Design Rationale for Domain Model

Advanced requirements Chosen: (a) and (c)

Domain Model and Design Decisions

Player

1. There are currently two types of players - a human and a computer.
2. Each player has 9 tokens at the start of each game. Each player must have at least 3 tokens for the game to continue, otherwise the game ends. Note that a player can have 2 tokens, and this event triggers a game to end.
3. A player can perform one specific action (see Action), depending on which phase (see Phase) of the game they are in.
4. Since a computer player is essentially simulating the behaviour of another human player, a computer player can perform any action that a human player can.
5. However, a computer player can be different from a human player in multiple aspects. For example, a computer player may also need incorporate additional algorithms to compute their moves.

Game

1. A game must be played by exactly 2 players.
2. A game must be shown on one display to convey latest information to the players whenever the board changes.
3. A game must be played on a board, and a board only represents one game.
4. A main menu has 3 types of games to choose from
5. There can be multiple game types (player versus player, player versus computer and tutorial mode).

6. Each game type is played according to the same rules shown above (1-4).

7. Each game type might have different combinations of player types.

Abstraction of Game and Player

1. This design choice is inspired the Liskov Substitution Principle.

2. Collectively, the abstraction of these two classes introduces flexibility into the game by making different combinations of game setups possible.

3. For example, a player (HumanPlayer) may prefer to play a game with a computer (PvCGame). However, they (HumanPlayer) may prefer to play another game (PvPGame), with another human (HumanPlayer) after the first game.

4. A game with different player types and game modes does not change how a game of 9 Men's Morris is played. In other words, these two components can be interchanged, without changing the game rules and logic.

TutorialGame

1. Everything related to tutorial is hosted in the TutorialGame domain entity that inherits from the Game entity to have all functionality of a normal game.
2. TutorialGame includes prompts for the players to follow through. The domain responsible for prompts is TutorialPrompt and TutorialGame has many prompts.

Token

1. Tokens are manipulated by players in order to progress a game. A player with fewer tokens indicates that they are losing (arguably).

2. Tokens currently come in black and white to represent each player in a game.

3. All tokens behave similarly, but will look different from each other.

4. There may be more token colours introduced in the future.

Action

1. Actions represent the moves that a player can perform on a token.

2. A player can perform one or more actions per turn (they can move, form a mill, then remove an opponent's token in the same turn). Throughout the game, they would have performed many actions.

3. One action manipulates only one token.

4. A token can move or fly across the board. It can also be placed or removed by a player. All these actions share a common goal of manipulating the position or state of a token, but they do so differently.

5. For example, both moving and flying moves a token to another empty intersection. A token can only be moved to an adjacent intersection, but a token can fly to any intersection on the board.

Phase

1. A phase defines a set of allowable actions a player can perform.

2. A player can by in any one of three phases based on how they can manipulate their tokens. They can either place, move or fly tokens.

3. For example, a player with all 9 tokens currently in play is allowed to move and remove opponent tokens (MovingPhase), but they are not allowed to place or fly their tokens until they have 3 tokens (FlyingPhase).

4. Players can remove opponent tokens at any phase.

Abstraction of Action and Phase

1. Abstracting both action and phase makes new game rules easier to maintain and extend.

2. If a new game rule specifies that players can fly their tokens during the placing phase, it only needs to be added in the PlacingPhase class, without affecting other phases.
3. A new phase with a combination of new and existing actions can also be added just by creating new classes that implement Phase or Action.

Board

1. A board must exist in the game so players can interact with tokens.
2. A board is a visual indicator of how the game is progressing for both players.
3. The board is essentially a collection of 24 intersections.

Intersection

1. An intersection is an interactable part of a board.
2. Players play the game by moving tokens around different intersections. They can also place or remove tokens in an intersection.
3. One intersection can contain a maximum of one token, but it may also be empty.
4. One intersection neighbours at least two, but not more than four other intersections on the board.

Design choice for Board and Intersection

1. It is possible to store the position of each token in a single Board object. However, it would be hard to maintain whenever the board layout changed.
2. A board would also need to store the exact positions of every single intersection and keep track of every token which would make the code within this class very convoluted.
3. It was decided that certain responsibilities would be delegated to a new class called Intersection.
4. Each intersection is responsible for its own position on the board.

5. Each intersection records if it contains a token or not.
6. In this design, a board only needs to keep track of the relative positions of the intersections, and query each intersection to determine where all the tokens on a board are.

Assumptions

1. For the start turn the white tokens moves first.
2. For the main menu, we decided to allow the user to choose between new game, tutorial game and computers. We find this easier to implement the logic when coding.