Sprint 4 Document

Architecture and Design Rationales

Prepared by 3 Men's Morris
Jer (Arthur) Lin, Quoc (Harry) Han and Liangdi Wang

Contents

1. Revised User Stories

Revised User stories to cover the advanced requirement(s) only, following the recommended user story template and guidelines. If your stories have not changed since sprint one, submit them anyway and clearly indicate to us that there was no change.

1. As a player, I want to be able to play against a computer, so that I can play without another person present.
Acceptance criteria:

- There is an option to play vs computer on the starting screen
- The player plays a move, then the computer responds with a move
- The computer's move is shown on the screen
- The game continues with the player's turn after the computer's move

2. As a player, I want the computer to make moves that adhere to the game rules, so that I can play a fair game.
Acceptance criteria:

- The computer should make a move among all currently valid moves
- The computer can only make a legal move

3. As a computer player, I want a list of valid moves for the current turn, so that I can choose a move to play for the turn.
Acceptance criteria:

- When it is the computer’s turn to play a turn, return a list of all current legal moves
- Given the moves, the computer player also have access to the list and make a decision on the move

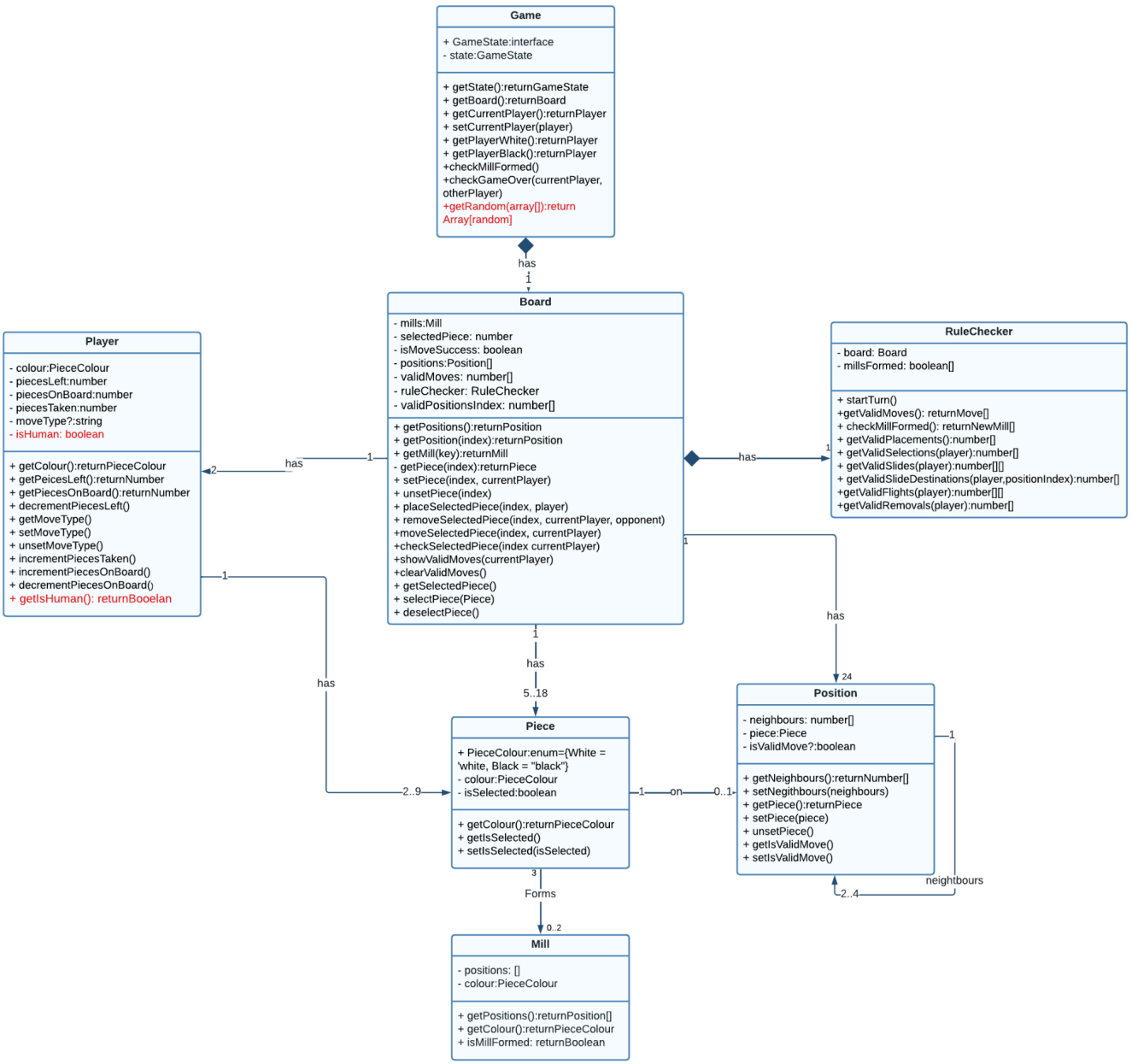
4. As a computer player, I want to be able to remove a piece when I create a mill, so that I can adhere to the game rule while attempting to win.

- If the computer makes a move and creates a mill, it should be prompted to make another move to remove a piece
- All the piece that can be removed must be presented as an options
- All the pieces formed must not be removed unless there are no single pieces left

5. As a computer player, I want to win the game when reaching the winning condition, so that I can play the game without concern of human bias.

- If the player only has a total of 2 pieces left, the computer will win.
- If the player doesn’t have any valid moves, the computer will win.

2. Revised Architecture



The computer class has been completely removed, it was noted that the computer will do everything a player does except the decision making step is automated. Hence, a boolean flag in the human class will suffice when it comes to distinguishing the two.

3. Design Rationales

3.1. Advanced requirement architecture rationale

When implementing the logic for the computer player, our group conducted a quick analysis to define the differences between the human player and the computer player. The result of the analysis suggests that the computer player is quite similar to the normal player. The only significant difference between those is the decision maker since the normal player's move is created manually by a human's click while the computer player's move is created automatically by a system's randomization. Hence, we have decided to reuse some of the methods that are used to generate the list of valid moves which the normal player can perform each turn, which are:

- getValidSelections(): used for generating a list of computer player's pieces on the board.
- getValidPlacements(), getValidSlides(): used for generating a list of positions the selected piece can be moved to.
- getValidRemovals(): used for generating a list of opponent pieces can be removed.

Due to the fact that the computer player can not interact manually with the board by click, we have created a method getRandom() to support the computer player to select randomly one of their pieces on the board and then use again getRandom() to select one of the valid moves that selected piece can perform. This logic will be applied for the computer player throughout the game regardless of their move's type each turn.

3.2. Architecture revision rationale

There are not many changes that we have done for this sprint 4 since most of the methods such as getValidSelections(), getValidPlacements(), getValidSlides() or getValidRemovals() can be reused for the computer player. The only change that we have made for class Player is adding a new boolean variable called "isHuman" to distinguish the Player instance as human player or computer player.

3.3. Advanced feature explanation

The chosen advanced feature is the implementation of an AI player. This feature was chosen in sprint 1, the team believes it is the feature that would make the game most complete. This is due to the fact that players can still play the game without the presence of another person, the enablement of single players will be the most appealing and bring the most value to the audience. As the project progresses and the team learns about human values, this feature starts to make even more sense for the team as it appeals to human's values of power and achievement. This refers to human's desire to showcase skills and achievements, having an AI to play against not will allow players to experiment with innovative moves and strategies without the need of another human. Note that risky moves may not necessarily work,

therefore it also preserves their public image and only shows off these strategies only when it is honed and refined to work efficiently.

The implementation of the AI player was seamless as the foundations built in first three sprints are well and were continuously improved to follow good object oriented practices and reduce code smells. As the separation of concerns is well defined between classes, the team is able to pinpoint the relevant class and implementation immediately. Knowing that the Game class is in charge with the responsibility to switch players, the team were able to delegate the switching mechanism between the human and AI to this class. Additionally, the implementation of the RuleChecker as a move validation class has also come in handy. As it returns all the valid moves for the current player, the team is able to easily use the output of this class as a list of options for the AI player. Therefore, it is clear the foundation of the code was designed to a high quality, allowing for the implementation for the AI player to happen without major effort in any form of refactoring and cleanup.

Evidence

Only a few main changes were made to successfully implement the advanced feature.

```
3   export class Player {
4       //Attributes for the player class
5       private colour: PieceColour;
6       private piecesLeft: number;
7       private piecesTaken: number;
8       private piecesOnBoard: number;
9       private isHuman: Boolean;
10      private moveType?: string;
11
12      //Constructor for the player class
13      constructor(colour: PieceColour, isHuman: Boolean) {
14          this.colour = colour;
15          this.piecesLeft = 9;
16          this.piecesTaken = 0;
17          this.piecesOnBoard = 0;
18          this.isHuman = isHuman;
19      }
```

A property added to Player class to check if it is a human player, along with getter and setter.

```
119     //Check if mill is formed and handle the case
120     checkMillFormed() {
121         const ruleChecker = this.getRuleChecker();
122         //If a mill is formed, the current player gets to remove a piece. Else, the other player gets to play
123         if (ruleChecker.checkMillFormed()) {
124             this.state.currentPlayer.setMoveType("remove");
125         } else {
126             this.updateCurrentPlayer();
127             //Before handing over the turn, check if the game is over
128             this.checkGameOver(this.state.currentPlayer, this.getOtherPlayer());
129             if (this.getIsGameOver()) {
130                 return
131             }
132             //If the other player is a computer, play a move for the computer
133             if (!this.state.currentPlayer.getIsHuman()) {
134                 this.playComputerMove();
135             }
136         }
137     }
```

Check if the game is over in the mill checking/turn passing function which is called at end of turns. This was previously done in the frontend but needed to be added here since the computer does not interact with the interface.

```
140     playComputerMove() {
141         const ruleChecker = this.getRuleChecker();
142         let index;
143         let selectedPiece;
144         //Based on the state of the game and the move type of the current player, play a move for the computer
145         switch (this.getCurrentPlayer().getMoveType()) {
146             //If the current computer is able to place a piece, place a piece at a random valid location
147             case "place":
148                 index = this.getRandom(ruleChecker.getValidPlacements())
149                 if (index !== undefined) {
150                     this.state.board.placeSelectedPiece(index, this.getCurrentPlayer());
151                 }
152                 break;
153             //If the current computer is able to slide a piece, slide a random piece to a random valid location
154             case "slide":
155                 selectedPiece = this.getRandom(ruleChecker.getValidSlides(this.state.currentPlayer));
156                 if (selectedPiece !== undefined) {
157                     this.state.board.checkSelectedPiece(selectedPiece[0], this.state.currentPlayer);
158                     this.state.board.setSelectedPiece(selectedPiece[0]);
```

Implement a function inside the game class for the computer to play a move. It uses the existing rule checker class to get valid moves for each type of move. It then plays one randomly using existing functions in the board class.

4. Meetings

25/05/2023, 6:00PM - 6:30PM

Location: In Person

Attendees: Liangdi, Harry (Quoc), Arthur (Jer)

Discussion: Initial work breakdown, understanding requirements.

29/05/2023, 10:30AM - 11:00AM

Location: Zoom

Attendees: Liangdi, Harry (Quoc), Arthur (Jer)

Discussion: Split and assign tasks into work breakdown structure, track progress. Discuss advanced feature implementation.

02/06/2023, 7:00PM - 7:30PM

Location: Zoom

Attendees: Liangdi, Harry (Quoc), Arthur (Jer)

Discussion: Explaining code, design. Go over remaining tasks, prepare for the video demo.

06/06/2023, 9:00PM - 11:20PM

Location: Zoom

Attendees: Liangdi, Harry (Quoc), Arthur (Jer)

Discussion: Record video demonstration.