# 等待和通知

1. 给出条件的定义

触发某事件所需要满足的要求 //持续执行的前置条件

1. 描述支持条件的API

wait()等待某个条件成立 ,notify(),notifyAll()在这个条件成立时通知处于等待的线程

1. 判断对错：wait（）方法可以被中断

对

1. 你会调用什么方法来唤醒在同一对象监听器上等待的全部线程

notifyAll

1. 判断对错：一条已经获取到锁的线程在调用Object的wait（）方法时并不会释放锁

错，会释放锁。

1. 给出条件队列的定义

//条件队列是一种数据结构，用于存储那些等待某个条件成立的线程，这些等待中的线程称为等等待集合。

1. 当你在同步上下文之外调用这组API的方法时会发生什么

死锁 //会导致IllegalMonitorStateException发生

1. 给出假唤醒的定义

线程不是因为通知、中断或者超时被唤醒

1. 为什么你需要在一个循环的上下文中调用wait（）方法？

为了确认唤醒时是否满足条件而不是被意外唤醒 //确保活跃性和安全

1. 创建一个等待的应用程序，示范一种被称作关卡的高级别的同步构造。这一构造允许多条线程达到同一个同步点（关卡），并且一直等待直到关卡被其它线程解锁，才能继续全部执行。

Main（）方法首先会为这些线程创建一个runnable，线程会在关卡处等待。Runnable打印一条消息表明这条线程正处于等待中，同时递增一个计数器，睡眠2秒，然后等待（确保考虑了假唤醒问题）。唤醒后，线程输出一条消息，表明线程终止。之后main（）方法创建3个线程对象并且启动这3条线程去执行runnable。接下来，main（）方法创建另一个runnable，它会反复地睡200毫秒直到计数器等于3，这时，它会通知所有处于等待中的线程。最后，main（）方法为第二个runnable创建一个线程对象并启动之。

