Pattern Report

PIETRO VALENTE

Summary

The only pattern used in this project is the *adapter pattern*.

This pattern is divided into two facets and both have been used: Class Adapter and Object Adapter.

The characteristic of the *Class Adapter* is to take advantage of inheritance to reuse code already present, adapting it to the need. This pattern was used for the *subList* methods of the *ListAdapter* and the *entrySet*, *keySet*, *values methods* of the *MapAdapter*; in fact, for each of these methods, a private class was created and an instance is returned.

Since by definition the *Class Adapter* cannot wrap an interface, since it must always derive from a base class; the classes linked to these methods extend the corresponding target classes (of the project), which in turn implement the required interfaces.

This solution was chosen, for these classes, because the signature of the methods is the same. So, this made it possible not to have to overwrite some methods that were limited to using other methods present in the class, in essence those that did not refer to private members. This alternative in fact exploits the dynamic binding of the Java language, that is, the behavior that if an inherited but not overwritten method invokes one that has been overwritten, the overwritten version is used.

The *Class Adapter* is a very good and useful solution if the signature of the methods is quite similar. Otherwise, you should override all methods not in the new target class, throwing an exception at it. In such a situation it is more convenient to use an *Object Adapter*.

The *Object Adapter* uses composition and can wrap classes or interfaces, or both. It can do this because it contains, as a private, encapsulated member, the instance of the class object or interface it wraps.

While the *Class Adapter* exploits inheritance, so the subclass is always also an instance of its superclass; the *Object Adapter* takes advantage of the composition, which guarantees greater freedom, in fact the new target class will be able to show only what will be necessary and established of the instance of the class present as a private member.

For this reason, this alternative was chosen for the target classes of the project: *ListAdapter, MapAdapter, SetAdapter* maintain an instance of their adaptee established (*Hashtable* for *MapAdapter* and *SetAdapter*, *Vector* for *ListAdapter*).

Having noticed that the behavior of the *CollectionAdapter* is the same as the *ListAdapter*, only with less functionality, the adaptee used for the *CollectionAdapter* is the *ListAdapter*. With a similar reasoning it was chosen to apply this adapter also to *VectorIterator* using as adaptee *ListIteratorAdapter*.