**面向对象程序设计实验**

**实验项目名称: 实验八 多线程**

**学生姓名: 范少帅 学号： 20181101047 班级： 18级计科3班**

**提交时间: 2019/12/4 指导教师: 司建辉**

格式说明: 正文 1.25倍行距 小四 宋体

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

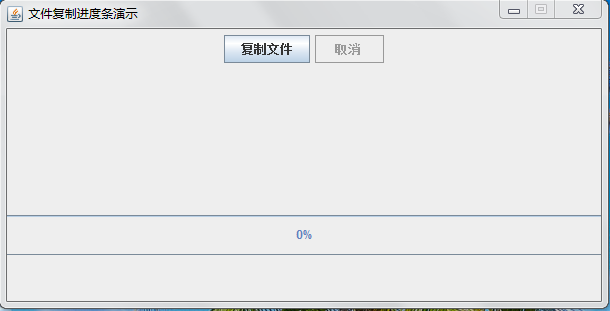
**（一）实验目的**

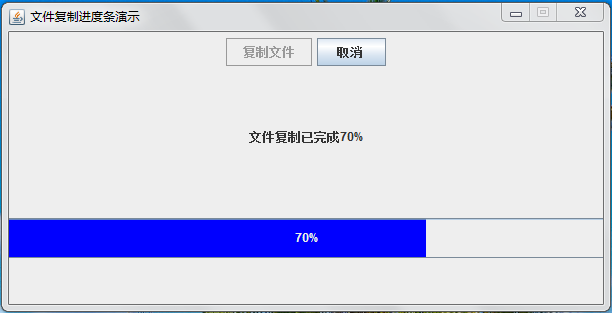
1、掌握多线程的编程方法。

2、掌握线程的启动、同步、等待和唤醒。

**（二）实验内容和实验原理**

1、利用线程模拟文件复制的进度显示，如下图所示，点击“复制文件”，开始显示进度条，并实时显示复制进度，复制完成后显示“文件复制已完成！”，点击“取消”，取消复制，进度条归零，点击“×”可以退出系统，注意复制按钮和取消按钮在同一时间，只有一个可用。

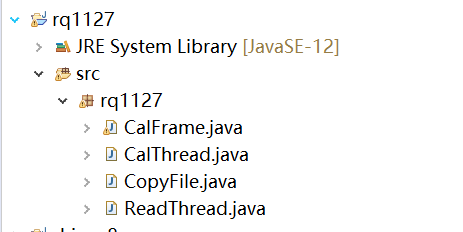


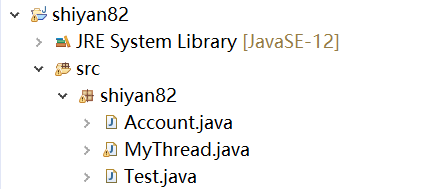


2、模拟一个银行存款的程序。假设有三个储户都去银行往同一个账户进行存款，一次存100，每人存三次。要求储户每存一次钱，账户余额增加100，并在控制台输出当前账户的余额。

**（三）实验步骤**

1．在MyEclipse环境中，新建工作区和类；





2．编辑并编译源程序；

3、运行。

**（四）实验报告要求**

1、提交实验源程序；

package rq1127;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class CopyFile extends JFrame implements ActionListener

{

private static final long serialVersionUID = 1L;

JButton button\_on,button\_off,button\_wait;

JLabel jbl\_show;

JProgressBar jpbar;

CopyFileThread copyfilethread;

public CopyFile ()

{

super("文件复制进度条演示");

this.setBounds(320, 240,460, 200);

this.setDefaultCloseOperation(EXIT\_ON\_CLOSE);

this.getContentPane().setLayout( new GridLayout(3,1));

this.setResizable(false);

JPanel jp1=new JPanel();

button\_on = new JButton("复制文件");

button\_on.addActionListener(this);

jp1.add(button\_on);

button\_wait = new JButton("暂停");

button\_wait.addActionListener(this);

button\_wait.setEnabled(false);

// jp1.add(button\_wait);

//this.getContentPane().add(jp1);

button\_off = new JButton("取消");

button\_off.addActionListener(this);

button\_off.setEnabled(false);

jp1.add(button\_off);

this.getContentPane().add(jp1);

JPanel jp2=new JPanel();

jbl\_show=new JLabel();

jp2.add(jbl\_show);

this.getContentPane().add(jp2);

JPanel jp3=new JPanel();

jpbar=new JProgressBar();

jpbar.setOrientation(JProgressBar.HORIZONTAL);

jpbar.setMinimum(0);

jpbar.setMaximum(100);

jpbar.setValue(0);

jpbar.setStringPainted(true);

jpbar.setForeground(Color.green);

jpbar.setPreferredSize(new Dimension(600,40));

jpbar.setBorderPainted(true);

jpbar.setIndeterminate(false);

jp3.add(jpbar);

this.getContentPane().add(jp3);

this.setVisible(true);

}

@SuppressWarnings("deprecation")

public void actionPerformed(ActionEvent e)

{

if(e.getSource()==button\_on)

{

copyfilethread=new CopyFileThread(jpbar,button\_on,jbl\_show);

copyfilethread.start();

button\_on.setEnabled(false);

button\_off.setEnabled(true);

}

if(e.getSource()==button\_off)

{

copyfilethread.stop();

button\_on.setEnabled(true);

button\_off.setEnabled(false);

jbl\_show.setText("");

jpbar.setValue(0);

}

}

class CopyFileThread extends Thread

{

JProgressBar jpbar;

JButton jbtn;

JLabel jbl;

CopyFileThread(JProgressBar jpbar,JButton jbtn,JLabel jbl)

{

this.jbl=jbl;

this.jbtn=jbtn;

this.jpbar=jpbar;

}

public void run()

{

for(int i=jpbar.getMinimum();i<=jpbar.getMaximum();i++)

{

try

{

jbl.setText("文件剩余"+(100-i)+"%");

jpbar.setValue(i);

Thread.sleep((int)Math.random()\*6 + 100);

}

catch (Exception e)

{

e.printStackTrace();

}

}

jbl.setText("文件已复制完成");

}

}

public static void main(String[] args)

{

new CopyFile();

}

}

package shiyan82;

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

/\*银行有一个账户。有两个储户分别向同一个账户存3000元，每次存1000，存3次。每次存完打印账户余额。注意：效果要是交替打印！(多线程实现)\*/

class cun implements Runnable{

int sum=0;

int a = 0;

int b =0;

int c = 0;

Lock loc = new ReentrantLock();

static boolean bool=true;

public void run(){

for(int i=1;i<=6;i++){

synchronized(this){

notify();//先唤醒另一个线程

try {

Thread.sleep(100);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

if(sum<900){

sum+=100;

if(Thread.currentThread().getName().equals("账户一")){

a=100;

System.out.println(Thread.currentThread().getName()+"存了"+a);

System.out.println("目前余额"+sum);

}

else if(Thread.currentThread().getName().equals("账户二"))

{

b=100;

System.out.println(Thread.currentThread().getName()+"存了"+b);

System.out.println("目前余额"+sum);

}

else

{

c=100;

System.out.println(Thread.currentThread().getName()+"存了"+c);

System.out.println("目前余额"+sum);

}

}

try {

wait();//打印完后等待下一次被唤醒

}

catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

}

}

public class Test {

public static void main(String[] args) {

cun c = new cun();

Thread t1 = new Thread(c,"账户一");

Thread t2 = new Thread(c,"账户二");

Thread t3 = new Thread(c,"账户三");

t1.start();

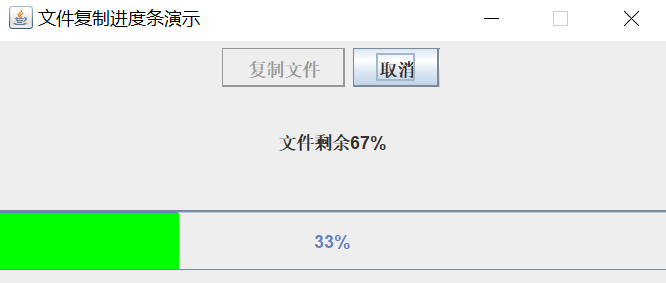
t2.start();

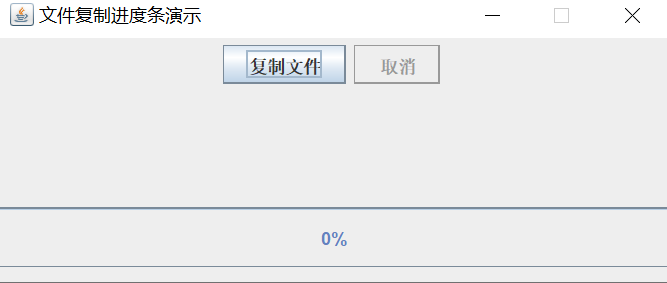
t3.start();

}

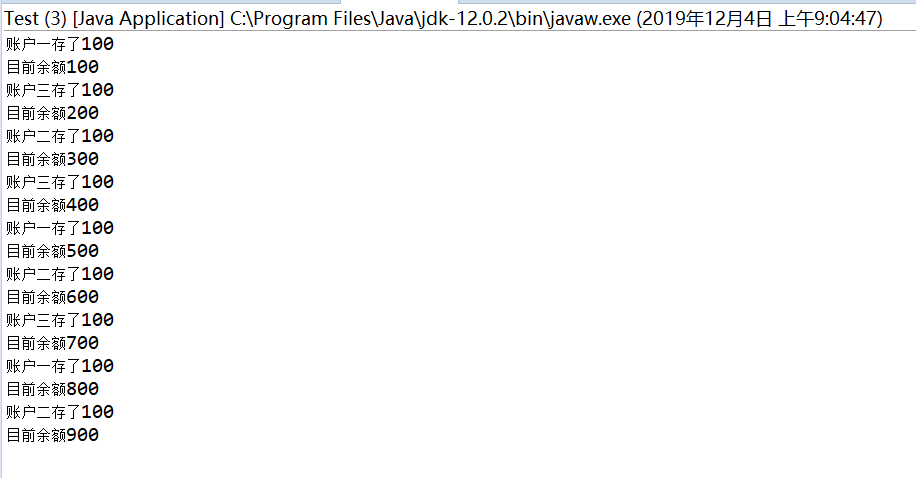
}

2、提交实验过程的截图；





2、



3、撰写心得体会。

多线程可以把任务分块执行，分块后可以同时进行而不用等待，这样效率更高。如下载文件，通过多线程就可以实现多文件下载。并且多线程可以模拟多块CPU的运行方式，因此，使用多线程是提高程序执行效率的最廉价的方式。