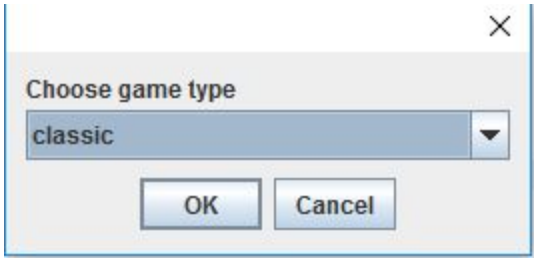
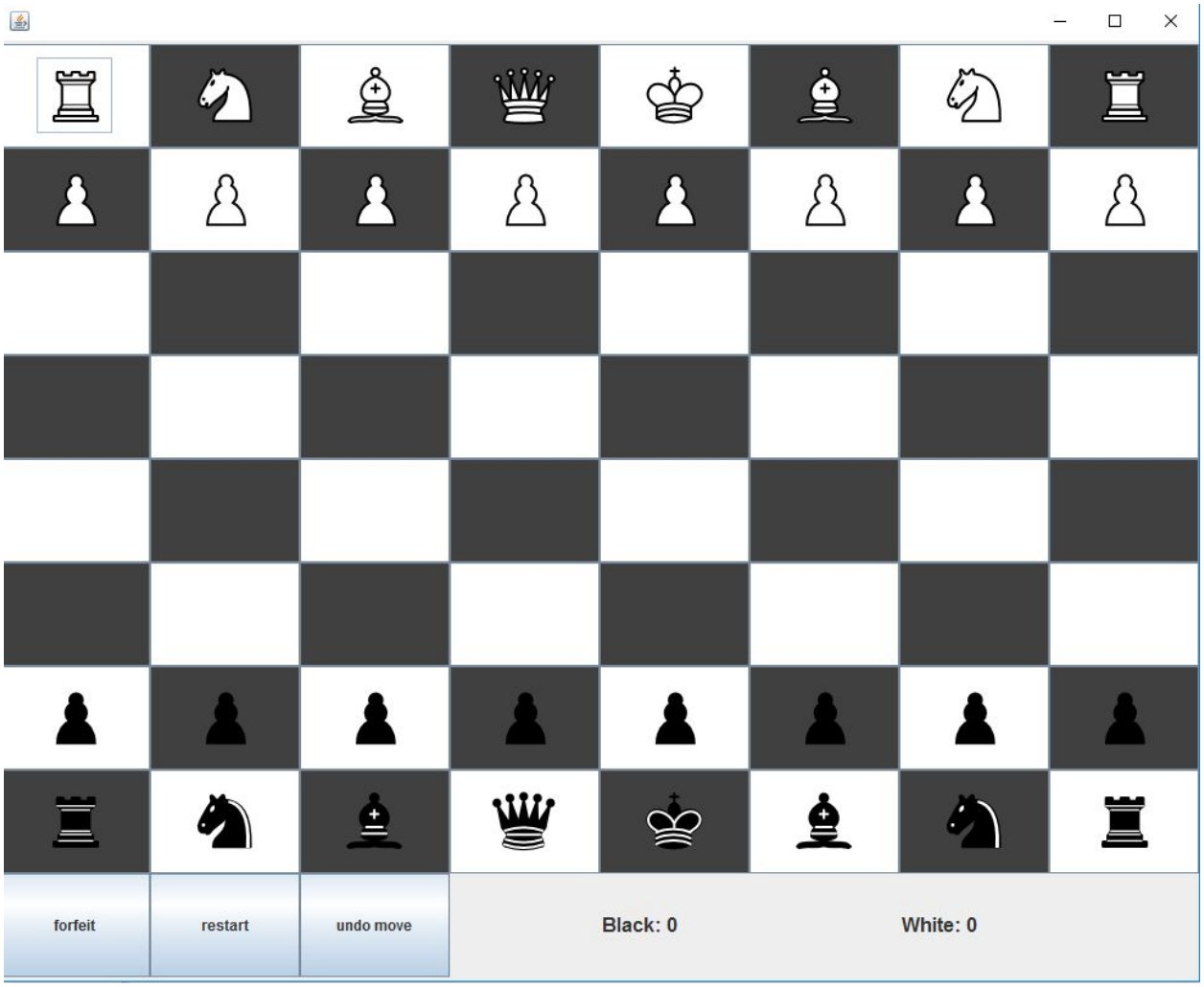
Run the game by running the ChessController.java as a java app. This should bring up the following pop-up:



For our first test, we will choose the classic game type so simply hit OK. This should bring up the following game board:

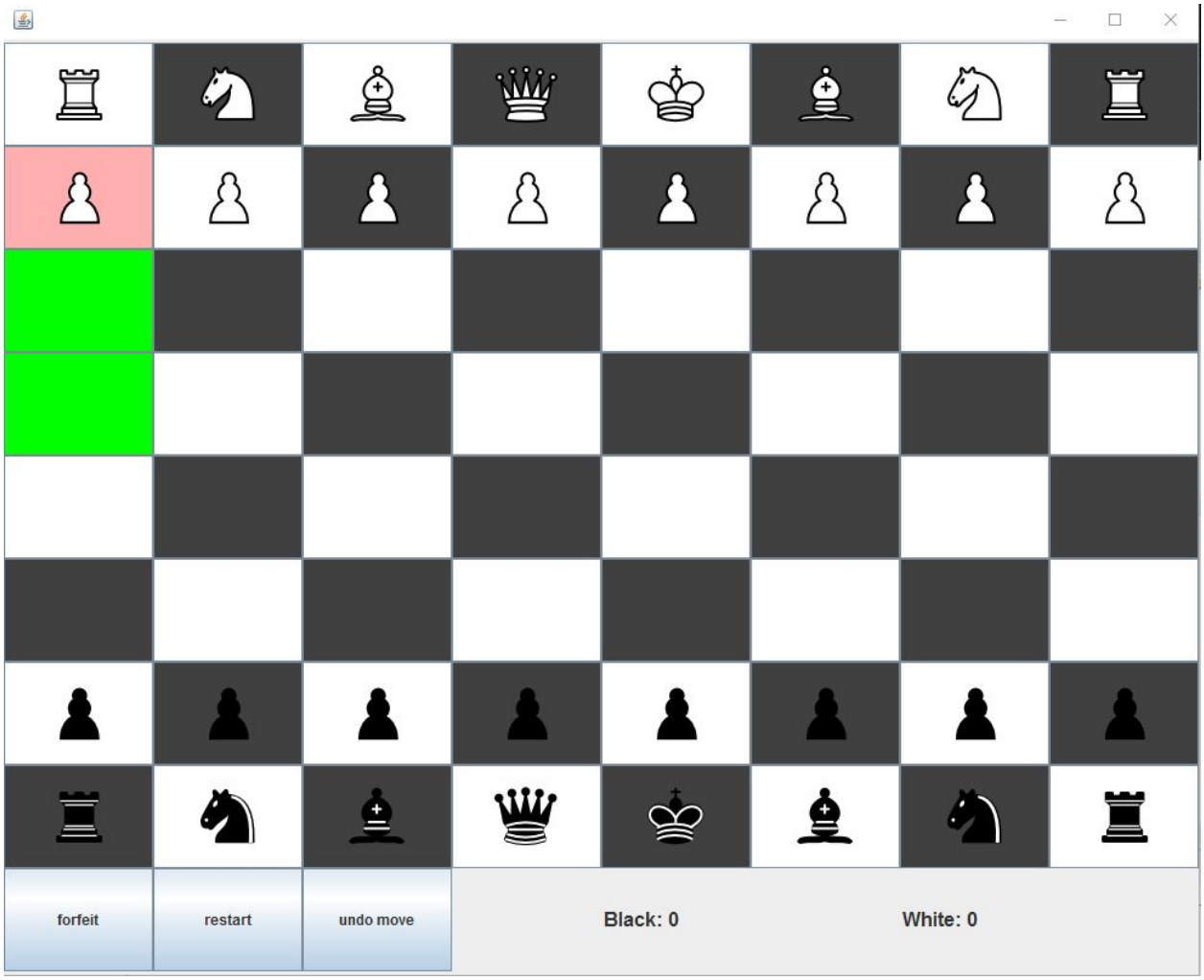


A couple things to immediately check:

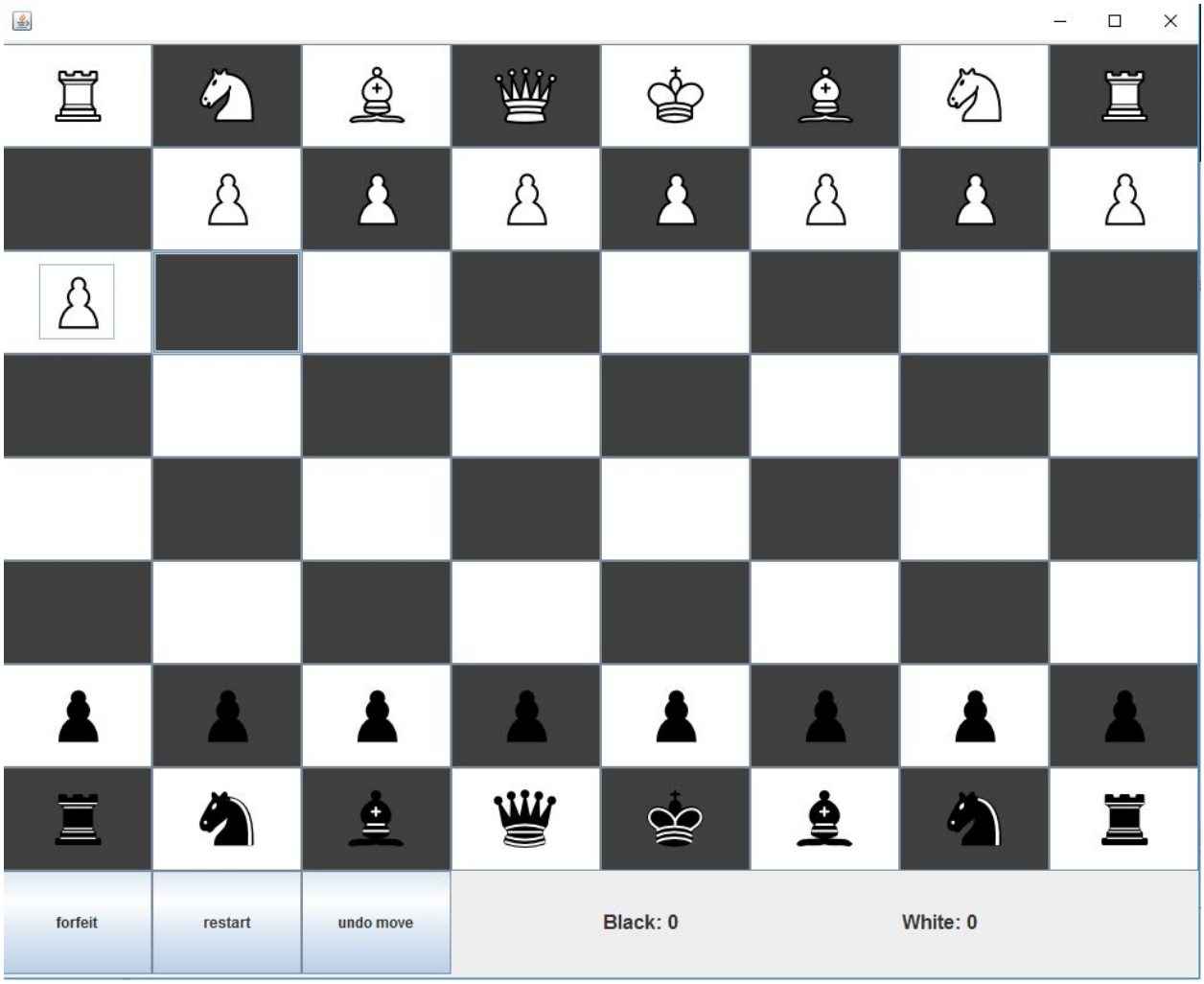
- The board should be 8x8

- The score for the black and white players should both be 0 in the bottom right
- The pieces should all be in the correct place for the standard rules of chess.

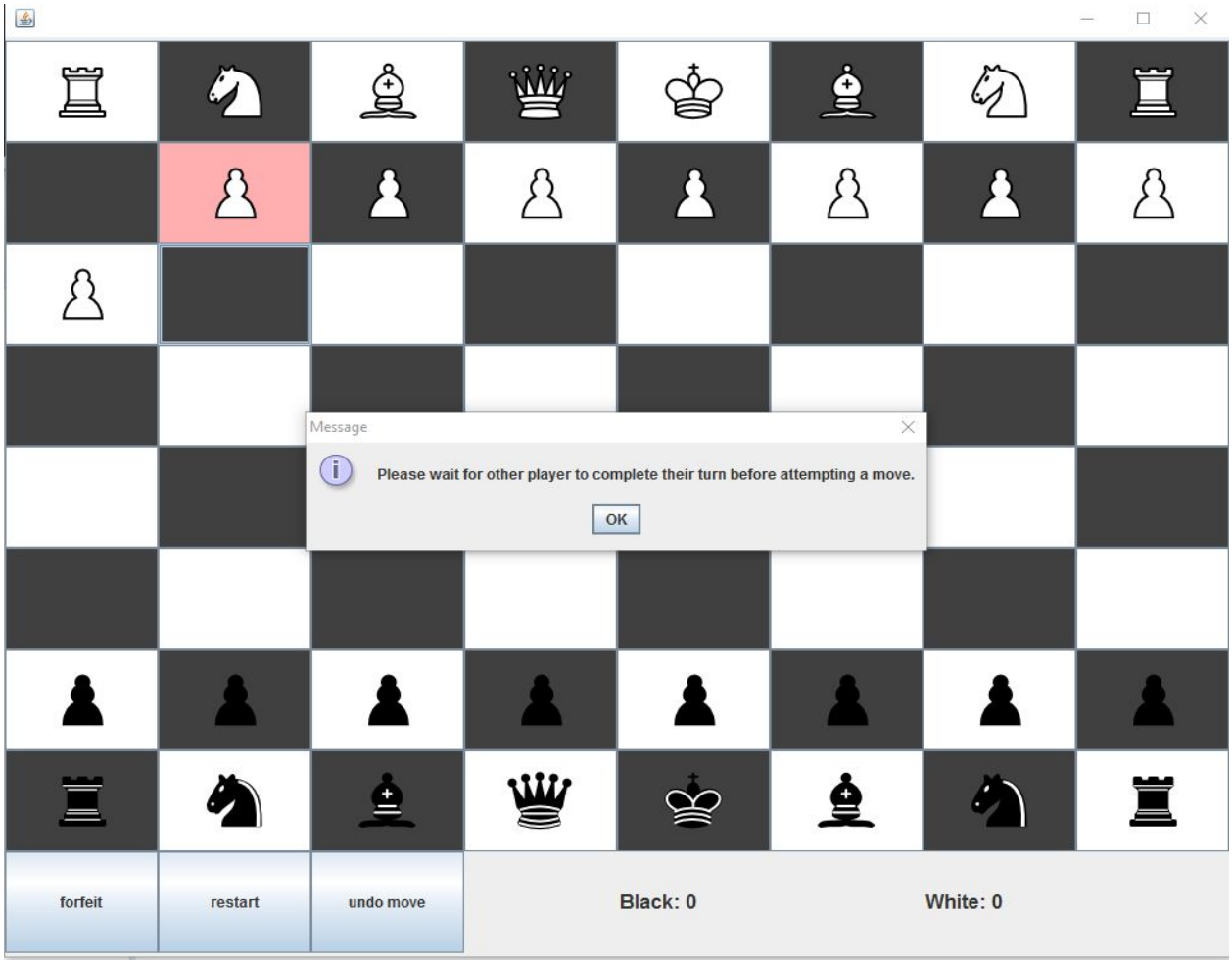
Next we will attempt to make our first move (white always goes first). Click on the leftmost white pawn. The pawn should then be highlighted pink, and the valid tiles it can move to should be highlighted green as seen below:



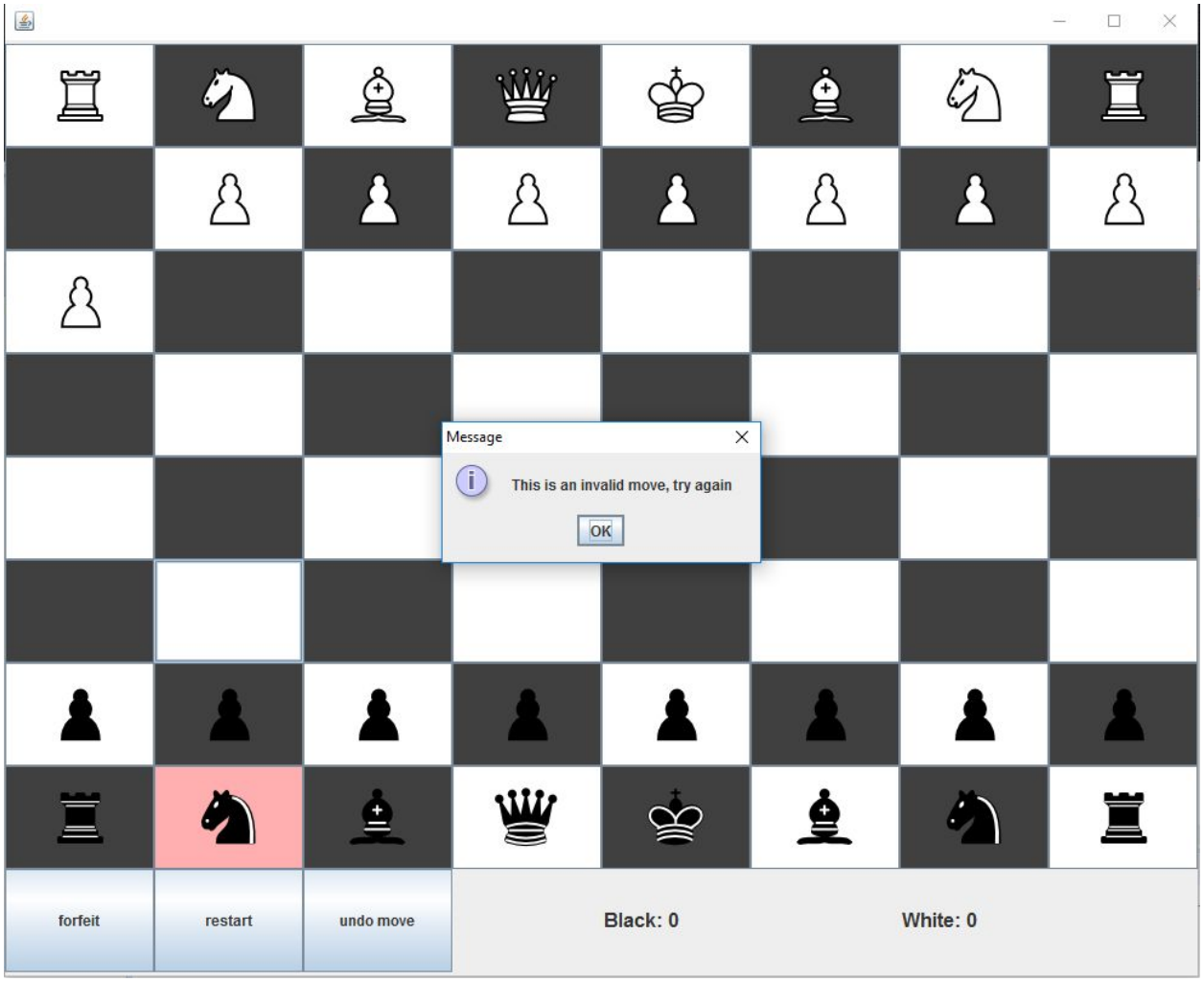
Now click on the first green tile to move the piece. This move should be reflected on the GUI:



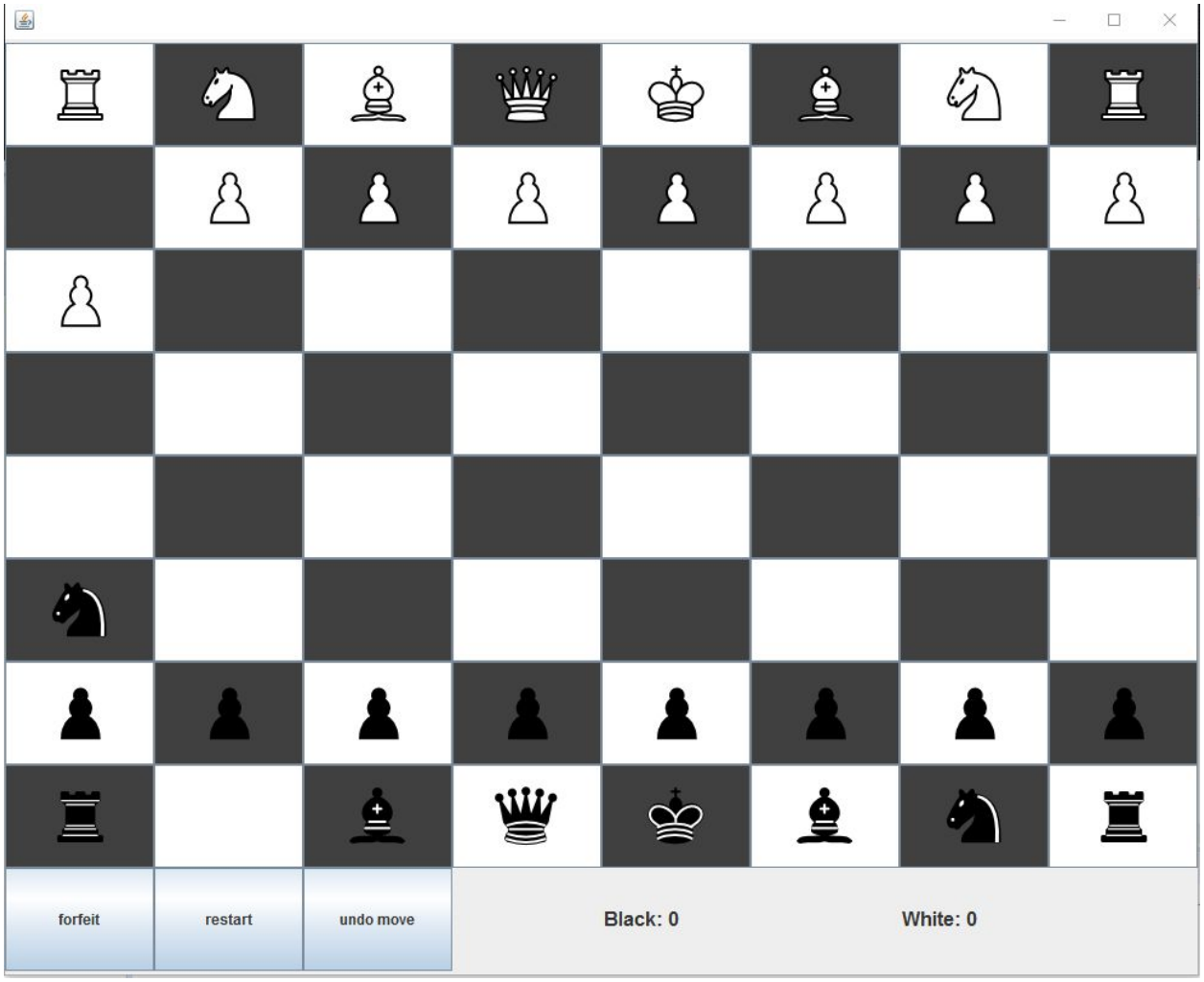
At this point let’s try to make another move with a white piece. Say the pawn next to the one we just moved. This should bring up an error message, since it is no longer white’s turn:



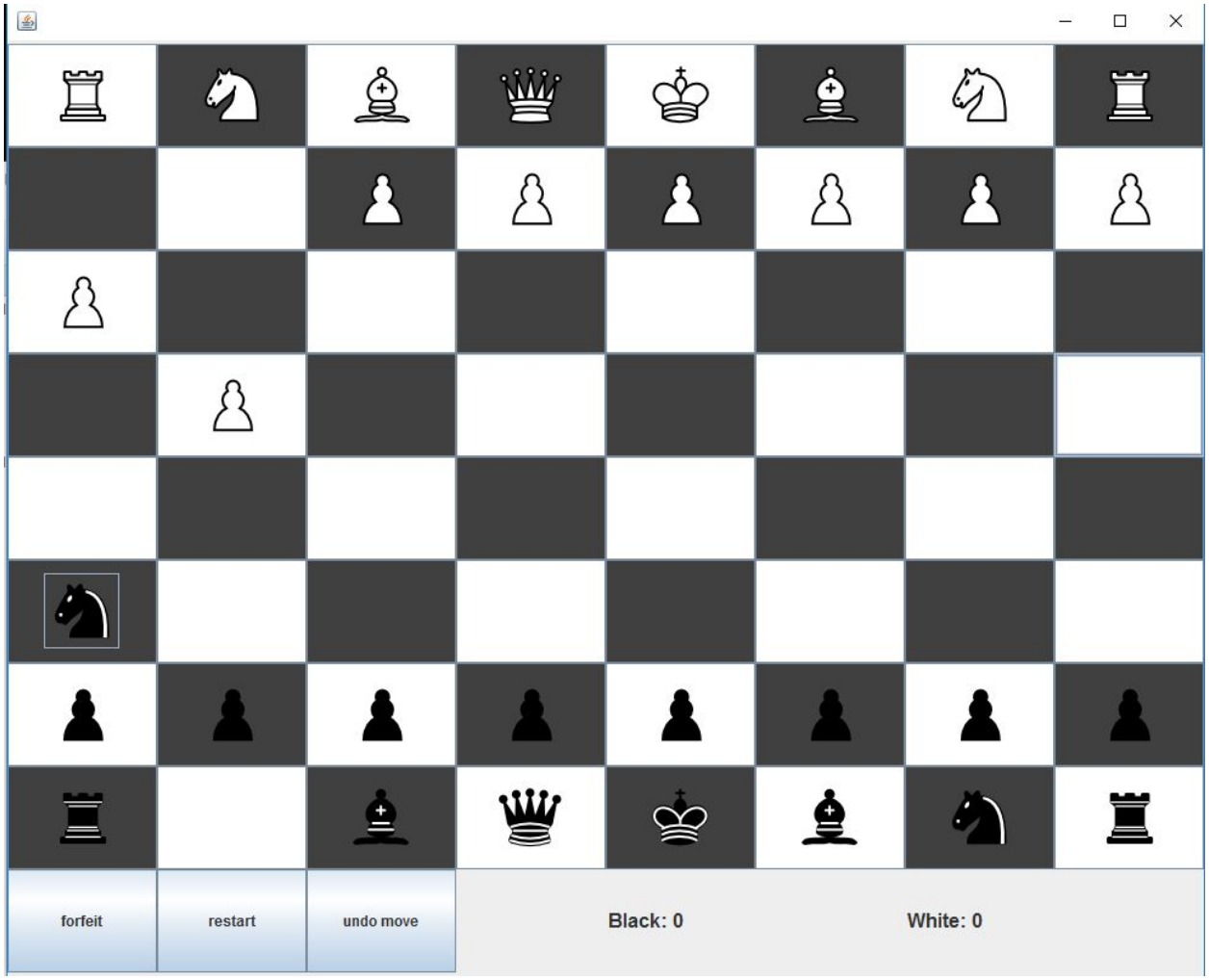
Click out of the error message and now try to move one of black's pieces, say the leftmost knight. This time, let's try to click on a tile that is not highlighted green (is not a valid move). This should bring up the following error message:



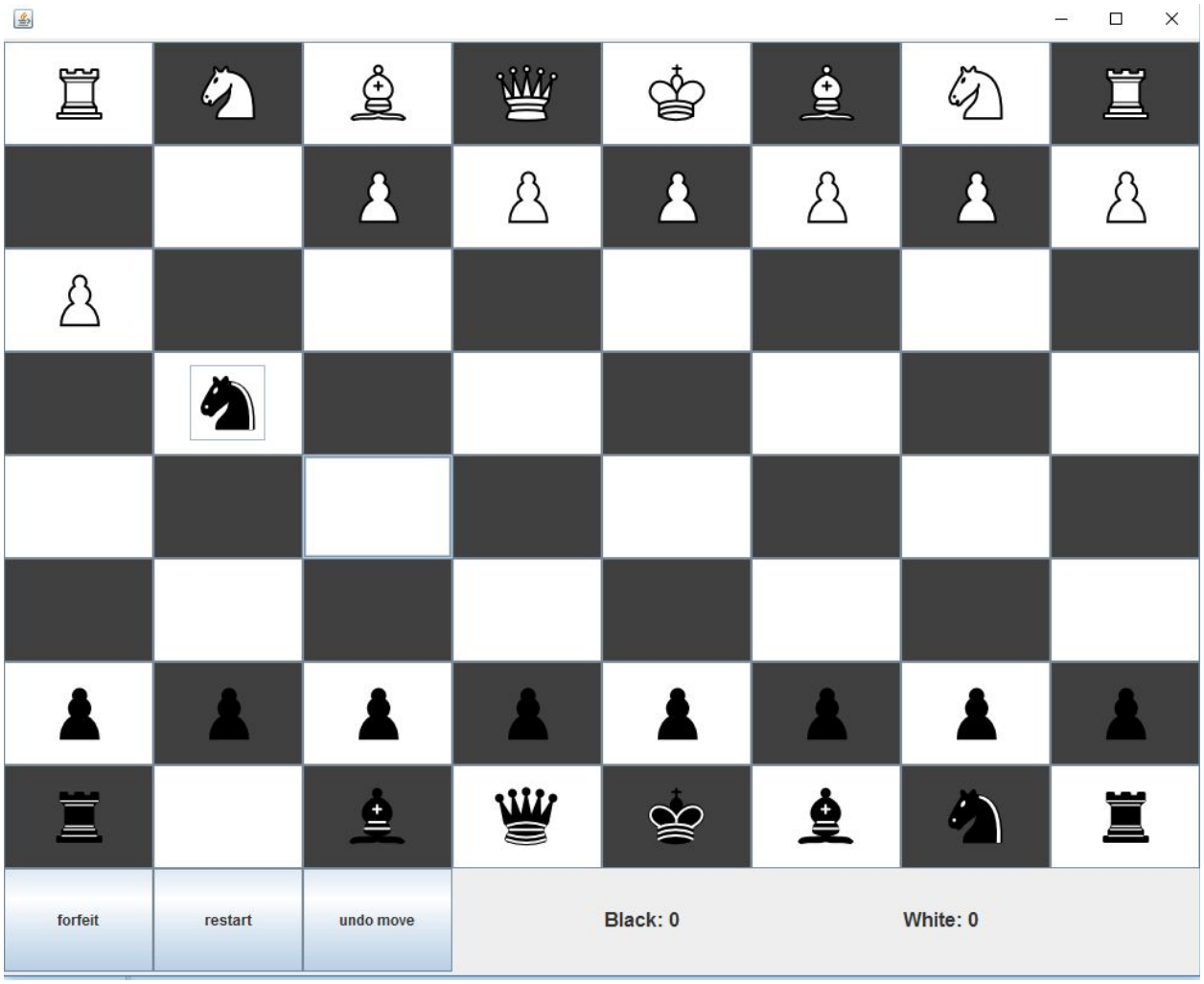
Now try to move the knight again, but this time to its leftmost valid (highlighted green) tile:



Now let’s move the second leftmost white pawn forward two spaces. The board state should then be:

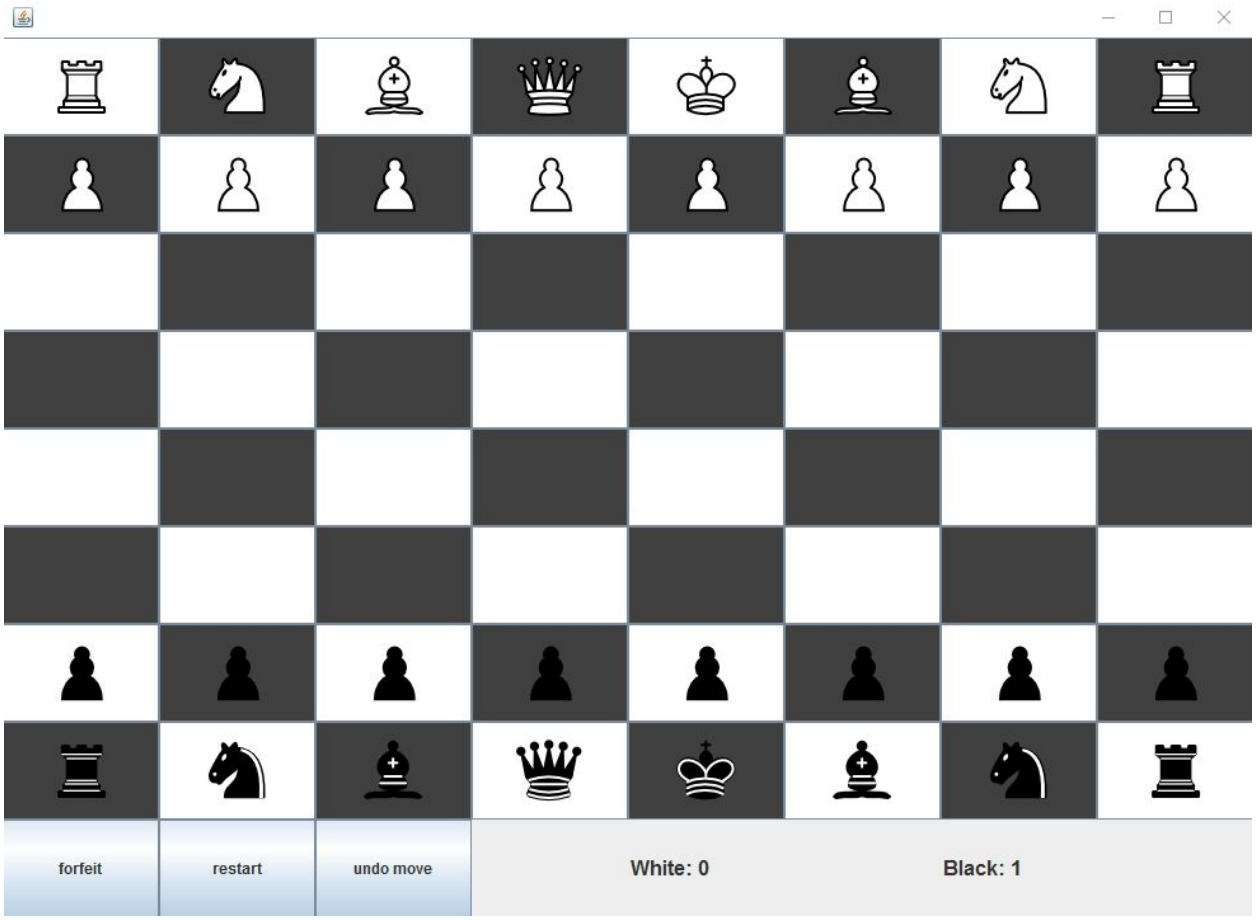


Now let’s capture that white pawn with the black knight we moved the previous turn. This should correctly move the white pawn from the board and replace it with the black knight:



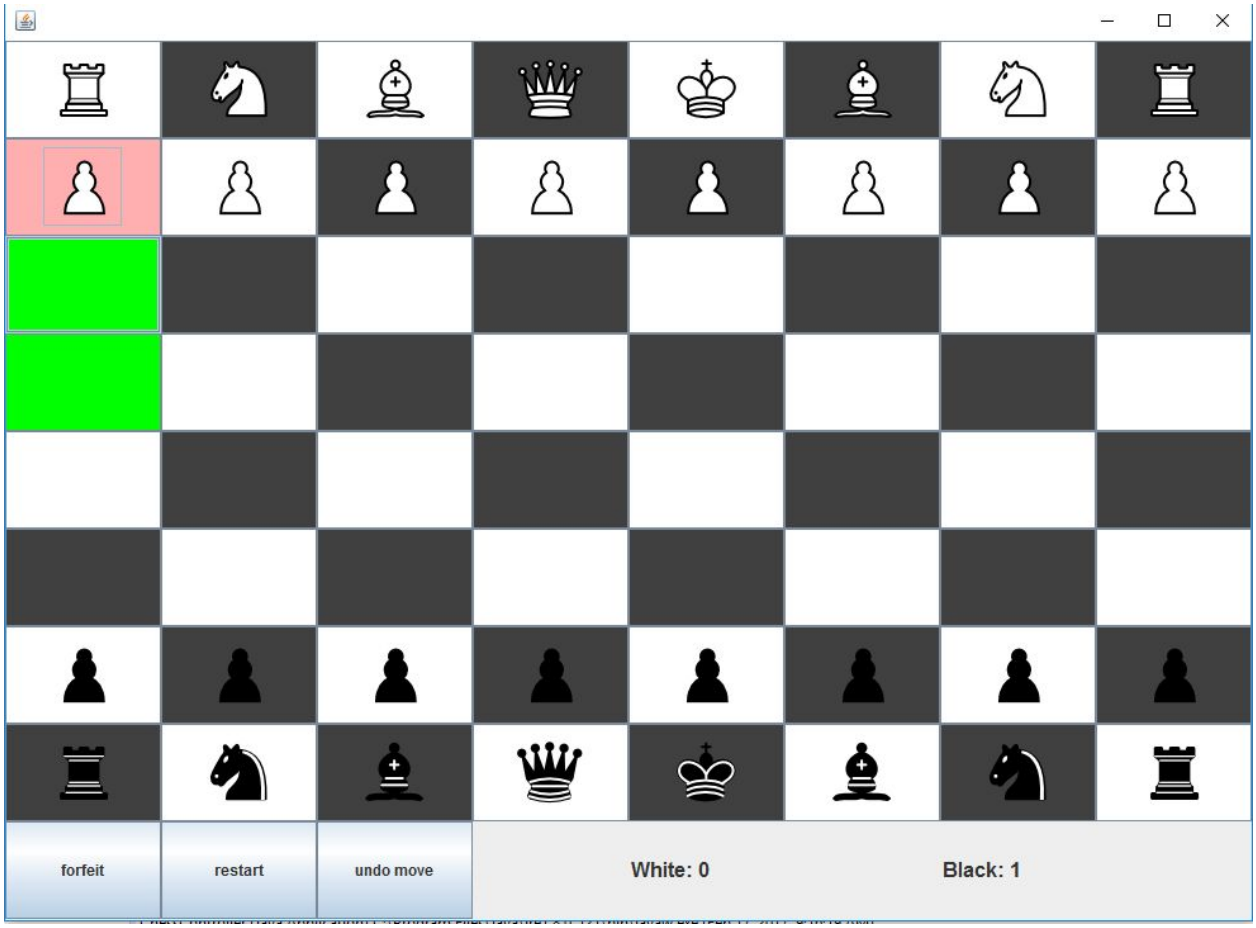
Now move the rightmost white pawn forward, and the black knight we have been moving up two spaces and right one space. The board state should now be:

Notice that the black player has now won the game, because the white king is in check and is unable to move since it is blocked on all sides. After hitting OK on the pop-up we should see:

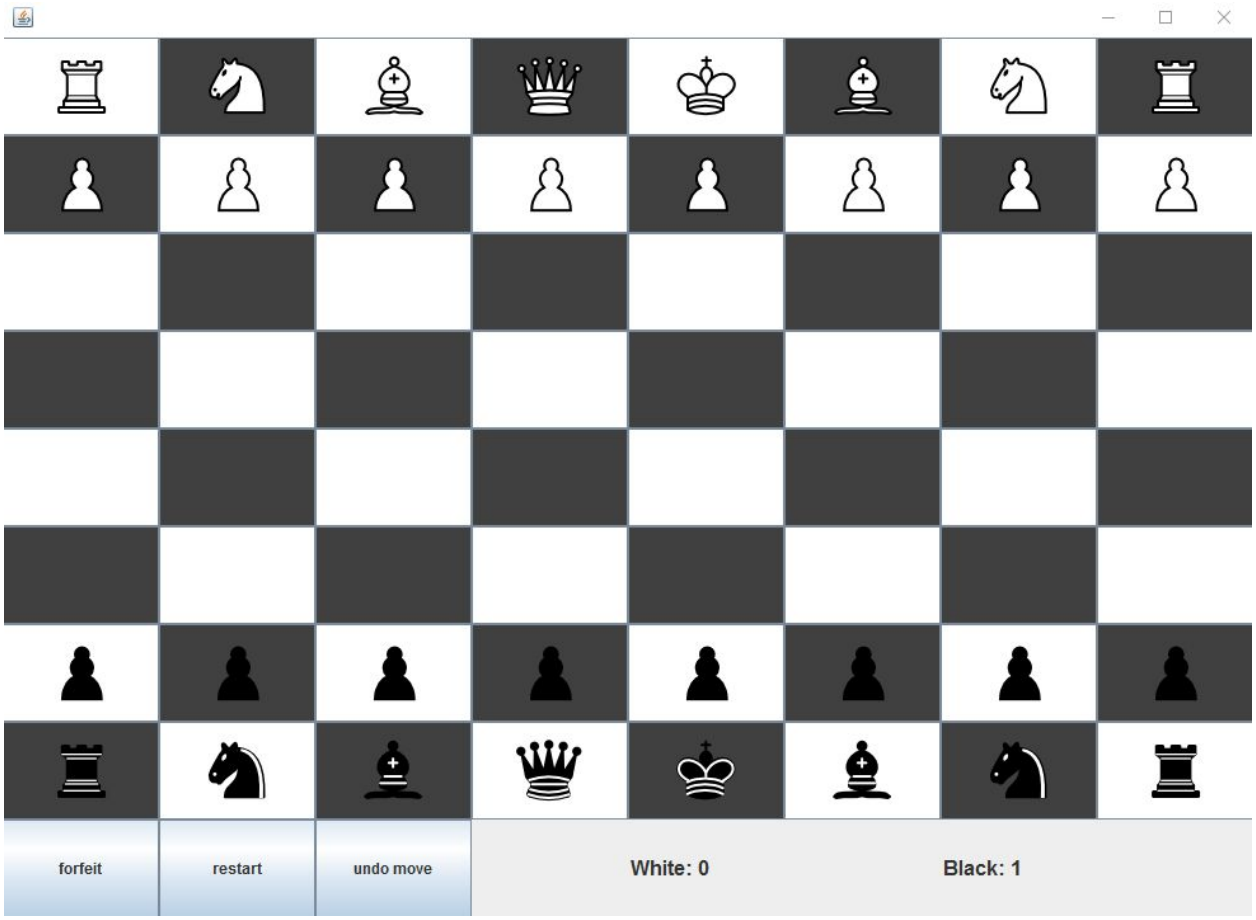


Note that the score in the right corner should reflect that black has won a single game now, and white has still not won any.

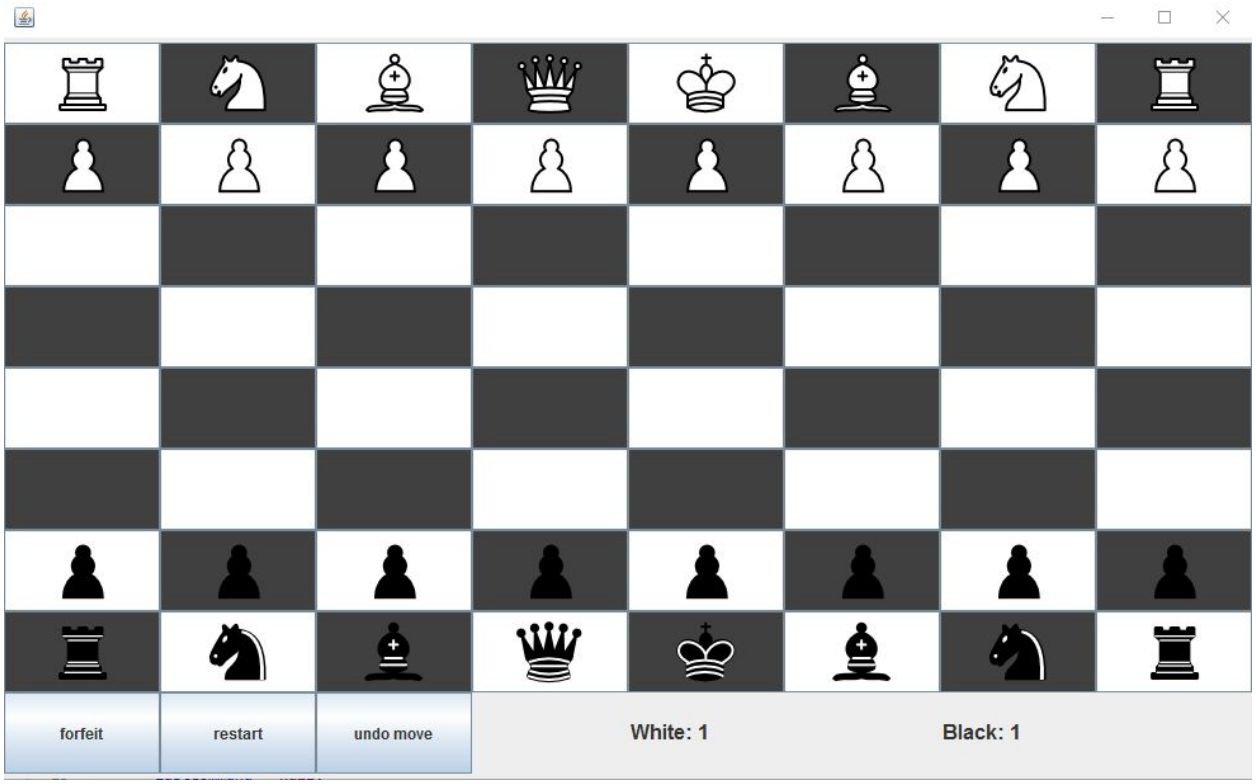
Now let's move the leftmost pawn of the white pieces again. Then, hit the "undo move" button in the bottom left. The board should look the same as it does above again. Now click on that pawn again and make sure it can still move 2 spaces forward (that is, it's "firstMove" flag is still set properly after the undo):



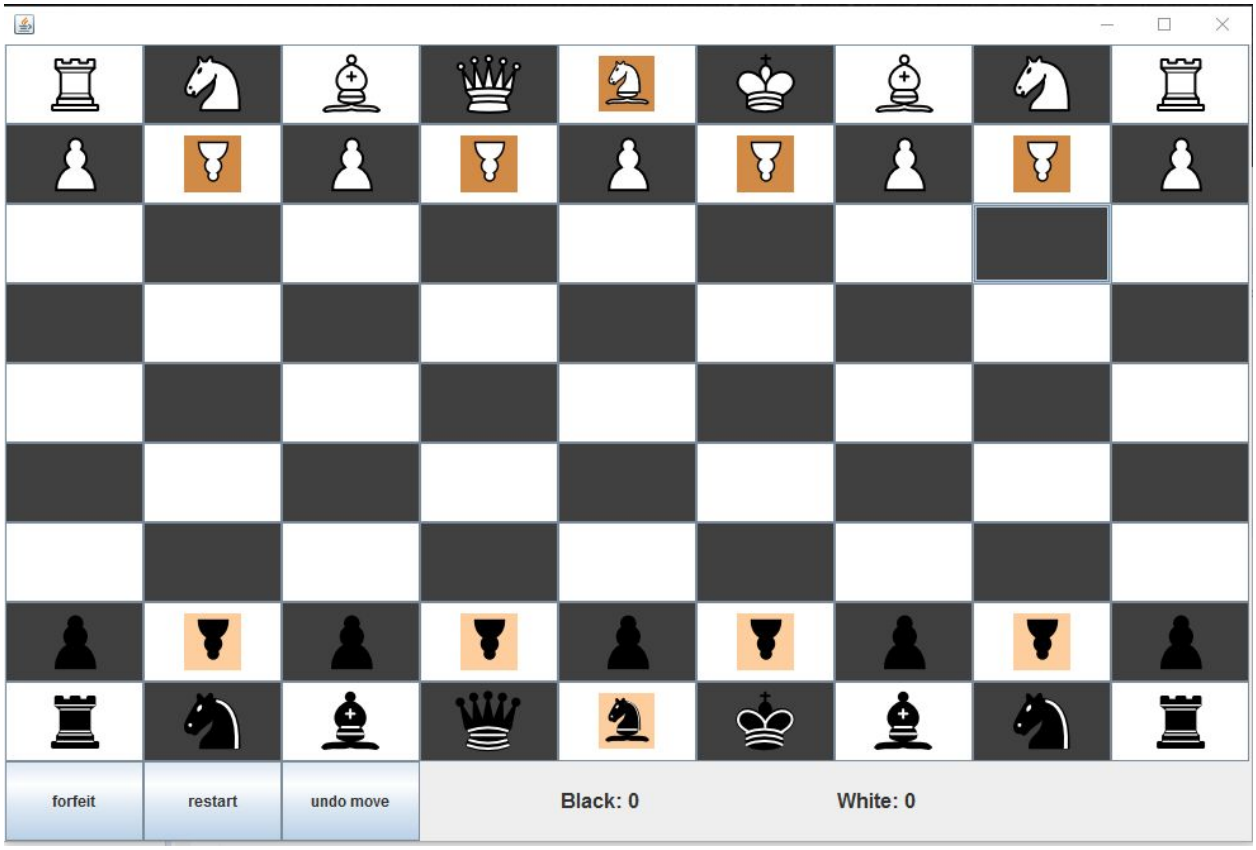
Go ahead and complete a forward move with that pawn. Then hit the restart button in the bottom right. The board should be back in its initial state, with the score still black:1 white:0:



The last piece of functionality of the GUI to test is the forfeit button. Move the white pawn forward as in the last game, then hit the forfeit button. Since black was the player to forfeit, the score should now be tied 1-1 with a fresh game appearing:



It appears that all elements of the GUI are working, but let's lastly look at the custom game. Exit out of the GUI and re-run the java app. This time select custom game in the drop-down. The initial board should look like:



Verify the following properities:

- The board is 9x9
- Pawns and berolina pawns alternate in the first row of each color
- The Princess piece is placed between the king and queen
- The score is 0 to 0.

All the tests run for the custom game should have the same results. Verify this now.

This confirms that the GUI elements are operating properly, please see the testing package for additional tests regarding the functionality of the GUI (e.g listeners are correctly being called, returning the correct results, etc.) and the underlying model of the game.