Design Document: Hotel Booking Broker System

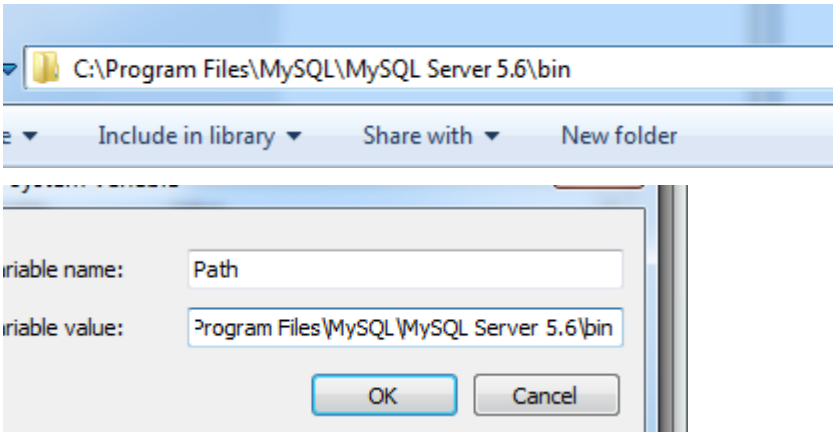
Name: Zhu Shuai ID: 26346915

Target:

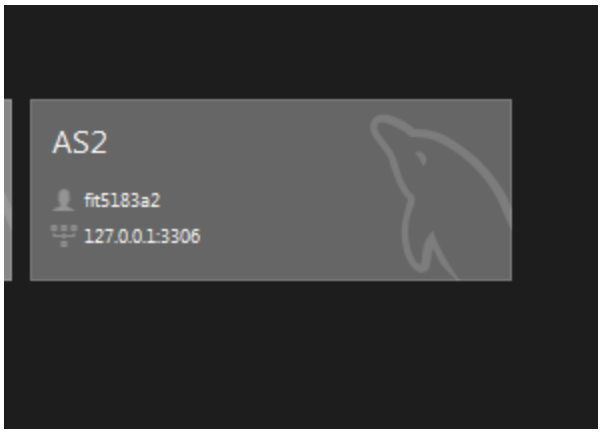
The level of assessment I have attempted is **D**, and this architecture of this system is 3 tiers: Client-Broker-Hotel server. The technology which I use between Client-side and Broker-side is **RMI**. Both **CORBA** and **RMI** are used between broker-side and hotel server-side.

Deployment instruction:

The most import thing is to import the database before you execute this system. First of all, configure the environment of the mySQl. Find your mySQL path add to the environment Variables

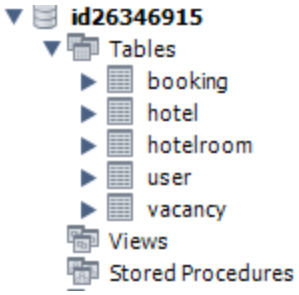


Then create a no password account "**fit5183a2**" on your MySQL workbench.



Enter the below command you can successfully import the **id26346915.sql** file.

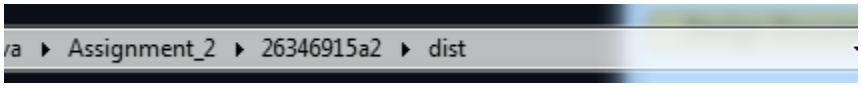
```
C:\Users\hp>mysql -ufit5183a2 -p <id26346915.sql
```

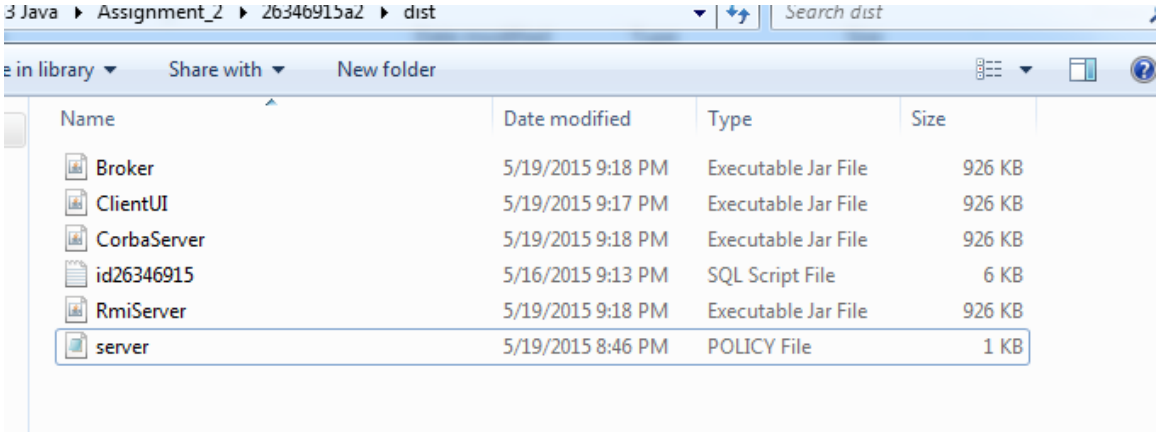


User manual or usage guidelines:

This is the guidelines under the DOS environment; please pay attention to every step before you execute the system.

Before you execute this system make sure that you have 4 jar package and server policy file in your specify directory.





Step1. First of all, you should start 2 different kinds of hotels server (2 types technologies to implements hotel server one is **RMI** the other is **CORBA**)

RMI hotel server start:

Command: java -Djava.security.policy=server.policy -jar RmiServer.jar

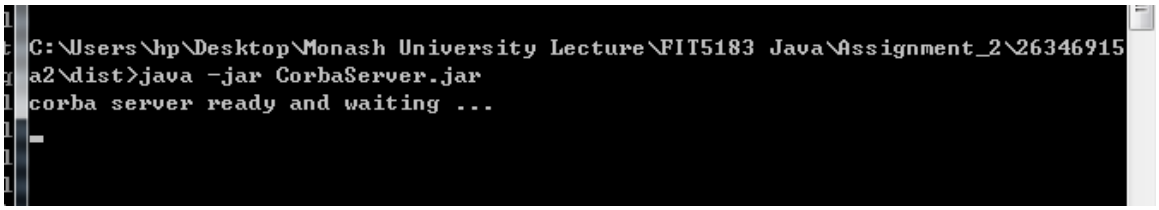
```
C:\Users\hp\Desktop\Monash University Lecture\FIT5183 Java\Assignment_2\26346915
a2\dist>java -Djava.security.policy=server.policy -jar RmiServer.jar
 rmi server run...
```

COBAR hotel server start:

Firstly, you should enter “tanmeserv” in command line:

```
C:\Users\hp>tnameserv
Initial Naming Context:
IOR:000000000000002b49444c3a6f6d672e6f72672f436f734e616d696e672f4e616d696e67436f
6e746578744578743a312e3000000000000010000000000000009a000102000000000e3137322e3136
2e3132302e3232000384000000045afabcb0000000020000f424000000001000000000000000020000
0008526f6f74504f41000000000d544e616d65536572766963650000000000000000800000001000000
0001140000000000000020000000100000020000000000000100010000000205010001000100200001
0109000000010001010000000026000000020002
TransientNameServer: setting port for initial object references to: 900
Ready.
```

Then



Step2. start the Broker server:

(when you start broker server you should indicate the technology you want use corba or rmi, here we just use rmi as an example. Of course you can use corba, but after you use rmi, you should restart the broker server with the parameter “corba”)

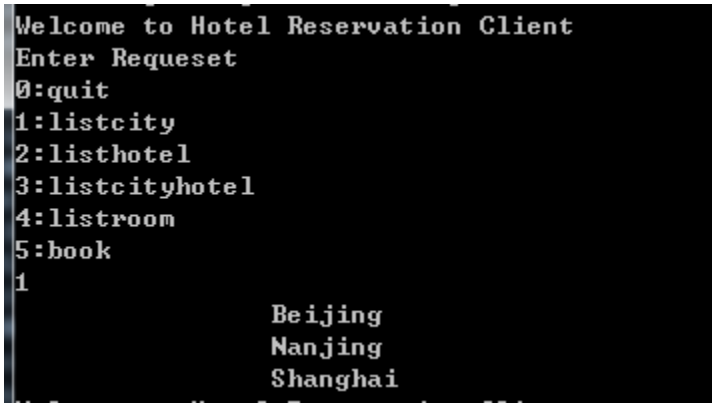
Step3. execute the client(just start one client to show how this system work ,you can start more than one client)

From the figure you can see that client should specify the argument too, because the client is executed in my own computer, so the argument is "localhost".

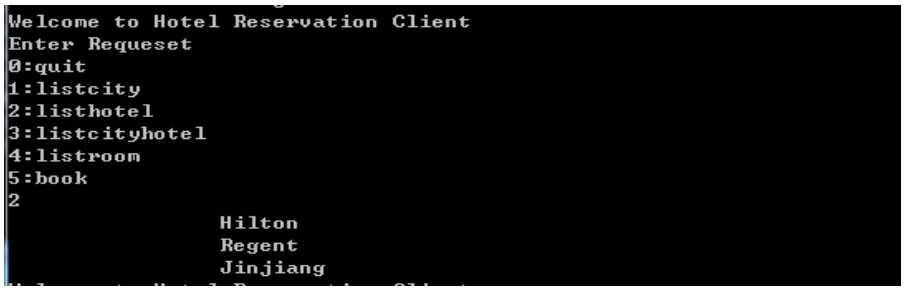
Step4. Function demonstrate

We can know that there are five function keys from the third step, here we demonstrate one by one.

list city:



list hotel:



list city hotel:

```
Welcome to Hotel Reservation Client
Enter Requeset
0:quit
1:listcity
2:listhotel
3:listcityhotel
4:listroom
5:book
3
please input city
Beijing
                Hilton
```

listroom:

```
4
please enter the city
Beijing
please enter the hotel
Hilton
               Beijing         Hilton          150             3 star
single
```

(we can see from the corba server-side)

```
C:\Users\hp\Desktop\Monash University Lecture\FIT5183 Java\Assignment_2\26346915
a2\dist>java -jar CorbaServer.jar
corba server ready and waiting ...
DataBase connecting sucess
```

book:

```
5
please enter the city
Beijing
Please enter the hotel name
Hilton
please enter the room type you want book<single or double>
single
please enter the check in date (formate:yyyy-mm-dd)
2015-04-01
please enter the check out date (formate: yyyy-mm-dd)
2015-04-00
wrong date formate
please enter the check in date (formate:yyyy-mm-dd)
2015-04-03
please enter the check out date (formate: yyyy-mm-dd)
2015-04-08
```

```
the room is availabe and your ID is321678.0
please enter your information
please enter your name
zhushuai
please enter your credit number
342222199202194430
please enter your email
1052785913@qq.com
booking sucess!!!
```

Because this system does not have the function of check booking, from the database we can see the state of booking.

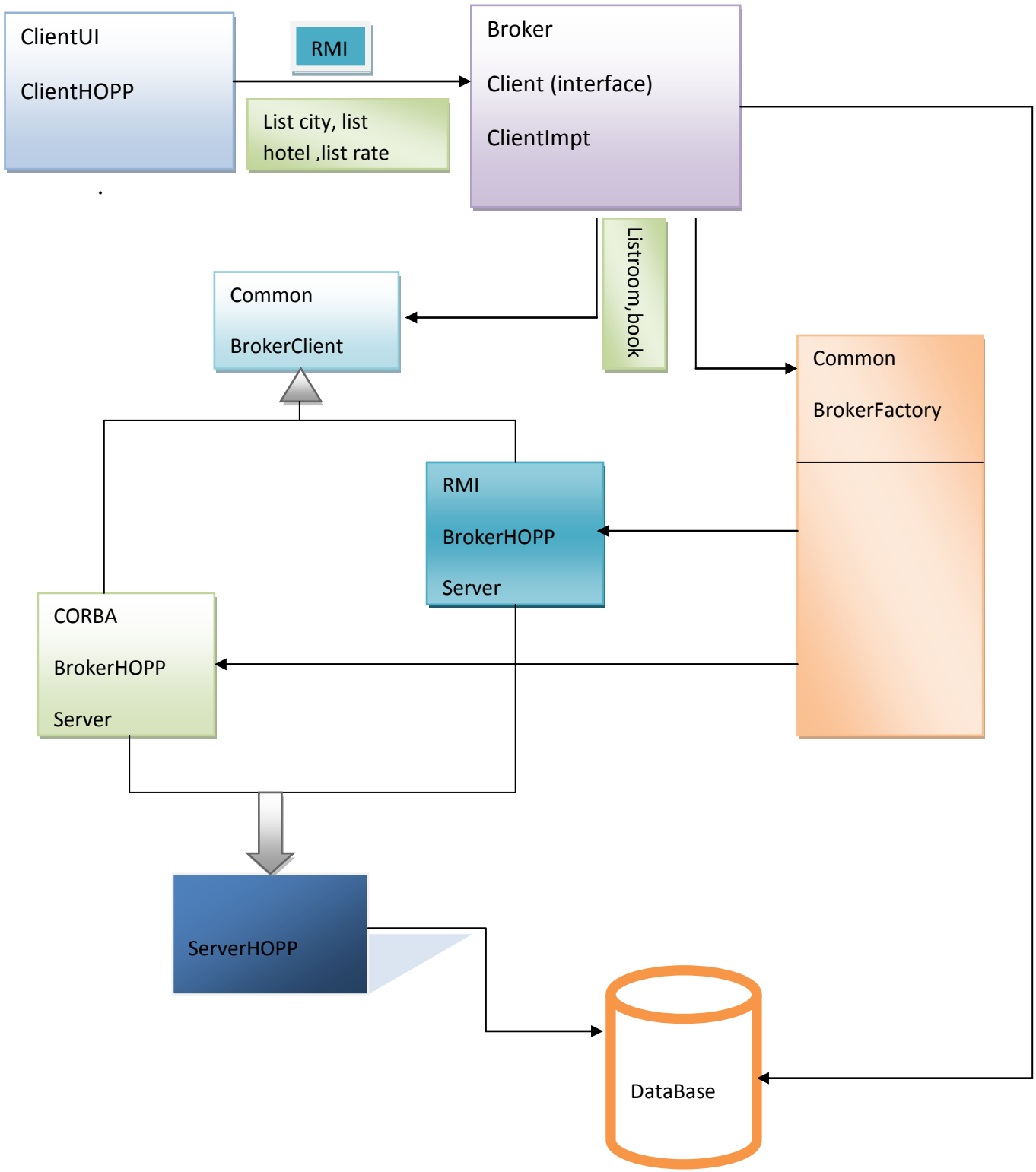
Booking table:

id	check_in	check_out	hotel	city	style
321678	2015-04-03	2015-04-08	Hilton	Beijing	single
(NULL)	*(NULL)*	*(NULL)*	*(NULL)*	*(NULL)*	*(NULL)*

Vacancy table:

ID	Riqi	AvaNo
1	1	10
1	2	10
1	3	9
1	4	9
1	5	9
1	6	9
1	7	9
1	8	9
1	9	10
1	10	10
1	11	10
1	12	10

System architecture diagram:



From the diagram we can see this Hotel Booking Broker System is a typical **3-tier** system, **Client- Broker-Hotel Server** 3 tier mechanism.

ClientUI is **independent** of the network code, it does not communicate directly with the Broker but through **ClientHOPP. ClientHOOP** will use the technology of **RMI** to communicate with **Boker.**

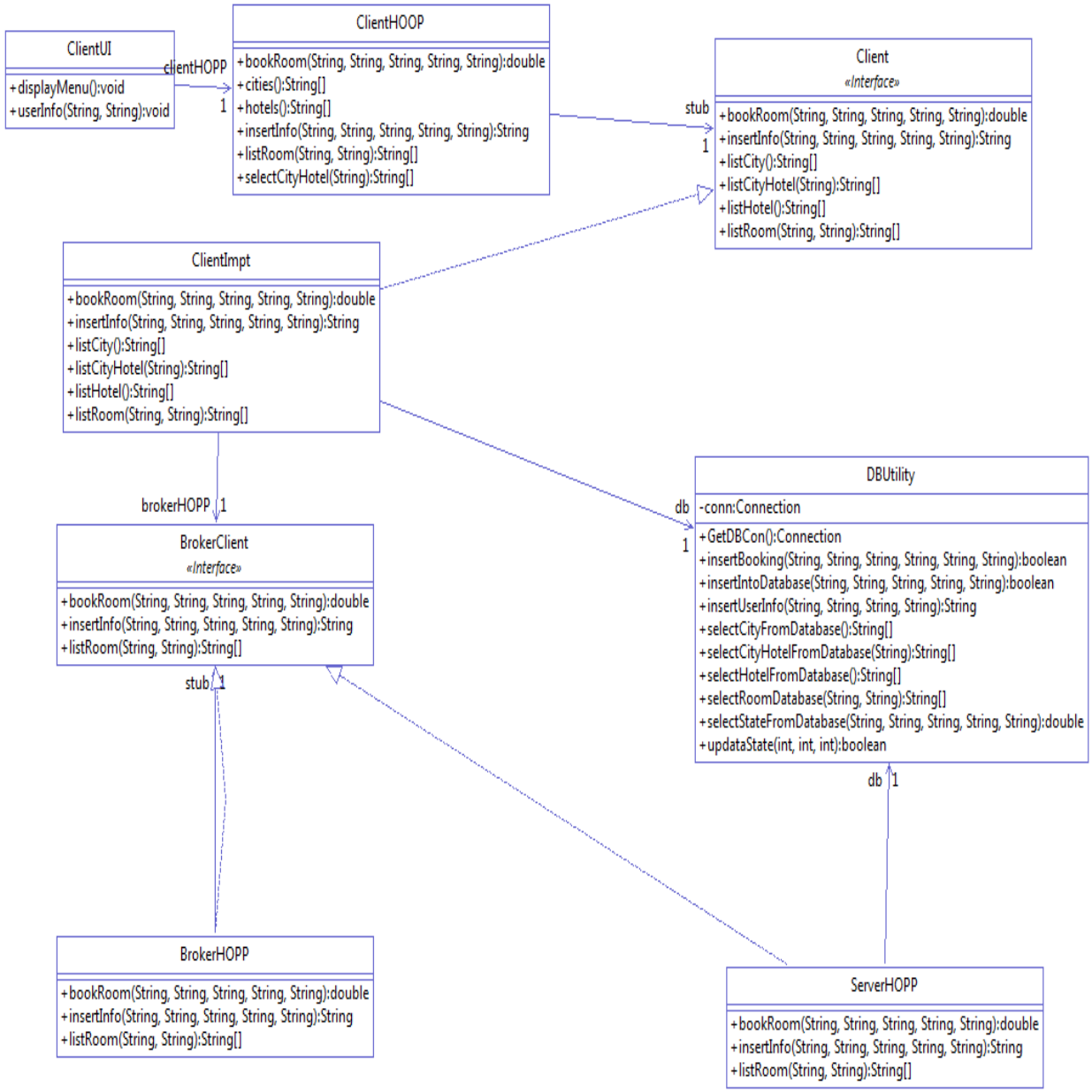
From the architecture we should know that when we list hotel, list city and list rate, The **Client-side** can accept the message directly from the **Broker-side** by the communication way of **RMI**.

But if we want list room and book room, Broker-side must communicate with hotel server. There are 2 kinds of technologies to implements the function of listing room and booking room. How to select one of the technologies to work? The method is that the Broker-side will create a path to communicate with the **RMI** server or **CORBA** server through the **BrokerFactory**. The BrokerFactory will generate special BrokerClient half-objects. It uses command line parameters passed from **Broker-side** to determine which half-object to return. All half –objects implements the **BrokerClient** interface. Then the half-object will get the information through **ServerHOOP.**

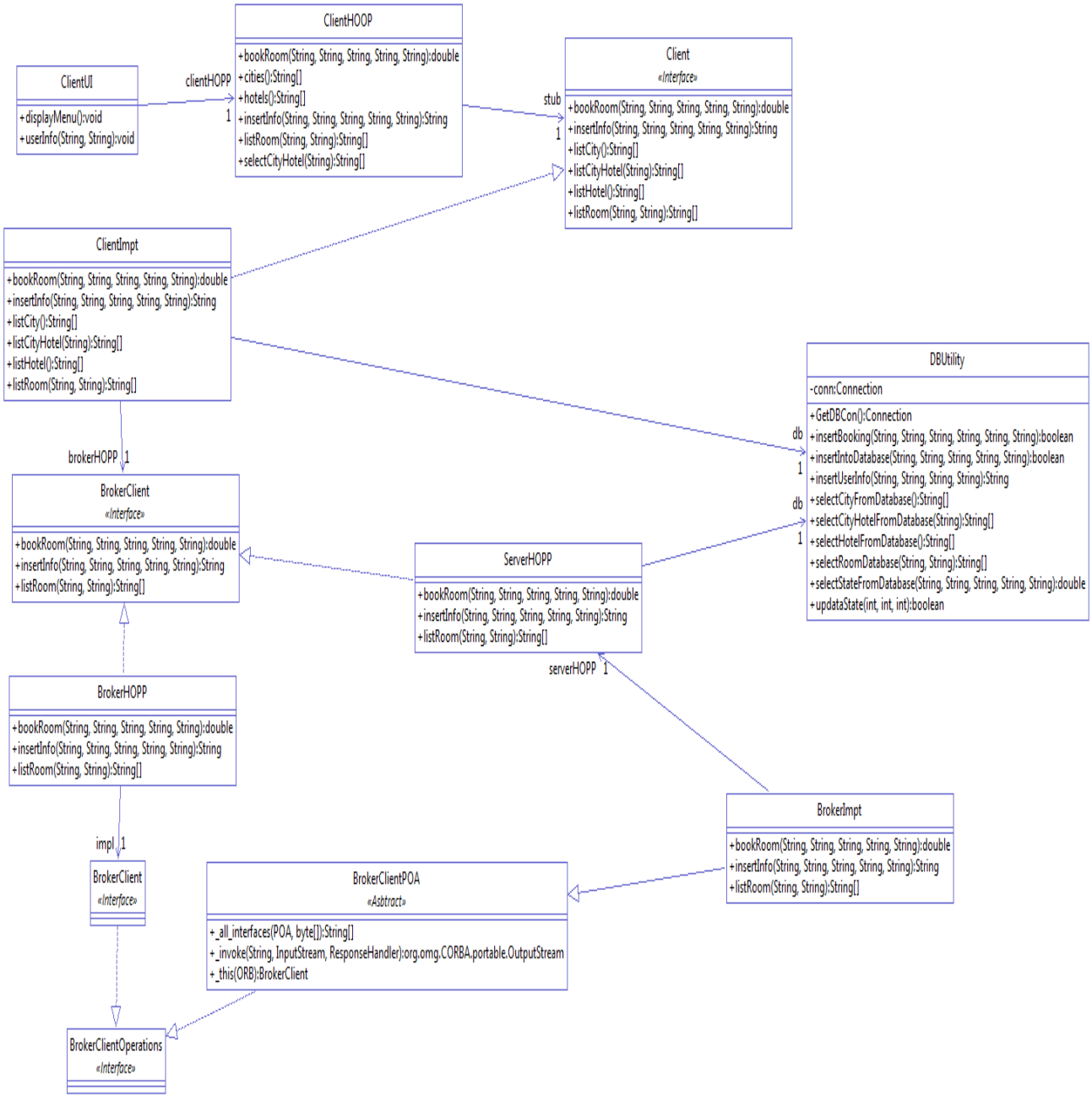
Message formats:

Request	Client	Broker	Broker Client	Server(RMI)	Server(Corba)	BrokerRES	ServerRES
List city	“1”	String city				String[]	
List hotel	“2”	String hotel				String[]	
Listcity hotle	“3”	String hotel				String[]	
List room	“4”	(String hotel, String city)	(String hotel, String city)	(String hotel, String city)	(String hotel, String city)	String[]	String[]
Book(not ava)	“5”	“city and hotel info”	“city and hotel info”	“city and hotel info”	“city and hotel info”	-1	-1
Book(ava)	“5”	“city and hotel info”	“city and hotel info”	“city and hotel info”	“city and hotel info”	**Double id**	**Double id**
quit	“0”						

UML class diagram of key components:



Communication between Broker-side and Hotel –
Side uses **RMI**



Communication between Broker-side and Hotel –
Side uses **COBAR**

Code explanation :(use two example two illustrate the interaction of the classes)

1. When **client** want to list city, the client class will call the function of the cities () then new a **ClientHOOP** instance to call the function of listcity() in class **ClientHOOP**. Then ClientHOOP use the **RMI** technology to send request to the **Broker**, **ClientImpt** will implements the **Client** interface, when the Broker accept the list city command then the ClientImpt will new a DBUtility instance to access the Database and return the message.

2. When client want to list room same step to send request to **Broker** by using **RMI**, the Broker could not respond directly to client but plays the role of client to send request to the hotel server through. There are 2 kinds of methods to implements list room one is **RMI** the other is **CORBA.**

RMI: The Broker-side will new a rim's half-object (i.g. **BrokerHOPP**) through the command line parameter (input "rmi") in the **Broker-side**, BrokerHOPP as the network object to communicate with the RMI-server. When the RMI-server receive the request, at the RMI-server-side will call the list room method in the **ServerHOOP** which implements the **BrokerClient** Interface. At last **ServerHOOP** will access the Database then return the message.

CORBAR: similar to rmi the Broker-side will new a corba's half-object (i.g.**BrokerHOPP**) through the command line parameter (input "corba") in the **Broker-side**, After the corba-server accept the list room command through some object (e.g. BrokerClientPOA, BrokerClientOperator and BrokerImpt). **ServerHOPP** will access the Database by creating the DBUtility instance.