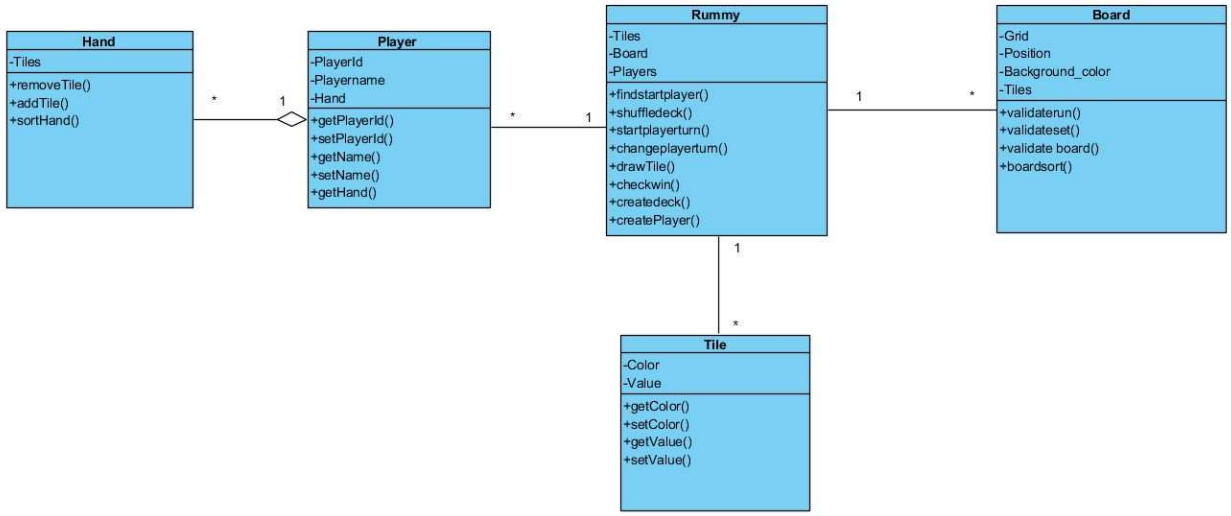
1 Introduction

In order to begin programming the Rummy-O game, it is important to first conceptualize the software design through diagrams and models. The design of this project was derived from four main types of UML diagrams; Class Diagram, Sequence Diagram, State Diagram and Component Diagram. These diagrams were extrapolated from the collective information gained from user stories, use cases and requirements. Having the four diagrams allows for the development of the project to be more streamlined and makes it easier to spread the work amongst the group members. (All the diagrams are located in the "Design" folder of the repository).

2 Class Diagram

A class diagram shows the interaction between the classes of a software project through relationships.

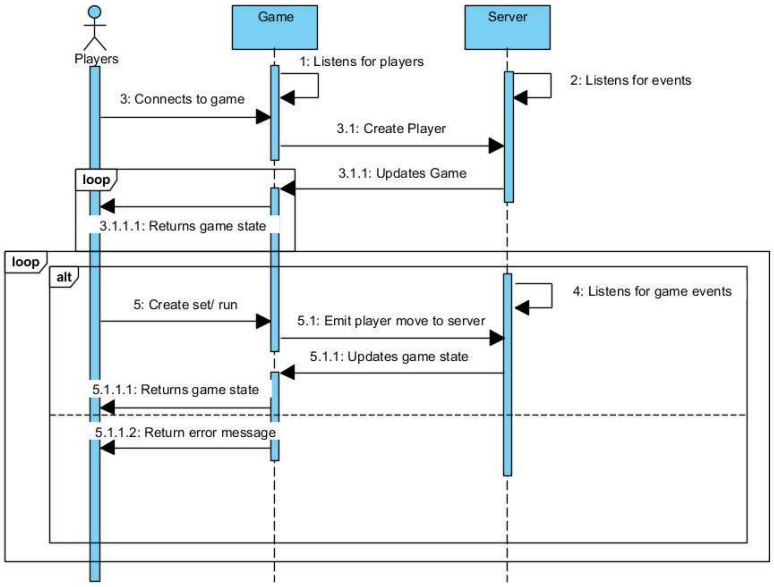


There are five classes used for the creation of the Rummy-O game as seen in the diagram. The Rummy class is the main driver class which communicates with all other classes. The Tile class defines a tile object with a colour and numerical value as well as its visual design. The Board class stores the current state of the game on a grid. It is updated every time melds are made and/or modified by players or tiles are drawn by players. Finally, the Player class handles all the functions that allow a player to interact with the Rummy-O game. It also contains the Hand class which stores all the tiles that a player has in their hand. The state of the Hand class is updated when a tile enters or leaves a player's hand. The adding or removing of tiles is governed by the Board class.

3 Sequence Diagram

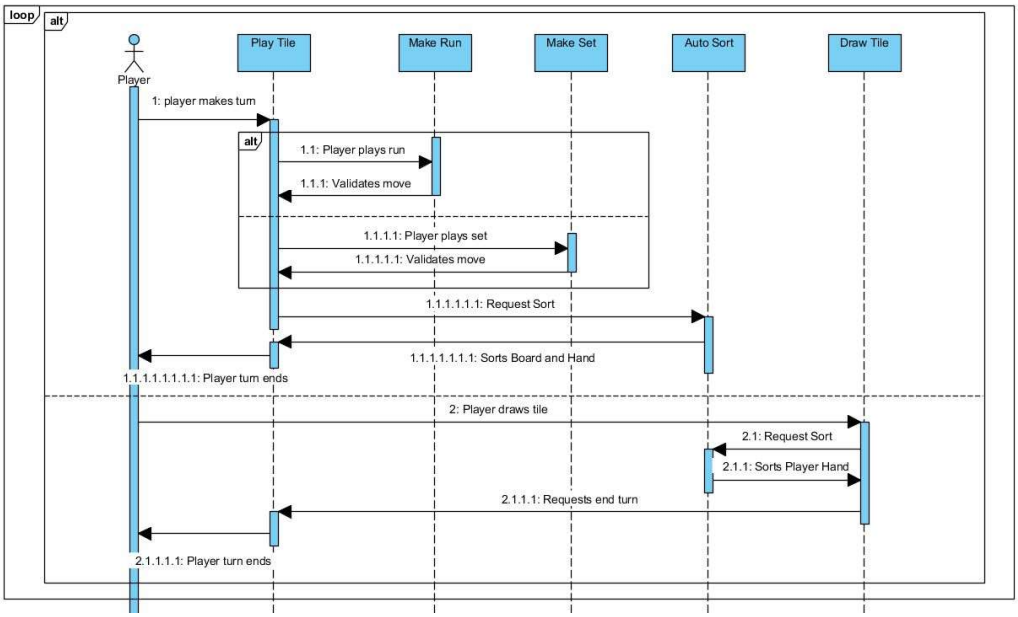
A sequence diagram depicts the interaction between objects in the order which these interactions take place. Follow the arrows in their numerical order to trace the sequences.

3.1 System Level Sequence Diagram:



This sequence diagram depicts the network structure; i.e. how the game communicates from the player's browser with the server. The game and server are constantly listening for events governed by either the player inputs or the game processes respectively.

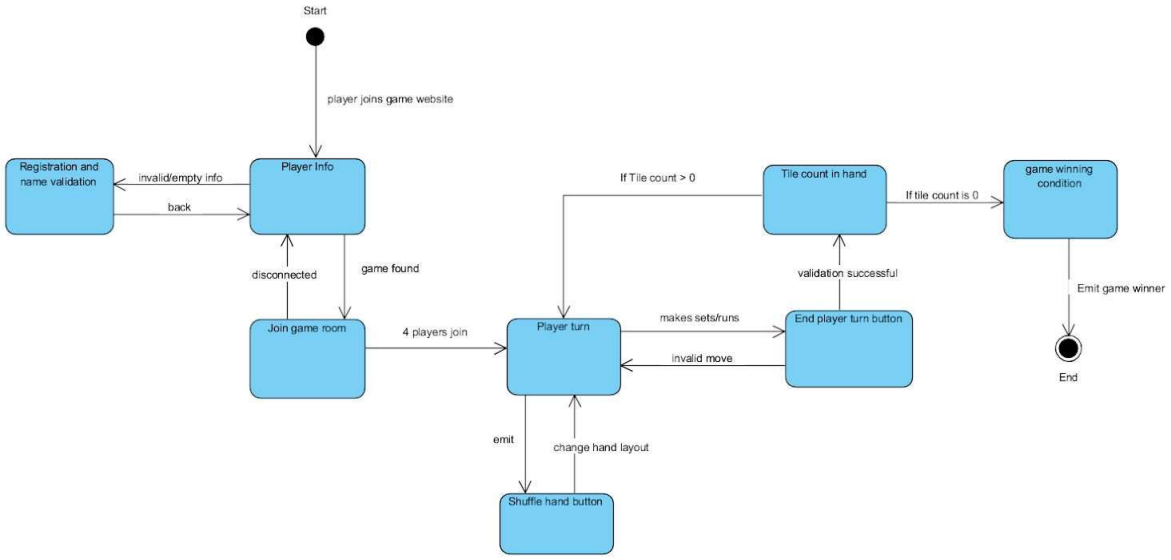
3.2 Class-Level Sequence Diagram:



This diagram displays the gameplay sequence for each player. Each turn will consist of the players making runs or sets which are validated by the board class to ensure legal plays. At the end of each turn the auto-sorting method is called for the players hand as well as the game board. Once a legal move is made or the player draws a tile. Their turn ends and the sequence is complete.

4 State Diagram

State diagrams show all of the different states that a software program can take and change to during its execution process.

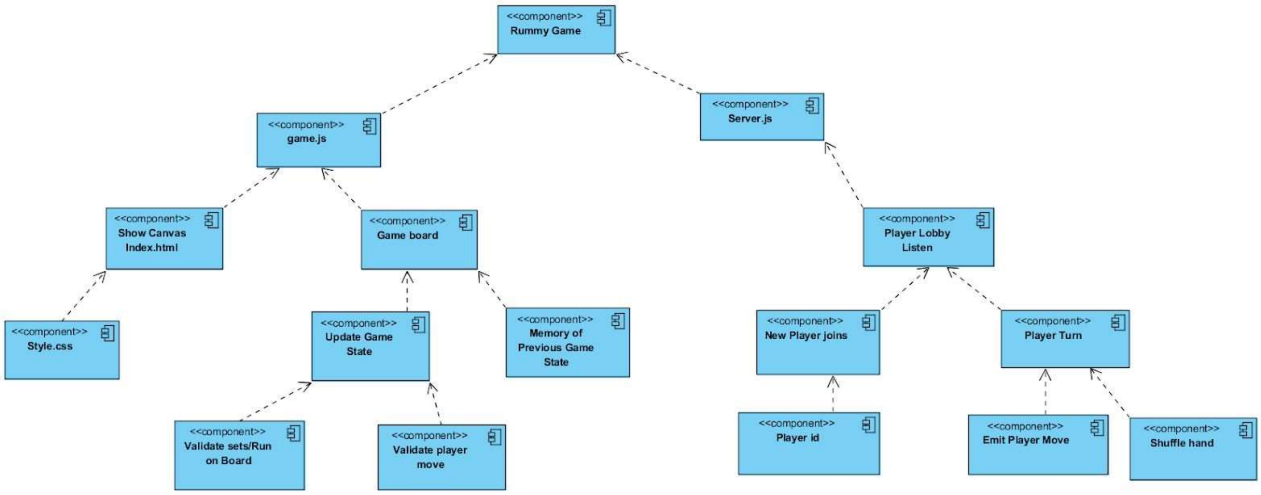


The game starts at the start dot and upon a player joining the game switches to the Player Info state. The player enters their required info and the game switches to the validation state. Once verified, the game begins finding other players during the Join Game Room state. When a game is found the game finds which players will take a turn first and then it switches to Player turn state. The player's inputs can cause the game to two distinct states of either shuffling the player's hand or ending the player's turn. Once a player's turn ends, win conditions are checked to see if a winner is found. If a win condition is found, the winner is emitted to each player's browser and the game ends.

5 Component Diagram

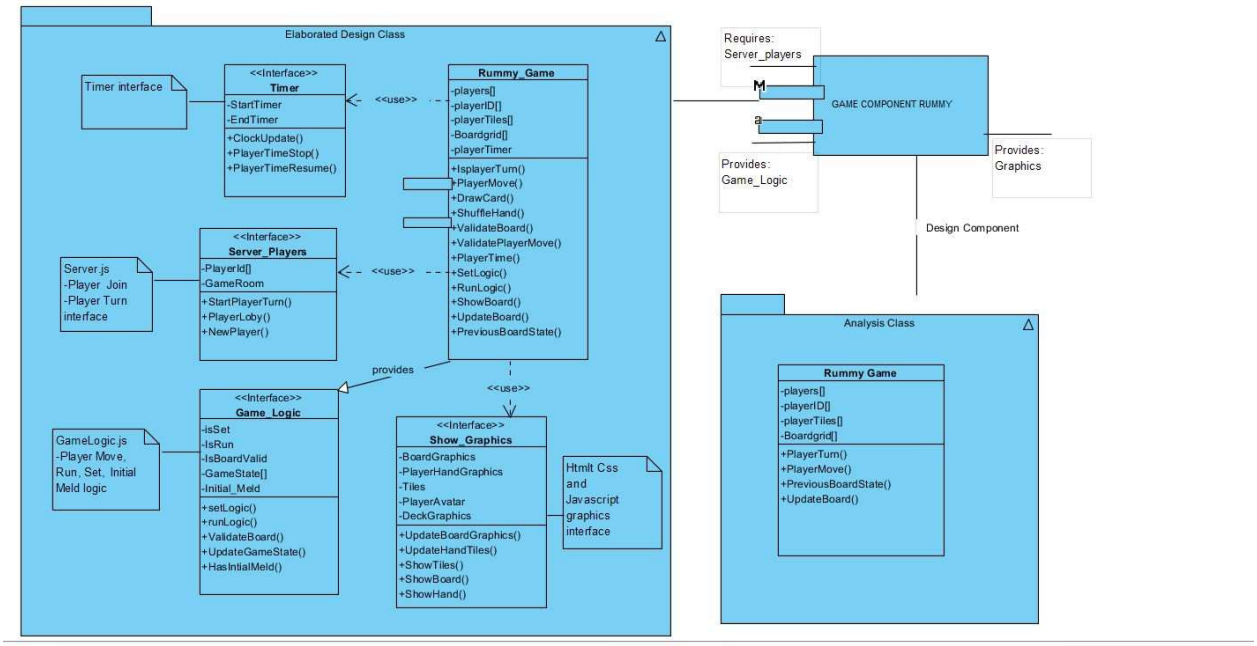
A component diagram describes the organization and wiring of the physical components in a system.

5.1 Basic Component Diagram



This component diagram shows the structure of the game and server scripts of the Rummy-O game.

5.2 Object Oriented View - Component Diagram



This component diagram elaborates the class designs and the responsibilities of each, while providing a visual diagram to follow while coding.

Conclusion

In conclusion, all these above diagrams assist in visualizing the project as well as help in understanding the minor responsibilities which need to be incorporated. Class and Sequence diagrams conceptualize the interaction between the game and the player. While the State diagram clarifies the communication linking the game and the server. Together these diagrams explicitly illustrate a simple and concise flow that will alleviate confusions while creating the game.