Project Report:
CS Board-A Chess Game

Yassir Fri
Ilyass Hakkou
Zineb abercha

Supervised by: Abdelwahed Assklou

Université Mohammed VI Polytechnique

School of Computer Science

19 Decembre 2022

Table of Contents

0.1 Introduction

CS Board is a desktop and web application that allows you to play chess against your friends it also has a checkers game, The game is built with the help of the following technologies:

1. Java Programming Language
2. JavaFx library
3. Scene Builder
4. WebSwing

0.2 Download and Installation

0.2.1 Desktop Application

The desktop application is available for Windows, Linux and Mac OS, you can download it from the github repository:

`https://github.com/YASSIRFRI/JavaProject`

you can also download the compiled jar file from the same link.

0.2.2 Web Application

The web application is currently not deployed, but will be soon deployed on WebSwing server.

0.3 Runnig the application

0.3.1 Desktop Application

To run the desktop application, you need to have Java installed on your machine, then you can run the jar file by double clicking on it or by running the following command in the terminal:

```
java --module-path "PATH-TO-JAVAFX-LIB" --add-modules

 javafx.controls,javafx.fxml,javafx.media -jar "PATH-TO-JAR-FILE"
```

```
PS C:\Users\Lenovo\Desktop> java --module-path "C:\Program Files\Java\javafx-sdk-19\lib" --add-modules javafx.controls,j
avafx.fxml,javafx.media -jar .\LBD3\JavaProject\ChessGame\ChessGame.jar
```

0.4 Gameplay

0.4.1 Desktop Application

To start, select either the chess or checkers option from the main menu. Then enter your name and the name of your opponent, the default names are player 1 and player 2. Then you can specify the time limit for the game, the default time limit is 5 minutes. Then select the color you want to play as, and the color you want your opponent to play as. There are a lot of different color options to choose from, so you can customize the game to your liking. Once all these options are selected, click the "start game" button to begin playing.

To make a move in either game, simply select the piece you want to move and then select the square you want to move it to. The game will automatically enforce all of the rules of chess or checkers, so you don't have to worry about making illegal moves. If you make a move and then decide that you want to undo it, you can use the "reverse move" feature by pressing the arrow button on the bottom side of the board. This will take back the last move that was made and allow you to try a different move instead. You can undo as many moves as you want, until you reach the beginning of the game. When a pawn reaches the other side of the board in chess, a prompt will appear asking you if you want to promote the pawn to a queen, rook, bishop, or knight. You can then select the piece you want to promote to by clicking on it. If you don't want to promote the pawn, you can click the "cancel" button to cancel the promotion.

0.5 Features

1. Play chess and checkers against your friends
2. Access the game history and undo moves
3. Play with different color options
4. Play with different time limits
5. Play with different player names
6. Play with different board sizes
7. Listen to music while playing

8. Play with different 3 music options

0.6 Developer Guide

The game is build using Java programming language and JavaFx library, the menus are build using Scene Builder, and the web application is build using WebSwing web server.

The game architecture is based on the event driven programming paradigm. The chess game was built using the JavaFX framework, which is a toolkit for building rich and interactive applications for desktop, mobile, and web platforms.

The game features a visually appealing and intuitive interface, with clear graphics and layout designed to make it easy for users to understand the game and make their moves.

The game logic was implemented using a combination of custom code and external libraries, ensuring that all of the rules of chess are followed, and the game is fair and enjoyable for all players. User input is handled using event listeners and other JavaFX features, allowing players to easily interact with the game and make their moves using a mouse or other input device.

The chess game has been thoroughly tested and debugged, ensuring that it runs smoothly and reliably on a wide range of devices and platforms. The game and consists of several classes including Game, Gameboard, Chessboard, Checkerboard, Piece, Move, Promotion and Controller.

The Gameboard class represents the overall game board, which can be either a chess board or a checkers board depending on the game mode. The Chessboard and Checkerboard classes extend the Gameboard class and provide the specific rules and layout for each game mode. The Piece class represents the individual chess pieces, checkers pieces and the Controller class handles user information provided in the main menu.

0.6.1 Main Classes

- Game

- Gameboard
- Chessboard
- Checkerboard
- Piece
- Square
- Move
- Promotion
- Controller

Game

This class is the main class of the game, it contains the main method and the start method which is used to start the game. It also contains the main menu that is first displayed to the user when the game is started, it then passes the user information to the Controller class. which then passes it to the start game method to instantiate the game board and start the game.

Gameboard

This class represents the overall game board and contains the layout and graphics for the board and pieces It also extends event handler class, which allows it to listen for mouse clicks from user actions and trigger changes in the game state. It also handles basic game logic such as checking for legal moves and determining the winner.

Chessboard

This class extends the Gameboard class and adds the specific rules and layout for chess. It implements the move logic for chess pieces and enforces the rules of the game.It also overrides the handle method to handle the promotion of pawns to queens, rooks, bishops, or knights. This class implements the abstract method fillBoard() which is used to fill the board with the chess pieces. It also implements other methods like kingWillBeInThreat() and isKingInThreat().

Checkerboard

This class also extends the Gameboard class and adds the specific rules and layout for checkers. It implements the move logic for checkers pieces and enforces the rules of the game by using the getValidMoves method for both checkers king and checkers pawn. It also overrides the handle method to handle the promotion of the checkers pawn to a king, this method itself calls other helper methods such as getClickedSquare(), highlightValidMoves(), and removeHighlights() . This class implements the abstract method fillBoard() which is used to fill the board with the checkers pieces.

Square

This class represents a single square on the board and contains information about the piece that is currently on the square, if any. It also has an event listener that tracks user clicks. It the highlights the clicked squares accordingly. It has a Piece object called Placeholder that stores the image for the piece

Piece

This class represents the individual chess pieces, checkers pieces and the Controller class handles user information provided in the main menu.This abstract class represents the individual chess pieces and contains information about the piece's type, color, and current position on the board. It also handles the movement logic for each piece. It has an ImageView object that stores the image for the piece. This class is extended by the Pawn, Rook, Knight, Bishop, Queen, checkers pawn and king class, which implement the movement logic for each piece type, more specifically the getValidMoves method.

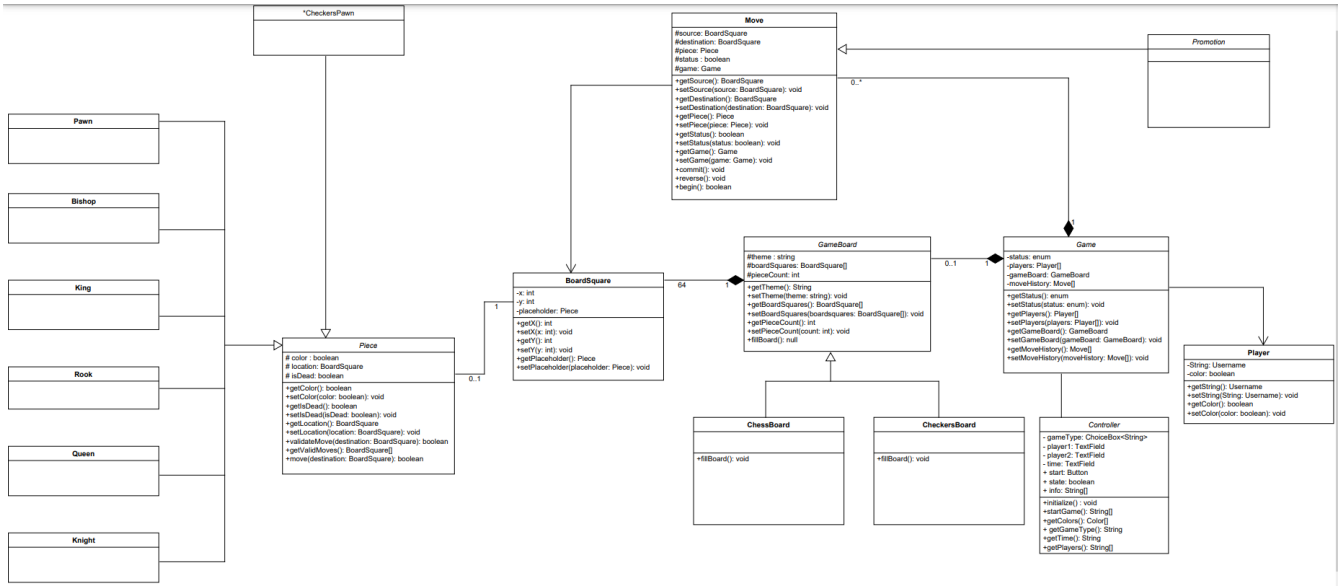
Controller

This class handles information provided in the main menu. It determines the color and name of each player, the time limit for the game, and the game mode. It also handles the logic for switching between the main menu and the game board.

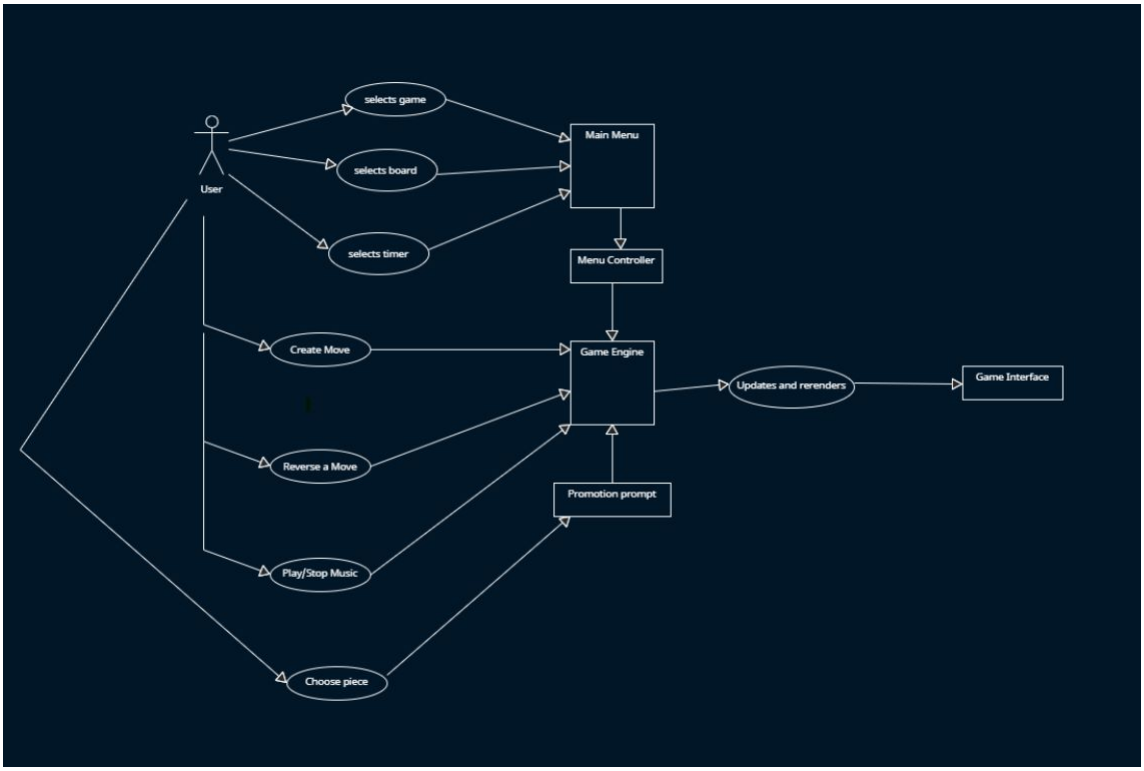
Move

This class represents a single move in the game and contains information about the piece that is being moved, the square that it is being moved from, and the square that it is being moved to. It also contains a boolean value that indicates whether or not the move is a capture. This class is used to store the history of moves in the game . It is also used to undo moves. And it also implements the reverseMove() method as well as the doMove() method that handles both of castling and enPassant moves for chess .This base class is extended by the a Promotion class, which handles the promotion of pawns to queens, rooks, bishops, or knights as well as the promotion of checkers pawn to a king.

UML Class Diagram



Flow Chart



0.6.2 Main Algorithms

- Legal moves
- Status of the king
- Castling feature
- EnPassant feature

Legal moves

To get the legal moves for a given piece, we run some for loops to get the standard chess valid moves for each piece, which is performed by the getValidMoves method. Then, we run another for loop over all the opposite team's pieces, and check if after making that move, the king can be killed or not. So, the final valid moves are given by the method getFinalValidMoves.

In this particular algorithm, some helper methods are used such as validate-Move, which returns whether the move will be safe for the king or not, which in turn relies on methods: isKingInThreat, and kingWillBeInThreat.

Status of the king

To detect whether the king is in check, checkmate, or stalemate, we use the methods described above, more specifically the isKingInThreat method. If it returns true, and all allies pieces have no legal moves, then the game is declared as checkmate. Otherwise, if all allies pieces have no legal moves and the king is not in check (i.e. isKingInThreat returns false), then the game is declared as draw or stalemate.

Castling feature

For the Castling feature, it is implemented inside the getFinalValidMoves method, simply by checking both sides of the board, and by checking if all conditions to do the castling are verified (such as all squares between the king and the rook are empty, and the king will not be in check in any of the squares he will move above). Then, the castling is handled further more in doMove method, to ensure that the king and the rook will move to their appropriate place.

EnPassant feature

For the EnPassant feature, an attribute called enPassantThreatedPawn is used, it is initialized everytime an enemy pawn moves two squares forward, otherwise, it is set to null. Then some code is added inside the doMove method to make the pawn disappear if it was killed using an EnPassant move, and to make sure that the killer pawn moves to his appropriate square.. .

0.7 Contributors

Shout-outs to:

Yassir Fri : 40 %

- Project management

- Overall architecture
- Application logic
- classes design
- Checkerboard Checkers pieces
- Menu Gui and Controller class
- Timer and Time limit
- Promotion
- Team coordination

Ilyass Hakkou:40%

- Implemented all methods related to getting legal moves for all chess pieces
- Improved the graphic user interface and the landing page.
- Handling user's mouse inputs
- EnPassant feature
- Castling feature
- Implemented doMove method
- Detecting Checkmate or Stalemate

Zineb Abercha:20%

- Checkersboard class
- CheckersPawn class
- Player class
- Designed Game class

0.8 Contribution

We are excited to invite developers to contribute to the development of this chess game. As a contributor, you will have the opportunity to work on a wide range of features and improvements, including AI opponents, graphical user interfaces, and gameplay mechanics. If you are interested in contributing to the project, please read the guidelines mentioned above and submit a pull request with your proposed changes. We welcome contributions from developers of all skill levels and encourage you to join our community of contributors.

Together, we can create a fantastic chess game that is enjoyed by players around the world. Thank you for considering contributing to this open source project!

0.9 license

WARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.