Othello Mini-Assignment
Due Date: Monday December 2nd at 11:59PM

Instructions

- Please read the handout extra carefully and play around with the Othello demo before you delve into these questions. We recommend completing this mini-assignment before you begin coding.
- Carefully think through the discussion questions and bring in your printed thoughts/responses to them. Be prepared to explain and discuss the tradeoffs of your design choices during section.
- As the name suggests, a “final project” is a lot more involved and bigger in size and complexity than your previous projects, so please give these questions a lot of careful thought. Trust us, that will save you time while coding.
- A reminder that the diagrams can be drawn in the digital program of your choice, but they cannot be hand-drawn.

Questions

1. What data structure(s) will you use to model the board? What type of objects will these data structures hold? How will you use them (be specific)?

2. Create the following diagrams. As always, you are allowed to use any digital software to draw this as you wish, but the diagram may not be hand-drawn.
   a. A complete inheritance diagram. Include any interfaces you will write or use.
   b. A logical containment diagram for your program. All classes used in your proposed design should be modelled in this diagram, including JavaFX classes. Think about the purpose of each class as you design your program and be prepared to discuss it.

3. How will you keep track of whose turn it is?

4. What is a copy constructor? How do you use it? What class would you want to use it for? Write an empty class with a copy constructor.

5. Provide pseudocode for the following methods:
   a. A recursive MiniMax AI algorithm. What parameters will this method take in?
i. Now that you have a recursive MiniMax AI algorithm, provide another section of pseudocode for how the compiler will actually read your method. As in, what is the order of lines that will actually be carried out/what is the flow of control? These should be the same lines of code, but now there is repeated lines and they are in a different order. For reference: Lecture 3 slide 67 onwards

b. Checking for a “sandwich” (i.e., when there is a valid move).

c. **Note:** These algorithms, especially MiniMax, are more complicated that have seen before, so this will likely be a substantial amount of pseudocode. Your pseudocode should translate almost line by line to Java. If your pseudocode involves helper methods, be sure to include those in your pseudocode. Do not just regurgitate the pseudocode we give you in the handout! Think critically and write detailed pseudocode.

6. Plan out steps 7 and 8 of the Coding Incrementally section of the handout. Break each of those larger steps into concrete methods that can be implemented and tested. Othello is a complicated game that will be much easier to code if you can break it down into small, testable pieces!