Project 3

The goal of this project is to test your ability to use the knowledge of Java that you have obtained up to this point in the course.
The focus of the project is

- decision structures
- looping structures
- arrays
- and methods

Instructions

The goal of this project is to create Tic-Tac-Toe game.

Program Functionality

➢ Program Steps

1. Display the current board state.
2. Take in a row and column as a position on the board for the user's move.
    a. If the user enters anything besides a valid row and column, tell the user it is invalid.
        i. Start over from step 2.
    b. If the user enters a position that already has an "X" or an "O", then tell the user that the position is occupied.
        ii. Start over from step 2.
    c. If the user enters a valid, unoccupied row and column, this should be considered the user's move.
        i. Display the board, now updated with the player's move
        ii. If the user's move results in a win, tell the user that she has won.
            A. Terminate the program.
        iii. If the user's move results in a tie (cat), tell the user that the game has ended in a tie (cat).
            A. Terminate the program.
        iv. The computer opponent should then randomly choose an open spot on the board.
        v. Display the board, now updated with the computer's move
        vi. If the computer's move results in a win, tell the user that the computer has won.
            A. Terminate the program.
        vii. Go to step 2.

Details

- The board should be represented by a 3 x 3 (two dimensional) array of characters.
- It should be obvious to the user how to enter coordinates and what the state of the board is at any time (See “Example Execution” Section).
- The computer’s moves should be completely random at the least. If you want to do more intelligent move making, you may receive extra points.
- You are REQUIRED to write the below methods (aside from the main method).

    - **public static void printBoard(char[][] board):** prints the board to screen
    - **public static boolean isMoveValid(String move):** checks if the move is a valid one, and returns true if the parameter is a valid move on the board, otherwise returns false
    - **public static boolean submitMove(String move, char[][] board, char player):** adds the move to the board, and returns true if the space specified in the move is not taken. If the space is taken, do not add the move and return false
    - **public static boolean isWinner(char [][] board, char player)**: Returns true if the player whose marker is passed as a parameter has won the game, false otherwise.
    - **public static boolean isCat(char[][] board)**: Returns true if there is a CAT (tie), false otherwise.
    - **public static String generateMove():** Returns a randomly generated move (you can use this for computer move)

- In the example provided below, the user goes first. If you want to randomly select which player goes first, this is also acceptable and will get you bonus points.
- All the normal rules to tic-tac-toe apply to this game. If you do not know the rules of tic-tac-toe, or have a specific question on the rules, feel free to ask the instructor.

Additional Notes

- Gathering of data for this project is done in main and “passed” to the methods as parameters. You may NOT declare any variables outside of the method in which it is being used (i.e., do not create any class level variables to eliminate the need to pass values via the method parameters)
- Every method MUST have a method header of comments.

- Example Execution

```
************
Tic-Tac-Toe!
 ************
Please enter the column and then row of your move.

   A  B  C
1 |  |  |  |
--------
2 |  |  |  |
---------
3 |  |  |  |

Player Move (X): banana
Invalid Input: Please enter the column and row of your move (Example: A1).

Player Move (X): A1

   A  B  C
1 |X|  |  |
---------
2 |  |  |  |
---------
3 |  |  |  |

Computer Move (O): C2

   A  B  C
1 |X|  |  |
---------
2 |  |  |O|
---------
3 |  |  | |

Player Move (X): C2
The space entered is already taken.

Player Move (X): B2

   A  B  C
1 |X|  |  |
---------
2 |  |X|O|
---------
3 |  |  |  |
Computer Move (O): B1

    A  B  C
1 |X|O|  |
---------
2 |  |X|O|
---------
3 |  |  |  |

Player Move (X): A3

    A  B  C
1 |X|O|  |
---------
2 |  |X|O|
---------
3 |X|  |  |

Computer Move (O): A2

    A  B  C
1 |X|O|  |
---------
2 |O|X|O|
---------
3 |X|  |  |

Player Move (X): C3

     A  B  C
1 |X|O|  |
---------
2 |O|X|O|
---------
3 |X|  |X|

Congrats, You win!!!
```

Project to do list:

- ☐ **Step 1. Requirements**
  - Make sure you completely understand the requirements and ask questions if you need clarification.
  - Make sure that you pay close attention to detail and follow the instructions very carefully ☺
- ☐ **Step 2. Design**
  - **Write UML statements and an algorithm/flowchart** for the solution to the problem (**main method**) and all **other methods** in a word document named **Project3Design.docx**. At the very top of the document type YOUR name and CSCI-1250-xxx (where xxx is your section number) on the right-hand side.
  - Use this format for each method:

    Method :
    UML Entry:
    Algorithm (or Flowchart):

  - Remember: your algorithm is your road map – you will follow it when writing your code. DO NOT WRITE YOUR CODE FIRST AND THEN YOUR ALGORITHM/FLOWCHART! (It should NOT be the last thing that you do.)
- ☐ **Step 3. Implementation**
  - Create a **class called Project3**.
  - Use your algorithm as a starting point for your comments throughout your program
  - Write your program one step at a time, i.e. make sure one thing works before going on to something else.
  - Complete the documentation of the application by inserting comments and adhering to programming standards
    - 1250 Coding Standards posted under Course Content in D2L
    - Pay particular attention to indenting, no word-wrapping when printed and comments
    - All methods MUST have a method header of comments
- ☐ **Step 4. Testing**
  - Test your program to make sure that it works!
- ☐ **Step 5. Delivery**
  - Create a folder named **LastnameFirstnameProject3** (replace Lastname and Firstname with your own last and first name)
  - **Create the javadocs** of your program**.** This is a simple as running the following command at the command prompt: ***javadoc Project3.java***
    You should place the resulting javadoc files (html and related files) in a folder named **javadocs**

- Place the completed java source code (**Project3.java**), design document (**Project3Design.docx**) and **javadocs** folder in the folder created named **LastnameFirstnameProject3**
- Zip up the folder containing your source code, design document, and javadocs. It should be named **LastnameFirstnameProject3.zip** (again, your first and last name will be substituted here)
- Drop the **zipped folder** into the dropbox in D2L.
- This must be completed and submitted to the dropbox by the due date/time. All projects turned in after the due date will be penalized 10% each day late up to max three days.
- If you have not completed the project, submit whatever you have. Partial credit is better than no credit at all.