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

**SCHOOL OF ENGINEERING AND TECHNOLOGY**

**FINAL ASSESSMENT FOR THE BSC (HONS) INFORMATION TECHNOLOGY; BSC (HONS) COMPUTER SCIENCE; BACHELOR of SOFTWARE ENGINEERING (HONS)YEAR 2**

**ACADEMIC SESSION 2024; SEMESTER 3**

**PRG2104: OBJECT ORIENTED PROGRAMMING**

**Project DEADLINE: Week 14**

**INSTRUCTIONS TO CANDIDATES**

# This assignment will contribute 50% to your final grade.

* This is an individual assignment.

**IMPORTANT**

# The University requires students to adhere to submission deadlines for any form of assessment. Penalties are applied in relation to unauthorized late submission of work.

# Coursework submitted after the deadline will be awarded 0 marks

**Lecturer’s Remark** (Use additional sheet if required)

I Zar Nie Aung (Name) 21074380 std. ID received the assignment and read the comments

****

15 Jan 2025

(Signature/date)

**Academic Honesty Acknowledgement**

“I Zar Nie Aung student name). verify that this paper contains entirely my own work. I have not consulted with any outside person or materials other than what was specified (an interviewee, for example) in the assignment or the syllabus requirements. Further, I have not copied or inadvertently copied ideas, sentences, or paragraphs from another student. I realize the penalties *(refer to page 16, 5.5, Appendix 2, page 44 of the student handbook diploma and undergraduate programme)* for any kind of copying or collaboration on any assignment.”

****

15 Jan 2025 ….................................. (Student’s signature / Date)

**Table of Content**

[**0.0 Project Source Location 3**](#_ipk318acxvo3)

[**1.0 Introduction 3**](#_h345qrw4kzb5)

[**2.0 Design and Development 6**](#_vuffz4ekq5zc)

[UML Class Diagram 6](#_7emf4z4xlyhv)

[UI Designs 12](#_6847l4sit50g)

[**3.0 Test Documentation 16**](#_ql5vot1qixgg)

[3.1 Entering Names for Game Activity 16](#_6e6rnja1sp1k)

[3.1.1 White Player and Black Player Filled 16](#_tnouc8zbqhgu)

[3.1.2 White Player Filled 16](#_6hvai5i4zris)

[3.1.3 Black Player Filled 18](#_a9n3fzg8w07y)

[3.1.4 Neither Players Filled 19](#_7z1f5ki1kw0j)

[3.2 Piece Event 20](#_ul7kfeakz0v)

[3.2.1 Pawn 20](#_91bs85jd8w7y)

[3.2.1.1 Pawn On Click Start 20](#_w7ovf0mzud62)

[3.3.1.2 Pawn On Move 20](#_6bft52o6j80w)

[3.2.1.3 Pawn On Click After First Move 21](#_p7nx8a4equn6)

[3.2.1.4 Pawn On Click Takeable 21](#_faaurpy8cbx0)

[3.2.1.5 Pawn On Take 21](#_ewj3ay2nnshr)

[3.2.2 Bishop 22](#_os04vyujxr6e)

[3.2.2.1 Bishop On Click 22](#_mx63w1ipjnbe)

[3.2.2.2 Bishop On Move and Takeable 22](#_2zgz7cuu0h76)

[3.2.2.3 Bishop On Take 23](#_611210ndzfi8)

[3.2.3 Knight 23](#_3ssf53h2991s)

[3.2.3.1 Knight On Click 23](#_skpygrmq0kci)

[3.2.3.2 Knight On Click, On Move, On Takeable 24](#_vuxcn7b1q2cl)

[3.2.3.3. Knight On Take 24](#_73dzxoxi962h)

[3.2.4 Rook 25](#_etmlemfb05jb)

[3.2.4.1 Rook On Click 25](#_y1qvvoj7apkm)

[3.2.4.2 Rook On Move, On Click, On Takeable 25](#_9eupkubr3dp6)

[3.2.4.3 Rook On Take, MoveTable On Scroll 26](#_to9f6kjo6z3h)

[3.2.5 Queen 26](#_h9by9medfde6)

[3.2.5.1 Queen On Click 26](#_4ukgam8p4nlz)

[3.2.5.2 Queen On Move 27](#_kr5jbvfl4itr)

[3.2.5.3 Queen On Move, On Click, On Takeable 27](#_k3x4ud22pvuv)

[3.2.5.4 Queen On Take 28](#_wc7j4m4i3d75)

[3.2.6 King 28](#_p6sqzb1cegpd)

[3.2.6.1 King On Click 28](#_mg8ykz4lfq2b)

[3.2.6.2 King On Move 29](#_w75a7qa5ow5k)

[3.2.6.3 King On Move, On Click, On Takeable 29](#_yso09z8mfr1l)

[3.2.6.4 King On Take 30](#_k3xt05jj96vq)

[3.3.Game-Over Event 30](#_bgqvga98104g)

[3.3.4.1 Black Won 30](#_wbvy7zedpmo3)

[3.3.4.2 White Won 31](#_cud9hnoj8gt)

[3.3.4.3 On Draw Event (Passive Play) 31](#_r49813nixp85)

[**Self-Reflection 32**](#_1ltqf699c7g7)

# 

# 0.0 Project Source Location

*To locate the github repository of the documented project, please use this link:*

[*https://github.com/sunwaydcis/final-project-TacticalObsidian*](https://github.com/sunwaydcis/final-project-TacticalObsidian)

*Zar Nie Aung (21074380)’s Github:*

<https://github.com/TacticalObsidian>

# 1.0 Introduction

Long Live the King (LLtK) is an adaptation to the standard chess game where the end goal of the is to take the opponent’s king piece. LLtK is played between 2 players on a more minimal chess engine. Players are not stopped during check conditions and are not prevented from making moves that might lead to one, especially if the move benefits their position. This leads to a more unique feel of chess, where players are to solely rely on their knowledge of chess to keep their king on the board, whilst also adapting to new strategies in order to knock their opponent’s king off the board unnoticed. With this new end goal, a game of LLtK is then played with the following set of rules:

* **Players are free to make any standard chess move except**
  + Pawn promotions - The premise of the game is to ‘kill’ the other king, making pawn promotions overpowered and highly valued early game. This also makes all existing pieces more valuable as they shall not be replaced later in game, forcing players to make due with what they currently have to win the game.
  + Castling - The game is built upon a limited game engine with the aim of forcing players into a more aggressive play style. Any defensive move like castling is not supported.
  + En passant - Similar to the above reason, the game engine was intentionally under developed, preventing special moves like en passant to be played. This way not only are players forced to be more aggressive, they are also forced to learn a new style of chess.
  + When making a check move, the player are advised to *not* inform their opponent - The premise of the game is to take the other king. Informing of the check, though not against the rules, would certainly undermine the player’s efforts at winning the game.
* **Players are to make an aggressive at least once in every 5 rounds**
  + The game offers minimal support to protect players from check conditions, forcing many to scan the board regularly. The level of pressure may lead to players intentionally making more defensive moves rather than offensive ones. To prevent this, all players must take one enemy piece at least once every 5 rounds, or else a draw will occur. Moreover, this can also be exploited by a player with lesser material, forcing the player with superior material to find a way to get the enemy king or another enemy piece as soon as possible. This approach does not stray far from standard chess, as when a player’s conditions are unfavorable, they can still avoid losing by forcing a draw condition.
* **The King must fall**
  + Through any means necessary, the player must be able to take the opponent player’s king.

In consequence to the above rules, LLtK is built surrounding the following functional requirements.

|  |  |
| --- | --- |
| User shall be able to move a piece | For all pieces the player can move, the game should display all the possible legal moves said piece can make.  The user shall be then able to select a tile on the board for the said piece to move to. |
| User shall be able to take a piece | If an enemy piece exists in range of the selected piece’s moves, the user shall be able to take the piece. |
| User shall be able to quit the game at any point | A game of chess can be relatively long and uninteresting. Because of this, the game view shall contain a quit button where users can leave the game easily. |
| User shall be able to keep track of the game progression | As per standard chess games, such as Chess.com, most players are able to see all the moves that have occurred throughout the game in order to keep track of game events and learn their opponent’s strategy.   To meet this requirement, the game view shall display these moves. |
| User shall be able to win, lose | As per the rules of the game, the user shall be able to win or lose depending on if they took the opponent’s king piece, or if their king piece was taken.  Moreover, if both players played passively for 5 rounds in a row, both players will end up in a draw. |
| User shall be able to display their name during the game play. | If a user is willing to put their name under a respective team, their name shall be displayed on the respective card of the game view. |

# 

# 2.0 Design and Development

LLtK is designed and developed using Scala 3 as the codebase following the **M**odel-**V**iew-**C**ontroller software model.. The User-Interfaces (Views) were designed using JavaFX components, and were controlled using controller classes built with the ScalaFX library.

## 2.1 UML Class Diagram

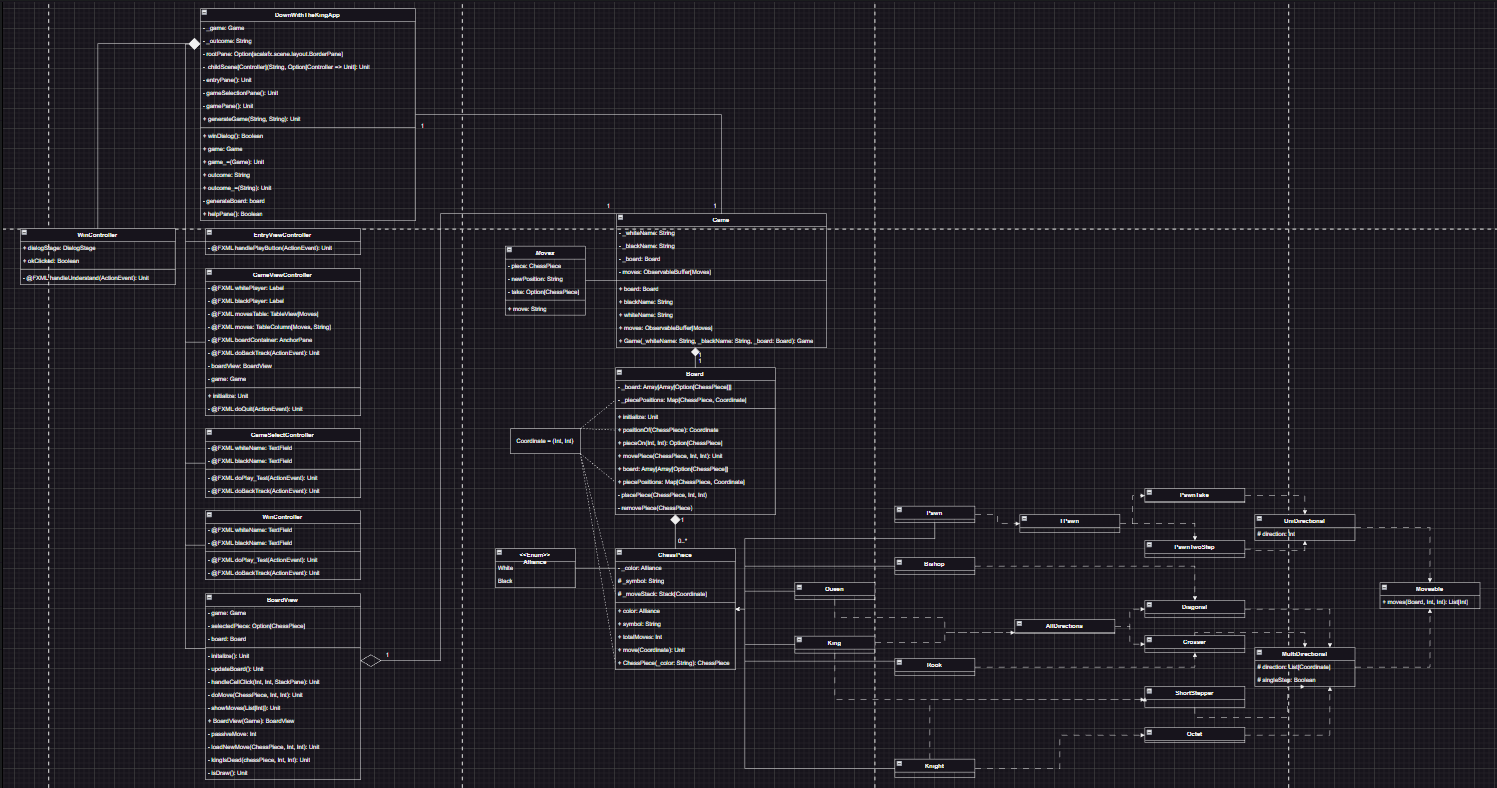


Figure 1 - Full UML Class Diagram of LLtK

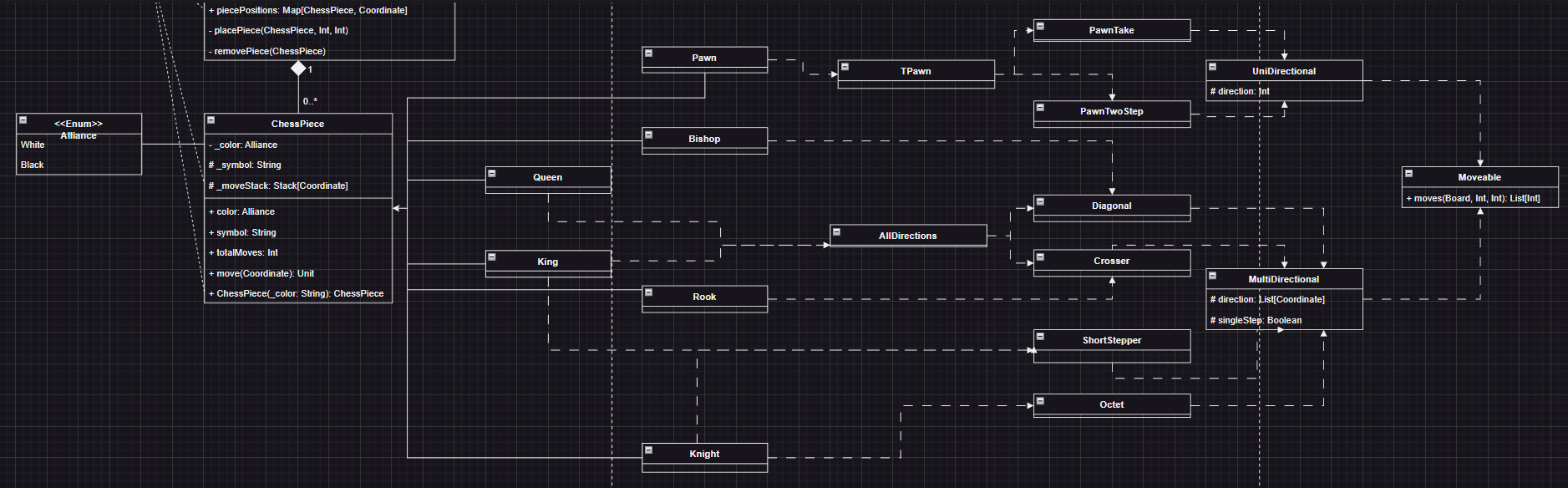


Figure 2 - ChessPieces and Traits

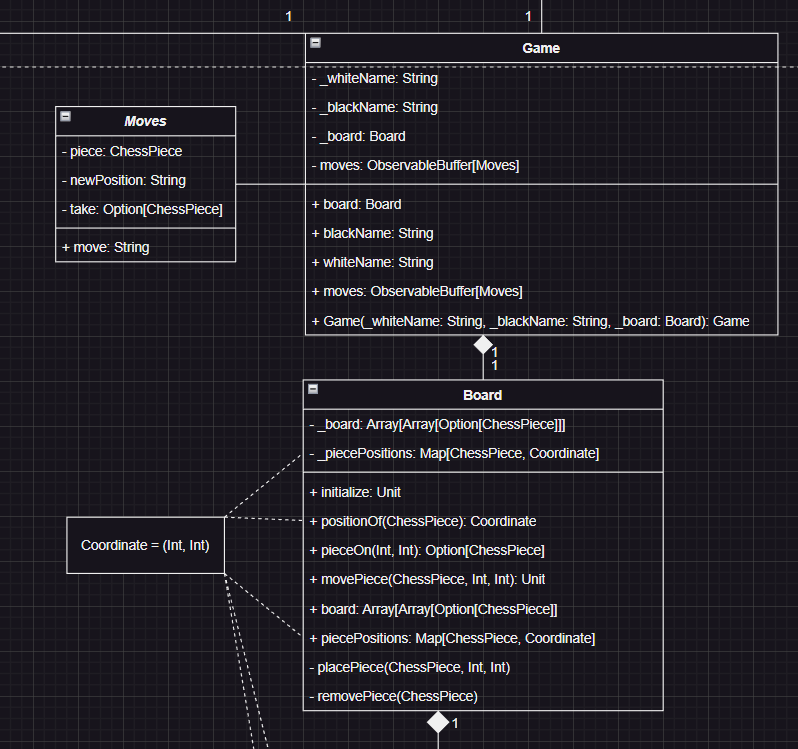


Figure 3 - Board, Game, and *Moves* (case class) classes.

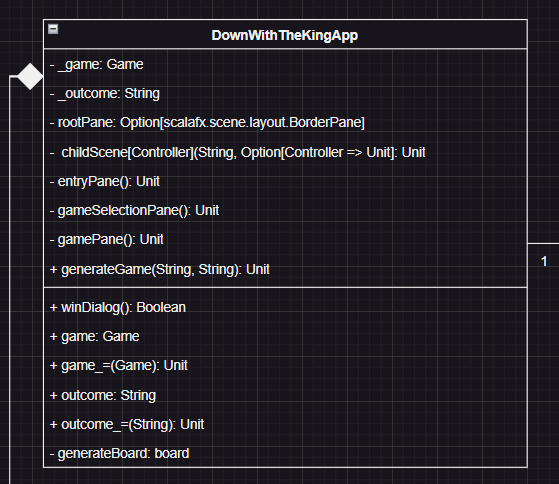


Figure 4 - Main App Object LongLiveTheKing (LLtK)

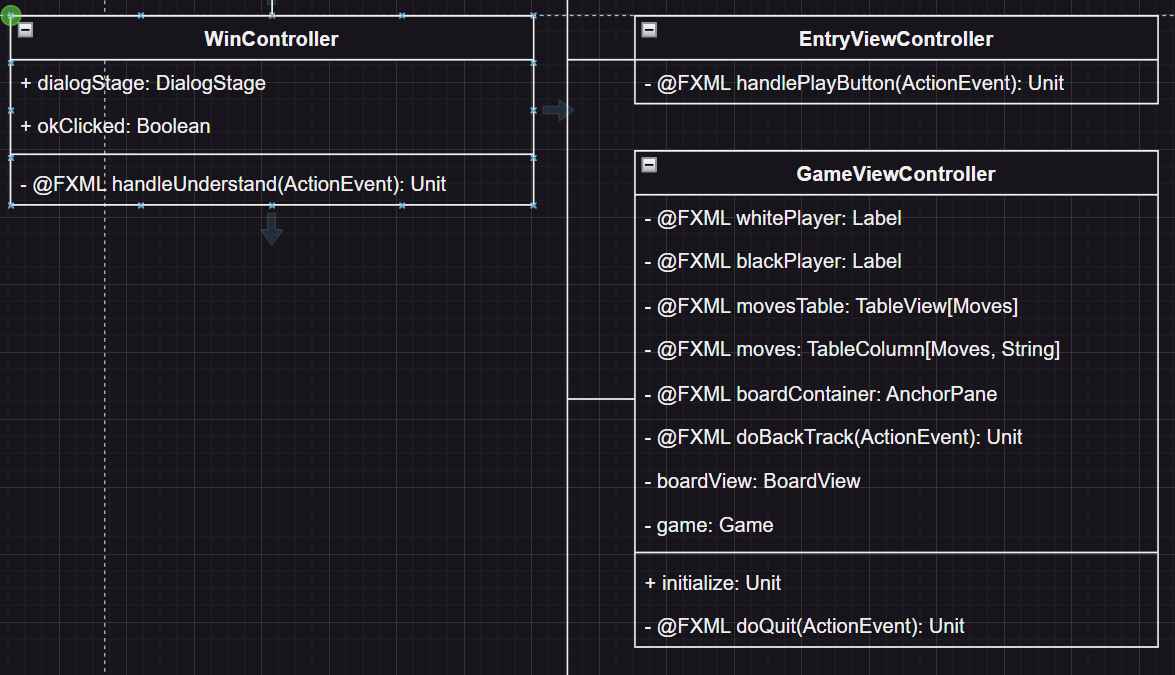


Figure 5A - Children Scenes of LLtK

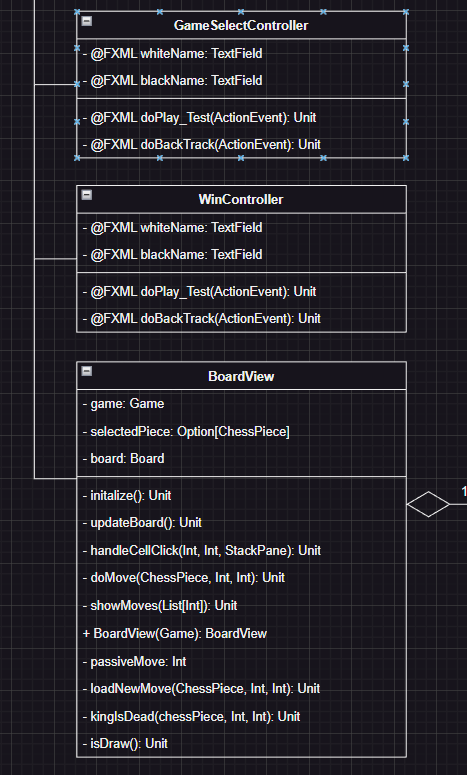


Figure 5B - Children Scenes of LLtK

## 

## 2.2 UI Designs

Provided below are the UI snippets that ChessTutor features. These UIs are built using SceneBuilder, a powerful UI building tool for JavaFX FXML GUIs.

RootLayout.fxml is the master view of the application. It’s responsible for containing common functions that the entire application will depend upon, and also contains key-bind methods to all game related functions, such as create a new game, quit game, and so on, as well as include assistance information. The root node is **BorderPane**, and in it’s center other main scenes will be loaded into it.

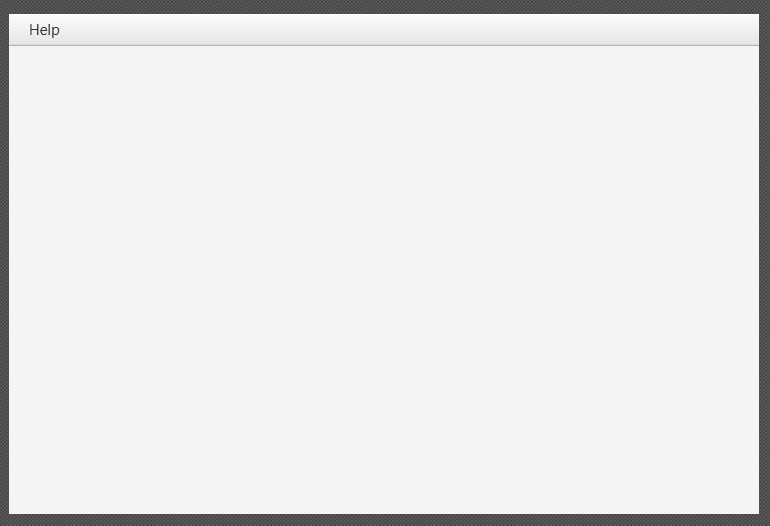


Figure 6 - RootLayout.fxml

EntryView.fxml is the main entry point view of the code. It is responsible for providing user’s a user-friendly navigation to two core components of the application, GameHistory.fxml (Fig 4) and GameSelect.fxml (Fig 5). This scene was built following standard HCI protocols to offer a user-friendly experience when navigating across different scenes.The root node is **AnchorPane**.

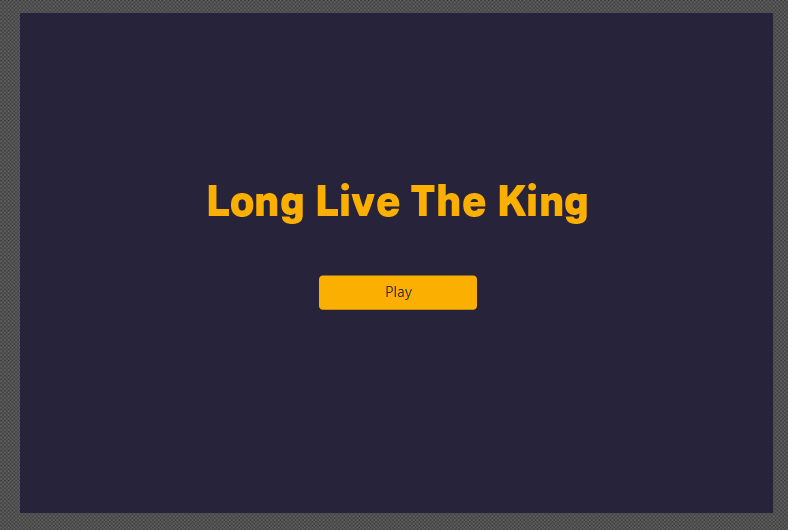


Figure 7 - EntryView.fxml

GameSelect.fxml is responsible for building the new game instance object that will be used by the GameView.fxml (Fig 6). On the top of the scene exists a back button bar to EntryView.fxml (Fig 1). In the center of the screen exists a set of game options for the user to select, including AI difficulties, user game team, and the max time per player. The root node is **AnchorPane**.

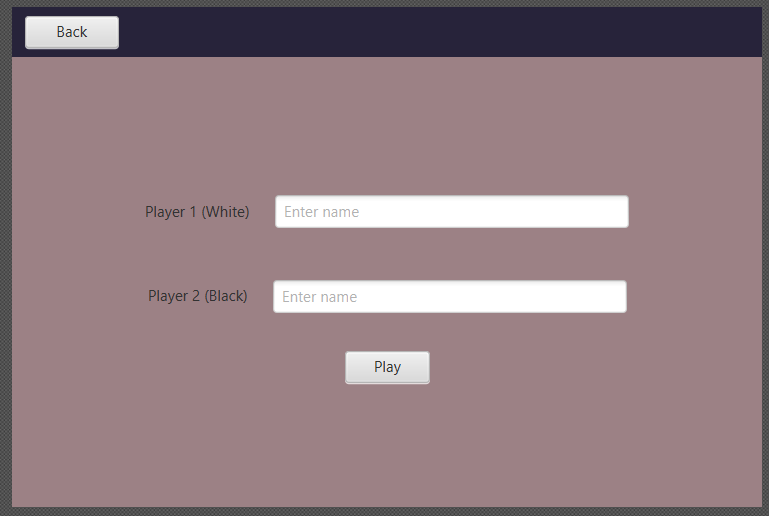


Figure 5 - GameSelect.fxml

GameView.fxml is responsible for loading game instance object information, and controlling the game logic. It is built using nested **SplitPanes**. On the left split pane, exists a horizontal **SplitPane**. On the top is a **TableView** containing the list of moves that has occurred throughout the game. On the bottom is a **GridPane** containing all game control buttons. On right split pane is a **BorderPane**. The top border pane features the black player’s information. the bottom border pane displays the white player’s information. The center border pane contains will be loaded with Board.fxml (Fig 3). The root node is **AnchorPane**.

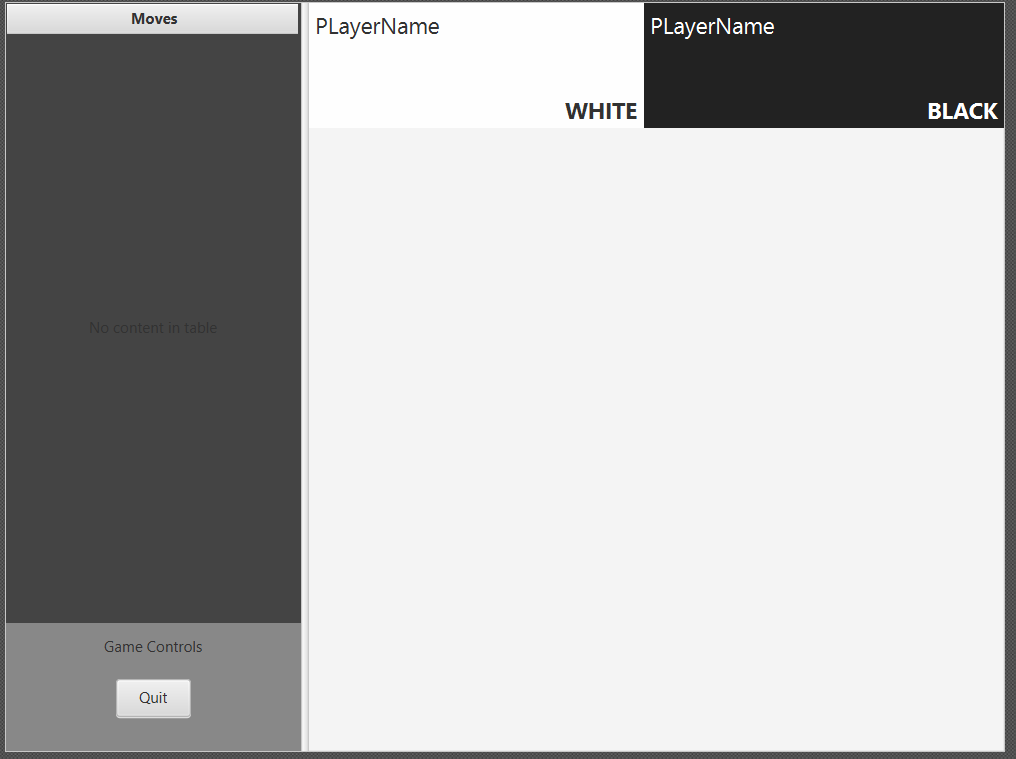


Figure 5 - GameView..fxml

WinView.fxml is responsible for displaying the game outcome message. It is built with nested VBox and HBox elements as they offer more flexible orientation.

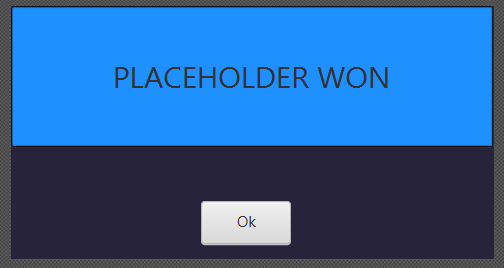


Figure 6 - WinView.fxml

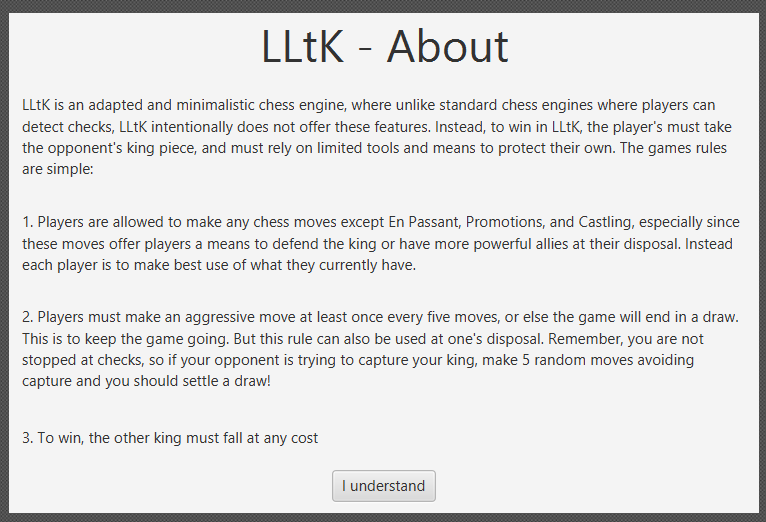


Figure 7 - RulesView.fxml

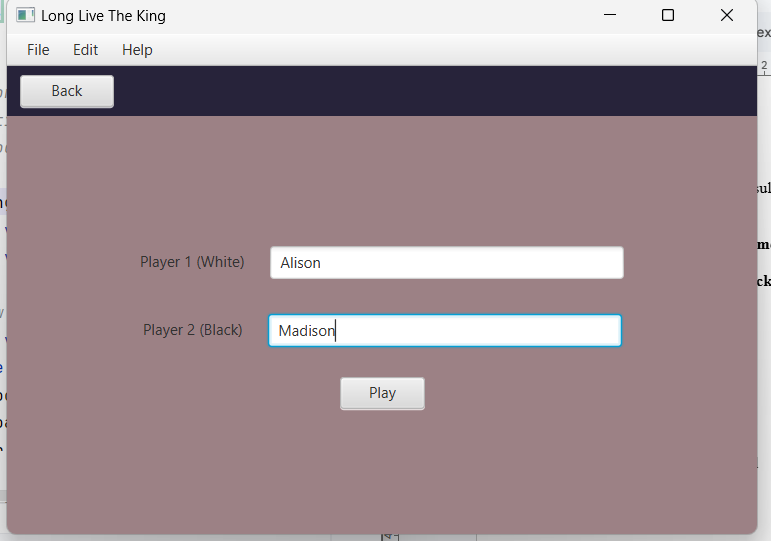
# 

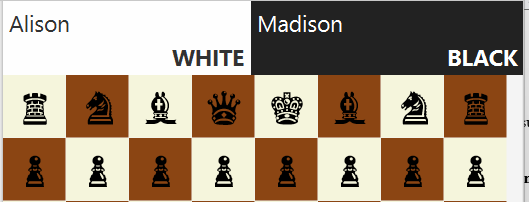
# 3.0 Test Documentation

In this section, all the test results of the proposed functional requirements of LLtK will be demonstrated here.

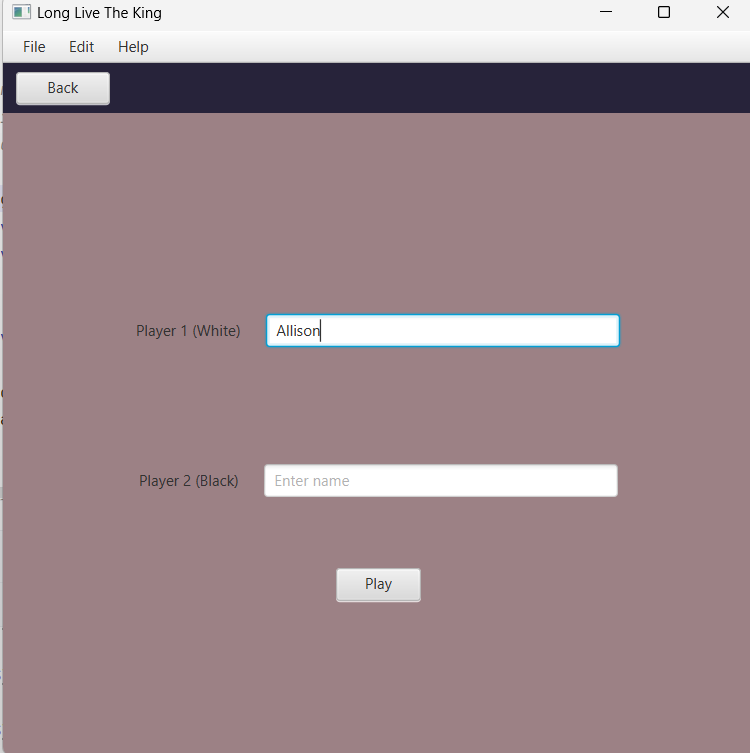
## 3.1 Entering Names for Game Activity

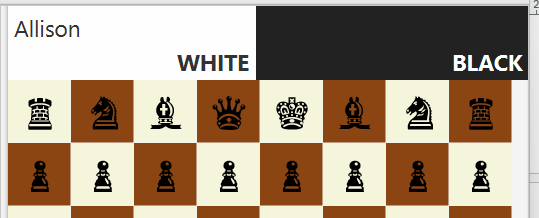
### 3.1.1 White Player and Black Player Filled



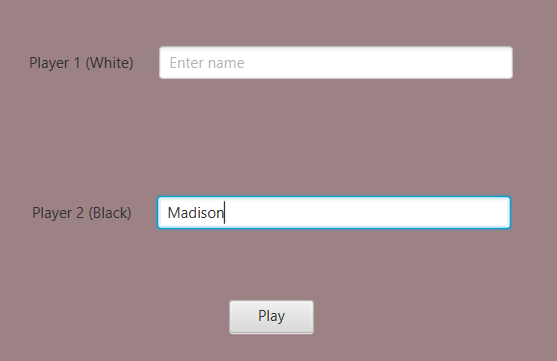


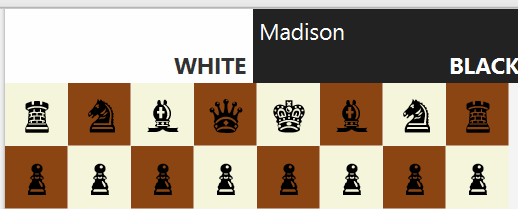
### 3.1.2 White Player Filled





### 3.1.3 Black Player Filled





### 3.1.4 Neither Players Filled

## 

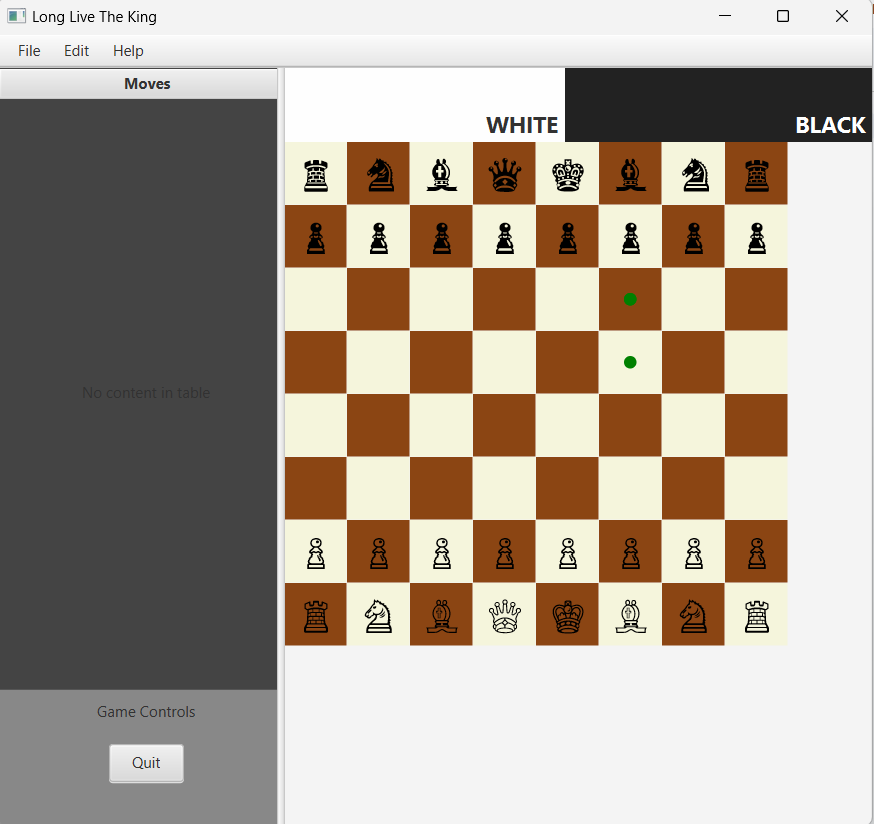
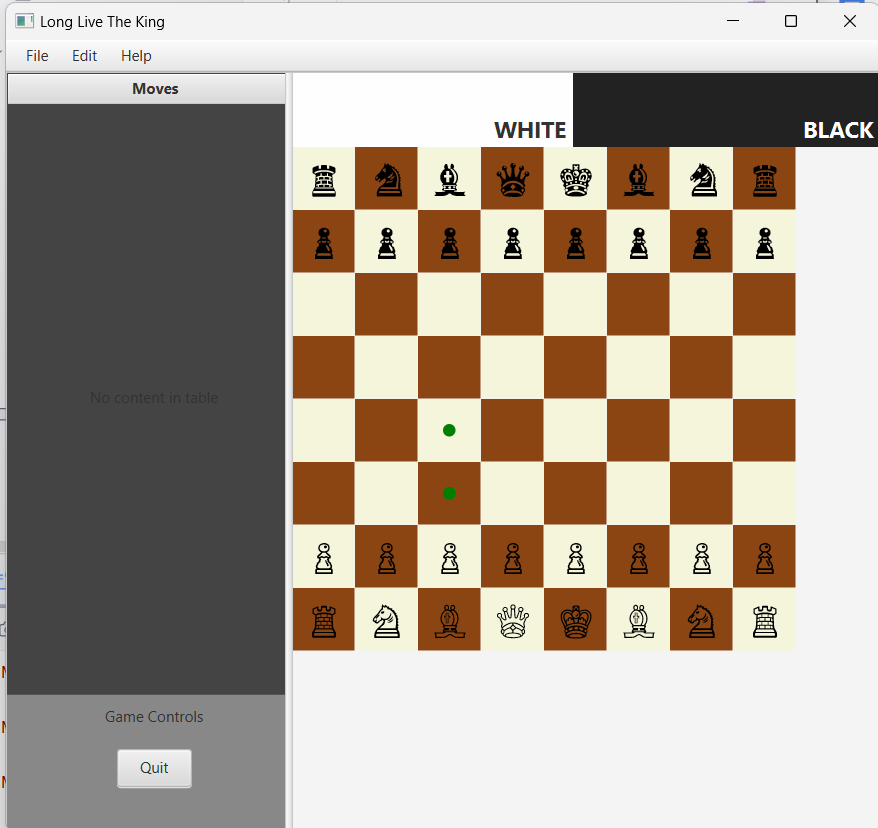


## 

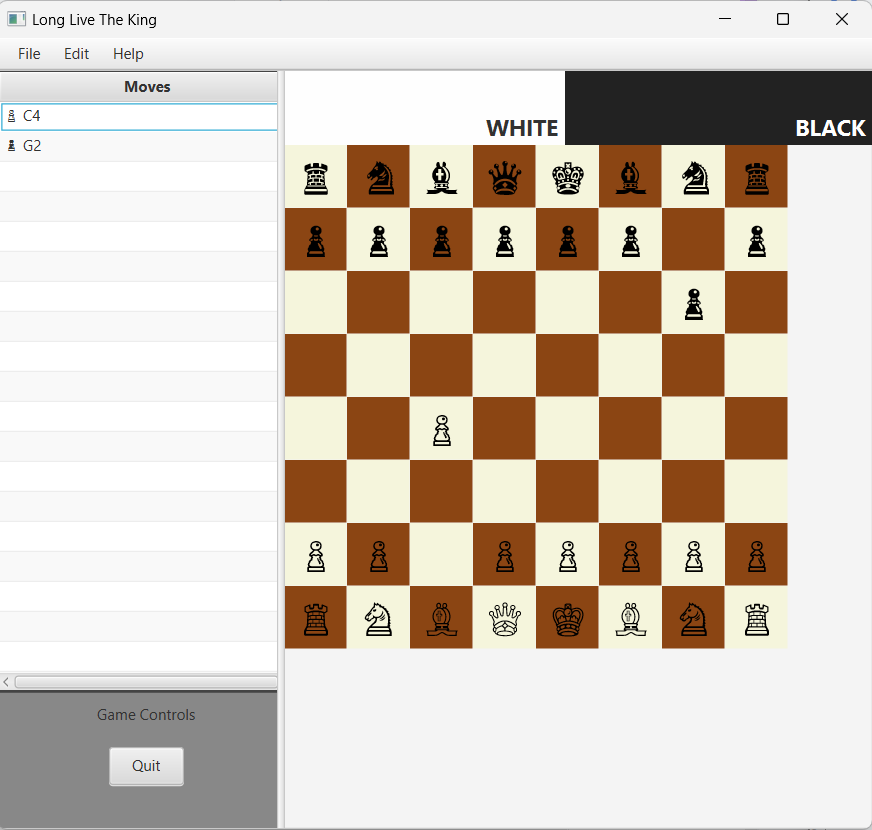
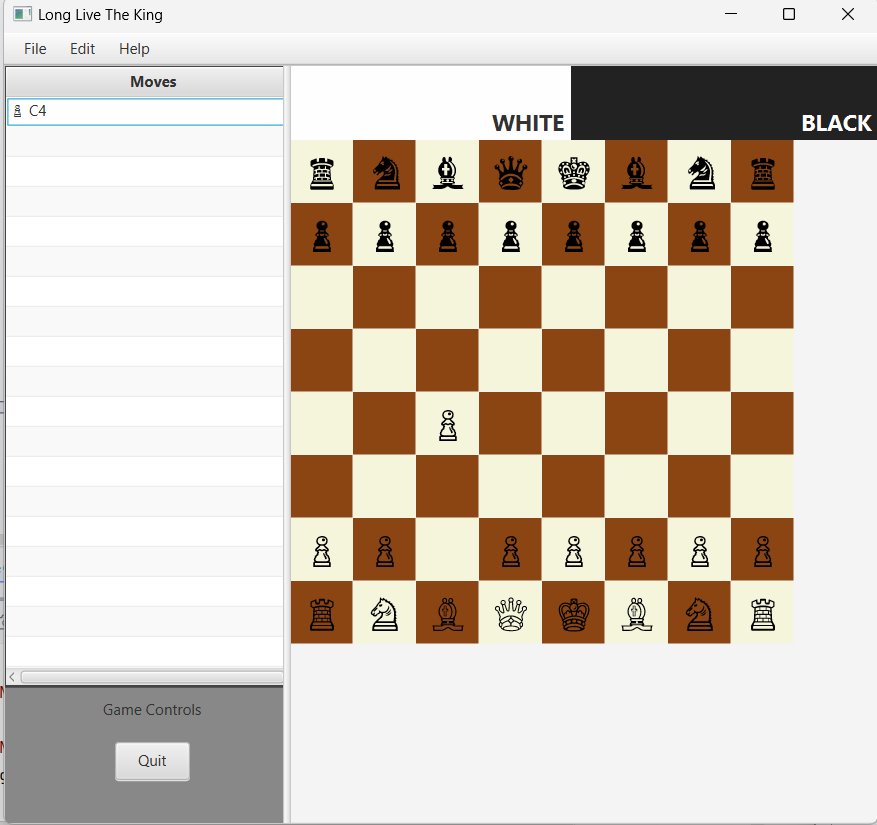
## 3.2 Piece Event

### 3.2.1 Pawn

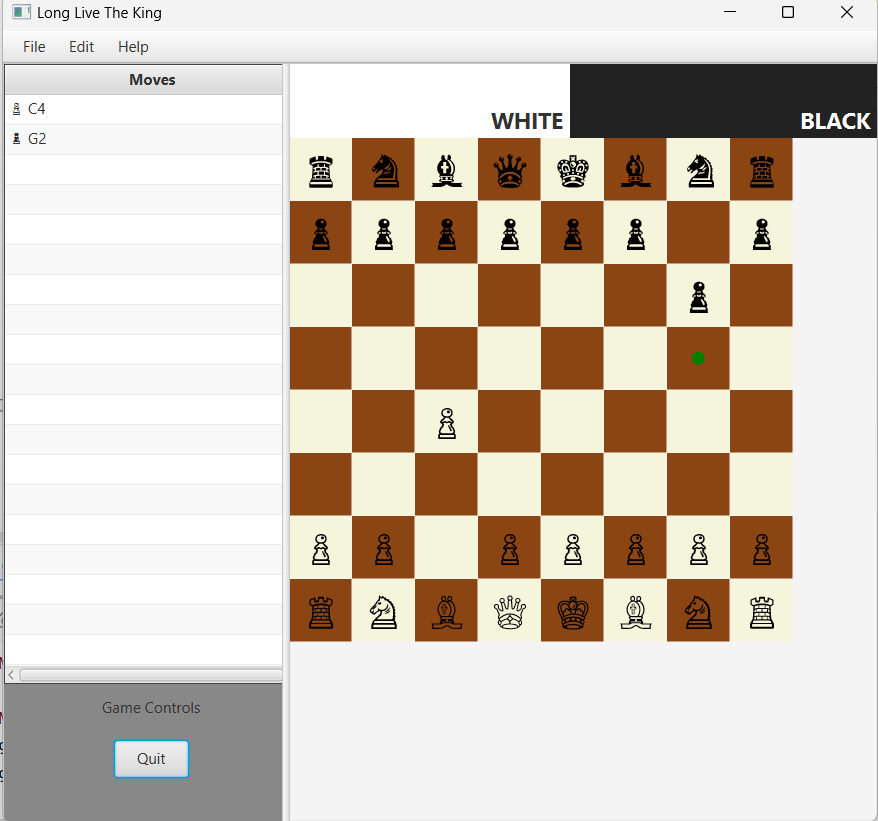
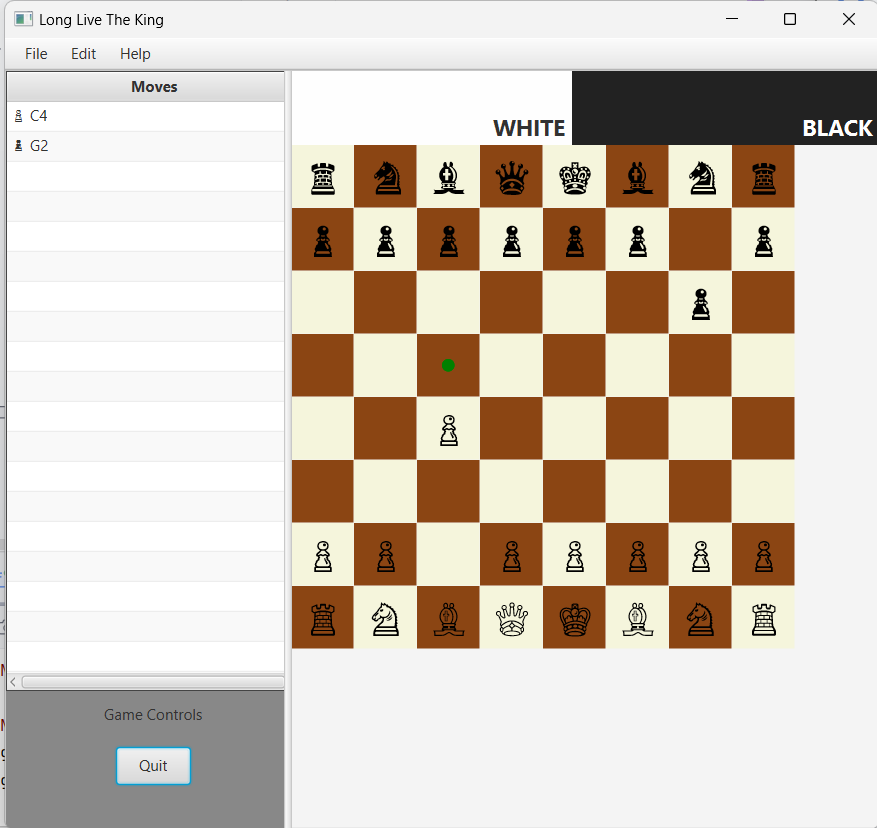
#### 3.2.1.1 Pawn On Click Start



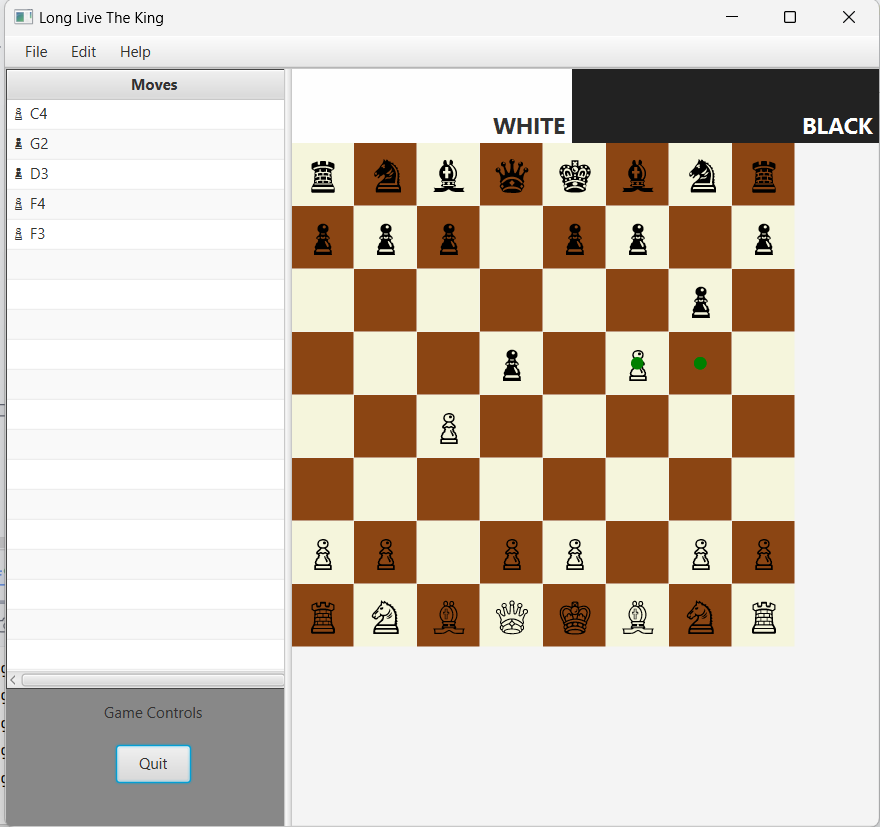
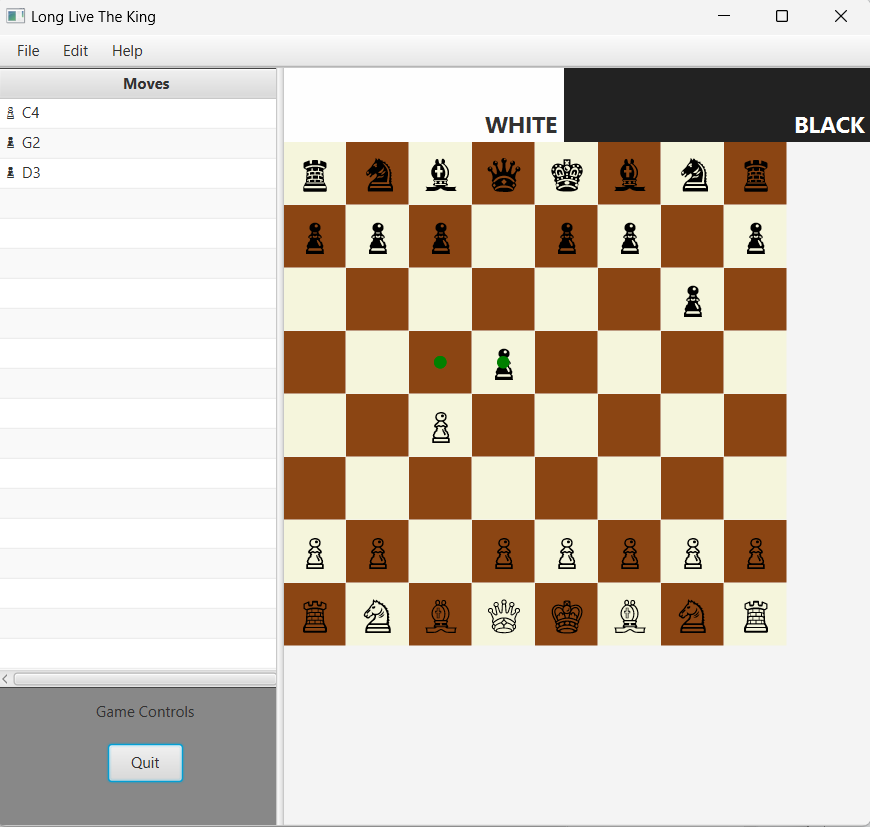
#### 3.3.1.2 Pawn On Move



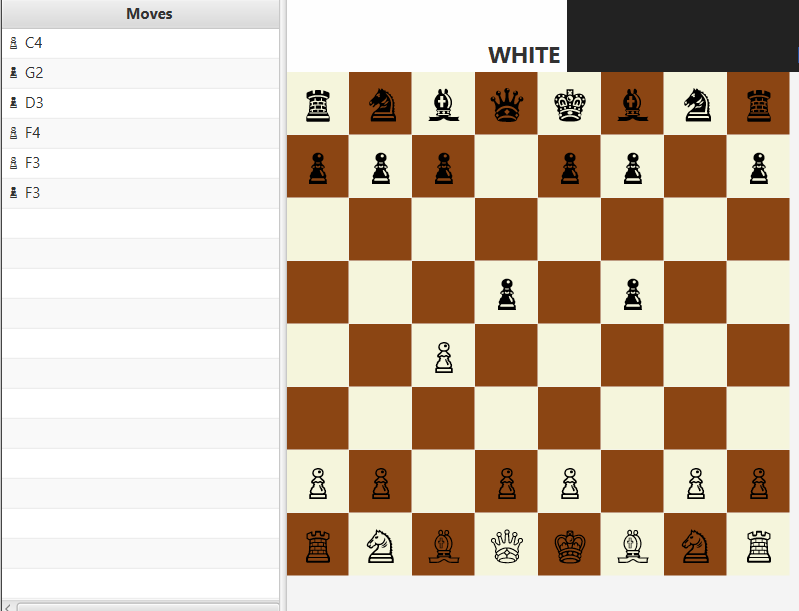
#### 3.2.1.3 Pawn On Click After First Move



#### 3.2.1.4 Pawn On Click Takeable

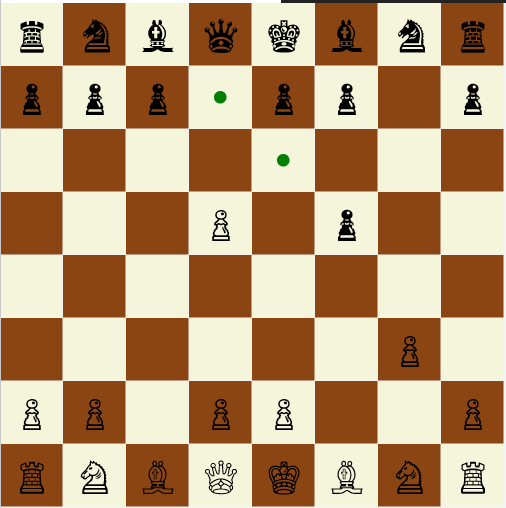


#### 3.2.1.5 Pawn On Take

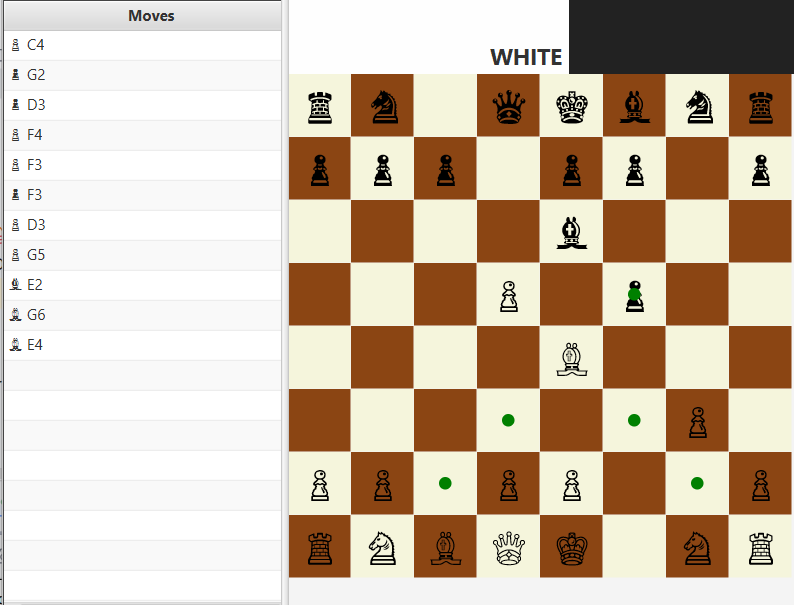


### 3.2.2 Bishop

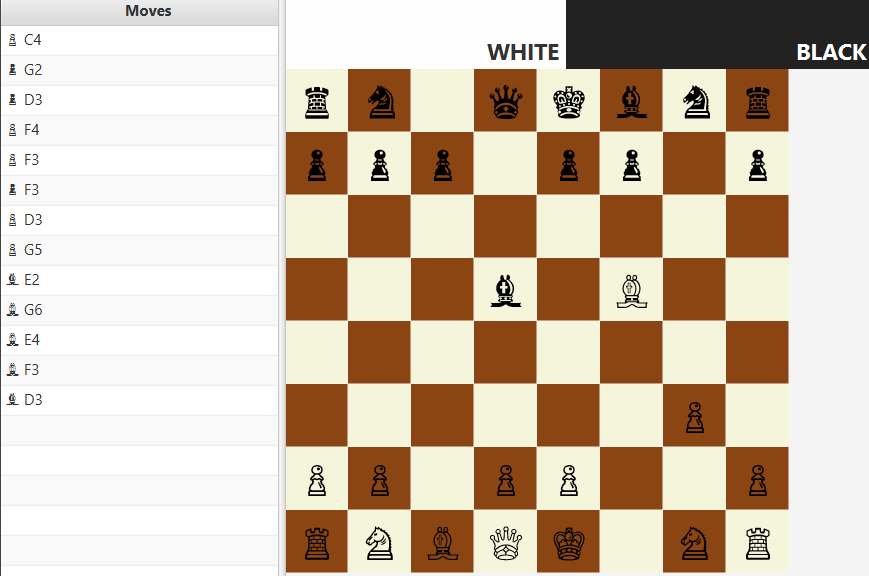
#### 3.2.2.1 Bishop On Click



#### 3.2.2.2 Bishop On Move and Takeable

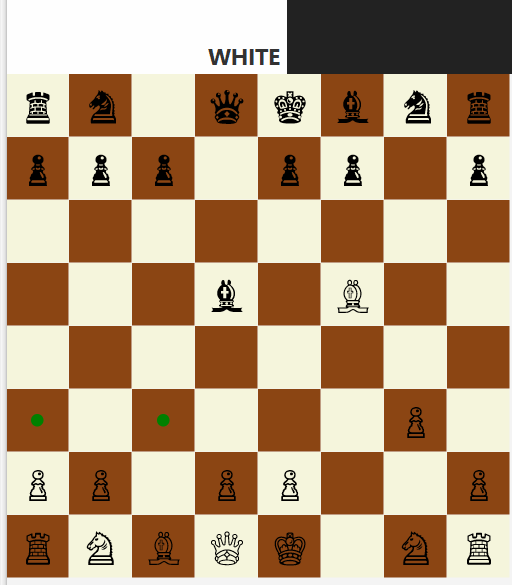


#### 3.2.2.3 Bishop On Take

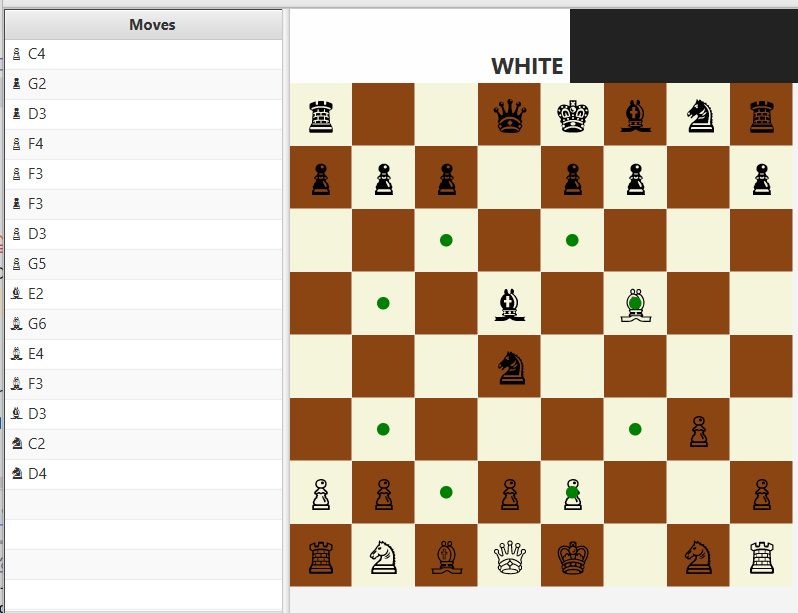


### 3.2.3 Knight

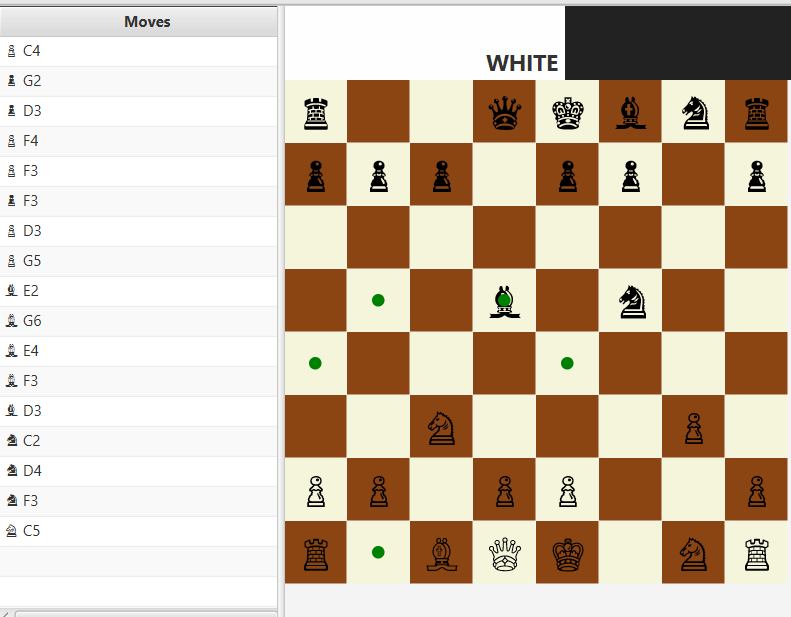
#### 3.2.3.1 Knight On Click



#### 3.2.3.2 Knight On Click, On Move, On Takeable

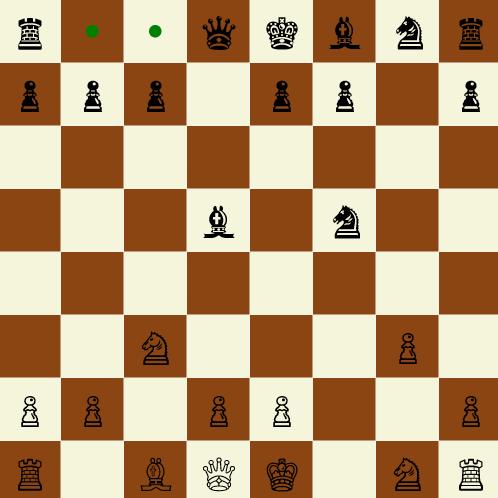


#### 3.2.3.3. Knight On Take

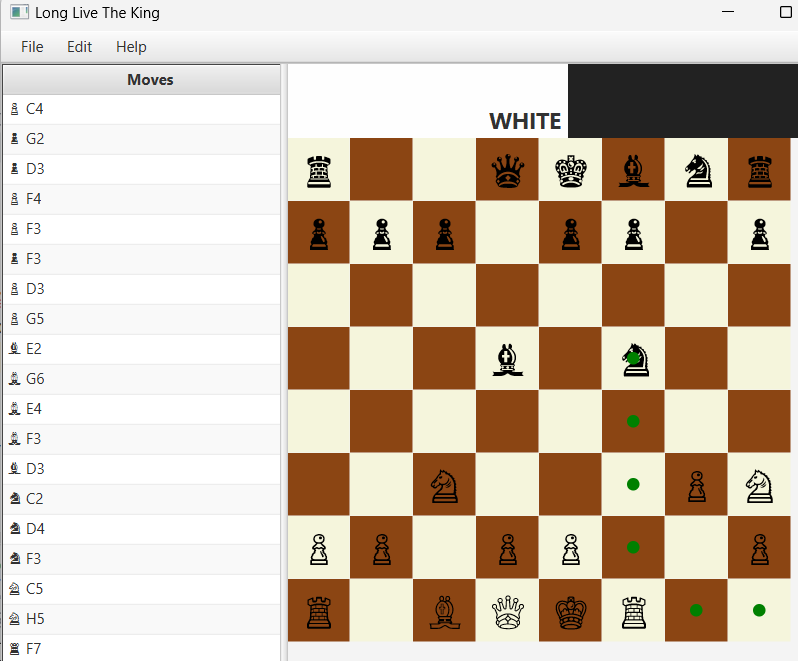


### 3.2.4 Rook

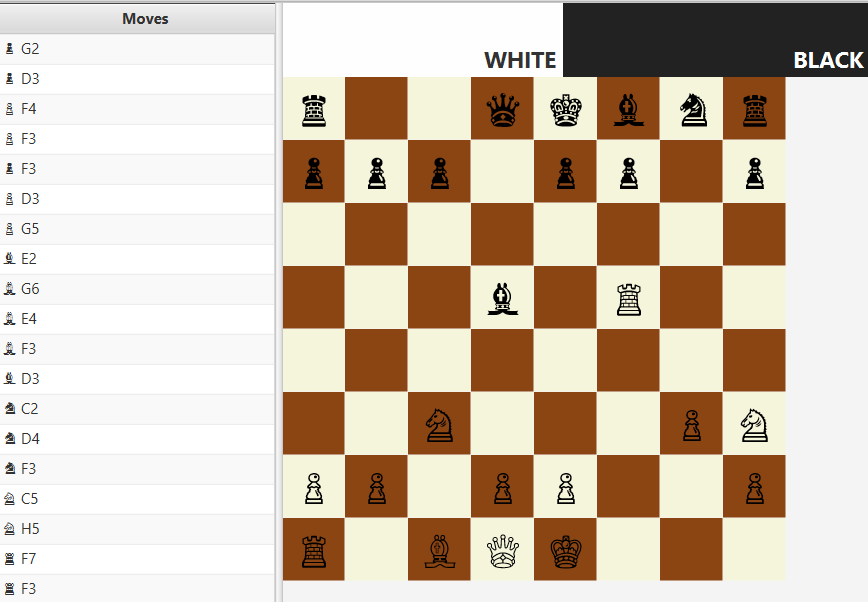
#### 3.2.4.1 Rook On Click



#### 3.2.4.2 Rook On Move, On Click, On Takeable

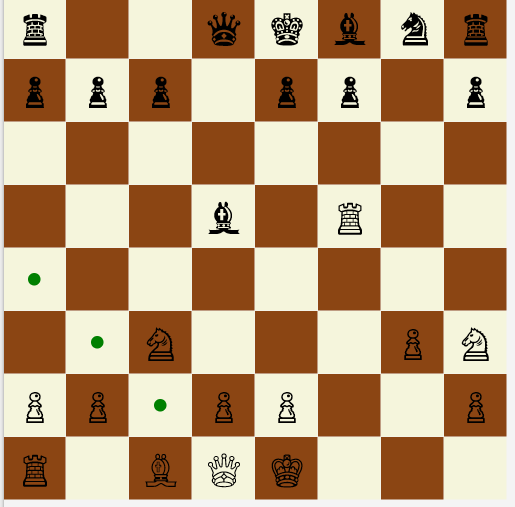


#### 3.2.4.3 Rook On Take, MoveTable On Scroll



### 3.2.5 Queen

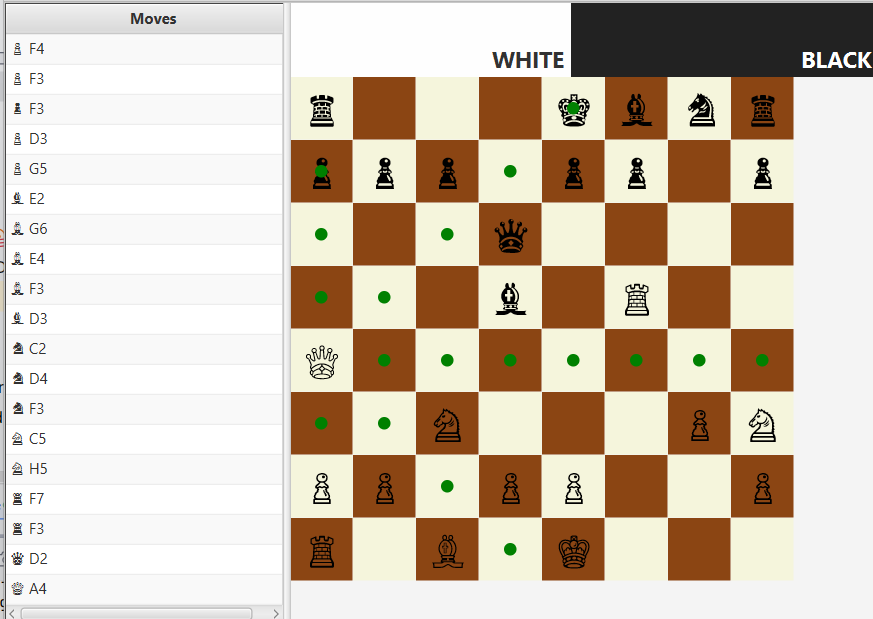
#### 3.2.5.1 Queen On Click



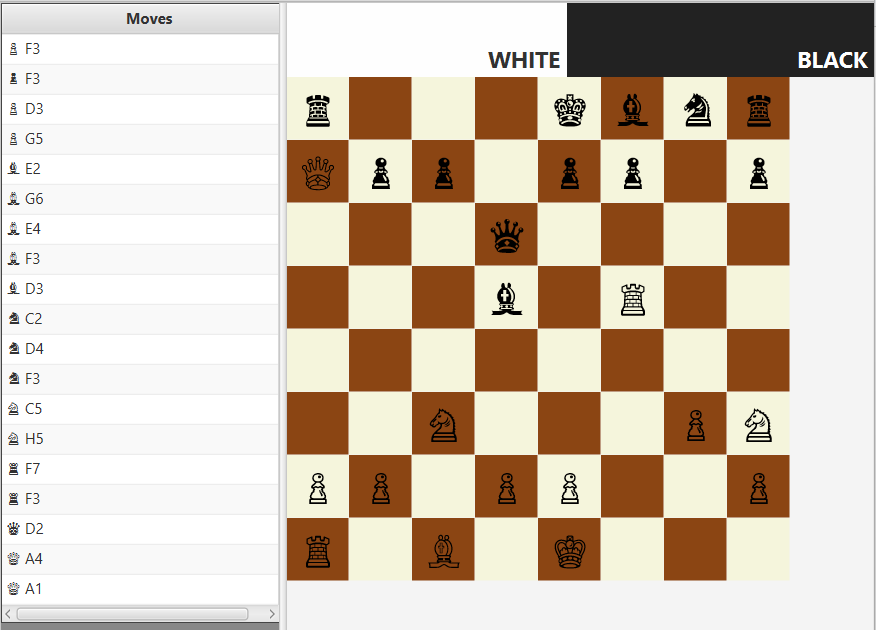
#### 3.2.5.2 Queen On Move



#### 3.2.5.3 Queen On Move, On Click, On Takeable

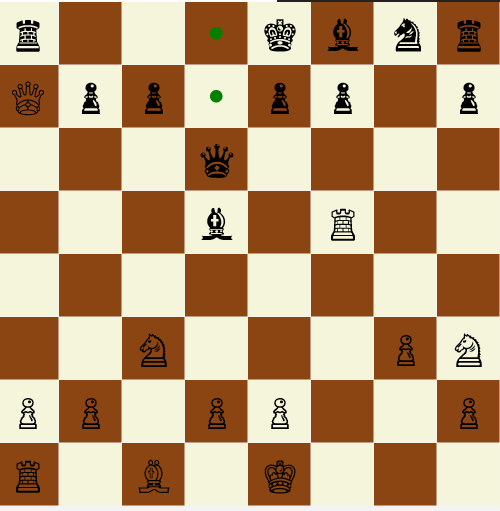


#### 3.2.5.4 Queen On Take

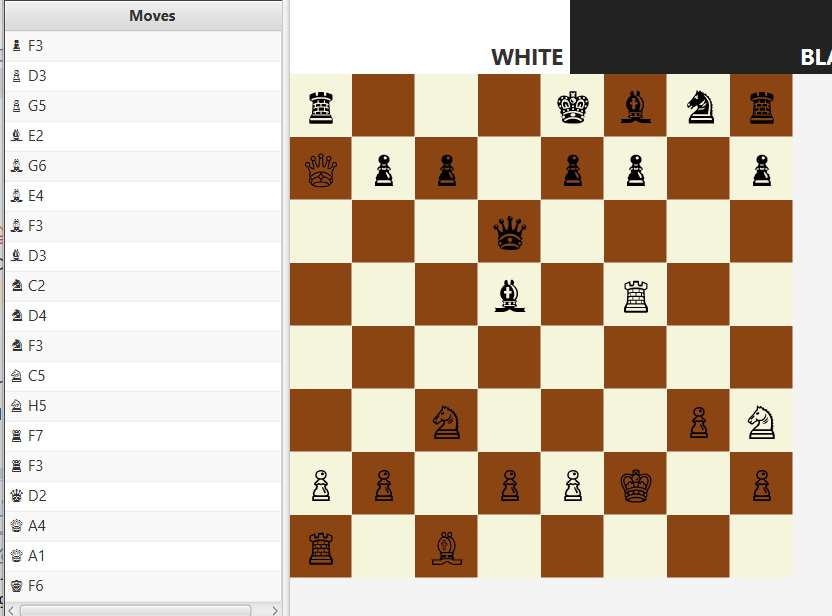


### 3.2.6 King

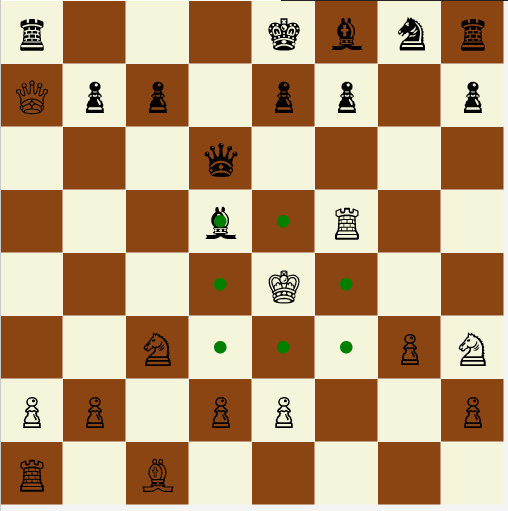
#### 3.2.6.1 King On Click



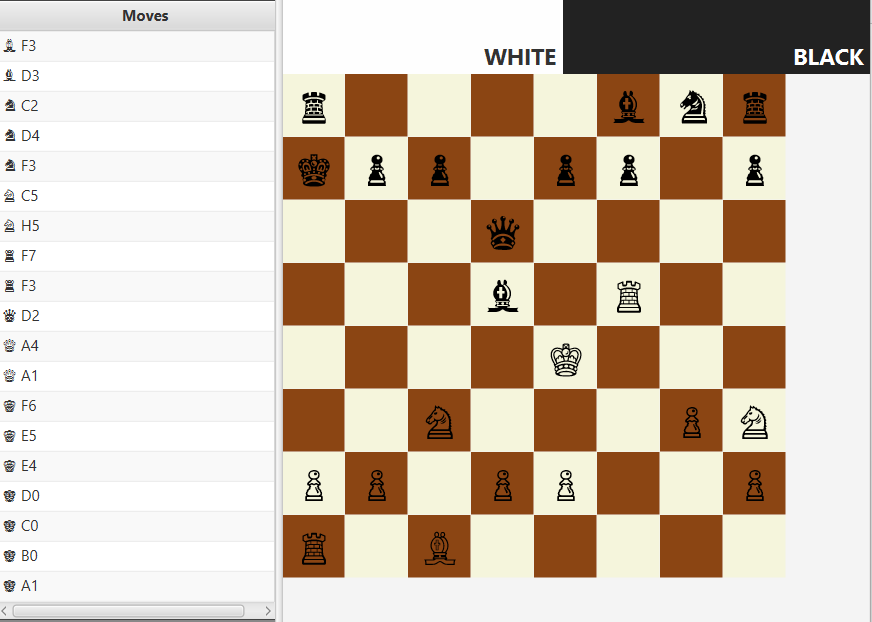
#### 3.2.6.2 King On Move



#### 3.2.6.3 King On Move, On Click, On Takeable

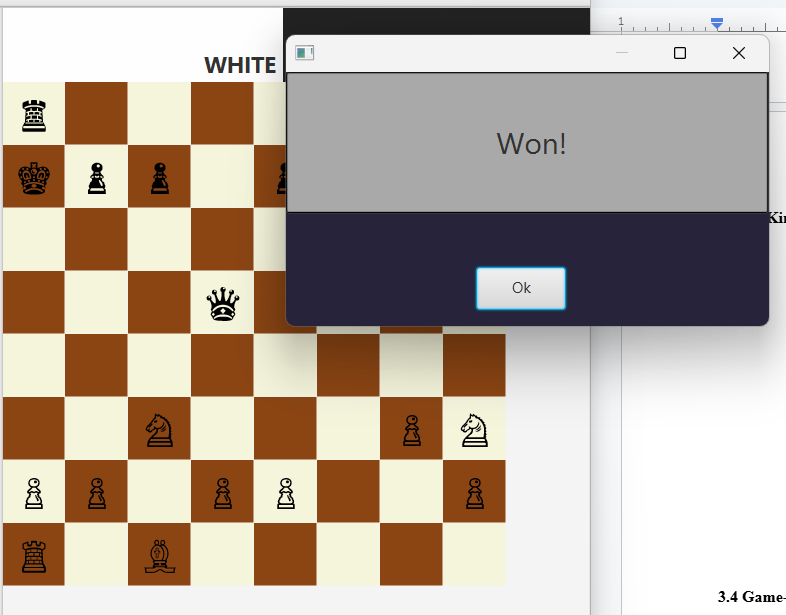


#### 3.2.6.4 King On Take

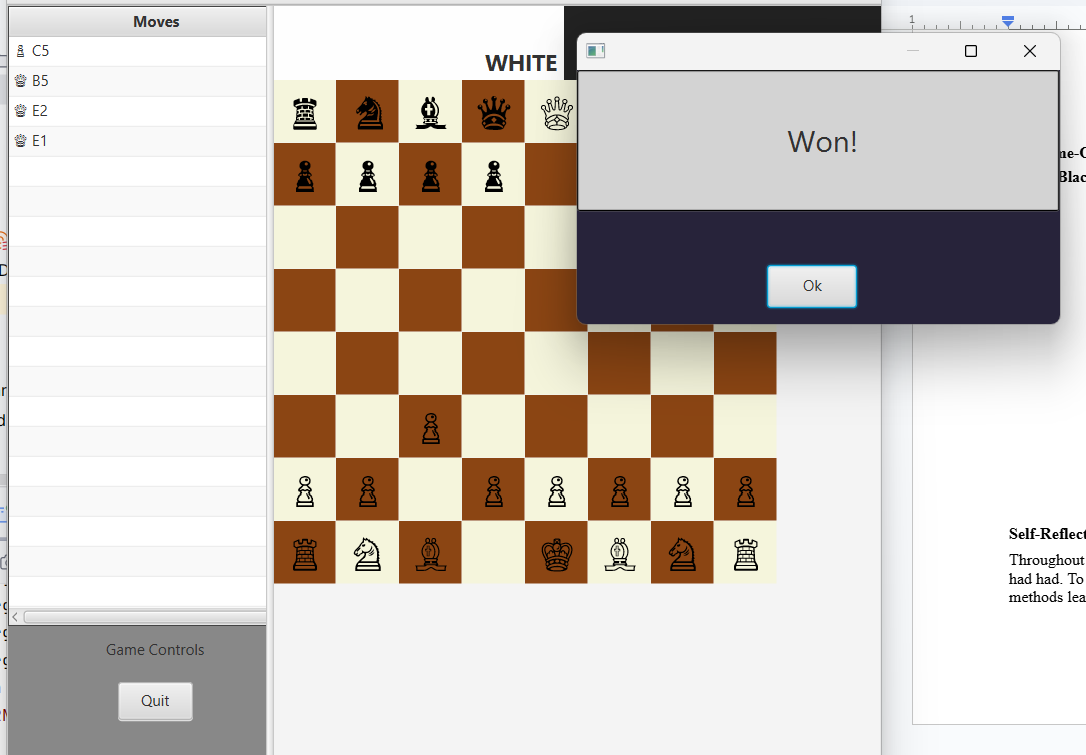


### 3.3.Game-Over Event

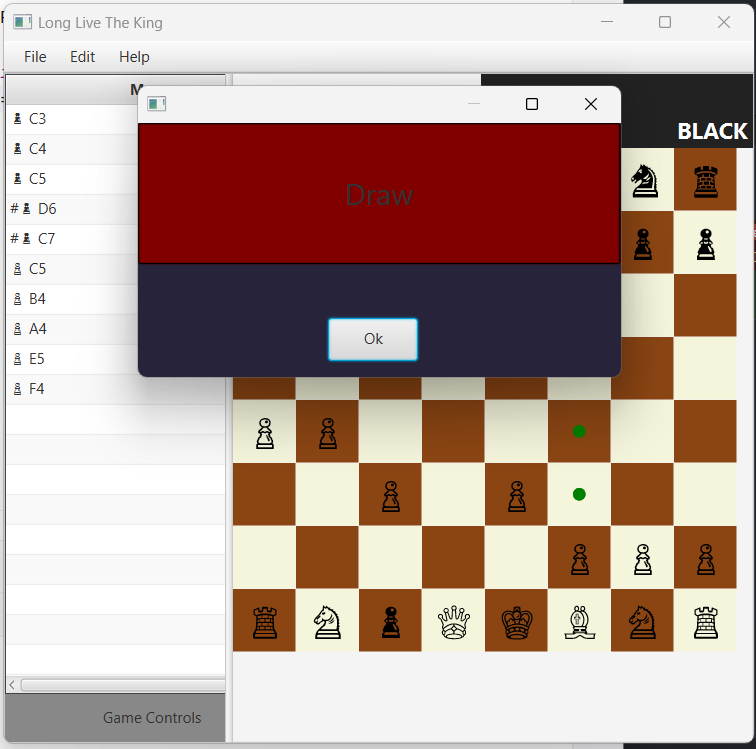
#### 3.3.4.1 Black Won

****

#### 3.3.4.2 White Won

****

#### 3.3.4.3 On Draw Event (Passive Play)

****

# 

# 4.0 Self-Reflection

Throughout this assessment, I had discovered many professional and personal problems that I had had. To start, I was beginning to overwhelm and doubt myself by strictly adhering to coding methods learnt throughout the second half of the pracit

# 5.0 References

Dr Chin Tek Min

Github CoPilot