

Criterion B – Design

Total word count (not including test plan): 313

Contents

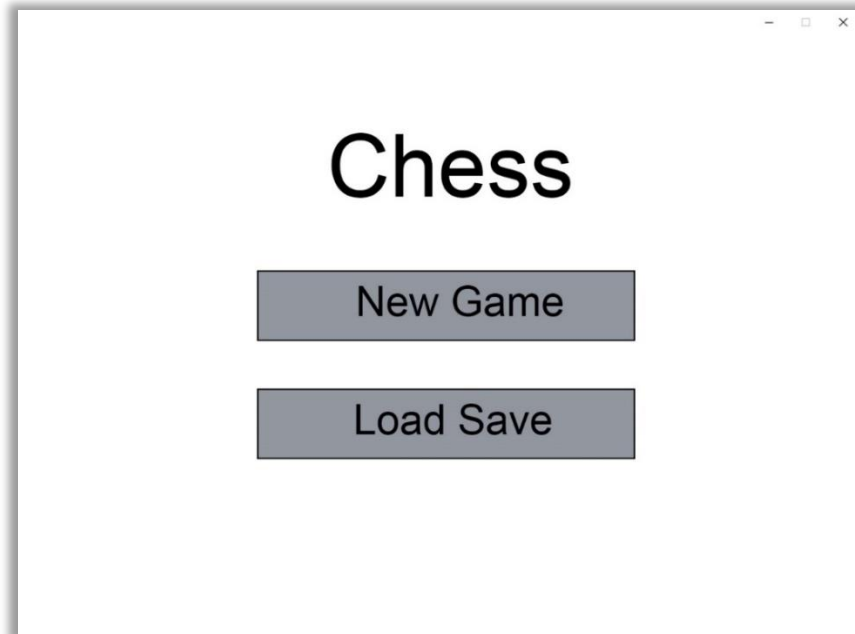
- Design of graphic interface
- Functionality
 - Menu hierarchy diagram
 - Game progression flow chart
 - Basic classes
 - Piece class
 - BoardLayout class
 - BoardLayoutManager class
 - Button click function pseudo code
 - Check and checkmate detection flow chart
 - Check
 - Checkmate
- Test plan

Design of graphic interface

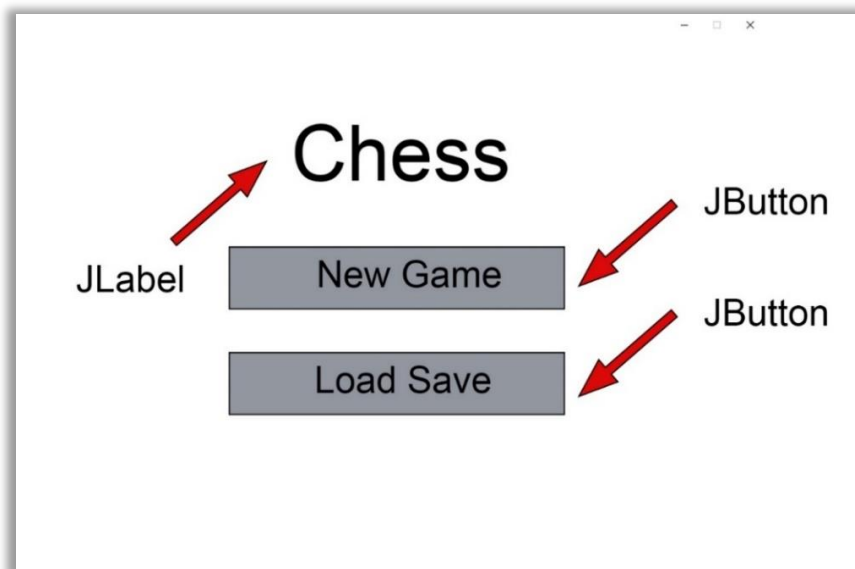
Menu screen window

This window will appear when the user launches the program. The first button will start a new game and the second button will start with the state of the game in the save file.

Design of interface



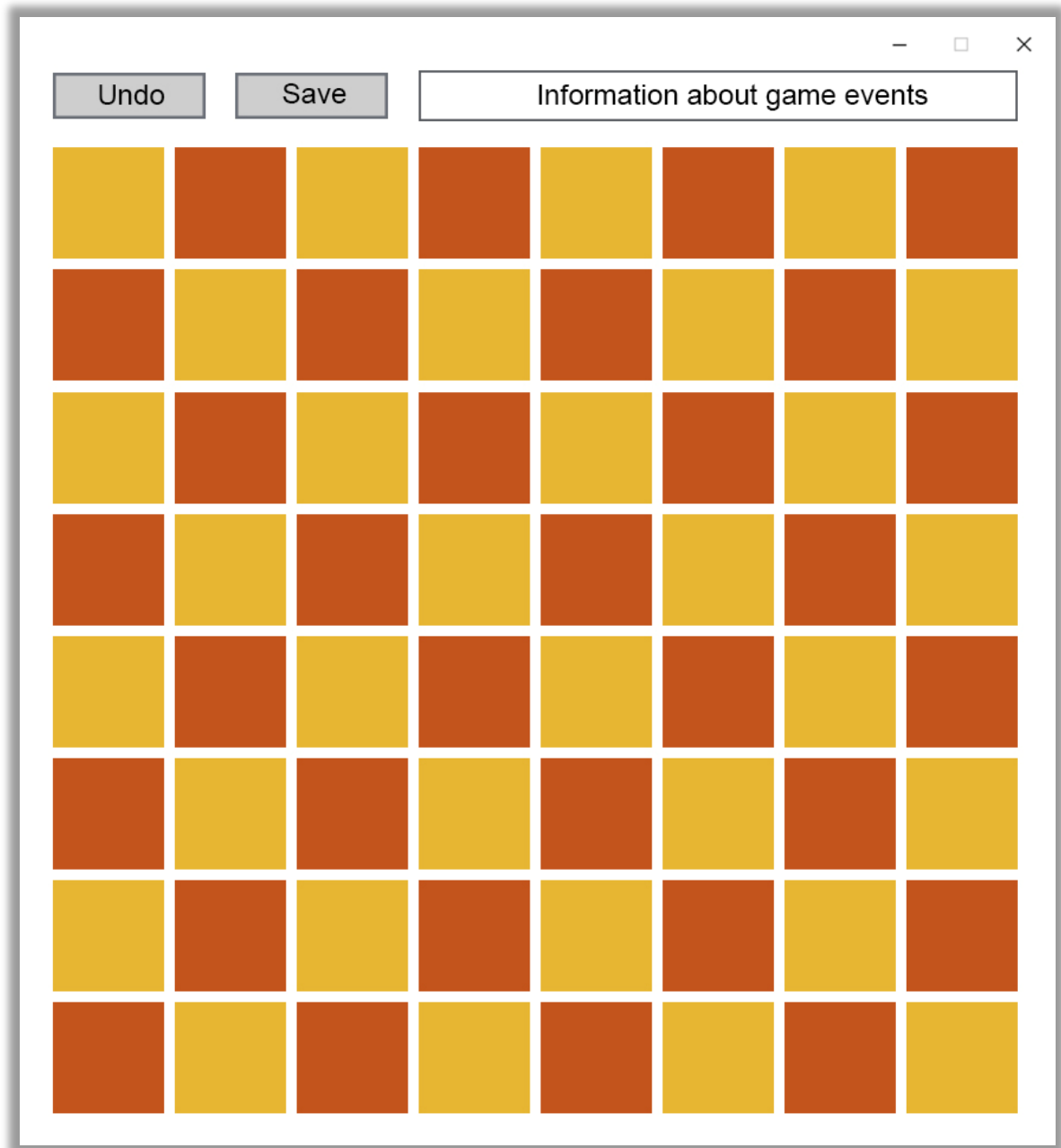
Labelling of components



Chess game window

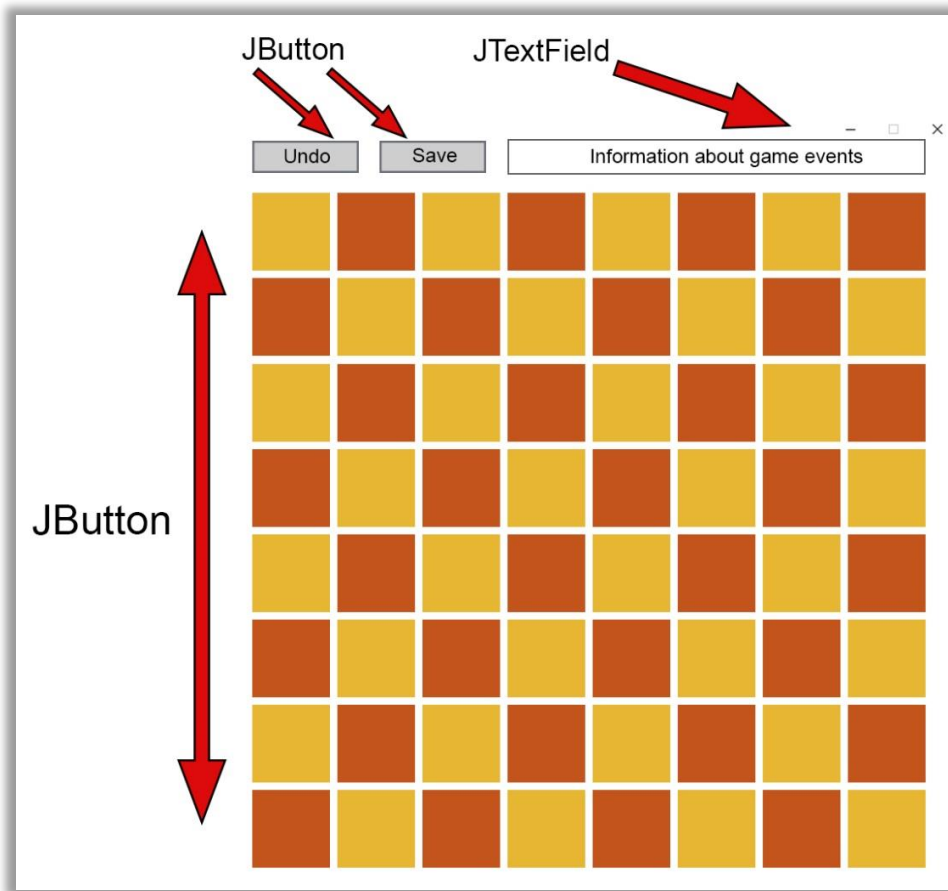
Each field of the chess board will be a button that can be clicked on by the players to interact with the pieces. Each piece will be represented as the title of the button that it is located on and its font will indicate the colour. The window will also contain a text field at the top of the application window that will display what action needs to be taken. There will be button next to it to undo moves and save button to save the current state of the game.

Design of interface





Labels of components



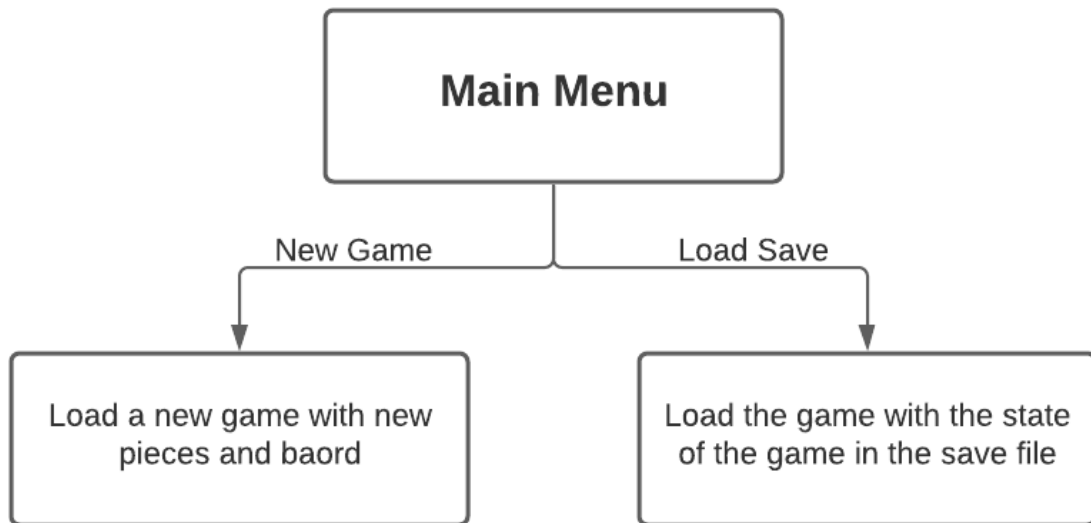
Sketches created using Adobe Photoshop CC

Section word count: 125

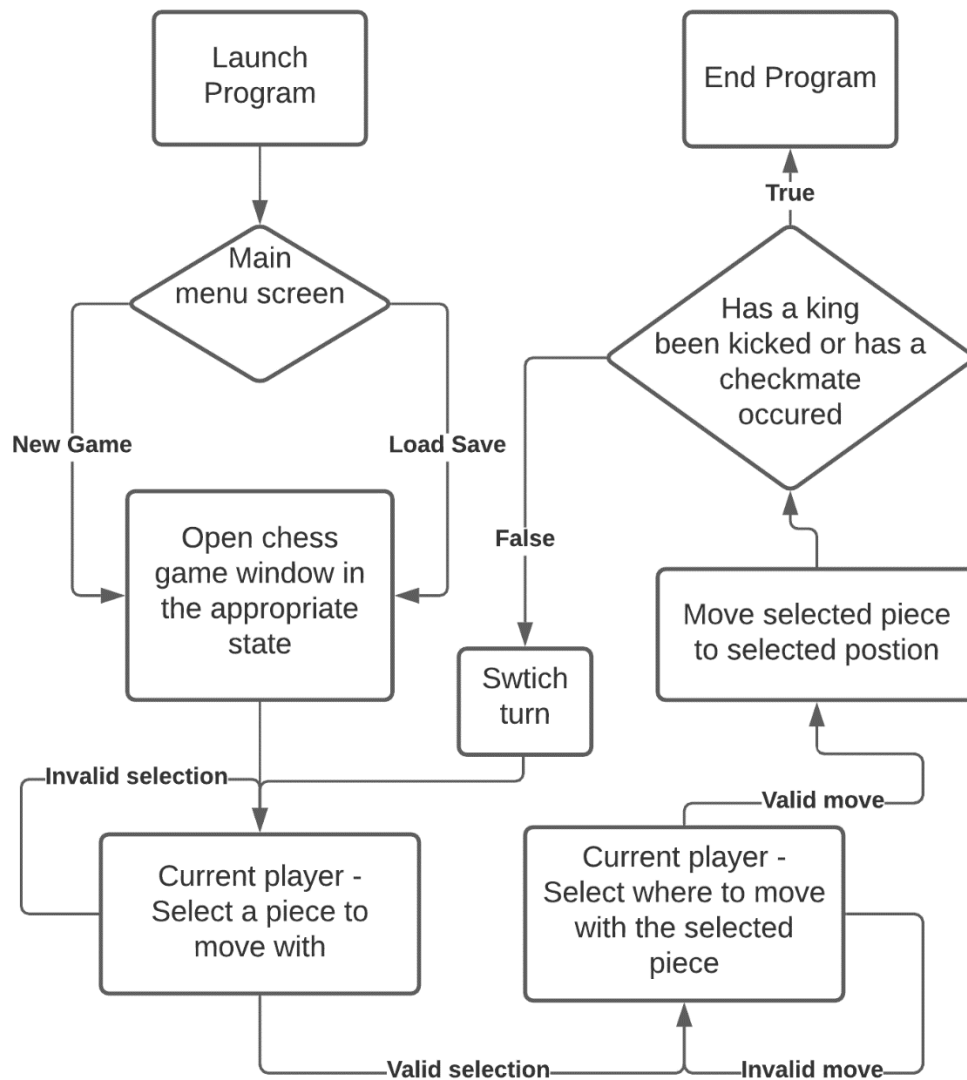
Functionality

The following section outlines user interaction and how the program will function.

Menu hierarchy diagram



Game Progression flow chart



Flowcharts created using lucidchart.com

Basic classes

Piece class

```
Class Piece

    Variable Name
    Variable Colour
    Variable PosX
    Variable PosY

    // Constructor function
    Function Piece(oName, oColour, oPosX, oPosY)

        Name = oName
        Colour = oColour
        PosX = oPosX
        PosY = oPosY

    End Function

    // Will return a boolean on if this Piece can move to the inputted position
    // This will vary based on each Piece
    Function IsValidMove(x, y)

End Class
```

Each Piece object will have a name, colour and a position value of where it is located on the board. Each type of Piece will have their own IsValidMove() function which will assert the validity of a potential move.

BoardLayout class

```
Class BoardLayout

    Variable PieceArray[][]

    // Constructor function
    Function BoardLayout(oPieceArray[][])

        PieceArray = oPieceArray

    End Function

    // Returns a boolean on if the inputted position is occupied
    Function IsAvalible(x, y)

        If PieceArray[x][y] Is NULL Then
            Return True
        Else
            Return False
        End If

    End Function

    // Returns the Piece at the inputted position
    Function PieceAt(x, y)

        Return PieceArray[x][y]

    End Function

End Class
```

The BoardLayout object will hold information about the location of Pieces on the board.

BoardLayoutManager class

```
Class BoardLayoutManager

    Variable BoardStack
    Variable Turn = "White"

    // Switches the turn colour
    Function SwitchTurn()

        If Turn Is "White" Then
            Turn = "Black"
        Else
            Turn = "White"
        End If

    End Function

    // Moves the inputted Piece to the inputted position (without validation)
    // The new move is recorded as a new BoardLayout that is added to the BoardStack
    Function MovePiece(Piece, x, y)

        Variable CurrentBoard = BoardStack.peek()
        Variable NewArray[][]

        // Creating a copy of CurrentBoard PieceArray
        For xx In Range 0 To 8
            For yy In Range 0 To 8
                CurrentBoard.PieceArray[xx][yy] = NewArray[xx][yy]
            End Loop
        End Loop

        // Inputted position in the array now holds the inputted Piece
        NewArray[x][y] = Piece
        // Previous position of that Piece is now empty
        NewArray[Piece.PosX][Piece.PosY] = NULL

        // Update the Piece's position to where it is now
        Piece.PosX = x
        Piece.PosY = y

        Variable NewBoard = BoardLayout(NewArray)

        // Add the new BoardLayout to the BoardStack
        BoardStack.push(NewBoard)

        SwitchTurn()

    End Function

End Class
```

The BoardLayoutManger object will control the movement of pieces in the game. It will contain a stack and whenever a Piece is moved, a new BoardLayout with the change implemented will be added to it.

Button click function pseudo code

```
// Position of the button that was clicked
Input ClickPosX
Input ClickPosY

// The BoardLayoutManager Object in play
Input BLM

// Boolean stating whether the button click is meant to select or move a Piece
Variable IsSelectionClick = True

// The Piece that the current player has selected and will move with
Variable SelectedPiece

Variable CurrentBoard = BLM.BoardStack.peek()

// If the current player needs to select a Piece to play with
If IsSelectionClick Is True Then

    // If there is a Piece at that position
    If Not CurrentBoard.isAvalible(ClickPosX, ClickPosY) Then

        // If the Piece is theirs
        If CurrentBoard.PeiceAt(ClickPosX, ClickPosY).Colour Is BLM.Turn Then

            // Set the SelectedPiece variable to the selected Piece
            SelectedPiece = CurrentBoard.PieceAt(ClickPosX, ClickPosY)

            // This is now False since a Piece has been selected
            IsSelectionClick = False

        End If

    End If

End If

// A Piece has been selected, Moving the selected Piece
Else

    // If the clicked position is a valid move for the selected Piece
    If SelectedPiece.IsValidMove(ClickPosX, ClickPosY) Then

        // Move the selected Piece to the selected position
        BLM.MovePiece(SelectedPiece, ClickPosX, ClickPosY)

        // This is now True since next player needs to select a Piece next
        IsSelectionClick = True

    End If

End If
```

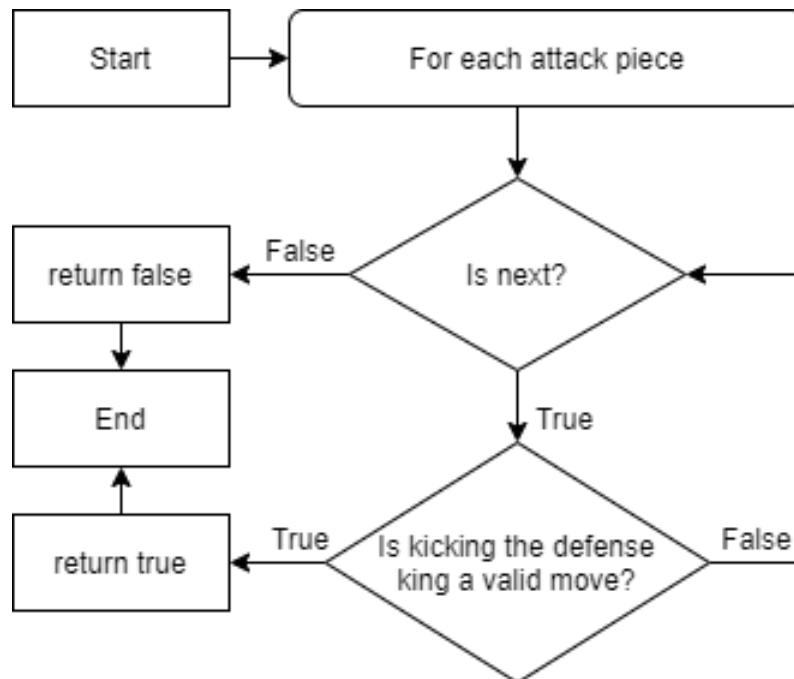
This function will run whenever a button on the chess board is clicked.

Check and checkmate detection

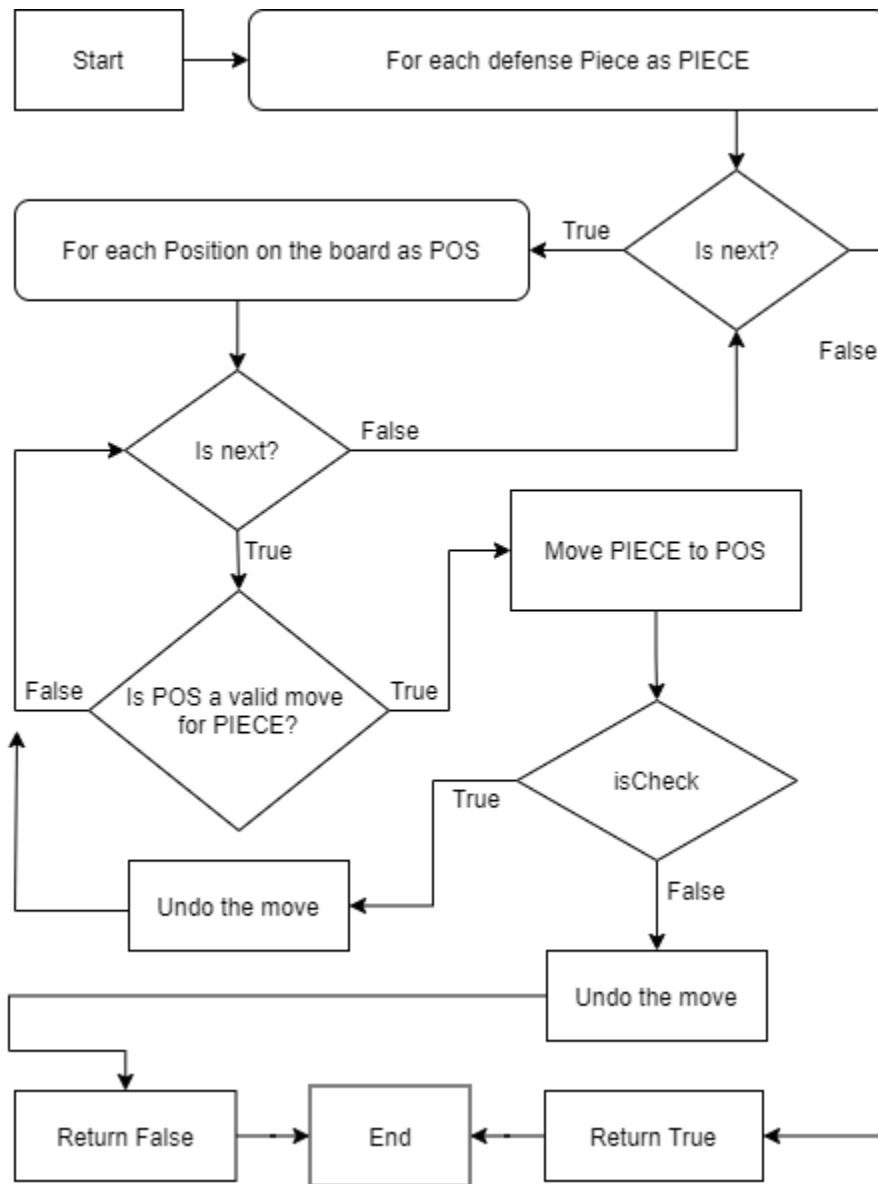
Whenever a Piece on board is moved, the program will examine the current BoardLayout to see if check has occurred. If a check has been detected, the program will also search for a potential checkmate.

The player that has just completed the move will have all their current pieces stored in data structure that will be referred to as “attack”, as these are attacking the king. The opponents Pieces will be referred to as “defence”.

Check flowchart



Checkmate flowchart

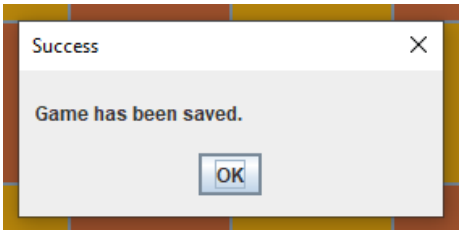
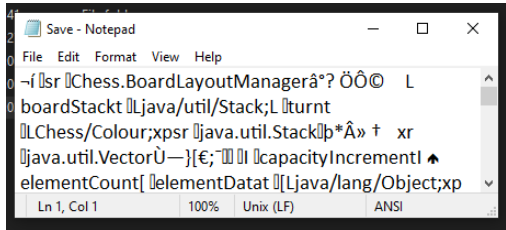


Section word count: 188

Flowcharts created using draw.io

Test plan

Test	Criteria number	Test description	Test data	Expected outcome
1	1, 3	When a button with a piece on it is clicked on, if it is the turn of that player whose piece was clicked on, that piece should get selected.	Clicking a white piece on the board when the player of the white pieces has their turn.	The piece clicked on should become selected and shown on the GUI.
2	1, 3, 6	When a button with a piece on it is clicked on, if it is the turn of that player whose piece was clicked on, the positions that piece can move should be highlighted.	Clicking a white piece on the board when the player of the white pieces has their turn.	The position to which the selected piece can move to are highlighted.
3	1, 3	When a button with a piece on it is clicked on, if it is the opposite turn of that player whose piece was clicked on, the program should indicate that it is not a valid selection.	Clicking a white piece on the board when the player of the black pieces has their turn.	The button that was clicked should be highlighted in red.
4	1, 3, 6	When piece is already selected, if a button is clicked that is not displayed as a possible move, the program should indicate that it is not a valid move.	Once a white piece has been selected, then clicking a button that is not highlighted as a possible move.	The button that was clicked should be highlighted in red.
5	1, 3	If no piece has been selected an empty position on the board is clicked on, the program should indicate that it is not a valid selection.	Clicking on button that has no piece on it when no prior selection has been made.	The button that was clicked should be highlighted in red.
6	1, 3	When a piece has already been selected, if the next button click is a piece of the same colour, the program should change the selected piece to the one on the button that was clicked.	Clicking on a white piece, when it is the white players turn, and another white piece has already been selected.	The possible moves of the newly selected piece should be highlighted on the board.
7	3, 4, 5	If a piece is moved and the opposing players king is now in check, the program should indicate a check.	Moving a white piece that can now kick the black king in the next turn.	The program should state that the opposing player is in check.

8	3, 4, 5	If a piece is moved and the opposing players king is now in checkmate and cannot be saved, the program should indicate that the player is in checkmate.	Moving a white piece that can now kick the black king in the next turn and nothing can prevent it.	The program should indicate that the opposing player is in check
9	7	When the undo button is clicked, the program should revert the game to state it was, prior to the last move.	White piece is moved to an unwanted position, the undo button is clicked.	The program should undo the move and revert the game to the state it was in previously.
10	7, 8	When the undo button is clicked and no piece has been moved yet, the program should ignore the request and state that no moves can be undone.	A new game is started, and the undo button is clicked immediately after.	The program should ignore the request and state that no move can be undone.
11	2	Upon opening the program, the Menu screen should open	The program is executed.	The Menu screen appears.
12	2	In the Menu screen, when the new game button is clicked, the chess game window should open with all pieces in their starting positions.	The new game button is clicked in the Menu screen.	The Menu screen disappears, the chess game window appears with pieces in their starting positions.
13	2	In the Menu screen, when the load game button is clicked, the chess game window should open with the previous state of the game when it was last saved.	The load game button is clicked in the Menu screen.	The Menu screen disappears, the chess game window appears in the state that it was last saved in.
14	9	When the save game button is clicked in the chess game window, a message dialog show either state an error or a success in saving the file.	The save game button is clicked on in the chess game window.	A message dialog appears stating that the game has been saved. The save file is either located or a new ones is created and its contents are altered.
	<div style="display: flex; justify-content: space-around;">   </div>			

