Linear Linked List
templated but will hold array of chars

```
public:
        insert(data_type & insert)
        remove(data_type & find_and_remove)
        get(data_type & find)
private:
        lll_node<data_type> head
```

Doubly Linked List
derive DLL node from LLL and add prev

```
public:
        insert(data_type & insert)
        remove(data_type & find_and_remove)
        get(data_type & find)
private:
        dll_node<data_type> head
```

Game
ABS which a card game needs to implement

```
public:
        // Accepts a variable list of arguments
which are the users that are playing
        virtual play(player & ...) = 0;
        // Games go until there are no more
turns to be made in which case this returns 0
        virtual int next_turn() = 0;
private:
        // either is a deck or has a deck I
haven't decided yet, probably is a deck
because all games need a deck
```

Player
derives from or uses an iostream

```
public:
        // underflow sends information to the
game
        virtual underflow
        // overflow receives information from the
game
        virtual overflow
        // The card you have chosen to send to
game, maybe to put somewhere or discard
        char card()
private:
        // Every player needs to keep track of
their hand of cards
        hand my_hand
```

Solitaire
derives from Game

```
public:
        // Takes the user that is playing
        play(player & ...)
        // Only one player so this will always
talk to that player
        int next_turn();
private:
        // Solitaire uses an LLL of arrays to
store the array of cards (chars) which have
been put at the top
        lll top;
        // Solitaire uses an array of DLLs to
manage the cards the player is manipulating
        dll columns[7]
```

Speed
derives from Game

```
public:
        // Accepts the users that are playing
        play(player & ...)
        // This will return 1 until the game is
over Sped will handle players in their own
thread so they can be speedy
        int next_turn();
private:
        // We need two LLLs to manage each
stack that players are putting cards on
        lll stack[2]
        // We will need a mutex because the
players are in threads
        mutex stack_in_use[2]
```