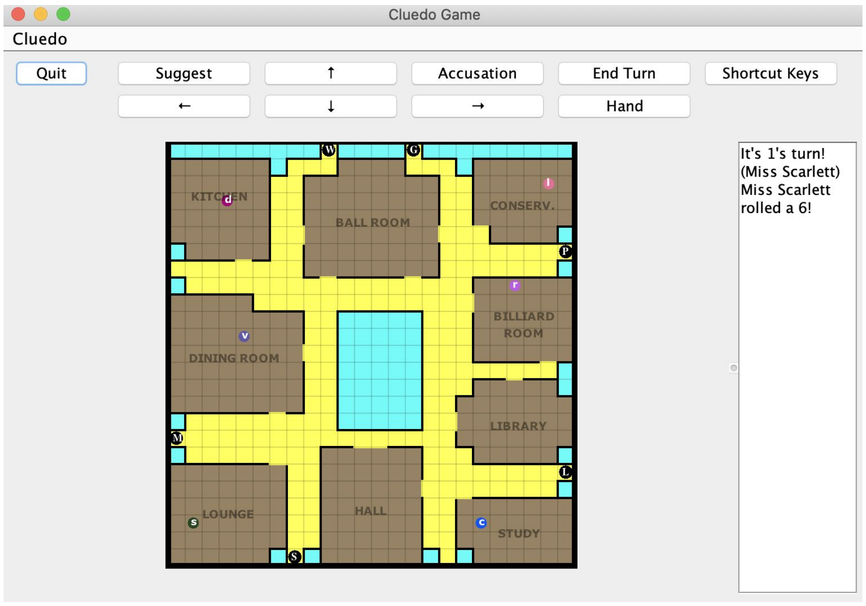
Reflection

GUI Discussion:
This is a brief look of our GUI design.



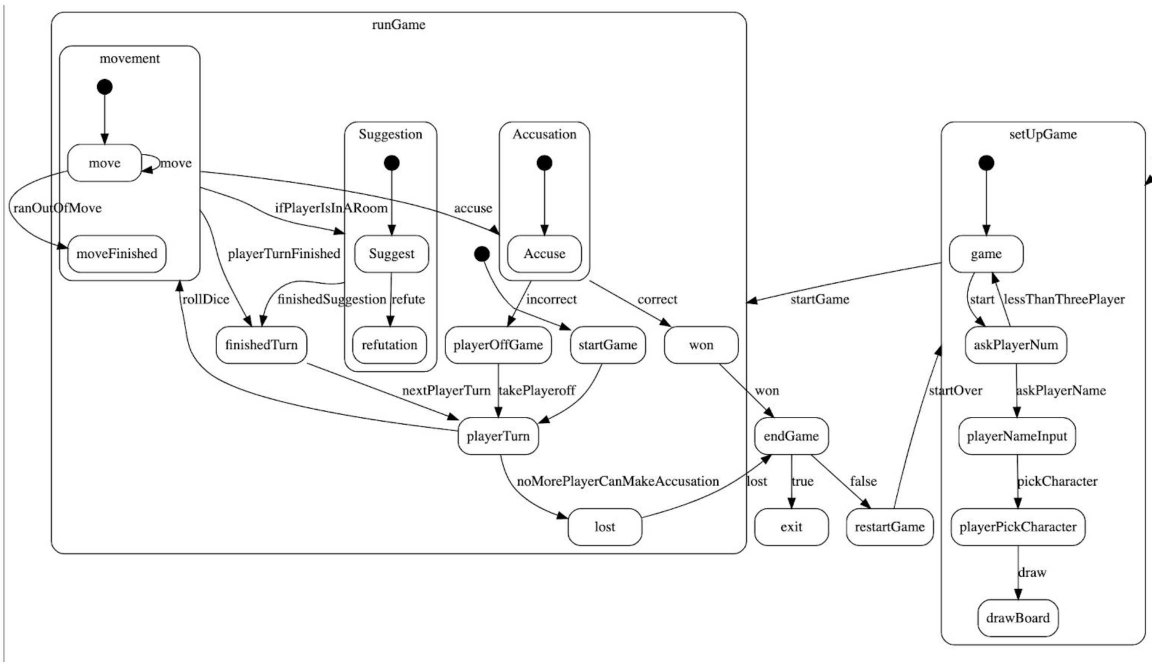
```
It's 1's turn!
(Miss Scarlett)
Miss Scarlett
rolled a 6!
```

Graphic output is made by MenuBar which is made by the JMenuBar and used by JPanel to create the initial panel which will add operators and other components to it. Operators are simply few Jbuttons which make players able to operate the game by adding an action listener to send requests to the model constructor. In our case, there is a String object in the board object in the game data class that receives messages from the controller.
We use JOptionPane to get information from players (players can choose from multi-options)such as getting player number, name and character or getting players' suggestions and accusations. The JDialog was used when players are trying to close the game to seek confirmation. We also added shortcuts which the game will operate using the keyboard using the keyStroke to get the key input and save it into the JButton object using getInputMap.put and save the action into the JButton using getActionMap.put . When JButton “shortcut keys” is pressed , it will come up with an instruction to show how to use them. Also there is a textOutPutArea which tells players the state of the game, such as the rolling dice number or show player's hand card, it is done by using JTextArea.

Design Discussion:
In the previous code, we have already implemented the GUI model. But we do have to modify it , one thing that really bothers us is that our code did not implement the MVC framework well so when we are modifying , bunch of codes are stuck together,when we change one little bit of the code, it will has impact to other part of code that are unnecessary to be changed . The code should be separated into 3 different parts: the model , view and controller so when we are modifying the view part (in our case, the draw method in class board) , it should not affect the other parts of the code. So we created a new class called gameData which represents a database instead of saving all the data inside the board, so when we are modifying the board , it will not affect datas stored inside the board class, now the board is also an object that is stored in the database.

A good design of the code that made it easier to add GUI is that, we manage to make GUI (which representing the controller part) to just send request instead of changing the model and data(board , players , cards) of the game, it will pass on the string I mentioned in the GUI Discussion which pass the string information representing request .for instance , for making character to move up, the controller send the String object “w” representing moving up , and then the model constructor receives the String and update the character position and then pass it on to the view part to update board. To make connections between each component model, a single request will do the job. And that will reduce the work we do, for example, to create a shortcut in the game, we do not have to reimplement the code, we can just send the same request as before to represent the requirement that we want. To make designing GUI easier, the main thing is to separate each part of the code to make the code easy to modify and manipulate.

State Diagram:
This is how our state diagram looks like.



The whole state diagram contains two nest states, setUpGame and runGame. The setUpGame contains everything needed for setting the game up including asking players’ number, players’ name input and their character input, after information is collected, we draw the initial board and finish setUp and start runGame.
The cluedo game is about each player taking turns to move and make suggestions and accusations. So after the game is started the first state is playerTurn which when every player finishes their turn and returns to it then the next player takes his turn until the game is won or lost. The next state after playerTurn is a super state movement, inside movement the player keeps moving until he runs out of moves. When the movement is finished, if the player is inside a room, then he can make a suggestion. The super state contains 2 states which when a suggestion is made the other player has to make refutations if they can. Also if a player ran out of move and is not in a room, he can either finish his turn or he can comes to the next super state accusation, if it’s correct means next state is won and the game is

won , if it’s incorrect then the player is off the game (can not make accusation anymore) then returns to playerTurn state which means the next player’s turn. When no more players can make an accusation which means the game is lost.Won or lost the game will both lead to the state endGame. Then the player can choose if he wants to restartGame which will bring him back to setUpGame state , if he does not want to restart, he can exit the game（exit state）.