Assignment #5 – Dealing Cards

Prepare to work on this assignment by creating a work folder called assign5 in your w-number folder. Call your program Dealer#.java, where the # is whichever version you are working on at the time. You must start with version 1 (Dealer1.java) and get it working before moving on to version2 (Dealer2.java), etc.

Version 1 (80 possible points)

Write a ConsoleProgram to create a deck of playing cards, shuffle them, and deal out 5 cards to the player. To do this:

- You will need an array of 52 cards. In the basic version of this program, each card is a string in the form “XY”, where X is the rank (A, 2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, A) and Y is a letter designating the suite (C, H, D, S).
- You will need to shuffle the cards by swapping the positions of two randomly selected cards in the array. For this, you should write and use three methods:
  - randRange – a method which takes 2 parameters: a minimum integer and a maximum integer. It creates a random integer in this range and returns it.
  - swap – a method which takes 3 parameters: an array of cards and two subscripts. It swaps the array elements with the given subscripts and returns nothing in its name.
  - shuffle – a method which takes 2 parameters: an array of cards and a number of times to swap cards. It shuffles by swapping two randomly selected cards the desired number of times (a possibly different pair each time). It returns nothing in its name (so it is a void method).
- You will need an array of 5 cards for the player’s hand. After shuffling the deck, you will fill this array with the first 5 cards from the deck.
- Display the unshuffled deck, then the shuffled deck, and finally the player’s hand. Indicate which is which in the output. For example, the output of one program run might look like this:

Unshuffled:
2C, 3C, 4C, 5C, 6C, 7C, 8C, 9C, TC, JC, QC, KC, AC, 2D, 3D, 4D, 5D, 6D, 7D, 8D, 9D, TD, JD, QD, KD, AD, 2H, 3H, 4H, 5H, 6H, 7H, 8H, 9H, TH, JH, QH, KH, AH, 2S, 3S, 4S, 5S, 6S, 7S, 8S, 9S, TS, JS, QS, KS, AS

Shuffled:
2C, QC, 3S, 9C, 9S, 6C, 6H, 3C, QC, 2H, 8C, 9H, AH, 4S, AS, TH, 6D, 3D, 2S, 8S, 5C, TC, 4C, 4D, 7C, 6D, 9D, JD, 4H, 5S, KH, 7H, JS, 8D, TD, KC, 3H, 2D, KD, AC, KS, JC, 5H, JH, 8H, 5D, 7S, TS, QD, 7D, 6S, QS

Your Hand:
2C, QC, 3S, 9C, 9S

To do this, you should write and use the following method:
- showCards – a method which takes 1 parameter: an array of cards (which does not contain any particular number of cards). It displays the cards neatly and in order (from the 0’th element up), and returns nothing in its name.

When you get this version of the card dealer working, make sure you include the required comments (including descriptive comments for each variable in your program), and then commit it with the log message "Dealer version 1" (it should be named Dealer1.java). If everything is correct, you will earn 80 points for Assignment #5.
Version 2 (85 possible points)

• Prompt the user for the size of their hand and the number of times to swap cards when shuffling. Modify your program to use these numbers.

When you get this version of the card dealer working, make sure you include the required comments (including descriptive comments for each variable in your program), and then commit it with the log message "Dealer version 2" (it should be named Dealer2.java). If everything is correct, you will earn 85 points for Assignment #5.

Version 3 (95 possible points)

• Find the most valuable card in the player’s hand. To do this, you should write and use the following methods:
  o cardVal – a method which takes 1 parameter: a single card. It returns the value of that card. Number cards are worth their number, Jacks are 11, Queens are 12, Kings are 13 and Aces are worth 14 points. Hint: remember that you can get the first character in a string with the charAt(0) method. For example, the first character in a string named cityName is found with cityName.charAt(0).
  o highCard – a method which takes 1 parameter: an array of cards (which may be any size). It returns the subscript of the card with the largest value.

• In addition to the output from version 1, display a line of the form “High Card: XX”, where XX is the most valuable card in the player’s hand. Using the example from version 1, it would add to the output:

  High Card: QC

When you get this version of the card dealer working, make sure you include the required comments (including descriptive comments for each variable in your program), and then commit it with the log message "Dealer version 3" (it should be named Dealer3.java). If everything is correct, you will earn 95 points for Assignment #5.

Version 4 (100 possible points)

• Change the program to a GraphicsProgram and display a picture of the player’s most valuable card. You can download a collection of card images from the files area of the class website in a zipped file called card_graphics.zip. You should still retain the original input and output in the IDE console area. Hint: Start by creating a method which takes a card string as a parameter and returns a string containing the corresponding graphic image filename.

When you get this version of the card dealer working, make sure you include the required comments (including descriptive comments for each variable in your program), and then commit it with the log message "Dealer version 4" (it should be named Dealer4.java). If everything is correct, you will earn 100 points for Assignment #5.
Extra credit versions. If you do extra credit in addition to Versions 1 thru 4, you **must** do Version 5. You may then do any or all of Versions 6 thru 8, although the indicated order is best.

**Version 5 (Extra Credit: +10 possible points)**

- Ask the user if they want to take the first N cards from the shuffled deck, where N is the hand size. If they do, deal the cards as before. If they do not, allow them to input the string for each card in their hand.

When you get this version of the card dealer working, make sure you include the required comments (including descriptive comments for each variable in your program), and then commit it with the log message "Dealer version 5" (it should be named Dealer5.java). If everything is correct, you will earn an additional 10 points for Assignment #5.

**Version 6 (Extra Credit: +30 possible points)**

- Determine if the player’s hand is a flush (all the same suite). If it is, display “XX high flush in SUITE”, where XX is the most valuable card in the player’s hand, and SUITE is either the word “clubs”, “diamonds”, “hearts”, or “spades”. For example, if player’s hand was:

  Your Hand:
  2C, QC, 3C, 9C, CS

  you would add to the output:

  QC high flush in clubs

  To do this, you should write and use the following method:
  - `isFlush` – a method which takes 1 parameter: an array of cards (which does not contain any particular number of cards). It returns a boolean true value if the cards are all in the same suite.

  - Hint: remember that you can get the second character in a string with the `charAt(1)` method. For example, the second character in a string named `cityName` is found with `cityName.charAt(1)`.

When you get this version of the card dealer working, make sure you include the required comments (including descriptive comments for each variable in your program), and then commit it with the log message “Dealer version 6” (it should be named Dealer6.java). If everything is correct, you will earn an additional 30 points for Assignment #5.

**Version 7 (Extra Credit: +40 possible points)**

- Determine if the player’s hand is a straight (if the cards could be arranged in rank order with no gaps, but not necessarily all the same suite. They don't have to actually be ordered this way in the hand). If it is, display “XX high straight”, where XX is the most valuable card in the player’s hand. For example, if player’s hand was:

  Your Hand:
  8C, QC, 9S, TC, JS
you would add to the output:

QC high straight

To do this, you should write and use the following method:

- isStraight – a method which takes 1 parameter: an array of cards (which does not contain any particular number of cards). It returns a boolean true value if the cards could be arranged in rank order with no gaps.

- Hint: Get the most valuable card in the hand and remember it. Then change it to the lowest possible card. Then get the most valuable card in the hand again, and compare it to the previous most valuable card. If it is one less in rank, remember it and do the same thing again. If you can keep doing this until all cards have been changed to the lowest possible card, then it must be a straight.

When you get this version of the card dealer working, make sure you include the required comments (including descriptive comments for each variable in your program), and then commit it with the log message "Dealer version 7" (it should be named Dealer7.java). If everything is correct, you will earn an additional 40 points for Assignment #5.

**Version 8 (Extra Credit: +20 possible points)**

- Determine if the player’s hand is a straight flush (both a straight and a flush). If it is, display “XX high straight flush in SUITE”, where XX is the most valuable card in the player’s hand, and SUITE is either the word “clubs”, “diamonds”, “hearts”, or “spades”. For example, if player’s hand was:

  Your Hand:
  8C, QC, 9C, TC, JC

  you would add to the output:

  QC high straight-flush in clubs

When you get this version of the card dealer working, make sure you include the required comments (including descriptive comments for each variable in your program), and then commit it with the log message "Dealer version 8" (it should be named Dealer8.java). If everything is correct, you will earn an additional 20 points for Assignment #5.

**What to turn in**

Commit versions 1 thru 5 in sequence as high as you can, as well as any of versions 6 thru 8, before 2:00 p.m. on the last day of classes, Friday, May 5, 2006. **This is not negotiable! No late assignments will be accepted, even if it would result in a failing grade! (see the syllabus).** Remember that you may commit multiple times for each version if you wish (and it is a good idea to do so for every major change or at the end of each work session). You don’t have to make it all the way up to any version beyond version 1, but of course, if you do, your grade will be higher (as long as it is correct). Remember to have fun, and get help from the instructor when you need it!