Clue Game

Manali Kale, Rio Parsons

- ## Status Summary

Work Done

During the first week of our project, our team focused on developing the basic functionality of the game. We started with creating the game board and character pieces. We also worked on the ability to make suggestions and accusations as well as check for winner functionalities.

Following is the work breakdown for the Clue Project :

Functionality	Team Member
Design Game board that accurately represents the layout of the original Clue board game	Manali
Allow players to choose the characters	Manali
Randomly determine the envelope and card deck - (Implementation of Singleton Pattern)	Rio
Implementation of Cards classes - Rooms, Suspects Weapon	Rio
Implementation of Command Pattern	Manali
AI player behavior	Rio
Implementation of Strategy Pattern	Rio

Changes/issues encountered

We have encountered a few issues with the initial design we created in Project 5. One of the issues we encountered was with the game's players and moving players across the board. Currently, the game will run with only AI players. These players can make guesses and prove each other wrong. Eventually, one player will win. We also realized that the game board was too large, so we decided to reduce the size of the board to make the game more manageable and easier to navigate.

Patterns

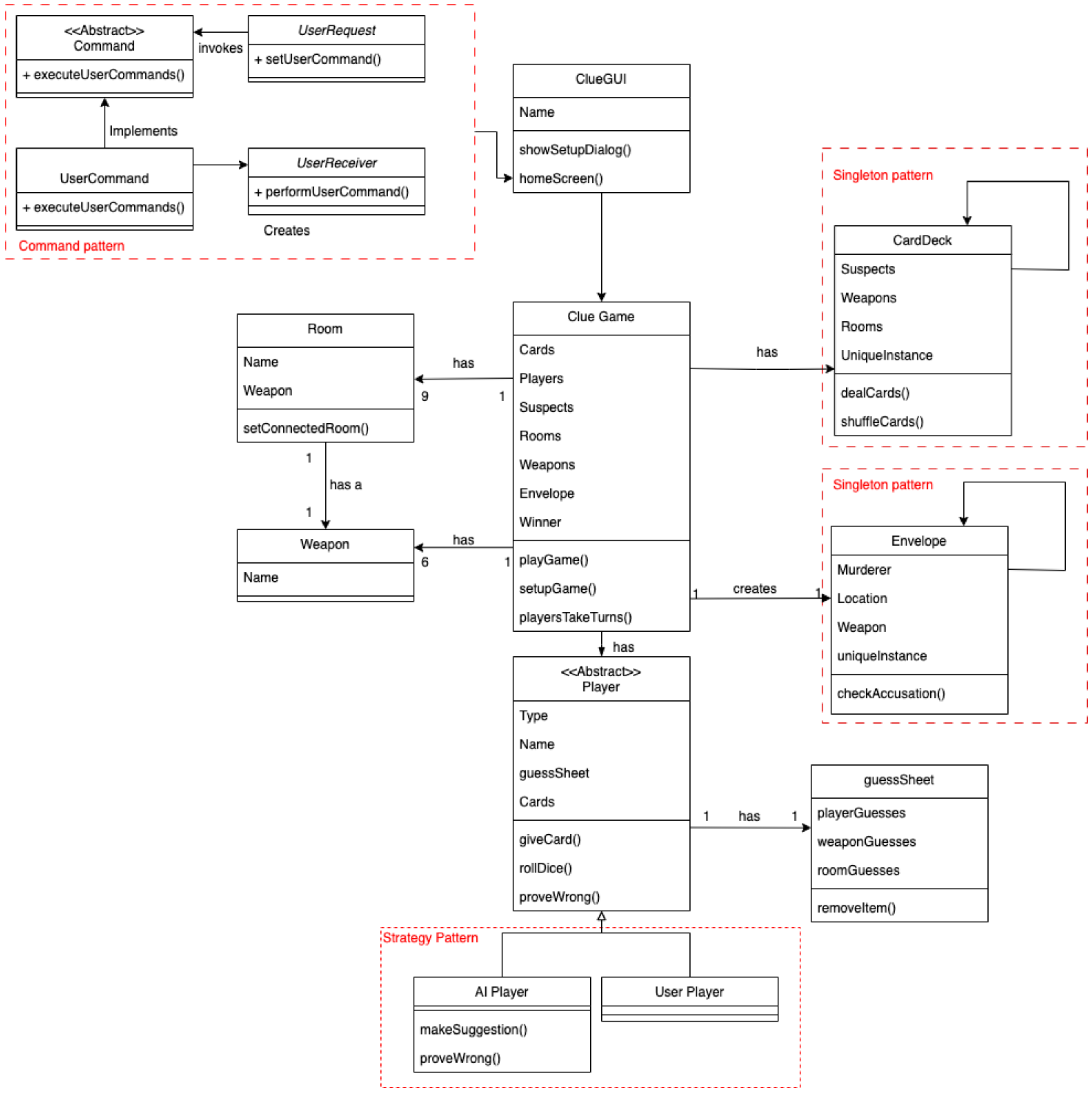
In our project, we have used several design patterns to help with the development process. These patterns have helped to improve the organization and maintainability of our code.

1. Singleton Pattern
   We have used the Singleton pattern to create a single instance of the Envelope and CardDeck classes which will be shared with all the players.

2. Strategy Pattern
   In our project, we use the Strategy pattern to define different strategies for handling player suggestions, based on the type of suggestion made (i.e., a suggestion about a character, a weapon, or a room). This pattern allows us to define different algorithms for handling each type of suggestion and swap them in and out as needed, without affecting the rest of the game logic.

3. Command Pattern
   In our project, we use the Command pattern to communicate from the user interface. Each user action in the game was encapsulated as a command object that could be executed as needed

- **Class Diagram**



- **Plan for the next iteration**

In order to finish the program, we will need to implement three general things: Player movement, user behavior, and user interface. We may also alter the algorithms for the AI players to make them more complex

Currently, the AI players can finish the game by guessing randomly. However, the players do not move. The game board is already created, so in order to allow them to move, we will need to write code to link the players to the board. We will then have to write an algorithm to decide where the AI players will move to. Additionally, we will have to find a way for the user to decide which spot they would like to move to.

The AI players are fully coded aside from moving behavior, however, the user player code has not been started. Many of the algorithms will be similar to those in the AI player code, however, they will vary due to the need for user input. The user will input the guesses they would like to make, as well as select the cards they want to show other players to prove their guess wrong.

Finally, we will need to finish the user interface, including graphics and prompts for the user to make decisions. This will include implementing the command pattern and observer pattern.

Currently, when the AI players make a guess they randomly select one item of each class (suspect, weapon, room) and make the guess. However, real players would make strategic decisions such as guessing two known entities and one unknown entity. Additionally, real players can gain information from other users' guesses. Additionally, when proving another player's guess incorrect, AI players randomly select one of their cards that could prove the guess incorrect. However, real players would make strategic decisions such as showing cards they have already shown in order to give out as little information as possible. We may add more logic to the AI player behavior, however, we do not know how much we should add. We do not want to make them so good that the user cannot win. Conversely, we do not want to make them so simple that the user always wins.