Mobile App Development - Spring 2021

Project 1: Tic-Tac-Toe

You will be making the game of Tic-Tac-Toe playing against the computer. You will work with a partner on the project. The repo contains a lot of starter code for the model, but is missing parts (marked with `TODO`) and has none of the views or controllers (except for the standard empty `Activity`).

The program must use 3 Activities:

1. Welcome Screen - the user is welcomed to the game and there is a spinner for the difficulty level (easy, medium, or hard) and a button to start the game
2. Play Screen - has a tic-tac-toe board with 9 buttons (you can use a table-layout for this like I did for the calculator). The buttons are initially blank. At the top or bottom there is a piece of text that says which pieces you are (X or O, make sure it isn't just the letter but actually tells the user what the letter means). In landscape mode the text is on one of the sides. You can click the buttons to place a piece. Obviously if the spot is already taken, nothing happens. Otherwise you play your piece.
3. Game Over Screen - once the round is over, a screen is shown telling you how it went (win, lose, or tie) and the overall number of wins by the player and the AI. There is a button to play again which goes back to the "Play Screen" activity starting a new round.

You must use the MVC pattern. Make sure to study the classes provided for the model:

- `Game` (extends `ViewModel`)
  Represents the entire game across many rounds. This class is entirely completed for you. This is the class you will primarily interact with from the activities (being the `ViewModel`) and will likely need nearly all of the methods implemented here.
- `Board`
  Represents a single game board with the pieces on it. The board is a 3x3 array of chars. The space char (`' '`) means an empty spot on the board while the `'X'` and `'O'` characters represent the player pieces. The `Game` class creates a new board for each round. This class is started by providing all of the public method declarations but not their code. You will need to write the code (look for the `TODO`s).
- `AI`
  Abstract base class for all of the different AIs. The main method here is the abstract `play()` method which each of the different AIs implement for different strategies. The other methods are utilities to be used by the various different AIs.
- `EasyAI` (extends `AI`)
  The easy AI simply plays randomly each turn.
- `MediumAI` (extends `AI`)
  The medium AI plays a winning move if available, then tries to block the opponent if they are about to win, and finally plays randomly. This AI is NOT complete, you must implement it. You may want to look at the methods available in the base AI class.
- `HardAI` (extends `AI`)
  The hard AI plays a "perfect" game - if you go first then you must play perfectly to win, if it will either tie or win. To accomplish this it follows the medium AI's rules but instead of playing randomly, it looks for locations to "fork" (generate two winning possibilities), block the opponent from forking, then tries the center, then tries a corner opposite an opponent's piece, any corner location, and finally any of the remaining empty locations.

Code Quality Requirements

- All classes and methods must be documented. All of the classes above are already documented, but every `Activity` (including the `MainActivity` if you keep it) also needs to be documented. Additionally, any private methods you add to the model classes need to be documented along with any methods added to the activities that are not `onCreate()`.
- Comment your code. In general, anything you would have to explain to a random student currently in CS2 unfamiliar with the project should be commented on.
- Do not add any additional public methods to the model. They are not needed.
- Reduce duplication and redundancy of code.
- Use good variable/field, method, `Intent` key, and resource names (strings and ids).
- Use good formatting (can just use Code > Reformat Code).
- Once you have completed a TODO remove the TODO comment (since it is done). You can use ⌘-Shift-F to search in all files and find them.

Suggested Approach

- Work on finishing the model first. The `Board` class is missing a ton and needs lots of help. Every single test fails (even the `AI` and `Game` ones) because they depend on the `Board` class.
- Create the activities next: start with the welcome screen which should be fairly easy and then move onto the play screen. Make sure you can transition from one to the next. The play screen is the most complicated (but mainly because the other two are extremely simple). Eventually you can add on the game over screen. The tricky part with this one is returning to the play screen and continuing the game.
- Before doing your final submission, run all tests, make sure all methods and classes are documented, look over your variable/field/method/key/resource names, and reformat your code.

Whenever you get stuck, ask for help. I can give just small nudges or confirm your direction so you don't waste hours of time going the wrong way, getting frustrated, and not learning.

You may be under the impression that getting help from a professor is a sign of weakness or a reason to be embarrassed. ***It absolutely is not***. In fact, quite the opposite. By asking questions you are telling me that you care and want to learn. I want everyone to ask questions. I ***expect*** everyone to ask questions.

Helpful Resources

You will likely need both of these things that haven't been covered in class yet so read up on them.

- String Resources with Parameterization: http://androiddevcorner.blogspot.com/2014/08/localized-getstring-with-parameters.html
- Using a String Array Resource with a Spinner: (easier than what we did in class) https://www.mysamplecode.com/2012/03/android-spinner-arrayadapter.html

Rubric (total of 100 points)

- 10 pts - Check-in on Feb 23rd, you must show satisfactory progress on this day in-class. Satisfactory progress would be most of `Board` working and an `Activity` or two started.
- 25 pts - Code quality. Yes, 25% of your grade is for code quality. See the section above for more info.
- 10 pts - Welcome Screen layout and `Activity`
- 20 pts - Play Screen layout and `Activity`
- 10 pts - Game Over Screen layout and `Activity`
- 20 pts - `Board` class
- 5 pts - `MediumAI` class

Extra Credit

- 2 pts for showing the final board on the Game Over Screen
- Up to 10 pts for adding a lot of polish to your program and making it look good. This is for exceptional polish, I expect a bit already for the standard grading. This includes updating the app icon, including images/icons in your program, making it user friendly and very nice looking, etc.