COMP6231/1 – Sections BB – Summer 2018

ASSIGNMENT #2
Due: Sunday, June 17th by midnight 11:59 PM

Important Note

- The work is realized in a **team of 4 students**.
- Submit code by the due date.
- Your program should be compiled, executed and return the expected results; otherwise a mark 0 (zero) will be assigned.

Distributed Class Management System (DCMS) using Java IDL

In this assignment, you are going to implement the distributed Class Management system (DCMS) from *Assignment #1* in CORBA using Java IDL. In addition to the 4 operations introduced in *Assignment #1* (namely, *createTRecord*, *createSRecord, getRecordCounts, editRecord*) the following operations also need to be implemented.

- *transferRecord* (*managerID*, *recordID*, *remoteCenterServerName*)

When a center manager invokes this method from his/her center, the server associated with this manager (determined by the *managerID* prefix) searches its hash map to find if the record with *recordID* exists. If it exists, the entire record is transferred to the *remoteCenterServer*. Note that the record should be removed from the hash map of the initial server and should be added to the hash map of the *remoteCenterServer* atomically. The server informs the manager whether the operation was successful or not and both the server and the manager store this information in their logs.

In addition to this operation, the previous operations (*createTRecord*, *createSRecord*, *getRecordCounts*, and *editRecord*) shall now have one additional parameter: *managerID*. This is used by the *CenterServers* to identify which manager conducted an invocation.

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

- Write the Java IDL interface definition for the modified DCMS with all the 5 specified operations.
- Implement the modified DCMS. You should design a server that maximizes concurrency. In other words, use proper synchronization that allows multiple managers to correctly perform operations on the same or different records at the same time.
- Test your application by running multiple managers with the 3 servers. Your test cases should check correct concurrent access of shared data, and the atomicity of *transferRecord* operation (e.g. what if a record being edited needs to be transferred and both operations were initiated at the same time?).

Your submission will be graded for correct and efficient implementation of the ***transferRecord*** 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

[**40%**] ***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.

[**60%**] ***The correctness of code:***

Your designed test scenarios to illustrate the correctness of your design. If your test scenarios do not cover all possible issues, you will lose part of marks up to 50%.

EDUCATIONAL GUIDELINES

- The work is realized in a team of 4 students.
- The delivery must be made no later than **Sunday, June 17th by midnight 11:59 PM** using the Website submission as mentioned in the course outline.
- If you are having difficulties understanding sections of this assignment, feel free to email the Teaching Assistants. It is strongly recommended that you attend the Lab sessions which will cover various aspects of the assignment.

NOTE

- CORBA is an old plugin and cannot be installed in new versions of Eclipse. The latest version in which the CORBA plugin can be installed is Eclipse Ganymede (3.4.2) that was released in 2009.