Clue Game

Manali Kale, Rio Parsons

- **Final State of System Statement**

  We have implemented a fully functional Clue Game using OOAD principles. The final version of the Clue game meets the requirements of the game. Following features were implemented :

  1. Start game, choose character and number of players
  2. Roll dice and move player
  3. Shuffle cards and assign cards to each player
  4. Randomly determine murderer, suspect and weapon and form envelope
  5. Make a guess or accusation, show card and prove guess wrong
  6. AI player and user player behaviors
  7. Maintain clue sheet to track clues
  8. Declare win or loss
  9. Implementation of Patterns - Singleton, Command, Observers and Strategy pattern
  10. Implementation of OOAD design principles

  Following changes were part of project 7 that changed from Project 5 and 6:

  1. Added Observer pattern for players
  2. Added player movement
  3. Added option to make accusation and guess and also prove guess wrong
  4. Added clue sheet to track clues
  5. Declare win or loss

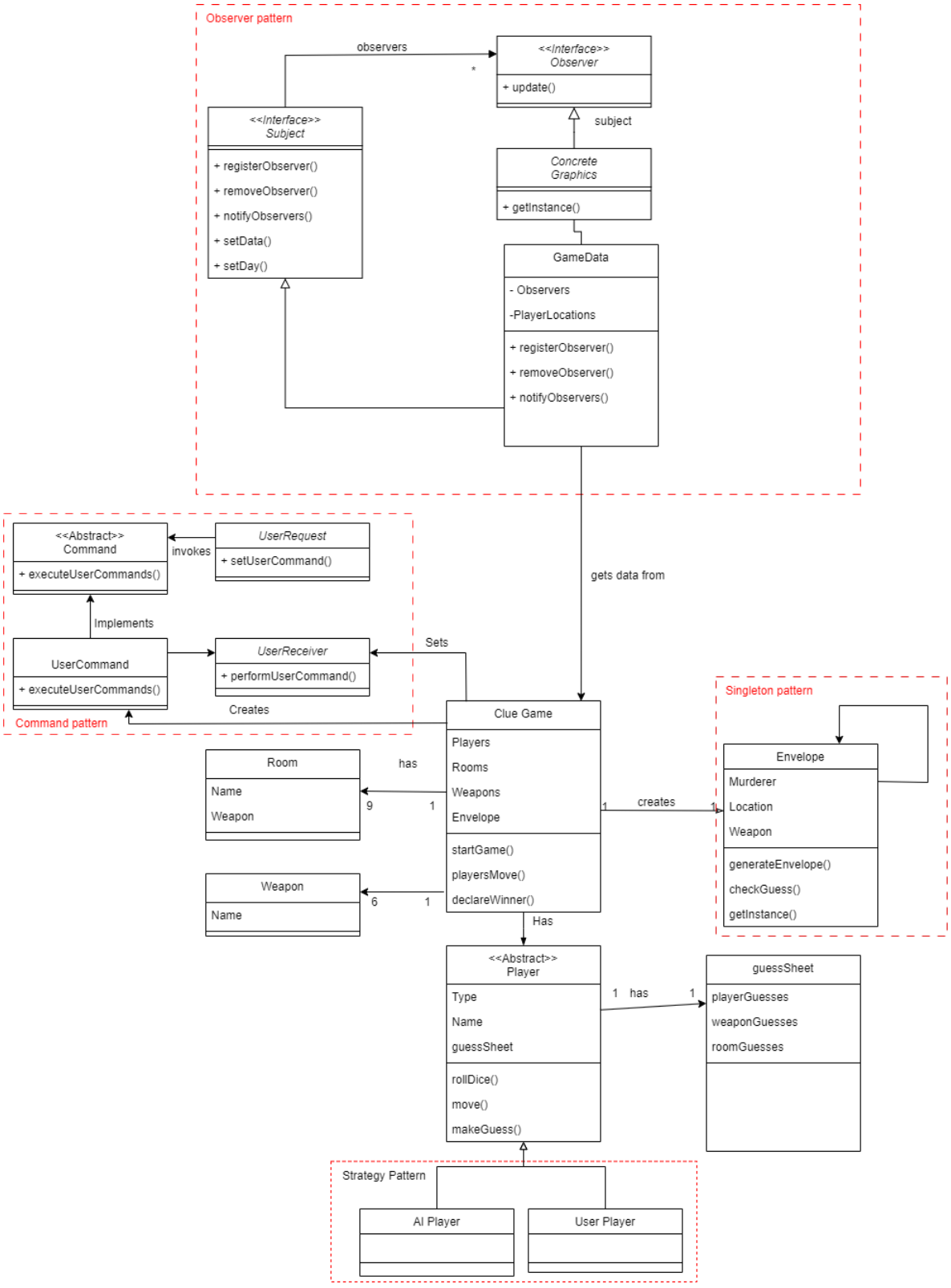
- **Third party code vs Original code statement**

  Images used in the game are taken from the following third party websites:

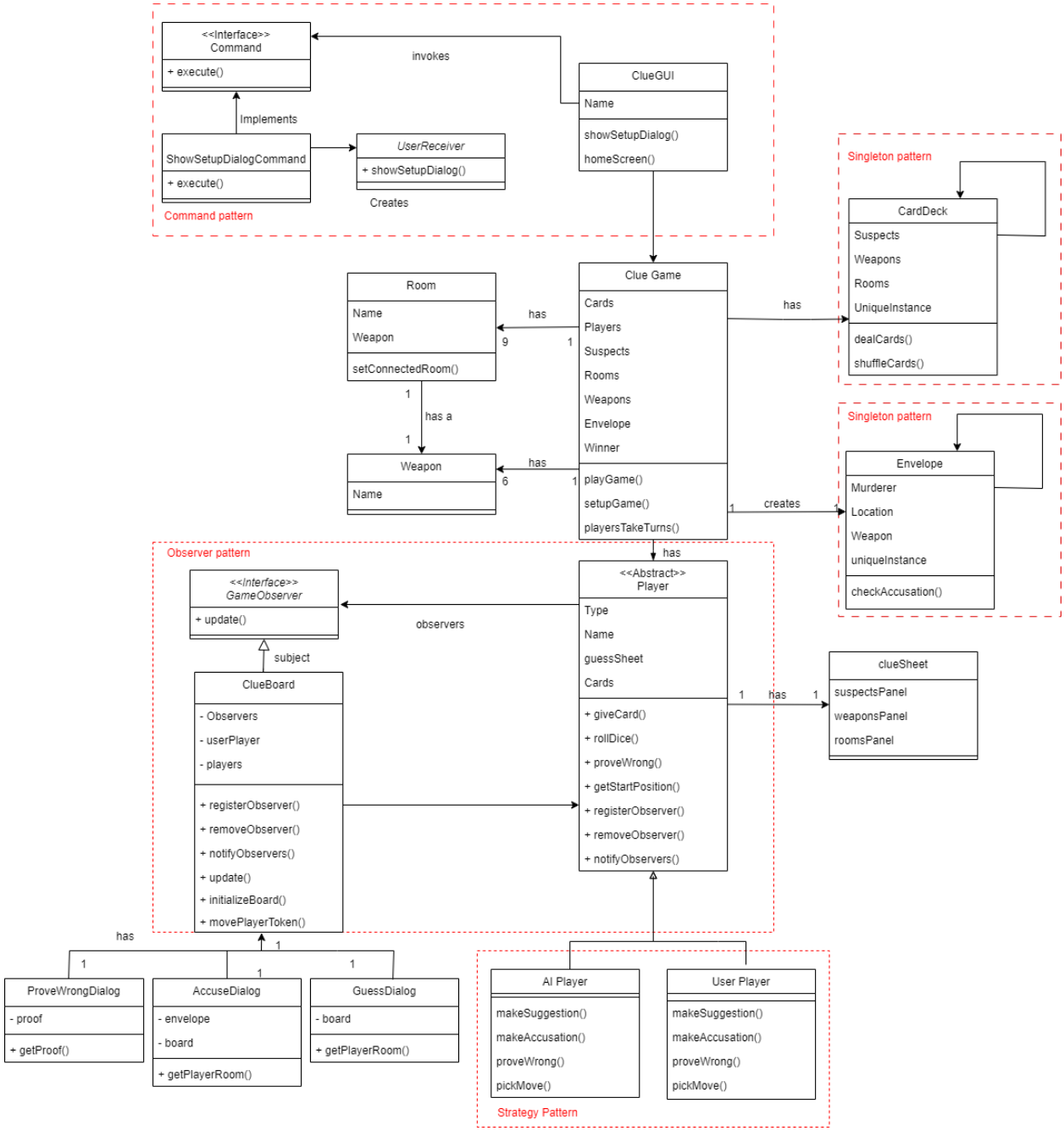
  1. https://www.cluewalkingexperience.com/
  2. https://www.housebeautiful.com/lifestyle/fun-at-home/a28860519/vote-for-new-room-in-clue-mansion-houzz-hasbro/
  3. https://www.ultraboardgames.com/clue/characters-and-weapons.php
  4. https://theartofmurder.com/forums/viewtopic.php?t=5045&start=140

- **Final Class Diagram and Comparison statement**

  Planned Class Diagram:



Final Class Diagram:



When we initially designed the project, we underestimated the number of classes that would be required to make the UI work. We had to add a GameBoard class that housed the board for the UI as well as the buttons. The GameBoard class also alerted the players when action was needed. We also had to add 3 dialog boxes which were subclasses of the GameBoard class. These were necessary in order to allow the user to select options such as which card to show to another player in order to prove them wrong. However, we were correct that the command pattern would be used for the UI.

- **Statement on the OOAD process for your overall Semester project**

1. **User vs AI player behavior**

   User and AI player behavior differs significantly, so we used the Strategy pattern to encapsulate the behavior that varied between the two types of players. This generally worked very well, however, the UI complicated this. Many of the user's behaviors were triggered by the UI, whereas the AI behavior was triggered in the game file. This made it hard to use the same function for both types of players as the UI decided many things for the user, such as where they move to, so the move() function was unnecessary for the user. In order to remedy this, we called functions from the UI in the user move() method.

2. **UI and Logic Integration**

   One of us coded the logic of the game without the UI initially and got it to a point where the game could run to completion with only AI players, but only in the command line. The other coded a UI that was functional but did not have the back end logic. We thought that these would be easy to integrate, however, this was not the case. This led to code that did not follow OOD principles very well, so we had to go through the code and reorganize it in order to fix the design. If we were to do this again we would create either the UI or the logic, then add the missing functionality. This would work better as it would be easier to design the code then implement it, instead of designing it while coding.

3. **UI Board Rendering and Spacing**

   While creating and modifying the game board for the UI, we had some issues with the JPanels. We had to create the board, then create each of the player icons and move them around the board. However, when moving the player icons the board would render incorrectly and rooms would be in the wrong locations.