# 2 Self-Referential Data; Methods for Simple Classes.

## Etudes

### 2.1 Etude

Work out the problem 6.1 in the textbook. ∎

### 2.2 Etude

Design the following methods for the shipment packages:

- Determine whether this package weighs less than some given weight.
- Determine the total volume of the package.
- Is one package going to the same customer as another package?
- Looking at this package and given another package, produce the one that has the smaller volume.

∎

## Main Assignment — Part 1

### 2.3 Problem (6.9)

Consider the following puzzle:

> A number of people want to cross a dilapidated bridge at night. Each person requires a different amount of time to cross the bridge. Together they have one battery-powered flashlight. Only two people can be on the bridge at any given time, due to its bad state of repair.
>
> Given the composition of the group and the life-time of the battery, the problem is to determine whether and how the entire group can cross the bridge.

Solving the puzzle (manually or with a program) requires a recording of what happens as members of the group moves back and forth across the river. Let's call the status of the "puzzle world" after each individual crossing a *state*.

Design a data representation for the states of the puzzle. Which elements does a state have to record? What is the initial state for this problem? Express it in your representation. What is the final state for this problem? ∎

## Main Assignment — Part 2

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

### 2.4 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 following questions:

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 recording request from the original one. (Method name: *newTime*)

∎

## Main Assignment — Part 3

We continue working with the classes that help us draw the city map and its attractions.

### 2.5 Problem

Besides hotels, museums, and hospitals, the map should also show other points of interest. Design two additional classes that represent schools and theaters. You may choose what information you want to provide for each, besides the name and its location. ∎

### 2.6 Problem

To navigate through the map we need to be able to answer the following questions:

- How far is this feature from some location on the map? (Method name *distanceTo*)

- How far is this feature on the map from another feature? (Method name *distanceFrom*)
- You want to know which of the two hotels (or for that matter, any two points of interest) is closer to the point of interest at which you are right now. The method should produce the closer point of interest. You need to think of which class will invoke the method. (Method name *findCloser*)