
JAVA 综合面试题

2010

目录

Java 面试题整理	9
Java 面向对象	9
1. super()与 this()的区别?	9
2. 作用域 public,protected,private,以及不写时的区别?	9
3. 编程输出如下图形。	9
4. JAVA 的事件委托机制和垃圾回收机制.....	10
5. 在 JAVA 中, 如何跳出当前的多重嵌套循环?	10
6. 什么是 java 序列化, 如何实现 java 序列化? (写一个实例).....	10
7. 一个".java"源文件中是否可以包括多个类(不是内部类)? 有什么限制?	10
8. 排序都有哪几种方法? 请列举。用 JAVA 实现一个快速排序?	10
9. Overload 和 Override 的区别。Overloaded 的方法是否可以改变返回值的类型?....	11
10. Final 类有什么特点?	11
11. 继承时候类的执行顺序问题,一般都是选择题,问你将会打印出什么?	11
12. 内部类的实现方式?.....	12
13. 用 JAVA 实现一种排序, JAVA 类实现序列化的方法(二种)?	13
14. 如在 COLLECTION 框架中, 实现比较要实现什么样的接口?	13
15. 用插入法进行排序代码如下.....	13
16. 编程: 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。 但是要保证汉字不被截半个, 如"我 ABC"4, 应该截为"我 AB", 输入"我 ABC 汉 DEF", 6, 应该输出为"我 ABC"而不是"我 ABC+汉的半个"。	14
15、Java 编程,打印昨天的当前时刻.....	15
16、文件读写,实现一个计数器.....	15
17、指出下面程序的运行结果。	16
18、抽象类和接口的区别?	16
19、什么是类的反射机制?.....	16
20、类的反射机制中的包及核心类?.....	17
21、得到 Class 的三个过程是什么?.....	17
22、如何唤起类中的一个方法?	17
23、如何将数值型字符转换为数字(Integer, Double)?	17
24、如何将数字转换为字符?	17
25、如何去小数点前两位, 并四舍五入。	17
26、如何取得年月日, 小时分秒?	18
27、如何取得从 1970 年到现在的毫秒数.....	18
28、如何获取某个日期是当月的最后一天?	18
29、如何格式化日期?	19
30、编码转换, 怎样实现将 GB2312 编码的字符串转换为 ISO-8859-1 编码的字符串。	19
32、String s = new String("xyz");创建了几个 String Object?.....	19
33、float 型 float f=3.4 是否正确?.....	19
35、说出一些常用的类, 包, 接口, 请各举 5 个.....	19
36、java 中会存在内存泄漏吗, 请简单描述。	20
37、java 中实现多态的机制是什么?	20
38、垃圾回收器的基本原理是什么? 垃圾回收器可以马上回收内存吗? 有什么办法主动	

通知虚拟机进行垃圾回收?	20
39、静态变量和实例变量的区别?	20
41、是否可以从一个 static 方法内部发出对非 static 方法的调用?	20
42、写 clone()方法时, 通常都有一行代码, 是什么?	20
43、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?	21
45、冒泡排序法.....	21
46、String and StringBuffer 的区别?	22
47、用 java 代码编写堆栈.....	22
48、集合的作用是什么?.....	23
49、集合的通用方法有那些?通用方法是什么?(操作).....	23
50、说出 ArrayList,Vector, LinkedList 的存储性能和特性 HashMap 和 Hashtable 的区别	24
51、Collection 和 Collections 的区别。.....	24
52、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?用 contains 来区分是否有重复的对象。还是都不用。	24
53、List, Set, Map 是否继承自 Collection 接口?.....	24
54、面向对象的特征有哪些方面.....	25
55、String 是最基本的数据类型吗?.....	25
56、int 和 Integer 有什么区别?	25
57、运行时异常与一般异常有何异同?	25
58、&和&&的区别?	25
59、final, finally, finalize 的区别?	25
62、heap 和 stack 有什么区别?	26
63、Static Nested Class 和 Inner Class 的不同?	26
64、什么时候用 assert?	26
65、GC 是什么? 为什么要有 GC?.....	26
66、short s1 = 1; s1 = s1 + 1;有什么错? short s1 = 1; s1 += 1;有什么错?.....	26
67、Math.round(11.5)等於多少? Math.round(-11.5)等於多少?.....	27
68、Java 有没有 goto?.....	27
69、给我一个你最常见到的 runtime exception.....	27
70、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)?.....	27
71、abstract 的 method 是否可同时是 static,是否可同时是 native, 是否可同时是 synchronized?	27
72、数组有没有 length()这个方法? String 有没有 length()这个方法?	28
73、构造器 Constructor 是否可被 override?.....	28
74、是否可以继承 String 类?	28
75、swtich 是否能作用在 byte 上, 是否能作用在 long 上, 是否能作用在 String 上?..	28
76、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?.....	28
77、编程题: 用最有效率的方法算出 2 乘以 8 等於几?.....	28
78、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?	28

79、当一个对象被当作参数传递到一个方法后,此方法可改变这个对象的属性,并可返回变化后的结果,那么这里到底是值传递还是引用传递?	29
80、四种会话跟踪技术.....	29
81、编程题:写一个 Singleton 出来。	29
83、Java 中的异常处理机制的简单原理和应用。	30
84、垃圾回收的优点和原理。并考虑 2 种回收机制。	30
85、描述一下 JVM 加载 class 文件的原理机制?	30
86、char 型变量中能不能存贮一个中文汉字?为什么?.....	30
88、写一个程序,从文件(c:\test.txt)中查出字符串"mobnet"出现的次数?	30
java 基础类库(io 流,集合类,线程,Socket,AWT,Swing,sql).....	31
1、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?	31
2、启动一个线程是用 run()还是 start()?.....	31
3、线程的基本概念、线程的基本状态以及状态之间的关系	31
4、多线程有几种实现方法,都是什么?同步有几种实现方法,都是什么?用什么关键字修饰同步方法? stop()和 suspend()方法为何不推荐使用?	31
用 synchoronized 修饰同步方法。	32
5、集合框架有什么?.....	32
12、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.....	32
13、同步和异步有和异同,在什么情况下分别使用他们?举例说明。	34
14、sleep() 和 wait() 有什么区别?.....	34
15、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?.....	34
17、输入输出流的理解:.....	34
18、请写一个程序的读写,要求用两种方式一种是低层流另一种是高层流。	35
19、如何列出某个目录下的所有文件.....	36
Socket.....	37
20、用 socket 通讯写出客户端和服务端端的通讯,要求客户发送数据后能够回显相同的数据?	37
23、介绍 JAVA 中的 Collection FrameWork(包括如何写自己的数据结构)?.....	39
24、请说出你所知道的线程同步的方法.....	39
jdbc 数据访问技术.....	39
1、JDBC 如何做事务处理?	39
2、写出几个在 Jdbc 中常用的接口	39
3、简述你对 Statement,PreparedStatement,CallableStatement 的理解.....	39
4、Java 中访问数据库的步骤?	40
5、JDBC 中的核心类及其作用是什么?.....	40
6、执行存储过程用那一个类,如何操作输出参数?(操作)	41
8、可能会让你写一段 Jdbc 连 Oracle 的程序.....	41
9、Class.forName 的作用?为什么要用?	41
10、Jdo 是什么?.....	41
11、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	41

Web 编程 Jsp&Servlet 技术.....	42
1、简单说说 tomcat 的配置?	42
2、JSP 中动态 INCLUDE 与静态 INCLUDE 的区别?	43
3、forward 和 redirect 的区别?	43
4、Servlet 的体系结构是什么?.....	43
Servlet.....	43
5、如何实现一个自定义的 servlet?	43
6、Servlet 的生命周期是什么?.....	43
7、jsp 就是一个 servlet 是否正确?.....	43
8、请罗列 jsp 中的脚本、指令及动作?	43
9、JSP 的内置对象及方法.....	44
10、说出在 JSP 页面里是怎么分页的?.....	44
11、include 的两种实现方式的区别?.....	47
12、jsp 页面中两种跳转方式分别是什么?有什么区别?.....	47
13、描述 JSP 和 Servlet 的区别、共同点、各自应用的范围.....	47
14、在 JSP 中如何读取客户端的请求, 如何确定某个 Jsp 文件的真实路径?	47
15、描述 Cookie 和 Session 的作用, 区别和各自的应用范围, Session 工作原理。	47
16、说明 Jsp 中 errorPage 的作用, 应用范围。	48
17、介绍在 Jsp 中如何使用 JavaBeans.....	48
19、简单介绍 JSP 的标记库.....	48
20、Servlet 中的核心类有那些, 各有什么特点?.....	48
21、Servlet 中重要的包有那些, 有什么区别?.....	49
22、说出 Servlet 的生命周期, 并说出 Servlet 和 CGI 的区别?	49
23、什么情况下调用 doGet()和 doPost()?	49
25、如何现实 servlet 的单线程模式	49
27、Request 对象的主要方法:	49
28、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?	50
30、Servlet 执行时一般实现哪几个方法?.....	50
5 Hibernate 持久层技术.....	51
1、在 myeclipse 加入 hibernate 环境的全过程是什么?.....	51
2、hibernate 的核心配置文件是什么及其作用?.....	51
3、hibernate 的核心类是什么, 它们的相互关系是什么?重要的方法是什么?.....	51
4、关联:	51
5、hibernate 中的 one-to-many 或 many-to-one 中常用的方式是什么?.....	52
6、Criteria 的作用?.....	52
7、DetachedCriteria 的作用?	52
8、Query.....	52
9、继承关系的实现.....	53
10、tomcat 连接池:在容器中预先产生了 n 个连接实例, 客户端不用重新实例化, 可以直接取。	54
11、对象的三大状态.....	55
12、hibernate 常见优化策略.....	56
6. iBatis 持久层技术	56

用 ibatis 的原因:	56
jdbc、hibernate、ibatis 的区别	56
ibatis 的核心配置文件:.....	57
ibatis 的核心类:.....	57
7 Struts 界面控制层技术.....	58
1、请说出 struts 框架的几大组件?	58
3、struts 的核心类有那些, 在 MVC 模式中其对应的关系是什么?	58
4、Struts 的处理请求的全过程是什么?.....	58
5、在 struts 中如何通过一个 url 找到一个 action, 它的核心配置文件是什么?.....	59
6、为什么使用 MVC, 其主要目的是什么?.....	59
7、对于 MVC 在 action 中对应有类有几种, 各有什么作用?.....	59
8、struts 的标记有几类, 请列举并说明其作用?	59
9、如何在 struts 中配置数据源在,什么文件?用什么标签?如何取出 DataSource?.....	60
10、如何在 jbuilder 中开发 struts?.....	60
11、如何实现 struts 的 validator 框架?	60
13、如何实现国际化?.....	62
国际化:不用修改代码, 就适用于不同的语言国家	62
8 JSF 界面控制层技术.....	62
1、Jsf 中的核心类用那些?有什么作用?.....	62
2、Jsf 中的 LiftCycle 六大生命周期是什么?	62
3、如何管理 web 层中的 Bean, 用什么标签。如何通过 jsp 页面与 Bean 绑定在一起进行处理?.....	62
4、Jsf 中导航的标签是什么?.....	63
5、jsf 中用户界面组件模型有几类, 各代表什么?.....	63
6、表格处理及取值.....	63
7、jsf 的标签库有哪些?	63
9 Spring 应用框架技术	64
1、Spring 和 Struts 的区别? struts: 是一种基于 MVC 模式的一个 web 层的处理。	
64	
2、什么是 aop, aop 的作用是什么?	64
3、aop 中的关键名词有些那些, 相互关系是什么?	65
4、依赖注入的方式有几种, 各是什么?.....	65
5、spring 中的核心类有那些, 各有什么作用?.....	65
6、ApplicationContext 的作用	65
7、如何实现资源管理.....	65
8、如何实现加入 web 框架中	65
9、如何实现事件处理.....	66
10、spring 的 ioc 及 di 代表什么意思?.....	66
、如何在 spring 中实现国际化?.....	66
12、spring 的配置的主要标签是什么?有什么作用?	67
13、spring 与 ejb2.0 的事务管理比较的优缺点?.....	67
14、spring 的 jdbc 与传统的 jdbc 有什么区别, 其核心类有那些?.....	67
15、在 spring 中有几种事务管理, 分别是什么?.....	67
16、在 spring 中如何配代码的事务管理?.....	68

17、在 spring 中如何配容器的事务管理, 相关的类有那些?	68
18、如果 spring 与 hibernate 结合在一起可以不需要 hibernate.cfg.xml 文件是否正确?	69
19、spring+hibernate 的配置文件中的主要类有那些?如何配置?	69
20、spring+hibernate 的代码实现中, 对于实现类一定继承于一个类是哪一个, 它有什么作用。	69
21、如何配置 spring+struts?	69
22、如何在 web 环境中配置 applicationContext.xml 文件?	69
24、Jsf 和 spring 的区别?	70
jsf: 是一种基于 MVC 模式的一个 web 层的处理, 粒度较 struts 较细。	70
Ejb 技术	70
1、weblogic 的热发布	70
2、在 ejb 中实现 one-to-many	70
3、ejb 所用的技术:	71
4、实现 ejb 几个接口,几个类?	71
5、实现 ejb 相关的配置文件是什么?	71
6、ejb 的分类?区别	71
7、本地接口与远程接口的区别。	72
8、请求处理的过程?	72
9、statefull 的生命周期	72
10、stateless 的生命周期	72
11、entityBean 的生命周期:	72
12、EJB 需直接实现它的业务接口或 Home 接口吗, 请简述理由。	73
13、EJB 的激活机制	73
14、EJB 是基于哪些技术实现的? 并说出 SessionBean 和 EntityBean 的区别,	73
15、EJB 的分类是什么?各有什么特点?	74
10、EJB 中主要的配置文件及作用是什么?	74
15、说出数据连接池的工作机制是什么?	74
16、EJB2.0 有哪些内容?分别用在什么场合? EJB2.0 和 EJB1.1 的区别?	74
18、EJB 与 JAVA BEAN 的区别?	75
19、EJB 的角色和三个对象	75
20、EJB 容器提供的服务	75
21、EJB 规范规定 EJB 中禁止的操作有哪些?	76
26、EJB 的基本架构	76
30、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	77
31 如何查看在 weblogic 中已经发布的 EJB?	77
WebService 技术	78
1、什么是 Web Service?	78
2、什么是 Web 容器?	78
3、应用服务器有那些?	78
5、如何给 weblogic 指定大小的内存?	78
6、如何设定的 weblogic 的热启动模式(开发模式)与产品发布模式?	78
7、如何启动时不需输入用户名与密码?	79
8、在 weblogic 管理制台中对一个应用域(或者说是一个网站,Domain)进行 jms 及 ejb 或	

连接池等相关信息进行配置后,实际保存在什么文件中?	79
9、说说 weblogic 中一个 Domain 的缺省目录结构?比如要将一个简单的 helloWorld.jsp 放入何目录下,然的在浏览器上就可打入 http://主机:端口号/helloworld.jsp 就可以看到运行结果了? 又比如这其中用到了一个自己写的 javaBean 该如何办?.....	79
12、CORBA 是什么?用途是什么?	79
13、说说在 weblogic 中开发消息 Bean 时的 persistent 与 non-persisten 的差别	79
14、WEB SERVICE 名词解释。JSWDL 开发包的介绍。JAXP、JAXM 的解释。SOAP、UDDI,WSDL 解释。	80
j2ee 模式 (MVC 模式、Model1, Model2)	80
1、j2ee 常用的设计模式? 说明工厂模式。	80
2、说说你所熟悉或听说过的 j2ee 中的几种常用模式?及对设计模式的一些看法.....	81
3、解释下面关于 J2EE 的名词	81
4、介绍 J2EE、J2SE、J2ME 的区别。	81
5、开发中都用到了那些设计模式?用在什么场合?	81
6、J2EE 是什么?	82
7、J2EE 是技术还是平台还是框架?	82
其他.....	82
1、当前主流的解析器有那些?.....	82
2、Dom 解析处理的过程是什么?.....	82
3、Sax 解析处理的过程是什么?.....	84
4、Dom 与 Sax 相比它们的优缺点是什么?.....	85
5、如何将 Dom 对象写入到文件中?.....	85
6、用 jdom 解析 xml 文件时如何解决中文问题?.....	85
7、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?	85
8、标准建模语言 UML 中的各种图?.....	86
9、BS 与 CS 的联系与区别。	86
10、Uml 的概念是什么?主要的工具是什么	87
Uml: 统一建模语言.....	87
11、Uml 的概念中的九大图形是什么?最重的三个图是什么?各有什么特点?.....	87
13、在类图中如何找类?.....	88

Java 面试题整理

Java 面向对象

1. super()与 this()的区别?

This ()：当前类的对象,super 父类对象。

Super ()：在子类访问父类的成员和行为,必须受类继承规则的约束

而 this 他代表当前对象,当然所有的资源都可以访问.

在构造函数中,如果第一行没有写 super(),编译器会自动插入.但是如果父类没有不带参数的构造函数,或这个函数被私有化了(用 private 修饰).此时你必须加入对父类的实例化构造.而 this 就没有这个要求,因为它本身就进行实例化的构造.

而在方法中 super 和 this 使用的方法就差不多了.只不过 super 要考虑是否能访问其父类的资源.

2. 作用域 public,protected,private,以及不写时的区别?

- Public:不同包、同一包、类内都可用
- Private: 类内
- Protected: 不同包的子类、同一包、类内都可用
- 不写时:同一包内、类内

3. 编程输出如下图形。

```
*****
*****
***
**
*
```

代码如下:

```
public class Print {
    public static void main(String[] args) {
        for (int i = 0; i < 5; i++) {
            for (int j = 5; j > i; j--) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

```
}  
}  
}
```

4. JAVA 的事件委托机制和垃圾回收机制

java 事件委托机制的概念, 一个源产生一个事件并将它送到一个或多个监听器那里。在这种方案中, 监听器简单的等待, 直到它收到一个事件。一旦事件被接受, 监听器将处理这个事件, 然后返回。

垃圾回收机制 垃圾收集是将分配给对象但不再使用的内存回收或释放的过程。如果一个对象没有指向它的引用或者其赋值为 null, 则次对象适合进行垃圾回收

5. 在 JAVA 中, 如何跳出当前的多重嵌套循环?

用 break; return 方法。

6. 什么是 java 序列化, 如何实现 java 序列化? (写一个实例)

序列化:

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, 接着, 使用 ObjectOutputStream 对象的 writeObject(Object obj) 方法就可以将参数为 obj 的对象写出(即保存其状态), 要恢复的话则用输入流。

7. 一个 ".java" 源文件中是否可以包括多个类 (不是内部类)? 有什么限制?

可以。如果这个类的修饰符是 public, 其类名与文件名必须相同。

8. 排序都有哪几种方法? 请列举。用 JAVA 实现一个快速排序?

排序的方法有: 插入排序 (直接插入排序、希尔排序), 交换排序 (冒泡排序、快速排序), 选择排序 (直接选择排序、堆排序), 归并排序, 分配排序 (箱排序、基数排序)

快速排序的伪代码。

9. Overload 和 Override 的区别。Overloaded 的方法是否可以改变返回值的类型？

方法的

重写 Override, 子类覆盖父类的方法, 将子类传与父类的引用调用的还是子类的方法。

重载 Overloading 一个类多个方法, 名称相同, 参数个数类型不同。
两者都是 Java 多态性的不同表现。

Overloaded 的方法是可以改变返回值的类型。

```
1, public class Ctest()  
{  
    Public static void main()  
    {  
System.out.println(8+8+"88"+8+8);  
    }  
}
```

168888

10. Final 类有什么特点？

属性常量

方法不可以 overriding

类不可以继承

11. 继承时候类的执行顺序问题,一般都是选择题,问你将会打印出什么？

答:父类:

```
package test;  
public class FatherClass  
{  
    public FatherClass()  
    {  
System.out.println("FatherClass Create");  
    }  
}
```

子类:

```
package test;  
import test.FatherClass;  
public class ChildClass extends FatherClass  
{
```

```
public ChildClass()
{
    System.out.println("ChildClass Create");
}
public static void main(String[] args)
{
    FatherClass fc = new FatherClass();
    ChildClass cc = new ChildClass();
}
}
```

输出结果:

```
C:>java test.ChildClass
FatherClass Create
FatherClass Create
ChildClass Create
```

12. 内部类的实现方式?

答: 示例代码如下:

```
package test;
public class OuterClass
{
    private class InterClass
    {
        Public InterClass()
        {
            System.out.println("InterClass Create");
        }
    }
    public OuterClass()
    {
        InterClass ic = new InterClass();
        System.out.println("OuterClass Create");
    }
    public static void main(String[] args)
    {
        OuterClass oc = new OuterClass();
    }
}
```

输出结果:

```
C:>java test/OuterClass
InterClass Create
OuterClass Create
```

13. 用 JAVA 实现一种排序，JAVA 类实现序列化的方法(二种)?

14. 如在 COLLECTION 框架中，实现比较要实现什么样的接口?

15. 用插入法进行排序代码如下

```
package test;
import java.util.*;
class InsertSort
{
    ArrayList al;
    public InsertSort(int num,int mod)
    {
        al = new ArrayList(num);
        Random rand = new Random();
        System.out.println("The ArrayList Sort Before:");
        for (int i=0;i<num ;i++ )
        {
            al.add(new Integer(Math.abs(rand.nextInt()) % mod + 1));
            System.out.println("al ["+i+"]="+al.get(i));
        }
    }
    public void SortIt()
    {
        Integer tempInt;
        int MaxSize=1;
        for(int i=1;i<al.size();i++)
        {
            tempInt = (Integer)al.remove(i);
            if(tempInt.intValue()>=((Integer)al.get(MaxSize-1)).intValue())
            {
                al.add(MaxSize, tempInt);
                MaxSize++;
                System.out.println(al.toString());
            } else {
                for (int j=0;j<MaxSize ;j++ )
                {
                    if
                    (((Integer)al.get(j)).intValue()>=tempInt.intValue())
                    {
                        al.add(j, tempInt);
                        MaxSize++;
                    }
                }
            }
        }
    }
}
```

```

System.out.println(al.toString());
break;
}
}
}
}
System.out.println("The ArrayList Sort After:");
for(int i=0;i<al.size();i++)
{
System.out.println("al["+i+"]="+al.get(i));
}
}
public static void main(String[] args)
{
InsertSort is = new InsertSort(10,100);
is.SortIt();
}
}

```

JAVA 类实现序列化的方法是实现 `java.io.Serializable` 接口
 Collection 框架中实现比较要实现 `Comparable` 接口和 `Comparator` 接口

16. 编程：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。但是要保证汉字不被截半个，如"我 ABC"4，应该截为"我 AB"，输入"我 ABC 汉 DEF"，6，应该输出为"我 ABC"而不是"我 ABC+汉的半个"。

答：代码如下：

```

public static void split(String source,int num) throws Exception
{
    int k=0;
    String temp="";
    for (int i = 0; i <source.length(); i++)
    {
        byte[] b=(source.charAt(i)+"").getBytes();
        k=k+b.length;
        if(k>num)
        {
            break;
        }
    }
}

```

```
        temp=temp+source.charAt(i);
    }
    System.out.println(temp);
}
```

15、Java 编程,打印昨天的当前时刻

```
public class YesterdayCurrent{
public void main(String[] args){
Calendar cal = Calendar.getInstance();
cal.add(Calendar.DATE, -1);
System.out.println(cal.getTime());
}
}
```

16、文件读写,实现一个计数器

```
public int getNum(){
int i = -1;
try{
String stri="";
BufferedReader in = new BufferedReader(new FileReader(f));
while((stri=in.readLine())!=null){
i = Integer.parseInt(stri.trim());
}
in.close();
}catch(Exception e){
e.printStackTrace();
}
return i;
}
public void setNum(){
int i = getNum();
i++;
try{
PrintWriter out=new PrintWriter(new BufferedWriter(new
FileWriter(f, false)));
out.write(String.valueOf(i)); //可能是编码的原因,如果直接写入int的话,
将出现java编码和windows编码的混乱,因此此处写入的是String
out.close();
}catch(Exception e){
e.printStackTrace();
}
```

```
}  
}
```

17、指出下面程序的运行结果。

```
class A{  
    static{  
        System.out.print("1");  
    }  
    public A(){  
        System.out.print("2");  
    }  
}  
class B extends A{  
    static{  
        System.out.print("a");  
    }  
    public B(){  
        System.out.print("b");  
    }  
}  
public class Hello{  
    public static void main(String[] args){  
        A ab = new B(); //执行到此处,结果: 1a2b  
        ab = new B(); //执行到此处,结果: 1a2b2b  
    }  
}
```

注:类的 static 代码段,可以看作是类首次加载(被虚拟机加载)执行的代码,而对于类的加载,首先要执行其基类的构造,再执行其本身的构造

18、抽象类和接口的区别?

- (1)接口可以被多重 implements,抽象类只能被单一 extends
 - (2)接口只有定义,抽象类可以有定义和实现
 - (3)接口的字段定义默认为:public static final,抽象类字段默认是 "friendly"(本包可见)
- 当功能需要累积时用抽象类,不需要累积时用接口。

19、什么是类的反射机制?

通过类(Class 对象),可以得出当前类的 fields、method、construtor、interface、superClass、modified 等,同是可以通过类实例化一个实例、设置

属性、唤醒方法。Spring 中一切都是反射、struts、hibernate 都是通过类的反射进行开发的。

20、类的反射机制中的包及核心类？

```
java.lang.Class  
java.lang.reflection.Method  
java.lang.reflection.Field  
java.lang.reflection.Constructor  
java.lang.reflection.Modifier  
java.lang.reflection.Interface
```

21、得到 Class 的三个过程是什么？

```
对象.getClass()  
    类.class 或 Integer.type(int) Integer.class(java.lang.Integer)  
Class.forName();
```

22、如何唤起类中的一个方法？

产生一个 Class 数组，说明方法的参数
通过 Class 对象及方法参数得到 Method
通过 method.invoke(实例, 参数值数组) 唤醒方法

23、如何将数值型字符转换为数字 (Integer, Double) ?

```
Integer.parseInt("1234")  
Double.parseDouble("123.2")
```

24、如何将数字转换为字符？

```
1+""  
1.0+""
```

25、如何去小数点前两位，并四舍五入。

```
double d=1256.22d;  
d=d/100;  
System.out.println(Math.round(d)*100);
```

26、如何取得年月日，小时分秒？

```
Calendar c=Calendar.getInstance();
    c.set(Calendar.YEAR, 2004);
    c.set(Calendar.MONTH, 0);
    c.set(Calendar.DAY_OF_MONTH, 31);
    System.out.println(c.get(Calendar.YEAR)+"
"+(c.get(Calendar.MONTH)+1)+" "+c.get(Calendar.DAY_OF_MONTH));
```

27、如何取得从 1970 年到现在的毫秒数

```
Java.util.Date dat=new Date();
Long now=dat.getTime();
```

28、如何获取某个日期是当月的最后一天？

当前日期加一天，若当前日期与结果的月份不相同，就是最后一天。

取下一个月的第一天，下一个月的第一天-1

```
public static void main(String[] args)
{
    Calendar c=Calendar.getInstance();
    c.set(Calendar.YEAR, 2004);
    c.set(Calendar.MONTH, 0);
    c.set(Calendar.DAY_OF_MONTH, 30);
    Calendar c1=(Calendar)c.clone();
    System.out.println(c.get(Calendar.YEAR)+"
"+(c.get(Calendar.MONTH)+1)+" "+c.get(Calendar.DAY_OF_MONTH));

    c.add(Calendar.DAY_OF_MONTH, 1);
    if(c.get(Calendar.MONTH)!=c1.get(Calendar.MONTH))
    {
        System.out.println("是最后一天");
    }
    else
    {
        System.out.println("不是取后一天");
    }
}
```

29、如何格式化日期？

```
Import java.text. SimpleDateFormat;
SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
Date dat=new Date();
//把日期转化为字符串
String str=sdf.format(dat);
System.out.println(str);
//将字符串转化为日期
Java.util.Date d1=sdf.parse("yyyy-mm-dd");
```

30、编码转换，怎样实现将 GB2312 编码的字符串转换为 ISO-8859-1 编码的字符串。

```
String a=new String("中".getBytes("gb2312"),"iso-8859-1");
```

```
String a=new String("中".getBytes("iso-8859-1"));
```

32、String s = new String("xyz");创建了几个 String Object?

New 了一个,"XYZ"本来又是一个
两个

33、float 型 float f=3.4 是否正确？

报错，应当是 float f=3.4f
如果是 float f=3(整数)正确

35、说出一些常用的类，包，接口，请各举 5 个

常用的类：BufferedReader BufferedWriter FileReader FileWriter
String Integer

常用的包：java.lang java.awt java.io java.util java.sql javax.xml
javax.servlet javax.ejb. java.net javax.faces

常用的接口：List Map Document NodeList EJBObject EJBHome
SessionBean EntityBean

36、java 中会存在内存泄漏吗，请简单描述。

会。如：`int i,i2; return (i-i2);` //when i 为足够大的正数,i2 为足够大的负数。结果会造成溢位，导致错误。

37、java 中实现多态的机制是什么？

静态的多态:方法名相同，参数个数或类型不相同。(overloading)

动态的多态:

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、垃圾回收器的基本原理是什么？垃圾回收器可以马上回收内存吗？有什么办法主动通知虚拟机进行垃圾回收？

动态内存

存放类实例

静态内存

类本身

垃圾收集主要针对的是动态内存，一般当内存不够用时会进行垃圾收集。或通过 `System.gc()` 手动收集，但不保证一定执行。

39、静态变量和实例变量的区别？

```
static i = 10; //常量
```

```
class A a; a.i =10; //可变
```

静态方法可以调用静态变量。

实现方法可以调用静态变量、实例变量

41、是否可以从一个 static 方法内部发出对非 static 方法的调用？

不可以, 如果其中包含对象的 `method()`；不能保证对象初始化.

42、写 clone()方法时，通常都有一行代码，是什么？

`Clone` 有缺省行为，`super.clone()`；他负责产生正确大小的空间，并逐位复制。

43、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？

Try: 执行部分，产生异常

Catch: 捕捉异常

Finally: 不管有没有异常都执行

Throws: 在方法声明处声明要抛出的异常，调用者必须对其进行处理。

Throw: 抛出一个异常

在 try 中可以抛出异常，一般与声明的异常相同。

自定义异常要继承于 Exception 或 Exception 的子类

45、冒泡排序法

//相邻两个数比较，将最小或最大的放到后面，最后面数的不参与比较

```
public class BubbleSort {
    private static int al[] = new int[10];
    public BubbleSort() {
        al[0]=2;
        al[1]=3;
        al[2]=23;
        al[3]=45;
        al[4]=1;
        al[5]=67;
        al[6]=23;
        al[7]=80;
        al[8]=35;
        al[9]=72;
    }
    public static void main(String[] args) {
        BubbleSort bs = new BubbleSort();
        System.out.println("排序前: ");
        display(al);

        for(int i=0;i<al.length;i++) {

            for (int j = 0; j < al.length-i-1; j++) {

                if(al[j]>al[j+1]) {
                    swap(j,j+1);
                }
            }
        }
    }
}
```

```

    }
    System.out.println();
    System.out.println("排序后: ");
    display(al);
}
private static void display(int[] al2) {
    for (int i = 0; i < al2.length; i++) {
        System.out.print(al2[i]+" ");
    }
}
private static void swap(int i, int j) {
    int temp = al[i];
    al[i]= al[j];
    al[j] = temp;
}
}

```

46、String and StringBuffer 的区别?

String:长度给定不可变,当多个字符串联合时,要先转为StringBuffer,再联合,速度慢。

StringBuffer:长度可变,可以将多个字符串值直接联合,效率高

47、用 java 代码编写堆栈

```

public class Stack {

    int[] data;
    int maxSize;
    int top;
    public Stack(int maxSize) {
        this.maxSize = maxSize;
        data = new int[maxSize];
        top = -1;
    }

    /**
     * 依次加入数据
     * @param data 要加入的数据
     * @return 添加是否成功
     */
    public boolean push(int data) {
        if(top+1== maxSize) {
            System.out.println("栈已满!");
            return false;
        }
    }
}

```

```

    }
    this.data[++top] = data;
    return true;
}

/**
 * 从栈中取出数据
 * @return 取出的数据
 */
public int pop() throws Exception{
    if(top==-1) {
        throw new Exception("栈已空!");
    }
    return this.data[top--];
}

public static void main(String[] args) throws Exception {
    Stack stack=new Stack(1000);
    stack.push(1);
    stack.push(2);
    stack.push(3);
    stack.push(4);
    stack.push(5);
    while(stack.top>=0)
    {
        System.out.println(stack.pop());
    }
}
}

```

48、集合的作用是什么？

数据的传送 增、删、改、查、containsAll，可以存放不同类型的对象。

49、集合的通用方法有那些？通用方法是什么？(操作)

集合 List 的遍历方法有：

Iterator:

Enumeration

For

Get

set

Collection 的通用方法有：

```
Iterator()
Add()
Clear();
remove()
```

50、说出 ArrayList, Vector, LinkedList 的存储性能和特性 HashMap 和 Hashtable 的区别

ArrayList Vector: 以数组的方式存储, 增、删慢, 查、改快

ArrayList: 线程不安全, 速度快

Vector: 线程安全, 速度慢 (synchronized)

LinkedList: 以单链表的方式存储, 增、删快, 查、改慢

HashMap 与 Hashtable 都实现的 Map 接口, Hashtable 线程安全, HashMap 线程不安全。

51、Collection 和 Collections 的区别。

Collection 是集合的根接口, 其下有 set 及 list

Collections 是集合的算法。

52、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? 用 contains 来区分是否有重复的对象。还是都不用。

在比较时先调用 hashCode 方法, 如果不相同, 证明不相等。

如果相同, 再调用 equals 方法, 如果 equals 方法相同, 证明相等, 不相同, 证明不相等。

==: 主要用在基本数据类型及引用

Equals: 主要是对象或对象引用的比较。

集合中是否包含某一个元素用 contains 来判断。

53、List, Set, Map 是否继承自 Collection 接口?

List, set 继承于 Collection

Map 没有继承于 Collection, 其相对是独立的。

属于 Collection 类型的对象, 可以通过构造函数将一个集合构造成另外一个集

合。

54、面向对象的特征有哪些方面

1. 抽象:

找共性，将共有的属性、方法放到父类中

2. 继承:

子类继承于父类，具有父类的所有属性与方法，可以重用，也可以覆盖。

3. 封装:

一个类包括多个属性及方法。

4. 多态性:

动态:

静态:

55、String 是最基本的数据类型吗?

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

java.lang.String 类是 final 类型的，因此不可以继承这个类、不能修改这个类。为了提高效率节省空间，我们应该用 StringBuffer 类

56、int 和 Integer 有什么区别?

Int 是基本数据类型，不是对象，占一个内存空间，没有方法。与其同类的有 long,char,doble
Integer 是封装类，具有方法及属性。与其同类的有 Long,Double.Float

57、运行时异常与一般异常有何异同?

运行时异常: java JVM 抛出的异常，代码中不用处理。

一般异常: 用户抛出的异常，如果用 throws 声明了，调用这个方法的代码必须对其处理。

58、&和&&的区别?

&:与: 左边若为 false 右边还执行。

&&:短路与,左边若为 false 右边不执行。

59、final, finally, finalize 的区别?

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。

`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。

算符可以用来决定某对象的类是否实现了接口。

62、heap 和 stack 有什么区别？

栈是一种线形集合，其添加和删除元素的操作应在同一段完成。栈按照后进先出的方式进行处理。

堆是栈的一个组成元素

63、Static Nested Class 和 Inner Class 的不同？

Static Nested Class 是被声明为静态（`static`）的内部类，它可以不依赖于外部类实例被实例化。而通常的内部类需要在外部类实例化后才能实例化。

64、什么时候用 assert？

`assertion`（断言）在软件开发中是一种常用的调试方式，很多开发语言中都支持这种机制。在实现中，`assertion` 就是在程序中的一条语句，它对一个 `boolean` 表达式进行检查，一个正确程序必须保证这个 `boolean` 表达式的值为 `true`；如果该值为 `false`，说明程序已经处于不正确的状态下，系统将给出警告或退出。一般来说，`assertion` 用于保证程序最基本、关键的正确性。`assertion` 检查通常在开发和测试时开启。为了提高性能，在软件发布后，`assertion` 检查通常是关闭的。

65、GC 是什么？为什么要有 GC？

GC 是垃圾收集的意思（`Garbage 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。

66、`short s1 = 1; s1 = s1 + 1;`有什么错？`short s1 = 1; s1 += 1;`有什么错？

`short s1 = 1; s1 = s1 + 1;`（`s1+1` 运算结果是 `int` 型，需要强制转换类型）
`short s1 = 1; s1 += 1;`（可以正确编译）

67、Math.round(11.5)等於多少? Math.round(-11.5)等於多少?

Math.round(11.5)==12 Math.round(-11.5)==-11 round 方法返回与参数最接近的长整数, 参数加 1/2 后求其 floor.

68、Java 有没有 goto?

java 中的保留字, 现在没有在 java 中使用。

69、给我一个你最常见到的 runtime exception

ArithmeticException, ArrayStoreException, BufferOverflowException, BufferUnderflowException, CannotRedoException, CannotUndoException, ClassCastException, CMMException, ConcurrentModificationException, DOMException, EmptyStackException, IllegalArgumentException, IllegalMonitorStateException, IllegalPathStateException, IllegalStateException, ImagingOpException, IndexOutOfBoundsException, MissingResourceException, NegativeArraySizeException, NoSuchElementException, NullPointerException, ProfileDataException, ProviderException, RasterFormatException, SecurityException, SystemException, UndeclaredThrowableException, UnmodifiableSetException, UnsupportedOperationException

一般异常:

IOException
FileNotFoundException
SQLException

70、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)?

接口可以继承接口。抽象类可以实现(implements)接口, 抽象类是否可继承实体类, 但前提是实体类必须有明确的构造函数。任何抽象类都是实际类 Object 的子类。

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

都不能

72、数组有没有 length()这个方法? String 有没有 length()这个方法?

数组没有 length()这个方法,有 length 这个属性
String 有 length()这个方法.

73、构造器 Constructor 是否可被 override?

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

74、是否可以继承 String 类?

String 类是 final 类故不可以继承。

75、switch 是否能作用在 byte 上,是否能作用在 long 上,是否能作用在 String 上?

switch (expr1) 中,expr1 是一个整数表达式。因此传递给 switch 和 case 语句的参数应该是 int、short、char 或者 byte。long,string 都不能作用于 switch。

76、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?

会执行,在 return 前执行。

77、编程题:用最有效率的方法算出 2 乘以 8 等於几?

$2 \ll 3$

78、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?

不对,有相同的 hash code。

79、当一个对象被当作参数传递到一个方法后，此方法可改变这个对象的属性，并可返回变化后的结果，那么这里到底是值传递还是引用传递？

是引用传递

基本数据类型:值

对象: 引用

80、四种会话跟踪技术

Cookie

Session

Hidden

url 重写

81、编程题：写一个 Singleton 出来。

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，在类初始化时实例化，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

83、Java 中的异常处理机制的简单原理和应用。

原理

有错直接转到异常处理部分或向上抛出。

应用:

JAVA 的异常就是错误，有两种一种是运行时，编码可以不用捕捉。一种是一般异常，如果 throws 声明了，必须进行处理。

84、垃圾回收的优点和原理。并考虑 2 种回收机制。

优点:

程序员不用管内存，jvm 自动完成，开发方便。运行优先非常低，程序无法清楚实例什么时候被销毁。

85、描述一下 JVM 加载 class 文件的原理机制？

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。

86、char 型变量中能不能存贮一个中文汉字?为什么？

能够定义成为一个中文的，因为 java 中以 uni code 编码，一个 char 占 16 个字节，所以放一个中文是没问题的

88、写一个程序，从文件（c:\test.txt）中查出字符串“mobnet”出现的次数？

java 基础类库(io 流, 集合类, 线程, Socket, AWT, Swing,sql)

1、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?

字节流, 字符流。字节流继承于 InputStream OutputStream, 字符流继承于 Reader Writer。在 java.io 包中还有许多其他的流, 低层流与调层流, 高层流主要是为了提高性能和使用方便。

2、启动一个线程是用 run()还是 start()?

启动一个线程是调用 start()方法, 启动线程并调用 run 方法。

3、线程的基本概念、线程的基本状态以及状态之间的关系

线程是进程内的并发, 没有自己内存空间, 共享进程的, 线程间的通信成本较低。Java 中的线程有四种状态分别是: 运行、就绪、挂起、结束。

4、多线程有几种实现方法,都是什么?同步有几种实现方法,都是什么? 用什么关键字修饰同步方法? stop()和 suspend()方法为何不推荐使用?

```
Extends Thread
Implements Runnable
同步
Public synchronized aa()
{
}

Public void cc(object aa)
{
    synchronized(aa)
    {
    }
}
```

```
}
```

用 `synchronized` 修饰同步方法。

答：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`
反对使用 `stop()`，是因为它不安全。它会解除由线程获取的所有锁定，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。结果很难检查出真正的问题所在。`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。

5、集合框架有什么？

Collection				Map
List		set		HashMap
ArrayList	LinkedList	HashSet	TreeSet	

12、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

```
public class TestThread
{
    private int j;
    public synchronized void inc()
    {
        j++;
        System.out.println(Thread.currentThread().getName() + "-inc: " +
j);
    }
    public synchronized void dec()
    {
        j--;
        System.out.println(Thread.currentThread().getName() + "-dec: " +
j);
    }
    public static void main(String[] args)
```

```
{
    TestThread t=new TestThread();
    for (int i = 0; i < 2; i++)
    {
        Thread inc=new Thread(new Inc(t));
        Thread dec=new Thread(new Dec(t));
        inc.start();
        dec.start();
    }
}
```

```
class Inc implements Runnable
{
    private TestThread obj;
    public Inc(TestThread obj)
    {
        this.obj=obj;
    }
    public void run()
    {
//        for (int i = 0; i < 100; i++)
//            {
//                this.obj.inc();
//            }
    }
}
```

```
class Dec implements Runnable
{
    private TestThread obj;
    public Dec(TestThread obj)
    {
        this.obj=obj;
    }
    public void run()
    {
//        for (int i = 0; i < 100; i++)
//            {
//                this.obj.dec();
//            }
    }
}
```

13、同步和异步有和异同，在什么情况下分别使用他们？举例说明。

同步：上一段代码没的完成，下一段必须等到上一段代码完成后才可以执行。如买票排队

异步：上一段代码没的完成，下一段不必等到上一段代码完成就可以执行。如手机发送短信。

14、sleep() 和 wait() 有什么区别？

Sleep 是指休眠给定的时间，当这个时间达到之后，线程会再次醒来。

Wait 是等待状态，多长时间不清楚，由另一个线程将其唤醒。

15、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？

如只其它方法是同步方法，不可以进入。如果不是可以进入。

17、输入输出流的理解：

在 java 使用流的机制进行数据的传送，从文件到内存是输入流，从内存到文件是输出流，输入流可以通过 read 读取，输出流以 write 或 print 写入，对于流可以是分为高层流和低层流，低层以一个字节或字符为单位进行处理，高层流以一批数据为单位进行处理。

FileInputStream (System.in) 至 InputSteamReader 至 BufferedReader
OutputSteam(System.out)至 printStream
FileReader 至 BufferedReader
FileWriter 至 PrintWri ter 或 bufferWri ter

分类：

字节(二进制)

FileInputStream(低层输入流)
FileOutputStream(低层输出流)

PrintStream(高层流) System.out.println()

字符(一个 char)

FileReader
FileWri ter

18、请写一个程序的读写，要求用两种方式一种是低层流另一种是高层流。

```
import java.io. FileWri ter;
import java.io. InputS tream;
import java.io. Buffer edWri ter;
import java.io. InputS treamR eader;
import java.io. Buffer edR eader;
import java.io. FileR eader;

public class Untitled1 {
    public static void writeFileChar() throws Exception {
        FileWri ter f = new FileWri ter("c:\\aa.txt");
        InputS tream is = System.in;
        int c = is.read();
        while (((char) c) != 'x') {
            f.wri te(c);
            c = is.read();
        }
        f.close();
        is.close();
    }

    public static void writeFileString() throws Exception {
        FileWri ter f = new FileWri ter("c:\\aa.txt");
        Buffer edWri ter bwr=new Buffer edWri ter(f);
        Buffer edR eader bf = new Buffer edR eader(new
InputS treamR eader(System.in));
        String c = bf.readLine();
        while (!c.equals("stop")) {
            bwr.wri te(c+"\r\n");
            c = bf.readLine();
        }
        bwr.close();
        f.close();
        bf.close();
    }

    public static void readFileChar() throws Exception {
        FileR eader f = new FileR eader("c:\\aa.txt");
        int c = f.read();
        while (c!=-1) {
            System.out.print((char)c);
        }
    }
}
```

```

        c=f.read();
    }
    f.close();
}

public static void readFileString() throws Exception {
    BufferedReader bf = new BufferedReader(new
FileReader("c:\\aa.txt"));
    String c = bf.readLine();
    while (c!=null)
    {
        System.out.println(c);
        c=bf.readLine();
    }
    bf.close();
}

public static void main(String[] args) throws Exception {
    readFileString();
}
}

```

19、如何列出某个目录下的所有文件

```

import java.io.File;
File f=new File("C:\\");
File[] f1=f.listFiles();
for(int i=0;i<f1.length;i++)
{
    if(f1[i].isDirectory())
    {
        System.out.println("directory is"+f1[i].getName());
    }
    else
    {
        System.out.println("file is"+f1[i].getName());
    }
}
}
}

```

1. 如何列出某个目录下的所有子目录

```

public static void main(String[] args) throws Exception
{
    getFile(new File("C:\\entityBean"), "\\t");
}

```

```

}
public static void getFile(File f,String sem) throws Exception
{
    System.out.println(sem+f.getName());
    File fl[]=f.listFiles();
    if(fl.length>=1)
    {
        for(int i=0;i<fl.length;i++)
        {
            if(fl[i].isDirectory())
            {
                getFile(fl[i],sem+"\t");
            }
        }
    }
}

```

2. 判断一个文件或目录是否存在

```

File f=new File("C:\\entityBean");
if(f.exists())
{
    System.out.println("exist");
}
else
{
    System.out.println("not exist");
}
Socket

```

20、用 socket 通讯写出客户端和服务端端的通讯，要求客户发送数据后能够回显相同的数据？

```

public class ServerSocket_1
{
    public static void main(String[] args)
        throws Exception
    {
        ServerSocket ss = new ServerSocket(4001);
        Socket s = ss.accept();
        BufferedReader br = new BufferedReader(new InputStreamReader(s.
            getInputStream()));
        PrintStream ps=new PrintStream(s.getOutputStream());
        String temp = br.readLine();
        while (true)

```

```

        {
            System.out.println("客户端: "+temp);
            ps.println(temp);
            if (temp.equals("stop"))
            {
                break;
            }
            temp = br.readLine();
        }
        br.close();
        ps.close();
        ss.close();
    }
}

public class ClientSocket
{
    public static void main(String[] args) throws Exception
    {
        Socket s = new Socket("localhost", 4001);
        PrintStream ps = new PrintStream(s.getOutputStream());
        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        BufferedReader br_server = new BufferedReader(new
InputStreamReader(s.
        getInputStream()));
        String temp = br.readLine();
        while (true)
        {
            ps.println(temp);
            temp = br_server.readLine();
            System.out.println("服务器的信息: " + temp);
            if (temp.equals("stop"))
            {
                break;
            }
            temp = br.readLine();
        }
        s.close();
        br.close();
        br_server.close();
    }
}

```

23、介绍 JAVA 中的 Collection Framework(包括如何写自己的数据结构)?

答: Collection Framework 如下:

```
Collection
├── List
│   ├── LinkedList
│   ├── ArrayList
│   └── Vector
│       └── Stack
└── Set
Map
├── Hashtable
├── HashMap
└── WeakHashMap
```

Collection 是最基本的集合接口, 一个 Collection 代表一组 Object, 即 Collection 的元素 (Elements)

Map 提供 key 到 value 的映射

24、请说出你所知道的线程同步的方法

HashTable 中的 put,get,remove

Vector 的相关方法。

jdbc 数据访问技术

1、JDBC 如何做事务处理?

```
Con.setAutoCommit(false)
Con.commit();
Con.rollback();
```

2、写出几个在 Jdbc 中常用的接口

PreparedStatement, CallableStatement, Statement, Connection, ResultSet

3、简述你对 Statement, PreparedStatement, CallableStatement 的理解

statement 用于执行静态 SQL 语句并返回它所生成结果的对象, 在执行时

确定 sql。

PreparedStatement 表示预编译的 SQL 语句的对象。SQL 语句被预编译并且存储在 PreparedStatement 对象中。然后可以使用此对象高效地多次执行该语句，可以传参数，在得到 PreparedStatement 对象时确定 sql。

CallableStatement 用于执行 SQL 存储过程的接口。如果有输出参数要注册说明是输出参数。

4、Java 中访问数据库的步骤？

1 连接 Oracle 数据库

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
Connection
```

```
con=DriverManager.openConnection("jdbc:oracle:thin:@localhost:1521:Database", "UserName", "Password")
```

1. 利用 JDBC 检索出表中的数据

```
Class.forName("");
```

```
Connection con=DriverManager.openConnection(" ", " ", " ")
```

```
PreparedStatement ps=con.prepareStatement("select * from [table]");
```

```
ResultSet rs=ps.executeQuery();
```

```
While(rs.next)
```

```
{
```

```
Rs.getString(1) 或 rs.getString(" 字段名" )
```

```
}
```

5、JDBC 中的核心类及其作用是什么？

```
DriverManager
```

```
Class.forName();
```

```
DriverManager.openConnection("","sa","")
```

```
Connection
```

```
PreparedStatement(Statement)
```

```
ResultSet rs=executeQuery() dql
```

```
While(rs.next())
```

```
{
```

```
}
```

```
executeUpdate() dml ddl
```

6、执行存储过程用那一个类，如何操作输出参数?(操作)

```
CallableStatement c=con. prepareCall ("{call getCustomerName(?,?)}");
c.setString(1,"1");
c.registerOutParameter(2,java.sql.Types.VARCHAR);
c.execute();
c.getString(2);
```

8、可能会让你写一段 Jdbc 连 Oracle 的程序.

```
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:acc
p","system","system");
```

9、Class.forName 的作用?为什么要用?

注册一个数据库驱动，将驱动加载到当前的 JVM 中。

10、Jdo 是什么?

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 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。（o/rMapping 工具 集合处理）

11、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

```
create or replace package myPack
is
    type c_type is ref cursor;
    procedure getPage(v_sql varchar2, pageSi ze number, pageI ndex number, c
out c_type);
end;

create or replace package body myPack
```

```

is
    procedure getPage(v_sql varchar2, pageSi ze number, pageI ndex number, c
out c_type)
    is
        pageTotal int:=0;
        pageFirstRow int:=0;
        pageLastRow int:=0;
        rowTotal int:=0;
    begin
        execute immediate 'select count(*)   from ('||v_sql||')' into
rowTotal ;
        pageTotal :=cei l (rowTotal /pageSi ze);
        if(pageI ndex<1) then
            raise_appl icati on_error(-20001,' 页数不能小于 1');
        end if;
        if(pageI ndex>pageTotal) then
            raise_appl icati on_error(-20001,' 页数太大， 不能读取');
        end if;
        pageFi rstRow:=(pageI ndex-1)*pageI ndex+1;
        pageLastRow:=pageFi rstRow+pageSi ze;
        open c for ' select * from '||v_sql||' where rownum<' ||
            pageLastRow||' minus select * from '||v_sql
            ||' where rownum<' ||pageFi rstRow;
    end;
end;

```

Web 编程 Jsp&Servlet 技术

1、简单说说 tomcat 的配置？

JAVA_HOME=JDK 的根目录

CATALINA_HOME=tomcat 的根目录

CATALINA-HOME\conf\server.xml:可以配置 tomcat 的端口，可以配置 tomcat 中下连接池。

CATALINA-HOME\common\lib:存放公用的类包

在 My ecl ipse 中如何配置 tomcat

在 ecl ipse 中 ， 选 择 windows->preference->MyEcl ipse->Appl icati onServer->Tomcat 选择 Tomcat 的安装目录， 并选择 TomCat 所需的 jdk， 选择 enable, 确定即可。

2、JSP 中动态 INCLUDE 与静态 INCLUDE 的区别？

jsp:include: 在运行时调用另一个页面，变量是可以重复的。

<%include file=""%>: 在转译时合在一起，会成为同一个类，变量不可以重复。

3、forward 和 redirect 的区别？

forward: 转发，在下一个页面中, request 保留上一个页面中的 request 的所有值

redirect: 跳转，不传递 request 对象。

4、Servlet 的体系结构是什么？

Servlet
GenericServlet
HttpServlet
自定义

5、如何实现一个自定义的 servlet？

extends HttpServlet 并覆盖 doPost 或 doGet 方法

在 web.xml 中进行部署

6、Servlet 的生命周期是什么？

Init
多次执行 doGet 或 doPost
destroy

7、jsp 就是一个 servlet 是否正确？

对

8、请罗列 jsp 中的脚本、指令及动作？

脚本

<%> <%=> <%!> <%---->

指令

<%@page contentType="text/html ; charset=utf-8"

```
language="java" import=""%>
    <%include file=""%>
    <%taglib uri="" prefix=""%>
```

动作:

```
<jsp:useBean class="" id="" scope="" > 在 scope 中如果没有实例化一个对象, 如果有直接用以前的。
```

```
<jsp:getProperty name="" property="" > 向一个 bean 中设置属性值
```

```
<jsp:forward > jsp 页的转发
```

```
<jsp:include page="" > 导入一个 jsp 页面
```

9、JSP 的内置对象及方法

request request 表示 HttpServletRequest 对象。取客户端表单域信息及 cookie, header, 和 session

response response 表示 HttpServletResponse 对象, 对客户端的响应返回文本、写 cookies。

out out 向客户端打印 html 文本。

pageContext : 当前 jsp 页面的上下文环境, 可以得到 session、request、application 等内置对象, 在自定义标签中使用的很多。

session session 表示一个请求的 javax.servlet.http.HttpSession 对象。

Session 一个用户多个页面共享同一变量。

application application 表示一个 javax.servlet.ServletContext 对象。存放容器级的变量。

config 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 page 表示从该页面产生的一个 servlet 实例

exception: 异常, 当 isErrorPage=true

10、说出在 JSP 页面里是怎么分页的?

页面需要保存以下参数: (数据库的分页及比较)

总行数: 根据 sql 语句得到总行数

每页显示行数: 设定值

当前页数: 请求参数

页面根据当前页数和每页行数计算出当前页第一行行数, 定位结果集到此行, 对结果集取出每页显示行数的行即可。

数据库:

Sql server:

```
SELECT TOP 页大小 *
```

```
FROM TestTable
```

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

```
--pageSi ze=5 页大小
--pageI ndex=2 所要的页
--如果有主键可以，没以及键不行
select top 5 *
from aa where a1 not in
(select top 5 a1 from aa order by a1)
order by a1;
```

```
oracle:
select * from ' ||v_sql ||' where rownum<' ||
      pageLastRow||' minus select * from ' ||v_sql
      ||' where rownum<' ||pageFi rstRow;
```

Session

先取出数据中的所有信息封装到对象并保存在 session 中，转发到 jsp 页面做如下处理。

```
<table border="1">
  <tr>
    <td>a1</td>
    <td>a2</td>
  </tr>
<%
  List l=(List)session.getAttribute("as");
  //一页显示多少行
  int pageSi ze=3;
  //总页数
  int pageCount=0;
  int currentPage=1;
  if(l!=null && l.size()>0)
  {
    pageCount=(l.size()/pageSi ze)+(l.size()%pageSi ze==0?0:1);
    if(request.getParameter("page")!=null)
    {
      currentPage=Integer.parseInt(request.getParameter("page"));
    }
    if(currentPage<1)
```

```

        {
            currentPage=1;
        }
        if(currentPage>pageCount)
        {
            currentPage=pageCount;
        }
        for      (int      i      =      (currentPage-1)*pageSize;      i
<(currentPage-1)*pageSize+pageSize; i++)
        {
            if(i>=l.size())
            {
                break;
            }
            Aa aa=(Aa)l.get(i);
            %>
            <tr>
            <td><%=aa.getA1()%></td>
            <td><%=aa.getA2()%></td>
            </tr>
            <%
            }
            }
            %>
            <tr>
            <td colspan="2">
            <%
            if(currentPage!=1)
            {
            %>
            <a href="page.jsp?page=1">首页</a>&nbsp;&nbsp;&nbsp;
            <a href="page.jsp?page=<%=currentPage-1%>">上一页</a>&nbsp;&nbsp;&nbsp;
            <%
            }
            if(currentPage!=pageCount)
            {
            %>
            <a href="page.jsp?page=<%=currentPage+1%>">下一页</a>&nbsp;&nbsp;&nbsp;
            <a href="page.jsp?page=<%=pageCount%>">最后一页</a>&nbsp;&nbsp;&nbsp;
            <%
            }
            %>
            </td>
            </tr>

```

</table>

11、include 的两种实现方式的区别？

<@include file>: 在将 jsp 生成 servlet 类前将两个文件和在一起，生成一个 java 类，一起运行的。所以是一家子，当中的变量名不能重名。

<jsp:include page>: 是两个类，是一个调用关系，在运行时动态的调用，不是一家子，可以重复变量。

12、jsp 页面中两种跳转方式分别是什么？有什么区别？

转发: 保留上次的 request

```
<jsp:forward>
    actionMapping.findForWard("");
    pageContext.forward();
    request.getRequestDispatcher("a.jsp").forward(request, response)
```

跳转: 不保留上次的 request

```
Response.sendRedirect("")
```

13、描述 JSP 和 Servlet 的区别、共同点、各自应用的范围

Jsp 主要在于页面的显示动态生成页面，可以与 html 标记一起使用，其还是要生成为一个 servlet。

Servlet: 主要是控制的处理，如调用业务层，跳转不同的 jsp 页面。

Mvc:

```
Jsp: v
Servlet: c
```

14、在 JSP 中如何读取客户端的请求，如何确定某个 Jsp 文件的真实路径？

```
Request.getParameter("")
<%=application.getRealPath("aa.jsp") %>
```

15、描述 Cookie 和 Session 的作用，区别和各自的应用范围，Session 工作

原理。

Cookie: 主要用在保存客户端，其值在客户端与服务端之间传送，不安全，存储的数据量有限。

Session: 保存在服务端，每一个 session 在服务端有一个 sessionID 作一个标识。存储的数据量大，安全性高。占用服务端的内存资源。

16、说明 Jsp 中 errorPage 的作用，应用范围。

正常页面中

```
%@page errorpage="error.jsp"%
```

错误页面

```
<%@page isErrorpage="true"%>
```

有一内置对象: exception

17、介绍在 Jsp 中如何使用 JavaBeans

```
<jsp:useBean class="" id="" scope=""/>
```

```
<%
```

```
    New 类();
```

```
%>
```

19、简单介绍 JSP 的标记库

做一个标记处理类 extends TagSupport

通过 tld 说明标记处理的类的前缀及后缀

在 web.xml 中说明 tld 文件

```
<taglib>
```

```
    <taglib-uri>
```

```
    <taglib-location>
```

```
</taglib>
```

在 jsp 页面是引用 tld<%@taglib uri=" " prefix=" " %>

20、Servlet 中的核心类有那些，各有什么特点？

ServletContext: 容器，放置全局变量

```
    setAttribute()
```

```
    getAttribute()
```

ServletConfig: 一个 servlet 的配置

```
    getInitParameter("名称")
```

HttpServletRequest: 封装的所有的请求

```
    getParameterValue("名称")
```

```
    getParameterValues("称")
```

```
    getSession();
```

```
    getAttribute("名称");
```

```
    getRequestDispatcher("a.jsp").forward(request, response)
```

HttpServletResponse: 响应

```
    getOut();
```

```
    sendRedirect("")
```

HttpSession: 一个用户多个页面共享同一变量
setAttribute("","")

21、Servlet 中重要的包有那些，有什么区别？

javax.servlet.*; javax.servlet.http.*;

22、说出 Servlet 的生命周期，并说出 Servlet 和 CGI 的区别？

Servlet 被服务器实例化后，容器运行其 init 方法，请求到达时运行其 service 方法，service 方法自动派遣运行与请求对应的 doXXX 方法（doGet，doPost）等，当服务器决定将实例销毁的时候调用其 destroy 方法。

与 cgi 的区别在于 **servlet 处理服务器进程中，它通过多线程方式运行其 service 方法，一个实例可以服务于多个请求，并且其实例一般不会销毁**，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。

23、什么情况下调用 doGet()和 doPost()？

Jsp 页面中的 form 标签里的 method 属性为 get 时调用 doGet()，为 post 时调用 doPost()。

25、如何现实 servlet 的单线程模式

在 doGet 及 doPost 方法前加入 synchronized

JSP:

```
<%@ page isThreadSafe="true"%>
```

27、Request 对象的主要方法：

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

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

28、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?

```
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;  
}
```

30、Servlet 执行时一般实现哪几个方法?

```
public void init(ServletConfig config)  
public ServletConfig getServletConfig()  
public String getServletInfo()  
public void service(ServletRequest request, ServletResponse response)  
public void destroy()
```

5 Hibernate 持久层技术

1、在 myeclipse 加入 hibernate 环境的全过程是什么？

Db-browsers 加入配置连接

新建工程

加入 hibernate 环境，指定 *. hbm. xml 及 Hi bernalteSessi onFactory 文件所在的位置

2、hibernate 的核心配置文件是什么及其作用？

Hi bernalte. cfg. xml : 数据库连接、指定相关的映射文件

*. hbm. xml : 具体的 o/r mappi ng 说明

3、hibernate 的核心类是什么，它们的相互关系是什么？重要的方法是什么？

Confi gurati on

Sessi onFactory

Sessi on 如下方法

Save

load

Update

Delete

Query q=CreateQuery(“from Customer where

customerName=: customerName”)

begi nTransacti on

close

Transacti on

Commi t()

4、关联:

one-to-many

i nverse: 主控方，外键的关系有谁控制

i nverse=false 是主控方，外键是由它控制的

i nverse=true 是被控方，外键与它没关系

要想实现主控方的控制必须将被控方作为主控方的属性

cascade: 级联

主表增从表增

主表修从表修

主表删从表删

lazy: 延迟

lazy=false: 一下将所有的内容取出, 不延时(常用)

lazy=true: 取出部分内容, 其余内容动态去取
通过 get 可以取出对方的所有内容

5、hibernate 中的 one-to-many 或 many-to-one 中常用的方式是什么?

主控方在 many 这边, 不及连删除

6、Criteria 的作用?

```
Criteria c=session.createCriteria(Customer.class);  
//设置条件  
c.add(Expression.ge(" 字段名" ," 值对象" ))  
    ge: >=  
    gt: >  
    le: <=  
    lt: <  
    eq: =  
//排序  
    c.addOrder(Order.asc(" 字段名" ))  
//分页  
    c.setFirstResult(1)//从第 2 行开始提取  
    c.setMaxResults(5)//返回 5 行
```

7、DetachedCriteria 的作用?

产生时不需要 session

```
DetachedCriteria dc= DetachedCriteria.forClass(Customer.class)  
Criteria c=Dc.getExecutableCriteria(session)
```

8、Query

1 个或多个属性查询:

```
Query query=session.createQuery("select customername, customerid  
from Customer")  
List l=query.list();  
For(int i=0; i<l.size(); i++)  
{  
    Object[] object=(Object[])l.get(i);  
    Object[0] object[1]  
}  
}
```

分组: "select count(*), productname from Product group by productname order by productname"

取值与属性一样

配置的查询, 在 *. hbm. xml 中

```
<query name="sql">
  <![CDATA[
    from Product where productid=:productid
  ]]>
</query>
```

```
Query query=session.getNamedQuery(sql);
```

联接 1

"from Customer as customer join fetch customer.buySet": 将多的放到 buySet 属性中, 得出的结是 Customer 有一个, Buy 有多个

联接 2

"from Customer as customer join customer.buySet": 得出的对象, customer 与 buy 是 1 对 1

子查询:

```
"from Customer as customer where (select count(*) from customer.buySet)>1"
```

9、继承关系的实现

1、两个表, 子类重复父类的属性。

2、一个表, 子类父类共用一个表

```
<class name="Users" table="users"
discriminator-value="Users">
  <discriminator column="DISCRIMINATOR_USERTYPE"
type="string"/>
<subclass name="admin" discriminator-value="admin">
  <property name="adminRemark" column="admin_remark"
type="string" />
</subclass>
</class>
```

3、两个表, 子类引用父类的主键, 享用公共的字段或属性。

```
<class name="Users" table="users">
  <id name="userid" column="USERID" type="string">
    <generator class="assigned"/>
  </id>
  <property name="pwd" column="pwd" type="string" />
<joined-subclass name="Guest" table="guest">
  <key column="USERID"/>
  <property name="guestRemark" column="guest_remark"
type="string" />
</joined-subclass>
```

```
</class>
```

```
批量删除
```

```
Query query=session.createQuery("update"或"delete");  
query.executeUpdate();
```

10、tomcat 连接池: 在容器中预先产生了 n 个连接实例, 客户端不用重新实例化, 可以直接取。

6.1、在 tomcat-5.0\conf\server.xml 中</host>前面加入如下内容

```
<Context path="/app1" docBase="app1" debug="0" reloadable="true"  
crossContext="true">
```

```
<Resource name="jdbc/sa" auth="Container"  
type="javax.sql.DataSource"/>
```

```
<ResourceParams name="jdbc/sa">
```

```
<parameter>
```

```
<name>factory</name>
```

```
<value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
```

```
</parameter>
```

```
<parameter>
```

```
<name>driverClassName</name>
```

```
<value>com.microsoft.jdbc.sqlserver.SQLServerDriver</value>
```

```
</parameter>
```

```
<parameter>
```

```
<name>url</name>
```

```
<value>jdbc:microsoft:sqlserver://127.0.0.1:1433;DatabaseName=jsp  
dev;SelectMethod=cursor</value>
```

```
</parameter>
```

```
<parameter>
```

```
<name>username</name>
```

```
<value>sa</value>
```

```
</parameter>
```

```
<parameter>
```

```
<name>password</name>
```

```
<value></value>
```

```
</parameter>
```

```
<parameter>
```

```
<name>maxActive</name>
```

```
<value>20</value>
```

```

</parameter>
  <parameter>
    <name>maxIdle</name>
    <value>10</value>
  </parameter>
  <parameter>
    <name>maxWait</name>
    <value>-1</value>
  </parameter>
</ResourceParams>
</Context>

```

6.2、将 sql-server 包拷贝到 C:\tomcat-5\common\lib

6.3、jdbc 测试代码

```

Context initCtx = new InitialContext();
Context envCtx = (Context)initCtx.lookup("java:comp/env");
ds = (DataSource)envCtx.lookup("jdbc/sa");
Connection conn = ds.getConnection();

```

6.4、hibernate 通过连接池实现连接

```

<session-factory name="foo">
  <property
name="connection.datasource">java:comp/env/jdbc/sa</property>  <!--指
定 tomcat 连接池-->
  <property name="show_sql">true</property>  <!-- 是否显示
sql -->
  <property
name="dialect">org.hibernate.dialect.SQLServerDialect</property>
<!--hibernate 的驱动管理器-->
  <mapping resource="com/accp/t15/Customer.hbm.xml"/>
</session-factory>

```

11、对象的三大状态

自由(transient)

与 session 无关

持久(persistent)

由 session 来管理

在持久状态中通过 get 方法取出对方

游离(detached)

被 session 抛弃

12、hibernate 常见优化策略

用最新版本的 hibernate

制定合理的缓存策略

采用合理的 session 管理机制

尽量使用延迟加载

many

大文本、大文件

设定合理的批处理参数(batch-size)

如有可能, 选用 uuid 作为主键生成器

如有可能, 选用基于 version 的乐观锁替代悲观锁

开发过程中, 打开 hibernate 的 SQL 日志输出(.hibernate.show_sql=true), 通过观察 hibernate 生成的 sql 语句进一步了解其实现原理, 从而指事实上更好的实现策略。

6. iBatis 持久层技术

用 ibatis 的原因:

只对开发团队提供几条 Select SQL (或存储过程) 以获取所需数据, 具体的表结构不予公开。

开发规范中要求, 所有牵涉到业务逻辑部分的数据库操作, 必须在数据库层由存储过程实现

系统数据处理量巨大, 性能要求极为苛刻, 这往往意味着我们必须通过经过高度优化的 SQL 语句

(或存储过程) 才能达到系统性能设计指标。

jdbc、hibernate、ibatis 的区别

jdbc:手动

手动写 sql

delete、insert、update 要将对象的值一个一个取出传到 sql 中,不能直接传入一个对象。

select:返回的是一个 resultset, 要从 ResultSet 中一行一行、一个字段一个字段的取出, 然后封装到一个对象中, 不直接返回一个对象。

ibatis 的特点:半自动化

sql 要手动写

delete、insert、update:直接传入一个对象

select:直接返回一个对象

hibernate:全自动
不写 sql,自动封装
delete、insert、update:直接传入一个对象
select:直接返回一个对象

ibatis 的核心配置文件:

sqlmapclient.xml : 数据库连接及相关 o/rMapping 的映射文件
(hibernate.cfg.xml)

sqlmapBuy.xml:具体的 o/rmapping 映射(*.hbm.xml)

四大标记:

select
update
delete
insert
procedure

ibatis 的核心类:

SqlMapClientBuilder:加载配置文件, 返回一个会话。

SqlMapClient:具体的会话

List list=queryForList("标签名",object);
Object obj=queryForObject("标签名",object);
delete("标签名",object)
update("标签名",object)
insert("标签名",object)

工程的使用:

将 ibatisfactory 拷贝到工程目录下

将 ms 的 3 个包及 ibatis 的三个包拷贝到/WEB-INF/lib 下

修改 ibatisfactory 中的 abatorConfig.xml 文件

进入 ibatisfactory 目录运行 java -jar abator.jar abatorConfig.xml true

将 sql 标记、select、update、insert、delete 的副本删除

spring&ibatis:

dataSource

sqlMapClient:SqlMapClientFactoryBean

configLocation

classpath:sqlMapClient.xml

/WEB-INF/sqlMapClient.xml

dataSource

transactionManager:DataSourceTransactionManager

dataSource

customerDao extends SqlMapClientDaoSupport

sqlMapClient

buyDao
 sqlMapClient
Facade
 buyDao
 customerDao
abstractProxy:TransactionProxyFactoryBean
 transactionManager
 transactionAttributes
facadeProxy
 target:façade

7 Struts 界面控制层技术

1、请说出 struts 框架的几大组件？

- 1、MVC
- 2、标签库
- 3、校验框架
- 4、国际化
- 5、tiles

3、struts 的核心类有那些，在 MVC 模式中其对应的关系是什么？

C

ActionServlet
RequestProcessor
Action
actionMapping(struts-config.xml)
actionForward

V

ActionForm
Jsp

M

Dao
Manager (facade)

4、Struts 的处理请求的全过程是什么？

url -> ActionServlet(process) -> RequestProcessor(process) -> 实例化 form
-> 填充 form 值 -> 进行校验 -> 实例化 action -> 调用 execute

5、在 struts 中如何通过一个 url 找到一个 action，它的核心配置文件是什么？

配置文件是 struts-config.xml

6、为什么使用 MVC，其主要目的是什么？

让 v 与 M 强制解耦，提高可重用性（旅馆的服务员(C)）

7、对于 MVC 在 action 中对应有类有几种，各有什么作用？

Action:

1) 基本的

DispatchAction:

2) 存在多个方法，根据页面传入的表单域的值调用不同的方法，表单域的名称在<action param="" />标记中进行配置

LookupDispatchAction

3) 多个按钮用同一个 action 的不同方法。实现 getMap 方法,说明每一个按钮在*.properties 中的键名及键值，在 struts-config.xml 通过 parameter 说明按钮的名称，按钮的值对应*.properties 的值，通过值找键名，通过键名找 Map 中的键名找到值就是相应的方法。

MappingDispatchAction: 未知

forwardAction: 直接跳转到不同页面不执行逻辑(???)

类所在的包: org.apache.struts.actions.ForwardAction

8、struts 的标记有几类，请列举并说明其作用？

Bean:

```
<bean:define scope="" name="" property="" id="" />
```

```
<bean:write name="customer" property="customerName" />
```

```
<bean:message key="" />国际化
```

Logic:

```
<logic:iterator> //将集合的内容取出
```

```
<logic:present> //
```

```
<logic:equals> //
```

Html:

```
<html:file>上传文件
```

```
<html:select property="sex">
```

```
<html:options collection="" property="" labelProperty="" />
```

```
</html:select>
```

9、如何在 struts 中配置数据源在,什么文件?用什么标签?如何取出 DataSource?

Struts-config.xml

```
<data-sources>
  <data-source key="data"
    type="org.apache.commons.dbcp.BasicDataSource">
    <set-property property="driverClassName"
      value="com.microsoft.jdbc.sqlserver.SQLServerDriver" />
    <set-property property="url"
      value="jdbc:microsoft:sqlserver://localhost:1433;databaseName=t23" />
    <set-property property="username" value="sa" />
    <set-property property="password" value="" />
  </data-source>
</data-sources>
```

DataSource

```
ds=(DataSource)this.getServlet().getServletContext().getAttribute("data");
Connection con=ds.getConnection();
```

10、如何在 jbuilder 中开发 struts?

工程

Web 模型 (选中 struts1.2.8, 会自动加在 web-inf 中生成 tld 及 struts-config.xml, 并加载相关的类包)

建一个 ActionForm

建一个 Action

修改 struts-config.xml 文件

11、如何实现 struts 的 validator 框架?

手动:

```
public class myForm extends ActionForm
{
    public ActionErrors validate()
    {
actionErrors.add(ActionErrors.GLOBAL_MESSAGE, new
ActionMessage("properties 中的键名"));
    }
}
```

```
}
struts-config.xml 中修改 action 的标签 validate=true
input="错误页面"
```

如果 validate 方法中的 ActionErrors 不为空且 size>0 时会回到 input 页面。

自动

```
public class myForm extends ValidateForm
{
    不能覆盖 validate 方法。
    //public void validate()
    //{

    //}
}
```

在 struts-config.xml 文件中加入插件

```
<plug-in
className="org.apache.struts.validator.ValidatorPlugIn">
    <set-property property="pathnames"
value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml" />
</plug-in>
```

修改 validation.xml 中的内容

```
errors.required={0} is required.
errors.minLength={0} can not be less than {1} characters.
```

```
<formset>
  <form name="loginForm">
    <field property="userName" depends="required">
      <arg0 key="userName" />
    </field>
    <field property="pwd" depends="required,minLength">
      <arg0 key="pwd" />
      <arg1 key="{var:minLength}" resource="false"/>
      <var>
        <var-name>minLength</var-name>
        <var-value>6</var-value>
      </var>
    </field>
  </form>
</formset>
```

```
struts-config.xml 中修改 action 的标签 validate=true
input="/错误页面"
```

13、如何实现国际化？

国际化:不用修改代码,就适用于不同的语言国家

本地化:如果要适应一个国家,要修改源代码

实现过程:

在 struts-config.xml 中用如下标签说明资源文件名,注意其只是文件名,没有语言_国家

```
<message-resources parameter="struts.ApplicationResources" />
```

在资源文件对应的目录 struts 中加入适应不同语言、国家的 properties 文件

```
ApplicationResources_zh_CN.properties 中国
```

```
ApplicationResources_en_US.properties us
```

如果不是英语要转码

```
native2ascii -encoding gb2312 源
```

```
ApplicationResources_zh_CN.properties
```

在 jsp 页面中用

```
<bean:message key="" />取出信息
```

8 JSF 界面控制层技术

1、Jsf 中的核心类用那些?有什么作用?

核心类

```
FacesServlet
```

```
LiftCycle FacesContext
```

2、Jsf 中的 LiftCycle 六大生命周期是什么?

恢复视图->应用请求值->校验->更新模型->调用应用程序->呈现响应

3、如何管理 web 层中的 Bean, 用什么标签。如何通过 jsp 页面与 Bean 绑定在一起进行处理?

```
<managed-bean>
```

```
  <managed-bean-name>checkNumber</managed-bean-name>
```

```
  <managed-bean-class>j sf1.CheckNumber</managed-bean-class>
```

```
  <managed-bean-scope>sessi on</managed-bean-scope>
```

```
</managed-bean>
```

4、Jsf 中导航的标签是什么？

```
<navigation-rule>
    <from-view-id>/main.jsp</from-view-id>
    <navigation-case>
        <from-outcome>success</from-outcome>
        <to-view-id>/result.jsp</to-view-id>
    </navigation-case>
</navigation-rule>
```

5、jsf 中用户界面组件模型有几类，各代表什么？

UI 组件、事件监听、显示、类型转换、验证

6、表格处理及取值

```
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;
DataModel: 代表一个表格，其可以从客户端传上来。
```

实例化:

```
DataModel dm=new ListDataModel();
```

将一个 List 值放入到 DataModel 中.

```
dm.setWrappedData(list)
```

将客户端当前行取出

```
(Logsv0) dm.getRowData()
```

对于多条的查询及增删改在模型中加入了如下属性:

模型的名称是 bean，其下的属性是

DataModel dm: 代表返回的行数据

V0 vo: 代表一行数据。

取一行:

```
vo=(V0) dm.getRowData();
```

通#{bean.vo.属性名}，如果修改对应信息自动修改。

增加一行:

```
this.vo=new V0();
```

通#{bean.vo.属性名}，显示肯定无值，但客户输入值提交后会

将值写入

7、jsf 的标签库有哪些？

核心: f

校验

```
<f:validateDoubleRange>
```

```
<f: validateLength>
转化
<f: convertDateTime pattern="yyyy-MM-dd"/>
<f: convertNumber type="currency"/>
选择框:
<f: selectItem/>
<f: selectItems/>
```

html:h

选择控件:

```
<h: selectOneRadio>: 一个 string
<h: selectManyCheckBox>: 一个 String 数组
<h: selectOneMenu>: 组合框架选择一个
<h: selectManyMenu>: 组合框架选择多个
<h: selectOneList>: 下拉列表选择一个
<h: selectManyList>: 下拉列表选择多个
```

表格:

```
<h: dataTable value="集合" var="集合中的一个">
  <h: column>
    <!--头-->
    <f: facet name="header">
      <h: outputText >
    </f: facet>
    <!--具体的行值-->
    <h: outputText value=""/>
  <h: column>
</h: dataTable>
```

9 Spring 应用框架技术

1、Spring 和 Struts 的区别?

struts: 是一种基于 MVC 模式的一个 web 层的处理。

Spring:提供了通用的服务, ioc/di aop,关心的不仅仅 web 层, 应当 j2ee 整体的一个服务, 可以很容易融合不同的技术 struts hibernate ibatis ejb remote springJDBC springMVC

2、什么是 aop, aop 的作用是什么?

Oop: 纵向的业务

Aop: oop 的一个横向的服务, 是对 oop 进一步的补充, 提供安全、事务、日志等的集中式处理, 相关的装备 before、around、after exception

3、aop 中的关键名词有些那些，相互关系是什么？

拦截器: 代理
装备(advice)
目标对象
关切点:条件
连接点:方法、属性

4、依赖注入的方式有几种，各是什么？

Setter
Interface
constructor

5、spring 中的核心类有那些，各有什么作用？

BeanFactory: 产生一个新的实例，可以实现单例模式
BeanWrapper: 提供统一的 get 及 set 方法
ApplicationContext: 提供框架的实现，包括 BeanFactory 的所有功能

6、ApplicationContext 的作用

beanFactory
国际化(getMessage)
资源管理:可以直接读取一个文件的内容(getResource)
加入 web 框架中(加入一个 servlet 或监听器)
事件处理

7、如何实现资源管理

使用
applicationContext.getResource("classpath: 文件名"): 在 src 根目录下，
在类路径下
applicationContext.getResource("classpath: /chap01/文件名"): 以 src
根目录下的基准往下走。
applicationContext.getResource("file:c:/a.properties"): 在系统文件目
录下。

8、如何实现加入 web 框架中

在 web.xml 中加入如下内容，在启动 web 服务器时加载
/WEB-INF/applicationContext.xml 中的内容。

```
<servlet>
<servlet-name>context</servlet-name>
<servlet-class>
org.springframework.web.context.ContextLoaderServlet
</servlet-class>
<load-on-startup>1</load-on-startup>
</servlet>
```

通过如下类得到 ApplicationContext 实例

```
WebApplicationContextUtils.getWebApplicationContext
```

9、如何实现事件处理

事件

Extends ApplicationEvent

监听器

Implements ApplicationListener

事件源

Implements ApplicationContextAware

在 applicationContext.xml 中配置事件源、监听器

先得到事件源，调用事件源的方法，通知监听器。

10、spring 的 ioc 及 di 代表什么意思？

Ioc:程序在运行过程中，根据配置文件动态加载所依赖的配置类

、如何在 spring 中实现国际化？

在 applicationContext.xml 加载一个 bean

```
<bean id="messageSource"
class="org.springframework.context.support.ResourceBundleMessageSource">
  <property name="basename">
    <value>message</value>
  </property>
</bean>
```

在 src 目录下建多个 properties 文件

对于非英文的要用 native2ascii -encoding gb2312 源 目转化文件相关内容

其命名格式是 message_语言_国家。

页面中的中显示提示信息，键名取键值。

当给定国家，系统会自动加载对应的国家的 properties 信息。

通过 applicationContext.getMessage(“键名” ,”参数” ,”区域”)取出相关的信息。

12、spring 的配置的主要标签是什么?有什么作用?

```
<beans>
    <bean id="" class="" init="" destroy="" singleton="">
        <property name="">
            <value></value>
        </property>
        <property name="">
            <ref local></ref>
        </property>
    </bean>
</beans>
```

13、spring 与 ejb2.0 的事务管理比较的优缺点?

测试:

Spring: poj o

Ej b: 二个接口一个类, 一堆配置文件

事务类型

Spring: jdbc jta hibernate

Ej b: jta

成本

Spring: 普通容器(tomcat jboss)

Ej b: webl ogic jboss

开发的周期:

Spring 远比 ej b 快.

14、spring 的 jdbc 与传统的 jdbc 有什么区别, 其核心类有那些?

Spring 的 jdbc: 节省代码, 不管连接(Connection), 不管事务、不管异常、不管关闭(con.close() ps.close)

JdbcTemplate(dataSource): 增、删、改、查

TransactionTemplate(transactionManager): 进行事务处理

15、在 spring 中有几种事务管理, 分别是什么?

代码管理的事务处理

TransactionTemplate 的 execute 方法中的内部类 TransactionCallback 中的 doInTransaction 方法中使用。

```
public void make()
```

```

    {
        TransactionTemplate jtm=new
TransactionTemplate(this.getTransactionManager());
        jtm.execute(new myClass1());
    }
public class myClass1 implements TransactionCallback
{

    public Object doInTransaction(TransactionStatus trans)
    {
        JdbcTemplate jdbc=new JdbcTemplate(dataSource);
        jdbc.execute("insert into customer(customerName)
values('b')");
        jdbc.execute("insert into customer(customerName)
values('b')");
        return null;
    }
}

```

容器管理的事务处理

16、在 spring 中如何配代码的事务管理？

Datasouce
 transactionManager
 userDao 要注入
 Datasouce
 transactionManager
 通过如下类实现
 TransactionTemplate
 JdbcTemplate

17、在 spring 中如何配容器的事务管理，相关的类有那些？

Datasouce
 transactionManager
 userDao 要注入
 Datasouce
 Proxy 代理
 Target: userDao: 代理对象(目标对象)
 transactionAttributes(那些方法需要事务处理)
 transactionManager(事务处理服务)

18、如果 spring 与 hibernate 结合在一起可以不需要 hibernate.cfg.xml 文件是否正确?

不需要

19、spring+hibernate 的配置文件中的主要类有那些?如何配置?

在 myeclipse 中先加入 spring 环境再加入 hibernate 环境。

如果 spring 与 hibernate 结合在一起可以不需要 hibernate.cfg.xml 文件是否正确?

spring+hibernate 的配置文件中的主要类有那些?如何配置?

- dataSource
- sessionFactory: hibernate.cfg.xml
- transactionManager
- userDao (extends HibernateDaoSupport)
 - sessionFactory
- facade
- proxy
 - sessionFactory
 - transactionManager
 - facade

20、spring+hibernate 的代码实现中,对于实现类一定继承于一个类是那一个,它有什么作用。

extends HibernateDaoSupport, 可以节省代码。

21、如何配置 spring+struts?

在 struts-config.xml 加入一个插件,通过它加载 applicationContext.xml
在 struts-config.xml 修改 action-mapping 标记,具体 action 交给了 DelegateActionProxy

通过 DelegateActionProxy 进入 spring 的环境。

在 spring 的 applicationContext.xml 加入 <bean name="/login" class="" singleton="false"/>

22、如何在 web 环境中配置 applicationContext.xml 文件?

```
<listener>  
  <listener-class>
```

```
        org.springframework.web.context.ContextLoaderListener
    </listener-class>
</listener>
或:
<servlet>
    <servlet-name>context</servlet-name>
    <servlet-class>
        org.springframework.web.context.ContextLoaderServlet
    </servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
```

通过如下方法取出 applicationContext 实例:

```
ApplicationContext
```

```
ac=WebApplicationContextUtils.getWebApplicationContext(this.getServletContext());
```

24、Jsf 和 spring 的区别?

jsf: 是一种基于 MVC 模式的一个 web 层的处理, 粒度较 struts 较细。

Spring:提供了通用的服务, ioc/di aop,关心的不仅仅 web 层, 应当 j2ee 整体的一个服务, 可以很容易融合不同的技术 struts hibernate ibatis ejb remote springJDBC springMVC

Ejb 技术

1、weblogic 的热发布

将 ear、jar、war 拷到 C:\bea\user_projects\domains\mydomain\applications 目录

weblogic 会自动发布

通过 jbuilder 将 ear 或 jar 或 war 部署到服务器上。

2、在 ejb 中实现 one-to-many

- 1、在 many 中的将外键字段属性删除
- 2、在删除 many 中的值时要将 Collection 转化为 ArrayList, 并反向遍历 ArrayList, 先删除 ArrayList 中值, 根据 ArrayList 删除反回的对象转化为 many 的远程接口, 通过远程接口将 many 删除

3、ejb 所用的技术:

Jndi: java naming directory inferface

Rmi: remote method invoke

4、实现 ejb 几个接口,几个类?

两个接口一个类

extends EJBHome

通过 jndi 得到 home 接口

create 方法调用服务端的 ejbCreate 方法, 在服务端产生一个 EntityBean 或 SessionBean 实例, 向客户端返回一个远程接口。

通过 find 方法在从服务端找到一个 EntityBean 实例, 向客户端返回一个远程接口。

extends EJBObject

在客户端通过 rmi 调用远程服务端方法。

通过 remove 方法调用服务端的 ejbRemove 方法, 将 EntityBean 删除

implements SessionBean

在服务端实现真正的处理, 实现核心业务

5、实现 ejb 相关的配置文件是什么?

ejb-jar.xml: 说明 ejb 的两个接口一个类的。

weblogic-ejb-jar.xml:说明 ejb 的 JNDI 名

weblogic-rdbms-ejb.xml:o-rMapping 实现数据库表、字段与 ejb 属性对应的关系。

ejb2.0 的开发

1、用 jbuilder 集成环境

2、可以用 eclipse 开发, 用源代码注释说明, 用 ant 执行 xdoclet, xdoclet 通过源代码注释自动生相关的配置、两个接口一个类。

```
/**
```

```
* @stateLess
```

```
* @remote
```

```
*/
```

6、ejb 的分类?区别

sessionBean

stateless:方法间不保留(1..1)

statefull:方法间保留(1..n)

entityBean:持久化

cmp:增删改容器

bmp:增删改手动 jdbc

message driver bean

异处的消息处理

7、本地接口与远程接口的区别。

EJBHome(在不同的进程或主机间调用,即不同的 jvm)

EJBObject

EJBLocalHome(在同一进程, 同是 jvm 中)

EJBLocalObject

8、请求处理的过程?

会话

url

factory

通过 jndi 得到一个 home 实例

在客户端通过 home 实例在服务端产生一个 sessionBean, 客户端返回一个接口

客户端通过远程接口调用方法。

9、statefull 的生命周期

不存在

setSessionContext

create--->ejbcreate

就绪:可以调用 remove 方法将 sessionBean 删除、可以调用服务端的任何方法。

ejbPassivate(从就绪到挂起)

ejbActivate(从挂起到就绪)

挂起

如果超时自动删除

10、stateless 的生命周期

不存在

setSessionContext

create--->ejbcreate

remove-->ejbremove

就绪

11、entityBean 的生命周期:

不存在

setEntityContext

create-->ejbcreate
入池
空房子，没加载数据
ejbActivate
ejbPassivate
就绪
remove-->ejbRemove
加载了数据库的数据

12、EJB 需直接实现它的业务接口或 Home 接口吗，请简述理由。

远程接口和 Home 接口不需要直接实现，
他们的实现代码是由服务器产生的，
程序运行中通过接口调用服务端产生的实例。

13、EJB 的激活机制

以 Stateful Session Bean 为例：其 Cache 大小决定了内存中可以同时存在的 Bean 实例的数量，根据 MRU 或 NRU 算法，实例在就绪和挂起状态之间迁移。

就绪:从文件到内存，调用 ejbActivate 方法
挂起:从内存到文件，调用 ejbPassivate 方法

14、EJB 是基于哪些技术实现的？并说出 SessionBean 和 EntityBean 的区别，

EJB 包括 Session Bean、Entity Bean、Message Driven Bean，基于 JNDI、RMI、JTA 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；

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.

15、EJB 的分类是什么?各有什么特点?

sessionBean: 主机重起消失

Stateless: 不记忆

StateFull: 一个用户多个操作可记忆

EntityBean: 持久的数据库中

Cmp: 容器通过 o/r mapping 实现数据的持久化，不写 sql，实现方便，在速度慢。

Bmp: 通过 jdbc 实现持久化, 实现复杂，速度快

messageBean: 提供异步处理。

10、EJB 中主要的配置文件及作用是什么?

EJB 部署时需要三个文件:

Ejb-jar.xml:

将二个接口一个类打包在一起，给 EJB 一个名称。

说明当前的 sessionBean 的事务是由容器处理的。

其在所有的服务器上通用的。

Weblogic-ejb-jar.xml:

将一个 EJB 名称，对应一个 JNDI，在仅对 weblogic 服务器

Weblogic-rdbms-jar.xml:

实现 o/r mapping 的说明, 相当于*.hbm.xml

15、说出数据连接池的工作机制是什么?

容器或相关的应用程序在其池中实例化多个连接，当应用程序在使用时，容器直接将池中的连接取出应用程序直接使用，同时当应用程序使用完后，容器还可以将连接收回。从而提高系统的效率。

16、EJB2.0 有哪些内容?分别用在什么场合? EJB2.0 和 EJB1.1 的区别?

sessionBean: 是一个过程

entityBean: 是持久化，代表的是一个业务实体，有主键。

Struts+sessionBean+entityBean

EJB2.0 加入的本地接口及本地 home

EJB1.1 中只有远程接口及远程 home

18、EJB 与 JAVA BEAN 的区别？

Java Bean 是可复用的组件，任何一个 Java 类都可以是一个 Bean。但通常情况下，Java Bean 是被容器所创建（如 Tomcat）的，所以 Java Bean 具有如下特点：

- 一个无参的构造器
- 实现 Serializable 接口
- 私有属性
- 公有 get set 方法

Enterprise Java Bean 是一个分布式组件，其特点是基于（RMI）技术的，可以被远程访问（跨进程、跨计算机）。EJB 必须被布署在 Webspere、WebLogic 容器中，不能直接访问 ejb，而是通过容器访问 ejb, 容器是 ejb 访问的一个代理。

19、EJB 的角色和三个对象

六个角色组成，分别是

- EJB 组件开发者（Enterprise Bean Provider）：sun
- 应用组合者（Application Assembler）真正的开发商
- 部署者（Deployer）：
- EJB 服务器提供者（EJB Server Provider）：sun ibm 小机
- EJB 容器提供者（EJB Container Provider）：weblogic jboss
- 系统管理员（System Administrator）：维护员

三个对象是 Remote（Local）接口、Home（Local Home）接口，Bean 类

20、EJB 容器提供的服务

主要提供

- 安全
- 事务管理
- 分布式
- jts
- 声明周期管理
- 代码产生
- 持续性管理
- 锁和并发管理等服务。

21、EJB 规范规定 EJB 中禁止的操作有哪些？

1. 不能操作线程和线程 API (线程 API 指非线程对象的方法如 notify, wait 等),
2. 不能操作 awt,
3. 不能实现服务器功能,
4. 不能对静态属性存取,
5. 不能使用 IO 操作直接存取文件系统,
6. 不能加载本地库.,
7. 不能将 this 作为变量和返回,
8. 不能循环调用。

26、EJB 的基本架构

答:一个 EJB 包括三个部分:

Remote Interface 接口的代码

```
package Beans;

import javax.ejb.EJBObject;

import java.rmi.RemoteException;

public interface Add extends EJBObject
{

//some method declare

}
```

Home Interface 接口的代码

```
package Beans;

import java.rmi.RemoteException;

import javax.ejb.CreateException;

import javax.ejb.EJBHome;

public interface AddHome extends EJBHome
```

```
{  
  
//some method declare  
  
}  
  
EJB 类的代码  
  
package Beans;  
  
import java.rmi.RemoteException;  
  
import javax.ejb.SessionBean;  
  
import javax.ejb.SessionContext;  
  
public class AddBean Implements SessionBean  
  
{  
  
//some method declare  
  
}
```

30、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。

31 如何查看在 weblogic 中已经发布的 EJB?

可以使用管理控制台，在它的 Deployment 中可以查看所有已发布的 EJB

WebService 技术

1、什么是 Web Service?

Web Service 就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。

使用的技术:

HTTP、XML、SOAP（简单对象访问协议）、WSDL

优点:

跨平台、跨语言、跨系统

SOAP 协议:

SOAP 协议 (Simple Object Access Protocol, 简单对象访问协议)

Tcp/ip → http → soap, soap 通过 xml 文件传送信息

缺点:

(1). WebService 使用了 XML 对数据封装, 会造成大量的数据要在网络中传输。

(2). WebService 规范没有规定任何与实现相关的细节, 包括对象模型、编程语言, 这一点, 它不如 CORBA。

2、什么是 Web 容器?

实现 J2EE 规范中 web 协议的应用. 该协议定义了 web 程序的运行时环境, 包括: 并发性, 安全性, 生命周期管理等等.

就是在 tomcat、weblogic 下运行 jsp、servlet、struts

3、应用服务器有那些?

BEA WebLogic Server, IBM WebSphere Application Server, Oracle9i Application Server, jBoss, Tomcat

5、如何给 weblogic 指定大小的内存?

在启动 Weblogic 的脚本中 (位于所在 Domain 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

6、如何设定的 weblogic 的热启动模式(开发模式)与产品发布模式?

可以在管理控制台中修改对应服务器的启动模式为开发或产品模式之一。或者修改服务的启动文件或者 commenv 文件, 增加 set PRODUCTION_MODE=true。

7、如何启动时不需输入用户名与密码？

修改服务启动文件，增加 WLS_USER 和 WLS_PW 项。也可以在 boot.properties 文件中增加加密过的用户名和密码。

8、在 weblogic 管理制台中对一个应用域(或者说是一个网站,Domain)进行 jms 及 ejb 或连接池等相关信息进行配置后,实际保存在什么文件中？

保存在此 Domain 的 config.xml 文件中，它是服务器的核心配置文件。

9、说说 weblogic 中一个 Domain 的缺省目录结构？比如要将一个简单的 helloWorld.jsp 放入何目录下,然的在浏览器上就可打入 http://主机:端口号 //helloworld.jsp 就可以看到运行结果了？又比如这其中用到了一个自己写的 javaBean 该如何办？

Domain 目录服务器目录 applications，将应用目录放在此目录下将可以作为应用访问，如果是 Web 应用，应用目录需要满足 Web 应用目录要求，jsp 文件可以直接放在应用目录中，Javabean 需要放在应用目录的 WEB-INF 目录的 classes 目录中，设置服务器的缺省应用将可以实现在浏览器上无需输入应用名。

12、CORBA 是什么？用途是什么？

CORBA 标准是公共对象请求代理结构 (Common Object Request Broker Architecture)，由对象管理组织 (Object Management Group，缩写为 OMG) 标准化。它的组成是接口定义语言 (IDL)，语言绑定 (binding: 也译为联编) 和允许应用程序间互操作的协议。其目的为：用不同的程序设计语言书写在不同的进程中运行，为不同的操作系统开发。

13、说说在 weblogic 中开发消息 Bean 时的 persistent 与 non-persistent 的差别

persistent 方式的 MDB 可以保证消息传递的可靠性，也就是如果 EJB 容器出现问题而 JMS 服务器依然会将消息在此 MDB 可用的时候发送过来，而 non-persistent 方式的消息将被丢弃。

14、WEB SERVICE 名词解释。JSWDL 开发包的介绍。JAXP、JAXM 的解释。 SOAP、UDDI,WSDL 解释。

Web Service 是基于网络的、分布式的模块化组件，它执行特定的任务，遵守具体的技术规范，这些规范使得 Web Service 能与其他兼容的组件进行互操作。

JAXP(Java API for XML Parsing) 定义了在使用 DOM, SAX, XSLT 的通用的接口。这样在你的程序中你只要使用这些通用的接口，当你需要改变具体的实现时候也不需要修改代码。

JAXM(Java API for XML Messaging) 是为 SOAP 通信提供访问方法和传输机制的 API。

WSDL 是一种 XML 格式，用于将网络服务描述为一组端点，这些端点对包含面向文档信息或面向过程信息的信息进行操作。这种格式首先对操作和消息进行抽象描述，然后将其绑定到具体的网络协议和消息格式上以定义端点。相关的具体端点即组合成为抽象端点（服务）。

SOAP 即简单对象访问协议(Simple Object Access Protocol)，它是用于交换 XML 编码信息的轻量级协议。

UDDI 的目的是为电子商务建立标准；UDDI 是一套基于 Web 的、分布式的、为 Web Service 提供的、信息注册中心的实现标准规范，同时也包含一组使企业能将自身提供的 Web Service 注册，以使别的企业能够发现的访问协议的实现标准。

j2ee 模式（MVC 模式、Model1, Model2）

1、j2ee 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 Responsibility（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，根据工厂模式实现的类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。

2、说说你所熟悉或听说过的 j2ee 中的几种常用模式?及对设计模式的一些看法

Session Facade Pattern: 使用 SessionBean 访问 EntityBean

Message Facade Pattern: 实现异步调用

EJB Command Pattern: 使用 Command JavaBeans 取代 SessionBean, 实现轻量级访问

Data Transfer Object Factory: 通过 DTO Factory 简化 EntityBean 数据提供特性

Generic Attribute Access: 通过 AttributeAccess 接口简化 EntityBean 数据提供特性

Business Interface: 通过远程(本地)接口和 Bean 类实现相同接口规范业务逻辑一致性

ejb 架构的设计好坏将直接影响系统的性能、可扩展性、可维护性、组件可重用性及开发效率。项目越复杂, 项目队伍越庞大则越能体现良好设计的重要性。

3、解释下面关于 J2EE 的名词

(1)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.

(2)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 and 广播.

(3)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.

(4)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.

(5)RMI: Remote Method Interface, 远程方法调用

4、介绍 J2EE、J2SE、J2ME 的区别。

J2ee: 企业级, 主要的 application server 的 web 及应用服务

J2se: 标准版, 没有 application server

J2me: 手机、pda 的嵌入式开发

5、开发中都用到了那些设计模式?用在什么场合?

每个模式都描述了一个在我们的环境中不断出现的问题, 然后描述了该问题的解决方案的核心. 通过这种方式, 你可以无数次地使用那些已有的解决方案, 无需在重复相同的工作. 主要用到了 MVC 的设计模式. 用来开发 JSP/Servlet 或者

J2EE 的相关应用。简单工厂模式等。

6、J2EE 是什么？

J2EE 是 Sun 公司提出的多层(multi-tiered), 分布式(distributed), 基于组件(component-base)的企业级应用模型 (enterprise 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(client tier)组件, web 层和组件, Business 层和组件, 企业信息系统(EIS)层。

7、J2EE 是技术还是平台还是框架？

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。

其他

1、当前主流的解析器有那些？

DOM: 文档对象模型(document object model)
SAX

2、Dom 解析处理的过程是什么？

```
package ss;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.Attr;
import org.w3c.dom.NodeList;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.Transformer;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

public class XmlParser
{
    public static void main(String[] args)
        throws Exception
```

```

    {
        xmlwriter();
    }

    public static void xmlparser()
        throws Exception
    {
        DocumentBuilderFactory xdf = DocumentBuilderFactory.newInstance();
        DocumentBuilder db = xdf.newDocumentBuilder();
        Document d = db.parse("C:\\A1\\customer.xml");
        NodeList nl = d.getElementsByTagName("customer");
        for (int i = 0; i < nl.getLength(); i++)
        {

            Element e = (Element) nl.item(i);
            Attr a = e.getAttributeNode("customerID");
            System.out.println(a.getNodeValue());
            NodeList nl1 = e.getElementsByTagName("customerName");
            System.out.println(nl1.item(0).getFirstChild().getNodeValue());
        }
    }

    public static void xmlwriter() throws Exception
    {
        DocumentBuilderFactory dbf=DocumentBuilderFactory.newInstance();
        DocumentBuilder db=dbf.newDocumentBuilder();
        Document d=db.parse("C:\\A1\\customer.xml");
        NodeList nl=d.getElementsByTagName("customerName");
        for(int i=0;i<nl.getLength();i++)
        {
            Element e=(Element) nl.item(i);
            System.out.println(e.getFirstChild().getNodeValue());

            e.getFirstChild().setNodeValue(e.getFirstChild().getNodeValue()+"1111111111111111");
        }
        TransformerFactory tff=TransformerFactory.newInstance();
        Transformer tf=tff.newTransformer();
        tf.transform(new DOMSource(d),new StreamResult("c:\\aa.xml"));
    }
}

```

DocumentBuilderFactory

DocumentBuiler db

Document d=db.parse(“具体文件路径”);

NodeList nl=d.getElementsByTagName(“节点名”)

3、Sax 解析处理的过程是什么？

```
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.SAXException;
import org.xml.sax.Attributes;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
public class SaxParser extends DefaultHandler
{
    public void characters(char[] ch, int start, int length)
        throws SAXException
    {
        String temp=new String(ch,start,length);
        System.out.println(temp);
    }

    public void endDocument()
        throws SAXException
    {
        System.out.println("正在开始一个文档");
    }

    public void endElement(String namespaceURI, String localName, String qName)
        throws SAXException
    {
        System.out.println("结束元素"+qName);
    }

    public void startDocument()
        throws SAXException
    {
        System.out.println("开始文档");
    }

    public void startElement(String namespaceURI, String localName,
        String qName, Attributes atts)
        throws SAXException
    {
        System.out.println("开始元素"+qName);
    }
    public static void main(String[] args) throws Exception
    {
        SAXParserFactory spf=SAXParserFactory.newInstance();
        SAXParser sp=spf.newSAXParser();
    }
}
```

```
        sp.parse("C:\\A1\\customer.xml",new SaxParser());
    }
}
```

4、Dom 与 Sax 相比它们的优缺点是什么？

DOM 可以访问任何一个节点，要将所有资源全部加载，比较耗内存,可以修改。

SAX 只能顺序读，不可随意访问，不可写，但速度快。

1. Dom 中的核心接口有那些？

Node

 TextNode

 Element

Attr

NodeList

Document

5、如何将 Dom 对象写入到文件中？

TransformerFactory

Transformer 通过如下方法进行处理

```
Transformer(DOMSource(Document d),ResultStream(OutputStream o))
```

6、用 jdom 解析 xml 文件时如何解决中文问题？

类文件是 utf-8，xml 文件头也应是 utf-8 如<?xml version="1.0" encoding="UTF-8" ?>

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

a: 两种形式 dtd schema,

b: 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),

c: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)

8、标准建模语言 UML 中的各种图？

静态图

用例图、类图、对象图、构件、部署、

动态图(行为图)

活动图,协作图,状态、时序

9、BS 与 CS 的联系与区别。

C/S 是 Client/Server 的缩写。服务器通常采用高性能的 PC、工作站或小型机，并采用大型数据库系统，如 Oracle、Sybase、Informix 或 SQL Server。客户端需要安装专用的客户端软件。

B/S 是 Brower/Server 的缩写，客户机上只要安装一个浏览器（Browser），如 Netscape Navigator 或 Internet Explorer，服务器安装 Oracle、Sybase、Informix 或 SQL Server 等数据库。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。

10、Uml 的概念是什么?主要的工具是什么

Uml: 统一建模语言

工具: Rational Rose

11、Uml 的概念中的九大图形是什么?最重的三个图是什么?各有什么特点?

类图

继承

关联:(1..n n..1 n..n)

依赖

实现接口

聚集

组成

类图的生命周期(开发的所有阶段都使用)

总体设计

详细设计

开发: 以类图进行开发

测试

Use-case 图

关系

用例与用例

包含(include): 必须调用

扩展(extends): 可以调用也可以不调用

角色与角色的关系

泛化

用例与角色的关系:

用例与角色: 通知

角色与用例: 调用

元素

用例

- 角色
- 系统边界
- Use-case 图的生命周期
 - 需求
 - 整体设计
 - 测试(单元测试、集成测试、系统测试、用户验收)
- 时序
 - 元素
 - 横坐标: 对象
 - 纵坐标: 时间
 - 作用
 - 找类
 - 找方法
 - 验证系统的正确
 - 生命周期
 - 详细设计
- 活动(流程图)
 - 作用
 - 分析业务
- 构件图
 - 作用:说明组件与组件之间的关系, 依赖关系
- 部署
 - 作用:说明安装软件的主机之间的关系, 系统运行时的性能的主要影响者。
- 协作
 - 对象与对象之间的调用协作。
- 状态
 - 一个对象状态在不同的动作下的变化。
- 对象
 - 说明对象

13、在类图中如何找类?

找名词