Checkers Game Data Model

Siyang Liu

1. An abstract base class for the player, it contains the following:
   1) Some basic information of the player, like name, id and etc.
   2) A n*n 2-D array to contains all the checker points (x, y coordinates) information player. If position (i, j) has a checker, then set checkers[i][j] to true, otherwise false.
   3) A variable checkerCount to record how many checkers the player has left.
   4) A getPossibleMoves method, passing through the points the player wants to move from, and the reference of another player, return all the possible positions that the current player can move from current point to.
   5) An abstract method that determine if the player can automatically move. Typically, this occur when the subclass of the Player contains the move AI code.
   6) If the subclass of the player can auto move, it must implement the autoMove method. It passes the reference to another player, then move by the AI code here, and update the check positions for its own and another player
   7) An abstract method move when it cannot auto move. Typically, it is human player. It contains the old position the player wants to move, and the new position the player wants to move to, and a reference for another player, in order to get and update the check positions for another player. If the move is illegal, it will throw an exception.
   8) A method getCurrentState, passing the reference to another player, return state enum of the current player. The enum contains IN_GAME, WIN, LOSE and DRAW.

```
public abstract class Player {
     protected String id;
     protected String name;
     protected boolean[][] checkers;
     protected int checkerCount;
     protected List<Point> getPossibleMoves(Point currentPoint, Player
 anotherPlayer);
     public abstract boolean canAutoMove();
     public abstract void autoMove(Player anotherPlayer);
     public abstract void move(Point oldPosition, Point newPosition, Player
anotherPlayer) throws IllegalMoveException;
     public State getCurrentState(Player anotherPlayer);
}
```

2. Two derived classes HumanPlayer and AIPlayer from Player class.
   1) HumanPlayer returns false in the canAutoMove method, and implements move method.
   2) AIPlayer return true in the canAutoMove method, and implements autoMove method.
3. A CheckerGame (Controller) class that contains the flowing information:
   1) Two references two all the 2 players, it can either be human player and computer player.
   2) A reference to the player that play the current turn. Initially, if player1 players first, then set it to player1. And it is same way for player2.
   3) A reference to the CheckBoard UI, as we did in Assignment 1, for the App to update the UI.
   4) A method to display the win information, if one player wins.

5) A method to display illegal move information, when the move method in Player class throws an exception of an illegal move.
6) A move method, that moves the current players. It may need the input from the user when it there are human players. It can use the autoMove method in the class, if it is AI player.

```
public class CheckerGame {
     private Player player1;
     private Player player2;
     private Player currentTurnPlayer;
     private CheckBoard board;
     private void displayWinInfomation();
     private void diaplayIllegalMoveInformation();
     private void move();
 }
```