SPRING 2004: **COP 3530** Data Structures

[Programming Assignment 1; Due January 27 in class.]

This assignment will be graded, but the points will not be used towards your grade for the class. Extra credit work will be graded.

Problem Description

You have been given a regular deck of 52 cards. At the start, the deck of cards is in **perfect increasing order**, i.e., the deck when placed face down has the four Aces at the top followed by the four twos, four threes, ..., four tens, four jacks, four queens, and four kings. Within the set of four Aces, the cards are ordered starting from the `Club Ace` at the top followed by the `Diamond Ace`, `Heart Ace`, and the `Spade Ace`. The same ordering of the suites occurs in the twos, threes and so on. Your first job is to implement classes called `aCard` and `deckOfCards`. The class `aCard` should have an overriding implementation of the `equals` and `toString` methods. It should also implement the `compareTo` method of the interface `Comparable`. Instead of storing the string representing the card, the class `aCard` should have one integer data field called `cardCode` to store an integer code for that card. For example, this integer code could refer to the position of that card in the initial deck. The method `toString` should take care of converting from the integer code to the string representing the card. For instance, according to the coding scheme mentioned above, code 0 would represent "Club Ace", code 6 would represent "Heart Two", and so on. The class `deckOfCards` should contain a list (array) of `aCard` and should be initialized to contain the deck in perfect increasing order. Also, define an interface called `Deck` and have the class `deckOfCards` implement `Deck`.

We define a **perfect shuffle** to be the following operation. The deck of cards is placed face down on a table. It is then separated into two equal half piles, and then the two half piles are "perfectly interleaved" (just like the way the professional card dealers shuffle at a casino). In other words any two cards that are next to each other in one of the half piles, will be separated by exactly one card from the other pile after the perfect shuffle. Care is taken so that the card that was on top of the deck before the shuffle remains on top after the shuffle (similarly, the card that was at the bottom of the deck before the shuffle remains at the bottom after the shuffle). Note that the method, as defined above, has no randomness in the result. Your next task is to define a method called `perfectShuffle` in the interface `Deck` and implement it in `deckOfCards`.

There is one special card in the deck – the *Jack of Spades*. You need to define a method called `findSpecial` in `Deck` and implement it in the class `deckOfCards`; this method will report the location of the special card (i.e., its position from the top of the deck, where the top card is said to have location 0 and the bottom card 51).

Now assume that a dealer deals 4 hands for a card game, i.e., the dealer deals one card per player iteratively until all the cards in the deck are distributed. So the first player (player number 0) would get cards at locations $0, 4, 8, 12, \ldots, 48$, while the second player (player number 1) gets cards at locations $1, 5, 9, 13, \ldots, 49$, and so on. You need to implement a method called `findMax` in the class `deckOfCards` that will report the "highest card" in player i's hand (assuming that the shuffler were to deal the current deck of cards). In order to determine the highest card, the cards in the deck in increasing order are as follows: `Club Ace`, `Diamond Ace`, `Heart Ace`, `Spade Ace`, `Two Clubs`, `Two Diamonds`, `Two Hearts`, `Two Spades`, `Three Clubs`, and so on, upto `King Spades`. The method `findMax` has four parameters. The first is an object of type `deckOfCards` and

the last is a **functor** that implements how to compare cards in the deck. The second parameter is the player number. The third is the number of players being dealt cards (in this case, this is 4).

Finally you need to write a main program that creates a new deck of cards, shuffles the deck 7 times, and after each shuffle reports both the location of the special card, as well as the highest card in the hand of player number 2 (assuming that the cards are dealt after that shuffle).

What to Submit

Do not print out the contents of the decks after each shuffle, although you may want to add a private method to do so for debugging purposes. Make sure to have a good comments section and run your code through Javadoc. Use runtime exceptions to check parameters of `findMax`. Print out the source code and output; also submit the source code on a labeled floppy disk. At the head of the program you should have your name and class information. Your program is due at the start of the class, not at the end of the class. If you cannot come to class, you need to slide it under my office door at least 15 minutes before class.

Academic Misconduct

You may talk to your friends about your homework. But the programming effort has to be your own. For every programming assignment, print out a statement at the top of the program stating that "the program is your work and you did not acquire it or part of it from elsewhere". Also, sign the above statement. If you have questions about what constitutes academic dishonesty, please consult the FIU web site at URL: `http://www.fiu.edu/provost/polman/sec2/sec2web2-44.htm`

Details

Here is the specification that you need to use. Note that the methods refer to the generic `Object` and uses **functors** to implement `findMax`:

```
public interface Deck
{
    public void perfectShuffle();
    public int locateSpecial();
    public Object findMax( Object [] a, int startIndex, int incrIndex, Comparator cmp )
}

import java.util.Comparator;
class aCard
{
     /* Some constructor to be implemented here */
     public String toString()        { /* Implementation not shown */ }
     public boolean equals(Object rhs) { /* Implementation not shown */ }
     public Object getCard()    { /* Implementation not shown */ }
     /* The private data field(s) here */
     private int cardCode;
}
```

```
class deckOfCards implements Deck
{
    /* Some constructor to be implemented here */
    public void perfectShuffle()    { /* Implementation not shown */ }
    public int locateSpecial()    { /* Implementation not shown */ }
    public Object findMax( Object [] a, int startIndex,
       int incrIndex, Comparator cmp ) { /* Implementation not shown */ }
    public Object getDeck()    { /* Implementation not shown */ }
    /* The private data field(s) here */
}

class ShuffleTest
{
    private static class usingPerfectOrder implements Comparator
    {
        public int compare( Object obj1, Object obj2 )
        { /* Implementation not shown */ }
    }
    public static void main( String [ ] args )
    { /* Implementation not shown */ }
}
```

Extra Credit

For extra credit you may try your hand at any of the various problems listed below. Make sure your documentation states clearly which of these problems you have correctly implemented. It is your responsibility to prove to me that your implementations of these parts work correctly.

1. Add a method to generate a "random shuffle", i.e., a random permutation of the cards. Your documentation should explain clearly the algorithm you are using to generate such a shuffled deck.

2. Write a program to count the number of "random shuffles" you need before the "Heart Queen" is at the top of the deck.

3. Write a program to count the number of "random shuffles" before player number 1 gets at least 6 cards from the Diamonds suit.