Given: Monday, February 25, 2013
Electronic Copy Due: Sunday, March 10, 2013 by 11:59 p.m.

Objectives

- Class design. Objects working together
- Constructors
- Arrays and ArrayLists
- To make multiple instances of a single class, and understand how they are distinct
- to use incremental problem solving strategies to break a large difficult problem (this assignment) into small, easy to solve problems (methods and classes)
- to utilise error checking throughout, to better handle bad user input
- Testing,Testing,Testing

Introduction

In this assignment, you will be writing several classes which will be used in Part 2 to create a card game. In this assignment, you will create at least a Card class, a Deck class, and a main class which will give a short menu of operations which can be performed by the Deck class.

The majority of the code for this assignment should be within the Card and Deck classes. Each selection from the menu described below. The card and deck class should not contain any input or output code.

Upon launching the program, the user should be presented with a menu. From the menu, there should be at least the following 6 options:

Choice	Description
N:	**New Deck** – This option should reset the deck to 52 cards, ordered from ace to king of spades, hearts, clubs, and diamonds in that order.
S:	**Shuffle Deck** – This option should shuffle the current deck without changing how many cards are in the deck.
T:	**Top Card** – This option should print the top card on the deck. The card should not be removed from the deck.
C:	**Cut the Deck** – This option should print the value of a random card in the deck, and then the deck should be cut at that point, resulting in the revealed card being at the bottom of the deck, and the card which was originally under that card at the top of the deck.
D:	**Deal Hands** – This option should deal two hands of cards. The user should be asked for the size of the hands, and then the hands should be dealt and printed. The cards in the hands should no longer be in the deck after this operation.
Q:	**Quit** – This option will exit your program.

Feel free to add other options to your menu, for debugging purposes. The options above must be selectable with the given letters, however. As before, both upper and lower case characters should be acceptable. The processing should continue until the user selects quit. An invalid input should print an error message and present the menu again.

Classes

For this assignment, you will need to create at least three classes (there could be more depending upon your design). You will need a Card class, a Deck class and a program class. You are required to properly use constructors for your classes.

The Card class should at least have the following requirements:

- It should have no interaction with the user (input or output)
- The rank and suit of the cards can be represented in any way you choose. But you must document how you are representing everything. Try to use constants rather than hard coded values where you can.
- There should be a method which returns a String of the form "*Rank* of *Suit*", where *Rank* is spelled out properly for Ace, Queen, King, Jack. Numerical otherwise. Here are some examples:
  - "Ace of Hearts"
  - "3 of Spades"
  - "Jack of Diamonds"
- There should be a method which will return true if two cards have the same rank and suit.
- The rest of the design is up to you.

The deck class should at least have the following requirements:

- It should have no interaction with the user (input or output)
- The Deck class will deal with setting up all the cards for a standard 52 deck of cards.
- It should handle shuffling the cards currently in the deck when requested
- There should be a method to get the top card of the deck without modifying the deck. This method will return an object of the Card class.
- There must be a method to cut the deck. This method will return the cut card, but that card will remain in the deck.
- There must be some way to generate hands of cards. The cards dealt into the hands must be removed from the deck so that they will not be dealt in any following hands. This method can be designed how you see fit.

You may need more methods than those that I have described for both classes.

Testing

You must also include a short test document that includes test cases (using black box test case generation) that verify at least that all 6 menu options work properly for your program.

Documentation

The program should be documented according to the standards used in COMP 1502. Each method and class (if classes are used) must have a javadoc comment describing what it does (as show in the class examples and the assignment solutions you have been provided with). In addition, inline documentation should be used as needed. Indentation and the use of constants will also be marked.

Submission

All of your source code should be submitted to the submit drive in a folder whose name includes your first and last name. The date and time stamp on your submission will determine if the assignment is on time.

Marking

Your work will be graded not only on program correctness, but also on quality of algorithm design, code style, readability, and quality of documentation. All programs submitted will be tested using a set of test data. Programs that do not compile will receive at **most** 40%.