User manual

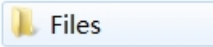
1.Level

I have accomplished the credit level.

This program is a 2-tier system which achieve the multiple clients talking to a single server. It is implemented satisfactorily, provides the required functionality, is robust, exhibits the application of sound design principles and code conventions.

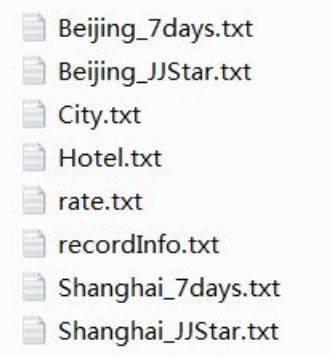
2.Guide

(1.) I use the text files to store my background data in a “Files” folder.



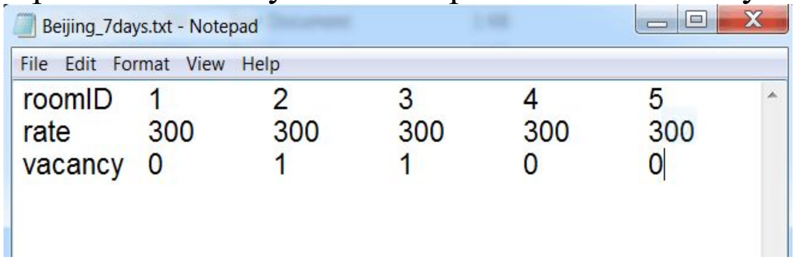
Files

In it, there are eight files as follow.



Beijing_7days.txt
Beijing_JJStar.txt
City.txt
Hotel.txt
rate.txt
recordInfo.txt
Shanghai_7days.txt
Shanghai_JJStar.txt

To ensure that each text file has its own function. For example: the Beijing_7days represents the information of the hotel named “7days” in Beijing. The text “Beijing_JJStar”, “Shanghai_7days”, “Shanghai_JJstar” are similar to the function of “Beijing_7days”. The data 0 or 1 behind the data vacancy represents the room which has been booked or not. “0” represents vacancy and “1” represents not vacancy.

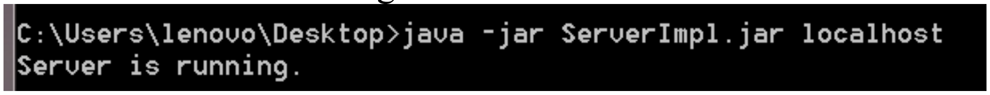


```
roomID  1    2    3    4    5
rate    300  300  300  300  300
vacancy 0    1    1    0    0
```

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

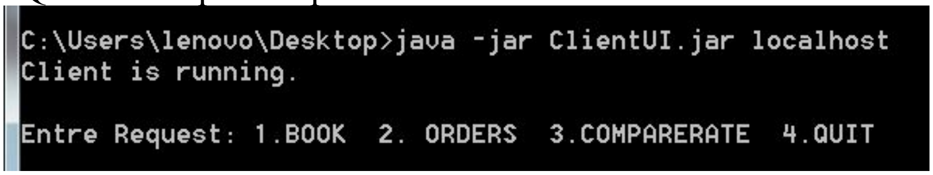
Use the command: *java -jar ServerImpl.jar localhost*

Then the server is running.



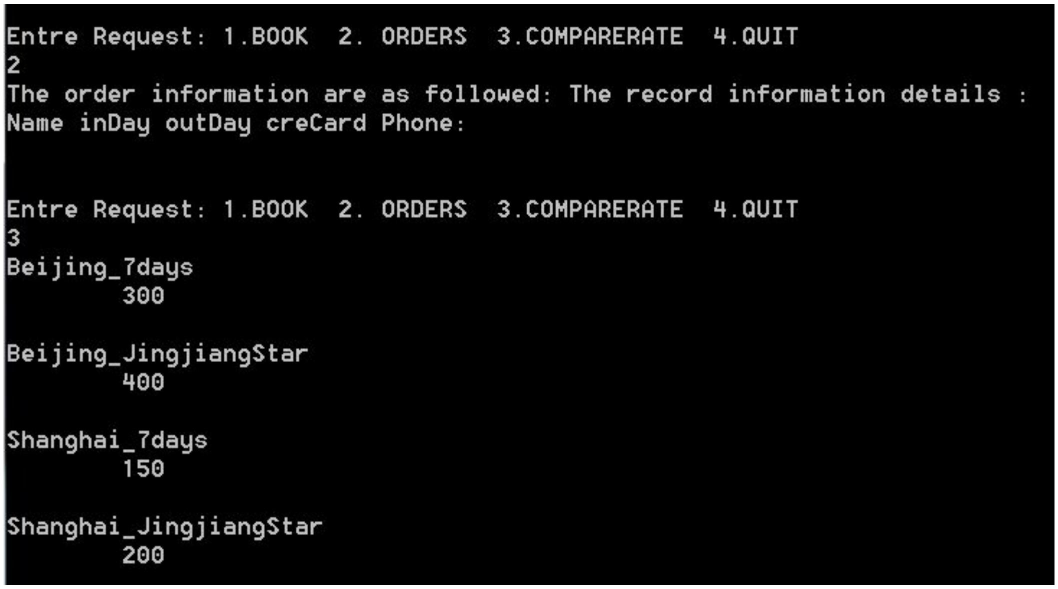
```
C:\Users\lenovo\Desktop>java -jar ServerImpl.jar localhost
Server is running.
```

(3.) Like the former operation, use the command: *java -jar ServerImpl.jar localhost*
Then the client is running and four main functions shown on the window.
"BOOK" for booking a room, "ORDER" for searching the orders,
"COMPARERATE" for comparing the rates from different hotels and
"QUIT" for quit the procedure.



```
C:\Users\lenovo\Desktop>java -jar ClientUI.jar localhost
Client is running.

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

(4.) Choose the function two and we can see there is no orders in it before no one book the room.
We can compare the rate before booking room because we can have a overview of rates of different hotels.

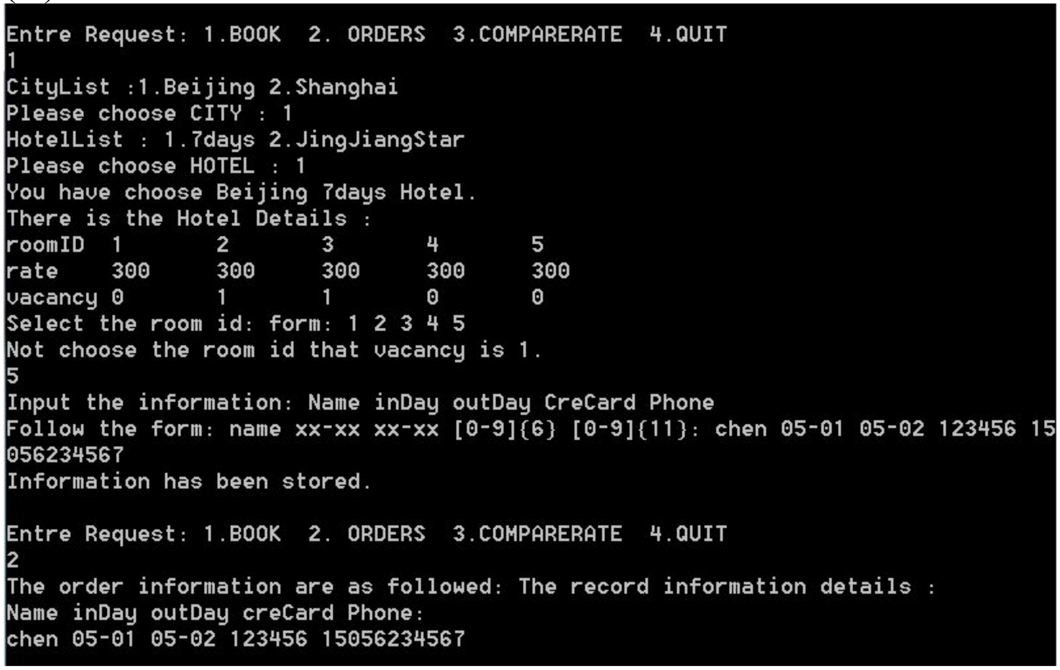


```
Entre Request: 1.BOOK  2. ORDERS  3.COMPARERATE  4.QUIT
2
The order information are as followed: The record information details :
Name inDay outDay creCard Phone:


Entre Request: 1.BOOK  2. ORDERS  3.COMPARERATE  4.QUIT
3
Beijing_7days
        300

Beijing_JingjiangStar
        400

Shanghai_7days
        150

Shanghai_JingjiangStar
        200
```

(5.) We can book the room now. Choose the function one.
First shown is the CityList, choose by yourself.
Second shown is the HotelList, choose by yourself.
Then the system returns your result and reveal the hotel details which you selected.
Then we meet the data I said above. If you choose the id that not 1 or 2 or 3 or 4 or 5, the system will recognized the error and enables you to input again.
Please do not choose the room which the value of "vacancy" is "1".
If you book the room which id is 5, then the system will set this vacancy to "1" when next client coming into this hotel will meet the changed data.

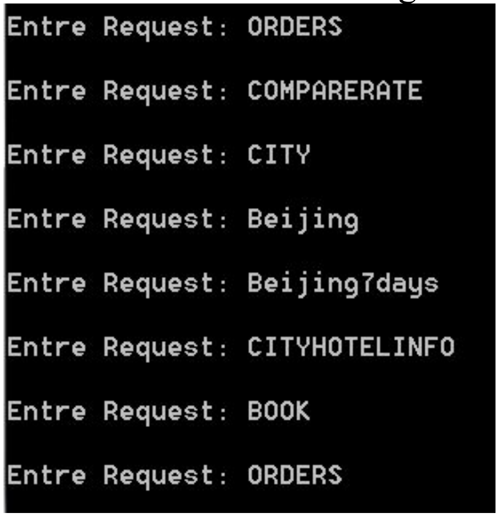
Next, you will input your personal information as the format seen in the windows -- with the delimiter “space” and the “Enter” in the last.
Then the information has been stored.

(6.) Now we can see the order in the function 2.



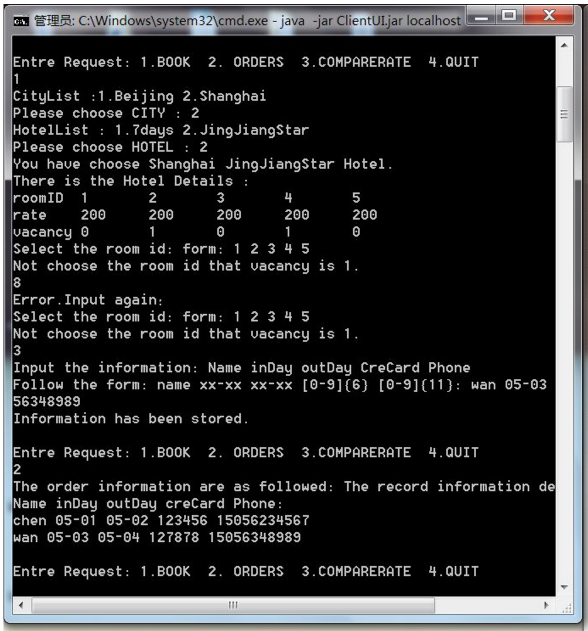
```
Entre Request: 1.BOOK  2. ORDERS  3.COMPARERATE  4.QUIT
1
CityList :1.Beijing 2.Shanghai
Please choose CITY : 1
HotelList : 1.7days 2.JingJiangStar
Please choose HOTEL : 1
You have choose Beijing 7days Hotel.
There is the Hotel Details :
roomID  1       2       3       4       5
rate    300     300     300     300     300
vacancy 0       1       1       0       0
Select the room id: form: 1 2 3 4 5
Not choose the room id that vacancy is 1.
5
Input the information: Name inDay outDay CreCard Phone
Follow the form: name xx-xx xx-xx [0-9]{6} [0-9]{11}: chen 05-01 05-02 123456 15
056234567
Information has been stored.

Entre Request: 1.BOOK  2. ORDERS  3.COMPARERATE  4.QUIT
2
The order information are as followed: The record information details :
Name inDay outDay creCard Phone:
chen 05-01 05-02 123456 15056234567
```

(7.) Go through this command in the client window, we can also see the server window. It can recognize all the client’s operations.

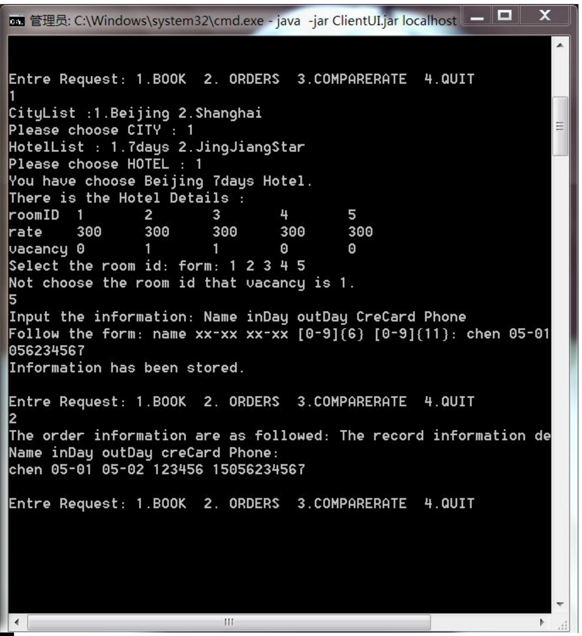


```
Entre Request: ORDERS

Entre Request: COMPARERATE

Entre Request: CITY

Entre Request: Beijing

Entre Request: Beijing7days

Entre Request: CITYHOTELINFO

Entre Request: BOOK

Entre Request: ORDERS
```

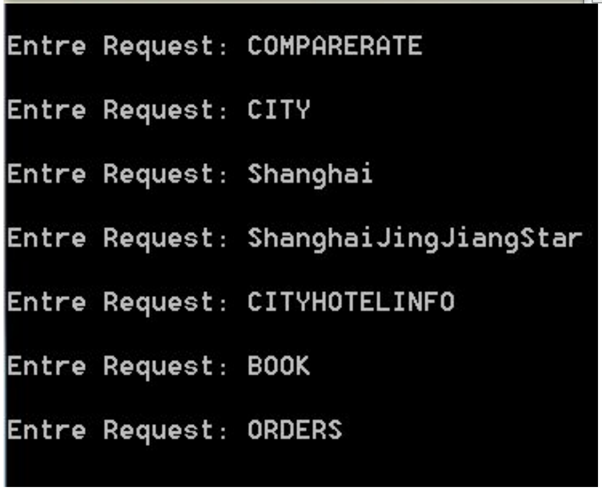
(8.) Because of the multiple thread, the project can be run with lots of clients in the same time.


管理员: C:\Windows\system32\cmd.exe - java -jar ClientUI.jar localhost

```
Entre Request: 1.BOOK  2. ORDERS  3.COMPARERATE  4.QUIT
1
CityList :1.Beijing 2.Shanghai
Please choose CITY : 2
HotelList : 1.7days 2.JingJiangStar
Please choose HOTEL : 2
You have choose Shanghai JingJiangStar Hotel.
There is the Hotel Details :
roomID  1       2       3       4       5
rate    200     200     200     200     200
vacancy 0       1       0       1       0
Select the room id: form: 1 2 3 4 5
Not choose the room id that vacancy is 1.
8
Error.Input again:
Select the room id: form: 1 2 3 4 5
Not choose the room id that vacancy is 1.
3
Input the information: Name inDay outDay CreCard Phone
Follow the form: name xx-xx xx-xx [0-9]{6} [0-9]{11}: wan 05-03
56348989
Information has been stored.

Entre Request: 1.BOOK  2. ORDERS  3.COMPARERATE  4.QUIT
2
The order information are as followed: The record information de
Name inDay outDay creCard Phone:
chen 05-01 05-02 123456 15056234567
wan 05-03 05-04 127878 15056348989

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

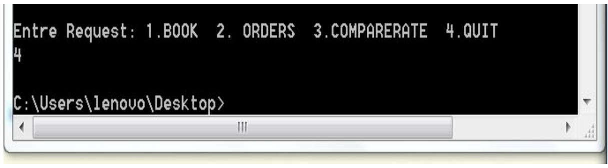

管理员: C:\Windows\system32\cmd.exe - java -jar ClientUI.jar localhost

```
Entre Request: 1.BOOK  2. ORDERS  3.COMPARERATE  4.QUIT
1
CityList :1.Beijing 2.Shanghai
Please choose CITY : 1
HotelList : 1.7days 2.JingJiangStar
Please choose HOTEL : 1
You have choose Beijing 7days Hotel.
There is the Hotel Details :
roomID  1       2       3       4       5
rate    300     300     300     300     300
vacancy 0       1       1       0       0
Select the room id: form: 1 2 3 4 5
Not choose the room id that vacancy is 1.
5
Input the information: Name inDay outDay CreCard Phone
Follow the form: name xx-xx xx-xx [0-9]{6} [0-9]{11}: chen 05-01
056234567
Information has been stored.

Entre Request: 1.BOOK  2. ORDERS  3.COMPARERATE  4.QUIT
2
The order information are as followed: The record information de
Name inDay outDay creCard Phone:
chen 05-01 05-02 123456 15056234567

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

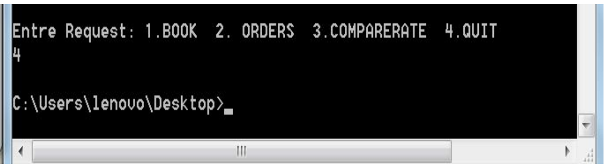


```
Entre Request: COMPARERATE

Entre Request: CITY

Entre Request: Shanghai

Entre Request: ShanghaiJingJiangStar

Entre Request: CITYHOTELINFO

Entre Request: BOOK

Entre Request: ORDERS
```

(9) Quit the program you can use the function 4 and the clients windows are shown as follow.

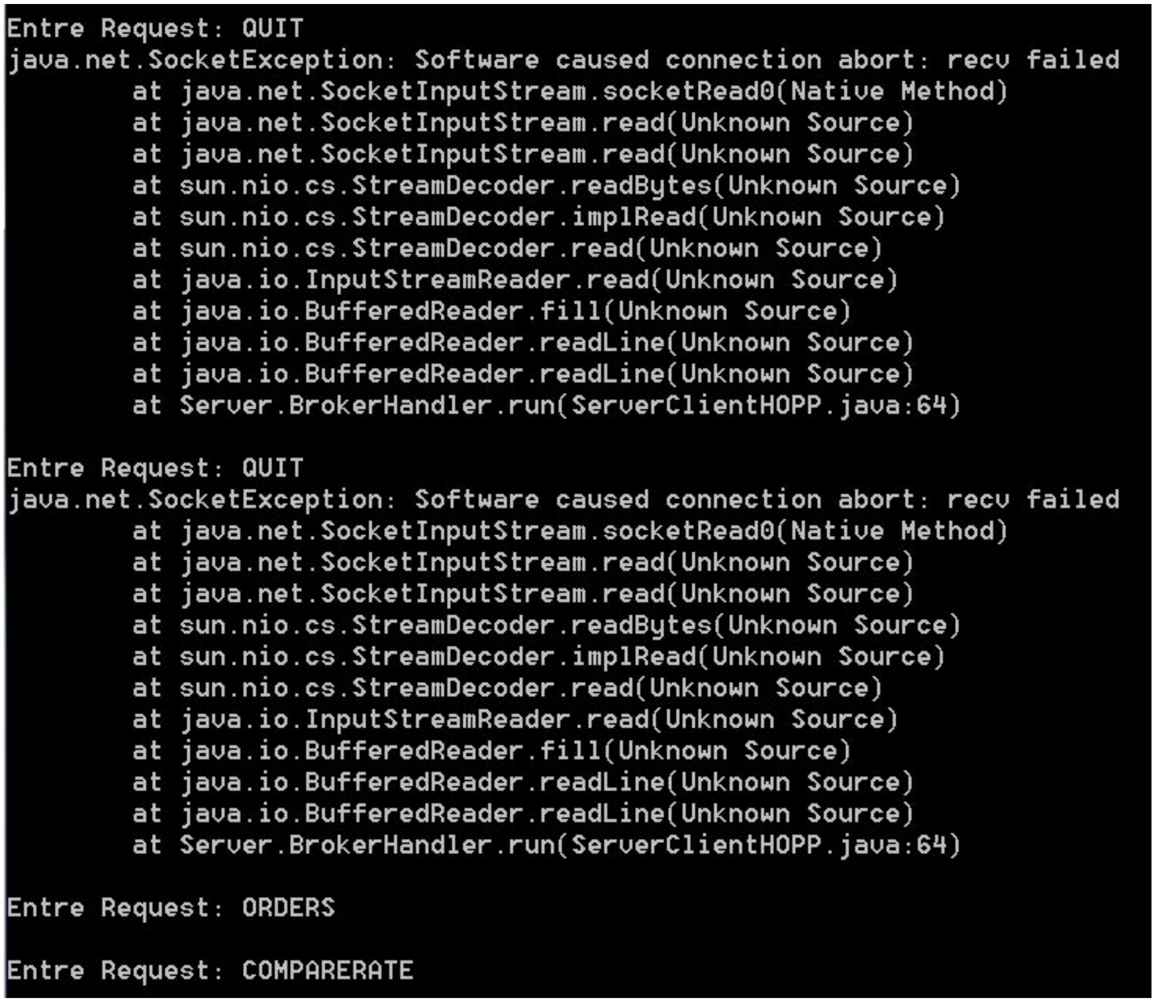


```
Entre Request: 1.BOOK  2. ORDERS  3.COMPARERATE  4.QUIT
4

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



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

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

(10.) Some wrong information will displayed on the server window but it has no effects on the following operation.



```
Entre Request: QUIT
java.net.SocketException: Software caused connection abort: recv failed
        at java.net.SocketInputStream.socketRead0(Native Method)
        at java.net.SocketInputStream.read(Unknown Source)
        at java.net.SocketInputStream.read(Unknown Source)
        at sun.nio.cs.StreamDecoder.readBytes(Unknown Source)
        at sun.nio.cs.StreamDecoder.implRead(Unknown Source)
        at sun.nio.cs.StreamDecoder.read(Unknown Source)
        at java.io.InputStreamReader.read(Unknown Source)
        at java.io.BufferedReader.fill(Unknown Source)
        at java.io.BufferedReader.readLine(Unknown Source)
        at java.io.BufferedReader.readLine(Unknown Source)
        at Server.BrokerHandler.run(ServerClientHOPP.java:64)

Entre Request: QUIT
java.net.SocketException: Software caused connection abort: recv failed
        at java.net.SocketInputStream.socketRead0(Native Method)
        at java.net.SocketInputStream.read(Unknown Source)
        at java.net.SocketInputStream.read(Unknown Source)
        at sun.nio.cs.StreamDecoder.readBytes(Unknown Source)
        at sun.nio.cs.StreamDecoder.implRead(Unknown Source)
        at sun.nio.cs.StreamDecoder.read(Unknown Source)
        at java.io.InputStreamReader.read(Unknown Source)
        at java.io.BufferedReader.fill(Unknown Source)
        at java.io.BufferedReader.readLine(Unknown Source)
        at java.io.BufferedReader.readLine(Unknown Source)
        at Server.BrokerHandler.run(ServerClientHOPP.java:64)

Entre Request: ORDERS

Entre Request: COMPARERATE
```