CONCORDIA UNIVERSITY

DEPARTMENT OF
COMPUTER SCIENCE AND SOFTWARE ENGINEERING

COMP 6231, Fall 2016 Instructor: R. Jayakumar

ASSIGNMENT 2

Issued: Oct. 17, 2016 Due: Nov. 7, 2016

Note: *The assignments must be done individually and submitted electronically.*

Distributed Flight Reservation System (DFRS) using Java IDL

In this assignment, you are going to implement the Distributed Flight Reservation System (DFRS) from Assignment 1 in CORBA using Java IDL. In addition to the 3 operations in Assignment 1 (namely, *bookFlight*, *getBookedFlightCount*, and *editFlightRecord*) the following operation also needs to be implemented.

- *transferReservation* (*PassengerID*, *CurrentCity*, *OtherCity*)

  This function is used by a manager (*ManagerID*) to transfer a previously made flight reservation i.e. passenger record (*PassengerID*) from the *CurrentCity* to the *OtherCity*. When a manager invokes this method through a *MangerClient* program, the server associated with the *CurrentCity* checks whether the passenger (*PassengerID*) has the specified flight reservation on it and then transfers the reservation with same details to the *OtherCity,* using UDP/IP messages (instead of invoking the *bookFlight* function at the other server). Note that creating the required reservation at the *OtherCity* and cancelling the existing reservation at the *CurrentCity* should be done atomically (that is, both should be done or none should be done) using UDP/IP messages.

In this assignment you are going to develop this modified DFRS application in CORBA using Java IDL. Specifically, do the following:

- Write the Java IDL interface definition for the modified DFRS with all the 4 specified operations.
- Implement the modified DFRS in CORBA. You should design a server that maximizes concurrency. In other words, use proper synchronization that allows multiple officers to perform operations for the same or different records at the same time.
- Test your application by running multiple clients (Passengers and Managers) with the 3 servers. Your test cases should check correct concurrent access of shared data, and the atomicity of *transferReservation* operation.

Your submission will be graded for correct and efficient implementation of the *transferReservation* operation in addition to correct use and implementation of mutual exclusion in accessing shared data and proper exploitation of concurrency to achieve high performance.

Marking Scheme

[30%] *Design Documentation*: Describe the techniques you use and your architecture, including the data structures. Design proper and sufficient test scenarios and explain what you want to test. Describe the most important/difficult part in this assignment. You can use UML and text description, but limit the document to 10 pages. Submit the documentation and code by the due date; print the documentation and bring it to your demo.

[70%] *Demo in the Lab*: You have to register for a 5-minute demo. Please come to the lab session and choose your preferred demo time in advance. You cannot demo without registering, so if you did not register before the demo week, you will lose 40% of the marks. Your demo should focus on the following.

[50%] C*orrectness of code:* Demo your designed test scenarios to illustrate the correctness of your design. If your test scenarios do not cover all possible issues, you'll lose part of mark up to 40%. You will also be evaluated on the implementation of your design.

[20%] *Questions:* You need to answer some simple questions (like what we've discussed during lab tutorials) during the demo. They can be theoretical related directly to your implementation of the assignment.

Questions

If you are having difficulties understanding sections of this assignment, feel free to email the Teaching Assistant Mr. Harpreet Narula at harpreetnarula005@gmail.com. It is strongly recommended that you attend the tutorial sessions which will cover various aspects of the assignment.