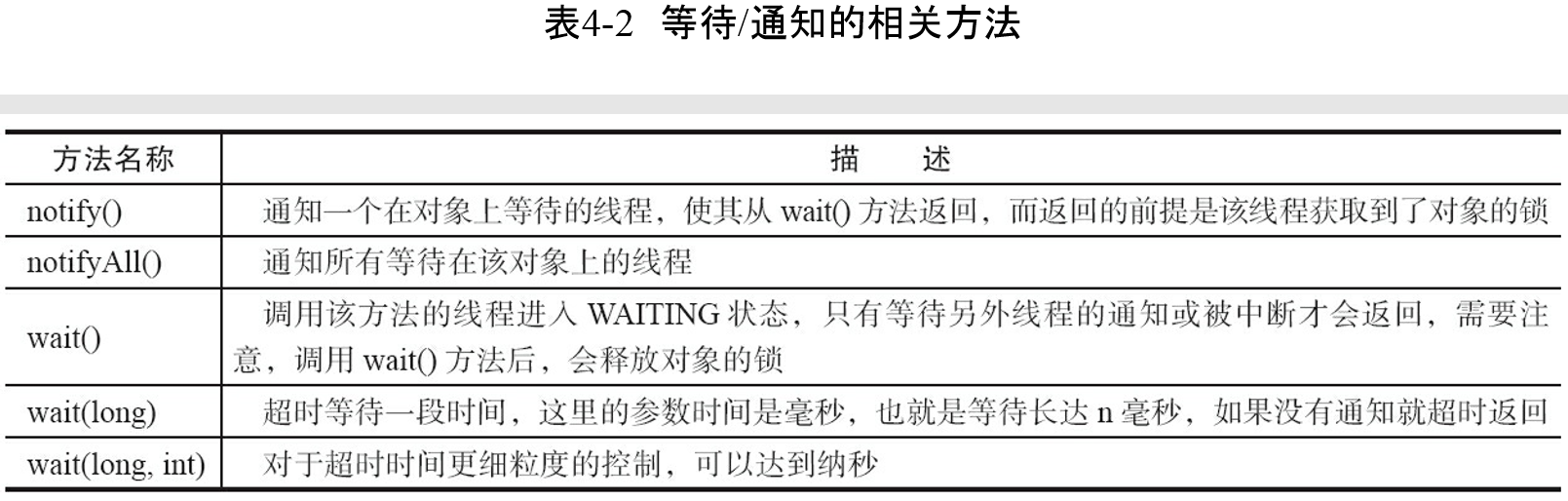
# 中断：

中断可以理解为线程的一个标识位属性，它表示一个运行中的线程是否被其他线程进行 了中断操作。中断好比其他线程对该线程打了个招呼，其他线程通过调用该线程的interrupt()

方法对其进行中断操作；

# 等待通知机制：



等待/通知机制，就是指一个线程A调用了对象O的wait（）方法进入等待状态，而另一个线程B调用了对象的notify（）或者notifyAll（）方法，线程A收到通知后从对象O的wait（）方法返回，进而执行后续操作；上述两个线程通过对象O来完成交互，而对象上的wait（）和notify/notifyAll（）的关系就如同开关信号一样，用来完成等待方和通知方之间的交互工作；

了调用wait()、notify()以 及notifyAll()时需要注意的细节，如下。

**1）使用wait()、notify()和notifyAll()时需要先对调用对象加锁。**

**2）调用wait()方法后，线程状态由RUNNING变为WAITING，并将当前线程放置到对象的**

**等待队列。**

**3）notify()或notifyAll()方法调用后，等待线程依旧不会从wait()返回，需要调用notify()或 notifAll()的线程释放锁之后，等待线程才有机会从wait()返回。**

**4）notify()方法将等待队列中的一个等待线程从等待队列中移到同步队列中，而notifyAll() 方法则是将等待队列中所有的线程全部移到同步队列，被移动的线程状态由WAITING变为 BLOCKED。**

**5）从wait()方法返回的前提是获得了调用对象的锁。**

# Thread.join()的使用

如果一个线程A执行了thread.join()语句，其含义是：当前线程A等待thread线程终止之后才从thread.join()返回。线程Thread除了提供join（）方法之外，还提供了join(long millis)和join(long millis,int nanos)两个具备超时特性的方法。这两个超时方法表示，如果线程thread在给定的超时时间里没有终止，那么将会从该超时方法中返回。

# Volatile：

1. 线程可见性
2. 防止指令重排序；

问题：DCL单例需不需要加volatile？

这里涉及到对象的半初始化状态；必须加volatile;

缓存行64个字节是CPU同步的基本单位，缓存行隔离会比伪共享效率要高；