# Laboon Chess

CS 1530 - SPRINT 5 DELIVERABLE

1 Dec 2016

| | |
|---|---|
| Craig Kodman (*) | [ *c-kodman* ] |
| David Bickford | [ *drb56* ] |
| Joe Meszar | [ *outkst* ] |
| David Tsui | [ *luckyleap* ] |

* SCRUM MASTER

## II. Summary of Sprint

For our final sprint, our group was successful in pushing out many final, critical user stories needed to complete our chess project. As well, we were able to fix many of our previous issues that were present during the previous iteration of the sprint. Some of our key features such as: being able to load and save the chess game, kibitzer chess messages, and GUI clean up were added during this sprint. Some of the issues regarding starting a new game in multiplayer and against the chess AI was also fixed. Overall, our product has satisfied all of the major features the customer requested. However, given more time, we can continue to improve our product with more test coverage and fixing minor defects that have been found.

During our sprint over the past two weeks, our team has been using the same methods of communication as the previous sprints. This communication involves mainly using slack messaging and phone communication for quick questions and meeting arrangements, personal meetings for scrum meeting, and working together physically for faster questions on specific code. Our scrum meeting has shifted to meeting immediately after class since that is when everyone is free. This meeting style, with slack, scrum meeting, and working together, has proven to be an effective method of communication as it is flexible and time efficient.

Some small disagreements arose during this sprint regarding the focus of the sprint. As this sprint is the final sprint, our resource time is severely constrained. We must be able to balance out which features to focus on and which features to drop. Some of our decisions in conflict involved the format in which we shall save our game and whether we should fix the bug in chess for en passant. In the end, we decided not to save the chess game in the standard Portable Game Notation (PGN) format and to push out the chess game with the bug where en passant does not work. These decisions were done in order to maximize engineering time on tests for loading and saving game, and getting the features for a new game so that the game is more playable.

We encountered some challenges during this sprint, which in turn, lowered our velocity. One challenge involve rendering the chess graphics when the game loads a new chess game. It turned out to be much more involved than initially anticipated as we did not have functions which dealt with translating Forsyth–Edwards Notation (FEN) string to standard algebraic notation (SAN), but we did have SAN to FEN string.

Another challenge occurred with using our FEN string after building tests for it. While attempting to get a list of valid chess moves, the FEN string was passed in to determine what moves could be made given the current state of the board. At first, our list of valid moves was incorrect and causing an *ArrayIndexOutOfBounds* exception. After stepping through the progression of events for that method, it came to light that our FEN string was actually incorrect with respect to the handling of the “Castling” events. This part of the string holds information determining if either team can perform a special Castling move where the King and the Rook are moved mid-way, in unison, to opposite sides of each other. The string was not being updated properly, and our tests were not checking this part of the FEN string; resulting in errors that were seemingly misguided.

Furthermore, we encountered a challenge including special chess events like *CASTLING*, *PAWN PROMOTION*, *EN PASSANT*, and *CHECKMATE*. Each required a specific set of events to occur; for example "Castling" requires that a team's king has never moved before, one or both of it's rooks has never moved before, there exists no chess pieces between the king and rook that will be castled, and (finally) the king does not castle *from*, *through*, **or** *into* Check. These requirements make it very difficult to program and to keep-track-of during the course of the chess game. In fact, "castling" and "en passant" both require constant observation of the board so that the events required for them to occur can properly be determined. Even if the pieces are in the correct location for such special chess moves, without constant observation it is not a direct indicator of being able to perform those moves.'

Yet another challenge we faced was how best to test the code. For most of the software development, it was difficult to test the GUI directly. Instead, we had to manually test the actual playing of the chess game. This was very time consuming. Oddly, one of the errors showed up only when there was only 2 legal moves left for one side. So it literally took a whole game played to find the error. And since the error was hard to decipher, multiple full games had to be played to determine how best to fix it. This error was caused by our logic in part of the FEN string generation (castling) that led to the program to incorrectly conclude that there were no legal moves remaining. The manual testing, while time consuming, was instrumental in finding multiple errors that were later fixed.

Some of our code was written by some members of the team that was later deleted and rewritten by other members of the team. This was never a problem for our team, as no one was sentimentally attached to their code and just wanted the product to work. But there had to be discussion on whether rewriting of the code would be beneficial to the project.

## III. User Stories

As a chess player,
I want an automated kibitzer,
So that I can pretend I am in a noisy environment

As a chess player,
I want to click “New Game”,
So that I can restart and play a new game of chess any time

As a chess player,
I want to save a game to a destination of my choice,
So that I remember where I saved the game

As a chess player,
I want to load a saved game,
So that I can return to a previous chess state I had saved

As a chess player,
I want to undo a move,
So that I can take back a move I had accidentally made

## IV. Link to Github

https://github.com/drb56/CS1530

## V. Defects Found

**Defect 1.** Special chess piece logic for does not work.

This defect was not fixed as the challenge of creating a working EN PASSANT move involved updating the functionality for the fen string and updating the graphics for the EN PASSANT. This update would require a lot of engineering time which could be spent on fixing other more critical defects that would prevent the player from playing the game.

*Reproduction Steps:*

1. Move pawn D2 to D4
2. Move pawn H7 to H6
3. Move pawn D4 to D5
4. Move pawn E7 to E5
5. Move pawn D5 to E6

*Expected Behavior:*

1. Pawn moves from D5 to E6

*Observed Behavior:*

1. There is no possible move for D5 to E6.

**Defect 2.** Once a game was finished, a new game could not be started. After "New Game" was chosen, no moves at the start of the new game were found to be legal. **This was eventually fixed by properly resetting the necessary game variables.**

*Reproduction Steps:*

1. Click on "File"
2. Click on "New Game"
3. Click on "Computer Black"
4. Play game until it completes
5. Click on "File"
6. Click on "New Game"
7. Click on "Computer Black"

*Expected Behavior:*

1. A new game should be started where the player can make any legal move white pieces

*Observed Behavior:*

1. When attempting to make a first legal move with white, none will be made as every single move will be found to be illegal.

**Defect 3.** Kibitzer does not stop when the game has ended.
**This defect was fixed by introducing an on/off variable that turns the kibitzer off when the game has ended; and on when a game has been *started*, *loaded*, or the *Undo* button was clicked.**

*Reproduction Steps:*

1. Play the game until Checkmate.

*Expected Behavior:*

1. The Kibitzer messages should stop when the game has ended (Checkmate).

*Observed Behavior:*

1. The Kibitzer was still generating random messages and showing them in the GUI.

**Defect 4.** Game registers illegal move when attempting a legal move.
**This error was eventually fixed by correcting logic in how our castling portion of the FEN string was generated.**

*Reproduction Steps:*

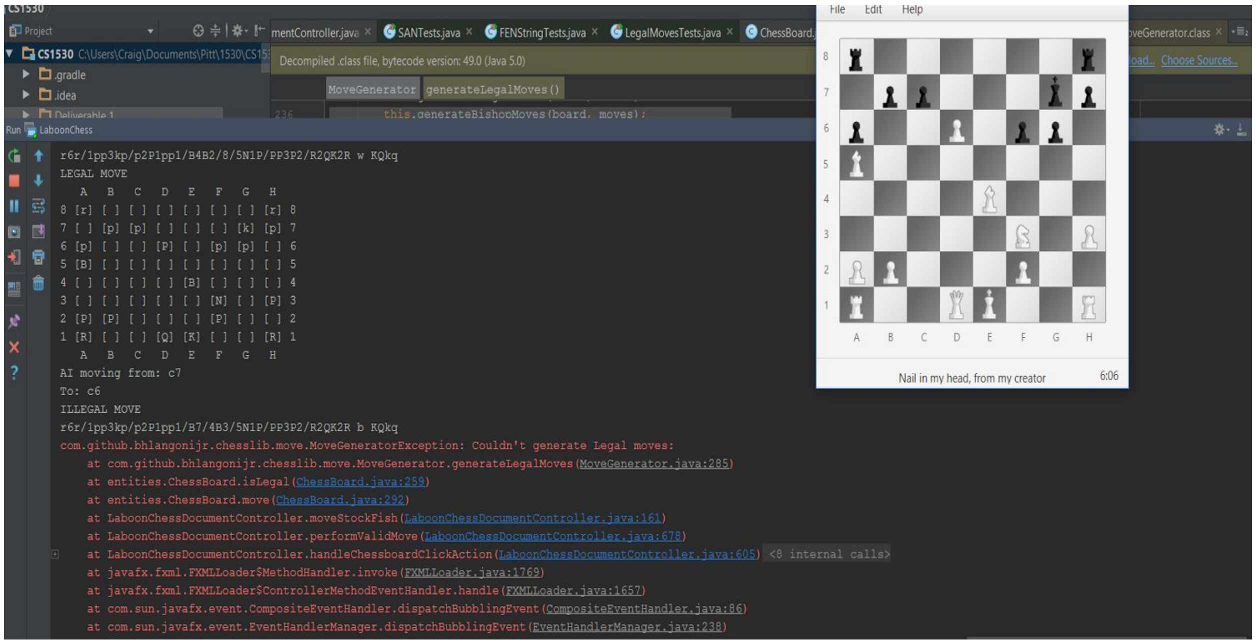
1. Play the game for some unspecified time as this was a random occurrence

*Expected Behavior:*

1. The game should continue as long as there is a legal move that can be executed

*Observed Behavior:*

1. At random times, when attempting a legal move, the game would crash with a *MoveGeneratorException* as seen:



```
this.generateBishopMoves(board, moves);
```

```
r6r/1pp3kp/p2P1pp1/B4B2/8/5N1P/PP3P2/R2QK2R w KQkq
LEGAL MOVE
   A   B   C   D   E   F   G   H
8 [r] [ ] [ ] [ ] [ ] [ ] [ ] [r] 8
7 [ ] [p] [p] [ ] [ ] [ ] [k] [p] 7
6 [p] [ ] [ ] [P] [ ] [p] [p] [ ] 6
5 [B] [ ] [ ] [ ] [ ] [ ] [ ] [ ] 5
4 [ ] [ ] [ ] [ ] [B] [ ] [ ] [ ] 4
3 [ ] [ ] [ ] [ ] [ ] [N] [ ] [P] 3
2 [P] [P] [ ] [ ] [ ] [P] [ ] [ ] 2
1 [R] [ ] [ ] [Q] [K] [ ] [ ] [R] 1
   A   B   C   D   E   F   G   H
AI moving from: c7
To: c6
ILLEGAL MOVE
r6r/1pp3kp/p2P1pp1/B7/4B3/5N1P/PP3P2/R2QK2R b KQkq
com.github.bhlangonijr.chesslib.move.MoveGeneratorException: Couldn't generate Legal moves:
    at com.github.bhlangonijr.chesslib.move.MoveGenerator.generateLegalMoves(MoveGenerator.java:285)
    at entities.ChessBoard.isLegal(ChessBoard.java:259)
    at entities.ChessBoard.move(ChessBoard.java:292)
    at LaboonChessDocumentController.moveStockFish(LaboonChessDocumentController.java:161)
    at LaboonChessDocumentController.performValidMove(LaboonChessDocumentController.java:678)
    at LaboonChessDocumentController.handleChessboardClickAction(LaboonChessDocumentController.java:605) <8 internal calls>
    at javafx.fxml.FXMLLoader$MethodHandler.invoke(FXMLLoader.java:1769)
    at javafx.fxml.FXMLLoader$ControllerMethodEventHandler.handle(FXMLLoader.java:1657)
    at com.sun.javafx.event.CompositeEventHandler.dispatchBubblingEvent(CompositeEventHandler.java:86)
    at com.sun.javafx.event.EventHandlerManager.dispatchBubblingEvent(EventHandlerManager.java:238)
```