Straights

<u>**Small breakdown of the UML : (Describing Classes used briefly)**</u>

1. We have a Game . The Game is like any board game. It comes with a deck of cards (deck) and a platform to play the game on (gameboard). Therefore the deck and the gameboard are compositions of the game.
2. There are 4 players in the game . These players obviously can exist without the game and hence the players are aggregations of the game.
3. A player can be a computer or a human being. They both play differently perhaps.
4. The players observe the gameboard to make a decision and the board observes the players' moves and changes its own state(outlook).

<u>**Class Compositions:**</u>

1. **Game:**

    Attributes:

    - Make protected playerList : contains the list of players in the game (This will be a vector of players.)

    Implement functions:

    - *make_game()* : adds players to the list , sets observers (players observe board and board observes players.

    - *start_game()* : lets player choose his type (c or h), shuffles_deck and deals cards. (These are methods of the composite Deck class)

    - start_round(): decides who goes first, has 13 iterations and 4 in each so the game is on . This is the main function where the game is actually played.

    - *decide_winner()*: based on the discard list of each player, scoring is done and results are displayed.

    -*shuffle ()* incorporates shuffle.cc to do this

    - *deal ()* deals cards to all the players

2. ***Player: (Abstract Base Class)***

   Attributes:

   - Club_list, spades_list, diamond_list and heart_list are vectors of strings that depend on the outcome of the shuffling of the deck.
   - legal _list this changes based on what the other players choose as their card.
   - Discard list is a vector of strings storing the cards discarded by the player

   All these attributes are initially initialized as empty and change on runtime.

   Implementation Method:

   - Play() this is a virtual method and depends on the attributes of the player.

3. **Human/ Computer :**

   Attributes:

   - These classes inherit attribute from the parents

   Implementation Methods:

   - play() is overridden since both computer and player can play differently
   - Notify () this function is overridden where the virtual methods are described in class OBSERVER.

4. **GameBoard:**

   Attributes:

   - Piles - This is a vector<vector<str>> where the inner vector describes a suite (pile) in the gameboard.

   Implementation Methods:

   - Nortify () is overridden. Look at OBSERVER class

5. **Deck:**

   Attributes :

   - card_deck - this is a vector of strings depicting all 52 cards.

6. **Subject and Observer:**

   Attributes:

   - Subject has observer list

- Rest is similar to q2 A4.

Game Flow

1. Create 4 players
2. Create an instance of game. (deck and game_board) are created within
3. make_game()- add the players to game and set observers
4. Start_game ()- for each player in the list point to h or c now dynamically
5. Shuffle()- shuffle cards
6. deal ()- deal cards to all players
7. Start_round()- have 13 repetitions for which each player goes once
8. Decide_winner ()- finds winner based on the discard list
9. Destroy players

Then there are internal functions like play, notify that are contained in these big functions

PLAN:

1. **Order of classes to implement**
    - **Deck and Game_board (By 5th Dec)**
    - **Player - Human/ Computer (By 7th Dec)**
    - **Subject and Observer (By 9th Dec)**
    - **Game (From 9th Dec to however long) and over all game flow**
    - **Additional stuff if time allows:**