Advanced Programming AssEx *Wesley Scott 2460681S*

Checkers Game Design Specification

Allows 2 players to participate in a multiplayer game of English Draughts via network.

The design specification discusses all necessary classes and their intended structure at the outset of development, as well as some groundwork on how communication and events will be handled.

Server.java:

- Should allow for 2 clients
- Server has its own instance of data model (board essentially), contains the runtime logic, maintains that data is coherent between clients.
- Decides legality of game moves made by clients to avoid sending updates unnecessarily
- Sends data allowing clients to update to current game state and view state after any other client has made an action within the game.

Client.java:

- Main 'controller' class
- Captures user input events via listeners, provide server with user actions to update data model.
- Client instance per user, sends user input to server (when relevant) to update server model data.
- Draws game board for each user, sync'd with server game state.

Data model (model classes):

- Contain data structures and algorithms for use in a server instance, serves both clients in a single game instance
- Should contain all necessary logic for a full playthrough, including win conditions
- Ideally separate from any Swing code

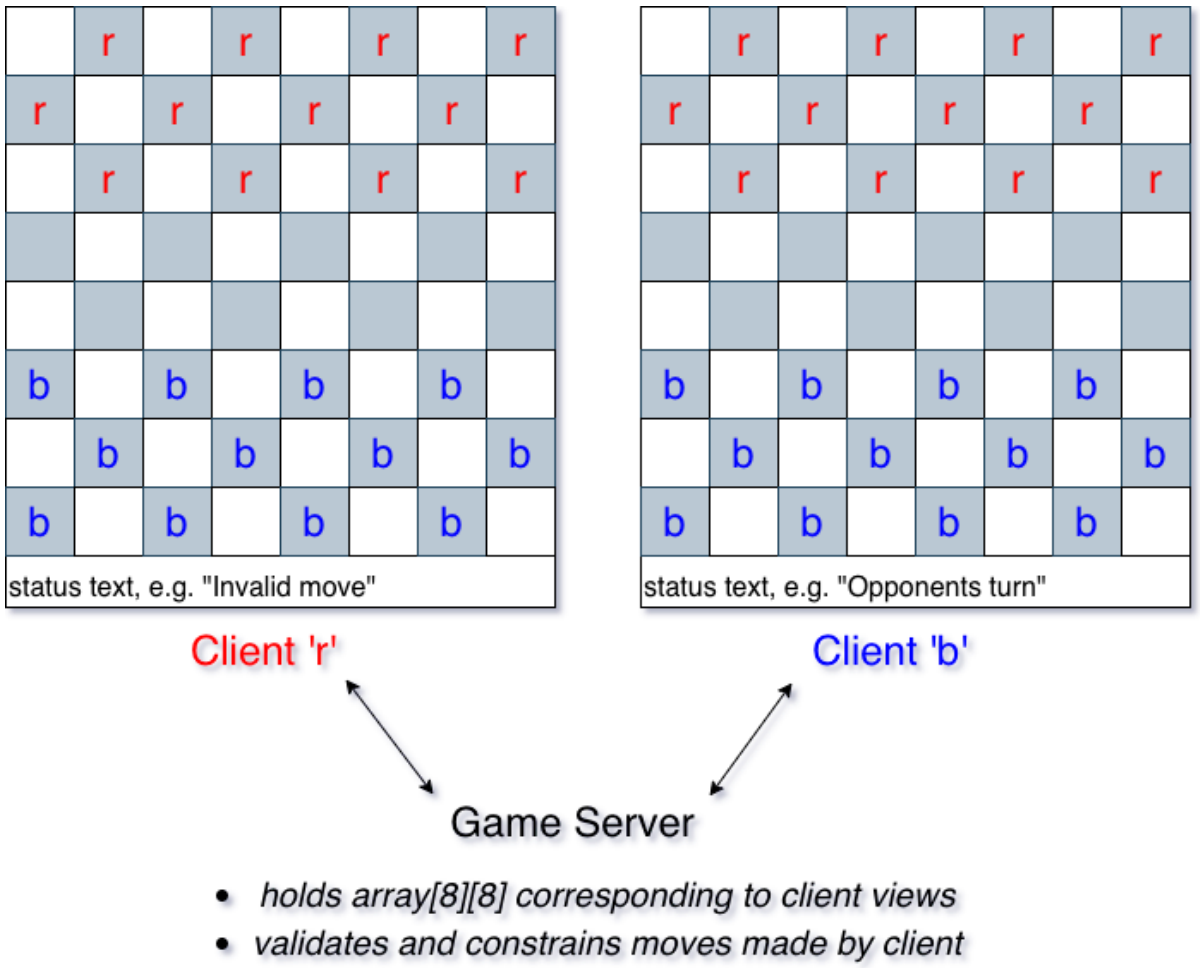
View classes:

- Contain all swing elements, used in conjunction with Client class.

Although class separation is necessary, during trial implementation it was clearer to have a client Class, which contained its relevant methods and sub classes, and do the same with the server, meaning there were no unclear associations.

Encoded messaging sent between the server and client are used in order to control states – a string prefix is checked to match conditions between server and client runtimes, simplifying the implementation of socket protocols.

See below initial design for client GUI:



Although UX is not part of the assignment, the aim of the implementation was to make the game state very clear to clients by using the status text bar as seen in many windows applications. A MouseListener will be used to make gameplay intuitive, similar to the Windows XP checkers game.

In order to run the application, it is recommended to use iTerm, allowing the user of multiple terminal tabs –

in one tab – locate repo root

type “javac Server.java” enter (required only prior to first run)
type “java Server” enter... which begins server runtime.

In 2 other tabs – locate repo root

type “javac Client.java” enter (required only prior to first run)
type “java Client” enter... which begins a single instance of the client runtime.