DESIGN DOCUMENTATION

Assignment 2 – DRRS CORBA System

Vu Dang Khoa Trinh

Student ID: 40012738
Submission date: October 20, 2021.
Course: SOEN 423
Section: H

UML

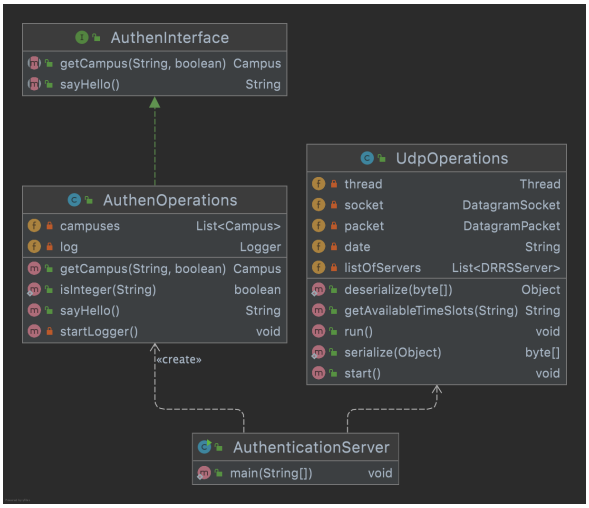

Figure 1: UML of package Authentication

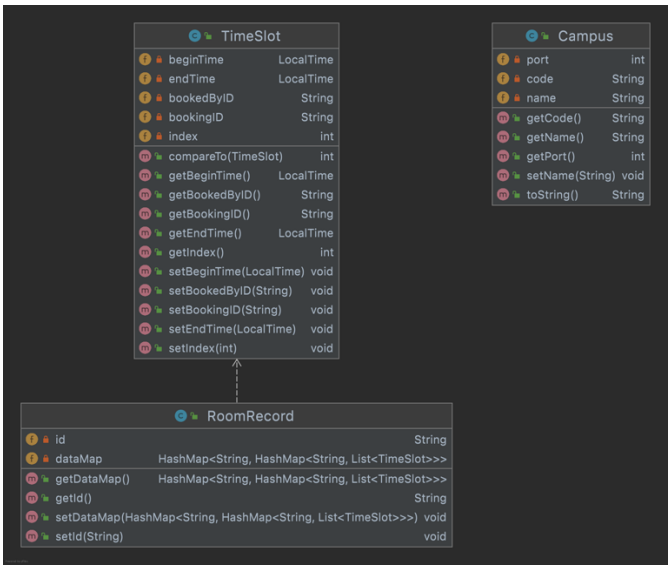

Figure 2: UML of package Business

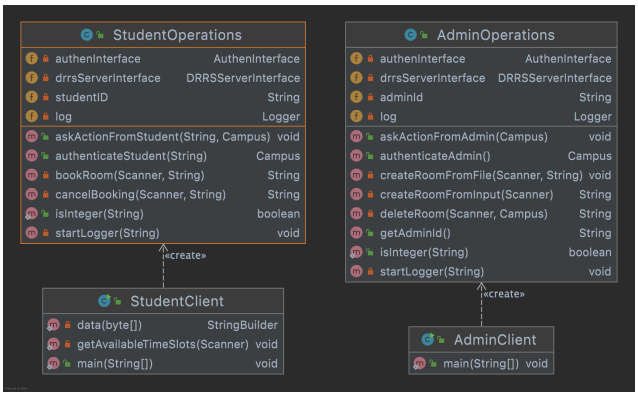

Figure 3: UML of package Client

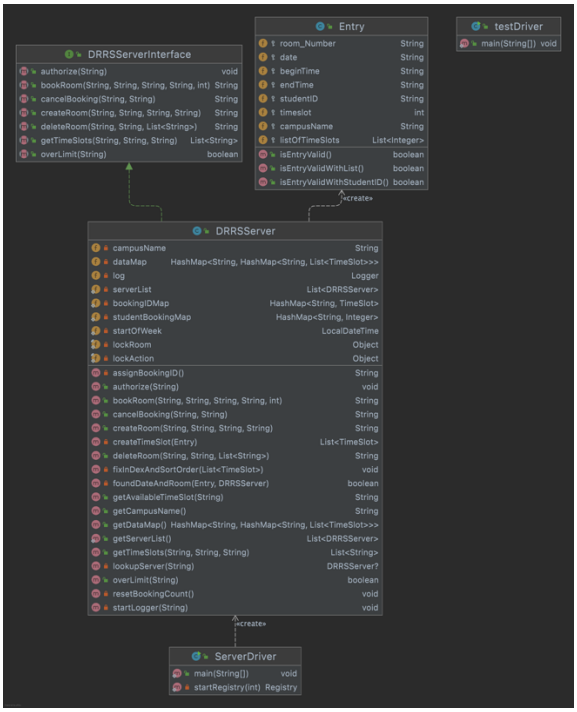

Figure 4: UML of package Server

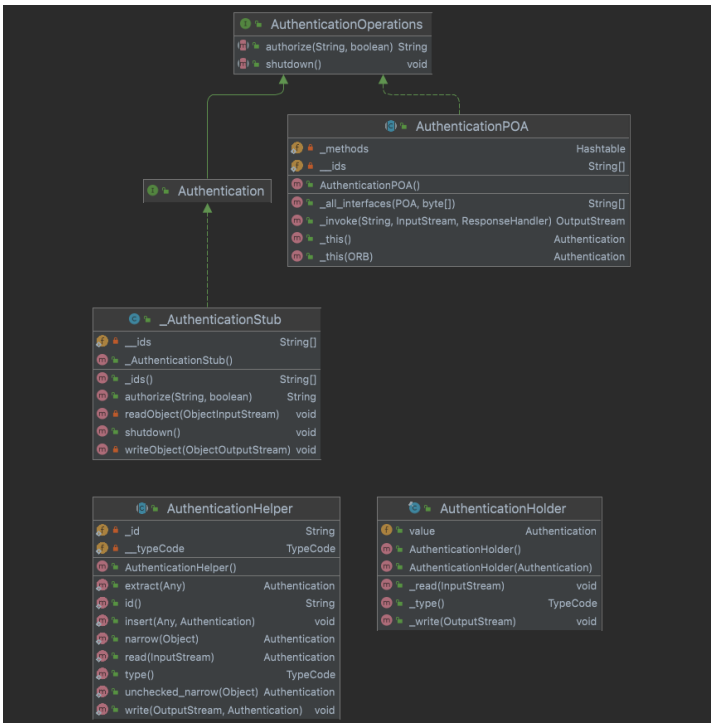

Figure 5: UML of AuthenticationApp

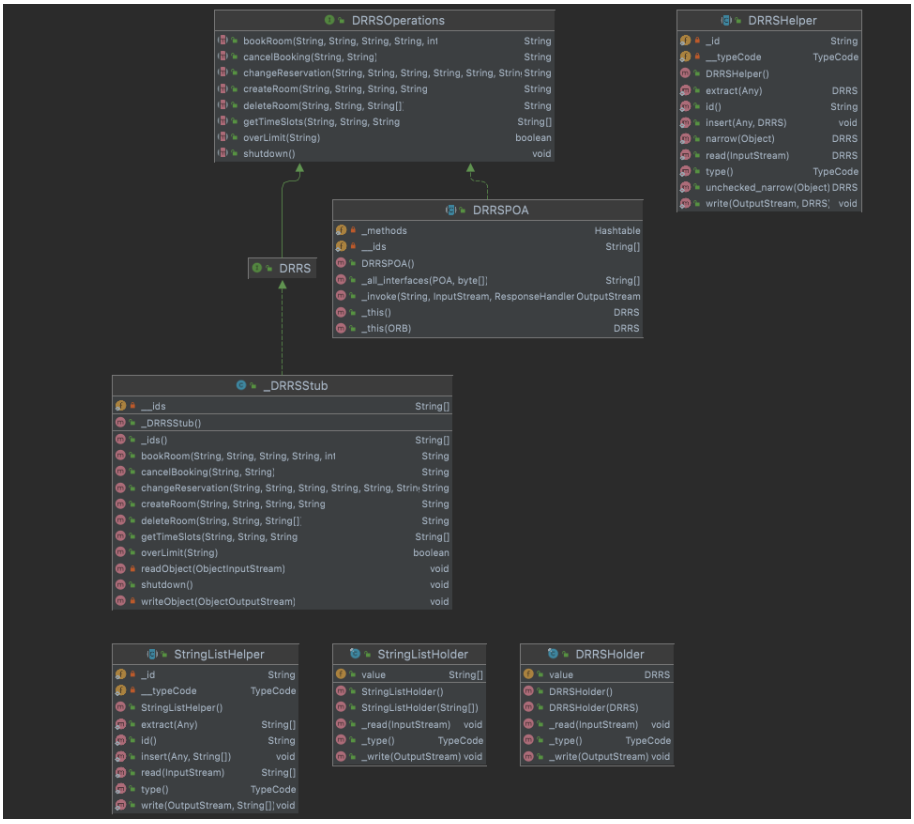

Figure 6: UML of DRRSApp

Techniques

The application is separated into six main packages:

1. DRRSApp
2. AuthenticationApp
3. Server
4. Client
5. Authentication
6. Business

Server

Package Server contains the code for the server side of CORBA implementation. It includes:

1. DRRSApp
    - This is the template of the Server in java, generated from the DRRSidl.idl file.
    - Inside, it has all the necessary classes to facilitate the communication through CORBA
2. DRRSServerInterface:
    - This is the template for any DRRSServer to implement. It includes the core functions such as createRoom(), deleteRoom(), bookRoom(), cancelBooking()
    - This interface implements the class Remote and makes sure that all of its implementations include the four aforementioned methods. These methods are what will be used by the Client side when they invoke a remote Server object.
3. DRRSServer:
    - This is the implementation of the DRRSServerInterface and its core methods.
    - It has the following members:
        - String campusName: holds the campus/server name when a new server is created

- LocalDateTime startOfWeek: holds the time that the server is created. It is then used to determine when a week has passed so that all the students' booking counts are reset.
- HashMap<> dataMap: this is a list that stores all the room and timeslot information of a server. When an admin creates or deletes a room, the information will be saved in here.
- List<> serverList: this is a static list that all servers share with each other. It has access to all the alive servers. This symbolically acts as the common database
- HashMap<> bookingIDMap: a static map that contains all booking IDs with their respective time slots.
- HashMap<> studentBookingMap: a static map that contains all student IDs and their booking counts. This map is used to prevent students from booking more than 3 time slot per week.
- Object lockRoom and lockAction: these two objects are used as locks to implement synchronization within the server.

4. ServerDriver:
    - This driver initializes and starts three separate DRRSServer objects: Westmount (port 10), Kirkland (port 20) and Dorval (port 30)
5. Entry
    - A helping class that stores information so that multiple classes can be passed within a method.
6. testDriver
    - used for testing

Authentication

Package Authentication contains the code for the authentication process. It includes:

1. AuthenticationApp

    a. This is the template of the Authentication Server in java, generated from the Authentication.idl file.
    b. Inside, it has all the necessary classes to facilitate the authentication through CORBA
2. AuthenInterface: this is the interface that has the core methods
3. AuthenOperations: this is the class that implements the interface above. It handles the authentification of clients.
4. UdpOperations: class that handles request going through UDP.
5. AuthenticationServer: initiates a AuthenOperations server object on port (7000)

Client

Client package contains the implementation of CORBA connection for the client side. It includes:

1. AdminOperations: is initiated whenever an admin logs in, keeps track of their information and handles all the requests that can be sent by them
2. AdminClient: responsible for connecting to the AuthenticationServer, authenticate the admin and then initiate an AdminOperations object to handle all his requests.
3. StudentOperations: is initiated whenever a student logs in, keeps track of their information and handles all the requests that can be sent by them
4. StudentClient: responsible for connecting to the AuthenticationServer, authenticate the student and then initiate a StudentOperations object to handle all his requests.

Business

Keeps all the helping classes that facilitate the data storage structure:

1. TimeSlot: Keep track of bookingId, studentID who books the slot, time
2. Campus: keeps track of the server name, port, code

Design

The Server side is separated into two distinct parts: the DRRSServer and the Authentication. This is to address the Separation of Concern design concept which spreads the tasks across different parts, instead of having one point of control which can be prone to errors.

Consequently, the client side was implemented in a way that they have to pass the Authentication process first in order to continue to connect to the DRRSServer.

The same idea is implemented inside the Server and Client side themselves, where an Operation class is created to handle only the requests once the authentication has been verified. Furthermore, having a separate operation object can help store the information of the user while he/she is signing in the system.

Data structure

As the application implements several mapping functions, HashMap is heavily used to store key-value entries such as bookingID-TimeSlot or Date-Room or Room-ListOfTimeSlot. This data structure is also helpful as it helps eliminate non-unique keys so that the system won't run into the situation of creating duplicates. However, due to the underlying functionalities of HashMap, the application was developed with heavy attention to situations where a duplicate key is potentially created, thus handling the situation appropriately.

Another data structure that is also heavily used in this application is ArrayList. To store information that does not require key-value mapping, this structure is significantly helpful in dynamically adding and deleting information, without the worry of managing the size and order.

Test

All the tests were done inside TestDriver class, which creates extensive possible situations:

- Create a brand new room successfully
- Create time slots in existed room successfully
- Attempt to create overlapping time slots in existed room
- Attempt to create same time slots in existed room
- Attempt to create room with the same name
- Delete a room successfully
- Attempt to delete a non-existent time slot
- Attempt to delete a non-existent room
- Attempt to delete a non-existent date
- Book a free time slot successfully
- Attempt to book a non-existent time slot
- Attempt to book a time slot in a non-existent room
- Attempt to book a time slot in a non-existent date
- Attempt to book a time slot in a non-existent campus
- Attempt to book a slot after room has been deleted
- Attempt to book a reserved time slot by the same user
- Attempt to book a reserved time slot by a different user
-

Challenging part

The most important and challenging part of this assignment is the establishment of the CORBA protocol, making sure that the server is created correctly and the client can connect to the server using the correct information.

Another challenging part is the establishment of UDP communication. The implementation requires more details such as serializing the data that are being passed through the connection and setting up the byte stream correctly to send the object.