SUMMARY:

The change color button only works before any moves are made.

DESCRIPTION:

The player can chance the colors for both the black and white chess pieces. This however only works properly when it is done before any moves are made. If colors are switched after a move has been made it does not change all pieces.

REPRODUCTION STEPS:

1. Run the chess program
2. Click a white piece and move it to any legal destination
3. Attempt to switch colors for the pieces via the button

EXPECTED BEHAVIOR:

The button should change colors for all pieces of that player whenever the button is clicked.

OBSERVED BEHAVIOR:

Piece colors only change if the color changes are made before any pieces have been moved. If a piece has been moved not all pieces will change color.

NOTES:

This problem is related to other defects in this sprint that all relate to the GUI class and how it displays information from the chess board in the backend to the GUI. It will be a major priority to fix in the next sprint.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SUMMARY:

The flip board button only work before any moves are made.

DESCRIPTION:

The player can flip the board to see it from the white player’s or the black player’s perspective. This however only works properly when it is done before any moves are made. If one attempts to flip the board after the first piece is moved it causes an error where only some of the pieces are flipped.

REPRODUCTION STEPS:

1. Run the chess program
2. Click a white piece and move it to any legal destination
3. Attempt to flip the board via the button.

EXPECTED BEHAVIOR:

The button should flip the board whenever the button is clicked.

OBSERVED BEHAVIOR:

The board will only properly flip before any pieces have been moved. If a piece has been moved not all pieces will end up in their correctly flipped position.

NOTES:

This problem is related to other defects in this sprint that all relate to the GUI class and how it displays information from the chess board in the backend to the GUI. It will be a major priority to fix in the next sprint.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SUMMARY:

The terminal will tell the user what piece exists on a given square and this information is stored in the backend. The GUI however is not properly synchronized and thus will not update the visual board with movement.

DESCRIPTION:

The player can make a move and be told whether or not the move is legal or not. The player is also forced to adhere to standard Chess turns where white moves and is then proceeded by black. The GUI however does not properly update with each move.

REPRODUCTION STEPS:

1. Run the chess program
2. Attempt to move a move any legal location.

EXPECTED BEHAVIOR:

The GUI should update with the piece being moved.

OBSERVED BEHAVIOR:

The backend recognizes that the piece has been moved, the GUI however is not updated.

NOTES:

This problem is related to other defects in this sprint that all relate to the GUI class and how it displays information from the chess board in the backend to the GUI. It will be a major priority to fix in the next sprint.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SUMMARY:

The terminal will tell the user what piece exists on a given square and this information is stored in the backend. The GUI however is not properly synchronized and thus will not update the visual board with movement.

DESCRIPTION:

The player can make a move and be told whether or not the move is legal or not. The player is also forced to adhere to standard Chess turns where white moves and is then proceeded by black. The GUI however does not properly update with each move.

REPRODUCTION STEPS:

1. Run the chess program
2. Attempt to move a move any legal location.

EXPECTED BEHAVIOR:

The GUI should update with the piece being moved.

OBSERVED BEHAVIOR:

The backend recognizes that the piece has been moved, the GUI however is not updated.

NOTES:

This problem is related to other defects in this sprint that all relate to the GUI class and how it displays information from the chess board in the backend to the GUI. It will be a major priority to fix in the next sprint.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SUMMARY:

The testCastle\_5test in the KingTest.java class fails.

DESCRIPTION:

Castling with the path being impeded is illegal in Chess but the program does not return false when attempting to make the move.

REPRODUCTION STEPS:

1. Change the assertTrue to assertFalse in line 168 in KingTest.java (Reason for change is in the notes)
2. Run gradle -test

EXPECTED BEHAVIOR:

The test should pass with assertFalse.

OBSERVED BEHAVIOR:

When line 168 in KingTest.java reads “assertFalse(test.movePiece(test\_square));” it fails.

NOTES:

This problem looks like it be fixed by refactoring the implementation of movePiece for the King class and having the Chess class validate the path for castling is not impeded, the destination is not occupied, and that the king never passes through a check while castling.

Line 168 has changed to assertTrue solely to pass the tests and run other parts of the program.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_