Project 2: TicTacToe

Due Mar 24, 2019 by 11:59pm **Points** 50 **Submitting** a website url

If you haven't done the Lab Assignment 5b, accept your GitHub project #2 assignment here:

Make sure you are logged into your GitHub account and please head over to the link, choose your myUSF username, and accept the assignment.

SECTION 02: https://classroom.github.com/a/vnwyIX3b

SECTION 01: https://classroom.github.com/a/8ODVZL4d

Click on the green "clone or download" button. Click "Use HTTPS". Copy the URL in the text box. Open your computer console/terminal, go into the directory where you want your lab assignment code to be stored (perhaps make a project2 folder), and type in "git clone" followed by your copied URL. For example:

```
git clone https://url.you.copied
```

Type in your username and password if prompted.

And voila, you have your own local copy on your computer of the starter code provided by the class repository. This is where you need to edit or create new java files for submission.

You may now ignore the HelloWorld.java program. It was solely for lab assignment 5b. However, use the git and GitHub instructions learned in lab 5b to know how to manipulate and submit your files.

Tic Tac Toe (50 points)

Write a class encapsulating the concept of a tic-tac-toe game as follows:

Player.java (10 points) - class represents a player

TicTacToe.java (20 points) - class represents the tictactoe board

TicTacToeDriver.java (20 points) - class that contains the logic of the game

- Two players will be playing, player 1 and player 2. Each is an object of the **Player** class. Maintain the player's name and the number of wins and losses they have accrued.
- The board, **TicTacToe** class, is represented by a 2-dimensional array of 9 integer elements.
  - The value 0 in the array indicates that this place is available, the value 1 indicates the space is occupied by player 1 and the value 2 indicates that this space is occupied by player 2.
- In the main method of your driver class, **TicTacToeDriver,** your program will simulate the tic-tac-toe game from the command line, doing the following:
  - Ask the players for the names.
  - Create a TicTacToe object and instantiate it.
  - Randomly designate who is player #1 and who is player #2.
  - In a loop, prompt for plays, as two integers *(int row, int col)*, from the user. At each iteration of the loop, you will need to call methods of the TicTacToe class to update the TicTacToe object's internal data. You need to keep track of who is playing (player 1 or 2), enforce the rules, check if either player has won the game.
  - If a player wins or there is a tie, you will need to exit the loop and present the result of the game.
- In your **TicTacToe** class, make sure to implement:
  - a default constructor instantiating the array representing the board.
  - a method that allows a player to make a move. This method needs the player number and the position played on the board.
  - a method checking if a play is legal (the spot desired is not taken).
  - a method checking if a player has won the game; you should break up this method into several methods if the code involved is too long. (for instance, check if a player has won the game by claiming the entire horizontal row).
  - a method that checks whether the game is a tie (if no player has won and all the squares have been played, the game is tied).
  - an implementation of the *public String toString()* method that will return a String that displays the current visual representation of the board at any time during the game. Make sure to use 'X' and 'O' instead of 1 and 2.
- BONUS:
  - Display the win/loss statistics of each player and allow players to play again if desired.

PROJECT 2 SUBMISSION:

From this point onwards, we will be using git and GitHub to submit your lab assignments and projects. You may use the website itself to upload your code, but using the command line gives you more control over what is going on. Also, the less you click around, the better!

Modify the README file, to include your name, date/time of submission and any information you want your grader to know (i.e. what works and what does not work.) Also, include any outside sources you used to complete this assignment (i.e. tutoring center, website, book).

Once you have pushed your final code onto Github for submission, please submit the URL of your repository on Canvas. This way, we can easily navigate to your code and see when you finished the project. Once you are done, do not resubmit (or push) your code once more. It will change the submission timestamp.

Recall that if you submit on time, you can possibly get 100% of the points. Past that, points will be docked for late submission.

WARNING:

It is easy to find and copy code for this project. However, try to figure this out on your own, especially since it is not a difficult problem to solve. We are trying to improve your coding and problem solving skills. You can only get better if you put in the time and effort, and actually do the work. Also, you have LOTS of time to complete this project.

GRADING:

ON TIME: If you submit on time, you will possibly get 50 points.

12 HOURS LATE: If you submit any time between midnight to 11:59am the next day (Monday), you can get a maximum of 40 points.

24 HOURS LATE: If you submit any time between noon to 11:59am the next day (the 24 hour mark), you can get a maximum of 30 points.

2 DAYS LATE: If you submit any time on Friday, you get a maximum of 20 points.

Beyond this, no submissions will be accepted.

‹ Previous Next ›

SUBMISSION

Submitted!
Mar 24, 2019 at 11:54pm
Submission Details
View the Original Page

Grade: 51 (50 pts possible)
Graded Anonymously: no

Comments:
+1 EXTRA CREDIT your program does not quit if wanting to quit the game, and does not keep scores updated if playing more than one game.
Arturo Galvan-Alarcon, Apr 9, 2019 at 12:09am