**COMP 6231 Assignment 1 Design Documentation**

Name: Yajing Liu Student ID: 40268996

This project is to develop a distributed health care management system (DHMS) for managing medical appointments. Admins can use this system to add, list, and remove appointments. And patients can use this system to book, list, and cancel appointments.

**Project Architecture**

A diagram of a computer

Description automatically generated

Figure 1 UML class diagram

As shown in Figure 1, there are nine classes in this project. The following are the details about these nine classes.

**Driver**: This class handles command line user interface. It will create new client object and call clients’ corresponding methods based on the input.

**Client**: This class is an abstract class handling all methods of clients.

**PatientClient**: This class is a subclass of Client. It calls server’s methods in bookAppointment, getAppointment and cancelAppointment. It calls printInvalidCommandMessage in the remaining methods.

**AdminClient**: This class is a subclass of Client. It calls server’s methods in all six methods.

**AppointmentRegistry**: This class is the registry of three servers. It creates local registry and rebinds three new server objects with names.

**AppointmentInterfaces**: This is an interface to define server behaviors.

**MontrealServer**: This class is the Montreal server. It implements methods in AppointmentInterfaces and uses main method to listen other servers’ requests.

**QuebecServer**: This class is the Quebec Server. It implements methods in AppointmentInterfaces and uses main method to listen other servers’ requests.

**SherbrookeServer**: This class is the Sherbrooke server. It implements methods in AppointmentInterfaces and uses main method to listen other servers’ requests.

**Major Techniques**

**RMI (in Java)**

I used RMI (in Java) for client-server communication. I create local registry and rebind three new server objects with names in AppointmentRegistry (Figure 2). I search the server objects by names (Figure 3) and call the corresponding methods in client (Figure 4). I use server to handle method calling and finish jobs.

A screen shot of a computer program

Description automatically generated

Figure 2 AppointmentRegistry code

A screen shot of a computer code

Description automatically generated

Figure 3 Get server object by name in Client

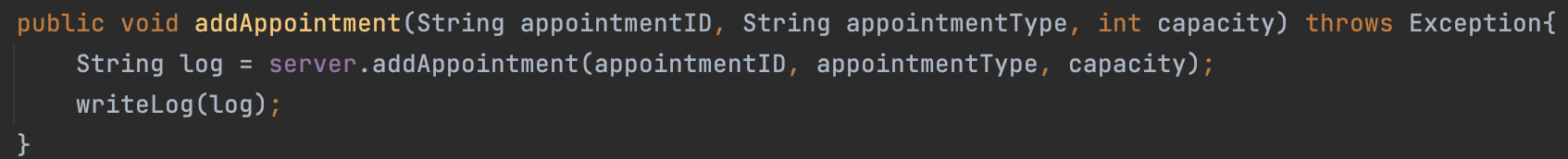


Figure 4 Call corresponding methods in AdminClient

A screen shot of a computer program

Description automatically generated

Figure 5 addAppointment in server

**UDP/IP (in Java)**

I use UDP/IP to handle server-server communication. Server can send request to get other servers’ appointments (Figure 6). And other servers can listen to the request and give response (Figure 7). The server which sends request can use the response to continue the work (Figure 6).

A screen shot of a computer program

Description automatically generated

Figure 6 Send UDP/IP request

A screen shot of a computer program

Description automatically generated

Figure 7 Receive UDP/IP request and sent response

**Test Scenarios**

1. To verify that server