**桂林电子科技大学 2019-2020学年 第1学期**

**《Java程序设计》实 验 报 告**

|  |
| --- |
| 指导教师意见：  教 师  签 名  成绩 |

实验名称  **实验七 多线程编程与网络编程**

计算机与信息安全 学院 **计算机类**  专 业

班 第 实 验 小 组

姓 名 奚锐 学 号  **1800300636**

实 验 日 期 2019 年 11 月 12 日

### 实验七 多线程编程与网络编程

### 实验目的

1、 掌握线程和多线程的概念。

2、 掌握创建线程的两种方法及其区别。

3、 了解线程的启动、终止、同步、互斥和优先级等概念。

4、 掌握 TCP 和 UDP 网络程序设计的模型。

5、 掌握 TCP 和 UDP 程序设计的基本方法和所使用的系统类。

6、 了解基于应用层 http 协议的程序的设计以及使用系统类的组播、广播等网络程序设计

### 实验内容

1、 多线程编程题目 1：编写一个程序，其功能是运行之后，其中有一个线程可以输出 20 次你的学 号，另一个线程会输出 20 次你的姓名。

2、 多线程编程题目 2：编写一个图形界面程序，运行之后，让其中有一个线程能在界面上“实时” 显示系统当前时间（精确到秒获取时间可查询 java.util.Calendar 类，它包含了多个用于获得系统时间 的函数） 。另让一个线程可以在界面上提示当前系统时间下用户该做什么工作（例如当程序判断出系统时 间现在是 8：00 到 9：00，则提示用户该上课；现在是 23：00 到 8：00，则提示用户该休息。 具体测试 时可以将时间段限制到秒级，以便及时查看到程序运行中提示信息的变化） 。

3、 网络编程题目 1：先熟悉所给的示例，了解网络编程的一般基本概念。然后两个同学为一组，分 别设计一对控制台下的 TCP 通信程序的客户端和服务器端，然后双方要能够相互发送和反馈信息进行无 限次连续通信，直到其中一方发送表示结束通信的“end”字符串，然后接收方也返回一个“end”，双方 结束通信。然后改进程序，将自己的程序改进为具有图形界面的软件（即设计成一个简单的网络聊天程 序）。可在本地机上做测试，或两个同学为一组在不同的机器上作测试都可以，要将服务器地址和端口号 作为参数，在运行中可以进行修改。对于本机上测试，服务器地址使用 127.0.0.1 或者本机实际 IP 地址。 在网内不同机器上测试，要给出正确的服务器 IP 地址或名称。

4、 网络编程题目 2：观察和分析 QQ 等即时通讯软件的界面和功能，通过查阅资料，思考和分析其 设计思想和实现方法，设计一款自己的通讯软件，实现诸如消息传递、好友管理、系统设置、文件收发 等若干功能。

### 分析与设计

1. 自定义一个类mythread继承自Thread并且重写其run方法，在run方法里循环输出一个字符串，再在main方法里创建两个mythread的实例并分别传入名字与学号；分别调用两个实例的start方法即可；
2. 自定义一个线程类，重写run方法，放置一个无限循环的while，每500ms检查一次当前时间（使用Calendar），如果过了9:00就提醒学习，过了23:00就提醒休息。实际可以缩短到10秒一次进行一次转换。
3. 写一个服务器端与客户端，服务器负责监听指定端口的TCP链接，当TCP连接成功时，服务器端开启一个新线程进行socket的信息收发，客户端则发送一个socket连接，当连接成功时时，也开启一个线程负责socket的收发；

问题一代码

、package helloWorld;

class MyThread extends Thread { //通过继承系统提供的Thread类

public String name;

public MyThread() {

//重写无参构造函数，若不需要也可去掉此函数

}

public MyThread(String name){

this.name=name;

}

public void run() {

for (int i = 1; i <= 20; i++) {

System.out.println(name);

// try {

// sleep(500);//睡眠500ms给其它线程运行机会，试试不要会怎样

// } catch (InterruptedException e) {

// }

}

}

}

public class HelloThread {

public static void main(String args[]) {

Thread t1 = new Thread(new MyThread("My name is HaeYe"),"1"); // 创建线程1的对象,并

//通过第二个参数将其命名为thread 1

Thread t2 = new Thread(new MyThread("My number is 112321"),"2"); // 创建线程2的对象,并

//通过第二个参数将其命名为thread 2

t1.start(); //调用start方法启动两个线程运行

t2.start(); //虽然t2的启动表面上好像在后面，实际上两个线程的执行并无先后之分，

//t2的执行甚至有可能在t1之前，看系统的调度情况，可以多执行几次看看

}

}

问题二代码：

package helloWorld;

import javax.swing.\*;

import java.awt.\*;

import java.util.Calendar;

class myclock extends Thread{

JLabel clock;

JLabel remind;

static int time;

public void setClock(JLabel clock) {

this.clock = clock;

}

public void setRemind(JLabel remind) {

this.remind = remind;

}

public void myclock(){

time=10;

}

@Override

public void run() {

while (true){

Calendar timer = Calendar.getInstance();

String t=String.valueOf(timer.get(Calendar.HOUR\_OF\_DAY)) + ":" + String.valueOf(timer.get(Calendar.MINUTE)) + ":" + String.valueOf(timer.get(Calendar.SECOND));

if(!t.equals(clock.getText())){

time--;

System.out.println(time);

if(time<0){

String t2=remind.getText();

if(remind.getText().equals("该学习了")){

remind.setText("该休息了");

}

else{

remind.setText("该学习了");

}

time=10;

}

}

clock.setText(t);

try {

Thread.sleep(100);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

public class CalenderReminder {

private JPanel panel1;

private JLabel Time;

private JLabel Reminder;

public static void main(String []args){

JFrame frame=new JFrame();

CalenderReminder myreminder =new CalenderReminder();

frame.setContentPane(myreminder.panel1);

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

frame.setSize(400,300);

frame.setVisible(true);

myreminder.Reminder.setBackground(new Color(0xA34228));

frame.repaint();

myclock a=new myclock();

a.setClock(myreminder.Time);

a.setRemind(myreminder.Reminder);

a.start();

}

}

问题三代码：

服务器端：

package helloWorld;

import javax.swing.\*;

import java.awt.\*;

import java.util.Calendar;

class myclock extends Thread{

JLabel clock;

JLabel remind;

static int time;

public void setClock(JLabel clock) {

this.clock = clock;

}

public void setRemind(JLabel remind) {

this.remind = remind;

}

public void myclock(){

time=10;

}

@Override

public void run() {

while (true){

Calendar timer = Calendar.getInstance();

String t=String.valueOf(timer.get(Calendar.HOUR\_OF\_DAY)) + ":" + String.valueOf(timer.get(Calendar.MINUTE)) + ":" + String.valueOf(timer.get(Calendar.SECOND));

if(!t.equals(clock.getText())){

time--;

System.out.println(time);

if(time<0){

String t2=remind.getText();

if(remind.getText().equals("该学习了")){

remind.setText("该休息了");

}

else{

remind.setText("该学习了");

}

time=10;

}

}

clock.setText(t);

try {

Thread.sleep(100);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

public class CalenderReminder {

private JPanel panel1;

private JLabel Time;

private JLabel Reminder;

public static void main(String []args){

JFrame frame=new JFrame();

CalenderReminder myreminder =new CalenderReminder();

frame.setContentPane(myreminder.panel1);

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

frame.setSize(400,300);

frame.setVisible(true);

myreminder.Reminder.setBackground(new Color(0xA34228));

frame.repaint();

myclock a=new myclock();

a.setClock(myreminder.Time);

a.setRemind(myreminder.Reminder);

a.start();

}

}

客户端：

package helloWorld;

import java.io.\*;

import java.net.Socket;

import java.util.Scanner;

public class IClient {

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

Socket socket1=new Socket("10.33.0.142",2333); //2288端口发起连接

try {

System. out .println("connection to server accepted:"+socket1); //连接成功，输出信息

BufferedReader in =new BufferedReader(new InputStreamReader(socket1.getInputStream()));

receiveThread newthread =new receiveThread(in);

newthread.start();

PrintWriter out=new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket1.getOutputStream())),true); //获取对输出流（发送 数据）的控制 //下面向服务器输出数据

out.println("this is the data from client"); //向服务器端发送一串数据

String dialog=null;

Scanner scan=new Scanner(System.in);

do {

dialog=scan.next();

out.println(dialog);

}while(dialog!="iclose");

}

finally {

System. out .println("client closing socket");

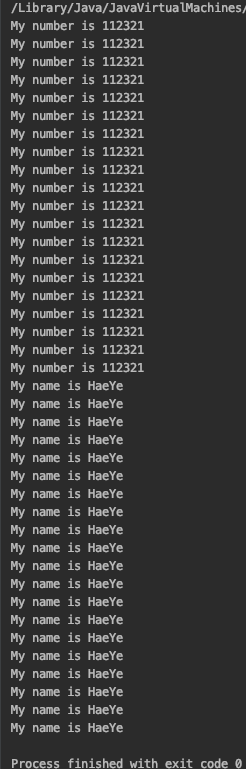
socket1.close();

}

}

}

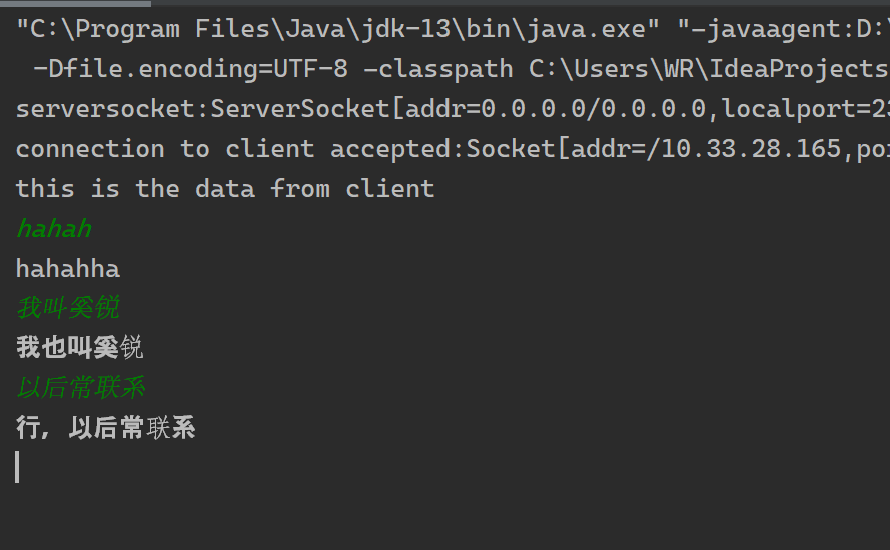
### 测试数据及运行结果



问题1运行结果



问题二运行结果



问题三服务器端



问题三客户端

### 问题与总结

掌握线程和多线程的概念，创建线程的两种方法及其区别。了解了线程的启动、终止、同步、互斥和优先级等概念。掌握 TCP 和 UDP 网络程序设计的模型 TCP 和 UDP 程序设计的基本方法和所使用的系统类。了解基于应用层 http 协议的程序的设计以及使用系统类的组播、广播等网络程序设计