Assignment 1 part 1: Othello Client UI.

Value: 8% of your overall grade.

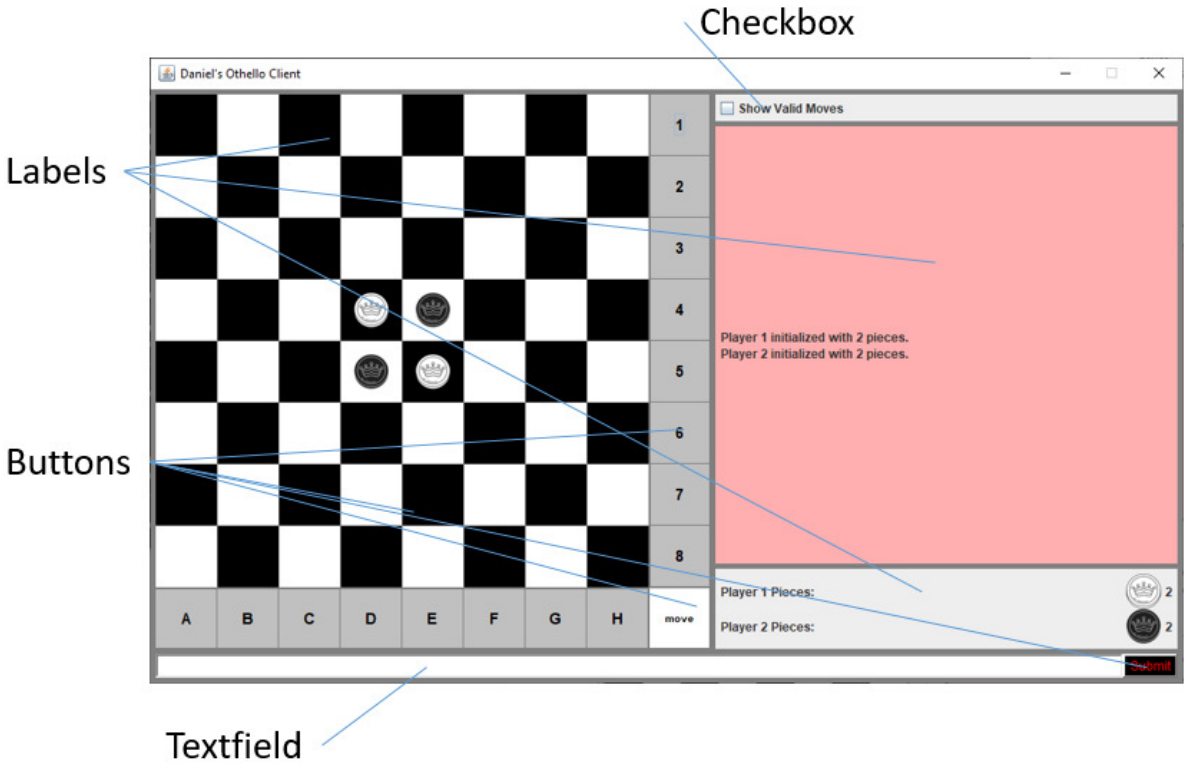
Due date: October 16th, 2020.

Purpose:

We will be initiating the development cycle of a product we want to push to market, a Java-based two-player Othello game. R&D has decided that Othello (historically also known as Reversi) will be very popular with the 18-27 demographic and we'd like to get our product out to market as soon as possible. You (and your team mate if you have one) have been hired as the UI coding specialist(s) to code the UI we have developed.

But seriously:

In this series of assignments, we will be creating an Othello game. This part of the assignment revolves around the creation of the UI. This assignment is based on material that has been covered in lectures, lab exercises, and hybrid activities. You need to replicate this UI as closely as you can, using the materials taught to you.



We'll be implementing the game logic in part 2 of this assignment, following the "Model-View-Controller" style. This is reflected in some of the requirements in this assignment, every requirement is here for a good reason.

Requirements:

Each square on the board must be 60x60 pixels, with black in the upper left corner. It will be helpful if you assign the black and white square colours to variables when drawing. The board must display the pieces as shown. Graphic files for the pieces will be provided to you. (You should add those images to your board in an area that's easily found in your code, as this will change for part 2 of this assignment.)

The pink zone to the right must have a preferred width of 450 pixels.

There are buttons arranged around the edges of the board, as depicted. These buttons must line up exactly with the board. These buttons are light gray.

The font on the "move" button must be 10 pt and otherwise be the default font. All other fonts must remain at the default size. This button has a white background.

The "submit" button has red font and a black background.

Thick borders are five pixels wide, in gray. There is a two-pixel border separating the buttons from the border, and a one-pixel border between the buttons.

The code must use the default look and feel.

The frame (or stage) must not be resizable.

On launch, the UI focus should be on the text field.

Your board labels must be contained in an 8x8 two-dimensional array.

All pieces should be centered in their labels (graphics for the pieces will be provided for you).

Your name (and that of your partner) should appear in the title bar, along with "Othello Client".

The overall UI should be around 1010 x 620 (± a small number of pixels).

Code:

You will need three classes for this assignment: *Othello*, *OthelloSplashScreen*, and *OthelloViewController*. All code must be in package "*othello*". (Case is important.)

The Othello class is the launch point. It is responsible for creating and launching your splash screen. You will find Lab 2 very useful for this purpose. It will then instantiate OthelloViewController and use it to complete the Stage/Frame.

OthelloViewController will assemble the UI, primarily in its default constructor. *NOTE: If you are using Swing*, OthelloViewController must extend JFrame. You may have to define your JFrame parameters in OthelloViewController instead of the Othello class.

If you are using JavaFX, OthelloViewController may extend Scene, Pane or any descended Pane subclass as you see fit.

It will be very helpful in the future if you define your board's square colours in fields (class "globals").

OthelloViewController must also have an inner class called Controller. You should instantiate Controller only once in your code.

The Controller class will have very basic code. It reads the action command (or user data) of the button that was pressed, and then prints it to the console. All buttons (and the checkbox) must have a unique action command.

Required method:

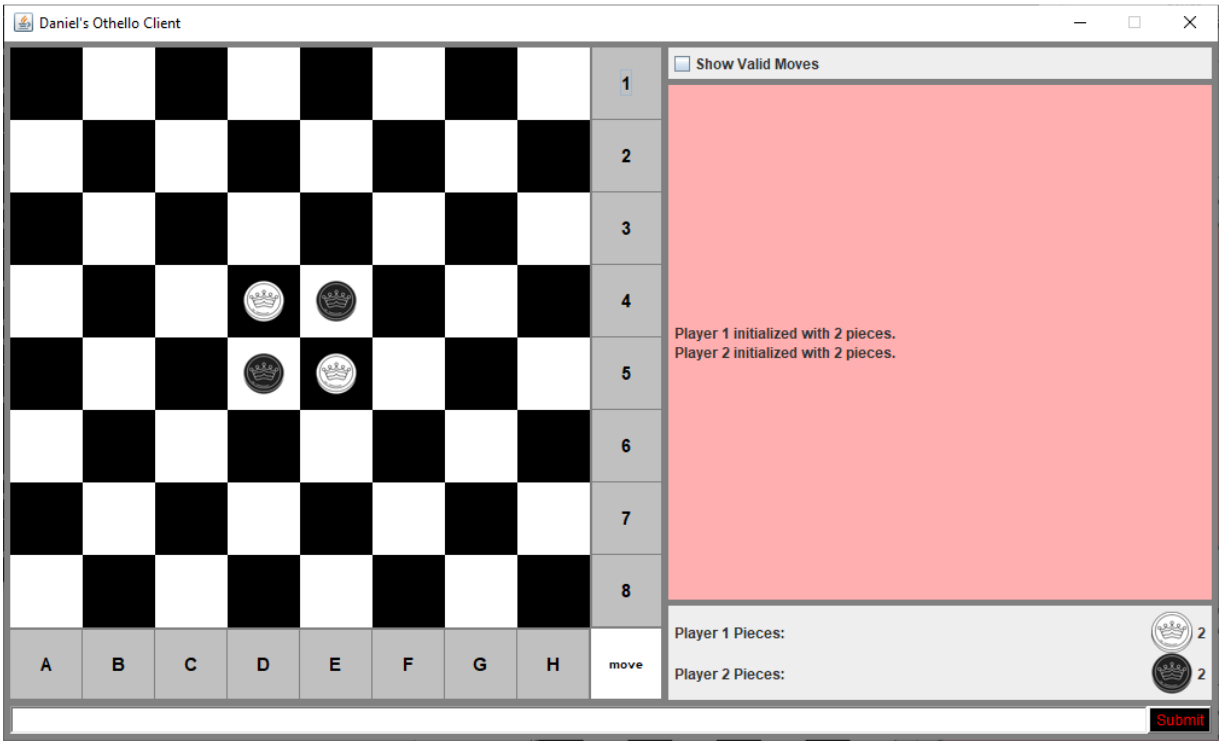
createButton(text, ac, fgc, bgc, handler)

- where "text" is the title of the button that the user will see.
- "ac" is the action command (or user data) for that button.
- "fgc" is the foreground (text) colour for the button.
- "bgc" is the background colour for the button.
- "handler" is registered as the event manager for that button.
- The method returns the newly created button.

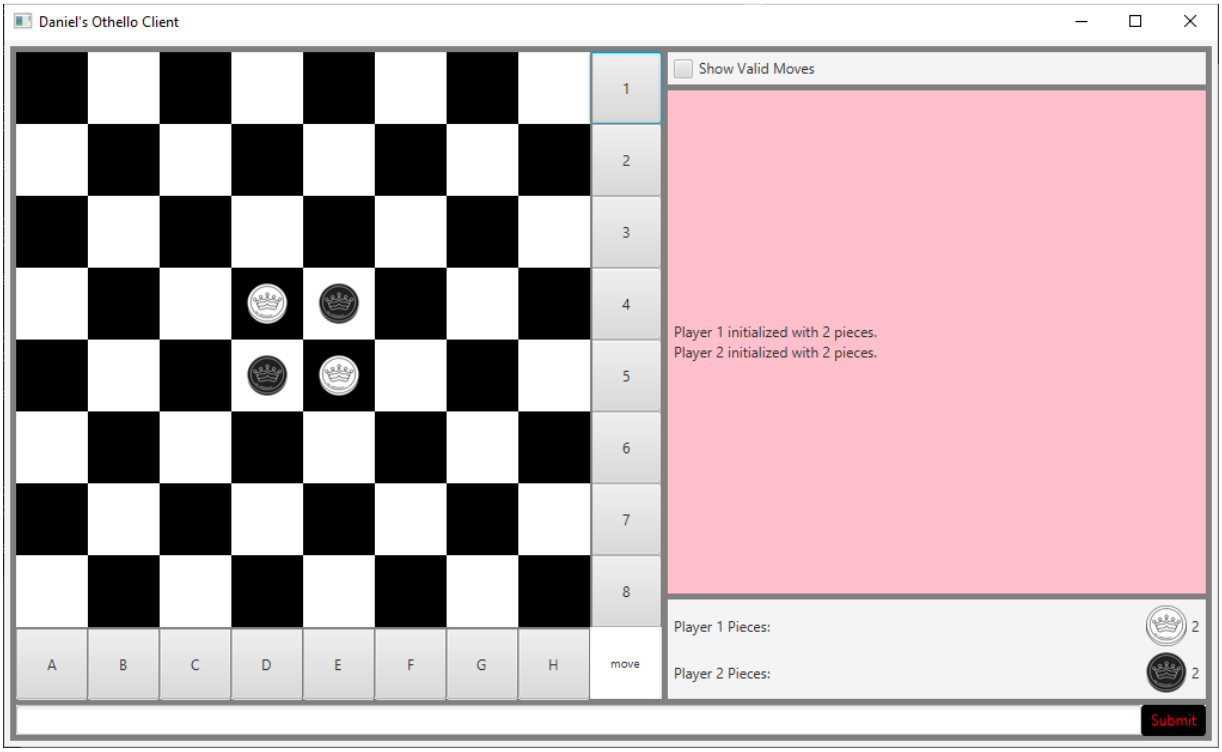
All buttons must be created with this method.

All screenshots were taken on a Windows 10 machine, which will be used for the marking. (See next page.)

Swing version:



JavaFX version



What to submit:

As per assignment submission guidelines, submit one zip file containing all the .class files, all the .java files, and all the image files.

All code must have the required headers, Javadoc comments, and be adequately commented as well. It should be neat and free of commented-out or still-active debug code.

Submit this zip file on Brightspace on or before the duedate.

Your code will be run from the command line, and I highly suggest you do the same. It should rely on no external libraries.

Final advice:

If you're in doubt about the implementation, try to make it look as close to the screenshots as possible. Your marks are largely based on how closely you can emulate it.

"*A picture is worth a thousand words. Or about 15 kilobytes of code.*" –Anonymous Java programmer.