**ASSIGNMENT-2**

**DISTRIBUTED ROOM RESERVATION SYSTEM (DRRS)**

**USING CORBA**

**CONCORDIA UNIVERSITY**

DISTRIBUTED SYSTEMS(COMP 6231)

SUBMITTED BY

MISBAHUDDIN ADIL SYED

STUDENT ID:40058698

**Distributed Room Reservation System (DRRS) using CORBA**

A Distributed System is a system that comprises of autonomous computers that are networked together using a distribution middleware. They help in sharing distinctive resources and capabilities to ensure users with a single and integrated coherent network.

The Distributed Room Reservation System is a distributed system that allows the Administrators to manage rooms by creating and deleting according to their availability. It allows the Students to manage their bookings by letting them book, cancel and check available rooms.

The system uses CORBA with Java IDL to facilitate the communication between remote objects in heterogonous environment. It also uses UDP connection to communicate and get available timeslots from peer servers.

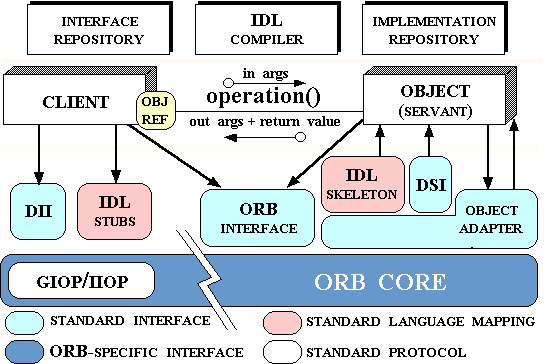
**CORBA:**

The Common Object Request Broker Architecture (CORBA) is an architecture and standard defined by the Object Management Group (OMG) for creating, distributing, and managing distributed program objects in a network.

CORBA facilitates communication between software written in different languages and running on different computers. Using CORBA, developers can ignore the specific implementation details from specific operating systems, programming languages, and hardware platforms. The method-call semantic between application objects residing either in the same or remote address spaces are normalized by CORBA (Same host or remote host).

CORBA uses an Interface definition language (IDL) to specify the interfaces that objects present to the implementations outside. CORBA then specifies a mapping from IDL to a specific implementation language like C++ or Java.

Picture taken from the internet.



**ORB:**

Object Request Broker (ORB) means that a client program (which may itself be an object) can request services from a server program or object without having to understand where the server is in a distributed network or what the interface to the server program looks like. General Inter-ORB Protocol (GIOP) is used to make requests or return replies between ORBs.

**OBJECT ADAPTER:**

ORB, on receiving request, forwards it to the object Adapter of the object. The object Adapter interacts with the object implementations skeleton, which performs data marshalling and invokes the called method in the object. It is responsible for creating remote object references for CORBA objects and activating remote objects.

**PORTABLE OBJECT ADAPTER:**

Portable Object Adapter is a particular type of object adapter that is defined by the CORBA specification. It allows an object implementation to function with different ORBs. It is a mechanism that connects a request using an object reference with the proper code to service the request.

**IMPLEMENTATION REPOSITORY:**

Implementation Repository is responsible for activating registered servers on demand and for locating servers that are currently running. Object implementations and object adapter names are stored with implementation repository when server programs are installed.

**INTERFACE REPOSITORY:**

Interface Repository is responsible for providing information about registered IDL interfaces to client and servers that require them.It supplies the names of the methods, names and types of arguments and exceptions.

**SKELETONS:**

Skeletons are generated in the language of the corresponding servers by an IDL compiler.

**CLIENT STUBS:**

These are generated in client language. The stub is generated from an IDL interface by an IDL Compiler for the client language. It marshals the arguments and un-marshals requests and replies.

**DYNAMIC INVOCATION INTERFACE:**

Dynamic Invocation Interface allows clients to make dynamic invocations on remote CORBA objects. It is used when it is not practical to employ proxies.

1. **ARCHITECTURE:**

The Distributed Room Reservation System is a distributed system which is used by the administrators to show the availability of the rooms and also by the students who reserve the rooms across the universities different campuses.

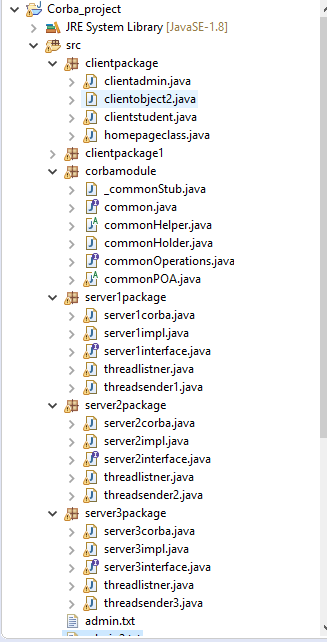
The Clients i.e. the Students and the Admins have varying accessibility. The Admins can either Create a room or Delete a room whereas the Students can Book a room, get available Time-Slots and Cancel the booking.

There are 3 campuses namely Dorval-Campus ( DVL), Kirkland-Campus (KKL) and Westmount-Campus (WST). Each Campus is assigned a Server. Each server has methods that are accessible by the Students and Admins separately.

The Student and the Admin and represented by respective classes. These classes can call remote methods present on the server through CORBA. Each student is associated with a particular campus which is known by the User Id. To get available Time-Slots present on other servers, the server associated with the student has to communicate with them using UDP Connection.

1. **STRUCTURE**

The Distributed Room Reservation System has the following Structure:



1. **SERVER METHODS**

Each Server contains methods that are separately accessible by Students and Admins respectively. The methods are

1. **Boolean CreateRoom(String rno,String date, String timeslots):**
2. **Boolean DeleteRoom(String rno, String date, String timeslot):**
3. **String Bookroom(String campusName, String rno, String date, String Timeslot):**
4. **String getAvailableTimeSlot (date):**
5. **String CancelBooking (bookingID):**

.

The methods accessible by **Admin** are:

1. **Boolean CreateRoom(String rno, String date, String timeslots):**

The method CreateRoom is a method that is only accessible by an Admin. An Admin, who has the information about the availability of the rooms can create a room record of availability of a room using this method. The admin has to provide Room Number, Date and the Time-Slot of the available room. It returns true if the record creation is successful else returns false.

1. **Boolean DeleteRoom(String rno, String date, String timeslot):**

The method DeleteRoom is another method that is only accessible by an Admin. An Admin can only delete a room only if it was earlier created by an Admin. It returns True if the Room Deletion was successful or else returns False.

The methods accessible by **Students** are:

1. **StringBookroom(String campusName,String rno,String date,String Timeslot):**

The method Bookroom is a method that is accessible by a Student. A Student can book a room by providing the Campus Name, Room Number, Date and the Timeslot. It returns a bookingID on successful booking of room.

1. **String getAvailableTimeSlot (date):**

This method checks the HashMap for the available Timeslots. It gets the available Timeslots from other Servers by communicating with them through UDP connection. It receives the count from other servers and returns all the timeslots as a String.

1. **Boolean CancelBooking (bookingID):**

This method checks the Hashmap for the given bookingID and deletes the booking record. It returns True if successful else returns False. It is accessible by a Student.

1. **String changeReservation(String studentid, String booking\_id, String new\_date,new** **String new\_campus\_name, String new\_room\_no, String new\_time\_slot):**

This method is accessible by a student. This method first checks if the new room is available, and if it is available it books the room on the new timeslot and then cancels the old timeslot. It returns the new bookingid.

**IDL INTERFACE:**

module corbamodule

{

interface common

{

boolean createroom(in string rno,in string date,in string timeslot);

boolean deleteroom(in string rno,in string date,in string timeslot);

string bookroom(in string campusName,in string rno,in string date,in string timeslot,in string UID);

string getAvailableTimeSlot (in string date);

string cancelBooking(in string bookingID,in string userid);

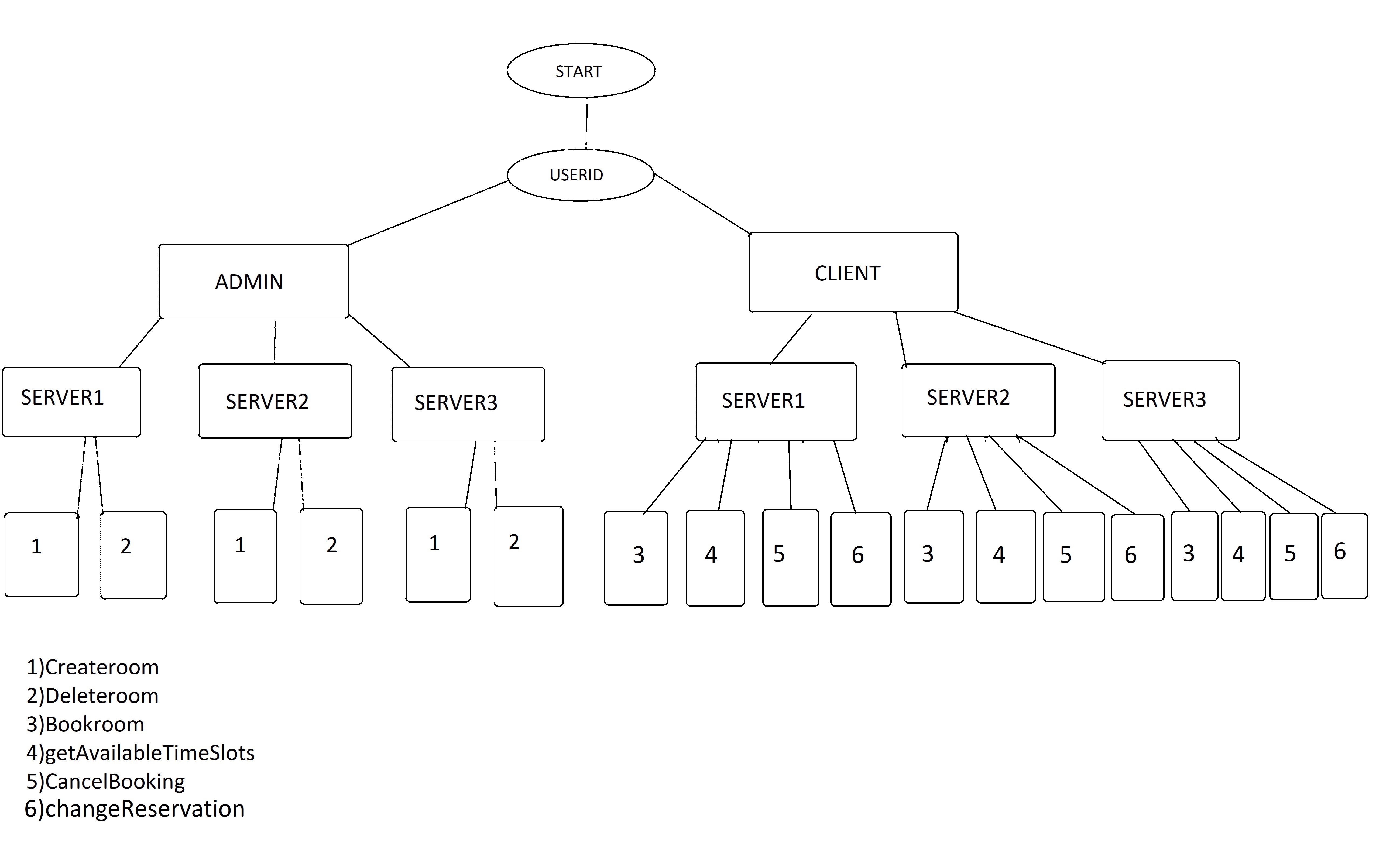
string changeReservation(in string studentid,in string booking\_id,in string new\_date,in string new\_campus\_name,in string new\_room\_no,in string new\_time\_slot);

};

};

1. **WORKFLOW**

The Workflow of the System has the following workflow:



The System first asks the user to enter the Userid and based on the 4th letter of the UserID, it redirects the control to the respective class. If the 4th letter is “A”, then the control is redirected to AdminClient and if 4th letter is “S”, then the control is redirected to StudentClient.

If the control is passed to AdminClient, it asks the user to enter a number associated with an action. 1 for createroom and 2 for deleteroom. Based on the Userid, the programs calls the method of the server associated with the Userid. 1 and 2 gives a call to the respective methods of the server using the server object through CORBA.

If the control is passed to StudentClient, it asks the user to enter a number associated with an action. 1 for bookroom, 2 for getAvailableTimeSlot, 3 for CancelBooking and 4 for changeReservation. Based on the Userid, the programs calls the method of the server associated with the Userid. 1, 2 and 3 gives a call to the respective methods of the server using the server object through CORBA.

The Createroom method on the server-side creates an entry in the HashMap for the given Date, Room number and Timeslot and makes the availability field Available and returns True to the Admin..

The Deleteroom method deletes the specified room on the specific Date and Timeslot from the HashMap.It returns a Boolean value depending upon the operation.

The Bookroom method sets the availability field with the bookingID in the HashMap and returns the bookingID to the User.

The getAvailableTimeSlots method iterates through the HashMap to get the count of the available rooms for the local server. It gets the count of other servers by communicating through UDP connection. It uses threadListner and threadSender class to send the count.

The CancelBooking method searches for the bookingID and the sets the availability field to available.It returns a Boolean value.

The ChangeReservation method checks if the new timeslot is available, books the new room, cancels previous booking and returns new booking id.

1. **DATABASE STRUCTURE**

The database Structure used here is nested HashMap.

The first HashMap ‘a’ has Date as the key and another HashMap whose key is the room number and its value is another HashMap whose key is Timeslot and value is booking id.

*a*.put(date, **new** HashMap<String,HashMap<String,String>>());

*a*.get(date).put(rno, **new** HashMap<String,String>());

*a*.get(date).get(rno).put(timeslot,"Available");

1. **CONCURRENCY**

Concurrency in distributed systems can be attained using Multithreading and Synchronization.

In this system, we need concurrency when many students access the methods such as bookroom(), getAvailableTimeSlots() and Cancelbooking(),ChangeReservation().

The logic of these methods are placed in respective classes and objects of these classes are called using dynamic threads.

The logic code is placed in the critical and synchronized block so that the modification of the HashMap does not overlap and cause data inconsistency.

1. **TEST CASES**

Test Cases using Admin Client and Student Client and by providing different inputs:

1. UserID :DVLA10005

createroom(String rno, String date, String timeslot)

INPUT:

101

28-10-2017

1:00-3:00

OUTPUT:

Room Created

2)UserID: DVLA10005

deleteroom(String rno, String date, String timeslot)

INPUT:

101

28-10-2017

1:00-3:00

OUTPUT:

Room deleted

3)UserID: KKLA10006

createroom(String roomnumber,String date,String t\_slot)

INPUT:

201

08-10-2017

10-1

OUTPUT:

Room Created

4)UserID:DVLA12345

createroom(String roomnumber,String date,String timeslot)

INPUT:

301

08-10-2017

10-1

OUTPUT:

Room Created

5)UserID:DVLS12345

bookroom(String campusName,String rno,String date,String timeslot,String UID)

INPUT:

DVL

101

08-10-2017

10-1

OUTPUT:

your booking id is 5406c32e-cd82-4096-9364-42e6292a4602

6) UserID: DVLS12345

changeReservation(studentid,bookingid,date,campusname,roomnumber,timeslot)

INPUT:

5406c32e-cd82-4096-9364-42e6292a4602

OUTPUT:

ab87h999-445d-9030-0999-009abd1c5678

7)UserID: DVLS12345

Cancelbooking(bookingID)

INPUT:

ab87h999-445d-9030-0999-009abd1c5678

Booking cancelled