**1.Collections.sort排序内部原理**

在Java 6中Arrays.sort()和Collections.sort()使用的是MergeSort，而在Java 7中，内部实现换成了TimSort，其对对象间比较的实现要求更加严格

**2.hashMap原理，java8做的改变**

从结构实现来讲，HashMap是数组+链表+红黑树（JDK1.8增加了红黑树部分）实现的。HashMap最多只允许一条记录的键为null，允许多条记录的值为null。HashMap非线程安全。ConcurrentHashMap线程安全。解决碰撞：当出现冲突时，运用拉链法，将关键词为同义词的结点链接在一个单链表中，散列表长m，则定义一个由m个头指针组成的指针数组T，地址为i的结点插入以T(i)为头指针的单链表中。Java8中，冲突的元素超过限制（8），用红黑树替换链表。

**3.String 和 StringBuilder 的区别**

1）可变与不可变：String不可变，每一次执行“+”都会新生成一个新对象，所以频繁改变字符串的情况中不用String，以节省内存。

2）是否多线程安全：StringBuilder并没有对方法进行加同步锁，所以是非线程安全的。StringBuffer和String均线程安全。

**4.Vector 与 Array 的区别**

1）ArrayList在内存不够时默认是扩展50% + 1个，Vector是默认扩展1倍。

2）Vector属于线程安全级别的，但是大多数情况下不使用Vector，因为线程安全需要更大的系统开销。

**5.HashMap 与 Hashtable 的区别**

1） 历史原因: Hashtable继承Dictonary类, HashMap继承自abstractMap

2） HashMap允许空的键值对, 但最多只有一个空对象，而HashTable不允许。

3） HashTable同步，而HashMap非同步，效率上比HashTable要高

**6.ConncurrentHashMap和hashtable比较（两个线程并发访问map中同一条链，一个线程在尾部删除，一个线程在前面遍历查找，问为什么前面的线程还能正确的查找到后面被另一个线程删除的节点）**

ConcurrentHashMap融合了hashtable和hashmap二者的优势。hashtable是做了同步的，即线程安全，hashmap未考虑同步。所以hashmap在单线程情况下效率较高。hashtable在的多线程情况下，同步操作能保证程序执行的正确性。但是hashtable是阻塞的，每次同步执行的时候都要锁住整个结构，ConcurrentHashMap正是为了解决这个问题而诞生的，

ConcurrentHashMap允许多个修改操作并发进行，其关键在于使用了锁分离技术（一个Array保存多个Object，使用这些对象的锁作为分离锁，get/put时随机使用任意一个）。它使用了多个锁来控制对hash表的不同部分进行的修改。在JDK 1.6中，有HashEntry结构存在，每次插入将新添加节点作为链的头节点（同HashMap实现），而且每次删除一个节点时，会将删除节点之前的所有节点拷贝一份组成一个新的链，而将当前节点的上一个节点的next指向当前节点的下一个节点，从而在删除以后有两条链存 在，因而可以保证即使在同一条链中，有一个线程在删除，而另一个线程在遍历，它们都能工作良好，因为遍历的线程能继续使用原有的链。

Java8中，采用volatile HashEntry保存数据，table元素作为锁；从table数组+单向链表加上了红黑树。红黑树是一种特别的二叉查找树，特性为：1.节点为红或者黑 2.根节点为黑 3.叶节点为黑 4.一节点为红，则叶节点为黑 5.一节点到其子孙节点所有路径上的黑节点数目相同。

**7.ArrayList与 LinkedList 的区别？**

最明显的区别是

ArrrayList 底层的数据结构是数组，支持随机访问，而 LinkedList 的底层数据结构书链表，不支持随机访问。使用下标访问一个元素，ArrayList 的时间复杂度是 O(1)，而 LinkedList 是 O(n)。LinkedList是双向链表

**8.Java 中，Comparator 与Comparable 有什么不同？**

Comparable 接口用于定义对象的自然顺序，是排序接口，而 comparator 通常用于定义用户定制的顺序，是比较接口。我们如果需要控制某个类的次序，而该类本身不支持排序(即没有实现Comparable接口)，那么我们就可以建立一个“该类的比较器”来进行排序。Comparable 总是只有一个，但是可以有多个 comparator 来定义对象的顺序。

**9.抽象类是什么？它与接口有什么区别？你为什么要使用过抽象类？**

抽象类是指不允许被实例化的类；一个类只能使用一次继承关系。但是，一个类却可以实现多个interface。

abstract class和interface所反映出的设计理念不同。其实abstract class表示的是"is-a"关系，interface表示的是"like-a"关系

实现抽象类和接口的类必须实现其中的所有方法。抽象类中可以有非抽象方法。接口中则不能有实现方法。但在Java8中允许接口中有静态默认的方法。

接口中定义的变量默认是public static final 型，且必须给其初值，所以实现类中不能重新定义，也不能改变其值。抽象类中的变量默认是 friendly 型，其值可以在子类中重新定义，也可以重新赋值。

子类中实现父类中的抽象方法时，可见性可以大于等于父类中的；而接口实现类中的接口 方法的可见性只能与接口中相同（public）。

用抽象类是为了重用。减少编码量，降低耦合性。

**10.描述 Java 中的重载和重写？**

重载和重写都允许你用相同的名称来实现不同的功能，但是重载是编译时活动，而重写是运行时活动。你可以在同一个类中重载方法，但是只能在子类中重写方法。重写必须要有继承

重写：1、在子类中可以根据需要对从基类中继承来的方法进行重写。2、重写的方法和被重写的方法必须具有相同方法名称、参数列表和返回类型。3、重写方法不能使用比被重写的方法更严格的访问权限。

重载的时候，方法名要一样，但是参数类型和个数不一样，返回值类型可以相同也可以不相同。无法以返回型别作为重载函数的区分标准。

**11.Collection与Collections的区别是什么？**

Collection是Java集合框架中的基本接口；

Collections是Java集合框架提供的一个工具类，其中包含了大量用于操作或返回集合的静态方法。

**12.Java中多态的实现原理**

所谓多态，指的就是父类引用指向子类对象，调用方法时会调用子类的实现而不是父类的实现。多态的实现的关键在于“动态绑定”。

**13.object中定义了哪些方法？**

clone(), equals(), hashCode(), toString(), notify(), notifyAll(),

wait(), finalize(), getClass()

**14.Java泛型和类型擦除？**

泛型即参数化类型，在创建集合时，指定集合元素的类型，此集合只能传入该类型的参数。类型擦除：java编译器生成的字节码不包含泛型信息，所以在编译时擦除：1.泛型用最顶级父类替换；2.移除。

**15.说出 5 个 JDK 1.8 引入的新特性？**

Java 8 在 Java 历史上是一个开创新的版本，下面 JDK 8 中 5 个主要的特性：

Lambda 表达式；允许像对象一样传递匿名函数 Stream API，充分利用现代多核 CPU，可以写出很简洁的代码 ；Date 与 Time API，最终，有一个稳定、简单的日期和时间库可供你使用 扩展方法，现在，接口中可以有静态、默认方法； 重复注解，现在你可以将相同的注解在同一类型上使用多次。

**16.java中public，private，protected以及默认关键字的访问范围：**

Protected可在包内及包外子类访问，default只能同一包内访问，prvate只能同一类

**17. 常用数据结构：**

集合，线性结构（数组，队列，链表和栈），树形结构，图状结构

**18.Java 中的 TreeMap 是采用什么树实现的？(答案)**

Java 中的 TreeMap 是使用红黑树实现的。

**19. 匿名内部类是什么？如何访问在其外面定义的变量？**

匿名内部类也就是没有名字的内部类，匿名内部类只能使用一次，它通常用来简化代码编写。

匿名内部类只能访问外部类的Final变量. Java 8更加智能：如果局部变量被匿名内部类访问，那么该局部变量相当于自动使用了final修饰。

**20. 如何创建单例模式？说了双重检查，他说不是线程安全的。如何高效的创建一个线程安全的单例？**

一种是通过枚举，一种是通过静态内部类。

**21.poll() 方法和 remove() 方法的区别？**

poll() 和

remove() 都是从队列中取出一个元素，但是 poll() 在获取元素失败的时候会返回空，但是 remove() 失败的时候会抛出异常。

**22.写一段代码在遍历 ArrayList 时移除一个元素**

使用迭代器。

Iterator itr = list.iterator();

while(itr.hasNext()) {if(…) { itr.remove();} }

**JVM**

**1.JVM如何加载一个类的过程，双亲委派模型中有哪些方法**

类加载过程：加载、验证（验证阶段作用是保证Class文件的字节流包含的信息符合JVM规范，不会给JVM造成危害）、准备（准备阶段为变量分配内存并设置类变量的初始化）、解析（解析过程是将常量池内的符号引用替换成直接引用）、初始化。

双亲委派模型中方法：双亲委派是指如果一个类收到了类加载的请求，不会自己先尝试加载，先找父类加载器去完成。当顶层启动类加载器表示无法加载这个类的时候，子类才会尝试自己去加载。当回到最开的发起者加载器还无法加载时，并不会向下找，而是抛出ClassNotFound异常。

方法：启动（Bootstrap）类加载器，标准扩展（Extension）类加载器，应用程序类加载器（Application ），上下文(Custom)类加载器。意义是防止内存中出现多份同样的字节码 。

**2.GC算法（什么样的对象算是可回收对象，可达性分析），CMS收集器**

jvm是如何判断一个对象已经变成了可回收的“垃圾”，一般是两个方法：引用记数法和根搜索算法。引用记数法没办法解决循环引用的问题，所以用根搜索。从一系列的”GC Roots“对象开始向下搜索，搜索走过的路径称为引用链。当一个对象到”GC Roots“之间没有引用链时，被称为引用不可达。引用不可到的对象被认为是可回收的对象。

几种垃圾收集器：1，Serial New/Serial Old(串行)，2，Parrallel New (并行)，3，Parrallel Scavenge，4，Parrallel Old，5，CMS（CMS收集器是一个以获得最短回收停顿时间为目标的收集器，它是一种并发收集器，采用的是Mark-sweep算法。），6，G1（是一款并行与并发收集器，并且可建立可预测的停顿时间模型，整体上是基于标记清理，局部采用复制）

**3.JVM分为哪些区，每一个区干吗的？**

1）方法区(method)：被所有的线程共享。方法区包含所有的类信息和静态变量。

2）堆(heap)：被所有的线程共享，存放对象实例以及数组，Java堆是GC的主要区域。

3）栈(stack)：每个线程包含一个栈区，栈中保存一些局部变量等。

4）程序计数器：是当前线程执行的字节码的行指示器。

**4.JVM新生代，老年代，持久代，都存储哪些东西？**

持久代主要存放的是Java类的类信息，与垃圾收集要收集的Java对象关系不大。所有新生成的对象首先都是放在年轻代的，年老代中存放的都是一些生命周期较长的对象。

**5.内存溢出和内存泄漏：**

内存溢出：程序申请内存时，没有足够的内存，out of memory；内存泄漏值垃圾对象无法回收，可以使用memory analyzer工具查看泄漏。

**6.进程与线程：**

进程值运行中的程序（独立性，动态性，并发性），线程指进程中的顺序执行流。区别是：1.进程间不共享内存 2.创建进程进行资源分配的代价要大得多，所以多线程在高并发环境中效率高。

**7.序列化与反序列化：**

序列化指将java对象转化为字节序列，反序列化相反。主要是为了java线程间通讯，实现对象传递。只有实现了Serializable或Externalizable接口类对象才可被序列化。

**8.64 位 JVM 中，int 的长度是多数？**

Java 中，int 类型变量的长度是一个固定值，与平台无关，都是 32 位。意思就是说，在 32 位 和 64 位 的Java 虚拟机中，int 类型的长度是相同的。

**9.Java 中 WeakReference 与 SoftReference的区别？**

Java中一共有四种类型的引用。StrongReference、 SoftReference、 WeakReference 以及 PhantomReference。

StrongReference 是 Java 的默认引用实现, 它会尽可能长时间的存活于 JVM 内，当没有任何对象指向它时将会被GC回收

WeakReference，顾名思义, 是一个弱引用, 当所引用的对象在

JVM 内不再有强引用时, 将被GC回收

虽然 WeakReference 与 SoftReference 都有利于提高 GC 和 内存的效率，但是 WeakReference ，一旦失去最后一个强引用，就会被 GC 回收，而 SoftReference 会尽可能长的保留引用直到 JVM 内存不足时才会被回收(虚拟机保证), 这一特性使得

SoftReference 非常适合缓存应用

**10.解释 Java 堆空间及 GC？**

当通过 Java 命令启动

Java 进程的时候，会为它分配内存。内存的一部分用于创建堆空间，当程序中创建对象的时候，就从对空间中分配内存。GC 是 JVM 内部的一个进程，回收无效对象的内存用于将来的分配。

**11.Java 中堆和栈有什么区别？**

JVM 中堆和栈属于不同的内存区域，使用目的也不同。栈常用于保存方法帧和局部变量，而对象总是在堆上分配。栈通常都比堆小，也不会在多个线程之间共享，而堆被整个 JVM 的所有线程共享。

**并发，锁**

**1.volatile关键字， Lock**

并发编程中：原子性问题，可见性问题，有序性问题。

volatile关键字能保证可见性，字能禁止指令重排序，但是不能保证原子性。可见性只能保证每次读取的是最新的值，但是volatile没办法保证对变量的操作的原子性。在生成的会变语句中加入Lock关键字和内存屏障。

Lock 实现提供了比使用synchronized 方法和语句可获得的更广泛的锁定操作，它能以更优雅的方式处理线程同步问题。用sychronized修饰的方法或者语句块在代码执行完之后锁自动释放，而用Lock需要我们手动释放锁

**2.MYSQL常用优化（sql优化，表结构优化等）**

SQL优化、表机构优化、索引优化、缓存参数优化

**3.java每改一点都需要重新编译打包部署，有没有更好的方法**

可以使用热加载

**4.进程间通信有哪几种方式？**

1）管道（Pipe），2）命名管道（named pipe），3）信号（Signal），4）消息（Message）队列，5）共享内存，6）内存映射（mapped memory），7）信号量（semaphore），8）套接口（Socket）

**5.Sychronized修饰静态方法，锁定类本身而不是实例，非静态方法锁定实例。**

**6. 操作系统什么情况下会死锁？**

所谓死锁：是指多个进程在运行过程中因争夺资源而造成的一种僵局。产生的原因：竞争资源：当系统中多个进程使用共享资源，并且资源不足以满足需要，会引起进程对资源的竞争而产生死锁。进程间推进的顺序非法：请求和释放资源的顺序不当，也同样会导致产生进程死锁

**7.产生死锁的四个条件：**

1.互斥条件（进程独占资源）2.请求与保持（进程因请求资源而阻塞时，对已获得的资源保持不放） 3.不剥夺条件（进程已获得的资源，在末使用完之前，不能强行剥夺） 4.循环等待（若干进程之间形成一种头尾相接的循环等待资源关系）

**8. 如何理解分布式锁？**

由于在平时的工作中，线上服务器是分布式多台部署的，经常会面临解决分布式场景下数据一致性的问题，那么就要利用分布式锁来解决这些问题。

**9. 线程同步与阻塞的关系？同步一定阻塞吗？阻塞一定同步吗？**

线程同步与否 跟 阻塞非阻塞没关系，同步是个过程，阻塞是线程的一种状态。多个线程操作共享变量时可能会出现竞争。这时需要同步来防止两个以上的线程同时进入临界区内，在这个过程中后进入临界区的线程将阻塞，等待先进入的线程走出临界区。

**10. 同步和异步有什么区别？**

同步和异步最大的区别就在于。一个需要等待，一个不需要等待。同步可以避免出现死锁，读脏数据的发生，一般共享某一资源的时候用，如果每个人都有修改权限，同时修改一个文件，有可能使一个人读取另一个人已经删除的内容，就会出错，同步就会按顺序来修改。

**11. 线程池**

根据系统自身的环境情况，有效的限制执行线程的数量，使得运行效果达到最佳。线程主要是通过控制执行的线程的数量，超出数量的线程排队等候，等待有任务执行完毕，再从队列最前面取出任务执行

**12. 如何调用 wait（）方法？使用 if 块还是循环？为什么？**

wait() 方法应该在循环调用，因为当线程获取到 CPU 开始执行的时候，其他条件可能还没有满足，所以在处理前，循环检测条件是否满足会更好。

wait()，notify（）和notifyall（）方法是java.lang.Object类为线程提供的用于实现线程间通信的同步控制方法。等待或者唤醒

**13. 实现线程的几种方法**

(1)继承Thread类，重写run函数

(2)实现Runnable接口，重写run函数

(3)实现Callable接口，重写call函数

**14. 什么是多线程环境下的伪共享（false sharing）？**

伪共享是多线程系统（每个处理器有自己的局部缓存）中一个众所周知的性能问题。缓存系统中是以缓存行（cache line）为单位存储的。缓存行是2的整数幂个连续字节，一般为32-256个字节。最常见的缓存行大小是64个字节。当多线程修改互相独立的变量时，如果这些变量共享同一个缓存行，就会无意中影响彼此的性能，这就是伪共享。

**网络、数据库**

**1.TCP如何保证可靠传输？三次握手过程？**

在TCP的连接中，数据流必须以正确的顺序送达对方。TCP的可靠性是通过顺序编号和确认（ACK）来实现的。TCP 连接是通过三次握手进行初始化的。三次握手的目的是同步连接双方的序列号和确认号并交换 TCP 窗口大小信息。第一次是客户端发起连接；第二次表示服务器收到了客户端的请求；第三次表示客户端收到了服务器的反馈。

**2. Linux下你常用的命令有哪些？**

1. cd命令用来改变所在目录。cd / 转到根目录中cd ~ 转到用户目录下

2. ls命令用来查看目录的内容。

3. cp命令用来拷贝文件cp

4.mv命令 mv t.txt Document 把文件t.txt 移动到目录Document中。

**3. 常用的hash算法有哪些？**

1.加法hash：所谓的加法Hash就是把输入元素一个一个的加起来构成最后的结果。

2.位运算hash：这类型Hash函数通过利用各种位运算（常见的是移位和异或）来充分的混合输入元素

3.乘法hash：33\*hash + key.charAt(i)

**4. 什么是一致性哈希？**

设计目标是为了解决因特网中的热点(Hot spot)问题，一致性hash算法提出了在动态变化的Cache环境中，判定哈希算法好坏的四个定义：1、平衡性(Balance) 2、单调性(Monotonicity) 3、分散性(Spread) 4、负载(Load)

**5. 数据库中的范式有哪些？**

第一范式----数据库中的表(所有字段值)都是不可分割的原子数据项。

第二范式----数据库表中的每一列都和主键相关，而不能只和主键的某一部分相关。

第三范式----数据库表中每一列数据都和主键直接相关，不能间接相关。范式是为了减小数据冗余。

**6. 数据库中的索引的结构？什么情况下适合建索引？**

数据库中索引的结构是一种排序的数据结构，数据库索引是通过B树和变形的B+树实现的。什么情况下不适合建立索引：1.对于在查询过程中很少使用或参考的列；对于那些只有很少数据值的列；对于那些定义为image，text和bit数据类型的列；当修改性能远大于检索性能。

根据系统自身的环境情况，有效的限制执行线程的数量，使得运行效果达到最佳。线程主要是通过控制执行的线程的数量，超出数量的线程排队等候，等待有任务执行完毕，再从队列最前面取出任务执行

**7. concurrent包下面，都用过什么？**

java.util.concurrent、java.util.concurrent.atomic和java.util.concurrent.lock

**8. 常用的数据库有哪些？redis用过吗？**

…

**9. 你知道的开源协议有哪些？**

GPL （GNU General Public License） ：GNU通用公共许可协议

LGPL （GNU Lesser General Public License） ：GNU宽通用公共许可协议

BSD

(Berkeley Software Distribution) :伯克利软件分发许可协议

MIT （Massachusetts Institute of Technology）：MIT之名源自麻省理工学院

Apache （Apache License） ：Apache许可协议

MPL （Mozilla Public License） ：Mozilla公共许可协议

**10.表单提交中，get和post区别**

1.get从服务器获取信息，post向服务器传信息

2.get传送数据量比较小，post可以比较大

3.get安全性比较低

**11. TCP 协议与 UDP 协议有什么区别？(answer答案)**

TCP（Tranfer Control Protocol）的缩写，是一种面向连接的保证传输的协议，在传输数据流前，双方会先建立一条虚拟的通信道。可以很少差错传输数据。

UDP(User DataGram Protocol)的缩写，是一种无连接的协议，使用UDP传输数据时，每个数据段都是一个独立的信息，包括完整的源地址和目的地，在网络上以任何可能的 路径传到目的地，因此，能否到达目的地，以及到达目的地的时间和内容的完整性都不能保证。

所以TCP必UDP多了建立连接的时间。相对UDP而言，TCP具有更高的安全性和可靠性。

TCP协议传输的大小不限制，一旦连接被建立，双方可以按照一定的格式传输大量的数据，而UDP是一个不可靠的协议，大小有限制，每次不能超过64K。