**Problem Statement**

**To implement “Checkers Game”**

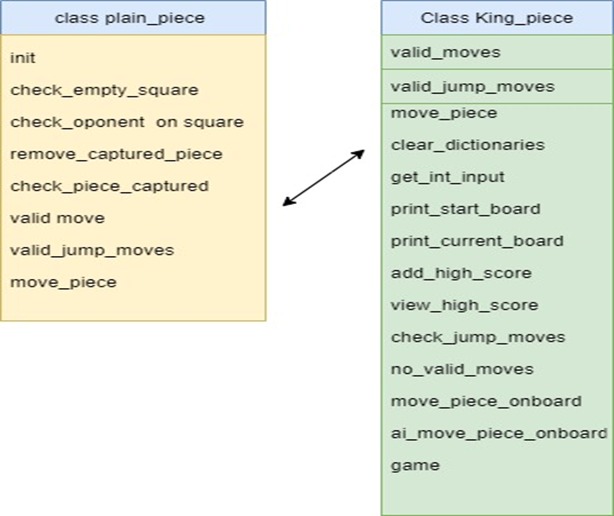
**TECHNICAL SPECIFICATIONS**

* **Programming language -**Python >v 3.1
* **Framework –** Tkinter, Pyqt
* **Modules/Library Used-**random, time

**Class Diagram**

Diagram

Description automatically generated



**class PlainPiece:’**

'''Class containing the plain pieces and its associated methods.'''

* All dictionaries has 1 and 2 as keys which represents player 1 and player 2
* Then We have another list If a piece can make another jump move, its coordinates will be appended to this list
* Dictionary to store the coordinates of each piece that can make a jump move. Used only for the AI-moves
* The next list Stores the coordinates of each piece that can make a move. Used only for the AI-moves
* A list Contains the current positions of each regular piece
* A list Contains the current positions of each king piece
* Another list stores The amount of squares on one side of the quadratic board

**Functions**

**1) Init-Use to make new player**

It will take 3 parameters X-Coordinate,Y-coordinate and player

**2) Check\_if\_empty\_square**

'''Checks if a square diagonally from the piece is empty.

The x-coordinate of the square you want to check relative to the piece's position

The y-coordinate of the square you want to check relative to the piece's position

return: Boolean variable. True if the square is empty. False if the square contains another piece

**3) check\_if\_opponent\_on\_square**

Checks if the opponents piece is on a square and adds it to the "valid jump moves"-list if that is the case.

The x-coordinate of the square you want to check relative to the piece's position

The y-coordinate of the square you want to check relative to the piece's position

return: (nothing)

4) **remove\_captured\_piece- '''Removes the captured piece from the dictionary.**

The x-coordinate of the captured piece relative to the new position of the capturing piece

The y-coordinate of the captured piece relative to the new position of the capturing piece

return (nothing)

**5) check\_which\_piece\_captured**

Checks which piece that has been captured.

:param new\_x: The new x-coordinate of the piece

:param new\_y: The new y-coordinate of the piece

:return: (nothing)

**6)** **valid\_moves**

Checks all the valid moves the piece can make and adds them to a list.

:return: The list containing all valid moves

7) **valid\_jump\_moves**

'''Checks all the valid jump moves the piece can make and adds them to a list

:return: A list containing all the possible jump moves

**Class KingPiece**

'''Inherits from the PlainPiece class and overrides some methods Overrides the valid\_moves- and the valid\_jump\_moves-methods to allow pieces to move backwards.

Also overrides the move\_piece-method to instead remove and add positions to the king\_pieces\_dictionary'''

**1) clear\_dictionaries**

'''Clears all the dictionaries. :return: (nothing) '''

2)**get\_int\_input**

'''Used to get an int form the user, asks again if input is not convertible to an integer.

:param prompt\_string: A string explaining what to input

:return: The int that was asked for

'''

3)**print\_start\_board**

Prints the starting board and adds every piece to the pieces dictionary.

:param empty\_rows: Used to calculate which rows that should be empty

:return:

# Uses the coordinates in the dictionaries to print the current board

**4) print\_current\_board**

Prints the current board using the dictionaries with all the positions.

**5) add\_high\_score**

adds information to the high score file about the winning player.

:param name: The name of the winning player

:param board\_size: The size of the board

:param start: The time at which the game started

:param end: The time at which the game ended

:return:

**6) view\_high\_score**

'Reads from the high score text file and prints the current high score list With the help of File module :return: (nothing)

**7)check\_jump\_moves**

Used to check if a player can take a piece.

# Stores all available jump moves that each player can make in lists

:param player: Which player it is

:return: Boolean variable. True if a jump move is possible and False if a jump move is not possible

**8) no\_valid\_moves**

'''Used to check if a player can make a valid move

:param player: Which player it is

:return: Boolean variable. True if no valid move is possible, False if there is a possible move

'''

**9)move\_piece\_on\_board**

Asks the player which move they want to make and moves the piece.

Asks for the coordinates the piece's current position and the coordinates where the player wants to move it.

If the players own piece is not selected or if the move is invalid, it will ask again.

:param player: The player whose turn it is

:return: (nothing)

**10) AI\_move\_piece\_on\_board**

Makes a random move for a player. Used to move a piece for the AI.

:param player: The player whose turn it

**Deployment Diagram**

**Diagram

Description automatically generated**

**Sequence Diagram**

**A picture containing application

Description automatically generated**

Diagram

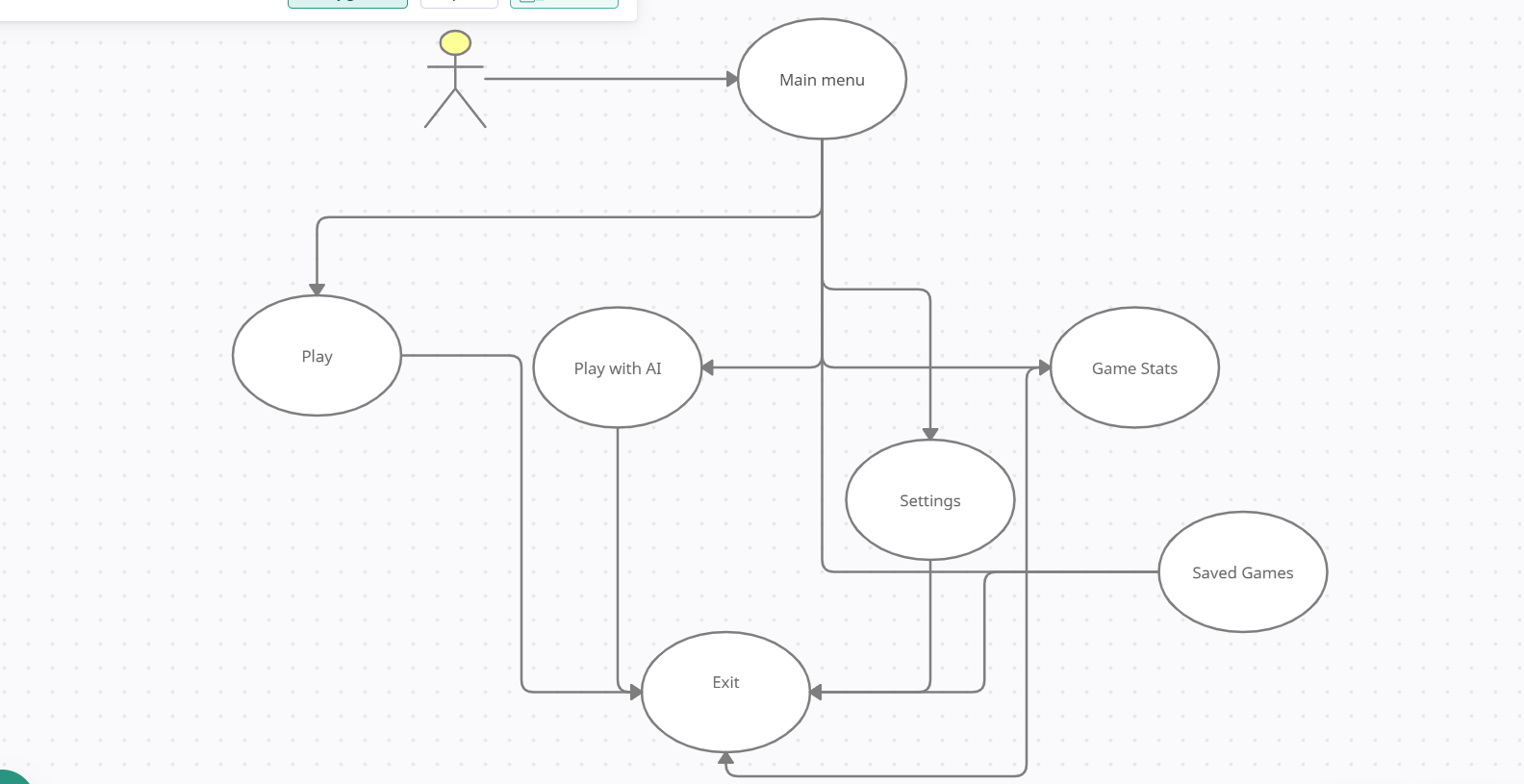
Description automatically generated

**Chart

Description automatically generated**

**UML Diagrams**

* **Menu Activity**

****

* **Saved Games**

**Diagram

Description automatically generated**

* **Game Stats**

**Diagram

Description automatically generated**

* **Play**

**Diagram

Description automatically generated**

* **Play with AI**

**Diagram

Description automatically generated**