A picture containing graphical user interface

Description automatically generated

COMP 6231 DISTRIBUTED SYSTEMS DESIGN

SUMMER 2021

DESIGN DOCUMENT FOR ASSIGNMENT 2

TEAM SMARTY PANTS

BINIT KUMAR - 40172005

ABDUL AZIZ RAZA SHAIK - 40182726

CONCORDIA UNIVERSITY

MONTREAL, QUEBEC, CANADA

JULY 2021

Introduction

**Distributed Class Management System (DCMS) using CORBA (Common Object Request Broker Architecture):**

Distributed Class Management System is a distributed system used by center managers to manage information regarding the teachers and students across different centers.

The three centers locations are:

* Montreal (MTL)
* Laval (LVL)
* Dollard-des-Ormeaux (DDO)

The server for each center (called CenterServer) maintains a number of Records. The two types of Records are:

* TeacherRecord
* StudentRecord

A Record a unique RecordID starting with TR - TeacherRecord or SR – StudentRecord.

Fields in TeacherRecord – First Name, Last Name, Address, Phone, Specialization and Location.

Fields in StudentRecord – First Name, Last Name, Courses Registered, Status and Status Date.

Managers with a unique ManagerID can perform operations:

* createTRecord
* createSRecord
* getRecordCounts
* editRecord
* **transferRecord** (new feature)

This application has a number of CenterServers(one per center) each implementing the above operations for that center and ManagerClients(one per center) invoking the manager’s operations at the associated CenterServeras necessary.

It has been developed using JAVA IDL - CORBA

Description of the technique used:

Common Object Request Broker Architecture (CORBA):

DCMS has been designed using CORBA which is the object oriented equivalent of remote procedure calls (RPC). ORB (Object Request Broker) is responsible for management of the remote access to objects. Common ORB architecture is a software bus for distributed objects. CORBA provides a nearly transparent access to remote objects from a local program using the client-server paradigm. Unlike RMI, CORBA has an advantage of being totally platform and language independent.

How does CORBA implementation look like:

A picture containing text, indoor, silver

Description automatically generated

Software-based communications interface

Diagram

Description automatically generated

As seen in the figure:

Client Program: ORB Library on the client side determines the location of the target object and sends a request for its access.

Server Program: ORB Library on the server side receives request, processes it and sends a reply to the client object.

Architecture of the system:

The DCMS architecture has 3 locations (Montreal, Laval and DDO) which are the clients of the system.

The Managers at these locations have access to the database which consists of the records: Student Record and Teacher Record. The managers can perform functions such as createTRecord, createSRecord, getRecordCounts, editRecord and transferRecord as per requirement.

ManagerClient is the client program which handles these requests and attempts to create, edit or transfer the appropriate record with the corresponding server associated with the manager in the form of a hashmap and each server also maintains a log containing the history of all the operations that have been performed on that server in a separate text file. By invoking getRecordCounts, the manager of a center can also get the number of record (both teacher and student) present in their respective database.

This system has multiple centers and one CenterServer for each of them which make use of a centralized ManagerClient to invoke the required method from the repository, for each operation it finds the information about the requesting server and invokes the corresponding operation.

Diagram

Description automatically generated

TEST CASES EXECUTED:

Test Setup:

**Machine-1 Configuration**

From an MS-DOS system prompt (Windows), enter:

*start orbd -ORBInitialPort 7719*

note: 7719 is the orb port of DDO server

Execute RunDDO.java to run DDO server

**Machine-2 Configuration**

From an MS-DOS system prompt (Windows), enter:

*start orbd -ORBInitialPort 7717*

note: 7717 is the orb port of MTL server

Execute RunMTL.java to run MTL server

**Machine-3 Configuration**

From an MS-DOS system prompt (Windows), enter:

*start orbd -ORBInitialPort 7718*

note: 7718 is the orb port of LVL server

Execute RunLVL.java to run LVL server

**In any of the machine above:**

Execute RunClient.java to run manager’s client

# Test Scenarios:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Expectation | Result |
| 1 | Manager creating Teacher records | Record should be saved in database | PASS |
| 2 | Manager creating Student records | Record should be saved in database | PASS |
| 3 | Managers requiring valid id to access the server Test Data: (MTLXXXX / LVLXXXX / DDOXXX) | Managers should be able to connect to the right server | PASS. |
| 4 | Managers not able to connect with invalid id | Invalid managers id will get an error message | PASS |
| 5 | Managers able to get all the record counts | The total count of all the records should be printed for each of the servers | PASS |
| 6 | Manager should be able to edit records | Manager should be able to modify records from the record id | PASS |
| 7 | Phone number validation in Teacher Records | Only numeric values in Phone number | PASS |
| 8 | Status Field validation in student record | Only Active and IActive status | PASS |
| 9 | Managers should not be allowed to edit any fields other than Address, Phone and Location in the Teacher Record. | Any fields other than Address, Phone and Location in the Teacher Record are not modified | PASS. |
| 10 | Managers should not be allowed to edit any fields other than Course Registered, Status and Status Date in the Student Record. | Any fields other than Course Registered, Status and Status Date in the Student Record are not modified | PASS |
| 11 | Transfer of record from one server to another | When transferring record using record id and server name, the new record is transferred to the new server | PASS |
| 12 | Deletion of record from current server after transferring the record to another server | After transferring record to another server, the record in the current server is being deleted successfully to avoid any duplicate data. | PASS |

CHALLENGES FACED:

* The most difficult and important part of the design and execution was creating the correct client-server architecture which would work exactly as given in the problem statement.
* Another challenge faced by us was to understand the specifics and limitations of the CORBA system and use it to implement our proposed architecture without any shortcomings in the required features.
* Third difficulty we faced was to maintain concurrency in the system.
* Getting the whole distributed system to run on a single system using different ports and sockets for the development purpose was also a tough task.

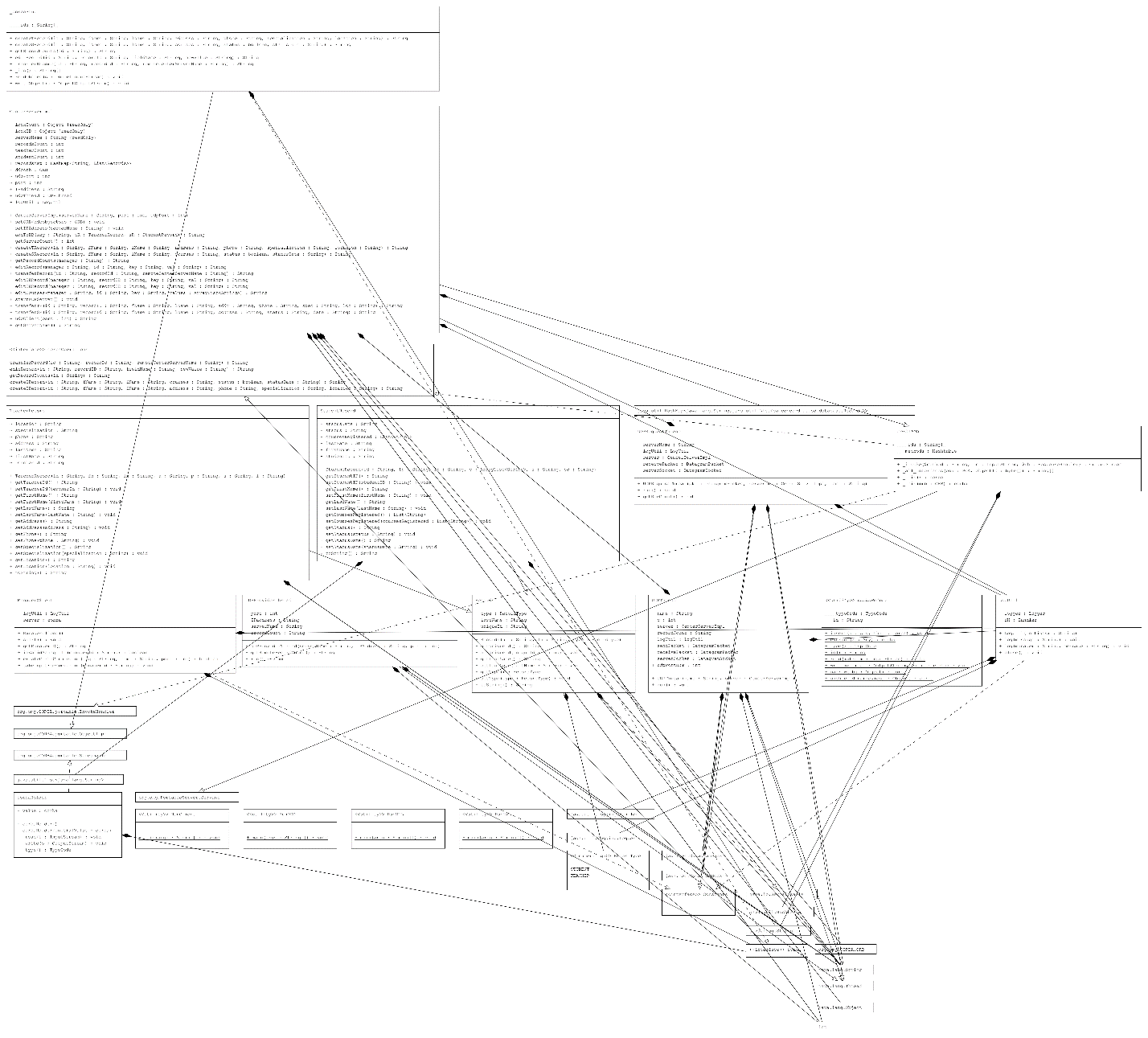
UML Diagrams:

Use case diagram:

Diagram

Description automatically generated

Class diagram





New Changes

Corba.idl

Text

Description automatically generated with medium confidence

* Every method includes manager id
* Corba.idl includes existing method from assignment1 as well as new method from assignment 2
* The generated package arch is under ca.concordia.dsd.arch package in the source code

CenterServerImpl.java



* CenterServerImpl class extends the automatically generated corbaPOA class



* The setORB function will be invovked when servant is initiated.

Text

Description automatically generated

* The class RunDDO.java RunMTL.java and RunLVL.java has only the main function, that start with initializing the ORB object by assigning the host and port programmatically.
* This class connects to ORB core, get the references to the RootPOA, initiate the servant and register it with the ORB, get the object reference to the servant and bind it to a naming context.
* At the end it also starts the internal UDP server event loop that can transfer the record when client trigger transfer record command to another server via UDP.

References:

* George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair, "*Distributed Systems Concepts and Design"*, Fifth Edition, Addison-Wesley, 2012, ISBN: 0-13-214301-1.
* M. L. Liu, "*Distributed Computing: Principles and Applications*", Pearson Addison-Wesley 2004, ISBN 0-201-79644-9.
* https://www.javatpoint.com/q/4780/corba
* https://en.wikipedia.org/wiki/Common\_Object\_Request\_Broker\_Architecture
* https://docs.oracle.com/javase/8/docs/technotes/guides/idl/jidlExample.html https://www.codejava.net/java-se/networking/java-udp-client-server-program-example