In this assignment, I wrote three classes, they are list as below:

* Card
* Deck
* NotFreecell

Each of the class is written in separate module.

Class Card

The class Card represents a card in play, it contains attribute card\_face and card\_suit.

In this class, the \_\_init\_\_ method takes two parameters, face and suit, the data type of both are integer. Face starts from 0 to 12, and suit starts from 0 to 3.

To make they readable to human, Two list are used here to map they to readable character.

The class has two attribute, card\_face, and card\_suit, so accessor and mutator are affored here.

* get\_face(self)
* set\_face(self, face)
* get\_suit(self)
* set\_suit(self, suit)

Finally, the \_\_str\_\_ method is implemented here to print readable string, the \_\_eq\_\_ method is also implemented here, which is used to compare with other Card object.

Class Deck

The class Deck represents a deck which is made up of cards.

The class Deck has only one attribute: cards, the attribute contains a list of cards.

I wrote these methods list as below:

* def \_\_init\_\_(): init the cards
* def shuffle(): shuffle the cards
* def add\_card(): add a new card to the deck
* def isIn(): judge if the card is in the deck
* def isEmpty(): if the deck is empty
* def \_\_len\_\_(): get the number of cards in deck
* def \_\_str\_(): return a readable string
* def draw(): draw all the cards in the deck
* def deal(): deal a card from the deck

Class NotFreecell

The class NotFreecell is defined in the freecell.py, it’s the main part of the game.

In this class, I did jobs list as below:

* Initiate the game, mainly initiate the deck, tableaus, cells and foundations. The deck must been shuffled before dealt, all cards are dealt from left to right in the tableaus.
* Start the game, show the game board, and game commands to user.
* Go to while loop, Prompt for user to input the command, parse the command, execute the command, judge if user win the game, display the game board again, if the user is not win, then prompt for user to input a command again.
* There are five kinds of motion in this game.
  + TC: Move card from tableaus to cells
  + TT: Move card from tableaus to foundations
  + TF: Move card from tableaus to foundations
  + CF: Move card from cells to foundations
  + CT: Move card from cells to tableaus

When parsing the command, we get the kind of movement, and the source card position, and destination position.

There are five kinds of movement, so I wrote five methods to handle the movement.

* + tableau\_to\_cell(self, tab, cell): move card from tableaus to cells
  + tableau\_to\_foundation(self, tab, fnd): move card from tableaus to foundations
  + tableau\_to\_tableau(self, tab, tab): move card from tableaus to tableaus
  + cell\_to\_foundation(self, cell, fnd): move card from cells to foundations
  + cell\_to\_tableau(self, cell, tab): move card from cells to tableau
* During each movement, we have to decide if the movement is valid, if valid return true, otherwise return false. So I wrote three methods to validate the move.
  + valid\_fnd\_move(self, src\_card, dest\_card): Decide if the movement to foundations is valid? If the move is not valid, raise an Exception

The conditions for the valid foundation move:

* + valid\_cell\_move(self, src\_card, dest\_card) Decide if the movement to the cells is valid?

Conditions for the valid cell move:

* + valid\_tab\_move(self, src\_card, dest\_card) Decide if the movement to tableaus is valid?

Conditions for the valid tableau move:

* Method is\_winner(self) is used to decide if the game was won? If all the foundations have 13 cards and the top of each pile if King, then the game was won and return True, otherwise return False.
* Method display(self) is used to display the game board, including the cells, foundations and tableaus
* Method start(self) is the entrance to the game.