Card Deck Case Study #1

When we have a small number of literal values that we want to keep in array, we can declare and initialize it by listing the values between curly braces, separated by commas. For example, we might use the following code in a program that processes playing cards.

```
String[] suit = { "Clubs", "Diamonds", "Hearts", "Spades" };
String[] rank = { "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack",
"Queen", "King", "Ace" };
```

After creating the two arrays, we can use them to print out a random card name, such as Queen of Clubs, as follows:

```
int i = (int) (Math.random() * rank.length);
int j = (int) (Math.random() * suit.length);
System.out.println(rank[i] + " of " + suit[j]);
```

This code generates random indices and then uses the indices to pick strings out of the arrays. Whenever the values of all array entries are known at compile time (and the size of the array is not too large) it makes sense to use this method of initializing the array—just put all the values in braces on the right hand side of an assignment in the array declaration. Doing so implies array creation, so the new keyword is not needed.

A more typical situation is when we wish to compute the values to be stored in an array. In this case, we can use array names with indices in the same way we use variable names on the left side of assignment statements. For example, we might use the following code to initialize an array of size 52 that represents a deck of playing cards, using the two arrays just defined:

```
String[] deck = new String[suit.length * rank.length];
for (int i = 0; i < suit.length; i++)
 for (int j = 0; j < rank.length; j++)
 deck[rank.length*i + j] = rank[i] + " of " + suit[j];
```

After this code has been executed, if you were to print out the contents of deck in order from deck[0] through deck[51] using System.out.println(), you would get the sequence

```
2 of Clubs
2 of Diamonds
2 of Hearts
2 of Spades
3 of Clubs
3 of Diamonds
...
Ace of Hearts
Ace of Spades
```

Card Deck Case Study #1

Frequently, we wish to exchange two values in an array. Continuing our example with playing cards, the following code exchanges the cards at position i and j:

```
String t = deck[i];
deck[i] = deck[j];
deck[j] = t;
```

When we use this code, we are assured that we are perhaps changing the order of the values in the array but not the set of values in the array. When i and j are equal, the array is unchanged. When i and j are not equal, the values a[i] and a[j] are found in different places in the array. For example, if we were to use this code with i equal to 1 and j equal to 4 in the deck array of the previous example, it would leave 3 of Clubs in deck[1] and 2 of Diamonds in deck[4].

The following code shuffles our deck of cards:

```
int N = deck.length;
for (int i = 0; i < N; i++)
{
 int r = i + (int) (Math.random() * (N-i));
 String t = deck[i];
 deck[i] = deck[r];
 deck[r] = t;
}
```

Proceeding from left to right, we pick a random card from deck[i] through deck[N-1] (each card equally likely) and exchange it with deck[i]. This code is more sophisticated than it might seem: First, we ensure that the cards in the deck after the shuffle are the same as the cards in the deck before the shuffle. Second, we ensure that the shuffle is random by choosing uniformly from the cards not yet chosen.