Sprint 4

MA_Friday2pm_Team22

User Stories/Architecture/Design Rationale

User Stories

Advanced requirement:
a.Considering that visitors to the student talent exhibition may not necessarily be familiar with 9MM, a tutorial mode needs to be added to the game. Additionally, when playing a match, there should be an option for each player to toggle “hints” that show all of the legal moves the player may make as their next move.

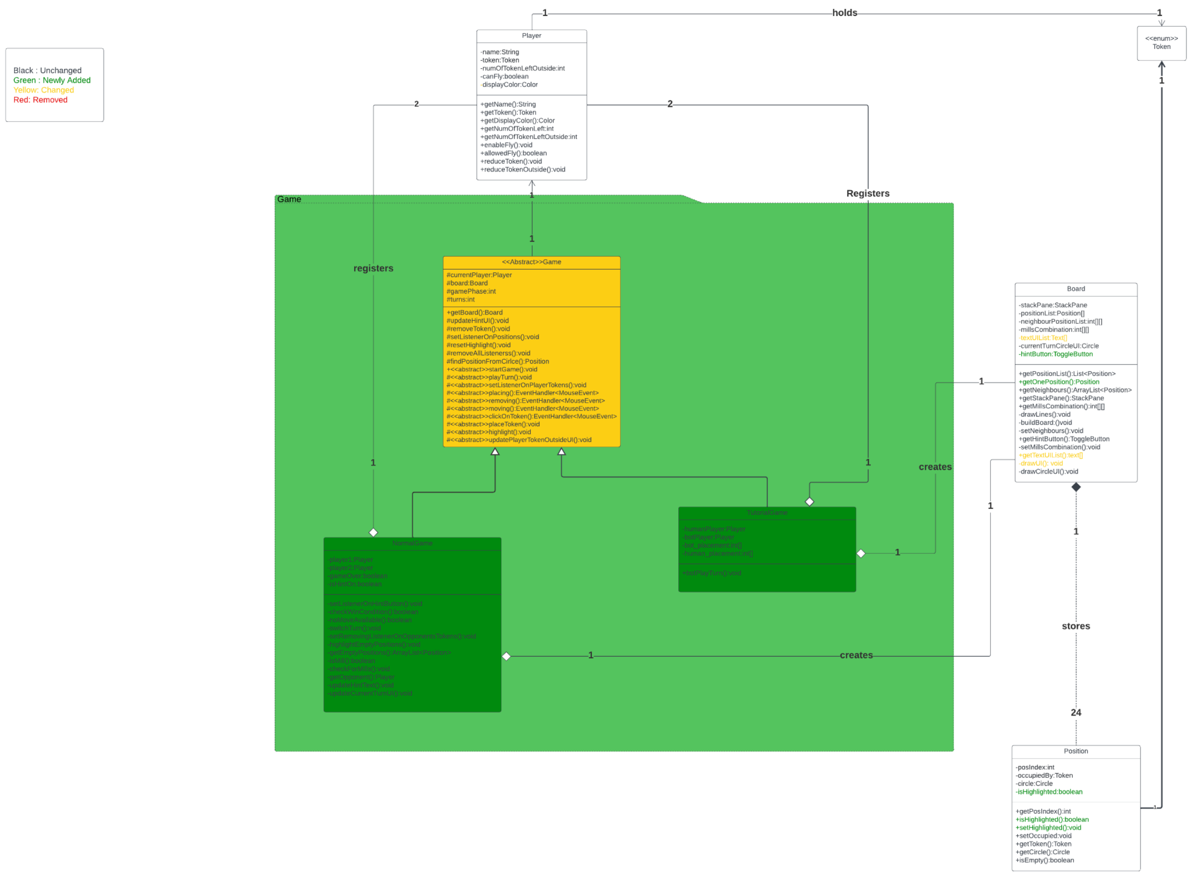
As a player,
I want to have a tutorial mode,
so that I can learn how to play the game.

As a player,
I want to have the option to toggle hints during a match,
so I can see all the legal moves available for my next move.

As a player,
I want the hints to be visually distinct and easily recognizable,
so I can quickly identify the available moves without any confusion.

All the user stories for the advanced requirement was completed.

Architecture



Design Rationale

The team’s selected advanced requirements for the Nine Men's Morris is to design a tutorial mode and hint toggle button for our tech-based final game software in sprint 4.

The reason for having Game class as an abstract class, extended by the NormalGame and TutorialGame classes, is based on the need for common functionality and specific requirements of each class. Both child classes utilise mouse events and listeners to handle user input from mouse clicks. By providing a common abstract class, we can encapsulate the shared code logic to promote reusability and enforce the child class to provide an implementation for the abstract method, ensuring that the expected behaviour is defined and maintained consistently. However, The NormalGame class is designed for gameplay between two human players, while the TutorialGame class is intended for gameplay between a player and a bot. By making the TutorialGame class extend the Game class, we can choose to implement code logic and flow differently from that of NormalGame class such as only allowing predefined moves for the player. This design choice ensures that the player’s actions are limited to the predefined moves set in the tutorial, providing a guided and controlled experience, while players in NormalGame are free to perform any valid move on the board.

The decision to group all Game-related classes within the “Game” package was made to encapsulate and organise the components related to the game functionality. The Game class only has 2 public methods which are startGame() and getBoard() for external components to interact with it and the remaining methods are generally private/protected methods used internally by Game class to handle user input, check game conditions and manage the flow of code executions.

Our Advanced feature was finalised very early because we always laid the groundwork in terms of skeleton code since sprint 2, such as for example the unofficial kind of hint button in sprint 3 as well as the isTutorial variable located in game for checking if the player chose to pick the tutorial.Not only for that reason, we also looked at it in terms of our own capabilities as coders and how difficult it would be if we were to choose another advanced requirement.. However just because our advanced features did not change, our design architecture did in terms of becoming more modular hence making it easier for us to implement the user stories for the advanced requirement.

Since we already have the functionality that the legal move will be highlighted when we want to place, move or remove the tokens in Sprint 3. So we just need to implement a function that shows the highlighted part when toggle on the hint button. To implement this functionality, we add a new boolean variable isHighlighted to position, this variable is to record which position should be highlighted in the following stage when toggle on the hint button. It doesn’t become false when the toggle is off so we can easily track it while toggling on and off. It only becomes false when we click on another position.

When creating the tutorial mode, it was found to be a lot more difficult to put into fruition than initial thoughts,but after the idea to make the Game class into an abstract class the full picture on how we were gonna build the tutorial mode came to be. The thought of a TutorialGame class and NormalGame class was thought of to make it easier to keep the changes from one game mode from affecting the other. This made building the tutorial mode easier because though many of the method names are the same, few to many of the code inside these methods were different then in the TutorialGame class. Another reason why implementing the tutorial mode was easier than first thought of was because we chose to hard code and simulate a normal game with different scenarios, rather than creating an AI that would play the game against the player and point out different scenarios. Another reason for the hard coding to limit the players ability to play which made it easier to implement the tutorial mode was due to the fact that multiple games also do this, such as clash of clans limiting what the player does as a sort of tutorial and telling them what to do and only allowing them to do that action.