实验报告二

题目：用JAVA语言实现双磁头引臂调度SCAN算法。

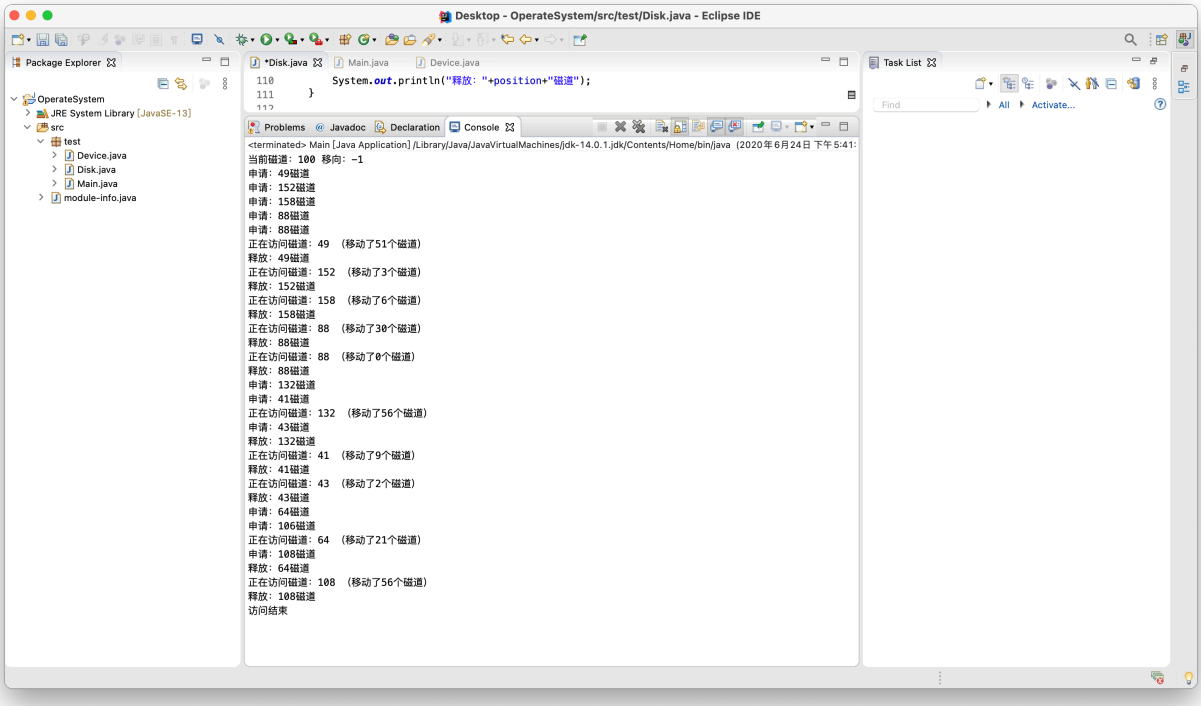
目的：理解硬件提供的多部件支持，并设计算法利用硬件支持提高系统性能。

要求：同实验1。

方法：同实验1。

说明：仍假定盘面上共有200个磁道，由外向内依次编号0,…,199，同一引臂上两个磁头head1和head2，二者相距100个磁道，复位时head1位于磁道0，head2位于磁道100，head1负责0..99磁道上的I/O请求，head2负责100..199磁道上的I/O请求。例如，当head1位于磁道35时，head2位于磁道135。

结果：



代码：

```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++)
```

```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
    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=0;
        if((dest-(dests.length/2))*(position-(dests.length/2))<0)
        {
            if(dest>(dests.length/2))
            {
                position=position+(dests.length/2);
            }
            else
            {
                position=position-(dests.length/2);
            }
        }
        d=Math.abs(position-dest);
        position=dest;
        System.out.println("正访问"+dest+" 磁道（移动了"+d+"个磁道）");
        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()
    {
        int head1;
        int head2;
        if(position<dests.length/2)
        {
            head1=position;
        System.out.println("释放：“+position+"磁道");
    }

    private int hasNext()
    {
        int head1;
        int head2;
        if(position<dests.length/2)
        {
            head1=position;
            head2=position+(dests.length/2);

        }
        else
        {
            head1=position-(dests.length/2);
            head2=position;
        }
        for(;head2<dests.length&&head1>=0;head1+=flag,head2+=flag)
        {
            if(dests[head1]>0)
            {
                return head1;
            }
            else if (dests[head2]>0)
            {
                return head2;
            }
        }
        return -1;
    }

}
```