**Socket**编程基础——

航班显示系统客户端的设计与实现

姓 名 梁寒冰 学号170341317

小组编号 28 小组其他成员的学号170341301 170341306 170341308 170341331

实验日期 2019年10月29日 实验报告日期2019年11月9日

成 绩 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. 实验目的

1、 理解基于TCP的网络连接、套接字的基本概念

2、 熟悉并熟练掌握Java中Socket类的使用

3、 熟练掌握基于TCP的网络客户端的数据读写操作和客户端软件设计

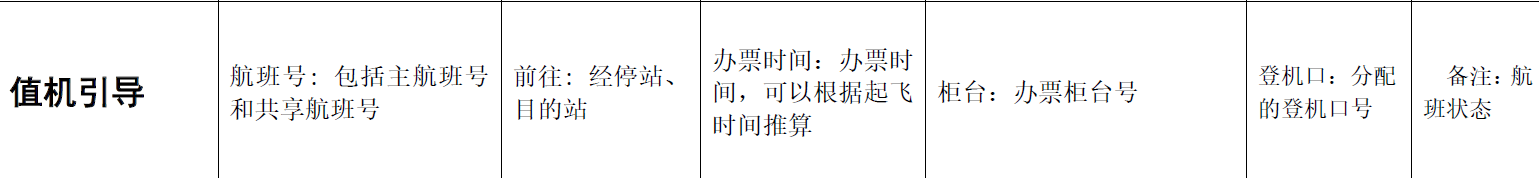
1. 实验环境

JDK，eclipse

文件里./src/ex/Ex3为本次实验内容。

1. 实验实际完成内容及结果分析(请先说明你在小组中所承担的任务)

负责值机引导：



1、 本次实验主要是要综合运用Socket 技术、GUI 和多线程技术，在完成实验二本地航班数据动态显示的软件基础上，进一步设计和开发网络版的航班动态显示系统的客户端程序，

2. 连接航班信息服务器（IP：10.5.25.193，端口：9999），获取航班的动态信息。航班信息服务器为每一个连接的客户端程序提供动态的航班数据，服务器发送的第一条信息是：共有18018 条航班数据待处理，然后服务器每隔100 毫秒发送一条新的航班数据，直到所有航班数据发送完毕（信息传输结束标志是：“no data!”），发送的航班数据同实验二

4.程序说明：程序首先封装所需数据成员为类，定义类对象数组Flight，循环对每一行数据进行筛选并赋给类对象相应的值，在选择过程中首先选择离港信息，选择后将其放入数组传给process并清屏，输出数据。

源码：

package ex;

import java.awt.BorderLayout;

import java.awt.EventQueue;

import java.beans.VetoableChangeListener;

import java.io.IOException;

import java.net.Socket;

import java.net.UnknownHostException;

import java.util.ArrayList;

import java.util.Collection;

import java.util.Collections;

import java.util.Hashtable;

import java.util.List;

import java.util.Scanner;

import java.util.Set;

import java.util.Vector;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.JScrollPane;

import javax.swing.JButton;

import javax.swing.JTable;

import javax.swing.SwingWorker;

import javax.swing.table.DefaultTableModel;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

public class Ex3 extends JFrame {

private JPanel contentPane;

private JTable table;

private DefaultTableModel defaultTableModel;

private ReceiveAndProcessData receiveAndProcessData;

Hashtable<String, airPlane> hashtable = new Hashtable<>();

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

Ex3 frame = new Ex3();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

}

);

}

/\*\*

\* Create the frame.

\*/

public Ex3() {

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 548, 356);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

contentPane.setLayout(new BorderLayout(0, 0));

setContentPane(contentPane);

JPanel panel = new JPanel();

contentPane.add(panel, BorderLayout.NORTH);

JButton button = new JButton("启动");

button.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

receiveAndProcessData = new ReceiveAndProcessData(hashtable);

receiveAndProcessData.execute();

}

});

panel.add(button);

table = new JTable();

defaultTableModel=new DefaultTableModel(

new Object[][] {

},

new String[] {

"航班号", "共享航班","经停站","目的站","办票时间","值机柜台","登机口","航班状态"

}

);

table.setModel(defaultTableModel);

JScrollPane scrollPane = new JScrollPane(table);

contentPane.add(scrollPane, BorderLayout.CENTER);

}

class ReceiveAndProcessData extends SwingWorker<String, ArrayList<airPlane>> {

Hashtable<String, airPlane> displayDataVecotr;

public ReceiveAndProcessData(Hashtable<String,airPlane> displayDataVecotr) {

// TODO 自动生成的构造函数存根{

this.displayDataVecotr = displayDataVecotr;

}

@Override

protected String doInBackground() throws Exception {

String line="";

try(Socket socket=new Socket("10.5.25.193",9999);

Scanner scanner=new Scanner(socket.getInputStream())){

System.out.println("服务器连接成功！！！");

while(!(line = scanner.nextLine()).equals("no data!"))

{

String ch = "";

String flno = "";

String ckno = "";

String[] tt = null;

//航班标识

Pattern p1 = Pattern.compile("(?<=ffid=).\*?(?=\\,)");

Matcher m1 = p1.matcher(line);

if(m1.find())

{

tt = m1.group().split("-");

ch = tt[3];

}

if(ch.equals("D"))

{

flno = tt[0]+tt[1];

airPlane Fight = new airPlane();

Fight.planeflno = flno;

//共享航班号

Pattern p2 = Pattern.compile("(?<=sfno=).\*?(?=\\,)");

Matcher m2 = p2.matcher(line);

if(m2.find())

{

if(!m2.group().equals("null"))

Fight.planesfno = m2.group();

}

//经停

Pattern p4 = Pattern.compile("(?<=arno=2, apcd=).\*?(?=\\,)");

Matcher m4 = p4.matcher(line);

if(m4.find())

{

if(!m4.group().equals("null"))

Fight.planearno2 = m4.group();

}

//目的

Pattern p5 = Pattern.compile("(?<=arno=3, apcd=).\*?(?=\\,)");

Matcher m5 = p5.matcher(line);

if(m5.find())

{

if(!m5.group().equals("null"))

Fight.planearno3 = m5.group();

}

//办票时间

Pattern p6 = Pattern.compile("(?<=fett=).\*?(?=\\,)");

Matcher m6 = p6.matcher(line);

if(m6.find())

{

if(!m6.group().equals("null"))

Fight.planefett = m6.group();

}

//柜台

Pattern p7\_ckno = Pattern.compile("(?<=ckno=).\*?(?=\\])");

Matcher m7\_ckno = p7\_ckno.matcher(line);

while(m7\_ckno.find())

{

Pattern p7\_code = Pattern.compile("(?<=code=).\*?(?=\\,)");

Matcher m7\_code = p7\_code.matcher(line);

if(m7\_code.find())

{

if(!m7\_code.group().equals("null"));

ckno = ckno + m7\_code.group() + ",";

}

}

Fight.planeckno = ckno;

//登机口

Pattern p8\_gtls = Pattern.compile("(?<=DFME\_GTLS\\[).\*?(?=\\])");

Matcher m8\_gtls = p8\_gtls.matcher(line);

if(m8\_gtls.find())

{

Pattern p8\_code = Pattern.compile("(?<=code=).\*?(?=\\,)");

Matcher m8\_code = p8\_code.matcher(m8\_gtls.group());

if(m8\_code.find())

{

Fight.planegtls = m8\_code.group();

}

}

//航班状态

Pattern p9 = Pattern.compile("(?<=ista=).\*?(?=\\,)");

Matcher m9 = p9.matcher(line);

if(m9.find())

{

Fight.planeitsa = m9.group();

}

if(displayDataVecotr.containsKey(flno))

{

airPlane newplane = displayDataVecotr.get(flno);

if(!Fight.planesfno.equals("")) {

newplane.planesfno = Fight.planesfno;

}

if(!Fight.planearno2.equals("")) {

newplane.planearno2 = Fight.planearno2;

}

if(!Fight.planearno3.equals("")) {

newplane.planearno3 = Fight.planearno3;

}

if(!Fight.planefett.equals("")) {

newplane.planefett = Fight.planefett;

}

if(!Fight.planeckno.equals("")) {

newplane.planeckno = Fight.planeckno;

}

if(!Fight.planegtls.equals("")) {

newplane.planegtls = Fight.planegtls;

}

if(!Fight.planeitsa.equals("")) {

newplane.planeitsa = Fight.planeitsa;

}

displayDataVecotr.replace(flno, newplane);

}

else {

displayDataVecotr.put(flno, Fight);

}

}

ArrayList<airPlane> ff = new ArrayList<airPlane>();

Set<String> flidSet=displayDataVecotr.keySet();

if(flidSet.size()!= 0)

{

for(String ss:flidSet) {

ff.add(displayDataVecotr.get(ss));

}

publish(ff);

}

}

}

return null;

}

@Override

protected void process(List<ArrayList<airPlane>> chunks) {

// TODO 自动生成的方法存根

super.process(chunks);

for(ArrayList<airPlane> v:chunks){

defaultTableModel.setRowCount(0);

for(airPlane flight:v) {

defaultTableModel.addRow(new String[] {

flight.planeflno,flight.planesfno,flight.planearno2,flight.planearno3,

flight.planefett,flight.planeckno,flight.planegtls,flight.planeitsa

});

}

}

}

}

}

1. 思考题
2. 什么是Socket？

网络上的两个程序通过一个双向的通信连接实现数据的交换，这个连接的一端称为一个socket。建立网络通信连接至少要一对端口号(socket)。socket本质是编程接口(API)，对TCP/IP的封装，TCP/IP也要提供可供程序员做网络开发所用的接口，这就是Socket编程接口。

1. 客户端与服务器的通信过程是怎样的？

(1) 连接 当客户在浏览器上键入或单击一个URL时，客户程序就在网络上搜索定位正确的服务器，并与其连接。

(2) 请求 在连接后，客户程序向服务器发出一个请求，要求服务器向自己发送指定的信息。

(3) 响应 服务器收到请求，就向客户端传回所需要的文件。

(4)关闭 服务器发出数据后，就断开与客户端的连接，以便接受其他客户的连接。