Practice Guidance of

Computer Network Programming

Software College of NEU

February 2017

Table of Contents

Table of Contents 2
Requirement 3
Overall Introduction 4
(一) Tasks 4
(二) Regulations 4
Project 1 Implement Network Communication Using FTP (8 hours) 6
A. Purpose 6
B. Content 6
C. Previewing 6
D. Basic steps 9
E. Submission 11
F. Critical thinking 11
Project 2 Implement Network Communication Using HTTP (8 hours) 12
A. Purpose 12
B. Content 12
C. Previewing 12
D. Basic Steps 13
E. Submission 16
F. Critical thinking 16
Project 3 Implement Chat Room Using Web Socket (12 hours) 17
A. Purpose 17
B. Content 17
C. Previewing 17
D. Basic Steps 19
E. Submission 20
F. Critical Thinking 20
Project 4 Distributed Agenda Service Based on RMI (Not Mandatory) 21
A. Purpose 21
B. Content 21
C. Previewing 22
D. Basic Steps 22
E. Submission 25
F. Critical Thinking 25

Requirement

Purpose of Computer Network Programming is letting students to enhance their understanding of how computer network operate. There are other materials that require student to master, such as architecture of Computer Network, TCP/IP, common command of computer network, using these command to debug CN. This lab would like students to get familiar with java net, multi-threads, method of Object-oriented and coding principles based on developing environment of Eclipse. Students should utilize their knowledge of computer network and Java in order to design their own program. We aim to cultivate students' ability to integrating theory with practice.

In this course, all student should manage to：

（1）Preview this guidance. Prepare for the content of this practice. Anticipate possible situation that may occur. Think and analyze these situations.

（2）Obey principle of lab room and teaching assistance in lab room. Take good care of facilities in lab room.

（3）Students shouldn't be late. Absence with no reason is not allowed. We require prove if there is really something emergency happened.

（4）Carefully debugging any problem that may occur, recording them, and give essential illustration and analysis.

（5）Write report, we attached template of lab report.

Checking will be divided into three part: ordinary attendance in lab including debugging and questioning to teachers, final presentation and practice reports.

Overall Introduction

(一) Tasks

This lab has three parts:

（1） Implement Network Communication using HTTP
（2） Implement Network Communication using FTP
（3） Implement a Chat Room System using Socket
（4） Distributed service for Agenda based on RMI (Not mandatory)

These three assignments is to letting students to enhance their understanding of how computer network operate. There are other materials that require student to master, such as architecture of Computer Network, TCP/IP, common command of computer network, using these command to debug CN. This lab would like students to get familiar with java net, multi-threads, method of Object-oriented and coding principles based on developing environment of Eclipse. Students should utilize their knowledge of computer network and Java in order to design their own program. We aim to cultivate students' ability to integrating theory with practice.

Attention：Requiring all students to submit intact source code, to submit their lab report and to attend final presentation.

(二) Regulations

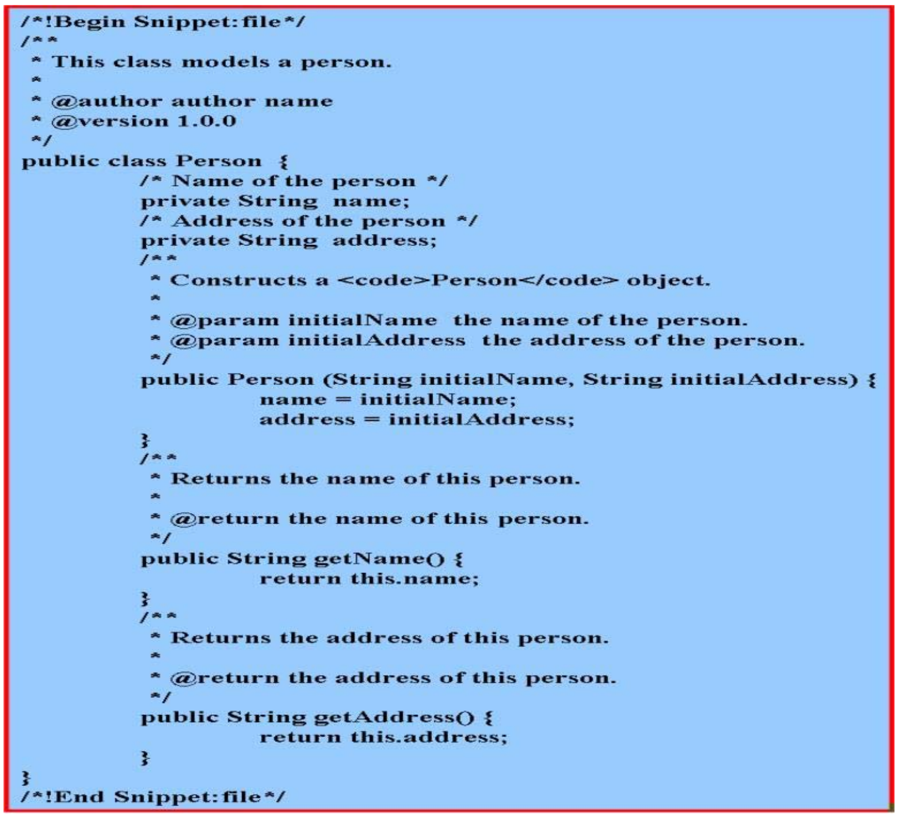
（1） All program should be save on a folder with folder's name "students' name＋last four bits of student numbers". All documents related to programs and all source codes should be put in that folder.

（2） Using favorite designing method, using object-oriented method to design and implement the system. Using knowledge, such as software engineering and data structure, to implement the system.

（3） Name of variables should be meaningful, for example：

```
private double salary;
```

（4） Codes should accord with Java coding principle, and codes should include appropriate commend. For example：



```
/*!Begin Snippet:file*/
/**
 * This class models a person.
 *
 * @author author name
 * @version 1.0.0
 */
public class Person  {
            /* Name of the person */
            private String  name;
            /* Address of the person */
            private String  address;
            /**
             * Constructs a <code>Person</code> object.
             *
             * @param initialName  the name of the person.
             * @param initialAddress  the address of the person.
             */
            public Person (String initialName, String initialAddress) {
                        name = initialName;
                        address = initialAddress;
            }
            /**
             * Returns the name of this person.
             *
             * @return the name of this person.
             */
            public String getName() {
                        return this.name;
            }
            /**
             * Returns the address of this person.
             *
             * @return the address of this person.
             */
            public String getAddress() {
                        return this.address;
            }
}
/*!End Snippet:file*/
```

（5） Complete content of lab according to credits and submit answers in time.

Project 1 Implement Network Communication Using FTP (8 hours)

A. Purpose

（1）Understanding principles of communication；

（2）Learning and using Socket and multi-threads to implement simple FTP server；

（3）Learning and using Socket to implement simple FTP client；

B. Content

We require students to complete following functions（Using PASV Mode）：

Name	Description	Priority
Getting list of files	Giving directory of users, getting files list, including file name, file size and creating data&time.	high
Uploading files	Uploading files to directory that user customize.	high
	Uploading folders to directory that user customize.	low
	Supporting resuming from break point.	low
Downloading files	Downloading files to directory that user customize.	high
	Uploading folders to directory that user customize.	low
	Supporting resuming from break point.	low

C. Previewing

（1）Work flow and commands of FTP

FTP uses 2 ports: one for data transition and one for command. These two ports usually can be 21(command) and 20(data).

1> Command Port

Generally, client have a socket in order to connect to FTP server. It is responsible for receive and send message. Some operations, such as login, changing directory, deleting files, etc, can be done through this connection.

2> Data Port

Operations that related to data, such as show file list, uploading, downloading files, etc, need another socket to manage.

If we use passive mode, normally server will send back a port number. Client need to open another socket to connect to this server port, and data will transmit through this new connection.

If we use active mode, normally client will send a port number to server, and client will listen to this port. Server will actively connect to this client port, and transmit data.

3> Active Mode (PORT)

In active mode, client randomly open a port with its number bigger than 1024, and we call this port as port N. Port N will connect to command port 21 of server. Client will also open another port which is port N+1, and this port N+1 will send command “port N+1” to the server. Then, server will actively connect its port 20 with clients’ port N+1.

FTP client just tell server its own port number, and server will try to connect to it. If clients have fire walls, because connection established outside, connections may be blocked.

4> Passive Mode (PASV)

We got another way to connect to FTP server which is using passive mode so that we can tackle problems that may rise due to fire walls. Both command and data connections will be initiated by clients, and there is no problem with fire walls. In passive mode, when we open a new FTP connection, client will create two new random local ports which is N & N+1 and N is bigger than 1024.

First port(N) will connect to 21 port of server, submitting PASV command. Then, server will establish a random port bigger than 1024, return something like “227 entering passive mode (127,0,0,1,4,18)”. It returns a message started with 227. In brackets, we have six numbers, first four is IP address of the server, and last two represent port number. Last second multiply to 256 and plus last one will generate the port number which belongs to server data port. If we get 227 entering passive mode (h1,h2,h3,h4,p1,p2), then port will be p1*256+p2, and ip address will be h1.h2.h3.h4. This means that a port is open at the server side. After a client receive this number, it will connect to data port of server using this number and transmit data.

5> Common FTP commands

Every commands of FTP consists of 3 or 4 letters, followed by parameters, splitting by space. All command end with “\r\n”.

In order to download or upload a file, firstly we need to log into FTP server. Until log out, we have to use USER, PASS, SIZE, CWD, RETR, PASV, PORT, REST and QUIT.

USER: Dictate user name. Usually, it is the first command after connecting.“USER gaoleyi\r\n”means user gaoleyi log in.

PASS: Dictate user password. This command follows USER.“PASS gaoleyi\r\n”： password is gaoleyi.

SIZE: Return a file size on the server side. “SIZE file.txt\r\n”： If file.txt exists, then return the size of it.

CWD: Change directory. For example：“CWD dirname\r\n”.

PASV: Enable server to listen at the server side, entering passive mode. E.g： “PASV\r\n”。

PORT: Tell server the listening port number of client, entering active mode. E.g: “PORT h1,h2,h3,h4,p1,p2”.

RETR: Downloading files.“RETR file.txt \r\n”： Downloading file.txt。

STOR: Uploading files.“STOR file.txt\r\n”： Uploading file.txt。

REST: This command doesn’t transmit file. Instead, it will omit specified point. This command should be followed by other commands that dictate file transmitting. “REST 100\r\n”： Specifies that the file transfer offset is 100 bytes.

QUIT: Close the connection.

6> FTP Response Code

After sending FTP command, servers should give responses.

Response Code consists of three digits:

First digit: Common command state, such as success, failure or incomplete.

Second digit: Classification of commands. E.g, 2 means commands relating to connection, and 3 means authentication.

Third digit: More details about the message. E.g:

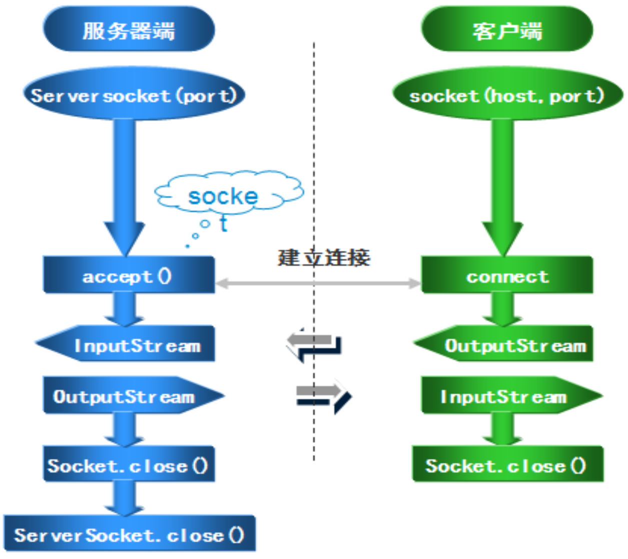
227 entering passive mode (127,0,0,1,4,18)

230 Logged on

250 CWD successful

（2）Socket Programming

• 基于TCP的Socket编程



• 利用TCP协议进行通信
两个应用程序，有主从之分
一个称为服务器程(Server)，
另外一个称为客户机程(Client)

• 交互过程

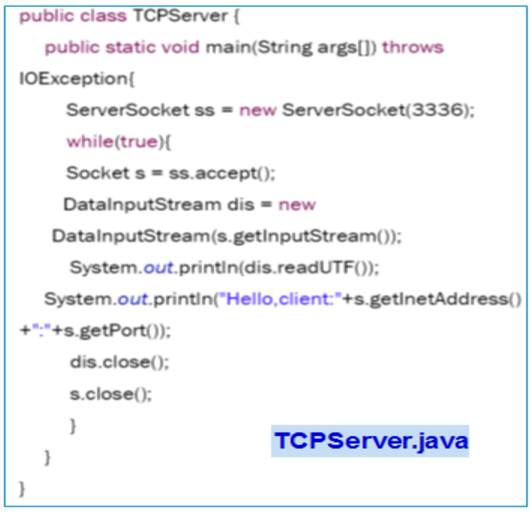
(1) 服务器程序创建一个ServerSocket,然后调用accept方法等待客户来连接

(2) 客户端程序创建一个socket并请求与服务器建立连接

(3) 刚才建立了连接的两个socket在一个单独的线程上对话

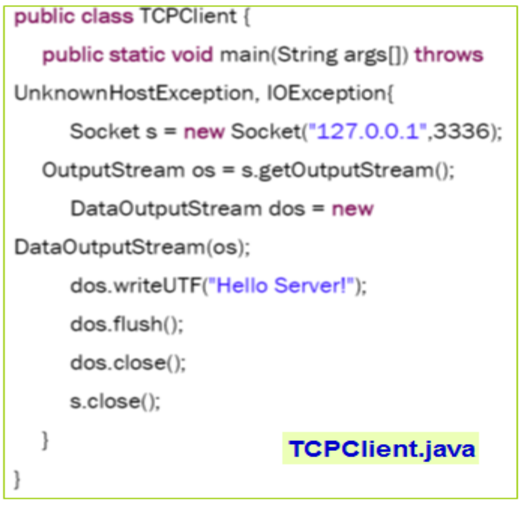
(4) 服务器开始等待新的连接请求

Typical Socket communication code:



```
public class TCPServer {
    public static void main(String args[]) throws
IOException{
        ServerSocket ss = new ServerSocket(3336);
        while(true){
        Socket s = ss.accept();
        DataInputStream dis = new
      DataInputStream(s.getInputStream());
        System.out.println(dis.readUTF());
    System.out.println("Hello,client:"+s.getInetAddress()
+":"+s.getPort());
        dis.close();
        s.close();
        }
    }
}
```

TCPServer.java

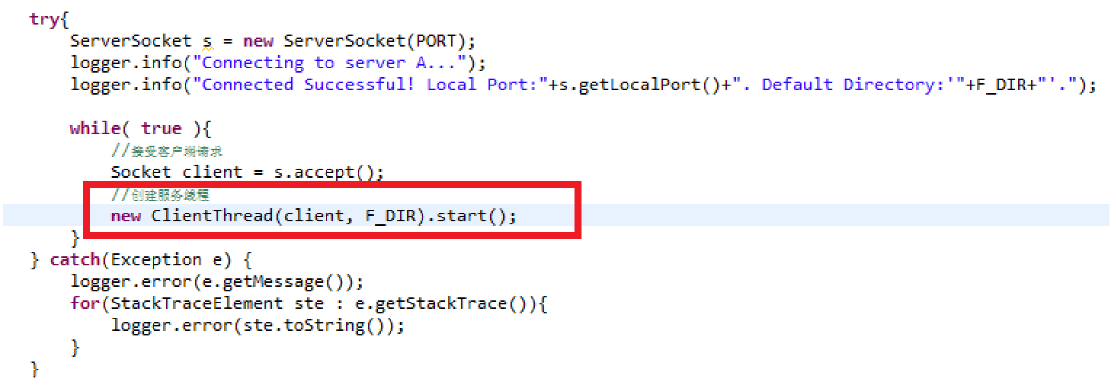


```
public class TCPClient {
    public static void main(String args[]) throws
UnknownHostException, IOException{
        Socket s = new Socket("127.0.0.1",3336);
    OutputStream os = s.getOutputStream();
        DataOutputStream dos = new
DataOutputStream(os);
        dos.writeUTF("Hello Server!");
        dos.flush();
        dos.close();
        s.close();
    }
}
```

TCPClient.java

（3）Multi-threads Programming

To make sure FTP server can tackle more than one request from clients at the same time, the server will allocate a thread for each client. For example:



```
try{
    ServerSocket s = new ServerSocket(PORT);
    logger.info("Connecting to server A...");
    logger.info("Connected Successful! Local Port:"+s.getLocalPort()+". Default Directory:'"+F_DIR+"'.");

    while( true ){
        //接受客户端请求
        Socket client = s.accept();
        //创建服务线程
        new ClientThread(client, F_DIR).start();
    }
} catch(Exception e) {
    logger.error(e.getMessage());
    for(StackTraceElement ste : e.getStackTrace()){
        logger.error(ste.toString());
    }
}
```

D.Basic steps

（1）Programming FTP Client

1> Getting files list

➢ Establishing socket connnection between clients and FTP server.

```
socket = new Socket(host, port);
reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
writer = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()));
```

➢ Send USER、PASS commands to FTP Servers.

```
writer.println(“USER root”);
writer.flush();
reponse = reader.readLine();
writer.println(“PASS root”);
writer.flush();
reponse = reader.readLine();
```

➢ Use PASV Command to get port number that the server is listening to, and establish data connection.

```
writer.println(“PASV”);
writer.flush();
reponse = reader.readLine(); // 227 entering passive mode    (h1,h2,h3,h4,p1,p2)
```

➢ Calculate data port number by p1*256+p2, connecting to data port, preparing receiving data.

```
Socket dataSocket = new Socket(ip, port1);
```

➢ Use List command to achieve files list.

```
writer.println(“List”);
writer.flush();
reponse = reader.readLine();
```

➢ Receiving data from data port.

```
DataInputStream dis = new DataInputStream(dataSocket.getInputStream());
            String s = "";
            while ((s = dis.~~readLine~~()) != null) {
                  String l = new String(s.getBytes("ISO-8859-1"), "utf-8");
                  System.out.println(l);
```

}

- After downloading, disconnect data port and sent QUIT command in order to quit.

```
writer.println("QUIT");
writer.flush();
reponse = reader.readLine();
```

2> Downloading files

- Client and FTP Server establish socket connection.
- Sent USER、PASS command to the server in order to log in.
- Use PASV Command to get port number that the server is listening to, and establish data connection.
- Use RETR command to download files.

```
writer.println("RETR filename");
writer.flush();
reponse = reader.readLine();
```

Receive data from data port, and save the data one local disks.

If the system support resuming from break point and a client already has local files, client will sent REST Size command first to determine the offset and send RETR command.

- After downloading, data ports will disconnect and client will send QUIT command to quit.

3> Uploading files

- Clients and FTP servers establish Socket connection.
- Clients send USER and PASS commands to FTP Server and log in.
- Clients use PASV command to get listening port of servers, establishing data connection.
- Clients use STOR command to upload files.

```
writer.println("STOR filename");
writer.flush();
response = reader.readLine();
```

Using data ports to send files to servers.

If the system support resuming from break point, clients will send SIZE command to check if the file size is greater than 0. If the file size if greater than 0, it means that files has already sent. Clients will then use InputStream.skip(length) to indicate offset of the file.

- After data transition, use QUIT to quit.

(2） Program servers

1. Server will allocate a new thread when a new client connect.
2. This new thread will receive requests from the client and deal with them.

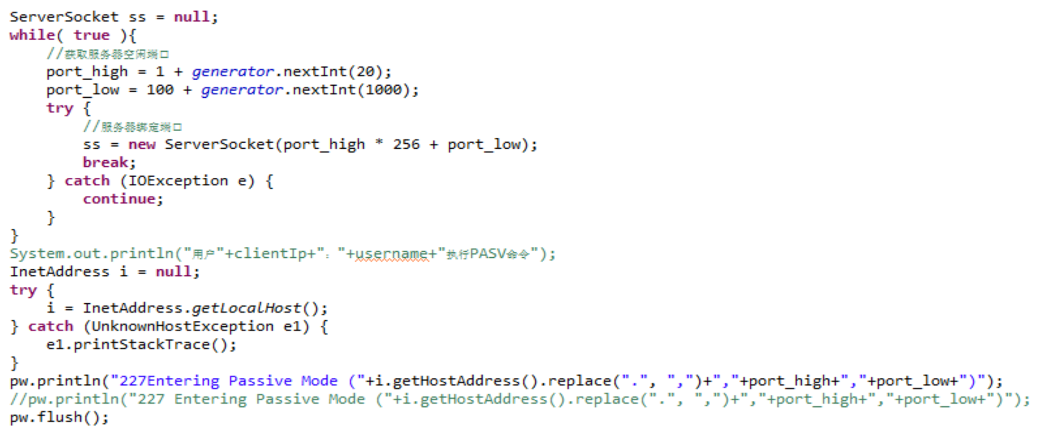
1> User Command

2> Pass Command

Using user names and password to check validation of users.

3> PASV Command

Get an available port of the server, send it to the client as a data port.



```
ServerSocket ss = null;
while( true ){
    //获取服务器空闲端口
    port_high = 1 + generator.nextInt(20);
    port_low = 100 + generator.nextInt(1000);
    try {
        //服务器绑定端口
        ss = new ServerSocket(port_high * 256 + port_low);
        break;
    } catch (IOException e) {
        continue;
    }
}
System.out.println("用户"+clientIp+"："+username+"执行PASV命令");
InetAddress i = null;
try {
    i = InetAddress.getLocalHost();
} catch (UnknownHostException e1) {
    e1.printStackTrace();
}
pw.println("227Entering Passive Mode ("+i.getHostAddress().replace(".", ",")+","+port_high+","+port_low+")");
//pw.println("227 Entering Passive Mode ("+i.getHostAddress().replace(".", ",")+","+port_high+","+port_low+")");
pw.flush();
```

4> Size Command

Returning the size of a file.

5> REST Command

Recording offset of files so that clients can use them later on.

6> RETR Command

Sending data through data ports.

7> STOR Command

Receiving data through data ports.

8> QUIT Command

Disconnect.

9> CWD Command

Set current directory the position that we can uploading and downloading files,

10> LIST Command

Listing information of files, including size, creating date&time and name.

E. Submission

Submit whole project of FTP client.
Submit whole project of FTP Server.

F. Critical thinking

Think Work flows and principles of FTP.

Project 2 Implement Network Communication Using HTTP (8 hours)

A.Purpose

（1）Understanding principles of HTTP communication；

（2）Learning and using Socket and multi-threads to implement simple HTTP server；

（3）Learning and using Socket to implement simple HTTP client；

B.Content

We require students to complete following functions（Using GET）：

Name	Description	Priority
Server	Offering MIME recourse like HTML, JPG, etc.	high
Client	Visiting servers, receiving HTML and JPG recourse, saving it to local disks.	high
	Visiting servers, receiving WMV recourse, saving it to local disks.	low

（1） Programming simple HTTP 1.1 client

Requirement：

- Using command line interface to build simple HTTP1.1 client.
- Simple HTTP1.1 client can connect to web server on the internet.
- Web servers on the internet can deal with simple requests from clients.
- Simple HTTP 1.1 client can receive response from servers, and display response header on command line, saving responses in a file.

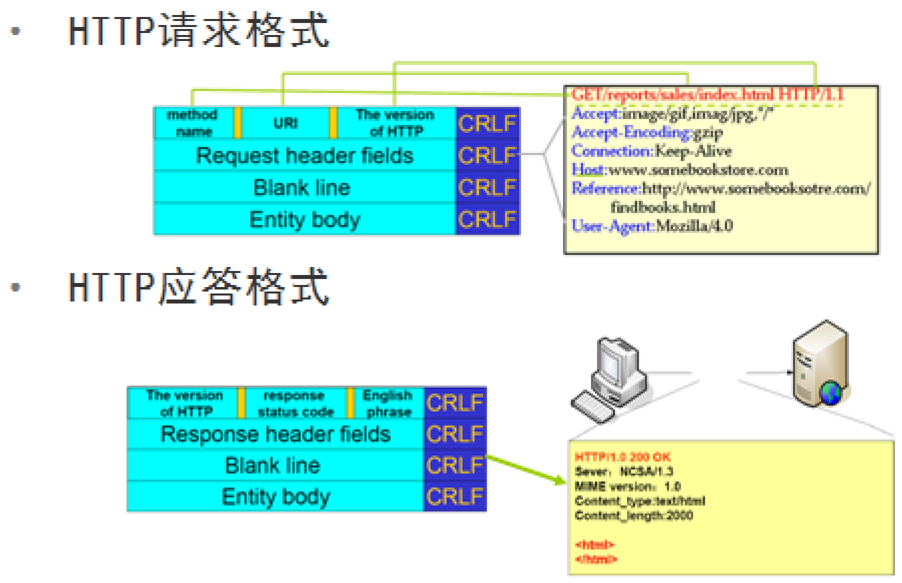
（2） Programming simple HTTP 1.1 server

Requirement：

- Build a simple HTTP1.1 server on port 8888.
- This server can response to all requests ignoring whether requests are valid. We require servers should give successful response when receiving HTTP GET method according to RFC.
- Servers should be on service all the time.
- Passing parameters on command line to determine servers' default welcome page. (E.g: Give c:\www as parameters on command line to servers, if clients send GET /test/index.html, then server should send C:\www\test\index.html to clients. It should have error message if the file doesn't exist）；(not mandatory)
- After successful response, server should recognize at least two kinds of files extension. (E.g. If extension of a file is .htm or .html, MIME type will be text/html)
- Server can response HTML pages embedded JPEG files.
- Can visit HTTP1.1 servers and get their web resources using Netscape or Internet Explorer.

C.Previewing

HTTP communication principles



- HTTP请求格式

```
GET/reports/sales/index.html HTTP/1.1
Accept:image/gif,imag/jpg,"/"
Accept-Encoding:gzip
Connection:Keep-Alive
Host:www.somebookstore.com
Reference:http://www.somebooksotre.com/
        findbooks.html
User-Agent:Mozilla/4.0
```

- HTTP应答格式

```
HTTP/1.0 200 OK
Sever:  NCSA/1.3
MIME version:  1.0
Content_type:text/html
Content_length:2000

<html>
</html>
```

D.Basic Steps

(1) Building HTTP Servers

1. Initialize ServerSocket, listening to client, allocating threads to tackle requests from clients.

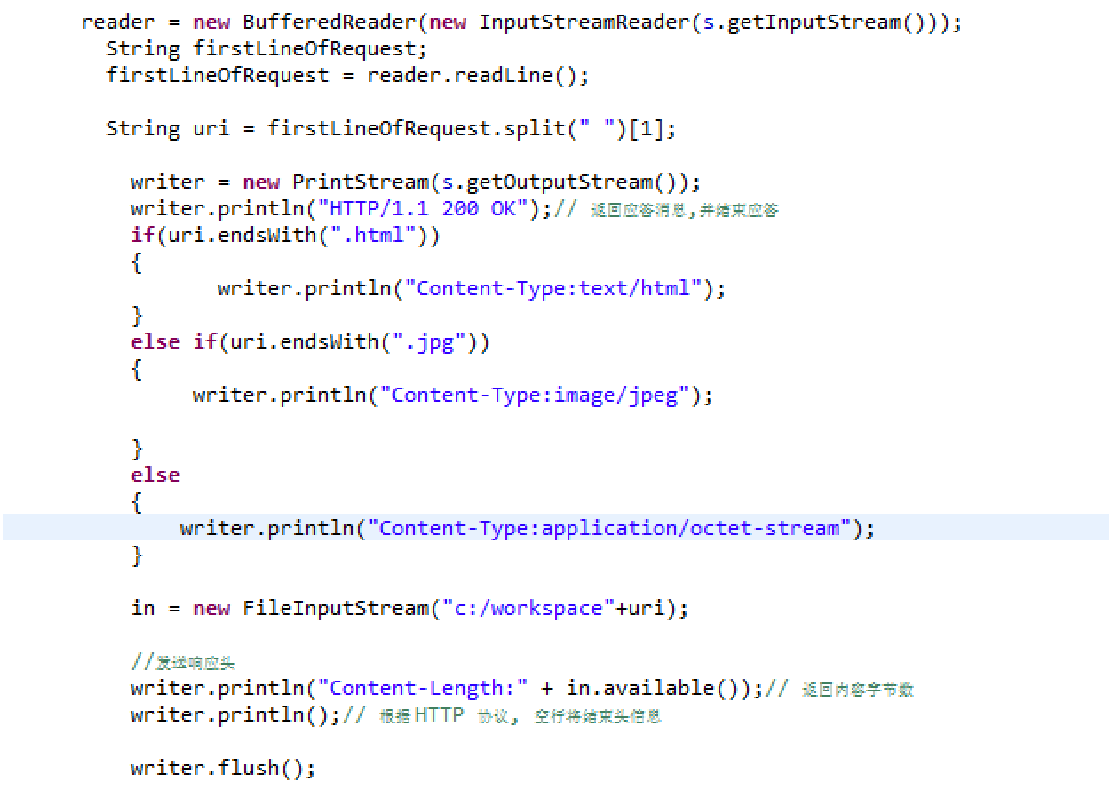
```
ServerSocket ss=null;
try {
    ss = new ServerSocket(8888);
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
};
while (true) {

    try {

          Socket s = ss.accept();
          TaskThread t = new TaskThread(s);
          t.start();
    } catch (IOException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }

}
```

2. According to suffix name of clients' files, send response type.



```
reader = new BufferedReader(new InputStreamReader(s.getInputStream()));
  String firstLineOfRequest;
  firstLineOfRequest = reader.readLine();

  String uri = firstLineOfRequest.split(" ")[1];

    writer = new PrintStream(s.getOutputStream());
    writer.println("HTTP/1.1 200 OK");// 返回应答消息,并结束应答
    if(uri.endsWith(".html"))
    {
           writer.println("Content-Type:text/html");
    }
    else if(uri.endsWith(".jpg"))
    {
         writer.println("Content-Type:image/jpeg");

    }
    else
    {
        writer.println("Content-Type:application/octet-stream");
    }

    in = new FileInputStream("c:/workspace"+uri);

    //发送响应头
    writer.println("Content-Length:" + in.available());// 返回内容字节数
    writer.println();// 根据HTTP 协议, 空行将结束头信息

    writer.flush();
```

3. Sending response data.

```
//Sending response body
                    byte[] b = new byte[1024];
                    int len = 0;
                    len = in.read(b);
                    while(len!=-1)
                    {
                        os.write(b, 0, len);
                        len =    in.read(b);
                    }
                os.flush();
```

4. If required recourse doesn’t exist, sent pure text response message.

```
//Sending response head
                    writer.println("HTTP/1.1 404 Not Found");
                    writer.println("Content-Type:text/plain");
                    writer.println("Content-Length:7");
                    writer.println();

                      //Send response body
                      writer.print("Content dose not exist");
                 writer.flush();
```

After compiling at server side, test your code by browsers:

E.g: Entering http://localhost:8888/login.html

       Entering http://localhost:8888/ios.jpg

(2) Building HTTP Client

1. Connecting to servers

```
    Socket s = new Socket(host, port);
```

2. Sent request header

```
        PrintStream writer = new PrintStream(s.getOutputStream());
        writer.println("GET /"+filename+" HTTP/1.1");
        writer.println("Host:localhost");
        writer.println("connection:keep-alive");
        writer.println();
        writer.flush();
```

3. Sent request body(In GET, requested data follows URL, while in POST, requesting data is put inside request body.)

4. Receiving response state

Successful response (Status code 200) - Saving resources to local disks

```
//Skip first 4 lines, starting to read response data.
InputStream in = s.getInputStream();
BufferedReader reader = new BufferedReader(new InputStreamReader(in));
String firstLineOfResponse = reader.readLine();//HTTP/1.1 200 OK
String secondLineOfResponse = reader.readLine();//Content-Type:text/html
String threeLineOfResponse = reader.readLine();//Content-Length:
String fourLineOfResponse = reader.readLine();//blank line
//Reading response data, saving files.
//success
byte[] b = new byte[1024];
OutputStream out = new FileOutputStream(savelocation+"/"+filename);
int len = in.read(b);
while(len!=-1)
{
    out.write(b, 0, len);
    len = in.read(b);
}
in.close();
out.close();
```

Response failure (Status code 404) - Print related info to console

```
//output error message
StringBuffer result = new StringBuffer();
String line;
while ((line = reader.readLine()) != null) {
        result.append(line);
}
reader.close();
System.out.println(result);
```

E. Submission

Submit whole project of HTTP client.
Submit whole project of HTTP Server.

F. Critical thinking

C
R
R

[illegible]

Project 3 Implement Chat Room Using Web Socket (12 hours)

A.Purpose

（1） Understand theories of WebSocket.
（2） Use tomcat server to establish chat room.

B. Content

We require students to complete following functions：

Name	Description	Priority
Multicast	More than 3 persons can chat at web pages.	high
Unicast	User log in, display online users, choose and talk to someone personally.	medium
Online Service	Online Service	low

C.Previewing

(1)Web Socket Concept

Nowadays, lots of web sites implemented instant communication. Technology used is polling, a method that browser send HTTP request to the server every given interval (can be one second) and the server will send back freshest data to clients. This traditional HTTP request model has a obvious drawback that clients need to continuously send request to the server, while HTTP request header is relatively long and data in the header is just a little. This will consume lots of bandwidth.

In Web Socket API, clients and servers need only “shack hands” at the beginning. Then, they will establish a fast path and transmitting data. In web Socket Protocol, we have two advantages of instant communication:

1. Header

Header used to communicate is as small as 2 Bytes.

2. Server Push

Servers needn’t passively receive request from browsers any more. Servers can now push data to clients.

Browser Request

```
GET /webfin/websocket/ HTTP/1.1
Host: localhost
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: xqBt3ImNzJbYqRINxEFlkg==
Origin: http:// Server address
Sec-WebSocket-Version: 13
```

Server Response

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
```

Connection: Upgrade
Sec-WebSocket-Accept: K7DJLdLooIwIG/MOpvWFB3y3FE8=

Implement the Browser of WebSocket

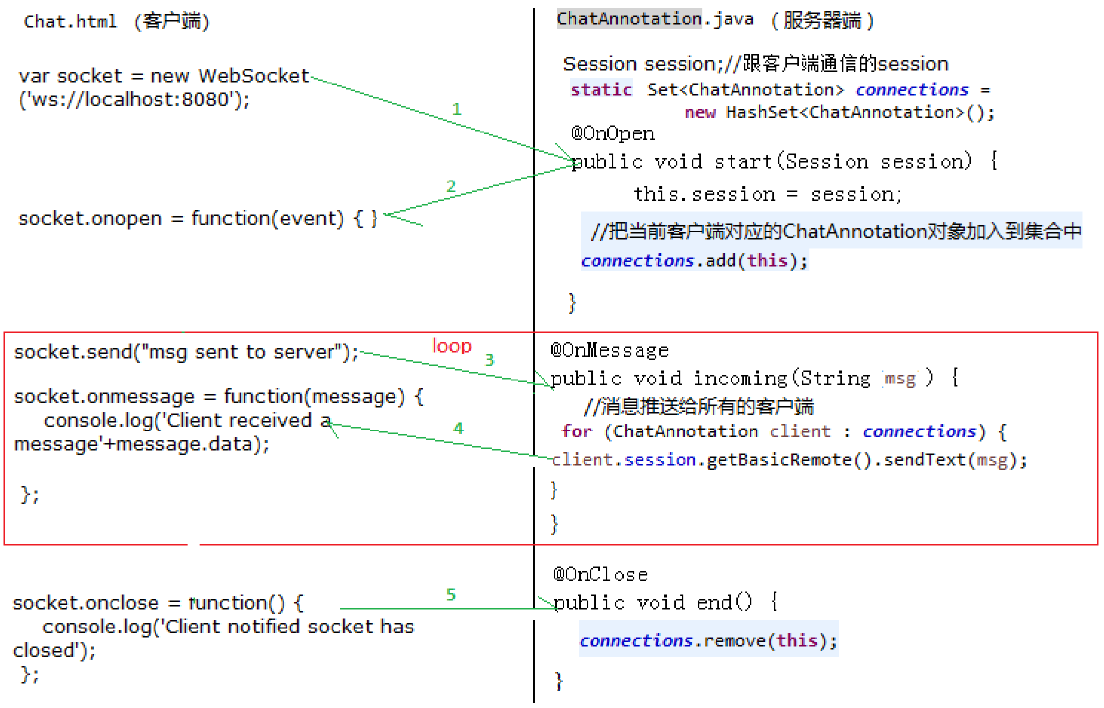
Chrome	Supported in version 4+
Firefox	Supported in version 4+
Internet Explorer	Supported in version 10+
Opera	Supported in version 10+
Safari	Supported in version 5+

A server implementing Web Socket

jetty 7.0.1 Version
resin
pywebsocket, apache http server Extension
apache tomcat 7.0.27 Version
Nginx 1.3.13 Version
jWebSocket java Implementation

What we should notice is that there is not standard api, and different implementations have their own api, so using websocket development tools have risks. Maybe developers will be locked on some platform or maybe developers will be force to update tools.

(2)Web Socket Communication Principle



```
Chat.html (客户端)

var socket = new WebSocket
('ws://localhost:8080');

socket.onopen = function(event) { }

socket.send("msg sent to server");

socket.onmessage = function(message) {
    console.log('Client received a
message'+message.data);

 };

socket.onclose = function() {
    console.log('Client notified socket has
closed');
 };
```

```
ChatAnnotation.java (服务器端)

 Session session;//跟客户端通信的session
  static  Set<ChatAnnotation> connections =
              new HashSet<ChatAnnotation>();
  @OnOpen
  public void start(Session session) {
          this.session = session;
     //把当前客户端对应的ChatAnnotation对象加入到集合中
    connections.add(this);

  }

@OnMessage
public void incoming(String msg ) {
   //消息推送给所有的客户端
 for (ChatAnnotation client : connections) {
client.session.getBasicRemote().sendText(msg);
}
}

@OnClose
public void end() {
    connections.remove(this);

}
```

1) Clients send requests, and server create a ChatAnnotation object. This will automatically run start method and take web socket session object as an input. Server will put current ChatAnnotation object into static set.
2) Client onopen method will automatically run.

3) Clients will trigger socket.send(), server will call incoming method to automatically run itself. Message of client will be added as parameter automatically. Servers can iterate static set, and send message to all clients.
4) After receiving message, method onmessage will automatically run on clients.If client log out, onclose method will automatically run on clients and end method will automatically run on servers. In end method, server will delete current ChatAnnotation object from static set.

D.Basic Steps

(1) Programming Client：

1） Clients send request, considering compatibility of browsers.

```
Var host = “ws://localhost:8080/ProjectName/ChatServlet”;
if ('WebSocket' in window) {
        Chat.socket = new WebSocket(host);
} else if ('MozWebSocket' in window) {
        Chat.socket = new MozWebSocket(host);
}
```

2) Add “enter”event at client input box. In that event, clients will send message to the server.

```
document.getElementById('chat').onkeydown = function(event) {
  if (event.keyCode == 13) {
                socket.send(document.getElementById('chat').value);
    }
};
```

3) Listening to the feedback of the server, print the message after receiving it.

```
socket.onmessage = function (message) {
                        var console = document.getElementById('console');
                        var p = document.createElement('p');
                        p.innerHTML = message.data;
                        console.appendChild(p);
                        };
```

(2) Programming Server:

1) Programming start method(@OnOpen), put server into the set, informing all clients that a new client connects.

```
@OnOpen
  public void start(Session session) {
          this.session = session;
          connections.add(this);
          String message = String.format("* %s %s", nickname, "has joined.");
          broadcast(message);
  }
```

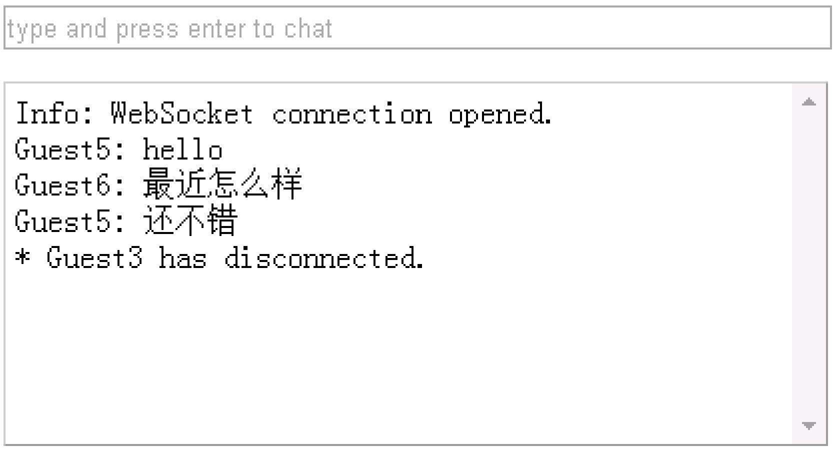
2) Programming incoming method (@OnMessage), receiving messages from clients, broadcast them to other clients.

```
@OnMessage
    public void incoming(String message) {
        broadcast(message);
    }
```

3) Programming end method(@OnClose), removing current server object.

```
@OnClose
    public void end() {
        connections.remove(this);
    }
```

(3)Booting up tomcat, running in browsers:



```
Info: WebSocket connection opened.
Guest5: hello
Guest6: 最近怎么样
Guest5: 还不错
* Guest3 has disconnected.
```

E. Submission

The project of chat room.

F. Critical Thinking

Compared to HTTP protocol, what’s the advantages of using WebSocket?

Project 4 Distributed Agenda Service Based on RMI (Not Mandatory)

A. Purpose

（1） Master basic concept of distributed system.
（2） Master ways to develop distributed system.
（3） Master mechanism of RMI.
（4） Program RMI using RMI tools in Java.

B. Content

（1） Programming RMI

Use RMI to build a distributed agenda service system. All clients should use sharing agenda service. This service include searching, adding, deleting agenda. Servers allow user to register and undo agenda.

➢ Register：New users should register first before using the system. Before registering, users shall offer a user name and a password. If the user name that a user enter has already been used, system need to inform a error message. If registration is successful, the system should also give a message.

register [username] [password]

➢ Add agenda：Users can add agenda to their schedule. Meeting should take place between two users. A user can not start a meeting without others. When adding meetings, the system require date&time of both beginning and ending, commend, users involved. After adding meetings, this meeting should be displayed to all users involved, which means all users in that meeting should be informed. If this meeting's schedule and another meeting's schedule of the same person conflict, the system should report error message. Whether an operation is successful, the system should report message to users.

add [username] [password] [otherusername] [start] [end] [title]

➢ Searching meetings：Offering a time interval, users can search all of his or her meetings during that interval(Include meeting held by himself/herself and meeting inviting him/her). Searching input is the beginning and ending of the interval. After searching, the system should give a list. This list include date&time of both beginning and ending, commend, users involved.

query [username] [password] [start] [end]

➢ Deleting meeting：Users can delete meetings that created by them. Deleting meeting inputs are identifier of the meeting and the execute user.

delete [username] [password] [meetingid]

➢ Clearing meeting：A user can delete all his/her meetings.

clear [username] [password]

This lab do NOT require permanent storage at the server side. If students get enough time, they can try relational database or file system after finishing requirements above.

C. Previewing

（1） Review Java Web Programming and knowledge related to it.
（2） Review RMI.

D. Basic Steps

【RMI (Remote Method Invocation) Concept】

To bridge network, running a process on other machines, traditional approach may confuse us. It is not only annoying but also risking. Best way is to send message to an object that run on remote machine and to get the result. The whole procedure is just like running on local machines. This “best way” is RMI.

Writing a distributed app using RMI, core three aspects as follow:

（1） Locating remote object

1. An app can use RMI name registrar server to register a remote commuting object.

2. Can pass and return a pointer or reference just like operation a normal object.

（2） Commuting with remote object

Underlying communication is done by RMI. For system developers, remote procedure call is just the same as regular Java call.

（3）Loading class byte code for objects to transmit

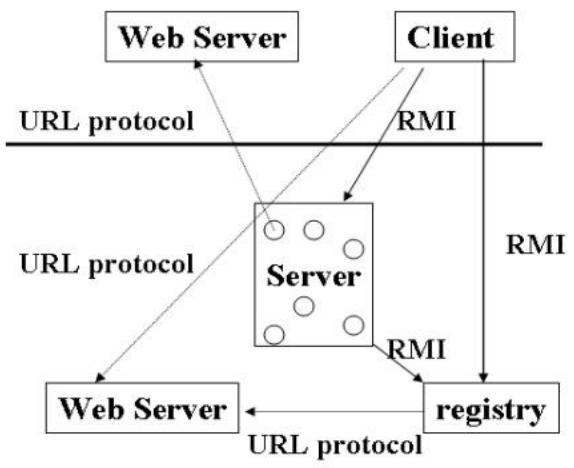
RMI allow caller to pass an object to the remote object RMI, so RMI support mechanism that can allocate objects.

【Work flows of RMI（Remote Method Invocation，远程函数调用）】

Systems that using RMI communication can be distinguished as two classes: Client and Server. Servers provide RMI service, and clients call methods of servers. RMI servers should register using loopup service. Then, clients are able to fine service of RMI servers. Once RMI servers registered, These servers will waiting for requests from clients.

RMI clients send RMI message to call methods on remote objects.Before call any remote service, clients should have a index of remote object. Once clients get the index, clients and remote server can commute with each others.

Client/Server RMI communicating procedure as follows:



Communication between clients and remote service is done by stub and skeleton objects in RMI. Stub is a agent object, and it will forward request of clients to RMI servers. Skeleton is responsible for listening requesting from clients and it will sent these requests to RMI server. Stub and skeleton objects’ communication is done by TCP Socket.

【Procedure of Programming RMI】

（1） Define RMI Service interface

Extend java.rmi.Remote interface

E.g:

```
public class RMILightBulb extends java.rmi.Remote{
        public void on() throws java.rmi.RemoteException;
        public void off() throws java.rmi.RemoteException;
        public Boolean isOn() throws java.rmi.RemouteException;
}
```

2） Implementing RMI service interface

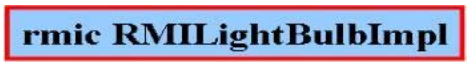
E.g:

```
public class RMILightBulbImpl
extends java.rmi.server.UnicastRemoteObject
implements RMILightBulb{
        public RMILightBulbImpl() throws java.rmi.RemoteException{
                setBulb(false);
        }
        private Boolean lightOn;
        public void on() throws java.rmi.RemoteException{
                setBulb(true);
        }
        public boolean off() throws java.rmi.RemoteException{
                setBulb(false);
        }
        public Boolean isOn() throws java.rmi.RemouteException{
                return getBulb();
        }
        public void setBulb(boolean value){
                lightOn=value;
        }
        public Boolean getBulb(){
                return lightOn;
        }
}
```

（3） Creating Stub and Skeleton Class

Stub class is responsible for distributing, while Skeleton class is responsible for dealing with RMI request. However, developers can not create these classes. If remote server exist, we can use rmic tool to create them.

E.g：



```
rmic RMILightBulbImpl
```

Then it will generate the following files：

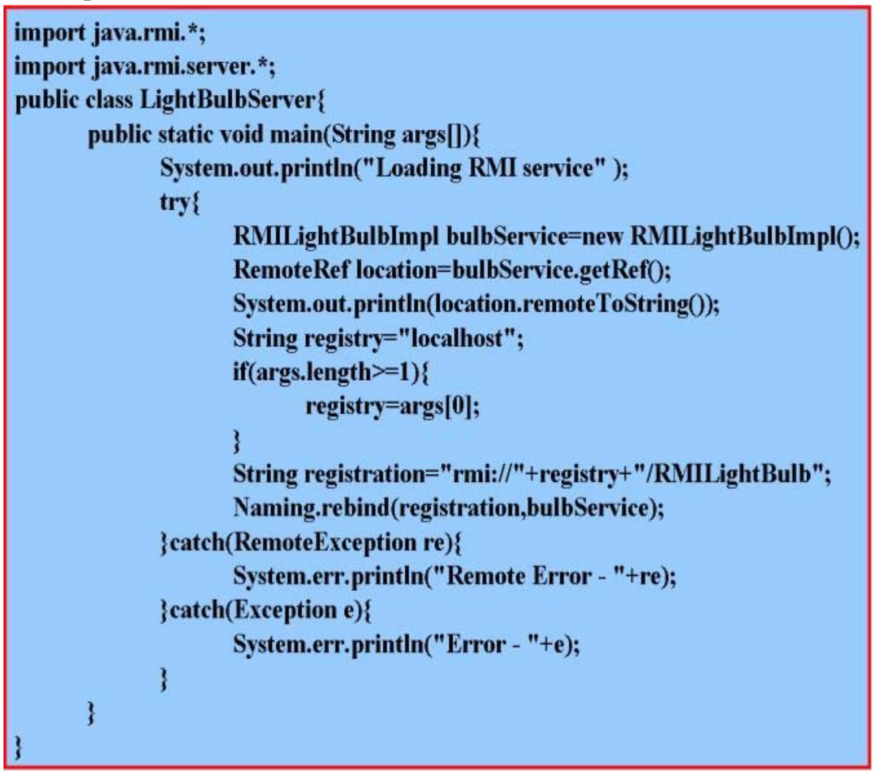
RMILightBulbImpl_Stub.class

RMILightBulbImpl_Skeleton.class

（4） Creating RMI Servers

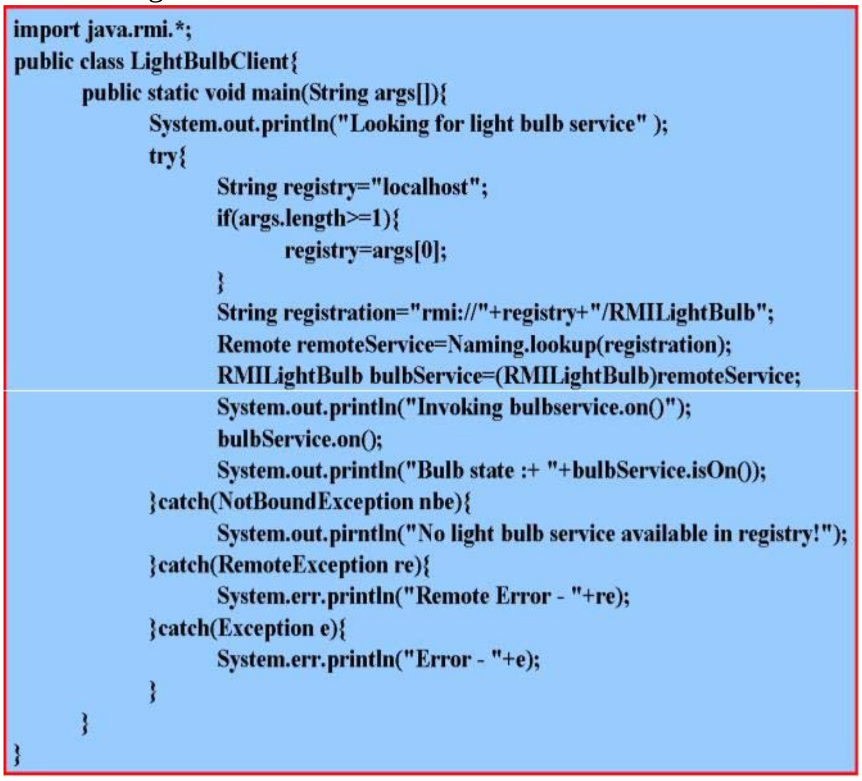
RMI servers are responsible for generating instances of RMI services.

For example：



```
import java.rmi.*;
import java.rmi.server.*;
public class LightBulbServer{
        public static void main(String args[]){
                System.out.println("Loading RMI service" );
                try{
                        RMILightBulbImpl bulbService=new RMILightBulbImpl();
                        RemoteRef location=bulbService.getRef();
                        System.out.println(location.remoteToString());
                        String registry="localhost";
                        if(args.length>=1){
                                registry=args[0];
                        }
                        String registration="rmi://"+registry+"/RMILightBulb";
                        Naming.rebind(registration,bulbService);
                }catch(RemoteException re){
                        System.err.println("Remote Error - "+re);
                }catch(Exception e){
                        System.err.println("Error - "+e);
                }
        }
}
```

（5） Creating RMI Clients



```
import java.rmi.*;
public class LightBulbClient{
        public static void main(String args[]){
                System.out.println("Looking for light bulb service" );
                try{
                        String registry="localhost";
                        if(args.length>=1){
                                registry=args[0];
                        }
                        String registration="rmi://"+registry+"/RMILightBulb";
                        Remote remoteService=Naming.lookup(registration);
                        RMILightBulb bulbService=(RMILightBulb)remoteService;
                        System.out.println("Invoking bulbservice.on()");
                        bulbService.on();
                        System.out.println("Bulb state :+ "+bulbService.isOn());
                }catch(NotBoundException nbe){
                        System.out.pirntln("No light bulb service available in registry!");
                }catch(RemoteException re){
                        System.err.println("Remote Error - "+re);
                }catch(Exception e){
                        System.err.println("Error - "+e);
                }
        }
}
```

【RMI running steps】

（1） Managing to compile all Java files

（2） Call rmiregistry

（3） Running the server at independent console.

For example：

java LightBulbServer hostname

（4） Running the client at independent console.

E. Submission

1．Requirement：

All essential files that can run agenda service.

readme.txt——Every command compiled and executed.

2．Write procedure of implementing the program in to report and submit the report. Students can use UML, such as class diagram, object diagram, sequential diagrams, use case diagram, state diagram and structural flow chart, to describe the procedure. Students can also list crucial source codes.

F. Critical Thinking

Try to think steps that programming RMI.