进程 vs 线程

进程:进程之间相对独立，包括很多线程。无法共享内存

进程;有独立的地址空间

线程：调用栈。函数的参数等，pc,程序指针 指向内存， TLS

线程没有独立的地址空间

存储：硬盘，内存，缓存，寄存器 速度：从慢到快 价格：从便宜到贵

操作系统：

寻址空间 32位 4G 只能访问4G 64位 10 ^19 Bytes 64位jvm 可以使更大内存，需要重新编译

寻址; int n = \*p; MOV EAX, [EBX]

网络传输： 7层 5层

不可靠：

滑动窗口：在TCP协议中使用 可以调整大小窗口 流量控制、解决拥塞

维持发送方/接收方缓冲区

关系型数据库：

基于关系代数理论

缺点：表结构不直观，实现复杂，速度慢

优点：健壮性高，社区庞大

非关系型数据库：

事务：

ACID四个特性：原子性 一致性 隔离性 持久性

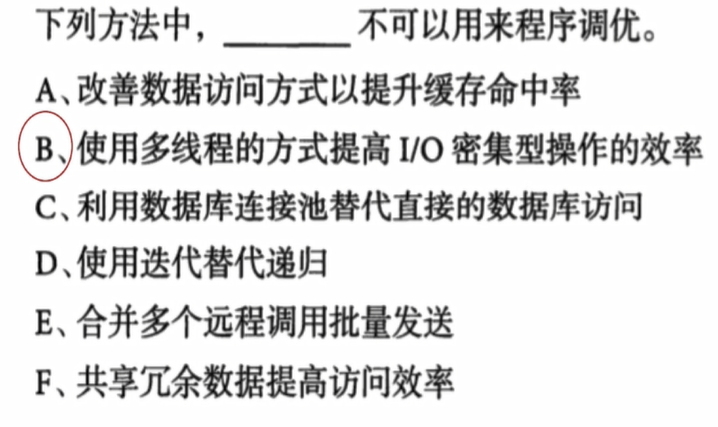
隔离级别:

Read uncommitted **脏读 事务A读到了事务B还没有提交的数据**

Read committed **不可重复读 在一个事务里面读取了两次某个数据，读出来的数据不一致**

Repeatable reads **幻读 在一个事务里面的操作中发现了未被操作的数据** 幻读出现的前提是并发的事务中有事务发生了插入、删除操作。

Serializable



程序设计语言的比较

类型检查：

编译时;c c++, java go

运行时;python js ruby perl

运行/编译：

编译成机器代码：c c++

编译成中间代码，在虚拟机运行：java c# 提高代码的移植性，跨平台

解释执行：python

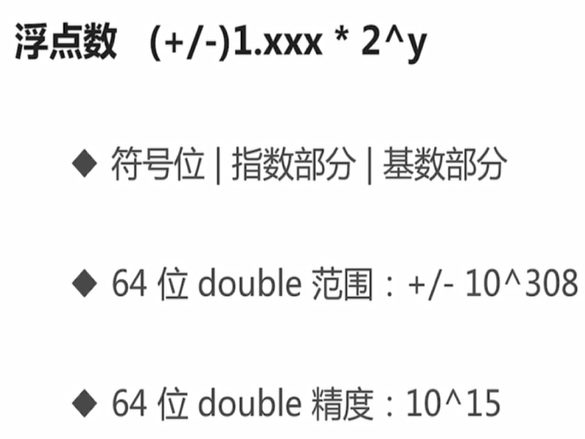
编程范式：面向过程 c vb

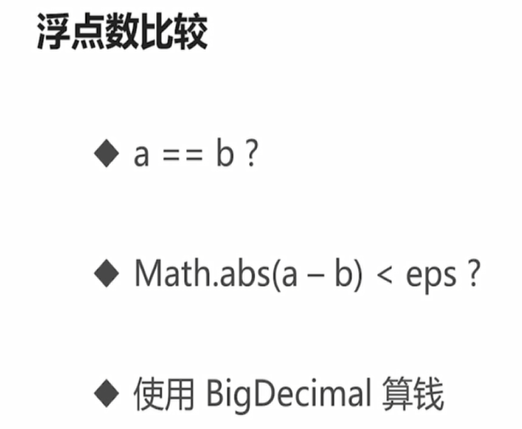
面向对象：java c++ c# Scala

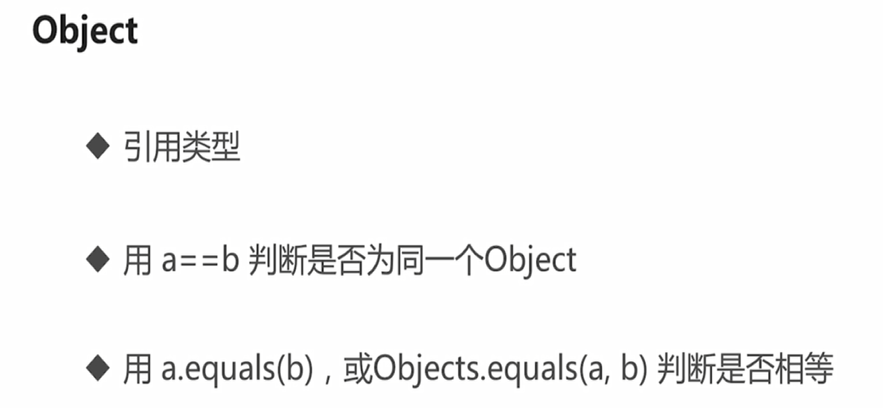
函数式：

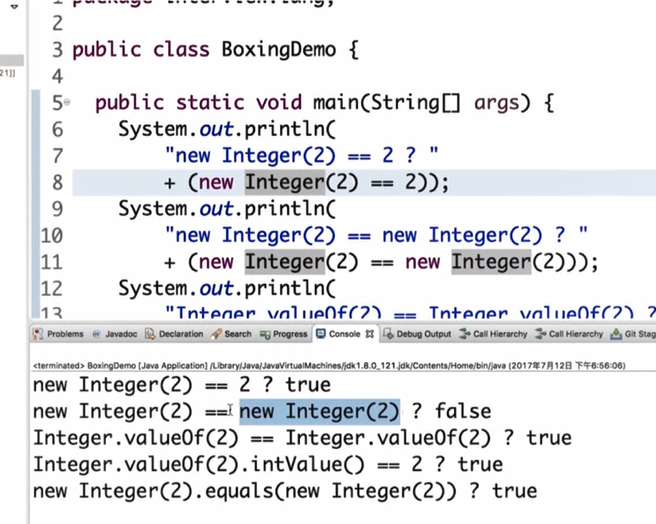
Int -2^31 -2^31 -1 共2^32

浮点数：









/\*\*  
\* **@Author** 宗坚  
\* **@Description** reverse linkedList  
\* such as :1 >2 >3 >4 >5 >null null < 1< 2 <3 <4 <5  
\* **@Since** 0:34 2019/2/26  
\* **@Param** [head]  
\* **@return** Coding.interviem.Node  
\*/  
**public** Node reverseLinkedList(Node head){  
 **if** (head == **null** || head.**next** ==**null**) {  
 **return** head;  
 }  
 Node newHead = reverseLinkedList(head.**next**);  
 head.**next**.**next** = head;  
 head.**next** = **null**;  
 **return** newHead;  
}



二分查找：

**public int** binarySearc(**int**[] arr, **int** k){  
 **int** a = 0;  
 **int** b = arr.**length**;  
 **while** (a < b) {  
 **int** m = a + (b - a) /2; *//小心a+b溢出* **if** (k <arr[m]) {  
 b = m;  
 } **else if** (k > arr[m]) {  
 a = m + 1;  
 }**else** {  
 **return** m;  
 }  
 }  
 **return** -1;  
}

前序遍历和中序遍历，， 求后续遍历

算法复杂度：

插入排序、选择排序 O(N^2)

归并排序、快速排序(最高（N^2）) O(N\*log(N) )

二分查找 O(log N)

算法的组合：区间合并

面向对象思想;

类与对象

类的成员变量 --- 对象状态

类的成员函数 ---对象的行为

类的静态变量 ---

类的静态函数 ---

Equals 相等 -🡪 hashcode 相等

接口

不能实现

接口与抽象类的比较：

~~实现角度~~：抽象类可以有实现，也可以不实现，但是需要使用abstract修饰

抽象类可以有成员变量

抽象类不可有多重继承，接口可以

接口不能有实现

封装：



一般使用private and public

派生类中修改封装可见性

增加可见性，不能减少

不可变性：不可变对象Immutable Objects 引用可以传递，可以缓存，线程安全

Final 修饰 类不可以被继承 函数不可以在派生类修改 变量不可以指向其他对象

Final 无法保证不可变性

泛型

List<E> 只允许放类型为E的数据

在运行的时候 List 和List<String> 没有区别



Singleton:确保全局最多只有一个对象

用于构造缓慢的对象，需要统一管理的资源

缺点;很多全局状态，线程安全性

创建：双重锁模式

继承关系：描述is-a关系

尽量使用组合代替继承

State模式、装饰者模式

使用new来创建的缺点

编译时必须决定创建哪个类的对象

参数意义不明确

并行计算的方法：

将数据拆分到每个节点

每个节点并行的计算出结果

将结果汇总

如何排序10G个元素？

归并排序

分多个节点排序，然后合并节点

利用小根堆PriorityQueue

多线程：

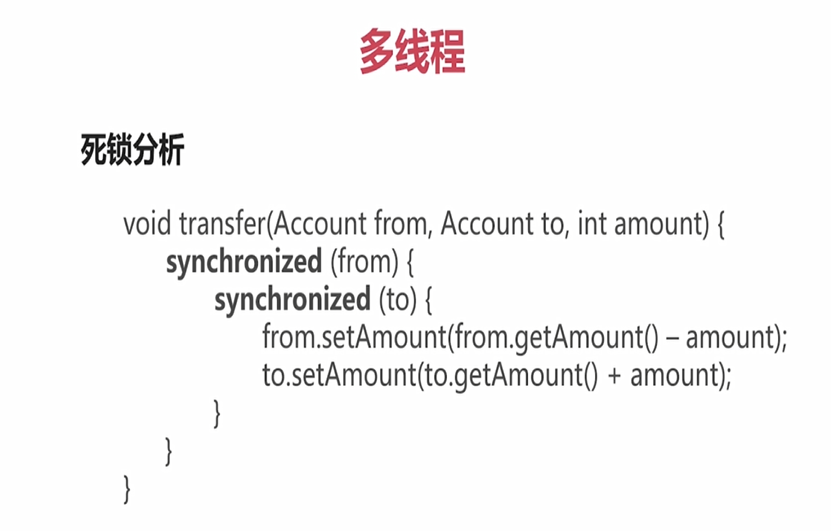
死锁分析

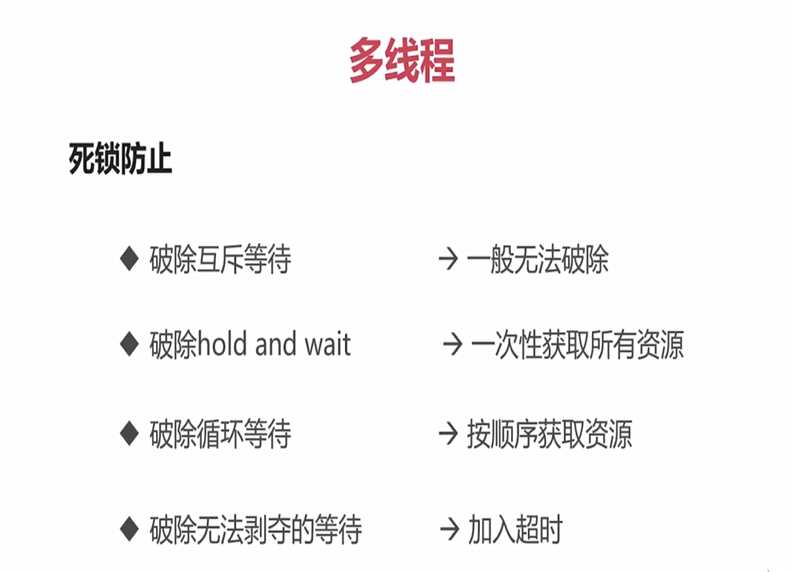
条件：互斥等待

Hold and wait

循环等待

无法剥夺的等待





线程池：

预先建立好线程，等待任务派发





资源管理

Java垃圾回收

不被引用的对象会被回收

Minor GC 、 Full GC 垃圾回收时所有运行暂停

结构化表达

提问：