使用单标记法实现互斥

实现平台：java

实现思路：

使用lock来标记当前进程的运行状态，在进入区时当lock为true时，表示有进程正在执行，这是该进程就会在原地循环等待，当lock为false时，表示处理机可以使用，这时进入临界区并把lock上锁，执行代码，结束区再把lock解锁，临界区使用权交给另一个进程。

PS: java进行的是线程操作，效果不是很好。

实现效果：

thread1取1元，账户还有65元
thread1取1元，账户还有64元
thread1取1元，账户还有63元
thread2取1元，账户还有62元
thread2取1元，账户还有61元

... ...

thread1取1元，账户还有6元
thread2取1元，账户还有5元
thread2取1元，账户还有4元
thread1取1元，账户还有3元
thread1取1元，账户还有2元
thread1取1元，账户还有1元
thread1取1元，账户还有0元

代码：

```
public class thread implements Runnable {
    String threads;//线程名
    int n = 66;//账户现在的余额

    static int isLock[] = new int[2];//未实现的peterson
    static boolean lock = false;//使用一个全局变量

    public synchronized void draw_money() {
        if (n > 0) {
            while (lock == true) ;//忙则等待
            lock = true;//上锁
            n--;//每次取1元
            System.out.println(this.threads + "取1元，账户还有" + n + "元");
```

```
        }
        lock = false;//解锁
        try {//休眠实现有限等待
            Thread.sleep(100);//取完后睡眠一秒
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }


    public void run() {
        while (n > 0) {
            threads = Thread.currentThread().getName();
            draw_money();
        }
    }

    public static void main(String[] args) {
        thread m = new thread();
        Thread thread1 = null, thread2 = null;

        thread1 = new Thread(m);//开m1，m2两个线程
        thread2 = new Thread(m);
        thread1.setName("thread1");
        thread2.setName("thread2");
        try {
            thread1.start();
            thread2.start();
        } catch (Exception e) {
        }
    }
}
```