Homework 3 (30 points)

cs1007 - Object-oriented programming and design in Java
Prof. Shlomo Hershkop
Dept of Computer Science
Columbia University
Spring 2006

Hw Due: Apr 17 11pm (electronically)

This homework is geared towards helping you understand how to work with objects in a dynamic environment while understanding what is happening behind the scenes.

You will be programming a graphical Othello game using java and an object oriented design approach. This assignment is very open ended, and you will be rewarded for originality as long as you complete the requirements.

1) Othello is sometimes called reversi
2) Although the default game for Othello is an 8 by 8 board, there is no reason it shouldn't work for any N by M board, you will need to ask the user for size before starting
3) The game should pause for 2 seconds before the computer executes its move, don't hard code this, so it should be easy for the user to change this value during the game play
4) You should be able to save and load a game at any point
5) I am providing a shell of the code to get you started, feel free to tweak it as you see fit
6) Although the reverse/Othello code is available everywhere on the internet, I expect you to write your own code….if you need help, you are expected to ask for it (from your instructor or TAs)
7) You will need to submit the final UML diagrams reflecting the actual working program, make sure you update it as you go, instead of scrambling at the last minute to make sure things are the same.
8) Your code will be organized in a single jar file, so that clicking on the jar file will launch the game, I will provide instructions in class for this.
9) Roughly the code I am providing for you to use is divided into 3 sections, this is one of many ways of logically dividing the code, feel free to change things as you see fit
    a. Board classes – these are classes related the actual board
        i. boardException – any exception thrown in the board, feel free to add more if you need
        ii. piece – class representing a single piece on the board
        iii. othelloMove – class representing an Othello move
        iv. abstractOthelloBoard – abstract class representing an Othello board
        v. othelloBoard – class which implements the actual board
            1. besides the methods I've sketched you need to define
                a. make sure serializable works
                b. equals
                c. toString
        vi. GraphicalOthelloBoard – class which is a graphical object and displays an internal Othello board to play the game
        vii. Piece – the individual piece in the game, notice the enum class here, and define equals

b. Othello – classes related to the game itself
    i. mainGameStart – the class with main in it
    ii. othelloException – exceptions to the logic playing the game
    iii. othelloLogic – the logic to play goes here, for example getting all valid moves, deciding

c. Player – all classes dealing with the players playing the game
    i. Playerinterface – you need to create an interface of what it means to be a player
    ii. simpleComputerplayer – will be the computer side of the game, using very simple logic to choose moves
    iii. feel free to find online a good strategy and implement it in another class (example complexComputerPlayer
    iv. humanPLayer – in this homework this will not mean anything, as you are going to be clicking as the human player, but you find this fun to define

NOTE: the code I am giving you will not compile directly since certain methods need to be defined etc.

Deliverables:

1) README – file with your name, a description of how to get things running etc if there are any issues with your program you would like the TA to know about, put a description here.
2) Othello.jar – all your .class files bundled in a jar executable (this will include a Manifest.txt file
3) Your source code tree
4) A graphic of your updated UML diagram.

Submission instructions are on the class website.

Please note, you need to configure your environment to get graphics working remotely if you choose to work off cunix from elsewhere. I will cover this in the next class.

Hint: Start early, and test often, and have fun!