Distributed System Design

COMP 6231

Instructor: R. Jayakumar

Assignment 2

Distributed Appointment Management System using Java IDL (CORBA)

BY: Raviraj Savaliya (40200503)

**Introduction:**

Distributed appointment management System (DAMS) for health care is a distributed system that has 3 hospitals namely Montreal (MTL), Quebec (QUE) and Sherbrooke (SHE). DAMS is managed and used by Patients and Admins. In this system Patient can perform few operations as listed below.

1. Book Appointment
2. Get Booked Appointment
3. Cancel Booked Appointment
4. List of all available Appointment
5. Swap Appointment

Whereas Admin can perform all patient’s operation and some additional operation such as:

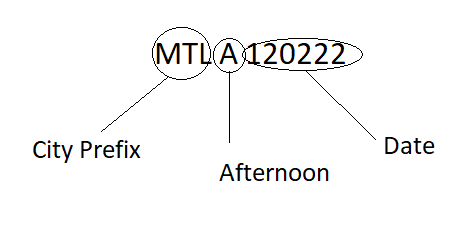
1. Add new Appointments
2. Remove Appointments

In DAMS, 3 servers are used for manage all clients requests. These servers name is:

1. Montreal
2. Quebec
3. Sherbrooke

Both patient and admins are identified by a unique adminID and patientID respectively, which is built from the acronym of their hospital’s city and a 4 digit number (e.g. MTLA1111 for admin and MTLP1111 is for patient). We can identify the user by the 4th character of ID. If it is A then admin and if P then patient. DAMS system also maintains logs for server and client.

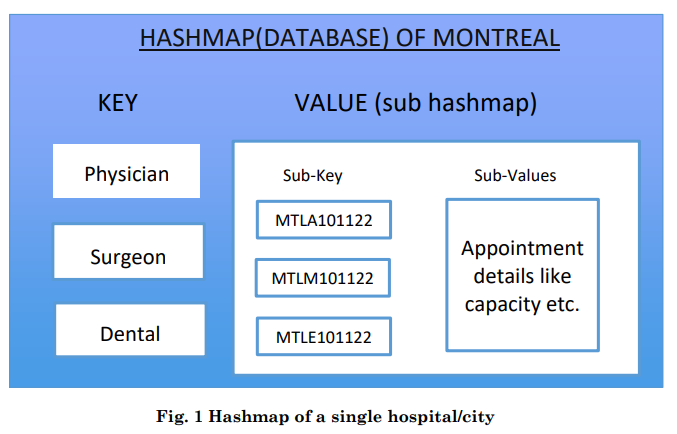
There are 3 types of admins for different server. They creates appointment slots of 3 different types such as Physician, Dental and Surgeon. There is three different time slot Morning(M),Afternoon(A) and Evening(E). Appointment ID is combination of city, time slot and appointment date (e.g. MTLM120222, QUEE120222).



**Data Management:**

The appointments data is stored in HashMap. Appointment type is key and Appointment Id

Is sub key and sub value is capacity of appointment.



**System overview:**

DAMS is designed in CORBA. CORBA Architecture and UDP are the major techniques. DAMS contains client which is connected to the to region wise servers. for example, If user is from Montreal then automatically Montreal server will be selected. Here, all servers are internally connected using the UDP. If Montreal’s user wants to book appointment of Quebec Hospitals, then Montreal server forward operation request to Quebec server. Quebec server perform that operation and send back the result to Montreal server.

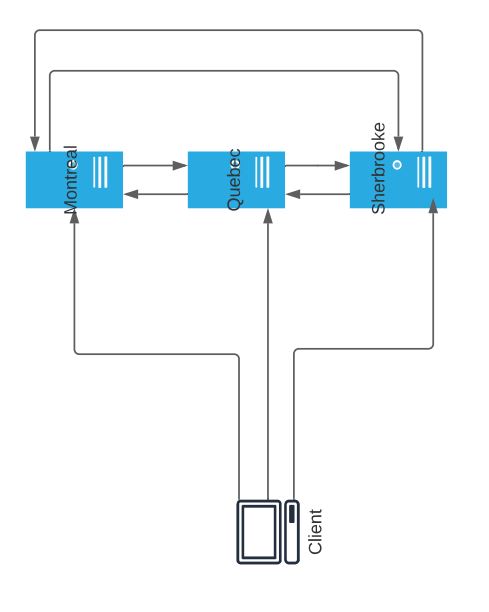


Fig.2 Server-client connection

As shown above, Server handles all request coming from the client and return a response. All server has their own HashMap to handle the data.

**Design Architecture:**

**ServerObjectInterface.idl**

This file contains all the operation that can be used by the client.

* Add appointment
* remove appointment
* list appointment
* book appointment
* get appointment schedule
* cancel appointment
* swap appointment

Compile file ServerObjectInterface.idl with IDL compiler. This will generate 6 files as shown in package ServerObjectInterfacezApp.

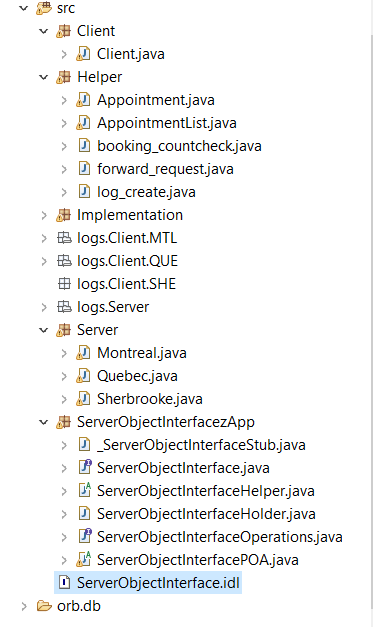


Fig.3 System Architecture

**Server:**

This package contains 3 Server file as shown in Fig.3. These files are responsible for start the server and running.

Each server has some common steps:

• Creates and initializes an ORB instance

• gets a reference to the root POA and activates the POAManager.

• Create servant and register it with ORB.

• Gets a CORBA object reference for a naming context in which to register the new CORBA object

• Gets the root naming context

• Bind the new object in the naming context

• Waits for invocations

**Implimentation:**

This package contains implimentation of all function that are mentioned in IDL file.

**Client:**

This is use for start the application. This file take the appropriate inputs from the either patient or admin and make request to server for invoking methods.

**Logs:**

This folder contains the all logs of server and client to track the application in case of failure. Client contains server wise directory and inside that there is log file by name of client. And server has 3 text file for each server.

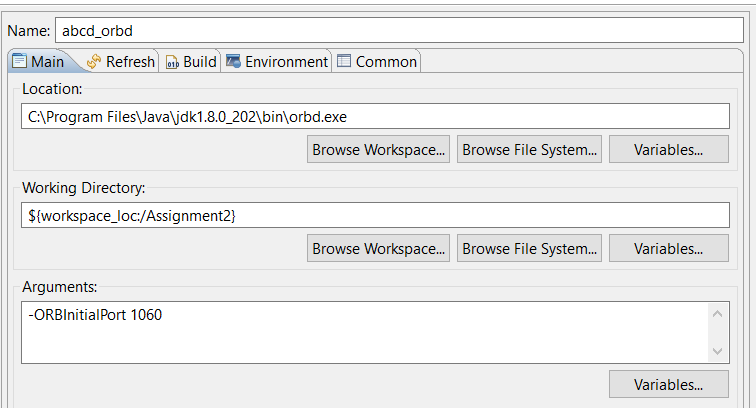
**Helper:**

Helper Package contains Common files that are used in implementation. It contains file to create log, check appointments count , forward request to another server and getter setter methods.

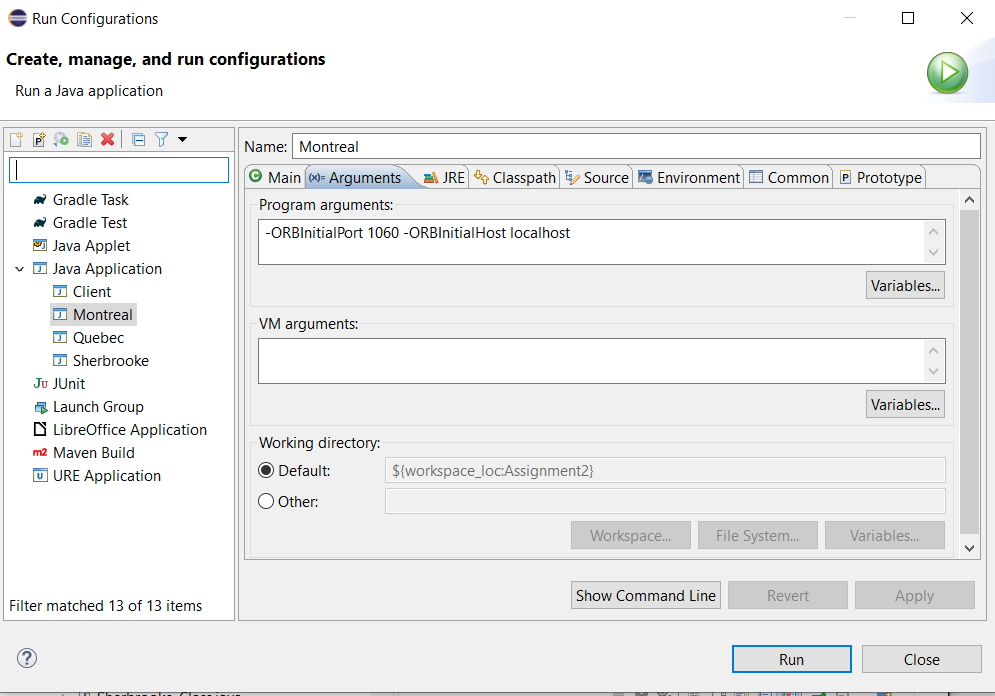
**Let’s Start Application**

Eclipse 2020-06 and JDK 1.8 is used for development of this application.

1. start orbd



1. Start All 3 server



1. Start Client same as Server

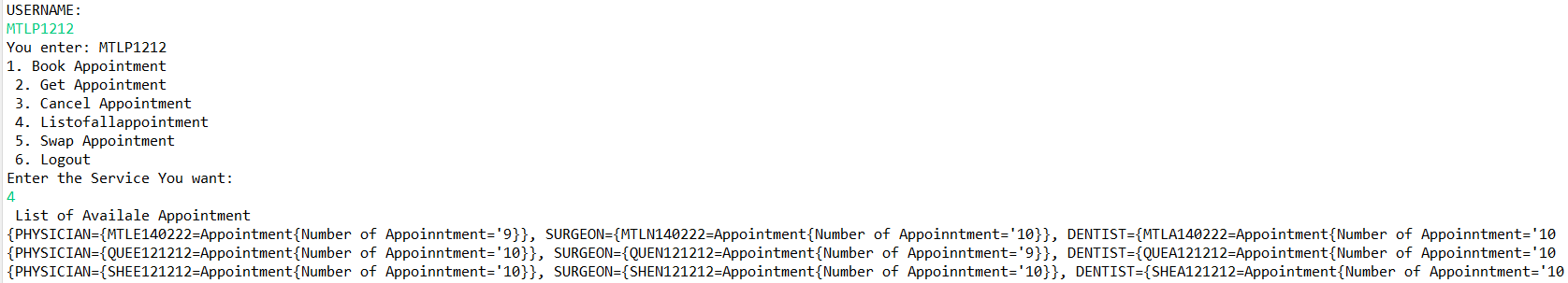
You are ready for the Perform Operations.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Num | USER | OPERATION | TEST CASES | VERIFY |
| 1 | Admin/Patient | Login | 1. Validate with valid UserID  2. UserID wise menus  3. Auto server selection from UserID | WORKING  WORKING  WORKING |
| 2 | Admin | Add Items | 1.Verify Appointment type  2.Verify Appointment id  3. Verify Appointment capacity  4. It will add a new item or change the capacity of current appointment | WORKING  WORKING  WORKING  WORKING |
| 3 | Admin | Remove items | 1.Verify Appointment Id  2.Verfiy Appointment type  3.It will remove all appointments capacity | WORKING  WORKING  WORKING |
| 4 | Admin/Patient | List of Appointments | 1.It will display the all appointments from all servers. | WORKING |
| 5 | Patient/Admin | Book Appointment | 1. Verify AppointmentID  2. verify Appointment Type  3. Book Appointment using inter server communication.  4.If appointment already book will display try again.  5.Verify user can book one appointment once only  6. Change the available appointment capacity if booked appointment successfully. | WORKING  WORKING  WORKING  WORKING  WORKING  WORKING |
| 6 | Patient/Admin | Get Appointment | 1.verify user has booked appointment  2.if not it will shows null  3.If appointment is booked then display list of appointments booked  4.Inter server communication | WORKING  WORKING  WORKING  WORKING |
| 7 | Patient/Admin | Cancel Appointment | 1.check if appointment is booked by user or not  2.if booked, verify appointment id and appointment type then cancel the appointment  3.Inter server communication | WORKING  WORKING  WORKING |
| 8 | Patient/Admin | Swap Appointment | 1.Verify the Input data  2.Swap appointment to same Region  3.Swap appointment to Different Region  4.Change the available appointment capacity if booked appointment successfully  5.Creating log file for each operation | WORKING  WORKING  WORKING  WORKING  WORKING |

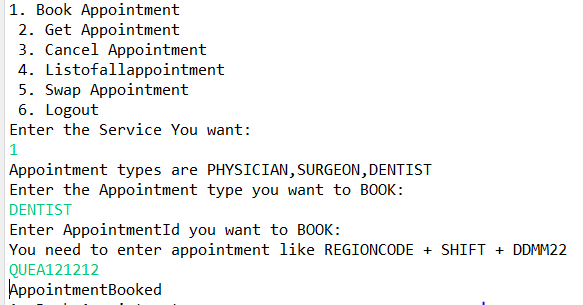
**Test Cases:**

**Screenshot:**

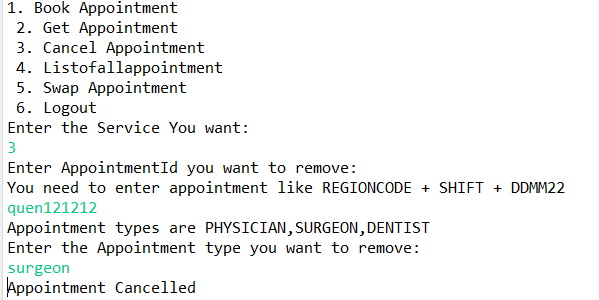
1> List of Available appointment



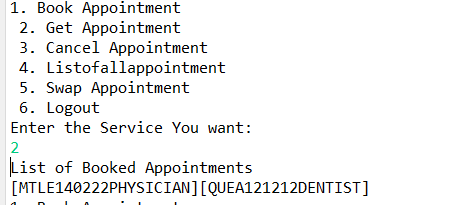
2>Appointment Booked



3> Cancel Appointment

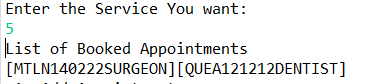


4>List of booked Appointment

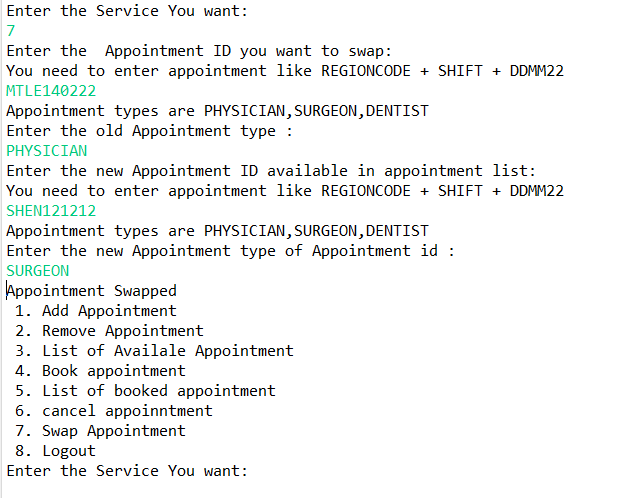


5>Appointment Swap





6> Swap appointment to another regions Appointment



7>Server Log

