User manual

1.Level

I have accomplished the distinct level.

This program is a three tiers system--client,broker,server having a complete separation of presentation from logical aspects which achieve the multiple clients talking to two servers named “ServerA” and “ServerB”. From the client to broker and broker to ServerB, I use the tech of RMI to achieve the remote connectivity and from the broker to ServerA, I use the tech of CORBA to achieve the remote connectivity. Each server in different cities has two hotels to be booked. It is implemented satisfactorily, provides the required functionality, is robust, exhibits the application of sound design principles and code conventions.

2.Guide

a) I use the MYSQL database named to store my background data. The database user name is fit5183a2 and the data are all stored in the database called “id26346966”.

```
C:\Users\lenovo\Desktop>mysql -u fit5183a2
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 393
Server version: 5.7.3-m13 MySQL Community Server (GPL)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| id26346966         |
| test               |
+--------------------+
3 rows in set (0.00 sec)
```

b) In “id26346966” database, there are 7 tables in it.

```
mysql> use id26346966;
Database changed
mysql> show tables;
+----------------------+
| Tables_in_id26346966 |
+----------------------+
| bj7days              |
| bjjjstar             |
| cityhotel            |
| roomcheck            |
| sh7days              |
| shjjstar             |
| user                 |
+----------------------+
7 rows in set (0.00 sec)
```

The tables named “bj7days”, “bjjjstar”, “sh7days”, “shjjstar” represent the four dependent hotels. The former two are in the same city and be controlled by same server and so does the latter two.

The “cityhotel” is used to present the cities and their corresponding hotels which can give the users a direct view on booking hotel.

```
mysql> select *from bj7days;
+---------------+--------+------+---------+
| hotelName     | roomID | rate | vacancy |
+---------------+--------+------+---------+
| beijing_7days | 100    | 300  | 0       |
| beijing_7days | 101    | 300  | 0       |
| beijing_7days | 102    | 300  | 0       |
| beijing_7days | 103    | 300  | 0       |
| beijing_7days | 104    | 300  | 0       |
| beijing_7days | 105    | 350  | 0       |
| beijing_7days | 106    | 350  | 0       |
| beijing_7days | 107    | 350  | 0       |
| beijing_7days | 108    | 350  | 0       |
| beijing_7days | 109    | 350  | 1       |
+---------------+--------+------+---------+
10 rows in set (0.03 sec)

mysql> select *from cityhotel;
+----+----------+--------+
| id | city     | hotel  |
+----+----------+--------+
| 1  | beijing  | 7days  |
| 2  | beijing  | jjstar |
| 3  | shanghai | 7days  |
| 4  | shanghai | jjstar |
+----+----------+--------+
4 rows in set (0.00 sec)
```

The “roomcheck” is used for recording the information of rooms which have been booked: roomID, checkInDay and checkOutDay.

Two things should be mentioned are the same room can be

booked repeatedly in different time periods and the value of "vacancy" are only "0" and "1". "0" represents a room no ordering until now and "1" represents at least one client has been booking in different time periods. That means the other users can also book the room with vacancy "1" just need the different time periods.

At last, the table "user" presents the users order information.

```
mysql> select *from roomcheck;
+--------+------------+-------------+
| roomID | checkInDay | checkOutDay |
+--------+------------+-------------+
| 109    | 2015-07-07 | 2015-07-09  |
| 309    | 2015-08-08 | 2015-08-09  |
| 109    | 2015-03-03 | 2015-03-04  |
+--------+------------+-------------+
3 rows in set (0.01 sec)

mysql> select *from user;
+----------------+-----------+------------+------------+---------+-------------+
| hotelName      | name      | inDay      | outDay     | creCard | phone       |
+----------------+-----------+------------+------------+---------+-------------+
| beijing_7days  | chen      | 2015-07-07 | 2015-07-09 | 123456  | 15150422448 |
| shanghai_7days | chentwo   | 2015-08-08 | 2015-08-09 | 123432  | 11111111111 |
| beijing_7days  | chenthree | 2015-03-03 | 2015-03-04 | 900090  | 22222222222 |
+----------------+-----------+------------+------------+---------+-------------+
3 rows in set (0.00 sec)
```

c) At first, the corba port should be opened like this.

```
Administrator: C:\Windows\system32\cmd.exe - tnameserv

C:\Users\fg\Desktop>tnameserv
Initial Naming Context:
IOR:000000000000002b49444c3a6f6d672e6f72672f436f734e616d696e672f4e616d696e67436f
6e746578744578743a312e300000000000010000000000000009a000102000000000e3137322e3136
2e3132302e313000038400000045afabcb0000000020000f424000000001000000000000000020000
0008526f6f74504f41000000000d544e616d6553657276696365000000000000000800000001000
000114000000000000002000000010000002000000000000100010000000205010001000100200001
01090000000100010100000000260000000020002
TransientNameServer: setting port for initial object references to: 900
Ready.
```

d) Export the java files with the runnable JAR file and then it can be used with the command in the dos system.

Use the command:

java -jar ServerA.jar

java -jar ServerB.jar

java -jar BrokerServer.jar
java -jar ClientUI.jar
Then the procedures are running.

```
C:\Users\lenovo\Desktop>java -jar ServerA.jar
Corba ServerA complete...ServerA to broker.

C:\Users\lenovo\Desktop>java -jar ServerB.jar
RMI Binding complete...ServerB to broker.

C:\Users\lenovo\Desktop>java -jar BrokerServer.jar
Corba broker complete...broker to ServerA.
RMI broker complete...broker to ServerB.
RMI broker complete...broker to client.

C:\Users\lenovo\Desktop>java -jar ClientUI.jar
RMI client complete...client to broker.

Entre Request: 1.BOOK  2. ORDERS  3.COMPARERATE  4.QUIT
```

If you are only run one of the ServerA and ServerB, you will get the different interface of the BrokerServer.

The result of ServerB not connected:

```
C:\Users\lenovo\Desktop>java -jar BrokerServer.jar
Corba broker complete...broker to ServerA.
Unable to connect to ServerB!
RMI broker complete...broker to client.
```

e) Then the client is running and four main functions shown on the window.

"BOOK" for booking a room.

"ORDERS" for searching the orders and for the users security you can only search the order with the right tel number which in your former order information.

"COMPARERATE" for comparing the rates from different hotels in the same city with the two orders asc and desc which can be selected.

"QUIT" for quit the procedure.

f) Choose the function two "ORDERS" and we can see the next step which allow us to provide the tel number. As long as you provide the right tel number, you can query the order information. In the following example, the tel number is provided in the former step.

```
Entre Request: 1.BOOK  2. ORDERS  3.COMPARERATE  4.QUIT
2
record Information:
input your tel number:

hotelName             name                  inDay                 outDay
        creCard               phone

Entre Request: 1.BOOK  2. ORDERS  3.COMPARERATE  4.QUIT
2
record Information:
input your tel number:
12312312
hotelName             name                  inDay                 outDay
        creCard               phone

Entre Request: 1.BOOK  2. ORDERS  3.COMPARERATE  4.QUIT
2
record Information:
input your tel number:
11111111111
hotelName             name                  inDay                 outDay
        creCard               phone
shanghai_7days        chentwo               2015-08-08            2015-08-09
        123432                11111111111

Entre Request: 1.BOOK  2. ORDERS  3.COMPARERATE  4.QUIT
```

g) Before we booking the room, we can have an overview of rate according to the function three “COMPARERATE”. We should firstly to select the city and know the hotels rate all in this city. The “asc” represents ascending order and “desc” represents descending order. This function can give us a price comparison and influence our hotel judgment.

```
Entre Request: 1.BOOK  2. ORDERS  3.COMPARERATE  4.QUIT
3
beijing                 shanghai
Please choose city : beijing
7days                   jjstar
compare rate! choose:  asc  desc
asc
hotelname             rate
beijing_7days         300
beijing_7days         350
beijing_jjstar        400
beijing_jjstar        450

Entre Request: 1.BOOK  2. ORDERS  3.COMPARERATE  4.QUIT
3
beijing                 shanghai
Please choose city : shanghai
7days                   jjstar
compare rate! choose:  asc  desc
desc
hotelname             rate
shanghai_jjstar       250
shanghai_jjstar       200
shanghai_7days        150
shanghai_7days        100
```

h) We can book now. Select the city, hotel and roomID. Then the procedure judge the checkInDay and checkOutDay which are available in three aspects:
   i. The checkOutDay is larger than checkInDay.
   ii. If the room have been booked(the value of “vacancy” is “1”), the procedure should be compare the existing inDay and outDay to ensure that the time periods are not overlap.
   iii. All the day format is yyyy-MM-dd, so the input string should be follow this convention otherwise should be input again.

```
Entre Request: 1.BOOK  2. ORDERS  3.COMPARERATE  4.QUIT
1
client booking:
beijing                 shanghai
Please choose city : beijing
7days                   jjstar
Please choose hotel : 7days
roomID                rate                  vacancy
100                   300                   0
101                   300                   0
102                   300                   0
103                   300                   0
104                   300                   0
105                   350                   0
106                   350                   0
107                   350                   0
108                   350                   0
109                   350                   1

please choose the room id: 100
please input your inDay: form yyyy-mm-dd 2015-12-12
please input your outDay: form yyyy-mm-dd 2015-12-09
please input your inDay: form yyyy-mm-dd 2015-12-12
please input your outDay: form yyyy-mm-dd 2015-12-15
please set you information:
please input your name: testOne
please input your card num: form [1-9]{1}[0-9]{5} 000000
wrong. input again
please input your card num: form [1-9]{1}[0-9]{5} 100000
please input your phone num: form [1-9]{1}[0-9]{10} 00000000000
wrong. input again
please input your phone num: form [1-9]{1}[0-9]{10} 10000000000
information has been saved.
```

```
Entre Request: 1.BOOK  2. ORDERS  3.COMPARERATE  4.QUIT
1
client booking:
beijing                 shanghai
Please choose city : beijing
7days                   jjstar
Please choose hotel : 7days
roomID                rate                  vacancy
100                   300                   1
101                   300                   0
102                   300                   0
103                   300                   0
104                   300                   0
105                   350                   0
106                   350                   0
107                   350                   0
108                   350                   0
109                   350                   1

please choose the room id: 100
please input your inDay: form yyyy-mm-dd 2015-12-10
please input your outDay: form yyyy-mm-dd 2015-12-17
this time periods has been booked.
wrong. input again
please input your inDay: form yyyy-mm-dd 2011-1-11
wrong. input again
please input your inDay: form yyyy-mm-dd 2016-01-01
please input your outDay: form yyyy-mm-dd 2016-01-13
please set you information:
please input your name: testTwo
please input your card num: form [1-9]{1}[0-9]{5} 190909
please input your phone num: form [1-9]{1}[0-9]{10} 12345678900
information has been saved.
```

```
C:\Users\lenovo\Desktop>java -jar ServerA.jar
Corba ServerA complete...ServerA to broker.
client choose the city: beijing, hotel: 7days.
client choose the city: beijing, hotel: 7days.
```

i) The ServerA serves the city “beijing” and the ServerB serves the city “shanghai”. Each city has two hotel called “7days” and “jjstar”. The inner function which contribute to select the city, hotel, get the information of room, order and rate will be implemented directly from the BrokerServer to MYSQL database. and the function which related to comfirm a roomID from different hotels in different cities should be implemented through BrokerServer to ServerA using the tech of “CORBA” or through BrokerServer to ServerB using the tech of “RMI”. So for example, if the ServerB is not start up, we cannot booking the hotel room in the “shanghai”.

```
Entre Request: 1.BOOK  2. ORDERS  3.COMPARERATE  4.QUIT
1
client booking:
beijing                 shanghai
Please choose city : shanghai
7days                   jjstar
Please choose hotel : jjstar
roomID                  rate                    vacancy
400                     200                     0
401                     200                     0
402                     200                     0
403                     200                     0
404                     200                     0
405                     250                     0
406                     250                     0
407                     250                     0
408                     250                     0
409                     250                     0

please choose the room id: 409
java.rmi.ServerException: RemoteException occurred in server thread; nested exce
ption is:
        java.rmi.ConnectException: Connection refused to host: 192.168.88.1; nes
ted exception is:
        java.net.ConnectException: Connection refused: connect
        at sun.rmi.server.UnicastServerRef.dispatch(Unknown Source)
        at sun.rmi.transport.Transport$1.run(Unknown Source)
        at sun.rmi.transport.Transport$1.run(Unknown Source)
        at java.security.AccessController.doPrivileged(Native Method)
        at sun.rmi.transport.Transport.serviceCall(Unknown Source)
        at sun.rmi.transport.tcp.TCPTransport.handleMessages(Unknown Source)
        at sun.rmi.transport.tcp.TCPTransport$ConnectionHandler.run0(Unknown Sou
rce)
        at sun.rmi.transport.tcp.TCPTransport$ConnectionHandler.run(Unknown Sour
ce)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source)
        at java.lang.Thread.run(Unknown Source)
        at sun.rmi.transport.StreamRemoteCall.exceptionReceivedFromServer(Unknow
n Source)
        at sun.rmi.transport.StreamRemoteCall.executeCall(Unknown Source)
        at sun.rmi.server.UnicastRef.invoke(Unknown Source)
        at java.rmi.server.RemoteObjectInvocationHandler.invokeRemoteMethod(Unkn
own Source)
        at java.rmi.server.RemoteObjectInvocationHandler.invoke(Unknown Source)
        at com.sun.proxy.$Proxy0.judgeRoomID(Unknown Source)
        at client.ClientImpl.setRoom(ClientImpl.java:131)
        at client.ClientUI.book(ClientUI.java:67)
        at client.ClientUI.loop(ClientUI.java:48)
        at client.ClientUI.main(ClientUI.java:28)
Caused by: java.rmi.ConnectException: Connection refused to host: 192.168.88.1;
nested exception is:
        java.net.ConnectException: Connection refused: connect
        at sun.rmi.transport.tcp.TCPEndpoint.newSocket(Unknown Source)
        at sun.rmi.transport.tcp.TCPChannel.createConnection(Unknown Source)
        at sun.rmi.transport.tcp.TCPChannel.newConnection(Unknown Source)
        at sun.rmi.server.UnicastRef.invoke(Unknown Source)
        at java.rmi.server.RemoteObjectInvocationHandler.invokeRemoteMethod(Unkn
own Source)
        at java.rmi.server.RemoteObjectInvocationHandler.invoke(Unknown Source)
        at com.sun.proxy.$Proxy2.judgeRoomID(Unknown Source)
        at broker.BrokerImpl.judgeRoomID(BrokerImpl.java:101)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
        at java.lang.reflect.Method.invoke(Unknown Source)
        at sun.rmi.server.UnicastServerRef.dispatch(Unknown Source)
        at sun.rmi.transport.Transport$1.run(Unknown Source)
        at sun.rmi.transport.Transport$1.run(Unknown Source)
        at java.security.AccessController.doPrivileged(Native Method)
        at sun.rmi.transport.Transport.serviceCall(Unknown Source)
```

```
        at sun.rmi.transport.tcp.TCPTransport.handleMessages(Unknown Source)
        at sun.rmi.transport.tcp.TCPTransport$ConnectionHandler.run0(Unknown Sou
rce)
        at sun.rmi.transport.tcp.TCPTransport$ConnectionHandler.run(Unknown Sour
ce)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source)
        at java.lang.Thread.run(Unknown Source)
Caused by: java.net.ConnectException: Connection refused: connect
        at java.net.DualStackPlainSocketImpl.connect0(Native Method)
        at java.net.DualStackPlainSocketImpl.socketConnect(Unknown Source)
        at java.net.AbstractPlainSocketImpl.doConnect(Unknown Source)
        at java.net.AbstractPlainSocketImpl.connectToAddress(Unknown Source)
        at java.net.AbstractPlainSocketImpl.connect(Unknown Source)
        at java.net.PlainSocketImpl.connect(Unknown Source)
        at java.net.SocksSocketImpl.connect(Unknown Source)
        at java.net.Socket.connect(Unknown Source)
        at java.net.Socket.connect(Unknown Source)
        at java.net.Socket.<init>(Unknown Source)
        at java.net.Socket.<init>(Unknown Source)
        at sun.rmi.transport.proxy.RMIDirectSocketFactory.createSocket(Unknown S
ource)
        at sun.rmi.transport.proxy.RMIMasterSocketFactory.createSocket(Unknown S
ource)
        ... 23 more
```

j) The function “quit” is to end the client process.

```
Entre Request: 1.BOOK  2. ORDERS  3.COMPARERATE  4.QUIT
4
quit...

C:\Users\lenovo\Desktop>_
```