实验报告一

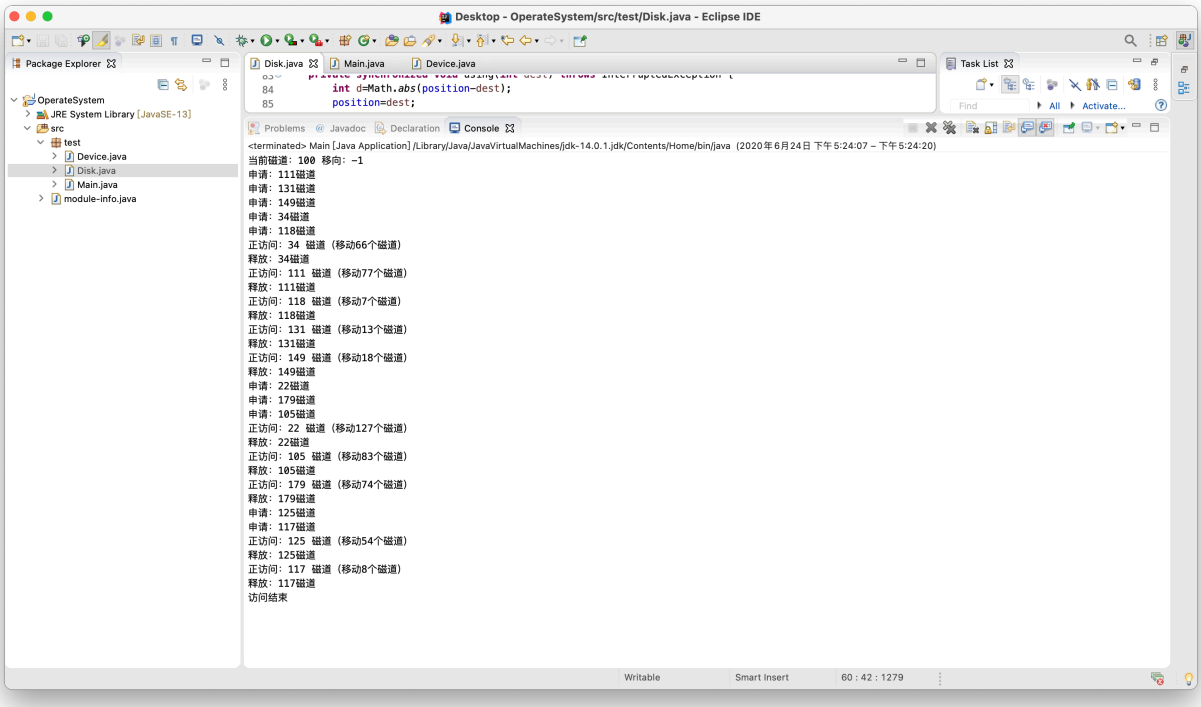
题目：用JAVA同步方法实现磁头引臂调度问题，采用SCAN算法。

目的：深入理解设备调度算法，能用同步方法解决调度问题。

要求：(1) 给出核心调度解法，用JAVA类实现，其中包含require(dest)和release()两个同步方法；(2) 创建若干线程或进程，分别提出某一磁道上某个磁盘块的访问请求，给出调度结果。

说明：(1)假定盘面上共有200个磁道，由外向内依次编号0,...,199，盘面只有一个移动磁头; (2)模拟访问磁盘时打印出磁道编号，并延迟一段时间以表示磁盘I/O操作时间; (3)假定所有块的I/O操作均为同步I/O。

结果：



```java
package test;

public class Device implements Runnable {
    Disk disk;

    public Device(Disk disk) {
        this.disk = disk;
    }

    @Override
    public void run() {
        while (disk.getTimes()>0)
        {
            int max=199,min=0;
            int randomDest = (int) (Math.random()*(max-min)+min);
            disk.require(randomDest);
            try {
                Thread.sleep(1500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

    }
}
```

```java
package test;

public class Main {
    public static void main(String[] args) {

        int threadNumber=6;
        int deviceNumber=threadNumber-1;
        Disk disk=new Disk(100,-1);
        Thread[]threads=new Thread[threadNumber];
        Device[]devices=new Device[deviceNumber];
        threads[0]=new Thread(disk,Integer.toString(0));
        for (int i=1;i<threadNumber;i++)
        {
            devices[i-1]=new Device(disk);
            threads[i]=new Thread(devices[i-1],Integer.toString(i));
        }

        for (int i=0;i<threadNumber;i++)
        {
            threads[i].start();
        }
    }
}
```

代码：

```java
package test;


public class Disk implements Runnable {
    int[]dests;
    int position;
    int flag;
    int times;

    public synchronized int getTimes() {
        return times;
    }

    public Disk(int position, int flag) {
        this.position = position;
        this.flag = flag;
        dests=new int[200];
        times=10;
        System.out.println("当前磁道：“+position+" 移向：“+flag);
    }

    @Override
    public void run() {
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        boolean result=false;
        while (times>0)
        {
            try {
                result=manage();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            if (result)
            {
                times--;
            }
        }
        System.out.println("访问结束");
    }

    private synchronized boolean manage() throws InterruptedException {
        int nextDest= scan();
        if(nextDest!=-1)
        {
            using(nextDest);
            release();
            return true;
        }
        else
        {
            return false;
        }
    }
    void require(int dest)
    {
        System.out.println("申请：“+dest+"磁道");
        dests[dest]++;
    }
    private int scan()
    {
        int next=-1;
        next=hasNext();
        if (next!=-1)
        {
            return next;
        }
        else
        {
            flag=-flag;
            next=hasNext();
            if(next!=-1)
            {
                return next;
            }
        }
        return -1;
    }

    private synchronized void using(int dest) throws InterruptedException {
        int d=Math.abs(position-dest);
        position=dest;
        System.out.println("正访问：“+dest+" 磁道（移动"+d+"个磁道）");
        //模拟访问过程
        try {
            Thread.sleep(1000);
        }catch (Exception e)
        {
        }
        return;
    }

    private synchronized void release()
    {
        dests[position]--;
        System.out.println("释放：“+position+"磁道");
    }

    private int hasNext()
    {
        for(int i=position;i<dests.length&&i>0;i+=flag)
        {
            if(dests[i]>0)
            {
                return i;
            }
        }
        return -1;
    }
}
```