Distributed Programming II

A.Y. 2014/15

Final test 1

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 4 into this directory.

The test consists of a programming exercise and a question.

Programming exercise

1. Modify your server so that it also includes the implementation of the operation that lets an operator console client cancel a flight instance (i.e. change its status to CANCELLED). The URL used by the server for receiving flight instance cancellation requests can be selected at your choice. Please, make just sure to use port numbers in the range from 8082 to 8999.

   The service must behave in such a way that the following requirements are satisfied:

   - Flight instance cancellation is possible only for flight instances that are not in the CANCELLED, DEPARTED or ARRIVED states.
   - If a flight instance that is in the BOARDING state is cancelled, no further boarding operation will be possible since cancellation.

   The server must be able to execute operations concurrently and allow at least up to 5 requests to be executed concurrently.

2. Create a client that can interact with your server and that takes the form of a Java library that implements the Java interface *it.polito.dp2.FDS.lab4.FDSFinalTest1Client*, given in source form. The client must satisfy the requirements stated in the comments of the interface. The client class must be named *it.polito.dp2.FDS.sol4.client.FDSFinalTest1ClientImpl*, and must have a default constructor (i.e. a constructor without arguments).

Apart from these new specifications, all the specifications given for Assignment 4 still apply.

In particular, all classes must be developed in the same packages used for the solution of Assignment 4 and stored in the same directories as the solution of Assignment 4. Make sure that the *ant* script `[root]/sol_build.xml` still works for the compilation of your new solution and modify it as necessary (`build-client` must build both the original client of Assignment 4 and the new client of this assignment). You can assume that, when your client is compiled and run, your server has already been deployed (i.e. your `FDSControlServer` application has already been started).

Question

Write the code of the method that performs the flight instance cancellation operation in your service implementation class and then, by referring to this code, explain if and how in your service implementation class you have avoided race conditions on the status of a flight instance, which can be modified both by a thread that executes flight instance cancellation and by a thread that starts boarding.

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

Correctness verification

In order to pass the exam your solution must pass at least the mandatory tests that are included in the archive (the sources are available in the folder `it.polito.dp2.FDS.lab4.tests`). These tests can be run by the ant script `buildTest1.xml` included in the *.zip* file, which also compiles your solution and runs the client and the server by calling your ant scripts. The command for running the tests is

```
ant –f buildTest1.xml run-final-test –Dseed=XXXX
```

There is a total of 6 junit tests. In order to pass the exam it is necessary to pass at least the `testGetStatus` and `testValidCancel` 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 4 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 buildTest1.xml make-final-zip
```

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