CPSC2150 – Checkers

Team Name: Team

Team Member 1: Eli Boccolucci

Team Member 2: Jonathan Flander

Team Member 3: Steven Spivack

Team Member 4: Luke Miller

Functional Requirements: As a <userRole> I <what/need/can> <goal> so that <reason>

Functional Requirement User Stories:

1. As a player, I need the game to accept command line input so that I can interact with the game using a keyboard.
2. As a player, I need the game to print the current state of the board at the start of each turn so that I know where to move next.
3. As a player, I need a clear visual indicator when a piece is “Kinged” to help strategize my next move
4. As a player, once I “king” my piece I need to move in all diagonal directions, so I can begin getting my opponents pieces from the other direction.
5. As a player, I need to be able to jump an opponent's piece, so I can advance my piece and try to win.
6. As a player, I need the game to recognize when all of my opponents' pieces are captured or blocked, so the game can be concluded.
7. As a player, I need the game to ask if I want to play again, so I have the option to play again without rerunning the program.
8. As a player, I need the game to validate the borders of the board so that it can prevent illegal moves like pieces being placed out of bounds.
9. As a player, I need the game to prevent me from moving to a spot that is currently occupied or blocked so that I follow the rules.
10. As a player, I need the game to declare a winner when one player’s pieces are all captured or blocked so that I know who wins.
11. As a player, I need to be able to move my piece in 2/4 diagonal directions to move my piece forward.
12. As a player, I need the game to recognize that if I select a piece that isn’t mine the game wont allow me to move it.

Non-Functional Requirements:

1. 8x8 game board size.
2. The blocks on the board should alternate colors from black to white.
3. Should have 24 pieces (12 black & 12 red).
4. Pieces should only move diagonally NE, NW, SE, SW.
5. The game should alternate turns after each player makes a move.
6. The game should respond from the user input without any delays.
7. The game interface should be easy to follow/navigate for the user.
8. The game shouldn’t have any unexpected crashes
9. Minimize memory space.
10. Have no gameplay issues and send messages when errors occur.
11. The game should be able to handle a growing amount of checker pieces and squares.
12. The gameplay interface should stay consistent throughout the game as it progresses.
13. Should be written in the language java. **Couldn’t Find In Auto-Grader
14. Should be developed and runnable with JDK17 and Junit 4. **Couldn’t Find In Auto-Grader
15. The games should be fully compatible with both windowsOS and LinuxOS
16. The input and output should be done through the terminal.