4 Methods for unions, self-referential and mutually referential data

4.1 Problem (12.6)

Collect and finish the *RoomEditor* class and connect it to the shape classes. ■

4.2 Problem (12.8)

Modify the presentation of the room so that it fits within the canvas and there is 20 point margin around the room. Make the margin black. ■

4.3 Problem (12.9)

Recall problem 2.1 (exercise 4.5 in the book):

> Develop a program that creates an on-line gallery from three different kinds of records: images (gif), texts (txt), and sounds (mp3). All have names for source files and sizes (number of bytes). Images also include information about the height, the width, and the quality of the image. Texts specify the number of lines needed for visual representation. Sounds include information about the playing time of the recording, given in seconds.

Develop the following methods for this program:

1. *timeToDownload*, which computes how long it takes to download a file at some given network connection speed (in bytes per second);
2. *smallerThan*, which determines whether the file is smaller than some given maximum size;
3. *sameName*, which determines whether the name of a file is the same as some given name. ■

4.4 Problem (13.1)

Collect all the pieces of *oneMonth* and insert the method definitions in the class hierarchy for logs. Develop examples and include them with the test suite. Draw the class diagram for this hierarchy. ■

4.5 Problem (13.2)

Suppose the requirements for the program that tracks a runner's log includes this request:

> The runner wants to know the total distance run in a given month.

Design the method that computes this number and add it to the class hierarchy of problem 4.4 (exercise 13.1 in the book).

Consider designing two different versions. The first should just follow the design recipe. The second should take into account that methods can compose existing methods and that this particular task can be seen as consisting of two separate tasks. Where would you put the second kind of method definition in this case? (See the next chapter.) ∎

4.6 Problem (13.4)

Suppose the requirements for the program that tracks a runner's log includes this request:

> The runner would like to see the log with entries ordered according to the pace computed in minutes per mile in each run, from the fastest to the slowest.

Design this sorting method.

Design the sorting method to sort the log with entries ordered according to the date of the entry. ∎

4.7 Problem (13.11 – OPTIONAL)

Figure 1 contains a class diagram that describes the GUI hierarchy of exercise 6.5 in the book. Note that one of its interfaces, *ITable*, refines another interface *IGUIComponent*.

Add templates to each box in this diagram. Do not assume anything about the return type; do not design classes. ∎

4.8 Problem

Modify the classes from the problem 3.1 to represent only the arithmetic expressions, as shown in the class diagram in figure 2. Design the method eval that produces an instance of *IntVal* that represents the integer value of the arithmetic expression. ∎

```
                                     +---------------+
                                     | IGUIComponent |  <---------------------------------+
                                     +---------------+                                    |
                                     +---------------+                                    |
                                             |                                            |
                                            / \                                           |
                                            ---                                           |
                                             |                                            |
       +--------------------+--------------------+-------------------+---------- ...      |
       |                    |                    |                   |                    |
+--------------+    +----------------+    +--------------+    +--------------+            |
| BooleanView  |    | TextFieldView  |    | OptionsView  |    | ColorView    |            |
+--------------+    +----------------+    +--------------+    +--------------+            |
| String label |    | String label   |    | String label |    | String label |            |
+--------------+    +----------------+    +--------------+    +--------------+            |
                                                                                          |
                                                                                          |
  ... ---------+                                                                          |
               |                                                                          |
           +--------+                                                                     |
           | ITable |<-------------+                                                      |
           +--------+              |                                                      |
           +--------+              |                                                      |
               |                   |                                                      |
              / \                  |                                                      |
              ---                  |                                                      |
               |                   |                                                      |
     ------------------            |                                                      |
     |                |            |                         +------+                     |
+---------+    +--------------+    |          +------------> | IRow |<-------------+      |
| MTTable |    | AddRow       |    |          |              +------+              |      |
+---------+    +--------------+    |          |              +------+              |      |
+---------+    | Table rest   |----+          |                 |                  |      |
               | IRow  first  |---------------+                / \                 |      |
               +--------------+                                ---                 |      |
                                                                |                  |      |
                                                         +------------------+      |      |
                                                         |                  |      |      |
                                                +------------------+    +-------+  |      |
                                                | ConsRow          |    | MTRow |  |      |
                                                +------------------+    +-------+  |      |
                                                |                  |    +-------+  |      |
                                                | GUIComponent fst |----------------------+
                                                | IRow rst         |---------------+
                                                +------------------+
```

Figure 1: A hierarchy of GUI components

4.9 Problem

Given a Java program as defined in the Problem 3.2 determine whether a given class is correctly defined. Divide the task into the following steps:

1. Determine whether a given name is a valid type name in a program.

2. Verify that all interfaces defined in a class within a given program are valid interface name. Use a helper that determines whether every interface name in a list of interface definitions is contained in the given list of interfaces. **Make sure you follow the design recipe.**

```
         +------+
         | IExp |<----------------------------+
         +------+                             |
           / \                                |
           ---                                |
            |                                 |
       ----------------------------           |
       |                          |           |
+-----------+          +-----------------+    |
| IntVal    |          | OpExp           |    |
+-----------+          +-----------------+    |
| int value |        +-| IOperator rator |    |
+-----------+        | | IExp rand1      |----+
                     | | IExp rand2      |----+
                     | +-----------------+
                     |
                     |
                     v
                +-----------+
                | IOperator |
                +-----------+
                     / \
                     ---
                      |
               ------------------
               |                |
         +-----------+    +-----------+
         | PlusOp    |    | TimesOp   |
         +-----------+    +-----------+
         | String op |    | String op |
         +-----------+    +-----------+
```

Figure 2: Representing arithmetic expressions

3. Verify that in a class every field type represents a valid type in the given program. Also make sure that the type of the field is not the same as the name of the class in which it is defined.

4. Combine the above methods to verify that a class definition is valid in the given program.

Note: This does not constitute a complete set of condition for a correct program definition. Additionally, we would also have to make sure no type names are repeated, and no field names are repeated within a given class definition. There may be other concerns as well.. ∎

4.10 Writing Problem

Writing assignments are separate from the rest of the assignment for the week. You should work on this assignment alone, and hand in your work typed on a paper before the first lecture that follows the due date for the rest of the homework. The answer should be about two paragraphs long — not to exceed half a page or

300 words.

Software errors can be very costly - in term of financial losses, violating person's privacy, or even physical harm or death. Find two examples of the situations where the software error caused such damage. Describe each error, its impact, and how it could have been prevented. Include references to the sources you consulted. ∎