1. Step 1 – underline all the nouns

Checker’s Game Requirements

In this game, two users play Checkers using the same computer and input their moves at each round via

the terminal. When the game starts, the program outputs—on the terminal—the board with the pieces in

their initial position. The board should be displayed as shown in the following example:

a b c d e f g h

+-------------------------------------------------+

8 | [ ] [R\_P] [ ] [R\_P] [ ] [R\_P] [ ] [R\_P] | 8

7 | [R\_P] [ ] [R\_P] [ ] [R\_P] [ ] [R\_P] [ ] | 7

6 | [ ] [R\_P] [ ] [R\_P] [ ] [R\_P] [ ] [R\_P] | 6

5 | [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] | 5

4 | [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] | 4

3 | [W\_P] [ ] [W\_P] [ ] [W\_P] [ ] [W\_P] [ ] | 3

2 | [ ] [W\_P] [ ] [W\_P] [ ] [W\_P] [ ] [W\_P] | 2

1 | [W\_P] [ ] [W\_P] [ ] [W\_P] [ ] [W\_P] [ ] | 1

+-------------------------------------------------+

a b c d e f g h

The first character (‘R’ or ‘W’) represents the piece’s color (Red or White) and the second one the piece’s

type (i.e., ‘P’ = Pawn, ‘K’ = King3). Each row of the board is indexed by a number (starting from the

bottom row with ‘1’) and each column is indexed by a character (starting from the left with ‘a’).

The game then asks the first player to enter their move on the terminal. Users declare their moves as

in the following: [current piece position]X[future piece position]; for example by

typing [a3]X[b4] they want to move a piece from position a3 into position b4. The validity of the

move must be checked; if not valid the user has to enter another move until they enter a valid one. After the

player inserted a valid move, the program prints an updated version of the board based on the new positions

of the pieces.

Afterwards, the program asks the next user to move, unless the game is finished, in which case it prints

out who the winner is.

1. Class selection rationale:

Physical objects: computer -> do not need

Conceptual entities:

-game, round -> game

-move

-board

-piece

Interfaces to the system terminal, program -> do not need to model them.

-user, the first player, the next user -> player -> probably we do not need if we don’t need to input name etc of users. ( just to put player 1/2 wins do not need such class)

More attributes than classes:

* Position or Column&raw (probably attribute pf piece)
* Number and character (int, char)
* Validity (probably attribute in move)
* Version ( attribute of board)

Candidate classes: game, move, board, piece

[oop - Object Oriented Design for a Chess game - Stack Overflow](https://stackoverflow.com/questions/4168002/object-oriented-design-for-a-chess-game)