Comp 6231: Distributed System Design
Winter 2022

Assignment 3: Distributed Appointment Management System (DAMS) Using WebServices

Course Instructor: Prof. R. Jayakumar
Lab Instructor: Mr. Brijesh Lakkad

Student: Mr. Munj Bhavesh Nayak
Student ID: 40195590

Table of Contents

Overview

The Distributed Appointment Management System (DAMS) consists of 3 different hospitals who have their own individual server, totalling 3 servers:

- Montreal Server (MTL)
- Sherbrooke Server (SHE)
- Quebec Server (QUE)

This system consists of 2 types of clients:

- Admins: who carry out the main management of the entire system.
- Patients: the end users of the system.

In this assignment, it has to be ensured that these clients are connected to their own servers with the JAX WS , and also that the connection between the 3 servers happens using the UDP/IP Socket Programming.

Admin Specific Functions:

- addAppointment(): Admins can add 3 different types of appointments (Physician, Dental, and Surgeon) in their own servers.
- removeAppointment(): Admins can remove any previously added Appointments from their respective servers. If any patient had booked an appointment before the admin removed it, the patient must be booked for the closest next available appointment.
- listAppointmentAvailability(): Display all the available appointments of the selected type from all the servers.

Admin/Patient Functions:

- bookAppointment(): Patients can book an appointment from the 3 appointment types and select the date that the admin has added. They can also book an appointment in servers other than their own but it has a weekly limit of 3 appointments.
- getAppointmentSchedule(): show the appointment schedule of the patient.
- cancelAppointment(): patients can cancel their appointments.
- swapAppointment(): patients can swap a booked appointment with another existing appointment (a bookAppointment + cancelAppointment). It needs to be atomic.

The patients are recognized with their patientID consisting of a serverID, clientType (Admin or Patient), and a 4 digit unique identifier key.

Appointments are recognized with their appointmentType: Physician/Surgeon/Dental and appointmentID (serverID + appointmentSlot (M/A/E) + appointmentDate)

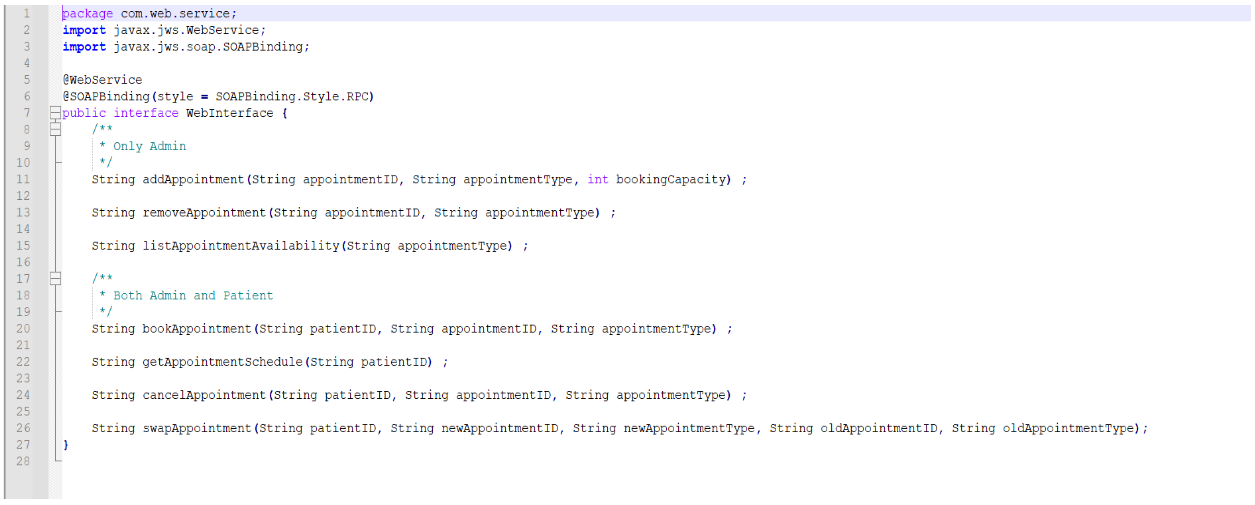
Bothe servers and clients maintain individual log files.

Implementation

- Client-Server communication is done by SOAP based Webservice
    - Montreal Service Address: http://localhost: http://localhost:8080/montreal?wsdl
    - Quebec Service Address: http://localhost: http://localhost:8080/quebec?wsdl
    - Sherbrooke Service Address: http://localhost: http://localhost:8080/sherbrook?wsdl
    - @WebService(endpointInterface = "com.web.service.WebInterface")
- Server-Server communication is done using the UDP/IP Socket Programming
    - Montreal UDP Port: 8888
    - Quebec UDP Port: 7777
    - Sherbrooke UDP Port: 6666
- To reduce the duplication of the code and ease the development, only one server Implementation is used with 3 instances, as discussed/advised in the lab.
- Clients and Servers maintain separate individual log files.
- Server Files are located at src/Logs/Server/ServerName.txt
- Client Files are located at src/Logs/Client/ClientID.txt
- To ensure maximum concurrency, concurrentHashMaps were used to store the data.
- The most crucial thing in the implementation was to avoid a deadlock/infinite looping in the UDP calls.
- The most challenging part to implement was the AppointmentManager since it is the heart of the entire system dealing with all the important operations.

- Synchronized blocks and methods were used in some cases to ensure a Thread-Safe operation.
- For the atomicity of the swapAppointment method, the newAppointment was booked first (similar to reserving a spot) and then if it was successful, the oldAppointment was canceled. If the oldAppointment couldn't be canceled, the new appointment(reservation) was canceled.
- A shutdown() method was added for shutting down the ORB.

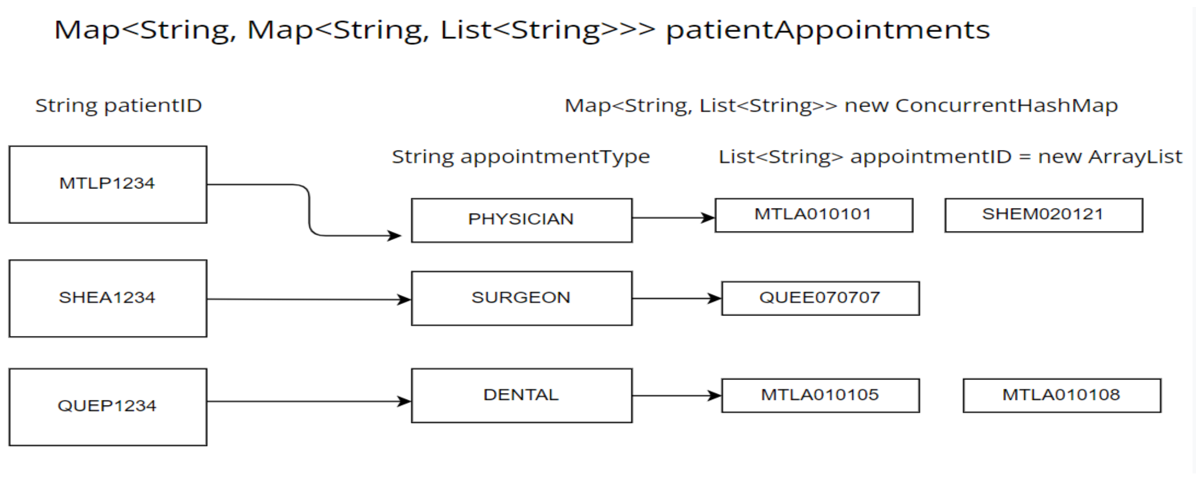
WebInterface:

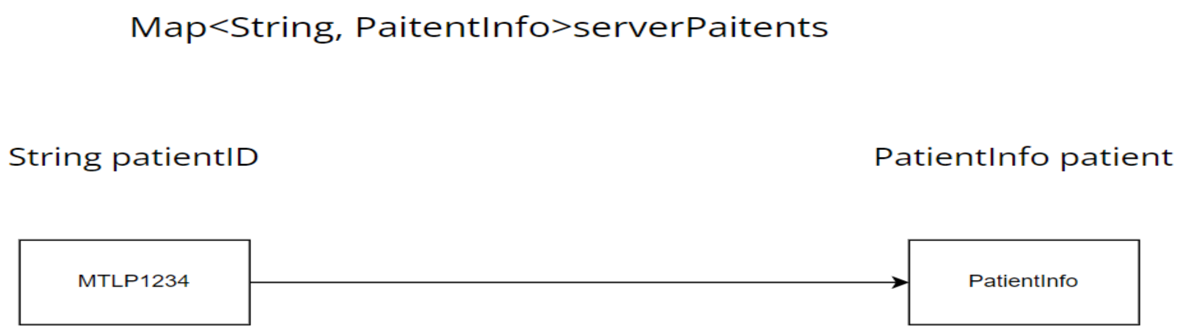


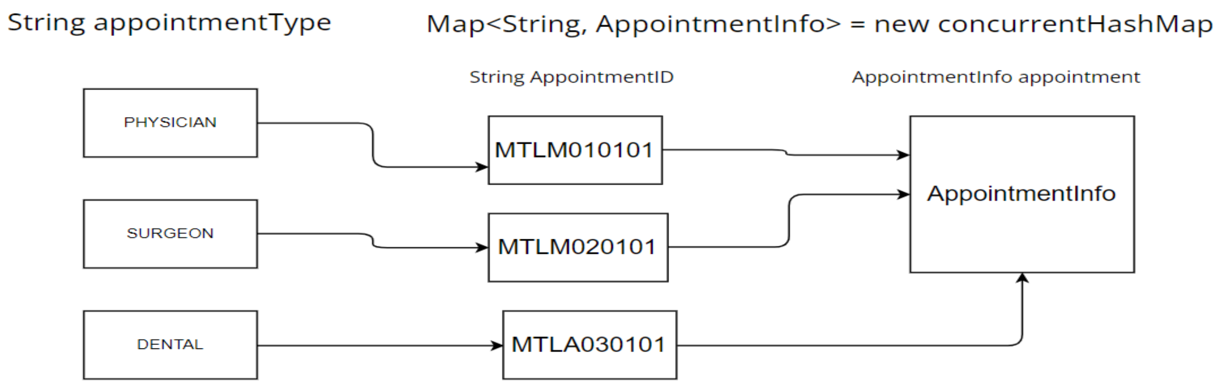
```
package com.web.service;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;

@WebService
@SOAPBinding(style = SOAPBinding.Style.RPC)
public interface WebInterface {
    /**
     * Only Admin
     */
    String addAppointment(String appointmentID, String appointmentType, int bookingCapacity) ;

    String removeAppointment(String appointmentID, String appointmentType) ;

    String listAppointmentAvailability(String appointmentType) ;

    /**
     * Both Admin and Patient
     */
    String bookAppointment(String patientID, String appointmentID, String appointmentType) ;

    String getAppointmentSchedule(String patientID) ;

    String cancelAppointment(String patientID, String appointmentID, String appointmentType) ;

    String swapAppointment(String patientID, String newAppointmentID, String newAppointmentType, String oldAppointmentID, String oldAppointmentType);
}
```

Data Structures

Following are the DataFlow Diagrams of the Data Structures involved:


Map<String, Map<String, List<String>>> patientAppointments
String patientID
Map<String, List<String>> new ConcurrentHashMap
String appointmentType
List<String> appointmentID = new ArrayList
MTLP1234
SHEA1234
QUEP1234
PHYSICIAN
SURGEON
DENTAL
MTLA010101
SHEM020121
QUEE070707
MTLA010105
MTLA010108





Testing Scenarios

1. Type of Test: Login

   Scenario: UserID

   Cases: AdminID, PatientID
2. Type of Test: Menu

   Scenario: UserID, Logout

   Cases: AdminID, PatientID, Logout
3. Type of Test: Add Appointment

   Scenario: Admin adds an Appointment

   Cases:
   - Invalid AppointmentID: not Added
   - New AppointmentID: Added
   - Existing AppointmentID: Capacity Updated
   - Duplicate Appointment: Not Allowed
   - AppointmentID of other Servers: Not Allowed
4. Type of Test: Remove Appointment

   Scenario: Admin Removes an Appointment

   Cases:
   - Invalid AppointmentID
   - AppointmentID Doesn't Exist
   - AppointmentID with no registrations: Appointment Removed
   - AppointmentID with a registration: get the patient the earliest next available appointment.
   - AppointmentID of other Server: not allowed

5. Type of Test: List Appointment Availability

   Scenario: Listing all available appointments.

   Cases:

   - List all the appointments available from all the servers.
   - Enforce an appointment Type so only that type of appointments are listed.

6. Type of Test: Book Appointment

   Scenario: Patient tries to book an appointment

   Cases:

   - On Own Server: Allowed
   - If capacity is full: not allowed
   - On other servers: weekly limit of 3 appointments.
   - Invalid AppointmentID: not allowed
   - Invalid PatientID: not allowed

7. Type of Test: Get Appointment Schedule

   Scenario: Patient tries to get a schedule of their appointments.

   Cases:

   - Show the appointment Schedule of a patient
   - Invalid patientID: not allowed
   - No bookings: empty return
   - Patient doesn't exist: create new patient

8. Type of Test: Cancel Appointment

   Scenario: Patient tries to cancel an appointment.

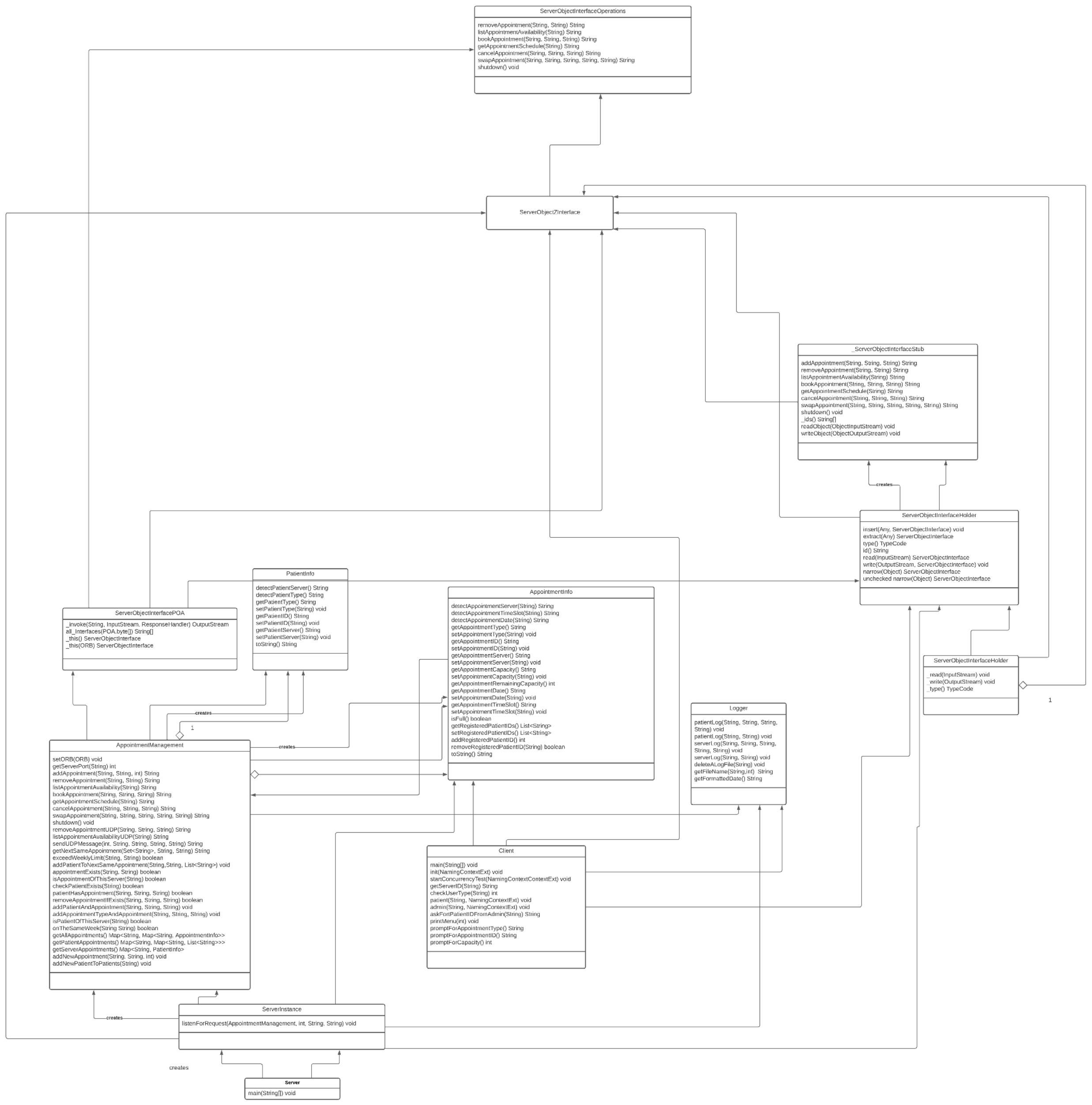
   Cases:

   - Cancel at own server: OK
   - Cancel at other Server: OK
   - Cancel a non booked appointment: Not allowed
   - Invalid AppointmentID/PatientID: Not allowed

9. Type of Test: SwapAppointment
   Scenario: Patient tries to swap an already booked appointment.
   Cases:
   - New appointment has no capacity - Status: false
   - Old appointment doesn't exists, and given new AppointmentID exists - Status: false
   - Old appointment exists, and given new AppointmentID doesn't exists - Status: false
   - Old appointmentID city equals patient's city and new appointmentID city equals patient's city happening in same week - Status: true
   - Old appointmentID city equals to patient's city and new appointmentID city equals to patient's city and not happening in same week - Status: true
   - Old appointmentID city not equals to patient's city and new appointmentID city equals to patient's city and happening in same week - Status: true
   - Old appointmentID city not equals to patient's city and new appointmentID city equals to patient's city and not happening in same week - Status: true
   - Old appointmentID city equals to patient's city and new appointmentID city not equals to patient's city and happening in same week Limit ==3 - Status: false
   - Old appointmentID city equals to patient's city and new appointmentID city not equals to patient's city and happening in same week Limit < 3 - Status: true

- Old appointmentID city equals to patient's city and new appointmentID city not equals to patient's city and not happening in same week - Status: false
- Old appointmentID city equals to patient's city and new appointmentID city not equals to patient's city and not happening in same week Limit < 3 - Status: true
- Old appointmentID city not equals to patient's city and new appointmentID city not equals to patient's city and happening in same week Limit == 3 - Status: true
- Old appointmentID city not equals to patient's city and new appointmentID city not equals to patient's city and not happening in same week Limit < 3 - Status: true
- Old appointmentID city not equals to patient's city and new appointmentID city not equals to patient's city and not happening in same week Limit < 3 - Status: true
- Extreme Test Case: Book 5 appointments on varied servers such that 4 fall in the same week and the 5th one falls in the next week. Take a patient who books 3 appointments that are on servers different than its own and fall in the same week. Now, the patient will try to book the 4th appointment and "Fail" as they have exceeded the weekly limit. Now, they should swap one of the appointments with the 5th one that was in the different week. Now try to book the 4th one that fell in the week as the other 3 and get a "Success".
  This test case checks the BookAppointment, CancelAppointment, SwapAppointment, UDP messaging, getAppointmentSchedule, as well as all the server ports and Logs.

Class Diagram


ServerObjectInterfaceOperations
removeAppointment(String, String) String
listAppointmentAvailability(String) String
bookAppointment(String, String, String) String
getAppointmentSchedule(String) String
cancelAppointment(String, String, String) String
swapAppointment(String, String, String, String, String) String
shutdown() void
ServerObjectZInterface
_ServerObjectInterfaceStub
addAppointment(String, String, String) String
removeAppointment(String, String) String
listAppointmentAvailability(String) String
bookAppointment(String, String, String) String
getAppointmentSchedule(String) String
cancelAppointment(String, String, String) String
swapAppointment(String, String, String, String, String) String
shutdown() void
_ids() String[]
readObject(ObjectInputStream) void
writeObject(ObjectOutputStream) void
creates
ServerObjectInterfaceHolder
insert(Any, ServerObjectInterface) void
extract(Any) ServerObjectInterface
type() TypeCode
id() String
read(InputStream) ServerObjectInterface
write(OutputStream, ServerObjectInterface) void
narrow(Object) ServerObjectInterface
unchecked narrow(Object) ServerObjectInterface
PatientInfo
detectPatientServer() String
detectPatientType() String
getPatientType() String
setPatientType(String) void
getPatientID() String
setPatientID(String) void
getPatientServer() String
setPatientServer(String) void
toString() String
ServerObjectInterfacePOA
_invoke(String, InputStream, ResponseHandler) OutputStream
all_Interfaces(POA,byte[]) String[]
_this() ServerObjectInterface
_this(ORB) ServerObjectInterface
AppointmentInfo
detectAppointmentServer(String) String
detectAppointmentTimeSlot(String) String
detectAppointmentDate(String) String
getAppointmentType() String
setAppointmentType(String) void
getAppointmentID() String
setAppointmentID(String) void
getAppointmentServer() String
setAppointmentServer(String) void
getAppointmentCapacity() String
setAppointmentCapacity(String) void
getAppointmentRemainingCapacity() int
getAppointmentDate() String
setAppointmentDate(String) void
getAppointmentTimeSlot() String
setAppointmentTimeSlot(String) void
isFull() boolean
getRegisteredPatientIDs() List<String>
setRegisteredPatientIDs() List<String>
addRegisteredPatientID() int
removeRegisteredPatientID(String) boolean
toString() String
ServerObjectInterfaceHolder
_read(InputStream) void
_write(OutputStream) void
_type() TypeCode
1
creates
1
AppointmentManagement
creates
setORB(ORB) void
getServerPort(String) int
addAppointment(String, String, int) String
removeAppointment(String, String) String
listAppointmentAvailability(String) String
bookAppointment(String, String, String) String
getAppointmentSchedule(String) String
cancelAppointment(String, String, String) String
swapAppointment(String, String, String, String, String) String
shutdown() void
removeAppointmentUDP(String, String, String) String
listAppointmentAvailabilityUDP(String) String
sendUDPMessage(int, String, String, String, String) String
getNextSameAppointment(Set<String>, String, String) String
exceedWeeklyLimit(String, String) boolean
addPatientToNextSameAppointment(String,String, List<String>) void
appointmentExists(String, String) boolean
isAppointmentOfThisServer(String) boolean
checkPatientExists(String) boolean
patientHasAppointment(String, String, String) boolean
removeAppointmentIfExists(String, String, String) boolean
addPatientAndAppointment(String, String, String) void
addAppointmentTypeAndAppointment(String, String, String) void
isPatientOfThisServer(String) boolean
onTheSameWeek(String String) boolean
getAllAppointments() Map<String, Map<String, AppointmentInfo>>
getPatientAppointments() Map<String, Map<String, List<String>>>
getServerAppointments() Map<String, PatientInfo>
addNewAppointment(String, String, int) void
addNewPatientToPatients(String) void
Logger
patientLog(String, String, String, String) void
patientLog(String, String) void
serverLog(String, String, String, String, String) void
serverLog(String, String) void
deleteALogFile(String) void
getFileName(String,int) String
getFormattedDate() String
Client
main(String[]) void
init(NamingContextExt) void
startConcurrencyTest(NamingContextContextExt) void
getServerID(String) String
checkUserType(String) int
patient(String, NamingContextExt) void
admin(String, NamingContextExt) void
askFortPatientIDFromAdmin(String) String
printMenu(int) void
promptForAppointmentType() String
promptForAppointmentID() String
promptForCapacity() int
ServerInstance
listenForRequest(AppointmentManagement, int, String, String) void
creates
creates
Server
main(String[]) void