4 Methods and Equality for Self Referential Data

4.1 Problem

Recall the Homework 3, Problem 3.7 that dealt with the items in the grocery store. Design the following methods:

1. *same*, which determines whether two grocery items are the same;
2. *cheaperItem*, which produces the grocery item that is cheaper, given in terms of the unit cost. This method consumes only one argument. ∎

4.2 Problem

Recall the Homework 3, Problem 3.8 that dealt with reading lists.

Design the following methods for these classes:

1. *same*, which determines whether two lists of books contain the same books in the same order.
2. *sort*, which sorts the list of books by year.
3. *oldBooks*, which produces a list of books published before 1950.
4. *contains*, which determines whether a list of books contains a given book. ∎

4.3 Problem

Recall the Homework 3, Problem 3.3 that dealt with shapes. Extend the class hierarchy with a new subclass that represents a combination of two shapes, layered on top of each other.

The locations of the two shapes in the combinations are given relative to the location of the combined shape. For instance, if the location of the dot is given as (20, 40), and the location of the combo shape that has the dot as one of its two layers is (30, 50), then if the combined shape is drawn, the location of the dot relative to the graphics window origin will be (50, 90).

Design the following methods for these classes:

1. *same*, which determines whether two shapes are the same.
2. *move*, which produces a new shape moved by the specified distance (the change in the x and y coordinates). ∎

4.4 Problem

A GUI *Component* is one of the following:

- Checkbox
- Textfield
- OptionsView
- ColorView
- Table

The first four components are *primitive* components, while the *Table* is a *compound* component. Each component contains a label and some additional data.

- The data for a *Table* is a list of *Row*s. A *Row* is a list of *Component*s.
- Data for each of the *primitive* components consists of the default value to be displayed, specified as a String, and also the preferred width and height for this component.

1. Draw the class diagram for this collection of classes.
2. Design the methods needed to count the number of *primitive* GUI components in a given GUI *Component*.
3. Design the methods needed to determine the height of a given GUI *Component*. The height of a table is the sum of the heights of the *Row*s. The height of a *Row* is the maximum of its components' heights. ∎

4.5 Problem

The following class diagram defines data that represents web pages.

```
        +----------------------------------+
        |                                  |
        v                                  |
+---------------+                          |
| WebPage       |                          |
+---------------+                          |
| String header |                          |
| ALoHTML body  |--+                       |
+---------------+  |  +----------------+   |
                   v  v                |   |
              +---------+              |   |
              | ALoHTML |              |   |
              +---------+              |   |
              +---------+              |   |
                 / \                   |   |
                 ---                   |   |
                  |                    |   |
       -------------------             |   |
       |                 |             |   |
  +----------+     +--------------+    |   |
  | MTLoHTML |     | ConsLoHTML   |    |   |
  +----------+     +--------------+    |   |
  +----------+  +--| AHTML first  |    |   |
                |  | ALoHTML rest |----+   |
                |  +--------------+        |
                v                          |
       +-------------+                     |
       | AHTML       |                     |
       +-------------+                     |
       | String word |                     |
       +-------------+                     |
             / \                           |
             ---                           |
              |                            |
     ----------------                      |
     |              |                      |
  +------+    +------------+               |
  | Word |    | Link       |               |
  +------+    +------------+               |
  +------+    | WebPage wp |---------------+
              +------------+
```

1. Translate this data definition into Java classes and make examples of the data.

2. Design the method *allWords*, which produces a list of all words in the web page (all strings, including the header string).

3. Design the method *pages*, which produces the list of *immediate* words on a page. That is, it consumes a *WebPage* and produces a list of String. An *immediate* word on a list of *HTML* elements is defined as follows:

   - an *HTML* element that is a *Word* the method extracts the *word* from the *Word*.
   - for an *HTML* element that is a *Link*, the method extracts the *word* from the *Link*.

4. Design the method *occurs*, which determines whether the given *word* occurs in the web page or its embedded pages. ∎