CS136: Computer Science II – Fall 2019

Extra Credit	Points	Announced	Due
04	30[1]	Nov-11	Nov-22

Introduction

The purpose of this assignment is to exercise your algorithmic thinking.

General Guidelines

Read the following guidelines carefully before working on this assignment.

1. This is an *individual* homework assignment. You may discuss ideas, ask questions or explain things to your colleagues. Nevertheless, you should solve the problem(s) independently.
2. You should submit your *own work*. Material brought from elsewhere (e.g., the Internet[2], a classmate, submission at a previous offering…) is not acceptable.
3. A program with syntax errors (aka compilation errors) will receive *zero* points.

Submission Instructions

1. Submissions via email will not be accepted. The assignment should be submitted via BBLearn by the due date.
2. For question 1, submit Java files (i.e. with .java extension) with the names specified in the problem description. Other file types (e.g. .class, .zip, .jar, .doc, .pdf…) are not acceptable and will receive *zero* points.
3. Make sure that your code compiles and runs without errors when the supplied compilation and execution commands are used.
4. When you use an IDE (e.g., NetBeans, Eclipse…) for writing Java programs, the IDE will automatically use packages and add package statements to your code files. Java files with the package statement will compile but will not run when the below commands are used. So, make sure to remove the package statements from the code you are submitting.
5. Your code must have Javadoc-style comments for all classes, methods, and fields that you write.
6. Make sure that your code compiles without errors when the following command is used:

```
javac *.java
```

7. Make sure that your code runs without errors when the following command is used:

```
java PlayTicTacToe
```

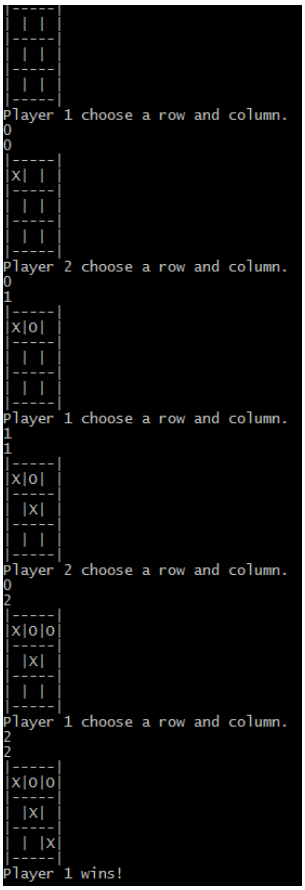
[1] The homework will be graded out of 30 points, but it is worth 3% (i.e. 3 points) of the overall course score.
[2] Unless explicitly asked to do so.

CS136: Computer Science II – Fall 2019

Penalties	
Item	**Points Deducted**
The program doesn't compile using the supplied command(s)	All
The program doesn't run using the supplied command(s)	All
Improper file format	All

[30 points] Question #1

Write a program that plays tic-tac-toe. The tic-tac-toe game is played on a 3 × 3 grid. The game is played by two players, who take turns. In our tic-tac-toe , the first player is always the X player, and the second one is the O player. The player who has formed a horizontal, vertical, or diagonal sequence of three marks wins. Your Java program should draw the game board, ask the user for the coordinates of the next mark, change the players after every successful move, and finally, pronounce the winner, or a draw. If a player enters invalid values for the mark row or column, your program should display an error message and ask the player to re-enter new values. A snapshot to the right shows a game.



Implement this game using two classes: `TicTacToe` and `PlayTicTacToe`. The class `TicTacToe` has all the data the logic needed for the game. The class `PlayTicTacToe` is the main class, and its sole task is to create a TicTacToe object and call its `play` method.

An option for representing the data of the `TicTacToe` board is to use a 3x3 array. A cell in the array can be 0, 1, or 2. 0 means an empty cell. 1 implies a cell marked by the X player. 2 means a cell marked by the O player.

The `TicTacToe` class should have these functions:

- `boolean isValidMove(int row, int col)` : Checks whether a move is a valid one. A move is valid if the `row` and `col` values are valid, and the cell is empty.
- `boolean wonDiagonal(int player)` : Checks whether `player` has won the game along diagonal lines.
- `boolean wonStraightLines(int player)` : Checks whether `player` has won the game along straight lines.
- `boolean win(int player)` : Checks whether `player` has won the game.
- `void drawBoard()` : Draws the board on the screen.
- `void play()` : Creates an empty board, draws it on the screen, and then keeps 1) asking for moves from the user, 2) checking the move validity, 3) checking for a winner,

4) drawing the board. If a winner is found, the winner is declared, and the game stops. If all cells get marked and a winner is not found, a draw is declared.

Grading Rubric	
Item	**Points**
Correct implementation of isValidMove	2
Correct implementation of wonDiagonal	5
Correct implementation of wonStraightLines	5
Correct implementation of win	2
Correct implementation of drawBoard	4
Correct implementation of play	10
Correct implementation of PlayTicTacToe	2
Missing JavaDoc comment (per occurrence)	-1

With best wishes
Dr. Mohamed Elwakil