4 Understanding Complex Data

Portfolio Problems

Problems:

1. Problem 15.8 on page 175
2. Problem 15.11 on page 176
3. This problem continues the work on mobiles we have started during one of the earlier lectures. The file mobile-methods-lecture.java contains the data definitions, examples of data, and the method totalWeight. Design the method draw() that consumes a Canvas and a Posn that represents the point where the top of the mobile will hang. The method draws the mobile with black lines for the struts, and for the hanging lines. For a simple mobile, there should be a disk of the appropriate color and with the size proportionate to its weight shown at the end of the line.

Pair Programming Assignment

4.1 Problem

Warm up by finishing the problems from Lab 3 that dealt with lists of Strings. Then work out the following problems:

A. Design the method maxLength that computes the maximum length of all Strings in a list of Strings. If the list is empty, the result should be -1.

B. Convert the method maxLength to the accumulator style — name the new method maxLength2 and the helper method maxLengthAcc.

C. Design the method shortWords that produces a list of all Strings that are shorter than the given number.

D. Design the method startingWith that produces a list of all words that start with the given letter. Provide the starting letter as a String of length one — for example "c" or "Z". Java String class defines the following method:
4.2 Problem

Finish the work on the second part of Lab 4, dealing with employee company chart. Make sure you understand what information the data represents. Then work out the following problems:

A. We would like to rank all employees as follows. The worker at the bottom, who is not a boss of any other workers, has rank one. The rank of every boss is defined as one more than the number of levels of subordinates. For the example given in the lab, Mike has rank 2, Dave has rank 4.

Design the method rank that determines the rank of an employee.

B. Design the method sortByRank that produces a list of employees sorted by their rank from a list of employees. Of course, there will be some workers of the same rank. So, the resulting list may look like this:

```((Jack rank 3) (Jill rank 2) (Joe rank 2) (Jenny rank 1) (Jesse rank 1) (Jodi rank 1))```

Of course, each entry would be the instance of Emp.