Software Design Document

for

Chesstackle

Version 1.0 approved

Prepared by Snake Army (Group 1)
Burmeister Uruchurtu, Alec
Blanco, Bruno
Lockman, Cory
Lemmon, Dylan
Currie Buckner, Ty Baron

Texas State University

03/06/2018

Contents

1. Introduction

1.1 Purpose

This program is intended to mimic a game of chess, following standard chess rules including piece movement.

1.2 System Overview

Each piece will be its own class, inheriting its functions from a 'piece' class. The board will be managed by a database which will be updated by a manager class. The board will update and redraw based on changes to the database.

1.3 Definitions, Acronyms and Abbreviations

- Castling – Castling consists of moving the King two squares towards a Rook on the player's first rank, then moving the Rook to the square over which the King crossed. Castling may only be done if the King and Rook involved have not moved this game, the squares between the King and the Rook involved are unoccupied, the King is not in check, and the King does not cross over or end on a square in which it would be in check.

- Check – A state in which the King will be taken in the opponent's next move. The side in check may not end their move with their King still in check.

- Mate – End of the game. The King is mated when check state cannot be removed via legal moves.

- Bishop – Each player starts the game with two Bishops. White having them start on squares c1 and f1, and black having them start on squares c8 and f8. The Bishop can only move diagonally and will always stay in squares of the same color it started the game in (a Bishop starting on a white square will never move into a black square).

- King – The King is the most important piece in the game, a mated King ends the game with the mated King's side losing. Each player only has one King, white having it start on e1 and black having it start on e8. The King can only move one space horizontally, diagonally, or vertically. The King cannot move into a position that would cause it to be in check.

- Knight – Each player starts the game with two Knights. White having them start on squares b1 and g1, and black having them start on squares b8 and g8. The Knight moves by first moving 1 square horizontally or vertically, then 1 square diagonally in an outward direction. The Knight may also jump over pieces of either color to reach its destination.

- Pawn – The lowest value piece in a chess game that can move one square forward (or two on its first move) and can only capture another piece when moving one square diagonally. Each player starts with eight Pawns on the second rank of the board from each players perspective. If a Pawn reaches the opponent's end of the board it can be promoted to any other piece.

- Queen – Each player starts the game with only one Queen. White having it start on square d1 and black having it start on square d8. The Queen may move horizontally, vertically, and diagonally any number of squares but may not jump over pieces.
- Rook – Each player starts the game with two Rooks. White having them start on squares a1 and h1, and black having them start on squares a8 and h8. The Rook can only move horizontally or vertically across the chess board.
    - This portion will be updated as necessary as the document grows.

1.4 Supporting Materials

1. Bodlaender, Hans. “The Rules of Chess.” The Chess Variant Pages, www.chessvariants.com/d.chess/chess.html
2. “Chessboard.” Wikipedia, Wikimedia Foundation, 24 Jan. 2018, en.wikipedia.org/wiki/Chessboard.

This portion will be updated as necessary as the document grows.

2 Architecture

2.1 Class Diagram

Figure 2.1 shows the class diagram for the chess software. The diagram is represented by the Unified Modeling Language.

a. **Templates**

Function Template

Function details will be formatted in the following manner:

Purpose: Description of how the function is used in the software.

Inputs: Description of the parameters of the function.

Outputs: Description of the return values of the function.

Calls: Functions called by this function.

Called: Functions this function calls.

Procedure: Description of the procedure of the function.

2.2 chessGame class

The chessGame class shall contain methods responsible for creating various piece objects and placing them on the chess board.

2.2.1 chessGame

Purpose: The constructor of the chessGame class is responsible for the creation and placement of the various pieces upon the chess board. Additionally, the chessGame constructor initializes the values of the turn and turnNumber variables.

Inputs: None

Outputs: None.

Calls:

- Piece.moveLocation(int) method.
- Piece.Piece(int) method.
- Knight.Knight(int) method.
- Rook.Rook(int) method.
- Bishop.Bishop(int) method.
- Queen.Queen(int) method.
- King.King(int) method.
- Pawn.Pawn(int) method.

Called: Board Class as global variable.

Procedure:

- Instantiates the Piece class as an array of size 64.
- Loops through the array to instantiate the various individual piece classes and assigns them to a location in the array. The array is used to place pieces on the Board.
  - Rook, King, Queen, Bishop, Knight, Pawn
- Initializes the turn and turnNumber variables, responsible for:
  1) Changing turns in the game (turn).
  2) Keeping track of the number of turns in the game(turnNumber).

2.2.2 changeTurn

Purpose: This function changes turns in the game.

Inputs: None.

Outputs: None

Calls: None

Called: TilePanel.TilePanel()

Procedure:

- If turn = 0, switch it to one, else turn = 0.

2.2.3 turnUp

Purpose: The purpose of this function is to provide the framework for the user to quit the game before a winner is decided.

Inputs: Description of the parameters of the function.

Outputs: Description of the return values of the function.

Calls: None

Called: TilePanel.TilePanel()

Procedure:

- Increments turnNumber when called.

2.2.4 gameOver

Purpose: The purpose of this function is to end the game with a winner.

Inputs: None

Outputs: None

Calls: None

Called:

- Board.checkForMoves()
- Board.SurrenderPanel()

Procedure:

- Check what player’s turn it is.
- If turn = 1; white wins.
- Else; black wins.

2.3 Board Class

The Board class is called by the strtGUI class (top-level). The Board class shall contain methods to handle events dealing with the user interface, as well as provide the user with game functionality.

2.3.1 Board

Purpose: Creates the various JFrames necessary for the GUI.

Inputs: None.

Outputs: None.

Calls:

- JFrame.setLayout()
- JFrame.setSize()
- JFrame.add()
- JFrame.setDefaultCloseOperation()
- JFrame.setVisible()

Called: strtGUI.main()

Procedure:

- Create new JFrame for the Chess Board.
- Initialize the Layout of the Board.
- Create new JFrame for the Grave.
- Add the Frames to the Game Frame.

2.4 GravePanel Class

The GravePanel class is responsible for providing the framework to add captured pieces to the graveyard.

2.4.1 GravePanel

Purpose: The constructor for the GravePanel class shall create a new Label and add it to the Grave Panel.

Inputs: None

Outputs: None

Calls:

- JPanel.add()
- JLabel.add()

Called:

- Board.Java()

Procedure:

- Gather a captured piece's information.
- Add the piece to the graveyard via JLabel.

2.5 BoardPanel class

This class provides the framework for creating the individual panels on the chess board. The class is also responsible for adding the panels to the board.

2.5.1 BoardPanel

Purpose: The constructor for the BoardPanel shall add the individual tiles to the board.

Inputs: a GravePanel object is passed into the parameters.

Outputs: None

Calls:

- JFrame.add()
- JFrame.setSize()
- Validate()

Called: Board.java

Procedure:

- Loop to populate an ArrayList with TilePanels and assign the panels to a position in the chess board.

2.5.2 drawBoard

Purpose: This function shall draw the chess board GUI.

Inputs: chessGame object responsible for game creation and initialization.

Outputs: None

Calls: TilePanel()

Called: TilePanel.run()

Procedure:

- Shall utilize a loop to construct game tiles and add them to the chess Game.

2.6 TilePanel Class

The TilePanel class shall create the functionality of the GUI as well as provide the system with general decision-making logic when it comes to piece movement. This class gives the user functionality.

2.6.1 TilePanel

Purpose: The constructor for the TilePanel class is responsible for providing the GUI with functionality as well as the user with ability to play the game.

Inputs: BoardPanel Object, GravePanel Object, and Integer tileID

Outputs: none

Calls:

- JPanel.setPreferredSize()
- GridBagLayout()
- assignTileColor()
- assignTileSprite()
- Piece.validMove()
- GravePanel.graveYardAdd()
- ChessGame.changeTurn()
- ChessGame.turnUp()

Called: BoardPanel.java

Procedure:

- Check if player selected a tile with a piece on it.
- Check if player selected a piece on the team who is taking their turn.
- If the player has selected a piece, call highlightTile() to highlight available path.
- If the path is valid, the system shall allow the piece to move or capture an enemy piece on the selected tile.
- Update piece positions, graveyard, the board GUI, and switch turns.

2.6.2 drawTile

Purpose: The purpose of this function is to provide the structure to create tiles.

Inputs: chessGame object

Outputs: None

Calls:

- removeAll()
- assignTileColor()
- assignTileSprite()
- validate()
- repaint()

Called: BoardPanel.drawBoard

Procedure:

- Assign color and necessary board pieces to specific tiles when called.

2.6.3 assignTileSprite

Purpose: The purpose of this function is place a piece on a specific tile.

Inputs: None

Outputs: None

Calls:

- JLabel.setText()
- Validate()

Called:

- drawTile()
- TilePanel.TilePanel()

Procedure:

- Create JLabel and set the text of the JLabel to the name of the piece.
- Add new Label with the piece name to the tile.

2.6.4 assignTileColor

Purpose: The purpose of this function is to assign different tiles colors to represent the accepted chess board.

Inputs: None

Outputs: None

Calls:

- Swing.setBackground

Called:

- drawTile()
- TilePanel.TilePanel()

Procedure:

- Check if the result of the tile number divided by 8 is even or odd
- If even, set the background color white, if odd set the background color black.

2.6.5 highlightTile

Purpose: The purpose of this function is to highlight the tiles that are a part of a valid move to the selected piece.

Inputs: Int selected; Describing the location of the tile selected

Outputs: None

Calls:

- Piece.validMove()

Called:

- TilePanel.TilePanel()

Procedure:

- If a tile is in the valid path of the selected piece, highlighted = true.

2.6.6 availableMove()

Purpose: The purpose of this function is to check if there are pieces in the path of the current move.

Inputs:

- int start
- int end

Outputs: Boolean

Calls:

- chessGame.Piece[].getImageName()
- chessGame.Piece[].getAlly()
- chessGame.Piece[].getPieceType()

Called:

- TilePanel.mousePressed()

Procedure:

- Check piece type.
- If it’s a Pawn, check if there is a piece in the valid path.
- If king or knight, return true.
- If neither, set bounds for the board so pieces cannot move off the board horizontally.
- Set bounds for the board so pieces cannot move off the board vertically.
- Set bound for the board so pieces cannot move off the board diagonally.

2.6.7 checkForCheck()

Purpose: The purpose of this function is to return the team the is in check

Inputs: None

Outputs: Int

Calls:

- chessGame.Piece[].getImageName()
- chessGame.Piece[].getAlly()
- availableMove()
- chessGame.Piece[].validMove()

Called:

- TilePanel.mousePressed()

Procedure:

- Set boolean to signal whether black is found to false
- Traverse until the black king is found, save the location.
- Repeat process to find the white king.
- If the player makes an available and valid move to the black king, the function returns a 1 to signal black is in check.
- Repeat process for a white King.

2.6.8 checkForMoves

Purpose: The purpose of this function is to highlight the tiles that are a part of a valid move to the selected piece.

Inputs: Int selected; Describing the location of the tile selected

Outputs: None

Calls:

- chessGame.Piece.validMove()
- availableMove()
- chessGame.Piece.getAlly()
- chessGame.Piece.validMove()
- Board.BoardPanel.drawBoard()
- chessGame.gameOver()

Called:

- TilePanel.run()

Procedure:

- Checks the current player’s piece and find available moves.
- If the number of available moves = 0, the gameOver() function is called to determine a winner.

2.7 Bishop Class

This class provides the framework to give the bishop movement across the board.

2.7.1 validMove

Purpose: The purpose of this function is to create the bishop's ability to move generally by validating the move made by user.

Inputs:

- int start: The starting position of where player is at.
- int end: The ending position of where the player wants to go to.

Outputs: bool valid: The Boolean that tells the process whether the move is valid or not.

Calls: None.

Called: None.

Procedure: if the move made is diagonal, and within the boundaries then set valid = true.

2.8 King Class

This class provides the framework to give the king movement across the board.

2.8.1 validMove

Purpose: The purpose of this function is to create the king's ability to move generally by validating the move made by user.

Inputs:

- int start: The starting position of where player is at.
- int end: The ending position of where the player wants to go to.

Outputs: bool valid: The Boolean that tells the process whether the move is valid or not.

Calls: None.

Called: None.

Procedure: if the move made is adjacent to starting position, then set valid = true.

2.9 Knight Class

This class provides the framework to give the knight movement across the board.

2.9.1 validMove

Purpose: The purpose of this function is to create the knight's ability to move generally by validating the move made by user.

Inputs:

- int start: The starting position of where player is at.
- int end: The ending position of where the player wants to go to.

Outputs: bool valid: The Boolean that tells the process whether the move is valid or not.

Calls: None.

Called: None.

Procedure: if the move made lands in the difference between start and end position where the move made is L shaped and within boundaries, then set valid = true.

2.10 Pawn Class

This class provides the framework to give the pawn movement across the board.

2.10.1 validMove

Purpose: The purpose of this function is to create the pawn's ability to move generally by validating the move made by user.

Inputs:

- int start: The starting position of where player is at.
- int end: The ending position of where the player wants to go to.

Outputs: bool valid: The Boolean that tells the process whether the move is valid or not.

Calls: None.

Called: None.

Procedure: If the user wants to go up one space, or two spaces if it has yet to move that pawn, then set valid = true.

2.11 Queen Class

This class provides the framework to give the queen movement across the board.

2.11.1 validMove

Purpose: The purpose of this function is to create the queen's ability to move generally by validating the move made by user.

Inputs:

- <u>int start:</u> The starting position of where player is at.
- <u>int end</u>: The ending position of where the player wants to go to.

Outputs: <u>bool valid:</u> The Boolean that tells the process whether the move is valid or not.

Calls: None.

Called: None.

Procedure: This checks the rules for diagonal, or straight moves in any direction, and if So, then valid = true.

2.12 Rook Class

This class provides the framework to give the Rook movement across the board.

2.12.1 validMove

Purpose: The purpose of this function is to create the rook's ability to move generally by validating the move made by user.

Inputs:

- int start: The starting position of where player is at.
- int end: The ending position of where the player wants to go to.

Outputs: bool valid: The Boolean that tells the process whether the move is valid or not.

Calls: None.

Called: None.

Procedure: Checks horizontal and vertical moves, and that are within boundaries, and if so, then return valid = true.

2.13 Piece Class

This superclass provides the framework to give to all the pieces that will extend it.

Attributes:

- Private String imageName;
- Private String Name;
- Private int aliance;
- Private int location;
- Public Boolean hasMoved;

2.13.1 moveLocation

Purpose: The purpose of this function is to create the pawn's ability to move generally by validating the move made by user.

Inputs:

- int start: The starting position of where player is at.
- int end: The ending position of where the player wants to go to.

Outputs: bool valid: The Boolean that tells the process whether the move is valid or not.

Calls: None.

Called: None.

Procedure Sets valid to true if user wants to move up one space, or 2 if it has not moved.

2.13.2 getLocation

Purpose: The purpose of this function is to create the pawn's ability to move generally by validating the move made by user.

Inputs:

- int start: The starting position of where player is at.
- int end: The ending position of where the player wants to go to.

Outputs: bool valid: The Boolean that tells the process whether the move is valid or not.

Calls: None.

Called: None.

Procedure: Sets valid to true if user wants to move up one space, or 2 if it has not moved.

2.13.3 getName

Purpose: The purpose of this getter function is to return the pieces name.

Inputs: None.

Outputs: String name: The String returned is the name of the piece.

Calls: None.

Called: None.

Procedure: Simple getter for the name of the piece.

2.13.4 setName

Purpose: The purpose of this setter function is to set the name of each piece to id them.

Inputs: int nm: Name of the piece. (rook, pawn, queen, etc.)

Outputs: None.

Calls: None.

Called: None.

Procedure: Simple setter function that sets the piece's name.

2.13.5 setImageName

Purpose: The purpose of this setter function is to set image name for sprite use.

Inputs: int in: Name of the image of the piece (.png)

Outputs: None.

Calls: None.

Called: None.

Procedure: Simple setter function that sets the image's name.

2.13.6 setAlly

Purpose: The purpose of this setter function is to set the alliance of the piece.

Inputs: int ally: The number of the alliance you want it to be in.

Outputs: None.

Calls: None.

Called: None.

Procedure: Simple setter function that sets the alliance of the piece.

2.13.7 getAlly

Purpose: The purpose of this getter function is to return the pieces alliance (color).

Inputs: None.

Outputs: int alliance: The integer returned tells what alliance the piece is in.

Calls: None**.**

Called: None.

Procedure: Simple getter for the alliance.

2.13.8 getImageName

Purpose: The purpose of this getter function is get the image name.

Inputs: None.

Outputs: String imageName: Returns this class' string attribute

Called: None.

Procedure: Simple getter for the image name.

2.13.9 validMove

Purpose: The purpose of this function is to create the piece's ability to move generally by validating the move made by user.

Inputs:

- int start: The starting position of where player is at.
- int end: The ending position of where the player wants to go to.

Outputs: bool valid: The Boolean that tells the process whether the move is valid or not.

Called: None.

Procedure: Depending on what the criteria for the piece is, it will check and return true or False.