Checker Game Specifications

From: CPSC2150Fall2015
  Dr. Yvon Feaster

To: __________________________

Date: _________________________

Overall Goal:
Produce a high quality graphical user interface (GUI) networked checker game that supports two players and is portable to all computer architectures.

Specifications:
The final product should include as a minimal the following:

1. To ensure portability you must use Java as the programming language.

2. To provide a GUI you may choose to use the Java Swing libraries or JavaFX libraries. The final produce must use a GUI, this is not a text based game.

3. The checker game must follow the Standard U.S. Rules as stated below:
   a. Checkers is played by two players. Each player begins the game with 12 colored discs. (Typically, one set of pieces is black and the other red.)
   b. The board consists of 64 squares, alternating between 32 dark and 32 light squares. It is positioned so that each player has a light square on the right side corner closest to him or her.
   c. Each player places his or her pieces on the 12 dark squares closest to him or her.
   d. Black moves first. Players then alternate moves.
   e. Moves are allowed only on the dark squares, so pieces are always limited to forward moves (toward the opponent).
   f. A piece making a non-capturing move (not involving a jump) may move only one square.
   g. When a piece is captured, it is removed from the board.
   h. If a player is able to make a capture, there is no option – the jump must be made. If more than one capture is available, the player is free to choose whichever he or she prefers.
   i. When a piece reaches the furthest row from the player who controls that piece, it is crowned and becomes a king.
j. Kings are limited to moving diagonally, but may move both forward and backward. (Remember that single pieces, i.e. non-kings, are always limited to forward moves.)

k. Kings may combine jumps in several directions –forward and backward—on the same turn. Single pieces may shift direction diagonally during a multiple capture turn, but must always jump forward (toward the opponent).

l. A player wins the game when the opponent cannot make a move. In most cases, this is because all of the opponent’s pieces have been captured, but it could also be because all of his pieces are blocked in. The source of these rules were from the following website.

http://boardgames.about.com/cs/checkersdraughts/ht/play_checker.htm

4. The final version of the game must be networked. Meaning two people on the Clemson University Campus, each having a copy of the code must be able to play each other.

5. The checker pieces must be able to be dragged by the mouse to and from squares.

6. The checker piece must not to be able to be dragged to an unoccupiable space. Ex. A checker must not be able to be placed on a light colored space. Also, it should not be able to be placed between two squares. The checker should snap to the center of the closest occupiable square when the mouse click is released.

7. The basic colors for the board must be in the red and black family.

8. The basic colors of the checker pieces must be black and red.

9. The colors of the board squares and the checker pieces must be distinguishable.

10. The basic checker pieces must resemble a round standard checker piece. When a checker piece is crowned King you must change the look of the piece to clearly indicate it is a King. (EX. draw a star in the middle or a large K)

11. You must implement a text box that indicates the name of the player whose turn it is. Example: It is Alice’s turn or It is Bob’s turn.

12. If the player makes an illegal move you must alert the player, using a text box, that they have made an illegal move. If an illegal move is made, the checker should snap back to the square it was moved from.

13. Your code must be documented well.
14. You must use good software development techniques.

15. On the final version of the product code you must use java doc.

16. You must use Git – version control with this project.

To meet the specifications as stated above in a timely manner, I have developed three deliverables. You are expected to meet the deliverable timetable. To encourage deliverable deadlines to be met I offer 5% extra pay for each deliverable that is delivered within 48 hours prior to the deliverable's deadline. Payment will be given for each deliverable.

**Deliverable 1:**
**Due: September 23, 2015**

At a minimal, this deliverable has three basic requirements:

1. You should produce an 8 X 8 checkerboard with alternating light and dark colored squares, as described in the 3.b above and shown below.

2. You should also provide 12 light and 12 dark checkers. The checkers should be centered on the dark squares of the checkerboard.

3. You should allow the checkers to be dragged to any unoccupied dark square. If the checker is moved to an unoccupiable square it should snap back to the square it was moved from.
Deliverable 2:
Due: October 21, 2015

This deliverable has several basic requirements.

1. Revise the checkers to look more realistic, an example is provided below. You should also develop a creative way to determine the king. As an example, draw a star in the middle of the checker. This is not the only option, be creative.

2. Implement anti-aliasing to smooth the edges of the checkers and/or the squares.

3. This version of the game should have all rules implemented. The rules we will follow are based on the Standard U.S. Rules listed above.

4. Provide a text box that will print a message that will indicate whose turn it is. As an example, “It is Alice’s turn” or “It is Bob’s turn”.

5. If an illegal move is made display a message that the move was illegal and the checker should snap back to the square the checker was dragged from.

6. Display a message indicating the winner of the game.
Deliverable 3:
Due: November 25, 2015

In this deliverable you will add the remainder of the requirements for this product.

1. You will implement double-buffered graphics and resizing of the GUI-based checkers game.

2. You will add network playing to the checkers game. This will require you to implement sockets.

3. The player with the darker checker will go first. Take care to orient the board to show the local player pieces at the bottom of the screen and the NetPlayer at the top.

4. To prevent freezing during network communication you will need to implement threading.

5. You will implement java doc.

Optional Enhancements:

1. Allow players to chat with each other. See example below. (10 % extra payment)

2. Provide a drop down menu that allows the user to choose from various color schemes for the board and the checkers. (5% extra payment)

3. Drop down menu that will allow the user to choose other images, icons, and shapes for the checker. If you choose this option you must include an image or icon to represent the king. This option should also allow the two players to choose differing sets of icons or images. (5% extra payment)

4. Any other approved enhancement that you would like to implement. (% to be determined)
NOTE: This is a specification from the client. If there is anything that is not clear it is your responsibility, as the software developer, to contact the client to clarify the requirements.