# 创建线程的两种方法

继承Thread 覆写run方法

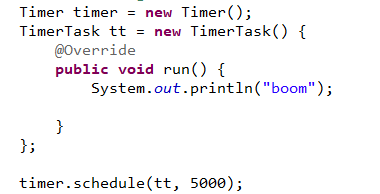
实现Runnable接口 赋值给Thread

# 定时器

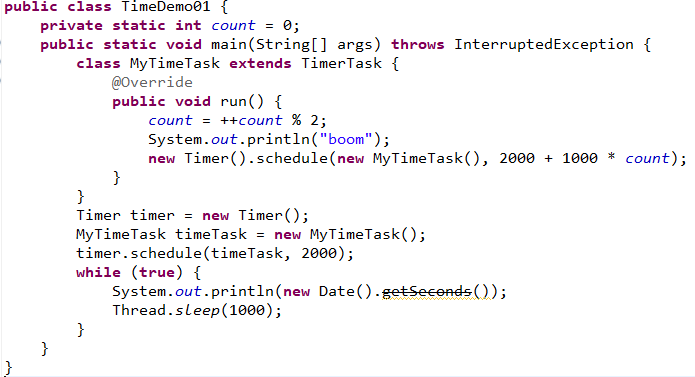
Timer类和TimeTask类

Timer类可以调度任务，TimerTask则是通过在run()方法里实现具体任务

当Timer的构造器被调用时，它创建了一个线程，这个线程可以用来调度任务

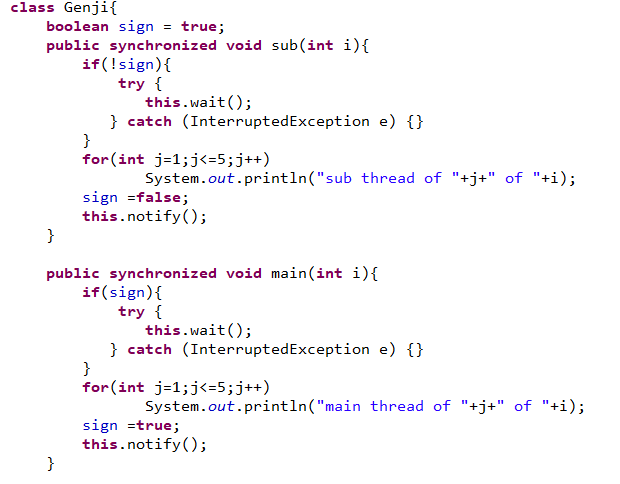


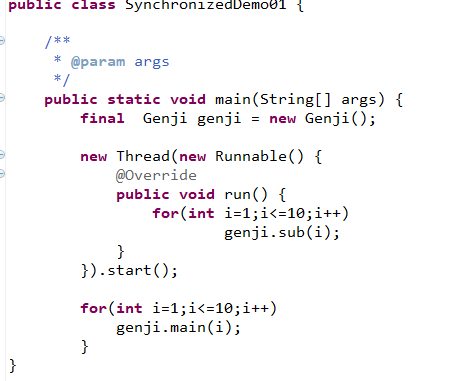
实现线程交互(生成多个线程 执行内容再生成线程)



### 线程通信

变量访问 调用wait notify等函数

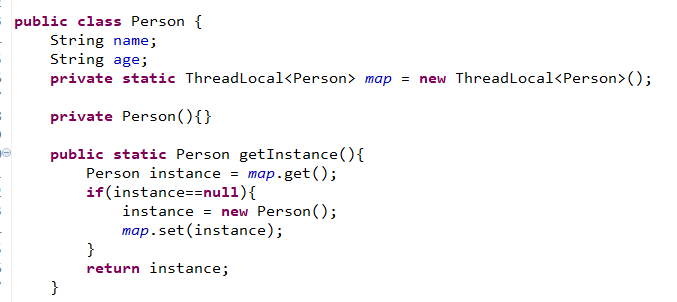


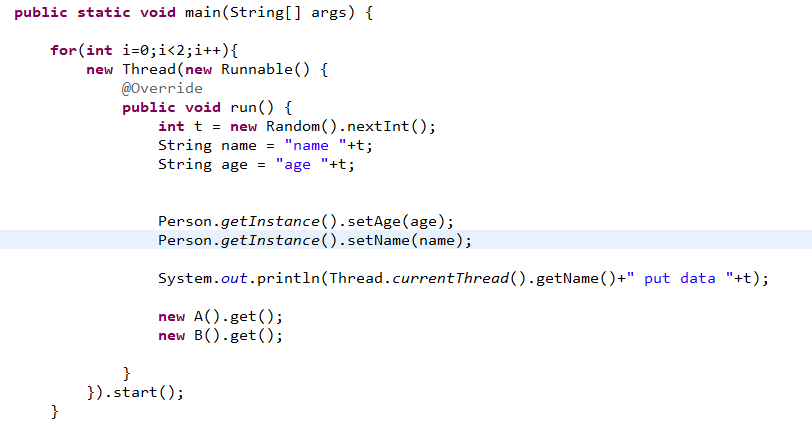


### 线程共享变量

多个数据打包后存放到ThreadLocal中

单例模式加载包装类





### JAVA5

aomic等类

### 线程池Executors

Executors提供一系列静态方法创建线程池

1：固定线程池



2：动态线程池 内部线程数可变



3：单例线程池 （线程死亡后可启动新的线程）



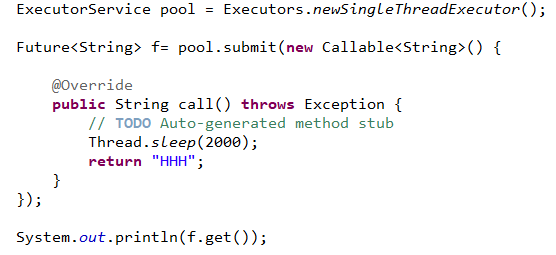
4：调度线程池



内部可执行调度任务

### Callable和Future

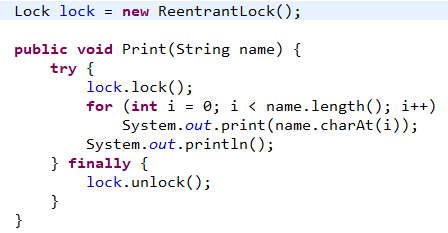
Callable返回泛型数据 Future接受泛型数据



### 锁

Lock(接口)

1：线程锁 ReentrantLock(实现类)



2：读写锁 (读上读锁 写上写锁)



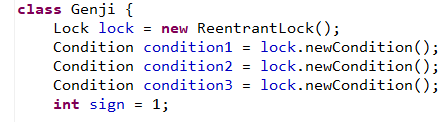
3：阻塞Condition

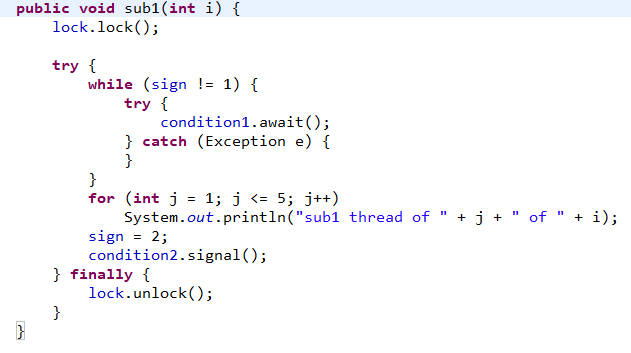


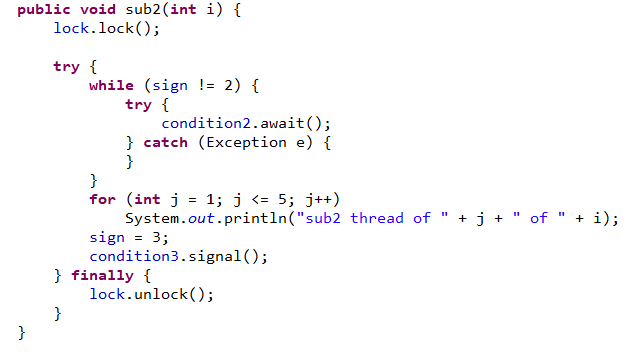
cocondition.signal();类似notify();

condition.await();作用类似wait();

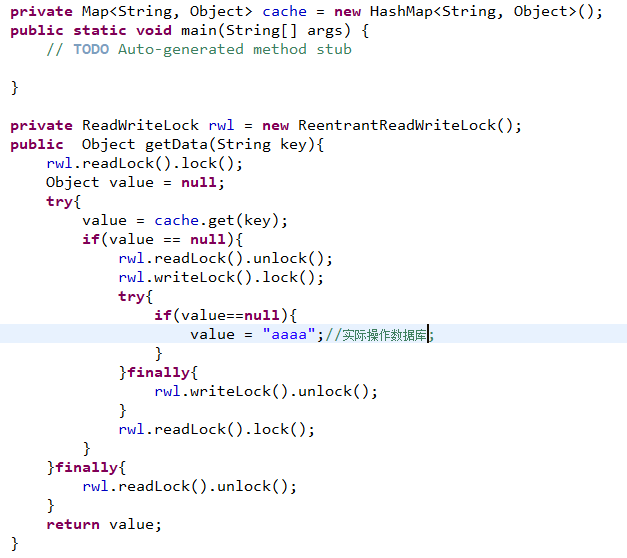
实现多线程轮流执行







面试题：设计一个缓存系统



# 锁

悲观锁：每次拿数据默认别人修改了数据 所以拿了数据会上锁 别的线程会阻塞

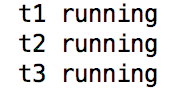
乐观锁：默认别人没有修改 所以不会上锁 更新时会检查是否有人修改 使用版本号机制可以实现

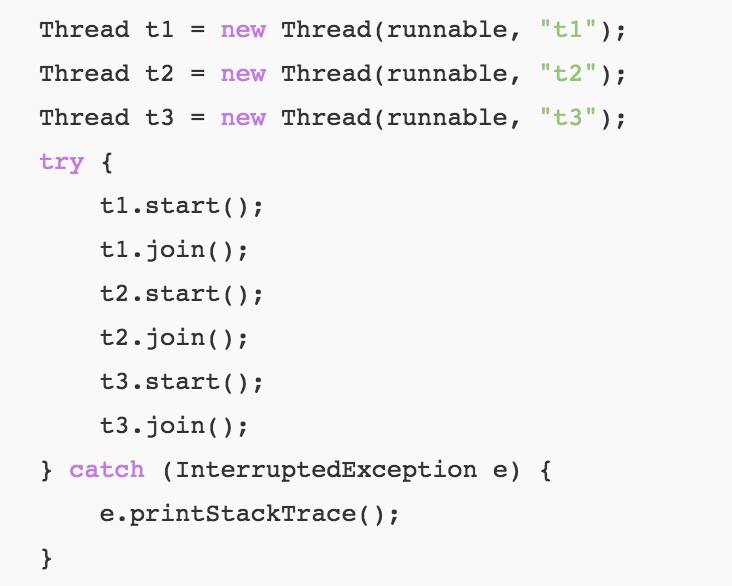


# 线程面试问题

## 三个线程如何特定顺序执行

join方法:在a线程中使用b.join会等b执行完毕再继续执行





## 主线程,子线程轮询问题

## 线程安全的计数器

使用AtomicInteger类,可以对long,boolean,int原子操作

实现原理,每次检查是否被其他线程修改过,是则返回并重新请求

