DISTRIBUTED SYSTEMS

Università Degli Studi Di Parma

Prof Agostino Poggi

Lorenzo Rotteglia • Matr. 219734 • lorenzo.rotteglia@studenti.unipr.it • 24 aprile 2013

IMMUTABLE OBJECT INTERFACE AND VALIDATION

Report

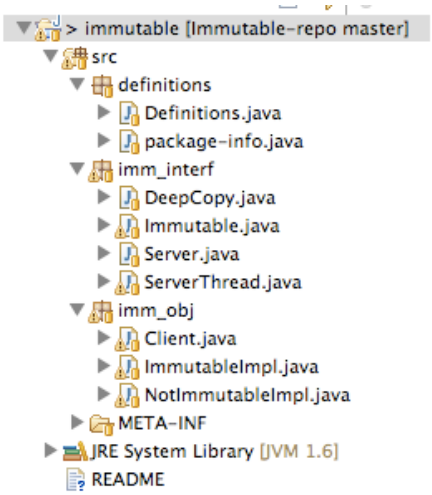
This project aims to provide an interface and a validation methodology to build immutable objects.

A STRATEGY FOR DEFINING IMMUTABLE OBJECTS

The following rules define a simple strategy for creating immutable objects. Not all classes documented as "immutable" follow these rules. This does not necessarily mean the creators of these classes were sloppy – they may have good reason for believing that instances of their classes never change after construction. However, such strategies require sophisticated analysis and are not for beginners.

- Don't provide "setter" methods – methods that modify fields or objects referred to by fields.
- Make all fields final and private.
- Don't allow subclasses to override methods. The simplest way to do this is to declare the class as `final`. A more sophisticated approach is to make the sonstructor private and construct instances in factory methods.
- If the instance fields include references to mutable objects, don't allow those objects to be changed:
    - Don't provide methods that modify the mutable objects.
    - Don't share references to the mutable objects. Never store references to external, mutable objects passed to the constructor; if necessary, create copies, and store references to the copies. Similarly, create copies of your internal mutable objects when necessary to avoid returning the originals in your methods.

SOFTWARE REALIZED FOR THE PROJECT



Package explorer

The picture above represent the content of the distributed archive.

It is possible to download a read-only repository with the distributed code at the following link : `git://github.com/duga9cu/Immutable-repo.git`

Software description

The project contains two core programs : a client and a server. They are named with remarkable creativity `Client.java` and `Server.java`

The client program is in charge of instantiate some sort of object and send it to the server. The imm_obj package contains already two predefined objects, one immutable and the other not immutable, again with the creative names: `ImmutableImpl.java` and `NotImmutableImpl.java`

The server on the other hand is in charge to receive the object and *validate* its "immutability" (these two functionalities are contained in the file `ServerThread.java`) that is to check whether it implements correctly the `Immutable interface`[1].

In the file `Definitions.java` is possible to chose which type of object the client is gonna send.

[1] provided in the imm_interf package, in the file Immutable.java

LAUNCHING THE PROGRAM

To use the software follow these simple step:

1. Select the functioning mode in Definitions.java (true or false)[2]
2. Run Server.java
3. Run Client.java
4. Wait for the results in the shell..

FUNCTIONING OF THE SOFTWARE:

1. the client instantiate an object which implements the interface Immutable.
2. the implementation of the function `validateMyself()` must return the object itself.
3. the client then sends the object willing to validate to the server.
4. the server then accept the client's object and pass it to a thread called serverThread.
5. the serverThread uses reflection and processors to check whether the object is really immutable and print to the shell its response.

For more information about the implementation you can check the javadoc provided with the code.

[2] In step 1 you can set the boolean variable SEND_IMMUTABLE to chose whether to send to the server an `ImmutableImpl` object or a `NotImmutableImpl` object. In either case the server will try and validate the "immutability" of the given object and will print on shell is judgement.