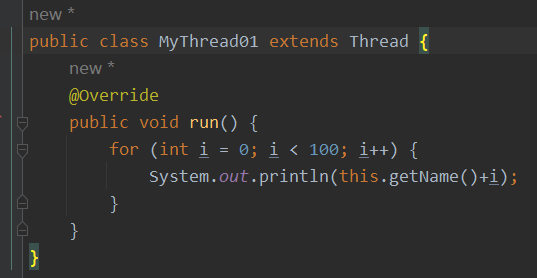
# 多线程

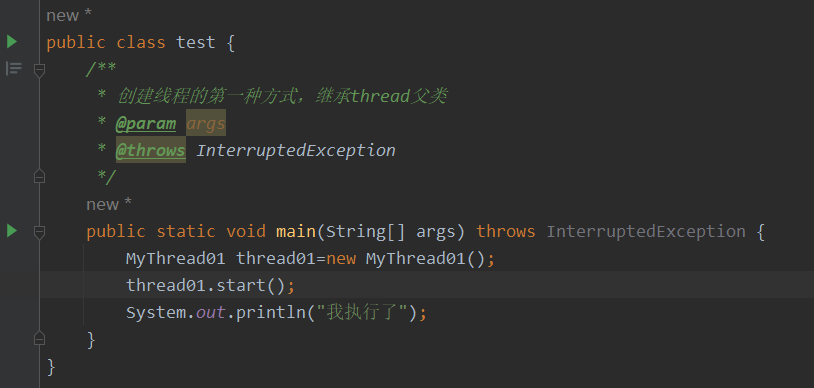
在操作系统中分进程，然后再java进程中又有线程

## 创建线程的三种常规方式

### 继承thread（线程类）



重写run方法在run方法中写当前线程需要执行的任务



在测试类创建继承的线程类并且运行线程，此方法无返回值

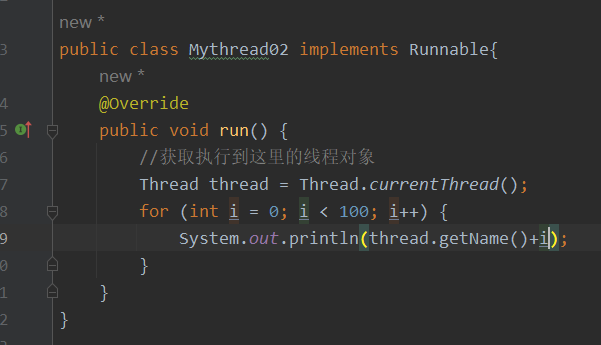
### 实现runnable接口（无返回值）

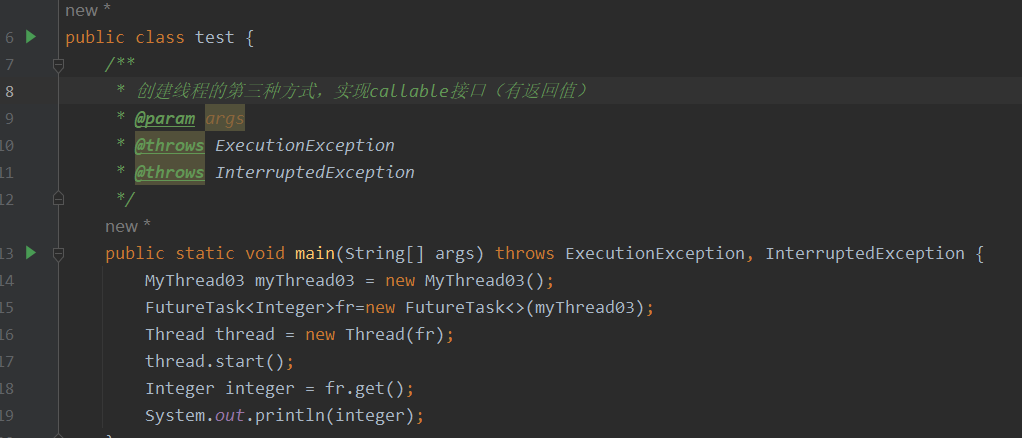




一二方法均无返回值

### 实现callable接口（有返回值，泛型需要定义返回值类型）





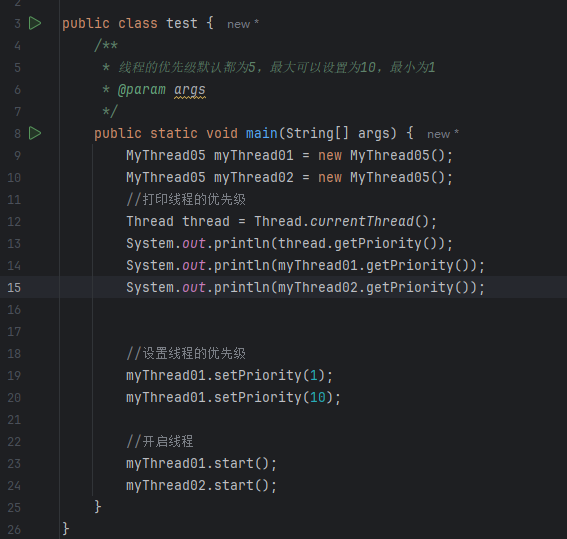
## 多线程的常规方法



在线程中的优先级最小是1最大是10，默认是5；

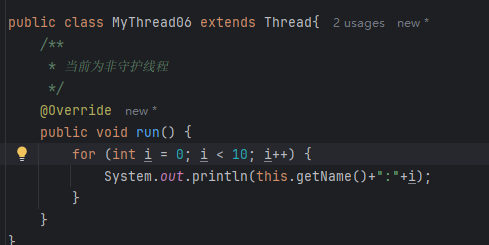


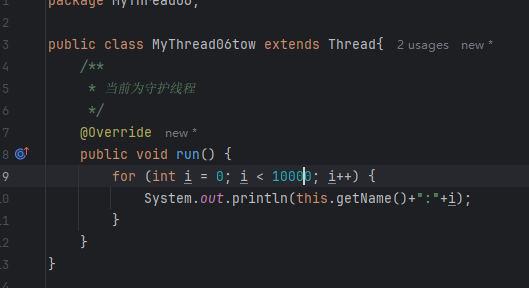
## 线程的优先级

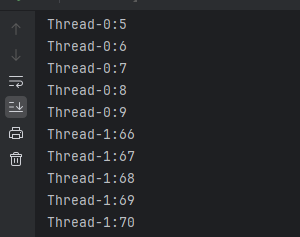


## 守护线程

当非守护线程全部执行完了 ，守护线程就没有存在的必要了，就会自行结束，举例，例如当聊天窗口中聊天窗为一个进程，下载文件为一个进程，当聊天窗口关闭了，则下载文件则自动关闭。







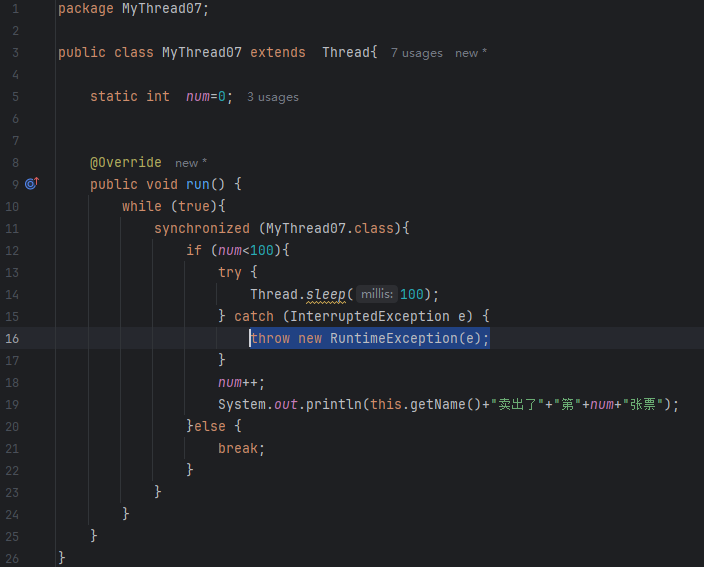
当非守护线程都结束了，即使守护线程的任务没完成也会结束

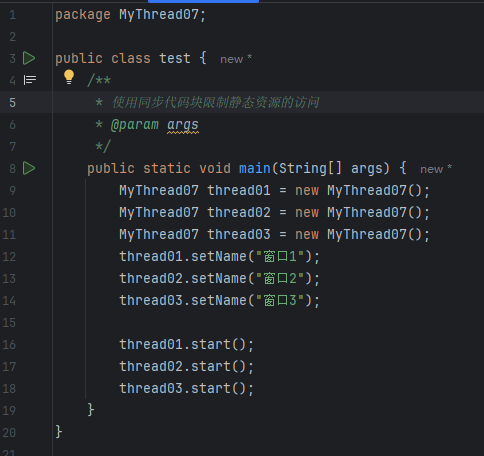
## 同步代码块

同步代码块（synchronized block）在 Java 中用于控制对共享资源的访问，以确保线程安全。它的主要作用是防止多个线程同时访问同一资源，从而避免数据的不一致性和潜在的并发问题。

注意\*在string中StringBuffer: 是线程安全的，所有的方法都是同步的。这意味着它可以在多个线程中安全使用，但由于同步的开销，性能较低，StringBuilder: 是非线程安全的，不同步，性能更高，适合单线程环境下使用。StringBuffer的所有方法都是用synchronized修饰的

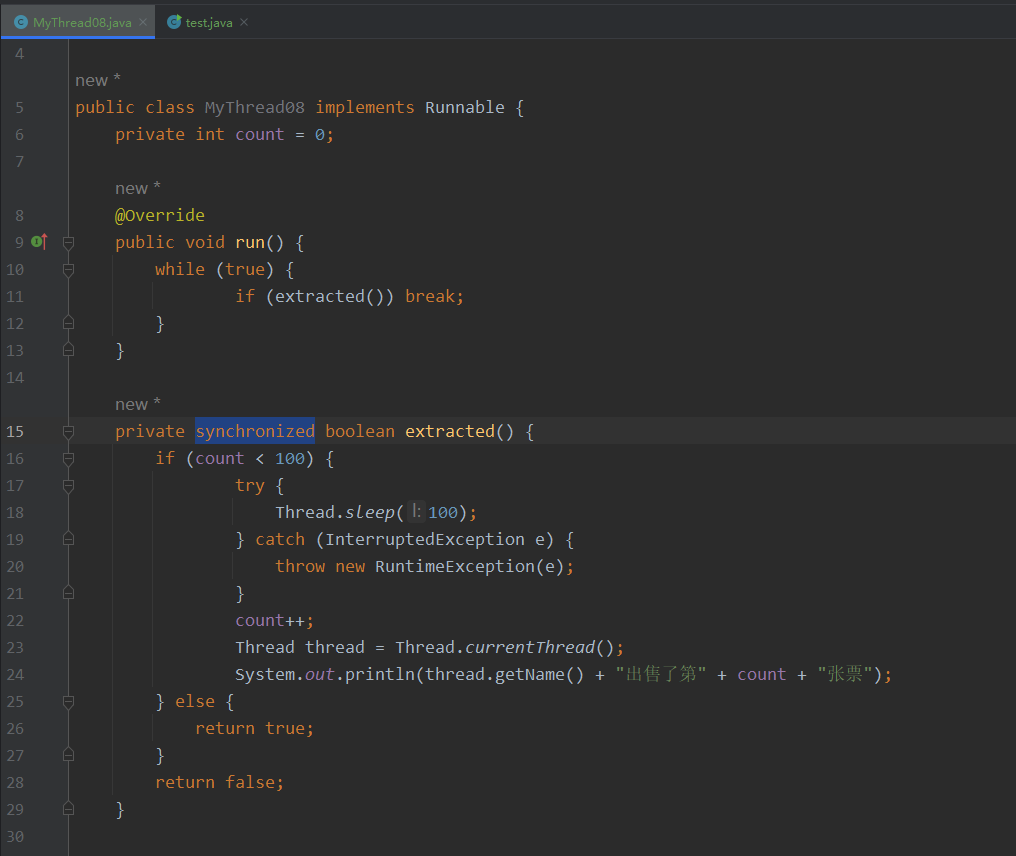
使用synchronized限制静态资源的访问，参数里面需要放一个唯一的对象作为锁，一般我们区当前类的字节码文件





## 同步方法

将同步代码块的代码抽取出来做成同步方法 需要在方法上加synchronize关键字



## Lock锁

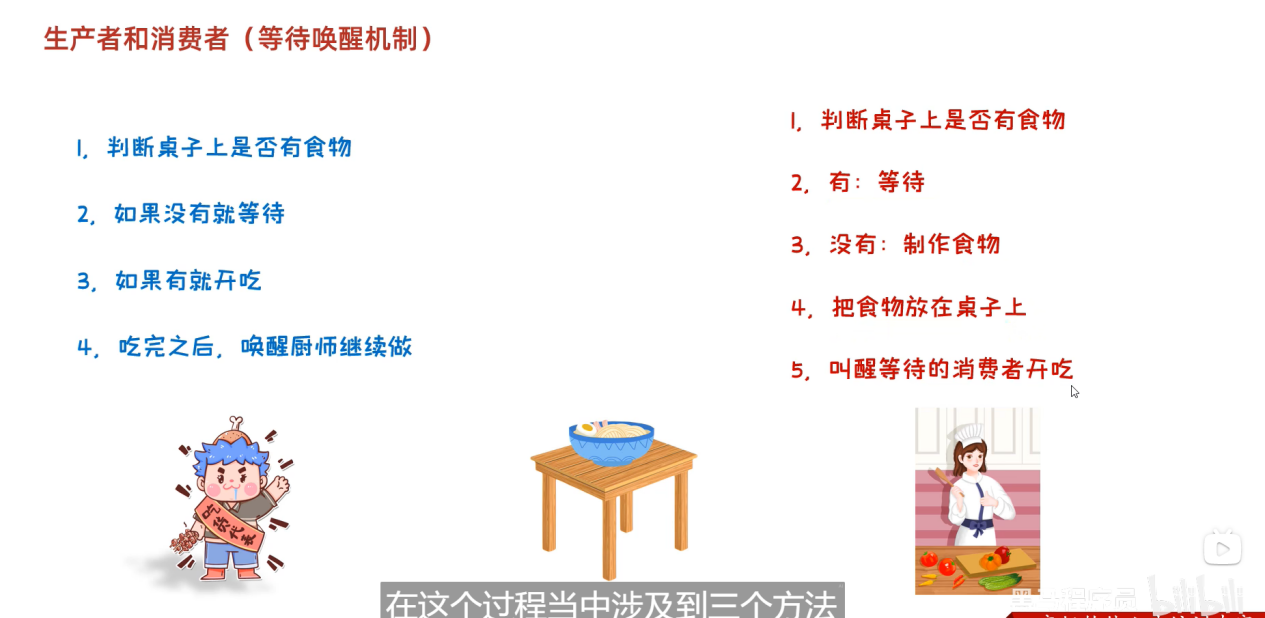
synchronize是关键字，而Lock 是一个接口，提供比 synchronized 更灵活的锁机制，synchronize是自动开锁和关锁，而lock是需要手动开关锁的，lock锁需要提前创建子类锁对象ReentrantLock，使用里面的.lock上锁使用.unlock解锁



## 死锁

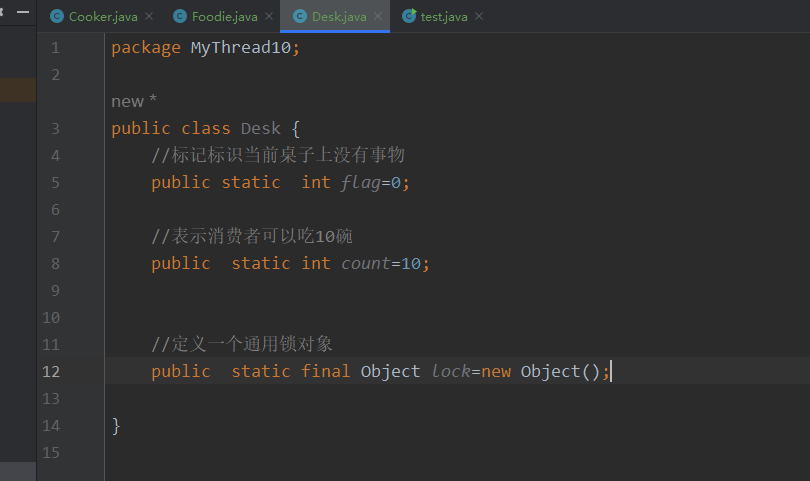
死锁是一个错误，是等待资源抢夺导致卡死，避免锁的时候写锁的时候不要让两个锁嵌套

## 生产者与消费者（等待唤醒机制）

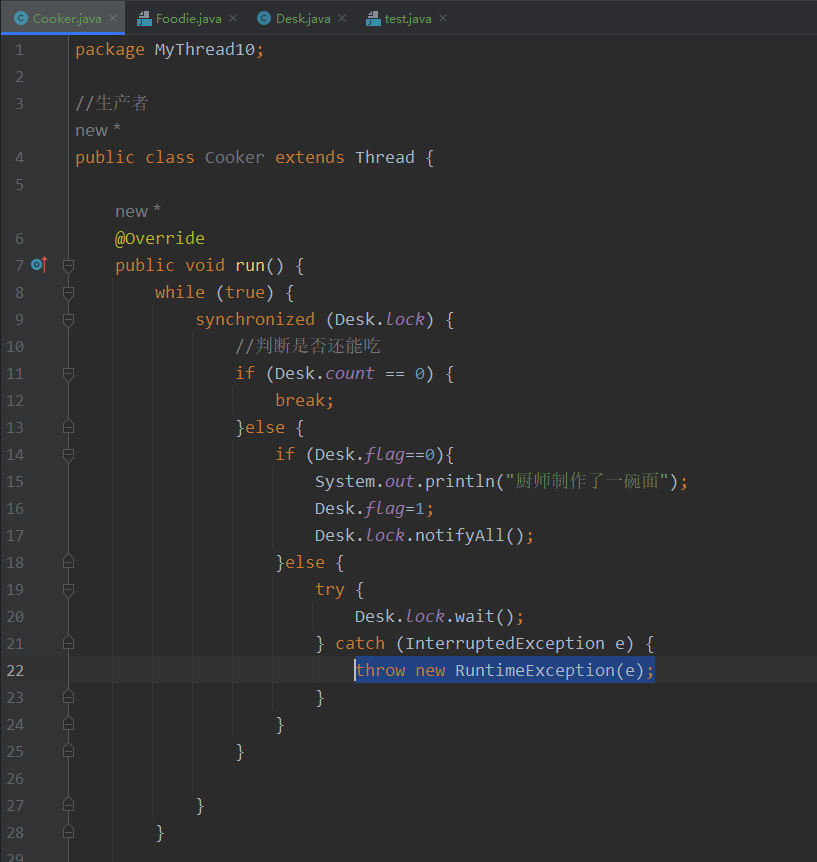




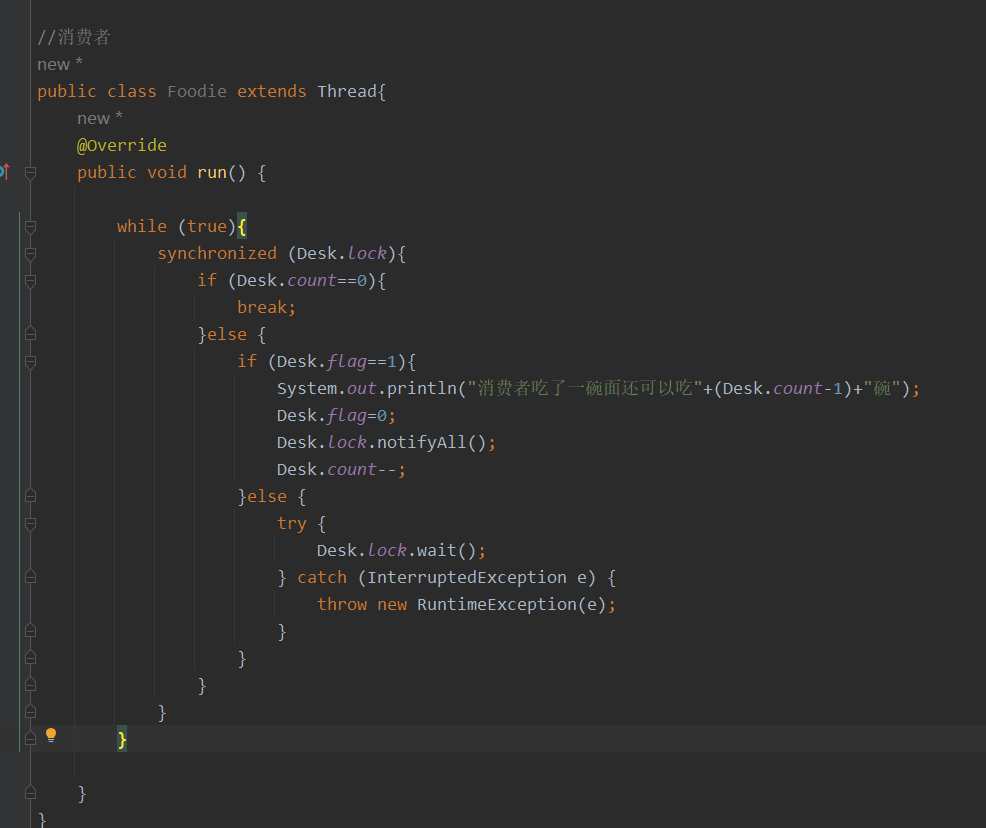
桌子类：定义一个变量，定义线程结束条件，定义一个锁对象



生产者类：判断是否还需要生产，判断是否结束，并且调用方法唤醒或等待

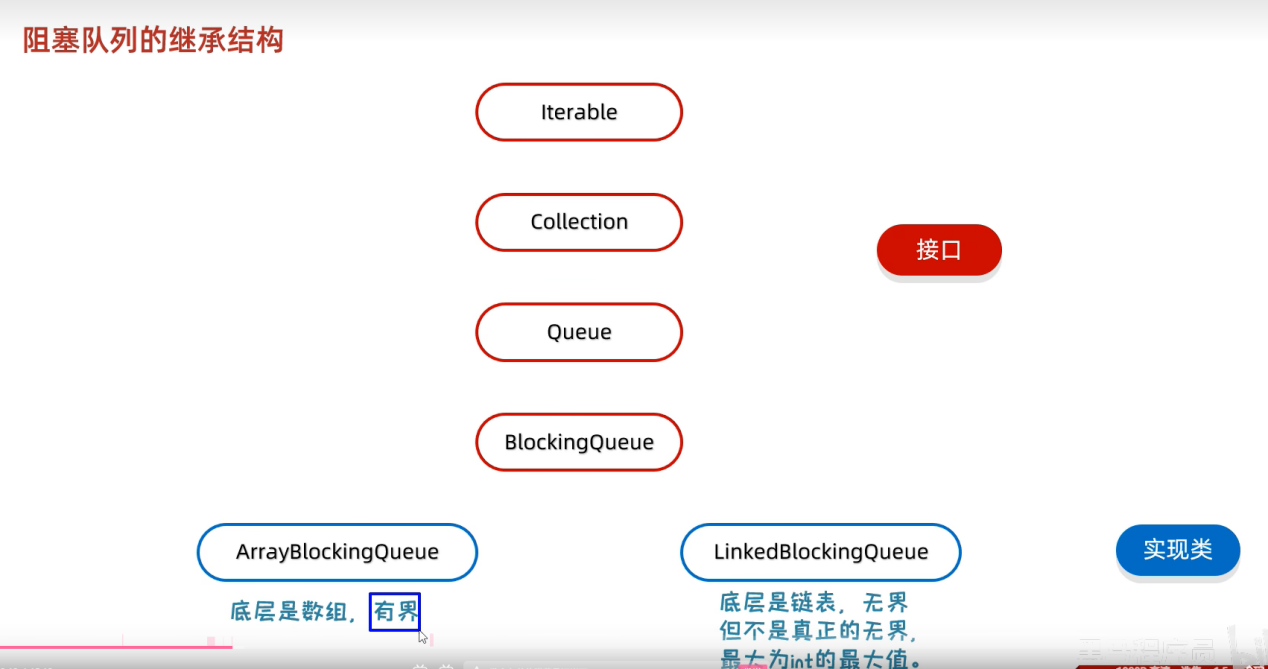


消费者类：

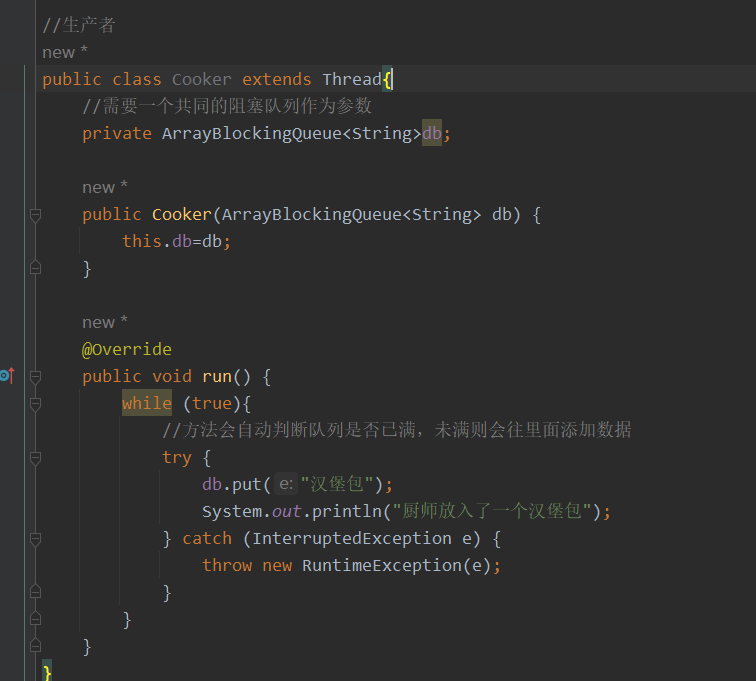


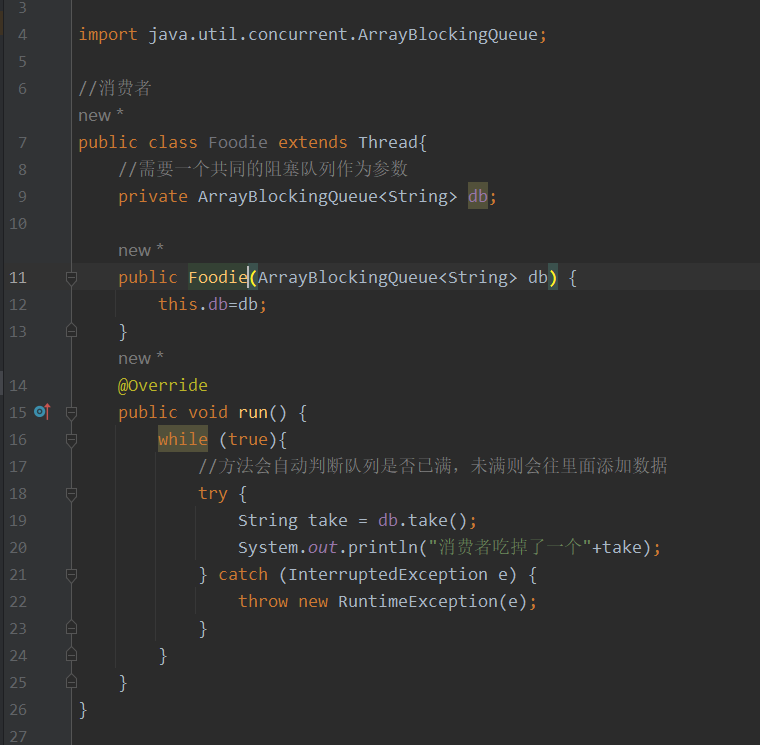
## 阻塞队列的实现

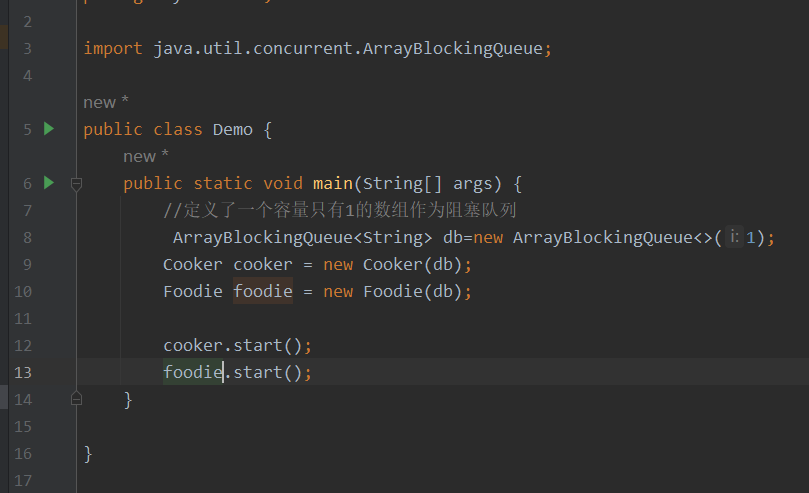
定义一个阻塞队列作为信号量，生产者和消费者必须使用同一个阻塞队列



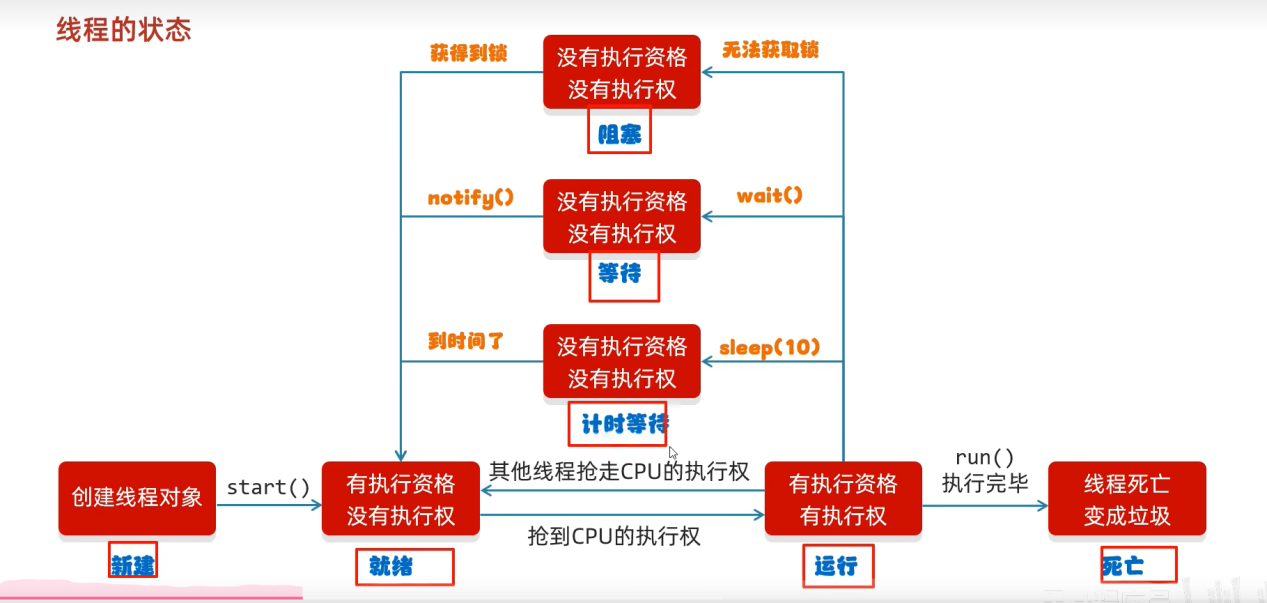


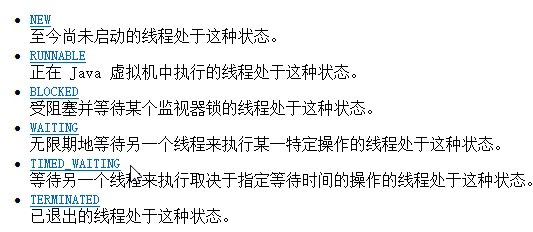






## 线程的状态





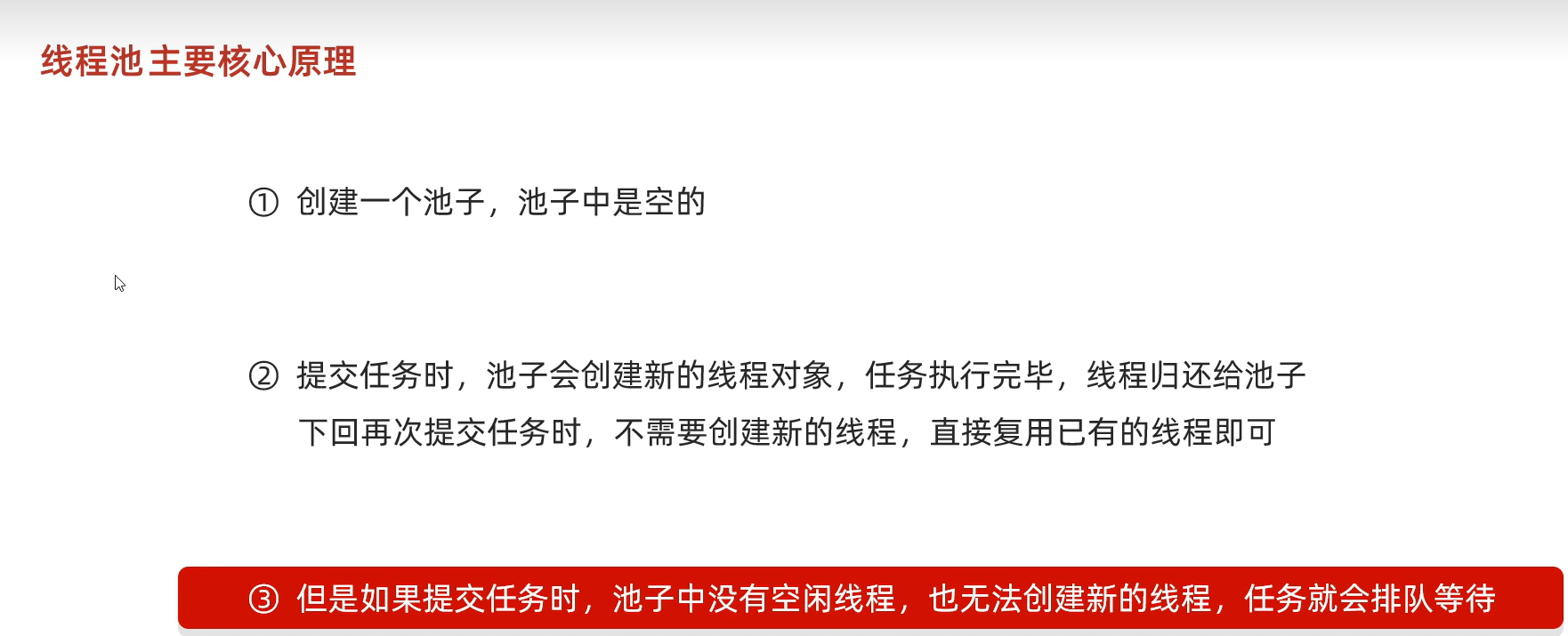
标准六个状态



## 线程池

常规创建线程用完即销毁，但是线程池相当于一个线程容器，可以在里面存放线程用到的时候拿出来用完放回去，不设置线程池的大小线程池的上线为int最大值（相当于无上线），设置上线，任务太多任务则会在阻塞队列排队

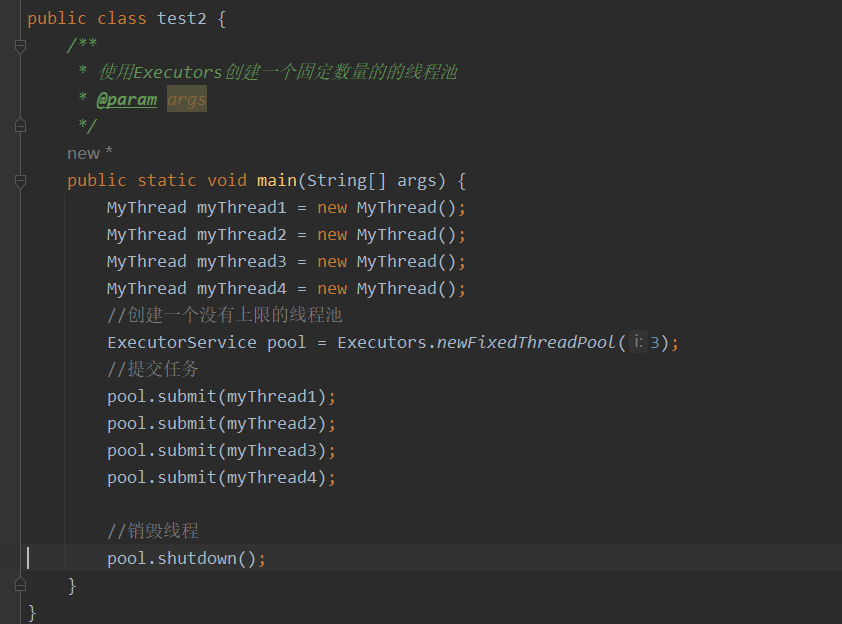




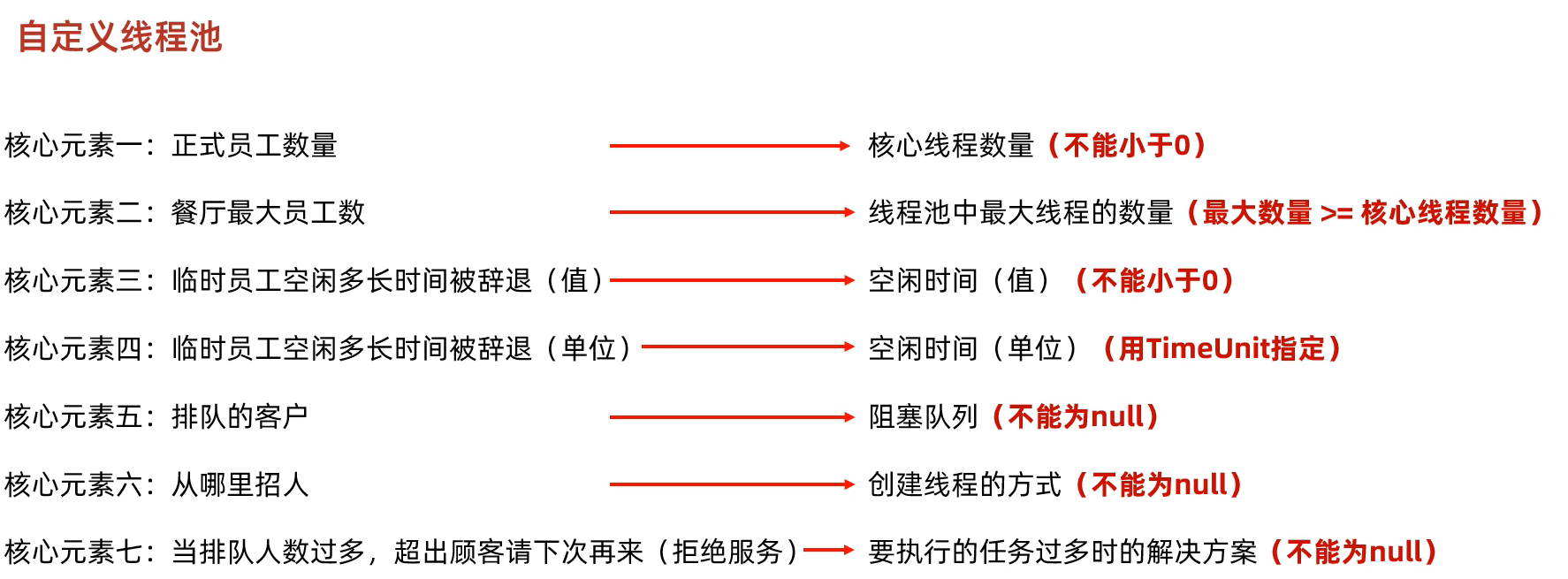
### 工具类创建

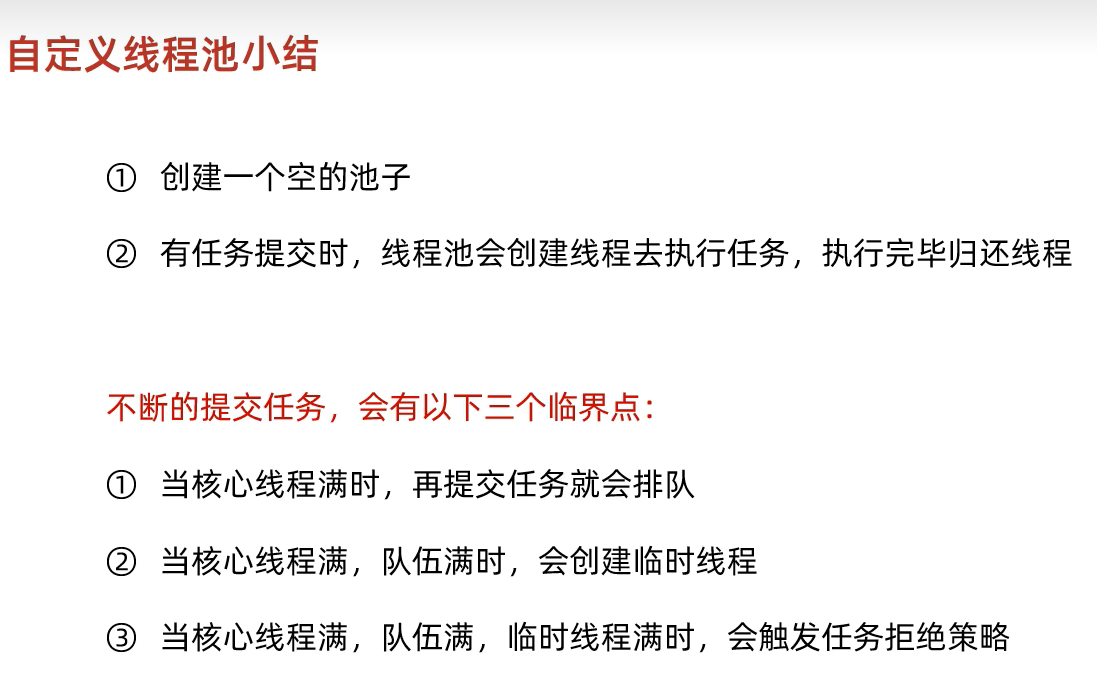


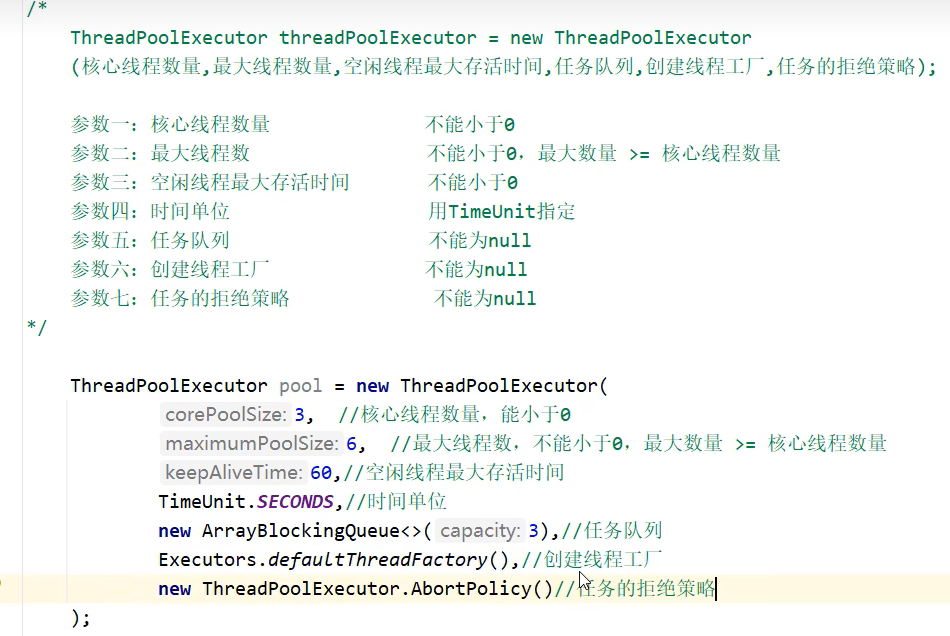




### 自定义线程池

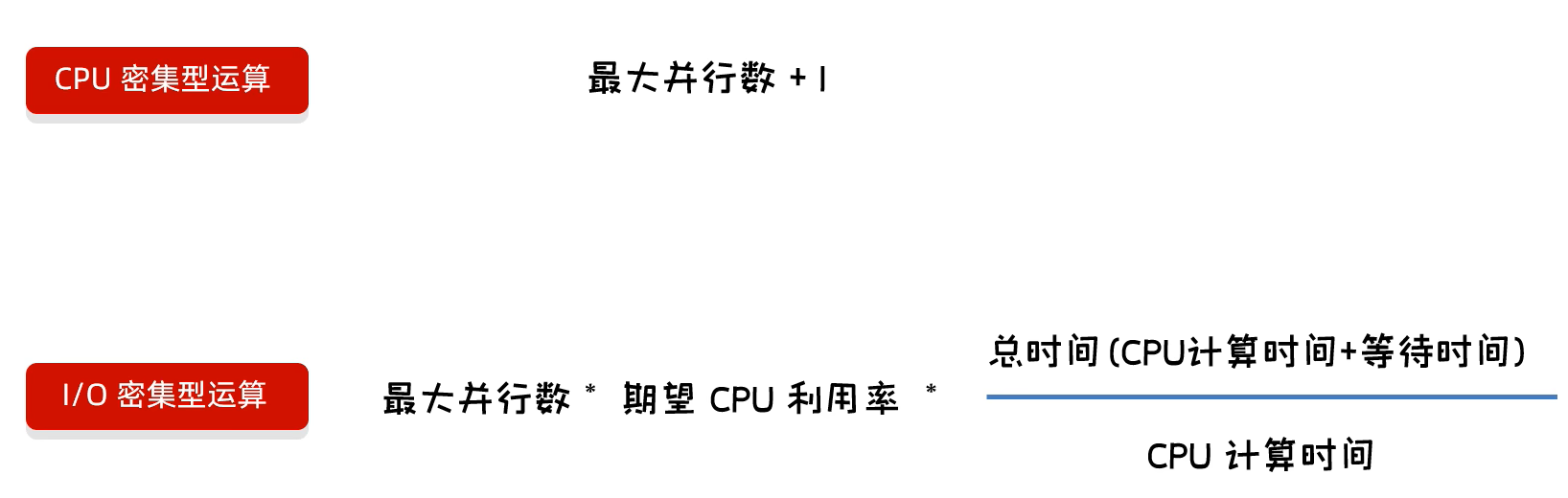


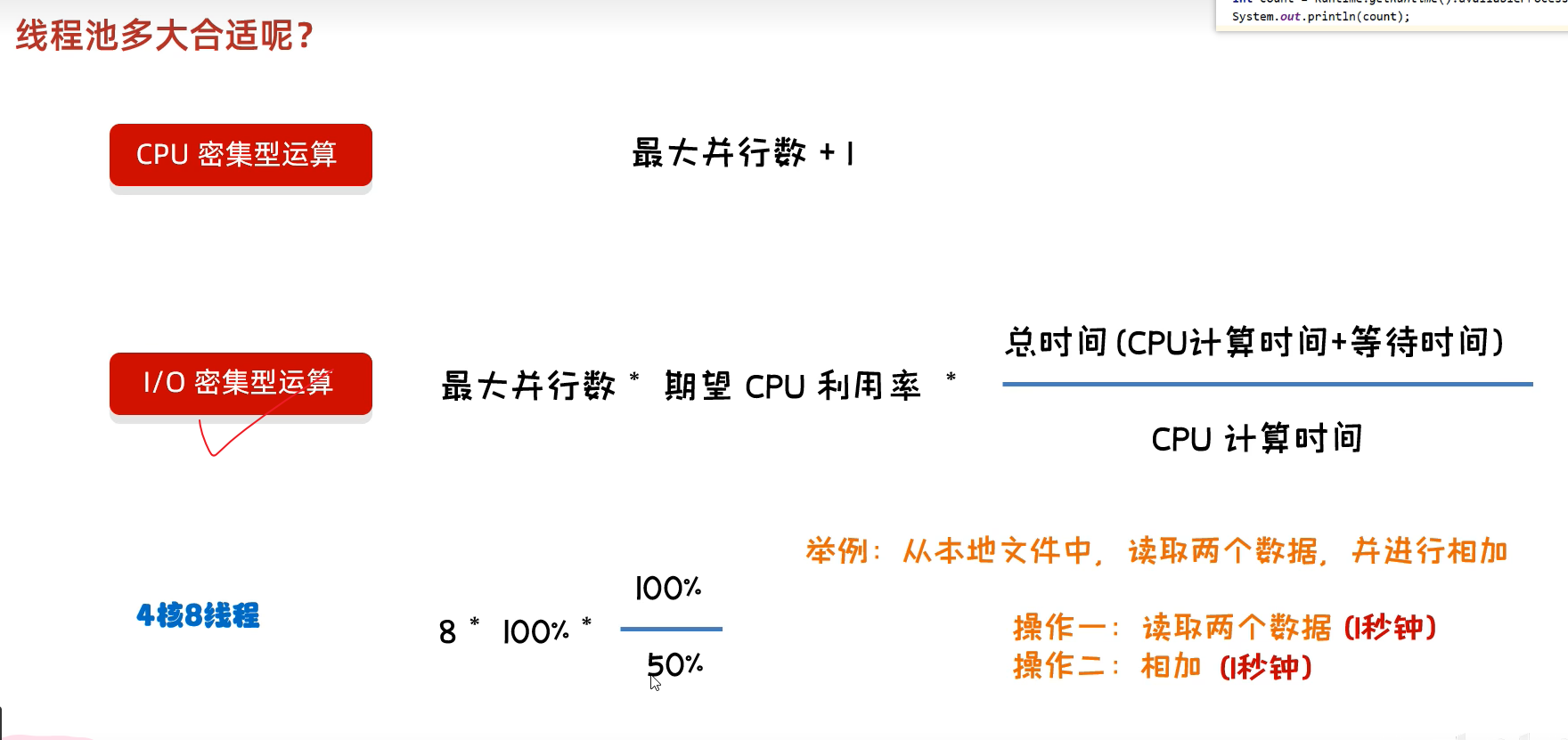






### 线程池大小





一般的项目都是i/0密集型，时间用工具计算