2 Self-Referential Data; Methods for Classes.

Portfolio Problems

- **FEDEX shipping**

  Work out the problem 6.1 in the textbook.

- **Method for FEDEX shipping**

  Design the following methods for the shipment packages:

  - Determine whether this package weighs less than some given weight. (Method name *lighterThan*)
  - Determine the total volume of the package. (Method name *totalVolume*)
  - Is one package going to the same customer as another package? (Method name *sameCustomer*)
  - Looking at this package and given another package, produce the one that has the smaller volume. (Method name *getSmaller*)

- **Soccer Team**

  Work out the problem 5.9 in the textbook.

Pair Programming Assignment

Data for a company organizational chart

2.1 Problem

Work out the problem 6.7 in the textbook.

TiVo recordings

All problems in this part become one Beginner ProfessorJ program. Use the data definitions and examples from your previous homework whenever appropriate.

We continue working with the classes that help us set up the recording of TV shows.

2.2 Problem

The program that keeps track of the programs you wish to record has to make sure your choices are valid. To do so, it needs to be able to answer the questions given below:

To make sure your data definitions are flexible enough, make sure that the date of the show and the time of the show are represented as two different fields. The class *ShowTime* then combines the day and the time of the show. Finally, the class *TVshow* also includes the title of the show and the channel on which it is being shown.

As you work through the problems, pay attention to the one task one method rule and delegate the work to the class that should be responsible for providing the answer.

1. What is the total recording time for a show?

   Decompose this problem into the following parts:

   - Design the method *minutes* in the class *ClockTime* that computes the minutes since midnight that this time instance represents.
   - Design the method *duration* in the class *TimeSlice* that computes the duration of this time slice in minutes.
   - Design the method *duration* in the class *ShowTime* that computes the duration of this show in minutes.
   - Design the method *recordingTime* in the class *TVshow* that determines the recording time for this show.

   Once you are comfortable with designing methods, you may combine some of these steps. The purpose of this exercise is to show how to properly delegate the responsibility for a task to the class that can correctly handle the request. This design allows us to change the way the time is represented, or the way that the time slice is represented, without affecting the rest of the program. For example, the time slice may be represented by its starting time and its duration in minutes — yet the *TVshow* class will not be affected by this change.

   Note: In the remaining exercises in this part make sure you follow a similar design strategy as above: each task should be delegated to the class that can correctly handle the request. At this point, no *shortcuts* are allowed.

2. Is the time when one show ends before the time when another show begins? (Method name: *endsBefore*)

3. Does one show starts before another show starts? (Method name: *startsBefore*)

4. Is one show the same as another one, other than the time for the recording? (Method name: *sameShow*)

5. You find out that your favorite show has been rescheduled and you know the new starting and ending time. Design the method that produces the new show time recording request from the original one. (Method name: *newTime*)