**Java课程上机报告**

## 上机题目：Java的多线程编程

张俊华 16030199025 (组长)

李金鑫 16030199026

李天浩 16030199027

1. **小组名单**

|  |  |  |
| --- | --- | --- |
| 学号 | 姓名 | 工作 |
| 16030199025 | 张俊华 | 完成程序、练习例程、调试、小组讨论 |
| 16030199026 | 李金鑫 | 完成程序、练习例程、调试、小组讨论 |
| 16030199027 | 李天浩 | 完成程序、练习例程、调试、小组讨论 |

**二、题目**

编写一个程序，程序模拟某电影院三个售票窗口同时出售电影票的过程。

1. 电影票顺序出售，程序模拟显示售票的详细过程（如：“窗口X出售编号XXX电影票”）。
2. 三个窗口同时出票，出票间隔采用随机控制。
3. 不能重复出售相同的电影票。

程序应具有良好的人机交互性能，即：程序应向用户提示功能说明，并可根据用户的功能选择，执行对应的功能，并给出带详细描述信息的最终执行结果。

**三、题目分析**：

**本实验要求编写一个程序，实现以下功能：**

1. 电影票顺序出售，程序模拟显示售票的详细过程（如：“窗口X出售编号XXX电影票”）

2. 三个窗口同时出票，出票间隔采用随机控制。

3. 不能重复出售相同的电影票。

1. 电影票顺序出售，程序模拟显示售票的详细过程（如：“窗口X出售编号XXX电影票”）

设置Conductor（售票员）类，Customer（顾客）类，模拟售票过程。Conductor包含SellaTicket方法，调用该方法，出售一张票，Customer 类包含BuyaTicket方法，该方法，调用指定售票员的SellaTicket。

1. 三个窗口同时出票，出票间隔采用随机控制。

程序初始化时，创建多个Conductor线程，开始出票时，间隔随机时间生成一个Customer线程，Customer随机调用指定Conductor的SellaTicket方法实现多窗口同时出票，随机时间出票

1. 不能重复出售相同的电影票

构建TicketBox（票箱）类，包含synchronized修饰的PushaTicket（出票）方法，售票员通过调用PushaTicket函数获取所得到的票的编号，由于存在对象锁，可以保证在高并发情况下票号不会相同。

**四、程序实现：**

张俊华：

**1．实验环境：**

IntelliJ IDEA 2017.1.2

Build #IU-171.4249.39, built on April 25, 2017

JRE: 1.8.0\_112-release-736-b16 amd64

JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o

Windows 10 10.0

**2．实现过程：**

在开始编写程序前，考虑生活中购票的实际情况，先对程序的整体结构和各个类之间的关系进行设计。定义以下类：

Conductor：售票员，为顾客销售票

Customer：顾客，向指定售票员购买票

TicketBox：票箱，为售票员提供票

Actions: 调用各种方法，程序功能实现

UserInterface：输入输出，与用户进行交互

Conductor继承Thread，实现多线程，完成多窗口同时售票，Conductor对象在程序开始执行时创建，每一个conductor拥有以下属性：

Name：名称

Speed：出票速度

TicketsBox：出票票箱

构造方法：

Conductor(String name,long speed,TicketsBox ticketsBox){  
 this.name = name;  
 this.speed = speed;  
 this.ticketsBox = ticketsBox;  
}

出票方法

*/\*\*  
 \*出售一张票  
 \*****@author*** *张俊华 16030199025  
 \*/*public void SellaTicket(int customerIndex) throws InterruptedException {  
 *sleep*(speed);  
 Outputer.*PrintTickets*(this.name,this.ticketsBox.PushaTicket(),customerIndex);  
  
}

Customer继承Thread，实现多线程，完成多顾客同时买票，Customer对象在程序运行时随机创建，每一个Customer拥有以下属性：

Index：顾客编号

Conductor：顾客选择的售票员对象

构造函数：

Customer(int index, ArrayList<Conductor> conductors) throws InterruptedException {  
 System.*out*.println("迎来了第" +(index)+ "位顾客...");  
 this.index=index;  
 this.conductors = conductors;  
}

买票方法：

public void BuyaTicket(Conductor conductor) throws InterruptedException {  
 conductor.SellaTicket(index);  
}

TicketBox类储存了票总数，有synchronized修饰的Pushaticket方法，调用该方法按顺序返回票的编号

*/\*\*  
 \* Created by 张俊华 on 2017/6/22.  
 \*  
 \** ***@author*** *张俊华.  
 \** ***@Time*** *2017/6/22 16:22.  
 \*/*public class TicketsBox {  
 private int ticketsNumber = 0;  
 private int totalTickets;  
 TicketsBox(int totalTickets){  
 this.totalTickets = totalTickets;  
 }  
 */\*\*  
 \*票箱出一张票  
 \*****@author*** *张俊华 16030199025  
 \*/* public synchronized int PushaTicket(){  
 ticketsNumber++;  
 if (ticketsNumber<=totalTickets){  
 return ticketsNumber;  
 }else {  
 return -1;  
 }  
 }  
}

定义Action类，程序运行时先对Action初始化，Action中有以下字段：

ArrayList<Conductor> conductors 售票员列表

ticketsBox 出票票箱

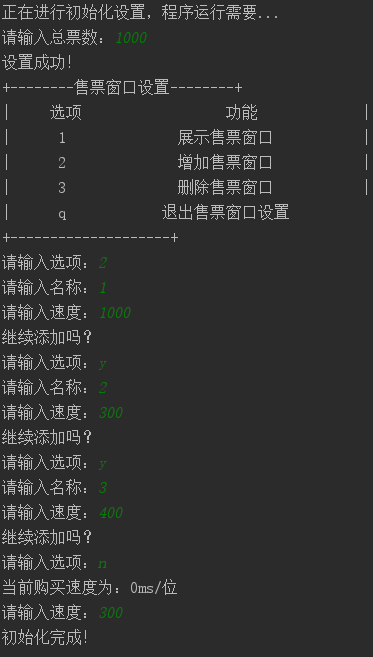
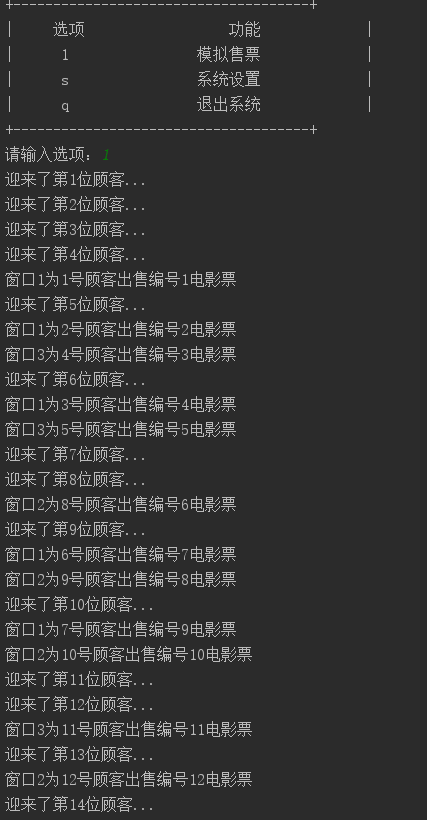
构造方法：

Actions(){  
 System.*out*.println("正在进行初始化设置，程序运行需要...");  
 SetTicketsBox();  
 SetConductors();  
 SetBuyingSpeed();  
 System.*out*.println("初始化完成！");  
}

调用各方法，实现模拟购票：

*/\*\*  
 \* 开始售票  
 \*  
 \** ***@author*** *张俊华 16030199025  
 \*/*public void StartSell() throws InterruptedException {  
 for (Conductor conductor:conductors){  
 conductor.start();  
 }  
 int index = 0;  
  
 while (true){  
  
 Customer customer = new Customer(++index,conductors);  
 customer.start();  
 Thread.*sleep*(buyingSpeed);  
 }  
  
}

运行结果：

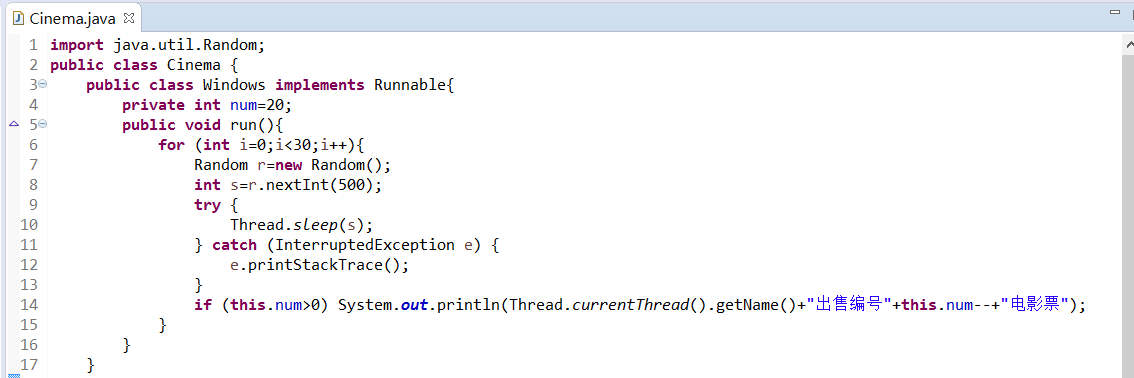
 

**李金鑫:**

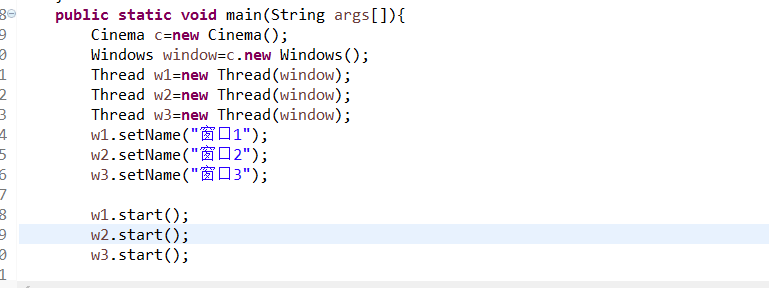
**实现过程：**

因为是三个窗口同时卖票，用Runnable实现多线程，资源共享。

程序中设定的票数为20张，将中间的间隔设置为500毫秒之内的随机数。



之后设定三个线程，分别命名窗口1、2、3。再启动线程即可。



**运行结果：**

****

# 李天浩:

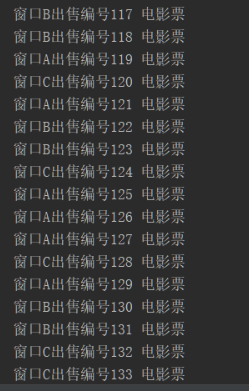
# 实现过程：

通过Runable实现多线程

Random实现随机的休眠

import java.util.Random;  
class Ticket implements Runnable{  
 private int TicketNum = 1000;  
 private boolean flag = true;  
 private synchronized void sale()  
 {  
 if(TicketNum<=0)  
 {  
 flag = false;  
 return ;  
 }  
 TicketNum--;  
 System.*out*.print(Thread.*currentThread*().getName()+"出售编号");  
 System.*out*.printf("%03d 电影票\n",(999-TicketNum));  
 }  
  
 @Override  
 public void run() {  
 Random r=new Random();  
 while(flag)  
 {  
 sale();  
 try {  
 Thread.*sleep*(Math.*abs*(r.nextInt(1000)));  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
}

# 运行结果



二．小组讨论内容：

**张俊华**：

两位同学均较好的完成了题目要求，实现了多线程同时出票，其中李金鑫同学对票号修改的操作未使用对象锁，可能会在多并发操作时出现数据异常情况，程序尚需改进。

**李金鑫：**

两位同学都很好的完成了题目要求，其中张俊华同学写的接近现实，李天浩同学简洁清晰，都很好处理了异常。

**李天浩：**

张俊华同学的思路是先产生票再去卖票,可以说是一个不同的思路吧,总体实现很好,值得我们去学习

李金鑫同学较好的完成了题目要求，实现了多线程同时出票，但是其关键方法未使用对象锁，程序尚需改进。

**六、个人总结：**

**张俊华：**

通过这次上机，我对java的多线程操作有了较为深刻的认识，了解掌握了java线程调度和控制的基本操作，学会使用对象锁解决多线程并发操作中的问题。

这次上机也加强了我对面向对象编程的理解，通过构建类抽象模拟真实世界，使程序有了较好的执行效果。

**李金鑫：**

通过这次上机，我对java的多线程有了更深刻的认识，特别是理解了多线程访问的共享方式，认识了java对现实问题的模拟。

其次，又学会了java中对随机数的运用，线程间休息间隔的不确定更提高了程序对现实的模拟度。

**李天浩：**

这次上机我学习了如何使用java的多线程,如何对线程进行调度控制,如何模拟真实世界现实问题的模拟,Random类的初步使用.

这次上机也增强了我对现实问题的理解,通过线程的引入来更好的解决问题.