PROJECT Design Documentation

Team Information

- Team name: Resistance
- Team members
  - Justin Lam
  - Alan Tan
  - Jesse Chen
  - Elijah Cantella
  - Jay Gogri

Executive Summary

This is the documentation for our WebCheckers project. This project was made through the use of the Spark framework and FreeMarker template engine.

Purpose

To allow users to sign in, sign out, and enter a game with another player by clicking on the opponent's name in the home page.

Requirements

During the first sprint, we were given two main tasks to complete

- Implement a Sign In interface so that users can sign in and sign out
  - Name restrictions
    - No duplicate names
    - No null names such as double quotes (“)
  - List out all the users currently online only if the user is sign in
- Implement a Game interface so that users can start a game with each other
  - Utliizing that list a user can click on another player and start a game with him/her
    - If selected player is in a game than, a message should show that a game cannot be started
  - The pieces on the board should be displayed properly, with challenger as red and challenged as white, with respective pieces on the bottom of each user's board
  - Pieces should be draggable and droppable on valid places on the board

During the second sprint, we decided to complete most of the MVP laid out for us.

- Implement the Player Movement
  - Basic movement of one diagonal space
  - Capture movement of two diagonal spaces, and also a necessary opponent piece to capture
  - King movement needs to expand on both of the above
- Implement the Win/Loss scenarios
  - Basically if either player captures all the pieces of the opponent, he/she will win the game.
  - If a player has no more pieces to play with, he/she loses the game.

During the third sprint, the rest of the project had to be completed and bugfree. We decided to complete the MVP and also two Enhancements features.

- Implement the Player Quit, which completes the Win/Loss scenarios.
  - Either one of the players has the option to quit or forfeit the game.
  - This in turn will send both players to the score page with the appropriate message.

- Implement the Enhancements features.
  - Pause and Resume
    * Players can pause the game at any point of the game, and a message will appear noting who paused the game.
    * This makes all the piece not draggable, essentially making the game not playable until a player resumes the game.
  - Game Chat
    * Players can talk to each other while playing the game.

Definition of MVP

The Minimun Viable Product should be a product that can sign a user in and out (if such user is already signed in), and start a game with properly aligned pieces. With the properly aligned pieces, a player should not be able to move his/her opponent's pieces, and shall only be able to move their own according to a turn based system. The game should work as intended by the American rules.

MVP Features

- Sign In
  - Player Sign-In
  - Player Sign-Out
- Game Play
  - Player Setup
  - Disc Placement
- Player Turn
  - Player Movement
  - Player Capture
  - Player King
- End Game
  - Player Win
  - Player Lose
  - Player Quit
- Enhancements
  - Pause
  - Resume
  - Game Chat

Roadmap of Enhancements

- Have fun (:

Application Domain

This section describes the application domain.

Overview of Major Domain Areas

The main domain areas are the Player, CheckersGame, Board, Squares, and Pieces. These define the checkers game that two players will play utilizing pieces on a board.

Details of each Domain Area

- Pieces are utilized to represent either Player on the board, which is either a red or white piece.
- Players play a CheckersGame which is played on a Board represented by Squares.
- Players will take turns to move their own pieces and ultimately win the game.

Architecture

This section describes the application architecture.

Summary

The WebCheckers webapp will use a Java-based web server, this is done using the Spark web micro framework and the FreeMarker template engine to handle HTTP requests and generate HTTP responses.

Overview of User Interface

The application's user interface consists of two parts, the server UI and the Client UI. In the Server UI it is made up of the various routes we implemented so that the Spark framework and the FreeMarker template engine's are able to handle HTTP request and HTTP responses. We also alter and made new ftl files in order to create the Client UI, which is a combinations of HTML and CSS. Together these two parts made up the UI views and UI controllers.

UI Tier

The user of the application interacts with the UI tier, then the UI tier interacts with the Application and Model tiers. We implemented various routes in order display proper information back to the user.

- GetChatRoute
  - This class allows the two players to talk to each other while playing the game.
- PostChatRoute
  - This class keeps the messages between the players visible and ensures that they don't disappear.
- GetGameRoute
  - This class deals with initializing a game, where a player had clicked on an opponent in the home screen to face against.
    - If the player had clicked on an opponent already in a game then he/she will be sent back to the home page with a error message and may click on another opponent.
    - If it is a valid game, then the player will be the red piece and the opponent will be the white piece with respective pieces on the bottom of whoever's screen.
- PostGameRoute
  - This class deals with the ongoing game that was created in GetGameRoute.
    - Whenever a move is made by the user (either side), this route takes in the information and uses it to update the game itself.
    - The information will be incorporated well into the game but updating the board and views (GetGameRoute).
- GetHomeRoute
  - This is the home screen of the whole application. This is what the player will see as the first thing he/she opens up the web browser.
  - In the navigation bar the player may sign in.
  - Only when he/she is signed may he/she see all the players online and click on someone to face against.

- GetPauseRoute
  - This class allows the game to be paused until later notice (Resume).
- PostPauseRoute
  - Ensures the game is paused indefinitely until later notice (Resume).
- GetScoreRoute
  - This class deals with the end game page.
    - Contains a message regarding the end game state of who won and lost.
- GetSignInRoute
  - When a player decides to sign in, this class allows the player to sign in with a username
- PostSignInRoute
  - When the player enters a username of their choice, this handles if the username is a valid username or not.
  - Returns a error message if the username is taken or if it is a empty username.
- GetSignOutRoute
  - Signs a player out and returns the player to the home page.

Application Tier

The Application tier holds the logic that controls the flow of the application.

- GameCenter
  - This class initializes a CheckersGame instance.
  - If there is an existing game instance, it will return that instead, so a game wouldn't be lost.
- PlayerLobby
  - Holds all the online players so that they can be displayed in the home page.
- PlayerServices
  - Each player needs to be part of a GameCenter, where they will be part of a CheckersGame
- TurnAdministrator
  - Checks for end game scenarios, such as no remaining moves or no pieces left.

Model Tier

The Model holds all the of the core domain logic.

- CheckersGame
  - Is part of the GameCenter.
  - Holds a Board.
- Board
  - Initializes the board by passing in white or black tiles to the Row class.
- Row
  - Creates each row on the board using the Square class.
  - Places all the pawn pieces in the rows on the board
- Square
  - Contains information on each individual square on the board.
  - If the square is valid, if it has a Piece.
- Piece
  - Contains information on a singular piece, the color and type.
- Player
  - Contains information on each user of the application as long as they are signed in.
  - Holds his/her username.
- Movement (sub directory)
  - Move
    - An abstract class that defines a move
  - NormalRegularMove

    * Extends the Move class
    * Defines a move that can be made by a pawn moving forward
  - NormalCaptureMove
    * Extends the Move class
    * Defines a capturing move that can be made by a pawn moving forward
  - KingRegularMove
    * Extends the Move class
    * Defines a move that can be made by a king moving forward/backward
  - KingCaptureMove
    * Extends the Move class
    * Defines a capturing move that can be made by a king moving forward/backward
  - PossibleMoves
    * A class that determines the moves that can be made by the player, represented by a list

Sub-system GameCenter

This section describes the detail design of sub-system GameCenter.

Purpose of the sub-system

Essentially this the brains of the game part of the application, it is responsible for assigning responsiblities onto other classes. The game hiearachy is basically a GameCenter grabbing a game instance either new or existing (determined in the PlayerServices), then the game has a Board, which has a Row containing Squares with Pieces on it. Then all of these is handled with by the GetGameRoute which displays everything.

Sub-system TurnAdministrator

This section describes the detail design of sub-system TurnAdministrator.

Purpose of the sub-system

Essentially this is what determines what an end game is. This basically pulls information off of the PossibleMoves class, which in turn uses information in the other classes in the Movement sub directory. It is responsible for forcing the respective player to make a capture move whenever it is available instead of making a regular forward move. In addition it ends the game whenever it senses that one player has captured all the opposing pieces or no moves can be made.

Static Models

Dynamic Models

Code Metrics

This section will describe the code metric measurements for the following categories.

- Chidamber-Kemerer metrics
- Complexity metrics
- Javadoc coverage metrics
- Lines of code metrics
- Martin package metrics

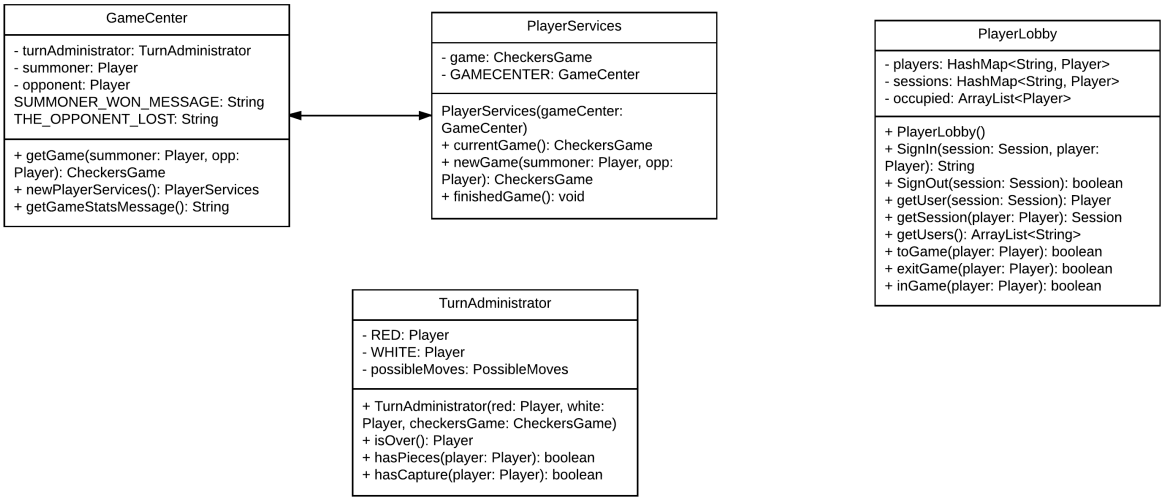

Figure 1: UML for the Application Tier

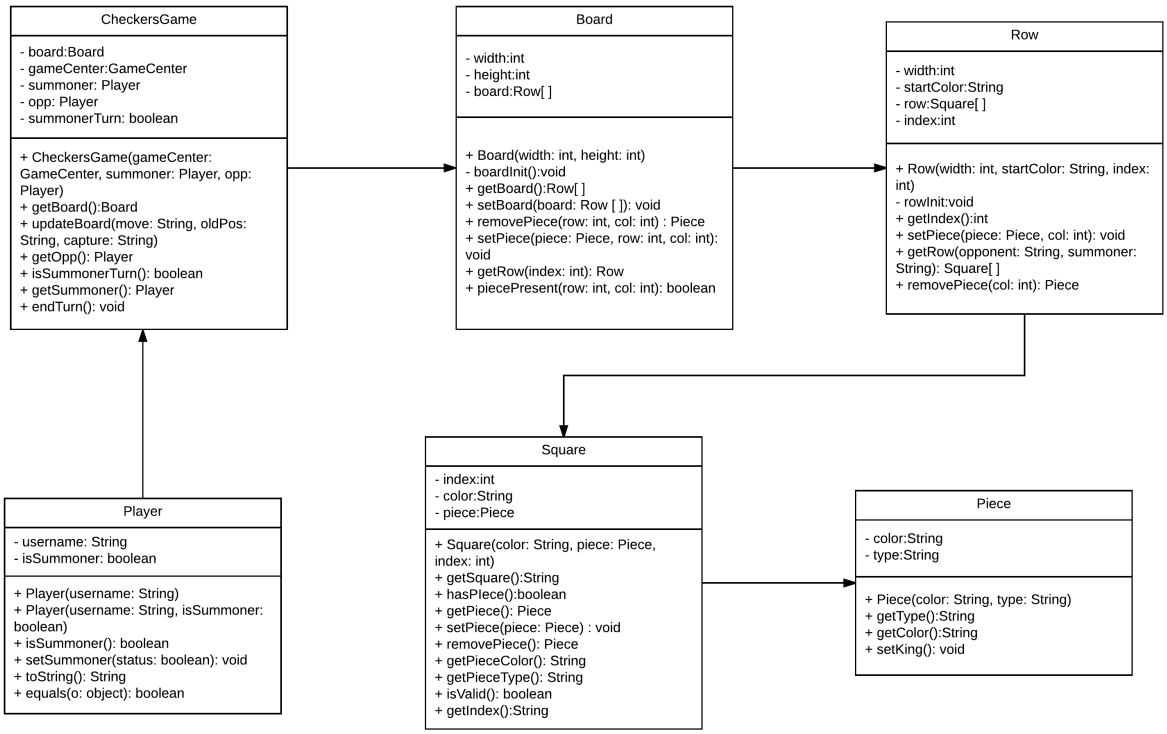

Figure 2: UML for the Model Tier

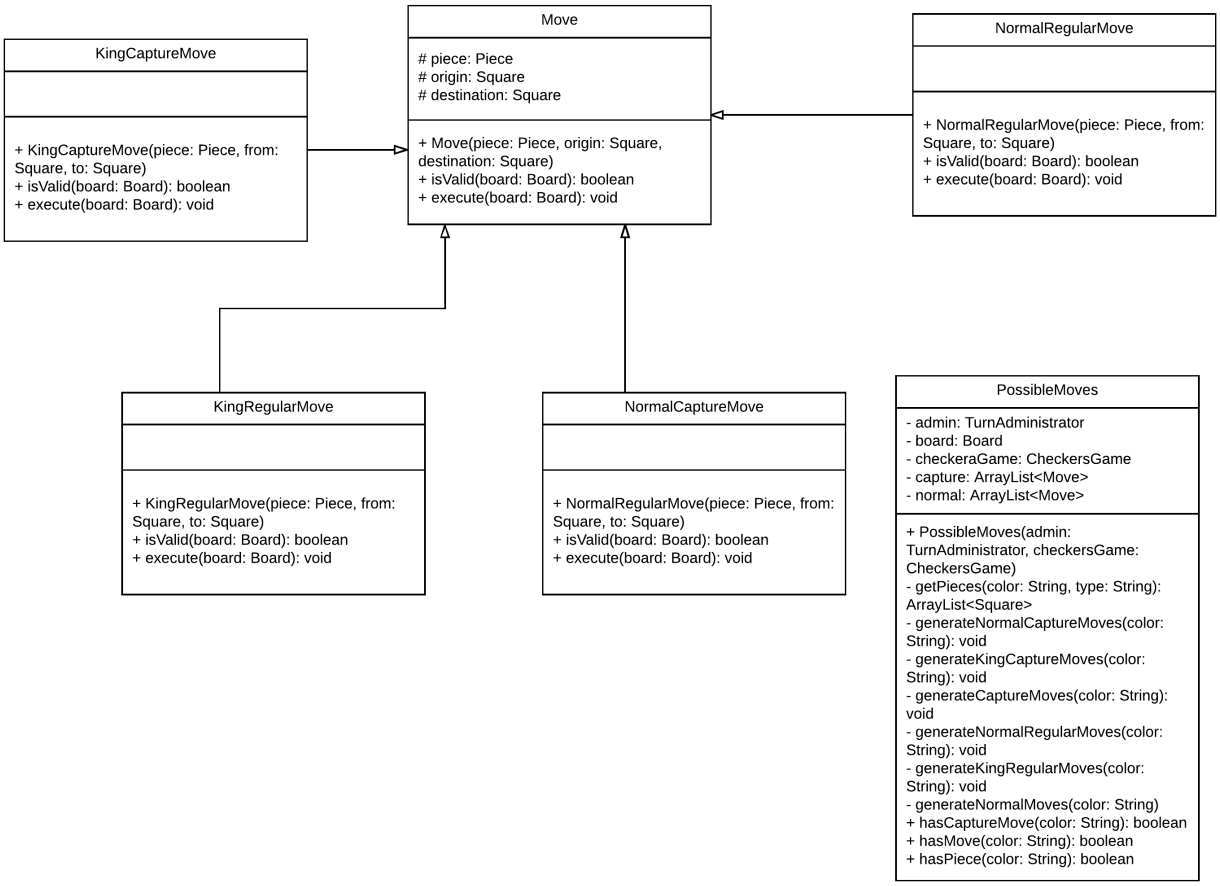

Figure 3: UML for the Movement sub directory

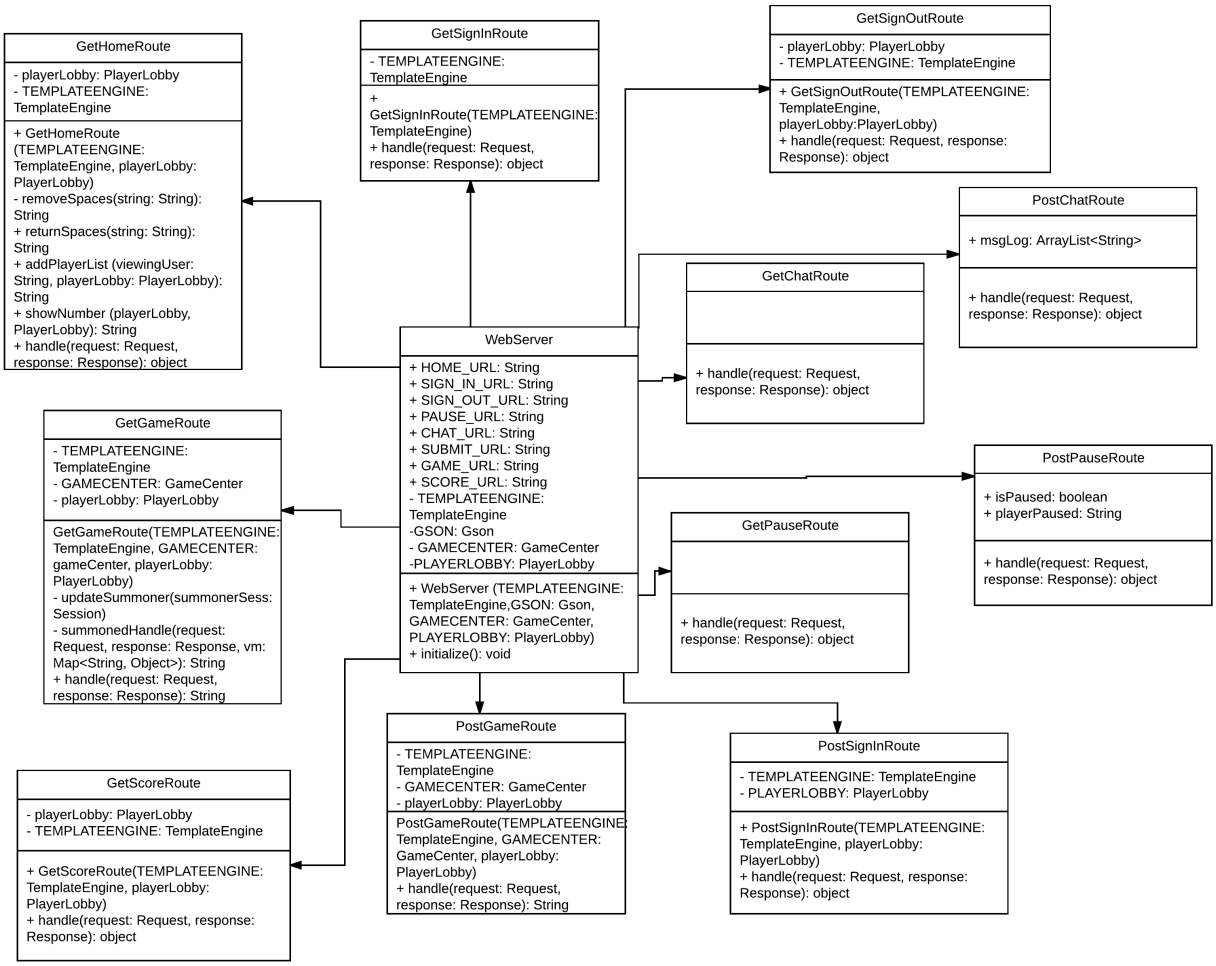

Figure 4: UML for the UI Tier

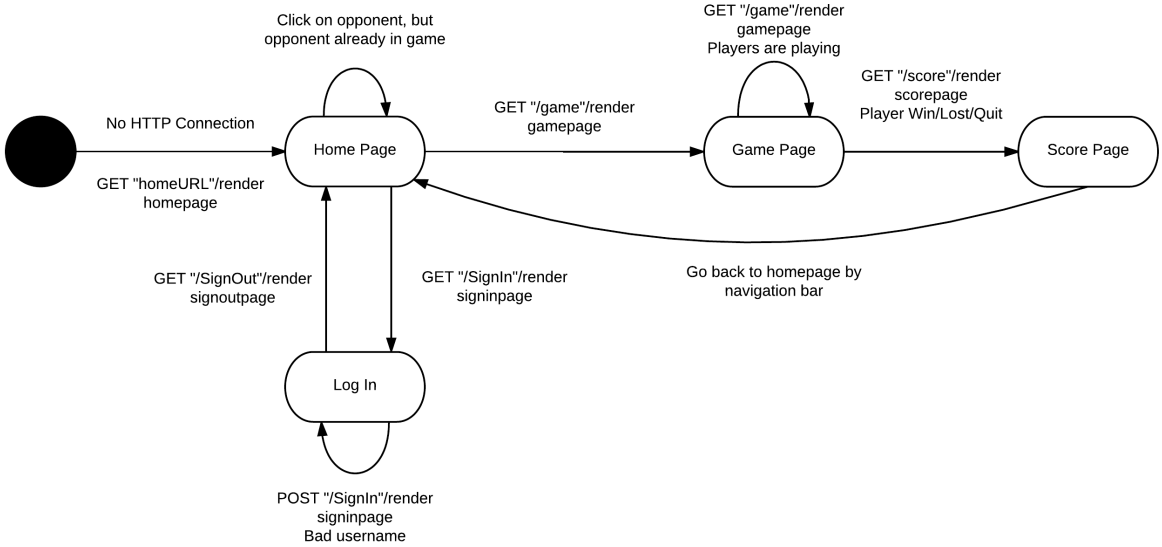

Figure 5: State Diagram

Chidamber-Kemerer Metrics

This metric determines coupling between object classes, and lack of cohesion in methods.

We did well in this aspect since, nothing was marked red when calculating this metric.

Complexity Metrics

This metric is cyclomatic complexity, and is the count of execution paths through a method.

Given this result, we can see that they are a couple of classes that were marked red.

- OCavg (Average Operations Complexity) - amount of some computational resource used by the algorithm, averaged over all possible inputs
  - KingCaptureMove - scored a 4.33
    - Mainly because for a king, it can move backwards and forwards, there are many more types of captures than a regular capture.
    - Decided not to change this class, although it can take up resources.
  - NormalCaptureMove - scored a 3.67
    - Has to cycle through a valid capture move that is made by a pawn, and it is reasonable for it to be taking up resources to do this.
    - Decided not to change this class, although it can take up resources.
  - PossibleMoves - scored a 4.45
    - Needs to determine ALL possible moves that can be made at any given time, it communicates with 4 other classes to accomplish this task and this is justified.
    - Decided not to change this class, although it can take up resources.
  - GetGameRoute - scored a 3.75
    - As the largest route class made, it takes cares of four separate scenarios.
      - Player (summoner) goes into the game
      - Player (opponent) gets dragged into game
      - Player clicks on opponent that is in a game
      - Player forfeits
    - Decided not to change this class, although it can take up resources.
- WMC (Weighted Method Complexity) - sum of complexities of all class methods
  - PossibleMoves
    - As described above, this class should require a lot of resources to accomplish its given task.
    - Decided not to change this class, although it can take up resources.
  - Square
    - This is due to all the switch cases that were implemented.
    - Decided not to change this class, although it can take up resources.

Javadoc Coverage Metrics

This metric determines the percentage of code coverage we have accomplished

Although nothing is marked in red, one can see that we don't have a good average coverage percentage.

Lines of Code Metrics

This metric shows the lines of code that was created for this project. This is broken down into categories and per method.

Nothing is marked in red, and the results of this code metric was expected. This project required a lot of lines of code to finish and we think this is a good graphical representation of our project.

Metrics Chidamber-Kemerer metrics for Project 'checkers-app' from Tue, 5 Dec 2017 16:08:48 EST

Class metrics

class	CBO	DIT	LCOM	NOC	RFC	WMC
com.webcheckers.appl.GameCenter	11	1	3	0	10	5
com.webcheckers.appl.PlayerLobby	15	1	1	0	24	21
com.webcheckers.appl.PlayerLobbyTest	2	1	1	0	26	11
com.webcheckers.appl.PlayerServices	7	1	1	0	5	4
com.webcheckers.appl.TurnAdministrator	6	1	1	0	9	10
com.webcheckers.Application	3	1	1	0	19	3
com.webcheckers.model.Board	12	1	1	0	24	19
com.webcheckers.model.BoardTest	4	1	1	0	22	4
com.webcheckers.model.CheckersGame	12	1	4	0	20	11
com.webcheckers.model.CheckersGameTest	4	1	1	0	13	1
com.webcheckers.model.Move	1	1	1	0	2	2
com.webcheckers.model.Movement.KingCap	6	2	1	0	14	13
com.webcheckers.model.Movement.KingReg	5	2	1	0	10	7
com.webcheckers.model.Movement.Move	8	1	2	4	3	1
com.webcheckers.model.Movement.Move.M	0		0		0	0
com.webcheckers.model.Movement.NormalC	6	2	1	0	14	11
com.webcheckers.model.Movement.NormalF	5	2	1	0	10	7
com.webcheckers.model.Movement.Possible	12	1	1	0	32	49
com.webcheckers.model.MoveTest	1	1	1	0	6	2
com.webcheckers.model.Piece	14	1	2	0	4	4
com.webcheckers.model.PieceTest	1	1	1	0	11	3
com.webcheckers.model.Player	17	1	1	0	7	8
com.webcheckers.model.PlayerTest	1	1	1	0	20	5
com.webcheckers.model.Row	8	1	1	0	16	16
com.webcheckers.model.RowTest	3	1	1	0	13	3
com.webcheckers.model.Square	12	1	3	0	19	32
com.webcheckers.model.SquareTest	2	1	1	0	20	4
com.webcheckers.ui.GetChatRoute	2	1	1	0	7	3
com.webcheckers.ui.GetGameRoute	10	1	1	0	38	15
com.webcheckers.ui.GetGameRouteTest	8	1	1	0	29	6
com.webcheckers.ui.GetHomeRoute	10	1	2	0	26	10
com.webcheckers.ui.GetHomeRouteTest	4	1	1	0	22	3
com.webcheckers.ui.GetPauseRoute	2	1	1	0	5	1
com.webcheckers.ui.GetScoreRoute	5	1	1	0	17	3
com.webcheckers.ui.GetSignInRoute	4	1	1	0	11	2
com.webcheckers.ui.GetSignInRouteTest	2	1	1	0	16	2
com.webcheckers.ui.GetSignOutRoute	5	1	1	0	14	2
com.webcheckers.ui.GetSignOutRouteTest	4	1	1	0	18	2
com.webcheckers.ui.InterfaceVariable	8	1	0	0	0	0
com.webcheckers.ui.MyModelAndView	5	1	1	0	1	2
com.webcheckers.ui.PostChatRoute	2	1	1	0	10	3
com.webcheckers.ui.PostGameRoute	8	1	1	0	18	3
com.webcheckers.ui.PostGameRouteTest	8	1	1	0	16	2
com.webcheckers.ui.PostPauseRoute	2	1	1	0	8	3
com.webcheckers.ui.PostSignInRoute	6	1	1	0	20	3
com.webcheckers.ui.PostSignInRouteTest	5	1	1	0	20	3
com.webcheckers.ui.WebServer	14	1	1	0	16	2
Total						**326**
Average	6.21	1.09	1.17	0.09	14.57	6.94

Figure 6: Chidamber-Kemerer result

Method metrics | Class metrics | Package metrics | Module metrics | Project metrics

lass	OCavg	WMC
com.webcheckers.appl.GameCenter	1.67	5
com.webcheckers.appl.PlayerLobby	2.33	21
com.webcheckers.appl.PlayerLobbyTest	1.22	11
com.webcheckers.appl.PlayerServices	1.00	4
com.webcheckers.appl.TurnAdministrator	2.50	10
com.webcheckers.Application	1.00	3
com.webcheckers.model.Board	1.46	19
com.webcheckers.model.BoardTest	1.00	4
com.webcheckers.model.CheckersGame	1.38	11
com.webcheckers.model.CheckersGameTest	1.00	1
com.webcheckers.model.Move	1.00	2
com.webcheckers.model.Movement.KingCap	4.33	13
com.webcheckers.model.Movement.KingReg	2.33	7
com.webcheckers.model.Movement.Move	1.00	1
com.webcheckers.model.Movement.Move.M		0
com.webcheckers.model.Movement.NormalC	3.67	11
com.webcheckers.model.Movement.NormalF	2.33	7
com.webcheckers.model.Movement.Possible	4.45	49
com.webcheckers.model.MoveTest	1.00	2
com.webcheckers.model.Piece	1.00	4
com.webcheckers.model.PieceTest	1.00	3
com.webcheckers.model.Player	1.33	8
com.webcheckers.model.PlayerTest	1.00	5
com.webcheckers.model.Row	2.00	16
com.webcheckers.model.RowTest	1.00	3
com.webcheckers.model.Square	2.00	32
com.webcheckers.model.SquareTest	1.00	4
com.webcheckers.ui.GetChatRoute	3.00	3
com.webcheckers.ui.GetGameRoute	3.75	15
com.webcheckers.ui.GetGameRouteTest	1.00	6
com.webcheckers.ui.GetHomeRoute	1.67	10
com.webcheckers.ui.GetHomeRouteTest	1.00	3
com.webcheckers.ui.GetPauseRoute	1.00	1
com.webcheckers.ui.GetScoreRoute	1.50	3
com.webcheckers.ui.GetSignInRoute	1.00	2
com.webcheckers.ui.GetSignInRouteTest	1.00	2
com.webcheckers.ui.GetSignOutRoute	1.00	2
com.webcheckers.ui.GetSignOutRouteTest	1.00	2
com.webcheckers.ui.InterfaceVariable		0
com.webcheckers.ui.MyModelAndView	1.00	2
com.webcheckers.ui.PostChatRoute	3.00	3
com.webcheckers.ui.PostGameRoute	1.50	3
com.webcheckers.ui.PostGameRouteTest	1.00	2
com.webcheckers.ui.PostPauseRoute	3.00	3
com.webcheckers.ui.PostSignInRoute	1.50	3
com.webcheckers.ui.PostSignInRouteTest	1.00	3
com.webcheckers.ui.WebServer	1.00	2
Total		**326**
Average	1.80	6.94

Figure 7: Complexity result

Metrics Javadoc coverage metrics for Project 'checkers-app' from Tue, 5 Dec 2017 16:06:33 EST

Method metrics | Class metrics | Package metrics | Module metrics | Project metrics

class	Jf	JLOC	Jm
com.webcheckers.appl.GameCenter	0.00%	0	0.00%
com.webcheckers.appl.PlayerLobby	0.00%	51	100.00%
com.webcheckers.appl.PlayerLobbyTest	0.00%	32	100.00%
com.webcheckers.appl.PlayerServices	0.00%	0	0.00%
com.webcheckers.appl.TurnAdministrator	0.00%	6	0.00%
com.webcheckers.Application	0.00%	16	33.33%
com.webcheckers.model.Board	0.00%	0	0.00%
com.webcheckers.model.BoardTest	0.00%	0	0.00%
com.webcheckers.model.CheckersGame	0.00%	0	0.00%
com.webcheckers.model.CheckersGameTest	0.00%	0	0.00%
com.webcheckers.model.Move	0.00%	5	0.00%
com.webcheckers.model.Movement.KingCap	100.00%	21	100.00%
com.webcheckers.model.Movement.KingReg	100.00%	21	100.00%
com.webcheckers.model.Movement.Move	0.00%	14	66.67%
com.webcheckers.model.Movement.Move.M	0.00%	0	100.00%
com.webcheckers.model.Movement.NormalC	100.00%	21	100.00%
com.webcheckers.model.Movement.NormalF	100.00%	20	100.00%
com.webcheckers.model.Movement.Possible	0.00%	5	0.00%
com.webcheckers.model.MoveTest	0.00%	0	0.00%
com.webcheckers.model.Piece	0.00%	3	0.00%
com.webcheckers.model.PieceTest	0.00%	0	0.00%
com.webcheckers.model.Player	0.00%	15	33.33%
com.webcheckers.model.PlayerTest	0.00%	0	0.00%
com.webcheckers.model.Row	0.00%	0	0.00%
com.webcheckers.model.RowTest	0.00%	0	0.00%
com.webcheckers.model.Square	0.00%	0	0.00%
com.webcheckers.model.SquareTest	0.00%	12	75.00%
com.webcheckers.ui.GetChatRoute	0.00%	3	100.00%
com.webcheckers.ui.GetGameRoute	0.00%	8	50.00%
com.webcheckers.ui.GetGameRouteTest	11.11%	19	66.67%
com.webcheckers.ui.GetHomeRoute	0.00%	37	66.67%
com.webcheckers.ui.GetHomeRouteTest	0.00%	9	66.67%
com.webcheckers.ui.GetPauseRoute	0.00%	3	100.00%
com.webcheckers.ui.GetScoreRoute	0.00%	23	100.00%
com.webcheckers.ui.GetSignInRoute	0.00%	23	100.00%
com.webcheckers.ui.GetSignInRouteTest	0.00%	12	100.00%
com.webcheckers.ui.GetSignOutRoute	0.00%	10	50.00%
com.webcheckers.ui.GetSignOutRouteTest	0.00%	0	0.00%
com.webcheckers.ui.InterfaceVariable	0.00%	6	100.00%
com.webcheckers.ui.MyModelAndView	0.00%	10	100.00%
com.webcheckers.ui.PostChatRoute	0.00%	3	100.00%
com.webcheckers.ui.PostGameRoute	0.00%	8	100.00%
com.webcheckers.ui.PostGameRouteTest	7.69%	10	50.00%
com.webcheckers.ui.PostPauseRoute	0.00%	3	100.00%
com.webcheckers.ui.PostSignInRoute	0.00%	22	100.00%
com.webcheckers.ui.PostSignInRouteTest	0.00%	0	0.00%
com.webcheckers.ui.WebServer	23.08%	62	100.00%
Total		**513**	
Average	2.25%	10.91	37.91%

Figure 8: Coverage result

Metrics Lines of code metrics for Project 'checkers-app' from Tue, 5 Dec 2017 16:08:24 EST

Package metrics | Module metrics | File type metrics | Project metrics

package	LOC	LOC(rec)	LOCp	LOCp(rec)	LOCt	LOCt(rec)
	26	6,678	26	5,615	0	1,063
com		3,403		2,340		1,063
com.webcheckers	83	3,403	83	2,340	0	1,063
com.webcheckers.appl	407	407	258	258	149	149
com.webcheckers.model	913	1,453	586	1,126	327	327
com.webcheckers.model.Movement	540	540	540	540	0	0
com.webcheckers.ui	1,460	1,460	873	873	587	587
public		2,722		2,722		0
public.css	248	248	248	248	0	0
public.img	375	375	375	375	0	0
public.js	5	2,099	5	2,099	0	0
public.js.game	815	2,094	815	2,094	0	0
public.js.game.model	108	108	108	108	0	0
public.js.game.modes		964		964		0
public.js.game.modes.play	770	770	770	770	0	0
public.js.game.modes.replay	97	97	97	97	0	0
public.js.game.modes.spectator	97	97	97	97	0	0
public.js.game.util	207	207	207	207	0	0
spark		527		527		0
spark.template		527		527		0
spark.template.freemarker	527	527	527	527	0	0
Total	**6,678**		**5,615**		**1,063**	
Average	417.38		350.94		66.44	

Figure 9: Lines of Code result

Martin Package Metrics

This metric shows fan-out and fan-in coupling, instability and abstractness.

Metrics Martin packaging metrics for Project 'checkers-app' from Tue, 5 Dec 2017 16:05:59 EST

Package metrics

package	A	Ca	Ce	D	I
com.webcheckers	0.00	0	2	0.00	1.00
com.webcheckers.appl	0.00	12	9	0.57	0.43
com.webcheckers.model	0.00	16	13	0.55	0.45
com.webcheckers.model.Movement	0.14	7	12	0.23	0.63
com.webcheckers.ui	0.00	15	38	0.28	0.72
Total					
Average	0.02	10.00	14.80	0.33	0.60

Figure 10: Martin Package result

The results of this metric shows us that we didn't violate the low coupling principle, as both the fan-out and fan-in coupling were low and not marked in red. Since both couplings were low, the instability was also low.