Assignment 01: building a custom fully working Reversi widget

Introduction:

In this assignment you will be tasked with making a fully working widget that implements the game of Reversi. In this game you are tasked with covering the board with your pieces by converting your opponents pieces into those of your own. It is suggested that you find an online version of Reversi and play a few games first so you can understand the rules that must be implemented when you produce your own version of the game.

Notes:

Code must be submitted by 2013-11-17 at 23:55. All code submitted after this time will be marked in accordance with the standard late submission rules (these are in your student handbook). There is a source file layed out for you to use already as a starting point on the Moodle. All considered you will be writing somewhere between 160 to 180 lines of Java Swing code.

In addition to the marking scheme below I will penalise people for any of the following mistakes.

- code that fails to compile -20%
- missing source files -10%
- code that is not commented -10%

The problems below are not in any logical order. It is up to you to determine the order of implementation. working on problem 1 then 2 then 3 is a very bad idea. Try to figure out a way you can get these quickly tested and working at a time.

There are 172 units in this assessment (each one marking a line of code in my solution) so each unit is worth 0.58% out of 100%. please bear in mind that you may have more or less lines than my solution but once it does the job requested then it will still score full marks.

Problems:

01) Write a constructor for the Main class that sets a window size of 655x675 pixels, sets a title and the default close operation to exit on close. it should generate a ReversiWidget and attach it directly to the content pane (5 units)

02) In the Main class modify the main method generate a new Main object and set it to be visible (2 units)

03) In the ReversiWidget constructor generate 3 colour objects for cyan, black, and white using full RGB values, initialise the game state, and add a mouse listener (6 marks)

04) Modify ReversiWidget mousePressed method to determine a value for oldx and oldy that maps to a board position when the left mouse button is pressed. (4 units)

05) Modify ReversiWidget mouseReleased method to determine a value for a newx and a newy that maps to a board position where the left mouse button is released, if newx, newy match oldx, oldy a move should be attempted in that position (6 units)

06) Modigy ReversiWidget paintComponent to get a 2D graphics object, draw a cyan background, then draw the grid and the pieces. (5 units)

07) Modify the attemptMove method to do the following

- check if the game is in play if not return immediately (2 units)
- check if the current slot denoted by (x,y) is empty if not return immediately (2 units)
- check if there is an opposing player's piece in any of the adjacent 8 board positions, if not return immediately (12 units)
- see if there is a reverse chain available for (x,y) if not return immediately (3 units)
- place the piece and start the reversing process in all 8 directions if there is a chain available (9 units)
- swap the players update the player scores, see if the end game has been reached and repaint the widget (4 units)

08) Modify the checkPiece method to return the piece that is in the given x,y position. If the position is out of bounds then return -1 to indicate this condition (4 units)

09) Modify the determineChain method to do the following

- Check if there is an opposing player's piece immediately in the given direction (dx, dy). if no such piece exists then return immediately (4 units)
- keep searching in that direction until either a an edge or a current player's piece is found. If the current player's piece is found the return true, if an edge is found then return false (18 units)

10) Modify determineEndGame method to do the following

- If there are no empty spaces on the board then return true, print a console message stating this (4 units)
- If either player has lost all their pieces then return true, print a console message stating this (4 units)
- check all the empty slots with the current player to see if there is a valid move available. if there is no move available then end the game stating that there is no moves available for whoever is the current player. (15 units)
- if none of the previous three conditions are satisfied then return false (1 unit)

11) Modify the drawGrid method to do the following

- set the background colour of grid to be black (1 unit)
- draw 4 border lines around the edge of the board (4 units)
- draw all the intermediate horizontal and vertical lines to divide the board into an 8x8 board (4 units)

12) Modify the drawPieces method to do the following

- go through each cell of the board (4 units)
- if there is a number 1 piece in the current cell then draw a white piece (4 units)
- if there is a number 2 piece in the current cell then draw a black piece (3 units)

13) Modify the initialState method to do the following

- initialise all cells in the array to have a value of zero (3 units)
- set the middle 4 cells to have the start game condition of Reversi (2 units)
- set the scores for both players, set the game to be in play and set the current player (3 units)

14) Modify the reverseChainAvailable method to do the following

- check each of the adjacent 8 cells to see if there is a chain available (should use the determineChain method) and return the appropriate result (5 units)

15) Modify the reversePieces method to do the following.

- if the current player is player 1 then reverse all of player 2's pieces in that direction (8 units)
- if the current player is player 2 then reverse all of player 1's pieces in the given direction (7 units)

16) Modify the swapPlayers method to swap the current player (5 units)

17) Modify the updatePlayerScores method to go through each cell on the board and count up the current score of both players (10 units)