Thread类

Runnable接口

当线程要sleep休眠时，其必须捕捉InterruptedException异常

yield()让出处理器

Java 语言中的 volatile 变量可以被看作是一种 “程度较轻的 synchronized”

Java实现同步的五种方法：

1. 同步方法：

即有synchronized关键字修饰的方法，由于java的每一个对象都有一个内置锁，当用此关键字修饰方法时，内置锁会保护整个方法。在调用该方法前，需要获得内置锁，否则就处于阻塞状态

（synchronized也可以修饰静态方法，如果调用该静态方法，将会锁住整个类）

1. 同步代码块

同步是一种高开销的操作，因此应该尽量减少同步的内容，没必要同步整个方法，使用该关键字同步关键代码即可

1. 使用特殊域变量（volatile）实现线程同步

Volatile关键字为域变量的访问提供了一种免锁机制

使用volatile修饰域相当于告诉虚拟机该域可能会被其他线程更新

因此每次使用该值域就要重新计算，而不是使用寄存器中的值

Volatile不会提供任何源自操作，它也不能用来修饰final类型的变量

（正确使用volatile变量的条件）

对变量的写操作不依赖于当前值

该变量没有包含在具有其他变量的不定式中

1. 使用重入锁实现线程同步

Java.util.concurrent支持同步

ReenreantLock类的常用方法：

ReenreantLock（）：创建一个ReentranLock实例

Lock（）:获得锁

Unclok（）：释放锁

1. 使用局部变量实现线程同步

如果使用ThreadLocal管理变量，则每一个使用该变量的线程都获得该变量的副本，副本之间相互独立，这样每一个线程都可以随意修改自己的变量副本，而不会对其它线程有影响