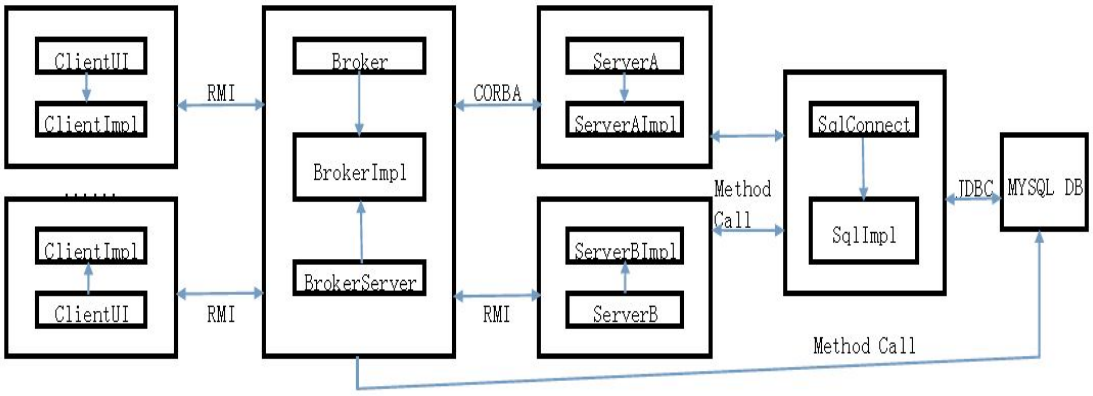
Hotel Booking Design Document

1.Architecture

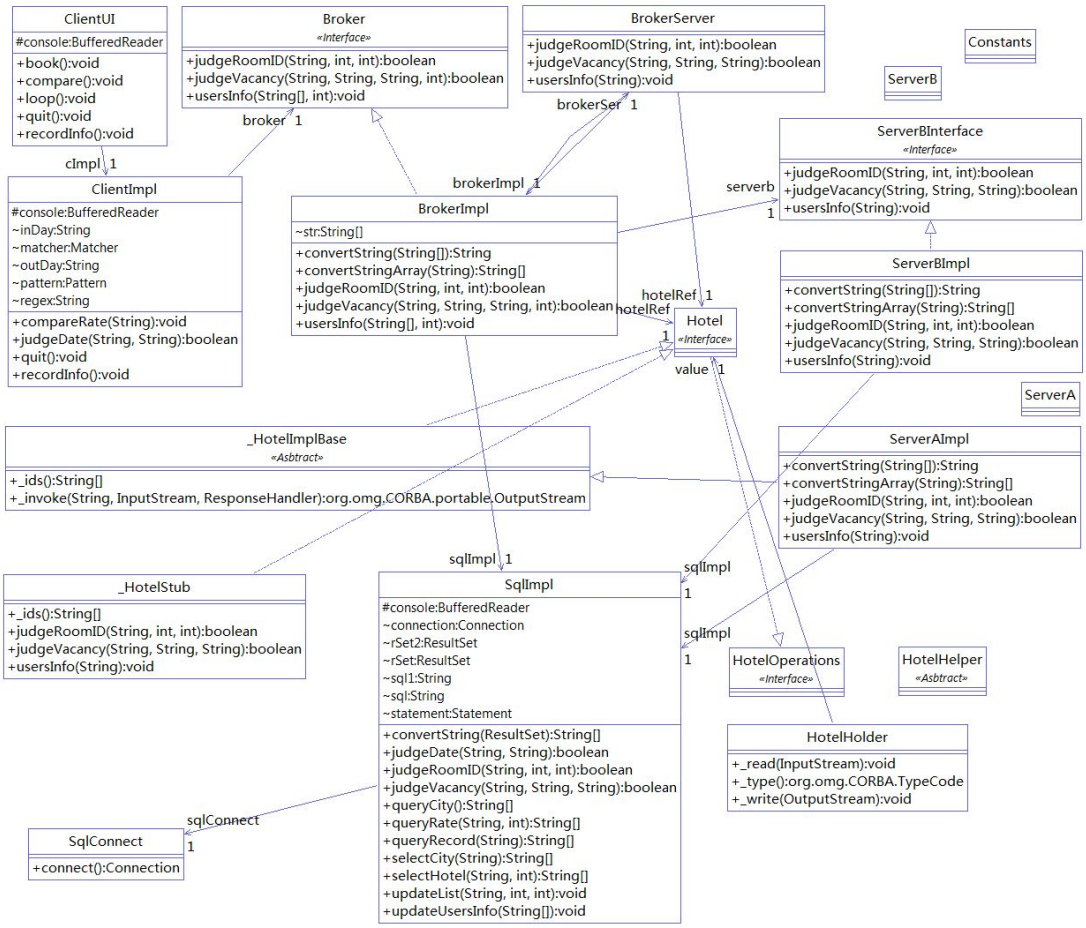


Introduction:

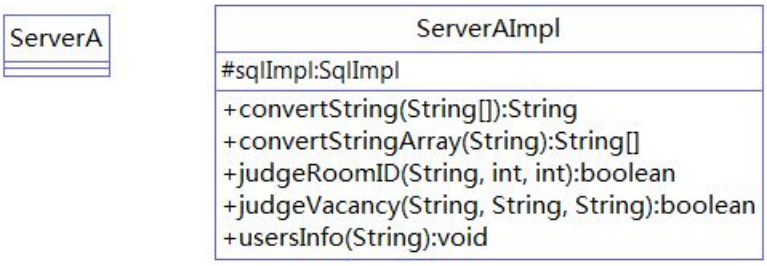
This Hotel Booking is a typical three tiers system with the client, broker and server. The section of sql is another part of the server which can be separated for clear viewing. Client can visit the broker to get the city, hotel and room information, but must to visit server if booking. The connection from the client to broker and broker to ServerB use the RMI technology and from the broker to ServerA uses the CORBA technology. The two servers are separate and independent. The main functions are booking room, comparing rate, checking available and searching order. When users set up a client, they will input request through the client. Then the client will send responding request to broker, the broker will deliver the request to the corresponding server. The related server visit the background MYSQL databases to check the information and return the data result back to the broker then broker to the client. It is implemented satisfactorily, provides the required functionality, is robust, exhibits the application of sound design principles and code conventions.

2.UML Class diagram

The overall diagram.



a) serverA
   i. ServerA.java
   ii. ServerAImpl.java



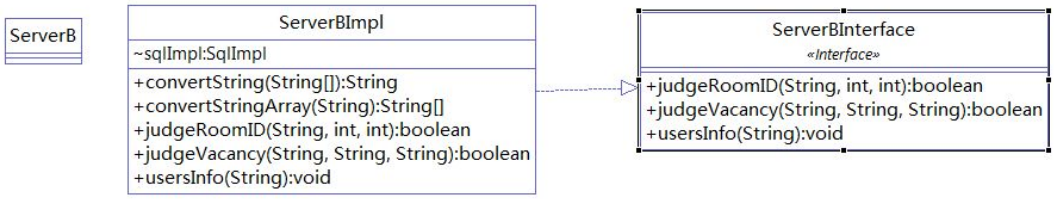
Server has the responsibility to receive messages from broker and do some handling work to send the message to do jobs on the mysql databases. When server gets result from responding tables in database, it will do some handling work again and return the message to the broker.

There are two classes in the serverA package: ServerA and ServerAImpl. The ServerA declares the information of the tech of CORBA and the ServerAImpl realize the functions details. The package called corba.hotelbooking will be introduced below is related to the technology of CORBA.

They as a middle system for building distributed application , use object methods instead of procedure call to realize the remote communication.

The class including functions are as follows: convertStringArray(String), convertString(String[]),judgeRoomID(String,int,int), judgeVacancy(String,String,String), userInfo(String).

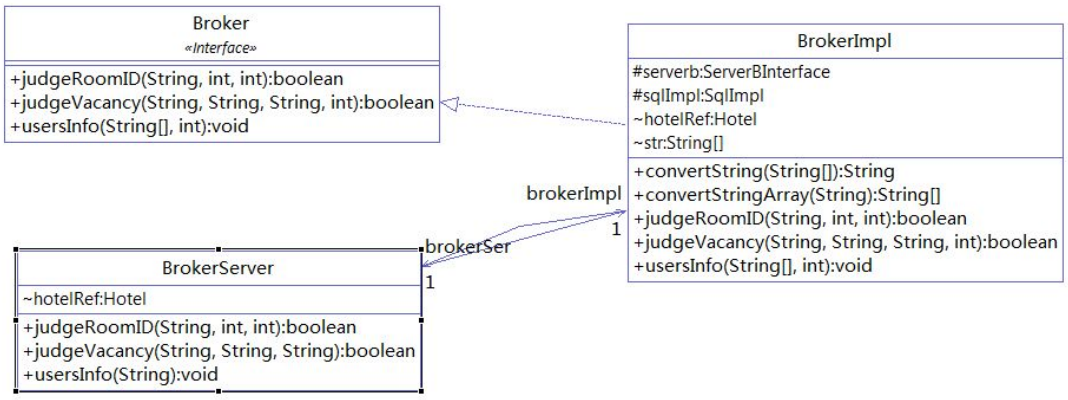
b) serverB
   i. ServerB.java
   ii. ServerBImpl.java



There are two classes in the serverB package: ServerB and ServerBImpl. The ServerB declares the information of the tech of RMI and the ServerBImpl realize the functions details. They as a middle system for building distributed application to realize the remote communication.

The class including functions are as follows: convertStringArray(String), convertString(String[]),judgeRoomID(String,int,int), judgeVacancy(String,String,String), userInfo(String).

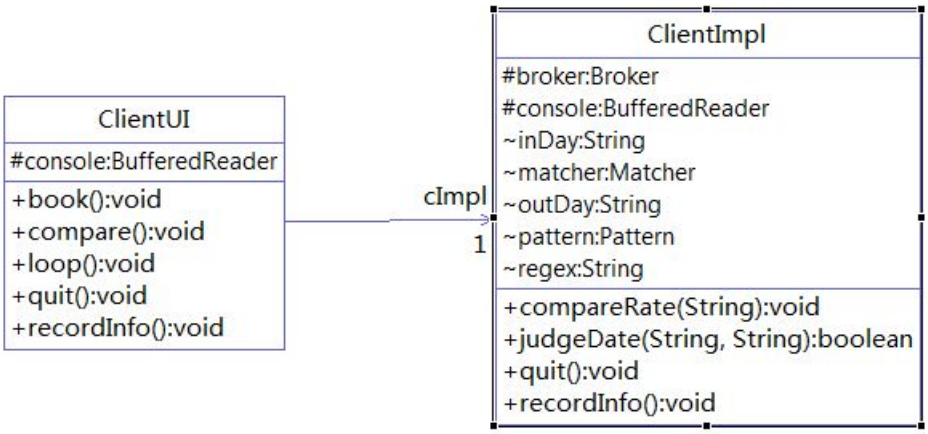
c) broker
   i. Broker.java
   ii. BrokerImpl.java
   iii. BrokerServer.java



This is a bridge between client and server to realize different communication using RMI or CORBA technology, but for inquiry function it can directly connect to the background databases. The Broker is an interface, BrokerServer achieve the connectivity and BrokerImpl achieve the specific function.

The class including functions are as follows: convertStringArray(String), convertString(String[]),judgeRoomID(String,int,int), judgeVacancy(String,String,String), userInfo(String).

d) client
   i. ClientUI.java
   ii. ClientImpl.java

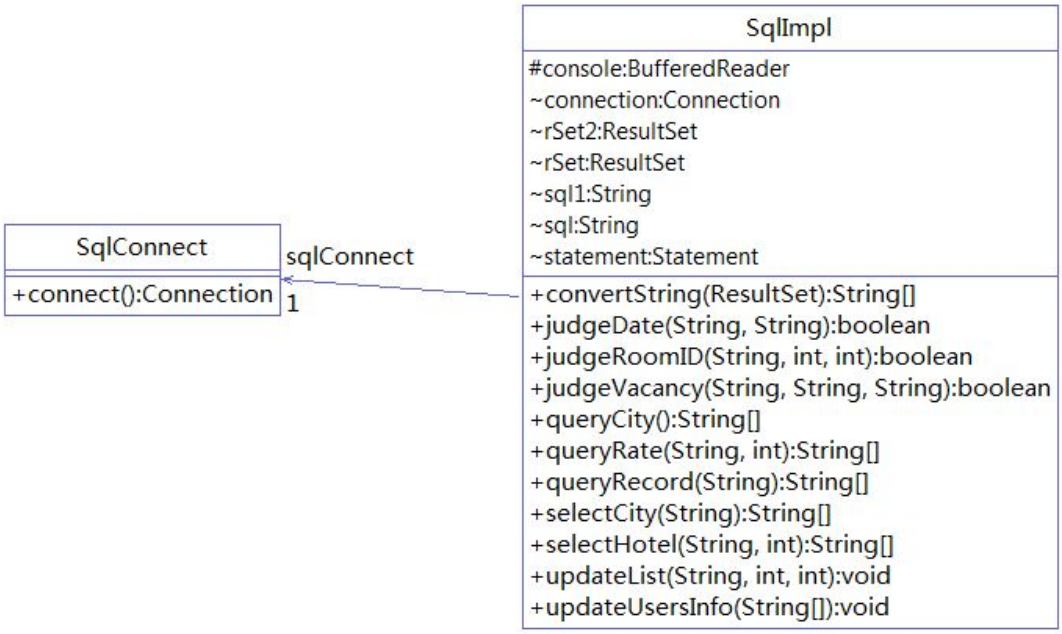


Client has the responsibility to receive the requirements from the users and send them to the broker, then client will display results to users. There are two Class in the package Client: ClientUI and ClientImpl. The major functions connecting with broker and doing operations are declared in the ClientImpl including: compareRate(String), recordInfo(), judgeDate(String,String), quit(). The ClientUI class is the UI and giving definitions to all the functions. When user opens a client, it means setting up a clientUI object, and the clientUI object will create a ClientUI object, the functions just call the responding functions in the ClientImpl.

The class including functions are as follows: compareRate(String), judgeDate(String,String), quit(),

recordInfo().

e) sql
   i. SqlConnect.java
   ii. SqlImpl.java



This is a part of the server. For clear viewing, it can be separated independently. The server call it then connect to the background databases.

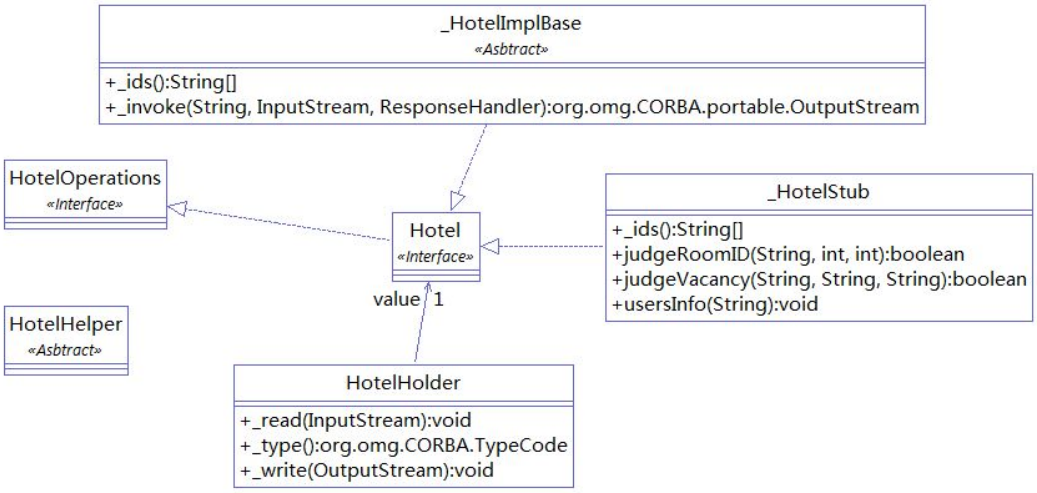
The class including functions are as follows: convertString(ResultSet), judgeRoomID(String,int,int), judgeDate(String,String), queryRate(String,int), queryCity(), queryRecord(String), selectCity(String), selectHotel(String,int), judgeVacancy(String,String,String), updateUsersInfo(String[]), updateList(String,int,int).

f) corba.hotelbooking
   i. _HotelImplBase.java
      An abstract class comprising the server skeleton and provides basic CORBA functionality for the server and implements the Hotel interface.
   ii. _HotelStub.java
      This is the client stub and provide CORBA functionality for the client.
   iii. Hotel.java
      Contains the java version of IDL interface.
   iv. HotelHelper.java

Provides auxiliary functionality.

v. HotelHolder.java
Holds a public instance member of type Hotel.

vi. HotelOperations.java
Contains the java method signatures for all operations in IDL file.



The six java files are generated by the following file called Hotel.idl. These files serve the CORBA.

Hotel.idl

```
module corba {
    module hotelbooking {


        interface Hotel{

            void setRoom(in string str,in long count,in long temp);

            void usersInfo(in string str);

            boolean judgeRoomID(in string str,in long count,in
            long temp);

            string getHotelName(in long i);

            boolean judgeVacancy(in string id,in string inDay,in
            string outDay);

        };
    };
};

```

g) constants

i. Constants.java

Constants

Create constants has lots benefits and convenient in replacing, searching and understanding. Some information can be stored and preprossessed. The constants I have used in the program as followed.

```
public class Constants {
    public final static String BOOK = "BOOK";
    public final static String RECORDINFO = "ORDERS";
    public final static String COMPARE =
"COMPARERATE";
    public final static String QUIT = "QUIT";
    public final static String ERROR = "wrong. input again";
    public static final String CR_LF = "\r\n";
    }
```

3.Design of MYSQL

a) User name: fit5183a2
b) No password
c) Database name: id26346966
d) Tables: bj7days, bjjjstar, sh7days, shjjstar, cityhotel, roomcheck and user
e) Tables structure: