

Java 实现 CD 出租销售店实验报告

软件学院软件工程专业 2019 级 2 班 韦诗睿 1913184

一、实验内容

(一) 主要内容：运用 Java 线程、同步的相关知识实现一个 CD 出租销售店的模拟程序。

(二) 主要功能：

1. CD 店基本定义：具有可租、可售列表，并具有租、还、销售、进货 CD 的方法。店中可租 CD 列表 10 张，可售 CD 列表 10 种，每种 10 张。
2. 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。
3. 销售线程：可以有两个或两个以上，售卖的时间为 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。
4. 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。

二、设计分析

(一) 线程分析：

由题目要求可知，因为租借列表和销售列表并不相同，且租借缺货靠归还补齐，而销售缺货靠进货补齐，因此 CD 店的功能可分为两部分独立实现：一部分为进货和销售，另一部分为租借和归还，两部分互不干扰。

1. 主线程 **ControlThread**：控制所有线程的启动和运行。设置进货线程一个，销售线程 4 个，租借线程 4 个。程序运行时间两分钟。

2. 进货和销售线程：

(1) 进货线程 **GetInThread**：调用 **CD_collect::getin()** 进货函数。在获得锁的情况下固定 1 秒启动一次，可被销售线程唤醒。

(2) 销售线程 **saleThread**：采用生成随机数的方式生成销售的种类以及数量，为方便起见将一次购买的多张 CD 设置为同一类。当要购买的 CD 缺货时，采取随机生成布尔值的方式选择等候或者放弃，若选择等候或欲买的 CD 数量充足，

则调用 `CD_collect::sale()` 销售函数。采用生成 200 以内随机数的方式定义售卖的时间，销售完后线程休眠 1 秒让其他线程执行。

3. 租借线程 `RentThread`：采用生成随机数的方式生成租借的种类。当要租借的 CD 缺货时，采取随机生成布尔值的方式选择等候或者放弃，若选择等候或租借的 CD 未被借出，则调用 `CD::rent()` 租借函数。采用生成 200 以内随机数的方式定义租借的时间。

（二）同步分析：

1. 租借：

由题，可租借的 CD 为单类单张，且由于租借和归还的规则，因此同步的对象为单张 CD，而非租借的 CD 列表。设置类 CD 定义单种 CD 的信息。

（1）类 CD 的基本定义：

单种类 CD 的信息包括 CD 的名称 `name`，以及该类 CD 的数量 `num`。在租借时设置 `num = 1`，在销售时设置 `num = 10`。

（2）租借方法：`void rent(int id, CD cd)`：

由于同步的对象为单张 CD，因此将该 `rent` 方法加上 `synchronized` 关键字。调用时首先判断该 CD 是否在店内，若不在则等候其他线程归还；若在则将 `num` 设置为 0 表示租借状态，等待 200~300 秒的随机时间，将 `num` 重新设为 1 表示归还状态，唤醒其他想要借同一张 CD 的线程。

（3）结果展示：

1) 多线程租借不同 CD 并归还：

```
RentThread4 Try Rent CD_rent5
Wed May 26 21:38:19 CST 2021 Thread4 Rent CD_rent5
RentThread1 Try Rent CD_rent4
Wed May 26 21:38:19 CST 2021 Thread1 Rent CD_rent4
RentThread2 Try Rent CD_rent7
Wed May 26 21:38:19 CST 2021 Thread2 Rent CD_rent7
RentThread3 Try Rent CD_rent10
Wed May 26 21:38:19 CST 2021 Thread3 Rent CD_rent10
Wed May 26 21:38:19 CST 2021 CD_rent4 has been returned by Thread1
Wed May 26 21:38:19 CST 2021 CD_rent5 has been returned by Thread4
Wed May 26 21:38:19 CST 2021 CD_rent7 has been returned by Thread2
Wed May 26 21:38:19 CST 2021 CD_rent10 has been returned by Thread3
```

2) 租借线程 4 想要借第二张 CD, 等待租借线程 1 归还。

```
RentThread4 Try Rent CD_rent2
CD_rent2 isn't available
CD_rent2 wait others to return
Wed May 26 21:38:21 CST 2021 CD_rent2 has been returned by Thread1
RentThread2 Try Rent CD_rent8
Wed May 26 21:38:22 CST 2021 Thread2 Rent CD_rent8
RentThread3 Try Rent CD_rent10
Wed May 26 21:38:22 CST 2021 Thread3 Rent CD_rent10
Wed May 26 21:38:22 CST 2021 CD_rent8 has been returned by Thread2
Wed May 26 21:38:22 CST 2021 CD_rent10 has been returned by Thread3
Wed May 26 21:38:22 CST 2021 Thread4 Rent CD_rent2
RentThread1 Try Rent CD_rent9
Wed May 26 21:38:23 CST 2021 Thread1 Rent CD_rent9
```

2. 进货和销售:

由题目要求可知, 当缺货时需要紧急启动进货线程, 且每次进货要补齐全部 CD 列表, 因此同步的对象非单类 CD, 而是整个 CD 列表。因此设置类 CD_collect, 以整个 CD 列表为对象进行操作。

(1) CD_collect 基本定义:

- 1) 可售 CD 数组 cdlist, 长度为 10, 元素类型为 CD。
- 2) 采用锁的方式进行线程同步: Lock l
- 3) 设置锁上的 Condition 控制进货和销售线程间的通信
分别为: getincondition 以及 salecondition。

(2) 进货方法: void getin()

- 1) 获得锁后, 遍历可售 CD 数组 cdlist, 重置每种 CD 的数量为 10。
- 2) 若为常规进货 (Normal GetIn), 即获得锁的情况下等待 1 秒进货, 则设置 getincondition 等待 1 秒。使用 Condition.awaitNaos(long time) 方法返回被唤醒时剩余的时间: 若剩余的时间小于等于 0, 则证明为常规进货。
- 3) 若为紧急进货, 则在进货结束后唤醒 salecondition 的等待队列, 即补货完成后销售线程可继续购买该类 CD。

(3) 销售方法: void sale(int n,int id,int type)

- 1) 获得锁后, 检查该类 CD 是否库存充足, 若不足, 则唤醒 getincondition 的等待队列, 即实行强制进货 (Force to GetIn), 同时设置 salecondition 等待, 即该销售线程等待 CD 补齐。
- 2) 当进货完成或库存充足时, 改变该类 CD 的数量表示售出, 释放锁, 该线程睡眠 1 秒。

(4) 结果展示:

1) 进货展示:

显示某类 CD 进货状态以及进货完成后的库存数量。

常规进货:

```
Normal GetIn!  
Wed May 26 22:07:31 CST 2021 GetIn Start:  
CD_saled1 GetIn Start  
CD_saled1 Current Storage:10  
CD_saled2 GetIn Start  
CD_saled2 Current Storage:10  
CD_saled3 GetIn Start  
CD_saled3 Current Storage:10  
CD_saled4 GetIn Start  
CD_saled4 Current Storage:10  
CD_saled5 GetIn Start  
CD_saled5 Current Storage:10  
CD_saled6 GetIn Start  
CD_saled6 Current Storage:10  
CD_saled7 GetIn Start  
CD_saled7 Current Storage:10  
CD_saled8 GetIn Start  
CD_saled8 Current Storage:10  
CD_saled9 GetIn Start  
CD_saled9 Current Storage:10  
CD_saled10 GetIn Start  
CD_saled10 Current Storage:10
```

2) 销售展示:

Try: 显示某销售线程尝试购买某类型的 CD 多个。

若销售成功, 则显示销售的时间、销售线程 id 和想要销售的 CD 类型以及数量。

销售结束后显示该类型 CD 的库存。

A. 正常销售:

```
saledThread1 Try Sale CD_saled9 4  
Wed May 26 22:03:09 CST 2021 Thread1 Sale 4copies CD_saled9  
CD_saled9 Current Storage: 6  
saledThread4 Try Sale CD_saled3 5  
Wed May 26 22:03:09 CST 2021 Thread4 Sale 5copies CD_saled3  
CD_saled3 Current Storage: 5  
saledThread3 Try Sale CD_saled1 5  
Wed May 26 22:03:09 CST 2021 Thread3 Sale 5copies CD_saled1  
CD_saled1 Current Storage: 5  
saledThread2 Try Sale CD_saled8 1  
Wed May 26 22:03:09 CST 2021 Thread2 Sale 1copies CD_saled8  
CD_saled8 Current Storage: 9
```

B. 当库存不足时可以放弃（yellow），也可以等候(blue)。等候时立即唤醒进货线程进行进货（Force to get in.）【为出现该效果，设置不同线程均购买同一类CD。】

```
CD_saled2 Current Storage: 0
saledThread3 Try Sale CD_saled2 3
saledThread3 give up CD_saled2
saledThread3 Try Sale CD_saled2 1
CD_saled2's stock is not enough now
CD_saled2 wait to GetIn
Wed May 26 22:07:26 CST 2021 Force to get in
Wed May 26 22:07:26 CST 2021 GetIn Start:
CD_saled1 GetIn Start
CD_saled1 Current Storage:10
CD_saled2 GetIn Start
CD_saled2 Current Storage:10
CD_saled3 GetIn Start
CD_saled3 Current Storage:10
CD_saled4 GetIn Start
CD_saled4 Current Storage:10
CD_saled5 GetIn Start
CD_saled5 Current Storage:10
CD_saled6 GetIn Start
CD_saled6 Current Storage:10
CD_saled7 GetIn Start
CD_saled7 Current Storage:10
CD_saled8 GetIn Start
CD_saled8 Current Storage:10
CD_saled9 GetIn Start
CD_saled9 Current Storage:10
CD_saled10 GetIn Start
CD_saled10 Current Storage:10
Wed May 26 22:07:26 CST 2021 Thread3 Sale 1copies CD_saled2
CD_saled2 Current Storage: 9
```

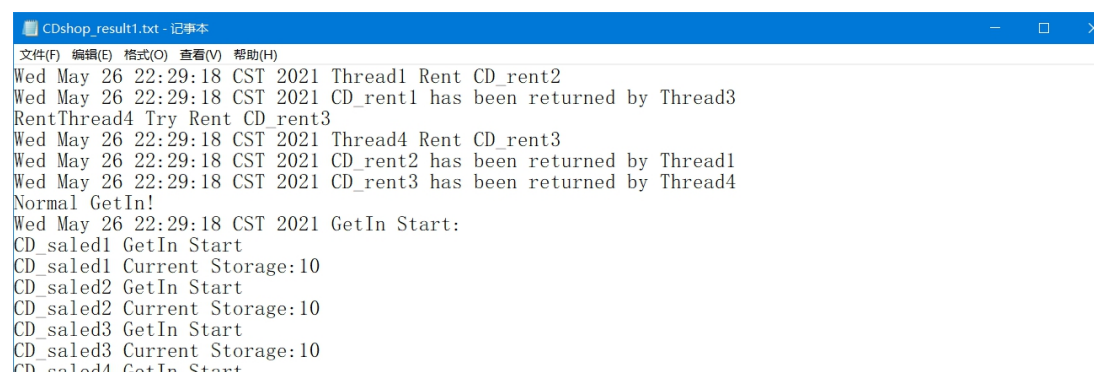
三、结果展示：购买、进货、租借还的纪录（时间及行为）

两次记录皆记录于附件文件 record.txt 中

GitHub 地址：

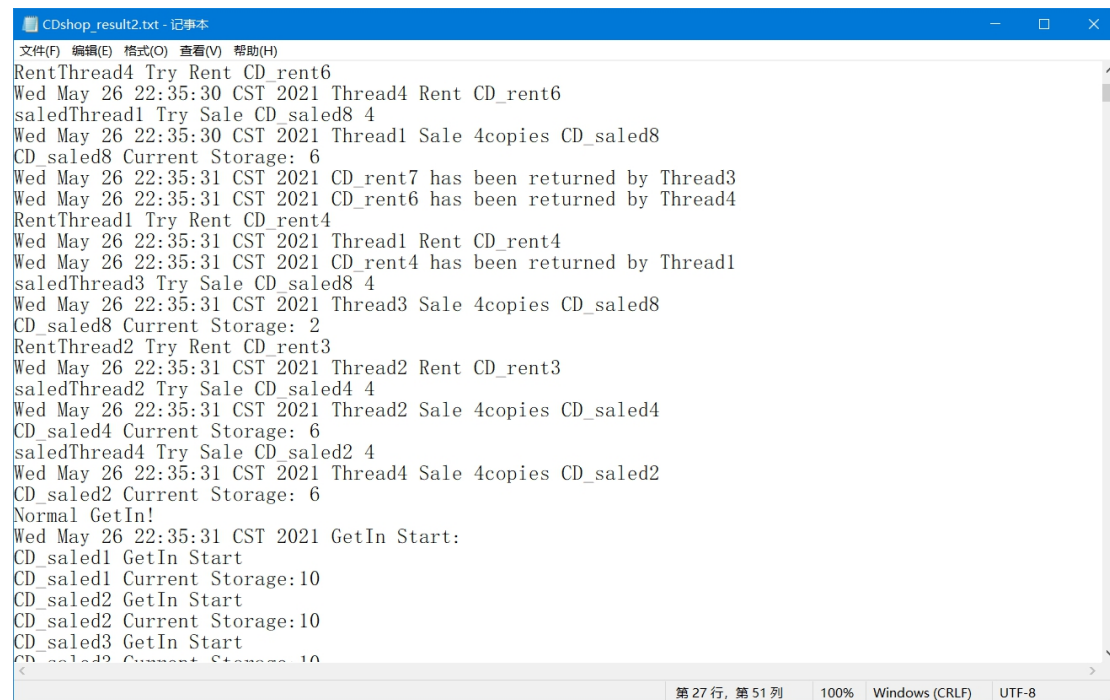
https://github.com/RuiNov1st/2021NKU_java_course/tree/main/CD_shop

1. Result1:



```
CDshop_result1.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
Wed May 26 22:29:18 CST 2021 Thread1 Rent CD_rent2
Wed May 26 22:29:18 CST 2021 CD_rent1 has been returned by Thread3
RentThread4 Try Rent CD_rent3
Wed May 26 22:29:18 CST 2021 Thread4 Rent CD_rent3
Wed May 26 22:29:18 CST 2021 CD_rent2 has been returned by Thread1
Wed May 26 22:29:18 CST 2021 CD_rent3 has been returned by Thread4
Normal GetIn!
Wed May 26 22:29:18 CST 2021 GetIn Start:
CD_saled1 GetIn Start
CD_saled1 Current Storage:10
CD_saled2 GetIn Start
CD_saled2 Current Storage:10
CD_saled3 GetIn Start
CD_saled3 Current Storage:10
CD_saled4 GetIn Start
```


2. Result2:



```
CDshop_result2.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
RentThread4 Try Rent CD_rent6
Wed May 26 22:35:30 CST 2021 Thread4 Rent CD_rent6
saledThread1 Try Sale CD_saled8 4
Wed May 26 22:35:30 CST 2021 Thread1 Sale 4copies CD_saled8
CD_saled8 Current Storage: 6
Wed May 26 22:35:31 CST 2021 CD_rent7 has been returned by Thread3
Wed May 26 22:35:31 CST 2021 CD_rent6 has been returned by Thread4
RentThread1 Try Rent CD_rent4
Wed May 26 22:35:31 CST 2021 Thread1 Rent CD_rent4
Wed May 26 22:35:31 CST 2021 CD_rent4 has been returned by Thread1
saledThread3 Try Sale CD_saled8 4
Wed May 26 22:35:31 CST 2021 Thread3 Sale 4copies CD_saled8
CD_saled8 Current Storage: 2
RentThread2 Try Rent CD_rent3
Wed May 26 22:35:31 CST 2021 Thread2 Rent CD_rent3
saledThread2 Try Sale CD_saled4 4
Wed May 26 22:35:31 CST 2021 Thread2 Sale 4copies CD_saled4
CD_saled4 Current Storage: 6
saledThread4 Try Sale CD_saled2 4
Wed May 26 22:35:31 CST 2021 Thread4 Sale 4copies CD_saled2
CD_saled2 Current Storage: 6
Normal GetIn!
Wed May 26 22:35:31 CST 2021 GetIn Start:
CD_saled1 GetIn Start
CD_saled1 Current Storage:10
CD_saled2 GetIn Start
CD_saled2 Current Storage:10
CD_saled3 GetIn Start
CD_saled3 Current Storage:10
```

四、心得体会

本次实验学习了如何使用线程、同步以及锁。首先是关键字 `synchronized` 的学习使用，一开始疑惑于同步的对象到底是单种 CD 还是所有 CD、是对象还是对象的方法。最后分析“进货要补齐所有 CD”的功能后确定采取同步所有 CD 的方式较为简单，但其实这一设计不符合生活实际，不同线程应当可以同时访问不同类的 CD。但由于不知如何能够同步单种 CD 的同时获取所有 CD 的锁进行立即进货，因此选择采取这一方法。

其次是 `lock` 的学习使用，由于在实现购买缺货部分时，使用 `synchronized` 尝试多次仍只能实现停止该购买进程的同时随机唤醒其他购买线程或进货进程，无法实现立即调用进货线程的功能，因此选择采用 `lock` 里的 `condition` 方法对唤醒条件施加约束，达到立即唤醒进货线程的要求。

在学习使用同步和多线程时，发现代码难以调试，很多时候输出产生死锁和死循环，但问题到底出在哪需要很久的测试和观察，对锁的使用尚未达到熟练。

但深感这次 CD 店的题目设置很有意思，贴近生活，`coding` 时很愉快。

五、源代码

GitHub 地址:

https://github.com/RuiNov1st/2021NKU_java_course/tree/main/CD_shop

(一) 主函数:

```
1. import java.io.IOException;
2. import java.util.*;
3. import java.util.logging.*;
4. import java.util.logging.FileHandler;
5. import java.util.logging.Logger;
6. import java.util.logging.SimpleFormatter;
7. import java.util.concurrent.locks.*;
8.
9. //主线程
10. public class CDshop {
11.     public static void main(String []args) {
12.         //使用锁
13.         Lock l = new ReentrantLock();
14.         //设置进货和销售条件
15.         Condition salecondition;
16.         Condition getincondition;
17.         getincondition = l.newCondition();
18.         salecondition = l.newCondition();
19.
20.         //设置日志
21.         Logger mylogger = Logger.getLogger("CDshop");
22.         //输入到文件中
23.         FileHandler f;
24.         try {
25.             f = new FileHandler("D:\\java_file\\HelloWorld\\cdshop.txt");
26.             SimpleFormatter sf = new SimpleFormatter();
27.             f.setFormatter(sf);
28.             mylogger.addHandler(f);
29.         } catch (SecurityException e) {
30.             // TODO Auto-generated catch block
31.             e.printStackTrace();
32.         } catch (IOException e) {
33.             // TODO Auto-generated catch block
34.             e.printStackTrace();
35.         }
36.
37.         //可售 CD 列表有十种
38.         CD_collect cc = new CD_collect(mylogger,l,salecondition,getincondition);
```

```

39. //可租 CD 列表 10 张
40. CD_pro []cdrentlist = new CD_pro[10];
41. for(int i = 0;i<10;i++) {
42.     cdrentlist[i] = new CD_pro();
43. }
44. //给销售和租借的 CD 类别起名字
45. for(int i = 0;i<10;i++) {
46.     cdrentlist[i].name = "CD_rent"+String.valueOf(i+1);
47.     cdrentlist[i].num = 1;//每种 CD 只有一张
48. }
49. //传入线程中
50. ControlThread_pro control = new ControlThread_pro(cc,cdrentlist,mylogger,
    l,getincondition , salecondition);
51. control.start();
52. }
53.
54. }

```

(二) 控制线程 ControlThread:

```

1. //控制所有线程的启动和运行
2. class ControlThread_pro extends Thread{
3.     //控制 CD
4.     CD_pro []cd_saled;
5.     CD_pro []cdrentlist;
6.     CD_collect cc;
7.     Logger lg;
8.     Lock l;
9.     Condition getincondition;
10.    Condition salecondition;
11.
12.    ControlThread_pro(CD_collect cc,CD_pro []cdrentlist,Logger lg,Lock l,Con
        dition getincondition,Condition salecondition){
13.        this.cc = cc;
14.        this.cdrentlist =cdrentlist;
15.        this.lg = lg;
16.        this.l = l;
17.        this.salecondition = salecondition;
18.        this.getincondition = getincondition;
19.    }
20.
21.    @Override
22.    //去启动其他所有的线程
23.    public void run() {
24.        //直接启动进货线程

```



```

25.    GetInThread_pro gt= new GetInThread_pro(cc,l,getincondition,saleconditio
n);
26.    gt.start();
27.    //启动两个或两个以上的销售线程
28.    new saleThread_pro(cc,gt,lg,l,getincondition,salecondition).start();
29.    new saleThread_pro(cc,gt,lg,l,getincondition,salecondition).start();
30.    new saleThread_pro(cc,gt,lg,l,getincondition,salecondition).start();
31.    new saleThread_pro(cc,gt,lg,l,getincondition,salecondition).start();
32.
33.    //租借进程
34.    new RentThread(cdrentlist,lg).start();
35.    new RentThread(cdrentlist,lg).start();
36.    new RentThread(cdrentlist,lg).start();
37.    new RentThread(cdrentlist,lg).start();
38.
39.    //控制线程睡眠
40.    //控制线程结束则所有线程结束
41.    try {
42.        Thread.sleep(120*1000);
43.    } catch (InterruptedException e) {
44.        // TODO Auto-generated catch block
45.        e.printStackTrace();
46.    }
47.
48.    }
49.    }

```

(三) CD 类:

```

1.    //CD 的类
2.    class CD_pro{
3.        String name;
4.        int num=10;
5.
6.        //租借进程
7.        synchronized void rent(int id,CD_pro cd,Logger lg) {
8.            //要借的CD 没了
9.            while(this.num==0) {
10.                //lg.warning(name+"s stock is not enough now\n"+name+" wait others to r
eturn \n");
11.                System.out.print(name+"isn't avaliable\n");
12.                System.out.print(name+" wait others to return \n");
13.                notifyAll();
14.            try {
15.                wait();
16.            } catch (InterruptedException e) {

```

```

17.    // TODO Auto-generated catch block
18.    e.printStackTrace();
19.    }
20.    }
21.    //借到了
22.    this.num = 0;
23.    //lg.info(new Date()+"Thread"+id+" "+"Rent "+name+"\n");
24.    System.out.print(new Date()+" "+"Thread"+id+" "+"Rent "+name+"\n");
25.    Random time = new Random();
26.    try {
27.        //随机等候 200~300ms 然后归还
28.        notifyAll();
29.        wait(time.nextInt(100)+200);
30.        //Thread.sleep(time.nextInt(100)+200);
31.        this.num = 1;//归还 CD
32.        //lg.info(name+" has been returned by"+" Thread"+id+"\n");
33.        System.out.print(new Date()+" "+name+" has been returned by"+" Thread"+
            id+"\n");
34.        notifyAll();//唤醒其他想要借同一张 CD 的线程
35.    } catch (InterruptedException e) {
36.        // TODO Auto-generated catch block
37.        e.printStackTrace();
38.    }
39.    //启动其他线程
40.    try {
41.        //买完后正在执行的线程主动放弃 CPU 让其他线程执行
42.        Thread.sleep(1000);
43.    } catch (InterruptedException e) {
44.        // TODO Auto-generated catch block
45.        e.printStackTrace();
46.    }
47.    }
48.    }

```

(四) CD_collect 类:

```

1.    //把进货和销售看成集合操作
2.    class CD_collect{
3.        Lock l;
4.        private Condition getincondition;
5.        private Condition salecondition;
6.        int type = 10;
7.        Logger lg;
8.        //可售 CD 列表有十种
9.        CD_pro []cdlist =new CD_pro[10];

```

```

10.  CD_collect(Logger lg,Lock l, Condition salecondition, Condition getinconditi
    on) {
11.    this.l= l;
12.    this.getincondition = getincondition;
13.    this.salecondition = salecondition;
14.    this.lg = lg;
15.    for(int i = 0;i<10;i++) {
16.      cdlist[i] = new CD_pro();
17.      cdlist[i].name = "CD_saled"+String.valueOf(i+1);
18.    }
19.  }
20.  //进货
21.  void getin() throws InterruptedException {
22.    l.lock();
23.    for(int i = 0;i<10;i++) {
24.      cdlist[i].num = 10;
25.      //lg.info(cdlist[i].name+" GetIn Start\n"+cdlist[i].name+" Current Storage:
        "+cdlist[i].num+"\n");
26.      System.out.print(cdlist[i].name+" GetIn Start\n");
27.      System.out.print(cdlist[i].name+" Current Storage:"+cdlist[i].num+"\n");
28.    }
29.    salecondition.signalAll();
30.    long remain = getincondition.awaitNanos(1000000000);
31.    if(remain<=0) {
32.      System.out.print("Normal GetIn!\n");
33.    }
34.    l.unlock();
35.  }
36.
37.  //销售
38.  void sale(int n,int id,int type,GetInThread_pro gt) {
39.    l.lock();
40.    //库存不足
41.    while(this.cdlist[type].num-n<0) {
42.      //lg.warning(cdlist[type].name+"s stock is not enough now\n"+cdlist[type].
        name+" wait to GetIn\n");
43.      //System.out.print("current Storage: "+this.cdlist[type].num+"\n");
44.      System.out.print(cdlist[type].name+"s stock is not enough now\n");
45.      //System.out.print(cdlist[type].name+" wait to GetIn\n");
46.      System.out.print(cdlist[type].name+" wait to GetIn\n");
47.      try {
48.        System.out.print(new Date()+" Force to get in\n");
49.        getincondition.signal();
50.        salecondition.await();

```

```

51.     } catch (InterruptedException e1) {
52.         // TODO Auto-generated catch block
53.         e1.printStackTrace();
54.     }
55. }
56. //库存充足，可以售卖
57. cdlist[type].num = cdlist[type].num-n;
58. //lg.info(new Date()+"Thread"+id+" "+"Sale "+n+"copies "+cdlist[type].name+"\n"+cdlist[type].name+" Current Storage: "+cdlist[type].num+"\n");
59. System.out.print(new Date()+" "+"Thread"+id+" "+"Sale "+n+"copies "+cdlist[type].name+"\n");
60. System.out.print(cdlist[type].name+" Current Storage: "+cdlist[type].num+"\n");
61.
62. l.unlock();
63. //启动其他线程
64. try {
65.     //买完后正在执行的线程主动放弃 CPU 让其他线程执行
66.     Thread.sleep(1000);
67. } catch (InterruptedException e) {
68.     // TODO Auto-generated catch block
69.     e.printStackTrace();
70. }
71.
72. }
73. }

```

(五) 进货线程 GetInThread:

```

1. //进货线程
2. //每次补齐可售 CD 列表
3. class GetInThread_pro extends Thread{
4.     Lock l;
5.     //控制 CD
6.     CD_pro []cdlist;
7.     CD_collect cc;
8.     Logger lg;
9.     Condition salecondition;
10.    Condition getincondition;
11.    //构造函数
12.    GetInThread_pro(CD_collect cc, Lock l, Condition getincondition, Condition salecondition){
13.        this.salecondition = salecondition;
14.        this.getincondition = getincondition;
15.        this.l = l;
16.        this.cc = cc;

```

```

17.  this.setDaemon(true);//一直处于循环状态的线程一定要设置为daemon 否则主线程无法结束
18.  }
19.
20.  //同步的是当前的CD 列表对象
21.  public void run(){
22.      while(true) {
23.          //lg.info("GetIn Start:\n");
24.          System.out.print(new Date()+" "+"GetIn Start:\n");
25.          try {
26.              cc.getin();
27.          } catch (InterruptedException e) {
28.              // TODO Auto-generated catch block
29.              e.printStackTrace();
30.          }
31.      }
32.  }
33.  }

```

(六) 销售线程 saleThread:

```

1.  //销售线程
2.  class saleThread_pro extends Thread{
3.      Lock l;
4.      Condition salecondition;
5.      Condition getincondition;
6.      GetInThread_pro gt;
7.      //销售CD 列表
8.      CD_pro [] cd_saled;
9.      CD_collect cc;
10.     //区分不同的线程
11.     static int ids;
12.     int id;
13.     Logger lg;
14.     //构造函数
15.     saleThread_pro(CD_collect cc,GetInThread_pro gt,Logger lg,Lock l,Condition getincondition,Condition salecondition){
16.         this.salecondition = salecondition;
17.         this.getincondition = getincondition;
18.         this.l = l;
19.         this.gt = gt;
20.         this.cc = cc;
21.         this.setDaemon(true);//一直处于循环状态的线程一定要设置为daemon 否则主线程无法结束
22.         id = ++ids;
23.         this.lg = lg;

```

```

24.     }
25.     //防止里面的代码拼错了
26.     //保证不会有书写错误
27.     @Override
28.     public void run() {
29.         //种类
30.         Random t = new Random();
31.         //数量
32.         Random r = new Random();
33.         while(true) {
34.             try {
35.                 Thread.sleep(r.nextInt(200));
36.             } catch (InterruptedException e) {
37.                 // TODO Auto-generated catch block
38.                 e.printStackTrace();
39.             } //200 以内的休眠时间
40.
41.             int saletype = t.nextInt(10); //销售十以内的随机种类
42.             int salenum = r.nextInt(5)+1; //销售五以内的随机数量
43.
44.             //id 为 x 的销售线程尝试销售某种 CD 多少张
45.             //lg.info("saledThread"+id+" Try Sale "+cc.cdlist[saletype].name+" "+salenum+"\n");
46.             System.out.print("saledThread"+id+" Try Sale "+cc.cdlist[saletype].name+" "+salenum+"\n");
47.             //要买的 CD 的库存不够
48.             if(cc.cdlist[saletype].num<salenum) {
49.                 //随机选择等候还是放弃
50.                 if(r.nextBoolean()) {
51.                     cc.sale(salenum, id, saletype, gt);
52.                 } else {
53.                     //lg.info("saledThread"+id+" give up"+" "+cc.cdlist[saletype].name+"\n");
54.                     System.out.print("saledThread"+id+" give up"+" "+cc.cdlist[saletype].name+"\n");
55.                 }
56.             } else {
57.                 //库存充足, 可以购买
58.                 cc.sale(salenum, id, saletype, gt);
59.             }
60.
61.         }
62.     }
63.

```


64. }

(七) 租借线程 RentThread:

```
1. //租借进程
2. //租借的是单张，因此不采用 CD 集合的方式同步
3. class RentThread extends Thread {
4.     CD_pro []cdrentlist;
5.     //区分不同的线程
6.     static int ids;
7.     int id;
8.     Logger lg;
9.     //构造函数
10.    RentThread(CD_pro[] cd,Logger lg){
11.        this.cdrentlist = cd;
12.        this.setDaemon(true);//一直处于循环状态的线程一定要设置为daemon 否则主线程无法结束
13.        id = ++ids;
14.        this.lg = lg;
15.    }
16.
17.    @Override
18.    public void run() {
19.        //种类
20.        Random t = new Random();
21.        //时间
22.        Random r = new Random();
23.        while(true) {
24.            //启动时间为 200ms 以内的随机数
25.            try {
26.                Thread.sleep(r.nextInt(200));
27.            } catch (InterruptedException e) {
28.                // TODO Auto-generated catch block
29.                e.printStackTrace();
30.            }//200 以内的休眠时间
31.
32.            int renttype = t.nextInt(10)+1;//销售十以内的随机种类
33.            //获取到同步对象
34.            synchronized(cdrentlist[renttype-1]) {
35.                //id 为x 的销售线程尝试销售某种 CD 多少张
36.                //lg.info("RentThread"+id+" Try Rent "+cdrentlist[renttype-1].name+"\n");
37.                System.out.print("RentThread"+id+" Try Rent "+cdrentlist[renttype-1].name+"\n");
38.                //要买的 CD 的库存不够
39.                if(cdrentlist[renttype-1].num==0) {
40.                    //随机选择等候还是放弃
```

```
41.     if(r.nextBoolean()) {
42.         cdrentlist[renttype-1].rent(id,cdrentlist[renttype-1],lg);
43.     }else {
44.         //lg.info("RentThread"+id+" give up"+" "+cdrentlist[renttype-1].name+"\n");
45.         System.out.print("RentThread"+id+" give up"+" "+cdrentlist[renttype-1].name+"\n");
46.     }
47. }else {
48.     //库存充足，可以租借
49.     cdrentlist[renttype-1].rent(id,cdrentlist[renttype-1],lg);
50. }
51.
52. }
53. }
54. }
55. }
```