**《Java技术》实验报告**

**实验三： 2019 年7 月16日**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **学院** | | 计算机与信息学院 | | **专业班级** | | 计算机20-01 | **姓名** | 叶春亭 | |  | | **成绩** | |  |
| **课程**  **名称** | | Java技术 | | | **实验项目**  **名 称** | 实验三 基于GUI的网络通信编程程序设计 | | |  | | **指导 教师** | | 薛峰 | |
| 教师评语 |  | | 教师签名：  年 月 日 | | | | | | | | | | | |

**一．实验目的**

1．掌握Java中GUI程序的编写，包括事件监听机制。

2．掌握Java的网络通信编程，ServerSocket，Socket类的使用。

3．掌握Java中多线程的编程，Thread类，Runnable接口的使用。

4．掌握用面向对象的方法分析和解决复杂问题。

二、实验原理

1、GUI的一些类和组件的应用

2、基于 网络通信和多线程的编程。

3、熟练应用ServerSocket，Socket类，Thread类，Runnable接口。

**三、使用硬件、软件环境**

PC 计算机一台，配置为CPU为1.5G，内存为512M，硬盘为40G，安装Windows10操作系统。

另外，使用JCreator，JDK1.6.0等软件

**四、实验过程、步骤及原始记录(算法、原程序、测试结果，分析等)**

编写程序完成以下功能：

1.设计一个基于GUI的客户-服务器的通信应用程序，如图1，图2所示。

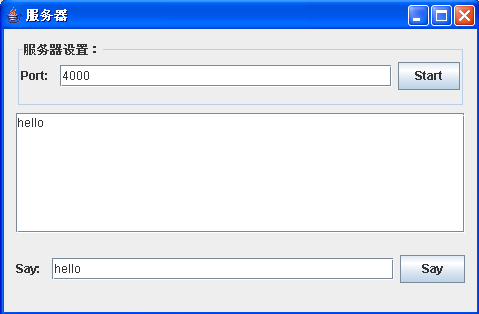


图1 Socket通信服务器端界面



图2 Socket通信客户端界面

2.图1为Socket通信服务器端界面，点击该界面中的【Start】按钮，启动服务器监听服务（在图1界面中间的多行文本区域显示“Server starting…”字样）。图2为Socket通信客户端界面，点击该界面中的【Connect】按钮与服务器建立链接，并在图2所示界面中间的多行文本区域显示“Connect to server…”字样，当服务器端监听到客户端的连接后，在图1界面中间的多行文本区域追加一行“Client connected…”字样，并与客户端建立Socket连接。

3.当图1所示的服务器端和图2所示的客户机端建立Socket连接后，编程实现服务端、客户端之间的“单向通信”：在客户端的输入界面发送消息，在服务端接收该消息，并将接收到对方的数据追加显示在多行文本框中

* **实验思路：**

1、服务器和客户端之前分开写一个java

2、依靠Socket和ServerSocket()实现客户端和服务器之前的连接

3、GUI的基本知识actionPerformed（）

4、保证双向连接，使用线程，读入会阻塞线程，有多次读入，不用线程会一直卡住

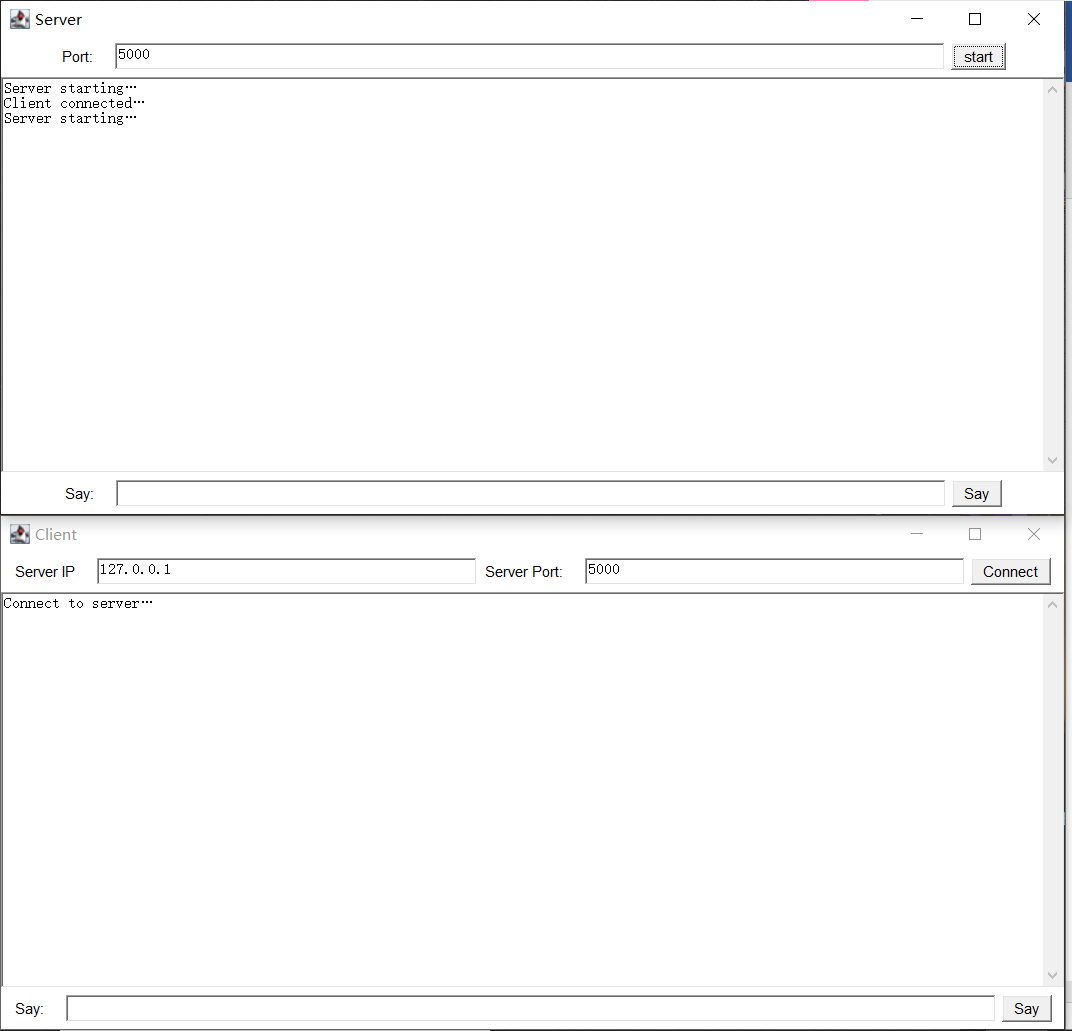
创建两个窗口文件类继承Frame类

排版GUI窗口的内容，包括Button，textFiled类等组件和容器

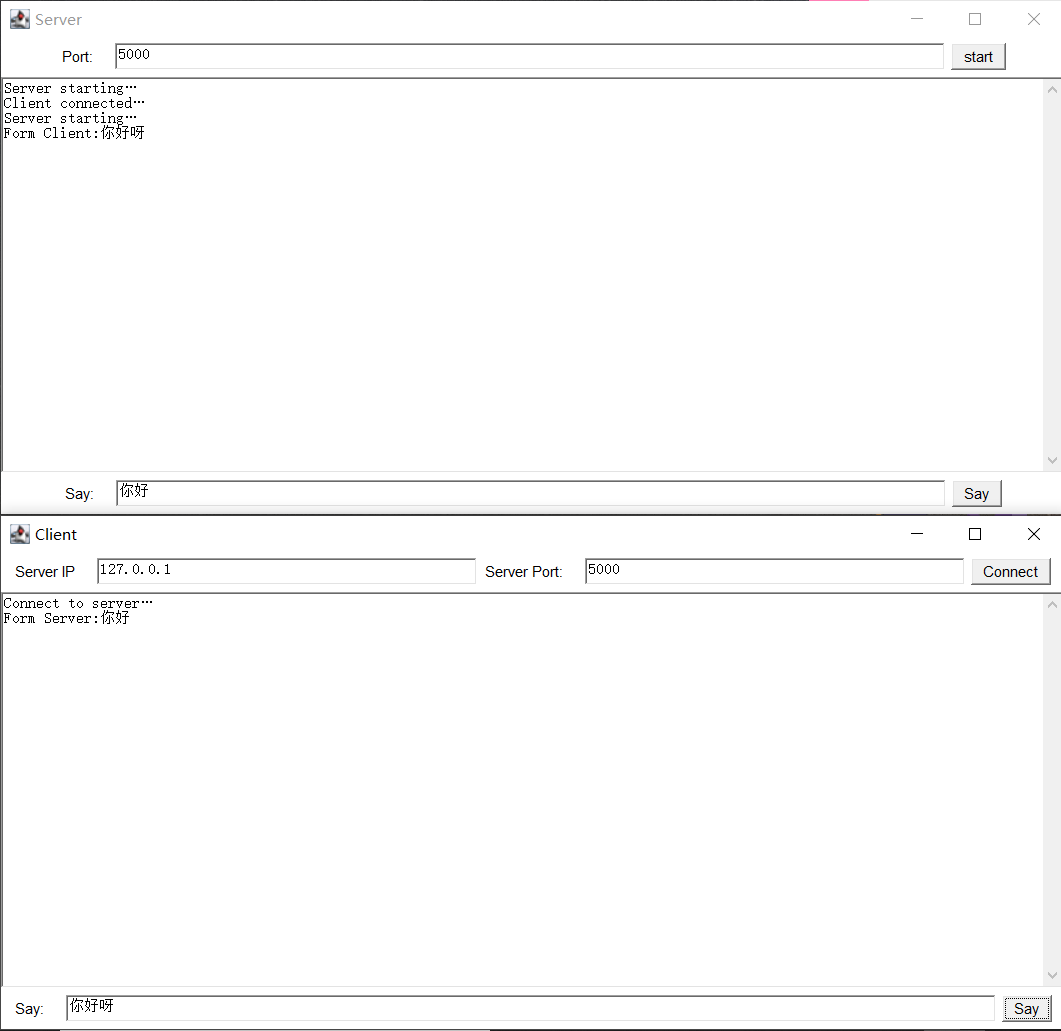
还有事件监听 对象的创建，用到了Socket类 和ServerSocket类来建立两个端口的连接，并且运用了多线程技术将两个文件同时运行，以保证客户 端和服务器端同时运行并接受消息。

以下是运行过程：

建立连接成功



实现双方互相接收消息：



源码：

**package** three;

**import** java.awt.\*;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.io.\*;

**import** java.net.ServerSocket;

**import** java.net.Socket;

**public** **class** ServerFrame **extends** Frame **implements** ActionListener{

Label [] labels = **new** Label[5];

TextField [] textFields = **new** TextField[5];

Button [] buttons = **new** Button[5];

TextArea [] textAreas= **new** TextArea[5];

Panel [] panels = **new** Panel[5];

ServerSocket listen1=**null**;

Socket Client=**null**;

//Choice ClientChoice = new Choice();

//HashMap<String,Socket> socketHashMap = new HashMap<String, Socket>();

**public** ServerFrame(String title){//构造函数

**super**(title);

}

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

ServerFrame f= **new** ServerFrame("Server");

f.DrawGUI();

}

**void** DrawGUI() {//基本绘图

labels[0] = **new** Label("Port:");

labels[1] = **new** Label("Say:");

textFields[0] = **new** TextField(80);

textFields[1] = **new** TextField(80);

buttons[0] = **new** Button(" start ");

buttons[1] = **new** Button(" Say ");

textAreas[0] =**new** TextArea(8,80);

panels[0] = **new** Panel();

panels[1] = **new** Panel();

**for**(**int** i =0;i<=1;++i){

panels[i].add(labels[i]);panels[i].add(textFields[i]);panels[i].add(buttons[i]);

}

add(panels[0],"North");

add(panels[1],"South");

add(textAreas[0],"Center");

pack();

setVisible(**true**);

buttons[0].addActionListener(**this**);//绑定按钮和事件

buttons[1].addActionListener(**this**);

}

**public** **void** actionPerformed(ActionEvent e) {

Object source =e.getSource();

**if**(source == buttons[0]) {

**try** {

**int** port = Integer.*parseInt*(textFields[0].getText().trim());

textAreas[0].append("Server starting…"+'\n');

listen1 = **new** ServerSocket(port);

Client = listen1.accept();

textAreas[0].append("Client connected…"+'\n');

Thread ServerListen = **new** Thread(**new** Server\_ListenThread(Client));

ServerListen.start();//开始Server的读入线程

} **catch** (Exception e1) {

e1.printStackTrace();

}

}**else**{

Thread ServerSpeaker = **new** Thread(**new** Server\_Speaker(Client));

ServerSpeaker.start();//开始Server的讲话线程

}

}

**class** Server\_ListenThread **implements** Runnable{//实现Runnable接口实现run函数

Socket Client;

**public** Server\_ListenThread(Socket Client){

**this**.Client=Client;

}

**public** **void** run() {

**try** {

InputStream ios = **this**.Client.getInputStream();

BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(ios));

String tem =reader.readLine();

**while**(tem!=**null**){

textAreas[0].append("Form Client:"+tem+'\n');

tem=reader.readLine();

}

reader.close();

} **catch** (Exception e1) {

e1.printStackTrace();

}

}

}

**class** Server\_Speaker **implements** Runnable{

Socket Client;

**public** Server\_Speaker(Socket Client){

**this**.Client=Client;

}

**public** **void** run() {

**try** {

OutputStream os = Client.getOutputStream();

BufferedWriter writer = **new** BufferedWriter(**new** OutputStreamWriter(os));

String message = textFields[1].getText();

writer.write(message);

writer.newLine();

writer.flush();

}**catch** (Exception e){

e.printStackTrace();

}

}

}

}

**五、实验结论、分析、思考题与心得体会**

本实验让我明白了网络通信中服务器端 是如何与客户端建立起连接的，

也懂得了多线程对程序的好处

还学会了用GUI界面化编程，让界面显得不再那么枯燥