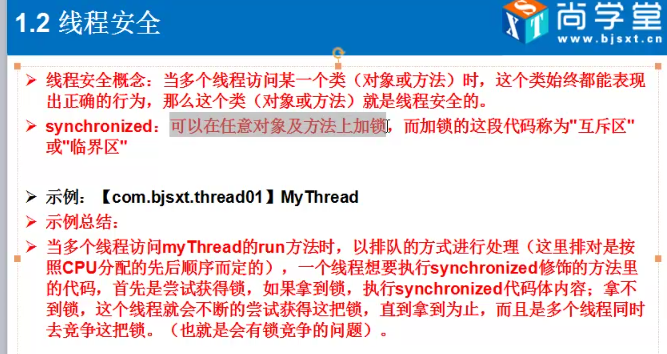
# (001-4)并发编程基础篇和中级篇

## 线程基础

线程安全：当多个线程访问某一个类（对象或者方法）时，这个类始终能表现出正确的行为，那么这个类就是线程安全的。

Synchronized 可以在任意对象及方法上加锁，而加锁的这段代码称为“互斥区”或者“临界区”。

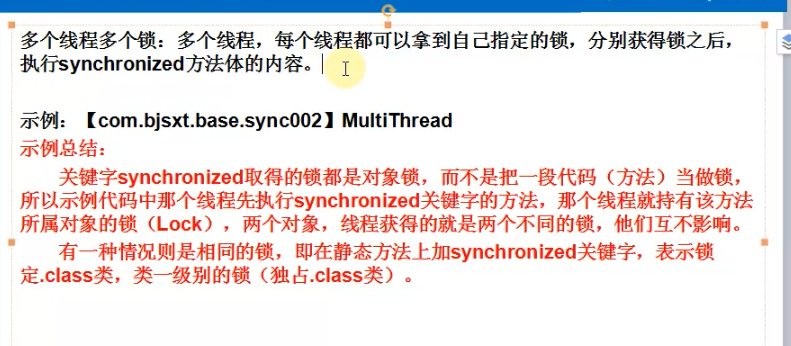


只有一个厕所，只能有一个人使用，只有前一个人使用完毕，才能让后面的人使用。

## 锁竞争的问题

多个线程抢一把锁的问题。

## 多个线程多个锁

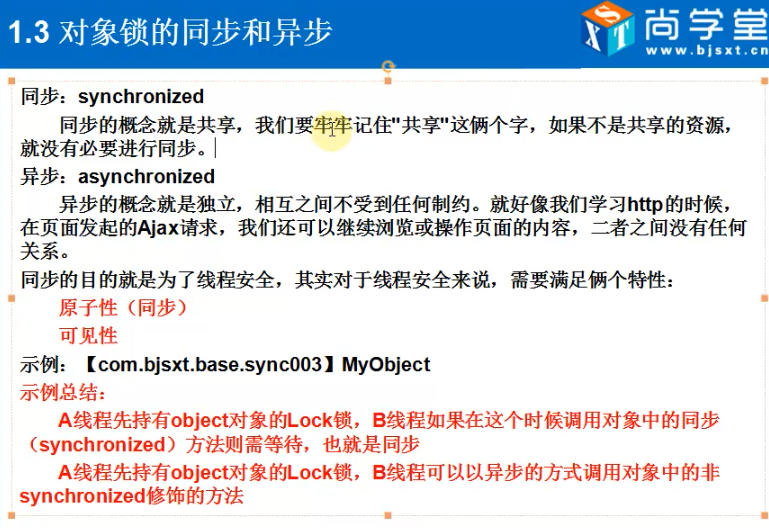


一个对象一把锁。

只加synchronized是对象锁，每个对象都有独自的一把锁。

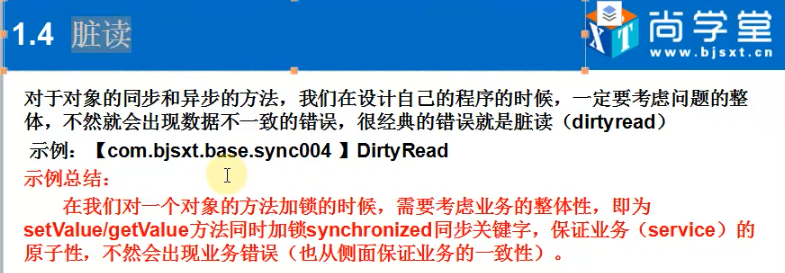
加上static synchronized 是类锁，一个类只有一把锁，所有对象共享该锁。

## 对象锁的同步和异步



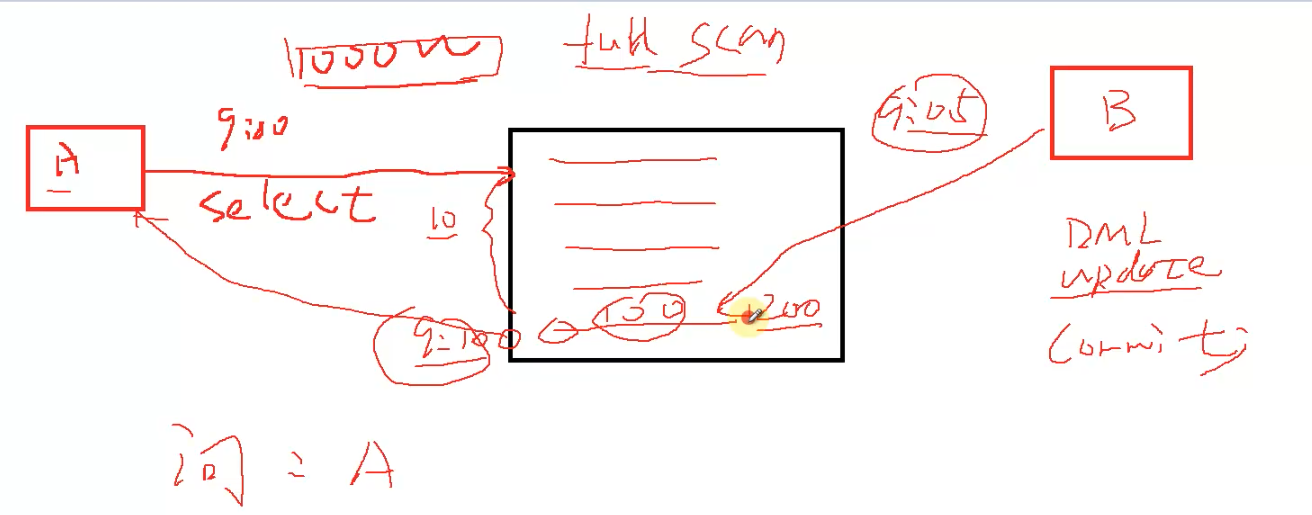
类似Ajax请求。

## 脏读



## ACID

原子性（Atomicity）、一致性（Consistency）、隔离性（Isolation）、持久性（Durability）



A 看到的是100 ，还是200？

因为Oracle 一致性读，看到是100。

Undo 概念： 像日志，客户端执行DML操作时，会执行undo.

当执行到该记录，发现该条记录有变化，他会到undo从查找，老记录。

Snapshot too old.

9点读，一定看到的是9点那一时刻的数据。

## MySQL InnoDB四个事务级别 与 脏读、不重复读、幻读

1. MySQL InnoDB事务隔离级别

脏读、可重复读、幻读

MySQL InnoDB事务的隔离级别有四级，默认是“可重复读”（REPEATABLE READ）。

· 1）.未提交读（READUNCOMMITTED）。另一个事务修改了数据，但尚未提交，而本事务中的SELECT会读到这些未被提交的数据（脏读）( 隔离级别最低，并发性能高 )。

· 2）.提交读（READCOMMITTED）。本事务读取到的是最新的数据（其他事务提交后的）。问题是，在同一个事务里，前后两次相同的SELECT会读到不同的结果（不重复读）。会出现不可重复读、幻读问题（锁定正在读取的行）

· 3）.可重复读（REPEATABLEREAD）。在同一个事务里，SELECT的结果是事务开始时时间点的状态，因此，同样的SELECT操作读到的结果会是一致的。但是，会有幻读现象（稍后解释）。会出幻读（锁定所读取的所有行）。

· 4）.串行化（SERIALIZABLE）。读操作会隐式获取共享锁，可以保证不同事务间的互斥（锁表）。

‘

四个级别逐渐增强，每个级别解决一个问题。

· 1）.脏读。另一个事务修改了数据，但尚未提交，而本事务中的SELECT会读到这些未被提交的数据。

· 2）.不重复读。解决了脏读后，会遇到，同一个事务执行过程中，另外一个事务提交了新数据，因此本事务先后两次读到的数据结果会不一致。

· 3）.幻读。解决了不重复读，保证了同一个事务里，查询的结果都是事务开始时的状态（一致性）。但是，如果另一个事务同时提交了新数据，本事务再更新时，就会“惊奇的”发现了这些新数据，貌似之前读到的数据是“鬼影”一样的幻觉。

具体地：

1). 脏读

首先区分脏页和脏数据

脏页是内存的缓冲池中已经修改的page，未及时flush到硬盘，但已经写到redo log中。读取和修改缓冲池的page很正常，可以提高效率，flush即可同步。脏数据是指事务对缓冲池中的行记录record进行了修改，但是还没提交！！！，如果这时读取缓冲池中未提交的行数据就叫脏读，违反了事务的隔离性。脏读就是指当一个事务正在访问数据，并且对数据进行了修改，而这种修改还没有提交到数据库中，这时，另外一个事务也访问这个数据，然后使用了这个数据。

2). 不可重复读

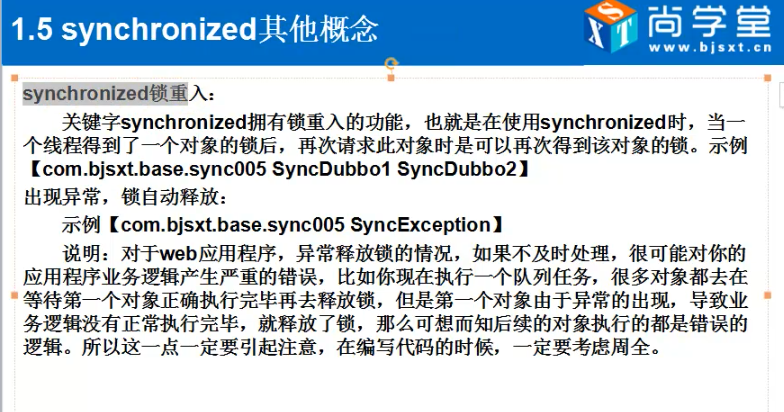
是指在一个事务内，多次读同一数据。在这个事务还没有结束时，另外一个事务也访问该同一数据。那么，在第一个事务中的两次读数据之间，由于第二个事务的修改，第二个事务已经提交。那么第一个事务两次读到的的数据可能是不一样的。这样就发生了在一个事务内两次读到的数据是不一样的，因此称为是不可重复读。例如，一个编辑人员两次读取同一文档，但在两次读取之间，作者重写了该文档。当编辑人员第二次读取文档时，文档已更改。原始读取不可重复。如果只有在作者全部完成编写后编辑人员才可以读取文档，则可以避免该问题

3). 幻读 :

是指当事务不是独立执行时发生的一种现象，例如第一个事务对一个表中的数据进行了修改，这种修改涉及到表中的全部数据行。同时，第二个事务也修改这个表中的数据，这种修改是向表中插入一行新数据。那么，以后就会发生操作第一个事务的用户发现表中还有没有修改的数据行，就好象发生了幻觉一样。例如，一个编辑人员更改作者提交的文档，但当生产部门将其更改内容合并到该文档的主复本时，发现作者已将未编辑的新材料添加到该文档中。如果在编辑人员和生产部门完成对原始文档的处理之前，任何人都不能将新材料添加到文档中，则可以避免该问题。

互联网金融、的并发量没有电商那么高。

## Synchronized锁重入



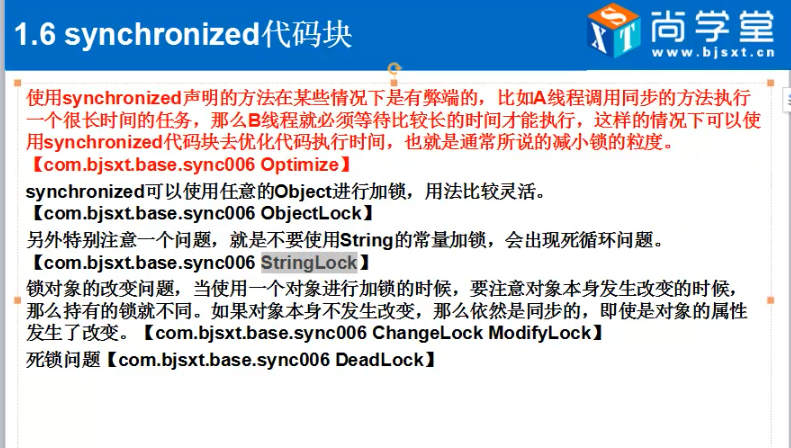
|  |
| --- |
| */\*\*  \* synchronized的重入  \** ***@author*** *alienware  \*  \*/* **public class** SyncDubbo2 {   **static class** Main {  **public int i** = 10;  **public synchronized void** operationSup(){  **try** {  **i**--;  System.***out***.println(**"Main print i = "** + **i**);  Thread.*sleep*(100);  } **catch** (InterruptedException e) {  e.printStackTrace();  }  }  }    **static class** Sub **extends** Main {  **public synchronized void** operationSub(){  **try** {  **while**(**i** > 0) {  **i**--;  System.***out***.println(**"Sub print i = "** + **i**);  Thread.*sleep*(100);   **this**.operationSup();  }  } **catch** (InterruptedException e) {  e.printStackTrace();  }  }  }    **public static void** main(String[] args) {    Thread t1 = **new** Thread(**new** Runnable() {  @Override  **public void** run() {  Sub sub = **new** Sub();  sub.operationSub();  }  });    t1.start();  }     } |

碰到异常，会释放锁。

Storm： 做分布式计算的；拆分成bolt

PLSQL For循环加上begin end. 才能加上Exception。

## Synchronized代码块



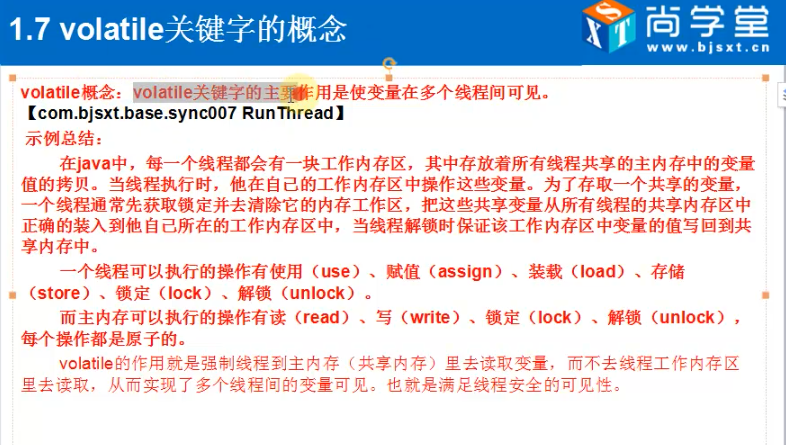
String 常量 在常量池中只有一个引用。

不要synchronized(“aa”); 在synchronized 不要修改String 字符串。

对象中属性发生变化，不影响锁的持有。

## volatile关键字

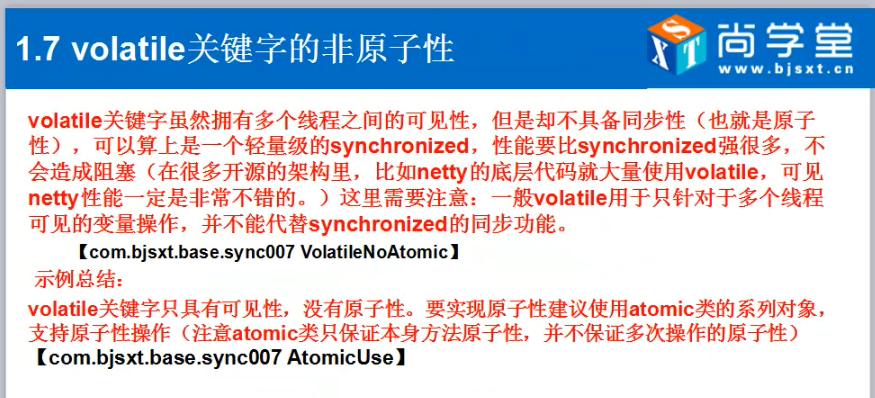
使变量在多个线程间可见。



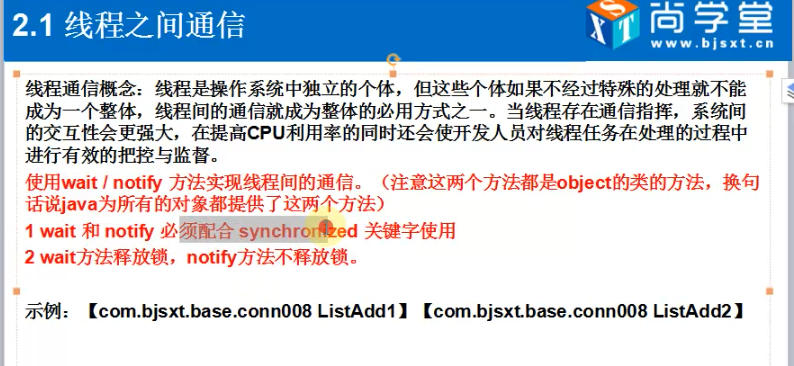


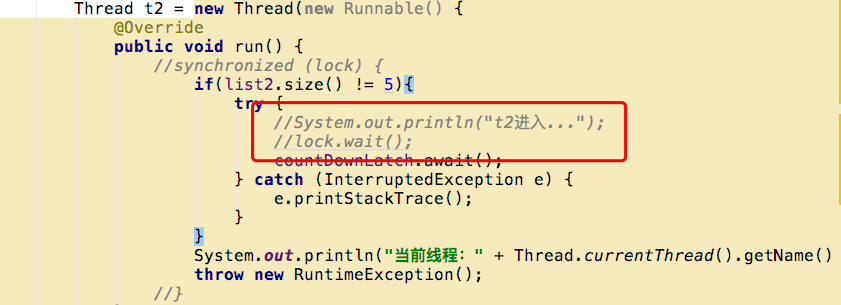
CAS 非阻塞算法

无锁并行计算。



## 线程通信

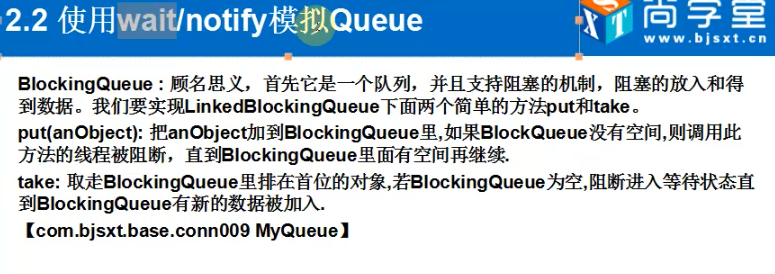




只是进入了，一直阻塞到这里，没有后面的执行了



## Queue模拟



# (005)并发编程高级篇-锁的高级深化