Distributed Programming II

A.Y. 2019/20

Final Test 2

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 (60% of marks) and two questions (40% of marks). The exam can be passed only if the programming exercise solution passes the mandatory tests (see Correctness Verification). The time for the test is 2 hours and 10 minutes.

Programming exercise

1. Modify your service developed in Assignment 3 so that the service also allows clients to add all the items that are in a bookshelf A to another bookshelf B. After this operation items from bookshelf A will belong also to bookshelf B. If this add operation would cause the maximum number of items of bookshelf B to be exceeded, it must fail.

   All the other features of the service must remain unchanged.

2. Create a new client for your service, named Client2, which implements the interface `it.polito.dp2.BIB.ass3.test2.Client2`, given in source form (the interface is self-explaining, look at the comments). As you can see, this interface extends the Client interface used in Assignment 3 by adding a new method for the new add operation. All the classes of your Client2 must be in the package `it.polito.dp2.BIB.sol3.test2.client`. In the same package, create a factory class for your Client2 named `Client2Factory` that extends the abstract factory `it.polito.dp2.BIB.ass3.test2.Client2Factory` and, through the method `newClient2()`, creates an instance of your concrete class that implements the `it.polito.dp2.BIB.ass3.test2.Client2` interface.

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

Modify the *ant* script `[root]/sol_build.xml` so that it works with the modified service and it includes the compilation of your new client. As usual, you can assume that, when your client is compiled and run, your web service has already been deployed.

Questions

1. Explain your design choices about the modified service. In the explanation, specify how you manage the concurrency problems that could arise when multiple clients are trying the new add operation together with other operations.
2. Explain how uncaught exceptions could cause information leakage and if/how you avoided it in your project.

The answer to these questions must be written in a text file named `[root]/answers.txt`

Correctness verification

In order to pass the exam you have to implement at least points 1. and 2. of the Programming

exercise, and your solution must pass at least the mandatory tests that are included in the archive (the original tests of Assignment 3 plus the tests `testAddAllFewItems`, `testAddAllMoreItems` from the new junit test suite `it.polito.dp2.BIB.ass3.test2.tests.BibTests2`). These tests can be run by the ant script `buildTest2.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 the tests is

```
ant -f buildTest2.xml run-tests
```

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 buildTest2.xml make-zip
```

Important: make sure you are using the `make-zip` target from `buildTest2.xml` (rather than the one from `build.xml`)!

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