SWEN 225
Assignment 1 Reflection

The first thing that we did was create the crc cards to plan out how we want the game to be set up. We had a pretty good idea of how we wanted the game to be set up so the crc cards were followed almost identically. Once the crc cards were complete because the program was required to be text based, me and Jesse started off by creating a concept for the text based output of the board/grid. Based on the concept I created a board class and a toString() method inside of it to display the output of the board to the terminal to give the player a visual representation of the game. At this point all I had was a 24 x 24 text grid, to complete the game's output I needed to add the estates, grey areas and the characters into the board. I created a class for each of those objects, and gave them the bare minimum code required to run (such as coordinates and names) as at this stage I just needed them to display an output. The toString method that I created was created in an adaptable way meaning that nothing was hard coded. Therefore if necessary I had the ability to easily change the size and position of any object and the output would be able to handle such changes. This also made displaying things such as moving characters easy as all that needed to happen was change the specific characters coordinates. With the board output complete I referenced the crc cards to ensure that each class was fulfilling its required tasks e.g the board class needed to contain the coordinates of everything on it, the characters on the board, the estates visible, the grey areas, and the game's current state. With this completed I had created a good base for the rest of the team to build from as I had created most of the classes necessary. As a team we split up tasks between us, my main task was to continue with UI and to also create the cluedo cards. For the UI I found that having the board print straight to the terminal was very bulky and made the game very difficult to follow as everything was cluttered together due to the board printing to the terminal every time something changes. To fix this I decided to have the board print separately in a JPanel, which allows us to display the board, update it without all the clutter in the terminal and still use the system in/out which we are required to use. Once I had completed the UI my other main task was to create the card system for the players. To do this I created a class named “Card” containing an enum state determining each card type i.e Character, Estate and Weapon. Along with the enum states I gave the class the bare minimum variables and methods like name, toString and equals. With the card base class created I added a card Set into the character class so that each player can have their own set of cards. To complete the card setup I created a board constructor and created a function that creates and adds all the cards to a large deck which then removes one card of each type to make the three murder cards and then distributes the remaining cards equally between the players. Doing this in the board constructor means that whenever a new board is created the cards are distributed before the game begins. The other large contributions I made was another UI update which rerouted the system.out to another Jpanel connected to the main board instead of the terminal, this gave us a lot of extra functionality for the guessing system as this update allowed us to easily clear the output to ensure that no information such as player cards aren’t shared between players. Besides my tasked contributions another large contribution was to the movement controls for the players, as a team we decided that there were two ways to implement movement, one would be to roll the dice and then click on the square that we want the character to move to and then calculate whether that move is possible with the number of moves we rolled. The other way was to move the character one move at a time until we run out of moves, this way we decided was the easiest way to implement the movement. To go with this decision we also decided that we should use wasd key inputs to move the characters. With my two main tasks completed I continued to make small updates and bug fixes to mine and my teammates' work up until the program was set as complete. The program may be complete and working but there are many things that I would want to see improved, especially regarding the quality of the code itself, the end result of the team's code is very unorganised and is not optimal or efficient (This is mainly due to alot . One suggestion I have regarding my own work is that I should have added methods such as board.update(), board.updateText() and board.clearText() which would essentially do as their name states. Small methods like this would have made my contribution to the program a bit cleaner and easier to follow.