Serie 5

Content:
– Design Patterns : Decorator - Iterator - JUnit : Dispenser

(19) Design Pattern : Decorator - Starbuzz Coffee

Read and study Chapter 3 of [2] ; solve the three proposed exercices (i.e. the "Sharpen your pencil" boxes).

– Improve the solution proposed in [2] (on page 107) for handling different sizes of beverages. One would like not to use the `if...else if...else if...` block in the method `cost()`.
– Draw the class diagram of your final solution.
– Draw a sequence diagram showing the collaborations that take place when the method `cost()` is invoked on a "Decaf - Milk - Moccha".
– Draw a sequence diagram showing the collaborations that take place when the method `getSize()` is invoked on a "Decaf - Milk - Moccha".
This chapter is freely available at O'Reilly [4] and a copy can also be found on [3], as well as a slighty adapted version of the source code.

(16) Design Pattern : Iterator - Traversing a two-dimensional matrix

The `Matrix` class implements a simple matrix.

One would like to access this data structure sequentially, but in two different manners.

1. Develop two iterators :
   – one accessing the data structure in "Row-Column" (`iterator.RowColumnIterator`) way,
   – and the other in the "Column-Row" (`iterator.ColumnRowIterator`) way.
   Devise the iterators in a way that other iterators can be easily added using the `Iterator` interface.
2. Add the methods related to the creation of iterators to the class `Matrix`.
   Warning : apart from the creation of the iterators, no other operations of the class `Matrix` can be modified or added.
3. Complete the `TestMatrix` program so that its execution provides the following result (for a 3x2 matrix) :
   – `1-1 1-2 2-1 2-2 3-1 3-2` if the matrix provides a "Row-Column" iterator, and
   – `1-1 2-1 3-1 1-2 2-2 3-2` if the matrix provides a "Column-Row" iterator.

On [3] you can find the base code for following classes (also see Figure 1) :
– the class `MatrixTest`
– the class `matrix.Matrix`
– the interface `iterator.Iterator`
– the interface `iterator.Iterable`

(24) Dispenser JUnit

Based upon the solution of the dispenser add unit tests [1] for each implementation of the ADT DISPENSER[G], respecting the naming convention of the lecture.

1. Tests are located in the `src/junittest` directory.
2. Dispenser implementations are located in the `src/main` directory.
3. Import the given code into Netbeans.
4. Don't forget to add the `src/junittest` directory to the test package folder.
5. Add the following dependencies as *Java Sources Classpath* :
   – Add the JUnit jar file (`junit-4.8.1.jar`) of your `libs/SoftEngLibs/junit4.8.1`.
   – Add the `src/main` directory.

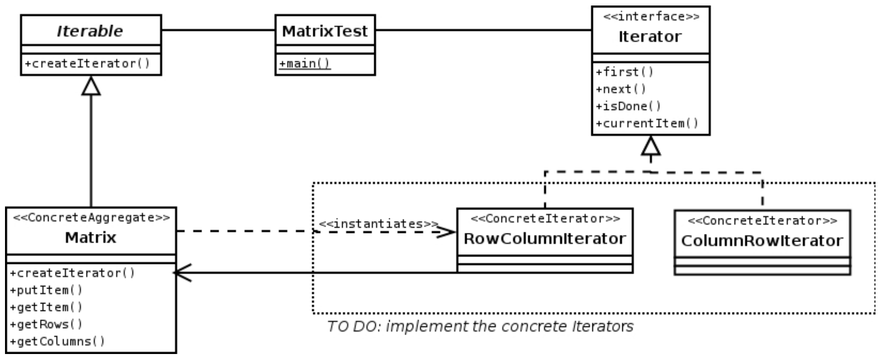

Figure 1 – Class diagram for the matrix iterator.

6. For each implementation of a stack create a class in the corresponding package which does the unit test for the class.
7. Add enough unit test for testing a maximum of code.
8. You can also try the coverage-report target to see how well your code is tested.

Références

[1] JUnit, 2011. http://www.junit.org (accessed April 17, 2012).

[2] Eric FREEMAN, Elisabeth FREEMAN, Kathy SIERRA, and Bert BATES. *Head First Design Patterns*. O'Reilly, 2004.

[3] Jacques Pasquier. Génie logiciel I, 2013. http://moodle2.unifr.ch/course/view.php?id=1252 (accessed Apr 17, 2013).

[4] O'Reilly Store. *Head First Desing Patterns*. O'Reilly, 2006. http://www.oreilly.com/catalog/hfdesignpat/ (accessed April 17, 2012).