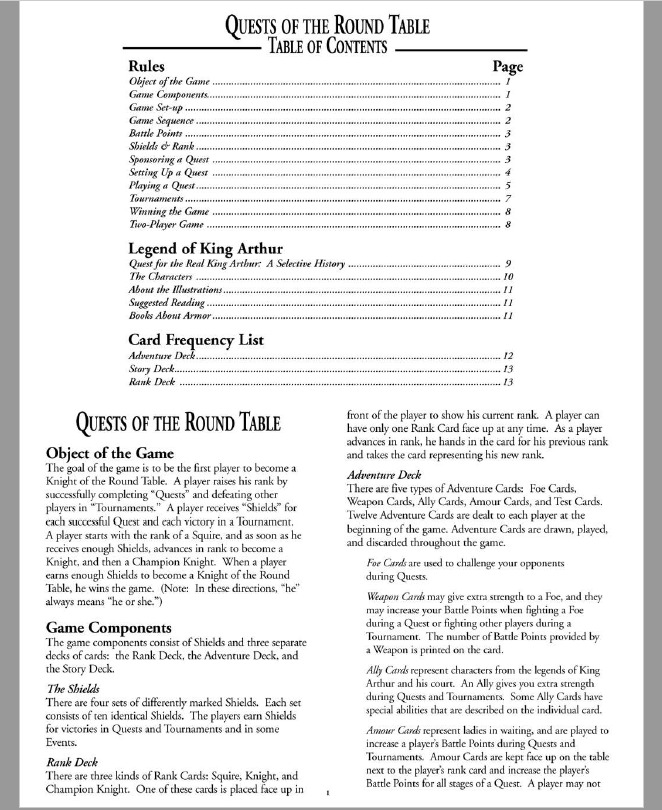
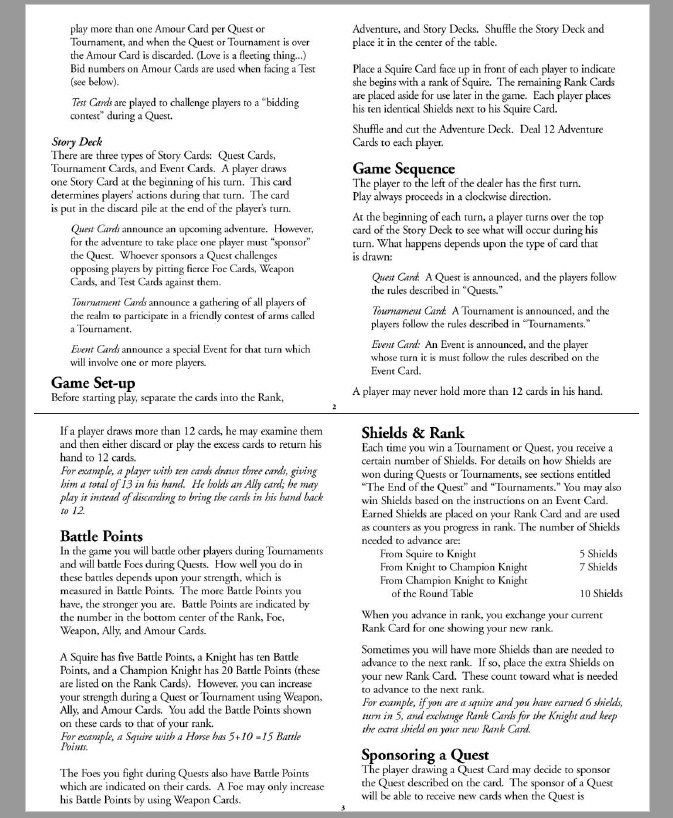
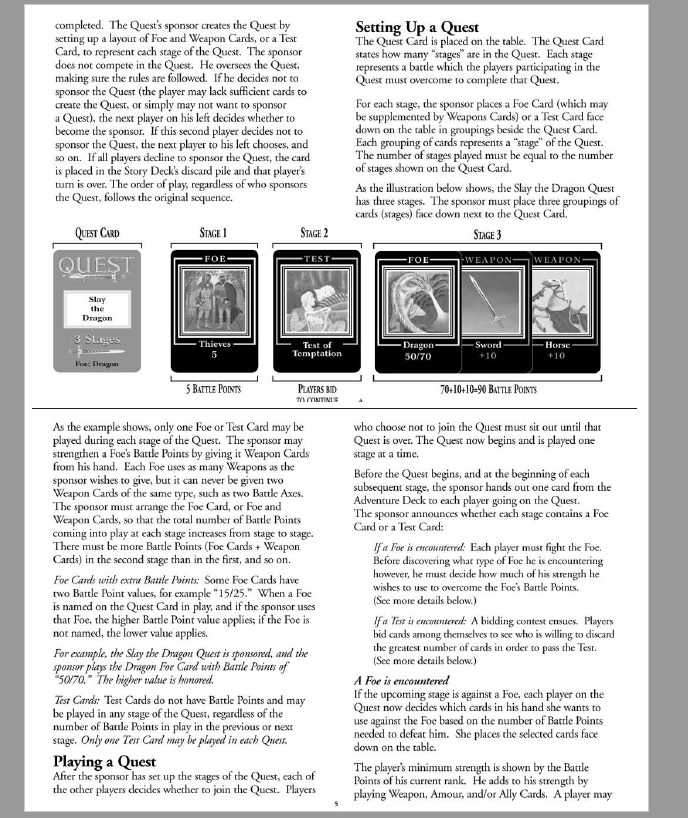
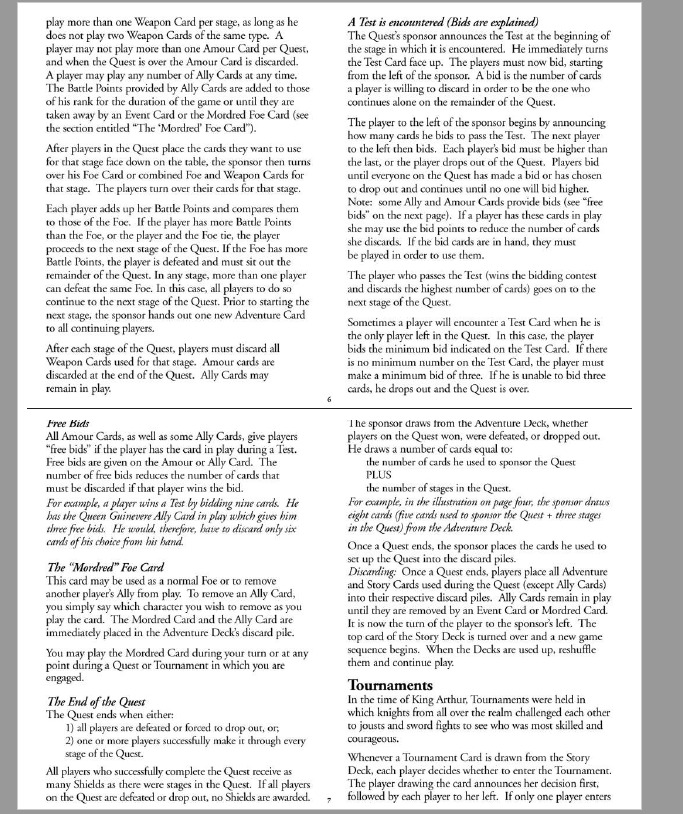
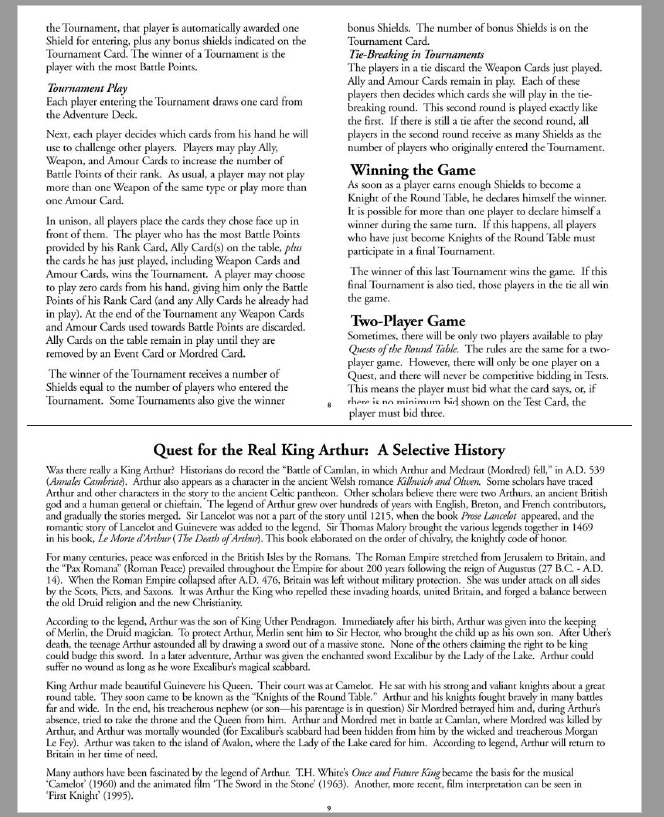
****

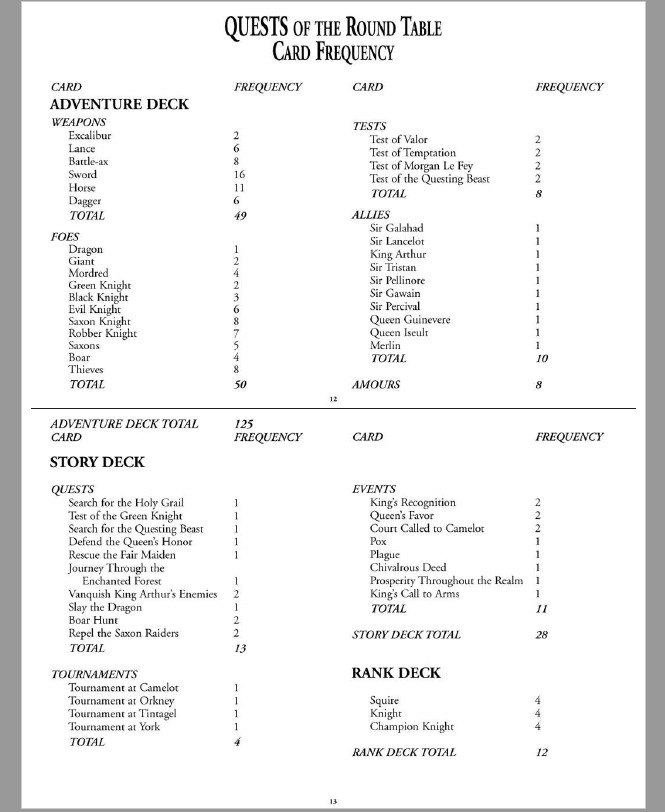
****

****

****

****

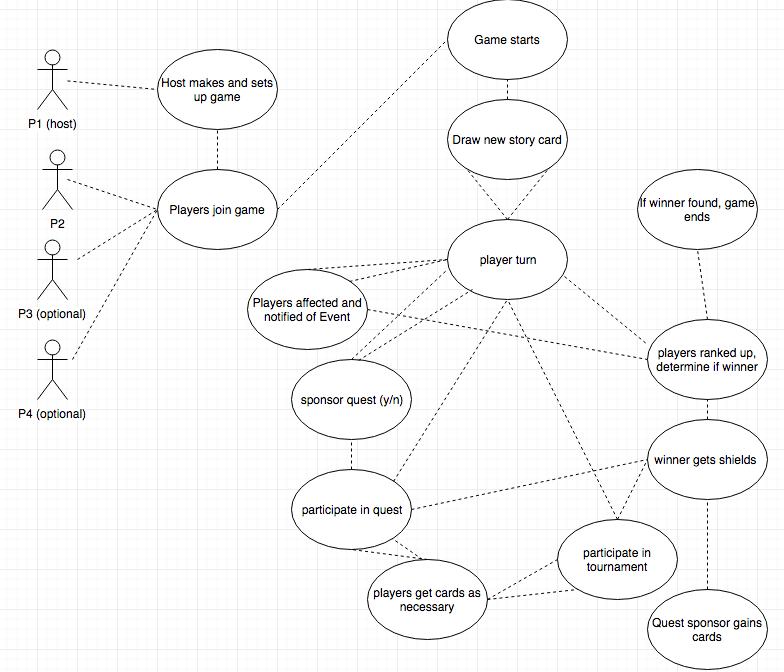
****

****

**Assumptions**

|  |  |  |
| --- | --- | --- |
| **ID** | **Assumption** | **Justification** |
| **A-01** | The game must have minimum two players when created by host, max four. Can only start when adequate number of players have joined. | It does not make sense to start a game when the correct number of players have not connected and joined the game session. |
| **A-01** | If race conditions are not enabled, the game will have all players present and responsive on their turns. | Without race conditions the game relies on player response to send back to the server and continue to the game, and will not continue till it gets it. |
| **A-02** | Whomever makes the game (the host) must start the game, only one game can be made at a time. | Only the host can start and set up the game, does not make sense for all players to be able to. |
| **A-03** | Server must be restarted to start a new game,  whether interrupting the current one, or the current one has ended with a winner. | Only one game can be set up and run at a time, multiple running would break the code. |
| **A-04** |  |  |
| **A-05** | Every player must wait their turn to complete a certain stage to be completed, if race conditions are off | The players have to wait their turns. No one can move forward to the next stage unless that of the player before has been completed or passed |
| **A-06** | If race conditions are on, players must play within the time frame or be skipped. | It is not possible to have race conditions are make the game turn based completely simultaneously. Hence we have timeouts. |

**Use Case Diagram 1: Turn Based**



**Use Case Diagram 2: Racing**

**Use Case Descriptions**

|  |  |
| --- | --- |
| **UC-01** | **Admin Initializes Server** |
| **Description** | Host makes and sets up game. |
| **Actors** | Host/First player |
| **Triggering Event** | First player initializes the start of the game |
| **Pre-Condition** | First player / host has successfully connected to server |
| **Main Sequence** | 1. Host connects and enters name 2. Host specifies number of players and game room name 3. Host make game joinable |
| **Post-Condition** | Other Players Join Game |
| **Resulting Event** | Waits for other players to connect to the game |
| **Alternative Scenarios** | N/A |

|  |  |
| --- | --- |
| **UC-02** | **Players join game** |
| **Description** | Other players join game host has made |
| **Actors** | All players not host |
| **Triggering Event** | Host makes and sets up game |
| **Pre-Condition** | Player successfully connected, game room exists |
| **Main Sequence** | 1. Players join game room |
| **Post-Condition** | Required number of players have joined |
| **Resulting Event** | Game starts |
| **Alternative Scenarios** | N/A |

|  |  |
| --- | --- |
| **UC-03** | **Game starts** |
| **Description** | Other players join game host has made |
| **Actors** | Host |
| **Triggering Event** | Host clicks start game |
| **Pre-Condition** | Correct number of players in game room |
| **Main Sequence** | 1. Host clicks start game 2. Set up screen, can rig cards for each player, ranks, shields 3. Can choose racing conditions for game or turn based 4. Accept options and start |
| **Post-Condition** | None |
| **Resulting Event** | Draw new story card |
| **Alternative Scenarios** | N/A |

|  |  |
| --- | --- |
| **UC-04** | **Draw new story card** |
| **Description** | New story card drawn from deck, shown to all players |
| **Actors** | All players |
| **Triggering Event** | Game starts |
| **Pre-Condition** | None |
| **Main Sequence** | 1. Card drawn from story deck 2. Remove from story deck, added to discard story deck |
| **Post-Condition** | None |
| **Resulting Event** | Player turn |
| **Alternative Scenarios** | Story deck empty, in which case discard story deck shuffled, added to story deck to fill it again, discard deck reset |

|  |  |
| --- | --- |
| **UC-05** | **Player turn** |
| **Description** | Determining who’s turn it is, i.e. who drew the story card |
| **Actors** | All players / current turn player |
| **Triggering Event** | Draw new story card |
| **Pre-Condition** | Successfully drew story card, winner not yet found |
| **Main Sequence** | 1. Based on player turn, do on of multiple events, asking player who’s turn it is first |
| **Post-Condition** | None |
| **Resulting Event** | Event, Sponsor quest, participate in quest, tournament |
| **Alternative Scenarios** | Story deck empty, in which case discard story deck shuffled, added to story deck to fill it again, discard deck reset |

|  |  |
| --- | --- |
| **UC-06** | **Event** |
| **Description** | Event card drawn, players affected by it lose/gain shields, cards, etc |
| **Actors** | All players affected by event |
| **Triggering Event** | Player turn |
| **Pre-Condition** | Story card drawn last was an event |
| **Main Sequence** | 1. Players shields, cards, allies in play, etc will be appropriately modified by the event 2. Players will be notified of what event they were just affected by, with the changes showing up either in their hand or scoreboard |
| **Post-Condition** | None |
| **Resulting Event** | Player turn, players ranked up |
| **Alternative Scenarios** | N/a |

|  |  |
| --- | --- |
| **UC-07** | **Sponsor quest** |
| **Description** | Offer sponsorship of quest |
| **Actors** | Current player (by turn) |
| **Triggering Event** | Player turn |
| **Pre-Condition** | Story card drawn last was a quest |
| **Main Sequence** | 1. Offer sponsorship of quest to player, if player wants to, check if they can in fact sponsor it 2. If yes, ask them for stages until valid 3. If no go to player turn then ask the next player for sponsorship until sponsor is found or all players have said no |
| **Post-Condition** | Sponsor found or all players asked |
| **Resulting Event** | Player turn, participate in quest |
| **Alternative Scenarios** | N/a |

|  |  |
| --- | --- |
| **UC-08** | **Participate in quest** |
| **Description** | Sponsor found, ask other players if they want to join quest |
| **Actors** | All players not sponsor |
| **Triggering Event** | Player turn, Sponsor quest |
| **Pre-Condition** | Story card drawn last was a quest, sponsor found successfully |
| **Main Sequence** | 1. Ask current player by turn if they want to join 2. Ask next player 3. If more than one participant, run quest 4. Else, no quest run 5. Sponsor gets cards |
| **Post-Condition** | None |
| **Resulting Event** | Player turn, winner gets shields |
| **Alternative Scenarios** | N/a |

|  |  |
| --- | --- |
| **UC-09** | **Players get cards** |
| **Description** | Players get cards as necessary |
| **Actors** | All players participating in quest/tournament (not sponsor of quest) |
| **Triggering Event** | Participate in quest, participate in tournament |
| **Pre-Condition** | Story card drawn last was quest or tournament, and players are participating |
| **Main Sequence** | 1. For tournament, all participants get one extra adventure card, then one again if they are in the tie 2. For quest, all participants (not sponsor) get one extra adventure card per stage |
| **Post-Condition** | If more than 12 cards in hand, player must discard |
| **Resulting Event** | Participate in quest, participate in tournament |
| **Alternative Scenarios** | N/a |

|  |  |
| --- | --- |
| **UC-10** | **Participate in tournament** |
| **Description** | Players join tournament and tournament is run |
| **Actors** | All players (then, all players that want to join) |
| **Triggering Event** | Player turn |
| **Pre-Condition** | Story card drawn last was a tournament |
| **Main Sequence** | 1. All players asked at once, regardless of player turn. But player turn still maintained for next story card. 2. If player decides to join, tournament is run 3. Tournament run |
| **Post-Condition** | None |
| **Resulting Event** | Winner gets shields |
| **Alternative Scenarios** | N/a |

|  |  |
| --- | --- |
| **UC-11** | **Winner gets shields** |
| **Description** | Winner(s) of quest / tournament get shields for winning |
| **Actors** | All winner of quest / tournament |
| **Triggering Event** | Participate in quest, participate in tournament |
| **Pre-Condition** | Quest / tournament completed |
| **Main Sequence** | 1. If no winners (quest), do nothing, else 2. Add shields for all winners |
| **Post-Condition** | Shields a player has, has been modified |
| **Resulting Event** | Sponsor gains cards (if quest), players ranked up determine if winner |
| **Alternative Scenarios** | N/a |

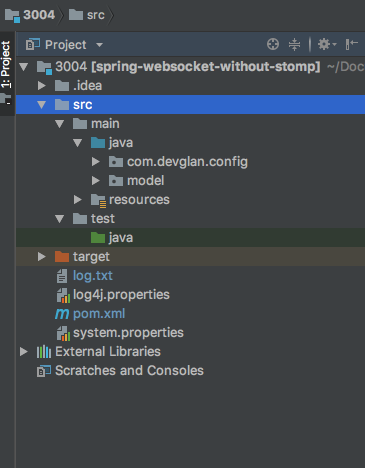
|  |  |
| --- | --- |
| **UC-12** | **Quest sponsor gains cards** |
| **Description** | Sponsor gains adventure cards += cards used + number stages in quest |
| **Actors** | Quest sponsor |
| **Triggering Event** | Winner gets shields |
| **Pre-Condition** | Quest completed |
| **Main Sequence** | 1. Sponsor gains cards 2. Discards if cards in hand > 12 |
| **Post-Condition** | Cards in hand for sponsor heavily modified |
| **Resulting Event** | N/A |
| **Alternative Scenarios** | N/A |

|  |  |
| --- | --- |
| **UC-13** | **Players ranked up, determine if winner** |
| **Description** | Players ranked up, winner found if anyone is now K.O.T.R |
| **Actors** | All players |
| **Triggering Event** | Winner gets shields, Players affected event |
| **Pre-Condition** | Tournament/Event done, or Event drawn |
| **Main Sequence** | 1. Calculate new ranks based on shields gained 2. Determine if anyone is winner, i.e. if they are now a knight of the round (K.O.T.R) |
| **Post-Condition** | Winner not found |
| **Resulting Event** | Player turn, Game ends (if winner found) |
| **Alternative Scenarios** | Winner found |

|  |  |
| --- | --- |
| **UC-14** | **Game ends** |
| **Description** | If winner found, game ends |
| **Actors** | Winning player, all players |
| **Triggering Event** | Players ranked up |
| **Pre-Condition** | Player has reached K.O.T.R (knight of the round) |
| **Main Sequence** | 1. Winner notified he/she won, other players notified who won |
| **Post-Condition** | Game ends |
| **Resulting Event** | None |
| **Alternative Scenarios** | None |

**Architecture Explained**

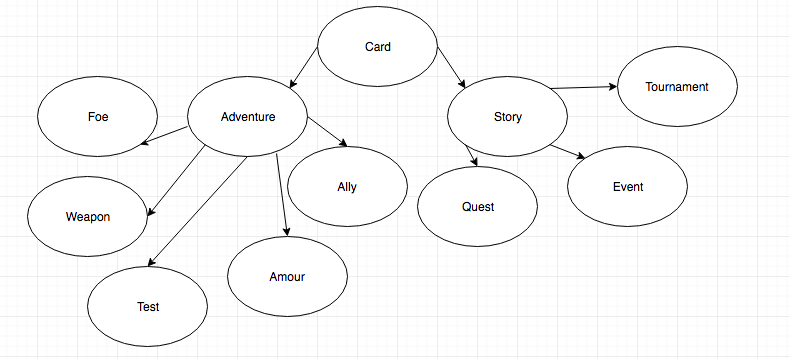
See picture below for folder layout:

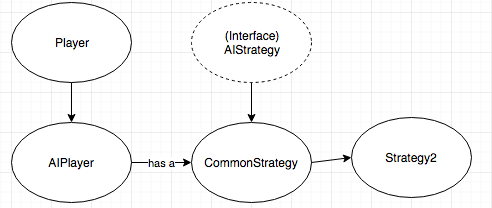


Resources, Target, com.devglan.config (Arun & Naseer)

**Model Architecture (backend) :**

The backend structure of the code was created with the usability benefits of polymorphism in mind, with classes appropriately divided to avoid excess typecasting. It was also written to well with the MVC pattern which we use, as well as the Strategy Pattern which was used for AI (although AI is not implemented, it’s pattern use and documentation will be shown here). For example, while Foe and Event are both subclasses of Card, they belong to different parents, Adventure and Story respectively. See figures below.





First we will touch on the AI structure even though it is not actually part of the game, the code is there. AIPlayer is a subclass of player so it could be added to the existing logic in many cases quite seamlessly. AIPlayer needed to be a separate class as AIPlayer has many attributes to function properly that a regular human player does not. AIPlayer has a CommonStrategy, which is an abstract class that implements AIStrategy interface. Strategy2 is a subclass of CommonStrategy and subtype of the interface. Commong strategy contains a lot of functionality that would be prevelant across both strategy1 and strategy2 preventing a large amount of code reuse, will Strategy2 only has functionality in it that, generally speaking, is exclusive to that particular strategy and can’t be used by another strategy. Strategy1 is not written, but the plan was once it was written to use a simple factory to instantiate objects of the respective strategies for each AIPlayer that is made.

Back to the other parts of the game, as you can see now by the picture the divison of Card into adventure and Story prevents the needs for excessive typecasting, because instead of all the cards being in once deck they can be split into two decks, housed in GameBoard class, Adventure and Story decks respectively. This also maks sense from a game logic point of view.

GameBoard in the main class in the backend and contains the vast majority of back end functionality, however it is not burdened by having to do too much. This is because most of its work is pure logic aided by various attributes and states in other classes. For example, when dealing with events, the respective event cards tell the GameBoard which players should be affected by the card based on event attributes, then GameBoard finds them. It does not have to determine who’s affected AND find them, which would be clunky and too much code for one class. Additionally, when the server needs something from the back end, it provides all necessary information for GameBoard to stick purely to its purpose. For example, the server will provide the indexes of selected cards for a player, that player which was found by web socket session, and whether it wants to validate cards or not. GameBoard simply has to execute the desired commands, it does NOT have to deal with the main control flow of the game after.

That’s not to say it does not have functionality for control flow, but that functionality is to help the server manage the control flow, and not for GameBoard in any way to deal with that itself, because if it did it would once more overburden itself. The idea to separate our game into parts like this game from Spring with uses D.I. (dependency injection) and I.O.C (inversion of control) to decouple crucial elements.

While we did not end up relying on beans, we still did use the principles of D.I. to decouple core components of our game, so we can pull and push and work on different parts of the game separately without having to worry about the other side of the functionality and whether what we were doing would break that part of the game or not.

Interaction diagrams for server client : Arun & Naseer