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:

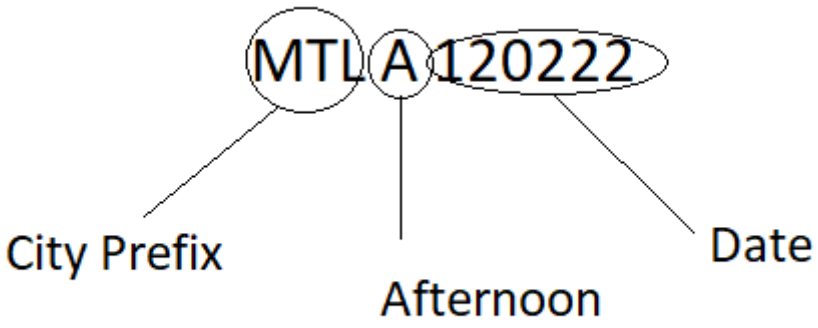
1. Add new Appointments
2. Remove Appointments

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.

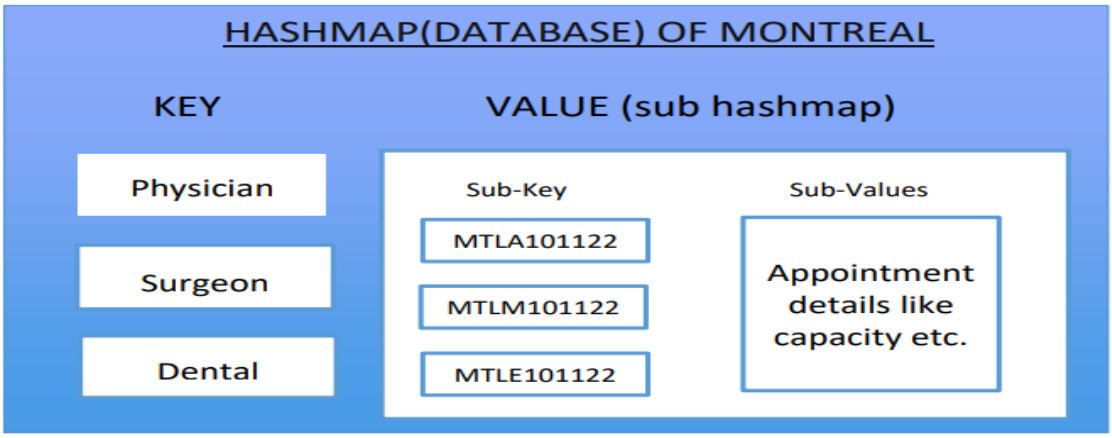


Fig. 1 Hashmap of a single hospital/city

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.

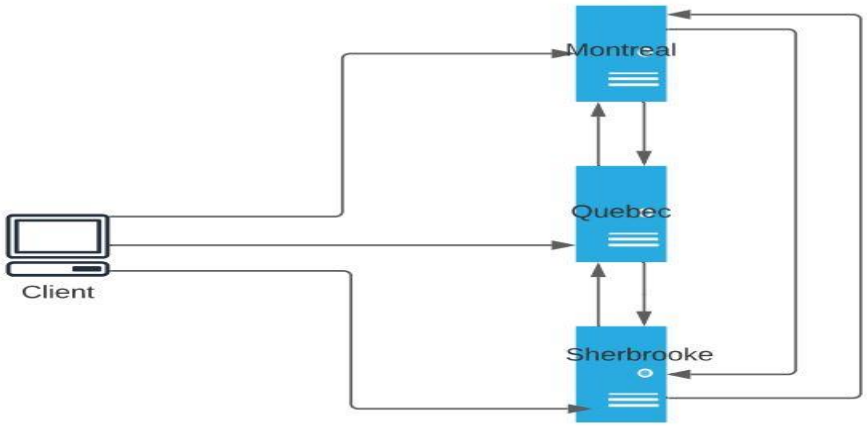


Fig.2 Server-client connection

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.

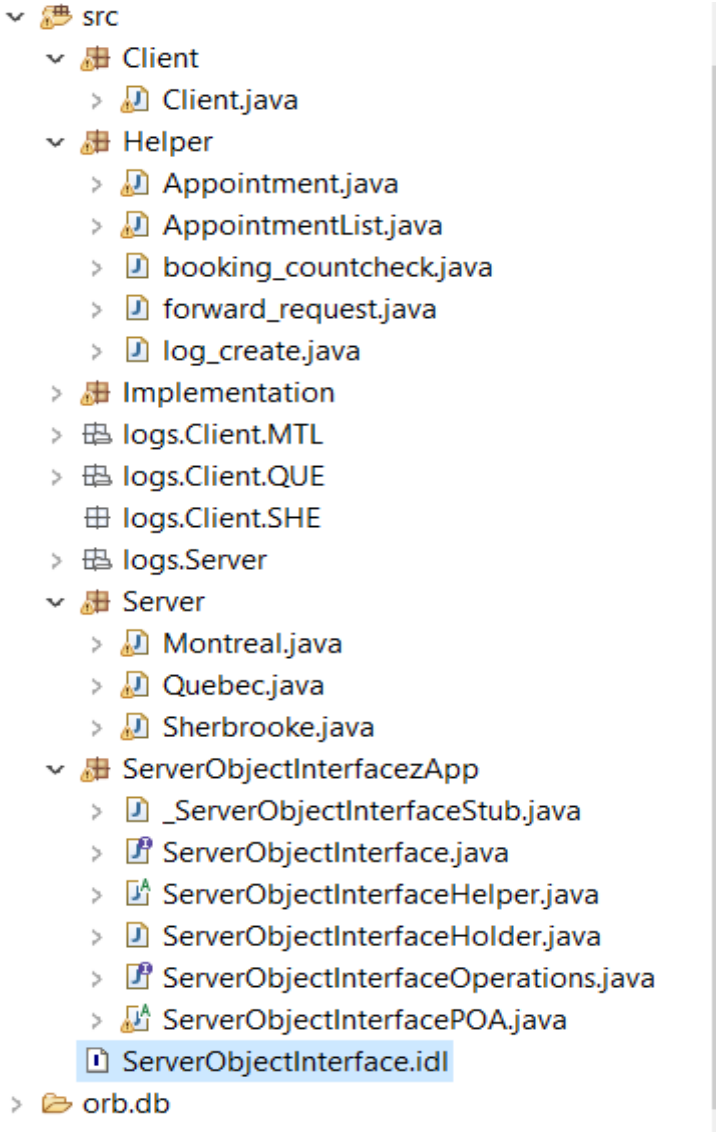


Fig.3 System Architecture

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.

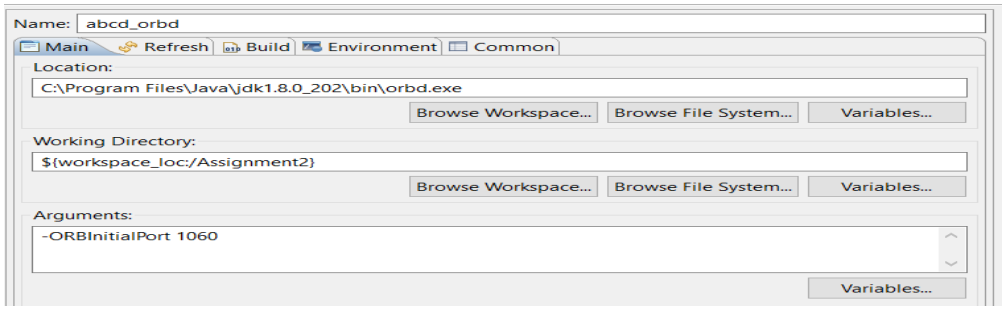
Helper:

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.

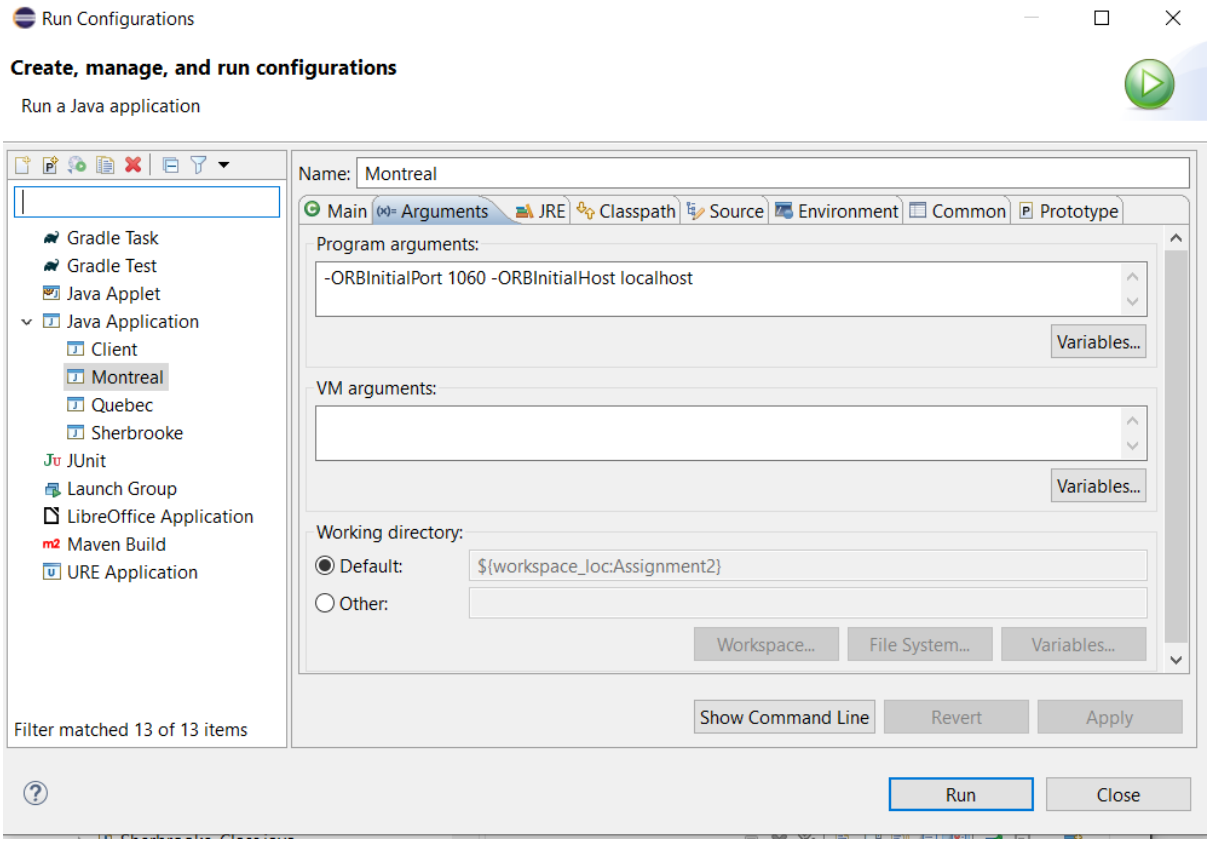
Let's Start Application

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

1. start orbd



2. Start All 3 server



3. Start Client same as Server

You are ready for the Perform Operations.

Test Cases:

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

Screenshot:

1> List of Available appointment

```
USERNAME:
MTLP1212
You enter: MTLP1212
1. Book Appointment
 2. Get Appointment
 3. Cancel Appointment
 4. Listofallappointment
 5. Swap Appointment
 6. Logout
Enter the Service You want:
4
 List of Availale Appointment
{PHYSICIAN={MTLE140222=Appointment{Number of Appoinntment='9}}, SURGEON={MTLN140222=Appointment{Number of Appoinntment='10}}, DENTIST={MTLA140222=Appointment{Number of Appoinntment='10
{PHYSICIAN={QUEE121212=Appointment{Number of Appoinntment='10}}, SURGEON={QUEN121212=Appointment{Number of Appoinntment='9}}, DENTIST={QUEA121212=Appointment{Number of Appoinntment='10
{PHYSICIAN={SHEE121212=Appointment{Number of Appoinntment='10}}, SURGEON={SHEN121212=Appointment{Number of Appoinntment='10}}, DENTIST={SHEA121212=Appointment{Number of Appoinntment='10
```

2>Appointment Booked

```
1. Book Appointment
 2. Get Appointment
 3. Cancel Appointment
 4. Listofallappointment
 5. Swap Appointment
 6. Logout
Enter the Service You want:
1
Appointment types are PHYSICIAN,SURGEON,DENTIST
Enter the Appointment type you want to BOOK:
DENTIST
Enter AppointmentId you want to BOOK:
You need to enter appointment like REGIONCODE + SHIFT + DDMM22
QUEA121212
AppointmentBooked
```

3> Cancel Appointment

```
1. Book Appointment
 2. Get Appointment
 3. Cancel Appointment
 4. Listofallappointment
 5. Swap Appointment
 6. Logout
Enter the Service You want:
3
Enter AppointmentId you want to remove:
You need to enter appointment like REGIONCODE + SHIFT + DDMM22
quen121212
Appointment types are PHYSICIAN,SURGEON,DENTIST
Enter the Appointment type you want to remove:
surgeon
Appointment Cancelled
```

4>List of booked Appointment

```
1. Book Appointment
 2. Get Appointment
 3. Cancel Appointment
 4. Listofallappointment
 5. Swap Appointment
 6. Logout
Enter the Service You want:
2
List of Booked Appointments
[MTLE140222PHYSICIAN][QUEA121212DENTIST]
```

5>Appointment Swap

```
List of Booked Appointments
[MTLE140222PHYSICIAN][QUEA121212DENTIST]
1. Book Appointment
 2. Get Appointment
 3. Cancel Appointment
 4. Listofallappointment
 5. Swap Appointment
 6. Logout
Enter the Service You want:
5
Enter the  Appointment ID you want to swap:
You need to enter appointment like REGIONCODE + SHIFT + DDMM22
MTLE140222
Appointment types are PHYSICIAN,SURGEON,DENTIST
Enter the old Appointment type :
PHYSICIAN
Enter the new Appointment ID available in appointment list:
You need to enter appointment like REGIONCODE + SHIFT + DDMM22
MTLN140222
Appointment types are PHYSICIAN,SURGEON,DENTIST
Enter the new Appointment type of Appointment id :
SURGEON
Appointment Swapped
```

```
Enter the Service You want:
5
List of Booked Appointments
[MTLN140222SURGEON][QUEA121212DENTIST]
```

6> Swap appointment to another regions Appointment

```
Enter the Service You want:
7
Enter the  Appointment ID you want to swap:
You need to enter appointment like REGIONCODE + SHIFT + DDMM22
MTLE140222
Appointment types are PHYSICIAN,SURGEON,DENTIST
Enter the old Appointment type :
PHYSICIAN
Enter the new Appointment ID available in appointment list:
You need to enter appointment like REGIONCODE + SHIFT + DDMM22
SHEN121212
Appointment types are PHYSICIAN,SURGEON,DENTIST
Enter the new Appointment type of Appointment id :
SURGEON
Appointment Swapped
 1. Add Appointment
 2. Remove Appointment
 3. List of Available Appointment
 4. Book appointment
 5. List of booked appointment
 6. cancel appoinntment
 7. Swap Appointment
 8. Logout
Enter the Service You want:
```

7>Server Log

```
DATE: 2022-02-23 09:07:33 PM|Action: getAppointmentSchedule | Parameters: USER:allservernull | Action Status: Success | Resonse: getAppointmentSchedule done
DATE: 2022-02-23 09:07:33 PM|Action: List of Appointments | Parameters: USER: MONserverMTLA1212 | Action Status: Success | Resonse: QUEServeraccessed
DATE: 2022-02-23 09:07:33 PM|Action: Appointment BookedQUEE121212 | Parameters: USER: QUEserverMTLA1212/ AppointmentId: QUEE121212/ AppointmentType: PHYSICIAN
| Action Status: Success | Resonse: TRUE
DATE: 2022-02-23 09:07:33 PM|Action: Appointment BookedQUEE121212 | Parameters: USER:queserverMTLA1212/ AppointmentId: QUEE121212/ AppointmentType: PHYSICIAN/
NumofAppointment:  | Action Status: Success | Resonse: Appointment added
DATE: 2022-02-23 09:08:10 PM|Action: Sending request to que server for swapappointmentQUEE121212PHYSICIANtoSHEN121212PHYSICIAN | Parameters: USER:
MONservertoQUEserverMTLA1212/ oldAppointmentId: QUEE121212/ oldAppointmentType: PHYSICIAN/ NewAppointmentId: SHEN121212/ newAppointmentType: SURGEON | Action
Status: Success | Resonse: true
```