Design Patterns - Abstract Factory

July 4, 2020

We would like to build a set of classes in order to design computing algorithms. These algorithms will be used by an application in a transparent way. More we would like to integrate a mechanism allowing to minimize the number of calculus when we call them several times.

1 Abstract Produce

- Give a declaration of the interface `Algorithm`. The methods which belong to this class are : `String getName()` and `double getVal()`. Give the implementation of these two classes which implement the interface `Algorithm` :

  1. `class Fibonacci` $(U(I) = U(I-1) + U(I-2), U(0) = 1, U(1) = 1)$
  2. `class Padovan` $(U(I) = U(I-2) + U(I-3), U(0) = 1, U(1) = 1, U(2) = 1)$

2 Lazy Evaluation

By using delegation and without modification of the previous algorithm implementation, we would like to build a version of the two algorithms which use a memory buffer. The values already computed are store in memory (RAM) and when the function `getVal(int)` will be called the result will be return directly without new computing.

In the main program, there are no difference comparing to the original version, the objects in the buffer memory will be considered as the objects coming from previous algorithms and this version of Padovan algorithm with buffer (cache) memory will work as Padovan without cache (idem for Fibonacci).

- Implement your solution
- Write a `junit` test which verifies if 2 instances of the `Algorithm` class return the same results for the first 10 integers (from 0 to 9), Use your test to verify that your implemetation with the cache works well.

3 Abstract Factory

We would like to hide to the client which algorithm type he uses (with or without cache). By using an `abstract factory`, you will propose and justify a solution to do that. Only the 2 algorithms Fibonacci and Padovan will be considered.

4 Introspection

By using introspection mechanism coming from java (see documentation of the class `Class`) give an implementation of a program which instantiate a factory depending on the name of the concrete class given as parameter (on the command line) then which compute the value of Fibonacci of the rank 5.

For example, the following line will create a class with the name XXXX and will use it to instanciate a class which computes Fibonacci with the value 5 and print the result.

```
java Exemple XXXX 5
```