作者：微信公众号JavaQ  
链接：<https://www.nowcoder.com/discuss/31667?type=0&order=0&pos=25&page=1>  
来源：牛客网

这个主题的内容之前分三个篇幅分享过，导致网络上传播的比较分散，所以本篇做了一个汇总，同时对部分内容及答案做了修改，欢迎朋友们吐槽、转发。因为篇幅长度和时间的原因，部分答案我没有亲自总结。更多精彩内容请关注我的微信公众号：JavaQ。

**1.面向对象和面向过程的区别**

**面向过程**  
优点：性能比面向对象高，因为类调用时需要实例化，开销比较大，比较消耗资源;比如单片机、嵌入式开发、Linux/Unix等一般采用面向过程开发，性能是最重要的因素。  
缺点：没有面向对象易维护、易复用、易扩展  
**面向对象**  
优点：易维护、易复用、易扩展，由于面向对象有封装、继承、多态性的特性，可以设计出低耦合的系统，使系统更加灵活、更加易于维护  
缺点：性能比面向过程低

**2.Java的四个基本特性（抽象、封装、继承，多态）**

抽象：就是把现实生活中的某一类东西提取出来，用程序代码表示，我们通常叫做类或者接口。抽象包括两个方面：一个是数据抽象，一个是过程抽象。数据抽象也就是对象的属性。过程抽象是对象的行为特征。  
封装：把客观事物封装成抽象的类，并且类可以把自己的数据和方法只让可信的类或者对象操作，对不可信的进行封装隐藏。封装分为属性的封装和方法的封装。  
继承：是对有着共同特性的多类事物，进行再抽象成一个类。这个类就是多类事物的父类。父类的意义在于抽取多类事物的共性。  
多态：允许不同类的对象对同一消息做出响应。方法的重载、类的覆盖正体现了多态。

**3.重载和重写的区别**

重载：发生在同一个类中，方法名必须相同，参数类型不同、个数不同、顺序不同，方法返回值和访问修饰符可以不同，发生在编译时。  
重写：发生在父子类中，方法名、参数列表必须相同，返回值小于等于父类，抛出的异常小于等于父类，访问修饰符大于等于父类；如果父类方法访问修饰符为private则子类中就不是重写。

**4.构造器Constructor是否可被override**

构造器不能被重写，不能用static修饰构造器，只能用public  
private protected这三个权限修饰符，且不能有返回语句。

**5.访问控制符public,protected,private,以及默认的区别**

private只有在本类中才能访问；  
public在任何地方都能访问；  
protected在同包内的类及包外的子类能访问；  
默认不写在同包内能访问。  
6是否可以继承String类#  
String类是final类故不可以继承，一切由final修饰过的都不能继承。

**7.String和StringBuffer、StringBuilder的区别**

**可变性**  
String类中使用字符数组保存字符串，private  
final char value[]，所以string对象是不可变的。StringBuilder与StringBuffer都继承自AbstractStringBuilder类，在AbstractStringBuilder中也是使用字符数组保存字符串，char[]  
value，这两种对象都是可变的。  
**线程安全性**  
String中的对象是不可变的，也就可以理解为常量，线程安全。AbstractStringBuilder是StringBuilder与StringBuffer的公共父类，定义了一些字符串的基本操作，如expandCapacity、append、insert、indexOf等公共方法。StringBuffer对方法加了同步锁或者对调用的方法加了同步锁，所以是线程安全的。StringBuilder并没有对方法进行加同步锁，所以是非线程安全的。  
**性能**  
每次对String 类型进行改变的时候，都会生成一个新的String 对象，然后将指针指向新的String 对象。StringBuffer每次都会对  
StringBuffer 对象本身进行操作，而不是生成新的对象并改变对象引用。相同情况下使用  
StirngBuilder 相比使用  
StringBuffer 仅能获得10%~15% 左右的性能提升，但却要冒多线程不安全的风险。

**8.hashCode和equals方法的关系**

equals相等，hashcode必相等；hashcode相等，equals可能不相等。

**9.抽象类和接口的区别**

**语法层次**  
抽象类和接口分别给出了不同的语法定义。  
**设计层次**  
抽象层次不同，抽象类是对类抽象，而接口是对行为的抽象。抽象类是对整个类整体进行抽象，包括属性、行为，但是接口却是对类局部（行为）进行抽象。抽象类是自底向上抽象而来的，接口是自顶向下设计出来的。  
**跨域不同**  
抽象类所体现的是一种继承关系，要想使得继承关系合理，父类和派生类之间必须存在"is-a"  
关系，即父类和派生类在概念本质上应该是相同的。对于接口则不然，并不要求接口的实现者和接口定义在概念本质上是一致的，仅仅是实现了接口定义的契约而已，"like-a"的关系。

**10.自动装箱与拆箱**

装箱：将基本类型用它们对应的引用类型包装起来；  
拆箱：将包装类型转换为基本数据类型；  
Java使用自动装箱和拆箱机制，节省了常用数值的内存开销和创建对象的开销，提高了效率，由编译器来完成，编译器会在编译期根据语法决定是否进行装箱和拆箱动作。

**11.什么是泛型、为什么要使用以及泛型擦除**

泛型，即“参数化类型”。  
创建集合时就指定集合元素的类型，该集合只能保存其指定类型的元素，避免使用强制类型转换。  
Java编译器生成的字节码是不包涵泛型信息的，泛型类型信息将在编译处理是被擦除，这个过程即类型擦除。泛型擦除可以简单的理解为将泛型java代码转换为普通java代码，只不过编译器更直接点，将泛型java代码直接转换成普通java字节码。  
类型擦除的主要过程如下：  
1）.将所有的泛型参数用其最左边界（最顶级的父类型）类型替换。  
2）.移除所有的类型参数。

**12.Java中的集合类及关系图**

List和Set继承自Collection接口。  
Set无序不允许元素重复。HashSet和TreeSet是两个主要的实现类。  
List有序且允许元素重复。ArrayList、LinkedList和Vector是三个主要的实现类。  
Map也属于集合系统，但和Collection接口没关系。Map是key对value的映射集合，其中key列就是一个集合。key不能重复，但是value可以重复。HashMap、TreeMap和Hashtable是三个主要的实现类。  
SortedSet和SortedMap接口对元素按指定规则排序，SortedMap是对key列进行排序。

**13.HashMap实现原理**

具体原理参考文章：  
<http://zhangshixi.iteye.com/blog/672697>  
<http://www.admin10000.com/document/3322.html>

**14.HashTable实现原理**

具体原理参考文章：  
<http://www.cnblogs.com/skywang12345/p/3310887.html>  
<http://blog.csdn.net/chdjj/article/details/38581035>

**15.HashMap和HashTable区别**

1）.HashTable的方法前面都有synchronized来同步，是线程安全的；HashMap未经同步，是非线程安全的。  
2）.HashTable不允许null值(key和value都不可以) ；HashMap允许null值(key和value都可以)。  
3）.HashTable有一个contains(Object  
value)功能和containsValue(Object  
value)功能一样。  
4）.HashTable使用Enumeration进行遍历；HashMap使用Iterator进行遍历。  
5）.HashTable中hash数组默认大小是11，增加的方式是old\*2+1；HashMap中hash数组的默认大小是16，而且一定是2的指数。  
6）.哈希值的使用不同，HashTable直接使用对象的hashCode； HashMap重新计算hash值，而且用与代替求模。

**16.ArrayList和vector区别**

ArrayList和Vector都实现了List接口，都是通过数组实现的。  
Vector是线程安全的，而ArrayList是非线程安全的。  
List第一次创建的时候，会有一个初始大小，随着不断向List中增加元素，当List 认为容量不够的时候就会进行扩容。Vector缺省情况下自动增长原来一倍的数组长度，ArrayList增长原来的50%。

**17.ArrayList和LinkedList区别及使用场景**

**区别**  
ArrayList底层是用数组实现的，可以认为ArrayList是一个可改变大小的数组。随着越来越多的元素被添加到ArrayList中，其规模是动态增加的。  
LinkedList底层是通过双向链表实现的， LinkedList和ArrayList相比，增删的速度较快。但是查询和修改值的速度较慢。同时，LinkedList还实现了Queue接口，所以他还提供了offer(),  
peek(), poll()等方法。  
**使用场景**  
LinkedList更适合从中间插入或者删除（链表的特性）。  
ArrayList更适合检索和在末尾插入或删除（数组的特性）。

**18.Collection和Collections的区别**

java.util.Collection 是一个集合接口。它提供了对集合对象进行基本操作的通用接口方法。Collection接口在Java 类库中有很多具体的实现。Collection接口的意义是为各种具体的集合提供了最大化的统一操作方式。  
java.util.Collections 是一个包装类。它包含有各种有关集合操作的静态多态方法。此类不能实例化，就像一个工具类，服务于Java的Collection框架。

**19.Concurrenthashmap实现原理**

具体原理参考文章：  
<http://www.cnblogs.com/ITtangtang/p/3948786.html>  
<http://ifeve.com/concurrenthashmap/>

**20.Error、Exception区别**

Error类和Exception类的父类都是throwable类，他们的区别是：  
Error类一般是指与虚拟机相关的问题，如系统崩溃，虚拟机错误，内存空间不足，方法调用栈溢等。对于这类错误的导致的应用程序中断，仅靠程序本身无法恢复和和预防，遇到这样的错误，建议让程序终止。  
Exception类表示程序可以处理的异常，可以捕获且可能恢复。遇到这类异常，应该尽可能处理异常，使程序恢复运行，而不应该随意终止异常。

**21.Unchecked**

Exception和Checked Exception，各列举几个#  
**Unchecked Exception:**  
a. 指的是程序的瑕疵或逻辑错误，并且在运行时无法恢复。  
b. 包括Error与RuntimeException及其子类，如：OutOfMemoryError,  
UndeclaredThrowableException, IllegalArgumentException,  
IllegalMonitorStateException, NullPointerException, IllegalStateException,  
IndexOutOfBoundsException等。  
c. 语法上不需要声明抛出异常。

**Checked Exception:**  
a. 代表程序不能直接控制的无效外界情况（如用户输入，数据库问题，网络异常，文件丢失等）  
b. 除了Error和RuntimeException及其子类之外，如：ClassNotFoundException,  
NamingException, ServletException, SQLException, IOException等。  
c. 需要try catch处理或throws声明抛出异常。

**22.Java中如何实现代理机制(JDK、CGLIB)**

JDK动态代理：代理类和目标类实现了共同的接口，用到InvocationHandler接口。  
CGLIB动态代理：代理类是目标类的子类，用到MethodInterceptor接口。

**23.多线程的实现方式**

继承Thread类、实现Runnable接口、使用ExecutorService、Callable、Future实现有返回结果的多线程。

**24.线程的状态转换**

20140828202610671.jpg

**25.如何停止一个线程**

参考文章：  
<http://www.cnblogs.com/greta/p/5624839.html>

**26.什么是线程安全**

线程安全就是多线程访问同一代码，不会产生不确定的结果。

**27.如何保证线程安全**

对非安全的代码进行加锁控制；  
使用线程安全的类；  
多线程并发情况下，线程共享的变量改为方法级的局部变量。

**28.synchronized如何使用**

synchronized是Java中的关键字，是一种同步锁。它修饰的对象有以下几种：  
1). 修饰一个代码块，被修饰的代码块称为同步语句块，其作用的范围是大括号{}括起来的代码，作用的对象是调用这个代码块的对象；  
2). 修饰一个方法，被修饰的方法称为同步方法，其作用的范围是整个方法，作用的对象是调用这个方法的对象；  
3). 修改一个静态的方法，其作用的范围是整个静态方法，作用的对象是这个类的所有对象；  
4). 修改一个类，其作用的范围是synchronized后面括号括起来的部分，作用主的对象是这个类的所有对象。

**29.synchronized和Lock的区别**

主要相同点：Lock能完成synchronized所实现的所有功能  
主要不同点：Lock有比synchronized更精确的线程语义和更好的性能。Lock的锁定是通过代码实现的，而synchronized是在JVM层面上实现的，synchronized会自动释放锁，而Lock一定要求程序员手工释放，并且必须在finally从句中释放。Lock还有更强大的功能，例如，它的tryLock方法可以非阻塞方式去拿锁。Lock锁的范围有局限性，块范围，而synchronized可以锁住块、对象、类。

**30.多线程如何进行信息交互**

void notify() 唤醒在此对象监视器上等待的单个线程。  
void notifyAll() 唤醒在此对象监视器上等待的所有线程。  
void wait() 导致当前的线程等待，直到其他线程调用此对象的notify()方法或notifyAll()方法。  
void wait(long timeout) 导致当前的线程等待，直到其他线程调用此对象的notify()方法或notifyAll()方法，或者超过指定的时间量。  
void wait(long timeout, int nanos) 导致当前的线程等待，直到其他线程调用此对象的notify()方法或notifyAll()方法，或者其他某个线程中断当前线程，或者已超过某个实际时间量。

**31.sleep和wait的区别(考察的方向是是否会释放锁)**

sleep()方法是Thread类中方法，而wait()方法是Object类中的方法。  
sleep()方法导致了程序暂停执行指定的时间，让出cpu该其他线程，但是他的监控状态依然保持者，当指定的时间到了又会自动恢复运行状态，在调用sleep()方法的过程中，线程不会释放对象锁。而当调用wait()方法的时候，线程会放弃对象锁，进入等待此对象的等待锁定池，只有针对此对象调用notify()方法后本线程才进入对象锁定池准备。

**32.多线程与死锁**

死锁是指两个或两个以上的进程在执行过程中，因争夺资源而造成的一种互相等待的现象，若无外力作用，它们都将无法推进下去。  
产生死锁的原因：  
一.因为系统资源不足。  
二.进程运行推进的顺序不合适。  
三.资源分配不当。

**33.如何才能产生死锁**

产生死锁的四个必要条件：  
一.互斥条件：所谓互斥就是进程在某一时间内独占资源。  
二.请求与保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放。  
三.不剥夺条件:进程已获得资源，在末使用完之前，不能强行剥夺。  
四.循环等待条件:若干进程之间形成一种头尾相接的循环等待资源关系。

**34.死锁的预防**

打破产生死锁的四个必要条件中的一个或几个，保证系统不会进入死锁状态。  
一.打破互斥条件。即允许进程同时访问某些资源。但是，有的资源是不允许被同时访问的，像打印机等等，这是由资源本身的属性所决定的。所以，这种办法并无实用价值。  
二.打破不可抢占条件。即允许进程强行从占有者那里夺取某些资源。就是说，当一个进程已占有了某些资源，它又申请新的资源，但不能立即被满足时，它必须释放所占有的全部资源，以后再重新申请。它所释放的资源可以分配给其它进程。这就相当于该进程占有的资源被隐蔽地强占了。这种预防死锁的方法实现起来困难，会降低系统性能。  
三.打破占有且申请条件。可以实行资源预先分配策略。即进程在运行前一次性地向系统申请它所需要的全部资源。如果某个进程所需的全部资源得不到满足，则不分配任何资源，此进程暂不运行。只有当系统能够满足当前进程的全部资源需求时，才一次性地将所申请的资源全部分配给该进程。由于运行的进程已占有了它所需的全部资源，所以不会发生占有资源又申请资源的现象，因此不会发生死锁。  
四.打破循环等待条件，实行资源有序分配策略。采用这种策略，即把资源事先分类编号，按号分配，使进程在申请，占用资源时不会形成环路。所有进程对资源的请求必须严格按资源序号递增的顺序提出。进程占用了小号资源，才能申请大号资源，就不会产生环路，从而预防了死锁。

**35.什么叫守护线程，用什么方法实现守护线程**

守护线程是为其他线程的运行提供服务的线程。  
setDaemon(boolean on)方法可以方便的设置线程的Daemon模式，true为守护模式，false为用户模式。

**36.Java线程池技术及原理**

参考文章：  
<http://www.importnew.com/19011.html>  
<http://www.cnblogs.com/dolphin0520/p/3932921.html>

**37.java并发包concurrent及常用的类**

这个内容有点多，参考文章：  
并发包诸类概览：[http://www.raychase.net/1912](http://www.raychase.net/1912" \t "_blank)  
线程池：[http://www.cnblogs.com/dolphin0520/p/3932921.html](http://www.cnblogs.com/dolphin0520/p/3932921.html" \t "_blank)  
锁：[http://www.cnblogs.com/dolphin0520/p/3923167.html](http://www.cnblogs.com/dolphin0520/p/3923167.html" \t "_blank)  
集合：[http://www.cnblogs.com/huangfox/archive/2012/08/16/2642666.html](http://www.cnblogs.com/huangfox/archive/2012/08/16/2642666.html" \t "_blank)

**38.volatile关键字**

用volatile修饰的变量，线程在每次使用变量的时候，都会读取变量修改后的最的值。volatile很容易被误用，用来进行原子性操作。  
Java语言中的volatile变量可以被看作是一种“程度较轻的  
synchronized”；与  
synchronized 块相比，volatile 变量所需的编码较少，并且运行时开销也较少，但是它所能实现的功能也仅是synchronized的一部分。锁提供了两种主要特性：互斥（mutual  
exclusion）和可见性（visibility）。互斥即一次只允许一个线程持有某个特定的锁，因此可使用该特性实现对共享数据的协调访问协议，这样，一次就只有一个线程能够使用该共享数据。可见性必须确保释放锁之前对共享数据做出的更改对于随后获得该锁的另一个线程是可见的，如果没有同步机制提供的这种可见性保证，线程看到的共享变量可能是修改前的值或不一致的值，这将引发许多严重问题。Volatile变量具有synchronized的可见性特性，但是不具备原子特性。这就是说线程能够自动发现volatile  
变量的最新值。  
要使volatile变量提供理想的线程安全，必须同时满足下面两个条件：对变量的写操作不依赖于当前值；该变量没有包含在具有其他变量的不变式中。  
第一个条件的限制使volatile变量不能用作线程安全计数器。虽然增量操作（x++）看上去类似一个单独操作，实际上它是一个由读取－修改－写入操作序列组成的组合操作，必须以原子方式执行，而volatile不能提供必须的原子特性。实现正确的操作需要使x 的值在操作期间保持不变，而volatile  
变量无法实现这点。  
每一个线程运行时都有一个线程栈，线程栈保存了线程运行时候变量值信息。当线程访问某一个对象时候值的时候，首先通过对象的引用找到对应在堆内存的变量的值，然后把堆内存变量的具体值load到线程本地内存中，建立一个变量副本，之后线程就不再和对象在堆内存变量值有任何关系，而是直接修改副本变量的值，在修改完之后的某一个时刻（线程退出之前），自动把线程变量副本的值回写到对象在堆中变量。这样在堆中的对象的值就产生变化了。

read and load 从主存复制变量到当前工作内存  
use and assign 执行代码，改变共享变量值  
store and write 用工作内存数据刷新主存相关内容  
其中use and  
assign 可以多次出现，但是这一些操作并不是原子性，也就是在read load之后，如果主内存count变量发生修改之后，线程工作内存中的值由于已经加载，不会产生对应的变化，所以计算出来的结果会和预期不一样。

**39.Java中的NIO，BIO，AIO分别是什么**

BIO:同步并阻塞，服务器实现模式为一个连接一个线程，即客户端有连接请求时服务器端就需要启动一个线程进行处理，如果这个连接不做任何事情会造成不必要的线程开销，当然可以通过线程池机制改善。BIO方式适用于连接数目比较小且固定的架构，这种方式对服务器资源要求比较高，并发局限于应用中，JDK1.4以前的唯一选择，但程序直观简单易理解。  
NIO:同步非阻塞，服务器实现模式为一个请求一个线程，即客户端发送的连接请求都会注册到多路复用器上，多路复用器轮询到连接有I/O请求时才启动一个线程进行处理。NIO方式适用于连接数目多且连接比较短（轻操作）的架构，比如聊天服务器，并发局限于应用中，编程比较复杂，JDK1.4开始支持。  
AIO:异步非阻塞，服务器实现模式为一个有效请求一个线程，客户端的I/O请求都是由OS先完成了再通知服务器应用去启动线程进行处理.AIO方式使用于连接数目多且连接比较长（重操作）的架构，比如相册服务器，充分调用OS参与并发操作，编程比较复杂，JDK7开始支持。

**40.IO和NIO区别**

一.IO是面向流的，NIO是面向缓冲区的。  
二.IO的各种流是阻塞的，NIO是非阻塞模式。  
三.Java NIO的选择器允许一个单独的线程来监视多个输入通道，你可以注册多个通道使用一个选择器，然后使用一个单独的线程来“选择”通道：这些通道里已经有可以处理的输入，或者选择已准备写入的通道。这种选择机制，使得一个单独的线程很容易来管理多个通道。

**41.序列化与反序列化**

把对象转换为字节序列的过程称为对象的序列化。  
把字节序列恢复为对象的过程称为对象的反序列化。  
对象的序列化主要有两种用途：  
一.把对象的字节序列永久地保存到硬盘上，通常存放在一个文件中；  
二.在网络上传送对象的字节序列。  
当两个进程在进行远程通信时，彼此可以发送各种类型的数据。无论是何种类型的数据，都会以二进制序列的形式在网络上传送。发送方需要把这个Java对象转换为字节序列，才能在网络上传送；接收方则需要把字节序列再恢复为Java对象。

**42.常见的序列化协议有哪些**

Protobuf, Thrift, Hessian, Kryo

**43.内存溢出和内存泄漏的区别**

内存溢出是指程序在申请内存时，没有足够的内存空间供其使用，出现out of  
memory。  
内存泄漏是指分配出去的内存不再使用，但是无法回收。

**44.Java内存模型及各个区域的OOM，如何重现OOM**

这部分内容很重要，详细阅读《深入理解Java虚拟机》，也可以详细阅读这篇文章[http://hllvm.group.iteye.com/group/wiki/2857-JVM](http://hllvm.group.iteye.com/group/wiki/2857-JVM" \t "_blank)

**45.出现OOM如何解决**

一. 可通过命令定期抓取heap dump或者启动参数OOM时自动抓取heap dump文件。  
二. 通过对比多个heap dump，以及heap dump的内容，分析代码找出内存占用最多的地方。  
三. 分析占用的内存对象，是否是因为错误导致的内存未及时释放，或者数据过多导致的内存溢出。

**46.用什么工具可以查出内存泄漏**

一. Memory  
Analyzer－是一款开源的JAVA内存分析软件，查找内存泄漏，能容易找到大块内存并验证谁在一直占用它，它是基于Eclipse  
RCP(Rich Client Platform)，可以下载RCP的独立版本或者Eclipse的插件。  
二. JProbe－分析Java的内存泄漏。  
三.JProfiler－一个全功能的Java剖析工具，专用于分析J2SE和J2EE应用程序。它把CPU、执行绪和内存的剖析组合在一个强大的应用中，GUI可以找到效能瓶颈、抓出内存泄漏、并解决执行绪的问题。  
四. JRockit－用来诊断Java内存泄漏并指出根本原因，专门针对Intel平台并得到优化，能在Intel硬件上获得最高的性能。  
五. YourKit-.NET & Java Profiling业界领先的Java和.NET程序性能分析工具。  
六.AutomatedQA －AutomatedQA的获奖产品performance profiling和memory debugging工具集的下一代替换产品，支持Microsoft,Borland, Intel, Compaq 和 GNU编译器。可以为.NET和Windows程序生成全面细致的报告，从而帮助您轻松隔离并排除代码中含有的性能问题和内存/资源泄露问题。支持.Net 1.0,1.1,2.0,3.0和Windows 32/64位应用程序。  
七.Compuware DevPartner Java Edition－包含Java内存检测,代码覆盖率测试,代码性能测试,线程死锁,分布式应用等几大功能模块

**47.Java内存管理及回收算法**

阅读这篇文章：<http://www.cnblogs.com/hnrainll/archive/2013/11/06/3410042.html>

**48.Java类加载器及如何加载类(双亲委派)**

阅读文章：  
[https://www.ibm.com/developerworks/cn/java/j-lo-classloader/（推荐）](https://www.ibm.com/developerworks/cn/java/j-lo-classloader/%EF%BC%88%E6%8E%A8%E8%8D%90%EF%BC%89)  
<http://blog.csdn.net/zhoudaxia/article/details/35824249>

**49.xml解析方式**

一.DOM(JAXP  
Crimson解析器)  
二.SAX  
三.JDOM  
四.DOM4J  
**区别：**  
一.DOM4J性能最好，连Sun的JAXM也在用DOM4J。目前许多开源项目中大量采用DOM4J，例如大名鼎鼎的hibernate也用DOM4J来读取XML配置文件。如果不考虑可移植性，那就采用DOM4J.  
二.JDOM和DOM在性能测试时表现不佳，在测试10M  
文档时内存溢出。在小文档情况下还值得考虑使用DOM和JDOM。虽然JDOM的开发者已经说明他们期望在正式发行版前专注性能问题，但是从性能观点来看，它确实没有值得推荐之处。另外，DOM仍是一个非常好的选择。DOM实现广泛应用于多种编程语言。它还是许多其它与XML相关的标准的基础，因为它正式获得W3C  
推荐(与基于非标准的Java模型相对)，所以在某些类型的项目中可能也需要它(如在JavaScript中使用DOM)。  
三.SAX表现较好，这要依赖于它特定的解析方式－事件驱动。一个SAX检测即将到来的XML流，但并没有载入到内存(当然当XML流被读入时，会有部分文档暂时隐藏在内存中)。

**50.Statement和PreparedStatement之间的区别**

一.PreparedStatement是预编译的,对于批量处理可以大大提高效率. 也叫JDBC存储过程  
二.使用  
Statement 对象。在对数据库只执行一次性存取的时侯，用  
Statement 对象进行处理。PreparedStatement  
对象的开销比Statement大，对于一次性操作并不会带来额外的好处。  
三.statement每次执行sql语句，相关数据库都要执行sql语句的编译，preparedstatement是预编译得,  
preparedstatement支持批处理  
四.  
**代码片段1:**  
String updateString = "UPDATE COFFEES SET SALES = 75 " + "WHERE  
COF\_NAME LIKE ′Colombian′";  
stmt.executeUpdate(updateString);  
**代码片段2:**  
PreparedStatement updateSales = con.prepareStatement("UPDATE COFFEES SET  
SALES = ? WHERE COF\_NAME LIKE ? ");  
updateSales.setInt(1, 75);  
updateSales.setString(2, "Colombian");  
updateSales.executeUpdate();  
片断2和片断1的区别在于，后者使用了PreparedStatement对象，而前者是普通的Statement对象。PreparedStatement对象不仅包含了SQL语句，而且大多数情况下这个语句已经被预编译过，因而当其执行时，只需DBMS运行SQL语句，而不必先编译。当你需要执行Statement对象多次的时候，PreparedStatement对象将会大大降低运行时间，当然也加快了访问数据库的速度。  
这种转换也给你带来很大的便利，不必重复SQL语句的句法，而只需更改其中变量的值，便可重新执行SQL语句。选择PreparedStatement对象与否，在于相同句法的SQL语句是否执行了多次，而且两次之间的差别仅仅是变量的不同。如果仅仅执行了一次的话，它应该和普通的对象毫无差异，体现不出它预编译的优越性。  
五.执行许多SQL语句的JDBC程序产生大量的Statement和PreparedStatement对象。通常认为PreparedStatement对象比Statement对象更有效,特别是如果带有不同参数的同一SQL语句被多次执行的时候。PreparedStatement对象允许数据库预编译SQL语句，这样在随后的运行中可以节省时间并增加代码的可读性。  
然而，在Oracle环境中，开发人员实际上有更大的灵活性。当使用Statement或PreparedStatement对象时，Oracle数据库会缓存SQL语句以便以后使用。在一些情况下,由于驱动器自身需要额外的处理和在Java应用程序和Oracle服务器间增加的网络活动，执行PreparedStatement对象实际上会花更长的时间。  
然而，除了缓冲的问题之外，至少还有一个更好的原因使我们在企业应用程序中更喜欢使用PreparedStatement对象,那就是安全性。传递给PreparedStatement对象的参数可以被强制进行类型转换，使开发人员可以确保在插入或查询数据时与底层的数据库格式匹配。  
当处理公共Web站点上的用户传来的数据的时候，安全性的问题就变得极为重要。传递给PreparedStatement的字符串参数会自动被驱动器忽略。最简单的情况下，这就意味着当你的程序试着将字符串“D'Angelo”插入到VARCHAR2中时，该语句将不会识别第一个“，”，从而导致悲惨的失败。几乎很少有必要创建你自己的字符串忽略代码。  
在Web环境中，有恶意的用户会利用那些设计不完善的、不能正确处理字符串的应用程序。特别是在公共Web站点上,在没有首先通过PreparedStatement对象处理的情况下，所有的用户输入都不应该传递给SQL语句。此外，在用户有机会修改SQL语句的地方，如HTML的隐藏区域或一个查询字符串上，SQL语句都不应该被显示出来。

**51.servlet生命周期及各个方法**

参考文章<http://www.cnblogs.com/xuekyo/archive/2013/02/24/2924072.html>

**52.servlet中如何自定义filter**

参考文章[http://www.cnblogs.com/javawebsoa/archive/2013/07/31/3228858.html](http://www.cnblogs.com/javawebsoa/archive/2013/07/31/3228858.html" \t "_blank)

**53.JSP原理**

参考文章<http://blog.csdn.net/hanxuemin12345/article/details/23831645>

**54.JSP和Servlet的区别**

(1)JSP经编译后就变成了“类servlet”。  
(2)JSP由HTML代码和JSP标签构成，更擅长页面显示；Servlet更擅长流程控制。  
(3)JSP中嵌入JAVA代码，而Servlet中嵌入HTML代码。

**55.JSP的动态include和静态include**

(1)动态include用jsp:include动作实现，如<jsp:include page="abc.jsp" flush="true" />，它总是会检查所含文件中的变化，适合用于包含动态页面，并且可以带参数。会先解析所要包含的页面，解析后和主页面合并一起显示，即先编译后包含。  
(2)静态include用include伪码实现，不会检查所含文件的变化，适用于包含静态页面，如<%@  
include file="qq.htm" %>，不会提前解析所要包含的页面，先把要显示的页面包含进来，然后统一编译，即先包含后编译。

**56.Struts中请求处理过程**

参考文章<http://www.cnblogs.com/liuling/p/2013-8-10-01.html>

**57.MVC概念**

参考文章<http://www.cnblogs.com/scwyh/articles/1436802.html>

**58.Springmvc与Struts区别**

参考文章：  
<http://blog.csdn.net/tch918/article/details/38305395>  
<http://blog.csdn.net/chenleixing/article/details/44570681>

**59.Hibernate/Ibatis两者的区别**

参考文章<http://blog.csdn.net/firejuly/article/details/8190229>

**60.Hibernate一级和二级缓存**

参考文章<http://blog.csdn.net/windrui/article/details/23165845>

**61.简述Hibernate常见优化策略**

参考文章<http://blog.csdn.net/shimiso/article/details/8819114>

**62.Springbean的加载过程(推荐看Spring的源码)**

参考文章[http://geeekr.com/read-spring-source-1-how-to-load-bean/](http://geeekr.com/read-spring-source-1-how-to-load-bean/" \t "_blank)

**63.Springbean的实例化(推荐看Spring的源码)**

参考文章[http://geeekr.com/read-spring-source-two-beans-initialization/](http://geeekr.com/read-spring-source-two-beans-initialization/" \t "_blank)

**64.Spring如何实现AOP和IOC(推荐看Spring的源码)**

参考文章[http://www.360doc.com/content/15/0116/21/12385684\_441408260.shtml](http://www.360doc.com/content/15/0116/21/12385684_441408260.shtml" \t "_blank)

**65.Springbean注入方式**

参考文章<http://blessht.iteye.com/blog/1162131>

**66.Spring的事务管理**

这个主题的参考文章没找到特别好的，[http://blog.csdn.net/trigl/article/details/50968079这个还可以。](http://blog.csdn.net/trigl/article/details/50968079%E8%BF%99%E4%B8%AA%E8%BF%98%E5%8F%AF%E4%BB%A5%E3%80%82)

**67.Spring事务的传播特性**

参考文章<http://blog.csdn.net/lfsf802/article/details/9417095>

**68.springmvc原理**

参考文章<http://blog.sina.com.cn/s/blog_7ef0a3fb0101po57.html>

**69.springmvc用过哪些注解**

参考文章<http://aijuans.iteye.com/blog/2160141>

**70.Restful有几种请求**

参考文章，[http://www.infoq.com/cn/articles/designing-restful-http-apps-roth，该篇写的比较全。](http://www.infoq.com/cn/articles/designing-restful-http-apps-roth%EF%BC%8C%E8%AF%A5%E7%AF%87%E5%86%99%E7%9A%84%E6%AF%94%E8%BE%83%E5%85%A8%E3%80%82)

**71.Restful好处**

(1)客户-服务器：客户-服务器约束背后的原则是分离关注点。通过分离用户接口和数据存储这两个关注点，改善了用户接口跨多个平台的可移植性；同时通过简化服务器组件，改善了系统的可伸缩性。  
(2)无状态：通信在本质上是无状态的，改善了可见性、可靠性、可伸缩性.  
(3)缓存：改善了网络效率减少一系列交互的平均延迟时间，来提高效率、可伸缩性和用户可觉察的性能。  
(4)统一接口：REST架构风格区别于其他基于网络的架构风格的核心特征是，它强调组件之间要有一个统一的接口。

**72.Tomcat，Apache，JBoss的区别**

Apache:HTTP服务器(WEB服务器)，类似IIS，可以用于建立虚拟站点，编译处理静态页面，可以支持SSL技术，支持多个虚拟主机等功能。  
Tomcat:Servlet容器，用于解析jsp，Servlet的Servlet容器，是高效，轻量级的容器。缺点是不支持EJB，只能用于java应用。  
Jboss:应用服务器，运行EJB的J2EE应用服务器，遵循J2EE规范，能够提供更多平台的支持和更多集成功能，如数据库连接，JCA等，其对Servlet的支持是通过集成其他Servlet容器来实现的，如tomcat和jetty。

**73.memcached和redis的区别**

(1)性能对比：由于Redis只使用单核，而Memcached可以使用多核，所以平均每一个核上Redis在存储小数据时比Memcached性能更高。而在100k以上的数据中，Memcached性能要高于Redis，虽然Redis最近也在存储大数据的性能上进行优化，但是比起Memcached，还是稍有逊色。  
(2)内存使用效率对比：使用简单的key-value存储的话，Memcached的内存利用率更高，而如果Redis采用hash结构来做key-value存储，由于其组合式的压缩，其内存利用率会高于Memcached。  
(3)Redis支持服务器端的数据操作：Redis相比Memcached来说，拥有更多的数据结构和并支持更丰富的数据操作，通常在Memcached里，你需要将数据拿到客户端来进行类似的修改再set回去。这大大增加了网络IO的次数和数据体积。在Redis中，这些复杂的操作通常和一般的GET/SET一样高效。所以，如果需要缓存能够支持更复杂的结构和操作，那么Redis会是不错的选择。

**74.如何理解分布式锁**

参考文章：  
<http://blog.csdn.net/zheng0518/article/details/51607063>  
[http://blog.csdn.net/nicewuranran/article/details/51730131。](http://blog.csdn.net/nicewuranran/article/details/51730131%E3%80%82)

**75.你知道的开源协议有哪些**

常见的开源协议有GPL、LGPL、BSD、Apache Licence  
vesion 2.0、MIT，详细内容参考文章:  
[http://blog.jobbole.com/44175/、http://www.ruanyifeng.com/blog/2011/05/how\_to\_choose\_free\_software\_licenses.html。](http://blog.jobbole.com/44175/%E3%80%81http:/www.ruanyifeng.com/blog/2011/05/how_to_choose_free_software_licenses.html%E3%80%82)

**76.json和xml区别**

XML:  
(1)应用广泛，可扩展性强，被广泛应用各种场合；  
(2)读取、解析没有JSON快；  
(3)可读性强，可描述复杂结构。  
JSON:  
(1)结构简单，都是键值对；  
(2)读取、解析速度快，很多语言支持；  
(3)传输数据量小，传输速率大大提高；  
(4)描述复杂结构能力较弱。

**77.设计模式**

参考文章：[http://www.cnblogs.com/beijiguangyong/archive/2010/11/15/2302807.html#\_Toc281750445。](http://www.cnblogs.com/beijiguangyong/archive/2010/11/15/2302807.html#_Toc281750445%E3%80%82)

**78.设计模式的六大原则**

参考文章[http://www.uml.org.cn/sjms/201211023.asp。](http://www.uml.org.cn/sjms/201211023.asp%E3%80%82)

**79.用一个设计模式写一段代码或画出一个设计模式的UML**

参考文章<http://www.cnblogs.com/beijiguangyong/archive/2010/11/15/2302807.html#_Toc281750445>

**80.高内聚，低耦合方面的理解**

参考文章[http://my.oschina.net/heweipo/blog/423235。](http://my.oschina.net/heweipo/blog/423235%E3%80%82)

**81.深度优先和广度优先算法**

推荐看书籍复习！可参考文章：  
<http://blog.163.com/zhoumhan_0351/blog/static/3995422720098342257387/>  
<http://blog.163.com/zhoumhan_0351/blog/static/3995422720098711040303/>  
<http://blog.csdn.net/andyelvis/article/details/1728378>  
<http://driftcloudy.iteye.com/blog/782873>

**82.排序算法及对应的时间复杂度和空间复杂度**

推荐看书籍复习！可参考文章：  
<http://www.cnblogs.com/liuling/p/2013-7-24-01.html>  
<http://blog.csdn.net/cyuyanenen/article/details/51514443>  
<http://blog.csdn.net/whuslei/article/details/6442755>

**83.排序算法编码实现**

参考<http://www.cnblogs.com/liuling/p/2013-7-24-01.html>

**84.查找算法**

参考<http://sanwen8.cn/p/142Wbu5.html>

**85.B+树**

参考<http://www.cnblogs.com/syxchina/archive/2011/03/02/2197251.html>

**86.KMP算法**

推荐阅读数据复习！参考<http://www.cnblogs.com/c-cloud/p/3224788.html>

**87.hash算法及常用的hash算法**

参考<http://www.360doc.com/content/13/0409/14/10384031_277138819.shtml>

**88.如何判断一个单链表是否有环**

参考文章：  
<http://www.jianshu.com/p/0e28d31600dd>  
[http://my.oschina.net/u/2391658/blog/693277?p={{totalPage}}](http://my.oschina.net/u/2391658/blog/693277?p=%7B%7BtotalPage%7D%7D)

**89.队列、栈、链表、树、堆、图**

推荐阅读数据复习！

**90.linux常用命令**

参考<http://www.jianshu.com/p/03cfc1a721b8>

**91.如何查看内存使用情况**

参考<http://blog.csdn.net/windrui/article/details/40046413>

**92.Linux下如何进行进程调度**

推荐阅读书籍复习，参考文章：  
<http://www.cnblogs.com/zhaoyl/archive/2012/09/04/2671156.html>  
<http://blog.csdn.net/rainharder/article/details/7975387>

**93.产生死锁的必要条件**

参考<http://blog.sina.com.cn/s/blog_5e3604840100ddgq.html>

**94.死锁预防**

参考<http://blog.sina.com.cn/s/blog_5e3604840100ddgq.html>

**95.数据库范式**

参考<http://www.360doc.com/content/12/0712/20/5287961_223855037.shtml>

**96.数据库事务隔离级别**

参考<http://blog.csdn.net/fg2006/article/details/6937413>

**97.数据库连接池的原理**

参考<http://blog.csdn.net/shuaihj/article/details/14223015>

**98.乐观锁和悲观锁**

参考<http://www.open-open.com/lib/view/open1452046967245.html>

**99.如何实现不同数据库的数据查询分页**

参考<http://blog.csdn.net/yztezhl/article/details/20489387>

**100.SQL注入的原理，如何预防**

参考<https://www.aliyun.com/zixun/content/3_15_245099.html>

**101.数据库索引的实现(B+树介绍、和B树、R树区别)**

参考文章：  
<http://blog.csdn.net/kennyrose/article/details/7532032>  
<http://www.xuebuyuan.com/2216918.html>

**102.SQL性能优化**

参考文章：  
<http://database.51cto.com/art/200904/118526.htm>  
<http://www.cnblogs.com/rootq/archive/2008/11/17/1334727.html>

**103.数据库索引的优缺点以及什么时候数据库索引失效**

参考文章：  
<http://www.cnblogs.com/mxmbk/articles/5226344.html>  
<http://www.cnblogs.com/simplefrog/archive/2012/07/15/2592527.html>  
<http://www.open-open.com/lib/view/open1418476492792.html>  
<http://blog.csdn.net/colin_liu2009/article/details/7301089>  
<http://www.cnblogs.com/hongfei/archive/2012/10/20/2732589.html>

**104.Redis的数据类型**

参考<http://blog.csdn.net/hechurui/article/details/49508735>

**105.OSI七层模型以及TCP/IP四层模型**

参考文章：  
<http://blog.csdn.net/sprintfwater/article/details/8751453>  
<http://www.cnblogs.com/commanderzhu/p/4821555.html>  
<http://blog.csdn.net/superjunjin/article/details/7841099>

**106.HTTP和HTTPS区别**

参考：  
<http://blog.csdn.net/mingli198611/article/details/8055261>  
<http://www.mahaixiang.cn/internet/1233.html>

**107.HTTP报文内容**

参考文章：  
<https://yq.aliyun.com/articles/44675>  
<http://www.cnblogs.com/klguang/p/4618526.html>  
<http://my.oschina.net/orgsky/blog/387759>

**108.get提交和post提交的区别**

参考文章：  
<http://www.cnblogs.com/hyddd/archive/2009/03/31/1426026.html>  
<http://www.jellythink.com/archives/806>

**109.get提交是否有字节限制，如果有是在哪限制的**

参考<http://www.jellythink.com/archives/806>

**110.TCP的三次握手和四次挥手**

阅读<http://www.jianshu.com/p/f7d1010fa603>

**111.session和cookie的区别**

参考<http://www.cnblogs.com/shiyangxt/archive/2008/10/07/1305506.html>

**112.HTTP请求中Session实现原理**

参考<http://blog.csdn.net/zhq426/article/details/2992488>

**113.redirect与forward区别**

参考<http://www.cnblogs.com/wxgblogs/p/5602849.html>

**114.TCP和UDP区别**

参考<http://www.cnblogs.com/bizhu/archive/2012/05/12/2497493.html>

**115.DDos攻击及预防**

参考文章：  
<http://blog.csdn.net/huwei2003/article/details/45476743>  
<http://www.leiphone.com/news/201509/9zGlIDvLhwguqOtg.html>