4 Understanding Complex Data

Portfolio Problems

Problems:

1. Problem 15.8 on page 175
2. This problem continues the work on mobiles we have started during the lab.

   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.
3. This problem continues the work on lists of `String`s we have started during the lab. Design the following two methods — use a helper method with accumulator and make sure the purpose statement explains the meaning of the accumulator:

   ```
   // combine all Strings in this list into one
   String combine();

   // find the length of the longest word in this list
   int maxLength();
   ```

   For the second method you will need to know that the class `String` defines the method

   ```
   // compute the length of this String
   int length();
   ```

Pair Programming Assignment

4.1 Problem

Warm up by finishing the problems from Lab 4 that dealt with lists of `String`s. Then work out the following problems:

A. Design the method `shortWords` that produces a list of all `String`s that are shorter than the given number.

B. 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:

```
// does this String starts with the given String
boolean startsWith(String s)
```

4.2 Problem

Start with the file **ExcelCells.java**.

For this problem you will use the classes that represent the values in the cells of a spreadsheet. For each cell we record the row and column where the cell is located, and the data stored in that cell. The data can either be a numerical (integer) value or a formula. Each formula can be one of three possible functions: `+` (representing addition), `mn` (producing the minimum of the two cells), or `*` (computing the product) and involves two other cells in the computation.

A. Make an example of the following spreadsheet segment:

```
   |     A    |     B    |     C    |     D    |     E     |
---+----------+----------+----------+----------+-----------+
1  |    8     |     3    |     4    |     6    |     2     |
---+----------+----------+----------+----------+-----------+
2  | mn A1 E1 | + B1 C1  |          |          | * B2 D1   |
---+----------+----------+----------+----------+-----------+
3  | * A1 A2  | + B2 B1  |          |          | mn A3 D1  |
---+----------+----------+----------+----------+----------+
4  |          | + B3 B2  |          |          | mn B4 D1  |
---+----------+----------+----------+----------+-----------+
5  |          | + B4 B3  |          |          | * B5 E4   |
---+----------+----------+----------+----------+-----------+
```

B. Draw on paper this spreadsheet and fill in the values that should show in each cell.

C. Design the method `value` that computes the value of this cell.

D. Design the method `countFun` that computes the number of function applications needed to compute the value of this cell.

E. Design the method `countPlus` that computes the number of `Plus` applications needed to compute the value of this cell.

   Make sure you design templates, use helper methods, and follow the containment and inheritance arrows in the diagram.

4.3 Problem

Revise the solution to the problem from last week that dealt with bank accounts.

A. Define an abstract class `AAccount` and lift into it all fields that are common to all accounts.

B. Revise the method `amtAvailable` for the classes that represent bank accounts: can it be lifted to the abstract class? - or does it have to be defined in each class anyway?

C. Revise the method `moreAvailable` that determines whether one account has more available for withdrawal than another account: can it be lifted to the abstract class? - or does it have to be defined in each class anyway?

D. Revise the method `withdraw` that produces a new account with the given amount withdrawn: can it be lifted to the abstract class? - or does it have to be defined in each class anyway?

   Note: Later we will learn how we can signal that the transaction is not valid.

E. Define the method `sameName` that determines whether two accounts have the same name.

4.4 Problem

Make a final revision of your game. Where appropriate, add lists of game components, or other collection of objects.

The grading rubric for this problem will be as follows:

- 4 points — well designed and readable data definitions and code

- 4 points — examples of world at the start, at the end, and during the game
- 4 points — well designed methods, the design follows one task - one method rule, the methods are defined in the appropriate classes
- 4 points — tests for all methods