Assignment 3: Deck of Cards

Due: Week seven. Check the course web site for the exact due date and time, which are campus-specific.

Web stuff: See the course web site for starter code, hints, and announcements about the assignment.

Purpose: Provide practice writing classes that implement interfaces, and thus meeting specifications. Provide practice writing code that manipulates a linked structure.

Cards: This assignment is about a deck of cards. Some of you may never have played with cards. A deck of cards is a pile of cardboard rectangles, each of which has two things printed on it: a "suit" (either "hearts", "diamonds", "clubs", or "spades"), and a "face value" (either a number between 2 and 10 inclusive, or one of the special values "jack", "queen", "king" and "ace"). There are many different games that can be played with cards. Most games begin by "dealing" some or all of the cards to the players, that is, passing the cards out among the players.

For this assignment, you don't need to know about card games; you'll write code that builds and manipulates a deck, but does not play a game.

Overview: The ultimate goal of this assignment is to write a class that implements a deck of playing cards using as a doubly-linked list.

Your task: **Step 1**

We have defined a new ADT:

Data:
A two-ended list is a hybrid of a stack and a queue; it contains objects, and one can insert or remove from either end as desired (but not the middle).

Operations:

- Add an item at the front of the list.
- Add an item at the back of the list.
- Retrieve the contents of the first item in the list
- Retrieve the contents of the last item in the list
- Remove and retrieve the contents of the first item in the list.
- Remove and retrieve the contents of the last item in the list.
- Return the number of items in the list.

A `TwoEndedList` interface is available on the course web site.

Design and write a class called `LinkedTwoEndedList` that implements the `TwoEndedList` interface using a doubly-linked list. We have provided a node class for you.

Test your class thoroughly. You can put your code for testing inside the `main()` method of your `LinkedTwoEndedList` class, or you can write a separate class and put the testing code in its `main()`.

Step 2

A `Deck` interface is available on the course web site. Design and write a class called `ListDeck` that implements the `Deck` interface using an instance of your `LinkedTwoEndedList` class. Below are details on how some parts of the class must work:

- Your `loadDeck()` method must read cards into the deck from the `MyStreamTokenizer` that is passed to it, and must add these cards one at a time to the bottom of the deck. A data file defining the cards in a standard deck is provided on the assignment web site.
- Your `shuffle()` method must work as follows: Make a new instance of `LinkedTwoEndedList` and move a random number of cards (between 2 and 6) from the top of the old deck, one after the other, to the bottom of the new deck. Then move a random number of cards (again between 2 and 6) from the the *bottom* of old deck, one after the other, to the bottom of the new deck. Keep doing this until the old deck is empty — that is, until all the cards have been transfered to the new deck. Then repeat the entire process two more times, to achieve a thorough shuffling.
- To get the random numbers it needs, `shuffle()` must use the instance of `MyRandom` that is passed to it. It is the client code's responsibility, not `Deck`'s, to construct this `MyRandom` and give it a seed. For example, `DeckDriver`'s `main()` does this.
- Your `deal()` method must use the `toString` method in the `Card` class provided, which makes it easy to print the cards in columns. An example of the output we expect from `deal()` will be provided on the Hints page for Assignment 3.

Use the `DeckDriver` class provided to test your `Deck` class. Do additional testing as you see fit.

What to submit: You will not hand in a paper submission for this assignment. Submit the `.java` files for your `LinkedTwoEndedList` and `ListDeck` classes electronically. Campus-specific requirements regarding submit location are provided on the different campus web sites.