COLBY COLLEGE
Department of Mathematics
and Computer Science

CS115
Spring 1998

Allen B. Downey

Assignment 12: Arrays of objects

Due: Friday 7 May

Congratulations and welcome to the last assignment of the semester. I know the last few weeks of class are crazy, but I hope you will all have a chance to work on this assignment a bit, since it contains what I think is some of the most interesting material.

Again, I have to warn you that there is information you will need to do this assignment that we have not got to in lecture yet, so if you get stuck, hang tight and we will get to it.

Also, this assignment builds on top of the Card class from Homework 10. If you are not satisfied with your solution to that assignment, feel free to use my version from the class web page.

Exercises from the book

In Chapter 12 there are several methods that I sketch but don't fill in completely. You should type them in and clean them up:

1. Modify `printCard` so that it uses an array of Strings instead of a chained conditional.
2. Type in `sameCard` and test it. Make sure you are clear on the difference between shallow and deep equality.
3. Use the code in Section 12.6 to write a method called `buildDeck` that creates and returns a fully-populated array of cards.
4. Write a method called `printDeck` that takes an array of cards and prints them one per line.
5. Type in and test `findCard`.

Shuffling and sorting

1. Write a method called `swapCards` that takes a deck (array of cards) and two indices, and that switches the cards at those two locations.

   HINT: it should switch the references to the two cards, rather than the contents of the two objects. This is not only faster, but it makes it easier to deal with programs in which cards are aliased.
2. Write a method called `shuffleDeck` that uses the algorithm in Section 12.8. You can use the `randInt` method from the solutions to Homework 11. Look it over carefully, it may be different from what you wrote.
3. Write method called `findLowestCard` that uses the `compareCard` method to find the lowest card in a given range of the deck (from `low` to `high`, including both).
4. Write a method called `sortDeck` that arranges a deck of cards from lowest to highest.

Merge

1. Type in the `subdeck` method from the alternate version of Chapter 13, and test it.
2. Using the pseudocode in Chapter 13, write the method called `merge`.
3. Write the simple version of `mergeSort`, the one that divides the deck in half and then uses `sortDeck` to sort the two halves. Then it uses merge to create a new, fully-sorted deck.

   Note that in previous assignments when we "broke" arrays in half, we just used index ranges to specify the halves. We didn't actually create halves. For `mergeSort` we actually create new decks and subdecks using `subdeck` and `merge`.
4. Write the fully recursive version of `mergeSort`.