**Name : Tong Wu CWID: A20410395**

Table of Contents

[PART 1: 2-7](#_Toc441827106)

[PART 2: 8-11](#_Toc441827106)

**PART 1:**

**Server code:**

**package** Assignment1;

**import** java.io.IOException;

**import** java.net.ServerSocket;

**import** java.net.Socket;

/\*\*

\*

\* **@author** TongWu A20410395 Group11

\*

\*/

**public** **class** Server {

**static** **int** *i*=0;

**public** **static** **void** main(String[] args) **throws** IOException {

//1 create serversocket object

ServerSocket ss = **new** ServerSocket(8888);

/\*2 invoke accept method and wait for client connection

\* in order to count quantities so i use thread

\*/

System.***out***.println("\*\*\*\*\*The server is about to start waiting for client connections\*\*\*\*\*\*\*");

**while**(**true**) {

Socket s=ss.accept();

*i*++;

**new** Thread(**new** Task(s)).start();

}

}

}

**Client1 code:**

**package** Assignment1;

**import** java.io.BufferedReader;

**import** java.io.BufferedWriter;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.io.OutputStreamWriter;

**import** java.net.Socket;

**import** java.util.Scanner;

/\*\*

\*

\* **@author** TongWu A20410395 Group11

\*

\*/

**public** **class** Client1 {

**public** **static** **void** main(String[] args) {

**try** {

Socket s = **new** Socket("localhost", 8888);

BufferedWriter request = **new** BufferedWriter(**new** OutputStreamWriter(s.getOutputStream()));

BufferedReader respond = **new** BufferedReader(**new** InputStreamReader(s.getInputStream()));

String back = **null**;

StringBuffer sb = **new** StringBuffer ();

Scanner sc = **new** Scanner(System.***in***);

String info=**null**;

**int** index;

**while**((info=sc.nextLine())!=**null**) {

request.write(info);

request.write("end");

request.newLine();

request.flush();

**if**(info.equals("exit")) {

**break**;

}

**while**((back=respond.readLine())!=**null**) {

**if**((index=back.indexOf("end"))!=-1){

sb.append(back.substring(0, index));

back=sb.toString();

sb.delete(0, index);

System.***out***.println("i am client，server says:"+back);

**break**;

}

}

}

s.shutdownOutput();

s.shutdownInput();

request.close();

respond.close();

s.close();

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

**Client2 code:**

**package** Assignment1;

**import** java.io.BufferedReader;

**import** java.io.BufferedWriter;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.io.OutputStreamWriter;

**import** java.net.Socket;

**import** java.util.Scanner;

/\*\*

\*

\* **@author** TongWu A20410395 Group11

\*

\*/

**public** **class** Client1 {

**public** **static** **void** main(String[] args) {

**try** {

Socket s = **new** Socket("localhost", 8888);

BufferedWriter request = **new** BufferedWriter(**new** OutputStreamWriter(s.getOutputStream()));

BufferedReader respond = **new** BufferedReader(**new** InputStreamReader(s.getInputStream()));

String back = **null**;

StringBuffer sb = **new** StringBuffer ();

Scanner sc = **new** Scanner(System.***in***);

String info=**null**;

**int** index;

**while**((info=sc.nextLine())!=**null**) {

request.write(info);

request.write("end");

request.newLine();

request.flush();

**if**(info.equals("exit")) {

**break**;

}

**while**((back=respond.readLine())!=**null**) {

**if**((index=back.indexOf("end"))!=-1){

sb.append(back.substring(0, index));

back=sb.toString();

sb.delete(0, index);

System.***out***.println("i am client，server says:"+back);

**break**;

}

}

}

s.shutdownOutput();

s.shutdownInput();

request.close();

respond.close();

s.close();

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

**Task code:**

**package Assignment1;**

**import java.io.BufferedReader;**

**import java.io.BufferedWriter;**

**import java.io.InputStreamReader;**

**import java.io.OutputStreamWriter;**

**import java.net.Socket;**

**import java.util.regex.Pattern;**

**/\*\***

**\***

**\* @author TongWu A20410395 Group11**

**\***

**\*/**

**public class Task implements Runnable {**

**private Socket s;**

**Pattern p = Pattern.compile("\\d\\\*\\d");**

**public Task(Socket socket) {**

**this.s = socket;**

**}**

**public void run() {**

**try {**

**handleSocket();**

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**}**

**public void handleSocket() throws Exception {**

**BufferedReader request = new BufferedReader(new InputStreamReader(s.getInputStream()));**

**BufferedWriter respond = new BufferedWriter(new OutputStreamWriter(s.getOutputStream()));**

**String info = null;**

**StringBuilder sb = new StringBuilder();**

**int index;**

**/\*i can imitate intercom and just print the info's content before "end"**

**by using StringBuffer and String's substring method such as the model like : "copy that" \*/**

**while((info=request.readLine())!=null) {**

**if((index=info.indexOf("end"))!=-1) {**

**sb.append(info.substring(0, index));**

**info = sb.toString();**

**sb.delete(0, index);**

**System.out.println("i am server，client says:"+info);**

**}**

**if(info.equals("HelloServer")) {**

**respond.write("515OK");**

**respond.write("end");**

**respond.newLine();**

**respond.flush();**

**}**

**// I can calculate the multiplication of any two tens by using regex**

**else if(p.matcher(info).matches()){**

**String [] s1 = info.split("\\\*");**

**int result = 0;**

**result= (Integer.parseInt(s1[0]))\*(Integer.parseInt(s1[1]));**

**respond.write(info+"="+String.valueOf(result));**

**respond.write("end");**

**respond.newLine();**

**respond.flush();**

**}else if(info.equals("count")) {**

**respond.write(String.valueOf(Server.i));**

**respond.write("end");**

**respond.newLine();**

**respond.flush();**

**}else if(info.equals("exit")){**

**respond.write("exit");**

**respond.write("end");**

**respond.newLine();**

**respond.flush();**

**}**

**}**

**s.shutdownInput();**

**s.shutdownOutput();**

**respond.close();**

**request.close();**

**s.close();**

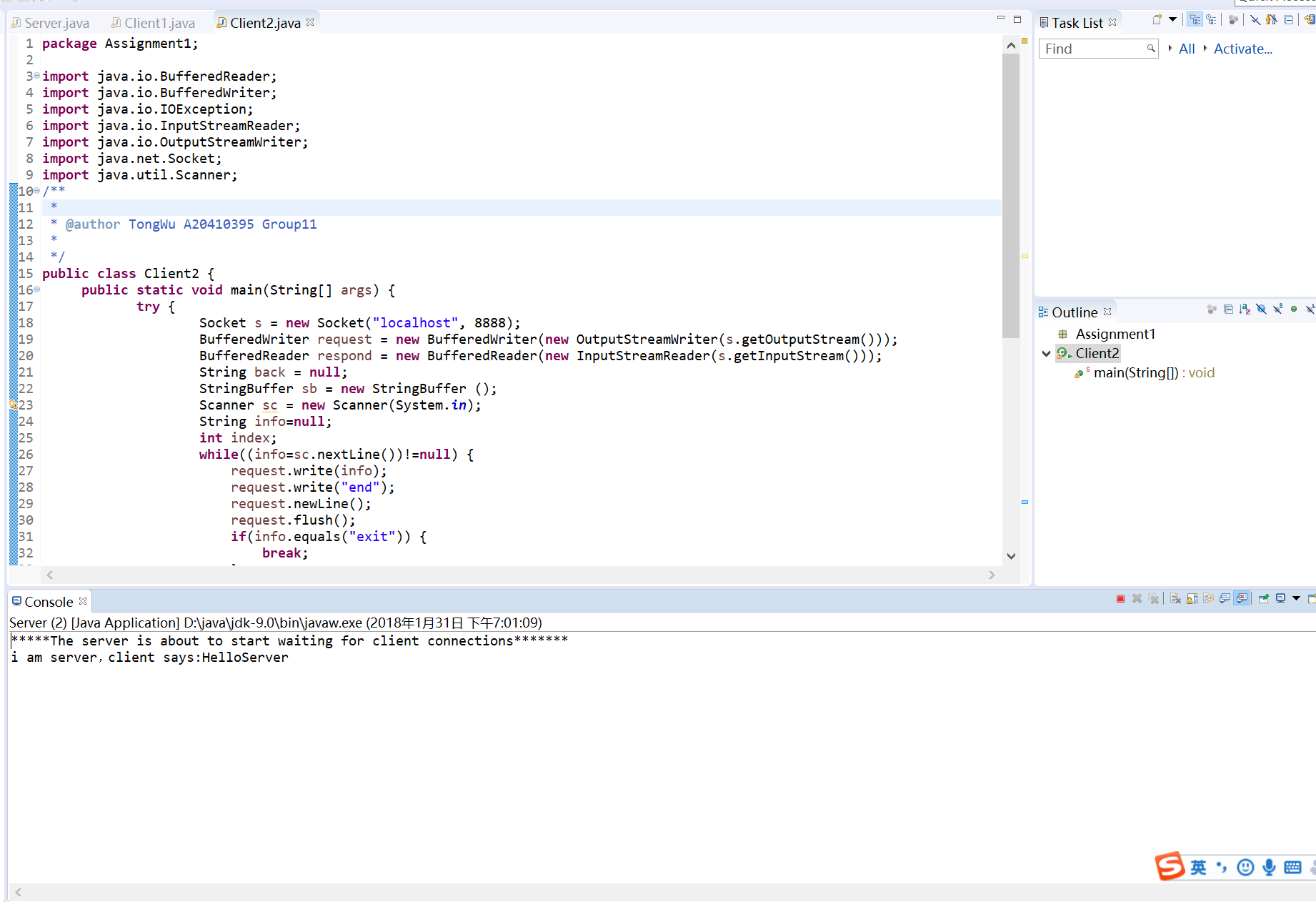
**}**

**}**

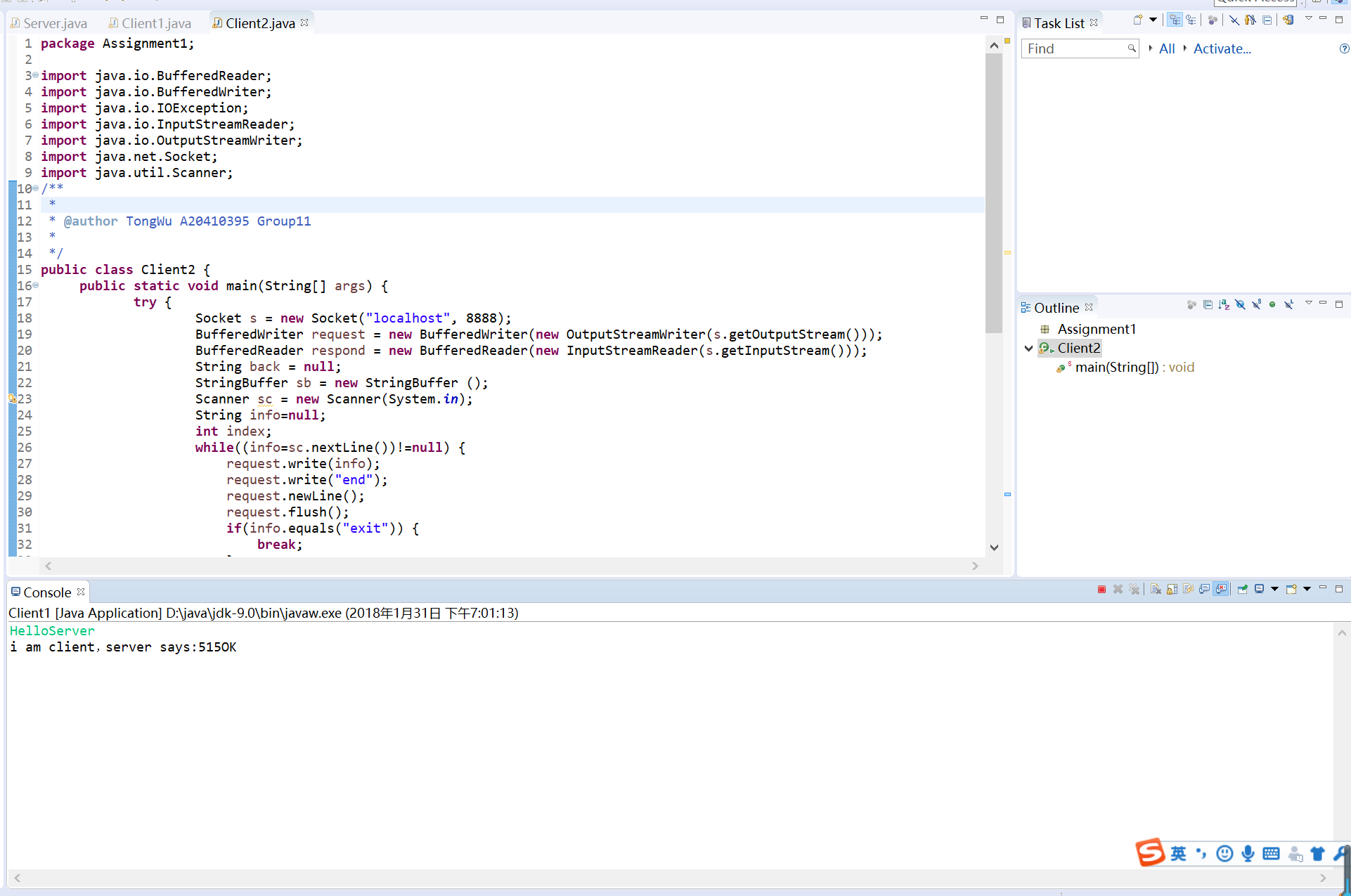
**PART 2:**

**Screenshot:**

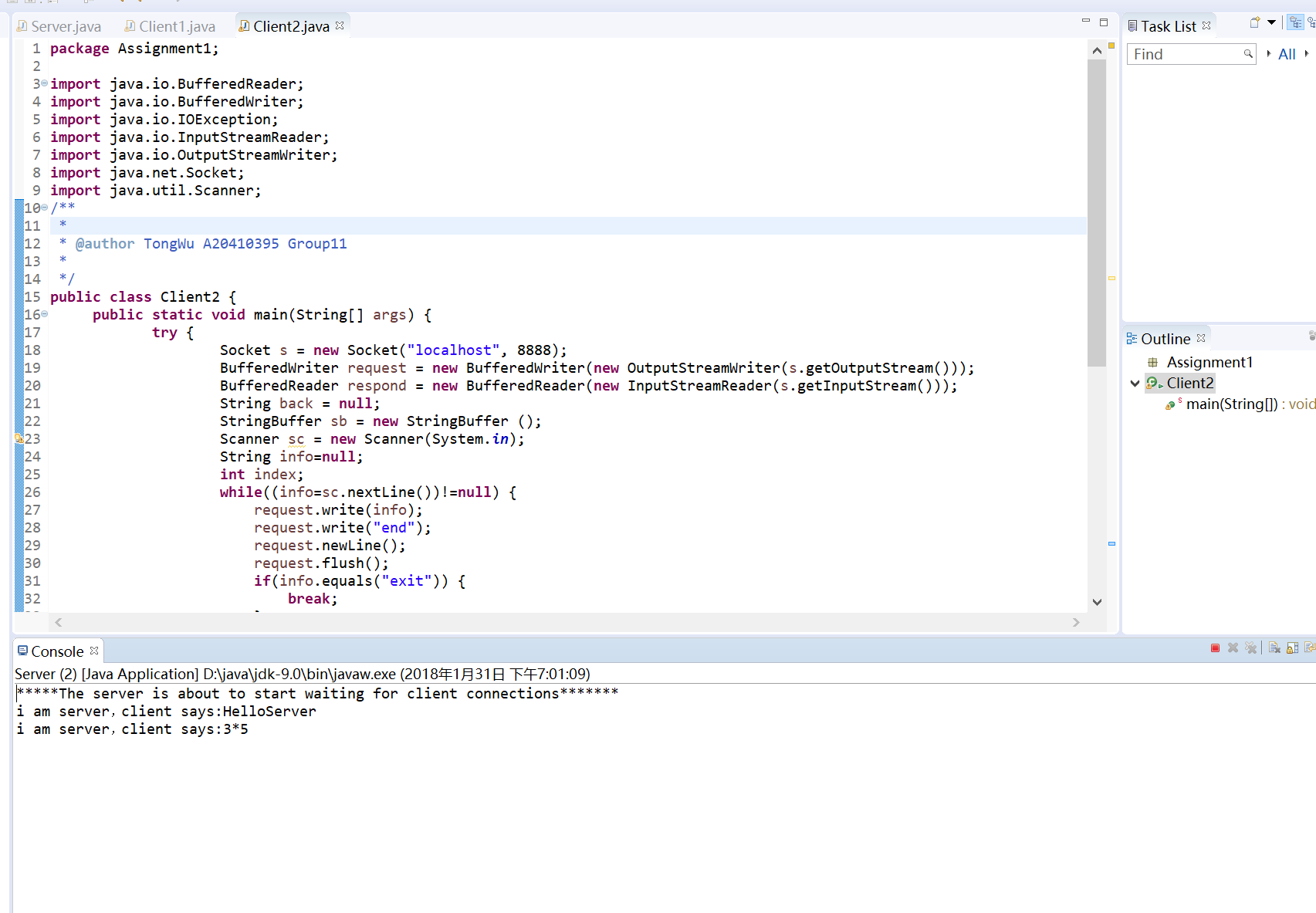
Client1 requests



Server responds client1



Client2 requests



Server responds client2

