实验五**UDP**通信——

基于**UDP**的航班显示系统的设计与实现

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12月20日 实验报告日期2019年12月20日

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

1. 实验目的

1、 理解基于 UDP 的 网络程序的通信原理和工作过程

2、 熟悉并熟练掌握 Java 中 DatagramSocket 、 DatagramPacket 类的使用

3、 熟练掌握基于 UDP 的网络应用程序的设计及创建

二、

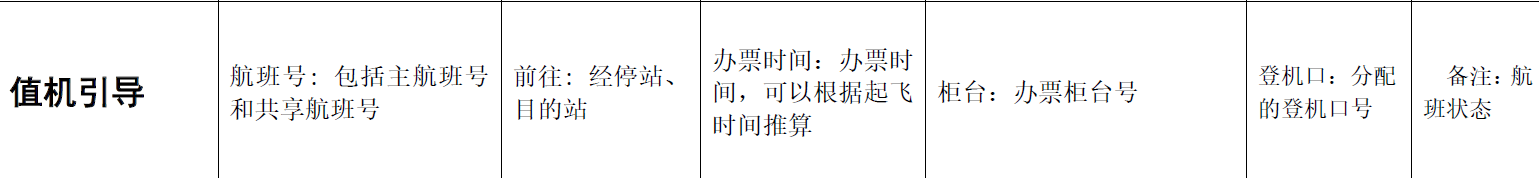
1. 实验环境

JDK，eclipse

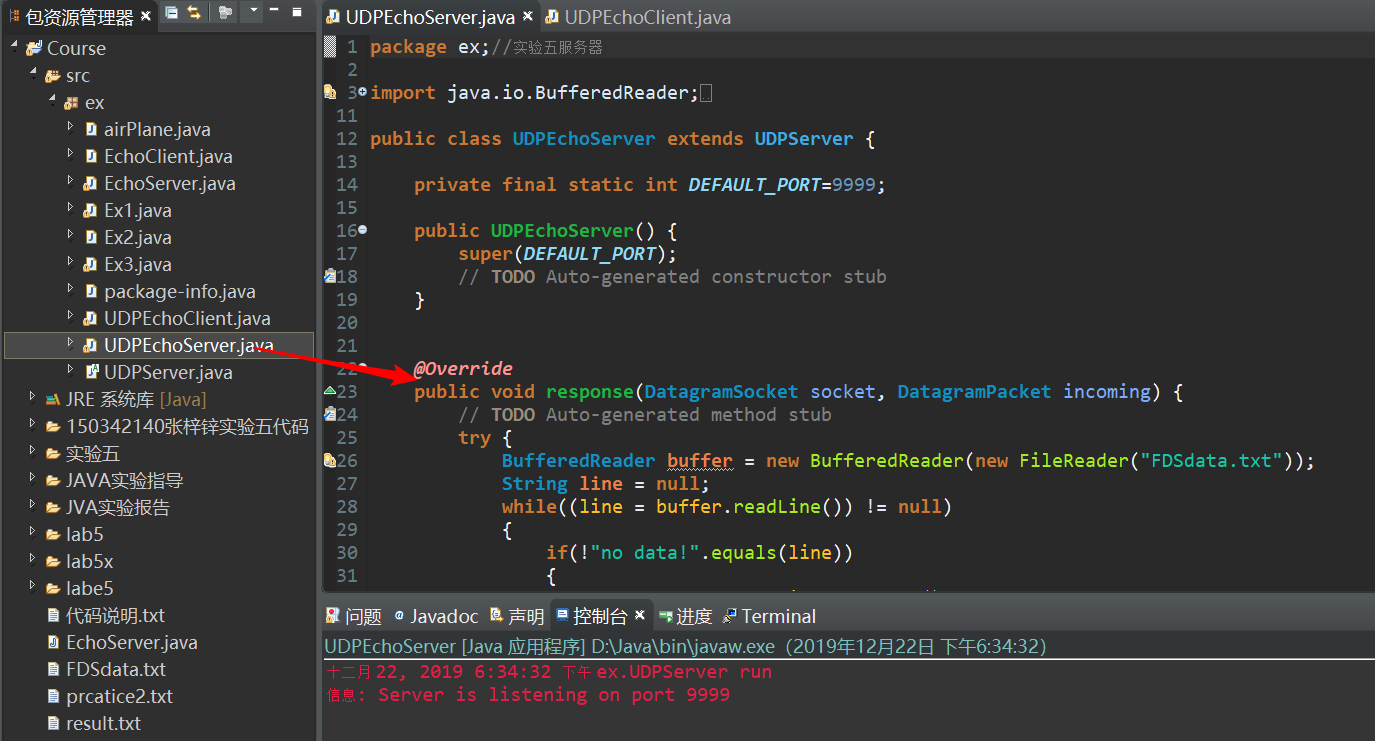
文件里./src/ex/UDPEchoSever和UDPEchoClient为本次实验内容。

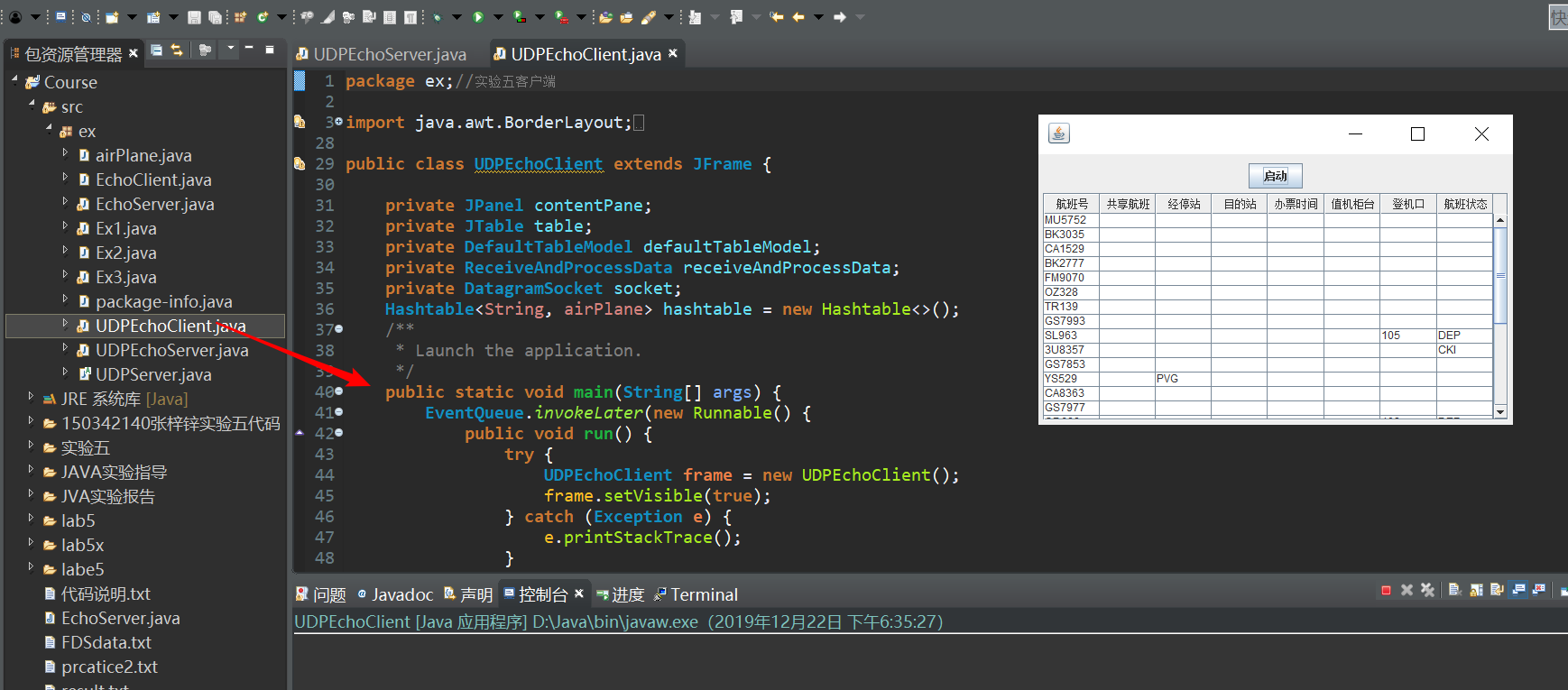
1. 实验实际完成内容及结果分析(请先说明你在小组中所承担的任务)

负责值机引导：



1. 本次实验主要是要综合运用，在完成实验三、实验四航班动态显示系统的服务器和客户端程序的基础上，设计并开发基于UDP通信机制的航班显示系统（包括服务器和客户端）的应用软件。
2. 给出基于 UDP 的航显系统的设计方案（包括设计思路，类结构和对象模型）创建应用服务器的监听端口（ UDP 端口： 9999 ），从 航班信息文本文件fdsdata.txt 获取航班的动态信息，并向已 知航显终端提供动态的航班数据，发送信息数据要求同课内实验四（建议一行数据发送一个 UDP数据报）





源码：

**UDPEchoSever:**

package ex;//实验五服务器

import java.io.BufferedReader;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.util.Scanner;

public class UDPEchoServer extends UDPServer {

private final static int DEFAULT\_PORT=9999;

public UDPEchoServer() {

super(DEFAULT\_PORT);

// TODO Auto-generated constructor stub

}

@Override

public void response(DatagramSocket socket, DatagramPacket incoming) {

// TODO Auto-generated method stub

try {

BufferedReader buffer = new BufferedReader(new FileReader("FDSdata.txt"));

String line = null;

while((line = buffer.readLine()) != null)

{

if(!"no data!".equals(line))

{

byte[] outData = line.getBytes();

incoming = new DatagramPacket(outData, outData.length,incoming.getAddress(),incoming.getPort());

socket.send(incoming);

Thread.sleep(50);

}

else {

break;

}

}

String receiveData = new String(incoming.getData(),0,incoming.getLength(),"gb2312");

System.out.println(receiveData);

socket.send(incoming);

} catch (IOException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

} catch (InterruptedException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

}

public static void main(String[] args) {

UDPServer server=new UDPEchoServer();

new Thread(server).start();

}

}

UDPEchoClient:

package ex;//实验五客户端

import java.awt.BorderLayout;

import java.awt.EventQueue;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.net.UnknownHostException;

import java.util.ArrayList;

import java.util.Hashtable;

import java.util.List;

import java.util.Set;

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.io.IOException;

import java.awt.event.ActionEvent;

public class UDPEchoClient extends JFrame {

private JPanel contentPane;

private JTable table;

private DefaultTableModel defaultTableModel;

private ReceiveAndProcessData receiveAndProcessData;

private DatagramSocket socket;

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

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

UDPEchoClient frame = new UDPEchoClient();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

}

);

}

/\*\*

\* Create the frame.

\*/

public UDPEchoClient() {

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 {

socket=new DatagramSocket();

byte[] outputData="startsend".getBytes();

DatagramPacket outputPacket=new DatagramPacket(outputData,outputData.length,InetAddress.getByName("localhost"),9999);

socket.send(outputPacket); //给EchoServer发送数据报

while(true)

{

String line="";

DatagramPacket inputPacket=new DatagramPacket(new byte[512],512);

socket.receive(inputPacket);

line=new String(inputPacket.getData(),0,inputPacket.getLength());

if(!"no data!".equals(line))

{

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);

}

}

}

else

{

socket.close();

}

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. DatagramSocket、DatagramPacket类的关系？

DatagramPacket与DatagramSocket位于java.net包中。DatagramPacket表示存放数据的数据报，DatagramSocket表示接受或发送数据报的套接字由这两个类所有构成的网络链接是基于UDP协议，是一种不可靠的协议。之所以不可靠是因为发送方不负责数据是否发送成功，接收方收到数据时也不会向发送方反馈成功消息，容易导致信息的丢失

1. 请描述典型UDP网络服务器和客户机的通信过程。
2. 打开通信信道，并连接到服务器在主机的端口，对应UDP进程。
3. 向服务器发出请求报文，等待接收应答
4. 从服务器方收到最终应答
5. 服务器打开通信信道，通知本地主机在某一端口接收请求
6. 等待客户请求
7. 接收请求