Java 多线程同步的五种方法

1. 同步方法

用synchronized关键字修饰方法。 由于java的每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。

2. 同步代码块

用synchronized关键字修饰语句块。被该关键字修饰的语句块会自动被加上内置锁，从而实现同步

注：同步是一种高开销的操作，因此应该尽量减少同步的内容。通常没有必要同步整个方法，使用synchronized代码块同步关键代码即可。

3. Volatile

a.volatile关键字为域变量的访问提供了一种免锁机制

b.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

c.因此每次使用该域就要重新计算，而不是使用寄存器中的值

d.volatile不会提供任何原子操作，它也不能用来修饰final类型的变量

volatile不能保证原子操作导致的，因此volatile不能代替 synchronized。此外volatile会组织编译器对代码优化。它的原理是每次要线程要访问volatile修饰 的变量时都是从内存中读取，而不是存缓存当中读取，因此每个线程访问到的变量值都是一样的。这样就保证了同步。

4. 使用重入锁实现线程同步

在JavaSE5.0中新增了一个java.util.concurrent包来支持同步。ReentrantLock类是可重入、互斥、实现了Lock接口的锁， 它与使用synchronized方法和快具有相同的基本行为和语义，并且扩展了其能力。

ReenreantLock类的常用方法有：

ReentrantLock() : 创建一个ReentrantLock实例

lock() : 获得锁

unlock() : 释放锁

注：ReentrantLock()还有一个可以创建公平锁的构造方法，但由于能大幅度降低程序运行效率，不推荐使用

5. ThreadLocal

ThreadLocal 类的常用方法

ThreadLocal() : 创建一个线程本地变量

get() : 返回此线程局部变量的当前线程副本中的值

initialValue() : 返回此线程局部变量的当前线程的"初始值"

set(T value) : 将此线程局部变量的当前线程副本中的值设置为value

ThreadLocal的原理：

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，而不会对其他线程产生影响。

即每个线程运行的都是一个副本，也就是说存钱和取钱是两个账户，只是名字相同而已，两个线程间的count没有关系。所以就会发生上面的效果。

ThreadLocal与同步机制

a.ThreadLocal与同步机制都是为了解决多线程中相同变量的访问冲突问题

b.前者采用以”空间换时间”的方法，后者采用以”时间换空间”的方式

ThreadLocal并不能替代同步机制，两者面向的问题领域不同。

1：同步机制是为了同步多个线程对相同资源的并发访问，是为了多个线程之间进行通信的有效方式；

2：而threadLocal是隔离多个线程的数据共享，从根本上就不在多个线程之间共享变量，这样当然不需要对多个线程进行同步了。