared and constructed in the attributes section of the class. fullBlocks variable is used as a counter that is incremented after each successful insertion in the cell. It will state the total number of occupied cells at any time.

The class constructor is used to initialize all elements of the array board to '-' .
As the fullBlocks variable implements the encapsulation method, a setter and a getter were used to access and modify it. Since it could be set by only the previous value + 1, the setter does not have any parameters. Another method of that class is printBoard, it will print the board after each player makes his move according to the format set that will be seen in the sample runs.

insert method is used to assign the player input value to the cell specified. It has 3 parameter, one stating the row, another stating the column and the third stating the **Player** object. It uses the method getInputValue of the **Player** class to get the symbol for that **Player** object.

The methods used for checking for a winner are 2. isDraw method would check whether the inserted cells is equal to the total number of cells, if found equal and nobody won then it will announce a draw.

isWinner method checks after each cell is inserted whether this cell may result in a win or not, it has the parameter of the **Player** object that placed that cell. The getters for the row and column would be called so the checking can happen upon that cell entered.

Algorithm for checking:

Upon constructing and initialising the array, we increased its size by 2 rows from above, 2 rows from below, 2 columns from the left and 2 columns from the right (Shown in red). Resulting in an increase by 4 for the wanted rows and columns dimensions “*ARRAY_EXTENSION”.* The white cells are the cells that would be played in and the ones that will be displayed for the players. This step was made to facilitate checking in an easier way.

	0	1	2	3	4	5	6	7	8	9	10
0	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
1	🟥	🟥	1	2	3	4	5	6	7	🟥	🟥
2	🟥	1	?	?	?	?	?	?	?	🟥	🟥
3	🟥	2	?	?	?	?	?	?	?	🟥	🟥
4	🟥	3	?	?	**X**	?	?	?	?	🟥	🟥
5	🟥	4	?	?	?	?	?	?	?	🟥	🟥
6	🟥	5	?	?	?	?	?	?	?	🟥	🟥
7	🟥	6	?	?	?	?	?	?	?	🟥	🟥
8	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
9	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥

There are 12 possibilties that may result in a win if the player placed in [3] [3] (index [4][4]) for instance. The check is emplemented in the function using that particular numbered order found in the following table, the function will be terminated upon reaching the first match

1.

?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	**X**	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?

2.

?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	**X**	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?

3.

?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	**X**	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?

4.

?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	**X**	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?

5.

?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	**X**	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?

6.

?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	**X**	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?

7.

?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	**X**	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?

8.

?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	**X**	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?

9.

?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	**X**	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?

10.

?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	**X**	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?

11.

?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	**X**	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?

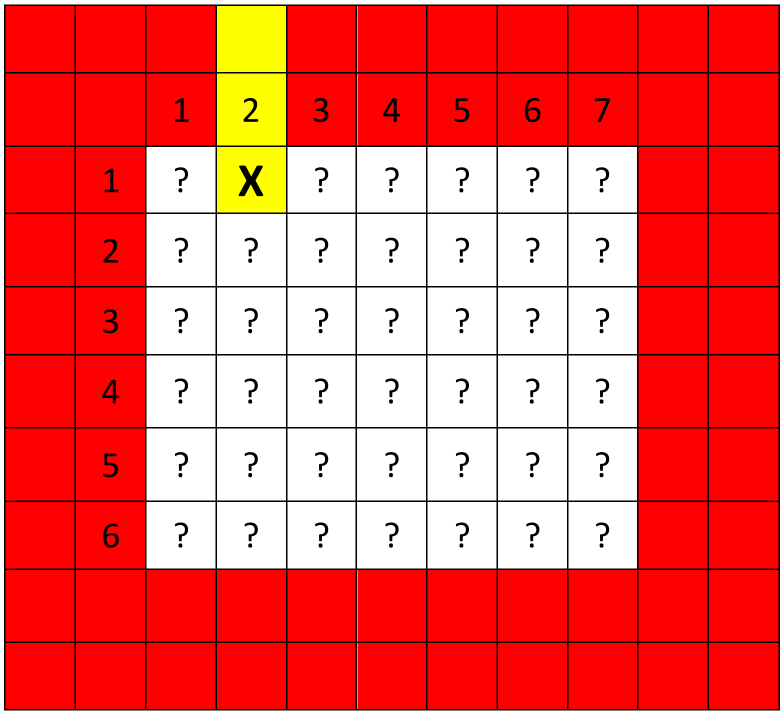
12.

?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	**X**	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?

Increasing the array size by 4 rows and 4 columns is done solely for the sake of the check function. By that increase, if we check for an element on the borders, no exception errors resulting from accessing invalid index number would occur "ArrayIndexOutOfBoundsException". It will not affect the logic for the check algorithm as well, meaning that the check would not be altered according to the place of the cell to be checked.

Consider checking an element placed in [1][2] (index [2][3]).

The first check is highlighted in yellow. Although the check is made in cells that will not be displayed, but it will not result in an error as those cells are already reserved for the array from the beginning. And the checks would continue …….



🟥	🟥	🟥	🟨	🟥	🟥	🟥	🟥	🟥	🟥	🟥
🟥	🟥	1	2	3	4	5	6	7	🟥	🟥
🟥	1	?	**X**	?	?	?	?	?	🟥	🟥
🟥	2	?	?	?	?	?	?	?	🟥	🟥
🟥	3	?	?	?	?	?	?	?	🟥	🟥
🟥	4	?	?	?	?	?	?	?	🟥	🟥
🟥	5	?	?	?	?	?	?	?	🟥	🟥
🟥	6	?	?	?	?	?	?	?	🟥	🟥
🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥

Code:

```
package tictactoe;
import javax.swing.JOptionPane;

public class Board {

	public static final int ROWS = 6;//ROWS is a constant for the number of rows of the 
board.
	public static final int COLUMNS = 7;//COLUMNS is a constant for the number of columns 
of the board.
	//Our 2 dimensional array will be extended by 4 rows(2 up and 2 down) and 4 columns(2 
right and 2 left) to facilitate checking who won the game.
	public static final int ARRAY_EXTENSION = 4;
	public final int NUM_BLOCKS = ROWS * COLUMNS;//Constant for the total number of 
blocks(cells).
	public char[][] board = new char[ROWS+ARRAY_EXTENSION][COLUMNS+ARRAY_EXTENSION];//2d 
array used for the game.
	private int fullBlocks;//Variable starting by 0, and will be incremented by 1 after 
every turn, representing the number of full cells.
	public Board()//Constructor that fills the 2d array by '-';
	{
		int i, j;
		for (i = 0; i < ROWS+ARRAY_EXTENSION; i++)
		{
			for (j = 0; j < COLUMNS+ARRAY_EXTENSION; j++)
			{
				board[i][j] = '-';
			}
		}
	}
	public int getFullBlocks()//to get the number of full blocks.
	{
		return fullBlocks;
	}
	public void incrementFullBlocks() { //the increment the full blocks by 1 after each 
turn.
		fullBlocks++;
	}
	public void printBoard() { //A method to print the board for the players after each 
turn with the new changes.
		int i, j;
		System.out.printf(" ");
		for(j=0;j<COLUMNS;j++) {
			System.out.printf("   %d   ",j+1);
		}
		System.out.printf("\n\n");
		for (i = 0; i < ROWS; i++) {
			System.out.printf("%d",i+1);
			for (j = 0; j < COLUMNS; j++) {
				System.out.printf("   %c   ", board[i+(ARRAY_EXTENSION / 
2)][j+(ARRAY_EXTENSION/2)]);
			}
```

```
                        System.out.printf("\n\n");
                }
        }
        public void insert(int row, int column, Player player) {//A method that insert 'X' or
'O' in the cell chosen by the player.
                board[row][column] = player.getInputValue();
        }
        public boolean isDraw() {
                if(getFullBlocks() == NUM_BLOCKS) { //It is a draw when the number of full
blocks is equal to the total number of blocks.
                        JOptionPane.showMessageDialog(null, "Draw!");
                        return true;//It returns true when it is a draw
                }
                return false;//It is not a draw.
        }
        public boolean isWinner(Player player) {//Check if the player has won.
                int row = player.getRowInput();//get the row where the player inserted 'X' or
'O'.
                int column = player.getColumnInput();//get the column.
                if(board[row][column]==board[row-1][column]&&(board[row][column]==board[row-
2][column]||board[row][column]==board[row+1][column])) {
                        return true;}
                else
if(board[row][column]==board[row+1][column]&&board[row][column]==board[row+2][column]) {
                        return true;}
                else if(board[row][column]==board[row][column-
1]&&(board[row][column]==board[row][column-2]||board[row][column]==board[row][column+1])) {
                        return true;}
                else
if(board[row][column]==board[row][column+1]&&board[row][column]==board[row][column+2]) {
                        return true;}
                else if(board[row][column]==board[row-
1][column+1]&&(board[row][column]==board[row-
2][column+2]||board[row][column]==board[row+1][column-1])) {
                    return true;}
                else if(board[row][column]==board[row+1][column-
1]&&board[row][column]==board[row+2][column-2]){
                    return true;}
                else if(board[row][column]==board[row-1][column-
1]&&(board[row][column]==board[row-2][column-
2]||board[row][column]==board[row+1][column+1])) {
                    return true;}
                else
if(board[row][column]==board[row+1][column+1]&&board[row][column]==board[row+2][column+2]){
                    return true;}

                return false;//The player has not won.
                }

}
```

2) Player Class

The **Player** class clearly shows how the encapsulation principle was used in our code. All the attributes were set private for which the accessors and mutators methods will be used to modify or access them.

The attributes used are:

- playerNumber that stores the number for the player. It will carry the value of 1 or 2.
- playerName that will store the name for the **Player** object.
- rowInput that will store the row number for the last cell added by the **Player** object.
- columnInput that will store the column number for the last cell added by the **Player** object.
- inputValue that stores the characters ‘X’ or ‘O’ according to whether which **Player** object is used.

The constructor is used to set the inputValue of the object to ‘X’ for player 1 and ‘0’ for player 2. In addition to all the setters and getters methods used, the class has play(Board board) method that prompts the user for his row and column input. The method checks whether the input values are out of range or not. If found out of range, a loop will be executed until a successful input is reached. The method would then check for the wanted cell if its vacant or preoccupied by a previous move. If found preoccupied, it would call the play method recursively, else it would use the setters of the row and column and would call the insert method in **Board** class using the getters of the row and column and this to send the current **Player** object. The fullBlocks of **Board** class would then be incremented by 1 to state that a new cell has been occupied. It’s used as a counter that can have a maximum value of 42.

Code:

```java
package tictactoe;
import javax.swing.JOptionPane;

public class Player {
	//All the instance variables are private and to have access on them we use 
the setters and getters (Encapsulation)
	private int playerNumber;//The number of the player 1 or 2
	private String playerName;//The name of the player
	private int rowInput;//The number of row that the player chooses each turn
	private int columnInput;//The number of column that the player chooses each 
turn
	private char inputValue;//Each player has an input value 'X' or 'O'

	public Player(int number) {//The player constructor has as input a number 
which is the player's number
		playerNumber = number;
		if (number == 1)
			setInputValue('X');//Setting the input value as 'X' for player 1 
and 'O' for player 2.
		else
			setInputValue('O');
    }
	//Getters and setters for all the instance variables.
	public int getPlayerNumber() {
		return playerNumber;
	}
	public void setPlayerNumber(int number) {
		playerNumber = number;
	}
	public String getPlayerName() {
		return playerName;
	}
	public void setPlayerName(String name) {
		playerName = name;
	}
	public int getRowInput() {
		return rowInput;
	}
	public void setRowInput(int rowInput) {
		this.rowInput = rowInput;
	}
	public int getColumnInput() {
		return columnInput;
	}
	public void setColumnInput(int columnInput) {
		this.columnInput = columnInput;
	}
	public char getInputValue() {
		return inputValue;
	}
	public void setInputValue(char ch) {
		inputValue = ch;
	}

	public void play(Board board)//A method to let a player play.
	{
		int r, c;
		String row = JOptionPane.showInputDialog(null, "Hi " + getPlayerName()
+ " (Player " + getPlayerNumber() + ") " + ", please enter the row: ");
	       r = Integer.parseInt(row);//Convert the string to an integer.
		while(r>Board.ROWS||r<1) {//Check that the row is not out of range.
			row = JOptionPane.showInputDialog(null, "Incorrect row choice!
Please re-enter your choice\nEnter row number: ");
			    r = Integer.parseInt(row);
		}
		String column = JOptionPane.showInputDialog(null, "Please enter the
column: ");
		c = Integer.parseInt(column);
		while(c>Board.COLUMNS||c<1) {//Check that the column is not out of
range.
			column = JOptionPane.showInputDialog(null, "Incorrect column
choice! Please re-enter your choice\nEnter column number: ");
			    c = Integer.parseInt(column);
		}
		if (board.board[r+1][c+1] != '-') {//Check that the cell chosen by the
player is empty or already filled.
			JOptionPane.showMessageDialog(null, "INVALID PLACE");
			play(board);//recall the method from the beginning to let the
user re-enter the row and column.
		}
		else {//if the row and column are valid:
			setRowInput(r+1);//We increment the row and column by 1 the fill
the correct the cell.
			setColumnInput(c+1);
			board.insert(getRowInput(), getColumnInput(), this);//We call the
insert method from Board class to fill the required cell.
			board.incrementFullBlocks();//Increment the number of full blocks
by 1 after the player has played.
			}
		}
	}
```

3) Game Class

The **Game** class is the class where the programme starts. It has the main method that creates an object of the class **Board** to be named board and 2 objects for the 2 players from the class **Player** to be named player1 and player2. The main would ask the user to enter player 1 name and player 2 name respectively using GUI (JOptionPane). A welcome message would then be displayed to the user and the board would be printed on the console. Player 1 would then be prompted to enter the row and column choice using the play method of the **Player** class. After a successful insertion, the isWinner method would be invoked to check whether player 1 won the game using that move or not, if he won, he will be congratulated using a GUI message, if not, it will check for a draw using the isDraw method. If player 1 did not win neither his move made a draw, the board will be displayed and the same sequence would be made with player 2. The function arguments would only differ according to the **Player** object used. All these statements are implemented in a while loop that will break if a win or a draw is reached.

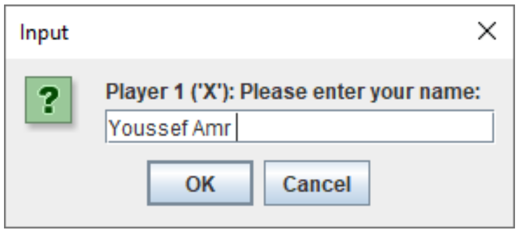
The final board would then be displayed and a thanking message would appear for the user stating the game termination.

Code:

```
package tictactoe;
import javax.swing.JOptionPane;

public class Game {
	public static void main(String[] args) {
		Board board = new Board(); //create an instance from the class Board.
		Player player1 = new Player(1); //create an instance for player 1.
		Player player2 = new Player(2); //create an instance for player 2.
		//Ask each player for his name and setting his name using the set method.
		String name1 = JOptionPane.showInputDialog(null, "Player 1 ('X'): Please enter 
your name: ");
		player1.setPlayerName(name1);
		String name2 = JOptionPane.showInputDialog(null, "Player 2 ('O'): Please enter 
your name: ");
		player2.setPlayerName(name2);
		//The board is printed and players ready to play.
		JOptionPane.showMessageDialog(null, "Look at the board and enjoy playing");
		System.out.println("Welcome to tic-tac-toe!");
	    board.printBoard();
	    //Loop until a player wins or it is a draw.
		while(true)
		{
			player1.play(board);//Player 1 turn.
			if(board.isWinner(player1))//Check if player 1 wins.
					{
						JOptionPane.showMessageDialog(null, "Player 1 ('X') 
wins, congratulations "+player1.getPlayerName()+" !");
						break;
					}
			if(board.isDraw())//Check is it is a draw.
				break;
			System.out.println("");
			System.out.println("Board now: ");
		    board.printBoard(); //Print the board after player 1 has played.
			player2.play(board); //Now player 2 plays.
			if(board.isWinner(player2))//Check if player 2 wins.
					{
						JOptionPane.showMessageDialog(null, "Player 2 ('O') 
wins, congratulations "+player2.getPlayerName()+" !");
						break;
					}
			if(board.isDraw())//Check is it is a draw.
				break;
			System.out.println("");
			System.out.println("Board now: ");//Print the board after player 2 has 
played.
		    board.printBoard();
		}
		System.out.println("Final Board: ");
		board.printBoard();//The board is printed for one last time.
		System.out.println("THANKS FOR PLAYING"); } }
```

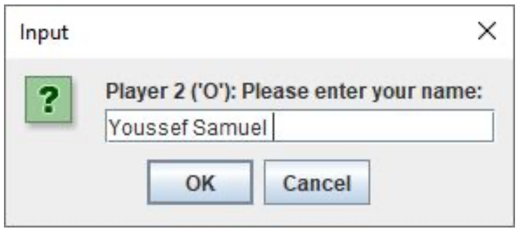
Sample Runs

Upon running the programme, a window will appear to the user to prompt him to enter the name for the first player, then he will be asked to enter the name for second player as well. As soon as he confirms both, a welcome message would appear and the board will then be printed on the console.


Input

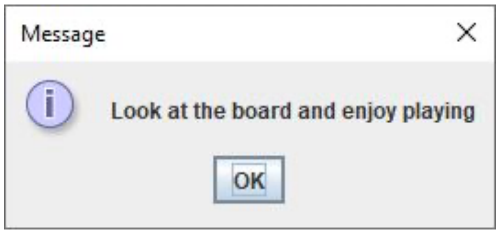
Player 1 ('X'): Please enter your name:

Youssef Amr

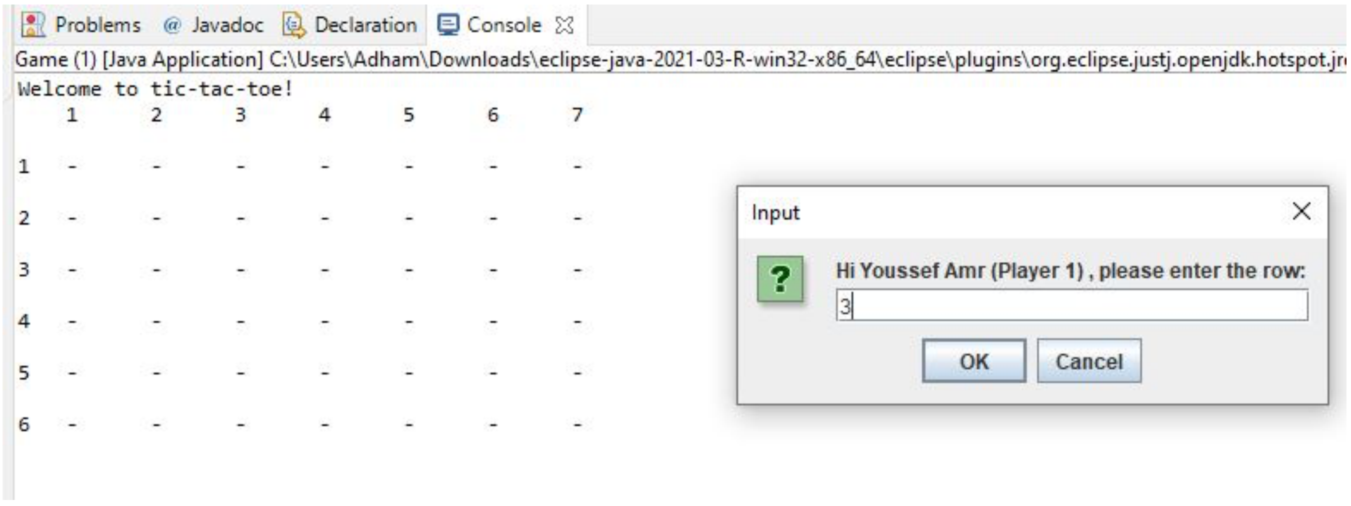

Input

Player 2 ('O'): Please enter your name:

Youssef Samuel


Message

Look at the board and enjoy playing



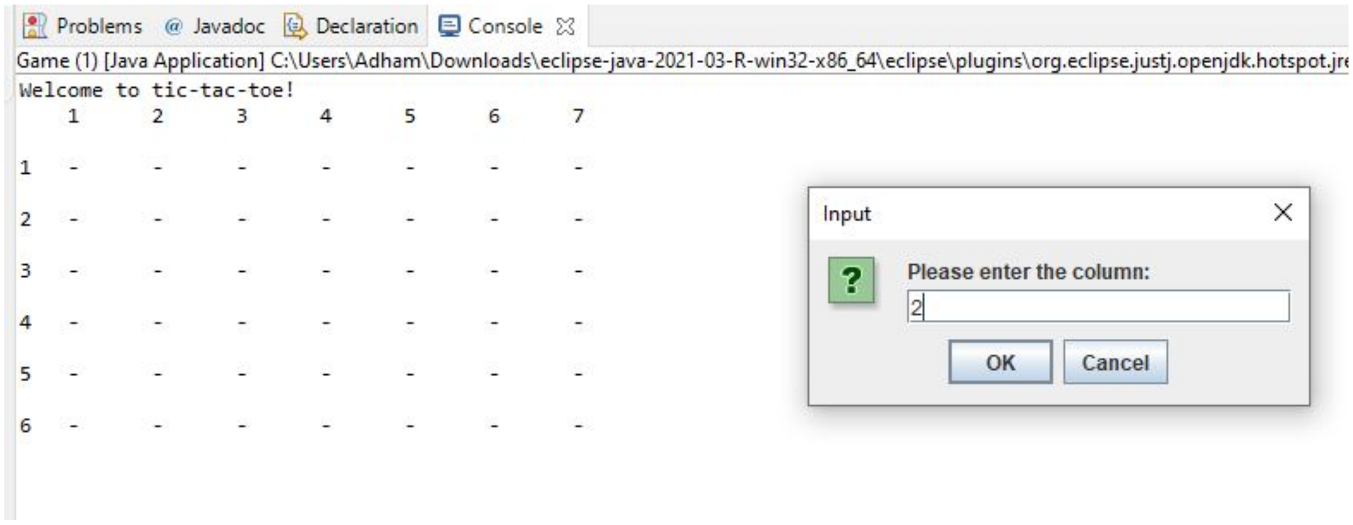
```
Game (1) [Java Application] C:\Users\Adham\Downloads\eclipse-java-2021-03-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jr
Welcome to tic-tac-toe!
    1     2     3     4     5     6     7

1   -     -     -     -     -     -     -

2   -     -     -     -     -     -     -

3   -     -     -     -     -     -     -

4   -     -     -     -     -     -     -

5   -     -     -     -     -     -     -

6   -     -     -     -     -     -     -
```

Input

Hi Youssef Amr (Player 1) , please enter the row:

3

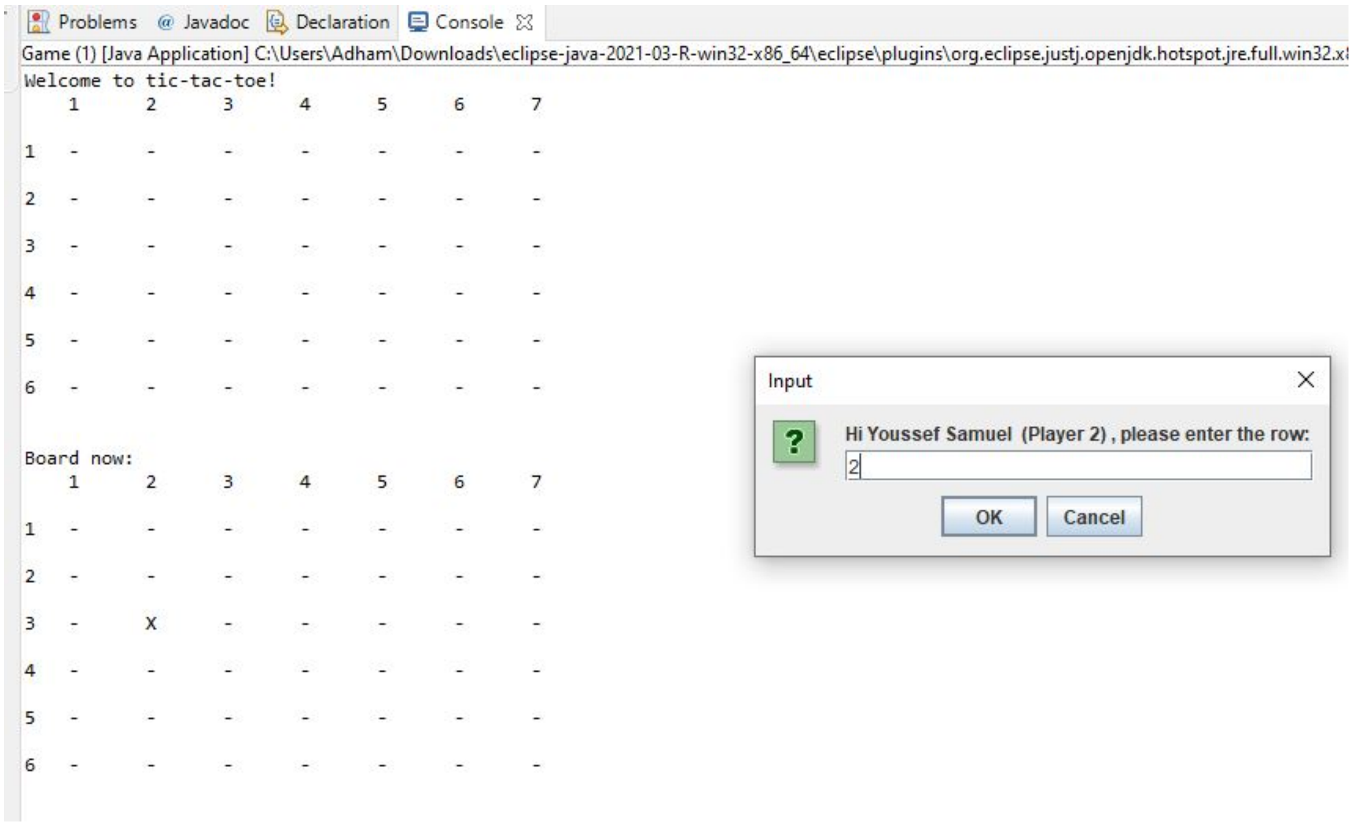
Player 1 will be asked to enter his row choice for the first move.



```
Game (1) [Java Application] C:\Users\Adham\Downloads\eclipse-java-2021-03-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre
Welcome to tic-tac-toe!
    1     2     3     4     5     6     7

1   -     -     -     -     -     -     -

2   -     -     -     -     -     -     -

3   -     -     -     -     -     -     -

4   -     -     -     -     -     -     -

5   -     -     -     -     -     -     -

6   -     -     -     -     -     -     -
```

```
Input
Please enter the column:
2
OK   Cancel
```

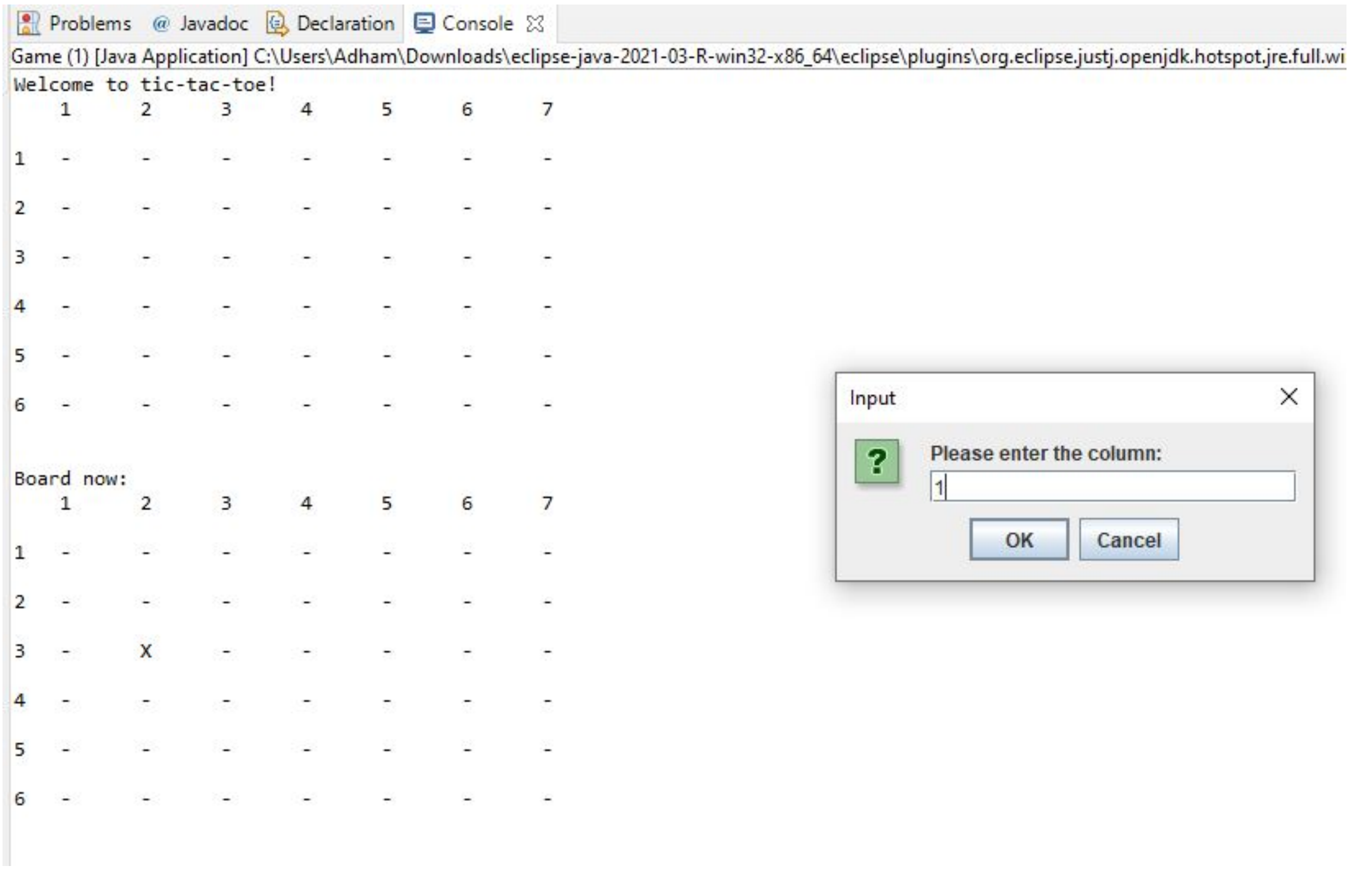
He will then be asked to enter his column choice



```
Game (1) [Java Application] C:\Users\Adham\Downloads\eclipse-java-2021-03-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x
Welcome to tic-tac-toe!
    1     2     3     4     5     6     7

1   -     -     -     -     -     -     -

2   -     -     -     -     -     -     -

3   -     -     -     -     -     -     -

4   -     -     -     -     -     -     -

5   -     -     -     -     -     -     -

6   -     -     -     -     -     -     -


Board now:
    1     2     3     4     5     6     7

1   -     -     -     -     -     -     -

2   -     -     -     -     -     -     -

3   -     X     -     -     -     -     -

4   -     -     -     -     -     -     -

5   -     -     -     -     -     -     -

6   -     -     -     -     -     -     -
```

```
Input
Hi Youssef Samuel  (Player 2) , please enter the row:
2
OK   Cancel
```

The cell was found empty so player 1 symbol ‘X’ was placed in it. The board is displayed thereafter and player 2 is asked to enter his preferences.



Problems @ Javadoc Declaration Console

```
Game (1) [Java Application] C:\Users\Adham\Downloads\eclipse-java-2021-03-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.wi
Welcome to tic-tac-toe!
    1     2     3     4     5     6     7

1   -     -     -     -     -     -     -

2   -     -     -     -     -     -     -

3   -     -     -     -     -     -     -

4   -     -     -     -     -     -     -

5   -     -     -     -     -     -     -

6   -     -     -     -     -     -     -


Board now:
    1     2     3     4     5     6     7

1   -     -     -     -     -     -     -

2   -     -     -     -     -     -     -

3   -     X     -     -     -     -     -

4   -     -     -     -     -     -     -

5   -     -     -     -     -     -     -

6   -     -     -     -     -     -     -
```

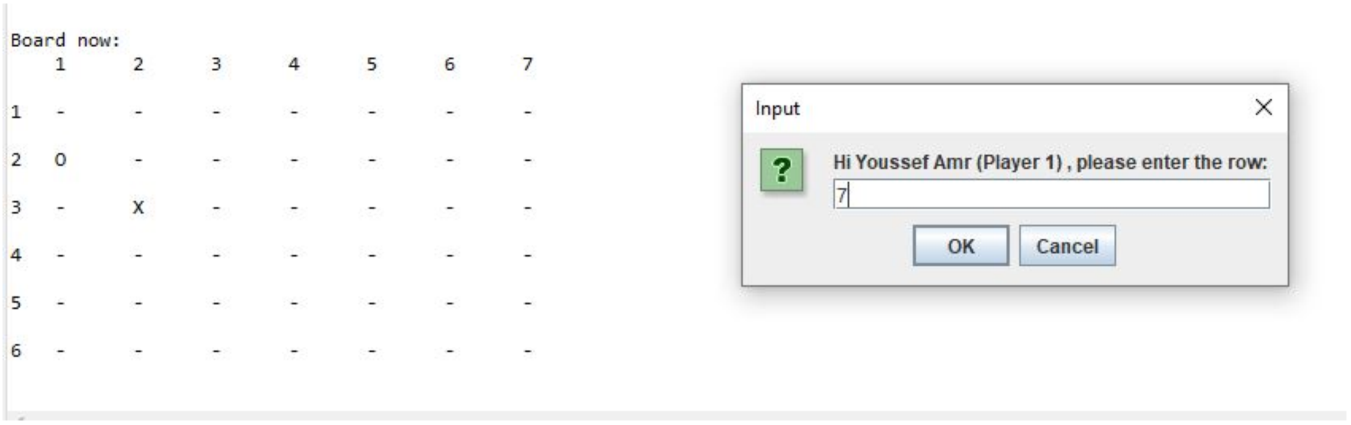
Input

Please enter the column:

1

OK Cancel

Player 2 symbol is placed in his cell choice [2][1].



```
Board now:
    1     2     3     4     5     6     7

1   -     -     -     -     -     -     -

2   O     -     -     -     -     -     -

3   -     X     -     -     -     -     -

4   -     -     -     -     -     -     -

5   -     -     -     -     -     -     -

6   -     -     -     -     -     -     -
```

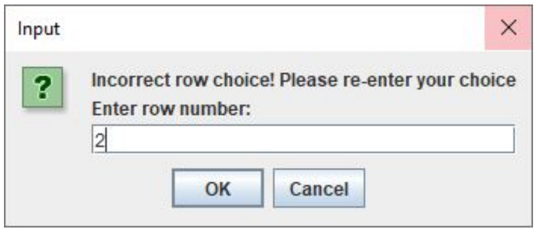
Input

Hi Youssef Amr (Player 1) , please enter the row:

7

OK Cancel

Player 1 is then asked to enter his row choice. If he attempts to enter an invalid index number, an error message would be appear and he would be asked to re-enter his choice.

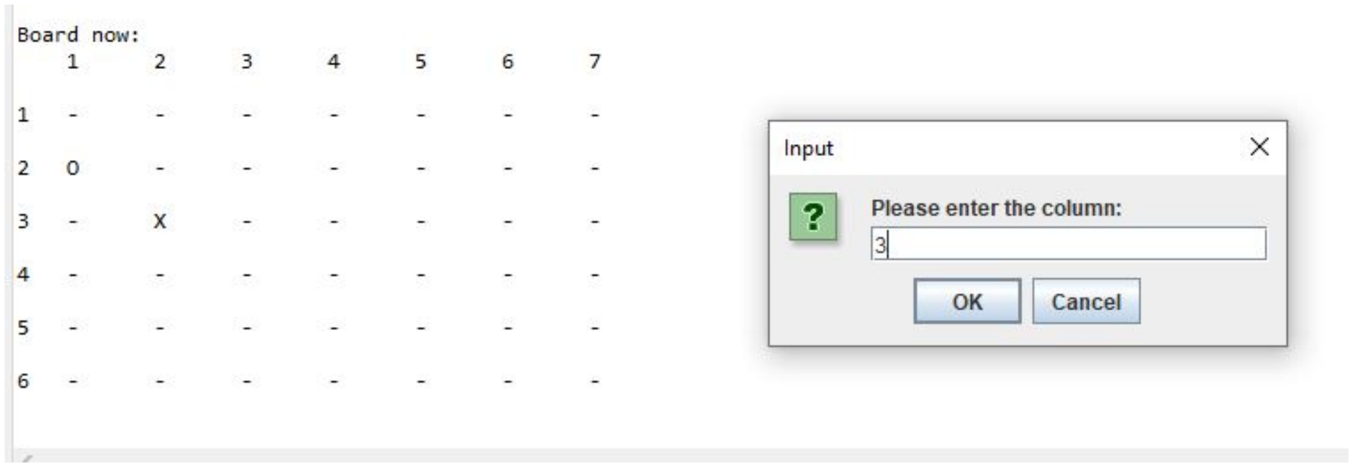


Input

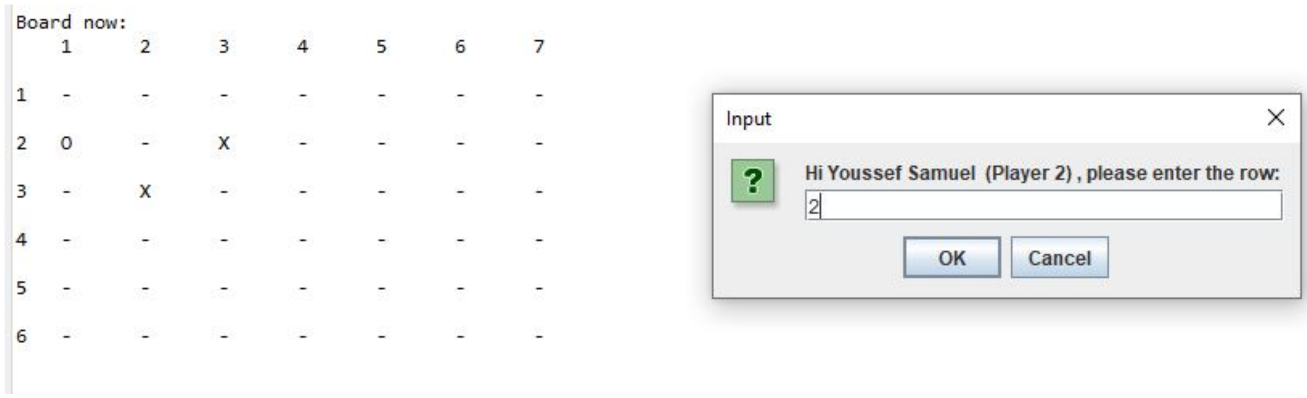
Incorrect row choice! Please re-enter your choice
Enter row number:

2

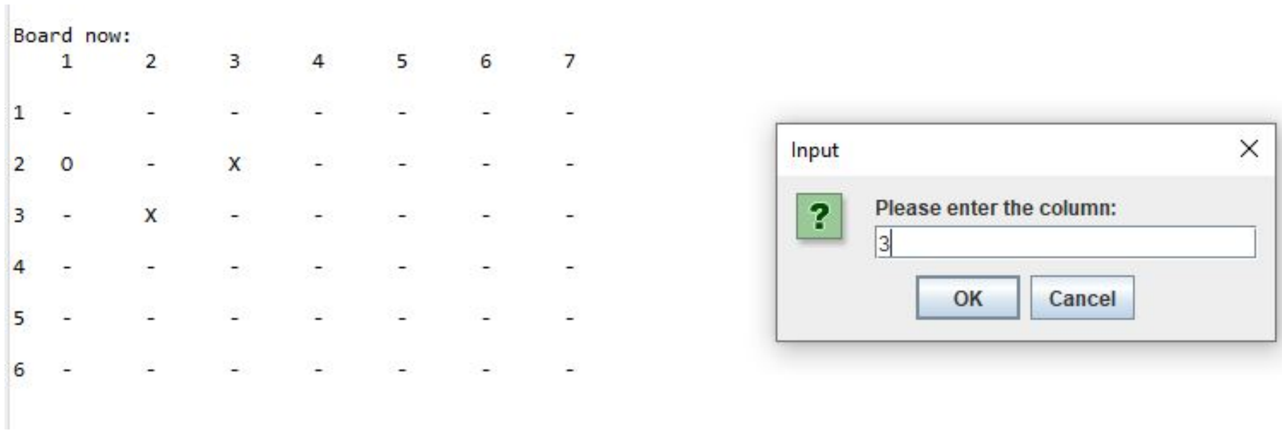
OK Cancel



```
Board now:
    1       2       3       4       5       6       7

1   -       -       -       -       -       -       -

2   O       -       -       -       -       -       -

3   -       X       -       -       -       -       -

4   -       -       -       -       -       -       -

5   -       -       -       -       -       -       -

6   -       -       -       -       -       -       -
```

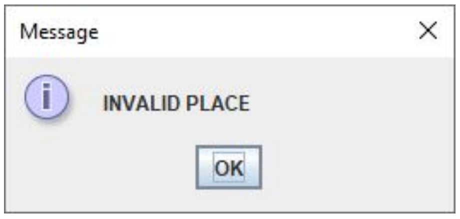


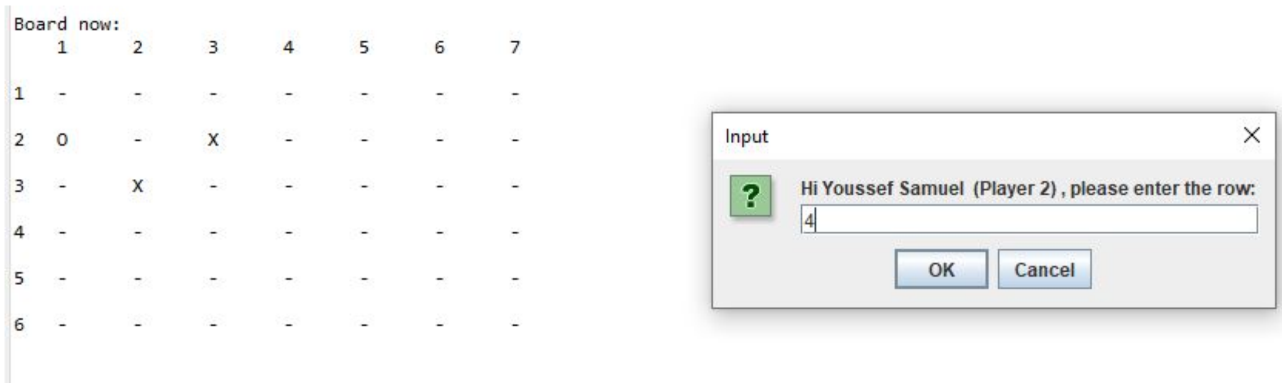
```
Board now:
    1       2       3       4       5       6       7

1   -       -       -       -       -       -       -

2   O       -       X       -       -       -       -

3   -       X       -       -       -       -       -

4   -       -       -       -       -       -       -

5   -       -       -       -       -       -       -

6   -       -       -       -       -       -       -
```



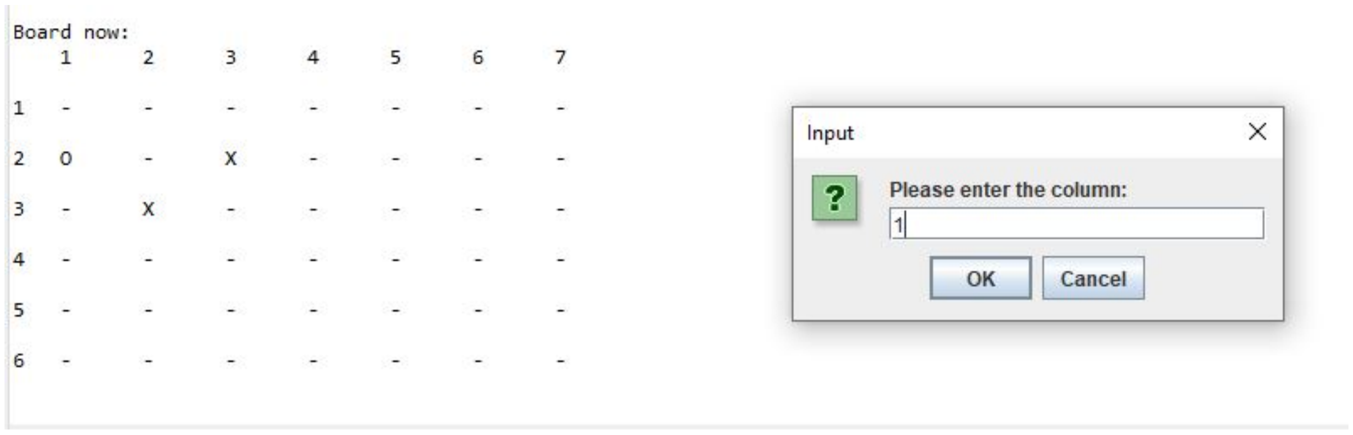
```
Board now:
    1       2       3       4       5       6       7

1   -       -       -       -       -       -       -

2   O       -       X       -       -       -       -

3   -       X       -       -       -       -       -

4   -       -       -       -       -       -       -

5   -       -       -       -       -       -       -

6   -       -       -       -       -       -       -
```

Player 2 is then asked to enter his choice for his cell after a successful insertion from player 1. If he attempts to enter the row and column for a preoccupied cell, an error message would appear an he would be asked to reenter his choices.



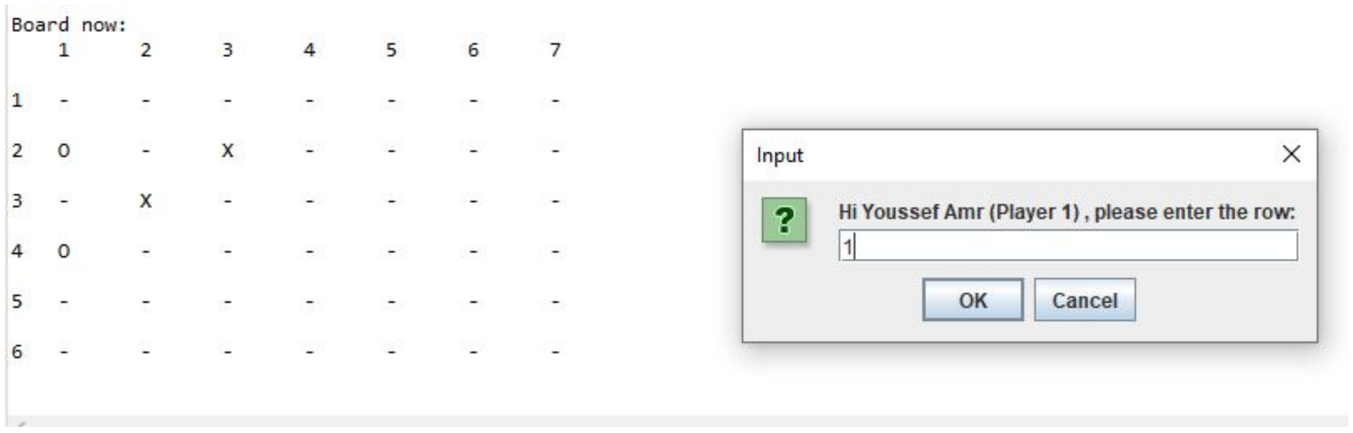


```
Board now:
    1      2      3      4      5      6      7

1   -      -      -      -      -      -      -

2   O      -      X      -      -      -      -

3   -      X      -      -      -      -      -

4   -      -      -      -      -      -      -

5   -      -      -      -      -      -      -

6   -      -      -      -      -      -      -
```



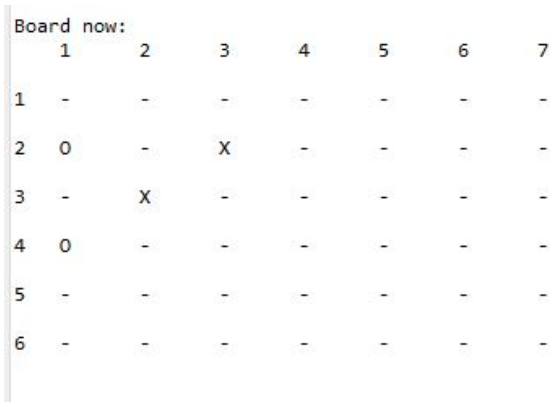
```
Board now:
    1      2      3      4      5      6      7

1   -      -      -      -      -      -      -

2   O      -      X      -      -      -      -

3   -      X      -      -      -      -      -

4   -      -      -      -      -      -      -

5   -      -      -      -      -      -      -

6   -      -      -      -      -      -      -
```

Player 2 successfully inserts his symbol in cell [4][1].

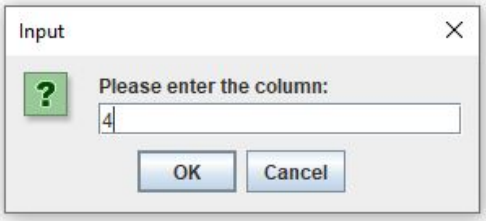


```
Board now:
    1      2      3      4      5      6      7

1   -      -      -      -      -      -      -

2   O      -      X      -      -      -      -

3   -      X      -      -      -      -      -

4   O      -      -      -      -      -      -

5   -      -      -      -      -      -      -

6   -      -      -      -      -      -      -
```

Player 1 is then prompted to enter his row choice.



```
Board now:
    1       2       3       4       5       6       7

1   -       -       -       -       -       -       -

2   O       -       X       -       -       -       -

3   -       X       -       -       -       -       -

4   O       -       -       -       -       -       -

5   -       -       -       -       -       -       -

6   -       -       -       -       -       -       -
```

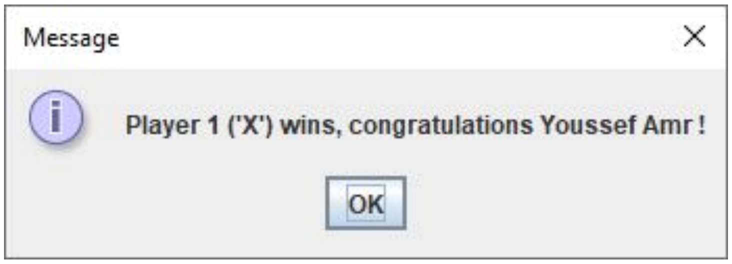


Input

Please enter the column:

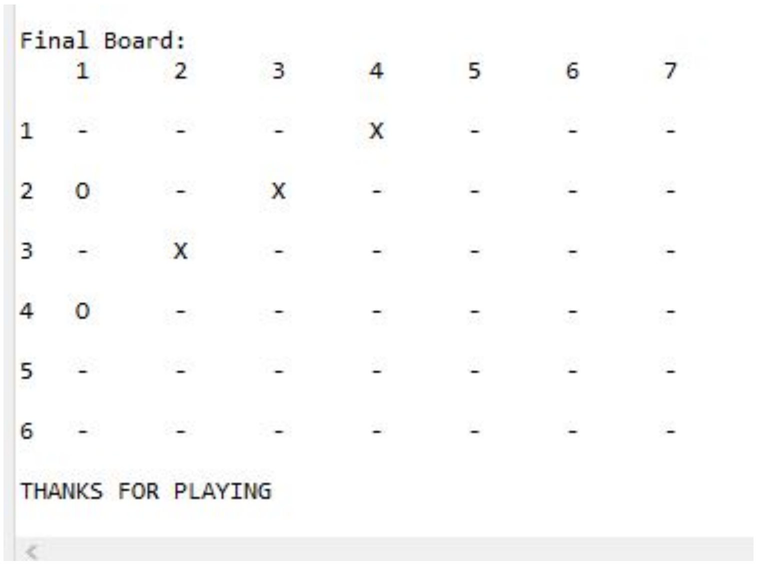
4

Upon checking for his cell choice [1][4], it resulted that this move made his win. A congratulations message would appear and the final board would then be printed.



Message

Player 1 ('X') wins, congratulations Youssef Amr !



```
Final Board:
    1       2       3       4       5       6       7

1   -       -       -       X       -       -       -

2   O       -       X       -       -       -       -

3   -       X       -       -       -       -       -

4   O       -       -       -       -       -       -

5   -       -       -       -       -       -       -

6   -       -       -       -       -       -       -

THANKS FOR PLAYING
```