Lab7（并发）实验报告

马斓轩 1813076

一、实验内容

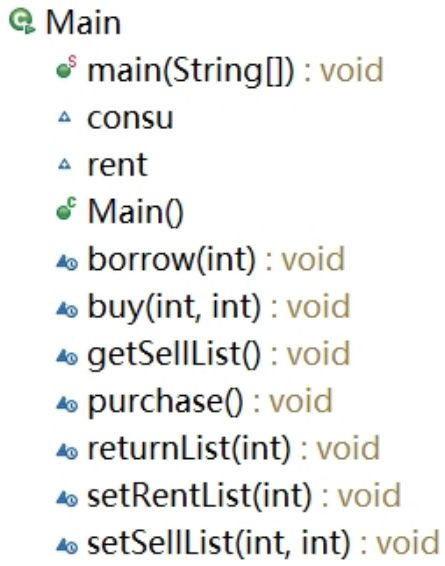
实现一个 CD 出租销售店的模拟程序，该 CD 租售店，具有可租 cd 列表，可售 cd 列表。并具有租、还 cd，销售、进货 CD 方法。程序运行 2 分钟以后停止。最后列出购买、进货、租借还的纪录（时间及行为）。 程序运行两次。

二、实验设计

算法实现:

（1）CD 出租销售店对象设计。

对象 shop，成员变量有两个，分别是出租列表和销售列表。出租列表为 boolean 型数组，销售列表为 int 型数组，储存当前型号的剩余数量。主要方法有 borrow（出租）、购买（buy）和进货（purchase），还有对应 borrow 方法的 setRentList 以及 returnList 方法用来改变出租列表 CD 的状态，对应 buy 方法的 setSellList 方法用来更新当前商店中的销售列表数量，以及实现了补货功能的方法 getSellList，每次进货将所有缺少的 CD 型号补满。



（2）设计三个线程，分别完成进货、销售和租借行为。

①进货线程：只有一个，固定的每 1 秒启动一次，但是如果临时缺货则购买线程发送消息紧急启动一次，每次补齐可售 CD 列表。进货线程调用了 Main 类中的 purchase 方法，通过 sleep 实现 1 秒休眠后继续进行。

②销售线程：可以有两个或两个以上，启动的时间为 200ms 以内的随机数。购买数量为 5 以内的随机数。如果 cd 数量不足则随机选择等候或放弃。销售线程调用了 Main 类中的 buy 方法，购买型号和购买数量都是随机的。用 Random 实现 200ms 以内的随机数，运行结束后 sleep 时长为该随机数。

③租借线程：可以有两个或两个以上，租借 CD 店的可租借 CD，启动时间为 200ms 以内的随机数。租借序号为 1-10 随机序号的 CD，如果该 CD 已经出租则随机选择等候或者放弃。如果可以借到 CD 则随机等候 200~300ms 然后归还。租借线程调用了 borrow 方法和 returnList 方法，在二者之间进行休眠 200~300ms，运行结束后 sleep 时长为 200ms 以内的随机数。

```
//进货线程
class purThreads extends Thread{
    Main shop;
    public purThreads(Main shop,int type) {
        super();
        this.shop=shop;
    }
    public void run() {
        while(true) {
            try {
                shop.purchase();
                sleep(1000);
            }catch(InterruptedException e){
                e.printStackTrace();
            }
        }
    }

}
```

```
//销售线程
class sellThreads extends Thread{
    Main shop;

    public sellThreads(Main shop) {
        super();
        this.shop=shop;
    }
    public void run() {
        while(true) {
            try {
                Random r=new Random();
                Random ty=new Random();
                shop.buy(r.nextInt(5)+1,ty.nextInt(10)+1);
                sleep(new Random().nextInt(200));
            }catch(InterruptedException e){
                e.printStackTrace();
            }
        }
    }

}
```

```
//租借线程
class rentThreads extends Thread{
    Main shop;

    public rentThreads(Main shop) {
        super();
        this.shop=shop;
    }
    public void run() {
        while(true) {
            try {
                Random r=new Random();
                Random r2=new Random();
                int num=r.nextInt(10)+1;
                shop.borrow(num);
                int wait=r2.nextInt(100)+200;
                sleep(wait);
                shop.returnList(num);
                System.out.println(getId()+" "+(new Date())+" "+num+"号归还成功");
                sleep(new Random().nextInt(200));
            }catch(InterruptedException e){
                e.printStackTrace();
            }
        }
    }
}
```

（3）Main 类中的主要方法设计。

①purchase 方法：首先对目前销售列表进行扫描，若发现其中一个不满足 10 个，调用 getSellList 方法进货，将所有型号的 CD 补满。若全部都满则 wait 释放对象锁。

②buy 方法：当目前所需的型号数量不足，调用 getSellList 方法紧急补货。假如数量足够，调用 setSellList 方法更新销售列表的数量。

③borrow 方法：判断当前型号是否已被出租，若出租就要 wait 释放对象锁，

否则出借。

```
synchronized void purchase() throws InterruptedException {
    int num=0;
    int type=0;
    for(int i=1;i<=10;i++) {
        if(consu[i]==10) {
            num++;
        }
        else {
            type=i;
            break;
        }
    }
    while(num==10) {
        Thread t=Thread.currentThread();
        System.out.println(t.getId()+" "+(new Date())+" "+"不需要进货");
        wait();
    }
    Thread t=Thread.currentThread();
    System.out.println(t.getId()+" "+(new Date())+" "+"现存"+type+"号"+consu[type]+"个");
    getSellList();
    System.out.println(t.getId()+" "+(new Date())+" "+"所有种类已补齐");
    notifyAll();
}
```

```
synchronized void buy(int i,int type) throws InterruptedException {
    while(consu[type]<i) {
        Thread t=Thread.currentThread();
        System.out.println(t.getId()+" "+(new Date())+" "+type+"号数量不足,差"+(i-consu[type])+"个");
        getSellList();
        System.out.println(t.getId()+" "+(new Date())+" "+"所有种类已补齐");
        wait();
    }
    Thread t=Thread.currentThread();
    System.out.println(t.getId()+" "+(new Date())+" "+"现存"+type+"号"+consu[type]+"个");
    setSellList(i,type);
    System.out.println(t.getId()+" "+(new Date())+" "+"售出"+type+"号"+i+"个后还剩"+consu[type]+"个");
    notifyAll();
}
```

```
synchronized void borrow(int type) throws InterruptedException {
    while(!rent[type]) {
        Thread t=Thread.currentThread();
        System.out.println(t.getId()+" "+(new Date())+" "+type+"号已出租");
        wait();
    }
    Thread t=Thread.currentThread();
    System.out.println(t.getId()+" "+(new Date())+" "+type+"号可以出借");
    setRentList(type);
    System.out.println(t.getId()+" "+(new Date())+" "+type+"号成功出借");
    notifyAll();
}
```

（4）测试代码部分

创建 7 个线程，一个进货线程，销售线程和租借线程各 3 个。将全部线程设置为 Daemon 线程，保证程序的正常停止。主线程休眠两分钟，使各子线程运行。运行结束后输出最终店里的两个列表状态。

```
Main shop=new Main();
Random r=new Random();
//一个进货线程
st[0]=new purThreads(shop,r.nextInt(10)+1);

for(int i=1;i<4;i++) {
    //三个租借线程
    st[i]=new rentThreads(shop);
    //三个销售线程
    st[i+3]=new sellThreads(shop);
}
for(int i=0;i<7;i++) {
    st[i].setDaemon(true);
    st[i].start();
}
```

```
try {
    Thread.sleep(120000);
} catch (InterruptedException e) {
    // TODO Auto-generated catch block
    System.out.print("error");
}
System.out.println("出租列表");
for(int j=1;j<=10;j++) {
    System.out.print(j+" "+shop.rent[j]+" ");
}
System.out.println();
System.out.println("出售列表");
for(int j=1;j<=10;j++) {
    System.out.print(j+" "+shop.consu[j]+" ");
}
```

三、思考与分析

整个程序的设计参考了“生产者与消费者”的结构。目前代码设计中仍存在一些问题。

①对于 sleep 和 wait 的运用还不熟悉。尤其是对“固定一秒调用一次进货线程”并没有很好的实现。由于 1000ms 相比 200ms 时间较长，进货线程很容易无法竞争到 CPU，只有当进货线程为第一个运行线程时，才能满足后续 CPU 要求。

②尽管已经将出租列表状态初始值设置为 true，在运行中有时会先出现“某型号已出租”。这从代码角度是解释不通的，因为出租和归还是同一线程中按顺序的两部分，几乎不可能从直接跳过出租部分，单从输出的结果中确实存在这样的问题。

③在 buy 方法中，如果数量足够进行销售的时候，输出结果上也可能出现跳过第一句输出而直接进行出售列表更新，再继续输出。

对于一个线程中的多个任务可以用阻塞式队列结构实现按顺序实现，但对于以上设计来说，并不能很好满足出租和归还的要求。因为归还受到了阻塞，不能再自由归还。

附录：源代码

```
import java.io.FileNotFoundException;
import java.io.PrintStream;
import java.util.Date;
import java.util.Random;


public class Main {
    boolean[] rent=new boolean[11];
    int[] consu=new int[11];

    public Main() {
        for(int i=1;i<=10;i++) {
            rent[i]=true;
        }
    }
    synchronized void setRentList(int type) {
        rent[type]=false;
    }
    synchronized void returnList(int type) {
        if(!rent[type])
            rent[type]=true;
    }
    synchronized void setSellList(int i,int type) {
        consu[type]-=i;
    }
    synchronized void getSellList() {
        for(int i=1;i<=10;i++) {
            if(consu[i]!=10) {
                int lack=10-consu[i]%10;
                consu[i]+=lack;
            }
        }
    }
    synchronized void purchase() throws InterruptedException {
        int num=0;
        int type=0;
        for(int i=1;i<=10;i++) {
            if(consu[i]==10) {
                num++;
            }
            else {
                type=i;
                break;
```

```
            }
        }
        while(num==10) {
            Thread t=Thread.currentThread();
            System.out.println(t.getId()+" "+(new Date())+" "+"不需要进货");
            wait();
        }
        Thread t=Thread.currentThread();
        System.out.println(t.getId()+" "+(new Date())+" "+"现存"+type+"号
"+consu[type]+"个");
        getSellList();
        System.out.println(t.getId()+" "+(new Date())+" "+"所有种类已补齐");
        notifyAll();
    }
    synchronized void buy(int i,int type) throws InterruptedException {
        while(consu[type]<i) {
            Thread t=Thread.currentThread();
            System.out.println(t.getId()+" "+(new Date())+" "+type+"号数量不
足,差 "+(i-consu[type])+"个");
            getSellList();
            System.out.println(t.getId()+" "+(new Date())+" "+"所有种类已补齐
");
            wait();
        }
        Thread t=Thread.currentThread();
        System.out.println(t.getId()+" "+(new Date())+" "+"现存"+type+"号
"+consu[type]+"个");
        setSellList(i,type);
        System.out.println(t.getId()+" "+(new Date())+" "+"售出"+type+"号
"+i+"个后还剩"+consu[type]+"个");
        notifyAll();
    }
    synchronized void borrow(int type) throws InterruptedException {
        while(!rent[type]) {
            Thread t=Thread.currentThread();
            System.out.println(t.getId()+" "+(new Date())+" "+type+"号已出租
");
            wait();
        }
        Thread t=Thread.currentThread();
        System.out.println(t.getId()+" "+(new Date())+" "+type+"号可以出借
");
        setRentList(type);
        System.out.println(t.getId()+" "+(new Date())+" "+type+"号成功出借
```

```
");
        notifyAll();
    }


    public static void main(String[] args)  {
        PrintStream out = null;
        try {
            out = new PrintStream("D:\\out2.txt");
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        System.setOut(out);
        Thread[] st=new Thread[7];
        Main shop=new Main();
        Random r=new Random();
        //一个进货线程
        st[0]=new purThreads(shop,r.nextInt(10)+1);

        for(int i=1;i<4;i++) {
            //三个租借线程
            st[i]=new rentThreads(shop);
            //三个销售线程
            st[i+3]=new sellThreads(shop);
        }
        for(int i=0;i<7;i++) {
            st[i].setDaemon(true);
            st[i].start();
        }
        try {
            Thread.sleep(120000);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            System.out.print("error");
        }
        System.out.println("出租列表");
        for(int j=1;j<=10;j++) {
            System.out.print(j+" "+shop.rent[j]+" ");
        }
        System.out.println();
        System.out.println("出售列表");
        for(int j=1;j<=10;j++) {
            System.out.print(j+" "+shop.consu[j]+" ");
```

```
        }
    }

}
//进货线程
class purThreads extends Thread{
    Main shop;
    public purThreads(Main shop,int type) {
        super();
        this.shop=shop;
    }
    public void run() {
        while(true) {
            try {
                shop.purchase();
                sleep(1000);
            }catch(InterruptedException e){
                e.printStackTrace();
            }
        }
    }

}
//租借线程
class rentThreads extends Thread{
    Main shop;

    public rentThreads(Main shop) {
        super();
        this.shop=shop;
    }
    public void run() {
        while(true) {
            try {
                Random r=new Random();
                Random r2=new Random();
                int num=r.nextInt(10)+1;
                shop.borrow(num);
                int wait=r2.nextInt(100)+200;
                sleep(wait);
                shop.returnList(num);
                System.out.println(getId()+" "+(new Date())+" "+num+"号归还
成功");

                sleep(new Random().nextInt(200));
```

```
            }catch(InterruptedException e){
                e.printStackTrace();
            }
        }
    }
}
//销售线程
class sellThreads extends Thread{
    Main shop;

    public sellThreads(Main shop) {
        super();
        this.shop=shop;
    }
    public void run() {
        while(true) {
            try {
                Random r=new Random();
                Random ty=new Random();
                shop.buy(r.nextInt(5)+1,ty.nextInt(10)+1);
                sleep(new Random().nextInt(200));
            }catch(InterruptedException e){
                e.printStackTrace();
            }
        }
    }


    }
```