**《面向对象程序设计(Java)》**

**实验报告**

姓 名 石家晖

学 号 2019081238

学 院 软件工程学院

专 业 软件工程

年 级 2019级

指导教师 文立玉

**成都信息工程大学软件工程学院**

**2020年9月**

# 实验8：

1. **实验目的**

1. 掌握线程的创建方法。

2. 熟悉线程的启动等方法的使用

3. 了解线程同步

4. 掌握synchronized关键字的使用

1. **实验要求**

1. 实验在4学时内完成。

2. 1人1组独立完成。

1. **实验设备与环境**

JDK13, NEOVIM，Linux

1. **设计思路与具体实现**

1. 建立三个线程，并且同时运行它们。当运行时输出线程的名称。

(1)设计思路

首先创建主类，使主类继承Thread类 实现run方法，run方法负责输出线程名称

(2)实验步骤

①．创建一个Java类

②．在类中创建三个线程，调用start()方法启动这三个线程

③．保存文件，调试并编译运行程序。

(3)关键代码

public static void main (String[] args) {

Thread t1 = new MyThread ("thread 1"); //创建三个线程

Thread t2 = new MyThread ("thread 2");

Thread t3 = new MyThread ("thread 3");

t1.start();//调用start()方法启动这三个线程

t2.start();

t3.start();

}

}

class MyThread extends Thread {

public MyThread (String str) {

super (str);

}

public void run() {

for (int i = 0; i < 3; i++) {

System.out.println ("The thread is:" + Thread.currentThread().getName() );

try {

Thread.sleep ( (int) (Math.random() \* 1000) );

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

(4)程序测试过程

①编译执行程序

②检查输出结果是否与预期相符

(5)程序运行结果(运行效果)



图8-1

2.

(1)设计思路

一个类实现计数 一个类实现打印 一个类实现存储，其中计数和打印继承Thread在主类中创建一个计数和一个打印线程，进行计数和打印

(2)实验步骤

①．创建三个类Counter， Printer，Storage

②．创建主类，在该类中定义main函数，在main函数中定义Storage对象、Counter对象和 Printer对象，创建Counter线程和Printer线程并启动

③．保存文件，调试并编译运行程序。

(3)关键代码

import java.util.\*;

class Storage {

ArrayList<Integer> number=new ArrayList<Integer>();

public void setNum(int num) {

this.number.add(num);

}

public int getNum(int index) {

return number.get(index);

}

}

class Counter extends Thread{

Storage s=new Storage();

int num=0;

public Counter(String str,Storage s) {

super(str);

this.s=s;

}

public void run()

{

while(true) {

try

{

System.out.println("Counter写入"+num);

s.setNum(num);

num++;

Thread.sleep(1000);

}

catch(Exception e)

{

e.printStackTrace();

}

}

}

}

class Printer extends Thread{

Storage s=new Storage();

int count=0;

public Printer(String str,Storage s)

{

super(str);

this.s=s;

}

public void run() {

while(true)

{

if(count<s.number.size())

{

System.out.println("Printer输出"+s.getNum(count));

count++;

try

{

Thread.sleep(1000);

}catch(Exception e)

{

e.printStackTrace();

}

}

}

}

}

public class threadTwo

{

public static void main(String[] args)

{

Storage se=new Storage();

Counter count=new Counter("Counter",se);

Printer print=new Printer("Printer",se);

count.start();

print.start();

}

}

(4)程序测试过程

①编译执行程序

②检查输出结果是否与预期相符

(5)程序运行结果(运行效果)

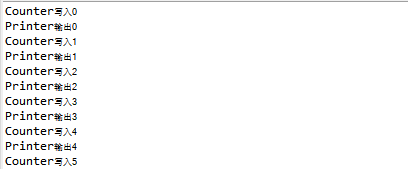


图8-2

3. 修改第2题的程序，添加适当代码，以确保每个数字都恰好只被打印一次。

(1)设计思路

一个类实现计数 一个类实现打印 一个类实现存储，其中计数和打印继承Thread在主类中创建一个计数和一个打印线程，进行计数和打印 在关键部分进行同步处理

(2)实验步骤

①．创建三个类Counter， Printer，Storage

②．创建主类，在该类中定义main函数，在main函数中定义Storage对象、Counter对象和 Printer对象，创建Counter线程和Printer线程并启动

③．保存文件，调试并编译运行程序。

(3)关键代码

import java.util.ArrayList;

class Storage {

ArrayList<Integer> num=new ArrayList<Integer>();

public void setNum(int num)

{

this.num.add(num);

}

public int getNum(int index)

{

return num.get(index);

}

}

class Counter extends Thread{

Storage s=new Storage();

int num=0;

public Counter() {}

public Counter(String str,Storage s) {

super(str);

this.s=s;

}

public void run() {

synchronized(s) {

while(true) {

try {

s.notify();

System.out.println("Counter写入"+num);

s.setNum(num);

num++;

s.wait();

Thread.sleep(1000);

}catch(InterruptedException e) {

e.printStackTrace();

}

}

}

}

}

class Printer extends Thread{

Storage s=new Storage();

int count=0;

public Printer() {}

public Printer(String str,Storage s) {

super(str);

this.s=s;

}

public void run() {

synchronized(s) {

while(true) {

if(count<s.num.size()) {

System.out.println("Printer输出"+s.getNum(count));

count++;

try {

s.notify();

s.wait();

Thread.sleep(1000);

}catch(InterruptedException e) {

e.printStackTrace();

}

}

}

}

}

}

public class threadThree {

public static void main(String[] args) {

// TODO Auto-generated method stub

Storage s=new Storage();

Counter coun=new Counter("Counter",s);

Printer prin=new Printer("Printer",s);

coun.start();

prin.start();

}

}

(4)程序测试过程

①编译执行程序

②检查输出结果是否与预期相符

(5)程序运行结果(运行效果)

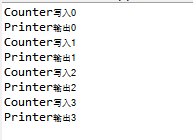


图8-3

1. **实验总结**

通过本次实验，我理解了java中线程的概念和知识，了解了java线程的基本状态分为新建、就绪、运行、阻塞、等待、同步、其他阻塞、死亡状态，并且我掌握了线程的创建方法。熟悉线程的启动等方法的使用，了解了Thread类、Runable接口的不同用法和应用，了解了start方法，了解线程同步掌握synchronized关键字的使用。

1. **附录（程序代码）**

1.

public class Main1 {

public static void main (String[] args) {

Thread t1 = new MyThread ("thread 1"); //创建三个线程

Thread t2 = new MyThread ("thread 2");

Thread t3 = new MyThread ("thread 3");

t1.start();//调用start()方法启动这三个线程

t2.start();

t3.start();

}

}

class MyThread extends Thread {

public MyThread (String str) {

super (str);

}

public void run() {

for (int i = 0; i < 3; i++) {

System.out.println ("The thread is:" + Thread.currentThread().getName() );

try {

Thread.sleep ( (int) (Math.random() \* 1000) );

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

2.

import java.util.ArrayList;

class Storage {

ArrayList<Integer> num = new ArrayList<Integer>();

public void setNum (int num) {

this.num.add (num);

}

public int getNum (int index) {

return num.get (index);

}

}

class Counter extends Thread {

Storage s = new Storage();

int num = 0;

public Counter() {}

public Counter (String str, Storage s) {

super (str);

this.s = s;

}

public void run() {

synchronized (s) {

while (true) {

try {

s.notify();

System.out.println ("Counter写入" + num);

s.setNum (num);

num++;

s.wait();

Thread.sleep (1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

}

class Printer extends Thread {

Storage s = new Storage();

int count = 0;

public Printer() {}

public Printer (String str, Storage s) {

super (str);

this.s = s;

}

public void run() {

synchronized (s) {

while (true) {

if (count < s.num.size() ) {

System.out.println ("Printer输出" + s.getNum (count) );

count++;

try {

s.notify();

s.wait();

Thread.sleep (1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

}

}

public class Main2{

public static void main (String[] args) {

Storage s = new Storage();

Counter coun = new Counter ("Counter", s);

Printer prin = new Printer ("Printer", s);

coun.start();

prin.start();

}

}

3.

import java.util.\*;

class Storage {

ArrayList<Integer> number = new ArrayList<Integer>();

public void setNum (int num) {

this.number.add (num);

}

public int getNum (int index) {

return number.get (index);

}

}

class Counter extends Thread {

Storage s = new Storage();

int num = 0;

public Counter (String str, Storage s) {

super (str);

this.s = s;

}

public void run() {

while (true) {

try {

System.out.println ("Counter写入" + num);

s.setNum (num);

num++;

Thread.sleep (1000);

} catch (Exception e) {

e.printStackTrace();

}

}

}

}

class Printer extends Thread {

Storage s = new Storage();

int count = 0;

public Printer (String str, Storage s) {

super (str);

this.s = s;

}

public void run() {

while (true) {

if (count < s.number.size() ) {

System.out.println ("Printer输出" + s.getNum (count) );

count++;

try {

Thread.sleep (1000);

} catch (Exception e) {

e.printStackTrace();

}

}

}

}

}

public class Main3{

public static void main (String[] args) {

Storage se = new Storage();

Counter count = new Counter ("Counter", se);

Printer print = new Printer ("Printer", se);

count.start();

print.start();

}

}