Project 1: Clue Game

**Overview of Program:**

It's a simple version of the popular murder mystery board game, where you must guess which person committed the crime. The murder has a perpetrator, a weapon and a location. The player must correctly guess all 3 to win the game. There is a card for each possible guest, weapon and room. At the beginning of the game one card of each type is taken and hidden in an envelope (that represents the correct guess) The remaining cards are dealt out to the players. Each turn the player must move to a new location and make a guess (guest, weapon, location). However the location is set to their current location, so the player must move to a location to use it in their guess. Only one player can be in a location at any given time and they must move to a new location every turn. Going in a preset order, if any player holds a card representing any of the 3 parts to the guess and therefore can prove the guess wrong, they must show the player that card (if a given player has 2 or more cards that would prove the guess wrong, they only have to show one). The guessing player can then eliminate that element from future guesses. The program instantiates a number of CPU players for the user to play against ( I might make this configurable at game setup). Those CPU players will make random guesses. The game ends when someone guesses all 3 aspects of the murder correctly.

**Complexity:**

There are multiple factors to deal with. There is tracking of upto 6 players (user and remaining CPU players, including turns and cards they hold and locations they are in). There are 9 locations(rooms) and there are 9 weapons. There is also limited connectivity between rooms, so players must navigate to get to a room if they want to use that location in their guess.

Global Namespaces:

Guests - Professor Plum, Miss Scarlet, Reverend Green, Mrs White, Colonel Mustard,Madame Peacock

Weapons - Candlestick, Knife, Hammer, Gun, Rope, Wrench, Axe, Poison, Baseball Bat

Rooms - Library, Hall, Kitchen, Basement, Garage, Dining Room, Theater, Conservatory, Study

**Overview of Classes:**

**Entity Classes:**

- Player: There are upto 6 instances of this class, one of which is the user. The player has a name attribute, a location attribute (that initialized to a different starting location), they have a hand of cards. Players share the same namespace as Guest. **Attributes: Name(Guest), Location (Room), Cards (list of Card objects)**
- Card: There are 24 instances of a card (6 guests, 9 rooms and 9 locations). Cards represent the cards that are either in the envelope or dealt to the players**.**The type can be Guest, Room or Weapon. The name corresponds to a valid choice given the type. **Attributes: Type, Name**
- Location: There are 9 instances of this class. Each location will have a name, the player in the location and a list of connecting locations. Locations share the same name space as

Room **Attributes: Name, Occupant (Player), Adjoining Rooms (list of connecting Location Objects)**

- Guess: This class contains the elements of a guess. A special instance of this class will be the Envelope or Correct Guess and will be created at the beginning of the game and populated with a randomly selected attributes. **Attributes: 3 Cards objects, one each of Guest, Weapon, Room types**

**Control Classes:**

- Game: This class runs an instance of the game. It has class attributes to track the number of players, number of guesses. It will leverage the other classes to display game map, display users cards and provide help, it uses the following methods
    - Setup: This method deals with the initial setup of the game. Initializing the CPU players and User Player objects and building the envelope with correct cards and randomly dealing out/distributing the remaining cards.
    - Turns**:** This method class deals with tracking and executing turns, it gets the validated input from Validate and executes the users turn by determining and managing the response to the guess and well as creating the auto generated play of the CPU players for that turn. Might create some sub methods to separate user turn from CPU turns.

- Validate: This class contains methods to validate the users input at different points in the game. It can validate to make sure the user is entering valid commands, making valid guesses and moves and enforcing the user is playing within the rules of the game. It will be responsible for providing useful error messages when needed

- Help: This class contains attributes and methods that will provide help to the user, for example - instructions, print map, possibly hints…etc.