Distributed Programming II

A.Y. 2016/17

Sample Final Test

All the material needed for this test is included in the *.zip* archive where you have found this file. Extract the archive to an empty directory where you will work, and copy your solution of Assignment 3 into this directory.

The test consists of a programming exercise (6 points) and a question (4 points). The exam can be passed only if the programming exercise solution passes the mandatory tests (at least test `testBasicLoadUnloadIsolationPolicy`). In this case, the proposed mark will be the sum of the points got for the test and a base evaluation mark in the range 16-20, assigned on the basis of the evaluation of the submitted assignments (16 points granted because your assignments passed the mandatory tests at submission time and 4 extra points based on the evaluation of one extra aspect of your assignments).

Programming exercise

1. Modify your service developed in Assignment 3 so that, for what concerns point 3) of the set of functionalities specified in part a, it allows, in addition to reachability policies, also an extra type of policy named "isolation policy". An isolation policy specifies that two sets of nodes must be isolated from each other, i.e. no node of the second set must be reachable from a node of the first set. The modified service must allow the creation, removal and updating of isolation and reachability policies. Note that, for what concerns points 4) and 5), related to verification, the service must behave as in the original version (i.e. the service must allow only the verification of reachability policies, not the verification of isolation policies). Add an extra resource to your service, with URL http://localhost:8080/NffgService/rest/totalNumberOfPolicies, representing the total number of policies loaded into the service (including both reachability and isolation policies). This resource must be readable as text/plain via a GET method which must return its decimal representation.

   All the other features of the service must remain unchanged.

2. Extend your client1 so that it implements the extended interface `it.polito.dp2.NFFG.lab3.test0.NFFGClient0`, given in source form (the interface is self-explaining, look at the comments). Create another factory class for your new client1 named `it.polito.dp2.NFFG.sol3.client1.NFFGClient0Factory` that extends the abstract factory `it.polito.dp2.NFFG.lab3.test0.NFFGClient0Factory` and, through the method `newNFFGClient0()`, creates an instance of your concrete class that implements the `it.polito.dp2.NFFG.lab3.test0.NFFGClient0` interface.

3. Modify your client2 so that it can read information about all the policies stored in the service, also including isolation policies. The modified client must implement the same set of interfaces implemented by the original client, but, when returning `PolicyReader` objects, it has to use, in addition to `ReachabilityPolicyReader` and `TraversalPolicyReader`, also the interface `IsolationPolicyReader`, given in source form (in the same package).

Apart from these new specifications, all the specifications given for Assignment 3 still apply.

In particular, all classes must be developed in the same packages used for the solution of Assignment 3 and stored in the same directories as the solution of Assignment 3. Make sure that the

ant script `[root]/sol_build.xml` still works for the compilation of your new solution and modify it as necessary. You can assume that, when your client is compiled and run, your web services have already been deployed.

Question

Explain how, in the code of your service, you avoid race conditions on the number of policies (as the addition of a new policy may be requested concurrently with the operation for reading this number).

The answer to this question must be written in a text file named `[root]/answer.txt`

Correctness verification

In order to pass the exam you have to implement at least points 1. and 2., and your solution must pass at least the mandatory tests that are included in the archive (the original tests of Assignment 3 plus the test `testBasicLoadUnloadIsolationPolicy` from the additional junit test suite `it.polito.dp2.NFFG.lab3.test0.tests.NFFGTests0`). These tests can be run by the ant script `buildTest0.xml` included in the *.zip* file, which also compiles your solution, packages and deploys your service to Tomcat, and runs your clients. The Tomcat server and Neo4J must be both running when the tests are launched. The command for running only the tests specific for this assignment (with the exclusion of the original ones of Assignment 3) is

```
ant -f buildTest0.xml run-tests -Dseed=XXXX
```

The total number of junit tests in this case is 3.

The full set of tests (including the original ones of Assignment 3) can be launched by running the `run-tests-full` target:

```
ant -f buildTest0.xml run-tests-full -Dseed=XXXX
```

As this target may take a long time to complete, initially you are suggested to use the `run-tests` target.

Submission format

A single *.zip* file must be submitted, including all the files that are part of your solution (including the files of your solution of Assignment 3 that you are re-using). The *.zip* file to be submitted must be produced by issuing the following command (from the `[root]` directory):

```
ant -f buildTest0.xml make-final-zip
```

Important: check the contents of the zip file named `solution.zip` after having run the command!