COLBY COLLEGE
Department of Mathematics
and Computer Science

CS115 **Allen B. Downey**
Spring 1998

Assignment 10: Cards and Decks

Due: Wednesday 22 April

The purpose of this assignment is to write a class definition for playing cards, and implement algorithms for shuffling and sorting decks.

Reading

Continue reading Chapter 5, and read Section 7.4.

The Card class

1. Create a new Java project called Card and make it an Application. The Card class will define a new type of object that will represent a playing card. It will have two instance variables, both integers, that will represent the rank and suit of the card. Declare two instance variables named `rank` and `suit` at the top of the class definition.

2. Write two constructors for the Card class. One should take no arguments and initialize both instance variables to 0. The other should take the rank and suit as arguments.

3. Write a `toString` method for this class. Probably the way you will want to do this is to create two global final arrays of Strings, named `ranks` and `suits`:

```
public static final String[] suits =
    { "Clubs", "Diamonds", "Hearts", "Spades" };
public static final String[] ranks =
    { "narf", "Ace", "2", "3", "4", "5", "6", "7", "8",
      "9", "10", "Jack", "Queen", "King" };
```

   Recall that both global variables and instance variables are declared at the beginning of the class definition, outside of any method, and that you can identify global variables by the presence of the keyword `static` (and instance variables by its absence).

4. Within `toString` you should use the instance variables as indices into the global arrays in order to select the appropriate string for a given rank or suit. For example, if the `suit` is 0, then the expression `suits[suit]` generates the String `"Clubs"`. In other words, the value 0 represents the suit Clubs, the value 1 represents the suit Diamonds, etc.

   Given all that, why is the first element of the ranks array `"narf"`?

5. Write a `main` method that creates a new card using either constructor, and prints the card using `toString`.

Compare cards

1. Write a static (or class) method called `compareCards` that takes two cards as arguments and returns an integer with value -1 if the first card outranks the second, 1 if the second outranks the first, and 0 if the two cards are the same. One card outranks another if its rank is higher or, if the two ranks are the same, if its suit is higher. The ranks are basically in numerical order, except that Ace is the highest rank instead of the lowest. The order of the suits is the order they appear in in the arrays above (Spades > Hearts > Diamonds > Clubs).

2. Invoke your method from `main` and test that it works.

3. Write a second method called `compare` that is functionally identical to `compareCards` except that it should be an instance method that compares the current object (`this`) to a second card that is passed as an argument.

4. Invoke this method from `main`, too.

Make a deck

1. Write a static method called `newDeck` that creates an array of 52 Cards and that fills the array with 52 new Cards. For now, you should make all the cards identical, like a trick deck that only has Aces of Spades. You should use `new` once to create the array, and 52 times (inside a loop) to create each of the cards.

2. Write a static method called `printDeck` that takes an array of cards and prints the cards in the deck in order. Your code should not assume that the deck has 52 cards; instead, use the `length` field of the array object to determine the number of cards.

3. Now modify `newDeck` so that instead of 52 identical cards, the deck contains exactly one of each card.

Shuffle the deck

1. Write a static method called `swapCards` that takes two indices as arguments and that switches the two cards at the indicated positions. Remember that you can switch objects by switching the references to the objects rather than having to move the data (instance variables) in the objects.

   If the previous paragraph doesn't make sense to you, see the figure. If it still doesn't make sense, talk to Allen or a TA.

2. The following method generates a random number between `low` and `high`, including both low and high. As you type it in, be sure you understand how it works. Also, you might want to test it to make sure that you typed it correctly and I haven't messed up.

```
public static int randomInt (int low, int high)
{
    double x = Math.random() * (high - low + 1);
    return (int) x + low;
}
```

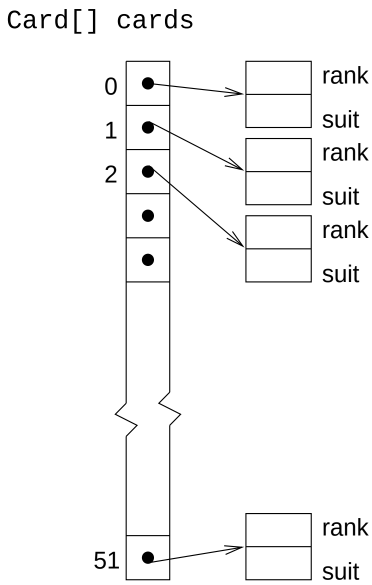

Figure 1: An array of cards is really an array of references to cards, which is why the cards have to be allocated with `new` individually. It is also why you can swap cards just by swapping references, rather than having to copy the data (`rank` and `suit`) in each object.

3. Use `swapCard` and `randomInt` to write a static method named `shuffleDeck` that takes an array of cards as an argument and shuffles the deck in such a way that every card has an equal probability of being at any position in the deck. As with the other methods that operate on decks, your code should not assume that the deck has 52 cards.

Sort the deck

1. Write a static method called `findHighestCard` that takes an array of cards and an index as arguments, and that finds and returns the index of the highest-ranking card in the deck, starting at the given index. Of course, you will want to use the `compare` method to compare cards to one another.

2. Write a static method called `sortDeck` that takes an array of cards as an argument and that sorts the deck so that the highest ranking card is first. Using the `findHighestCard` method, you can write this method in only about four lines, and you will find that the result is disturbingly similar to `shuffleDeck`.

What to turn in

1. Please prepare and print your code as usual. Don't forget to write your name in the upper-right corner, even if your name also appears in the code (as it should).