当一个方法后面声明可能会抛出InterruptedException 异常时，说明该方法是可能会花一点时间，但是可以取消的方法。

抛InterruptedException的代表方法有：

1. java.lang.Object 类的 wait 方法

2. java.lang.Thread 类的 sleep 方法

3. java.lang.Thread 类的 join 方法

需要花点时间的方法 执行wait方法的线程，会进入等待区等待被notify/notify All。在等待期间，线程不会活动。执行sleep方法的线程，会暂停执行参数内所设置的时间。 执行join方法的线程，会等待到指定的线程结束为止。 因此，上面的方法都是需要花点时间的方法.

可以取消的方法 因为需要花时间的操作会降低程序的响应性，所以可能会取消/中途放弃执行这个方法。

这里主要是通过interrupt方法来取消。

1. sleep方法与interrupt方法 interrupt方法是Thread类的实例方法，在执行的时候并不需要获取Thread实例的锁定，任何线程在任何时刻，都可以通过线程实例来调用其他线程的interrupt方法。 当在sleep中的线程被调用interrupt方法时，就会放弃暂停的状态，并抛出InterruptedException异常，这样一来，线程的控制权就交给了捕捉这个异常的catch块了。

2. wait方法和interrupt方法 当线程调用wait方法后，线程在进入等待区时，会把锁定接触。当对wait中的线程调用interrupt方法时，会先重新获取锁定，再抛出InterruptedException异常，获取锁定之前，无法抛出InterruptedException异常。

3. join方法和interrupt方法 当线程以join方法等待其他线程结束时，一样可以使用interrupt方法取消。因为join方法不需要获取锁定，故而与sleep一样，会马上跳到catch程序块

-- interrupt方法干了什么？

interrupt方法其实只是改变了中断状态而已。 而sleep、wait和join这些方法的内部会不断的检查中断状态的值，从而自己抛出InterruptEdException。 所以，如果在线程进行其他处理时，调用了它的interrupt方法，线程也不会抛出InterruptedException的，只有当线程走到了sleep, wait, join这些方法的时候，才会抛出InterruptedException。若是没有调用sleep, wait, join这些方法，或者没有在线程里自己检查中断状态，自己抛出InterruptedException，那InterruptedException是不会抛出来的。 isInterrupted方法，可以用来检查中断状态 Thread.interrupted方法，可以用来检查并清除中断状态。

https://www.cnblogs.com/canda/p/7835823.html