Dominic Pilla

Professor Dale Musser

CMP_SC 4330

12 February 2018

Checkers Game Data Model Concept

Player – Public Class

New players will be created with the Player class. The turns the players take will be determined by the controller in a random order. Player 1 will move first on the board, followed by Player 2.

Fields:

- firstName – String
- lastName – String
- wins – int
- losses – int
- gamesPlayed – int
- score - int
- checkers – [Checker]

Checker – Public Class

The Checker class will be utilized for controlling the checkers for each player. The top side of the board will be controlled by Player 1. Player 1's checker colors will be RED. The bottom set of Checkers will be controlled by Player 2. Player 2's checker colors will be BLUE. When it's a Players turn they will be able to move their checkers, from the array of checkers listed in the Checker's field within Player.

Fields:

- Kinged – bool
- Row - int
- Column – int

Methods:

- *Move* – This method within the Checker class will first (assuming nulls have been checked) check to see if the Player is moving within the grid. For Player 1, it will check if the move is +1 row and either -1 or +1 for the left or right movements across columns, and vise-versa for Player 2. When a Player has a Kinged checker then the row check will be superseded and go straight to column movements. There will also be a need for checking if a user is trying to move across 2 rows, and if there is an enemy Checker between the move then it will need take that Checker and approve the move.
- *Build* – This factory method within the Checker class will be used to build out new Checker pieces. This will be useful when instantiating new pieces on the board, because instead of managing Circle objects, I'll be able to manage Checker objects. All of the data can all be held within one object, instead of two.

Scoreboard – Public Class

The Scoreboard class will help to manage the score and read/write of data for the game. The score will be managed by the Scoreboard, along with Player statistics like wins, losses, games played, and score. Each checker you overtake in the game will count as 100 points, and kings will count as 300 points. This will introduce a competitive aspect to the game. The Scoreboard class will also manage the winning and losing screens. Since this class will be managing Players it will be necessary for it to hold references to each Player class.

Fields:

- Player1 – Player
- Player2 – Player
- totalScoreForGame – int
- timePlayed – double

Method:

- *writeGameToFile* – This method will write the finished game in a JSON format to a file. This will log both Players statistics, the total score, and time played. It will be in an array format within the JSON file so that many games can be listed and retrieved.

- *readGamesFromFile* – This method will retrieve the games from the JSON file and prepare them for the UI. This will easily allow the developer to make a single call to retrieve the array of previously played games.