Reflection

assumptions :

1. I assumed that it is a game designed for multiplayer playing on one platform. which means multiple people are using one device to run and take turns to play the game we design.
2. The initial players' position cannot be changed, if there are 3 players playing the game, the other characters that were selected will also be appear on the board, just cannot be moved.
3. Even the murderer himself would not know if he is the murderer or not because the murder circumstances are randomly formed.
4. A player can make accusation no matter where the player is inside a room or not, but suggestion have to be made inside a room.
5. A suggestion and accusation have to be made when there is no move steps left, cannot do them during middle of moving.
6. There is no causal relationship between suggestion and accusation. A player can make an accusation even if no suggestion has been made before.Suggestion just gives players a clue.

CRC cards

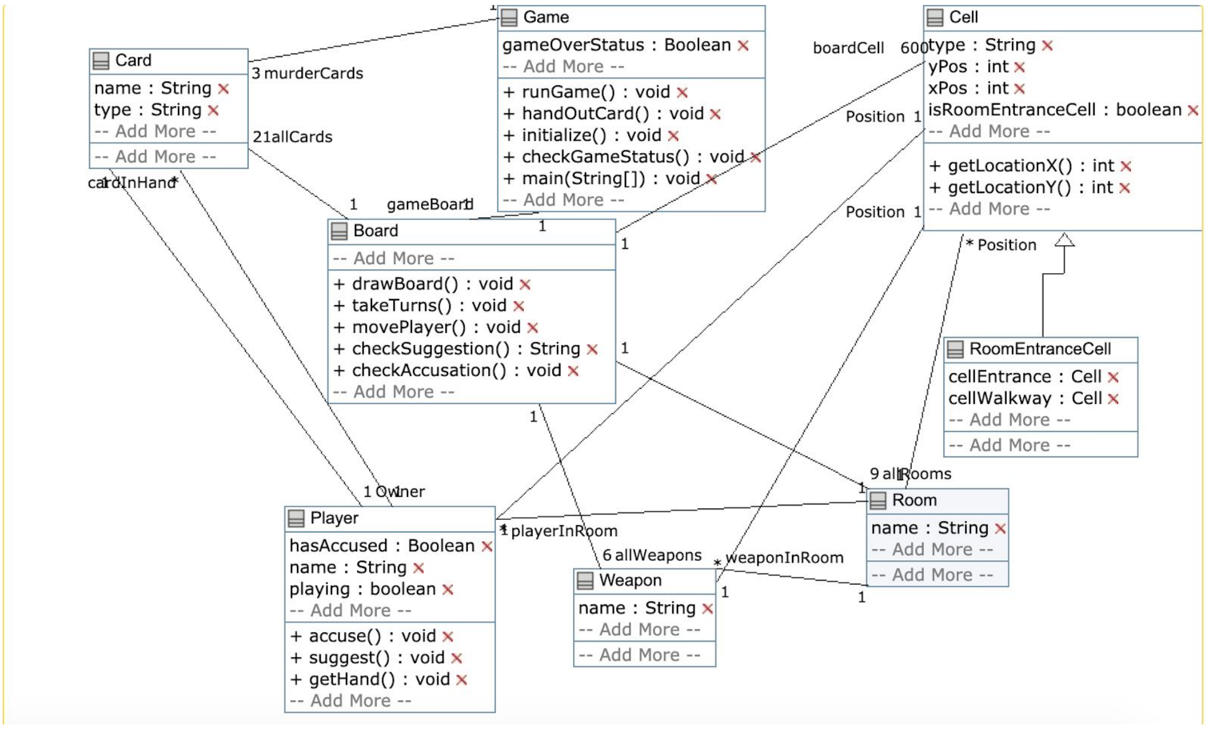
We created our CRC card according to textual analysis description as what we learned in the lecture to create the initial CRC cards .

The usage for CRC Cards helped::

1. The initial CRC cards gave us an initial overall view of the game that we are about to design.and helped us with our class diagram to complete our design
2. Me and my group member Yun can communicate our idea more easily by making changes to CRC cards and as we both have different opinions on the design, sharing ideas and debates make our design more complete.
3. The CRC cards isn't very complete. The responsibility part helped us to create new class to complete our design. For example, our initial CRC cards contains class "Board" and it has responsibility to "Aggregate the location" which is correspond to coordinate. I cannot just store 24*25 of x coordinate and y coordinate(which is 24*25*2) in the board class.So I create "Cell" class to store location also have responsibility of containing players and weapons.
4. Also the collaboration part helped me with building class diagram. For example,the Player class collaborate with Card and has responsibility of knows cards in hand. So when I am making class diagram I can simply make an association between Player and Card without overthink.
5. Works the same with coding when implementing method and making fields.The Responsibility and Correspond part can give clear view of what method do I need to implement and save time from wasting time. for instance, when I make accusation and suggestion part of the code, we have pre thought about this when building up CRC cards, we wanted to make accuse and suggest method in the Player class and check it if matches in board class , so I was coding, I can simply start code without rebuild constructor.

Game Logic of Class Diagram

Here is the Class Diagram of the game:



Game Class : Game class is to initialise every field needed to create a game. Including initialising everything needed to create a board and use them to create a board Object and make the board to draw. After the board is drawn , start to run the game (runGame method), which lets players start to move around and make suggestions and accusations(takeTurn method) until the gameOverStatus store in the field is true.
Board Class: Board class include a lot of information including 24*25 2D array of Cells, List of allPlayer,list of all Weapon,list of all Rooms, list of All cards,list of Murder Circustances cards, list of room entrance cell and list of player who is playing. It has a method to draw boards using these information and has taketurn() method make players take turns to behave which are used in game class, take turn method are called in the game class . Check accusation and Suggestion are in the board class , if accusation is correct, gameOver Statue will be true. All the changes will apply to board's field immediately and redraw.
Cell class : determines positions ,can be used to draw board, movePlayer etc. has subclass RoomEntranceCell.
RoomEntranceCell Class: determines positions for room entrance, be used to determine door position.
Player Class: store player's name , 1 Cell to determine the position and a list of cards in hand. The accuse and suggest methods are in the player class which return a list of Cards and will be called and check if it is correct when running the game. (checker method on board).
Weapon Class:stores weapon name and 1 Cell to determine the position.
RoomClass: contains the name of the room and players and weapons in the room.
Card Class:Contains the name and type of the card, can be owned by the player to refute suggestion, check accusation etc.

Contribution && Chanllenge::
CRC cards were first created by my groupmate Yun(there are only 2 of us because we couldn't contact the third member), Then we discussed and exchanged opinions to create the initial CRC

cards, then I created the class diagram using CRC cards.After we finished coding, we updated the class diagram and CRC cards together.
For the coding bit ,Yun initialised the field needed to initialise the game such as murder circumstances and dealt card to player then he created the GUI class for the game then we made the constructor of game class together, I initialised every object needed for board and did the constructor bit for the board to draw , take turns,moves and accusation,suggestion bit in Player class and checker method in board. We both pretty much did half of work.
The most challenging part is initialises set up ,make constructor and run game method because the Game Class’s fields can not synchronous with Board class. So to change board statue ,we might need to create new board over and over. So,we decided to just make changes in the Board to make it convenient(fields in Game is just for setup) this is why the take turn method is in the Board class. When implementing the board drawing, all the positions really bothered me ,how to determine the door position was a challenge to me but I came up with a way to create a RoomEntranceCell class as a subclass to Cell to store the door’s position. The other classes are mostly field to store objects and get, set methods which might mostly be auto created by umple .