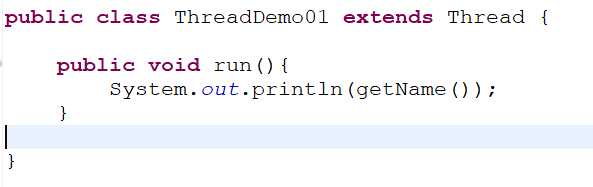
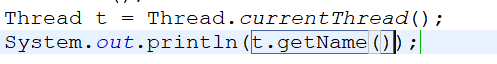
# 多线程

1. 多线程需要实现Runnable接口的run方法或者继承Thread类重写run方法。
2. 创建和启动一个线程，①创建Thread子类对象②调用父类的start 方法，使该线程开始执行，jvm调用改线程的run方法③只能调用一次；start才开线程。
3. 线程默认名，主线程为“mian”,其他为Thread-num；通过getName()来获取线程名

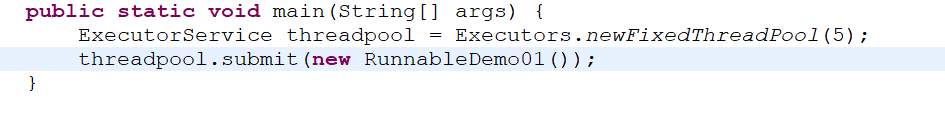


1. 获取main的执行线程对象 static Thread currentThread()来获取当前的线程。



1. 设置线程名使用setName(string) 使用子类对象调用setName();后调用start();
2. Static Sleep(毫秒值);Thread.sleep，休息多少毫秒后自动运行；
3. 实现Runnable接口需要实现run方法，启动线程过程①创建实现Runnable的对象②使用Thread的构造函数参数为实现Runnable的对象,再调用start();
4. Object 类中Wait()进入等待，不能自动运行，只能使用notify()去唤醒
5. 受阻塞：线程具有cpu的执行资格，等待cpu资源；
6. 休眠等待，无cpu的执行资格。
7. 线程池的创建，使用Executors线程池创建工厂类Executors.newFixedThreadPool(线程池中线程的个数)返回的是

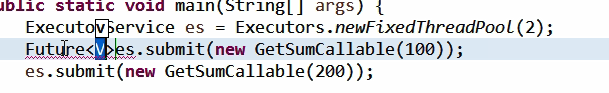
ExecutorService接口，接触此接口的实现类调用sumit(Runnable);线程不死。使用shutdown()干掉线程池。



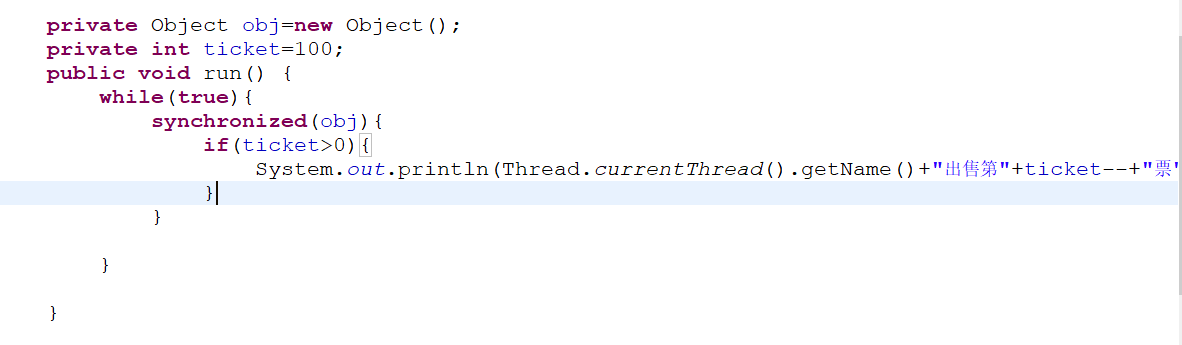
1. 使用线程程序的第三方法ExecutorService threadpool=Executors.*newFixedThreadPool*(5);

Threadpool.sumit(callable接口)

1. 异步计算使用线程池， 调用callable子类，在子类使用有参构造器，来传参

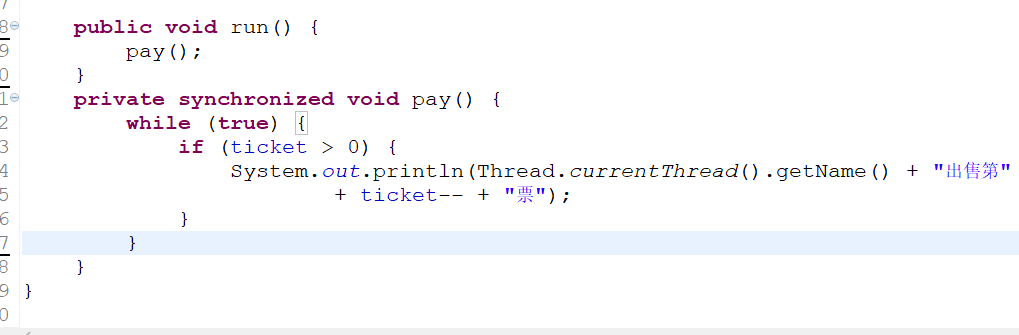


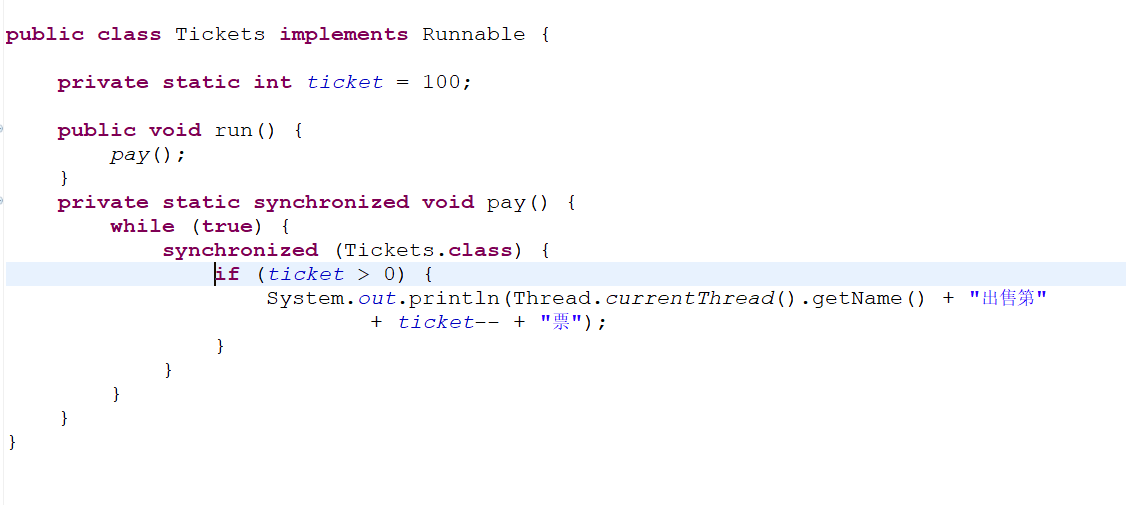
1. 解决线程不安全的方法① 同步代码块synchronized(任意对象（对象监视器）){线程操作的共享数据}



过程：线程遇到同步代码块后，线程判断同步锁还又没有同步锁，如果有进入同步中，去执行线程代码，执行完毕后出去了同步代码块，线程在将锁对象还回去。

②同步方法 在方法的返回值之前加入synchronized,同步对象是本类的引用this，如果方法是静态，有同步但是不是this，锁是本类的.class





③Jdk1.5Lock接口替換了synchronized，

