CSE 231 Spring 2010

Programming Project 9

Assignment Overview
This assignment will give you more experience on the use of classes.

This assignment is worth 50 points (5.0% of the course grade) and must be **completed and turned in before 11:59 on Monday, April 5th , 2009.**

Background
You will implement the Spider Solitaire card game. It is a fairly simple game and does not have many strategies. Our goal is to simply enforce the rules for players. To play the game on Windows, just open the Start menu and choose Games, click the Spider Solitaire. If not Windows, you can find many examples anywhere online.

More important than anything, familiarize yourself with the game by playing the game before considering programming. There are typically three difficulty levels. We only require two of them: beginner with one suit, and advanced with four suits.

Important Note: There is a difference in our game from online versions. 1 suit means all the cards should be the same suit (that is, only numeric order matters). 4 suits mean there are the normal 4 suits and movement of runs of cards must to be of the same suit. In our game, when playing 1 suit all 4 suits are shown, but the game runs as if there is only one suit (that is, only number matters).

The game play proceeds as follows:

1. The Start
    a. Put two decks of cards (104 cards) together as a “stock”.
    b. There is one tableau. It begins with 10 stacks with 6 cards in each of the first 4 stacks and 5 cards in each of the remaining 6 stacks. In each stack, initially, only the top card is visible.
    c. Cards are moved from one stack of the tableau to another. Line up runs of cards from king through ace in the same suit to remove them.
    d. You can deal one card to each stack of the tableau from stock, until there are no cards left in stock.
2. The Goal
    a. The object is to remove cards from play in the fewest moves possible.
3. Rules of Play
    a. Tableau
        i. You can move a card from the bottom/end of a stack to an empty stack.
        ii. You can move a card from the bottom/end of a stack to a card with the next highest value, **regardless of suit or color**.
        iii. You can move a run of cards all of the same suit and in numeric order, as if they were one card.
        iv. When you create a run of cards from king through ace, they will be removed from the playing field.
    b. Stock
        i. If you cannot make any moves on the Tableau, you can take cards from the stock.
        ii. **Different**! When the stock is used, a card is added to the end of each Tableau column.

        iii. There must be at least one card in each stack before you can deal a new row of cards.
    c. Score
        i. Initially you have 500 points, each move costs 1 point.
        ii. If you remove a run of cards from king through ace, you get 100 points.

Your program allows a user to play the game, ensuring that they follow the rules.

Requirements
Implement the game in Python. You can also try the example game *spider.py* to get a feel for the game. Copy spider.py and either implementation25.pyc or implementation26.pyc (depending on whether you are running python 2.5 or python 2.6) to your computer if you use IDLE.
Requirements are:

1. Use the provided Card and Deck Class, found in the cards.py file in the project directory. **Do not modify the cards.py program** as you will only turn in proj09.py. Just import cards.py into your proj09.py.
2. You must use functions in this game. Implementations without functions will not be graded.
3. Create one function to be called play(), which starts the play of the game.
4. If the user makes a move that is illegal, you must inform them of the error and let user choose another move.
5. You must determine if a winning position is achieved. If so, report it and stop the game, printing out a “Winning” message.
6. The provided demo uses the following prompts, **which you are also required to use**:
    a. m [number of cards] [stack A] [stack B]' to move a certain number of cards from stack A to stack B"
    b. “d” to deal cards
    c. “h” gives help to the user (see the demo program)
    d. “q” means quit
7. The demo program checks for the following errors, **which you are also required to check for**:
    a) trying to move cards that violate the rules
    b) checking on user input to capture incorrect commands
    c) whether the move command is in correct format
    d) whether it is a valid move trying to deal more cards when there are no cards left in the stock or not all stacks have at least one card
8. The demo program provides a template for the output format of your program. **You are required to use the output format of the demo program for your program**

Card and Deck Classes plus Display function
We provide a module named ***cards*** that contains a card class and a deck class. We also provide a sample piece of code that demonstrates how to use the cards module. The card and deck classes are general purpose for developing card games so they contain many methods that may not be used in any particular implementation. You are welcome to use all of them, but do not be surprised if there are many that you do not need for this project. For example, in my implementation I used *set_hidden*, *show_card*, *get_rank*, and *get_suit* functions of Card and *deal*, *cards_left*, *shuffle*, *add_deck* and constructor method of Deck. That’s about it.

The get_rank() method returns the rank of the card: 1 for ace, 2-10 for number cards 2-10 (respectively), 11 for Jack, 12 for Queen, 13 for King.

Deliverables
You must use handin to turn in the file **proj09.py** – this is your source code solution; be sure to include your section, the date, the project number and comments describing your code. Please be sure to use the specified file name, and save a copy of your proj09.py file to your H drive as a backup.

Other good information

Notes:
1. Play with the provided demo program and get a feel for the game
2. Look carefully at the example cardsDemo.py program. It imports the cards module and uses the two classes and gives you a better idea how you can use them. These classes provide methods you may not need, but they should provide almost any method you do need.
3. When using class methods remember the parenthesis—no error is generated for missing parenthesis, but results will not be what you expect.
4. There are multiple parts to the game (setup, printing, game play, starting). Address each one individually and then put them together.
5. You should avoid dealing a card from an empty stock.
6. You should make sure there is at least one card in each stack before you can deal a new row of cards.
7. For difficulty level of beginner, all suits are regarded as the same suit.
8. For playing the game, begin by assuming perfect input. Get that working and add error checking later.
9. Add as much error checking as you can. Error conditions could occupy quite a bit of code!