CPSC 170-- Assignment #1

Playing Cards: Crazy-8's Solitaire
Due Tuesday Feb 12, 2:50pm

Description

We want to have a graphical program that plays Crazy 8's Solitaire. This game is a variation of the popular card game, Crazy 8's; you will essentially be implementing a one-handed version. To play:

- One deck of 52 cards is used.
- 6 cards are initially dealt to a hand, and one card is dealt to a discard (play) pile.
- The objective of the game is to empty your hand by systematically discarding cards to the play pile.
- To play:
    - Select a card from the hand that matches either the suit or the value of the top card of the discard pile.
    - 8's are considered wild (or crazy). When a 8-card is played, the game should prompt the player for a new suit. The next play must match the designated suit.
    - If there are no matching cards, the player choose to draw a new card from the deck.
    - The game is won if the hand is empty
    - The game is lost if either a) the hand exceeds 12 cards or b) the deck is empty and there are no valid discards.

Phase I: Design & JavaDoc.

This assignment builds off of the Card class and the PileOfCards class that we worked on last week in lab. The first stage of this assignment is to think about the design of the classes that are needed. There are essentially two categories of classes needed to complete this assignment: the classes that support the data structures for the game and the classes that support the GUI. Ultimately, you will need to implement the following classes containing the methods listed:

Classes supporting Data Structures

.

- Modify **Card** class
    - int getSuit()
    - int getValue()
    - boolean equals (Card C)
    - static ImageIcon getBackIcon
    - static ImageIcon getBlankIcon
    - static int indexOfSuit(String target)

- Create a class **DeckOfCards** that extends PileOfCards
  - Constructor
  - Shuffle
  - AddCard (Not Permitted)
- Create a class **HandOfCards** that extends PileOfCards
  - boolean removeCard(card c)
  - card getCard(int index)
  - int size()
- Create a class **PlayPile** that extends PileOfCards
  - Overrides AddCard
  - RemoveCard (Not Permitted)

Classes supporting GUI.

- Create a class **CardLabel** that extends JLabel
  - public CardLabel(card C, boolean visible)
  - getCard
  - setCard (Card C)
  - setNull
- Create a class **PlayPanel** (extending JPanel)
  - Has a Deck and a PlayPile
  - Has CardLabels to represent a the deck and the PlayPile
  - Card DrawCard()
  - boolean playCard(Card C)
- Create a class **HandPanel** (extending JPanel)
  - Has an array of Card Labels showing the hand
  - Has a button to draw a card.
  - Implements a MouseListener to allow selection of cardLabels
  - Implements an ActionListener to react to the Button
  - private updateLabels();

To complete Phase I, you should study the classes described above. Prepare the skeleton code with comments to generate the JavaDoc that completely describes the various classes and supporting methods. Make an appointment with me sometime before Friday afternoon to discuss your understanding of the program design. This meeting will probably last 20-30 minutes.

Phase II : Implementation

Once you have a clear understanding of the design, you should implement your solution. As usual you should implement this in steps, and test segments of your code for correctness as you are implementing. This will likely require you to write some smaller driver programs for your own piece of mind. You should find that a majority of the methods to be implemented are very straightforward and direct. If you have done a thorough job for Phase I, you will have identified only two or three methods that are will require significant effort.

Other Requirements: As usual use good programming practices that include appropriate naming of variables, correct indentation, and documentation. Just because you are producing JavaDoc for method-level comments, does not mean that you should forget about using internal documentation where necessary to explain what the code is doing.

Turn in: Printed copies of all source code files related to your project. Tar your project directory and send the tar file as an email attachment.

Academic Integrity Reminder: Programming assignments are to be your own work. You may get help on the specifics of the assignment from no one except the instructor. You may not show your program to anyone or look at anyone else's program or share ideas with anyone about how to write the program.