Distributed Programming II

A.Y. 2014/15

Assignment n. 3

All the material needed for this assignment is included in the *.zip* archive where you have found this file. Please extract the archive to an empty directory (that will be called *`[root]`*) where you will work.

The assignment consists of developing a client, for a web service named *FDSBookingService*, using JAX-WS "dynamic proxy" programming. The service is meant for booking sellers. It provides read access to the lists of passengers already booked for the various flight instances of the FDS system, and provides the possibility to do other bookings. The service can be started on your own local machine by running the command

```
$ ant run-server
```

Once the service has been started, the WSDL can be obtained as usual at the URL where the service is available, i.e. `http://localhost:8181/FDSBookingService?wsdl`

The server provides the service for a randomly generates set of flight instances. The seed for generation can be specified on the command line in the following form:

```
$ ant -Dseed=XXX run-server
```

The semantics of the operations made available by the service is intuitive. It is explained in the WSDL document.

The client to be developed has to take the form of a class named *FDSBookingClientImpl* that implements the *FDSBookingClient* interface and has the default (i.e. no arguments) constructor. Even if it is not mandatory that this class has a main method, you are suggested to create a main for preliminary testing of your class.

The classes of the solution must be written entirely in package `it.polito.dp2.FDS.sol3`, with all their sources stored in folder `[root]/src/it/polito/dp2/FDS/sol3/`.

The actual URL used by the client class to contact the service must be customizable: the actual URL has to be read as the value of the `it.polito.dp2.FDS.sol3.URL` system property.

The artifacts have to be generated from the WSDL using the ant script provided with this assignment. The generation of artifacts can be started by running the command

```
$ ant compile-wsdl
```

As you can see by inspecting the ant script, the `compile-wsdl` target gets the WSDL from the server, using the same service location specified in the system property mentioned above. For this reason, the server must be running when this target is invoked. If you need custom bindings, you have to write them in the file `[root]/custom/binding.xml`.

The client class must be robust and interoperable, without dependencies on locales.

Correctness verification

Before submitting your solution, you are expected to verify its correctness and adherence to all the specifications given here. In order to be acceptable for examination, your assignment must pass at least all the automatic mandatory tests. Note that these tests check just part of the functional specifications! In particular, they only check that the booking operation works as expected.

Other checks and evaluations on the code will be done at exam time (i.e. passing all tests does not

guarantee the maximum of marks).

All the automatic tests use the random data generator provided with this assignment. The *.zip* file of this assignment includes a set of tests like the ones that will run on the server after submission. Tests can be run by the ant script included in the *.zip* file, which also compiles your solution. Of course, before running the tests you must have started your server. Then, you can run the tests using the `runFuncTest` target.

Note: ensure to run the server before running the tests. This is an example of how to properly run the two commands

```
ant run-server
ant runFuncTest
```

Submission format

A single *.zip* file must be submitted, including all the files that have been produced. The *.zip* file to be submitted can be produced by issuing the following command (from the `[root]` directory)

```
$ jar -cf lab3.zip src/it/polito/dp2/FDS/sol3/**/*.java custom/*
```

Note that the *.zip* file **must not** include the files generated automatically.