New Bulgarian University

NETB375 Практика по програмиране

Zdravko Donev F74219, Semester 5

Sofia, 2017

Contents

Task description and sub problem covered in this documentation

Task Variant 1 – AI chess game

Using the programming language C++ and programming framework **Qt** implement a computer program the famous chess game. The game must give two modes of play: human against human (on the same computer), and human against CPU. The program must have the following features:

- fully functioning GUI that represents the chess board and all pieces;
- white piece can be moved if it is a white player's move, and the same for black pieces;
- each piece is allowed to perform only valid moves according to game rules;
- AI complexity is left to project developers, and it can be the simplest possible -- a random move.

This part of the documentation describes the parts of the chess game project that are related to the User Interface.

Overview of the project

The project is divided into several parts which are implemented from different persons and are merged into a single final project. For the purpose of source control, we are using GitHub. My part of the project is the Graphical User Interface.

The project is implemented using Qt Creator 4.0.2 IDE and follows international and local (within the team) code writing and formatting conventions and rules.

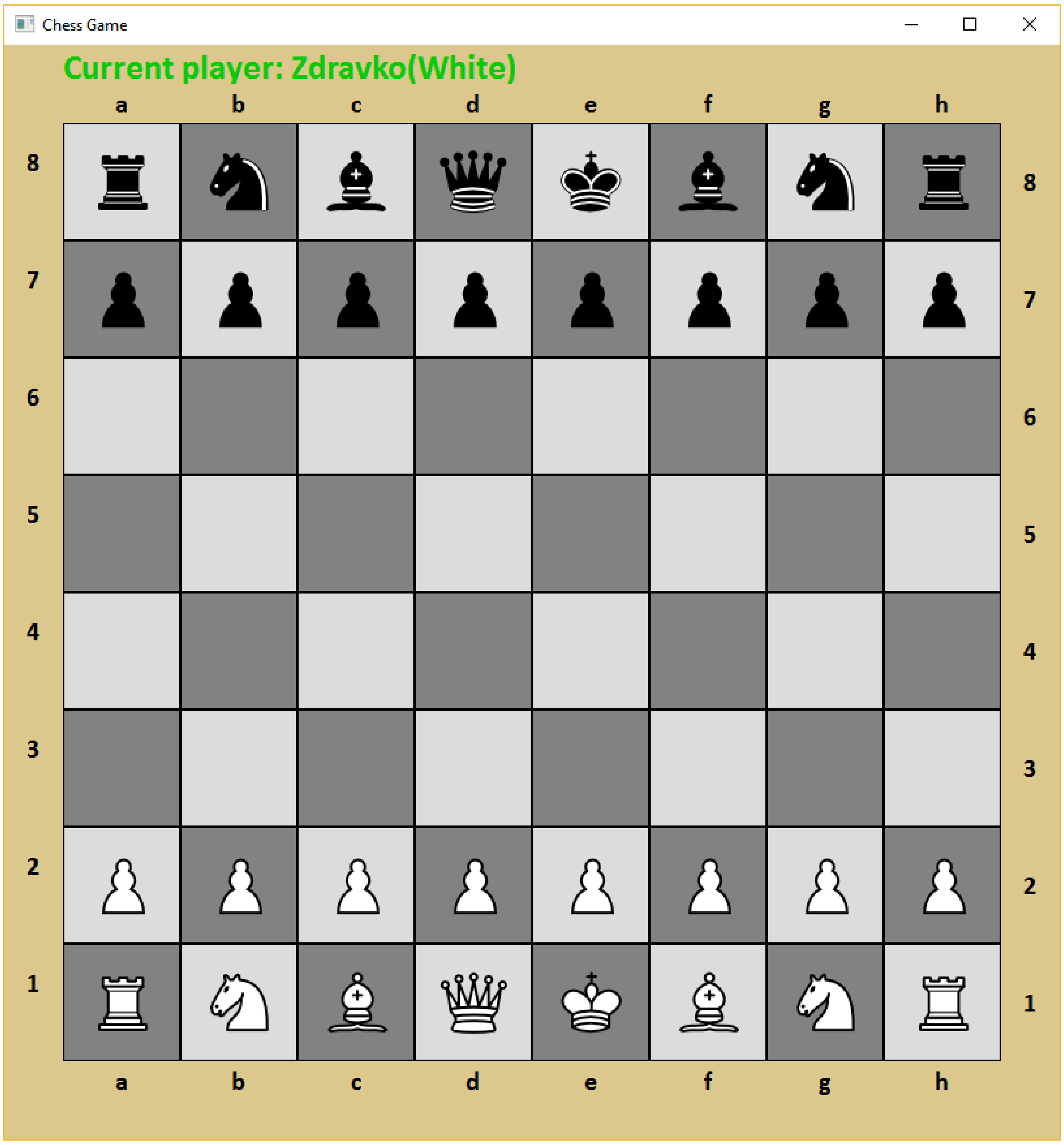
The User Interface is designed to be responsive and to have elements with contrasting colors, so it would be easy and pleasantly to use. The size of all windows that are shown on the screen, during the execution of the program, is calculated at runtime. This approach aims the application to look the same on computers with different screen resolutions.

All images of pieces are downloaded from Wikipedia.

Design and implementation of GUI

The GUI part consists of the following classes:

- ***MainWindow***



This class is the default created class by *Qt Creator* when creating a new project. Its purpose is to draw the chess board and to communicate with the *Controller* class. The board is actually two 2D arrays of *CellButtons*, where the first one(*BackgroundBoard*) is showing the static chess-colors

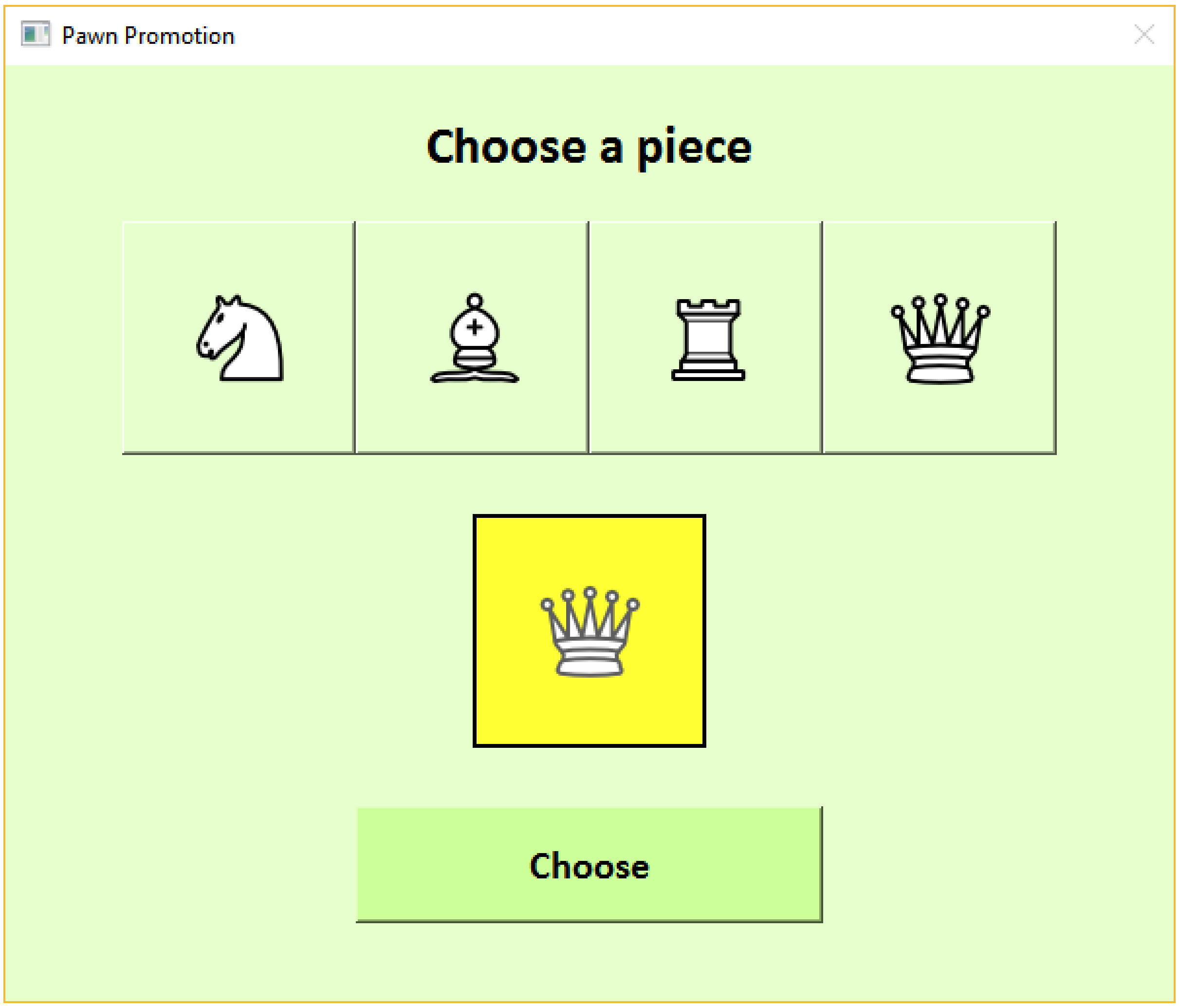
of the board and the second one(*Board*) is showing the pieces. If there is no piece to show, the cell in the *Board* is transparent. There exist labels that show the row(8-1) and the column(a-h) on each of the four sides of the board.
The pieces have hover-events on which event, the background of the piece is changed, which shows that the pointer of the mouse is over the piece. When a piece is clicked, both the piece and all it's possible cells to which it can move are highlighted(their backgrounds are changed to different color).
On the left-top corner of the window, the name of the current player is shown.
On the right-top corner of the window, there is a label that is visible if the current player is in check.

The class consists of the following functions:

- ***createBoard*** - Creates the board with buttons that are used to contain pieces.
- ***createBackgroundBoard*** - Creates the board with buttons that are in the background i.e. that have no pieces on them.
- ***createLabels*** - Initializes the labels around the board that show the "name" of the cell. Also initializes the label for the "check" and "checkmate" status
- ***initBackgroundBoardColor*** - Sets the color of the background buttons to the traditional chess-like color. This way, the method removes any highlighted fields.
- ***isSelected*** - Checks if the cell at the provided coordinates is highlighted(selected).

- ***drawState*** - Draws the current state: all pieces currently on the board, the status of the game, and which player is on turn. Checks wether the game is in Checkmate state or it has a pawn promotion.
- ***setWindowSize*** - Calculates the size of the labels, board, cells on the board and the window.
- ***highlightCells*** - Highlights the selected cells of possible moves.
- ***drawCurrentPlayer*** - Shows which player is currently on turn.
- ***showGameOver*** - Opens a dialog that alerts for a finished game and shows additional information about the game.
- ***handleBackgroundAndHighlightedFields*** - When background field or highlighted field is clicked, this method is called. One method is used, since the actions taken when those situations occur are taken are the same.
- ***handlePieceClick*** - This method is called when a piece is clicked.

- ***ChoosePieceWindow***



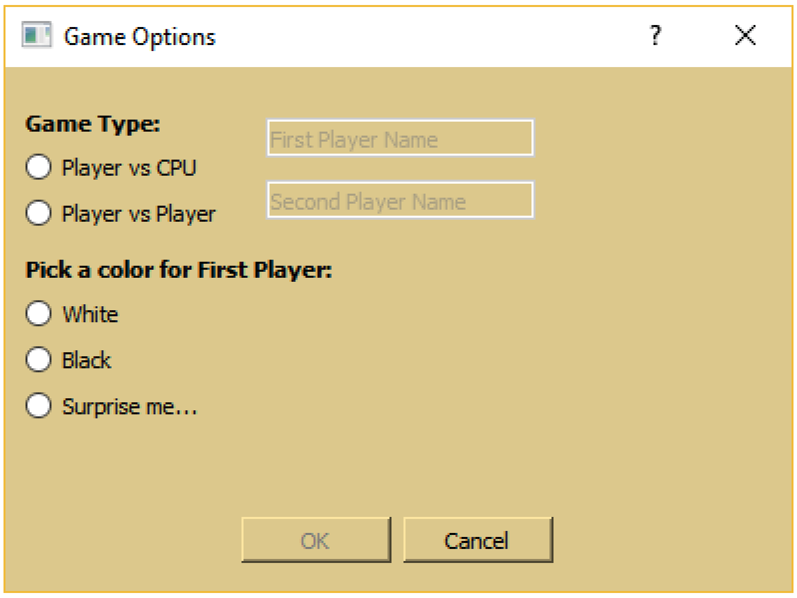
This class is used to show a dialog window that appears when the user wants to choose a piece that will replace the pawn which has reached the end of the grid.

The class consists of the following functions:

- ***getSelectedPieceType*** - Returns the selected piece type for pawn promotion.
- ***setWindowSize*** - Initializes the size of the dialog.
- ***initCaption*** - Initializes a label with caption of the dialog.
- ***initPieces*** - Initializes the pieces on the fields.

- ❖ ***initChooseButton*** - Button that submits the choose action of the user.
- ❖ ***setPiece*** - Sets the given piece image on the provided index of a *CellButton*.
- ❖ ***handlePieceClick*** - This method highlights the piece being clicked.
- ❖ ***handleChooseClick*** - When the user clicks the submit button, the dialog closes, and the game continues.

- ***GameOptionsWindow***



This class is used to show a dialog at the start of the application. The dialog helps to setup the game, like choosing the name of the players and choosing the game type.

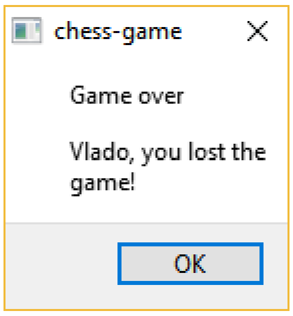
The class consists of the following functions:

- ❖ ***getSelectedGameType*** - Returns the selected game type.
- ❖ ***getSelectedColor*** - Returns the selected color of the first player.
- ❖ ***getFirstPlayerName*** - Returns the name of the first player.
- ❖ ***getSecondPlayerName*** - Returns the name of the second player.
- ❖ ***on_radioButton_gameType_clicked*** - Event trigered when a button in the game type radio group is clicked. This method

enables/disables appropriate fields and stores the current selected game type.

- ❖ ***on_radioButton_color_clicked*** - Event trigered when a button in the color radio group is clicked. This method enables/disables appropriate fields and stores the current selected color of first player.
- ❖ ***on_lineEdit_firstPlayer_textChanged*** - Fires event to check if the field is filled.
- ❖ ***on_lineEdit_secondPlayer_textChanged*** - Fires event to check if the field is filled.

- ***GameOverWindow***



chess-game

Game over

Vlado, you lost the game!

- ***UIHelperFunc***

This class is used as a namepace to store helper functions that the UI can use in order to visualize appropriate and consistent data.

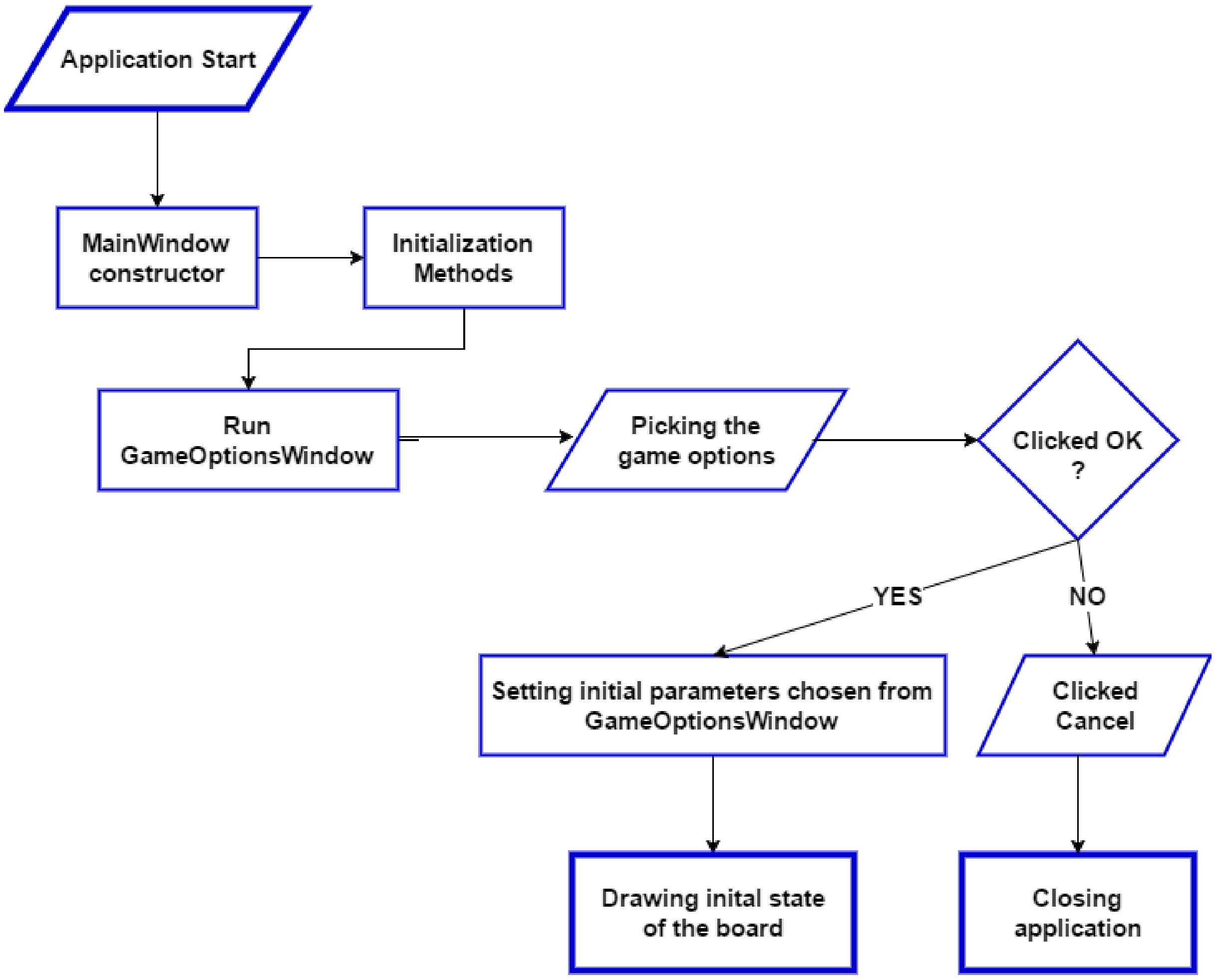
The class consists of the following functions:

- ❖ ***getPieceFileName*** – Given an object of a subclass of a piece, this method returns the filename that should be looked for in the directory with images.
- ❖ ***getFormBackgroundStyleSheet*** – Returns a stylesheet format of the background of the windows that are visualized.
- ❖ ***getBackgroundStyleSheet*** – Returns a stylesheet format of a field in the board.

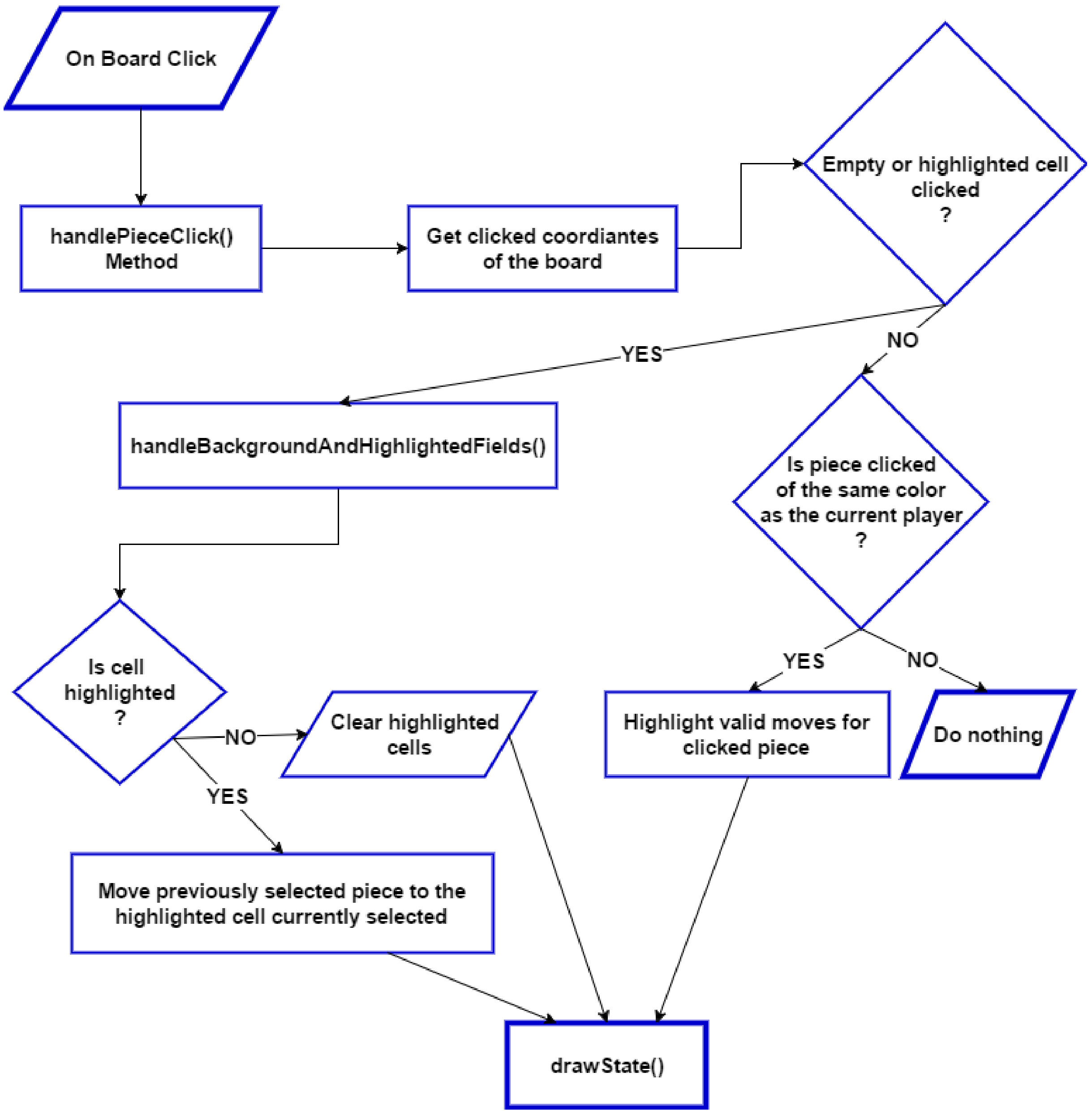
❖ ***getBackgroundAndHoverStyleSheet*** - Returns a stylesheet format of a field in the board which has a hover event.

Flow diagrams

Application Start Diagram

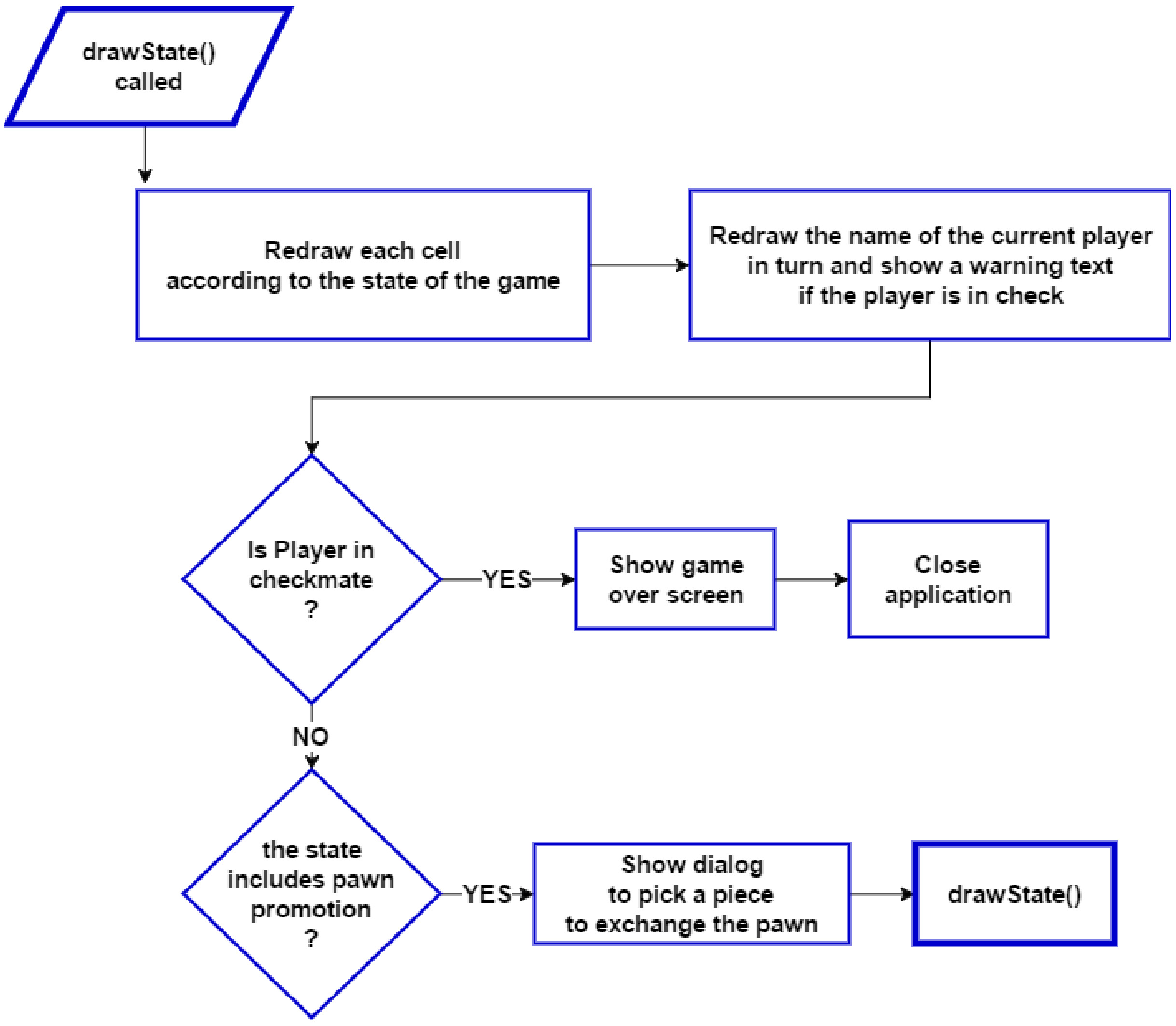


On Board Selected Diagram



On Board Click
handlePieceClick() Method
Get clicked coordiantes of the board
Empty or highlighted cell clicked ?
YES
NO
handleBackgroundAndHighlightedFields()
Is piece clicked of the same color as the current player ?
Is cell highlighted ?
NO
YES
Clear highlighted cells
YES
NO
Highlight valid moves for clicked piece
Do nothing
Move previously selected piece to the highlighted cell currently selected
drawState()

drawState Method Diagram


drawState() called
Redraw each cell according to the state of the game
Redraw the name of the current player in turn and show a warning text if the player is in check
Is Player in checkmate ?
YES
Show game over screen
Close application
NO
the state includes pawn promotion ?
YES
Show dialog to pick a piece to exchange the pawn
drawState()