# 实验3 TCP网络编程

**一、实验目的**

1、熟悉TCP网络编程的一般操作。

2、理解多线程的作用。

3、理解ArrayList集合类的作用。

4、理解事件处理机制。

1. 了解swing编程。

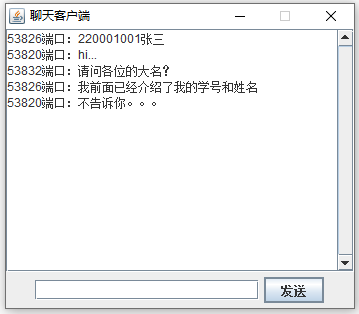
**二、实验报告要求**

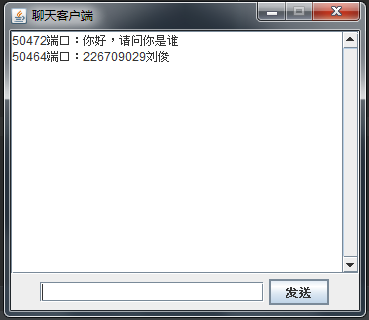
1、实验报告提交word文档，命名为“学号+姓名+实验3”。

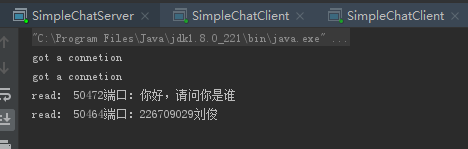
2、本次实验共3题

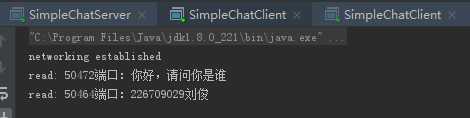
（1）第1、2题：给出**完整代码**，以及服务器和客户端的**运行结果**。

（2）第3题：给出服务器、以及客户端的**运行结果**。客户端的运行截图中，应展示至少两个客户端之间的对话，并在第一条聊天记录中出现本人的学号、姓名。









1. 可以附上调试程序的过程中，出现的问题、以及解决方法。【可选】

**三、实验内容**

1、通过TCP网络编程实现：每次运行客户端，都能够从服务器中获得不同的诗歌的名字。请在【代码一】~【代码六】填写代码。

（1）新建名为”ex3”的项目（Java应用程序）。在项目中，新建名为“mypackage”的包，在该包中新建Java类，命名为“DailyPoetryClient”。在DailyPoetryClient.java文件中，定义客户端类。客户端类的主要功能：打印输出从服务器中读取的诗歌名。

package mypackage;

import java.io.\*;

import java.net.\*;

public class DailyPoetryClient {

public void go() {

try {

Socket s = new Socket("127.0.0.1",5000);

//【代码一】

String poetry = reader.readLine();

System.out.println("Today's Poetry: "+poetry);

reader.close();

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

DailyPoetryClient client = new DailyPoetryClient();

client.go();

}

}

（2）在“mypackage”包中，新建Java类，命名为“DailyPoetryServer”。在DailyPoetryServer.java文件中，定义服务器类。服务器类的主要功能：向客户端随机输出一首诗歌名。

package mypackage;

import java.net.\*;

import java.io.\*;

public class DailyPoetryServer {

String [] poetrys = /\*【代码二】\*/;

/\*数组的内容包括："太阳颂"、"生如夏花"、 "飞鸟集"、 "园丁集"、   
"吉檀迦利"、 "闲暇"以及**学生本人的姓名**\*/

public void go() {

try {

ServerSocket serverSocket = /\*【代码三】\*/;

//【代码四】

PrintWriter writer = new PrintWriter(sock.getOutputStream());

String poetry = getPoetry(); //获取诗歌名

writer.println(poetry);

//【代码五】

System.out.println(poetry);

} catch (Exception e) {

e.printStackTrace();

}

}

//随机返回诗歌名

private String getPoetry() {

int random = (int)(Math.random()\*poetrys.length);

return /\*【代码六】\*/;

}

public static void main(String [] args) {

DailyPoetryServer server = new DailyPoetryServer();

server.go();

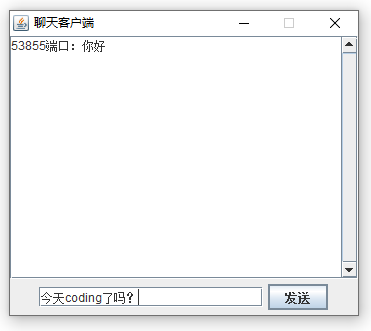
}

}

2、通过多线程编程，改写服务器类DailyPoetryServer，以实现该服务器能同时服务多个客户端。

【说明】新建Java类，命名为“DailyPoetryServer2”。在DailyPoetryServer2.java文件中，定义多线程的服务器类DailyPoetryServer2。

3、创建简单的TCP聊天程序。



（1）新建Java类，命名为“SimpleChatClient”。在SimpleChatClient.java文件中，定义客户端类。客户端类的主要功能：实现聊天界面，当用户在文本框中输入一行文本，单击“发送”按钮，文本信息将传送给服务器。

import java.io.\*;

import java.net.\*;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class SimpleChatClient {

JTextArea incoming;

JTextField outgoing;

BufferedReader reader;

PrintWriter writer;

Socket socket;

public static void main(String[] args) {

new SimpleChatClient().go();

}

public void go() {

JFrame frame = new JFrame("聊天客户端");

JPanel mainPanel = new JPanel();

JPanel panel = new JPanel();

incoming = new JTextArea(15, 30);

incoming.setLineWrap(true);//设置在行过长的时候是否要自动换行

incoming.setWrapStyleWord(true);//设置在单词过长的时候是否要把长单词移到下一行。

incoming.setEditable(false);

JScrollPane qScrollPane = new JScrollPane(incoming);

qScrollPane.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL\_SCROLLBAR\_ALWAYS);

qScrollPane.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL\_SCROLLBAR\_NEVER);

outgoing = new JTextField(20);

JButton sendButton = new JButton("发送");

//设置按钮的监听器

sendButton.addActionListener(new SendButtonListener());

panel.add(outgoing);

panel.add(sendButton);

mainPanel.setLayout(new BoxLayout(mainPanel, BoxLayout.Y\_AXIS));

mainPanel.add(qScrollPane);

mainPanel.add(panel);

frame.getContentPane().add(BorderLayout.CENTER, mainPanel);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

//窗体关闭，则退出应用程序

frame.setResizable(false); //禁止改变窗体大小

frame.pack();

frame.setVisible(true);

frame.getRootPane().setDefaultButton(sendButton);

//设置在窗口中按下回车键时，单击了sendButton

setUpNetWoking();//连接服务器

//创建一个线程，用于接收服务器的消息

Thread readerThread = new Thread(new IncomingReader());

readerThread.start();

}

//将“发送”按钮的事件处理类定义为内部类

public class SendButtonListener implements ActionListener {

public void actionPerformed(ActionEvent ev) {

//请手动输入代码

try {

writer.println(outgoing.getText());

writer.flush();

} catch (Exception e) {

e.printStackTrace();

}

outgoing.setText("");

outgoing.requestFocus();

}

}

private void setUpNetWoking() {

//请手动输入代码

try {

socket = new Socket("127.0.0.1", 5000);

InputStreamReader streamReader = new InputStreamReader(socket.getInputStream());

reader = new BufferedReader(streamReader);

writer = new PrintWriter(socket.getOutputStream());

System.out.println("networking established");

} catch (Exception e) {

e.printStackTrace();

}

}

//将接收服务器端消息的线程定义为内部类

public class IncomingReader implements Runnable {

//请手动输入代码

public void run(){

String message;

try {

while ((message = reader.readLine()) != null){

System.out.println("read: " + message);

incoming.append(message + "\n");

}

} catch (IOException e) {

e.printStackTrace();

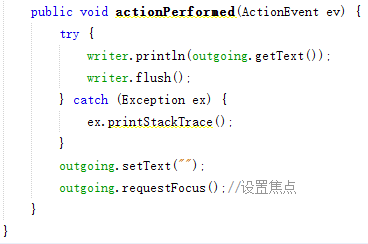
}

}

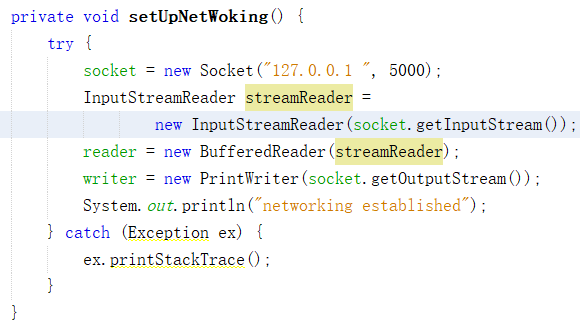
}

}

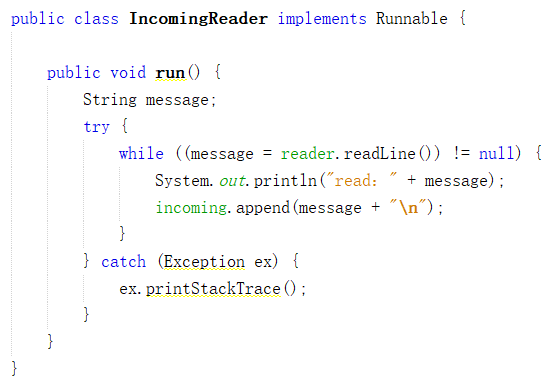
public void actionPerformed((ActionEvent ev)的实现代码：



setUpNetWoking()的实现代码：



IncomingReader类的实现代码如下：



（2）新建Java类，命名为“SimpleChatServer”。在SimpleChatServer.java文件中，定义服务器类。服务器的主要功能：将接收到的信息，群发给所有的客户端。

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class SimpleChatServer {

ArrayList<PrintWriter> clientOutputStreams;

public static void main(String[] args) {

new SimpleChatServer().go();

}

public void go() {

//请手动输入代码

clientOutputStreams = new ArrayList<>();

try {

ServerSocket serverSocket = new ServerSocket(5000);

while (true){

Socket clientSocket = serverSocket.accept();

PrintWriter writer = new PrintWriter(clientSocket.getOutputStream());

clientOutputStreams.add(writer);

Thread t = new Thread(new ClientHandler(clientSocket));

t.start();

System.out.println("got a connetion");

}

} catch (IOException e) {

e.printStackTrace();

}

}

public class ClientHandler implements Runnable {

BufferedReader reader;

Socket socket;

public ClientHandler(Socket clientSocket) {

//请手动输入代码

try {

socket = clientSocket;

InputStreamReader isReader = new InputStreamReader(socket.getInputStream());

reader = new BufferedReader(isReader);

} catch (Exception e) {

e.printStackTrace();

}

}

public void run() {  
 String message;  
 try {  
 while ((message = reader.readLine()) != null) {  
 String printMessage = socket.getPort() + "端口："+message;  
 System.out.println("read： " + printMessage);  
 tellEveryone(printMessage);  
 }  
 } catch (Exception ex) {  
 //ex.printStackTrace();  
 }  
}

}

public void tellEveryone(String message) {

//请手动输入代码

Iterator<PrintWriter> it = clientOutputStreams.iterator();

while (it.hasNext()){

try {

PrintWriter writer = it.next();

writer.println(message);

writer.flush();

} catch (Exception e) {

e.printStackTrace();

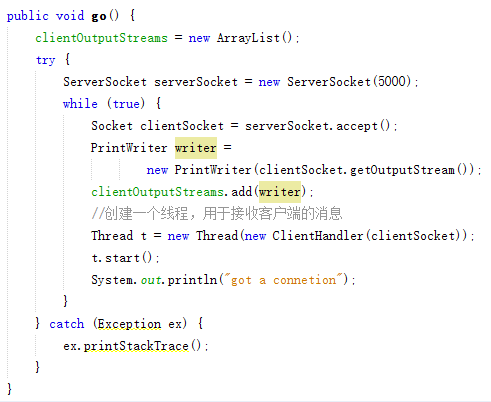
}

}

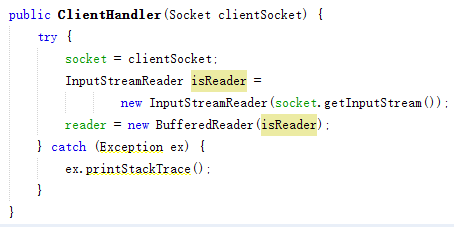
}

}

public void go()的实现代码：



public ClientHandler(Socket clientSocket)的定义：



public void tellEveryone(String message)的定义：

