面试题大全

Java 高级软件工程师

班　　级：_____

姓　　名：_____

联系电话：_____

写在前面的话

从我们翻开厚厚面试题学习开始，预示着我们离开学校的脚本越来越近了，就业压力越来越大了。偶尔会在脑海中勾画、想像我们以后工作的情景了，而步入社会参与竞争的第一步就是从学习如何面试！如何找工作开始。面试包括人事交流沟通与技术问答等相关环节，在这本面试题书中包含了这两部分内容，第一部分主要讲人事问题的一些常见问题，第二部分讲技术问答。在技术面试问答题中无法将所有公司可能问到的所有问题都一一收录进去；也无法对每个问题，每个知识做深入精细的分析讲解，因为这是一个没有结点的工作，所以除了学习这本书上的面试问答的知识外与课程相关的知识我们也要去学习。

在学校期间我们在学习老师教授给我们的专业知识的同时还有些知识是无法在学校的课堂中吸取到的，如：**缓存 cache、事务与并发、海量数据、Lucene、NoSQL、Hadoop、系统日志、单点登录、UI 框架、项目管理与集成、设计模式、移动互联网开发、各公司自有开发框架**等等，还有很多很多，因为软件开发行业所包含的技术包罗万象。我们虽然了目前软件行业的主流技术，但也有一些企业，因架构师或项目经理的技术特点，会选择一些我们未接触或未听说过的的技术架构，那么作为即将步入软件工程师行业的我们，必须要有重新学习新技术的心态和作好准备学习新技术的准备。同样，在以后的工作过程中也会遇到一些我们之前未接触过的技术或未遇到的项目 Bug，这些都在要求我们在今后的工作和学习中还需要不断的学习，另外还要持有一颗：生命不息，学习不止的心。只有这样，我们才能成为软件行业与时俱进的技术人才。

最后也希望每位 IT 学子，通过面试题能够获得更多的收获，学有所成。也预祝每们学子能找到自己满意的工作。

一、简历制作与面试技巧

➢ 简历相关

1) 简历要求

简历主要是用人单位用来认识你、了解你的一种方式或一种参考，这点很重要。因为用人单位和你都不太了解对方，无论对你或者用人单位而言都是第一次打交道，在刚开始接触中。对你而言简历是你想要展现你的过去的一种形式。另外简历是我们在求职过程中与用人单位打交道的第一环节，这个环节过关了，才有可能进入下一环节。所以我们要认真的准备我们的简历，把我们优秀的过去通过简历来展现给面试官；让面试官通过简历来认识、了解你过去的经历。

2) 简历内容

包括个人基本信息、专业技能、工作经历、项目经验、所获证书、个人评价等内容。这些内容决定了你是否符合招聘岗位要求。

基本信息：面试官通过这部分来了解你是男，还是女，年龄、工作经验、应聘岗位、最高学历，看看这些方面是否符合他的招聘要求。

专业技术：面试官通过专业技术主要了解你对各种专业技能的掌握和使用情况，对比你所具有的这些技能是否符合他所招聘的岗位技能。

工作经历：面试官主要来了解你之前是否有工作的经验的人，还是应届生等等。

项目经验：面试官主要看看你是否有从事过相关工作的实际开发经验。

所获证书：看看是何种证书，证书的含金量怎么样。

人个评价：看看你的综合素质、个人性格等方面

3) 简历示例

◆ **基本信息**

个人信息

姓　　名：	XXXX	性　　别：	XX
年　　龄：	22	现 居 地：	XX 市 XX 区
毕业院校：	湖北大学	工作经验：	2 年
专　　业	计算机科学与技术	学　　历：	本科
联系电话：	XXXX	应聘职位：	JAVA 软件工程师
电子邮件：	XXXX		

◆ **专业技能**

专业技能

- 熟悉 C、C#、Java 等程序设计语言。
- 熟练掌握 HTML、XML、CSS 层叠样式、Javascript 等前端脚本语言
- 熟练使用 Struts1、Struts2、Spring3、Hibernate3、MyBatis、JQuery 等主流开源框架进行应用开发。
- 熟练使用 Oracle，SQLServer，MySQL 关系型数据库。
- 熟练使用 Tomcat，Weblogic，JBoss，Apache HTTP Server 等服务器。
- 熟练使用 MyEclipse，Dreamweaver，PL/SQL，PowerDesiger 等开发工具。
- 熟悉 JBPM 工作流引擎原理

◆ 工作经历

工作经验

工作时间	单位	职位
2010.3—2012.12	武汉智诚信息科技有限公司	Java 工程师
2009.5--2010.1	中地数码武汉分公司	Java 工程师

◆ 项目经验

项目经验 1

✧2010 年 4 月--2010 年 10 月：

- **项目名称：** 湖北黄冈市房地产税收一体化系统
- **开发框架：** Struts2.0+Hibernate3.0+Spring2.0+Jquery+Xfire
- **开发环境：** jdk、tomcat、eclipse
- **数据库**: Oracle10
- **项目背景：**为了实现客户方无纸化办公，提高办事效率，以及实现对下属各税务分局进行统一的税收管理，该管理系统包括:税源信息管理、纳税评估、行政执法、综合查询、系统设置、信息变更、电子档案、保有环节、信息比对、零散契税 10 个大模块。
- **责任描述：**

（1）首页电子地图显示：主要基于电子地图框架 MapEasy 进行二次开发，在原型功能上实现地物标注、等级缩放、瓦片移动、地图切换等功能。二次开发后通过地理位置显示各税务局所属片区土地交易信息、在建项目工程信息、并按坐标进行标注显示、查看、预览效果图，实现更方便、快捷、直观的操作和显示功能。

（2）各行政区税费统计：主要根据不同地税分局统计出欠税纳税人数、欠税总金额、应缴总金额、已缴总金额等，并以图形方式显示数据。

（2）系统设置：主要包括用户管理、权限设置、个人纳税人管理、单位纳税人信息管理、土地管理、税率设置、地图管理、基础数据管理。

➢ 面试环节

1) 面试准备

1、了解清楚岗位需求与岗位职责。
2、检查简历描述是否与所应聘的岗位需求一致。
3、了解用人单位的一些基本信息，如：从事什么行业的，公司产品、岗位要求
4、准备好用人单位所需的一切证件，以免白跑一趟。

2) 面试过程

1、普通话标准。
2、表现要自信、大方。
3、沟通交流过程中表情自然、得当。
4、着装要让人感觉很正自然、看着很舒服。
5、交流过程中懂得谦让和尊重面试官

3) 面试自我介绍

1、介绍个人基本信息，包括毕业院校。
2、介绍之前的工作经历，包括岗位名称
3、介绍之前参与的项目，包括客户方名称、技术架构

4) 面试注意事项

内容过多，附网址：***http://job.group.iteye.com/group/wiki/?category_id=13***
上面的版块叫《求职宝典》，是 iteye(专业 Java 技术站点)上主要针对 IT 行业求职过程分析和求职问题注意事项，包括如何写简历、如何跟 IT 经理面谈等内容。作为一个即将步入 IT 行业的初级程序员来说里面的内容不可错过。

二、职场相关

人在求职或学习生活过程中，很多知识和信息是通过环境给我们传达的，下面我们随着这些问题来学习一下工作环境中的学问，请回答以下题目：

- **学习教育、培训相关**

1、你们学校是属于几本？

2、你的学历是全日制，还是非全日制？

3、你为什么选择你所学的专业？有什么原因？

4、你能列举一下：大一、大二、大三、大四分别学了哪些课程？你比较喜欢哪门课程？为什么喜欢？

5、你毕业的毕业论文是什么题目？主要内容是什么？花了多长时间？是否有答辨，你的导师是怎么评论你的论文的？

6、你毕业时修了多少学分？毕业时取得了哪些证书？是否有学位证？

7、你的英语过了几级？英语的成绩是多少？

8、说说你的大学生活？让你印象最深的是什么？

9、你们学校有几个校区？每个校区有什么特点？你在哪个校区？附近有什么标志性的建筑？怎么到你们学校？

10、你目前的档案在哪里？可以转到我们公司吗？

- **职场相关**

1、你们公司在 XX（市）的什么地方？公司有多少人？有多少个部门？

2、你们公司的网址是什么？联系电话是多少？

3、你们公司的大致的组织机构是怎么构成的？你属于哪个部门，你们部门在职责的是什么？你在公司的职位？是的直接汇报对象哪个职位？

4、你们公司主要面向哪些行业的？有哪些客户，列举一下？主要有做过哪些系统？

5、你的项目经理叫什么名字，能介绍一下你对他的了解和看法吗？

6、你在公司参与过哪些项目？你们公司跟哪些其它的公司有过合过？

7、为什么离职？你有离职证明吗？

8、你在公司做项目时，遇到困难怎么克服的？举例说明。

9、你对加班怎么看？

10、你和同事之间的关系怎么样？你怎么看待同事间相处关系？

11、你所工作的城市有什么特色？城市有什么特产可以向我们介绍的？有什么好玩的景点可以推荐一下吗？

12、平时下班后做些什么？周末在家主要做什么？

13、你的项目代码可以拿来我们看看吗？

14、谈谈你对工资的要求？

15、你以前的工资构成？

16、你的工资扣税吗？是怎么扣的？

17、你原来有保险吗？保险号是多少？你原来的公司还有什么福利？

18、你来来的工作是向谁汇报，一般会汇报一些什么内容？

19、你平时一般会访问哪些技术网站？

三、技术面试题

（一）Java 部分

1、列举出 JAVA 中 6 个比较常用的包【天威诚信面试题】

【参考答案】

java.lang;java.util;java.io;java.sql;java.awt;java.net;javax.swing

2、JDK 中哪些类是不能继承的？【信雅达面试题】

【参考答案】

不能继承的是类是那些用 final 关键字修饰的类。一般比较基本的类型或防止扩展类无意间破坏原来方法的实现的类型都应该是 final 的。

3、String 是最基本的数据类型吗？【天能智健面试题】

【参考答案】

基本数据类型包括 byte、int、char、long、float、double、boolean 和 short。

所以 String 不属于基本数据类型范畴内，但 String 属于最常见一种引用类型。

4、short s1 = 1; s1 = s1 + 1;有什么错? short s1 = 1; s1 += 1;有什么错？【博炎科技面试题】

【参考答案】

对于 short s1 = 1; s1 = s1 + 1; 由于 s1+1 运算时会自动提升表达式的类型，所以结果是 int 型，再赋值给 short 类型 s1 时，编译器会提示错误，需要强制转换类型。

对于 short s1 = 1; s1 += 1;由于 += 是 java 语言规定的运算符，Java 编译器会对它进行特殊处理，因此可以正确编译。

【分析】

主要考查几种基本数据类型在运算时的，由低到高会自动转换，而由高到低时会强制转换。

5、Java 对象初始化顺序？【腾鹏科技面试题】

【参考答案】

分为两种，一种是本类的初始化，一种是含有父类的初始化顺序。这里分开来说，

本类的初始化顺序是：静态变量、静态初始化块、变量、初始化块、构造函数

继承类的初始化顺序是：父类静态变量、父类静态初始化块、子类静态变量、子类静态初始块、父类变量、父类初始化块、父类构造函数、子类变量、子类初始化块、子类构造函数。

【分析】

```
static{
   System.out.println("静态块");
  }
  {
   System.out.println("初始化模块");
  }
  public ClassName() {
   System.out.println("构造方法");
  }
```

说明：

原则上回答全面的话，应该是完整的说出带有继承的这种类的初始化过程，下面有个步骤可以参考：

1.装载程序的时候首先找到的是它的基（父）类,如果有多层基（父）类则会一级一级的往上找最后找到根基（父）类。

2.执行根基础（父）类中的 static 初始化,再执行下一个衍生类中的 static,依此类推,一直保持这个顺序。

3.此时类已经装载完毕,开始创建对象,所有的基本数据类型都会设成它们的默认值,对象句柄设为 null

4.调用基础（父）类的构造方法,基础（父）类的构建采用与衍生类构造方法完全相同的处理过程。

5.构造方法初始完之后,进行变量的初始化。

6.执行构造方法中剩余的部分。

6、写几个线程安全类，不安全的，支持排序的类名？【软通动力面试题】

【参考答案】

- 线程安全类：Vector 、Hashtable、Stack。
- 线程不安全的类：ArrayList、Linkedlist、HashSet、TreeSet、HashMap、TreeMap 等。
- 支持排序的类有 HashSet、LinkedHashSet、TreeSet 等（Set 接口下的实现都支持排序）

【分析】

此题主要考查集合框架的知识。在集合框架中 Collection 接口为集合的根类型，提供集合操作的常用 API 方法，该接口下派生出两个子接口，一个是不支持排序的 List 接口，一个是有自身排序的 Set 接口，所以回答排序与不排序分别从两接口的实现中在作答。线程安全上来说，Vector 类比同属于 List 接口的 ArrayList 要早，是一个线程安全的类，在 JDK1.2 以后才推出一个异步的 ArrayList 类,比 Vector 类效率高。同理 Stack 继承自 Vector 也线程安全的类，另外在在 Map 接口的实现在 Hashtable 也是个线程安全的类。

7、哪几个方法可以实现一个线程？【上海华信面试题】

【参考答案】

一是继承 Thread ，重写 Thread 类的方法 run 方法；另种是实现 runnable 接口并实现 run 方法。

【分析】

考查线程的基本实现，很多公司喜欢考查这方面知识，另外补充一下关于线程的 run 方法，在多线程 API 中启动一个线程是调用 start()方法，线程进入就绪状态。

8、STOP()和 SUSPEND()不推荐使用的原因？

【参考答案】

stop()是因为它不安全。它会解除由线程获取的所有锁定,当在一个线程对象上调用 stop()方法时，这个线程对象所运行的线程就会立即停止，假如一个线程正在执行：synchronized void { x = 3; y = 4;} 由于方法是同步的，多个线程访问时总能保证 x,y 被同时赋值，而如果一个线程正在执行到 x = 3;时，被调用了 stop()方法，即使在同步块中，它也干脆地 stop 了，这样就产生了不完整的残废数据。而多线程编程中最最基础的条件要保证数据的完整性,所

以请忘记线程的 stop 方法，以后我们再也不要说“停止线程”了。而且如果对象处于一种不连贯状态，那么其他线程能在那种状态下检查和修改它们。

suspend()方法容易发生死锁。调用 suspend()的时候，目标线程会停下来，但却仍然持有在这之前获得的锁定。此时，其他任何线程都不能访问锁定的资源，除非被"挂起"的线程恢复运行。对任何线程来说，如果它们想恢复目标线程，同时又试图使用任何一个锁定的资源，就 会造成死锁。所以不应该使用 suspend()，而应在自己的 Thread 类中置入一个标志，指出线程应该活动还是挂起。若标志指出线程应该挂起，便用 wait()命其进入等待状态。若标志指出线程应当恢复，则用一个 notify()重新启动线程。

【分析】

9、″==″和 equals 方法有什么区别？【中科软】

【参考答案】

==操作符专门用来比较两个变量的值是否相等，也就是用于比较变量所对应的内存中所存储的数值是否相同，要比较两个基本类型的数据或两个引用变量是否相等，只能用==操作符。

如果一个变量指向的数据是对象类型的，那么，这时候涉及了两块内存，对象本身占用一块内存（堆内存），变量也占用一块内存，例如 Objet obj = new Object();变量 obj 是一个内存，new Object()是另一个内存，此时，变量 obj 所对应的内存中存储的数值就是对象占用的那块内存的首地址。对于指向对象类型的变量，如果要比较两个变量是否指向同一个对象，即要看这两个变量所对应的内存中的数值是否相等，这时候就需要用==操作符进行比较。

equals 方法是用于比较两个独立对象的内容是否相同，就好比去比较两个人的长相是否相同，它比较的两个对象是独立的。例如，对于下面的代码：

```
String  a=new  String(″foo″);
String  b=new  String(″foo″);
```

两条 new 语句创建了两个对象，然后用 a,b 这两个变量分别指向了其中一个对象，这是两个不同的对象，它们的首地址是不同的，即 a 和 b 中存储的数值是不相同的，所以，表达式 a==b 将返回 false，而这两个对象中的内容是相同的，所以，表达式 a.equals(b)将返回 true。

在实际开发中，我们经常要比较传递进行来的字符串内容是否等，例如，String input = …;input.equals(“quit”),如果一个类没有自己定义 equals 方法，那么它将继承 Object 类的 equals 方法，Object 类的 equals 方法的实现代码如下：

```
boolean  equals(Object  o){
   return  this==o;
}
```

这说明，如果一个类没有自己定义 equals 方法，它默认的 equals 方法（从 Object 类继承的）就是使用==操作符，也是在比较两个变量指向的对象是否是同一对象，这时候使用 equals 和使用==会得到同样的结果，如果比较的是两个独立的对象则总返回 false。如果你编写的类希望能够比较该类创建的两个实例对象的内容是否相同，那么你必须覆盖 equals 方法，由你自己写代码来决定在什么情况即可认为两个对象的内容是相同的。

10、静态变量和实例变量的区别？

【参考答案】

在语法定义上的区别：静态变量前要加 static 关键字，而实例变量前则不加。

在程序运行时的区别：实例变量属于某个对象的属性，必须创建了实例对象，其中的实例变量才会被分配空间，才能使用这个实例变量。静态变量不属于某个实例对象，而是属于类，所以也称为类变量，只要程序加载了类的字节码，不用创建任何实例对象，静态变量就会被分配空间，静态变量就可以被使用了。总之，实例变量必须创建对象后才可以通过这个对象来使用，静态变量则可以直接使用类名来引用。

例如，对于下面的程序，无论创建多少个实例对象，永远都只分配了一个 staticVar 变量，并且每创建一个实例对象，这个 staticVar 就会加 1；但是，每创建一个实例对象，就会分配一个 instanceVar，即可能分配多个 instanceVar，并且每个 instanceVar 的值都只自加了 1 次。

```
public class VariantTest
{
    public static int staticVar = 0;
    public int instanceVar = 0;
    public VariantTest()
    {
        staticVar++;
        instanceVar++;
        System.out.println(“staticVar=” + staticVar + ”,instanceVar=” + instanceVar);
    }
}
```

备注：这个解答除了说清楚两者的区别外，最后还用一个具体的应用例子来说明两者的差异，体现了自己有很好的解说问题和设计案例的能力，思维敏捷，超过一般程序员，有写作能力！

11、构造器的名能不能和类的名字相同？

【参考答案】

构造器的名称必须与类名相同。

【分析】

构造器或构造函数（有些书这样叫）主要用来对类的成员变量进行初始化，当类创建实例时调用。

12、在一个主方法类可不可以调用一个非静态的方法？

【参考答案】

可以调用。因为 Java 的主方法（main）方法本身也是 static 类型方法，一个 static 类型方法，发起对另一个 static 方法的调用没有问题。

【分析】

静态方法可以调用其它的静态方法，但是不能调用非静态方法，这个好比 Java 中的类变量与实例变量的关系。类变量是被所有类成员共享，而实例变量只被该实例共享，

13、一个类中可不可以有 2 个公共的方法？

【参考答案】

可以。Java 中对公共方法的个数没有约束，但是对公共的类有约束，一个 Java 源文件中只能定义一个 public 类型的类。

14、GC 是什么，为什么要使用它？【阿斯拓】

【参考答案】

GC 是垃圾收集的意思（Gabage Collection）,内存处理是编程人员容易出现问题的地方，忘记或者错误的内存回收会导致程序或系统的不稳定甚至崩溃，Java 提供的 GC 功能可以自动监测对象是否超过作用域，从而达到自动回收内存的目的，Java 语言没有提供释放已分配内存的显示操作方法。

【分析】

15、说一下垃圾回收的原理，可以直接从内存中回收吗?

【参考答案】

Java 语言中一个显著的特点就是引入了垃圾回收机制，使 c++程序员最头疼的内存管理的问题迎刃而解，它使得 Java 程序员在编写程序的时候不再需要考虑内存管理。垃圾回收可以有效的防止内存泄露，有效的使用可以使用的内存。垃圾回收器通常是作为一个单独的低级别的线程运行,不可预知的情况下对内存堆中已经死亡的或者长时间没有使用的对象进行清除和回收，程序员不能实时的调用垃圾回收器对某个对象或所有对象进行垃圾回收,因为 Java 语言规范并不保证 GC 一定会执行。回收机制有分代复制垃圾回收和标记垃圾回收，增量垃圾回收。

【分析】

16、Java 的异常有哪几种，有什么区别？

【参考答案】

两大类，一般异常和运行时异常。一般异常，这些异常是在定义方法时声明抛出的,这些异常必需用 try catch 抛出，或 throws 处理，如果不处理，程序将编译失败。比如：IOException、FileNotFoundException、SQLException 等。

运行时异常是程序运行时可能报出的异常。可以用 try catch 抓取，也可以不做任何处理。例如：NullPointerException 异常就是一种比较常见的运行时异常。

【分析】

17、switch 语句能否作用在 byte 上，能否作用在 long 上，能否作用在 String 上?

【参考答案】

在 switch（表达式）中，括号表达式只能是一个整数表达式或者枚举常量（更大字体），整数表达式可以是 int 基本类型或 Integer 包装类型，由于，byte,short,char 都可以隐含转换为 int，所以，这些类型以及这些类型的包装类型也是可以的。显然，long 和 String 类型都不符合 switch 的语法规定，并且不能被隐式转换成 int 类型，所以，它们不能作用于 swtich 语句中。

18、Integer 与 int 的区别？

【参考答案】

int 是 java 提供的 8 种原始数据类型之一，另外 Java 为每个原始类型提供了封装类，Integer 是 java 为 int 提供的封装类。int 的默认值为 0，而 Integer 的默认值为 null，即 Integer 可以区分出未赋值和值为 0 的区别，int 则无法表达出未赋值的情况。

19、Java Reflection 是什么？【】

【参考答案】

JAVA 反射，Reflection 是 Java 程序开发语言的特征之一，它允许运行中的 Java 程序对自身进行检查，或者说"自审"，并能直接操作程序的内部属性。

【分析】

20、写几个 java.lang.Object 类中的方法名称。

【参考答案】

主要有 equals()、toString()、getClass()、hashCode()、clone()、 notify()、wait()、notify()方法。

【分析】

这种题能记多少个就说多少个，不一定要求你所有的都记住，但是要理解其中部分重要方法的含义和作用。

21、&和&&的区别？

【参考答案】

&和&&都可以用作逻辑与的运算符，表示逻辑与（and），当运算符两边的表达式的结果都为 true 时，整个运算结果才为 true，否则，只要有一方为 false，则结果为 false。

&&还具有短路的功能，即如果第一个表达式为 false，则不再计算第二个表达式。

&还可以用作位运算符，当&操作符两边的表达式不是 boolean 类型时，&表示按位与操作，我们通常使用 0x0f 来与一个整数进行&运算，来获取该整数的最低 4 个 bit 位。

【分析】

先说分别说两者的作用，再说出&&和&各自的不同之处。

22、数组有没有 length()这个方法，String 有没有 length()这个方法。

【参考答案】

数组没有 length()方法，但有 length 属性

String 有 length()方法。

【分析】

考查平时使用数组和字符串的一些细节，一般在使用

23、String s=new String(“xyz”)创建了几个对象

【参考答案】

2 个 string 对象,一个是=null 的 s，一个是=“xyz”的 string

两个或一个”xyz”对应一个对象，这个对象放在字符串常量缓冲区，常量”xyz”不管出现多少遍，都是缓冲区中的那一个。New String 每写一遍，就创建一个新的对象，它一句那个常量”xyz”对象的内容来创建出一个新 String 对象。如果以前就用过’xyz’，这句代表就不会创建”xyz”自己了，直接从缓冲区拿。

【分析】

24、能不能自己写个类，也叫 java.lang.String?

可以，但在应用的时候，需要用自己的类加载器去加载，否则，系统的类加载器永远只是去加载jre.jar包中的那个java.lang.String。由于在tomcat的web应用程序中，都是由webapp自己的类加载器先自己加载WEB-INF/classess目录中的类,然后才委托上级的类加载器加载，如果我们在tomcat的web应用程序中写一个java.lang.String，这时候Servlet程序

加载的就是我们自己写的java.lang.String，但是这么干就会出很多潜在的问题，原来所有用了java.lang.String类的都将出现问题。
虽然java提供了endorsed技术，可以覆盖jdk中的某些类，具体做法是….。但是，能够被覆盖的类是有限制范围，反正不包括java.lang这样的包中的类。

例如，运行下面的程序：

```
package java.lang;
public class String {
    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("string");
    }

}
```

报告的错误如下：

```
java.lang.NoSuchMethodError: main
Exception in thread "main"
```

这是因为加载了jre自带的java.lang.String，而该类中没有main方法。

25、你对面向对象思想的理解？

【参考答案】

面向对象编程（Object-Oriented Programming）简称 OOP 技术，是开发计算机应用程序的一种新方法、新思想。过去的面向过程编程中常常会导致所有的代码都包含在几个模块中，使程序难以阅读和维护，在做一些修改时常常牵一动百，使以后的开发和维护难以为继。而使用 OOP 技术，使用许多代码模块，每个模块都只提供特定的功能，它们是彼此独立的，可以增加代码重用的几率，更加有利于软件的开发、维护和升级。另外 OOP 的三大核心特性：继承、封装、多态的特性，使得在面对象编上能够设计出高内聚、低耦合的系统结构，使得系统更灵活、更容易扩展，而且成本较低，所以这一编程思想是目前一种应用最为普遍的软件设计思想。

【分析】

26、**最常见的 runtime exception 运行时异常？**

【参考答案】

ClassCastException（类型转换异常）、NumberFormatException（格式化异常）、ArrayIndexOutOfBoundsException（数组越界异常）、ArithmeticException（算术异常）、NullPointerException（空指针异常）等等

【分析】

这道题主要考查大家平时在项目开发过程中经常遇到的一些异常类型信息，通过这些异常来考查大家的项目经验与项目排错能力。

27、用 JDBC 来实现访问数据库记录可以采用下面的几个步骤：

【参考答案】

1、 通过驱动器管理器获取连接接口（Connection）。

2、 获得 Statement 或它的子类。

3、 指定 Statement 中的参数。

4、 通过 Statement 发送 SQL 语句。

5、 检查并处理返回的结果。

6、 关闭 Statement。

7、 关闭连接接

【分析】

28、Error 和 exception 的区别与联系？

【参考答案】

error 表示恢复不是不可能,但很困难的情况下的一种严重问题。比如说内存溢,网络故障等，不可能指望程序能处理的一类错误。

Exception 表示一种由程序设计或实现问题，像我们常说的异常处理，就是属于这类，一般程序可以捕获和处理这些异常。

【分析】

这道题的难点在 Error 很多时候由于我们无法重现这种 Error 导致很多同学甚至不知道 Error 到底是什么，所以很容易把题目中的两种错误划上等号。

29、String s = "Hello";s = s + " world!";这两行代码执行后，原始的 String 对象中的内容到底变了没有？

【参考答案】

没有。因为 String 被设计成不可变(immutable)类,所以它的所有对象都是不可变对象。在这段代码中，s 原先指向一个 String 对象，内容是 "Hello"，然后我们对 s 进行了+操作，那么 s 所指向的那个对象是否发生了改变呢？答案是没有。这时，s 不指向原来那个对象了，而指向了另一个 String 对象，内容为"Hello world!"，原来那个对象还存在于内存之中，只是 s 这个引用变量不再指向它了。

通过上面的说明，我们很容易导出另一个结论，如果经常对字符串进行各种各样的修改,或者说，不可预见的修改，那么使用 String 来代表字符串的话会引起很大的内存开销。因为 String 对象建立之后不能再改变，所以对于每一个不同的字符串，都需要一个 String 对象来表示。这时，应该考虑使用 StringBuffer 类，它允许修改，而不是每个不同的字符串都要生成一个新的对象。并且，这两种类的对象转换十分容易。

同时，我们还可以知道，如果要使用内容相同的字符串，不必每次都 new 一个 String。例如我们要在构造器中对一个名叫 s 的 String 引用变量进行初始化，把它设置为初始值，应当这样做：

```
public class Demo {
private String s;
...
public Demo {
s = "Initial Value";
}
...
}
```

而非

s = new String("Initial Value");
后者每次都会调用构造器，生成新对象，性能低下且内存开销大，并且没有意义，因为 String 对象不可改变，所以对于内容相同的字符串，只要一个 String 对象来表示就可以了。也就说，多次调用上面的构造器创建多个对象，他们的 String 类型属性 s 都指向同一个对象。
上面的结论还基于这样一个事实：对于字符串常量，如果内容相同，Java 认为它们代表同一个 String 对象。而用关键字 new 调用构造器，总是会创建一个新的对象，无论内容是否相同。
至于为什么要把 String 类设计成不可变类，是它的用途决定的。其实不只 String，很多 Java 标准类库中的类都是不可变的。在开发一个系统的时候，我们有时候也需要设计不可变类，来传递一组相关的值，这也是面向对象思想的体现。不可变类有一些优点，比如因为它的对象是只读的，所以多线程并发访问也不会有任何问题。当然也有一些缺点，比如每个不同的状态都要一个对象来代表，可能会造成性能上的问题。所以 Java 标准类库还提供了一个可变版本，即 StringBuffer。

30、Jdk1.5 的新特性？

【参考答案】

JDK1.5 的一个重要主题就是通过新增一些特性来简化开发，这些特性主要包括：泛型、ForEach 循环、自动装包/拆包、枚举、可变参数、静态导入这些。

【分析】

31、面向对象的特征有哪些方面？

【参考答案】

面向对象的编程语言有封装、继承 、多态等 3 个主要的特征。

◆ 封装

封装是保证软件部件具有优良的模块性的基础，封装的目标就是要实现软件部件的“高内聚、低耦合”，防止程序相互依赖性而带来的变动影响。面向对象的封装就是把描述一个对象的属性和行为的代码封装在一个“模块”中，也就是一个类中，属性用变量定义，行为用方法进行定义，方法可以直接访问同一个对象中的属性。

◆ 继承

在定义和实现一个类的时候，可以在一个已经存在的类的基础之上来进行，把这个已经存在的类所定义的内容作为自己的内容，并可以加入若干新的内容，或修改原来的方法使之更适合特殊的需要，这就是继承。继承是子类自动共享父类数据和方法的机制，这是类之间的一种关系，提高了软件的可重用性和可扩展性。

◆ 多态

多态是指程序中定义的引用变量所指向的具体类型和通过该引用变量发出的方法调用在编程时并不确定，而是在程序运行期间才确定，即一个引用变量倒底会指向哪个类的实例对象，该引用变量发出的方法调用到底是哪个类中实现的方法，必须在由程序运行期间才能决定。因为在程序运行时才确定具体的类，这样，不用修改源程序代码，就可以让引用变量绑定到各种不同的类实现上，从而导致该引用调用的具体方法随之改变，即不修改程序代码就可以改变程序运行时所绑定的具体代码，让程序可以选择多个运行状态，这就是多态性。多态性增强了软件的灵活性和扩展性。

32、JVM 工作原理？

运行 jvm 字符码的工作是由解释器来完成的。解释执行过程分三步进行：

代码的装入、代码的校验、和代码的执行。

装入代码的工作由“类装载器 class loader”完成。类装载器负责装入运行一个程序需要的所有代码，这也包括程序代码中的类所继承的类和被调用的类。当类装载器装入一个类时，该类被放在自己的名字空间中。除了通过符号引用自己名字空间以外的类，类之间没有其他办法可以影响其他类。在本台计算机的所有类都在同一地址空间中，而所有从外部引进的类，都有一个自己独立的名字空间。这使得本地类通过共享相同的名字空间获得较高的运行效率，同时又保证它们与从外部引进的类不会相互影响。当装入了运行程序需要的所有类后，解释器便可确定整个可执行程序的内存布局。解释器为符号引用与特定的地址空间建立对应关系及查询表。通过在这一阶段确定代码的内布局，java 很好地解决了由超类改变而使子类崩溃的问题，同时也防止了代码的非法访问。随后，被装入的代码由字节码校验器进行检查。校验器可以发现操作数栈益处、非法数据类型转化等多种错误。通过校验后，代码便开始执行了。

Java 字节码的执行有两种方式：

1）即时编译方式：解释器先将字节编译成机器码，然后再执行该机器码。

2）解释执行方式：解释器通过每次解释并执行一小段代码来完成 java 字节。

码程序的所有操作。

33、说说 Java 中的内存分配?

Java 把内存分成两种，一种叫做栈内存，一种叫做堆内存

在函数中定义的一些基本类型的变量和对象的引用变量都是在函数的栈内存中分配。当在一段代码块中定义一个变量时，java 就在栈中为这个变量分配内存空间，当超过变量的作用域后，java 会自动释放掉为该变量分配的内存空间，该内存空间可以立刻被另作它用。

堆内存用于存放由 new 创建的对象和数组。在堆中分配的内存，由 java 虚拟机自动垃圾回收器来管理。在堆中产生了一个数组或者对象后，还可以在栈中定义一个特殊的变量，这个变量的取值等于数组或者对象在堆内存中的首地址，在栈中的这个特殊的变量就变成了数组或者对象的引用变量，以后就可以在程序中使用栈内存中的引用变量来访问堆中的数组或者对象，引用变量相当于为数组或者对象起的一个别名，或者代号。

引用变量是普通变量，定义时在栈中分配内存，引用变量在程序运行到作用域外释放。而数组＆对象本身在堆中分配，即使程序运行到使用 new 产生数组和对象的语句所在地代码块之外，数组和对象本身占用的堆内存也不会被释放，数组和对象在没有引用变量指向它的时候，才变成垃圾，不能再被使用，但是仍然占着内存，在随后的一个不确定的时间被垃圾回收器释放掉。这个也是 java 比较占内存的主要原因。但是在写程序的时候，可以人为的控制。

34、final，finally，finalize 的区别。

【参考答案】

final 用于声明属性，方法和类，分别表示属性不可变，方法不可覆盖，类不可继承。内部类要访问局部变量，局部变量必须定义成 final 类型，例如，一段代码……

finally 是异常处理语句结构的一部分，表示总是执行。

finalize 是 Object 类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，可以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等。JVM 不保证此方法总被调用

35、Extends 和 Implement 的不同？

【参考答案】

extends 是继承父类，只要那个类不是声明为 final 或者那个类定义为 abstract 的就能继承，JAVA 中不支持多重继承，但是可以用接口来实现，这样就要用到 implements，继承只能继承一个类，但 implements 可以实现多个接口，用逗号分开就行了

比如 class A extends B implements C,D,E

36、抽象类是否可以没有抽象方法？为什么？

【参考答案】

可以在 java 中用 abstract 关键字来修饰一个类时，这个类叫做抽象类。

抽象类中不一定要包含 abstract 方法，但一个类中包含了 abstract 方法，则这个类必须声明为 abstract 类。

37、静态的多态和动态的多态的区别？

【参考答案】

静态的多态: 即为重载 ；方法名相同，参数个数或类型不相同。(overloading)

动态的多态: 即为重写；子类覆盖父类的方法，将子类的实例传与父类的引用调用的是子类的方法　实现接口的实例传与接口的引用调用的实现类的方法。

38、说出一些常用的类，包，接口，请各举 5 个?

【参考答案】

常用的类：String、StringBuffer、 Integer 、Vector、ArrayList、Hashtable等
常用的包：java.lang　 java.io　 java.util 、java.sql 。
常用的接口：集合中的List、Set、 Map接口；与Servlet API相关的Servlet接口、HttpServletRequest,HttpServletResponse,HttpSession接口等。

39、Collections 和 Collection 的区别【天晟科技】

【参考答案】

Collection 是个 java.util 下的接口，它是各种集合结构的父接口，定义了集合对象的基本操作方法。Collections 是个 java.util 下的工具类，它包含有各种有关集合操作的静态方法，主要是针对集合类的一个帮助类或者叫包装类，它提供一系列对各种集合的搜索，排序，线程安全化等操作方法。

40、Class.forName 的作用?为什么要用?

【参考答案】

按参数中指定的字符串形式的类名去搜索并加载相应的类,如果该类字节码已经被加载过，则返回代表该字节码的 Class 实例对象，否则，按类加载器的委托机制去搜索和加载该类，如果所有的类加载器都无法加载到该类，则抛出 ClassNotFoundException。加载完这个 Class 字节码后,接着就可以使用 Class 字节码的 newInstance 方法去创建该类的实例对象了。有时候，我们程序中所有使用的具体类名在设计时（即开发时）无法确定，只有程序运行时才能确定，这时候就需要使用 Class.forName 去动态加载该类，这个类名通常是在配置文件中配置的，例如，spring 的 ioc 中每次依赖注入的具体类就是这样配置的，jdbc 的驱动类名通常也是通过配置文件来配置的,以便在产品交付使用后不用修改源程序就可以更换驱动类名。

41、Socket 如何获取本地 ip 地址？

【参考答案】

InetAddress 类提供的 API 来访问。InetAddress.getLocalAddress()

【分析】

42、接口是否可继承接口？抽象类是否可实现(implements)接口？抽象类是否可继承具体类【天威诚信面试题】

【参考答案】

接口可以继承接口。抽象类可以实现(implements)接口，抽象类是否可继承具体类。抽象类中可以有静态的 main 方法。

备注：只要明白了接口和抽象类的本质和作用，这些问题都很好回答，你想想，如果你是 java 语言的设计者，你是否会提供这样的支持，如果不提供的话，有什么理由吗？如果你没有道理不提供，那答案就是肯定的了。

只有记住抽象类与普通类的唯一区别就是不能创建实例对象和允许有 abstract 方法。

43、用最有效率的方法算出 2 乘以 8 等於几?

【参考答案】

2 << 3

【分析】

因为将一个数左移 n 位，就相当于乘以了 2 的 n 次方，那么，一个数乘以 8 只要将其左移 3 位即可，而位运算 cpu 直接支持的，效率最高，所以 2 乘以 8 等於几的最效率的方法是 2 << 3。

44、char 型变量中能不能存贮一个中文汉字?为什么?

【参考答案】

char 型变量是用来存储 Unicode 编码的字符的，unicode 编码字符集中包含了汉字，所以，char 型变量中当然可以存储汉字啦。不过，如果某个特殊的汉字没有被包含在 unicode 编码字符集中，那么，这个 char 型变量中就不能存储这个特殊汉字。补充说明：unicode 编码占用两个字节，所以，char 类型的变量也是占用两个字节。

45、写 clone()方法时，通常都有一行代码，是什么？

【参考答案】

clone 有缺省行为，super.clone();因为首先要把父类中的成员复制到位，然后才是复制自己的成员。

46、说说常用集合类有哪些？有哪些方法？

【参考答案】

通常我们使用的集合类都大多是由 List、Set、Map 这三类接口派生出来的类，例如：ArrayList、Vector、LinkedList、Stack、TreeSet、Hashtable、HashMap 等

集合类的大部分方法都是由 Collection 接口定义的，主要包括有：

add(E e)、remove(Object e)、addAll(),remove()、**contains(Object obj)**、clear()等

47、请说出作用域 public，private，protected，以及不写时的区别?【天威诚信面试题】

【参考答案】

这四个作用域的可见范围如下表所示。

说明：如果在修饰的元素上面没有写任何访问修饰符，则表示 friendly。

作用域	同一类	同一 package	子孙类	其他 package
public	√	√	√	√
protected	√	√	√	×
friendly	√	√	×	×
private	√	×	×	×

备注：只要记住了有 4 种访问权限，4 个访问范围，然后将全选和范围在水平和垂直方向上分别按排从小到大或从大到小的顺序排列，就很容易画出上面的图了。

48、构造器 Constructor 是否可被 override? 【亿阳通讯面试题】

【参考答案】

构造器 Constructor 不能被继承，因此不能重写 Override，但可以被重载 Overload。

49、是否可以从一个 static 方法内部发出对非 static 方法的调用？ 【世承软件面试题】

【参考答案】

不可以。因为非 static 方法是要与对象关联在一起的，必须创建一个对象后，才可以在该对象上进行方法调用，而 static 方法调用时不需要创建对象，可以直接调用。

50、Math.round(11.5)等於多少? Math.round(-11.5)等於多少?【雾隐美地传媒】

【参考答案】

Math 类中提供了三个与取整有关的方法：ceil、floor、round，这些方法的作用与它们的英文名称的含义相对应，例如，ceil 的英文意义是天花板，该方法就表示向上取整，所以，Math.ceil(11.3)的结果为 12,Math.ceil(-11.3)的结果是-11；floor 的英文意义是地板，该方法就表示向下取整，所以，Math.floor(11.6)的结果为 11,Math.floor(-11.6)的结果是-12；最难掌握的是 round 方法，它表示“四舍五入”，算法为 Math.floor(x+0.5)，即将原来的数字加上 0.5 后再向下取整，所以，Math.round(11.5)的结果为 12，Math.round(-11.5)的结果为-11。

51、abstract class（抽象类）和 interface（接口）有什么区别? 【百度应用中心面试题】

【参考答案】

含有 abstract 修饰符的 class 即为抽象类，abstract 类不能创建的实例对象。含有 abstract 方法的类必须定义为 abstract class，abstract class 类中的方法不必是抽象的。abstract class 类中定义抽象方法必须在具体(Concrete)子类中实现，所以，不能有抽象构造方法或抽象静态方法。如果的子类没有实现抽象父类中的所有抽象方法，那么子类也必须定义为 abstract 类型。

接口（interface）可以说成是抽象类的一种特例，接口中的所有方法都必须是抽象的。接口中的方法定义默认为 public abstract 类型，接口中的成员变量类型默认为 public static final。

下面比较一下两者的语法区别：

1.抽象类可以有构造方法，接口中不能有构造方法。

2.抽象类中可以有普通成员变量，接口中没有普通成员变量

3.抽象类中可以包含非抽象的普通方法，接口中的所有方法必须都是抽象的，不能有非抽象的普通方法。

4. 抽象类中的抽象方法的访问类型可以是 public，protected 和（默认类型,虽然 eclipse 下不报错，但应该也不行），但接口中的抽象方法只能是 public 类型的，并且默认即为 public abstract 类型。

5. 抽象类中可以包含静态方法，接口中不能包含静态方法

6. 抽象类和接口中都可以包含静态成员变量，抽象类中的静态成员变量的访问类型可以任意，但接口中定义的变量只能是 public static final 类型，并且默认即为 public static final 类型。

7. 一个类可以实现多个接口，但只能继承一个抽象类。

下面接着再说说两者在应用上的区别：

【分析】

这道题的思路是先从总体解释抽象类和接口的基本概念，然后再比较两者的语法细节，最后再说两者的应用区别。比较两者语法细节区别的条理是：先从一个类中的构造方法、普通成员变量和方法（包括抽象方法），静态变量和方法，继承性等方面来回答。

52、Collection 框架中实现比较要实现什么接口?

【参考答案】

Comparable、Comparator 接口

53、是否可以继承 String 类?

【参考答案】

String 类是 final 类故不可以继承。

54、String 和 StringBuffer 的区别

【参考答案】

JAVA 平台提供了两个类：String 和 StringBuffer，它们可以储存和操作字符串，即包含多个字符的字符数据。String 类表示内容不可改变的字符串。而 StringBuffer 类表示内容可以被修改的字符串。当你知道字符数据要改变的时候你就可以使用 StringBuffer。典型地，你可以使用 StringBuffers 来动态构造字符数据。另外，String 实现了 equals 方法，new String("abc").equals(new String("abc")的结果为 true,而 StringBuffer 没有实现 equals 方法，所以，new StringBuffer("abc").equals(new StringBuffer("abc")的结果为 false。

String 覆盖了 equals 方法和 hashCode 方法，而 StringBuffer 没有覆盖 equals 方法和 hashCode 方法，所以，将 StringBuffer 对象存储进 Java 集合类中时会出现问题。

55、StringBuffer 与 StringBuilder 的区别

【参考答案】

StringBuffer 和 StringBuilder 类都表示内容可以被修改的字符串，StringBuilder 是线程不安全的，运行效率高，如果一个字符串变量是在方法里面定义，这种情况只可能有一个线程访问它，不存在不安全的因素了，则用 StringBuilder。如果要在类里面定义成员变量，并且这个类的实例对象会在多线程环境下使用，那么最好用 StringBuffer。

56、try {}里有一个 return 语句，那么紧跟在这个 try 后的 finally {}里的 code 会不会被执行，什么时候被执行，在 return 前还是后? 【杭州天眼科技】

【参考答案】

答案是在 return 之前。

【分析】

程序代码的运行结果：

```
public  class Test {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println(new Test().test());;
    }
    static int test()
    {
        int x = 1;
        try
        {
            return x;
        }
        finally
        {
            ++x;
        }
    }

   }
```

```
   ---------执行结果 ---------
   1
```

运行结果是 1，为什么呢？主函数调用子函数并得到结果的过程，好比主函数准备一个空罐子，当子函数要返回结果时，先把结果放在罐子里，然后再将程序逻辑返回到主函数。所谓返回，就是子函数说，我不运行了，你主函数继续运行吧，这没什么结果可言，结果是在说这话之前放进罐子里的。

下面的程序代码输出的结果是多少？

```
public class  smallT
{
    public static void  main(String args[])
    {
        smallT t  = new  smallT();
        int  b  =  t.get();
        System.out.println(b);
    }

    public int  get()
    {
        try
        {
```

```
            return 1 ;
        }
        finally
        {
            return 2 ;
        }
    }
}
```

返回的结果是 2。
我可以通过下面一个例子程序来帮助我解释这个答案，从下面例子的运行结果中可以发现，try 中的 return 语句调用的函数先于 finally 中调用的函数执行，也就是说 return 语句先执行，finally 语句后执行，所以，返回的结果是 2。Return 并不是让函数马上返回，而是 return 语句执行后，将把返回结果放置进函数栈中，此时函数并不是马上返回，它要执行 finally 语句后才真正开始返回。
在讲解答案时可以用下面的程序来帮助分析：

```
public  class Test {
    /**
     * @param args add by zxx ,Dec 9, 2008
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println(new Test().test());;
    }

    int test()
    {
        try
        {
            return func1();
        }
        finally
        {
            return func2();
        }
    }

    int func1()
    {
        System.out.println("func1");
        return 1;
    }
    int func2()
    {
```

```
        System.out.println("func2");
        return 2;
    }
}
----------执行结果----------------
func1
func2
2
```

结论：finally 中的代码比 return 和 break 语句后执行。

57、Java 中的异常处理机制的简单原理和应用。

【参考答案】

异常是指 java 程序运行时（非编译）所发生的非正常情况或错误，与现实生活中的事件很相似，现实生活中的事件可以包含事件发生的时间、地点、人物、情节等信息，可以用一个对象来表示，Java 使用面向对象的方式来处理异常，它把程序中发生的每个异常也都分别封装到一个对象来表示的，该对象中包含有异常的信息。

Java 对异常进行了分类，不同类型的异常分别用不同的 Java 类表示，所有异常的根类为 java.lang.Throwable，Throwable 下面又派生了两个子类：Error 和 Exception，Error 表示应用程序本身无法克服和恢复的一种严重问题，程序只有死的份了，例如，说内存溢出和线程死锁等系统问题。Exception 表示程序还能够克服和恢复的问题，其中又分为系统异常和普通异常，系统异常是软件本身缺陷所导致的问题，也就是软件开发人员考虑不周所导致的问题，软件使用者无法克服和恢复这种问题，但在这种问题下还可以让软件系统继续运行或者让软件死掉，例如，数组脚本越界（ArrayIndexOutOfBoundsException），空指针异常（NullPointerException）、类转换异常（ClassCastException）；普通异常是运行环境的变化或异常所导致的问题，是用户能够克服的问题，例如，网络断线，硬盘空间不够，发生这样的异常后，程序不应该死掉。

java 为系统异常和普通异常提供了不同的解决方案，编译器强制普通异常必须 try..catch 处理或用 throws 声明继续抛给上层调用方法处理，所以普通异常也称为 checked 异常，而系统异常可以处理也可以不处理，所以，编译器不强制用 try..catch 处理或用 throws 声明，所以系统异常也称为 unchecked 异常。

58、多线程有几种实现方法?同步有几种实现方法?

【参考答案】

多线程有两种实现方法，分别是继承 Thread 类与实现 Runnable 接口。

同步的实现方面有两种，分别是 synchronized,wait 与 notify 。

a. wait():使一个线程处于等待状态，并且释放所持有的对象的 lock。
b. sleep():使一个正在运行的线程处于睡眠状态，是一个静态方法，调用此方法要捕捉 InterruptedException 异常。
c. notify():唤醒一个处于等待状态的线程，注意的是在调用此方法的时候，并不能确切的唤醒某一个等待状态的线程，而是由 JVM 确定唤醒哪个线程，而且不是按优先级。
d. allnotity():唤醒所有处入等待状态的线程，注意并不是给所有唤醒线程一个对象的锁，而是让它们竞争。

59、启动一个线程是用 run()还是 start()？

【参考答案】

启动一个线程是调用 start()方法，使线程就绪状态，以后可以被调度为运行状态，一个线程必须关联一些具体的执行代码，run()方法是该线程所关联的执行代码。

60、内部类可以引用外部类的成员吗？有没有什么限制？

【参考答案】

完全可以。如果不是静态内部类，那没有什么限制！

如果你把静态嵌套类当作内部类的一种特例，那在这种情况下不可以访问外部类的普通成员变量，而只能访问外部类中的静态成员。

61、List 和 Map 区别?【软件动力】

【参考答案】

一个是存储单列数据的集合，另一个是存储键和值这样的双列数据的集合，List 中存储的数据是有顺序，并且允许重复；Map 中存储的数据是没有顺序的，其键是不能重复的，它的值是可以有重复的。

62、ArrayList 和 Vector 的区别【博炎科技】

【参考答案】

这两个类都实现了 List 接口(List 接口继承了 Collection 接口)，他们都是有序集合，即存储在这两个集合中的元素的位置都是有顺序的，相当于一种动态的数组，我们以后可以按位置索引号取出某个元素，并且其中的数据是允许重复的。

接着说 ArrayList 与 Vector 的区别，这主要包括两个方面：

1、同步性：

Vector 是线程安全的，也就是说是它的方法之间是线程同步的，而 ArrayList 是线程序不安全的，它的方法之间是线程不同步的。如果只有一个线程会访问到集合，那最好是使用 ArrayList，因为它不考虑线程安全，效率会高些；如果有多个线程会访问到集合，那最好是使用 Vector，因为不需要我们自己再去考虑和编写线程安全的代码。

备注：对于 Vector&ArrayList、Hashtable&HashMap，要记住线程安全的问题，记住 Vector 与 Hashtable 是旧的，是 java 一诞生就提供了的，它们是线程安全的，ArrayList 与 HashMap 是 java2 时才提供的，它们是线程不安全的。

2、数据增长：

ArrayList 与 Vector 都有一个初始的容量大小，当存储进它们里面的元素的个数超过了容量时，就需要增加 ArrayList 与 Vector 的存储空间，每次要增加存储空间时，不是只增加一个存储单元，而是增加多个存储单元，每次增加的存储单元的个数在内存空间利用与程序效率之间要取得一定的平衡。Vector 默认增长为原来两倍，而 ArrayList 的增长为原来的 1.5 倍。ArrayList 与 Vector 都可以设置初始的空间大小，Vector 还可以设置增长的空间大小，而 ArrayList 没有提供设置增长空间的方法。

63、heap 和 stack 有什么区别。

【参考答案】

Java 的内存分为两类，一类是栈内存，一类是堆内存。栈内存是指程序进入一个方法时，会为这个方法单独分配一块私属存储空间，用于存储这个方法内部的局部变量，当这个方法结束时，分配给这个方法的栈会释放，这个栈中的变量也将随之释放。

堆是与栈作用不同的内存，一般用于存放不放在当前方法栈中的那些数据，例如，使用

new 创建的对象都放在堆里，所以，它不会随方法的结束而消失。方法中的局部变量使用 final 修饰后，放在堆中，而不是栈中。

64、Java 类实现序列化的方法（二种）？如在 collection 框架中实现排序，要实现什么样的接口

【参考答案】

java.io.Serializable 接口或实现 Externalizable 接口。

Collection 框架中实现比较要实现 Comparable 接口或 Comparator 接口，并实现比较方法

65、JAVA 实现向数据库添加一列。

【参考答案】

```
Connection con = null;
ResultSet rs = null;
Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");
   String url="jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=db_name";
   Connection con = DriverManager.getConnection(url,"","");
StateManager  sm =con.createStateMent();
String sql = " alter table student  add  age  int; ";
rs = sm.excute(sql);
```

66、什么是 Java 序列化，如何实现 java 序列化？或者请解释 Serializable 接口的作用。

【东软国际】

【参考答案】

序列化就是一种用来处理对象流的机制，所谓对象流也就是将对象的内容进行流化。可以对流化后的对象进行读写操作，也可将流化后的对象传输于网络之间。序列化是为了解决在对对象流进行读写操作时所引发的问题。

序列化的实现：将需要被序列化的类实现 Serializable 接口，该接口没有需要实现的方法，implements Serializable 只是为了标注该对象是可被序列化的，然后使用一个输出流（如：FileOutputStream）来构造一个 ObjectOutputStream（对象流）对象，使用 ObjectOutputStream 对象的 writeObject(Object obj)方法就可以将参数为 obj 的对象写出，那么在另一端，通过 ObjectInputStream 对象的 readObject(Object obj)获取到字节流数据后，要将字节流转换成原对象，这叫反序列化，以便将数据存储在文件中或在网络传输。

Serializable 接口描述启用其序列化功能，未实现此接口的类将无法使其任何状态序列化或反序列化。Serializable 接口没有方法或字段，仅用于标识可序列化的语义，标识实现了该接口的对象属性可被序列化。

67、Java 中有几种类型的流？JDK 为每种类型的流提供了一些抽象类以供继承，请说出他们分别是哪些类？

【参考答案】

字节流，字符流两种类型流。字节流继承于 InputStream、 OutputStream；字符流继承于 Reader、Writer。 其它与 IO 操作相关的类都是派生至上述 4 个抽象类，如字节相关的:FileInputStream、FileOutputStream 类;字符相关的:BufferedReader、BufferedWriter

类

68、用 JDBC 如何调用存储过程

【参考答案】

通过JDBC组件中的CallableStatement接口实现调用存储过程。

核心代码如下：

```
Class.forName("com.mysql.jdbc.Driver");
Connection conn=
DriverManager.getConnection("jdbc:mysql:///test","root","root");
CallableStatement cstmt = cn.prepareCall("{call
insert_Student(?,?,?)}");
cstmt.registerOutParameter(3,Types.INTEGER);
cstmt.setString(1, "wangwu");
cstmt.setInt(2, 25);
cstmt.execute();
```

69、JAVA 事件有哪些模式？

【参考答案】

1、事件直接驱动模式。它的特点是直接而且快，是必须经常使用的，主要适合于迅速处理 前台的命令，通常就是我们说的 command（命令）模式。。2.监控式事件模式。主要借助第三者来监控和触发事件，就是通常我们说的观察者模式。特点是： 有一个观察者置身事外在定期独立运行着，我们将我们要监听的事件向这个观察者注册，这样观察者就 代替我们来监听这个事件，应用客户端通过观察者来获得事件状况。

【分析】

70、JVM 加载 class 文件原理？

【参考答案】

所谓装载就是寻找一个类或是一个接口的二进制形式并用该二进制形式来构造代表这个类或是这个接口的 class 对象的过程.

在 Java 中，类装载器把一个类装入 Java 虚拟机中，要经过三个步骤来完成：装载、链接和初始化,其中链接又可以分成校验、准备、解析

装载：查找和导入类或接口的二进制数据；

链接：执行下面的校验、准备和解析步骤，其中解析步骤是可以选择的；

校验：检查导入类或接口的二进制数据的正确性；

准备：给类的静态变量分配并初始化存储空间；

解析：将符号引用转成直接引用；

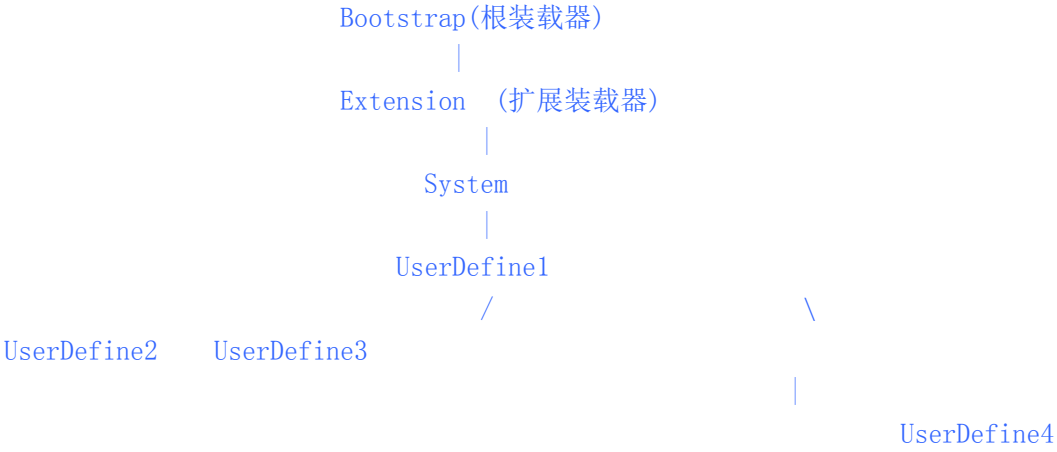
初始化：激活类的静态变量的初始化 Java 代码和静态 Java 代码块

JVM 中类的装载是由 ClassLoader 和它的子类来实现的,Java ClassLoader 是一个重要的 Java 运行时系统组件。它负责在运行时查找和装入类文件的类

一个 Java 应用程序使用两种类型的类装载器：根装载器(bootstrap)和用户定义的装载器(user-defined)。

根装载器以某种默认的方式将类装入，包括那些 Java API 的类。在运行期间一个 Java 程序能安装用户自己定义的类装载器。根装载器是虚拟机固有的一部分，而用户定义的类装载器则不是，它是用 Java 语言写的，被编译成 class 文件之后然后再被装入到虚拟机，并像

其它的任何对象一样可以被实例化。 Java 类装载器的体系结构如下所示：



Java 的类装载模型是一种代理(delegation)模型。当 JVM 要求类装载器 CL(ClassLoader)装载一个类时,CL 首先将这个类装载请求转发给他的父装载器。只有当父装载器没有装载并无法装载这个类时,CL 才获得装载这个类的机会。这样, 所有类装载器的代理关系构成了一种树状的关系。树的根是类的根装载器(bootstrap ClassLoader) , 在 JVM 中它以"null"表示。除根装载器以外的类装载器有且仅有一个父装载器。在创建一个装载器时, 如果没有显式地给出父装载器, 那么 JVM 将默认系统装载器为其父装载器

下面针对各种类装载器分别进行详细的说明：

根(Bootstrap) 装载器:该装载器没有父装载器，它是 JVM 实现的一部分，从 sun.boot.class.path 装载运行时库的核心代码。

扩展(Extension) 装载器:继承的父装载器为根装载器,不像根装载器可能与运行时的操作系统有关，这个类装载器是用纯 Java 代码实现的，它从 java.ext.dirs (扩展目录)中装载代码。

系统(System or Application) 装载器:装载器为扩展装载器，我们都知道在安装 JDK 的时候要设置环境变量(CLASSPATH)，这个类装载器就是从 java.class.path(CLASSPATH 环境变量)中装载代码的，它也是用纯 Java 代码实现的，同时还是用户自定义类装载器的缺省父装载器。

小应用程序(Applet) 装载器: 装载器为系统装载器，它从用户指定的网络上的特定目录装载小应用程序代码。

71、SOCKET 中有几中连接方式，各有什么区别？

【参考答案】

Sockets 有两种主要的操作方式:面向连接(TCP/IP)的和无连接(UDP)的。无连接的操作使用数据报协议，无连接的操作是快速的和高效的,但是数据安全性不佳. 面向连接的操作使用 TCP 协议.面向连接的操作比无连接的操作效率更低,但是数据的安全性更高

【分析】

72、抽象类能否被实例化 ？抽象类的作用是什么？

【参考答案】

抽象类一般不能被实例化；抽象类通常不是由程序员定义的，而是由项目经理或模块设计人 设计抽象类的原因通常是为了规范方法名 抽象类必须要继承，不然没法用，作为模块设计者，可以把让底层程序员直接用得方法直接调用，而一些需要让程序员覆盖后自己做得方法则定义称抽象方法

【分析】

73、Linkedlist、Arraylist 内部是如何实现的（更深入的问了 LinkedList 与 ArrayList 的区别） 【天威诚信面试题】

【参考答案】

ArrayList 的内部实现是基于内部数组 Object[]，它更像是对数组实现的一种封装，所以在向 ArrayList 的前面或中间插入数据时，必须将其后的所有数据相应的后移，这样必然要花费较多时间。

而 LinkedList 的内部实现是基于一组双向链表实现的存储特性，所以提供了链表一样访问的 API 接口，它们在性能上有很大的差别。当你访问 Linkedlist 链表中的某个元素时，就必须从链表的一端开始沿着连接方向一个一个元素地去查找，直到找到所需的元素为止，所以，当你的操作是在一列数据的前面或中间添加或删除数据，并且按照顺序访问其中的元素时，就应该使用 LinkedList 了。

而当你的操作是在一列数据的后面添加数据而不是在前面或中间，并且需要随机地访问其中的元素时，使用 ArrayList 会提供比较好的性能。

【分析】

74、Hashtable 的原理【北辰网络】

【参考答案】

通过节点的关键码确定节点的存储位置,即给定节点的关键码 k,通过一定的函数关系 H(散列函数),得到函数值 H(k),将此值解释为该节点的存储地址

75、JDBC 中的 PreparedStatement 相比 Statement 的好处？

【参考答案】

预编译语句 java.sql.PreparedStatement ，扩展自 Statement,不但具有 Statement 的所有能力而且具有更强大的功能。不同的是，PreparedStatement 是在创建语句对象的同时给出要执行的 sql 语句。这样，sql 语句就会被系统进行预编译，执行的速度会有所增加，尤其是在执行大语句的时候，效果更加理想

76、sleep()和 wait()区别

【参考答案】

sleep() 方法：线程主动放弃 CPU，使得线程在指定的时间内进入阻塞状态，不能得到 CPU 时间，指定的时间一过，线程重新进入可执行状态。典型地，sleep() 被用在等待某个资源就绪的情形：测试发现条件不满足后，让线程阻塞一段时间后重新测试，直到条件满足为止。

wait( ) ：与 notify()配套使用，wait()使得线程进入阻塞状态，它有两种形式，一种允许指定以毫秒为单位的一段时间作为参数，另一种没有参数，当指定时间参数时对应的 notify() 被调用或者超出指定时间时线程重新进入可执行状态，后者则必须对应的 notify() 被调用

（**网上的答案**：sleep 是线程类（Thread）的方法，导致此线程暂停执行指定时间，给执行机会给其他线程，但是监控状态依然保持，到时后会自动恢复。调用 sleep 不会释放对象锁。wait 是 Object 类的方法，对此对象调用 wait 方法导致本线程放弃对象锁，进入等待此对象的等待锁定池，只有针对此对象发出 notify 方法（或 notifyAll）后本线程才进入对象锁定池准备获得对象锁进入运行状态。）

sleep 就是正在执行的线程主动让出 cpu，cpu 去执行其他线程，在 sleep 指定的时间过后，cpu 才会回到这个线程上继续往下执行，如果当前线程进入了同步锁，sleep 方法并不会释放锁，即使当前线程使用 sleep 方法让出了 cpu，但其他被同步锁挡住了的线程也无法得到执行。wait 是指在一个已经进入了同步锁的线程内，让自己暂时让出同步锁，以便其他正在等待此锁的线程可以得到同步锁并运行，只有其他线程调用了 notify 方法(notify 并不释放锁，只是告诉调用过 wait 方法的线程可以去参与获得锁的竞争了，但不是马上得到锁，因为锁还在别人手里，别人还没释放。如果 notify 方法后面的代码还有很多，需要这些代码执行完后才会释放锁，可以在 notfiy 方法后增加一个等待和一些代码，看看效果)，调用 wait 方法的线程就会解除 wait 状态和程序可以再次得到锁后继续向下运行。对于 wait 的讲解一定要配合例子代码来说明，才显得自己真明白。

```
package com.huawei.interview;
public class MultiThread {
    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        new Thread(new Thread1()).start();
        try {
            Thread.sleep(10);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        new Thread(new Thread2()).start();
    }


    private static class Thread1 implements Runnable
    {

        @Override
        public void run() {
            // TODO Auto-generated method stub
//由于这里的Thread1和下面的Thread2内部run方法要用同一对象作为监视器，我们这里不能用this，因为在Thread2里面的this和这个Thread1的this不是同一个对象。我们用MultiThread.class这个字节码对象，当前虚拟机里引用这个变量时，指向的都是同一个对象。
            synchronized (MultiThread.class) {

                System.out.println("enter thread1...");

                System.out.println("thread1 is waiting");
                try {
```

```
//释放锁有两种方式，第一种方式是程序自然离开监视器的范围，也就是离开了
synchronized关键字管辖的代码范围,另一种方式就是在synchronized关键字管辖的代码内
部调用监视器对象的wait方法。这里，使用wait方法释放锁。
                    MultiThread.class.wait();
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }

                System.out.println("thread1 is going on...");
                System.out.println("thread1 is being over!");
            }
        }
    }

    private static class Thread2 implements Runnable
    {

        @Override
        public void run() {
            // TODO Auto-generated method stub
            synchronized (MultiThread.class) {

                System.out.println("enter thread2...");

                System.out.println("thread2 notify other thread can release wait
status..");
//由于notify方法并不释放锁， 即使thread2调用下面的sleep方法休息了10毫秒，但
thread1仍然不会执行，因为thread2没有释放锁，所以Thread1无法得不到锁。
MultiThread.class.notify();
            System.out.println("thread2 is sleeping ten millisecond...");
                try {
                    Thread.sleep(10);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }

                System.out.println("thread2 is going on...");
                System.out.println("thread2 is being over!");

            }
        }
```

```
    }
})
```

77、概述反射和序列化

【参考答案】

Reflection:是 Java 被视为动态语言的一个关键性质。这个机制允许程序在运行时透过 Reflection APIs 取得任何一个已知名称的 class 的内部信息，包括其 modifiers（诸如 public, static 等等）、superclass(例如 Object)、实现之 interfaces(例如 Cloneable)，也包括 fields 和 methods 的所有信息，并可于运行时改变 fields 内容或唤起 methods。

序列化:就是一种用来处理对象流的机制，所谓对象流也就是将对象的内容进行流化。可以对流化后的对象进行读写操作，也可将流化后的对象传输于网络之间。序列化是为了解决在对对象流进行读写操作时的问题。

78、Java 中实现多态的机制是什么?

【参考答案】

重写，重载

方法的重写 Overriding 和重载 Overloading 是 Java 多态性的不同表现。

重写 Overriding 是父类与子类之间多态性的一种表现，重载 Overloading 是一个类中多态性的一种表现。如果在子类中定义某方法与其父类有相同的名称和参数，我们说该方法被重写(Overriding)。子类的对象使用这个方法时，将调用子类中的定义，对它而言，父类中的定义如同被“屏蔽”了。

果在一个类中定义了多个同名的方法，它们或有不同的参数个数或有不同的参数类型，则称为方法的重载(Overloading)。Overloaded 的方法是可以改变返回值的类型。

79、Overload 和 Override 的区别？Overloaded 的方法是否可以改变返回值的类型？【软通动力】

【参考答案】

Overload 是重载的意思，Override 是覆盖的意思，也就是重写。

重载 Overload 表示同一个类中可以有多个名称相同的方法，但这些方法的参数列表各不相同（即参数个数或类型不同）。

重写 Override 表示子类中的方法可以与父类中的某个方法的名称和参数完全相同，通过子类创建的实例对象调用这个方法时，将调用子类中的定义方法，这相当于把父类中定义的那个完全相同的方法给覆盖了，这也是面向对象编程的多态性的一种表现。子类覆盖父类的方法时，只能比父类抛出更少的异常，或者是抛出父类抛出的异常的子异常，因为子类可以解决父类的一些问题,不能比父类有更多的问题。子类方法的访问权限只能比父类的更大，不能更小。如果父类的方法是 private 类型，那么，子类则不存在覆盖的限制，相当于子类中增加了一个全新的方法。

是否可以改变返回值类型，在重载的定义中，与方法是什么类型返回值无关。

【分析】

override 可以翻译为覆盖，从字面就可以知道，它是覆盖了一个方法并且对其重写，以求达到不同的作用。对我们来说最熟悉的覆盖就是对接口方法的实现，在接口中一般只是对方法进行了声明，而我们在实现时，就需要实现接口声明的所有方法。除了这个典型的用法以外，我们在继承中也可能会在子类覆盖父类中的方法。在覆盖要注意以下的几点：

1、覆盖的方法的标志必须要和被覆盖的方法的标志完全匹配，才能达到覆盖的效果；

2、覆盖的方法的返回值必须和被覆盖的方法的返回一致；
3、覆盖的方法所抛出的异常必须和被覆盖方法的所抛出的异常一致，或者是其子类；
4、被覆盖的方法不能为 private，否则在其子类中只是新定义了一个方法，并没有对其进行覆盖。

overload 对我们来说可能比较熟悉，可以翻译为重载，它是指我们可以定义一些名称相同的方法，通过定义不同的输入参数来区分这些方法，然后再调用时，VM 就会根据不同的参数样式，来选择合适的方法执行。在使用重载要注意以下的几点：
1、在使用重载时只能通过不同的参数样式。例如，不同的参数类型，不同的参数个数，不同的参数顺序(当然,同一方法内的几个参数类型必须不一样,例如可以是 fun(int,float)，但是不能为 fun(int,int))；
2、不能通过访问权限、返回类型、抛出的异常进行重载；
3、方法的异常类型和数目不会对重载造成影响；
4、对于继承来说，如果某一方法在父类中是访问权限是 priavte，那么就不能在子类对其进行重载，如果定义的话，也只是定义了一个新方法，而不会达到重载的效果。

80、ClassLoader 如何加载 class 。

【参考答案】

jvm 里有多个类加载，每个类加载可以负责加载特定位置的类，例如，bootstrap 类加载负责加载 jre/lib/rt.jar 中的类， 我们平时用的 jdk 中的类都位于 rt.jar 中。extclassloader 负责加载 jar/lib/ext/*.jar 中的类，appclassloader 负责 classpath 指定的目录或 jar 中的类。除了 bootstrap 之外，其他的类加载器本身也都是 java 类，它们的父类是 ClassLoader。

81、ArrayList 如何实现插入的数据按自定义的方式有序存放

【参考答案】

实现 Comparable 比较接口，并实现 compareTo 方法。排序的方法，取决于 compareTo 方法中的比较定义的返回值,一般有 3 个返回值：1、-1、0 表示不同的比较结果。

程序示例：

```
class MyBean implements Comparable{
    public int compareTo(Object obj){
        if(! obj instanceof MyBean)
            throw new ClassCastException();
        MyBean other = (MyBean) obj;
        return age > other.age?1:age== other.age?0:-1;
    }
}
class MyTreeSet {
    private ArrayList  datas = new ArrayList();
    public void add(Object obj){
        for(int i=0;i<datas.size();i++){
            if(obj.compareTo(datas.get(i) != 1){
                datas.add(i,obj);
            }
        }
```

```
    }
}
```

82、hashCode 方法的作用？

【参考答案】

hashcode 这个方法是用来鉴定 2 个对象是否相等的。hashcode 方法一般用户不会去调用，比如在 hashmap 中，由于 key 是不可以重复的，他在判断 key 是不是重复的时候就判断了 hashcode 这个方法,而且也用到了 equals 方法。这里不可以重复是说 equals 和 hashcode 只要有一个不等就可以了！所以简单来讲，hashcode 相当于是一个对象的编码。我们一般在覆盖 equals 的同时也要覆盖 hashcode，让他们的逻辑一致。

83、abstract 的 method 是否可同时是 static,是否可同时是 native，是否可同时是 synchronized?

【参考答案】

abstract 的 method 不可以是 static 的,因为抽象的方法是要被子类实现的,而 static 与子类扯不上关系!

native 方法表示该方法要用另外一种依赖平台的编程语言实现的,不存在着被子类实现的问题，所以，它也不能是抽象的，不能与 abstract 混用。例如，FileOutputSteam 类要硬件打交道，底层的实现用的是操作系统相关的 api 实现，例如，在 windows 用 c 语言实现的，所以，查看 jdk 的源代码，可以发现 FileOutputStream 的 open 方法的定义如下：

```
private native void open(String name) throws FileNotFoundException;
```

如果我们要用 java 调用别人写的 c 语言函数，我们是无法直接调用的，我们需要按照 java 的要求写一个 c 语言的函数，又我们的这个 c 语言函数去调用别人的 c 语言函数。由于我们的 c 语言函数是按 java 的要求来写的，我们这个 c 语言函数就可以与 java 对接上，java 那边的对接方式就是定义出与我们这个 c 函数相对应的方法，java 中对应的方法不需要写具体的代码，但需要在前面声明 native。

关于 synchronized 与 abstract 合用的问题，我觉得也不行，因为在我几年的学习和开发中，从来没见到过这种情况，并且我觉得 synchronized 应该是作用在一个具体的方法上才有意义。而且，方法上的 synchronized 同步所使用的同步锁对象是 this，而抽象方法上无法确定 this 是什么。

84、Anonymous Inner Class (匿名内部类) 是否可以 extends(继承)其它类，是否可以 implements(实现)interface(接口)?

【参考答案】

可以继承其他类或实现其他接口。不仅是可以，而是必须!

85、JAVA 语言如何进行异常处理，关键字：throws,throw,try,catch,finally 分别代表什么意义？在 try 块中可以抛出异常吗？

【参考答案】

Java 使用面向对象的方式来处理异常，它把程序中发生的每个异常也都分别封装到一个对象来表示的，该对象中包含有异常的信息。而 throws\throw\try、catch、finally 就是 Java 中用来对异常进行处理的几个关键字，在 Java 编程中规容 Java 编译器强制普通异常必须 try..catch 处理或用 throws 声明继续抛给上层调用方法处理,一般异常必须要求被捕获和处理，而系统异常可以处理也可以不处理，所以编译器不强制用 try..catch 处理或用 throws、throw 声明异常。而 finally 一般与 try 或 try\catch 一起使用做为异常的最后

处理出口。

86、同步和异步有何异同，在什么情况下分别使用他们？举例说明。

【参考答案】

如果数据将在线程间共享。例如正在写的数据以后可能被另一个线程读到，或者正在读的数据可能已经被另一个线程写过了，那么这些数据就是共享数据，必须进行同步存取。当应用程序在对象上调用了一个需要花费很长时间来执行的方法，并且不希望让程序等待方法的返回时，就应该使用异步编程，在很多情况下采用异步途径往往更有效率。

87、当一个线程进入一个对象的一个 synchronized 方法后，其它线程是否可进入此对象的其它方法?

【参考答案】

分几种情况：

1. 其他方法前是否加了 synchronized 关键字，如果没加，则能。
2. 如果这个方法内部调用了 wait，则可以进入其他 synchronized 方法。
3. 如果其他个方法都加了 synchronized 关键字，并且内部没有调用 wait，则不能。
4. 如果其他方法是 static，它用的同步锁是当前类的字节码，与非静态的方法不能同步，因为非静态的方法用的是 this。

88、线程的基本概念、线程的基本状态以及状态之间的关系

【参考答案】

一个程序中可以有多条执行线索同时执行，一个线程就是程序中的一条执行线索，每个线程上都关联有要执行的代码，即可以有多段程序代码同时运行，每个程序至少都有一个线程，即 main 方法执行的那个线程。如果只是一个 cpu，它怎么能够同时执行多段程序呢？这是从宏观上来看的，cpu 一会执行 a 线索，一会执行 b 线索，切换时间很快，给人的感觉是 a,b 在同时执行，好比大家在同一个办公室上网，只有一条链接到外部网线，其实，这条网线一会为 a 传数据，一会为 b 传数据，由于切换时间很短暂，所以，大家感觉都在同时上网。

状态：就绪，运行，synchronize 阻塞，wait 和 sleep 挂起，结束。wait 必须在 synchronized 内部调用。

调用线程的 start 方法后线程进入就绪状态，线程调度系统将就绪状态的线程转为运行状态，遇到 synchronized 语句时，由运行状态转为阻塞，当 synchronized 获得锁后，由阻塞转为运行，在这种情况可以调用 wait 方法转为挂起状态，当线程关联的代码执行完后，线程变为结束状态。

89、**简述 synchronized 和 java.util.concurrent.locks.Lock 的异同** ？

【参考答案】

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。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。

90、HashMap 和 Hashtable 的区别？【北通网科】

【参考答案】

HashMap 是 Hashtable 的轻量级实现（非线程安全的实现），他们都实现 Map 接口，主要区别在于 HashMap 允许空（null）键值（key）,由于非线程安全，在只有一个线程访问的情况下，效率要高于 Hashtable。
HashMap 允许将 null 作为一个 entry 的 key 或者 value，而 Hashtable 不允许。
HashMap 把 Hashtable 的 contains 方法去掉了，改成 containsvalue 和 containsKey。因为 contains 方法容易让人引起误解。
Hashtable 继承自 Dictionary 类，而 HashMap 是 Java1.2 引进的 Map interface 的一个实现。
最大的不同是，Hashtable 的方法是 synchronized 的，而 HashMap 不是，在多个线程访问 Hashtable 时，不需要自己为它的方法实现同步，而 HashMap 就必须为之提供外同步。
Hashtable 和 HashMap 采用的 hash/rehash 算法都大概一样，所以性能不会有很大的差异。

91、List、Set、Map 是否继承自 Collection 接口?

【参考答案】

List、Set 是，Map 不是； Map 接口定义的是 Key-Value 存储的特性，与 List 和 Set 不同,Map 在存储对象时,先要定义这个对象的 key 的值,再存入与这个 key 相对应的 Object，Map 集合的取值时根据存入的 key（关键字）来获取与这个关键字对应的对象。

92、List、Map、Set 三个接口，存取元素时，各有什么特点？

【参考答案】

首先，List 与 Set 具有相似性，它们都是单列元素的集合，所以，它们有一个功共同的父接口 Collection 接口。Set 里面不允许有重复的元素，即不能有两个相等的对象 。

List 表示有先后顺序的集合， 当我们多次调用 add(Obj e)方法时，每次加入的对象就像火车站买票有排队顺序一样，按先来后到的顺序排序。

Map 与 List 和 Set 不同，它是双列的集合每次存储时，要存储一对 key/value，不能存储重复的 key，这个重复的规则也是按 equals 比较相等。取则可以根据 key 获得相应的 value，即 get(Object key)返回值为 key 所对应的 value。另外，也可以获得所有的 key 的结合。

【分析】

总结:List 以特定次序来持有元素，可有重复元素。Set 无法拥有重复元素,内部排序。Map 保存 key-value 值，value 可多值。上面是大致不同，另外上述 3 个只是接口，而具体实现类中，用法大同小异，只是实现的数据结构不同，例如 List 接口下的 LinkedList 主要实现了双链表的存储特点，Vector 是线程安全的集合类。

93、说出 ArrayList,Vector, LinkedList 的存储性能和特性。【大唐动力面试题】

【参考答案】

ArrayList 和 Vector 都是使用数组方式存储数据，此数组元素数大于实际存储的数据以便增加和插入元素，它们都允许直接按序号索引元素，但是插入元素要涉及数组元素移动等内存操作，所以索引数据快而插入数据慢，Vector 由于使用了 synchronized 方法（线程安全），通常性能上较 ArrayList 差，而 LinkedList 使用双向链表实现存储，按序号索引数据需要进行前向或后向遍历，但是插入数据时只需要记录本项的前后项即可，所以插入速度较快。

LinkedList 也是线程不安全的，LinkedList 提供了一些方法，使得 LinkedList 可以被当作堆栈和队列来使用。

94、如何去掉一个 Vector 集合中重复的元素

【参考答案】

```
Vector newVector = new Vector();
for (int i=0;i<vector.size();i++)
{
    Object obj = vector.get(i);
    if(!newVector.contains(obj);
        newVector.add(obj);
}
```

还有一种简单的方式，HashSet set = new HashSet(vector);

95、Set 里的元素是不能重复的，那么用什么方法来区分重复与否呢？是用==还是 equals()？它们有何区别?

【参考答案】

Set 里的元素是不能重复的，元素重复与否是使用 equals()方法进行判断的。

96、**两个对象值相同(x.equals(y) == true)，但却可有不同的 hash code，这句话对不对?**

【参考答案】

对。

如果对象要保存在 HashSet 或 HashMap 中，它们的 equals 相等，那么，它们的 hashcode 值就必须相等。

如果不是要保存在 HashSet 或 HashMap，则与 hashcode 没有什么关系了，这时候 hashcode 不等是可以的，例如 arrayList 存储的对象就不用实现 hashcode 方法。

97、字节流与字符流的区别

【参考答案】

要把一片二进制数据数据逐一输出到某个设备中，或者从某个设备中逐一读取一片二进制数据，不管输入输出设备是什么，我们要用统一的方式来完成这些操作，用一种抽象的方式进行描述，这个抽象描述方式起名为 IO 流，对应的抽象类为 OutputStream 和 InputStream ，不同的实现类就代表不同的输入和输出设备，它们都是针对字节进行操作的。

在应用中，经常要完全是字符的一段文本输出去或读进来，用字节流可以吗？计算机中的一切最终都是二进制的字节形式存在。对于“中国”这些字符，首先要得到其对应的字节，然后将字节写入到输出流。读取时，首先读到的是字节，可是我们要把它显示为字符，我们需要将字节转换成字符。由于这样的需求很广泛，人家专门提供了字符流的包装类。

底层设备永远只接受字节数据，有时候要写字符串到底层设备，需要将字符串转成字节再进行写入。字符流是字节流的包装，字符流则是直接接受字符串，它内部将串转成字节，再写入底层设备，这为我们向 IO 设别写入或读取字符串提供了一点点方便。

字符向字节转换时，要注意编码的问题，因为字符串转成字节数组，

其实是转成该字符的某种编码的字节形式，读取也是反之的道理。

讲解字节流与字符流关系的代码案例：

```
import java.io.BufferedReader;
import java.io.FileInputStream;
```

```
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.InputStreamReader;
import java.io.PrintWriter;
public class IOTest {
  public static void main(String[] args) throws Exception {
     String str = "中国人";
     /*FileOutputStream fos  = new FileOutputStream("1.txt");

     fos.write(str.getBytes("UTF-8"));
     fos.close();*/

     /*FileWriter fw = new FileWriter("1.txt");
     fw.write(str);
     fw.close();*/
     PrintWriter pw = new PrintWriter("1.txt","utf-8");
     pw.write(str);
     pw.close();

     /*FileReader fr = new FileReader("1.txt");
     char[] buf = new char[1024];
     int len = fr.read(buf);
     String myStr = new String(buf,0,len);
     System.out.println(myStr);*/
     /*FileInputStream fr = new FileInputStream("1.txt");
     byte[] buf = new byte[1024];
     int len = fr.read(buf);
     String myStr = new String(buf,0,len,"UTF-8");
     System.out.println(myStr);*/
     BufferedReader br = new BufferedReader(
             new InputStreamReader(
                 new FileInputStream("1.txt"),"UTF-8"
                 )
             );
     String myStr = br.readLine();
     br.close();
     System.out.println(myStr);
  }

}
```

98、java 里面的 io 跟 nio 有什么区别

【参考答案】

1、Java NIO 和 IO 之间第一个最大的区别是，IO 是面向流的，NIO 是面向缓冲区的。

2、Java IO 的各种流是阻塞的。而 Java NIO 的非阻塞模式，使一个线程从某通道发送请求读取数据，但是它仅能得到目前可用的数据，如果目前没有数据可用时，就什么都不会获取。而不是保持线程阻塞，所以直至数据变的可以读取之前，该线程可以继续做其他的事情。

3、选择器上，Java IO 无选择器，而 NIO 有选择器，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。

99、Java 中会存在内存泄漏吗，请简单描述。

【参考答案】

所谓内存泄露就是指一个不再被程序使用的对象或变量一直被占据在内存中。java 中有垃圾回收机制，它可以保证一对象不再被引用的时候，即对象变成了孤儿的时候，对象将自动被垃圾回收器从内存中清除掉。由于 Java 使用有向图的方式进行垃圾回收管理，可以消除引用循环的问题，例如有两个对象，相互引用，只要它们和根进程不可达的，那么 GC 也是可以回收它们的，例如下面的代码可以看到这种情况的内存回收：

```
package com.huawei.interview;
import java.io.IOException;
public class GarbageTest {
    /**
     * @param args
     * @throws IOException
     */
    public static void main(String[] args) throws IOException {
        // TODO Auto-generated method stub
        try {
            gcTest();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        System.out.println("has exited gcTest!");
        System.in.read();
        System.in.read();
        System.out.println("out begin gc!");
        for(int i=0;i<100;i++)
        {
            System.gc();
            System.in.read();
            System.in.read();
        }
    }

    private static void gcTest() throws IOException {
```

```
        System.in.read();
        System.in.read();
        Person p1 = new Person();
        System.in.read();
        System.in.read();
        Person p2 = new Person();
        p1.setMate(p2);
        p2.setMate(p1);
        System.out.println("before exit gctest!");
        System.in.read();
        System.in.read();
        System.gc();
        System.out.println("exit gctest!");
    }

    private static class Person
    {
        byte[] data = new byte[20000000];
        Person mate = null;
        public void setMate(Person other)
        {
            mate = other;
        }
    }
}
```

java 中的内存泄露的情况：长生命周期的对象持有短生命周期对象的引用就很可能发生内存泄露，尽管短生命周期对象已经不再需要，但是因为长生命周期对象持有它的引用而导致不能被回收，这就是 java 中内存泄露的发生场景，通俗地说，就是程序员可能创建了一个对象，以后一直不再使用这个对象，这个对象却一直被引用，即这个对象无用但是却无法被垃圾回收器回收的，这就是 java 中可能出现内存泄露的情况，例如，缓存系统，我们加载了一个对象放在缓存中(例如放在一个全局 map 对象中)，然后一直不再使用它，这个对象一直被缓存引用，但却不再被使用。

检查 java 中的内存泄露，一定要让程序将各种分支情况都完整执行到程序结束，然后看某个对象是否被使用过，如果没有，则才能判定这个对象属于内存泄露。

如果一个外部类的实例对象的方法返回了一个内部类的实例对象，这个内部类对象被长期引用了，即使那个外部类实例对象不再被使用，但由于内部类持久外部类的实例对象，这个外部类对象将不会被垃圾回收，这也会造成内存泄露。

[] 是对象已不可到达，而内存又没有回收，真正的内存黑洞。

而 Java 的泄漏，则是因为各种原因，对象对应用已经无用，但一直被持有，一直可到达。

总结原因无外乎几方面：

1. 被生命周期极长的集合类不当持有，号称是 Java 内存泄漏的首因。
   这些集合类的生命周期通常极长，而且是一个辅助管理性质的对象，在一个业务事务运行完后，如果没有将某个业务对象主动的从中清除的话，这个集合就会吃越来

越多内存，可以用 WeakReference，如 WeakHashMap，使得它持有的对象不增加对象的引用数。

2. Scope 定义不对，这个很简单了，方法的局部变量定义成类的变量，类的静态变量等。
3. 异常时没有加 finally{}来释放某些资源，JDBC 时代也是很普遍的事情。
4. 另外一些我了解不深的原因，如：Swing 里的 Listener 没有显式 remove；内部类持有外部对象的隐式引用；Finalizers 造成关联对象没有被及时清空等。

内存泄漏的检测

有不少工具辅助做这个事情的，如果手上一个工具也没有，可以用 JDK 自带的小工具：

- 看看谁占满了 Heap？
  用 JDK6 的 jmap 可以显示运行程序中对象的类型，个数与所占的大小
  先用 jps 找到进程号，然后 jmap -histo pid 显示或 jmap -dump:file=heap_file_name pid 导出 heap 文件
- 为什么这些对象仍然可以到达？
  用 jhat(Java Heap Analysis Tool) 分析刚才导出的 heap 文件。
  先 jhat heap_file_name，然后打开浏览器 http://localhost:7000/ 浏览。

100、Hashcode 和 Equals 的联系

【参考答案】

首先 equals()和 hashcode()这两个方法都是从 object 类中继承过来的，主要用来比较对象时进行调用。在 object 类中定义如下：

a)、如果两个对象相同，那么它们的 hashCode 值一定要相同；

b)、如果两个对象的 hashCode 相同，它们并不一定相同 上面说的对象相同指的是用 eqauls 方法比较。

101、Tread 和 Threadlocal 的作用及区别？

【参考答案】

答：threadlocal 是 线程局部变量（thread local variable），为每一个使用该线程的线程都提供一个变量值的副本，使每一个线程都可以独立地改变自己的副本，而不会和其他线程的副本产生冲突。

102、TCP 和 UDP 的区别？

【参考答案】

TCP/IP 的运输层有两个不同的协议：①用户数据报协议 UDP ②传输控制协议 TCP

二者最大区别:TCP 是面向连接的,而 UDP 是无连接的.区别大致如下:

1）UDP 传送的数据单位协议是 UDP 报文或用户数据报,TCP 传送的数据单位协议是 TCP 报文段。

2）UDP 发送数据之前不需要建立连接,因此减少了开销和发送之前的时延。TCP 提供面向连接的服务,不提供广播或多播服务。

3）对方的运输层在收到 UDP 报文后,不需要给出任何确认。TCP 则需要确认。

4）UDP 没有拥塞控制,因此网络出现的拥塞不会使源主机的发送速率降低,也不保证可靠交付,因此主机不需要维持具有许多参数的、复杂的连接状态表。TCP 要提供可靠的、面向连接的运输服务,因此不可避免地增加了许多的开销,这不仅使协议数据单元的首部增大很多,还要占用许多的处理机资源。

5）UDP 用户数据报只有 8 个字节的首部开销,比 TCP 的 20 个字节的首部要短。

103、启动一个线程用什么方法？【北京永中软件面试题】

【参考】

使用 Thread 类的 start()方法来启动一个线程，使线程进入就绪状态。如果自定义的类是 Thread 类的子类的话，可以直接使用 Start()来启，如果是实现的 Runnable 接口的话，还要将该类的实例作为参数传入到 Thread 对象中来启动。

104、作用域 public 等写不写的区别？【北京永中软件面试题】

【参考】

作用域不写将采用默认的作用域,默认作用域的访问权限是包的权限，也就是除本包中的所有类能访问，不同包只有子类能访问。

105、同步和异步有何异同

【参考答案】

同步(synchronized)和异步(asynchronized)是对于多线程(multi-threading)而言的

同步可防止并发 主要出于数据安全的考虑

如果数据将在线程间共享。例如正在写的数据以后可能被另一个线程读到，或者正在读的数据可能已经被另一个线程写过了，那么这些数据就是共享数据，必须进行同步存取。

异步允许并发

ajax 技术通常都是异步实现的，异步主要使用在当应用程序在对象上调用了一个需要花费很长时间来执行的方法，并且不希望让程序等待方法的返回时，就应该使用异步编程，在很多情况下采用异步途径往往更有效率。

106、Static 方法和 static 字段有什么用处？可以被重载吗？

【参考答案】

用 static 修饰的方法叫类方法，被所有实例共享；static 修饰的字段为类变量，被所有实例共享，在使用类变量时，一般会结合 final 一起使用定义类常量，不允许被其它的类实例修改。

可以被重载，重载只是参数类型、顺序和个数不同。

107、JDK6 的新特性？

【参考答案】

1、在 JDK6 中 AWT 新增加了两个类:Desktop 和 SystemTray。前者可以用来打开系统默认浏览器浏览指定的 URL,打开系统默认邮件客户端给指定的邮箱发邮件,用默认应用程序打开或编辑文件(比如,用记事本打开以 txt 为后缀名的文件),用系统默认的打印机打印文档;后者可以用来在系统托盘区创建一个托盘程序。

2、轻量级 Http Server。

JDK6 提供了一个简单的 Http Server API,据此我们可以构建自己的嵌入式 Http Server,它支持 Http 和 Https 协议,提供了 HTTP1.1 的部分实现，没有被实现的那部分可以通过扩展已有的 Http Server API 来实现,程序员必须自己实现 HttpHandler 接口。

3、嵌入式数据库 Derby 。Derby 并不是一个新的数据库产品,它是由 IBM 捐献给 Apache 的 DB 项目的一个纯 Java 数据库,JDK6.0 里面带的这个 Derby 的版本是 10.2.1.7,支持存储过程

和触发器;有两种运行模式,一种是作为嵌入式数据库,另一种是作为网络数据库,前者的数据库服务器和客户端都在同一个 JVM 里面运行。

4、使用 Compiler API。动态编译 Java 源文件，Compiler API 结合反射功能就可以实现动态的产生 Java 代码并编译执行这些代码，有点动态语言的特征。这个特性对于某些需要用到动态编译的应用程序相当有用， 比如 JSP Web Server，当我们手动修改 JSP 后，是不希望需要重启 Web Server 才可以看到效果的，这时候我们就可以用 Compiler API 来实现动态编译 JSP 文件，当然，现在的 JSP Web Server 也是支持 JSP 热部署的，现在的 JSP Web Server 通过在运行期间通过 Runtime.exec 或 ProcessBuilder 来调用 javac 来编译代码，这种方式需要我们产生另一个进程去做编译工作，不够优雅而且容易使代码依赖与特定的操作系统；Compiler API 通过一套易用的标准的 API 提供了更加丰富的方式去做动态编译,而且是跨平台的。

其它略.....像还有用 Console 开发控制台程序、JTable 的排序和过滤、插入式注解等。。。。

（二）Java web 部分

1. Servlet 和 jsp 页面过滤器 Filter 的作用及配置

【参考答案】

过滤器是一个驻留在服务器端的 Web 组件，它可以截取客户端和服务器端资源之间的请求与响应信息，并对这些信息进行过滤。作用：用户请求审查、用户数据转换、统一认证、输出数据进行压缩、对请求响应进行加密。

配置：一般在 web.xml 中配置，通过<filter>和<filter-mapping>元素来完成的。

2. JSP 内置对象作用，如何取 Cookie 的方法

【参考答案】

使用 request 对象的 getCookies()方法取到所有客户端 cookies 信息。

(1) HttpServletRequest 类的 request 对象

作用：代表请求对象，主要用于接受客户端通过 HTTP 协议连接传输到服务器端的数据。

(2) HttpServletResponse 类的 respone 对象

作用：代表响应对象，主要用于向客户端发送数据

(3) JspWriter 类的 out 对象

作用：主要用于向客户端输出数据;

Out 的基类是 JspWriter

(4) HttpSession 类的 session 对象

作用：主要用于来分别保存每个用户信息，与请求关联的会话;

会话状态维持是 Web 应用开发者必须面对的问题。

(5) ServletContext 类的 application 对象

作用：主要用于保存用户信息，代码片段的运行环境;

它是一个共享的内置对象，即一个容器中的多个用户共享一个 application 对象，故其保存的信息被所有用户所共享.

(6) PageContext 类的 pageContext 对象

作用：管理网页属性,为 JSP 页面包装页面的上下文，管理对属于 JSP 中特殊可见部分中已命名对象的访问，它的创建和初始化都是由容器来完成的。

(7) ServletConfig 类的 config 对象

作用：代码片段配置对象，表示 Servlet 的配置。

(8) Object 类的 Page（相当于 this）对象

作用：处理 JSP 网页，是 Object 类的一个实例，指的是 JSP 实现类的实例，即它也是 JSP 本身，只有在 JSP 页面范围之内才是合法的。

(9)Exception

作用：处理 JSP 文件执行时发生的错误和异常

3. 通过部署描述文件（web.xml）可以配置哪些功能？

【参考答案】

1、配置项目的欢迎页面。

2、配置 Servlet 访问 URL

3、配置 Web 容器、Servlet 的初始化参数

4、配置错误页面，可以通过异常编号进行错误页面跳转。

5、Servlet 加载优先级。

6、Web 容器监听器。
7、Web 请求过滤器。
8、设置会话的过期时间。

4. JSP 有哪些的动作？分别有什么作用？

【参考答案】

<jsp:include>：在页面被请求的时候引入一个文件。
<jsp:useBean>：寻找或者实例化一个 JavaBean。
<jsp:setProperty>：设置 JavaBean 的属性。
<jsp:getProperty>：输出某个 JavaBean 的属性。
<jsp:forward>：把请求转到一个新的页面。
<jsp:param>:在请求转发与 include 中进行页面传参。
<jsp:plugin>：根据浏览器类型为 Java 插件生成 OBJECT 或 EMBED 标记。

【分析】

主要说出常用的即可，分类罗列，一类与 JavaBean 相关的动作，<jsp:useBean >\<jsp:setProperty>\<jsp:getProperty>，另一类是转发包含的指令<jsp:include>\<jsp:forwrad>\<jsp:param>

5. JSP 与 SERVLET 区别

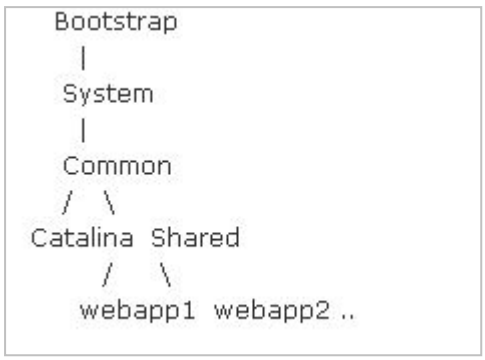
【参考答案】

JSP 在本质上就是 SERVLET,但是两者的创建方式不一样.Servlet 完全是 JAVA 程序代码构成，擅长于流程控制；而.JSP 由 HTML 代码和 JSP 标签构成，可以方便地编写动态网页.因此在实际应用中采用 Servlet 来控制业务流程，而采用 JSP 来生成动态网页，同时在 MVC 设计模式中 JSP 充当视图层，而 Servlet 位于控制层。另外 JSP 也是 Servlet 技术的扩展，本质上就是 Servlet，就是 JSP 的另一种简易体现形式，因为 JSP 编译后就是一个“类 servlet”，再经由 JVM 编译生成 Java 类文件来执行。

6. Tomcat 的 class 加载的优先顺序一览

【参考答案】

加载顺序图示如下：



第一步：加载 JVM 类库。一般是加载由虚拟机提供的基本的运行时类和系统扩展目录（$JAVA_HOME/jre/lib/ext)下的 JAR 包。
第二步:加载系统环境变量的类库。这个加载器用来加载 CLASSPATH 环境变量中指定的类。
第三步：加载 Tomcat 下面 common 文件夹下面的公共类库。
第四步：加载自己需要的 catalina 类库。
第五步：webapps 下面自己应用的类库，包括 webapp1、webapp1.....等。

7. BS 与 CS 的联系与区别 。【极地信息面试题】

【参考答案】

C/S 是 Client/Server 的缩写,表示客户端需要安装专用的客户端软件与服务器交互。B/S 是 Brower/Server 的缩写，客户机上只要安装一个浏览器（Browser），如 Internet Explorer。在这种结构下，用户界面完全通过 WWW 浏览器实现，一部分事务逻辑在前端实现，但是主要事务逻辑在服务器端实现。浏览器通过Web Server 同数据库进行数据交互。

C/S 与 B/S 区别：

1．硬件环境不同:

C/S 一般建立在专用的网络上，小范围里的网络环境，局域网之间再通过专门服务器提供连接和数据交换服务.

B/S 建立在广域网之上的，不必是专门的网络硬件环境,例与电话上网，租用设备.信息自己管理. 有比 C/S 更强的适应范围，一般只要有操作系统和浏览器就行

2．对安全要求不同

C/S 一般面向相对固定的用户群，对信息安全的控制能力很强. 一般高度机密的信息系统采用 C/S 结构适宜. 可以通过 B/S 发布部分可公开信息.

B/S 建立在广域网之上，对安全的控制能力相对弱，可能面向不可知的用户。

3．对程序架构不同

C/S 程序可以更加注重流程，可以对权限多层次校验，对系统运行速度可以较少考虑.

B/S 对安全以及访问速度的多重的考虑，建立在需要更加优化的基础之上. 比 C/S 有更高的要求 B/S 结构的程序架构是发展的趋势，从 MS 的.Net 系列的 BizTalk 2000 Exchange 2000 等，全面支持网络的构件搭建的系统. SUN 和 IBM 推的 JavaBean 构件技术等,使 B/S 更加成熟.

4．软件重用不同

C/S 程序可以不可避免的整体性考虑，构件的重用性不如在 B/S 要求下的构件的重用性好.

B/S 对的多重结构,要求构件相对独立的功能. 能够相对较好的重用.就入买来的餐桌可以再利用,而不是做在墙上的石头桌子

5．系统维护不同

C/S 程序由于整体性，必须整体考察，处理出现的问题以及系统升级. 升级难.可能是再做一个全新的系统

B/S 构件组成,方面构件个别的更换,实现系统的无缝升级. 系统维护开销减到最小.用户从网上自己下载安装就可以实现升级.

6．处理问题不同

C/S 程序可以处理用户面固定，并且在相同区域，安全要求高需求，与操作系统相关. 应该都是相同的系统

B/S 建立在广域网上，面向不同的用户群，分散地域，这是 C/S 无法作到的. 与操作系统平台关系最小.

7．用户接口不同

C/S 多是建立的 Window 平台上,表现方法有限,对程序员普遍要求较高

B/S 建立在浏览器上，有更加丰富和生动的表现方式与用户交流. 并且大部分难度减低,减低开发成本.

8．信息流不同

C/S 程序一般是典型的中央集权的机械式处理，交互性相对低

B/S 信息流向可变化，B-B B-C B-G 等信息、流向的变化，更像交易中心。

8. Servlet 与 CGI 的区别。【大唐动力面试题】

【参考答案】

Servlet 主要是运行在服务器端的一个组件，基于 Java 语法构建，延用 Java 的传统优势——可移植、稳健、易开发。CGI 不可移植，为某一特定平台编写的 CGI 应用只能运行于这一环境中。每一个 CGI 应用存在于一个由客户端请求激活的进程中，并且在请求被服务后被卸载。这种模式将引起很高的内存、CPU 开销，而且在同一进程中不能服务多个客户。

Servlet 对 CGI 的最主要优势在于一个 Servlet 被客户端发送的第一个请求激活，然后它将继续运行于后台，等待后续请求。每个请求将生成一个新的线程，而不是一个完整的进程，多个客户能够在同一个进程中同时得到服务，也就是说它是基于多线程模式。而 CGI 对每个请求都产生新的进程，服务完成后就销毁，所以效率上低于 servlet。

9. Servlet 中的 init（）方法什么时候被调用？

【参考答案】

当客户端第一次请求该 Servlet 时，由容器调用该 Servlet 类的 init()方法对该 Servlet 进行初始化，该初始始化方法只被调用一次。

10. 用什么方法使服务器关闭之后，session 所保存的信息不会丢失？

【参考答案】

使用 cookie 实现，服务器端将需要保存的信息，通过 Cookie 并写入客户端磁盘中，下次访问时，客户端浏览器携带写入的信息提交至服务器，这样可以使信息不会因服务器关闭而丢失。

【分析】

主要考查 Cookie 的相关知识，比如还有一道题问：如果登录某个网站后，想在本周内都不用重新登录，也是通过 Cookie 实现的

11. Forward 与 Redirect 的区别?有哪些方式实现

【参考答案】

forward 是服务器资源转发，服务器直接访问目标地址的 URL，把那个 URL 的响应内容读取过来，然后把这些内容再发给浏览器，客户机并不知道发送的内容是从哪儿来的，所以地址栏中还是原来的地址；redirect 则是服务器收到请求后发送一个状态头给客户，客户将再请求一次，这里多了两次网络通信的来往。forward 会将请求状态和信息发至下一个 jsp 或 Servlet。redirect 是送到 client 端后再一次请求,信息不被保留，就是我们说的无法获取 request 中的参数。

实现方式：

HttpServletResponse 接口 sendRedirect（）方法进行重定向转发

RequestDispatcher.接口 forward（）方法进行请求转发

12. Servlet 的生命周期分为 3 个阶段？ 和 CGI 的区别？

【参考答案】

Servlet 的生命周期主要由 3 个过程组成。

(1)init()方法：服务器初始化 servlet。

(2)service()方法：初始化完毕，servlet 对象调用该方法响应客户的请求。

(3)destroy()方法：调用该方法消灭 servlet 对象。

其中，init()方法只在 servlet 第一次被请求加载的时候被调用一次，当有客户再请求 servlet 服务时，web 服务器将启动一个新的线程，在该线程中，调用 service 方法响应客户的请求。

【分析】

后一个问题，与 CGI 的区别，可以参考前面的答案。

13. 描述 Cookie 和 Session 的作用？区别和各自的应用范围？Session 工作原理。【北京科瑞明】

【参考答案】

Cookie 和 Session 都是用来服务器端和客户端进行会话跟踪的一种技术。

区别：Cookie 只能传输字符数据，字符是通过加密后写到客户端，下次请求时协带至服务器端，Cookie 协带的数据大小为 4KB，对数据量超出 4KB 的数据，无法处理，Cookie 数据一般是通过加密后存储在客户端，而 Session 在服务器端和浏览器缓存中都保存在 ID 的值，通过此 ID 来识别唯一的客户端。Session 对数据存储的大小没有限制，但存储的信息加重服务器的负载，另外 Session 在分布式服务器的使用上也有限制，Session 无法跨域，也就是多台服务器无法共享会话。

Session 原理：当客户端用户访问时，服务器都为每个用户分配一个唯一的会话 ID（Session ID) 保存在服务器内存中，服务器响应客户端时，将 Session ID 写入浏览器缓存中，当下次客户端请求时，就会将该 Session ID 携带至服务器，服务器再根据 ID 比对，识别不同客户端请求，以此方式来不断维持服务器和客户端状态跟踪。

14. dao 是什么及作用

【参考答案】

DAO 是数据库访问接口,负责管理与数据库打交道操作对象,将对数据维护与操作的方法通过 DAO 封装起来，使得这些操作可以从系统的业务逻辑中独立出来,从而使系统中的数据库操作变得统一和简单化，方便开发人员对组件职责进行划分。

15. 解释一下什么是 servlet

【参考答案】

Servlet 是用 Java 编写的服务器端程序，由服务器端调用和执行的 Java 类。是使用 Java Servlet 应用程序设计接口(API)及相关类和方法来构建的，而 Java Servlet API 定义了 servlet 和 Java 使能的服务器之间的一个标准接口，这使得 Servlets 具有跨服务器平台的特性，当客户机发送请求至服务器时，服务器可以将请求信息发送给 Servlet，并让 Servlet 建立起服务器返回给客户机的响应。Servlet 的功能很广泛，例如可以创建响应至客户端的所有 HTML 元素内容和动态数据内容，与其它组件通讯等等。

16. HTTP 请求的 GET 与 POST 方式的区别

【参考答案】

GET 和 POST.是 HTTP 定义的与服务器交互的不同方法，是使用 HTTP 的标准协议动词，用于编码和传送变量名/变量值对参数，并且使用相关的请求语义。

1、Get 方式在通过 URL 提交数据，数据在 URL 中可以看到；POST 方式，数据放置在 HTML HEADER 内提交，无法在地址栏看到。

2、GET 方式提交的数据最多只能有 1024 字节，而 POST 则没有此限制。

3、GET 一般用作小数据量的请求，POST 一般用作大数据量的请求，如：附件。

17. 什么情况下调用 doGet()和 doPost()？

【参考答案】

根据客户端的请求的方式来决定调用哪个方法处理请求，如果客户端采用 GET 方式发送请求，服务器端则采用 doGET()来处理，如果采用 post 方式，服务器端则采用 doPOST()

18. request 对象的主要方法

【参考答案】

setAttribute(String name,Object)：设置名字为 name 的 request 的参数值
getAttribute(String name)：返回由 name 指定的属性值
getAttributeNames()：返回 request 对象所有属性的名字集合，结果是一个枚举的实例
getCookies()：返回客户端的所有 Cookie 对象，结果是一个 Cookie 数组
getCharacterEncoding()：返回请求中的字符编码方式
getContentLength()：返回请求的 Body 的长度
getHeader(String name)：获得 HTTP 协议定义的文件头信息
getHeaders(String name)：返回指定名字的 request Header 的所有值，结果是一个枚举的实例
getHeaderNames()：返回所以 request Header 的名字，结果是一个枚举的实例
getInputStream()：返回请求的输入流，用于获得请求中的数据
getMethod()：获得客户端向服务器端传送数据的方法
getParameter(String name)：获得客户端传送给服务器端的有 name 指定的参数值
getParameterNames()：获得客户端传送给服务器端的所有参数的名字，结果是一个枚举的实例
getParametervalues(String name)：获得有 name 指定的参数的所有值
getProtocol()：获取客户端向服务器端传送数据所依据的协议名称
getQueryString()：获得查询字符串
getRequestURI()：获取发出请求字符串的客户端地址
getRemoteAddr()：获取客户端的 IP 地址
getRemoteHost()：获取客户端的名字
getSession([Boolean create])：返回和请求相关 Session
getServerName()：获取服务器的名字
getServletPath()：获取客户端所请求的脚本文件的路径
getServerPort()：获取服务器的端口号
removeAttribute(String name)：删除请求中的一个属性

19. request.getAttribute() 和 request.getParameter() 有何区别?

【参考答案】

request.getAttribute()获取在请求对象中设置的属性，该方法返回对象为 Object 类型，而 getParameter()方法是获取指定的请求参数值，返回值为 String 类型的字符串。

20. jsp 有哪些内置对象?作用分别是什么？分别有什么方法？ 【软通动力面试题】

【参考答案】

答:JSP 共有以下 9 个内置的对象：

1) request 用户端请求，此请求会包含来自 GET/POST 请求的参数
2) response 网页传回用户端的回应

3) pageContext 网页的属性是在这里管理
4) session 与请求有关的会话期
5) application servlet 正在执行的内容
6) out 用来传送回应的输出
7) config servlet 的构架部件
8) page JSP 网页本身
9) exception 针对错误网页，未捕捉的例外

具体说明如下：

request 表示 HttpServletRequest 对象。它包含了有关浏览器请求的信息，并且提供了几个用于获取 cookie, header, 和 session 数据的有用的方法。

response 表示 HttpServletResponse 对象，并提供了几个用于设置送回 浏览器的响应的方法（如 cookies,头信息等）

out 对象是 javax.jsp.JspWriter 的一个实例，并提供了几个方法使你能用于向浏览器回送输出结果。

pageContext 表示一个 javax.servlet.jsp.PageContext 对象。它是用于方便存取各种范围的名字空间、servlet 相关的对象的 API，并且包装了通用的 servlet 相关功能的方法。

session 表示一个请求的 javax.servlet.http.HttpSession 对象。Session 可以存贮用户的状态信息

applicaton 表示一个 javax.servle.ServletContext 对象。这有助于查找有关 servlet 引擎和 servlet 环境的信息

config 表示一个 javax.servlet.ServletConfig 对象。该对象用于存取 servlet 实例的初始化参数。

page 表示从该页面产生的一个 servlet 实例

21. JSP 的常用指令

【参考答案】

共三种，分别是:

1) <%@include >用来在 JSP 页面包含静态资源
2) <%@taglib >用来指定 JSP 页面标签类型
3) <%@page >用来指定页面相关属性

22. 页面间对象传递的方法

【参考答案】

request、session、application、Cookie 等,其中比较常用的像 request、Session。request 主要是应用在同一请求周期内，可能进行对象或参数的共享传递。而 Session 主要可以应用于同一客户端会话周期内进行参数属性的共享。

23. MVC 的各个部分都有那些技术来实现?如何实现?

【参考答案】

MVC 是 Model－View－Controller 的简写。Model 代表的是应用的业务逻辑（通过 JavaBean，EJB 组件实现）， View 是应用的表示面（由 JSP 页面产生），Controller 是提供应用的处理过程控制（一般是一个 Servlet），通过这种设计模型把应用逻辑，处理过程和显示逻辑分成不同的组件实现。这些组件可以进行交互和重用。

24. 我们在 web 应用开发过程中经常遇到输出某种编码的字符，如 iso8859-1 等，如何输出一个某种编码的字符串？

【参考答案】

```
public String translate (String str) {
   String tempStr = "";
   try {
     tempStr = new String(str.getBytes("ISO-8859-1"), "GBK");
     tempStr = tempStr.trim();
   }
   catch (Exception e) {
     System.err.println(e.getMessage());
   }
   return tempStr;
 }
```

25. Web 系统安全因素有哪些？

【参考答案】

1)操作系统、后台数据库的安全问题：这里指操作系统和后台数据库的漏洞，配置不当，如弱口令等等，导致黑客、病毒可以利用这些缺陷对网站进行攻击。

2)Web 发布系统的漏洞 ：Web 业务常用的发布系统(即 Web 服务器)，如 IIS、Apache 等，这些系统存在的安全漏洞，会给入侵者可乘之机。

3)Web 应用程序的漏洞：主要指 Web 应用程序的编写人员，在编程的过程中没有考虑到安全的因素，使得黑客能够利用这些漏洞发起对网站的攻击，比如 SQL 注入、跨站脚本攻击等等。

4)自身网络的安全状况：网站服务器所处的网络安全状况也影响着网站的安全，比如网络中存在的 DoS 攻击等，也会影响到网站的正常运营。

26. web 运用程序的稳定、安全需要考虑哪些？

【参考答案】

Web 服务器的性能考虑主要有：并发用户数、事务安全、负载均衡、时段流量、网络带宽网格安全等。

网络安全方面：

- 关键数据的保护，例如用户数据等
- 功能服务的正常提供。
- 网站的防攻击能力。
- 对异常灾害的恢复能力。

程序性能：

- 响应请求并运行得出结果的时间。
- 错误的检测和拦截。
- 扩展性。

27. 介绍几种 WEB 服务器，问 Tomcat 里面的文件组成？

【参考答案】

Apache、Tomcat、Jetty、Jboss、Websphere、WebLogic

bin:启动、关闭 Tomcat 的命令。
common/lib:网络编程的 jar 文件。
conf:配置文件。
logs:日志文件.
server:自带的 web 应用(三个).
shared:所有 web 应用都可以访问的内容.
temp:临时.
webapps:默认站点文件夹.
work:jsp 生成的类.

Tomcat 是一个开放源代码、运行 servlet 和 JSP Web 应用软件的基于 Java 的 Web 应用软件容器。Tomcat Server 是根据 servlet 和 JSP 规范进行执行的，因此我们就可以说 Tomcat Server 也实行了 Apache-Jakarta 规范且比绝大多数商业应用软件服务器要好。

28. 将 ISO8859-1 字符串转成 GB2312 编码，语句为？

【参考答案】

String s=new String(text.getBytes(“iso8859-1”),”gb2312”)

29. <jsp:include>与<%@include>的区别

【参考答案】

上面两种都是在当前页面加载或包含另一页面内容，而使用<jsp:include>动作标签实现时它总是会检查所含文件中的变化，适合用于包含动态页面，并且包含的页页要符合 web 容器语法要求,因为被包含的页面会被 web 引擎进行编译和加载，同时包含时会发送 http 请求，所以可以携带参数。

而<%@include>是用伪码实现,定不会检查所含文件的变化,只是简单的将被加载的资源拷贝到了当前，而这种包含更多的是一种重用，不能实现参数共享。

30. 说出数据连接池的工作机制是什么？

【参考答案】

web 服务器启动时会建立一定数量的池连接，并一直维持不少于此数目的池连接。客户端程序需要连接时，池驱动程序会返回一个未使用的池连接并将其表记为忙。如果当前没有空闲连接，池驱动程序就新建一定数量的连接，新建连接的数量有配置参数决定。当使用的池连接调用完成后，池驱动程序将此连接表记为空闲，其他调用就可以使用这个连接。
实现方式，返回的 Connection 是原始 Connection 的代理，代理 Connection 的 close 方法不是真正关连接，而是把它代理的 Connection 对象还回到连接池中。

31. 如何实现 Servlet 单线程

【参考答案】

<%@ page isThreadSafe="false"%>

32. 哪些方法可以提高 JDBC 性能?

【参考答案】

1. 使用数据连接池(Connection Pool), 避免使用 DriverManager.getConnection（）。
2. 合理的配置数据连接池参数，合理如何设置数据连接池的初始大小
3. 选择合适的事务等级，按照不同的数据库操作类型选择不同的事务等级。

4. 及时关闭 Connection，不关闭的话会严重影响系统的性能，甚至造成系统罢工

5. 优化 Statement

1) 选择合适的 Statement, 根据不同的数据库操作选择 Statement, PreparedStatement 或者 CallableStatement

2) 尽可能的使用 batch, 这样可以减少调用 JDBC 的次数。

3) Statement 执行完毕后关闭 Statement

6. 优化你的 SQL, 尽量减少你的结果集，不要每次都"select * from XXX"

7. 使用一些缓存工具进行缓存，特别是大数据查询。

33. 实现会话跟踪有哪几个方式？

【参考答案】

1、URL 重写

2、隐藏表单域

3、Cookie

4、Session

34. Web 容器里面的对象存活周期？

【参考答案】

当然由 web 容器进行创建管理的对象主要有 application，session，request，page 这四个级别的对象，而这 4 种级别的对象，根据它们自身的特点来管理所持的对象，如：request 中的对象的生命周期就是在请求范围内，Session 在是会话周期内，page 是在当前 JSP Page 内，Application 是在服务器启、停的周期内。

【分析】

总的来说这道题有点没明白，提问者想问的是东西是什么。看到题第一反应以为是问 Servlet 的生存周期，因为说到 Web 容器对象，一般指的是 Servlet，Servlet 组件是由容器进行创建、调用和管理的，所以首先想到了 Servlet 的存活周期。

35. 浏览器页面与 Tomcat 的交互过程？

【参考答案】

当一个 JSP 页面第一次被访问的时候，JSP 引擎将执行以下步骤：

（1）将 JSP 页面翻译成一个 Servlet，这个 Servlet 是一个 java 文件，同时也是一个完整的 java 程序

（2）再由 java 编译器对这个 Servlet 进行编译，得到可执行 class 文件

（3）再由 JVM 解释器来解释执行 class 文件，生成向客户端发送的应答，然后发送给客户端

以上三个步骤仅仅在 JSP 页面第一次被访问时才会执行，以后的访问速度会因为 class 文件已经生成而大大提高。

36. 什么是断点续传，HTTP 是否支持上传下载，原理？【大唐动力面试题】

所谓断点续传，也就是要从文件已经下载的地方开始继续下载。客户端在请求时，除了其它的信息外，需要增加一条参数，告诉服务器从哪里开始传，在读取时从指定的字节数的开始位向后读取。简单示例：

```
RandomAccess oSavedFile = new RandomAccessFile("down.zip","rw");
```

```
long nPos = 100000; //表示从 100000 字节位置开始读取。
// 定位文件指针到 nPos 位置
oSavedFile.seek(nPos);
byte[] b = new byte[1024];
int nRead;
// 从输入流中读入字节流，然后写到文件中
while((nRead=input.read(b,0,1024)) > 0)
{
oSavedFile.write(b,0,nRead);
}
```

（三）SQLServer 数据库

1. SQL 有哪三种注入方式？SQL 安全

动态 SQL 拼装注入、SQL 溢出漏洞、获取管理员权限、

2. 数据库事务及隔离级别【中恒互联】

隔离级别：脏读、幻读、一致读、不可重复读、更新丢失

1. 更新丢失（Lost update）：两个事务都同时更新一行数据但是第二个事务却中途失败退出导致对数据两个修改都失效了这是系统没有执行任何锁操作因此并发事务并没有被隔离开来

2. 脏读（Dirty Reads）：一个事务开始读取了某行数据但是另外一个事务已经更新了此数据但没有能够及时提交。这是相当危险很可能所有操作都被回滚

3. 不可重复读（Non-repeatable Reads）：一个事务对同一行数据重复读取两次但是却得到了不同结果。例如在两次读取中途有另外一个事务对该行数据进行了修改并提交

4. 两次更新问题（Second lost updates problem）：无法重复读取特例，有两个并发事务同时读取同一行数据然后其中一个对它进行修改提交而另一个也进行了修改提交这就会造成第一次写操作失效

5. 幻读（Phantom Reads）：也称为幻像（幻影）。事务在操作过程中进行两次查询，第二次查询结果包含了第一次查询中未出现的数据（这里并不要求两次查询 SQL 语句相同）这是因为在两次查询过程中有另外一个事务插入数据造成的

3. 事务四大属性

原子性、一致性、隔离性、持久性。

4. 说说存储过程定义，并描述一下优点和缺点？【奇谷网络】

语法：

```
CREATE PROCEDURE getUserInfo_PROC
    AS
        begin
      //过程体
         end
  GO
```

优点：

- 减轻网络流量
- 可被作为一种安全机制来利用
- 允许标准组件式编程
- 能够实现较快的执行速度

缺点：

- 可移值性差
- 重构复杂

5. 数据库中为什么要映射主外健?什么是事务处理?【海天华光】

主外键：保持数据完整性。

数据库事务是指作为单个逻辑工作单元执行的一系列操作。事务可以确保除非事务性单元内的所有操作都成功完成，否则不会永久更新面向数据的资源。事务内相关操作组合为一个要么全部成功要么全部失败，可以简化错误恢复并使应用程序更加可靠。一个逻辑工作单元要成为事务，必须满足所谓的 ACID(原子性、一致性、隔离性和持久性)属性。

6. SqlServer 的完整性约束?

– 主键约束（Primary Key Constraint）：要求主键列数据唯一，并且不允许为空
– 唯一约束（Unique Constraint）：要求该列唯一，允许为空，但只能出现一个空值。
– 检查约束（Check Constraint）：某列取值范围限制、格式限制等，如有关年龄的约束
– 默认约束（Default Constraint）：某列的默认值，如我们的男性学员较多，性别默认为“男”
– 外键约束（Foreign Key Constraint）：用于两表间建立关系，需要指定引用主表的那列

7. 一张学生表 student，有字段班级 classid，学号 id，成绩 grade

（1）求各个班的平均成绩

```
Select classid,avg(grade) from student group by classid
```

（2）查询出比该班平均成绩高的学生的所有信息

```
select w.id,w. grade,b.avg_ grade from student w,(
select classid,avg(grade) avg_grade from student group by classid) b
   where w. classid =b. classid and w.grade >b.avg_grade;
```

8. 写一条 SQL 语句，查询姓张学生中平均成绩大于 75 的学生信息

```
select * from student where name in (select name from student
where name like '张%' group by name having avg(score) > 75)
```

9. 查一下每门课程都大于 80 的学生姓名

学生表 student　分数表 grade

```
select s.name from student s where s.id not in(select g.studentid from grade g where
g.marks<=80)
```

或者

```
select s.name from student s where not exists(select 1 from grade g where
g.studentid=s.id and g.marks<=80)
```

10. truncate 与 delete 的区别？（delete from table 和 truncate table 的区别!)【天晟科技面试题】

truncate 是 DDL 語言.delete 是 DML 語言 DDL 語言是自動提交的.命令完成就不可回滚.truncate 的速度也比 delete 要快得多.

详细说明：

相同点:truncate 和不带 where 子句的 delete, 以及 drop 都会删除表内的数据
不同点:
1. truncate 和 delete 只删除数据不删除表的结构(定义)
drop 语句将删除表的结构被依赖的约束(constrain),触发器(trigger),索引(index);依赖于该表的存储过程/函数将保留,但是变为 invalid 状态.
2.delete 语句是 dml,这个操作会放到 rollback segement 中,事务提交之后才生效;如果有相应的 trigger,执行的时候将被触发.
truncate,drop 是 ddl, 操作立即生效,原数据不放到 rollback segment 中,不能回滚.操作不触发 trigger.
3.delete 语句不影响表所占用的 extent, 高水线(high watermark)保持原位置不动
显然 drop 语句将表所占用的空间全部释放
truncate 语句缺省情况下见空间释放到 minextents 个 extent,除非使用 reuse storage; truncate 会将高水线复位(回到最开始).
4.速度,一般来说: drop> truncate > delete
5.安全性:小心使用 drop 和 truncate,尤其没有备份的时候.否则哭都来不及
使用上,想删除部分数据行用 delete,注意带上 where 子句. 回滚段要足够大.
想删除表,当然用 drop
想保留表而将所有数据删除. 如果和事务无关,用 truncate 即可. 如果和事务有关,或者想触发 trigger,还是用 delete.

11. 一学生表！有班及 id，学号！成绩！一求平均成绩！二求比平均成绩高的学生的所有信息

select id,avg(成绩) from table group by id
select * from table where 成绩 >(select 成绩 from (select id,avg(成绩) 成绩 from table group by id) as a)

12. 查询出 M 页的 N 行数据(分页的实现，求第 M 也的记录数)

Select top N from table where id not in (select top (M-1)*N id from table)

13. 数据库三范式是什么?

第一范式（1NF）:
字段具有原子性,不可再分。所有关系型数据库系统都满足第一范式）
数据库表中的字段都是单一属性的，不可再分。例如，姓名字段，其中的姓和名必须作为一个整体，无法区分哪部分是姓，哪部分是名，如果要区分出姓和名，必须设计成两个独立的字段。

第二范式（2NF）:
第二范式（2NF）是在第一范式（1NF）的基础上建立起来的，即满足第二范式（2NF）必须先满足第一范式（1NF）。
要求数据库表中的每个实例或行必须可以被惟一地区分。通常需要为表加上一个列，以存储各个实例的惟一标识。这个惟一属性列被称为主关键字或主键。
第二范式（2NF）要求实体的属性完全依赖于主关键字。所谓完全依赖是指不能存在仅依赖主关键字一部分的属性，如果存在，那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体,新实体与原实体之间是一对多的关系。为实现区分通常需要为表加上一个列，以存储各个实例的惟一标识。简而言之，第二范式就是非主属性非部分依赖于主关键字。

第三范式（3NF）：

满足第三范式（3NF）必须先满足第二范式（2NF）。简而言之，第三范式（3NF）要求一个数据库表中不包含已在其它表中已包含的非主关键字信息。

所以第三范式具有如下特征：

1，每一列只有一个值

2，每一行都能区分。

3，每一个表都不包含其他表已经包含的非主关键字信息。

例如，帖子表中只能出现发帖人的 id，而不能出现发帖人的 id，还同时出现发帖人姓名，否则，只要出现同一发帖人 id 的所有记录，它们中的姓名部分都必须严格保持一致，这就是数据冗余。

14. join 与 left join 的区别：

inner join(等值连接) 只返回两个表中联结字段相等的行

left join(左联接) 返回包括左表中的所有记录和右表中联结字段相等的记录

right join(右联接) 返回包括右表中的所有记录和左表中联结字段相等的记录

15. 用 sql 语句分页：

Mysql 数据库：

```
SELECT TOP 页大小 * FROM table1 WHERE id NOT IN (
        SELECT TOP 页大小*(页数-1) id FROM table1 ORDER BY id
        ) ORDER BY id
```

Oracle 数据库：

在 ORACLE 大数据量下的分页解决方法。一般用截取 ID 方法，还有是三层嵌套方法。

截取 ID 的方法

```
select * from emp a, (select empno,rownum as num from emp)b where
a.empno=b.empno and b.num between 5 and 7;
```

三层嵌套

```
SELECT * FROM ( SELECT A.*, rownum r FROM ( SELECT * FROM emp ) A WHERE
rownum <=7 ) B WHERE r >5;
```

16. SQL 编程题 1：

有表 Table_1 ,数据如下

ID	NAM	JIEGUO
1	A	WIN
2	A	LOST
3	A	WIN
4	A	LOST
6	B	WIN
5	A	WIN
7	B	LOST
8	B	LOST

要求统计出，如下结果

```
NAM       Win_num   Lost_num
   A                        3                      2
   B                      1                      2
```

答：

```
select nam,
count(case jieguo when  'WIN'  THEN 1 ELSE Null END) as Win_num,
count(case jieguo when  'LOST' THEN 1 ELSE Null END) as Lost_num
from TABLE_1
group by nam
```

或者

```
select nam,
sum(case jieguo when 'WIN' THEN 1 ELSE 0 END) as Win_num,
sum(case jieguo when 'LOST' THEN 1 ELSE 0 END) as Lost_num
from TABLE_1
group by nam
```

17. SQL 编程题 2：

给下面这样的一个表记录：

```
---------------------------------
CUSTOMER_NAME          TRADE_NAME        NUM
A                                        甲                  2
B                                  乙              4
C                                  丙              1
A                                  丁              2
B                                  丙              5
```

给出所有购入商品为两种或两种以上的购物人记录。

答：

方法一（推荐，使用了 count(distinct 列名) ）

```
select * from T1
where customer_name in
(select customer_name
from T1 as b
group by customer_name
having count(distinct trade_name)>=2)
```

说明：

having count ，发生在分组之后，对每个分组包含的行，进行统计 count(distinct trade_name)，根据这个统计值（每个分组一个），确定是否输出此分组。

方法二（容易看迷糊，用了 2 次 group by）

```
select * from T1 WHERE CUSTOMER_NAME IN
(SELECT * FROM
(select CN1 as CN from (select customer_name AS CN1,trade_name AS TN1 from T1 a
s B group by customer_name,trade_name) as A group by CN1
having count(TN1)>=2) as B )
```

说明：
1.group by customer_name,trade_name ：用于排除 （customer_name + trade_name）重复的行，只留一行）。
2.group by CN1 having count(TN1)>=2 ：用于排除只买了一种商品的顾客。

18. SQL 编程题 3：

Student 表有三列,分别是姓名、课程、成绩(见下表)，其中 stu_name 列有约 1000 行，course 列除了已经列出来的 马哲、数学、英语、语文外，还有一些学科种类没有列完（即 学科种类数不定），未参加考试的学生 Mark 列值为 0 。

stu_name	course	mark
城南	马哲	70
城南	数学	65
城南	英语	58
城南	语文	79
李四	马哲	61
李四	数学	80
李四	英语	77
李四	语文	80
王朝	马哲	52
王朝	数学	55
王朝	英语	59
王朝	语文	90
张三	马哲	66
张三	数学	88
张三	英语	61
张三	语文	70

要求查询：
1、每一门课程都及格的学生的姓名
2、总分排名在前三名的学生的姓名
答：
1、

```
SELECT stu_name
FROM T2
GROUP BY stu_name
HAVING (COUNT(CASE WHEN mark >= 60 THEN 1 ELSE NULL END) = COUNT(Mark))
```

说明：此题的关键点是 “学科种类数不定” 和 “每一门都及格”，此点用 COUNT(CASE WHEN mark >= 60 THEN 1 ELSE NULL END) = COUNT(Mark) 来实现。

还有更加简单、易理解的解答：
从 原始表 中 排出 那些 不及格的人，就可以啦。

```
SELECT Distinct  name   FROM StudentScore
WHERE name     NOT     IN
(Select Distinct name from StudentScore where score<60);
```

2、

```
select top 3 stu_name,sum(mark) as 总分
from T2
group by stu_name
order by sum(mark) desc
```

还有一种解答，SQL2005 以上版本才能用的。

```
SELECT
ROW_NUMBER()   OVER(ORDER BY SUM(SCORE) DESC)   AS    ID,
NAME,
SUM(SCORE)   AS   TOTAL
FROM StudentScore
GROUP BY name;
```

19. SQL 编程题 4：

表 className 中有如下分类:

classID	className
1	衣服
2	裤子
5	帽子
10	鞋子

表 productInfo 有如下记录:

productID	productName	parentID	clickNum	
1	男士衣服	1	90	--衣服类别中点击率最高
2	女士衣服	1	80	
3	男士裤子	2	70	
4	女士裤子	2	90	--裤子类别中点击率最高
5	男士帽子	5	15	
6	女士帽子	5	30	--帽子类别中点击率最高
7	男士鞋子	10	65	--鞋子类别中点击率最高
8	女士鞋子	10	52	
9	女士鞋子 1	10	54	

现在要求分别把衣服,裤子,帽子,鞋子这些类别中点击率最高的一条记录找出来,然后再降序排列,结果应如下:

productID	productName	clickNum
1	男士衣服	90
4	女士裤子	90
7	男士鞋子	65
6	女士帽子	30

答:

```
select productid,productname,clicknum
from ProductInfo
inner join
(select parentid as P_id,max(clicknum) as Maxclick
from   ProductInfo   group by parentid) as A
on T3.parentid=A.P_id and T3.clicknum=A.Maxclick
order by clicknum desc,productid asc
```

20. SQL 编程题 5:

有三个表:
一张 老师表 TecTable,字段是 T_ID,T_NAME;
一张 学生表 StuTable,字段是 S_ID,S_NAME;
一张 班级表 ClassTable,字段是 T_ID,S_ID,C_NAME,其中 C_NAME 的取值只有‘大班’和‘小班’。
请查询出符合条件的老师的名字,条件是老师在大班中带的学生数大于此老师在小班中带的学生数。

答:

未验证

```
select
TecTable.Techer_ID,
Techer_NAME,
count(case Class_name when '大班' then 1 else null end) as Big_stu_num,
count(case Class_name when '小班' then 1 else null end) as Little_stu_num
from TecTable,StuTable,ClassTable
where
  TecTable.Techer_ID=ClassTable.Techer_ID and StuTable.Stu_id=ClassTable.Stu_id
group by Techer_ID,Techer_NAME
having Big_stu_num > Little_stu_num
```

网上还有一种解法:

```
select * from T,
(select count(*) as x,T_ID from C where c_name='小班' group by T_ID) a,
(select count(*) as x,T_ID from C where c_name='大班' group by T_ID) b where b.
x >a.x and a.T_ID=b.T_ID and T.T_ID=b.T_ID
```

group by 后面还要多加个班级名。

21. SQL 程序题 6（行列转换）

已知一个表 T4, 结构为：

stu_name	course	result
张三	语文	20
张三	数学	30
张三	英语	50
李四	语文	70
李四	数学	60
李四	英语	90

问：怎样通过 SQL 的 Select 语句输出以下结构的数据

stu_name	语文	数学	英语
张三	20	30	50
李四	70	60	90

答：

```
SELECT
A1.STU_NAME,A1.RESULT AS '语文',A2.RESULT AS '数学',A3.RESULT AS '英语'
FROM
T4 A1 INNER JOIN T4 A2 ON A1.STU_NAME=A2.STU_NAME
INNER JOIN T4 A3 ON A2.STU_NAME=A3.STU_NAME
WHERE
A1.COURSE='语文' AND A2.COURSE='数学' AND A3.COURSE='英语'
```

22. SQL 编程题 7：

有一个学生表，结构如下：

id	name	course	score
1	学生 1	语文	88.50
2	学生 1	数学	95.00
3	学生 1	英语	85.50
4	学生 2	语文	100.00
5	学生 2	数学	76.00
6	学生 2	英语	70.50
7	学生 3	语文	68.50

8 学生 3 数学 77.50
9 学生 3 英语 91.50
10 学生 4 语文 72.50
11 学生 4 数学 86.50
12 学生 4 英语 77.50

要求进行行转列，得出如下结果：

name	语文	数学	英语
学生 2	100.00	76.00	70.50
学生 1	88.50	95.00	85.50
学生 4	72.50	86.50	77.50
学生 3	68.50	77.50	91.50

答：

```
select
name,
sum(case cource when '语文' then score else 0 end) as '语文',
sum(case cource when '数学' then score else 0 end) as '数学',
sum(case cource when '英语' then score else 0 end) as '英语'
from Score
group by name
order by '语文' desc
```

（四）Oracle 数据库

1. oracle 中 row_id 理解

ORACLE 的 row_id 是一个伪列,其个是为 18 个字节可将这 18 个字节用 6363 来划分,分别表示段编号,数据文件编号,数据块

2. 嵌入式数据库和传统数据库的区别【大唐动力面试题】

嵌入式数据库主要像：SQLite、

传统数据库服务器：SQL Server 、Oracle、MySQL

嵌入式数据库：SQLite 的主要特点：

1. 支持事件，不需要配置，不需要安装，也不需要管理员；

2. 支持大部分 SQL92；

3.一个完整的数据库保存在磁盘上面一个文件，同一个数据库文件可以在不同机器上面使用，最大支持数据库到 2T，字符和 BLOB 的支持仅限制于可用内存；

4.整个系统少于 3 万行代码，少于 250KB 的内存占用(gcc)，大部分应用比目前常见的客户端/服务端的数据库快，没有其它依赖

5. 源代码开放，代码 95%有较好的注释，简单易用的 API。官方带有 TCL 的编译版本。

关系数据库特点：

1、更好的安全性、多用户管理

2、强大的数据管理能力，如索引、视图等关系对象

3、强大的数据库编程式的设计，像 T-SQL、存储过程、游标

4、丰富的数据类型

其它略。。。。

3. Inserted 和 deleted 的含义？

inserted 表反映插入或更新操作时插入的记录

deleted 表反映删除或更新操作时删除的记录

4. 函数和过程的区别？

- **存储过程：**
  1) 一般用于在数据库中完成特定的业务或任务
  2) 可以定义返回类型，也可以不定义返回类型
  3) SQL 语句中不可以调用
- **函数：**
  1) 一般用于特定的数据查询或数据转转换处理
  2) 申请时必须要定义返回类型，且程序体中必须定义 return 语句。
  3) 不能独立执行，必须作为表达式的一部分调用
  4) SQL 语句中可以调用。

5. 数据库优化的方案

建立主键，为数据库创建索引，建立存储过程，触发器，可提高查询速度。

6. Oracle 中有哪几种索引

1.单列索引与复合索引

一个索引可以由一个或多个列组成，用来创建索引的列被称为“索引列”。

单列索引是基于单列所创建的索引，复合索引是基于两列或者多列所创建的索引。

2.唯一索引与非唯一索引

唯一索引是索引列值不能重复的索引，非唯一索引是索引列可以重复的索引。

无论是唯一索引还是非唯一索引，索引列都允许取 NULL 值。默认情况下，Oracle 创建的索引是不唯一索引。

3.B 树索引

B 树索引是按 B 树算法组织并存放索引数据的，所以 B 树索引主要依赖其组织并存放索引数据的算法来实现快速检索功能。

4.位图索引

位图索引在多列查询时，可以对两个列上的位图进行 AND 和 OR 操作，达到更好的查询效果。

5.函数索引

Oracle 中不仅能够直接对表中的列创建索引，还可以对包含列的函数或表达式创建索引，这种索引称为“位图索引

7. 数据库索引的优点和缺点【首航财务】

优点：

1、通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性。

2、可以大大加快数据的检索速度，这也是创建索引的最主要的原因。

3、可以加速表和表之间的连接，特别是在实现数据的参考完整性方面特别有意义。

4、在使用分组和排序子句进行数据检索时，同样可以显著减少查询中分组和排序的时间。

5、通过使用索引，可以在查询的过程中，使用优化隐藏器，提高系统的性能。

缺点：

1、创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加。

2、索引需要占物理空间，除了数据表占数据空间之外，每一个索引还要占一定的物理空间，如果要建立聚　簇索引，那么需要的空间就会更大。

3、当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，这样就降低了数据的维护速度。

8. 触发器有几种？

共 2 种，一种 DML 触发，就是遇到 DML 事件时触发执行，像 insert\update\delete。一种 DDL 触发，遇到 DDL 事件时触发，像 Login Datatabase、更改数据库状态、create 语句等。

9. oracle 中除了数据库备份，还有什么方法备份？

Oracle 数据库有三种标准的备份方法，它们分别是导出/导入(EXP/IMP)、热备份和冷备份。导出备份是一种逻辑备份，冷备份和热备份是物理备份。

10G 有几种新功能进行备份，像数据磅

10. 写出删除表中重复记录的语句 oracle

```
delete from people
where peopleId in (select   peopleId from people group by   peopleId   having
```

```
count(peopleId) > 1)  and rowid not in (select min(rowid) from  people group by
```

11. 数据库里面游标，索引是怎么用的？

declare cur cursor keyset for

get 返回 null,load classnotfoundException

12. .在 Oracle 中数据库中的一个表中，这个表没有主键 id 也没有特定标示来查数据，怎么查？

利用伪列标识进行查询。

13. 用两种方式根据部门号从高到低，工资从低到高列出每个员工的信息。

```
employee:
     eid,ename,salary,deptid;
 select * from employee order by deptid desc,salary
```

14. 列出各个部门中工资高于本部门的平均工资的员工数和部门号，并按部门号排序。【上海锦星信息面试题】

```
select a.ename,a.salary,a.deptid
 from emp a,
    (select deptd,avg(salary) avgsal from emp group by deptid ) b
 where a.deptid=b.deptid and a.salary>b.avgsal ;
```

15. union 和 union all 有什么不同?

假设我们有一个表 Student，包括以下字段与数据：

```
drop table student;
create table student
(
id int primary key,
name nvarchar2(50) not null,
score number not null
);
insert into student values(1,'Aaron',78);
insert into student values(2,'Bill',76);
insert into student values(3,'Cindy',89);
insert into student values(4,'Damon',90);
insert into student values(5,'Ella',73);
insert into student values(6,'Frado',61);
insert into student values(7,'Gill',99);
insert into student values(8,'Hellen',56);
insert into student values(9,'Ivan',93);
insert into student values(10,'Jay',90);
commit;
```

Union 和 Union All 的区别。

```
select *
from student
where id < 4
union
select *
from student
where id > 2 and id < 6
```

结果将是

```
1       Aaron       78
2       Bill       76
3       Cindy       89
4       Damon       90
5       Ella       73
```

如果换成 Union All 连接两个结果集，则返回结果是：

```
1       Aaron       78
2       Bill       76
3       Cindy       89
3       Cindy       89
4       Damon       90
5       Ella       73
```

可以看到，Union 和 Union All 的区别之一在于对重复结果的处理。

UNION 在进行表链接后会筛选掉重复的记录，所以在表链接后会对所产生的结果集进行排序运算，删除重复的记录再返回结果。实际大部分应用中是不会产生重复的记录，最常见的是过程表与历史表 UNION。如：

```
select * from gc_dfys
union
select * from ls_jg_dfys
```

这个 SQL 在运行时先取出两个表的结果，再用排序空间进行排序删除重复的记录，最后返回结果集，如果表数据量大的话可能会导致用磁盘进行排序。

而 UNION ALL 只是简单的将两个结果合并后就返回。这样，如果返回的两个结果集中有重复的数据，那么返回的结果集就会包含重复的数据了。

从效率上说，UNION ALL 要比 UNION 快很多，所以，如果可以确认合并的两个结果集中不包含重复的数据的话，那么就使用 UNION ALL，

16. 用一条 SQL 语句 查询出每门课都大于 80 分的学生姓名

name	kecheng	fenshu
张三	语文	81
张三	数学	75
李四	语文	76
李四	数学	90
王五	语文	81
王五	数学	100
王五	英语	90

准备数据的 sql 代码：

```
create table score(id int primary key auto_increment,name varchar(20),subject varchar(20),score int);
insert into score values
(null,'张三','语文',81),
(null,'张三','数学',75),
(null,'李四','语文',76),
(null,'李四','数学',90),
(null,'王五','语文',81),
(null,'王五','数学',100),
(null,'王五 ','英语',90);
```

提示：当百思不得其解时，请理想思维，把小变成大做，把大变成小做，

答案：

```
A: select distinct name from score  where  name not in (select distinct name from score where score<=80)

B:select distince name t1 from score where 80< all (select score from score where name=t1);
```

17. 所有部门之间的比赛组合

一个叫 department 的表，里面只有一个字段 name,一共有 4 条纪录，分别是 a,b,c,d,对应四个球对，现在四个球对进行比赛，用一条 sql 语句显示所有可能的比赛组合.

答：

```
select a.name, b.name
from team a, team b
where a.name < b.name
```

18. 每个月份的发生额都比 101 科目多的科目

请用 SQL 语句实现：从 TestDB 数据表中查询出所有月份的发生额都比 101 科目相应月份的发生额高的科目。请注意：TestDB 中有很多科目，都有 1－12 月份的发生额。

AccID：科目代码，Occmonth：发生额月份，DebitOccur：发生额。

数据库名：JcyAudit，数据集：Select * from TestDB

准备数据的 sql 代码：

```
drop table if exists TestDB;
create table TestDB(id int primary key auto_increment,AccID varchar(20), Occmonth date, DebitOccur bigint);
insert into TestDB values
(null,'101','1988-1-1',100),
(null,'101','1988-2-1',110),
(null,'101','1988-3-1',120),
(null,'101','1988-4-1',100),
(null,'101','1988-5-1',100),
(null,'101','1988-6-1',100),
```

```
(null,'101','1988-7-1',100),
(null,'101','1988-8-1',100);
--复制上面的数据，故意把第一个月份的发生额数字改小一点
insert into TestDB values
(null,'102','1988-1-1',90),
(null,'102','1988-2-1',110),
(null,'102','1988-3-1',120),
(null,'102','1988-4-1',100),
(null,'102','1988-5-1',100),
(null,'102','1988-6-1',100),
(null,'102','1988-7-1',100),
(null,'102','1988-8-1',100);
--复制最上面的数据，故意把所有发生额数字改大一点
insert into TestDB values
(null,'103','1988-1-1',150),
(null,'103','1988-2-1',160),
(null,'103','1988-3-1',180),
(null,'103','1988-4-1',120),
(null,'103','1988-5-1',120),
(null,'103','1988-6-1',120),
(null,'103','1988-7-1',120),
(null,'103','1988-8-1',120);
--复制最上面的数据，故意把所有发生额数字改大一点
insert into TestDB values
(null,'104','1988-1-1',130),
(null,'104','1988-2-1',130),
(null,'104','1988-3-1',140),
(null,'104','1988-4-1',150),
(null,'104','1988-5-1',160),
(null,'104','1988-6-1',170),
(null,'104','1988-7-1',180),
(null,'104','1988-8-1',140);
--复制最上面的数据，故意把第二个月份的发生额数字改小一点
insert into TestDB values
(null,'105','1988-1-1',100),
(null,'105','1988-2-1',80),
(null,'105','1988-3-1',120),
(null,'105','1988-4-1',100),
(null,'105','1988-5-1',100),
(null,'105','1988-6-1',100),
(null,'105','1988-7-1',100),
(null,'105','1988-8-1',100);
```

答案：

```
select distinct AccID from TestDB
```

```
where AccID not in
    (select TestDB.AccIDfrom TestDB,
         (select * from TestDB where AccID='101') as db101
    where TestDB.Occmonth=db101.Occmonth and TestDB.DebitOccur<=db101.DebitOccur
    );
```

19. 统计每年每月的信息

```
year    month amount
1991     1         1.1
1991     2         1.2
1991     3         1.3
1991     4         1.4
1992     1         2.1
1992     2         2.2
1992     3         2.3
1992     4         2.4
```

查成这样一个结果

```
year m1    m2    m3    m4
1991 1.1 1.2 1.3 1.4
1992 2.1 2.2 2.3 2.4
```

提示：这个与工资条非常类似，与学生的科目成绩也很相似。

准备 sql 语句：

```
drop table if exists sales;
create table sales(id int auto_increment primary key,year varchar(10), month
varchar(10), amount float(2,1));
insert into sales values
(null,'1991','1',1.1),
(null,'1991','2',1.2),
(null,'1991','3',1.3),
(null,'1991','4',1.4),
(null,'1992','1',2.1),
(null,'1992','2',2.2),
(null,'1992','3',2.3),
(null,'1992','4',2.4);
```

答案一、

```
select sales.year ,
(select t.amount from sales t where t.month='1' and t.year= sales.year) '1',
(select t.amount from sales t where t.month='1' and t.year= sales.year) '2',
(select t.amount from sales t where t.month='1' and t.year= sales.year) '3',
(select t.amount from sales t where t.month='1' and t.year= sales.year) as '4'
from sales  group by year;
```

20. 显示文章标题，发帖人、最后回复时间

表：id,title,postuser,postdate,parentid

准备 sql 语句：

```
drop table if exists articles;
create table articles(id int auto_increment primary key,title varchar(50), postuser varchar(10), postdate datetime,parentid int references articles(id));
insert into articles values
(null,'第一条','张三','1998-10-10 12:32:32',null),
(null,'第二条','张三','1998-10-10 12:34:32',null),
(null,'第一条回复 1','李四','1998-10-10 12:35:32',1),
(null,'第二条回复 1','李四','1998-10-10 12:36:32',2),
(null,'第一条回复 2','王五','1998-10-10 12:37:32',1),
(null,'第一条回复 3','李四','1998-10-10 12:38:32',1),
(null,'第二条回复 2','李四','1998-10-10 12:39:32',2),
(null,'第一条回复 4','王五','1998-10-10 12:39:40',1);
```

答案：

```
select a.title,a.postuser,
    (select max(postdate) from articles where parentid=a.id) reply
from articles a where a.parentid is null;
```

21. 删除除了 id 号不同,其他都相同的学生冗余信息

2.学生表 如下:

id 号	学号	姓名	课程编号	课程名称	分数
1	2005001	张三	0001	数学	69
2	2005002	李四	0001	数学	89
3	2005001	张三	0001	数学	69

A: delete from tablename where id 号 not in(select min(id 号) from tablename group by 学号,姓名,课程编号,课程名称,分数)

实验：

```
create table student2(id int auto_increment primary key,code varchar(20),name varchar(20));
insert into student2 values(null,'2005001',' 张 三 '),(null,'2005002',' 李 四 '),(null,'2005001','张三');
```

//如下语句，mysql 报告错误，可能删除依赖后面统计语句，而删除又导致统计语句结果不一致。

```
delete from student2 where id not in(select min(id) from student2 group by name);
```

//但是，如下语句没有问题：

```
select * from student2 where id not in(select min(id) from student2 group by name);
```

//于是，我想先把分组的结果做成虚表，然后从虚表中选出结果，最后再将结果作为删除的条件数据。

```
delete from student2 where id not in(select mid from (select min(id) mid
from student2 group by name) as t);
```

或者：

```
delete from student2 where id not in(select min(id) from (select * from s
tudent2) as t group by t.name);
```

22. 航空网的几个航班查询题：

表结构如下：

```
flight{flightID,StartCityID ,endCityID,StartTime}
city{cityID, CityName)
```

实验环境：

```
create table city(cityID int auto_increment primary key,cityName varchar(20));
create table flight (flightID int auto_increment primary key,
    StartCityID int references city(cityID),
    endCityID  int references city(cityID),
    StartTime timestamp);
//航班本来应该没有日期部分才好，但是下面的题目当中涉及到了日期
insert into city values(null,'北京'),(null,'上海'),(null,'广州');
insert into flight values
    (null,1,2,'9:37:23'),(null,1,3,'9:37:23'),(null,1,2,'10:37:23'),(null,2,3,'
10:37:23');
```

1、查询起飞城市是北京的所有航班，按到达城市的名字排序

参与运算的列是我起码能够显示出来的那些列，但最终我不一定把它们显示出来。各个表组合出来的中间结果字段中必须包含所有运算的字段。

```
  select  * from flight f,city c
    where f.endcityid = c.cityid and startcityid =
    (select c1.cityid from city c1 where c1.cityname = "北京")
    order by c.cityname asc;
```

```
mysql> select flight.flightid,'北京' startcity, e.cityname from flight,city e wh
ere flight.endcityid=e.cityid and flight.startcityid=(select cityid from city wh
ere cityname='北京');
```

```
mysql> select flight.flightid,s.cityname,e.cityname from flight,city s,city e wh
ere flight.startcityid=s.cityid and s.cityname='北京' and flight.endCityId=e.cit
yID order by e.cityName desc;
```

2、查询北京到上海的所有航班纪录（起飞城市，到达城市，起飞时间，航班号）

```
select c1.CityName,c2.CityName,f.StartTime,f.flightID
from city c1,city c2,flight f
where f.StartCityID=c1.cityID
and f.endCityID=c2.cityID
and c1.cityName='北京'
and c2.cityName='上海'
```

3、查询具体某一天（2005-5-8）的北京到上海的的航班次数

```
select count(*) from
(select c1.CityName,c2.CityName,f.StartTime,f.flightID
from city c1,city c2,flight f
where f.StartCityID=c1.cityID
and f.endCityID=c2.cityID
and c1.cityName='北京'
and c2.cityName='上海'
and 查帮助获得的某个日期处理函数(startTime) like '2005-5-8%'
```

mysql 中提取日期部分进行比较的示例代码如下：

```
select * from flight where date_format(starttime,'%Y-%m-%d')='1998-01-02'
```

23. **查出比经理薪水还高的员工信息：**

```
select e.* from employees e,employees m where e.managerid=m.id and e.sala
ry>m.salary;
```

24. 求出小于 45 岁的各个老师所带的大于 12 岁的学生人数

数据库中有 3 个表 teacher 表，student 表，tea_stu 关系表。

teacher 表 teaID name age

student 表 stuID name age

teacher_student 表 teaID stuID

要求用一条 sql 查询出这样的结果

1.显示的字段要有老师 name, age 每个老师所带的学生人数

2 只列出老师 age 为 40 以下，学生 age 为 12 以上的记录

预备知识：

1.sql 语句是对每一条记录依次处理，条件为真则执行动作（select,insert,delete,update）

2.只要是迪卡尔积，就会产生“垃圾”信息，所以，只要迪卡尔积了，我们首先就要想到清除“垃圾”信息

实验准备：

```
drop table if exists tea_stu;
drop table if exists teacher;
drop table if exists student;
create table teacher(teaID int primary key,name varchar(50),age int);
create table student(stuID int primary key,name varchar(50),age int);
create table tea_stu(teaID int references teacher(teaID),stuID int
```

references student(stuID));
insert into teacher values(1,'zxx',45), (2,'lhm',25) , (3,'wzg',26) , (4,'tg',27);
insert into student values(1,'wy',11), (2,'dh',25) , (3,'ysq',26) , (4,'mxc',27);
insert into tea_stu values(1,1), (1,2), (1,3);
insert into tea_stu values(2,2), (2,3), (2,4);
insert into tea_stu values(3,3), (3,4), (3,1);
insert into tea_stu values(4,4), (4,1), (4,2) , (4,3);

结果：2→3,3→2,4→3

解题思路：
1 要会统计分组信息，统计信息放在中间表中：
select teaid,count(*) from tea_stu group by teaid;

2 接着其实应该是筛除掉小于 12 岁的学生，然后再进行统计，中间表必须与 student 关联才能得到 12 岁以下学生和把该学生记录从中间表中剔除，代码是：
select tea_stu.teaid,count(*) total from student,tea_stu
where student.stuid=tea_stu.stuid and student.age>12 group by tea_stu.teaid

3.接着把上面的结果做成虚表与 teacher 进行关联，并筛除大于 45 的老师
select teacher.teaid,teacher.name,total from teacher ,*(select tea_stu.tea*
id,count(*) total from student,tea_stu where student.stuid=tea_stu.stuid and stu
dent.age>12 group by tea_stu.teaid) as tea_stu2 where teacher.teaid=tea_stu2.tea
id and teacher.age<45;

25. 求出发帖最多的人：

```
select authorid,count(*) total from articles
group by authorid
having total=
    (select max(total2) from (select count(*) total2 from articles group by authorid)
as t);

select t.authorid,max(t.total) from
     (select authorid,count(*) total from articles ) as t
```

这条语句不行，因为 max 只有一列，不能与其他列混淆。

```
select authorid,count(*) total from articles
group by authorid having total=max(total)
```

也不行。

26. 一个用户表中有一个积分字段，假如数据库中有 100 多万个用户，若要在每年第一天凌晨将积分清零，你将考虑什么，你将想什么办法解决?

```
alter table drop column score;
alter table add colunm score int;
```

可能会很快，但是需要试验，试验不能拿真实的环境来操刀，并且要注意，
这样的操作时无法回滚的，在我的印象中，只有 inert update delete 等 DML 语句才能回滚，对于 create table,drop table ,alter table 等 DDL 语句是不能回滚。
解决方案一，update user set score=0;
解决方案二，假设上面的代码要执行好长时间，超出我们的容忍范围，那我就 alter table user drop column score;alter table user add column score int。

下面代码实现每年的那个凌晨时刻进行清零。

```
Runnable runnable =
    new Runnable(){
        public void run(){
            clearDb();
            schedule(this,new Date(new Date().getYear()+1,0,0));
            }
            };

schedule(runnable,
    new Date(new Date().getYear()+1,0,1));
```

27. 一个用户具有多个角色，请查询出该表中具有该用户的所有角色的其他用户。

```
select count(*) as num,tb.id
from
 tb,
 (select role from tb where id=xxx) as t1
where
 tb.role = t1.role and tb.id != t1.id
group by tb.id
having
    num = select count(role) from tb where id=xxx;
```

28. xxx 公司的 sql 面试

```
Table EMPLOYEES Structure:
  EMPLOYEE_ID      NUMBER        Primary Key,
  FIRST_NAME       VARCHAR2(25),
  LAST_NAME       VARCHAR2(25),
  Salary number(8,2),
  HiredDate DATE,
  Departmentid number(2)
Table Departments Structure:
  Departmentid number(2)        Primary Key,
  DepartmentName  VARCHAR2(25).
```

（2）基于上述 EMPLOYEES 表写出查询：写出雇用日期在今年的，或者工资在[1000,2000]

之间的，或者员工姓名（last_name）以’Obama’打头的所有员工，列出这些员工的全部个人信息。（4 分）

```
select * from employees
where Year(hiredDate) = Year(date())
    or (salary between 1000 and 200)
    or left(last_name,3)='abc';
```

(3) 基于上述 EMPLOYEES 表写出查询：查出部门平均工资大于 1800 元的部门的所有员工，列出这些员工的全部个人信息。（4 分）

```
mysql> select id,name,salary,deptid did from employee1 where (select avg(salary)
 from employee1 where deptid = did) > 1800;
```

(4) 基于上述 EMPLOYEES 表写出查询：查出个人工资高于其所在部门平均工资的员工，列出这些员工的全部个人信息及该员工工资高出部门平均工资百分比。（5 分）

```
select employee1.*,(employee1.salary-t.avgSalary)*100/employee1.salary
from employee1,
    (select deptid,avg(salary) avgSalary from employee1 group by deptid) as t
where employee1.deptid = t.deptid and employee1.salary>t.avgSalary;
```

29. 大数据量下的分页解决方法。【齐谷网络】

答：最好的办法是利用 sql 语句进行分页，这样每次查询出的结果集中就只包含某页的数据内容。再 sql 语句无法实现分页的情况下，可以考虑对大的结果集通过游标定位方式来获取某页的数据。

sql 语句分页，不同的数据库下的分页方案各不一样，下面是主流的三种数据库的分页 sql：

sql server:

```
    String sql =
    "select top " + pageSize + " * from students where id not in" +
 "(select top " + pageSize * (pageNumber-1) + " id from students order by id)" +
 "order by id";
```

mysql:

```
    String sql =
    "select * from students order by id limit " + pageSize*(pageNumber-1) + "," +
pageSize;
```

oracle:

```
    String sql =
     "select * from " +
     (select *,rownum rid from (select * from students order by postime desc) where
rid<=" + pagesize*pagenumber + ") as t" +
     "where t>" + pageSize*(pageNumber-1);
```

30. 写一个用 jdbc 连接并访问 oracle 数据的程序代码

```
private final   static String driverString = "oracle.jdbc.driver.OracleDriver";
    private final static   String url="jdbc:oracle:thin:@localhost:1521:orcl";
    public static Connection getOracleDB()
    {
        Connection conn=null;
```

```
        try {
            Class.forName(driverString);
            conn= DriverManager.getConnection(url,"scott","123");
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return conn;
    }
```

（五）JS\jQuery\XML 部分

1、 JavaScript 有哪几种数据类型

简单：Number，Boolean，String，Null，Undefined

复合：Object，Array，Function

2、 在 js 编码中 innerHTML,outhtml,innertext 区别

- ✓ innerHTML 设置或获取位于对象起始和结束标签内的 HTML
- ✓ outerHTML 设置或获取对象及其内容的 HTML 形式
- ✓ innerText 设置或获取位于对象起始和结束标签内的文本
- ✓ outerText 设置(包括标签)或获取(不包括标签)对象的文本

3、 写一个简单的 json 对象？

```
var json_ =
{
    stuId : 2012002,
    stuName: ‘李华明’,
    stuAge:20,
    stuSex:’男’
}
```

如果一个 JavsScript 对象转 JSON 可以通过 eval()函数转换成 JSON 对象进行访问。

4、 什么是 json ，jquery?

JSON: (javaScript Object Notation)是一种轻量级的数据交换格式。

JSON 两种结构：

名称/值对的集合，不同的语言中，它被理解为对象，记录，结构，字典，哈希表，有键列表，关联数组。

值的有序列表，数组

jQuery:

jQuery 由美国人 John Resig 创建　是一个优秀的 javascript 框架　使用户能够方便的处理 HTML documents　events　实现动画效果，并且方便地为网站提供 AJAX 交互。

5、 jQuery 中有 id 为 foo 的对象有 att 属性,如何获取 att 属性的值？

$(‘foo’).att(‘attr’).val()

6、 jQuery 中添加文本怎么写在 div 中

【参考】

可以通过 jQuery 提供的元素选择器或 ID 选择器为实现，如：

$(‘div’).append(‘Hello ’);要求只能有一个 div 否则$(‘div’)返回的是数组

$(‘#ID 名称’).append(“hello”);

7、 你在公司是怎么用 jquery 的？

【参考】

在项目中是怎么用的是看看你有没有项目经验(根据自己的实际情况来回答)你用过的选择器啊，复选框啊，表单啊，ajax 啊，事件等。配置 Jquery 环境 下载 jquery 类库 在 jsp 页面引用 jquery 类库即可

<script type="text/javascript" src="jquery/jquery-1.7.2.min.js"/>

接下来通过在<script>

$(function(){

);

</script>

8、 你为什么要使用 jquery？

【参考】

答：因为 jQuery 是轻量级的框架，大小不到 30kb,它有强大的选择器，出色的 DOM 操作的封装，有可靠的事件处理机制(jQuery 在处理事件绑定的时候相当的可靠)，完善的 ajax(它的 ajax 封装的非常的好，不需要考虑复杂浏览器的兼容性和 XMLHttpRequest 对象的创建和使用的问题。) 出色的浏览器的兼容性。 而且支持链式操作，隐式迭代。行为层和结构层的分离，还支持丰富的插件，jquery 的文档也非常的丰富。

9、 你觉得 jquery 有哪些好处？

答案同上

10、你使用 jquery 遇到过哪些问题，你是怎么解决的？

【参考】

答：这个答案是开发的，看你是否有相关的项目经验。例 前台拿不到值，JSON 可是出现的错误(多了一个空格等)这编译是不会报错的 jquery 库与其他库冲突：

1>如果其他库在 jquery 库之前导入的话

① 我们可以通过 jquery.noconflict()将变量的$的控制权过度给其他库

② 自定义快捷键,用一个变量接住 jquery.noconflict()

③ 通过函数传参

2>如果 jquery 库在其他库之前导入就直接使用 jquery 今天在处理一个数据问题时，发现 jQuery.ajax()方法返回的值一直有问题，清除缓存后数据无误，多次测试后发现返回的值都是之前的值，并且一直未执行 url(后台为 JAVA，设置断点一直未进入)。在网上查找下,发现是未设置 type 的原因。如果没设置 jQuery.ajax 的 type="Post"，那么 ajax 就会默认 type="Get"，这就会导致之前数据被缓存起来。加上 type="Post"，问题解决！

11、你知道 jquery 中的选择器吗，请讲一下有哪些选择器？

【参考】

jQuery 中的选择器大致分为:基本选择器，层次选择器，过滤选择器，表单选择器

12、jquery 中的选择器 和 css 中的选择器有区别吗？

【参考】

jQuery 选择器支持 CSS 里的选择器，jQuery 选择器可用来添加样式和添加相应的行为 CSS 中的选择器是只能添加相应的样式

13、你觉得 jquery 中的选择器有什么优势？

【参考】

简单的写法 $('ID') 来代替 document.getElementById()函数支持 CSS1 到 CSS3 选择器完

善的处理机制(就算写错了 id 也不会报错)

14、你在使用选择器的时候有有没有什么觉得要注意的地方？

【参考】

1) 选择器中含有".","#","[" 等特殊字符的时候需要进行转译
2) 属性选择器的引号问题
3) 选择器中含有空格的注意事项

15、jquery 对象和 dom 对象是怎样转换的？

【参考】

答 ：jquery 转 DOM 对象:jQuery 对象是一个数组对象，可以通过[index]的丰富得到相应的 DOM 对象还可以通过 get[index]去得到相应的 DOM 对象。DOM 对象转 jQuery 对象:$(DOM 对象)

16、你是如何使用 jquery 中的 ajax 的？

【参考】

如果是一些常规的 ajax 程序的话，使用 load(),$.get(),$.post(),就可以搞定了， 一般我会使用的是$.post() 方法。如果需要设定 beforeSend(提交前回调函数),error(失败后处理),success(成功后处理)及 complete(请求完成后处理)回调函数等，这个时候我会使用$.ajax()

17、你觉得 jquery 中的 ajax 好用吗，为什么？

【参考】

答: 好用的。因为 jQuery 提供了一些日常开发中夙瑶的快捷操作，例 load，ajax，get，post 等等，所以使用 jQuery 开发 ajax 将变得极其简单，我们就可以集中精力在业务和用户的体验上，不需要去理会那些繁琐的 XMLHttpRequest 对象了。

18、jquery 中$.get()提交和$.post()提交有区别吗？

1) $.get() 方法使用 GET 方法来进行异步请求的。 $.post() 方法使用 POST 方法来进行异步请求的。
2) get 请求会将参数跟在 URL 后进行传递，而 POST 请求则是作为 HTTP 消息的实体内容发送给 Web 服务器的，这种传递是对用户不可见的。
3) get 方式传输的数据大小不能超过 2KB 而 POST 要大的多
4) GET 方式请求的数据会被浏览器缓存起来，因此有安全问题。

19、jquery 中的 load 方法一般怎么用的？

【参考】

答：load 方法一般在 载入远程 HTML 代码并插入到 DOM 中的时候用通常用来从 Web 服务器上获取静态的数据文件。

如果要传递参数的话，可以使用$.get() 或 $.post()

20、在 jquery 中你是如何去操作样式的？

【参考】

addClass() 来追加样式

removeClass() 来删除样式

toggle() 来切换样式

21、简单的讲叙一下 jquery 是怎么处理事件的，你用过哪些事件？

【参考】

答: 首先去装载文档，在页面家在完毕后，浏览器会通过 javascript 为 DOM 元素添加事件。

22、你使用过 jquery 中的动画吗，是怎样用的？

【参考】

答:使用过。hide() 和 show() 同时修改多个样式属性。像高度，宽度，不透明度。fadeIn() 和 fadeOut() fadeTo() 只改变不透明度 slideUp() 和 slideDown() slideToggle() 只改变高度 animate() 属于自定义动画的方法。

23、你使用过 jquery 中的插件吗？

【参考】

答:看个人的实力和经验来回答了。

24、你一般用什么去提交数据，为什么？

【参考】

答:一般我会使用的是$.post() 方法。如果需要设定 beforeSend(提交前回调函数),error(失败后处理),success(成功后处理)及 complete(请求完成后处理)回调函数等，这个时候我会使用 $.ajax()

25、在 jquery 中引入 css 有几种方式？

【参考】

答:四种 行内式，内嵌式，导入式，链接式

26、你在 jquery 中使用过哪些插入节点的方法，它们的区别是什么？

【参考】

答:append(),appendTo(),prepend(),prependTo(),after(),insertAfter() before(),insertBefore() 大致可以分为 内部追加和外部追加

append() 表式向每个元素内部追加内容。

appendTo() 表示 讲所有的元素追加到指定的元素中。例$(A)appendTo(B) 是将 A 追加到 B 中 下面的方法解释类似

27、你使用过包裹节点的方法吗，包裹节点有方法有什么好处？

【参考】

答: wrapAll(),wrap(), wrapInner() 需要在文档中插入额外的结构化标记的时候可以使用这些包裹的方法应为它不会帛画原始文档的语义

28、jquery 中如何来获取或和设置属性？

【参考】

jQuery 中可以用 attr()方法来获取和设置元素属性

removeAttr() 方法来删除元素属性

29、如何来设置和获取 HTML 和文本的值？

【参考】

html()方法 类似于 innerHTML 属性 可以用来读取或者设置某个元素中的 HTML 内容 注意：html() 可以用于 xhtml 文档 不能用于 xml 文档

text() 类似于 innerText 属性 可以用来读取或设置某个元素中文本内容。
val() 可以用来设置和获取元素的值

30、你 jquery 中有哪些方法可以遍历节点？
【参考】
children() 取得匹配元素的子元素集合,只考虑子元素不考虑后代元素
next() 取得匹配元素后面紧邻的同辈元素
prev() 取得匹配元素前面紧邻的同辈元素
siblings() 取得匹配元素前后的所有同辈元素
closest() 取得最近的匹配元素 find() 取得匹配元素中的元素集合 包括子代和后代

31、子元素选择器 和后代选择器元素有什么区别？
【参考】
答:子代元素是找子节点下的所有元素,后代元素是找子节点或子节点的子节点中的元素

32、在 jquery 中可以替换节点吗？
【参考】
答：可以 在 jQuery 中有两者替换节点的方式 replaceWith() 和 replaceAll() 例如在 hao are you 替换成 I am fine
$('p').replaceWith('I am fine'); replaceAll 与 replaceWith 的用法前后调换一下即可。

33、你觉得 beforeSend 方法有什么用？
【参考】
答：发送请求前可以修改 XMLHttpRequest 对象的函数，在 beforeSend 中如果返回 false 可以取消本次的 Ajax 请求。XMLHttpRequest 对象是唯一的参数所以在这个方法里可以做验证

34、siblings() 方法 和 $('prev~div')选择器是一样的嘛？
【参考】
答: $('prev~div') 只能选择'#prev'元素后面的同辈 元素而 siblings()方法与前后的文职无关，只要是同辈节点就都能匹配。

35、你在 ajax 中使用过 JSON 吗，你是如何用的？
【参考】
答:使用过，在$.getJSON() 方法的时候就是。因为 $.getJSON() 就是用于加载 JSON 文件的

36、有哪些查询节点的选择器？
【参考】
答：我在公司使用过
:first 查询第一个，
:last 查询最后一个，
:odd 查询奇数但是索引从 0 开始
:even 查询偶数，

:eq(index)查询相等的 ，
:gt(index)查询大于 index 的 ，
:lt 查询小于 index
:header 选取所有的标题等

37、nextAll() 能 替代$('prev~siblindgs')选择器吗？

【参考】

答:能。 使用 nextAll() 和使用$('prev~siblindgs') 是一样的

38、jQuery 中有几种方法可以来设置和获取 样式

答 ： addClass() 方法，attr() 方法

39、 $(document).ready()方法和 window.onload 有什么区别？

【参考】

答: 两个方法有相似的功能，但是在实行时机方面是有区别的。

1) window.onload 方法是在网页中所有的元素(包括元素的所有关联文件)完全加载到浏览器后才执行的。
2) $(document).ready() 方法可以在 DOM 载入就绪时就对其进行操纵，并调用执行绑定的函数。

40、 jQuery 是如何处理缓存的？

【参考】

答 ： 要处理缓存就是禁用缓存.

1) 通过$.post() 方法来获取数据，那么默认就是禁用缓存的。
2) 通过$.get()方法 来获取数据，可以通过设置时间戳来避免缓存。可以在 URL 后面加上+(+new Date)例 $.get('ajax.xml?'+(+new Date),function () { //内容
});
3) 通过$.ajax 方法来获取数据，只要设置 cache:false 即可。

41、 $.getScript()方法 和 $.getJson() 方法有什么区别？

【参考】

1) $.getScript() 方法可以直接加载.js 文件，并且不需要对 javascript 文件进行处理 ，javascript 文件会自动执行。
2) $.getJson() 是用于加载 JSON 文件的 ，用法和$.getScript()

42、你读过哪些有关于 jQuery 的书吗？

【参考】

《jquery 基础教程》 《jquery 实战》 《锋利的 jquery》 《巧用 jquery》 《jQuery 用户界面库学习指南》等

43、$("#msg").text(); 和 $("#msg").text("new content");有什么区别？

【参考】

1 $("#msg").text() 是 返回 id 为 msg 的元素节点的文本内容

2 $("#msg").text("new content"); 是 将“new content” 作为普通文本串写入 id 为 msg 的元素节点内容中, 页面显示粗体的new content

44、radio 单选组的第二个元素为当前选中值，该怎么去取？

【参考】

答 : $('input[name=items]').get(1).checked = true;

45、选择器中 id，class 有什么区别？

【参考】

答：在网页中 每个 id 名称只能用一次，class 可以允许重复使用

46、你使用过哪些数据格式，它们各有什么特点？

【参考】

答: HTML 格式,JSON 格式,javascript 格式,XML 格式

1) HTML 片段提供外部数据一般来说是最简单的。
2) 如果数据需要重用，而且其他应用程序也可能一次受到影响，那么在性能和文件大小方面具有优势的 JSON 通常是不错的选择。
3) 而当远程应用程序未知时，XML 则能够为良好的互操作性提供最可靠的保证。

47、jQuery 能做什么？

【参考】

1) 获取页面的元素
2) 修改页面的外观
3) 改变页面大的内容
4) 响应用户的页面操作
5) 为页面添加动态效果
6) 无需刷新页面，即可以从服务器获取信息
7) 简化常见的 javascript 任务

48、在 ajax 中 data 主要有几种方式？

【参考】

三种，html 拼接的，json 数组，form 表单经 serialize()序列化的。

49、jQuery 中的 hover()和 toggle()有什么区别？

【参考】

hover()和 toggle()都是 jQuery 中两个合成事件。hover()方法用于模拟光标悬停事件。toggle()方法是连续点击事件。

50、你知道 jQuery 中的事件冒泡吗，它是怎么执行的，何如来停止冒泡事件？

【参考】

答 : 知道,事件冒泡是从里面的往外面开始触发。在 jQuery 中提供了 stopPropagation()方法可以停止冒泡。

51、例如 单击超链接后会自动跳转，单击"提交"按钮后表单会提交等，有时候我想阻止这些默认的行为，该怎么办？

【参考】

可以用 event.preventDefault()或 在事件处理函数中返回 false，即 return false;

52、jquery 表单提交前有几种校验方法？分别为？？

【参考】

formData:返回一个数组，可以通过循环调用来校验

jaForm：返回一个 jQuery 对象，所有需要先转换成 dom 对象

fieldValue：返回一个数组　　beforeSend()

53、在 jquery 中你有没有编写过插件，插件有什么好处？你编写过那些插件？它应该注意那些？

【参考】

插件的好处：对已有的一系列方法或函数的封装，以便在其他地方重新利用，方便后期维护和提高开发效率

插件的分类：封装对象方法插件 、封装全局函数插件、选择器插件

注意的地方：

1) 插件的文件名推荐命名为 jquery.[插件名].js，以免和其他的 javaScript 库插件混淆 。
2) 所有的对象方法都应当附加到 jQuery.fn 对象上，而所有的全局函数都应当附加到 jQuery 对象本身上。
3) 插件应该返回一个 jQuery 对象，以保证插件的可链式操作。
4) 避免在插件内部使用$作为 jQuery 对象的别名,而应使用完整的 jQuery 来表示，这样可以避免冲突或使用闭包来避免。
5) 所有的方法或函数插件，都应当一分好结尾，否则压缩的时候可能出现问题。在插件头部加上分号，这样可以避免他人的不规范代码给插件带来影响。
6) 在插件中通过$.extent({})封装全局函数,选择器插件，扩展已有的 object 对象
7) 通过$.fn.extend({})封装对象方法插件

54、怎样给 jquery 动态附加新的元素?那么怎样给新生成的元素绑定事件呢？

【参考】

jQuery 的 html()可以给现在元素附加新的元素 直接在元素还未生成前就绑定肯定是无效的，因为所绑定的元素目前根本不存在。所以我们可以通过 live 和 livequery 来动态绑定事件

55、IE 与 FireFox 脚本兼容性问题

(1) window.event：

表示当前的事件对象，IE 有这个对象，FF 没有，FF 通过给事件处理函数传递事件对象

(2) 获取事件源

IE 用 srcElement 获取事件源，而 FF 用 target 获取事件源

(3) 添加，去除事件

IE：element.attachEvent(“onclick”, function) element.detachEvent(“onclick”, function)

FF ： element.addEventListener(“click”, function, true) element.removeEventListener(“click”, function, true)

(4) 获取标签的自定义属性

IE：div1.value 或 div1[“value”]

FF：可用 div1.getAttribute(“value”)

(5) document.getElementByName()和 document.all[name]

IE；document.getElementByName()和 document.all[name]均不能获取 div 元素

FF：可以

(6) input.type 的属性
IE：input.type 只读
FF：input.type 可读写
(7) innerText textContent outerHTML
IE：支持 innerText, outerHTML
FF：支持 textContent
(8) 是否可用 id 代替 HTML 元素
IE：可以用 id 来代替 HTML 元素
FF：不可以
这里只列出了常见的，还有不少

56、css+div 的优势

Div+CSS 标准的优点:
1.大大缩减页面代码，提高页面浏览速度,缩减带宽成本;
2.结构清晰，容易被搜索引擎搜索到，天生优化了 seo
3.缩短改版时间。只要简单的修改几个 CSS 文件就可以重新设计一个有成百上千页面的站点。
4.强大的字体控制和排版能力。CSS 控制字体的能力比糟糕的 FONT 标签好多了，有了 CSS，我们不再需要用 FONT 标签或者透明的 1 px GIF 图片来控制标题，改变字体颜色，字体样式等等。
5.CSS 非常容易编写。你可以象写 html 代码一样轻松地编写 CSS。
6.提高易用性。使用 CSS 可以结构化 HTML
7.可以一次设计，随处发布。
更好的控制页面布局

57、DIV 和 Span 的区别?

DIV(division)是一个块级元素，可以包含段落、标题、表格，乃至诸如章节、摘要和备注等。而 SPAN 是行内元素，SPAN 的前后是不会换行的，它没有结构的意义，纯粹是应用样式，当其他行内元素都不合适时，可以使用 SPAN
在 HTML 视图中工作时，可以在 <DIV> 内编辑文本，将某些字包含在 <SPAN> 元素内，以强调那些字。与 <DIV> 不同，<SPAN> 和它周围的文本一起移动

58、css 是什么

层叠样式表，用来进行页面样式设计，美化页面显示。

59、xml 有哪些解析技术，有什么区别？

有 DOM,SAX,STAX 等
DOM:处理大型文件时其性能下降的非常厉害。这个问题是由 DOM 的树结构所造成的，这种结构占用的内存较多，而且 DOM 必须在解析文件之前把整个文档装入内存,适合对 XML 的随机访问 SAX:不现于 DOM,SAX 是事件驱动型的 XML 解析方式。它顺序读取 XML 文件，不需要一次全部装载整个文件。当遇到像文件开头，文档结束，或者标签开头与标签结束时，它会触发一个事件，用户通过在其回调事件中写入处理代码来处理 XML 文件，适合对 XML 的顺序访

问

60、ajax 的工作原理？

Ajax 基本上就是把 JavaScript 技术和 XMLHttpRequest 对象放在 Web 表单和服务器之间。当用户填写表单时，数据发送给一些 JavaScript 代码而不是 直接发送给服务器。相反，JavaScript 代码捕获表单数据并向服务器发送请求。同时用户屏幕上的表单也不会闪烁、消失或延迟。换句话说，JavaScript 代码在幕后发送请求，用户甚至不知道请求的发出。更好的是，请求是异步发送的，就是说 JavaScript 代码（和用户）不用等待服务器的响应。因此用户可以继续输入数据、滚动屏幕和使用应用程序。

然后，服务器将数据返回 JavaScript 代码（仍然在 Web 表单中），后者决定如何处理这些数据。它可以迅速更新表单数据，让人感觉应用程序是立即完成的，表单没有提交或刷新而用户得到了新数据。JavaScript 代码甚至可以对收到的数据执行某种计算，再发送另一个请求，完全不需要用户干预！这就是 XMLHttpRequest 的强大之处。它可以根据需要自行与服务器进行交互，用户甚至可以完全不知道幕后发生的一切。结果就是类似于桌面应用程序的动态、快速响应、高交互性的体验。

61、HTML 的 form 提交之前如何验证数值文本框的内容全部为数字？否则的话提示用户并终止提交?

```
<form onsubmit=' return chkForm(this)' >
<input type="text" name="d1"/>
<input type="submit"/>
</form>
<script type=" text/javascript" />
function chkForm(this)
    {
        var value = thist.d1.value;
        var len = value.length;
        for(var i=0;i<len;i++)
        {
            if(value.charAt(i)>"9" || value.charAt(i)<"0")
            {
                alert("含有非数字字符");
                return false;
            }
        }
        return true;
    }
</script>
```

62、请写出用于校验 HTML 文本框中输入的内容全部为数字的 javascript 代码

```
   <input type="text" id="d1" onblur=" chkNumber (this)"/>
<script type=" text/javascript" />
```

```
function chkNumber(eleText)

    {
        var value = eleText.value;
        var len = value.length;
        for(var i=0;i<len;i++)
        {
            if(value.charAt(i)>"9" || value.charAt(i)<"0")
            {
                alert("含有非数字字符");

                eleText.focus();
                break;
            }
        }
    }
</script>
```

除了写完代码，还应该在网页上写出实验步骤和在代码中加入实现思路，让面试官一看就明白你的意图和检查你的结果。

63、xml 有哪些解析技术?区别是什么？【北京科瑞明】

答:

有 DOM,SAX,STAX 等

DOM:处理大型文件时其性能下降的非常厉害。这个问题是由 DOM 的树结构所造成的，这种结构占用的内存较多，而且 DOM 必须在解析文件之前把整个文档装入内存,适合对 XML 的随机访问 SAX:不现于 DOM,SAX 是事件驱动型的 XML 解析方式。它顺序读取 XML 文件，不需要一次全部装载整个文件。当遇到像文件开头，文档结束，或者标签开头与标签结束时，它会触发一个事件，用户通过在其回调事件中写入处理代码来处理 XML 文件，适合对 XML 的顺序访问

STAX:Streaming API for XML (StAX)

讲解这些区别是不需要特别去比较

64、在 javascript 中设置定时调用 myfun()函数(周期为 1 秒)的代码是？

setInterval(myfun(),1000)

65、XML 文档定义有几种形式？它们之间有何本质区别？解析 XML 文档有哪几种方式？

两种形式 dtd 、schema，

本质区别:schema 本身是 xml 的，可以被 XML 解析器解析(这也是从 DTD 上发展 schema 的根本目的)。

解析的中有：DOM、SAX、JDOM、DOM4J 等。

DOM:处理大型文件时其性能下降的非常厉害。这个问题是由 DOM 的树结构所造成的，这种结构占用的内存较多，而且 DOM 必须在解析文件之前把整个文档装入内存,适合对 XML 的随机访问

SAX:不现于 DOM,SAX 是事件驱动型的 XML 解析方式。它顺序读取 XML 文件，不需

要一次全部装载整个文件。当遇到像文件开头，文档结束，或者标签开头与标签结束时，它会触发一个事件，用户通过在其回调事件中写入处理代码来处理 XML 文件，适合对 XML 的顺序访问 。

66、HTTP 请求返回的状态码有哪些？分别有什么含义？

3XX：重定向，这类状态码代表需要客户端采取进一步的操作才能完成请求。通常，这些状态码用来重定向，后续的请求地址（重定向目标）在本次响应的 Location 域中指明。

302：请求的资源现在临时从不同的 URI 响应请求

303：对应当前请求的响应可以在另一个 URI 上被找到，而且客户端应当采用 GET 的方式访问那个资源。

4XX：请求错误，这类的状态码代表了客户端看起来可能发生了错误，妨碍了服务器的处理。

像，403、404、405 错误

5XX：服务器错误，这类状态码代表了服务器在处理请求的过程中有错误或者异常状态发生，也有可能是服务器意识到以当前的软硬件资源无法完成对请求的处理。

像，500、501、502 等错误

67、二叉树遍历有几种方法【中恒互联】

有 3 种方法。先序遍历、中序遍历、后序遍历

68、用 JavaScript 写一个小时钟，网页内容如下：

2013 年 12 月 23 日　12：23：30 每秒钟跳动一次

参考：

```
<script type="text/javascript">
document.write('<div id="time"></div>');
function showTime(){
var time = new Date();
document.getElementById("time").innerHTML=time.getFullYear()+"　　年
"+(time.getMonth()+1)+"　　月　　"+time.getDate()+"　　日
"+time.getHours()+":"+time.getMinutes()+":"+time.getSeconds();
}
setInterval(showTime,500);
</script>
```

69、JSON 和 XML 的优缺点

【参考答案】

1) 在可读性方面，JSON 和 XML 的数据可读性基本相同。JSON 和 XML 的可读性可谓不相上下，一边是建议的语法，一边是规范的标签形式，很难分出胜负。
2) 在可扩展性方面，XML 天生有很好的扩展性，JSON 当然也有，没有什么是 XML 能扩展，JSON 不能的。
3) 在编码难度方面，XML 有丰富的编码工具，比如 Dom4j、JDom 等，JSON 也有 json.org 提供的工具，但是 JSON 的编码明显比 XML 容易许多，即使不借助工具也能写出 JSON 的代码，可是要写好 XML 就不太容易了。
4) 在解码难度方面，XML 的解析得考虑子节点父节点，让人头昏眼花，而 JSON 的解析难度几乎为0。这一点 XML 输的真是没话说。

5) 在流行度方面，XML 已经被业界广泛的使用，而 JSON 才刚刚开始，但是在 Ajax 这个特定的领域，未来的发展一定是 XML 让位于 JSON。到时 Ajax 应该变成 Ajaj(Asynchronous Javascript and JSON)了。
6) JSON 和 XML 同样拥有丰富的解析手段。
7) JSON 相对于 XML 来讲，数据的体积小。
8) JSON 与 JavaScript 的交互更加方便。
9) JSON 对数据的描述性比 XML 较差。
10) JSON 的速度要远远快于 XML。

（六）Struts、Spring、Hibernate、Mybatis 框技术

1. Struts2.0 有几种标签库

【参考答案】

UI 标签 、控制标签、数据标签、杂项标签

2. struts2 必备包有哪些？

【参考答案】

commons-fileupload-1.2.1.jar

freemarker-2.3.13.jar

ognl-2.6.11.jar

struts2-core-2.1.6.jar

xwork-2.1.2.jar

3. Hiberbate 优化方法有那些？

【参考答案】

1) 尽量使用 many-to-one，避免使用 one-to-many
2) 灵活使用单向 one-to-many
3) 不用一对一，使用多对一代替一对一
4) 配置对象缓存，不使用集合缓存
5) 一对多使用 Bag 多对一使用 Set
6) 继承使用显示多态 HQL:from object polymorphism="exlicit" 避免查处所有对象
7) 消除大表，使用二级缓存

4. 应用服务器与 Web Server 的区别

【参考答案】

Web 服务器(Web Server)

Web 服务器可以解析(handles)HTTP 协议。当 Web 服务器接收到一个 HTTP 请求，会返回一个 HTTP 响应，例如送回一个 HTML 页面。为了处理一个请求，Web 服务器可以响应一个静态页面或图片， 进行页面跳转，或者把动态响应的产生委托给一些其它的程序例如 CGI 脚本，JSP 脚本，servlets，ASP 脚本，服务器端 JavaScript，或者一些其它的服务器端技术。无论它们(译者 注:脚本)的目的如何，这些服务器端的程序通常产生一个 HTML 的响应来让浏览器可以浏览。

应用程序服务器

通过各种协议，可以包括 HTTP，把商业逻辑暴露给客户端应用程序。Web 服务器主要 是处理向浏览器发送 HTML 以供浏览，而应用程序服务器提供访问商业逻辑的途径以供客户端应用程序使用。应用程序使用此商业逻辑就象你调用对象的一个方法 (或过程语言中的一个函数)一样。

应用程序服务器的客户端(包含有图形用户界面(GUI)的)可能会运行在一台 PC、一个 Web 服务器或者甚至是其它的应用程序服务器上。在应用 程序服务器与其客户端之间来回穿梭的信息不仅仅局限于简单的显示标记。相反，这种信息就是程序逻辑。 正是由于这种逻辑取得了数据和方法调用的形式而不是静态 HTML，所以客户端才可以随心所欲的使用这种被暴露的商业逻辑。

在大多数情形下，应用程序服务器是通过组件的应用程序接口把商业逻辑暴露 (给

客户端应用程序)的，例如基于 J2EE 应用程序服务器的 EJB 组件模型。此外，应用程序服务器可以管理自己的资源，例如，安全，事务处理，资源池， 和消息。就象 Web 服务器一样，应用程序服务器配置了多种可扩展和容错技术。

5. 如何设置 Hibernate 二级缓存

【参考答案】

1、首先要打开二级缓存，在 hibernate.cfg.xml 中添加如下配置：

<property name="hibernate.cache.use_second_level_cache">true</property>

2、Hibernate 的二级缓存使用第三方的缓存工具来实现，所以我们需要指定 Hibernate 使用哪个缓存工具。如下配置指定 Hibernate 使用 EhCache 缓存工具。

<property

name="hibernate.cache.provider_class">org.hibernate.cache.EhCacheProvider

</property>

3、Hibernate 在默认情况下并不会对所有实体对象进行缓存，所以，我们需要指定缓存哪些对象，在实体对象的映射文件中（相应的<class>标签内部），添加如下配置：

<cache usage="read-only"/>

usage="read-only"是“只读”缓存策略。

6. ApplicationContext 和 BeanFactory 有什么区别？

【参考答案】

BeanFactory 实际上是实例化，配置和管理众多 bean 的容器。 这些 bean 通常会彼此合作，因而它们之间会产生依赖，而这种依赖只是体现在 Bean 与 Bean 之间的依赖 这些依赖关系可以通过配置来反映，而 **ApplicationContext beans** 包是 **BeanFactory** 的子类，提供了以编程的方式管理和操控 bean 的基本功能，而 context 包增加了 ApplicationContext，它以一种更加**面向框架的方式**增强了 BeanFactory 的功能，简单说他增强了**面向 Web 容器的功能**。ApplictionContext 完全采用声明式方式来使用容器，甚至不用去手工创建它，Web 应用的启动进程中用它启动 ApplicationContext。当然用编程的方式创建一个 ApplicationContext 对象可以有以下几种方式或实现：

1、ClassPathXmlApplicationContext:从类路径中的 XML 文件载入上下文定义信息，把上下文定义文件当作类路径资源。

2、FileSystemXmlApplicationContext:从文件系统中的 XML 文件载入上下文定义信息。

3、XmlWebApplicationContext：从 Web 系统中的 XMl 文件载入上下文信息。

7. Spring MVC 与 Struts2 MVC 的不同之处 ?【杭州网阙科技】

【参考答案】

1、请求处理机制：spring mvc 是基于方法的设计，而 sturts 是基于类，每次发一次请求都会实例一个 action，每个 action 都会被注入属性，而 spring 基于方法，粒度更细。

2、参数传递：struts 是在接受参数的时候，可以用属性来接受参数，这就说明参数是让多个方法共享的。

3、设计思想上：struts 更加符合 oop 的编程思想， spring 就比较谨慎，在 servlet 上扩展。

4、intercepter 的实现机制：struts 有以自己的 interceptor 机制，spring mvc 用的是独立的 AOP 方式。这样导致 struts 的配置文件量还是比 spring mvc 大，虽然 struts 的配置能继承，所以我觉得论使用上来讲，spring mvc 使用更加简洁，开发效率 Spring MVC 确实比 struts2 高。

8. Spring 使用了哪些设计模式，这样用有什么好处？

【参考答案】

最典型的像：工厂模式，Spring 的 IOC 容器就是一个大的 Bean 实例的工厂，负责 Bean 的周期管理。单例模式，这个和 Spring 的 IOC 一起的，既然是 IOC 是一个大工厂，那个 Bean 对象为减少内存开销就需要提供单例特征。适配器模式，在 Spring 的 AOP 编程中随处可见 Adapter 模式的应用。代理模式，为其它对象提供一种代理访问的机制。观察者模式，当对象发生变化时，其它对象需要得到相应更新，Spring 中应用也较为普遍。

【分析】

就说上面几个了，这种问题无法一追溯，把所有 Spring 用到的设计模式一一回答出来，重要的是你只需要让面试官觉得你对 Spring 有较深的理解和应用即可。

9. Hibernate 有哪几种查询数据的方法？

【参考答案】

hibernate 查询有三种方式：HQL 查询、结构化 SQL 查询、QBC 查询

10. 3 个框架在项目在项目当中的用，BaseDao 是用来做什么的。

【参考答案】

DAO 组件主要提供数据库访问操作，针对不同数据源表持久化操作进行了封装，这样可以提供其它层的访问接口，使得组件之间解耦。而 BaseDAO 是这些所有这些不同的持久化的 DAO 的公共 API 进行了封装，进一步抽象提取，使其它各组件 DAO 从 BaseDAO 中派生，增强系统的重用性、维护性、扩展性。

11. ThreadLocal 在项目中的实际意义？

【参考答案】

ThreadLocal 和其它同步机制相比从另一个角度来解决多线程的并发访问，它为每一个线程维护一个和该线程绑定的变量的副本，从而隔离了多个线程的数据，每一个线程都拥有自己的变量副本，从而也就没有必要对该变量进行同步了。还提供了线程安全的共享对象，在编写多线程代码时，可以把不安全的整个变量封装进 ThreadLocal

ThreadLocal 可以大量减少参数的传递，可以使代码简洁，但一个线程会绑定多个自己定义的局部对象，ThreadLocal 是抽象在线程上的对象创建工厂，目前的 Tomcat5 使用了线程池，一个线程处理一个 request，这样 ThreadLocal 对象可以抽象的绑定在 request 生命周期，不会存在线程危机，而且线程池也平衡了这些 ThreadLocal

12. Spring 对多种 ORM 框架提供了很好的支持，结合事务管理描述 spring 从哪些方面提供了对 Hibernate 的支持。

【参考答案】

1、SessionFactory 管理。使用 Spring 整合 Hibernate 时我们不需要 hibernate.cfg.xml 文件。首先，在 applicationContext.xml 中配置数据源（dataSource）bean 和 session 工厂（sessionFactory）bean。

2、持入层管理。Spring 提供了 HibernateTemplate，用于持久层访问，无需打开 Session 及关闭 Session。它只要获得 SessionFactory 的引用，将可以只读地打开 Session，并在持久化访问结束后关闭 Session，对持久层逻辑，通用的操作（如对数据库中数据的增，删，改，查）有 HibernateTemplate 完成。

3、对 DAO 的支持：通过继承 HibernateDaoSupport 实现 DAO 基本操作。

4、对事务支持：Spring 的声明式事务和编程式事务，很好的将持久化的操作纳入事务管理。

13. Hibernate 的 session.save()与 session.saveOrUpdate()的区别？

【参考答案】

1.save（）方法，调用 save 方法时，首先会在 session 缓存中查找保存对象如果实体对象已经处于 Persient 状态，直接返回；否则并将保存至数据库，对象变为持久状态。

2.saveOrUpdate（）方法：和 save 方法一样首先在 session 缓存中查找，判断对象是否为为保存状态，如果对象处于 Persient，不执行操作，处于 Transient 执行 save 操作，处于 Detached 调用 saveOrUpdate 将对象与 session 重新关联(简单的说就是该方法会先看该对象是否已经存在，如果已经存在就更新，否则新增保存)，如果此时 saveOrUpdate 的对象与另一个与 Session 关联的对象持有相同的持久化标识则抛出相同的标识符异常。

14. Hibernate 中 sedssion.get()与 session.load()的区别？

【参考答案】

Session.load/get 方法均可以根据指定的实体类和 id 从数据库读取记录，并返回与之对应的实体对象。其区别在于：

1) 如果未能发现符合条件的记录，get 方法返回 null，而 load 方法会抛出一个 ObjectNotFoundException。

2) load 支持延迟加载，get 不支持

3) load 方法可返回实体的代理类实例，而 get 方法永远直接返回实体类。

4) load 方法可以充分利用内部缓存和二级缓存中的现有数据，get 方法则仅仅在内部缓存中进行数据查找，如没有发现对应数据，将越过二级缓存，直接调用 SQL 完成数据读取。

15. annotation 的使用方法和用途主要分为几类？

【参考答案】

1、内建 Annotation——Java5.0 版在 java 语法中经常用到的内建 Annotation：

2、开发者自定义 Annotation：由开发者自定义 Annotation 类型；

3、使用第三方开发的 Annotation 类型

16. Struts2 是怎么实现 MVC 的？

【参考答案】

MVC 就是 model view controller，三种组件，Struts2 实现 Controller 主要由一系列的前端过滤器（Filter）实现 Controller。Model 层主要由 Struts2 的 Action 组件实现。View 层主要是由 Struts2 的标签库、Freemarker 等组件实现视图层

17. 介绍下 hibernate【上海数字政通】

【参考答案】

Hibernate 是一个开放源代码 Java 语言下的对象关系映射解决方案。它为面向对象的领域模型到传统的关系型数据库的映射，提供了一个使用方便的框架。Hibernate 也是目前 Java 开发中最为流行的数据库持久层框架。将软件开发人员从大量相同的数据持久层相关编程工作中解放出来，Hibernate 不仅负责从 Java 类到数据库表的映射（还包括从 Java 数据类型到 SQL 数据类型的映射），还提供了面向对象的数据查询检索机制，从而极大地缩短的手动处

理 SQL 和 JDBC 上的开发时间。

18. 在 Struts2 中,配置文件中通配符是怎么表示和使用的？使用通配符后有什么好处。【上海咯勤】

示例1：调用相同 Action 中的不同方法

```
<action name="*Action" class="Jcuckoo.LoginRegistAction" method="{1}">
            <result name="input">/login.jsp</result>
            <result name="error">/error.jsp</result>
            <result name="success">/welcome.jsp</result>
</action>
```

其中表达式{1}的值 name 属性值中第一个*的值。

如果用户请求的 URL 为 loginAction.action,则调用 Jcuckoo.LoginRegistAction 中的 login 方法;

如果用户请求的 URL 为 registerAction.action,则调用 Jcuckoo.LoginRegistAction 中的 register 方法

示例2：带转视图

```
<action name="crud_*" class="Jcuckoo.CrudAction" method="{1}">
            <result name="input">/input.jsp</result>
            <result>/{1}.jsp</result>
</action>
```

当处理结果是 input 时，会转到/input.jsp 页面

当处理结果是 success 时，

如果 crud_create.action，则会执行 Jcuckoo.CrudAction 中的 create 方法，并且跳转到/create.jsp;

如果 crud_delete.action，则会执行 Jcuckoo.CrudAction 中的 delete 方法，并且跳转到/delete.jsp;

优点：

使用通配符能规范命名格式，简洁配置文件，加速开发效率，也是 Struts 倡导的一种开发模式。

19. Hibernate 中的 HQL 和 criteria 的区别？

【参考答案】

1. QBC(Query by Criteria)查询对查询条件进行了面向对象封装，符合编程人员的思维方式;
2. HQL(Hibernate Query Language)查询提供了更加丰富的和灵活的查询特性，在涵盖 Criteria 查询的所有功能的前提下，提供了类似标准 SQL 语句的查询方式，同时也提供了更加面向对象的封装。

20. 介绍 hibernate 延持加载属性

【参考答案】

Hibernate 通过 lazy 来指定加载策略，一般值为 true 或 false,。设为 flase 表示立即加载 ，true 表过延迟加载。

21. 列举你接触过的框架，说明特点和原理

【参考答案】

Struts2 的特点：自己总结一下。

Hibernate 特点：

1. 基于 JDBC 的主流持久化框架，是一个优秀的 ORM 实现，对 JDBC 访问数据库的代码做了封装，大大简化了数据访问层繁琐的重复性代码。

2. hibernate 使用 Java 反射机制，而不是字节码增强程序来实现透明性。

3. hibernate 的性能非常好，因为它是个轻量级框架。映射的灵活性很出色。它支持各种关系数据库，从一对一到多对多的各种复杂关系。

Spring 特点：

Spring 框架的主要优势之一低侵入式的架构思想，实现了 IOC 容器。另外一个 AOP 的编程也在很多应用场合下地位较重。提供了对各种不同类型框架的支持，如：Web 容器、持入层、事务层、其它 J2EE 组件等。

22. 解释一下 IOC,以及 spring 的举例

【参考答案】

IOC 称为控制反转，也叫依赖注入，ioc 是 Spring 的核心组件，它通过配置文件，将需要创建的对象以池的方式管理，将实例注入到需要的对象中区，是对象依赖于注入而不依赖于实现，解决了各个组件的耦合度，使得项目在后期的维护和扩展上非常方便。 如在 ssh 框架整合中，我们将 datasource 对象注入给 sessionFactory，再将 sessionFactory 注入给 dao 组件，再将 dao 组件注入给 struts 的 Action 组件，在将 action 对象注入给 struts 的拦截器。

23. 解释一下 mvc 以及熟悉的 mvc 框架

【参考答案】

MVC 是 Model－View－Controller 的简写。Model 代表的是应用的业务逻辑（通过 JavaBean，EJB 组件实现）， View 是应用的表示层（由 JSP、HTML、各种 Taglib 等组成），Controller 是提供应用程序的中心控制处理。通过这种设计模型把应用逻辑，处理过程和显示逻辑分成不同的组件实现，这些组件可以进行交互和重用，另外有利于维护。

Struts1、Struts2、Spring MVC、WebWork 等这些都是属于基于 MVC 模式实现的框架

24. Spring 工作原理

【参考答案】

内部最核心的就是 IOC 了，动态注入，让一个对象的创建不用 new 了，可以自动的生产，这其实就是利用 java 里的反射,反射其实就是在运行时动态的去创建、调用对象，Spring 就是在运行时，跟 xml Spring 的配置文件来动态的创建对象，和调用对象里的方法的 还有一个核心就是 AOP 这个就是面向切面编程，可以为某一类对象 进行监督和控制（也就是在调用这类对象的具体方法的前后去调用你指定的 模块)从而达到对一个模块扩充的功能。这些都是通过配置类达到的

Spring 目的：

就是让对象与对象（模块与模块）之间的关系没有通过代码来关联，都是通过配置类说明 管理的（Spring 根据这些配置 内部通过反射去动态的组装对象）要记住：Spring 是一个容器，凡是在容器里的对象才会有 Spring 所提供的这些服务和功能

25. Hibernate 中离线查询与在线查询的区别

【参考答案】

Criteria 和 DetachedCriteria 的主要区别在于创建的形式不一样， Criteria 是在线的，所以它是由 Hibernate Session 进行创建的；而 DetachedCriteria 是离线的，创建时无需 Session，DetachedCriteria 提供了 2 个静态方法 forClass(Class) 或 forEntityName(Name) 进行 DetachedCriteria 实例的创建。

26. Struts2 实现拦截器的原理？

【参考答案】

拦截器是 AOP 中的概念，它本身是一段代码，可以通过定义“织入点”，来指定拦截器的代码在“织入点”的前后执行，从而起到拦截的作用。而 Struts2 的 Interceptor，其拦截的对象是 Action 代码，可以定义在 Action 代码之前或者之后执行拦截器的代码。

1. 整个结构就如同一个堆栈，除了 Action 以外，堆栈中的其他元素是 Interceptor

2. Action 位于堆栈的底部。由于堆栈"先进后出"的特性，而这些都是围绕着 Action 的，当我们请求 Action 时，必须首先把位于 Action 上端的 Interceptor 拿出来执行。

27. Struts2 的实现原理。【高达软件】

【参考答案】

1、客户端初始化一个指向 Servlet 容器（例如 Tomcat）的请求

2、这个请求经过一系列的过滤器（Filter）（这些过滤器中有一个叫做 ActionContextCleanUp 的可选过滤器，这个过滤器对于 Struts2 和其他框架的集成很有帮助，例如：SiteMesh Plugin）;

3、接着 FilterDispatcher 被调用，FilterDispatcher 询问 ActionMapper 来决定这个请求是否需要调用某个 Action;

4、如果 ActionMapper 决定需要调用某个 Action，FilterDispatcher 把请求的处理交给 ActionProxy;

5、ActionProxy 通过 Configuration Manager 询问框架的配置文件，找到需要调用的 Action 类;

6、ActionProxy 创建一个 ActionInvocation 的实例。

7、ActionInvocation 实例使用命名模式来调用，在调用 Action 的过程前后，涉及到相关拦截器（Intercepter）的调用。

8、一旦 Action 执行完毕，ActionInvocation 负责根据 struts.xml 中的配置找到对应的返回结果。返回结果通常是（但不总是，也可能是另外的一个 Action 链）一个需要被表示的 JSP 或者 FreeMarker 的模版。在表示的过程中可以使用 Struts2 框架中继承的标签。在这个过程中需要涉及到 ActionMapper

28. 简述 spring 的事务传播行为和 隔离级别

【参考答案】

spring 的事务传播行为：

Spring 在 TransactionDefinition 接口中规定了 7 种类型的事务传播行为，它们规定了事务方法和事务方法发生嵌套调用时事务如何进行传播：

PROPAGATION_REQUIRED：如果当前没有事务，就新建一个事务，如果已经存在一个事务中，加入到这个事务中。这是最常见的选择。

PROPAGATION_SUPPORTS：支持当前事务，如果当前没有事务，就以非事务方式执行。

PROPAGATION_MANDATORY：使用当前的事务，如果当前没有事务，就抛出异常。

PROPAGATION_REQUIRES_NEW：新建事务，如果当前存在事务，把当前事务挂起。

PROPAGATION_NOT_SUPPORTED：以非事务方式执行操作，如果当前存在事务，就把当前事务挂起。

PROPAGATION_NEVER：以非事务方式执行，如果当前存在事务，则抛出异常。

PROPAGATION_NESTED：如果当前存在事务，则在嵌套事务内执行。如果当前没有事务，则执行与 PROPAGATION_REQUIRED 类似的操作。

Spring 的隔离级别

1、Serializable：最严格的级别，事务串行执行，资源消耗最大；

2、REPEATABLE READ：保证了一个事务不会修改已经由另一个事务读取但未提交（回滚）的数据。避免了“脏读取”和“不可重复读取”的情况，但是带来了更多的性能损失。

3、READ COMMITTED:大多数主流数据库的默认事务等级，保证了一个事务不会读到另一个并行事务已修改但未提交的数据，避免了“脏读取”。该级别适用于大多数系统。

4、Read Uncommitted：保证了读取过程中不会读取到非法数据。

29. Ibatis 框架和 Hibernate 框架各有什么特点？

【参考答案】

比较方面	Ibatis 框架	Hibernate 框架
从设计思路来看	从关系型开始到对象型的思路来解决数据库的操作问题	从对象的角度的思路来解决数据库的操作问题
从方便性上来看	半自动	全自动
从 sql 语句书写来看	要写 sql 语句	一般不用写，但是有时候我们还是要用书写 hql 语句的
从映射角度来看	映射 sql 语句的输入输出参数	对数据库表结构来进行映射

30. 为什么要用 ORM? 和 JDBC 有何不一样?

【参考答案】

orm 是一种思想，就是把 object 对象转变成数据库中的记录，或者把数据库中的记录转变成 object 对象，我们可以用 jdbc 来实现这种思想，orm 的思想本质就是建立是 JDBC 上，对 JDBC 的一种封装，这种封装可以省去了直接使用 jdbc 的繁琐细节，提高了开发效率，现在用的较多的 ORM 工具很多，一般我们公司采用的 ORM 框架主要有 hibernate 和 MyBatis。当然也听说一些其他 orm 工具，如 toplink,ojb 等。

31. 谈谈你对 Spring AOP 思想的理解。

【参考答案】

AOP（Aspect-OrientedProgramming，面向方面编程），可以说是 OOP（Object-Oriented Programing，面向对象编程）的补充和完善。OOP 引入封装、继承和多态性等概念来建立一种对象层次结构，用以模拟公共行为的一个集合。当我们需要为分散的对象引入公共行为的时候，OOP 则显得无能为力。也就是说，OOP 允许你定义从上到下的关系，但并不适合定义从左到右的关系。例如日志功能，日志代码往往水平地散布在所有对象层次中，而与它所散布到的对象的核心功能毫无关系。对于其他类型的代码，如安全性、异常处理和透明的持续性也是如此。这种散布在各处的无关的代码被称为横切（cross-cutting）代码，在 OOP 设计中，它导致了大量代码的重复，而不利于各个模块的重用。而 AOP 技术则恰恰相反，它利用一种称为“横切”的技术，剖解开封装的对象内部，并将那些影响了多个类的公共行为封

装到一个可重用模块，并将其名为“Aspect”，即方面。所谓“方面”，简单地说，就是将那些与业务无关，却为业务模块所共同调用的逻辑或责任封装起来，便于减少系统的重复代码，降低模块间的耦合度，并有利于未来的可操作性和可维护性。AOP 代表的是一个横向的关系，如果说“对象”是一个空心的圆柱体，其中封装的是对象的属性和行为；那么面向方面编程的方法，就仿佛一把利刃，将这些空心圆柱体剖开，以获得其内部的消息。而剖开的切面，也就是所谓的“方面”了。然后它又以巧夺天功的妙手将这些剖开的切面复原，不留痕迹。

使用“横切”技术，AOP 把软件系统分为两个部分：核心关注点和横切关注点。业务处理的主要流程是核心关注点，与之关系不大的部分是横切关注点。横切关注点的一个特点是，他们经常发生在核心关注点的多处，而各处都基本相似。比如权限认证、日志、事务处理。Aop 的作用在于分离系统中的各种关注点，将核心关注点和横切关注点分离开来。

实现 AOP 的技术，主要分为两大类：一是采用动态代理技术，利用截取消息的方式，对该消息进行装饰，以取代原有对象行为的执行；二是采用静态织入的方式，引入特定的语法创建“方面”，从而使得编译器可以在编译期间织入有关“方面”的代码。

32. 谈谈你对 Spring 的理解。

【参考答案】

1.Spring 实现了工厂模式的工厂类（在这里有必要解释清楚什么是工厂模式），这个类名为 BeanFactory（实际上是一个接口），在程序中通常 BeanFactory 的子类 ApplicationContext。Spring 相当于一个大的工厂类，在其配置文件中通过<bean>元素配置用于创建实例对象的类名和实例对象的属性。

2. Spring 提供了对 IOC 良好支持，IOC 是一种编程思想，是一种架构艺术，利用这种思想可以很好地实现模块之间的解耦。IOC 也称为 DI（Depency Injection）。

3. Spring 提供了对 AOP 技术的良好封装， AOP 称为面向切面编程，就是系统中有很多各不相干的类的方法，在这些众多方法中要加入某种系统功能的代码，例如，加入日志，加入权限判断，加入异常处理，这种应用称为 AOP。实现 AOP 功能采用的是代理技术，客户端程序不再调用目标，而调用代理类，代理类与目标类对外具有相同的方法声明，有两种方式可以实现相同的方法声明，一是实现相同的接口，二是作为目标的子类在，JDK 中采用 Proxy 类产生动态代理的方式为某个接口生成实现类，如果要为某个类生成子类，则可以用 CGLI B。在生成的代理类的方法中加入系统功能和调用目标类的相应方法，系统功能的代理以 Advice 对象进行提供，显然要创建出代理对象，至少需要目标类和 Advice 类。

33. 说说 struts1 与 struts2 的区别。【首都信息面试题】

【参考答案】

从以下几个方面来一一总结 Struts1与 Struts2的区别：

① **Servlet 依赖性**

由于 Action 在被调用时，**HttpServletRequest** 和 **HttpServletResponse** 被传递到 execute()方法中，Struts1中的 Action 对 Servlet 的 API 是有依赖性的.但如果在 Struts2 中，Action 就不会对容器有依赖了，因为 Action 是由简单的 POJO 组成的.在 Struts2 中，servlet 上下文以简单的 Map 的形式表现出来，这使得 Action 可以得到独立的测试.如果需要，Struts2也可以访问原始的请求与响应。不过，其他的框架元素减少或排除直接访问 HttpServetRequest 或 HttpServletResponse 的必要。

② **Action 类**

使用抽象类而不是接口设计是 Struts1设计上的问题,这已经在 Struts2中得到了解决.Struts1中的 Action 类需要继承框架中依赖的抽象基础类.但在 Struts2中,Action 类可能会也可能不会实现接口来启用可选的或者自定义的服务.在 Struts2中,Action 是不会依赖于容器的,因为它是由简单的POJO组成的.Struts2提供了一个基础的ActionSupport类来实现一些常用的接口。尽管这样,Action 接口仍然不是必须要继承的,任何含有 execute 方法的 POJO 对象都可以当作 Action 对象来用。

③ 验证

Struts1与 Struts2都支持通过 validate 方法的手动验证.Struts1使用 ActionForm 中的 validate 方法或者通过扩展 Commons Validator 来进行校验.然而,Struts2支持通过 Validate 方法与 Xwork 校验框架的手动验证. Xwork 框架支持验证链到子属性----使用为属性类文件和校验上下文定义的验证.

④ 线程模型

在 Struts1中,Action 资源一定是线程安全或者同步的.所以 Action 是 singletons 并且线程安全的,一定只有一个 Action 类的实例处理该 Action 的所有请求,singleton 策略限制了 Struts1的 Action 所能完成的,并且需要更加谨慎的开发.但是在 Struts2中,Action 对象对每一个请求都生成实例,所以在 Struts2中不存在线程安全的问题。

⑤ 易测性

测试 Struts1的程序会有些复杂.测试 Struts1 Action 的主要它依赖容器。但是在 Struts2中,Action 可以经由创建 Action 的实例,设置属性和调用方法来得到测试。Struts2中的测试是很简单的，不依赖于容器。

⑥ 获取输入

Struts1使用 ActionForm 来捕获输入，而且所有的 ActionForm 需要继承一个框架依赖的基类.由于 JavaBean 不能当作 ActionForm 来用,开发人员不得不创建冗繁的类来获取输入.不过 Struts2使用 Action 属性(例如输入属性不依赖于底层框架)这避免了需要创建第二个输入对象,从此冗繁减少了.此外在 Struts2中,Action 的属性可以通过标签在 web 页面中得到访问，POJO 表单对象和 POJO Action.甚至富对象类型,包括业务或域对象,都可以被当作输入/输出对象来使用。

⑦ 表达式语言

Struts1与 JSTL 整合,所以它使用 JSTL 表达式语言.Struts1的表达式语言含有遍历图表的基础对象,但是在集合和索引属性的支持上表现不好.Struts2同样支持 JSTL,但是它也支持一种更强大且灵活的表达式语言----“对象图标记语言”(OGNL)

⑧ 将绑定值到视图中

在视图层,Struts1使用标准的 JSP 来绑定对象(在模型层被处理的)到页面上下文来进行访问.然而 Struts2使用一种叫做值栈的技术,这使得标签可以访问值而不需将视图与正在呈递的对象类型连接起来.值栈允许重用一些属性名相同但类型不同的视图类型.

⑨ 类型转换

通常 Struts1的 ActionForm 属性都是 String 型的。Struts1使用 Commons-Beanutils 进行类型转换，这些针对每一个类的类型转换无法为每一个实例配置。然而 Struts2使用 OGNL 来进行类型转换.框架包含了针对基础类型,常见对象类型与原始类型的转换器。

⑩ Action 的生存周期

Struts1支持对每一个模块的请求处理器的分离(生命周期),但是同一模块下的所有 Action 必须共享相同的生命周期。Struts2支持通过拦截器栈为每一个 Action 创建不同的生命周期.自定义栈可以视需要对不同的 Action 使用.

34. 简述 Hibernate 和 JDBC 的优缺点？如何书写一个 one to many 配置文件

【参考答案】

1、封装了 jdbc，简化了很多重复性代码。

2、简化了 DAO 层编码工作，使开发更对象化了。

3、移植性好，支持各种数据库，如果换个数据库只要在配置文件中变换配置就可以了，不用改变 hibernate 代码。

4、支持透明持久化，因为 hibernate 操作的是纯粹的（pojo）java 类，没有实现任何接口，没有侵入性。

说明：

One to many 配置见下题

35. 写 Hibernate 的一对多和多对一双向关联的 orm 配置?

【参考答案】

配置 一对多

```
<!--            部门对员工的一对多配置-->
<set name="employees" inverse="true" cascade="delete">
                <key column="deptid"></key>
                <one-to-many class="com.tsinghua.manager.vo.EmployeeVO"/>
        </set>
```

多对一

```
<!-- 员工对部门的多对一配置 column 指定外键-->
<many-to-one name="department" class="com.tsinghua.manager.vo.DepartmentVO"
column="deptid"></many-to-one>
```

36. hibernate 的 inverse 属性的作用?

【参考答案】

在 Hibernate 中，术语 inverse 是反转的意思，在关联关系中，inverse="false"为主控方，由主控方负责维护对象的关联关系，如果设为主控方对象，主控对象更新，则负责更新另一方对象更新。

37. Spring 的依赖注入是什么意思？给一个 Bean 的 message 属性，字符串类型，注入值为 "Hello" 的 XML 配置文件该怎么写?

【参考答案】

依赖注入(Dependency Injection)和控制反转(Inversion of Control)是同一个概念。具体含义是：当某一个 Java 类，需要另一个 Java 类的协助时，在传统的程序设计过程中，通常由当前类（调用者）来创建被调用者的实例，然后使用被调用者的方法。但在 Spring 里，创建被调用者的工作不再由调用者来完成，而是由其它类（往往是工厂类）或容器（Spring IOC 容器）完成，当前调用者从其它类或容器中来获取被调用者的实例，这种方式称为控制反转; 创建被调用者实例的工作通常由 Spring 容器来完成，然后注入调用者，因此也称为依赖注入，这是 Spring 的一种编程思想的体现。

依赖注入在设计模式也体现得非常多，比如说工厂模式和构建模式，这种就是一个依赖注入的实现

```
<bean id="helloBean" class="com.spring.demo.HelloWorld">
        <property name="msg" value="Hello!"/>
 </bean>
```

38. JDO 是什么?

【参考答案】

JDO 是 Java 对象持久化的新的规范，为 java data object 的简称,也是一个用于存取某种数据仓库中的对象的标准化 API。JDO 提供了透明的对象存储，因此对开发人员来说，存储数据对象完全不需要额外的代码（如 JDBC API 的使用）。这些繁琐的例行工作已经转移到 JDO 产品提供商身上，使开发人员解脱出来，从而集中时间和精力在业务逻辑上。另外，JDO 很灵活，因为它可以在任何数据底层上运行。JDBC 只是面向关系数据库（RDBMS）JDO 更通用，提供到任何数据底层的存储功能，比如关系数据库、文件、XML 以及对象数据库（ODBMS）等等，使得应用可移植性更强。

39. Spring 与 EJB 的区别?

【参考答案】

① **提供商无关性**

EJB 3.0 是一个被设计为对提供商没有依赖性的开放的标准。EJB 3.0 规范由企业 JAVA 社区的主流开源组织和厂商共同编写和支持的。EJB 3.0 框架使开发者的应用程序实现可以独立于应用服务器。而 Spring 一直是一个非标准的技术，尽管你在任何应用服务器都上可以使用 Spring 框架，但基于 Spring 的应用仍然被限制于 Spring 本身和在你的应用中使用到的 Spring 提供的各种特别服务。

② **服务整合**

Spring 框架是建立在应用服务器和服务库之上，它的服务整合代码（如数据访问模板和 Helper 类）是基于框架的，并暴露给应用开发者。相反，EJB 3.0 框架是紧密整合到应用服务器中的，它的服务整合代码是封装在一个标准的接口下的。

③ **服务聚合的灵活性**

由于 Spring 中的服务整合代码是作为编程接口暴露给应用开发者的，因此开发人员可以根据需要来聚合多个服务。这个特性使你可以集成一个你自己的“轻量”级应用服务器。通常，EJB 3.0 应用服务器不提供给开发者这种按照你的需要来选择服务的灵活性。大多数情况，你会得到一系列已经预先打包好的特性，其中有些你可能是不需要的。

④ **声明式服务**

EJB 3.0 和 Spring 都将运行时服务（如事务管理、安全、日志、消息、和信息服务）连接给应用程序。由于这些服务同应用程序的业务逻辑并不是直接相关的，因此，它们不被应

用程序本身来管理。相反，这些服务被服务容器（如 EJB 3.0 和 Spring）以不可见的方式在运行时提供给应用程序。开发人员（或系统管理员）通过配置来告诉容器什么时候，以怎样的方式来应用这些服务。

⑤ **注射依赖**

Spring 和 EJB 3.0 都提供了大量的 DI 模式支持。但是，它们之间也有着根本的不同。Spring 支持了通常意义上的但是复杂的基于 XML 配置文件的注射依赖 API；EJB 3.0 支持的注射大多数通用服务对象（如，EJB 和容器对象）和 JNDI 对象，它通过简单的 JAVA 注解来完成。

40. SSH 框架的优缺点。【北京科瑞明面试题】

【参考答案】

先说说 Struts：

1．struts 第一个优点应该是实现了 MVC。

对 Servlet 依赖减少，低侵入式的设计

2．Action 线程安全的设计

3．功能强大的 OGNL 表达式使用。

4．支持多种复合视图，表现层的使用也多样化，像 JSP\freeMarker\Velocity。

5．拦截器的应用，实现了 AOP 的思想，方便重用与扩展

6．自动类型转换功能。

7．相对低粗度的数据验证功能

Struts 缺点：

Struts2 中 Action 中取得从 jsp 中传过来的参数时，如果页面过于复杂，会造成对象臃肿。

Spring 优点：

1、非常优秀的轻量级，低侵入式的框架。

2、IOC 的容器周期式的管理，降低组件的偶合。

3、对其它容器有非常好的支持 ，像对持久层的 Hibernate、Ibaits、TOP Link 等。

Spring 缺点：

1．Web 层的 MVC 框架单过于单薄，对页面框架的支持，跟其它框架还有很大的差距。

2．不是一站式解决方案。

3．使用了反射来管理其容器中的 bean，在程序中增大了内存和运行计算的时间。

4．部分组件功能还有待完善

Hibernate 的优点：

1、非常好的 ORM 的框架，在 MVC 的切分和 JDBC 的封装上做的很好。

缺点：

1、对复杂查询，多变的查询，完成起来有难度。

2、自动化程序高，改写灵活性不够。

2、缓存不是太高效，所以有些企业单独会选择缓存框架或者弃用 Hibernate 的原因之一。

41. Spring 有哪几种注入方式？

【参考答案】

3 种方法。构造属入、属性注入、接口注入

42. Hibernate 原理

【参考答案】

1.读取并解析配置文件

2.读取并解析映射信息，创建 SessionFactory

3.打开 Sesssion

4.创建事务 Transation

5.持久化操作

6.提交事务

7.关闭 Session

8.关闭 SesstionFactory

43. Spring MVC 工作机制及为什么要用?【杭州网阙科技】

【参考答案】

1、客户端所有的请求都提交给 DispatcherServlet,它会委托应用系统的其他模块负责负责对请求进行真正的处理工作。

2.DispatcherServlet 查询一个或多个 HandlerMapping,找到处理请求的 Controller.

3.DispatcherServlet 请请求提交到目标 Controller

4.Controller 进行业务逻辑处理后，会返回一个 ModelAndView

5.Dispathcher 查询一个或多个 ViewResolver 视图解析器,找到 ModelAndView 对象指定的视图对象

6.视图对象负责渲染返回给客户端。

44. spring 中常用的几种 advice

【参考答案】

BeforeAdvice （前织入）、AfterAdvice （后织入）、ArroundAdvice （前后织入）ThrowAdvice （异常织入）

45. Spring 中 bean 的配置 scope 表示什么含义？可以有哪几种取值。

【参考答案】

scope 表示 Bean 的生命周期或者叫 Bean 的作用域。scope 的值有两个：

1、singleton，为单例属性，即 Spring IoC 容器只会创建该 bean 的唯一一个实例，这也是默认的。

2、prototype 为原型属性，即每一次请求都会产生一个新的 bean 实例。

46. Hibernate 对象有几种状态？如何转换

【参考答案】

共 3 种状态，分别是：Transient 状态（瞬时）、Persient 状态（持久）、Detached（脱管状态）状态。

47. spring 中 OpenSessionInViewFilter 作用什么的，解决了什么问题

【参考答案】

解决 Hibernate 的 Session 的关闭与开启问题。

SSH 整合问题：Hibernate 允许对关联对象、属性进行延迟加载，但是必须保证延迟加载的操作限于同一个 Hibernate Session 范围之内进行。如果 Service 层返回一个启用了延迟加载功能的领域对象给 Web 层，当 Web 层访问到那些需要延迟加载的数据时，由于加

载领域对象的 Hibernate Session 已经关闭，这些导致延迟加载数据的访问异常。而 Spring 为我们提供的 OpenSessionInViewFilter 过滤器为我们很好的解决了这个问题。OpenSessionInViewFilter 的主要功能是用来把一个 Hibernate Session 和一次完整的请求过程对应的线程相绑定，允许在事务提交之后延迟加载显示所需要的对象。实现"Open Session in View"的模式 。

【补充说明】

配置如下：

```
<filter>
    <filter-name>Spring OpenSessionInViewFilter</filter-name>
<filter-class>org.springframework.orm.hibernate3.support.OpenSessionInViewFilter</filter-class>
    <init-param>
        <param-name>sessionFactoryBean</param-name>
        <param-value>sessionFactory</param-value>
    </init-param>
</filter>
<filter-mapping>
        <filter-name>Spring OpenSessionInViewFilter</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

一般在项目中 SSH 整合时，我们会加入上面的配置，解决延迟加载时的异常。

48. hibernate 有哪五个核心接口。

【参考答案】

Configuration 接口，SessionFactory 接口，Session 接口，Transaction 接口，Query 和 Criteria 接口

49. Hibernate 中有几种关系映射【嘉瑞互动】

【参考答案】

主要有单向一对一、单向一对多、单向多对一、单向多对多、双向一对一、双向一对多、双向多对多。

50. 介绍一下 Hibernate 的一级缓存与二级缓存。

缓存就是把以前从数据库中查询出来和使用过的对象保存在内存中(一个数据结构中)，这个数据结构通常是或类似 Hashmap，当以后要使用某个对象时，先查询缓存中是否有这个对象，如果有则使用缓存中的对象，如果没有则去查询数据库，并将查询出来的对象保存在缓存中，以便下次使用。Hibernate 的 Session 就是一种缓存，我们通常将之称为 Hibernate 的一级缓存，当想使用 session 从数据库中查询出一个对象时，Session 也是先从自己内部查看是否存在这个对象，存在则直接返回，不存在才去访问数据库，并将查询的结果保存在自己内部。由于 Session 代表一次会话过程，一个 Session 与一个数据库连接相关连，所以 Session 最好不要长时间保持打开，通常仅用于一个事务当中，在事务结束时就应关闭。并且 Session 是线程不安全的，被多个线程共享时容易出现问题。通常只有那种全局意义上的缓存才是真正的缓存应用，才有较大的缓存价值，因此，Hibernate 的 Session 这一级缓存的缓存作用并不明显，应用价值不大。

Hibernate 的二级缓存就是要为 Hibernate 配置一种全局缓存，让多个线程和多个事务都可以共享这个缓存。一般我们叫它 sessionFactory 缓存，也叫进程级的缓存，使用第 3 方

插件实现的，也值缓存实体，生命周期和 sessionFactory 一致，可以进行管理。

二级缓存具体实现：

首先配置第 3 放插件，我们用的是 EHCache，在 hibernate.cfg.xml 文件中加入
<propertyname="hibernate.cache.user_second_level_cache">true</property>
在映射中也要显示的调用，<cacheusage="read-only"/>
二级缓存之查询缓存：对普通属性进行缓存。如果关联的表发生了修改，那么查询缓存的生命周期也结束了。
在程序中必须手动启用查询缓存：query.setCacheable(true);

51. 如何在 WEB 里面配置 SPRING【首都信息面试题】

【参考答案】

主要配置 Spring　Web 监听器

52. Hibernate 中的 Cascade 属性有什么作用？

cascade 属性的作用是描述关联对象进行操作时的级联特性。可以有以下几种取值：

all : 所有情况下均进行关联操作。

save-update：(级联保存) 表明保存或更新当前对象时会级联保存或更新他所关联的对象。

none：所有情况下均不进行关联操作。这是默认值。

delete ：(级联删除)　级联删除所关联的对象。

all-delete-orphan：自动删除不再和 父对象关联的子对象。并且，在出现上面两种情况时执行上面两种的功能，可以说是一个全自动的属性值。

53. Struts1 优缺点

【参考答案】

优点：

1) 实现 MVC 模式，结构清晰,使开发者只关注业务逻辑的实现.
2) 有丰富的 tag 可以用 ,Struts 的标记库(Taglib)，如能灵活动用，则能大大提高开发效率
3) 页面导航
4) 使系统的脉络更加清晰。通过一个配置文件，即可把握整个系统各部分之间的联系,这对于后期的维护有着莫大的好处。尤其是当另一批开发者接手这个项目时，这种优势体现得更加明显。
5) 提供 Exception 处理机制。
6) 数据库链接池管理。
7) 支持 I18N（国际化）

缺点

1) 转到展示层时，需要配置 forward，如果有十个展示层的 jsp，需要配置十次 struts，而且还不包括有时候目录、文件变更，需要重新修改 forward，注意，每
2) Struts 的 Action 必需是 thread－safe 方式,它仅仅允许一个实例去处理所有的请求。所以 action 用到的所有的资源都必需统一同步，这个就引起了线程安全的问题。
3) 测试不方便. Struts 的每个 Action 都同 Web 层耦合在一起，这样它的测试依赖于 Web 容器，单元测试也很难实现
4) 类型的转换. Struts 的 FormBean 把所有的数据都作为 String 类型，它可以使用工具

Commons-Beanutils 进行类型转化。但它的转化都是在 Class 级别，而且转化的类型是不可配置的。

5) 对 Servlet 的依赖性过强. Struts 处理 Action 时必需要依赖 ServletRequest 和 ServletResponse，所有它摆脱不了 Servlet 容器。

6) 前端表达式语言方面集成了 JSTL，所以它主要使用 JSTL 的表达式语言来获取数据。可是 JSTL 的表达式语言在 Collection 和索引属性方面处理显得很弱。

7) 对 Action 执行的控制困难. Struts 创建一个 Action，如果想控制它的执行顺序将会非常困难。

8) 对 Action 执行前和后的处理. Struts 处理 Action 的时候是基于 class 的 hierarchies，很难在 action 处理前和后进行操作。

9) 对事件支持不够. 在 struts 中，实际是一个表单 Form 对应一个 Action 类(或 DispatchAction)，换一句话说：在 Struts 中实际是一个表单只能 对应一个事件。

54. 整合 spring 与 struts1 的方法，那种最好，为什么？

【参考答案】

答：1.第一种方法:

Struts 的 Action 继承 Spring 的 ActionSupport 类，并在 Action 中获取 Spring 的 ApplicationContext。这是最简单的一种整合方式，但有三个缺点：第一，Struts 与 Spring 紧密耦合，不能改换到其他 IoC 容器；第二，难以使用 Spring AOP 特性；第三，对于需要使用 DispatchAction 的 Struts 应用无能为力。

2.第二种方法:

在 Struts 的配置文件中，以 Spring 的 DelegatingRequestProcessor 类代替 Struts 的 RequestProcessor 类，并在 Spring 的配置文件中定义与 Struts 配置文件中<action-mappings>对应的 bean，从而将 Struts 的 Action 与 Spring 分开，并把 Struts 的动作置于 Spring 的控制之下。这种整合方式的优点是将不再依赖 Spring 这个特定的 IoC 容器，但必须依赖 Struts 的 RequestProcessor 类。

3.第三种方法:

通过 Spring 的 DelegatingActionProxy 类代理 Struts 的动作，即在 Struts 配置文件中，定义<action-mappings>的 type 属性全部改为 **DelegatingActionProxy**，而不是具体的类名,并在 Spring 配置文件中定义与 Struts 动作映射对应的 bean,从而将 Struts 的 Action 与 Spring 分开，并把 Struts 的动作置于 Spring 的控制之下。无疑，这是最灵活的一种整合方式。

【分析】

注意一般说几种整合方式所指的就是 Struts1

55. Struts1.2 的工作原理

【参考答案】

1.ActionServlet 核心控制器会拦截所有*.do 的请求

2.从 struts-config.xml 中找到用户请求的 Action

3.通过 struts-config.xml 中的配置再去找这个 Action 对应的 ActionForm,并实例化

4.把用户填写的数据自动填充到 ActionForm 中(调用 ActionForm 中的 setXX()方法填充)

5.同时把 ActionForm 放入到指定的范围中（request，session）

6.然后把请求转发给 Action

7.Action 获取 ActionForm 中的值然后调用业务逻辑层实现功能

8.在通过 ActionMapping 查找 Actionforward 实现转发;

56. 谈谈 Struts1 中的 ActionServlet。

【参考答案】

中心控制器，负责所以请求处理，并根据配置文件，将请求转到指定 Action 执行，并根据 Action 的 ActionForward 返回，转到指定视图。

57. Struts1.X 中 ActionServlet、 ActionForm 、Action、 ActionMapping 各起什么作用?

【参考答案】

a. Actionservlet 为控制器，接受用户请求并派发给相应的 Action 组件处理;
b. ActionForm 主要用于封装请求中的数据和简单验证
c. Action 组件具体对用户的请求进行处理
d. ActionMapping 封装当前用户请求 Action 相关的配置信息

58. Struts 1 的 Action 类与 Struts 2 的 Action 类区别?

【参考答案】

Struts 1 要求 Action 类要扩展自一个抽象基类。Struts 1 的一个共有的问题是面向抽象类编程而不是面向接口编程。

Struts 2 的 Action 类实现了一个 Action 接口，连同其他接口一起实现可选择和自定义的服务。

Struts 1 Action 类是单例类，因只有一个示例控制所有的请求。

Struts 2 Action 对象每一个请求都实例化对象，所以没有程安全的问题。

59. 谈谈你对 Struts1 的理解。

【参考答案】

1. struts 是一个按 MVC 模式设计的 Web 层框架，其实它就是一个大大的 servlet，这个 Servlet 名为 ActionServlet，或是 ActionServlet 的子类。我们可以在 web.xml 文件中将符合某种特征的所有请求交给这个 Servlet 处理，这个 Servlet 再参照一个配置文件（通常为/WEB-INF/struts-config.xml）将各个请求分别分配给不同的 action 去处理。

2. ActionServlet 把请求交给 action 去处理之前，会将请求参数封装成一个 formbean 对象（就是一个 java 类，这个类中的每个属性对应一个请求参数）

3.要说明的是， ActionServlet 把 formbean 对象传递给 action 的 execute 方法之前，可能会调用 formbean 的 validate 方法进行校验，只有校验通过后才将这个 formbean 对象传递给 action 的 execute 方法，否则，它将返回一个错误页面，这个错误页面由 input 属性指定，（看配置文件）作者为什么将这里命名为 input 属性，而不是 error 属性，我们后面结合实际的运行效果进行分析。

4.action 执行完后要返回显示的结果视图，这个结果视图是用一个 ActionForward 对象来表示的,actionforward 对象通过 struts-config.xml 配置文件中的配置关联到某个 jsp 页面，因为程序中使用的是在 struts-config.xml 配置文件为 jsp 页面设置的逻辑名，这样可以实现 action 程序代码与返回的 jsp 页面名称的解耦。

60. Struts1 中的 action 是安全线程么？为什么

【参考答案】

Servlet 是在多线程环境下的。即可能有多个请求发给一个 servelt 实例，每个请求是一个线程。

struts 下的 action 也类似，同样在多线程环境下。

为多线程环境编写代码。我们的 controller servlet 指挥创建你的 Action 类的一个实例，用此实例来服务所有的请求。因此，你必须编写线程安全的 Action 类。遵循与写线程安全的 servlet 同样的方针。

61. 在 Struts1 中 actionform 属于哪一层

【参考答案】

属于视图层组件，负责封装页面表单数据。

62. struts 控制器组件的主要包括?

【参考答案】

ActionServlet 组件：充当 struts 框架的中央控制器。

RequestProcessor 组件，充当每个子应用模块的请求处理器

Action 组件，负责处理一项具体的业务。

63. 常见的缓存技术举例【大唐动力面试题】

【参考答案】

操作系统磁盘缓存->减少磁盘机械操作

数据库缓存->减少文件系统 I/O

应用程序缓存->减少对数据库的查询

Web 服务器缓存->减少应用服务器请求

客户端浏览器缓存->减少对网站的访问

64. 以 Spring 为基础的 J2EE 项目中，一般如何分层？每个层的作用是什么？事务控制放在哪一层？

一般分为持久层（DAO 层）、业务层（Service 层）、控制层、视图层；

事务一般放在业务层，以一个业务作为一个事务分割的最小单位。

（七）EJB 与 WebServices 部分

1. EBJ 是基于哪些技术实现的和 JavaBean 的区别？

EJB 包括 Session Bean、Entity Bean、Message Driven Bean，基于 JNDI、RMI、JAT 等技术实现　EJB 是一个关于用 JAVA 语言开发的可部署的服务器端组件的组件体系结构。它是一个技术协议，能使组件运行于任何应用服务器,专门用来解决商务问题 JAVABEANS 是 JAVA 类，是由属性、事件和方法组成的 JAVA 组件，它们可以用来组成 JAVA 应用程序

Java Bean 是可复用的组件，对 Java Bean 并没有严格的规范，理论上讲，任何一个 Java 类都可以是一个 Bean。但通常情况下，由于 Java Bean 是被容器所创建（如 Tomcat）的，所以 Java Bean 应具有一个无参的构造器，另外，通常 Java Bean 还要实现 Serializable 接口用于实现 Bean 的持久性。Java Bean 实际上相当于微软 COM 模型中的本地进程内 COM 组件，它是不能被跨进程访问的。Enterprise Java Bean 相当于 DCOM，即分布式组件。它是基于 Java 的远程方法调用（RMI）技术的，所以 EJB 可以被远程访问（跨进程、跨计算机）。但 EJB 必须被布署在诸如 Webspere、WebLogic 这样的容器中，EJB 客户从不直接访问真正的 EJB 组件，而是通过其容器访问。EJB 容器是 EJB 组件的代理，EJB 组件由容器所创建和管理，客户通过容器来访问真正的 EJB 组件。

2. Webservice 有什么好处?

跨平台的可互操作性　　跨防火墙的通信　应用程序集成　软件和数据重用。

3. 什么是事物处理，J2EE 提供哪两种事物处理方式

事务（Transaction）是数据库管理系统提供的基本功能之一，可以把完成用户一个特定工作的一组操作看作是一个不可拆分的工作单元，所以事务也就是作业或任务。

JDBC：支持单一数据库连接事务

JTA：支持分布式事务

4. WEB SERVICE 的理解【大唐动力面试题】

Web Service 主要是为了使原来各孤立的站点之间的信息能够相互通信、共享而提出的一种接口。 Web Service 所使用的是 Internet 上统一、开放的标准，如 HTTP、XML、SOAP（简单对象访问协议）、WSDL 等，所以 Web Service 可以在任何支持这些标准的环境（Windows,Linux）中使用。注：SOAP 协议（Simple Object Access Protocal,简单对象访问协议）,它是一个用于分散和分布式环境下网络信息交换的基于 XML 的通讯协议。在此协议下，软件组件或应用程序能够通过标准的 HTTP 协议进行通讯。它的设计目标就是简单性和扩展性，这有助于大量异构程序和平台之间的互操作性，从而使存在的应用程序能够被广泛的用户访问。

5. J2EE 系统访问速度慢.从哪些方面可以优化

J2EE 性能的优化包括很多方面的，要达到一个性能优良的系统，除了关注代码之外，还应该根据系统实际的运行情况，从服务器软硬件环境、集群技术、系统构架设计、系统部署环境、数据结构、算法设计等方面综合考虑来优化。一般程序级别的优化首先考虑做数据缓存，数据库方面全做表的切割、数据分区、索引等这些方面来加快对数据的访问。

6. 说说你所知道的应用服务器？

BEA WebLogic Server，IBM WebSphere Application Server，Oracle9i Application Server，jBoss

7. 什么是 J2EE？J2EE 是技术还是平台还是框架？ 什么是 J2EE

答:

Je22 是 Sun 公司提出的多层(multi-diered)，分布式(distributed)，基于组件(component-base)的企业级应用模型(enterpriese application model).在这样的一个应用系统中，可按照功能划分为不同的组件，这些组件又可在不同计算机上，并且处于相应的层次(tier)中。所属层次包括客户层(clietn tier)组件,web 层和组件,Business 层和组件,企业信息系统(EIS)层。

J2EE 本身是一个标准，一个为企业分布式应用的开发提供的标准平台。

J2EE 也是一个框架，包括 JDBC、JNDI、RMI、JMS、EJB、JTA 等技术。

8. 请对以下在 J2EE 中常用的名词进行解释(或简单描述)

web 容器：给处于其中的应用程序组件（JSP，SERVLET）提供一个环境，使 JSP,SERVLET 直接更容器中的环境变量接口交互，不必关注其它系统问题。主要有 WEB 服务器来实现。例如：TOMCAT,WEBLOGIC,WEBSPHERE 等。该容器提供的接口严格遵守 J2EE 规范中的 WEB APPLICATION 标准。我们把遵守以上标准的 WEB 服务器就叫做 J2EE 中的 WEB 容器。

EJB 容器：Enterprise java bean 容器。更具有行业领域特色。他提供给运行在其中的组件 EJB 各种管理功能。只要满足 J2EE 规范的 EJB 放入该容器，马上就会被容器进行高效率的管理。并且可以通过现成的接口来获得系统级别的服务。例如邮件服务、事务管理。

JNDI：（Java Naming & Directory Interface）JAVA 命名目录服务。主要提供的功能是：提供一个目录系统，让其它各地的应用程序在其上面留下自己的索引，从而满足快速查找和定位分布式应用程序的功能。

JMS：（Java Message Service）JAVA 消息服务。主要实现各个应用程序之间的通讯。包括点对点和广播。

JTA：（Java Transaction API）JAVA 事务服务。提供各种分布式事务服务。应用程序只需调用其提供的接口即可。

JAF：（Java Action FrameWork）JAVA 安全认证框架。提供一些安全控制方面的框架。让开发者通过各种部署和自定义实现自己的个性安全控制策略。

RMI/IIOP:（Remote Method Invocation /internet 对象请求中介协议）他们主要用于通过远程调用服务。例如，远程有一台计算机上运行一个程序，它提供股票分析服务，我们可以在本地计算机上实现对其直接调用。当然这是要通过一定的规范才能在异构的系统之间进行通信。RMI 是 JAVA 特有的。

9. 如何给 weblogic 指定大小的内存?

在启动 Weblogic 的脚本中（位于所在 Domian 对应服务器目录下的 startServerName），增加 set MEM_ARGS=-Xms32m -Xmx200m，可以调整最小内存为 32M，最大 200M

10. 如何在 weblogic 中进行 ssl 配置与客户端的认证配置或说说 j2ee(标准)进行 ssl 的配置?

缺省安装中使用 DemoIdentity.jks 和 DemoTrust.jks KeyStore 实现 SSL，需要配置服务

器使用 Enable SSL，配置其端口，在产品模式下需要从 CA 获取私有密钥和数字证书，创建 identity 和 trust keystore，装载获得的密钥和数字证书。可以配置此 SSL 连接是单向还是双向的。

11. EJB 是基于哪些技术实现的？并说出 SessionBean 和 EntityBean 的区别，StatefulBean 和 StatelessBean 的区别？

EJB 包括 Session Bean、Entity Bean、Message Driven Bean，基于 JNDI、RMI、JAT 等技术实现。

SessionBean 在 J2EE 应用程序中被用来完成一些服务器端的业务操作，例如访问数据库、调用其他 EJB 组件。EntityBean 被用来代表应用系统中用到的数据。

对于客户机，SessionBean 是一种非持久性对象，它实现某些在服务器上运行的业务逻辑。

对于客户机，EntityBean 是一种持久性对象，它代表一个存储在持久性存储器中的实体的对象视图，或是一个由现有企业应用程序实现的实体。

Session Bean 还可以再细分为 Stateful Session Bean 与 Stateless Session Bean ，这两种的 Session Bean 都可以将系统逻辑放在 method 之中执行，不同的是 Stateful Session Bean 可以记录呼叫者的状态，因此通常来说，一个使用者会有一个相对应的 Stateful Session Bean 的实体。Stateless Session Bean 虽然也是逻辑组件，但是他却不负责记录使用者状态，也就是说当使用者呼叫 Stateless Session Bean 的时候，EJB Container 并不会找寻特定的 Stateless Session Bean 的实体来执行这个 method。换言之，很可能数个使用者在执行某个 Stateless Session Bean 的 methods 时，会是同一个 Bean 的 Instance 在执行。从内存方面来看， Stateful Session Bean 与 Stateless Session Bean 比较， Stateful Session Bean 会消耗 J2EE Server 较多的内存，然而 Stateful Session Bean 的优势却在于他可以维持使用者的状态。

12. EJB 包括（SessionBean,EntityBean）说出他们的生命周期，及如何管理事务的？

SessionBean：Stateless Session Bean 的生命周期是由容器决定的，当客户机发出请求要建立一个 Bean 的实例时，EJB 容器不一定要创建一个新的 Bean 的实例供客户机调用，而是随便找一个现有的实例提供给客户机。当客户机第一次调用一个 Stateful Session Bean 时，容器必须立即在服务器中创建一个新的 Bean 实例，并关联到客户机上，以后此客户机调用 Stateful Session Bean 的方法时容器会把调用分派到与此客户机相关联的 Bean 实例。

EntityBean：Entity Beans 能存活相对较长的时间，并且状态是持续的。只要数据库中的数据存在，Entity beans 就一直存活。而不是按照应用程序或者服务进程来说的。即使 EJB 容器崩溃了，Entity beans 也是存活的。Entity Beans 生命周期能够被容器或者 Beans 自己管理。

EJB 通过以下技术管理实务：对象管理组织（OMG）的对象实务服务（OTS），Sun Microsystems 的 Transaction Service（JTS）、Java Transaction API（JTA），开发组（X/Open）的 XA 接口。

13. EJB 的几种类型

【参考答案】

会话（Session Bean）Bean ，实体（Entity Bean ）Bean 消息驱动的（Message Driven Bean） Bean

会话 Bean 又可分为有状态（Stateful）和无状态（Stateless）两种。
实体 Bean 可分为 Bean 管理的持续性（BMP）和容器管理的持续性（CMP）两种 。

14. Tomcat 和 WebLogic 的区别？【杭州网阙科技】

【参考答案】

1) WLS 全面支持 J2EE 的标准规范和其他标准规范（Web Service, SSL, xml 等），同时 BEA 为众多规范组织的制定者之一，积极参与规范的制定； Tomcat 只支持部分 J2EE 标准，应用局限性强,不能够安全稳定的支持大并发
2) WLS：集群机制，支持分布式的应用；Tomcat：不支持
3) WLS：开发模式下，不用重起部署新 Web, EJB 应用；Tomcat：不支持
4) WLS 收费的应用服务器，支持 EJB 容器。Tomat 开源，但只实现了 Web 容器，不支持企业级应用。

（八）项目管理与设计模式

1. OO 设计原则？

单一责任原则、依赖倒置原则、开放封闭原则 OCP。

2. 什么情况下，不能用单例模式。

单例模式简单易用，但是也是所有设计模式中最容易滥用的模式。当你的类想得到很好的扩展时，不能使用单例模式。

3. 软件开发的流程是怎样的？【云巢动脉】

需求分析、概要设计、详细设计、编码、测试、交付、验收、维护

4. 描述工厂模式和单例优缺点 举例在什么情况下用。

单例模式优点：

单例 保证一个类只有单一的实例，也就是说你无法通过 New 来创建这个类的一个新实例 好处：当一个对象在程序内部只能有一个实例的时候，它可以保证我们不会重复创建，而是始终指向同一个对象。 缺点就是 它就是在内存上共享，都可以去访问它，而且多个用户访问的都是同一个实例，会造成线程不安全。

工厂模式优点：

第一，方便系统统一管理与维护，降低组件间耦合。

第二，它让具体的创建实例与客户端分离，客户端是通过它们的抽象接口操纵实例，使得实例名不会出现在客户端代码中。

5. 写适配器模式，观察者模式？【云巢动脉】

适配器模式：

Target：定义 Client 使用的与特定领域相关的接口

```
public interface Target
{
    void request();
}
```

Adaptee：现在需要适配的已经存在的接口

```
public class Adaptee
{
    public void specificRequest(){}
}
```

Adapter：对 Adaptee 的接口与 Target 接口进行适配

```
public class Adapter implements Target
{
    public Adapter(Adaptee adaptee)
    {
        super();
        this.adaptee = adaptee;
    }
    public void request()
    {
        adaptee.specificRequest();
    }
    private Adaptee adaptee;
}
```

观察者模式：

Subject：

1. Subject 可以加将 Observer 对象的引用保存在一个聚集中，每个 Subject 可以有任意个 Observer。
2. 提供注册和删除 Observer 对象的接口。

```
public interface Subject
{
    public void registerObserver(Observer o);
    public void removeObserver(Observer o);
    public void notifyObservers();
}
```

Observer：

1. 为那些在 Subject 发生改变时需获得通知的对象定义一个更新接口。

```
public interface Observer
{
    public void update();
}
```

Concrete Subject：

1. 将有关状态存入各 ConcreteObserver 对象，当它的状态发生改变时，向它的各个观察者发出通知。

```
public class ConcreteSubject implements Subject
{
    private ArrayList<Observer> observers;
    public ConcreteSubject()
    {
```

```
        observers = new ArrayList<Observer>();
    }
    public void registerObserver(Observer o)
    {
        observers.add(o);
    }
    public void removeObserver(Observer o)
    {
        int i = observers.indexOf(o);
        if (i >= 0)
        {
            observers.remove(i);
        }
    }
    public void notifyObservers()
    {
        for (int i = 0; i < observers.size(); i++)
        {
            observer = observers.get(i);
            observer.update();
        }
    }
    public void stateChanged()
    {
        notifyObservers();
    }
    public void setState()
    {
        stateChanged();
    }}
```

Concrete Observer：

1. 维护一个指向 ConcreteSubject 对象的引用。
2. 存储有关状态，这些状态应与 Subject 的状态保持一致。
3. 实现 Observer 的更新接口以使自身状态与 Subject 的状态保持一致。

```
public class ConcreteObserver implements Observer
{
    private Subject subject;
    public ConcreteObserver(Subject subject)
    {
        this.subject = subject;
        subject.registerObserver(this);
    }
    public void update()
    {
```

```
        // your code
    }}
```

6. 适配器模式解释【】

把一个类的接口变换成客户端所期待的另一种接口，从而使原本因接口原因不匹配而无法一起工作的两个类 能够一起工作。适配类可以根据参数返还一个合适的实例给客户端

7. 说说 UML

标准建模语言 UML。用例图,静态图(包括类图、对象图和包图),行为图,交互图(顺序图,合作图),实现图。

8. UML 中的几种关系？

依赖，关联，泛化，实现

9. UML 几种图及用途？

1) 用例图—描述系统的参与者与用例之间的关系
2) 类图—描述系统中的概念及它们之间的关系
3) 对象图—描述系统中实例及它们之间关系
4) 状态图—描述系统中对象所具备的状态，及触发状态改变的事件
5) 时序图—描述对象之间发送消息的时间顺序
6) 协作图—描述对象之间协作关系
7) 活动图—描述执行过程中涉及的活动及状态变化
8) 组件图—描述系统组件之间的关系
9) 配置图—描述系统的硬件及软件的物理结构

10. 常用的设计模式？说明工厂模式。

Java 中的 23 种设计模式：

Factory（工厂模式）， Builder（建造模式）， Factory Method（工厂方法模式），
Prototype（原始模型模式），Singleton（单例模式）， Facade（门面模式），
Adapter（适配器模式）， Bridge（桥梁模式）， Composite（合成模式），
Decorator（装饰模式）， Flyweight（享元模式）， Proxy（代理模式），
Command（命令模式）， Interpreter（解释器模式）， Visitor（访问者模式），
Iterator（迭代子模式）， Mediator（调停者模式）， Memento（备忘录模式），
Observer（观察者模式）， State（状态模式）， Strategy（策略模式），
Template Method（模板方法模式）， Chain Of Responsibleity（责任链模式）

工厂模式：

工厂模式是一种经常被使用到的模式，所指的是简单工厂模式，还有另一种工厂方法模式、单例模式，也用到一些工厂模式的思想。工厂模式可以根据提供的数据生成一组类中某一个类的实例，通常这一组类有一个公共的抽象父类并且实现了相同的方法，但是这些方法针对不同的数据进行了不同的操作。首先需要定义一个基类，该类的子类通过不同的方法实现了基类中的方法。然后需要定义一个工厂类，工厂类可以根据条件生成不同的子类实例。当得到子类的实例后,开发人员可以调用基类中的方法而不必考虑到底返回的是哪一个子类的实例，这就是工厂模式的实现。

【说明】

设计模式分为三大类：创建型，结构型，行为型。

分别是：

创建型（工厂、工厂方法、抽象工厂、单例）

结构型（包装、适配器，组合，代理）

行为（观察者，模版，策略）

说 6、7 种即可。

11. DAO 设计模式与 DELEGATE 模式

DAO(Data Access Object)模式实际上是两个模式的组合，即 Data Accessor 模式和 Active Domain Object 模式，其中 Data Accessor 模式实现了数据访问和业务逻辑的分离，而 Active Domain Object 模式实现了业务数据的对象化封装，一般我们将这两个模式组合使用。

DAO 模式通过对底层数据的封装，为业务层提供一个面向对象的接口，使得业务逻辑开发员可以面向业务中的实体进行编码。通过引入 DAO 模式，业务逻辑更加清晰，且富于形象性和描述性，这将为日后的维护带来极大的便利。试想，在业务曾通过 Customer.getName 方法获得客户姓名，相对于直接通过 SQL 语句访问数据库表并从 ResultSet 中获得某个字符型字段而言，哪种方式更加易于业务逻辑的形象化和简洁化？

Business Delegate 起到客户端业务抽象化的作用。它抽象化，进而隐藏业务服务的实现。使用 Business Delegate，可以降低表示层客户端和系统的业务服务之间的耦合程度。根据实现策略不同，Business Delegate 可以在业务服务 API 的实现中，保护客户端不受可能的变动性影响。这样，在业务服务 API 或其底层实现变化时，可以潜在地减少必须修改表示层客户端代码的次数。

12. 适配器模式与桥梁模式的区别

适配器模式把一个类的接口变换成客户端所期待的另一种接口，从而使原本因接口不匹配而无法在一起工作的两个类能够在一起工作。又称为转换器模式、变压器模式、包装模式（把已有的一些类包装起来，使之能有满足需要的接口）。适配器模式的用意是将接口不同而功能相同或者相近的两个接口加以转换，包括适配器角色补充一些源角色没有但目标接口需要的方法。就像生活中电器插头是三相的，而电源插座是两相的，这时需要一个三相变两相的转换器来满足。

比如，在 Java I/O 库中使用了适配器模式，象 FileInputStream 是一个适配器类，其继承了 InputStrem 类型，同时持有一个对 FileDiscriptor 的引用。这是将一个 FileDiscriptor 对象适配成 InputStrem 类型的对象形式的适配器模式。StringReader 是一个适配器类，其继承了 Reader 类型，持有一个对 String 对象的引用。它将 String 的接口适配成 Reader 类型的接口。等等。

桥梁模式的用意是要把实现和它的接口分开，以便它们可以独立地变化。桥梁模式并不是用来把一个已有的对象接到不相匹配的接口上的。当一个客户端只知道一个特定的接口，但是又必须与具有不同接口的类打交道时，就应该使用桥梁模式。

比如，JDBC 驱动器就是一个桥梁模式的应用，使用驱动程序的应用系统就是抽象化角色，而驱动器本身扮演实现化角色。应用系统和 JDBC 驱动器是相对独立的。应用系统动态地选择一个合适的驱动器，然后通过驱动器向数据库引擎发出指令就可以访问数据库中的数据。

13. 开发中都用到了那些设计模式?用在什么场合?

简单工厂模式、单例模式、观察者模式、适配器模式等。

四、程序题

1、请用 Java 写一个冒泡排序方法

【参考答案】

```
public static void Bubble(int a[]){
        for(int i=0;i<a.length-1;i++){
            for(int j=a.length-1;j>i;j--){
                if(a[j]<a[j-1]){
                    a[j]=a[j]+a[j-1];
                    a[j-1]=a[j]-a[j-1];
                    a[j]=a[j]-a[j-1];
                }
            }
        }
    }
```

2、子线程循环 10 次，接着主线程循环 100，接着又回到子线程循环 10 次，接着再回到主线程又循环 100，如此循环 50 次，请写出程序。

【参考答案】

最终的程序代码如下：

```
public class ThreadTest {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        new ThreadTest().init();

    }
    public void init()
    {
        final Business business = new Business();
        new Thread(
                new Runnable()
                {

                    public void run() {
                        for(int i=0;i<50;i++)
                        {
                            business.SubThread(i);
                        }
                    }
```

```
                }

        ).start();

        for(int i=0;i<50;i++)
        {
            business.MainThread(i);
        }
    }

    private class Business
    {
        boolean bShouldSub = true;//这里相当于定义了控制该谁执行的一个信号灯
        public synchronized void MainThread(int i)
        {
            if(bShouldSub)
                try {
                    this.wait();
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }

            for(int j=0;j<5;j++)
            {
                System.out.println(Thread.currentThread().getName() + ":i=" + i
+",j=" + j);
            }
            bShouldSub = true;
            this.notify();

        }


        public synchronized void SubThread(int i)
        {
            if(!bShouldSub)
                try {
                    this.wait();
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
```

```
            for(int j=0;j<10;j++)
            {
                System.out.println(Thread.currentThread().getName() + ":i=" + i
+",j=" + j);
            }
            bShouldSub = false;
            this.notify();
        }
    }
}
```

备注：不可能一上来就写出上面的完整代码，最初写出来的代码如下，问题在于两个线程的代码要参照同一个变量，即这两个线程的代码要共享数据，所以，把这两个线程的执行代码搬到同一个类中去：

```
package com.huawei.interview.lym;

public class ThreadTest {

    private static boolean bShouldMain = false;

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        /*new Thread(){
        public void run()
        {
            for(int i=0;i<50;i++)
            {
                for(int j=0;j<10;j++)
                {
                    System.out.println("i=" + i + ",j=" + j);
                }
            }
        }

    }.start();*/


        //final String str = new String("");

        new Thread(
                new Runnable()
                {
```

```
            public void run()
            {
                for(int i=0;i<50;i++)
                {
                    synchronized (ThreadTest.class) {
                        if(bShouldMain)
                        {
                            try {
                                ThreadTest.class.wait();}
                            catch (InterruptedException e) {
                                e.printStackTrace();
                            }
                        }
                        for(int j=0;j<10;j++)
                        {
                            System.out.println(
                                    Thread.currentThread().getName() +
                                    "i=" + i + ",j=" + j);
                        }
                        bShouldMain = true;
                        ThreadTest.class.notify();
                    }
                }
            }
        }
).start();

for(int i=0;i<50;i++)
{
    synchronized (ThreadTest.class) {
        if(!bShouldMain)
        {
            try {
                ThreadTest.class.wait();}
            catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        for(int j=0;j<5;j++)
        {
            System.out.println(
                    Thread.currentThread().getName() +

                    "i=" + i + ",j=" + j);
```

```
                }
                bShouldMain = false;
                ThreadTest.class.notify();
            }
        }
    }

}
```

下面使用 jdk5 中的并发库来实现的：

```
import java.util.concurrent.Executors;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;
import java.util.concurrent.locks.Condition;

public class ThreadTest
{
    private static Lock lock = new ReentrantLock();
    private static Condition subThreadCondition = lock.newCondition();
    private static boolean bBhouldSubThread = false;
    public static void main(String [] args)
    {
        ExecutorService threadPool = Executors.newFixedThreadPool(3);
        threadPool.execute(new Runnable(){
            public void run()
            {
                for(int i=0;i<50;i++)
                {
                    lock.lock();
                    try
                    {
                        if(!bBhouldSubThread)
                            subThreadCondition.await();
                        for(int j=0;j<10;j++)
                        {
                            System.out.println(Thread.currentThread().getName()
+ ",j=" + j);
                        }
                        bBhouldSubThread = false;
                        subThreadCondition.signal();
                    }catch(Exception e)
                    {
                    }
                    finally
```

```
                    {
                        lock.unlock();
                    }
                }
            }

        });
        threadPool.shutdown();
        for(int i=0;i<50;i++)
        {
                lock.lock();
                try
                {
                    if(bBhouldSubThread)
                            subThreadCondition.await();

                    for(int j=0;j<10;j++)
                    {
                        System.out.println(Thread.currentThread().getName()    +
",j=" + j);
                    }
                    bBhouldSubThread = true;
                    subThreadCondition.signal();
                }catch(Exception e)
                {
                }
                finally
                {
                    lock.unlock();
                }
        }
    }
```

3、写一个程序,把一个文件的数组按对角线做对称变换,并输出!

【参考答案】

一个正方形里面全数字,写一个程序,成对角线转变!　　我做的这个是 3 行 3 列的对角互换，也许转换规则不一样

```
public class testMain {
    public static void main(String[] args) {
        int a[][]=new int[3][3];
        int c=1;
        //初始化数据
        for(int i=0;i<3;i++){
            for(int j=0;j<3;j++){
```

```
                a[i][j]=c++;
}
        }
        System.out.println("转换之前：");
        for(int i=0;i<3;i++){
            for(int j=0;j<3;j++){
                System.out.print("a["+i+"]["+j+"]="+a[i][j]+"   ");
            }
            System.out.println("\n");
        }
        for(int i=0;i<3;i++){
            for(int j=0;j<3;j++){
                if((i+1<3&&j+1<3)&&i==j&&i!=0&&i!=3-i){
                    int temp=a[i-1][j-1];
                    a[i-1][j-1]=a[i+1][j+1];
                    a[i+1][j+1]=temp;
                    temp=a[i-1][j+1];
                    a[i-1][j+1]=a[i+1][j-1];
                    a[i+1][j-1]=temp;
                }
            }
        }
        System.out.println("转换之后：");
        for(int i=0;i<3;i++){
            for(int j=0;j<3;j++){
                System.out
print("a["+i+"]["+j+"]="+a[i][j]+"   ");
            }
            System.out.println("\n");
        }
    }
}
```

4、写一个方法，传入一个 int 型的数字，把它的四个字节码取出来，并且把它按大小顺序通过控制台输出？

【参考答案】

```
public static void main(String[] args) {
    int num = -800000000;
    String str = Integer.toBinaryString(num); //获得 num 的二进制
    if(num>=0) {   //如果输入的数为正数,位数可能不足 32 位，要补 0；负数肯定是 32 位
      if(str.length()<32) { //二进制不足 32 位，就在前面补 0
        int n0 = 32-str.length(); //看差几个 0
        String temp = "";
        for(int i=0;i<n0;i++) {
```

```
            temp = temp + "0"; //拼 0
          }
          str = temp + str;
        }
      }
      String s1 = str.substring(0, 8);
      String s2 = str.substring(8, 16);
      String s3 = str.substring(16, 24);
      String s4 = str.substring(24, 32);

      System.out.println(str);
      System.out.println(s1);
      System.out.println(s2);
      System.out.println(s3);
      System.out.println(s4);

      int n1=Integer.parseInt(s1,2);//以二进制把字符串解析为 10 进制的数
      int n2=Integer.parseInt(s2,2);
      int n3=Integer.parseInt(s3,2);
      int n4=Integer.parseInt(s4,2);
      System.out.println(n1);
      System.out.println(n2);
      System.out.println(n3);
      System.out.println(n4);      //整数大小自己比较吧

    }
```

【分析】

5、设计 4 个线程，其中两个线程每次对 j 增加 1，另外两个线程对 j 每次减少 1。写出程序。

【参考答案】

以下程序使用内部类实现线程，对 j 增减的时候没有考虑顺序问题。

```
public class ThreadTest1
{
private int j;
public static void main(String args[]){
   ThreadTest1 tt=new ThreadTest1();
   Inc inc=tt.new Inc();
   Dec dec=tt.new Dec();
   for(int i=0;i<2;i++){
       Thread t=new Thread(inc);
       t.start();
        t=new Thread(dec);
```

```
        t.start();
        }
    }
private synchronized void inc(){
    j++;
    System.out.println(Thread.currentThread().getName()+"-inc:"+j);
    }
private synchronized void dec(){
    j--;
    System.out.println(Thread.currentThread().getName()+"-dec:"+j);
    }
class Inc implements Runnable{
    public void run(){
        for(int i=0;i<100;i++){
        inc();
        }
    }
}
class Dec implements Runnable{
    public void run(){
        for(int i=0;i<100;i++){
        dec();
        }
    }
}
}

----------随手再写的一个-------------
class A
{
JManger j =new JManager();
main()
{
  new A().call();
}

void call
{
  for(int i=0;i<2;i++)
  {
     new Thread(
         new Runnable(){ public void run(){while(true){j.accumulate()}}}
     ).start();
     new          Thread(new          Runnable(){          public          void
```

```
    run(){while(true){j.sub()}}}).start();
    }
  }
  }

  class JManager
  {
    private j = 0;

    public synchronized void subtract()
    {
       j--
    }

    public synchronized void accumulate()
    {
       j++;
    }

  }
```

6、十六进制的 216 转换十进制是多少？

216 是 16 进制，转 10 进制:
＝2*16^2+1*16^1+6*16^0
=512+16+6
=536

以下函数 htoi 函数的功能是将一个十六进制数字的字符串,转换成它等价的十进制整数值。

```
Public int htoi(char s[])
{
Int i , n ;
N=0;
For(i=0 , s[i]<’ \0’ ;i++)
{
    If(s[i]>=0&&s[i]<=9) n=_______
    If(s[i]>=’ a’ &&s[i]<=’ f’ ) n=_________
    If(s[i]>=’ A’ &&s[i]<=’ F’ ) n=_________

}
Return (n);
}
```

7、如何把一段逗号分割的字符串转换成一个数组?

【参考答案】

可以说说我的思路：

1. 用正则表达式，代码大概为：String [] result = orgStr.split(“,”);
2. 用 StingTokenizer ,代码为：

```
StringTokenizer  tokener = StringTokenizer(orgStr,”,”);
   String [] result = new String[tokener .countTokens()];
   Int i=0;
   while(tokener.hasNext(){result[i++]=toker.nextToken();}
```

8、编写一个函数将一个十六进制数的字符串参数转换成整数返回。

【参考答案】

```
String str = “13abf”;
int len = str.length;
int sum = 0;
for(int i=0;i<len;i++){
    char c = str.charAt(len-1-i);
    int n = Character.digit(c,16);
    sum += n * (1<<(4*i));
}
```

其实，也可以用 Integer.parseInt(str,16)，但面试官很可能是想考我们的编码基本功。

9、读取一个文件在控制台打印出来

【参考答案】

```
File file = new File("E:\\JRadioButtonDemo.java");
    long file_length= file.length();
        try {
            //输入流
            FileInputStream input = new FileInputStream(file);
            byte b_data [] = new byte[(int)file_length];
            input.read(b_data);
            System.out.println(new String(b_data));
            input.close();
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
```

10、递归实现 1,1,2,3,5,8,….第 30 个数是多少？【上海菲耐德】

【参考答案】

```
public  static  int  Foo(int  i)
```

```
        {
                if  (i  <=  0)
                        return  0;
                else  if(i  >  0  &&  i  <=  2)
                        return  1;
                else  return  Foo(i  -1)  +  Foo(i  -  2);
        }
int  i=Foo(30);
System.out.println(i);
```

11、求一个字符串中第一个无重复的字符

```
    public static void getUniqueString(String str){
        boolean bool = true;
        for(int i=0;i<str.length()&&bool;i++){
            String s1 = str.substring(i, i+1);
            if(str.indexOf(s1, i+1)==-1){
                System.out.println(s1);
                bool = false;
            }
        }
    }
```

12、写一个递归函数，输入一个整数，反序输出这个整数

```
// 写一个递归函数，输入一个整数，反序输出这个整数
    public static void printOut(int n) {
        System.out.print(n % 10);
        if (n >= 10){
            printOut(n / 10);
        }
    }
```

13、有一个数据文件:123 34 17 651234 345....这些数据都是随机产生的,编写程序读出该文件.并将其以从大到小的顺序输出到另一个文件中.

```
public void readtext(){
    File file = new File("D:\test.txt");
    List list= new ArrayList();
    try {
        BufferedReader br=new BufferedReader(new FileReader(file));
        String data = "";
        String line = null;
        while ( (line = br.readLine()) != null) {
            data = data.concat(line);
        }
        StringTokenizer stoken = new StringTokenizer(data, " ");
```

```
        while (stoken.hasMoreTokens()) {
            int i = Integer.parseInt(stoken.nextToken());
            list.add(i);
        }
    } catch(Exception ex) {}
    String[] str = new String[list.size()];
    for(int i=0;i<list.size();i++){
        str[i]=list.get(i);
    }
    Object iTemp= null;
    for(int i=1;i<list.size();i++) {
        for(int j=list.size()-1;j>=i;j--) {
            if(str[j]>str[j-1]) {
                iTemp = str[j-1];
                str[j-1] = str[j];
                str[j] = iTemp;
            }
    }
    String result = "";
    for(int i=0;i<str.length;i++){
        result +=str[i]+" ";
    }
    //将 result 写入另外一个文件即可。
}
```

14、从一到十九共十九个数,打印出利用这十九个整数任意多个相加等于 20 所以可能性,每个数字在同一个算式中只出现一次.

```
public void test(){
    Integer[] a = new Integer[19];
    for(int i=1;i<20;i++){
        a[i-1]=i;
    }
    for(int i=0;i<18;i++){
    for(int j=18-i;j<18;j++)
        if(a[i]+a[j]==20)
            System.out.println(a[i]+"+"+a[i+1]+"="+20);
    }
}
```

15、一个字符串中可能存在 A-Z 的全角字符，写一个方法把里面的全角字符转变成半角字符？

答：采用建立字典表进行查找转换

```
public  static String translate(String s){
    String qj = "Ａ Ｂ Ｃ Ｄ Ｅ Ｆ Ｇ Ｈ Ｉ Ｊ Ｋ Ｌ Ｍ Ｎ Ｏ Ｐ Ｑ Ｒ Ｓ Ｔ Ｕ Ｖ Ｗ Ｘ Ｙ Ｚ";
```

```
    String bj = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    StringBuffer sb = new StringBuffer();
    for(int i=0;i<s.length();i++){
       char c = s.charAt(i);
       int pos = qj.indexOf(c);
       if(pos>=0){
          System.out.println(c + "," + pos);
          sb.append(bj.charAt(pos));
       }else{
          sb.append(c);
       }
    }
    return sb.toString();
  }
```

16、Stack 堆栈，实现进栈、出栈。【云巢动脉面试题】

```
package t1;
public class mystack {
    private Object[] data;
    private int top=-1;
    private int size;
    public mystack()
    {
        data=new Object[5];
        size=5;
    }
    public mystack(int size)
    {
        data=new Object[size];
        this.size=size;
    }
    public void push(Object obj)
    {
        if(this.isfull())
        {
            return ;
        }
        top++;
        data[top]=obj;
    }
    public Object pop() {
        if(this.isempty())
        {
```

```
            return null;
        }
        Object obj=data[top];
        top--;
        return obj ;
    }

    public boolean isfull()
    {
        if(top==data.length)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    public boolean isempty()
    {
        if(top==-1)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}
```

17、定义两个变量 a 和 b，不使用第三个变量，使两个值交换

```
public class testMain {
    public void test(int a,int b){
        System.out.println("交换前 a = "+a);
        System.out.println("交换前 b = "+b);
        a=a+b;
        b=a-b;
        a=a-b;
       System.out.println("交换后 a = " +a);
       System.out.print("交换后 b = "+b);
    }
    public static void main(String args[]){
        new testMain().test(10,13);
```

```
    }
}
```

18、针对一个分期付款，总期为 1 年，给定分期金额，期数和开始还款时间，计算出各期还款日期。

```
package  demo;
import  java.util.Calendar;
import  java.util.Date;
public  class  TestDemo  {
        //  分期付款，总期为 1 年，给定分期金额，期数和开始还款时间
        //  计算出各期还款日期
        public  void  huankuan(double  amount,int    num,Date  start){
                int  period  =  365/num;  //  一年中分期间隔天数

                Calendar  cal  =  Calendar.getInstance();
                cal.set(Calendar.YEAR,  start.getYear()+1900);
                cal.set(Calendar.MONTH,  start.getMonth());
                cal.set(Calendar.DATE,  start.getDate());

                for(int  i=1;i<=num;i++){
                        System.out.println(" 第" + i + "期还款日
期: " + cal.getTime().toLocaleString());
                        cal.add(Calendar.DATE,  period);
                }
        }
                public  static  void  main(String[]  args)  {
                TestDemo  demo  =  new  TestDemo();
                demo.huankuan(20000.00,  1,  new  Date());

        }
}
```

19、用一个方法查出宜个数值类型数组的最大值，用递归方式实现。【高达软件】

方法 1

```
public class Test1 {
    public static int a(int[] i,int j){
        if(i.length-1>j){
            if(i[j]>i[j+1]){
                i[j+1]=i[j];
            }
            return a(i,j+1);
        }else{
            return i[i.length-1];
        }
```

```
    }

}
```

方法 2 -- 非递归

```
public  static int  test(int  []num) {
int x=0;
int log   =  num.Length;
for(intt=0;t<log;t++){
if(num[t]>x){
 x=num[t];
}
}return  x;
}
```

方法 3 --- 递归 不改变原数组中的元素

```
public static int getMax(int[]a, int index,int max){
        int len = a.length;
        if(len==1){
            return a[len-1];
        }
        if(index==0){
            max = a[index];
        }
        if(index==len){
            return max;
        }
        if(max<a[index]){
            max = a[index];
        }
        index++;
        return getMax(a,index,max);
}
// 测试
int max = getMax(new int[]{2,5,18,3,38,10,2},0,0);
System.out.println(max);
```

20、用 C 编写将一个 100 以内的自然数分解质因数

```
/*  100 以内素数  */
#include<stdio.h>
main()
{
 int i,j;
 for(i=2;i<100;i++)
```

```
 {
  for(j=2;j<i;j++)
  {
if(i%j==0)
    break;
  }
  if(i==j)
  {
   printf("%d ",i);
  }
 }
}

/* 分解质因数*/
main()
{
    int   n,i;
    printf( "please   input   a   number:\n ");
    scanf( "%d ",&n);
    printf( "%d= ",n);
    for(i=2;i <=n;i++)
        while(n!=i)
        {
            if(n%i==0)
            {
                printf( "%d* ",i);
                n=n/i;
            } else{       break; }
        }
    printf( "%d ",n);
    getch();
}
```

21、在 main 方法中将字符串中的数字排序并输出 STRING A="56.89.5.3.75.98.98.26.15.44"

```
String s=" 56.89.5.3.75.98.98.26.15.44";
String s1[]=s. split (".");
Integer ii[]=new Integer[s1.length];
For(int i=0;i<s1.length;i++){
    ii[i]=Integer.parseInt(s1[i]);
}
Arrays.sort(ii);
for(Integer o: ii){
```

```
  System.out.println(o+"  s");
}
```

22、用 4 个 0，用你所知道的数学方法计算出 24

0 的阶乘等于 1　即 0！＝1 那么 4 个 0 就是 4 了

又 4 的阶乘为 2 4　 4！＝2 4

23、判断身份证：要么是 15 位，要么是 18 位，最后一位可以为字母，并写程序提出其中的年月日。

答：我们可以用正则表达式来定义复杂的字符串格式，(\d{17}[0-9a-zA-Z]|\d{14}[0-9a-zA-Z])可以用来判断是否为合法的 15 位或 18 位身份证号码。

因为 15 位和 18 位的身份证号码都是从 7 位到第 12 位为身份证为日期类型。这样我们可以设计出更精确的正则模式，使身份证号的日期合法，这样我们的正则模式可以进一步将日期部分的正则修改为[12][0-9]{3}[01][0-9][123][0-9]，当然可以更精确的设置日期。

在 jdk 的 java.util.Regex 包中有实现正则的类,Pattern 和 Matcher。以下是实现代码：

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class RegexTest {

    /**
     * @param args
     */
    public static void main(String[] args) {

        // 测试是否为合法的身份证号码
        String[] strs = { "130681198712092019", "13068119871209201x",
                "13068119871209201", "123456789012345", "12345678901234x",
                "1234567890123" };
        Pattern p1 = Pattern.compile("(\\d{17}[0-9a-zA-Z]|\\d{14}[0-9a-zA-Z])");
        for (int i = 0; i < strs.length; i++) {
            Matcher matcher = p1.matcher(strs[i]);
            System.out.println(strs[i] + ":" + matcher.matches());
        }

        Pattern p2 = Pattern.compile("\\d{6}(\\d{8}).*"); // 用于提取出生日字符串
        Pattern p3 = Pattern.compile("(\\d{4})(\\d{2})(\\d{2})");// 用于将生日字符串进行分解为年月日
        for (int i = 0; i < strs.length; i++) {
            Matcher matcher = p2.matcher(strs[i]);
            boolean b = matcher.find();
```

```
            if (b) {
                String s = matcher.group(1);
                Matcher matcher2 = p3.matcher(s);
                if (matcher2.find()) {
                    System.out
                            .println("生日为" + matcher2.group(1) + "年"
                                    + matcher2.group(2) + "月"
                                    + matcher2.group(3) + "日");
                }
            }

        }

    }

}
```

24、编写一个程序，将 a.txt 文件中的单词与 b.txt 文件中的单词交替合并到 c.txt 文件中，a.txt 文件中的单词用回车符分隔，b.txt 文件中用回车或空格进行分隔。

答：

```
package cn.itcast;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;

public class MainClass{
    public static void main(String[] args) throws Exception{
        FileManager a = new FileManager("a.txt",new char[]{'\n'});
        FileManager b = new FileManager("b.txt",new char[]{'\n',' '});
        FileWriter c = new FileWriter("c.txt");
        String aWord = null;
        String bWord = null;
        while((aWord = a.nextWord()) !=null ){
            c.write(aWord + "\n");
            bWord = b.nextWord();
            if(bWord != null)
                c.write(bWord + "\n");
        }

        while((bWord = b.nextWord()) != null){
            c.write(bWord + "\n");
        }
        c.close();
    }
```

```
}


class FileManager{

    String[] words = null;
    int pos = 0;
    public FileManager(String filename,char[] seperators) throws Exception{
        File f = new File(filename);
        FileReader reader = new FileReader(f);
        char[] buf = new char[(int)f.length()];
        int len = reader.read(buf);
        String results = new String(buf,0,len);
        String regex = null;
        if(seperators.length >1 ){
            regex = "" + seperators[0] + "|" + seperators[1];
        }else{
            regex = "" + seperators[0];
        }
        words = results.split(regex);
    }

    public String nextWord(){
        if(pos == words.length)
            return null;
        return words[pos++];
    }

}
```

25、编写一个程序，将 d:\java 目录下的所有.java 文件复制到 d:\jad 目录下，并将原来文件的扩展名从.java 改为.jad。

答：listFiles 方法接受一个 FileFilter 对象，这个 FileFilter 对象就是过虑的策略对象，不同的人提供不同的 FileFilter 实现，即提供了不同的过滤策略。

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FilenameFilter;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
public class Jad2Java {
    public static void main(String[] args) throws Exception {
```

```
        File srcDir = new File("java");
        if(!(srcDir.exists() && srcDir.isDirectory()))
                throw new Exception("目录不存在");
        File[] files = srcDir.listFiles(
            new FilenameFilter(){

                    public boolean accept(File dir, String name) {
                        return name.endsWith(".java");
                    }
                }
        );
        System.out.println(files.length);
        File destDir = new File("jad");
        if(!destDir.exists()) destDir.mkdir();
        for(File f :files){
            FileInputStream  fis = new FileInputStream(f);
            String destFileName = f.getName().replaceAll("\\.java$", ".jad");
            FileOutputStream      fos      =      new      FileOutputStream(new
File(destDir,destFileName));
            copy(fis,fos);
            fis.close();
            fos.close();
        }
    }

    private static void copy(InputStream ips,OutputStream ops) throws Exception{
        int len = 0;
        byte[] buf = new byte[1024];
        while((len = ips.read(buf)) != -1){
            ops.write(buf,0,len);
        }

    }
}
```

由本题总结的思想及策略模式的解析：

1.

class jad2java{

1. 得到某个目录下的所有的 java 文件集合

1.1 得到目录 File srcDir = new File("d:\\java");

1.2 得到目录下的所有 java 文件：File[] files = srcDir.listFiles(new MyFileFilter());

1.3 只想得到.java 的文件： class MyFileFilter implememyts FileFilter{

public boolean accept(File pathname){

```
                return pathname.getName().endsWith(".java")
            }
        }
```

2.将每个文件复制到另外一个目录，并改扩展名

2.1 得到目标目录，如果目标目录不存在，则创建之

2.2 根据源文件名得到目标文件名，注意要用正则表达式，注意.的转义。

2.3 根据表示目录的 File 和目标文件名的字符串，得到表示目标文件的 File。

//要在硬盘中准确地创建出一个文件，需要知道文件名和文件的目录。

2.4 将源文件的流拷贝成目标文件流，拷贝方法独立成为一个方法，方法的参数采用抽象流的形式。

//方法接受的参数类型尽量面向父类，越抽象越好，这样适应面更宽广。

```
}
```

分析 listFiles 方法内部的策略模式实现原理

```
File[] listFiles(FileFilter filter){
    File[] files = listFiles();
    //Arraylist acceptedFilesList = new ArrayList();
    File[] acceptedFiles = new File[files.length];
    int pos = 0;
    for(File file: files){
        boolean accepted = filter.accept(file);
        if(accepted){
            //acceptedFilesList.add(file);
            acceptedFiles[pos++] = file;
        }
    }

    Arrays.copyOf(acceptedFiles,pos);
    //return (File[])accpetedFilesList.toArray();

}
```

26、编写一个截取字符串的函数，输入为一个字符串和字节数，输出为按字节截取的字符串，但要保证汉字不被截取半个，如“我 ABC”，4，应该截取“我 AB”，输入“我 ABC 汉 DEF”，6，应该输出“我 ABC”，而不是“我 ABC+汉的半个”。

答：

首先要了解中文字符有多种编码及各种编码的特征。

假设 n 为要截取的字节数。

```
public static void main(String[] args) throws Exception{
    String str = "我 a 爱中华 abc def';
    String str = "我 ABC 汉";
    int num = trimGBK(str.getBytes("GBK"),5);
    System.out.println(str.substring(0,num) );
```

```
    }

    public static int  trimGBK(byte[] buf,int n){
        int num = 0;
        boolean bChineseFirstHalf = false;
        for(int i=0;i<n;i++)
        {
            if(buf[i]<0 && !bChineseFirstHalf){
                bChineseFirstHalf = true;
            }else{
                num++;
                bChineseFirstHalf = false;
            }
        }
        return num;
    }
```

27、有一个字符串，其中包含中文字符、英文字符和数字字符，请统计和打印出各个字符的个数。

答：String content = “中国 aadf 的 111 萨 bbb 菲的 zz 萨菲”;

```
HashMap map = new HashMap();
for(int i=0;i<content.length;i++)
{
    char c = content.charAt(i);
    Integer num = map.get(c);
    if(num == null)
        num = 1;
    else
        num = num + 1;
    map.put(c,num);
}
for(Map.EntrySet entry : map)
{
    system.out.println(entry.getkey() + “:” + entry.getValue());
}
```

如果一串字符如"aaaabbc中国1512"要分别统计英文字符的数量，中文字符的数量，和数字字符的数量，假设字符中没有中文字符、英文字符、数字字符之外的其他特殊字符。

```
int engishCount;
int chineseCount;
int digitCount;
for(int i=0;i<str.length;i++)
{
    char ch = str.charAt(i);
    if(ch>=’0’  && ch<=’9’)
```

```
    {
        digitCount++
    }
    else if((ch>=' a' && ch<=' z' ) || (ch>=' A' && ch<=' Z' ))
    {
        engishCount++;
    }
    else
    {
        chineseCount++;
    }
}
System.out.println(……);
```

28、从类似如下的文本文件中读取出所有的姓名，并打印出重复的姓名和重复的次数，并按重复次数排序：

1,张三,28
2,李四,35
3,张三,28
4,王五,35
5,张三,28
6,李四,35
7,赵六,28
8,田七,35

```
package com.huawei.interview;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.Comparator;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.TreeSet;
public class GetNameTest {
    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        //InputStream ips =
GetNameTest.class.getResourceAsStream("/com/huawei/interview/info.txt");
        //用上一行注释的代码和下一行的代码都可以，因为info.txt与GetNameTest类在
```

同一包下面，所以，可以用下面的相对路径形式

```
        Map results = new HashMap();
        InputStream ips = GetNameTest.class.getResourceAsStream("info.txt");
        BufferedReader in = new BufferedReader(new InputStreamReader(ips));
        String line = null;
        try {
            while((line=in.readLine())!=null)
            {
                dealLine(line,results);
            }
            sortResults(results);
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    static class User
    {
        public  String name;
        public Integer value;
        public User(String name,Integer value)
        {
            this.name = name;
            this.value = value;
        }

        @Override
        public boolean equals(Object obj) {
            // TODO Auto-generated method stub

            //下面的代码没有执行，说明往treeset中增加数据时，不会使用到equals方
法。
            boolean result = super.equals(obj);
            System.out.println(result);
            return result;
        }
    }

    private static void sortResults(Map results) {
        // TODO Auto-generated method stub
        TreeSet sortedResults = new TreeSet(
                new Comparator(){
                    public int compare(Object o1, Object o2) {
```

```
                        // TODO Auto-generated method stub
                        User user1 = (User)o1;
                        User user2 = (User)o2;
                        /*如果compareTo返回结果0，则认为两个对象相等，新的对象不会增加到集合中去
                         * 所以，不能直接用下面的代码，否则，那些个数相同的其他姓名就打印不出来。
                         * */

                        //return user1.value-user2.value;
                        //return user1.value<user2.value?-1:user1.value==user2.value?0:1;
                        if(user1.value<user2.value)
                        {
                            return -1;
                        }else if(user1.value>user2.value)
                        {
                            return 1;
                        }else
                        {
                            return user1.name.compareTo(user2.name);
                        }
                    }

                }
        );
        Iterator iterator = results.keySet().iterator();
        while(iterator.hasNext())
        {
            String name = (String)iterator.next();
            Integer value = (Integer)results.get(name);
            if(value > 1)
            {
                sortedResults.add(new User(name,value));
            }
        }

        printResults(sortedResults);
    }
    private static void printResults(TreeSet sortedResults)
    {
        Iterator iterator  = sortedResults.iterator();
        while(iterator.hasNext())
        {
```

```
            User user = (User)iterator.next();
            System.out.println(user.name + ":" + user.value);
        }
    }
    public static void dealLine(String line,Map map)
    {
        if(!"".equals(line.trim()))
        {
            String [] results = line.split(",");
            if(results.length == 3)
            {
                String name = results[1];
                Integer value = (Integer)map.get(name);
                if(value == null) value = 0;
                map.put(name,value + 1);
            }
        }
    }

}
```

29、写一个 Singleton 出来。

第一种：饱汉模式

```
public class SingleTon {
    private SingleTon(){
        }
    //实例化放在静态代码块里可提高程序的执行效率，但也可能更占用空间
    private final static SingleTon instance = new SingleTon();
    public static SingleTon getInstance(){
        return instance;
    }
}
```

第二种：饥汉模式

```
public class SingleTon {
    private SingleTon(){}

    private static instance = null;//new SingleTon();

    public static synchronized SingleTon getInstance(){
        if(instance == null)
            instance = new SingleTon();
        return instance;
    }
}
```

第三种：用枚举

```
    public enum SingleTon{
        ONE;

    }
```

第三：更实际的应用（在什么情况用单例）

```
public class SequenceGenerator{
    //下面是该类自身的业务功能代码
    private int count = 0;

    public synchronized int getSequence(){
        ++count;
    }

    //下面是把该类变成单例的代码
    private SequenceGenerator(){}
    private final static instance = new SequenceGenerator();
    public static SingleTon getInstance(){
        return instance;
    }

}
```

第四：

```
    public class MemoryDao
    {
    private HashMap map = new HashMap();

    public void add(Student stu1){
            map.put(SequenceGenerator.getInstance().getSequence(),stu1);
    }

   //把MemoryDao变成单例
  }
```

Singleton 模式主要作用是保证在 Java 应用程序中，一个类 Class 只有一个实例存在。

一般 Singleton 模式通常有几种种形式:

第一种形式: 定义一个类，它的构造函数为 private 的，它有一个 static 的 private 的该类变量，在类初始化时实例话，通过一个 public 的 getInstance 方法获取对它的引用,继而调用其中的方法。

```
public class Singleton {
private Singleton(){}
```

```
        //在自己内部定义自己一个实例，是不是很奇怪？
        //注意这是 private 只供内部调用
        private static Singleton instance = new Singleton();
        //这里提供了一个供外部访问本 class 的静态方法，可以直接访问
        public static Singleton getInstance() {
            return instance;
        }
    }
```

第二种形式:

```
public class Singleton {
    private static Singleton instance = null;
    public static synchronized Singleton getInstance() {
    //这个方法比上面有所改进，不用每次都进行生成对象，只是第一次
    //使用时生成实例，提高了效率！
    if (instance==null)
        instance＝new Singleton();
        return instance;
  }
}
```

其他形式:

定义一个类，它的构造函数为 private 的，所有方法为 static 的。

一般认为第一种形式要更加安全些

30、一个整数，大于 0，不用循环和本地变量，按照 n，2n，4n，8n 的顺序递增，当值大于 5000 时，把值按照指定顺序输出来。

例：n=1237

则输出为：

1237，

2474，

4948，

9896，

9896，

4948，

2474，

1237，

提示：写程序时，先致谢按递增方式的代码，写好递增的以后，再增加考虑递减部分。

```
public static void doubleNum(int n)
{
    System.out.println(n);
    if(n<=5000)
        doubleNum(n*2);
    System.out.println(n);
}
```

Gaibaota(N) = Gaibaota(N-1) + n

31、第 1 个人 10，第 2 个比第 1 个人大 2 岁，依次递推，请用递归方式计算出第 8 个人多大？

```
package cn.demo;
import java.util.Date;
public class A1 {

    public static void main(String [] args)
    {
        System.out.println(computeAge(8));
    }

    public static int computeAge(int n)
    {
        if(n==1) return 10;
        return computeAge(n-1) + 2;
    }
}
    public static void toBinary(int n,StringBuffer result)
    {

        if(n/2 != 0)
            toBinary(n/2,result);
        result.append(n%2);
    }
```

32、排序都有哪几种方法？请列举。用 JAVA 实现一个快速排序。

交换式排序、选择排序、插入排序、希尔排序、快速排序

```
public class QuickSort {
/**
* 快速排序
* @param strDate
* @param left
* @param right
*/
public void quickSort(String[] strDate,int left,int right){
String middle,tempDate;
int i,j;
i=left;
j=right;
middle=strDate[(i+j)/2];
do{
    while(strDate[i].compareTo(middle)<0&& i<right)
    i++; //找出左边比中间值大的数
```

```
    while(strDate[j].compareTo(middle)>0&& j>left)
    j--; //找出右边比中间值小的数
if(i<=j){ //将左边大的数和右边小的数进行替换
    tempDate=strDate[i];
    strDate[i]=strDate[j];
    strDate[j]=tempDate;
    i++;
    j--;
}
}while(i<=j); //当两者交错时停止

if(i<right){
quickSort(strDate,i,right);//从
}
if(j>left){
quickSort(strDate,left,j);
}
}
/**
   * @param args
   */
public static void main(String[] args){
String[] strVoid=new String[]{"11","66","22","0","55","22","0","32"};
QuickSort sort=new QuickSort();
sort.quickSort(strVoid,0,strVoid.length-1);
for(int i=0;i<strVoid.length;i++){
System.out.println(strVoid[i]+" ");
}
}
}
```

33、有数组 a[n]，用 java 代码将数组元素顺序颠倒

```
//用下面的也可以
//for(int i=0,int j=a.length-1;i<j;i++,j--) 是否等效于 for(int
i=0;i<a.length/2;i++)呢?
import java.util.Arrays;
public class SwapDemo{

    public static void main(String[] args){
        int [] a = new int[]{
                        (int)(Math.random() * 1000),
                        (int)(Math.random() * 1000),
```

```
                    (int)(Math.random() * 1000),
                    (int)(Math.random() * 1000),
                    (int)(Math.random() * 1000)

        };

        System.out.println(a);
        System.out.println(Arrays.toString(a));
        swap(a);
        System.out.println(Arrays.toString(a));
    }

    public static void swap(int a[]){
        int len = a.length;
        for(int i=0;i<len/2;i++){
            int tmp = a[i];
            a[i] = a[len-1-i];
            a[len-1-i] = tmp;
        }
    }
}
```

34、写一个方法，用一个 for 循环打印九九乘法表

```
        /**  *//**
         *  打印九九乘法口诀表
         */
        public  void  nineNineMulitTable(){
                for  (int  i  =  1,j  =  1;  j  <=  9;  i++)  {
                        System.out.print(i+"*"+j+"="+i*j+"  ");
                        if(i==j){
                                i=0;
                                j++;
                          System.out.println();
                        }
                }
        }
```

35、给定一个 java.util.Date 对象，如何转化为”2007-3-22 20:23:22”格式的字符串。【高达软件】

```
        /**  *//**
         *  将某个日期以固定格式转化成字符串
         *  @param  date
        *  @return  str
        */
```

```
        public  String  date2FormatStr(Date  date)
        {
            SimpleDateFormat  sdf  =  new  SimpleDateFormat("yyyy-MM-
dd  HH:mm:ss");
            String  str  =  sdf.format(date);
            return  str;
        }
```

36、金额转换，阿拉伯数字的金额转换成中国传统的形式如：（￥1011）—>（一千零一拾一元整）输出。

去零的代码：

return sb.reverse().toString().replaceAll(" 零 [拾 佰 仟]"," 零 ").replaceAll(" 零 + 万 "," 万 ").replaceAll("零+元","元").replaceAll("零+","零");

```
public class RenMingBi {

    /**
     * @param args add by zxx ,Nov 29, 2008
     */
    private static final char[] data = new char[]{
            '零','壹','贰','叁','肆','伍','陆','柒','捌','玖'
        };
    private static final char[] units = new char[]{
        '元','拾','佰','仟','万','拾','佰','仟','亿'
    };
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println(
                convert(135689123));
    }

    public static String convert(int money)
    {
        StringBuffer sbf = new StringBuffer();
        int unit = 0;
        while(money!=0)
        {
            sbf.insert(0,units[unit++]);
            int number = money%10;
            sbf.insert(0, data[number]);
            money /= 10;
        }
```

```
        return sbf.toString();
    }
    }
```

37、写一个方法，能够判断任意一个整数是否素数

```
/** *//**
     * 判断任意一个整数是否素数
     * @param num
     * @return boolean
     */
     public boolean isPrimeNumber(int num)
     {
        for (int i = 2; i <= Math.sqrt(num); i++) {
            if(num%i==0)
            {
                return false;
            }
        }
        return true;
     }
```

38、用 1、2、3、4、5 这 5 个数字，用 Java 写一个 Main 函数，打印所有不同的排序

```
static int[] bits = new int[] { 1, 2, 3, 4, 5 };
/**
 * @param args
 */
public static void main(String[] args) {
sort("", bits);
}
private static void sort(String prefix, int[] a) {
if (a.length == 1) {
System.out.println(prefix + a[0]);
}
for (int i = 0; i < a.length; i++) {
sort(prefix + a[i], copy(a, i));
}
}
private static int[] copy(int[] a,int index){
int[] b = new int[a.length-1];
System.arraycopy(a, 0, b, 0, index);
System.arraycopy(a, index+1, b, index, a.length-index-1);
return b;
}
```

39、 写一个方法，输入任意一个整数，返回它的阶乘

```
/** *//**
 *获得任意一个整数的阶乘
 n
 !
 */
 public int factorial(int num)
 {
  //递归
  if(num == 1)
  {
     return 1;
  }
  return num*factorial(num-1);
 }
```

40、写一个方法，用二分查找法判断任意整数在任意整数数组里面是否存在，若存在就返回它在数组中的索引位置，不存在返回-1

```
/**  *//**
  *二分查找特定整数在整型数组中的位置(递归)
  dataset
  data
  beginIndex
  endIndex
  index
  */
   public  int  binarySearch(int[]  dataset,int  data,int
beginIndex,int  endIndex){
        int  midIndex  =  (beginIndex+endIndex)/2;
        //如果查找的数要比开始索引的数据要小或者是比结束索
引的书要大，或者开始查找的索引值大于结束的索引值返回-1 没有查到
        if(data  <dataset[beginIndex]||data>dataset[endInde
x]||beginIndex>endIndex){
                return  -1;
        }
        if(data  <dataset[midIndex]){
                return  binarySearch(dataset,data,beginInde
x,midIndex-1);
        }else  if(data>dataset[midIndex])
        {
                return  binarySearch(dataset,data,midIndex+
1,endIndex);
        }else  {
```

```
                    return  midIndex;
                }
            }

            /**  *//**
              *二分查找特定整数在整型数组中的位置(非递归)
              dataset
              data
              index
              */
              public  int  binarySearch(int[]  dataset  ,int  data)
              {
                  int  beginIndex  =  0;
                  int  endIndex  =  dataset.length  -  1;
                  int  midIndex  =  -1;
                  if(data  <dataset[beginIndex]||data>dataset[endIn
dex]||beginIndex>endIndex){
                          return  -1;
                  }
                  while(beginIndex  <=  endIndex)  {
                          midIndex  =  (beginIndex+endIndex)/2;
                          if(data  <dataset[midIndex])  {
                                endIndex  =  midIndex-1;
                          }  else  if(data>dataset[midIndex])  {
                              beginIndex  =  midIndex+1;
                          }else  {
                              return  midIndex;
                          }
                  }
                  return  -1;
              }
```

41、如果一个序列的前四个数字分别是 2,9,28,65 请问第五个数字是?

$1^3+1=2$

$2^3+1=9$

$3^2+1=28$

$4^3+1=65$

所以继续的话应是

$5^3+1=126$

42、编写函数找出 1 到 1000 之内能被 3 整除且不是偶数的整数，并按个位数的大小从大到小排序

```
public List find() {
List list=new ArrayList();
    for (int i = 1; i <=1000; i++) {
    if (i%3==0 && i%2!=0) {
```

```
        list.add(i);
        }
        }
        return list;
}
```

43、用 java 方法编写计算指定目录下所有文件占空间大小

```
//返回文件大小 private void getFileSize()throws RuntimeException,IOException
 {//初始化文件大小为 0；
        this.longSize=0;
        //如果文件存在而且是文件，直接返回文件大小
        if(file.exists()&&file.isFile()){
          this.longSize= file.length();
        }
// 文 件 存 在 而 且 是 目 录 ， 递 归 遍 历 文 件 目 录 计 算 文 件 大 小      else
if(file.exists()&&file.isDirectory()){
          getFileSize(file);//递归遍历
 }else{
          throw new RuntimeException("指定文件不存在");
        }
 }
```

44、一个 list 中，有 b.a.b.c.b.b.写个方法去掉所有 b

```
        public List qub(List list) {
        List alist=new ArrayList();
        for (int i = 0; i < list.size(); i++) {
        String a=(String)list.get(i);
        if (!a.equals("b")) {
        alist.add(a);
        }
        }
        return alist;
        }
```

45、设计线程的生产者和消费者模式

```
package com;

import java.io.*;
import java.util.*;
public class Test1 {
    Vector v = new Vector();
    int index = -1;
    boolean isput = false;
    public synchronized void put(String name) throws InterruptedException{
          if(isput){
              wait();
          }
```

```
            v.add(name);
            index++;
            isput = true;
            notify();
            System.out.println(Thread.currentThread().getName());
    }
    public synchronized  void get() throws InterruptedException{
        if(!isput){
            wait();
        }
        System.out.println(Thread.currentThread().getName()+"
取:"+v.get(index));
        isput = false;
        notify();
    }
    public static void main(String[] args) throws CloneNotSupportedException,
FileNotFoundException, IOException, InterruptedException {
            Test1 t = new Test1();
            A a = new A(t);
            B b = new B(t);
            new Thread(a).start();
            new Thread(b).start();
    }
}
class A implements Runnable{
    Test1 t;
    public A(Test1 t){
        this.t = t;
    }
    @Override
    public void run() {
        int i  =0 ;
        while(true){
            if(i==0){
                try {
                    t.put("男");
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }else{
                try {
                    t.put("女");
                } catch (InterruptedException e) {
```

```
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
            i = (i+1)%2;
        }
    }
}
class B implements Runnable{
    Test1 t;
    public B(Test1 t){
        this.t = t;
    }
    @Override
    public void run() {
        // TODO Auto-generated method stub
        while(true){
            try {
                t.get();
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

46、写一个方法求两个数的公约数

```
import java.util.ArrayList;
import java.util.List;

public class MaxNum {
    int num=0;
    public List Maxs(int a){//先找出能被第一个数整除的数
        List list=new ArrayList();
        for (int i = 1; i <= a; i++) {
            if(a%i==0){
                list.add(i);
            }
        }
        return list;
    }
    public void Test(int a,int b){
        List list=new MaxNum().Maxs(a);
```

```
        for (int i = 0; i < list.size(); i++) {
            int bb=(Integer) list.get(i);
            if(b%bb==0){//找出能被第二个数整出并且也能被第二个数整除的数
                num=bb;
            }
        }
        System.out.println("最大公约数为："+num);
    }
    public static void main(String[] args) {
        new MaxNum().Test(100,1000);
    }

    }
```

47、实现字符串的反转。如：abcd 输出 dcba

答：

```
StringBuffer stringBuffer =
new StringBuffer().append("abc").reverse();
System.out.println(stringBuffer.toString());
```

48、编写程序将由数字及字符组成的字符串中的数字截取出来并按顺序输出，例如：“ABC137GMNQQ2049PN5FFF”输出结果应该为 01234579

```
package com.tarena;
import java.util.Arrays;
public class NumberSplitChar {
public static void main(String[] args) {
String str=”ABC137GMNQQ2049PN5FFF”;
char[] beforechars=str.toCharArray();
char[] afterchars=new char[beforechars.length];
int j=0;
for(int i=0;i<beforechars.length;i++){
if(beforechars[i]>=’0′ && beforechars[i]<=’9′){
afterchars[j++]=beforechars[i];
}
}
Arrays.sort(afterchars);
for(int i=(afterchars.length-j);i<afterchars.length;i++){
System.out.print(afterchars[i]);
}
}
}
```

五、现场招聘考试题

中*国际笔试题（一）

一、判断题（共 10 题，每小题 1 分，共 10 分，占 10%）

1．java 程序里,创建新的类对象用关键字 new。

2．类及其属性、方法可以同时有一个以上的修饰符来修饰。

3．抽象方法必须在抽象类中，所以抽象类中的方法都必须是抽象方法。

4．父类方法不能被子类方法覆盖。

5．一个 Java 类可以有多个父类。

6．一个数组中能够存储不同类型的数据。

7．final 类型的变量是符号常量，其值不能改变。

8．无论 Java 源程序包含几个类的定义，若该源程序文件以 B.java 命名，编译后生成的都只有一个名为 B.class 的字节码文件。

9．Java 类中不能存在同名的两个成员方法。

10．Java 源程序文件中是不区分字母的大小写的。

二、单项选择题（共 20 题，每小题 2 分，共 40 分，占 40%）

说明：请将正确答案填写在后面表格的相应位置上，否则不得分。

1．以下代码段执行后的输出结果为(　　)

```
    int   x=3;  int   y=10;
System.out.println(y%x);
```

（A）0　　（B）2　　（C）1　　（D）3

2．下列叙述中，错误的是(　　)

(A)父类不能替代子类　　(B)子类能够替代父类

(C)子类继承父类　　(D)父类包含子类

3．阅读下列代码后

```
public   class Person
  {public   static   void main(String args[ ])
     {int arr=new int[10];
       System.out.println(arr[1]);}}
```

正确的说法是(　　)

(A)编译时将产生错误　　(B)编译时正确，运行时将产生错误

(C)输出零　　(D)输出空

4. 在类方法中声明的变量，属于(　　)。

(A)局部变量　　(B)成员变量

(C)方法参数　　(D)异常处理参数

5. main 方法是 Java　Application 程序执行的入口点，关于 main 方法的方法头以下哪项是合法的(　　)

(A)public　static　void　main ()

(B)public　static　void　main (String[ ]　args)

(C)public static int　main (String[ ] arg)

(D)public　void　main (String　arg[ ])

6. 关于被私有访问控制符 private 修饰的成员变量，以下说法正确的是(　　)

（A）可以被三种类所引用：该类自身、与它在同一个包中的其他类、在其他包中的该类的子类

（B）可以被两种类访问和引用：该类本身、该类的所有子类

（C）只能被该类自身所访问和修改

（D）只能被同一个包中的类访问

7.类 Test1 定义如下：

```
1. public　class　Test1{
2.　　public　float　aMethod (float　a, float　b) {　}
3.
4. }
```

将以下哪种方法放在第 3 行是不合法的。(　　)

(A) public　float　aMethod (float　a,　float　b, float　c) {　}

(B) public　float　aMethod (float　c, float d) {　}

(C) public　int　aMethod (int　a,　int b) {　}

(D) private float　aMethod (int a, int b, int c) {　}

8. 设数组 Array 由以下语句定义

```
int age[ ]=new int[10];
```

则数组的第一个元素的正确引用方法为(　　)

A、age[1]　　B、Age[0]　　C、age[0]　　D、age[]

9. 设 x=5 则 y=x-- 和 y=--x 的结果，使 y 分别为(　　)

（A）5,5　　（B）5,6　　（C）5,4　　（D）4,4

10.一个可以独立运行的 Java 应用程序(　　)

(A) 可以有一个或多个 main 方法　　(B)只能有两个 main 方法

(C)可以有一个或零个 main 方法　　(D)只能有一个 main 方法

11. Java 用来定义一个新类时，所使用的关键字为 (　　)

（A）class　（B）public　（C）struct　（D）class 或 struct

12. 下面程序运行结果为(　　)

```
int j=0;
for(int i=0;i<=10;j+=i,i++);
System.out.print(j);
```

(A)45　　(B)55　　(C)50　　(D) 编译时不能通过

13. 以下程序执行后输出结果是(　　)。

```
public　class test
```

```
      {public   static   void main(String[ ]     args)
{int i;
for(i=1;i<5;i++)
switch(i)
{case 1:System.out.print(“first-”);
  case 2:System.out.print(“second-”); break;
  default:System.out.print(“others-”);
}}}
```

(A) first-second-others-second-others-others-others
(B) first-second-others- others-
(C) first-second- second- others-others-
(D) first-second- second-others-

14. Java 语言是(　　)。
(A)面向问题的解释型高级编程语言　　(B) 面向机器的低级编程语言
(C)面向过程的编译型高级编程语言　　(D) 面向对象的解释型高级编程语言

15. for(;;)是(　　)
（A）循环结构　　（B）分支结构　　（C）顺序结构

16. 设有对象 x 具有属性 a 则访问该属性的方法为(　　)
（A）a.x　　（B）a.x()　　（C）x.a　　（D）x.a()

17. System.out.print(12+3)的输出结果是(　　)
(A)15　　(B)123　　(C)”123 ”　　(D)”15”

18. 在 java 语言中，只有整型数据才能进行的运算是(　　)
(A)　*　　(B)　/　　(C)　%　　(D)　+

19.. 看以下程序：

```
boolean a=false;
boolean b=true;
boolean c=(a&&b)&&(!b);
int result=c= =false?1:2;
```

这段程序执行完后，c 与 result 的值是(　　)
(A) c=false;result=1; (B)c=true;result=2; (C)c=true;result=1;
(D)c=false;result=2;

20. 用 public 修饰的类称为(　　)
(A)静态类　　(B)抽象类　　(C)最终类　　(D)公有类

三、填空题（共 14 题，每空 1 分，共 20 分，占 20%）

1. 面向程序设计中的对象用_____表示其属性；用_____描述其行为。
2. 若 x = 5，y = 10，则 x < y 和 x >= y 的逻辑值分别为_____和_____。
3. 在类中可以定义多个具有相同名称、但参数不同的方法，该做法称为_____。
4. 数学中的 x 变量值范围是[-3,5]，那么在 java 程序中应该表示成 _____
若范围是[-∞,3]和[8,+∞]，则表示成_____ 。
5. _____方法是一种仅有方法头，没有具体方法体和操作实现的方法，该方法必须在抽象类之中定义。
6. System.out.println(015)的结果是_____。

7. 省略访问修饰符的类只能被_____中的类使用，称之具有_____访问特性。
8. 如果子类定义的成员变量与父类的成员变量同名，要表明使用子类的成员变量，可以在成员变量前加上关键字_____ 。
9. 在Java的基本数据类型中,char型采用Unicode编码方案,每个Unicode码占用_____字节内存空间，这样，无论是中文字符还是英文字符，都是占用_____字节内存空间。
10. 在JAVA语言中，字符串“ABC\tD\b\n”包括_____个字符。
11. 在Java程序设计中，一维数组的声明格式是_____。
12. 程序从_____类开始执行。
13. 以下方法fun的功能是求两个参数之和。

```
int fun ( int a, int b )
{ _____; }
```

14. 下面的程序创建一个3╳4二维数组，该数组元素的值为行和列的下标之和。请在横线处填入适当内容，使程序能正确执行。

```
 public class test
{
public   static   void main(String   args[ ])
{int arrayint=new int [3][4];
for(int i=0;i<=_____   ;i++ )
for(int j=0;j<=_____   ;j++ )}}
 arrayint[i][j]=i+j;}}
```

四、写出下列程序的运行结果（共3题，第1题8分，后2题每题6分，共20分，占20%）

1.

```
 class   Father
{  int x=0;
  public   void output( ){System.out.println(x);}}
class Son extends Father
{int x=2;
         public void output(int x)
         {super.output( );
System.out.println(this.x);
System.out.println(x);
System.out.println(super.x);}}
class Exclass
{   public static void main(String args[])
       {Father ob1=new Father();
         ob1.output( );
Son ob2=new Son();
         ob2.output(4 );
       }}
```

2.

```
public class ArrDemo2
{public static void main(String[] args)
                    { int a[ ][ ]=new int[5][5];
                                  for(int i=0;i<=a.length-1;i++)
                                       { a[i][0]=1;a[i][i]=1; }
                                    for(int i=2;i<=a.length-1;i++)
```

```
                                        for(int j=1;j<=i-1;j++)
                                      {a[i][j]=a[i-1][j-1]+a[i-1]
[j]; }
                                for(int i=0;i<=a.length-1;i++)
                                        {for(int j=0;j<=i;j++)
                                 System.out.print(" "+a[i][j]); }}}
```

五、编程题（共 1 题，每小题 10 分，共 10 分，占 10%）

1. 该程序共包括 2 个类，一个类为 Another，要求在其中定义方法 mul 实现 s=10!，返回值为整型，public 类为 Test，其 main 方法输出 10！的结果。

东南融*笔试题（二）

一、选择题。

1、关于 spring 说法错误的是（）

A. spring 是一个轻量级 JAVA EE 的框架集合

B. spring 包含一个“依赖注入”模式的实现

C. 使用 spring 可以实现声明事务

D. spring 提供了 AOP 方式的日志系统

2、关于依赖注入说法正确的是（）

A. 依赖注入的目标是在代码之外管理程序组建间的依赖关系

B. 依赖注入即是“面向接口”的编程

C. 依赖注入是面向对象技术的替代品

D. 依赖注入的使用会增大程序的规模

3、关于 AOP 错误的是（）

A. AOP 将散落在系统中的“方面”代码集中实现

B. AOP 有助于提高系统可维护性

C. AOP 已经表现出将要替代面向对象的趋势

D. AOP 是一种设计模式，Spring 提供了一种实现

4、关于 spring AOP 错误的是（）

A. 首先要编写方面代码，实现 MethodBeforeAdvice 接口

B. AOP 采取拦截方法调用方式实现，可以在调用方法前，后，抛出异常时拦截

C. AOP 采取代理的方式实现，常用代理对象的类别为
org.springframework.aop.framework.ProxyFactoryBean

D. AOP 可以在对目标（target）不做任何修改的情况下增加程序功能

5、在 Spring 配置文件 di.xml 中包含下面的配置代码，可以推断（）。

A. 下列语句运行时无异常
ApplicationContext context=new ClassPathXmlApplicationContext(“di.xml”);
A a=(A)content.getBean(“a”);

B. A 是抽象类

C. A 是接口

D. 类 A 中定义了 getM()方法

6、关于 Struts+Spring+Hibernate 集成框架，下列说法正确的是（）。

A. 使用 SSH 框架开发项目，由于增加了大量配置工作，从而会降低开发的效率

B. 三个框架可以用多种方式进行集成

C. 使用 SSH 框架开发不利于实现复杂的业务逻辑

D. 使用 SSH 框架开发不利于构造复杂的页面和交互流程

7、对于 Sping 和 Struts 的集成，下列说法错误的是（）。（选 1 项）

A. Action Bean 的实例可以由 Spring 创建
B. 在 Struts 配置文件中，Action Bean 的配置可以如下所示：
type= “org.springframework.web.struts.DelegatingActionProxy”>
C. 在 Spring 配置文件中，Action Bean 的配置可以如下所示：
D. 由于 Spring 本身也提供了 MVC 实现，所以不能与 Struts 集成

8、在 Spring 框架中，面向方面编程（AOP）的目标在于（）。 （选 1 项）
A. 编写程序时不用关心其依赖组件的实现
B. 将程序中涉及的公用问题集中解决
C. 封装 JDBC 访问数据库的代码，简化数据访问层的重复性代码
D. 实现页面的“无刷新”

9、在 Spring 中，配置 Hibernate 事务管
理器，（HibernateTransactionManager）时，需要注入的属性名称是（）。 （选 1 项）
A. dataSource
B. sessionFactory
C. baseHibernateDao
D. transactionProxyFactoryBean

10、在 Spring 的配置文件中，包含如下所示的配置代码，则下面说法正确的是
（）。
A. 可以通过下列代码获取对象实例：
ApplicationContext context = new
ClassPathXmlApplicationContext(“di.xml”);
TestBean t = (TestBean)context.getBean(“test. TestBean”);
B. TestBean 中一定有如下语句：
private String dp = “”;
C. TestBean 中一定有方法：public void setSp(String value)
D. 属性 dp 的类型可能是 int 也可能是 String

11.下面关于“依赖注入”的说法，错误的是（）。
A. 将组件间的依赖关系采取配置文件的方式管理，而不是硬编码在代码中
B. 降低了组件间的耦合，使程序更容易维护和升级
C. 促进了“面向接口”编程，使构建大规模程序更轻松
D. 需要定义大量接口，增加了编码复杂度

12.关于 Spring 的说法错误的是（）。
A. 通过 setter 方法实现依赖注入
B. 对 Hibernate 提供了支持，可简化 Hibernate 编码
C. 通过 AOP 可以轻松与 Hibernate 集成
D. 通过 AOP 实现了声明式事务管理

13.关于 Spring AOP，下面说法错误的是（）。
A. 支持前置通知、后置通知、环绕通知

B. 采用代理的方式实现
C. 在 Spring 2.0 以前的版本中，通过、和配置
D. 与“依赖注入”实现紧密结合

14. 下面 Spring 声明式事务的配置中，存在哪些错误？
A.应为 execution(* com.conghai.isale.biz.*(..))
B.应为 execution(* com.conghai.isale.biz.*.*(..))
C.应为 D. pointcut-ref=”txAdvice”
D.应为 pointcut-ref=”bizMethods”

二、简答题（18 题）

1. spring 工作机制？

2. Jdo 是什么？

3. 简述你对 IoC（Inversion of Control）的理解，

4. 简单描述 Spring framework 与 Struts 的不同之处，整合 Spring 与 Struts 有哪些方法，哪种最好，为什么？

5. 什么是 AOP？

6. 说出 Spring 的通知类型？

7. 简述 ProxyFactoryBean 的作用？

8. 叙述 Spring 对持久层支持采用的策略？

9. 叙述 AOP 事务的含义？

10. 简述 Spring 的事务机制？

深圳华*笔试题（三）

一、单项选择题

1．Java 是从（ ）语言改进重新设计。

A．Ada B．C++ C．Pasacal D．BASIC

2．下列语句哪一个正确（ ）

A． Java 程序经编译后会产生 machine code

B． Java 程序经编译后会产生 byte code

C． Java 程序经编译后会产生 DLL

D． 以上都不正确

3．下列说法正确的有（ ）

A． class 中的 constructor 不可省略

B． constructor 必须与 class 同名，但方法不能与 class 同名

C． constructor 在一个对象被 new 时执行

D． 一个 class 只能定义一个 constructor

4．提供 Java 存取数据库能力的包是（ ）

A．java.sql B．java.awt C．java.lang D．java.swing

5．下列运算符合法的是（ ）

A．&& B．<> C．if D．:=

6．执行如下程序代码

```
a=0;c=0;
do{
    --c;
    a=a-1;
}while(a>0);  后，C 的值是（ ）
```

A．0 B．1 C．-1 D．死循环

7．下列哪一种叙述是正确的（ ）

A． abstract 修饰符可修饰字段、方法和类

B． 抽象方法的 body 部分必须用一对大括号{ }包住

C． 声明抽象方法，大括号可有可无

D． 声明抽象方法不可写出大括号

8．下列语句正确的是（ ）

A． 形式参数可被视为 local variable

B． 形式参数可被字段修饰符修饰

C． 形式参数为方法被调用时，真正被传递的参数

D． 形式参数不可以是对象

9．下列哪种说法是正确的（ ）

A． 实例方法可直接调用超类的实例方法

B. 实例方法可直接调用超类的类方法
C. 实例方法可直接调用其他类的实例方法
D. 实例方法可直接调用本类的类方法

二、 多项选择题

1. Java 程序的种类有（ ）
A. 类（Class） B. Applet C. Application D. Servlet

2. 下列说法正确的有（ ）
A. 环境变量可在编译 source code 时指定
B. 在编译程序时，所能指定的环境变量不包括 class path
C. javac 一次可同时编译数个 Java 源文件
D. javac.exe 能指定编译结果要置于哪个目录（directory）

3. 下列标识符不合法的有（ ）
A. new B. $Usdollars C. 1234 D. car.taxi

4. 下列说法错误的有（ ）
A. 数组是一种对象
B. 数组属于一种原生类
C. int number=[]={31,23,33,43,35,63}
D. 数组的大小可以任意改变

5. 不能用来修饰 interface 的有（ ）
A. private B. public C. protected D. static

6. 下列正确的有（ ）
A. call by value 不会改变实际参数的数值
B. call by reference 能改变实际参数的参考地址
C. call by reference 不能改变实际参数的参考地址
D. call by reference 能改变实际参数的内容

7. 下列说法错误的有（ ）
A. 在类方法中可用 this 来调用本类的类方法
B. 在类方法中调用本类的类方法时可直接调用
C. 在类方法中只能调用本类中的类方法
D. 在类方法中绝对不能调用实例方法

8. 下列说法错误的有（ ）
A. Java 面向对象语言容许单独的过程与函数存在
B. Java 面向对象语言容许单独的方法存在
C. Java 语言中的方法属于类中的成员（member）
D. Java 语言中的方法必定隶属于某一类（对象），调用方法与过程或函数相同

9. 下列说法错误的有（ ）
A. 能被 java.exe 成功运行的 java class 文件必须有 main()方法

B. J2SDK 就是 Java API
C. Appletviewer.exe 可利用 jar 选项运行.jar 文件
D. 能被 Appletviewer 成功运行的 java class 文件必须有 main()方法

10. 给定如下 JAVA 程序片断:

```
class A{
    public A(){
        system.out.println(“A”);
    }
}
class B extends A{
     public B(){
         System.out.println(“B”);
     }
     public static void main(String[] args){
         B b=new B();
     }
} 上述程序将( ).(选择一项)
```

A、 不能通过编译
B、通过编译,输出为: A B
C、通过编译,输出为: B
D、通过编译,输出为: A

11. 某一 java 程序中有如下代码:

```
DataInputstream din=new DataInputstream (
                  new BufferedInputstream
                    (new FileInputstream(“employee.dat”) ));
```

假设在 employee.dat 文件中只有如下一段字符:abcdefg。则:System.out.println(din)在屏幕上打印 ().(选择一项)

A、A
B、B
C、97
D、98

12. 给定 java 代码片段，如下:

```
int i=0,j=-1;
switch(i){
   case 0,1:j=1;
   case 2:j=2;
}
System.out.print("j="+j);
```

编译运行，正确的是 ()。(选择一项)

A、程序编译出错
B、j=1
C、j=2
D、j=0

13、在 Java 中，下面关于构造函数的描述正确的是（）。（选择一项）
A、类必须有显式构造函数
B、它的返回类型是 void
C、它和类有相同的名称，但它不能带任何参数
D、以上皆非

14、 在 JAVA 编程中，Swing 包中的组件处理事件时，下面（）是正确的。（选择一项）
A、 Swing 包中的组件也是采用事件的授权得理模型来处理事件的
B、 Swing 包中组件产生的事件类型，也都带有一个 J 字母，如：JmouseEvent
C、Swing 包中的组件也可以采用事件的传递处理机制
D、 Swing 包的组件所对应的事件适配器也是带有 J 字母的，如：JmouseAdapter

15、
```
public class EqTest {
        public static void main(String args[]){
             EqTest e=new EqTest();
        }
        EqTest(){
             String s="Java"; String s2="java";
             //在这儿放置测试代码
             {
                System.out.println("相等");
             }else {
                System.out.println("不相等");
             }
        }
```
} 在上面的 Java 代码的注释行位置，放置（）测试代码能输出“相等”结果。（选择一项）
A、if(s==s2)
B、if(s.equals(s2))
C、if(s.equalsIgnoreCase(s2))
D、if(s.noCaseMatch(s2))

16、 在类的说明符中，被指定为私有的数据可以被以下（）访问。（选择一项）
A、程序中的任何函数
B、其他类的成员函数
C、类中的成员函数
D、派生类中的成员函数

17、 在 JAVA 编程中，以下（）命令用来执行 java 类文件。（选择一项）
A、javac
B、java
C、appletviewer
D、以上所有选项都不正确

18、Java 中，下面的方法可以正确的创建一个 Swing 标签组件是（）（选择两项）
A、ImageIcon icon = new ImageIcon(

B、ImageIcon icon = new ImageIcon(
C、ImageIcon icon=new ImageIcon(
D、JLabel label=new JLabel(SwingConstants.LEFT);

19、分析下面的用 Java 语言编写的 trythis()方法：

```
public void trythis(){
   try{
      System.out.print("a");
      problem();
   }catch(RuntimeException e){
       System.out.print("b");
   }catch(Exception e){
       System.out.print("c");
   }finally{
       System.out.print("d");
   }
   System.out.print("e");
}
```

当该方法被调用时，如果其中的 problem()方法引发一个 RuntimeException 类的异常，那么输出结果将是（）。（选择一项）

A、abcde
B、abd
C、abde
D、abe

20、

```
public class MyClass1{
    public static void main (String argv[]){
    }
    _____ class MyInner { }
}
```

在以上 java 代码中的横线上，可放置（）修饰符。（选择三项）

A、public
B、private
C、static
D、friend

21、

```
public class test3{
    public static void main(String args[]){
        for(int i=0;i<3;i++){
           for(int j=3; j>0;j--){
              if(i==j)
                  continue;
                  System.out.println(“i=”+i+”j=”+j)
           }
        }
    }
```

上面的 JAVA 代码编译运行后，下列选项中，（）会出现在输出结果中，（选择三项）

A、i=0 j=3

B、i=0 j=0
C、i=2 j=2
D、i=0 j=2
E、i=1 j=2

22、如下 Java 代码段，体现了（）概念。(选择一项)

```
.........
public void aMethod(String s){ .........}
public void aMethod(int i){.........}
public void aMethod(int I,float f){ .........}
.........
```

A、多继承
B、重载
C、重写
D、多态

23、java 语言中，下列时处理输出操作的所有类的基础的是（）(选择一个)
A、DataOutput
B、OutputStream
C、BufferedOutputStream
D、IOStream

24、在 jave 中，类 Worker 是类 Person 的子类，Worker 的构造方法中有一句”super()”,该语句 （)。(选择一项)
A、调用类 Worker 中定义的 super()方法
B、调用类 Person 中定义的 super()方法
C、调用类 Person 的构造函数
D、句法错误

25、在 Java 中，关于 final 关键字的说法正确的是（)。(选择两项)
A、如果修饰变量，则一旦赋了值，就等同一个常量
B、如果修饰类，则该类只能被一个子类继承
C、如果修饰方法，则该方法不能在子类中被覆盖
D、如果修饰方法，则该方法所在的类不能被继承

26、在 Java 中,下列选项表示字符”a”值的是().(选择一项)
A、'a'
B、"a"
C、new Character(a)
D、\000a

27、给定 java 代码片段，如下：

```
Integer a = new Integer(3);
Integer b = new Integer(3);
System.out.println(a==b); 运行后，这段代码将输出（)。(选择一项)
```

A、true

B、false
C、0
D、1

28、Java 程序中读入用户输入的一个值，要求创建一个自定义的异常，如果输入值大于10，使用 throw 语句显式地引发异常，异常输出信息为“something’s wrong!”，语句为 ()。（选择一项）
A、if (I>10) throw Exception(“something’s wrong!”);
B、if (I>10) throw Exception e (“something’s wrong!”);
C、if (I>10) throw new Exception(“something’s wrong!”);
D、if (I>10) throw new Exception e (“something’s wrong!”);

29、42.给定两个 java 程序,如下:Text.java 的编译运行结果是(). (选择一项)

```
pubilc interface Face{ int count=40; }
pubilc class Text implements Face{
    private static int counter;
    pubilc static void main(String[]args){
         System.out.println(++counter);
    }
}
```

A、40
B、41
C、0
D、1

30、给定一个 Java 程序代码，如下：运行编译后，输出结果是(). (选择一项)

```
pubilc class Test{
    int count = 9;
    pubilc void count1(){
        int count = 10;
        System.out.println("count1"+count);
    }
    pubilc void count2(){
        System.out.println("count2"+count);
    }
    pubilc static void main(String args[]){
        Test t =new Twst();
        t.count1();
        t.count2();
    }
}
```

A、count1=9 count2=9
B、count1=10 count2=9
C、count1=10 count2=10
D、count1=9 count2=10

31、给定 java 代码如下：要使用这段代码能够编译成功，横线处可以填入().(选择两项)

```
 pubilc int count(char c,int i,double d){   return_____;   }
```

A、c*i
B、c*(int)d
C、(int)c*d
D、i*d

32、 在 JAVA 编程中，Java 编译器会将 Java 程序转换为（）。（选择一项）
A、字节码
B、可执行代码
C、机器代码
D、以上所有选项都不正确

33、在 J2EE 中，下列元素经过排序的集合类是（）。（选择一项）
A、LinkedList
B、Stack
C、Hashtable
D、TreeSet

34、分析下面的 Java 程序：

```
public class yy {
    public static void main(String[] ards) throws Exception {
       try {
           throw new Exception();
       }catch(Exception e){
           System.out.println("Caught in main()");
       }
       System.out.println("nothing");
    }
```

} 输出结果为（）。（选择一项）
A、Caught in main() nothing
B、Caught in main()
C、nothing
D、没有任何输出

35、编译并运行下面的 Java 代码段：

```
   char c='a'; switch (c) {
       case 'a': System.out.println("a");
       default: System.out.println("default");
   }
```

输出结果是（）。（选择一项）
A、代码无法编译，因为 switch 语句没有一个合法的表达式
B、a default
C、a
D、default

36、在 Java 中，执行下面的语句后，c 的值为（）。（选择一项）

```
String s= "Jessica "; char c=s.charAt(6);
```

A、null www.87717.com
B、' '
C、'c '
D、'a '

37、在 J2EE 中，下面的代码中出现编译错误的是（）。（选择一项）
A、File f = new File("/","autoexec.bat");
B、DataInputStream din = new DataInputStream(
new FileInputStream("autoexec.bat"));
C、InputStreamReader in = new InputStreamReader(System.in);
D、OutputStreamWriter out = new OutputStreamWriter(System.in);

38、在 JavaSwing 编程中，要获取每次选择的 JComboBox 对象的选项值，可以使用()类型的监听器. (选择两项)
A、ActionListener
B、ltemListener
C、KeyListener
D、SelectionListener

39、在 Java 中，下面关于包的陈述中正确的是（）。（选择两项）
A、包的声明必须是源文件的第一句代码
B、包的声明必须紧跟在 import 语句的后面
C、只有公共类才能放在包中
D、可以将多个源文件中的类放在同一个包中

40、在 Java 中，要想使只有定义该类所在的包内的类可以访问该类，应该用（ ）关键字。（选择一项）
A、不需要任何关键字
B、private
C、final
D、protected

41、包 pack1 的类 class1 中有成员方法：
protected void method_1(){…}
private void method_2(){…}
public void method_3(){…}
void method_4(){…}，在包 pack2 中的类 class2 是 class1 的子类，你在 class2 中可以调用方法（）。（选择两项）
A、method_1
B、method_2
C、method_3
D、method_4

42、在 Java 语言中，小明在他的包 mypackage 中定义了类 My_Class，在 mypackage 的子包 mysubpackage 中也有个类 My_Class。小明用.import mypackage：引入包，执行其中的语句：My_Class NewClass=New My_Class();时，将发生（）。（选择一项）

A、创建一个类 mypackage.My_Class 对象
B、创建一个类 mypackage. Mysubpackage.My_Class 的对象
C、该语句是错误的
D、创建一个类 mypackage.My_Class 的对象和一个类 mypackage.
Mysubpackage.My_Class 的对象

43、在 JavaSwing 编程中，给定一个 java 程序 main 方法的代码片段如下:

```
JFrame jf=new JFrame();
jf.getContentpane().setLayout(null);
jf.setSize(200,200);
jf.setVisible(true); //a
```

要在界面上显示如下组件，则应在 A 处填入(). (选择一项)

A、JTextArea text = text JTextArea(100,100);
text.setBounds(10,10,150,100); jf.getContentpane().add(text);

B、JTextField text = text JTextField(100,100); JScrollpane text=new JScrollpane(text);
jf.setBounds(10,10,150,100);jf.getContentpane().add(jp);

C、JTextArea text = new JTextArea(100,1); JScrollpane jp=new JScrollpane(text);
jp.setBounds(10,10,150,100); jf.getContentpane().add(jp);

D、JTextArea text = new JTextArea(100,100);JScrollpane
jp=new JScrollpane(text); jp.setBounds(10,10,150,100);
jf.getContentpane().add(jp);

44、在 JAVA 语言中，你的按钮要实现下面的功能：当鼠标按下按钮时，显示“鼠标已按下”；当释放按键时，显示“鼠标已释放”。你必须具体定义接口 MouseListener 的（）方法。（选择两项）

A、mouseClicked
B、mouseEntered
C、mouseExited
D、mousePressed
E、mouseReleaseed

45、JAVA 中，为了辨别用户关闭窗口的时间，要实现监听器接口（）。(选择一项)

A、MouseListener
B、ActionListener
C、WindowListener
D、以上都要

46、在 Java 语言中，当一个类的某个变量声明为 protected 时下列说法正确的是（）。(选择两项)

A、只有同一类中的成员才能访问它
B、不同包中的任何其他类都能够访问它
C、同包中的任何其他类能够访问它
D、不同包中的子类可以访问该变量

47、在 Java 事件处理模型中，当按下鼠标按钮时，处理（）事件。(选择一项)
A、mouseReleased
B、mouseExited
C、mousePressed
D、mouseDown

48、String s1=new String(“Hello”); String s2=new String(“there”); String s3=new String("0"); 上面是 JAVA 程序的一些声明，以下选项中能够通过编译的是()。(选择一项)
A、 s3=s1+s2
B、 s3=s1&s2
C、 s3=s1||s2
D、 s3=s1&&s2

49、在 Java 中，调用 Math.random() 方法可能返回的结果是（）。(选择一项)
A、132.34
B、0.342
C、29.34E10
D、1.0009

50、 在 Java 语言中，Panel 默认的布局管理器是（）。(选择一项)
A、BorderLayout
B、FlowLayout
C、GridLayout
D、GridBagLayout

51、public class MyClass1 {
public static void main(String argv[]){ }
_____class MyInner{} } 在以上 Java 代码的横线上，可放置（）修饰符。(选择两项)
A、public
B、private
C、implements
D、friend

52、在 java 中，下列赋值语句正确的是（）。(选择二项)
A、char c='a';
B、char c="a";
C、char c=97;
D、char c=new Character('a');

53、与传统的过程编程不同，面向对象方法的主要思想是（）。(选择两项)
A、真实反映用户的实际需求
B、将现实世界的一切抽象为实体或对象
C、将现实世界细分为一个过程化实现
D、将软件组织成为对象的集合，将数据结构和行为结合在一起

54、在 Java 中，根据你的理解，下列方法（）可能是类 Orange 的构造方法。(选择 3 项)
A、Orange（）{…}
B、Orange（…）{…}
C、Public void Orange(){…}
D、Public Orange(){…}
E、Public Orange Constuctor(){…}

55、在 Java 中,()借口位于集合框架的顶层.　　(选择一项)
A、Map
B、Collection
C、Set
D、List

56、给定某 java 程序片段，如下： int i=1; int j=i++; if((i>++j)&&(i++==j)) i+=j; System.out.println(i); 该程序运行后，i 的输出结果为（）。(选择一项)
A、1
B、2
C、3
D、4

57、在 Java 中，假设我们有一个实现 ActionListener 接口的类，以下方法中（）能够为一个 Button 类注册这个类。(选择一项)
A、addListener()
B、addActionListener()
C、addButtonListener()
D、setListener()

58、44.在 JAVA 中的布局管理器，以下说法中错误的是(). (选择一项)
A、FlowLayout 以由上到下的方式从左到右排列组件
B、BorderLayout 使用 "东"."西"."南"."北","居中"来指定组件的位置
C、GridLayout 可以创建网格布局，网格布局中各组的大小可以任意调整
D、可以通过容器的 setLayout 方法为容器指定布局管理

59、研究下面的 Java 代码：

```
public class testException{
     public static void main(String args[]){
        int n[]={0,1,2,3,4};
        int sum=0;
        try {
            for(int i=1;i<6;i++)
                sum=sum+n[i];
            System.out.println("sum="+sum);
         } catch(ArrayIndexOutOfBoundsExpception e) {
             System.out.println("数组越界");
         } finally{
              System.out.println("程序结束");
```

```
            }
        }
    }
```

} 输出结果将是（）。(选择一项)

A、10 数组越界 程序结束

B、10 程序结束

C、数组越界 程序结束

D、程序结束

三、 判断题

1．Java 程序中的起始类名称必须与存放该类的文件名相同。( )

2．Unicode 是用 16 位来表示一个字的。( )

3．原生类中的数据类型均可任意转换。( )

四、问答题

1、在 java 中如果声明一个类为 final，表示什么意思？

2、父类的构造方法是否可以被子类覆盖（重写)?

3、请讲述 String 和 StringBuffer 的区别。

4、如果有两个类 A、B（注意不是接口），你想同时使用这两个类的功能，那么你会如何编写这个 C 类呢？

5、结合 Java 多线程分析 sleep()和 wait()方法的区别。

6、谈谈你对抽象类和接口的理解。

部分参考答案：
1、单选题
答案：B 答案：B 答案：C 答案：A 答案：A 答案：C
答案：D 答案：A 答案：D [1-9 题]
2、多选题
答案：BC 答案：BCD 答案：ACD 答案：BCD 答案：ACD
答案：ACD 答案：ACD 答案：ABC 答案：BCD
3、判断题
正确
正确
错误

姓名:_____ 电话:_____ 应职岗位:_____ 结果:_____

北京联动天*面试题（四）

一、选择题

1. 下列语句哪一个正确（ ）
A. Java 程序经编译后会产生 machine code
B. Java 程序经编译后会产生 byte code 也就是.class 文件
C. Java 程序经编译后会产生 DLL
D. 以上都不正确

2. 下列说法正确的有（ ）
A. class 中的 constructor 不可省略
B. constructor 必须与 class 同名，但方法不能与 class 同名
C. constructor 在一个对象被 new 时执行
D. 一个 class 只能定义一个 constructor

3. 提供 Java 存取数据库能力的包是（ ）
A. java.sql
B. java.awt
C. java.lang
D. java.swing

4. 下列运算符合法的是（ ）
A. &&
B. <>
C. if
D. :=

5. 执行如下程序代码

```
a=0;c=0;
  do{
     --c;
     a=a-1;
        }while(a>0);
```

后，C 的值是（ ）

A. 0 B. 1 C. -1 D. 死循环

6. 下列哪一种叙述是正确的（ ）

A. abstract 修饰符可修饰字段、方法和类

B. 抽象方法的 body 部分必须用一对大括号{ }包住

C. 声明抽象方法，大括号可有可无

D. 声明抽象方法不可写出大括号

7. 下列哪种说法是正确的（ ）

A. 实例方法可直接调用超类的实例方法

B. 实例方法可直接调用超类的类方法

C. 实例方法可直接调用其他类的实例方法

D. 实例方法可直接调用本类的类方法

8. 下列说法错误的有（BCD ）多选

A. 数组是一种对象

B. 数组属于一种原生类

C. int number=[]={31,23,33,43,35,63}

D. 数组的大小可以任意改变

9. 不能用来修饰 interface 的有（ACD ）多选

A. private B. public C. protected D. static

10. 下列说法错误的有（ ）多选

A. 在类方法中可用 this 来调用本类的类方法

B. 在类方法中调用本类的类方法时可直接调用

C. 在类方法中只能调用本类中的类方法

D. 在类方法中绝对不能调用实例方法

11. 下列说法错误的有（ ）多选

A. Java 面向对象语言容许单独的过程与函数存在

B. Java 面向对象语言容许单独的方法存在

C. Java 语言中的方法属于类中的成员（member）

D. Java 语言中的方法必定隶属于某一类（对象），调用方法与过程或函数相同

12、（）是用一组任意的存储单元存储线性表元素的一种数据结构

A、数组 B、链表 C、树 D、图

13、程序的局部变量存在于_____中，全局变量存在于_____中，动态申请数据存在于_____中
A、代码段　B、数据段　C、堆空间　D、程序

1-13 参考答案
B、CAACDD、BCD、ACD、　ACD、ABC、B、ACB

14、下面哪种不是 jquery 的选择器？(单选)
A、基本选择器 B、后代选择器 C、类选择器 D、进一步选择器
考点:jquery 的选择器 (C)

15、当 DOM 加载完成后要执行的函数，下面哪个是正确的？(单选)
jQuery(expression, [context]) B 、 jQuery(html, [ownerDocument]) C 、 jQuery(callback) D 、 jQuery(elements)
考点：jquery 的核心函数 (C)

16、下面哪一个是用来追加到指定元素的末尾的？(单选)

A、insertAfter() B、append() C、appendTo() D、after()
考点：jquery 的核心函数 (C)

17、下面哪一个不是 jquery 对象访问的方法？(单选)

A、each(callback) B、size() C、index(subject) D、index()
考点： jquery 的核心函数之对象访问 （D）

18、jquery 访问对象中的 size()方法的返回值和 jQuery 对象的_____属性一样.

考点：jquery 的核心函数之对象访问 （length）

19、jquery 中$(this).get(0)的写法和_____是等价的。

考点：jquery 的核心函数之对象访问 （$(this)[0]）

20、有这样一个表单元素
，想要找到这个 hidden 元素，下面哪个是正确的？(单选)

A、visible B、hidden C、visible() D、hidden()
考点：jquery 的选择器 （B）

21、如果需要匹配包含文本的元素，用下面哪种来实现？(单选)

A、text() B、contains() C、input() D、attr(name)
考点：jquery 的选择器 （B）

22、现有一个表格，如果想要匹配所有行数为偶数的，用_____实现，奇数的用_____实现。

考点：jquery 的选择器 （even,odd）

23、如果想要找到一个表格的指定行数的元素，用下面哪个方法可以快速找到指定元素？(单选)

A、text() B、get() C、eq() D、contents（）
考点：jquery 的选择器 （C）

24、在一个表单里，想要找到指定元素的第一个元素用_____实现，那么第二个元素用_____实现。

考点：jquery 的选择器 （first,eq(1)）

25、下面哪种不属于 jquery 的筛选？（单选）

A、过滤 B、自动 C、查找 D、串联
考点：jquery 的筛选 (B)

26、下面哪几种是属于 jquery 文档处理的？ （多选）

A、包裹 B、替换 C、删除 D、内部和外部插入
考点：jquery 的文档处理 (ABD)

27、如果想在一个指定的元素后添加内容，下面哪个是实现该功能的？（单选）

A、append(content) B、appendTo(content) C、insertAfter(content) D、after(content)
考点：jquery 的文档处理 (D)

28、在 jquery 中，用一个表达式来检查当前选择的元素集合，使用_____来实现，如果这个表达式失效，则返回_____值。

考点：jquery 的筛选 （is(expr),false）

29、在 jquey 中，如果想要从 DOM 中删除所有匹配的元素，下面哪一个是正确的？（单选）

A、delete() B、empty() C、remove() D、removeAll()

考点：jquery 的文档处理 （C）

30、在 jquery 中，想要给第一个指定的元素添加样式，下面哪一个是正确的？（单选）

A、first B、eq(1) C、css(name) D、css(name,value)
考点： jquery 的 css 处理 （C）

31、在编写页面的时候，如果想要获取指定元素在当前窗口的相对偏移，用_____来实现，该方法的返回值有两个属性，分别是_____和_____。

考点：jquery 的 css 处理 （offset,top,left）

32、在 jquery 中，如果想要获取当前窗口的宽度值，下面哪个是实现该功能的？ （单选）

A、width() B、width(val) C、width D、innerWidth()
考点：jquery 的 css 处理 （A）

33、在一个表单中，如果将所有的 div 元素都设置为绿色，实现功能是_____。

考点：jquery 的 css 处理 ($(“div”).css(“color”,”green”))

34、为每一个指定元素的指定事件（像 click）绑定一个事件处理器函数，下面哪个是用来实现该功能的？（单选）

A、trigger (type) B、bind(type) C、one(type) D、bind
考点：jquery 的事件操作 (B)

35、在 jquery 中，鼠标移动到一个指定的元素上，会触发指定的一个方法，实现该操作的是_____。

考点：jquery 的事件操作 (hover(over,out))

36、下面哪几个不是属于 jquery 的事件处理？(多选)

A、bind(type) B、click() C、change() D、one(type)
考点：jquery 的事件处理 (BC)

37、在一个表单中，如果想要给输入框添加一个输入验证，可以用下面的哪个事件实现？ (单选)

A、hover(over ,out) B、keypress（fn） C、change() D、change(fn)
考点：jquery 的事件处理 (D)

38、当一个文本框中的内容被选中时，想要执行指定的方法时，可以使用下面哪个事件来实现？（单选）
A、click(fn) B、change(fn) C、select(fn) D、bind(fn)
考点：jquery 的事件处理 (C)

39、在 jquery 中，想让一个元素隐藏，用_____实现，显示隐藏的元素用_____实现。

考点：jquery 的页面效果 （hide(),show()）

40、在一个表单中，用600毫秒缓慢的将段落滑上，用_____来实现。

考点：jquery 的页面效果 （$(“p”).slideUp(“slow”)）

在 jquery 中，如果想要自定义一个动画，用_____函数来实现。
考点：jquery 的页面效果 (animate(params, options))

二、css 测试题

css 的语法由_____、_____、_____三部分组成。
考点：css 的语法 (选择器、属性和值)

下面哪一个不是检索一个对象的定位方式的属性？ （单选）
A、static B、absolute C、fixed D、top
考点：css 的定位 （D）

下面哪个是 display 布局中用来设置对象以块显示，并添加新行的？ （单选）
A、inline B、none C、block D、compact
考点：css 的布局 (C)

设置边框的边框样式用_____实现，设置边框的颜色用_____实现。
考点：css 的边框 （border-style,border-color）

设置一个 div 的最小宽度为100像素，最大高度为50像素，实现这两个设置的方法是_____、_____。
考点：css 的尺寸 (min-width：100px; min-height：50px;)

为一个图片设置成背景图像后，设置背景图像在纵向和横向上平铺，下面哪个是实现该功能的？ (单选)
A、no-repeat B、repeat C、repeat-x D、repeat-y
考点：css 的背景 background (repeat)

想要设置一个对象外的线条轮廓的样式为虚线边框，用_____实现该功能。
考点：css 的轮廓 outline (outline-style: dashed)

在 css 中，如果想要阻止计数器增加的，用_____来实现。
考点：css 的内容 content (counter-increment : none)

在编写 css 样式的时候，如果想要设置标记容器和主容器之间的补白，用_____来表示。
考点：css 的列表 (marker-offset)

如果想设置表格的行与单元格的边框合并在一起，可以用_____来实现。
考点：css 的表格 （border-collapse：collapse）

如果想设置一个对象内文本的流动和方向，用_____标签来实现，该标签有两个值，分别是_____、_____。
考点：css 的文本 Text （text-layout-flow，horizontal、vertical-ideographic）

在编写 css 样式的时候，如果想要设置文本的对齐方式，下面哪一个不是 text-align 下的值？（单选）
A、left B、right C、top D、justify
考点：css 的文本 Text (C)

在 css 中，如果想要设置对象中文本的样式，下面哪一个不是该标签的值？(单选)
A、blink B、line-through C、color D、overline
考点：css 的字体 font （C）

在 css 中，设置滚动条的表面颜色，是用下面的哪个标签来实现的？（单选）
A、scrollbar-base-color B、scrollbar-face-color C、scrollbar-arrow-color
D、scrollbar-shadow-color
考点：css 的滚动条 scrollbar（B）

二、简答题

1、abstract class 和 interface 有什么区别?

2、error 和 exception 有什么区别?

3、接口是否可继承接口?抽象类是否可实现(implements)接口?抽象类是否可继承实体类(concrete class)?

4、HashMap 和 Hashtable 的区别。

5、谈谈 final, finally, finalize 的区别。

6、当一个线程进入一个对象的一个 synchronized 方法后，其它线程是否可进入此对象的其它方法?

7、try {}里有一个 return 语句，那么紧跟在这个 try 后的 finally {}里的 code 会不会被执行，什么时

8、String 和 StringBuffer 的区别。

9、说出 ArrayList,Vector, LinkedList 的存储性能和特性。

11、说出数据连接池的工作机制是什么?。

三、找错题

1.abstract class Name
{
    private String name;
    public abstract boolean isStupidName(String name) {}
}

abstract method 必须以分号结尾，且不带花括号

2.public class Something
{
    void doSomething ()
    {
        private String s = "";
        int l = s.length();
    }
}

局部变量前不能放置任何访问修饰符 (private，public，和 protected)。final 可以用来修饰局部变量 (final 如同 abstract 和 strictfp,都是非访问修饰符,strictfp 只能修饰 class 和 method 而非 variable)。

3.abstract class Something
{
    private abstract String doSomething ();
}

abstract 的 methods 不能以 private 修饰。abstract 的 methods 就是让子类 implement(实现)具体细节的，怎么可以用 private 把 abstract method 封锁起来呢? (同理，abstract method 前不能加 final)。

4.public class Something
{
    public int addOne(final int x)
    {
        return ++x;

```
    }
}
```

int x 被修饰成 final，意味着 x 不能在 addOne method 中被修改。

四、数据库

表名：t_student

ID	NAME	SEX	AGE	REMARK
1	张三	男	23	
2	李四	男	24	
3	王五	女	25	

1、 根据表 t_student，分别写出 mysql、sqlserver、oracle 分页语句（每页显示 15 条记录，检索第 2 页）

2、 求出表中全体人员平均年龄

3、 检索表中全部男生，并且按照年龄由大到小排序