Lab Homework #1: Tic-Tac-Toe Game [Total: 60]

Tic-Tac-Toe [60 marks]

Implement the game of Tic-Tac-Toe. Your game must consist of at least three (3) classes: Board, Player, CompPlayer, HumanPlayer, and TicTacToeApp. You can add more classes if you wish.

• The TicTacToeApp.java program will run the game, using a Board object and two Player objects.

• The Board class will be the board in the game. The board will consist of a 3x3 two dimensional array grid. The constructor of the Board class will create and initialize this array.

• The game board must be displayed before the player can make a move. The player can only choose the space that is empty. In the following, we demonstrate the display of the Board. The + indicates that these positions are empty and can be selected by the player. H stands for the human player move and C stands for the computer player move.

```
—————————————-
| H | + | + |
—————————————-
| + | C | H |
—————————————-
| H | C | + |
—————————————-
```

• The Player class will implement a player in the game. It consists of two private instance variables row, and col denoting the input for the next move.

• HumanPlayer will extend the Player class and allow for the user to input values for the row and col variables by using a Scanner object. User input will be two comma separated numbers. The first number is row (0, 1, or 2) and the second number is the column (0, 1, or 2). The input member function must verify the validity of the input. For example, (4, 3) would be an invalid input. Similarly, if the selected location from the user (row, col) is not empty in the board grid then the user should input these two values again in a loop.

• The CompPlayer will extend the Player class and automatically generate the next move. The class will examine the content of the Board and determine the move (row, col) that will increase the winning chance of the CompPlayer.
  - The CompPlayer will return the position of the next move (row, col)
  - CompPlayer should use the Board object in order to access the Board grid and calculate the next position for the move.

- The initial move can be randomized (when there are only one or no move on the Board). (1, 1) can be one such potential move.
- The CompPlayer will return the first position that increases the winning chance of the CompPlayer. For example, in the following Board configuration the CompPlayer should return the (0, 1) position as that would be the winning move.

```
-----------------
| H | + | + |
-----------------
| + | C | H |
-----------------
| + | C | + |
-----------------
```

- Similarly, the CompPlayer will select a position and ensure that it is not loosing the game. For example, in the following Board configuration the CompPlayer should return the (1, 0) position as that would be the defending move.

```
-----------------
| H | + | C |
-----------------
| + | C | H |
-----------------
| H | + | + |
-----------------
```

- Likewise, the CompPlayer will prioritize winning when selecting the next move. When there is a winning chance, it should select the winning move instead of defending the game. For example, in the following Board configuration the CompPlayer should return the (0, 1) position as that would be the winning move.

```
-----------------
| H | + | + |
-----------------
| + | C | H |
-----------------
| H | C | + |
-----------------
```

- After processing each move, the game should check the Board and determine win, loss, or tie status and report accordingly. The players will take turns. For example, the HumanPlayer will input the next move and then the CompPlayer will generate the next move. This sequence of turns will continue until the game reaches win, loss, or tie state — at this point the game should report the game status and provide option to the user to restart the game or exit the game.

Put your Board.java, TicTacToeApp.java, Player.java, HumanPlayer.java, CompPlayer.java and any other .java files you created in your submission.

The assignment rubrics for the work is given in the following:

Criterion	Details	Deductions
Classes	At minimum, we need the following java files Board.java, TicTacToeApp.java, Player.java, HumanPlayer.java, CompPlayer.java	The game does not let user input moves -40 The program does not validate the input -10 The computer does not generate any move -40 The game does not display the Board -30 The game can not determine win, loose, or draw status of the game -20 The program does not allow to restart/exit the game -20
Code quality	Identifier names, class names, proper use of public/private, ample comments in main, etc.	-15: incoherent, inconsistent coding style -10: comments were not used throughout the code
Test	Note, whether the program compile and run	The portion of the code of the game is a copy -60 The required java files not submitted -50 The program does not compile -60 -20: screenshot of the program run is not attached