Introduction

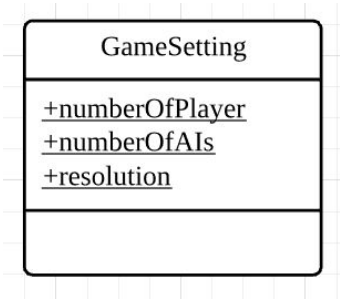
The purpose of our project is to implement the single-player and multiplayer version of the Splendor board game based on standard Java API. We completed the graphical user interface of this game using the MVC design pattern. The view components is designed and implemented with JavaFX along with .CSS file. AI player in the game is based on analysis on situations and calculation of weighted scores. We also plan to Implementation AI opponents in single-player mode and multiplayer mode with other users through network connection.

User stories

User Stories			
As a [persona]	**I [want / need / etc.]**	**So that [state reason]**	**Completion**
Features			
Player of the game	I want the game to have a single player mode	I am able to play the game alone offline.	Completed
Player of the game	I want the game to have AI opponents.	I am able to play with AI opponent when I play the game in the single player mode.	Completed
Player of the game	I want the game to have an offline multiplayer mode.	I am able to play the game with my friends on one computer.	Completed
Player of the game	I want to play the game through a graphical user interface.	I am able to know how many gem tokens I have and how many cards I possesses through the graphical display of the game	Completed
Player of the game	I want cards of the game to have distinguished front image and the cards' cost and value to be listed on the front sides of cards.	I am able to distinguish different card based on their appearances.	Completed
Administrator of the game	I want the game to have three general types of cards: low cost card with low values (prestige points), medium cost cards	Players have more options and flexibility in choosing which cards to buy based on how many gem tokens I have.	Completed

	with medium values, and high cost cards with high values.		
Rules			
Administrator	I want player to do any one of three actions in my turn: buying a card, getting gem tokens, reserving a card. They can only do one of these three actions in one turn.	Players play the game following the original rules of Splendor	Completed
Administrator	I want the game to have five different types of gem tokens, and one golden token which can serve as one of any other five types of tokens when player buys card.	I can increase the difficulty of the game and add more fun	Completed
player of the game	I want to reserve a card during the my turn when I don't have enough tokens to buy that card, and buy the reserved card later. I can only hold at most three reserved cards in hand.	I am able to reserve a high value card when I don't have enough gem tokens to buy it, and I am able to buy my reserved card later.	Completed
Administrator	I want to have discounts on the cost of cards (color) based on how many cards that I have already bought for this particular color.	The game can have multiple strategies to play	Completed
Administrator	I want the game to end once one of the players has got 15 points.	I am able to end the game at that point.	Completed
Administrator	I want nobles to come to player automatically when they are qualified.	players don't have to check their progress after every turn.	Completed
Administrator	I want the players to only hold at most 10 tokens at a time.	There are always tokens available for the other player to collect	Incomplete

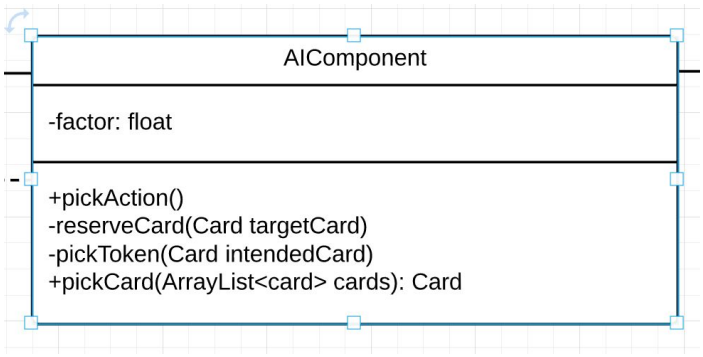
OOD



1. The GameSetting class serves as a place that allows other class make change of the number of AI players, the total players

and the winning points. The user can adjust through the slider in the GameLauncher. This allows them to change the player mode they want, either multiplayers ranging from 1-4 or multiplayers with ai, where ai number ranges from 0 to 3. Additionally, the winning points is ranging from 5 to 20. With the less winning points means, the users will play in a relative short gaming period and vice versa.

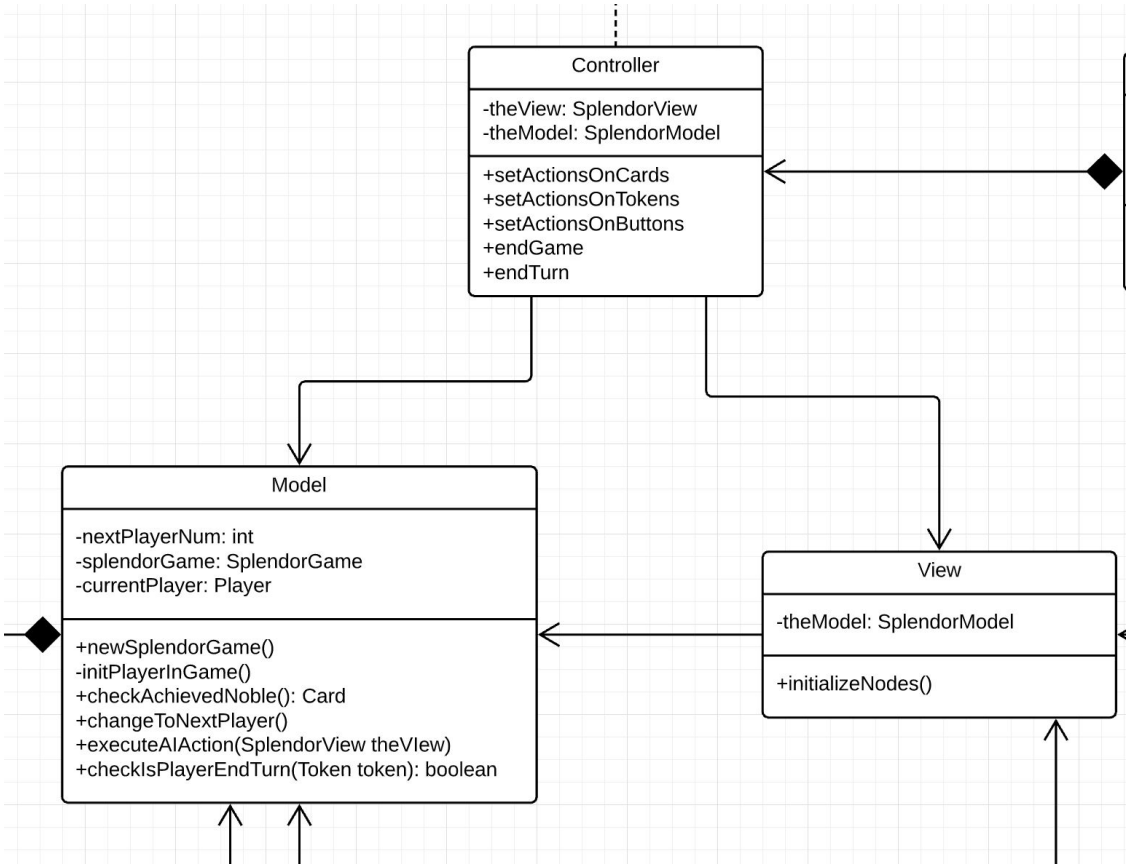
2. AI components pick actions based on the situation on the board. Some if statement are applied when we try to exclude those special circumstances, where the movement of the players in the game is limited (For both AI and Human Players). For each action, the AI either deal with cards or tokens. As a consequence, AI will analyze which card or which



tokens are of most benefits. Thus pick cards and tokens will be based on some basic score calculations. During these calculations, the different token and card are weighted according to the situation on board. AI has a factor which is being randomly generated during the instantiation. This factor allows different AI behaves differently.

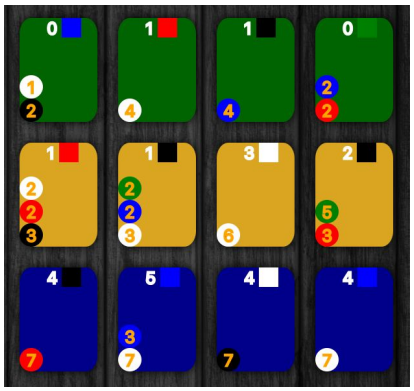
3. We design the view according to MVC design pattern. The view provides user with fascinating interface design. On the main view, the user are able to see their player's information including prestige point, tokens owned, card reserved, and card brought. Information of all the players are public. The board also includes all the cards that are available to buy with their price, value, and colors, the token number of different color

remaining on board, and the nobles on the board. After any of the player reaches the prerequisite of the noble, their icon will be added directly noble. There are four buttons at the right most interface to allow users to restart game, skip turn, and exit. These button allows users to change setting, skip turn when nothing to do within the turn and exit the game.



4. In order to provide different types of cards, we found a .CSV file online that contains all the information of cards. We read the file and stored the information as Card object in the local storage in .DAT file. Our Card object is a class representing the card on board with their values, colors, prices, and levels. So basically, the program of reading the file only needs to run once at the beginning of the game and the cards will be read from the local storage directly. When we create cards in the view, we use a StackPane for each card and

put colored squares to indicate the color type of the card, colored circles with numbers to indicate the prices of the cards, and integers on the card to indicate the points of the cards. Cards of different level has their own different colors -- level 1 cards are green; level 2 cards are yellow and level 3 cards are blue.



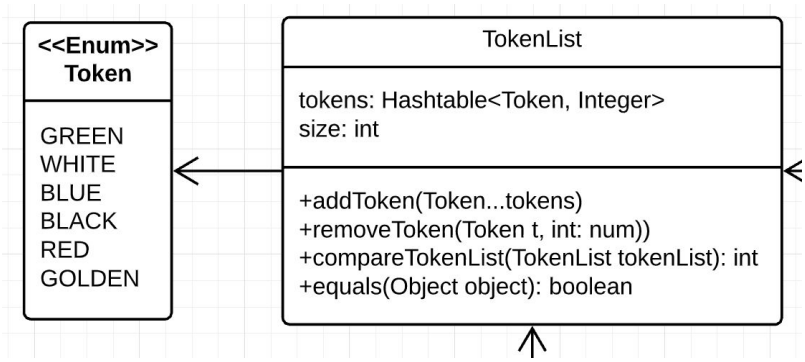
5. We designed a system count for the turn and the current player in the model. Through user's interaction with interface, we could know whether current player has finished his or her part of the game. There is an instance variable of current player obtained from the SplendorGame class, where all the players are stored in an ArrayList. Every change in the player through the controller will only affect the current player until this player end its turn.

SplendorModel	
Responsibility	Collaborator
Initiate the splendor game.	SplendorGame
End the turn of a player when he/she finishes the action	
End the game when there is one player winning the game and ask whether they want to play another round.	

After player get tokens, buy card, or reserve card, the turn will automatically end and change to the next player. At the end of each turn, the system will add nobles to player if

any of them finished the requirements on the nobles. But for each player in each turn, there is only one noble will be added at most.

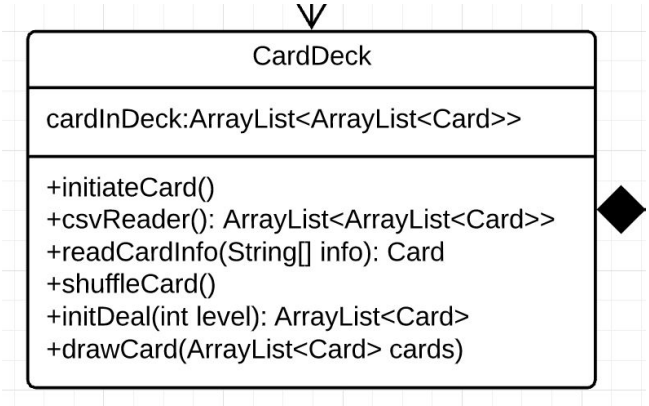
6. We have five different tokens along with a gold token which can be used to replace any of those five types of tokens. We used a customized data structure based on the HashTable to store the information of the five basic tokens and their number. Since we only have five variables to change, the HashTable is the best option for us to use. We also add more functions to this data structure, there are two types of overridden adding method which is convenient, using in multiple places in code. You can either add a TokenList to a TokenList or you can add any number of different tokens to the TokenList.



7. Players are allowed to reserve card in their turn when they are not able to afford the card. This action is based on fundamentally the same algorithm as we do in buy card, only different is, the card information is stored in ArrayList in Player object separated from the card brought by the player.. After the reservation the card count the score or discount for the later purchase, instead, it go the ArrayList for the later action by player.
8. The discount on buy card are automatically calculated based on the number of different cards color they have brought. For example, the red card can help player to save a red

token in the later purchase if the intended card to buy requires some red tokens. The whole process is embedded in player's purchase card method.

9. CardDeck object reads the cards' information from a csv file and use the data to generate a set of cards that players will use in the game. Since noble card is just a special type of card with different type of cost, we use the inheritance to enhance



10. Since players promote the actions such as collecting tokens, acquiring nobles, reserving card and buying cards, player controls the creation of a TokenList object, which keeps track of the token owned by the player, and the player has the references to a set of cards he possessed and Noble he acquired.