PROJECT Design Documentation

Team Information

- Team name: D_Team
- Team members
  - Jack Sipley
  - Aaron Schulte
  - Tyler Talarico
  - Alexic Chiang

Executive Summary

This web application is designed with the purpose of allowing users to play online checkers against other users. Users who connect are required to sign in, which simply requires the user to provide a valid username. Once signed in users are presented with a list of other users currently signed in, clicking another user's name will start a game if they are available. Users are presetned with a drag and drop checkers board until they have completed the game and returned to the home page.

Purpose

To allow users to play a game of drag and drop checkers against other users across the internet.

Glossary and Acronyms

> *Provide a table of terms and acronyms.*

Term	Definition
VO	Value Object
Position	A coordinate representation of the spaces on a checker board (row,column)

Requirements

This section describes the features of the application.

> *In this section you do not need to be exhaustive and list every story. Focus on top-level features from the Vision document and maybe Epics and critical Stories.*

Definition of MVP

> *Provide a simple description of the Minimum Viable Product.*

Two players must be able to sign in and play a game of checkers based upon the American rules. The game ends when either one player wins or one of the player resigns.

MVP Features

> *Provide a list of top-level Epics and/or Stories of the MVP.*

Start A Game

As a Player I want to start a game so that I can play checkers with an opponent.

Player Sign-In

As a Player I want to sign-in so that I can play a game of checkers.

Player Sign-Out

As a Player I want to sign-out so that I can stop playing.

King Me

As a Player I want my pieces to be kinged when they reach the end of the board so that my checker can move/jump forward or backwards.

Epic: Movement As a Player I want to move my pieces so that I can play a game of checkers.

Simple Move

As a Player I want to move my pieces to vacant locations so that I can play checkers.

Jump Move

As a Player I want to jump my opponent's pieces to capture them so that I can win.

Multiple Jump

As a Player I want to continue jumping my opponent's pieces if possible so that I can win.

Jump Move

As a Player I want to jump my opponent's pieces to capture them so that I can win.

Epic: End Game As a Player I want to end the game so that I can do other actions.

Victory

As a Player I want the game to end when I have won or lost so that I decide what to do after.

Resignation

As a Player I want to be able to resign at any point during a game so that I can quit or start another.

Roadmap of Enhancements

Provide a list of top-level features in the order you plan to consider them.

Epic: Multi-Game Interface The player should be equipped with an interface which can display multiple games.

Add Game

As the player I want another game displayed when asked to play another game.

Rematch

As a player I want to offer a rematch after a game so that I can play again.

Epic: AI-Logic The AI must be able to complete a game of checkers comparable to a human.

Where To Move

As a player I want the AI to have an idea of where it is in relation to the board and pieces.

Fight Or Flight

As an AI I want to know when to capture opponents' pieces or move someplace else.

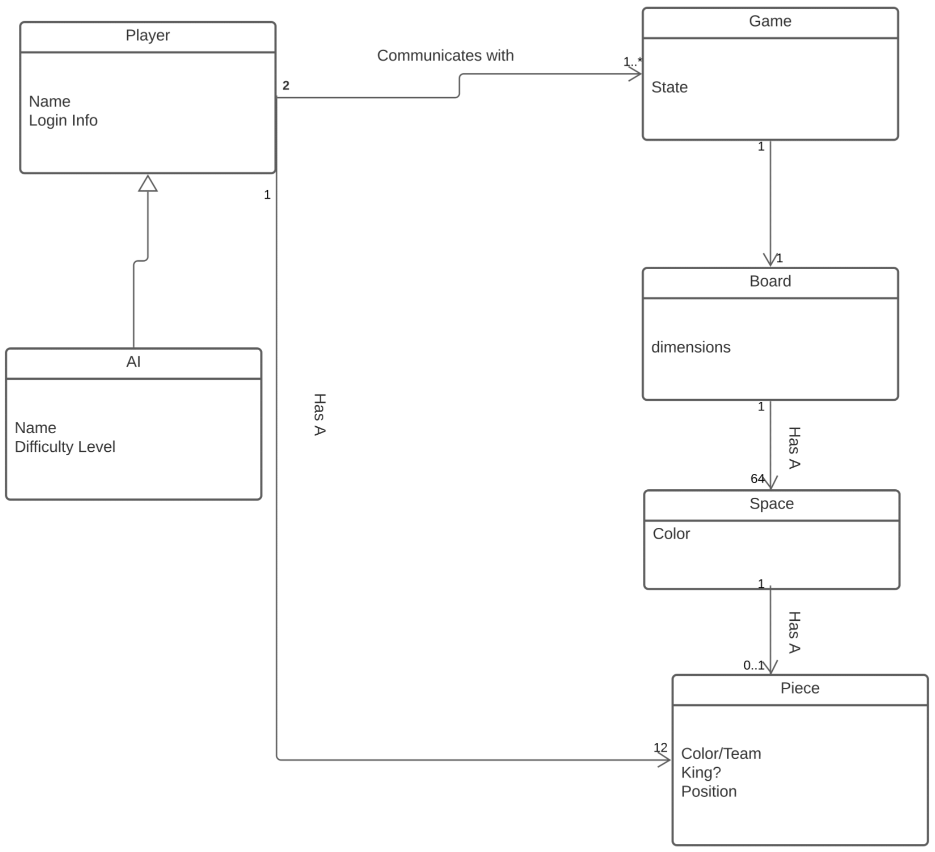

Figure 1: The WebCheckers Domain Model

Application Domain

This section describes the application domain.

Two players play in a game, each game consists of one board with 8x8 spaces. Pieces are set up on this board according to the American rules for checkers.

Architecture and Design

This section describes the application architecture.

Summary

The following Tiers/Layers model shows a high-level view of the webapp's architecture.

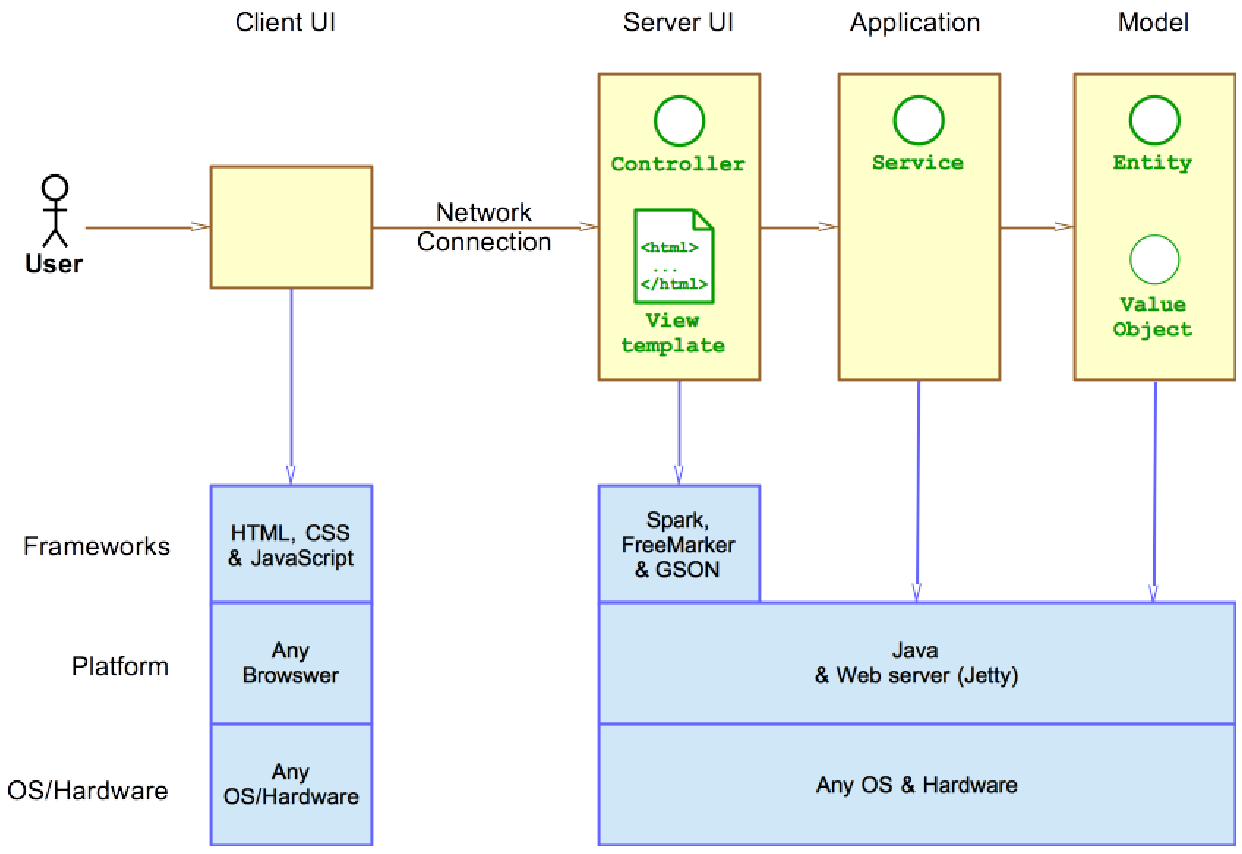

Figure 2: The Tiers & Layers of the Architecture

As a web application, the user interacts with the system using a browser. The client-side of the UI is composed of HTML pages with some minimal CSS for styling the page. There is also some JavaScript that has been provided to the team by the architect.

The server-side tiers include the UI Tier that is composed of UI Controllers and Views. Controllers are built using the Spark framework and View are built using the FreeMarker framework. The Application and Model tiers are built using plain-old Java objects (POJOs).

Details of the components within these tiers are supplied below.

Overview of User Interface

This section describes the web interface flow; this is how the user views and interacts with the WebCheckers application.

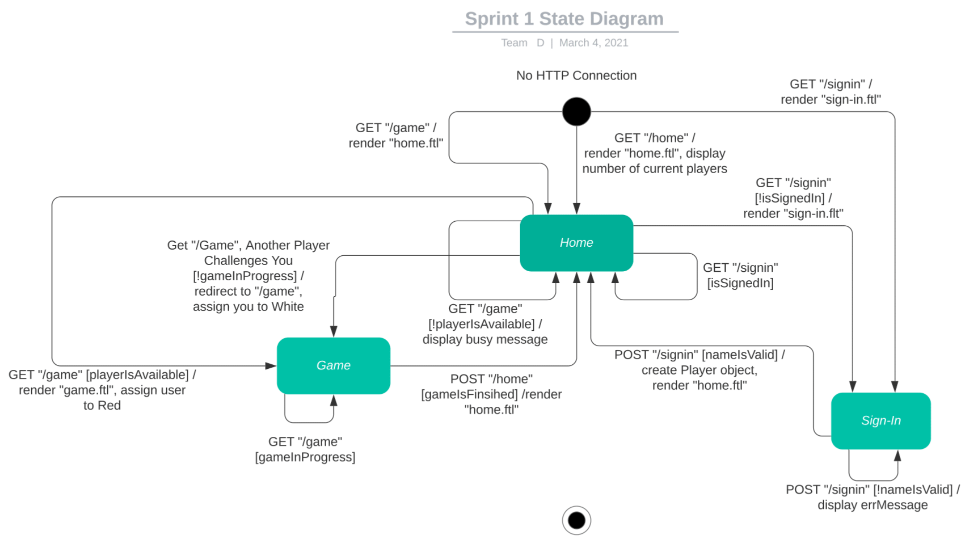
Sprint 1 State Diagram
Team D | March 4, 2021
No HTTP Connection
GET "/signin" / render "sign-in.ftl"
GET "/game" / render "home.ftl"
GET "/home" / render "home.ftl", display number of current players
GET "/signin" [!isSignedIn] / render "sign-in.flt"
Home
GET "/signin" [isSignedIn]
Get "/Game", Another Player Challenges You [!gameInProgress] / redirect to "/game", assign you to White
GET "/game" [!playerIsAvailable] / display busy message
Game
GET "/game" [playerIsAvailable] / render "game.ftl", assign user to Red
POST "/home" [gameIsFinsihed] /render "home.ftl"
POST "/signin" [nameIsValid] / create Player object, render "home.ftl"
Sign-In
GET "/game" [gameInProgress]
POST "/signin" [!nameIsValid] / display errMessage

Figure 3: The WebCheckers Web Interface Statechart

> *Provide a summary of the application's user interface. Describe, from the user's perspective, the flow of the pages in the web application.*

When a user connects they will be directed to the home page where there only option will be to click the "sign-in" button. This will take the user to the sign in page where the user is required to enter a valid username. Once a valid username is entered the user is taken back to the home page and presented with a list of online players. Clicking the name of an online player will take the user to the game page where they will play out their game of checkers until completion. Once completed, the user will be directed back to home.

UI Tier

> *Provide a summary of the Server-side UI tier of your architecture. Describe the types of components in the tier and describe their responsibilities. This should be a narrative description, i.e. it has a flow or "story line" that the reader can follow.*
>
> *At appropriate places as part of this narrative provide one or more static models (UML class structure or object diagrams) with some details such as critical attributes and methods.*

You must also provide any dynamic models, such as statechart and sequence diagrams, as is relevant to a particular aspect of the design that you are describing. For example, in WebCheckers you might create a sequence diagram of the `POST /validateMove` HTTP request processing or you might show a statechart diagram if the Game component uses a state machine to manage the game.

If a dynamic model, such as a statechart describes a feature that is not mostly in this tier and cuts across multiple tiers, you can consider placing the narrative description of that feature in a separate section for describing significant features. Place this after you describe the design of the three tiers.

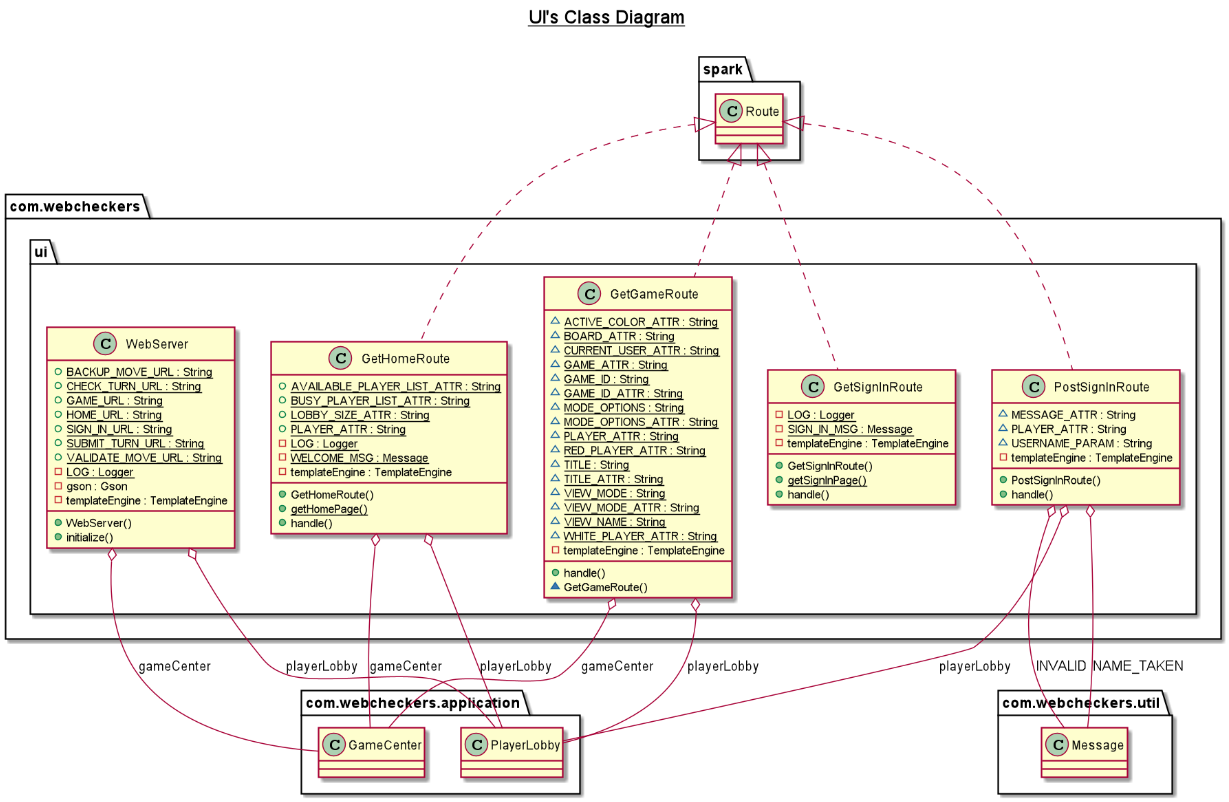

Figure 4: The UI's class diagram'

Application Tier

Provide a summary of the Application tier of your architecture. This section will follow the same instructions that are given for the UI Tier above.

Model Tier

Provide a summary of the Application tier of your architecture. This section will follow the same instructions that are given for the UI Tier above.

Design Improvements

Discuss design improvements that you would make if the project were to continue. These improvement should be based on your direct analysis of where there are problems in the code base which could be addressed with design changes, and describe those suggested design improvements.

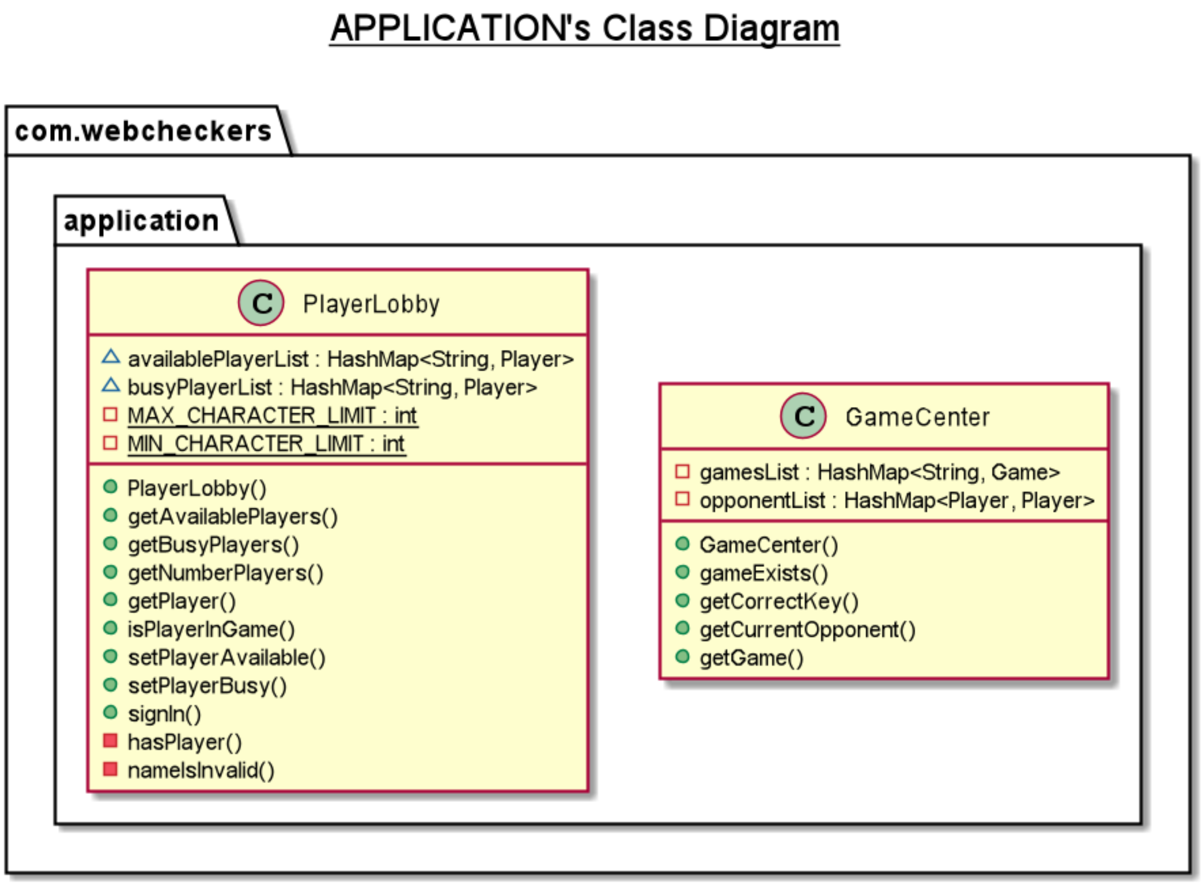

Figure 5: The Application's class diagram'

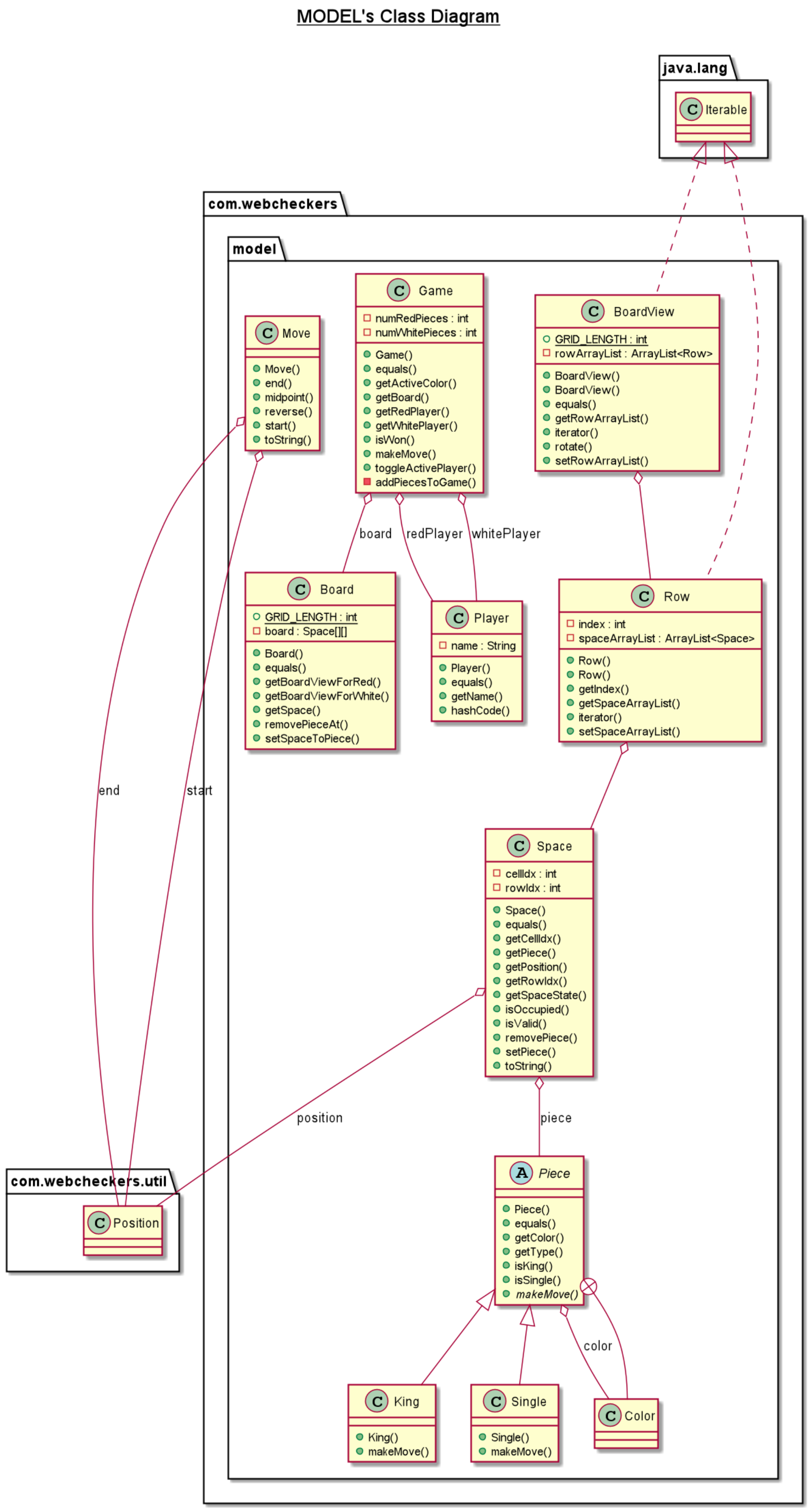

Figure 6: The Model's class diagram'

After completion of the Code metrics exercise, you will also discuss the resutling metric measurements. Indicate the hot spots the metrics identified in your code base, and your suggested design improvements to address those hot spots.

Testing

This section will provide information about the testing performed and the results of the testing.

Acceptance Testing

Report on the number of user stories that have passed all their acceptance criteria tests, the number that have some acceptance criteria tests failing, and the number of user stories that have not had any testing yet. Highlight the issues found during acceptance testing and if there are any concerns.

Currently the only user story to have completed acceptance testing is “Sign-In”

Unit Testing and Code Coverage

Discuss your unit testing strategy. Report on the code coverage achieved from unit testing of the code base. Discuss the team’s coverage targets, why you selected those values, and how well your code coverage met your targets. If there are any anomalies, discuss those.