□ □
□ □
□
□ □

唠唠并发

不同的进程之间通过一些锁粒度的通信机制来交换数据，包括：套字、信号处理器、共享内存、信号量以及文件等。

线程间会共享进程范围内的资。每个线程都会有各自的程序计数器（PC）、栈以及局部变量

同一个程中的所有线程共享进程中的内存地址空间

线程也被称为轻量级的进程，它是基本的调度单位（进程是资源分配的最小单位）多线程可以通过提处理器的资源利用率来提高系统的吞吐量

多线程优势

线程的风险

安全性

"　　"
"　　"

"　"

```
test1   0
test1   1
test2   0
test2   1
test2   2
test2   3
test2   4
test1   2
test1   3
test1   4
```

" "
" "

" "

```
Thread-1操作后的值为：8
Thread-2操作后的值为：8
Thread-1操作后的值为：7
Thread-2操作后的值为：6
Thread-2操作后的值为：5
Thread-1操作后的值为：4
Thread-1操作后的值为：2
Thread-2操作后的值为：2
Thread-1操作后的值为：0
Thread-2操作后的值为：0
```

多线程是并发执行的，他们共享想通过的内存地址空间

活跃性

某个操作无法执行下去的时候就会发生活跃性的问题

性能问题

性能问题包括很多：服务时间过长、响应不灵敏、吞吐率过低、资源消耗过高、可伸缩性较低

线程的安全性

线程安全 要对状态访问操作进行关联，特别是对共享的和可变的状态的访
。

JAVA中主要的同步机制是synchronized关键字

线程安全性定义：当多个线程访问某个类时，这个类始终能够表现除正确的行为，那么就称这个类是程安全的。

无状态对象一定是线程安全的。

原子性

竞态条件

在程序执行的过程中，由于执行时许出现不正确的结果。当某个计算的正确性取决于多个线程的交替行时序时，就会发生竞态条件。最常见的竞态条件类型是：“先检查后执行”的操作。

复合操作

加锁机制

内置锁

在JAVA中提供一种内置锁的机制来支持原子性：同步代码块（同步代码块包括两部分：一个是作为的对象引用，另一个是作为有这个锁保护的代码块）JAVA中内置锁相当于一种互斥锁。

以synchronized来修饰的方法就是一种横跨整个方法体的代码同步块。

重入锁

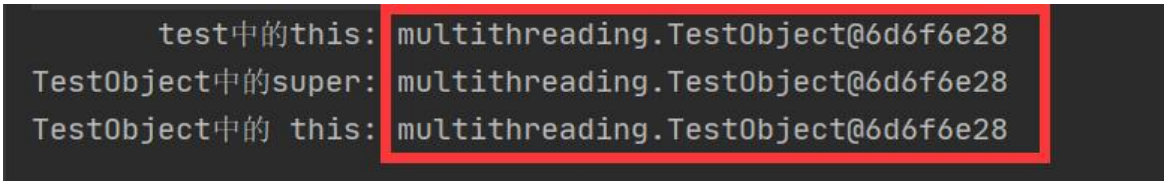
**内置锁是可重入的。“重入”意味着获取锁的操作粒度是“线程”而不是“调用
。重入的一种实现方法是为每个锁关联一个获取计数值和一个所有者线程。当某个线程请求一个未持的锁时，JVM将记下锁的持有者，并且将获得的锁计数值置为1.如果同一个线程再次获得这个锁，计器值递增，当线程退出同步代码块时，计数器值递减，计数器为0时，锁被释放。**

他们锁住的对象是一致的

"　　"

"　　"

"　　"



用锁来保护状态

象的内置锁与其状态之间没有内在的关联

活跃性与性能

由于service方法是一个synchronized方法

PS：当执行时间较长的计算或者可能无法快速完成的操作时（eg：网络I/O或控制台I/O）一定不要有锁。