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

一、 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。

2. 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

3. 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：

```
正在进行初始化设置，程序运行需要...
请输入总票数：1000
设置成功!
+--------售票窗口设置--------+
|     选项                功能          |
|      1              展示售票窗口        |
|      2              增加售票窗口        |
|      3              删除售票窗口        |
|      q             退出售票窗口设置
+--------------------+
请输入选项：2
请输入名称：1
请输入速度：1000
继续添加吗？
请输入选项：y
请输入名称：2
请输入速度：300
继续添加吗？
请输入选项：y
请输入名称：3
请输入速度：400
继续添加吗？
请输入选项：n
当前购买速度为：0ms/位
请输入速度：300
初始化完成!
```

```
+-----------------------------------+
|     选项                功能          |
|      1              模拟售票          |
|      s              系统设置          |
|      q              退出系统          |
+-----------------------------------+
请输入选项：1
迎来了第1位顾客...
迎来了第2位顾客...
迎来了第3位顾客...
迎来了第4位顾客...
窗口1为1号顾客出售编号1电影票
迎来了第5位顾客...
窗口1为2号顾客出售编号2电影票
窗口3为4号顾客出售编号3电影票
迎来了第6位顾客...
窗口1为3号顾客出售编号4电影票
窗口3为5号顾客出售编号5电影票
迎来了第7位顾客...
迎来了第8位顾客...
窗口2为8号顾客出售编号6电影票
迎来了第9位顾客...
窗口1为6号顾客出售编号7电影票
窗口2为9号顾客出售编号8电影票
迎来了第10位顾客...
窗口1为7号顾客出售编号9电影票
窗口2为10号顾客出售编号10电影票
迎来了第11位顾客...
迎来了第12位顾客...
窗口3为11号顾客出售编号11电影票
迎来了第13位顾客...
窗口2为12号顾客出售编号12电影票
迎来了第14位顾客...
```

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。

Cinema.java

```java
import java.util.Random;
public class Cinema {
    public class Windows implements Runnable{
        private int num=20;
        public void run(){
            for (int i=0;i<30;i++){
                Random r=new Random();
                int s=r.nextInt(500);
                try {
                    Thread.sleep(s);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                if (this.num>0) System.out.println(Thread.currentThread().getName()+"出售编号"+this.num--+"电影票");
            }
        }
    }
```

之后设定三个线程，分别命名窗口 1、2、3。再启动线程即可。

```
    public static void main(String args[]){
        Cinema c=new Cinema();
        Windows window=c.new Windows();
        Thread w1=new Thread(window);
        Thread w2=new Thread(window);
        Thread w3=new Thread(window);
        w1.setName("窗口1");
        w2.setName("窗口2");
        w3.setName("窗口3");

        w1.start();
        w2.start();
        w3.start();
```

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.