总结：

第一：

1.做过项目。

2.有看过哪些JDK源码，了解哪些常用库。

3.集合框架HashMap的扩容机制，ConcurrnetHashMap的原理

   那么HashMap什么时候进行扩容呢？当HashMap中的元素个数超过数组大小\*loadFactor时，就会进行数组扩容，loadFactor的默认值为0.75，这是一个折中的取值。也就是说，默认情况下，数组大小为16，那么当HashMap中元素个数超过16\*0.75=12的时候，就把数组的大小扩展为 2\*16=32，即扩大一倍，然后重新计算每个元素在数组中的位置，而这是一个非常消耗性能的操作，所以如果我们已经预知HashMap中元素的个数，那么预设元素的个数能够有效的提高HashMap的性能。

4.jvm内存模型与gc内存回收机制

**程序计数器**  
多线程时，当线程数超过CPU数量或CPU内核数量，线程之间就要根据时间片轮询抢夺CPU时间资源。因此每个线程有要有一个独立的程序计数器，记录下一条要运行的指令。线程私有的内存区域。如果执行的是JAVA方法，计数器记录正在执行的java字节码地址，如果执行的是native方法，则计数器为空。  
**虚拟机栈**  
线程私有的，与线程在同一时间创建。管理JAVA方法执行的内存模型。每个方法执行时都会创建一个桢栈来存储方法的的变量表、操作数栈、动态链接方法、返回值、返回地址等信息。栈的大小决定了方法调用的可达深度（递归多少层次，或嵌套调用多少层其他方法，-Xss参数可以设置虚拟机栈大小）。栈的大小可以是固定的，或者是动态扩展的。如果请求的栈深度大于最大可用深度，则抛出stackOverflowError；如果栈是可动态扩展的，但没有内存空间支持扩展，则抛出 OutofMemoryError。  
使用jclasslib工具可以查看class类文件的结构。

**本地方法区**  
和虚拟机栈功能相似，但管理的不是JAVA方法，是本地方法，本地方法是用C实现的。

**JAVA堆**  
线程共享的，存放所有对象实例和数组。垃圾回收的主要区域。可以分为新生代和老年代(tenured)。  
新生代用于存放刚创建的对象以及年轻的对象，如果对象一直没有被回收，生存得足够长，老年对象就会被移入老年代。  
新 生代又可进一步细分为eden、survivorSpace0(s0,from space)、survivorSpace1(s1,to space)。刚创建的对象都放入eden,s0和s1都至少经过一次GC并幸存。如果幸存对象经过一定时间仍存在，则进入老年代(tenured)。

**方法区**  
线程共享的，用于存放被虚拟机加载的类的元数据信息：如常量、静态变量、即时编译器编译后的代码。**也成为永久代。**如果hotspot虚拟机确定一个类的定义信息不会被使用，也会将其回收。回收的基本条件至少有：所有该类的实例被回收，而且装载该类的ClassLoader被回收

**垃圾回收算法**

**标记-清除算法(Mark-Sweep)**  
从根节点开始标记所有可达对象，其余没标记的即为垃圾对象，执行清除。但回收后的空间是不连续的。  
**复制算法(copying)**  
将内存分成两块，每次只使用其中一块，垃圾回收时，将标记的对象拷贝到另外一块中，然后完全清除原来使用的那块内存。复制后的空间是连续的。**复制算法适用于新生代，**因 为垃圾对象多于存活对象，复制算法更高效。在新生代串行垃圾回收算法中，将eden中标记存活的对象拷贝未使用的s1中，s0中的年轻对象也进入s1，如 果s1空间已满，则进入老年代；这样交替使用s0和s1。这种改进的复制算法，既保证了空间的连续性，有避免了大量的内存空间浪费。

**标记-压缩算法(Mark-compact)**  
适合用于**老年代**的算法（存活对象多于垃圾对象）。  
标记后不复制，而是将存活对象压缩到内存的一端，然后清理边界外的所有对象。

5.classloader结构，是否可以自己定义一个java.lang.String类，为什么？ 双亲代理机制。

双亲代理机制可以避免重复加载，当父亲已经加载了该类的时候，就没有必要子ClassLoader再加载一次。考虑到安全因素，我们试想一下，如果不使用这种 委托模式，那我们就可以随时使用自定义的String来动态替代java核心api中定义的类型，这样会存在非常大的安全隐患，而双亲委托的方式，就可以 避免这种情况，因为String已经在启动时就被引导类加载器（Bootstrcp ClassLoader）加载，所以用户自定义的ClassLoader永远也无法加载一个自己写的String，除非你改变JDK中 ClassLoader搜索类的默认算法。

6.了解哪些设计模式，6个设计原则分别是什么？每种设计原则体现的设计模式是哪个？

单一职责原则：就一个类而言，应该仅有一个引起它变化的原因。(单例模式)

开放封闭原则：软件实体（类、模块、函数等等）应该可以扩展，但是不可以修改。(工厂方法)

依赖倒转原则：A、高层模块不应该依赖底层模块，两个应该都依赖抽象；(解释器模式)

B、抽象不应该依赖细节，细节应该依赖抽象。

里氏代换原则：子类型必须能够替换掉它们的父类型。

合成/聚合复用原则：尽量使用合成/聚合，尽量不要使用继承。

迪米特法则：如果两个类不必彼此直接通信，那么这两个类就不应当发生直接的相互作用。

如果其中一个类需要调用另一个类的某一个方法的话，可以通过第三者转发这个调用(代理模式)

7.关于设计模式看了哪些书？书名是什么？

《设计模式》《设计模式之禅》《大话设计模式》

8.**uml模型图**画过哪些？ 类图中类之间的关系有哪些，区别分别是什么？

功能模型: 从用户的角度展示系统的功能，包括**用例图**。  
对象模型: 采用对象，属性，操作，关联等概念展示系统的结构和基础，包括类图。  
动态模型: 展现系统的内部行为。 包括序列图，活动图，状态图。

类的关系有泛化(Generalization)、实现（Realization）、依赖(Dependency)和关联(Association)。其中关联又分为一般关联关系和聚合关系(Aggregation)，合成关系(Composition)

9.画uml中类图时候用过一种虚线么？做什么用的？

10.做过应用相关性能测试的，举个例子，实际项目中怎么使用的。

用过并发框架相关的哪些内容

11.了解哪些osgi的框架？

Equinox eclipse

12有没有做过jvm内存调优，如何做的，举例子，用过哪些工具？

锁的本质是什么 java内部怎么实现的

aqs

ReentrantLock锁机制的实现是基于它的一个成员变量sync，这个Sync是 AbstractQueuedSynchronized(AQS)的一个子类（ps：sync类是ReentrantLock自己定义的一个内部类）。另 外在ReentrantLock内部还定义了另外两个类，分别是FairSync和NonFairSync，这两个类就是分别对应的锁公平分配和不公平分 配的两个实现，它们都继承自Sync（类图已经清晰的描述出来了继承结构）。有关锁的分配和释放逻辑都是封装在了AQS里面的（AQS是 AbstractQueuedSynchronized的简称，是JSR166规范中提出的一个基础的同步中心类或者说是同步框架，其在内部实现了大量的 同步操作，而且用户还可以在此类的基础上自定义自己的同步类），可见Sync和AQS是锁机制实现的核心类（AQS详述见下文）。

自旋锁原理

怎么实现不可重入锁，只能重入两次怎么做

两个超大数相乘

选项，消息队列和rpc该用谁，为什么不用另一个

cup内存屏障都有啥，volatile 用什么内存屏障 解释loadstorage

* 编译器引起的内存屏障
* 缓存引起的内存屏障
* 乱序执行引起的内存屏障

注解处理器是啥

怎么样使用永久内存 除了加载class 和使用常量池

第二视

1.自我介绍

2.看过哪些源码

3.java的io库的类结构图所用到的设计模式如何体现

说设计模式的话，用到了装饰模式和适配器模式，下面简要说明，

1、装饰者模式：在由 InputStream，OutputStream，Reader和Writer代表的等级结构内部，

有一些流处理器可以对另一些流处理器起到装饰作用，形成新的，具有改善了的功能的流处理器。

装饰者模式是Java I/O库的整体设计模式。这样的一个原则是符合装饰者模式的。

2、适配器模式：在由InputStream，OutputStream，Reader和Writer代表的等级结构内部，

有一些流处理器是对其它类型的流源的适配。这就是适配器模式的应用。

4.画出自己设计过的设计模式如何体现，画出结构图，并进行讲解。

5.画出自己做的架构的项目架构图 如何扩展等

6.数据库设计中主键id设计的原则

7.jvm内存调优用过哪些工具，jstate做什么用的？如何dump出当前线程状态？

8.并发框架是否有了解

9.classloader的双亲代理机制

10.应用服务器的jvm调优实际经验，如何做的，在哪里用到的

**1.内存调优**   
内存方式的设置是在catalina.sh中，调整一下JAVA\_OPTS变量即可，因为后面的启动参数会把JAVA\_OPTS作为JVM的启动参数来处理。   
  
具体设置如下：   
JAVA\_OPTS="$JAVA\_OPTS -Xmx3550m -Xms3550m -Xss128k -XX:NewRatio=4 -XX:SurvivorRatio=4"

**2.垃圾回收策略调优**   
垃圾回收的设置也是在catalina.sh中，调整JAVA\_OPTS变量。   
具体设置如下：   
JAVA\_OPTS="$JAVA\_OPTS -Xmx3550m -Xms3550m -Xss128k -XX:+UseParallelGC  -XX:MaxGCPauseMillis=100"   
具体的垃圾回收策略及相应策略的各项参数如下：

串行收集器（JDK1.5以前主要的回收方式）

-XX:+UseSerialGC:设置串行收集器

并行收集器（吞吐量优先）

示例：

java -Xmx3550m -Xms3550m -Xmn2g -Xss128k -XX:+UseParallelGC -XX:MaxGCPauseMillis=100

-XX:+UseParallelGC：选择垃圾收集器为并行收集器。此配置仅对年轻代有效。即上述配置下，年轻代使用并发收集，而年老代仍旧使用串行收集。

-XX:ParallelGCThreads=20：配置并行收集器的线程数，即：同时多少个线程一起进行垃圾回收。此值最好配置与处理器数目相等。

-XX:+UseParallelOldGC：配置年老代垃圾收集方式为并行收集。JDK6.0支持对年老代并行收集

-XX:MaxGCPauseMillis=100:设置每次年轻代垃圾回收的最长时间，如果无法满足此时间，JVM会自动调整年轻代大小，以满足此值。

-XX:+UseAdaptiveSizePolicy：设置此选项后，并行收集器会自动选择年轻代区大小和相应的Survivor区比例，以达到目标系统规定的最低相应时间或者收集频率等，此值建议使用并行收集器时，一直打开。

并发收集器（响应时间优先）

示例：java -Xmx3550m -Xms3550m -Xmn2g -Xss128k -XX:+UseConcMarkSweepGC

-XX:+UseConcMarkSweepGC：设置年老代为并发收集。测试中配置这个以后，-XX:NewRatio=4的配置失效了，原因不明。所以，此时年轻代大小最好用-Xmn设置。

-XX:+UseParNewGC: 设置年轻代为并行收集。可与CMS收集同时使用。JDK5.0以上，JVM会根据系统配置自行设置，所以无需再设置此值。

-XX:CMSFullGCsBeforeCompaction：由于并发收集器不对内存空间进行压缩、整理，所以运行一段时间以后会产生“碎片”，使得运行效率降低。此值设置运行多少次GC以后对内存空间进行压缩、整理。

-XX:+UseCMSCompactAtFullCollection：打开对年老代的压缩。可能会影响性能，但是可以消除碎片

11.在哪里获取最新资讯，逛什么论坛。最新的Swift语言有什么看法

12.设计原则与设计模式对应

13.servlet/filter作用原理配置

1) 客户端发送请求至服务器端；

2) 服务器将请求信息发送至 Servlet；

3) Servlet 生成响应内容并将其传给服务器。响应内容动态生成，通常取决于客户端的请求；

4) 服务器将响应返回给客户端。

在 Web 应用程序中，一个 Servlet 在一个时刻可能被多个用户同时访问。

这时 Web 容器将为每个用户创建一个线程来执行 Servlet。

如果 Servlet 不涉及共享资源的问题，不必关心多线程问题。

但如果 Servlet 需要共享资源，需要保证 Servlet 是线程安全的。

1 <filter>

2 <filter-name>logfilter</filter-name>

3 <filter-class>com.mine.test.LogFilter</filter-class>

4 </filter>

5 <filter-mapping>

6 <filter-name>logfilter</filter-name>

7 <url-pattern>/\*</url-pattern> <!--配置过滤的范围 后缀符合即过滤 此处为全部过滤-->

8 </filter-mapping>

14.ibatis in操作 以及一个属性的作用

<select id="findByIdsMap" resultMap="BaseResultMap">

Select

<include refid="Base\_Column\_List" />

from jria where ID in

<foreach item="item" index="index" collection="list"

open="(" separator="," close=")">

#{item}

</foreach>

</select>

15.spring aop 用了什么设计原则，自动注入配置是做什么用的

aop主要用的是动态代理，也就是代理模式(迪米特法则)

16.jboss的类加载器

17.session共享机制

目前的处理方式有如下几种：

1).使用Tomcat本身的Session复制功能

参考<http://ajita.iteye.com/blog/1715312>（Session复制的配置）

方案的有点是配置简单，缺点是当集群数量较多时，Session复制的时间会比较长，影响响应的效率

2).使用第三方来存放共享Session

目前用的较多的是使用memcached来管理共享Session，借助于memcached-sesson-manager来进行Tomcat的Session管理

参考<http://ajita.iteye.com/blog/1716320>（使用MSM管理Tomcat集群session）

3).使用黏性session的策略

对于会话要求不太强（不涉及到计费，失败了允许重新请求下等）的场合，

同一个用户的session可以由nginx或者apache交给同一个Tomcat来处理，

这就是所谓的session sticky策略，目前应用也比较多

参考：<http://ajita.iteye.com/blog/1848665>（tomcat session sticky）

nginx默认不包含session sticky模块，需要重新编译才行（windows下我也不知道怎么重新编译）

优点是处理效率高多了，缺点是强会话要求的场合不合适

18.做过最成功的一件事情是什么？

19.最大的争执是什么？

20.为什么

第三电：

1.现在公司负责什么？

2.项目主要目的是做什么的？

3.公司管理方式、项目问题反馈机制是什么？

4.Java 的序列化做什么用的 序列化id会出现哪些问题？

5.OSGi用过哪些？类加载器结构如何，如何在一个bundle中加载另外一个bundle中的一个类？

6.nio是否了解 阻塞之后通知机制是怎样的？

7.uml设计类图如何画，类之间关系以及区别

8.spring如何不许要配置文件加载bean定义，可能是问自动注解或者是properties文件定义bean

9.ibatis等框架是不是都是实际在使用的,技术细节

10.为什么想离职去阿里

//一些不记得了

第四电：

1.自我介绍

2.公司做什么，业务， 负责内容，汇报机制等

3.企业级应用安全相关

4.http协议，返回码，301与302区别

5.多线程并发用过哪些

6.应用服务器相关，谈最熟悉的

7.为什么离职

//这个太多不记得了，很多不太会。