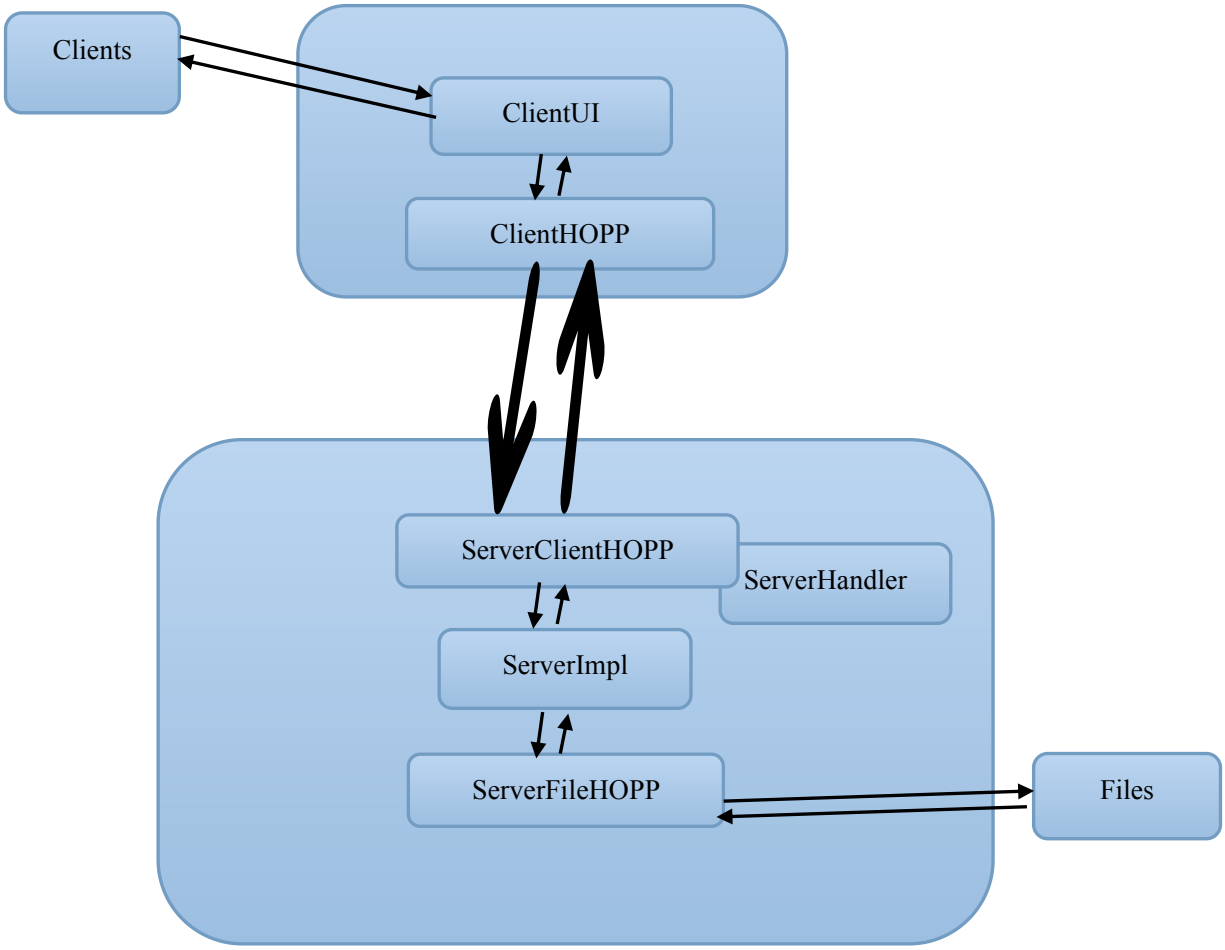
Hotel Booking Design Document

1. Architecture

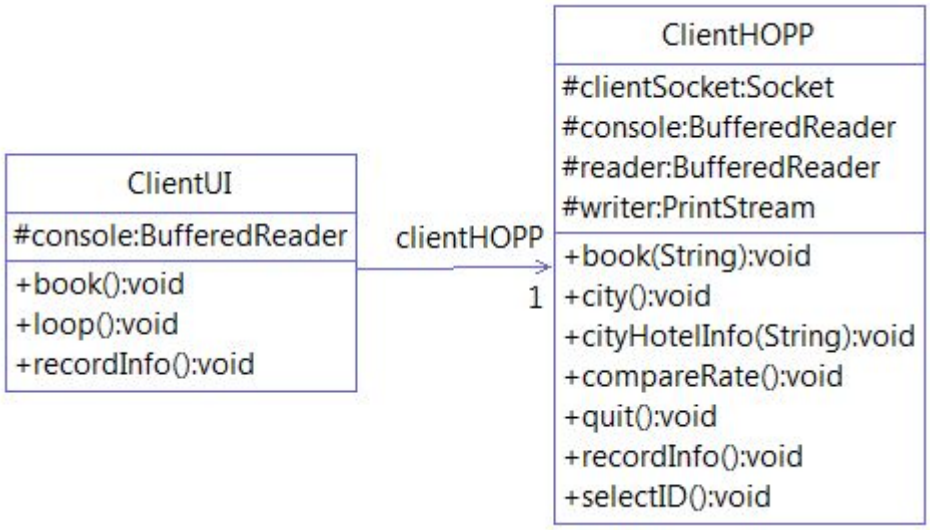


Introduction: This Hotel Booking is a typical multiple clients talking to a single server system. My system can be capable of handling at least two clients concurrently, the server handles all of the Hotel Booking requests. It is implemented satisfactorily, provides the required functionality, is robust, exhibits the application of sound design principles and code conventions. The server maintains information on the cities, hotels, their rates and bookings for each hotel. All of this can be done using local text and data files kept in memory. The main functions are booking room, comparing rate, checking available and searching order. The main body has 3 major components: client, server and text files. When users set up a client, they will input request through the client. Then the client will send responding request to server, the server will create a new thread. The server will analyze the message and query the files and return the result.

2. UML Class diagram

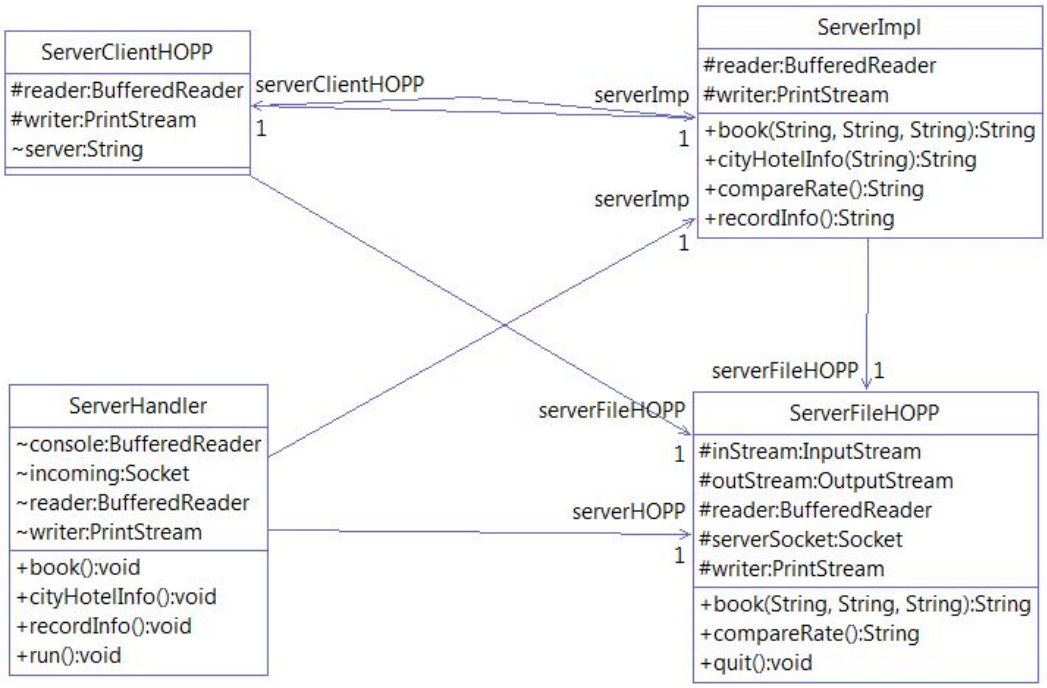
(1.) Client

The UML class diagrams:



Client has the responsibility to receive the requirements from the users and send them to server, then client will display results to users. There are two Class in the package Client: ClientUI and ClientHOPP. The major functions connecting with server and doing operations are declared in the ClientHOPP including: city, getCity, getHotel, getLocation, recordInfo, cityHotelInfo, selectID, book, compareRate, 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 ClientHOPP.

(2.) Server
The UML class diagrams:



Server has the responsibility to receive messages from client and do some handling work to send the message to do jobs on the files. When server gets result from responding files, it will do some handling work again and return the message to the client.

There are three classes in the server package: ServerImpl, ServerClientHOPP and ServerFileHOPP. The ServerClientHOPP declares an inner class ServerHandler who extends class Thread, at the same time, it is used to set up connection with client and receive messages from client. It will call the functions of class ServerImpl. Class ServerImpl's responsibility is connecting with class ServerFileHOPP, then ServerImpl will call the functions of ServerFileHOPP to do some work on the local files. The class ServerFileHOPP does all the jobs on the local files. The class including functions are as follows: recordInfo, cityHotelInfo, book, quit, getCity, getHotel, getLocation, compareRate.

(3.) Constants

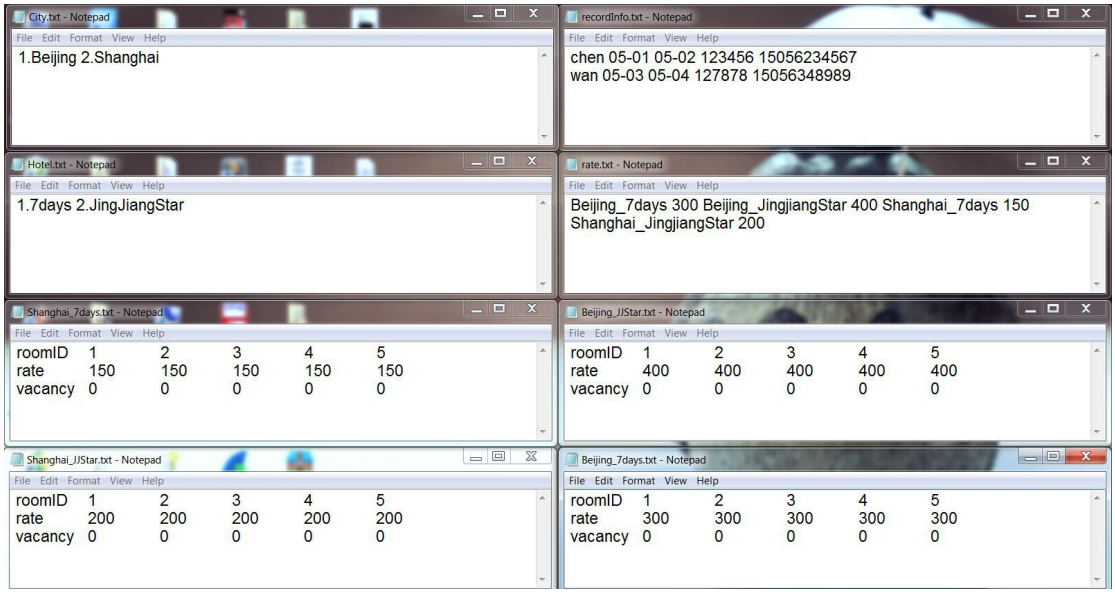
Create constants has lots benefits and convenient in replacing, searching and understanding. Some information can be stored and preprocessed. The constants I have used in the program as followed.

```
public class Constants {
    public static final int PORT = 8888;
    public static final String BOOK = "BOOK";
    public static final String RECORDINFO = "ORDERS";
    public static final String COMPARE = "COMPARERATE";
    public static final String CITYHOTELINFO = "CITYHOTELINFO";
```

```
    public static final String INPUTAGAIN = "Input again: ";
    public static final String DELIMITER = ":";
    public static final String CITY = "CITY";
    public static final String QUIT = "QUIT";
    public static final String CR_LF = "\r\n";
    public static final String ERROR = "Error.";
    public static final String SUCCEED = "Succeed.";

    public static final String SEVENDAYS = "7days";
    public static final String JJSTAR = "JingJiangStar";
    public static final String BEIJING = "Beijing";
    public static final String SHANGHAI = "Shanghai";

    public class Files {
        public static final String FILECITY = "Files/City.txt";
        public static final String FILEHOTEL = "Files/Hotel.txt";
        public static final String RECORDINFO = "Files/recordInfo.txt";
        public static final String RATE = "Files/rate.txt";

        public static final String BEIJING_SEVENDAYS =
"Files/Beijing_7days.txt";
        public static final String BEIJING_JINGJIANGSTAR =
"Files/Beijing_JJStar.txt";
        public static final String SHANGHAI_SEVENDAYS =
"Files/Shanghai_7days.txt";
        public static final String SHANGHAI_JINGJIANGSTAR =
"Files/Shanghai_JJStar.txt";
    }
}
```

3. Design of TEXT Files

I use various IO stream to achieve the text files succeed including BufferedReader, PrintWrite, FileWrite, FileReader. All the data should be read by line from the text files and then be processed in the Server port.



City.txt - Notepad

File Edit Format View Help

1.Beijing 2.Shanghai

recordInfo.txt - Notepad

File Edit Format View Help

chen 05-01 05-02 123456 15056234567
wan 05-03 05-04 127878 15056348989

Hotel.txt - Notepad

File Edit Format View Help

1.7days 2.JingJiangStar

rate.txt - Notepad

File Edit Format View Help

Beijing_7days 300 Beijing_JingjiangStar 400 Shanghai_7days 150
Shanghai_JingjiangStar 200

Shanghai_7days.txt - Notepad

File Edit Format View Help

roomID	1	2	3	4	5
rate	150	150	150	150	150
vacancy	0	0	0	0	0

Beijing_JJStar.txt - Notepad

File Edit Format View Help

roomID	1	2	3	4	5
rate	400	400	400	400	400
vacancy	0	0	0	0	0

Shanghai_JJStar.txt - Notepad

File Edit Format View Help

roomID	1	2	3	4	5
rate	200	200	200	200	200
vacancy	0	0	0	0	0

Beijing_7days.txt - Notepad

File Edit Format View Help

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