JAVA面试题集1

第一、谈谈final,finally,finalize的区别□ 最常被问到。

□ 第二，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)?□

答:写匿名内部类依赖于继承。

匿名内部类:无类名，无class关键字，

无构造，有继承但没有继承关键字，

有父类或父类接口，

有类体

```
public interface In1 {
   public void test();
}
```

==========

```
public class TestAnn {
   public void go(In1 in){
      in.test();
   }

      public static void main(String[] args) {
            TestAnn ta = new TestAnn();
            ta.go(new In1(){          //匿名内部类
                  public void test(){
                        System.out.println("power");
                  }
            });
            TestAnn ta1= new TestAnn();
            ta1.go(new In1(){
                  public void test(){
                        System.out.println("power2");
                  }
            });
      }
```

□□第三，Static Nested Class 和 Inner Class的不同，说得越多越好(面试题有的很笼统)。

第四，&和&&的区别。□这个问得很少。

□□第五，HashMap和Hashtable的区别75。□□常问。

□□第六，Collection 和 Collections的区别。□□你千万别说一个是单数一个是复数。

□□第七，什么时候用assert。

□□API级的技术人员有可能会问这个。

□□第八，GC是什么? 为什么要有GC?

□□基础。□

□□第九，String s = new String("xyz");创建了几个String Object?□

□□第十，Math.round(11.5)等於多少? Math.round(-11.5)等於多少?□

□□第十一，short s1 = 1; s1 = s1 + 1;有什么错? short s1 = 1; s1 += 1;有什么错?

□□面试题都是很变态的，要做好受虐的准备。

□□第十二，sleep() 和 wait() 有什么区别? □□搞线程的最爱。□

□□第十三，Java有没有goto?

JAVA相关基础知识

1、面向对象的特征有哪些方面

答：主要有以下四方面：

1.抽象：

抽象就是忽略一个主题中与当前目标无关的那些方面，以便更充分地注意与当前目标有关的方面。抽象并不打算了解全部问题，而只是选择其中的一部分，暂时不用部分细节。抽象包括两个方面，一是过程抽象，二是数据抽象。

2.继承：

继承是一种联结类的层次模型，并且允许和鼓励类的重用，它提供了一种明确表述共性的方法。对象的一个新类可以从现有的类中派生，这个过程称为类继承。新类继承了原始类的特性，新类称为原始类的派生类（子类），而原始类称为新类的基类（父类）。派生类可以从它的基类那里继承方法和实例变量，并且类可以修改或增加新的方法使之更适合特殊的需要。

3.封装：

封装是把过程和数据包围起来，对数据的访问只能通过已定义的界面。面向对象计算始于这个基本概念，即现实世界可以被描绘成一系列完全自治、封装的对象，这些对象通过一个受保护的接口访问其他对象。

4. 多态性：

多态性是指允许不同类的对象对同一消息作出响应。多态性包括参数化多态性和包含多态性。多态性语言具有灵活、抽象、行为共享、代码共享的优势，很好的解决了应用程序函数同名问题。

其他特征：抽象关联，聚合，组合，内聚，耦合

2、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。该为了提高效率节省空间，我们应用StringBuffer类

3、int 和 Integer 有什么区别

Java

提供两种不同的类型：引用类型和原始类型（或内置类型）。Int是java的原始数据类型，Integer是java为int提供的封装类。Java为每个原始类型提供了封装类。

原始类型封装类booleanBoolean charCharacter byteByte shortShort intInteger longLong floatFloat doubleDouble

引用类型和原始类型的行为完全不同，并且它们具有不同的语义。引用类型和原始类型具有不同的特征和用法，它们包括：大小和速度问题，这种类型以哪种类型的数据结构存储，当引用类型和原始类型用作某个类的实例数据时所指定的缺省值。对象引用实例变量的缺省值为

null，而原始类型实例变量的缺省值与它们的类型有关。

4、String 和StringBuffer的区别

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。这个String类提供了数值不可改变的字符串。而这个StringBuffer类提供的字符串进行修改。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。如果最后需要String，那么使用StringBuffer的toString()方法

5、运行时异常与一般异常有何异同？

运行时异常异常表示程序运行过程中可能出现的非正常状态，表示虚拟机的通常操作中可能遇到的异常，是一种常见运行错误。java编译器要求方法必须声明抛出可能发生的非运行时异常，但是并不要求必须声明抛出未被捕获的运行时异常。

6、说一说Servlet的生命周期，并说出Servlet和CGI的区别。

答:servlet有良好的生存期的定义，包括加载和实例化、初始化、处理请求以及服务结束。这个生存期由javax.servlet.Servlet接口的init,service和destroy方法表达。

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处于服务器进程中，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。7、说出ArrayList,Vector, LinkedList的存储性能和特性

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。

8、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。

9、Collection 和 Collections的区别。

□□Collection是集合类的上级接口，继承与他的接口主要有Set 和List.

Collections是针对集合类的一个帮助类，他提供一系列静态方法实现对各种集合的搜索、排序、线程安全化等操作。Collections没有实现任何接口。它直接继承了Object。

10、&和&&的区别

□□答:&是位运算符，表示按位与运算，&&是逻辑运算符，表示逻辑与（and）

另外，&也可以用于逻辑运算，与＆＆的区别在于＆是非短路运算符，＆＆是短路运算符

11、HashMap和Hashtable的区别。
HashMap是Hashtable的轻量级实现(非线程安全的实现)，他们都完成了Map接口，主要区别在于HashMap允许空(null)键值(key),由于非线程安全，效率上可能高于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Synchronize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。
12、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。
13、sleep() 和 wait() 有什么区别?
□□
答: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。
14、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Overriding和重载Overloading是Java多态性的不同表现。
重载Overload是一个类中多态性的一种表现，方法名必需相同,参数列表必需不同.(长度不同或类型不同),与返回值类型没有关系.
构造器也可以重载.方法名和参数列表都相同,只有返回类型不相同则是方法重复定义.编译出错.
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。如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。
15、error和exception有什么区别?
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。比如说内存溢出。不可能指望程序能处理这样的情况。
exception 表示一种设计或实现问题。也就是说，它表示如果程序运行正常，从不会发生的情况。
Error表示系统级的错误和程序不必处理的异常，我们无法处理它。
Exception表示是可以捕捉或者需要程序进行处理的异常。
16、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。
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。

17、abstract class和interface有什么区别?

声明方法的存在而不去实现它的类被叫做抽象类(abstract class)，它用于要创建一个体现某些基本行为的类，并为该类声明方法，但不能在该类中实现该类的情况。不能创建abstract类的实例。然而可以创建一个变量，其类型是一个抽象类，并让它指向具体子类的一个实例。不能有抽象构造函数或抽象静态方法。Abstract类的子类为它们父类中的所有抽象方法提供实现，否则它们也是抽象类为。取而代之，在子类中实现该方法。知道其行为的其它类可以在类中实现这些方法。

接口(interface)是抽象类的变体。在接口中，所有方法都是抽象的。多继承性可通过实现这样的接口而获得。接口中的所有方法都是抽象的，没有一个有程序体。接口只可以定义static final成员变量。接口的实现与子类相似，除了该实现类不能从接口定义中继承行为。当类实现特殊接口时，它定义(即将程序体给予)所有这种接口的方法。然后，它可以在实现了该接口的类的任何对象上调用接口的方法。由于有抽象类，它允许使用接口名作为引用变量的类型。通常的动态联编将生效。引用可以转换到接口类型或从接口类型转换，instanceof运算符可以用来决定某对象的类是否实现了接口。

一个只能继承一个抽象类，但却可以实现多个接口。抽象类中可以有也可以没有抽象方法。并且可以定义和常规类一样的变量和方法。而接口中所有的方法都是抽象的，所有的变量都是静态不可修改的。

18、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

19、forward 和redirect的区别

forward是服务器请求资源，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，浏览器根本不知道服务器发送的内容是从哪儿来的，所以它的地址栏中还是原来的地址。

redirect就是服务端根据逻辑,发送一个状态码,告诉浏览器重新去请求那个地址，一般来说浏览器会用刚才请求的所有参数重新请求，所以session,request参数都可以获取。

20、EJB与JAVA BEAN的区别？

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。

21、Static Nested Class 和 Inner Class的不同

□□答:Nested Class（一般是C++的说法），Inner Class (一般是JAVA的说法)。Java内部类与C++嵌套类最大的不同就在于是否有指向外部的引用上。注:

静态内部类(Inner Class)意味着1创建一个static内部类的对象，不需要一个外部类对象，2不能从一个static内部类的一个对象访问一个外部类对象

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。

22、JSP中动态INCLUDE与静态INCLUDE的区别？

动态INCLUDE用jsp:include动作实现 <jsp:include page="included.jsp" flush="true" />它总是会检查所含文件中的变化，适合用于包含动态页面，并且可以带参数。

静态INCLUDE用include伪码实现,定不会检查所含文件的变化，适用于包含静态页面:<%@ include file="included.htm" %>

23、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。

24、GC是什么? 为什么要有GC?

□□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。

25、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)

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

答: 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

总是返回接近0的数round方法返回与参数最接近的长整数，参数加1/2后求其floor.

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

答:两个，一个是对象，一个是对象的引用。

"xyz"本身作为字符常量，在汇编语言中应该作为常量放在数据段，Java有一个类似数据段的constant pool保存这个常量，在classloader加载这个类的时候就把"xyz"和这个类的其他一些信息放在constant pool new String("xyz")根据常量"xyz"在heap上创建String对象所以，一共两个对象

String(String original) Initializes a newly created String object so that it represents the same sequence of characters as the argument; in other words, the newly created string is a copy of the argument string.

29、Java有没有goto?

没有 goto为java中的保留字，现在没有在java中使用。

30、启动一个线程是用run()还是start()?

答:启动一个线程是调用start()方法，使线程所代表的虚拟处理机处于可运行状态，这意味着它可以由JVM调度并执行。这并不意味着线程就会立即运行。run()方法可以产生必须退出的标志来停止一个线程。直接调用run()方法不会产生线程，而是把它当作普通的方法调用，马上执行.

31、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。
32、应用服务器有那些？
BEA WebLogic Server，IBM WebSphere Application Server，Oracle9i Application Server，jBoss，Tomcat
33、给我一个你最常见到的runtime exception。
常见的运行时异常有如下这些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
34、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。接口间继承（extends），不能实现（implements）。
抽象类可以实现(implements)接口，但接口不能实现抽象类。抽象类间也用继承(extends).
抽象类是否可继承实体类，但前提是实体类必须有无参的构造函数。35、List, Set, Map是否继承自Collection接口？
List，Set是，Map不是
36、说出数据连接池的工作机制是什么？
J2EE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。
37、abstract的method是否可同时是static,是否可同时是native，是否可同时是synchronized?
答：都不能
38、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的属性。String有有length()这个方法。
39、Set里的元素能重复吗？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？
Set里的元素是不能重复的，用iterator()方法来区分重复与否。equals()是判读两个Set是否相等。
equals()和==方法决定引用值是否指向同一对象equals()在类中被覆盖，为的是当两个分离的对象的内容和类型相配的话，返回真值。
40、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。

41、是否可以继承String类?
String类是final类故不可以继承。
42、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?
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swtich。
43、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。
44、编程题: 用最有效率的方法算出2乘以8等於几? 有C背景的程序员特别喜欢问这种问题。
答:2 << 3 。如果是2的10次方则是:1<<10.
45、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。
46、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?
是值传递。Java
编程语言只有值传递参数。当一个对象实例作为一个参数被传递到方法中时，参数的值就是对该对象的引用。对象的内容可以在被调用的方法中改变，但对象的引用是永远不会改变的。
47、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?
答案:能，一个对象的synchronized方法只能由一个线程访问。但其他线程可以同时访问这个对象的非synchronized方法
48、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模式通常有几种种形式:
第一种形式: (醉汉式)
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: (懒汉式)

```
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
49、Java的接口和C++的虚类的相同和不同处。
由于Java不支持多继承，而有可能某个类或对象要使用分别在几个类或对象里面的方法或属性，现有的单继承机制就不能满足要求。与继承相比，接口有更高的灵活性，因为接口中没有任何实现代码。当一个类实现了接口以后，该类要实现接口里面所有的方法和属性，并且接口里面的属性在默认状态下面都是public static,所有方法默认情况下是public.一个类可以实现多个接口。
50、Java中的异常处理机制的简单原理和应用。
当JAVA程序违反了JAVA的语义规则时，JAVA虚拟机就会将发生的错误表示为一个异常。违反语义规则包括2种情况。一种是JAVA类库内置的语义检查。例如数组下标越界,会引发IndexOutOfBoundsException;访问null的对象时会引发NullPointerException。另一种情况就是JAVA允许程序员扩展这种语义检查，程序员可以创建自己的异常，并自由选择在何时用throw关键字引发异常。所有的异常都是java.lang.Thowable的子类。
51、垃圾回收的优点和原理。并考虑2种回收机制。
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。由于有个垃圾回收机制，Java中的对象不再有"作用域"的概念，只有对象的引用才有"作用域"。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楚和回收，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。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。
52、请说出你所知道的线程同步的方法。
wait():使一个线程处于等待状态，并且释放所持有的对象的lock。
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。
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。
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。
53、你所知道的集合类都有哪些？主要方法？
最常用的集合类是 List 和 Map。List 的具体实现包括 ArrayList 和
Vector，它们是可变大小的列表，比较适合构建、存储和操作任何类型对象的元素列表。List 适用于按数值索引访问元素的情形。
Map 提供了一个更通用的元素存储方法。Map
集合类用于存储元素对（称作"键"和"值"），其中每个键映射到一个值。
54、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。
55、char型变量中能不能存贮一个中文汉字?为什么?

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

如果用gbk的本地编码的话可以.如果用utf-8的话，可能不行。

56、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?

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

Thread是类继承，但是，当你写的某个类已经继承了其他的类，如JFrame等，你就只能用Runnable接口了(JAVA不支持多重继承)。

Runnable接口只有一个抽象方法，就是run()，你要重写。如果你要做的事情就是实现run的话，使用Runnable接口就简单了。一般用法如下:

```
  先 定义Thread clock;
public class Show implemens Runnable{
   clock = new Thread(this);
clock.start();
}
```

我曾经试过，括号内必须加上目标对象this，否则，start函数执行默认的抽象Runnable接口里的run函数，什么都不做，加上目标对象后，它就执行目标对象里的run函数，所以，目标对象类必须实现Runnable接口，并且重写run函数。

57、JSP的内置对象及方法。

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

58、线程的基本概念、线程的基本状态以及状态之间的关系

线程指在程序执行过程中，能够执行程序代码的一个执行单位，每个程序至少都有一个线程，也就是程序本身。

Java中的线程有四种状态分别是:运行、就绪、挂起、结束。

?新建 (Born) : 新建的线程处于新建状态?就绪 (Ready) : 在创建线程后，它将处于就绪状态，等待 start() 方法被调用?运行 (Running) : 线程在开始执行时进入运行状态?睡眠 (Sleeping) : 线程的执行可通过使用 sleep() 方法来暂时中止。在睡眠后，线程将进入就绪状态?等待 (Waiting) : 如果调用了 wait() 方法，线程将处于等待状态。用于在两个或多个线程并发运行时。?挂起 (Suspended) : 在临时停止或中断线程的执行时，线程就处于挂起状态。?恢复 (Resume) : 在挂起的线程被恢复执行时，可以说它已被恢复。?阻塞 (Blocked) – 在线程等待一个事件时(例如输入/输出操作)，就称其处于阻塞状态。?死亡 (Dead) – 在 run() 方法已完成执行或其 stop() 方法被调用之后，线程就处于死亡状态。

59、JSP的常用指令

isErrorPage(是否能使用Exception对象)，isELIgnored(是否忽略表达式)

60、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()。

61、servlet的生命周期

web容器加载servlet，生命周期开始。通过调用servlet的init()方法进行servlet的初始化。通过调用service()方法实现，根据请求的不同调用不同的do***()方法。结束服务，web容器调用servlet的destroy()方法。

62、如何现实servlet的单线程模式

□□答:<%@ page isThreadSafe="false"%>

63、页面间对象传递的方法

request，session，application，cookie等

64、**JSP和Servlet有哪些相同点和不同点，他们之间的联系是什么？**

JSP是Servlet技术的扩展，本质上是Servlet的简易方式，更强调应用的外表表达。JSP编译后是"类servlet"。Servlet和JSP最主要的不同点在于，Servlet的应用逻辑是在Java文件中，并且完全从表示层中的HTML里分离开来。而JSP的情况是Java和HTML可以组合成一个扩展名为.jsp的文件。JSP侧重于视图，Servlet主要用于控制逻辑。

65、四种会话跟踪技术

会话作用域ServletsJSP 页面描述

page否是代表与一个页面相关的对象和属性。一个页面由一个编译好的 Java servlet 类(可以带有任何的 include 指令，但是没有 include 动作)表示。这既包括 servlet 又包括被编译成 servlet 的 JSP 页面

request是是代表与 Web 客户机发出的一个请求相关的对象和属性。一个请求可能跨越多个页面，涉及多个 Web 组件(由于 forward 指令和 include 动作的关系)

session是是代表与用于某个 Web 客户机的一个用户体验相关的对象和属性。一个 Web 会话可以也经常会跨越多个客户机请求

application是是代表与整个 Web 应用程序相关的对象和属性。这实质上是跨越整个 Web 应用程序，包括多个页面、请求和会话的一个全局作用域

66、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

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
67、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。
68、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;
}
```

69、简述逻辑操作(&,|,^)与条件操作(&&,||)的区别。
区别主要答两点：a.条件操作只能操作布尔型的,而逻辑操作不仅可以操作布尔型,而且可以操作数值型
b.逻辑操作不会产生短路
70、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？
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)
71、简述synchronized和java.util.concurrent.locks.Lock的异同 ？
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。

72、EJB的角色和三个对象
一个完整的基于EJB的分布式计算结构由六个角色组成，这六个角色可以由不同的开发商提供，每个角色所作的工作必须遵循Sun公司提供的EJB规范，以保证彼此之间的兼容性。这六个角色分别是EJB组件开发者(Enterprise Bean Provider)、应用组合者(Application Assembler)、部署者(Deployer)、EJB服务器提供者(EJB Server Provider)、EJB 容器提供者(EJB Container Provider)、系统管理员(System Administrator)
三个对象是Remote(Local)接口、Home(LocalHome)接口，Bean类
73、EJB容器提供的服务
主要提供声明周期管理、代码产生、持续性管理、安全、事务管理、锁和并发行管理等服务。
74、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.不能对静态属生存取，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。
75、remote接口和home接口主要作用
remote接口定义了业务方法，用于EJB客户端调用业务方法。
home接口是EJB工厂用于创建和移除查找EJB实例
76、bean 实例的生命周期
对于Stateless Session Bean、Entity Bean、Message Driven Bean一般存在缓冲池管理，而对于Entity Bean和Statefull Session Bean存在Cache管理，通常包含创建实例，设置上下文、创建EJB Object(create)、业务方法调用、remove等过程，对于存在缓冲池管理的Bean，在create之后实例并不从内存清除，而是采用缓冲池调度机制不断重用实例，而对于存在Cache管理的Bean则通过激活和去激活机制保持Bean的状态并限制内存中实例数量。
77、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，实例在激活和去激活状态之间迁移，激活机制是当客户端调用某个EJB实例业务方法时，如果对应EJB Object发现自己没有绑定对应的Bean实例则从其去激活Bean存储中(通过序列化机制存储实例)回复(激活)此实例。状态变迁前会调用对应的ejbActive和ejbPassivate方法。
78、EJB的几种类型
会话(Session)Bean，实体(Entity)Bean 消息驱动的(Message Driven)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
79、客服端调用EJB对象的几个基本步骤
设置JNDI服务工厂以及JNDI服务地址系统属性，查找Home接口，从Home接口调用Create方法创建Remote接口，通过Remote接口调用其业务方法。
80、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
81、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。
82、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.
83、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。
84、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//helloword.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。
85、在weblogic中发布ejb需涉及到哪些配置文件
不同类型的EJB涉及的配置文件不同，都涉及到的配置文件包括ejb-jar.xml,weblogic-ejb-jar.xmlCMP实体Bean一般还需要weblogic-cmp-rdbms-jar.xml
86、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。
87、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
88、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。
89、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AttibuteAccess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。
90、说说在weblogic中开发消息Bean时的persistent与non-persisten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。
91、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()

92、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 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，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。
93、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，程序运行中对应实现类会作为对应接口类型的实例被使用。
94、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。
/ /使用快速排序方法对a[0 :n- 1]排序
从a[0 :n- 1]中选择一个元素作为m i d d l e，该元素为支点
把余下的元素分割为两段left 和r i g h t，使得l e f t中的元素都小于等于支点，而right中的元素都大于等于支点
递归地使用快速排序方法对left 进行排序
递归地使用快速排序方法对right 进行排序
所得结果为l e f t + m i d d l e + r i g h t
95、请对以下在J2EE中常用的名词进行解释(或简单描述)
容器：充当中间件的角色
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。
EAI:企业应用集成。是一种概念，从而牵涉到好多技术。J2EE技术是一种很好的集成实现。
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。
96、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？
Java通过面向对象的方法进行异常处理，把各种不同的异常进行分类，并提供了良好的接口。在Java中，每个异常都是一个对象，它是Throwable类或其它子类的实例。当一个方法出现异常后便抛出一个异常对象，该对象中包含有异常信息，调用这个对象的方法可以捕获到这个异常并进行处理。Java的异常处理是通过5个关键词来实现的:try、catch、throw、throws和finally。一般情况下是用try来执行一段程序，如果出现异常，系统会抛出(throws)一个异常，这时候你可以通过它的类型来捕捉(catch)它，或最后(finally)由缺省处理器来处理。
用try来指定一块预防所有"异常"的程序。紧跟在try程序后面，应包含一个catch子句来指定你想要捕捉的"异常"的类型。
throw语句用来明确地抛出一个"异常"。
throws用来标明一个成员函数可能抛出的各种"异常"。
Finally为确保一段代码不管发生什么"异常"都被执行一段代码。
可以在一个成员函数调用的外面写一个try语句，在这个成员函数内部写另一个try语句保护其他代码。
每当遇到一个try语句，"异常"的框架就放到堆栈上面，直到所有的try语句都完成。如果下一级的try语句没有对某种"异常"进行处理，堆栈就会展开，直到遇到有处理这种"异常"的try语句。
97、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。必须只有一个类名与文件名相同。
98、MVC的各个部分都有那些技术来实现?如何实现?
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。
99、java中有几种方法可以实现一个线程？用什么关键字修饰同步方法?
stop()和suspend()方法为何不推荐使用？
有两种实现方法，分别是继承Thread类与实现Runnable接口
用synchronized关键字修饰同步方法
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。

100、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InputStreamReader OutputStreamWriter。在java.io包中还有许多其他的流，主要是为了提高性能和使用方便。

101、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。

自己实现堆载的数据结构时有可能会出现内存泄露.

102、java中实现多态的机制是什么？

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。

103、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？

对于GC来说，当程序员创建对象时，GC就开始监控这个对象的地址、大小以及使用情况。通常，GC采用有向图的方式记录和管理堆(heap)中的所有对象。通过这种方式确定哪些对象是"可达的"，哪些对象是"不可达的"。当GC确定一些对象为"不可达"时，GC就有责任回收这些内存空间。可以。程序员可以手动执行System.gc()，通知GC运行，但是Java语言规范并不保证GC一定会执行。

104、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

105、什么是java序列化，如何实现java序列化？

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，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。

106、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.

107、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。

108、在JAVA中，如何跳出当前的多重嵌套循环？

用break; return 方法。

109、List、Map、Set三个接口，存取元素时，各有什么特点？

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。

110、J2EE是什么？

J2EE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。

111、UML方面

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。

112、说出一些常用的类，包，接口，请各举5个

常用的类：BufferedReader BufferedWriter FileReader FileWirter String Integer

常用的包：java.lang java.awt java.io java.util java.sql

常用的接口：Remote List Map Document NodeList Runnable Serializable ActionListener
Collection,Connection, Cloneable, Comparable

113、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。

114、jsp有哪些动作?作用分别是什么?

答:JSP共有以下6种基本动作：

□□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:plugin：根据浏览器类型为Java插件生成OBJECT或EMBED标记

115、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)

□□

答：匿名的内部类是没有名字的内部类。可以继承抽象(SDK1.5没有限制)，但一个内部类可以作为一个接口，由另一个内部类实现.可以继承其他类或完成其他接口，在swing编程中常用此方式。

116、应用服务器与WEB SERVER的区别？

应用服务器：Weblogic、Tomcat、Jboss

WEB SERVER：IIS、Apache

117、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。

118、LINUX下线程，GDI类的解释。

LINUX实现的就是基于核心轻量级进程的"一对一"线程模型，一个线程实体对应一个核心轻量级进程，而线程之间的管理在核外函数库中实现。

GDI类为图像设备编程接口类库。

119、STRUTS的应用(如STRUTS架构)

Struts是采用Java Servlet/JavaServer Pages技术，开发Web应用程序的开放源码的framework。

采用Struts能开发出基于MVC(Model-View-Controller)设计模式的应用构架。Struts有如下的主要功能：

一.包含一个controller servlet，能将用户的请求发送到相应的Action对象。

二.JSP自由tag库，并且在controller servlet中提供关联支持，帮助开发员创建交互式表单应用。

三.提供了一系列实用对象：XML处理、通过Java reflection APIs自动处理JavaBeans属性、国际化的提示和消息。

120、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。
121、内部类可以引用他包含类的成员吗？有没有什么限制？
一个内部类对象可以访问创建它的外部类对象的内容
122、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Java中使用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消息进行操作。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。
JAVA编程题
1. 现在输入n个数字，以逗号，分开；然后可选择升或者降序排序；按提交键就在另一页面显示按什么排序，结果为，提供reset

```
import java.util.*;
public class bycomma{
public static String[] splitStringByComma(String source){
if(source==null||source.trim().equals(""))
return null;
StringTokenizer commaToker = new StringTokenizer(source,",");
String[] result = new String[commaToker.countTokens()];
int i=0;
while(commaToker.hasMoreTokens()){
result[i] = commaToker.nextToken();
i++;
}
return result;
}
```

```
public static void main(String args[]){
String[] s = splitStringByComma("5,8,7,4,3,9,1");
int[] ii = new int[s.length];
for(int i = 0;iii[i] =Integer.parseInt(s[i]);
}
Arrays.sort(ii);
//asc
for(int i=0;iSystem.out.println(ii[i]);
}
//desc
for(int i=(s.length-1);i>=0;i--){
System.out.println(ii[i]);
}
}
}
```

2. 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。

```
package test.format;
import java.text.NumberFormat;
import java.util.HashMap;
public class SimpleMoneyFormat {
public static final String EMPTY = "";
public static final String ZERO = "零";
public static final String ONE = "壹";
public static final String TWO = "贰";
public static final String THREE = "叁";
public static final String FOUR = "肆";
public static final String FIVE = "伍";
public static final String SIX = "陆";
public static final String SEVEN = "柒";
public static final String EIGHT = "捌";
public static final String NINE = "玖";
public static final String TEN = "拾";
public static final String HUNDRED = "佰";
public static final String THOUSAND = "仟";
public static final String TEN_THOUSAND = "万";
public static final String HUNDRED_MILLION = "亿";
public static final String YUAN = "元";
public static final String JIAO = "角";
public static final String FEN = "分";
public static final String DOT = ".";
private static SimpleMoneyFormat formatter = null;
private HashMap chineseNumberMap = new HashMap();
private HashMap chineseMoneyPattern = new HashMap();
private NumberFormat numberFormat = NumberFormat.getInstance();
private SimpleMoneyFormat() {
numberFormat.setMaximumFractionDigits(4);
numberFormat.setMinimumFractionDigits(2);
numberFormat.setGroupingUsed(false);
```

```
chineseNumberMap.put("0", ZERO);
chineseNumberMap.put("1", ONE);
chineseNumberMap.put("2", TWO);
chineseNumberMap.put("3", THREE);
chineseNumberMap.put("4", FOUR);
chineseNumberMap.put("5", FIVE);
chineseNumberMap.put("6", SIX);
chineseNumberMap.put("7", SEVEN);
chineseNumberMap.put("8", EIGHT);
chineseNumberMap.put("9", NINE);
chineseNumberMap.put(DOT, DOT);
chineseMoneyPattern.put("1", TEN);
chineseMoneyPattern.put("2", HUNDRED);
chineseMoneyPattern.put("3", THOUSAND);
chineseMoneyPattern.put("4", TEN_THOUSAND);
chineseMoneyPattern.put("5", TEN);
chineseMoneyPattern.put("6", HUNDRED);
chineseMoneyPattern.put("7", THOUSAND);
chineseMoneyPattern.put("8", HUNDRED_MILLION);
}
public static SimpleMoneyFormat getInstance() {
if (formatter == null)
formatter = new SimpleMoneyFormat();
return formatter;
}
public String format(String moneyStr) {
checkPrecision(moneyStr);
String result;
result = convertToChineseNumber(moneyStr);
result = addUnitsToChineseMoneyString(result);
return result;
}
public String format(double moneyDouble) {
return format(numberFormat.format(moneyDouble));
}
public String format(int moneyInt) {
return format(numberFormat.format(moneyInt));
}
public String format(long moneyLong) {
return format(numberFormat.format(moneyLong));
}
public String format(Number moneyNum) {
return format(numberFormat.format(moneyNum));
}
private String convertToChineseNumber(String moneyStr) {
String result;
StringBuffer cMoneyStringBuffer = new StringBuffer();
for (int i = 0; i < moneyStr.length(); i++) {
cMoneyStringBuffer.append(chineseNumberMap.get(moneyStr.substring(i, i + 1)));
}
```

```
//拾佰仟万亿等都是汉字里面才有的单位，加上它们
int indexOfDot = cMoneyStringBuffer.indexOf(DOT);
int moneyPatternCursor = 1;
for (int i = indexOfDot - 1; i > 0; i--) {
cMoneyStringBuffer.insert(i, chineseMoneyPattern.get(EMPTY + moneyPatternCursor));
moneyPatternCursor = moneyPatternCursor == 8 ? 1 : moneyPatternCursor + 1;
}

String fractionPart = cMoneyStringBuffer.substring(cMoneyStringBuffer.indexOf("."));
cMoneyStringBuffer.delete(cMoneyStringBuffer.indexOf("."), cMoneyStringBuffer.length());
while (cMoneyStringBuffer.indexOf("零拾") != -1) {
cMoneyStringBuffer.replace(cMoneyStringBuffer.indexOf("零拾"), cMoneyStringBuffer.indexOf("零拾")
+ 2, ZERO);
}
while (cMoneyStringBuffer.indexOf("零佰") != -1) {
cMoneyStringBuffer.replace(cMoneyStringBuffer.indexOf("零佰"), cMoneyStringBuffer.indexOf("零佰")
+ 2, ZERO);
}
while (cMoneyStringBuffer.indexOf("零仟") != -1) {
cMoneyStringBuffer.replace(cMoneyStringBuffer.indexOf("零仟"), cMoneyStringBuffer.indexOf("零仟")
+ 2, ZERO);
}
while (cMoneyStringBuffer.indexOf("零万") != -1) {
cMoneyStringBuffer.replace(cMoneyStringBuffer.indexOf("零万"), cMoneyStringBuffer.indexOf("零万")
+ 2, TEN_THOUSAND);
}
while (cMoneyStringBuffer.indexOf("零亿") != -1) {
cMoneyStringBuffer.replace(cMoneyStringBuffer.indexOf("零亿"), cMoneyStringBuffer.indexOf("零亿")
+ 2, HUNDRED_MILLION);
}
while (cMoneyStringBuffer.indexOf("零零") != -1) {
cMoneyStringBuffer.replace(cMoneyStringBuffer.indexOf("零零"), cMoneyStringBuffer.indexOf("零零")
+ 2, ZERO);
}
if (cMoneyStringBuffer.lastIndexOf(ZERO) == cMoneyStringBuffer.length() - 1)
cMoneyStringBuffer.delete(cMoneyStringBuffer.length() - 1, cMoneyStringBuffer.length());
cMoneyStringBuffer.append(fractionPart);
result = cMoneyStringBuffer.toString();
return result;
}
private String addUnitsToChineseMoneyString(String moneyStr) {
String result;
StringBuffer cMoneyStringBuffer = new StringBuffer(moneyStr);
int indexOfDot = cMoneyStringBuffer.indexOf(DOT);
cMoneyStringBuffer.replace(indexOfDot, indexOfDot + 1, YUAN);
////////////////////////////////////////////////////////////////////////////////
```

一、Java基础方面

1、作用域public,private,protected,以及不写时的区别

□□答:区别如下:

□□作用域	当前类	同一package	子孙类	其他package
□□public	√	√	√	√
□□protected	√	√	√	×
□□friendly	√	√	×	×
□□private	√	×	×	×

27、ArrayList和Vector的区别,HashMap和Hashtable的区别

□□答:就ArrayList与Vector主要从二方面来说:

□□1、同步性:Vector是线程安全的,也就是说是同步的,而ArrayList是线程序不安全的,不是同步的

□□2、数据增长:当需要增长时,Vector默认增长为原来一倍,而ArrayList却是原来的一半

□□就HashMap与HashTable主要从三方面来说:

□□1、历史原因:Hashtable是基于陈旧的Dictionary类的,HashMap是Java 1.2引进的Map接口的一个实现

□□

2、同步性:Hashtable是线程安全的,也就是说是同步的,而HashMap是线程序不安全的,不是同步的

□□3、值:只有HashMap可以让你将空值作为一个表的条目的key或value

30、float型float f=3.4是否正确?

□□答:不正确。精度不准确,应该用强制类型转换,如下所示:float f=(float)3.4; float f = 3.4f;

31、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

32、抽象类与接口?

□□

答:抽象类与接口都用于抽象,但是抽象类(JAVA中)可以有自己的部分实现,而接口则完全是一个标识(同时有多实现的功能)。

□□JAVA类实现序例化的方法是实现java.io.Serializable接口

□□Collection框架中实现比较要实现Comparable 接口和 Comparator 接口

33、STRING与STRINGBUFFER的区别。

□□

答:STRING的长度是不可变的,STRINGBUFFER的长度是可变的。如果你对字符串中的内容经常进行操作,特别是内容要修改时,那么使用StringBuffer,如果最后需要String,那么使用StringBuffer的toString()方法

34、谈谈final, finally, finalize的区别

答:final—
修饰符(关键字)如果一个类被声明为final,意味着它不能再派生出新的子类,不能作为父类被继承。因此一个类不能既被声明为
abstract的,又被声明为final的。将变量或方法声明为final,可以保证它们在使用中不被改变。如果是引用类型,说明它不能指向其他的对象了.但对象还是可以改变的。被声明为final的变量必须在声明时给定初值,而在以后的引用中只能读取,不可修改。被声明为final的方法也同样只能使用,不能重载

□□finally—在异常处理时提供 finally
块来执行任何清除操作。表示无论是否出现异常总是执行。如果抛出一个异常,那么相匹配的 catch 子句就会执行,然后控制就会进入 finally 块(如果有的话)
□□finalize—方法名。Java 技术允许使用 finalize()
方法在垃圾收集器将对象从内存中清除出去之前做必要的清理工作。这个方法是由垃圾收集器在确定这个对象没有被引用时对这个对象调用的。它是在 Object
类中定义的,因此所有的类都继承了它。子类覆盖 finalize()
方法以整理系统资源或者执行其他清理工作。finalize()
方法是在垃圾收集器删除对象之前对这个对象调用的.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。

三、Jsp方面

70、jsp有哪些内置对象?作用分别是什么?
□□答:JSP共有以下9种基本内置组件(可与ASP的6种内部组件相对应):
request 用户端请求,此请求会包含来自GET/POST请求的参数
□□response 网页传回用户端的回应
□□pageContext 网页的属性是在这里管理
□□session 与请求有关的会话期
□□application servlet正在执行的内容
□□out 用来传送回应的输出
□□config servlet的构架部件
□□page JSP网页本身
□□exception 针对错误网页,未捕捉的例外 □□

73、两种跳转方式分别是什么?有什么区别?
□□答:有两种,分别为:
□□<jsp:include page="included.jsp" flush="true">□□<jsp:forward page= "nextpage.jsp"/>
□□
前者页面不会转向include所指的页面,只是显示该页的结果,主页面还是原来的页面。执行完后还会回来,相当于函数调用。并且可以带参数.后者完全转向新页面,不会再回来。相当于go to 语句。

四、Servlet方面

76、JAVA SERVLET API中forward() 与redirect()的区别?
□□
答:前者仅是容器中控制权的转向,在客户端浏览器地址栏中不会显示出转向后的地址;后者则是完全的跳转,浏览器将会得到跳转的地址,并重新发送请求链接。这样,从浏览器的地址栏中可以看到跳转后的链接地址。所以,前者更加高效,在前者可以满足需要时,尽量使用forward()方法,并且,这样也有助于隐藏实际的链接。在有些情况下,比如,需要跳转到一个其它服务器上的资源,则必须使用sendRedirect()方法。

77、Servlet的基本架构

□□答:public class ServletName extends HttpServlet {
public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
}
public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
}
}

五、Jdbc、Jdo方面

87、Class.forName的作用?为什么要用?

□□答:调用该访问返回一个以字符串指定类名的类的对象。

六、XML方面

91、xml有哪些解析技术?区别是什么?

□□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

定义方式:DTD(文档类型定义)和Schema。

□□STAX:Streaming API for XML (StAX)

92、你在项目中用到了xml技术的哪些方面?如何实现的?

□□

答:用到了数据存贮,信息配置两方面。在做数据交换平台时,将不同数据源的数据组装成XML文件,然后将XML文件压缩打包加密后通过网络传送给接收者,接收解密与解压缩后再同XML文件中还原相关信息进行处理。在做软件配置时,利用XML可以很方便的进行,软件的各种配置参数都存贮在XML文件中。

七、EJB方面

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

□□

答:规范内容包括Bean提供者,应用程序装配者,EJB容器,EJB配置工具,EJB服务提供者,系统管理员。这里面,EJB容器是EJB之所以能够运行的核心。EJB容器管理着EJB的创建,撤消,激活,去活,与数据库的连接等等重要的核心工作。JSP,Servlet,EJB,JNDI,JDBC,JMS.....

八、应用服务器方面

□□

九、J2EE及MVC方面

十三、代码与编程题

136、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
```

137、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
```

□□再一个例题:

```
public class OuterClass {
  private double d1 = 1.0;
  //insert code here }
```

You need to insert an inner class declaration at line 3. Which two inner class declarations are valid?(Choose two.)

```
A. class InnerOne{
     public static double methoda() {return d1;}
   }
B. public class InnerOne{
     static double methoda() {return d1;}
```

```
  }
C. private class InnerOne{
     double methoda() {return d1;}
  }
D. static class InnerOne{
     protected double methoda() {return d1;}
  }
E. abstract class InnerOne{
     public abstract double methoda();
  }
```

□□说明如下：

□□1.静态内部类可以有静态成员，而非静态内部类则不能有静态成员。故 A、B 错

□□2.静态内部类的非静态成员可以访问外部类的静态变量，而不可访问外部类的非静态变量；故 D 错

□□3.非静态内部类的非静态成员可以访问外部类的非静态变量。故 C 正确

□□4.答案为C、E

138、Java 的通信编程，编程题(或问答)，用JAVA SOCKET编程，读服务器几个字符，再写入本地显示？

□□答：Server端程序：

```
package test;
import java.net.*;
import java.io.*;
public class Server{
  private ServerSocket ss;
  private Socket socket;
  private BufferedReader in;
  private PrintWriter out;
 public Server() {
  try {
   ss=new ServerSocket(10000);
   while(true) {
    socket = ss.accept();
    String RemoteIP = socket.getInetAddress().getHostAddress();
    String RemotePort = ":"+socket.getLocalPort();
    System.out.println("A client come in!IP:"+RemoteIP+RemotePort);
    in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
    String line = in.readLine();
    System.out.println("Cleint send is :" + line);
    out = new PrintWriter(socket.getOutputStream(),true);
    out.println("Your Message Received!");
    out.close();
    in.close();
    socket.close();
   }
  }catch (IOException e) {
   out.println("wrong");
  }
 }
```

```
 public static void main(String[] args) {
  new Server();
 }
}
```

□□Client端程序:

```
package test;
import java.io.*;
import java.net.*;

public class Client {
 Socket socket;
 BufferedReader in;
 PrintWriter out;
 public Client() {
  try {
   System.out.println("Try to Connect to 127.0.0.1:10000");
   socket = new Socket("127.0.0.1",10000);
   System.out.println("The Server Connected!");
   System.out.println("Please enter some Character:");
   BufferedReader line = new BufferedReader(new InputStreamReader(System.in));
   out = new PrintWriter(socket.getOutputStream(),true);
   out.println(line.readLine());
   in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
   System.out.println(in.readLine());
   out.close();
   in.close();
   socket.close();
  }catch(IOException e) {
   out.println("Wrong");
  }
 }
 public static void main(String[] args) {
  new Client();
 }
}
```

139、用JAVA实现一种排序，JAVA类实现序列化的方法(二种)？
如在COLLECTION框架中，实现比较要实现什么样的接口？

□答: java类实现序列化需要实现java.io.Serializable接口。
在collection中实现比较要实现java.lang.Comparable接口或java.util.Comparator接口。
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();
}
}
```

140、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：

```
package test;
class  SplitString {
 String SplitStr;
 int SplitByte;
 public SplitString(String str,int bytes) {
  SplitStr=str;
  SplitByte=bytes;
  System.out.println("The String is:'"+SplitStr+"';SplitBytes="+SplitByte);
 }
 public void SplitIt() {
  int loopCount;
```

```
 loopCount=(SplitStr.length()%SplitByte==0)?(SplitStr.length()/SplitByte):(SplitStr.length()/SplitByte+1);
 System.out.println("Will Split into "+loopCount);
 for (int i=1;i<=loopCount ;i++ ) {
  if (i==loopCount){
   System.out.println(SplitStr.substring((i-1)*SplitByte,SplitStr.length()));
  } else {
   System.out.println(SplitStr.substring((i-1)*SplitByte,(i*SplitByte)));
  }
 }
}
public static void main(String[] args) {
 SplitString ss = new SplitString("test中dd文dsaf中男大3443n中国43中国人0ewldfls=103",4);
 ss.SplitIt();
}
}
```

141、JAVA多线程编程。

用JAVA写一个多线程程序，如写四个线程，二个加1，二个对一个变量减一，输出。

□□希望大家补上，谢谢

142、可能会让你写一段Jdbc连Oracle的程序,并实现数据查询.

答:程序如下:

```
package hello.ant;
import java.sql.*;
public class  jdbc {
 String dbUrl="jdbc:oracle:thin:@127.0.0.1:1521:orcl";
 String theUser="admin";
 String thePw="manager";
 Connection c=null;
 Statement conn;
 ResultSet rs=null;
 public jdbc() {
  try{
   Class.forName("oracle.jdbc.driver.OracleDriver").newInstance();
      c = DriverManager.getConnection(dbUrl,theUser,thePw);
   conn=c.createStatement();
  }catch(Exception e){
   e.printStackTrace();
  }
 }
 public boolean executeUpdate(String sql) {
  try {
   conn.executeUpdate(sql);
   return true;
  } catch (SQLException e) {
   e.printStackTrace();
   return false;
  }
 }
 public ResultSet executeQuery(String sql) {
  rs=null;
```

```
 try {
  rs=conn.executeQuery(sql);
 } catch (SQLException e) {
  e.printStackTrace();
 }
 return rs;
}
public void close() {
 try {
  conn.close();
  c.close();
 } catch (Exception e) {
  e.printStackTrace();
 }
}
public static void main(String[] args) {
 ResultSet rs;
 jdbc conn = new jdbc();
 rs=conn.executeQuery("select * from test");
 try {
  while (rs.next()) {
   System.out.println(rs.getString("id"));
   System.out.println(rs.getString("name"));
  }
 }catch(Exception e) {
  e.printStackTrace();
 }
}
}
```

143、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。

□□答:一种分页方法:

```
<%
 int i=1;
 int numPages=14;
 String pages = request.getParameter("page") ;
 int currentPage = 1;
 currentPage=(pages==null)?(1):{Integer.parseInt(pages)}
 sql = "select count(*) from tables";
 ResultSet rs = DBLink.executeQuery(sql) ;
 while(rs.next()) i = rs.getInt(1) ;
 int intPageCount=1;
 intPageCount=(i%numPages==0)?(i/numPages):(i/numPages+1);
 int nextPage ;
 int upPage;
 nextPage = currentPage+1;
 if (nextPage>=intPageCount) nextPage=intPageCount;
 upPage = currentPage-1;
 if (upPage<=1) upPage=1;
 rs.close();
```

```
  sql="select * from tables";
  rs=DBLink.executeQuery(sql);
  i=0;
  while((i<numPages*(currentPage-1))&&rs.next()){i++;}
%>
```

□□输出内容:

```
//输出翻页连接
合计:<%=currentPage%>/<%=intPageCount%><a href="List.jsp?page=1">首页</a><a
href="List.jsp?page=<%=upPage%>">上页</a>
<%
  for(int j=1;j<=intPageCount;j++){
  if(currentPage!=j){
%>
  <a href="list.jsp?page=<%=j%>">[<%=j%>]</a>
<%
  }else{
   out.println(j);
  }
  }
%>
<a href="List.jsp?page=<%=nextPage%>">下页</a><a
href="List.jsp?page=<%=intPageCount%>">末页</a>
```

144、用jdom解析xml文件时如何解决中文问题?如何解析?

□□答:看如下代码,用编码方式加以解决:

```
package test;
import java.io.*;
public class DOMTest {
 private String inFile = "c:\\people.xml";
 private String outFile = "c:\\people.xml";
 public static void main(String args[]) {
  new DOMTest(); }

 public DOMTest() {
  try {
   javax.xml.parsers.DocumentBuilder builder =
    javax.xml.parsers.DocumentBuilderFactory.newInstance().newDocumentBuilder();
      org.w3c.dom.Document doc = builder.newDocument();
      org.w3c.dom.Element root = doc.createElement("老师");
      org.w3c.dom.Element wang = doc.createElement("王");
    org.w3c.dom.Element liu = doc.createElement("刘");
      wang.appendChild(doc.createTextNode("我是王老师"));
      root.appendChild(wang);
      doc.appendChild(root);
      javax.xml.transform.Transformer transformer =
       javax.xml.transform.TransformerFactory.newInstance().newTransformer();
      transformer.setOutputProperty(javax.xml.transform.OutputKeys.ENCODING, "gb2312");
      transformer.setOutputProperty(javax.xml.transform.OutputKeys.INDENT, "yes");
      transformer.transform(new javax.xml.transform.dom.DOMSource(doc),
```

```
      new javax.xml.transform.stream.StreamResult(outFile));
  } catch (Exception e) {
   System.out.println (e.getMessage());
  }
 }
}
```

145、编程用JAVA解析XML的方式.

□□答:用SAX方式解析XML，XML文件如下:

```
<?xml version="1.0" encoding="gb2312"?>
<person>
  <name>王小明</name>
  <college>信息学院</college>
  <telephone>6258113</telephone>
  <notes>男,1955年生,博士，95年调入海南大学</notes>
</person>
```

□□事件回调类SAXHandler.java

```
import java.io.*;
import java.util.Hashtable;
import org.xml.sax.*;
public class SAXHandler extends HandlerBase {
  private Hashtable table = new Hashtable();
  private String currentElement = null;
  private String currentValue = null;
  public void setTable(Hashtable table) {
    this.table = table;
  }
  public Hashtable getTable() {
    return table;
 }
  public void startElement(String tag, AttributeList attrs) throws SAXException {
    currentElement = tag;
  }
  public void characters(char[] ch, int start, int length) throws SAXException {
    currentValue = new String(ch, start, length);
  }
  public void endElement(String name) throws SAXException {
    if (currentElement.equals(name)) {
      table.put(currentElement, currentValue);
    }
  }
}
```

JSP内容显示源码，SaxXml.jsp:

```
<HTML>
<HEAD>
<TITLE>剖析XML文件people.xml</TITLE>
</HEAD>
<BODY>
<%@ page errorPage="ErrPage.jsp" contentType="text/html;charset=GB2312" %>
<%@ page import="java.io.*" %>
```

```
<%@ page import="java.util.Hashtable" %>
<%@ page import="org.w3c.dom.*" %>
<%@ page import="org.xml.sax.*" %>
<%@ page import="javax.xml.parsers.SAXParserFactory" %>
<%@ page import="javax.xml.parsers.SAXParser" %>
<%@ page import="SAXHandler" %>
<%
File file = new File("c:\\people.xml");
FileReader reader = new FileReader(file);
Parser parser;
SAXParserFactory spf = SAXParserFactory.newInstance();
SAXParser sp = spf.newSAXParser();
SAXHandler handler = new SAXHandler();
sp.parse(new InputSource(reader), handler);
Hashtable hashTable = handler.getTable();
out.println("<TABLE BORDER=2><CAPTION>教师信息表</CAPTION>");
out.println("<TR><TD>姓名</TD>" + "<TD>" +
  (String)hashTable.get(new String("name")) + "</TD></TR>");
out.println("<TR><TD>学院</TD>" + "<TD>" +
  (String)hashTable.get(new String("college"))+"</TD></TR>");
out.println("<TR><TD>电话</TD>" + "<TD>" +
  (String)hashTable.get(new String("telephone")) + "</TD></TR>");
out.println("<TR><TD>备注</TD>" + "<TD>" +
  (String)hashTable.get(new String("notes")) + "</TD></TR>");
out.println("</TABLE>");
%>
</BODY>
</HTML>
```

146、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 jaax.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 javx.ejb.SessionContext;
public class AddBean Implements SessionBean
{
  //some method declare
}
```

147、如何校验数字型?

```
var re=/^\d{1,8}$|\.\d{1,2}$/;
var str=document.form1.all(i).value;
var r=str.match(re);
if (r==null) {
  sign=-4;
  break;
}else{
  document.form1.all(i).value=parseFloat(str);
}
```

148、将一个键盘输入的数字转化成中文输出(例如:输入1234567, 输出:一百二拾三万四千五百六拾七),请用java语言编一段程序实现!

```
public class Reader {
  private String strNum;
  private String strNumChFormat;
  private String strNumTemp;
  private int intNumLen;
  private String strBegin;
  public Reader(String strNum) {
    this.strNum = strNum;
  }
  public boolean check(String strNum) {
    boolean valid = false;

    if (strNum.substring(0,1).equals("0")){
     this.strNum = strNum.substring(1);
    }
    try {
      new Double(strNum);
      valid = true;
    } catch (NumberFormatException ex) {
      System.out.println("Bad number format!");
    }
    return valid;
  }
  public void init() {
    strNumChFormat = "";
    intNumLen = strNum.length();
    strNumTemp = strNum;
    strNumTemp = strNumTemp.replace('1', '一');
    strNumTemp = strNumTemp.replace('2', '二');
    strNumTemp = strNumTemp.replace('3', '三');
    strNumTemp = strNumTemp.replace('4', '四');
    strNumTemp = strNumTemp.replace('5', '五');
```

```
  strNumTemp = strNumTemp.replace('6', '六');
  strNumTemp = strNumTemp.replace('7', '七');
  strNumTemp = strNumTemp.replace('8', '八');
  strNumTemp = strNumTemp.replace('9', '九');
  strNumTemp = strNumTemp.replace('0', '零');
  strNumTemp = strNumTemp.replace('.', '点');
  strBegin = strNumTemp.substring(0, 1);
}

public String readNum() {
  if (check(strNum)) {
    init();
    try {
      for (int i = 1, j = 1, k = 1; i < intNumLen; i++) {
        if (strNumTemp.charAt(intNumLen - 1) == '零' && i == 1) {
          strNumChFormat = "位";
        }
        else if (strNumTemp.charAt(intNumLen - i) == '零' && j == 1) {
          strNumChFormat = "位" + strNumChFormat;
        }
        else if (strNumTemp.charAt(intNumLen - i) == '点') {
          j = 1;
          k = 1;
          strNumChFormat = strNumTemp.charAt(intNumLen - i) + strNumChFormat;
          continue;
        }
        else {
          strNumChFormat = strNumTemp.charAt(intNumLen - i) + strNumChFormat;
        }
        if (strNumTemp.charAt(intNumLen - i - 1) != '位' &&
            strNumTemp.charAt(intNumLen - i - 1) != '零') {
          if (j == 1 && i < intNumLen) {
            strNumChFormat = '拾' + strNumChFormat;
          }
          else if (j == 2 && i < intNumLen) {
            strNumChFormat = '百' + strNumChFormat;
          }
          else if (j == 3 && i < intNumLen) {
            strNumChFormat = '千' + strNumChFormat;
          }
        }
        if (j == 4 && i < intNumLen) {
          j = 0;
        }
        if (k == 4 && i < intNumLen) {
          strNumChFormat = '万' + strNumChFormat;
        }
        else if (k == 8 && i < intNumLen) {
```

```
          k = 0;
          strNumChFormat = '亿' + strNumChFormat;
        }
        j++;
        k++;
      }
      while (strNumChFormat.indexOf("位") != -1) {
        strNumChFormat = strNumChFormat.replaceAll("位", " ");
      }
      if (strNumChFormat.substring(0, 2) == "一拾") {
        strNumChFormat = strNumChFormat.substring(1, strNumChFormat.length());
      }
      if (strNumChFormat.indexOf("点") >= 0) {
        String rebegin = strNumChFormat.substring(0,
            strNumChFormat.indexOf("点"));
        String relast = strNumChFormat.substring(strNumChFormat.indexOf("点"),
            strNumChFormat.length());
        for (int i = 1; i <= relast.length(); i++) {
          relast = relast.replaceAll("拾", "");
          relast = relast.replaceAll("百", "");
          relast = relast.replaceAll("千", "");
          relast = relast.replaceAll("万", "");
          relast = relast.replaceAll("亿", "");
        }
        strNumChFormat = rebegin + relast;
      }
    }
    catch (ArrayIndexOutOfBoundsException ex) {
      ex.printStackTrace();
    }
    catch (Exception ex) {
      ex.printStackTrace();
    }
    int off = strNumChFormat.indexOf("点");
    strNumChFormat = strBegin + strNumChFormat.substring(0);
  }
  else {
    strNumChFormat = "";
  }
  return strNumChFormat;
}
public static void main(String args[]) {
  try {
    String number = args[0].toString();
    System.out.println("The number is: " + number);
    Reader reader = new Reader(number);
    System.out.println("Output String: " + reader.readNum());
  }
  catch (Exception ex) {
```

```
    System.out.println("Please input like that: javac Reader <number>");
   }
  }
}
```

149、JAVA代码查错

□□(1) 下面这段代码有什么错误？

```
abstract class Name {
  private String name;
  public abstract boolean isStupidName(String name) {}
}
```

□□答: abstract method必须以分号结尾，且不带花括号。

□□(2) 下面这段代码有错吗?

```
public class Something {
  void doSomething () {
    private String s = "";
    int l = s.length();
  }
}
```

□□答: 有错。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) 下面这段代码有错吗?

```
abstract class Something {
  private abstract String doSomething ();
}
```

□□答:

错。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)下面这段代码有什么错误？

```
public class Something {
  public int addOne(final int x) {
    return ++x;
  }
}
```

答: 这个错误比较明显。int x被修饰成final，意味着x不能在addOne method中被修改。

□(5) 下面这段代码有错吗？

```
public class Something {
  public static void main(String[] args) {
    Other o = new Other();
    new Something().addOne(o);
  }

  public void addOne(final Other o) {
    o.i++;
  }
}

class Other {
```

```
  public int i;
}
```

□□答案: 正确。在addOne method中，参数o被修饰成final。如果在addOne method里我们修改了o的reference (比如: o = new Other();)，那么如同上例这题也是错的。但这里修改的是o的member vairable (成员变量)，而o的reference并没有改变。

□□(6) 下面这段代码有错吗？

```
class Something {
  int i;
  public void doSomething() {
    System.out.println("i = " + i);
  }
}
```

□□答案: 正确。输出的是"i = 0"。int i属於instant variable (实例变量，或叫成员变量)。instant variable有default value。int的default value是0。

□□(7) 和上面一题只有一个地方不同，就是多了一个final。下面这段代码有错吗？

```
class Something {
  final int i;
  public void doSomething() {
    System.out.println("i = " + i);
  }
}
```

□□答案: 错。final int i是个final的instant variable (实例变量，或叫成员变量)。final的instant variable没有default value，必须在constructor (构造器)结束之前被赋予一个明确的值。可以修改为"final int i = 0;"。

□□(8) 下面这段代码看上去很完美，错在哪里呢？

```
public class Something {
  public static void main(String[] args) {
    Something s = new Something();
    System.out.println("s.doSomething() returns " + doSomething());
  }
  public String doSomething() {
    return "Do something ...";
  }
}
```

□□答案: 错。看上去在main里call doSomething没有什么问题，毕竟两个methods都在同一个class里。但仔细看，main是static的。static method不能直接call non-static methods。可改成"System.out.println("s.doSomething() returns " + s.doSomething());"。同理，static method不能访问non-static instant variable。

□□(9) 此处Something类的文件名叫OtherThing.java

```
class Something {
  private static void main(String[] something_to_do) {
    System.out.println("Do something ...");
  }
}
```

□□答案: 从没有人说过Java的Class名字必须和其文件名相同。但public class的名字必须和文件名相同，错误，main一定是public的。

□□(10) 下面这段代码有错吗？

```
interface A {
  int x = 0;
}
class B {
  int x =1;
}
class C extends B implements A {
  public void pX(){
    System.out.println(x);
  }
  public static void main(String[] args) {
    new C().pX();
  }
}
```

□

答案:错误。在编译时会发生错误(错误描述不同的JVM有不同的信息，意思就是未明确的x调用，两个x都匹配(就象在同时import java.util和java.sql两个包时直接声明Date一样)。对于父类的变量,可以用super.x来明确，而接口的属性默认隐含为 public static final.所以可以通过A.x来明确。

□□(11) 这个错误不容易发现

```
interface Playable {
  void play();
}
interface Bounceable {
  void play();
}

interface Rollable extends Playable, Bounceable {
  Ball ball = new Ball("PingPang");
}
class Ball implements Rollable {
  private String name;
  public String getName() {
    return name;
  }
  public Ball(String name) {
    this.name = name;  }
  public void play() {
    ball = new Ball("Football");
    System.out.println(ball.getName());
  }
}
```

答案: 错。"interface Rollable extends Playable, Bounceable"没有问题。interface可继承多个interfaces，所以这里没错。问题出在interface Rollable里的"Ball ball = new Ball("PingPang");"。任何在interface里声明的interface variable(接口变量，也可称成员变量)，默认为public static final。也就是说"Ball ball = new Ball("PingPang");"实际上是"public static final Ball ball = new

Ball("PingPang");"。在Ball类的Play()方法中，"ball = new Ball("Football");"改变了ball的reference，而这里的ball来自Rollable interface，Rollable interface里的ball是public static final的，final的object是不能被改变reference的。因此编译器将在"ball = new Ball("Football");"这里显示有错。

150、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。

□□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();
   }
   }
  }}
```

54. 写一个冒泡排序的程序。

```
                public class Sort
{           public static void main(String[] args)
            { int[] array = {2,3,4,1,43,432,1,344,234,2,3,43,32,434,3432,43,2432,432,4,43};
                int num = 0;
                  /* 冒泡排续 */
                  for(int i = 0;i < array.length;i++)
```

```
{ /* 每下底一个元素，则调换的次数减一。注意:j<array.length-i-
1,不减一则数组下标越界。*/
                for(int j = 0;j<array.length-i-1;j++)
                { int temp = 0;
                    /* 两两比较若前面的大于后面的则进行调换。*/
                    if(array[j] > array[j+1])
                    { temp = array[j];
                        array[j] = array[j+1];
                        array[j+1] = temp;
                    }
                }
            }
            /* 输出排序后的数组 */
            System.out.println("***************已排序的数组********");
            for(int i = 0;i < array.length;i++)
            { System.out.print(array[i]+"    ");
                num++;
                if(num == 5)//每行输出5个。
                { System.out.println();
                    num = 0;
                }
            }
        }}
```

5、串行化的注意事项以及如何实现串行化

答:如果有循环引用是不可以串行化的。对象输出流的WriteObject方法和 对象输入流的ReadObect方法

7、

线程的同步、如何实现线程的同步答:当两个或多个线程同时访问同一个变量，并且以个线程需要修改这个变量。就要用到线程同步。在Java 中，同步是通过 synchronized 关键字来定义的。

诺是想同步化某程序段，可以使用 synchronized(object){}方法，其中{}内的程序语句被同步化。

9、socket通信（tcp/udp区别及JAVA的实现方式）TCP——

传输控制协议，具有极高的可靠性，保证数据包按照顺序准确到达，但其也有着很高的额外负担。UDP——

使用者数据元协议，并不能保证数据包会被成功的送达，也不保证数据包到达的顺序，但其传输速度很快。大多数我们会使用TCP，偶尔才会动用UDP，如声音讯号，即使少量遗失，也无关紧要。

10、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在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

11、JDBC调用数据库的基本步骤

导入必要的类，装入JDBC驱动程序，识别数据源，分配一个Connection对象，分配一个Statement对象，使用Statement执行一个查询，从返回的ResultSet对象中检索数据，关闭ResultSet，关闭Statement对象，关闭Connection对象

12、解析XML文件的几种方式和区别
答:Dom解析 在内存中创建一个DOM树，能随机访问文件内容，也可以修改原文件内容SAX解析线性解析，不能随机访问，也无法修改原文件Dom解析要先用SAX解析创建DOM树
13、JAVA的四种基本权限的定义public private protected 默认
14、JAVA的国际化 答:Java 中提供了诺干国际化明感类，来实现国际化的。例如:dateformat timezone 等等。
16、谈谈你对swing mvc模式的理解？
答:Swing号称是完全按照MVC的思路来进行设计的。在设计开始前，Swing的希望能够达到的目标就包括: 模型驱动(Model-Driven)的编程方式。提供一套单一的API，但是能够支持多种视感(look-and-feel)，为用户提供不同的界面。严格的说，Swing中的MVC实际上是MVC的一个变体:M-VC。Swing中只显示的定义了Model接口，而在一个UI对象中集成了视图和控制器的部分机制。View和Control比较松散的交叉组合在一起，而更多的控制逻辑是在事件监听者部分引入的。
但是，这并没有妨碍在Swing中体现MVC的精髓。事实上，在Swing的开发初期，Swing确实是按照标准的MVC模式来设计的，但是很快的问题就出现了:View和Controller实际上是紧密耦合的，很难作出一个能够适应不同View的一般化的Controller来，而且，一般也没有很大的必要。
18、Java程序怎么优化？答:提高JAVA的性能，一般考虑如下的四个主要方面:
程序设计的方法和模式 (2) JAVA布署的环境。(3) JAVA应用程序的实现 (4) 硬件和操作系统
为了提高JAVA程序的性能，需要遵循如下的六个步骤。a) 明确对性能的具体要求 b)
了解当前程序的性能 c) 找到程序的性能瓶颈 d) 采取适当的措施来提高性能 e)
只进行某一方面的修改来提高性能 f) 返回到步骤c,继续作类似的工作，一直达到要求的性能为止。
动态查询如何实现？表的结构变化后，如果不需要修改程序，如何设计和实现查询？答:讲查询封装进存储过程中，通过调用存储过程实现动态调用;表结构发生变化后修改相应的存储过程即可再不修改程序的情况下实现查询。
2、
如何优化数据库，如何提高数据库的性能？答:优化数据库主要是优化查询语句，通过高性能的查询语句提高数据库的性能。
3、
设计数据库应注意那些问题答:首先应尽量满足三范式的要求，在一定程度上打破3范式的要求以提高数据库的性能。
4、表与表之间的关联关系答:分为3种:一对一、一对多、多对多。
5、
主键和外键的区别答:主键在本表中是唯一的、不可唯空的，外键可以重复可以唯空;外键和另一张表的主键关联，不能创建对应表中不存在的外键。
3. 数据类型之间的转换 如何将数值型字符转换为数字(Integer，Double) 如何将数字转换为字符
如何去小数点前两位，并四舍五入。
4. 日期和时间 如何取得年月日，小时分秒Date dat=new
Date();dat.getYear();dat.getMonth();dat.getDay();dat.getHours();...
如何取得从1970年到现在的毫秒数long
now=dat.getTime();如何获取某个日期是当月的最后一天如何格式化日期
DateFormate df=DateFormate.getInstance();df.Format(dat);
For if,else switch
6. 文件和目录(I/O)操作 如何列出某个目录下的所有文件 如何列出某个目录下的所有子目录
判断一个文件或目录是否存在 如何读写文件

4. 描述Cookie和Session的作用，区别和各自的应用范围，Session工作原理。
Cookie是在客户端开辟的一块可长期存储用户信息的地方；
Session是在服务器内存中开辟的一块存储用户信息的地方；当客户第一次访问服务器时，服务器在内存创建对应的Session给访问客户，
当客户离开或时间过期时；服务器自动销毁Session； Session是由容器管理的.
8. 描述一下你最常用的编程风格。
(1)
类名首字母应该大写。字段、方法以及对象（句柄）的首字母应小写。对于所有标识符，其中包含的所有单词都应紧靠在一起，而且大写中间单词的首字母。Java包（Package）属于一种特殊情况：它们全都是小写字母，即便中间的单词亦是如此。对于域名扩展名称，如com，org，net或者edu等，全部都应小写（这也是Java 1.1和Java 1.2的区别之一）。
(2)
为了常规用途而创建一个类时，请采取“经典形式”，并包含对下述元素的定义：equals()hashCode()toString()clone()（implement Cloneable）implement Serializable
(3)
对于自己创建的每一个类，都考虑置入一个main()，其中包含了用于测试那个类的代码。为使用一个项目中的类，我们没必要删除测试代码。若进行了任何形式的改动，可方便地返回测试。这些代码也可作为如何使用类的一个示例使用。
(4)
应将方法设计成简要的、功能性单元，用它描述和实现一个不连续的类接口部分。理想情况下，方法应简明扼要。若长度很大，可考虑通过某种方式将其分割成较短的几个方法。这样做也便于类内代码的重复使用（有些时候，方法必须非常大，但它们仍应只做同样的一件事情）。
(5)
设计一个类时，请设身处地为客户程序员考虑一下（类的使用方法应该是非常明确的）。然后，再设身处地为管理代码的人考虑一下（预计有可能进行哪些形式的修改，想想用什么方法可把它们变得更简单）。
(6)
使类尽可能短小精悍，而且只解决一个特定的问题。下面是对类设计的一些建议：一个复杂的开关语句：考虑采用“多形”机制数量众多的方法涉及到类型差别极大的操作：考虑用几个类来分别实现许多成员变量在特征上有很大的差别：考虑使用几个类
(7) 让一切东西都尽可能地“私有”——
private。可使库的某一部分“公共化”（一个方法、类或者一个字段等等），就永远不能把它拿出。若强行拿出，就可能破坏其他人现有的代码，使他们不得不重新编写和设计。若只公布自己必须公布的，就可放心大胆地改变其他任何东西。在多线程环境中，隐私是特别重要的一个因素——
只有private字段才能在非同步使用的情况下受到保护。
(8)
谨惕“巨大对象综合症”。对一些习惯于顺序编程思维、且初涉OOP领域的新手，往往喜欢先写一个顺序执行的程序，再把它嵌入一个或两个巨大的对象里。根据编程原理，对象表达的应该是应用程序的概念，而非应用程序本身。
(9) 若不得已进行一些不太雅观的编程，至少应该把那些代码置于一个类的内部。
(10)
任何时候只要发现类与类之间结合得非常紧密，就需要考虑是否采用内部类，从而改善编码及维护工作（参见第14章14.1.2小节的“用内部类改进代码”）。
(11) 尽可能细致地加上注释，并用javadoc注释文档语法生成自己的程序文档。

(12)
避免使用“魔术数字”，这些数字很难与代码很好地配合。如以后需要修改它，无疑会成为一场噩梦，因为根本不知道“100”到底是指“数组大小”还是“其他全然不同的东西”。所以，我们应创建一个常数，并为其使用具有说服力的描述性名称，并在整个程序中都采用常数标识符。这样可使程序更易理解以及更易维护。
(13) 涉及构建器和异常的时候，通常希望重新丢弃在构建器中捕获的任何异常——如果它造成了那个对象的创建失败。这样一来，调用者就不会以为那个对象已正确地创建，从而盲目地继续。
(14)
当客户程序员用完对象以后，若你的类要求进行任何清除工作，可考虑将清除代码置于一个良好定义的方法里，采用类似于cleanup()这样的名字，明确表明自己的用途。除此以外，可在类内放置一个boolean（布尔）标记，指出对象是否已被清除。在类的finalize()方法里，请确定对象已被清除，并已丢弃了从RuntimeException继承的一个类（如果还没有的话），从而指出一个编程错误。在采取象这样的方案之前，请确定finalize()能够在自己的系统中工作（可能需要调用System.runFinalizersOnExit(true)，从而确保这一行为）。
(15)
在一个特定的作用域内，若一个对象必须清除（非由垃圾收集机制处理），请采用下述方法：初始化对象；若成功，则立即进入一个含有finally从句的try块，开始清除工作。
(16)
若在初始化过程中需要覆盖（取消）finalize()，请记住调用super.finalize()（若Object属于我们的直接超类，则无此必要）。在对finalize()进行覆盖的过程中，对super.finalize()的调用应属于最后一个行动，而不应是第一个行动，这样可确保在需要基础类组件的时候它们依然有效。
(17)
创建大小固定的对象集合时，请将它们传输至一个数组（若准备从一个方法里返回这个集合，更应如此操作）。这样一来，我们就可享受到数组在编译期进行类型检查的好处。此外，为使用它们，数组的接收者也许并不需要将对象“造型”到数组里。
(18)
尽量使用interfaces，不要使用abstract类。若已知某样东西准备成为一个基础类，那么第一个选择应是将其变成一个interface（接口）。只有在不得不使用方法定义或者成员变量的时候，才需要将其变成一个abstract（抽象）类。接口主要描述了客户希望做什么事情，而一个类则致力于（或允许）具体的实施细节。
(19)
在构建器内部，只进行那些将对象设为正确状态所需的工作。尽可能地避免调用其他方法，因为那些方法可能被其他人覆盖或取消，从而在构建过程中产生不可预知的结果（参见第7章的详细说明）.
(20) 对象不应只是简单地容纳一些数据；它们的行为也应得到良好的定义。
(21)
在现成类的基础上创建新类时，请首先选择“新建”或“创作”。只有自己的设计要求必须继承时，才应考虑这方面的问题。若在本来允许新建的场合使用了继承，则整个设计会变得没有必要地复杂。
(22)
用继承及方法覆盖来表示行为间的差异，而用字段表示状态间的区别。一个非常极端的例子是通过对不同类的继承来表示颜色，这是绝对应该避免的：应直接使用一个“颜色”字段。
(23)
为避免编程时遇到麻烦，请保证在自己类路径指到的任何地方，每个名字都仅对应一个类。否则，编

译器可能先找到同名的另一个类，并报告出错消息。若怀疑自己碰到了类路径问题，请试试在类路径的每一个起点，搜索一下同名的.class文件。
(24) 在Java 1.1
AWT中使用事件“适配器”时，特别容易碰到一个陷阱。若覆盖了某个适配器方法，同时拼写方法没有特别讲究，最后的结果就是新添加一个方法，而不是覆盖现成方法。然而，由于这样做是完全合法的，所以不会从编译器或运行期系统获得任何出错提示——只不过代码的工作就变得不正常了。
(25)
用合理的设计方案消除“伪功能”。也就是说，假若只需要创建类的一个对象，就不要提前限制自己使用应用程序，并加上一条“只生成其中一个”注释。请考虑将其封装成一个“独生子”的形式。若在主程序里有大量散乱的代码，用于创建自己的对象，请考虑采纳一种创造性的方案，将些代码封装起来.
(26)
警惕“分析瘫痪”。请记住，无论如何都要提前了解整个项目的状况，再去考察其中的细节。由于把握了全局，可快速认识自己未知的一些因素，防止在考察细节的时候陷入“死逻辑”中。
(27) 警惕“过早优化”。首先让它运行起来，再考虑变得更快——
但只有在自己必须这样做、而且经证实在某部分代码中的确存在一个性能瓶颈的时候，才应进行优化。除非用专门的工具分析瓶颈，否则很有可能是在浪费自己的时间。性能提升的隐含代价是自己的代码变得难于理解，而且难于维护。
(28)
请记住，阅读代码的时间比写代码的时间多得多。思路清晰的设计可获得易于理解的程序，但注释、细致的解释以及一些示例往往具有不可估量的价值。无论对你自己，还是对后来的人，它们都是相当重要的。如对此仍有怀疑，那么请试想自己试图从联机Java文档里找出有用信息时碰到的挫折，这样或许能将你说服。
10.
如果系统要使用超大整数（超过long长度范围），请你设计一个数据结构来存储这种超大型数字以及设计一种算法来实现超大整数加法运算）。

```
public class BigInt() {
public static final long maxlong = ^0;
long[] ArrOne = new long[1000];
String intString="";
public int[] Arr(String s){
intString = s;
for(int i=0;i<ArrOne.leght;i++)
.....
```

49、列出某文件夹下的所有文件；
50、调用系统命令实现删除文件的操作；
51、实现从文件中一次读出一个字符的操作；
52、列出一些控制流程的方法；
54、编写了一个服务器端的程序实现在客户端输入字符然后在控制台上显示，直到输入"END"为止，让你写出客户端的程序；
在Jdbc进行数据库调用时，你经常采用什么方式执行sql语句？为什么不用其他方式（比较一下即可）
int类型在java中有多少位？（如果面试题目中有这样的问题，不是公司太牛就是公司太差）
你用过线程吗？请启动4个线程对一个静态变量进行加1操作。
线程是如何启动的？
每个类实力化时都调用父类的构造函数吗？如果是，那么都调用object类的构造函数吗

你懂得Ftp协议吗？如果不懂请问我告诉你Ftp协议命令格式及数据包的解析方法，你能用多长时间用java基本apI搞定一个ftp客户端程序（是这样的问题主要看你个人学习能力，一般也就是一人五天的工作量，不必要害怕，一般他不会给你五天做的，就是想看一下你的自信心及对工作的理解能力）
你知道java与C的通信?吗你会用那些协议进行通信？（其实也就是问socret通信）
请问java中的网络通信有那些方式，有什么区别？
String a=“”For limit
I=0;I<100000;I++)A=a+“A”把字符串成“A”连接100000次，上面方法不够好，请优化上面代码？（采用stringBuffer进行优化）
.
写出从数据库表Custom中查询No、Name、Num1、Num2并将Name以姓名显示、计算出的和以总和显示的SQL。SELECT No，Name AS ‘姓名’，Num1，Num2，（Num1+Num2）AS ‘总和’
FROM Custom
数据库方面：
1. 存储过程和函数的区别
存储过程是用户定义的一系列sql语句的集合，涉及特定表或其它对象的任务，用户可以调用存储过程，而函数通常是数据库已定义的方法，它接收参数并返回某种类型的值并且不涉及特定用户表。
2. 事务是什么？
事务是作为一个逻辑单元执行的一系列操作，一个逻辑工作单元必须有四个属性，称为ACID（原子性、一致性、隔离性和持久性）属性，只有这样才能成为一个事务：
原子性
事务必须是原子工作单元；对于其数据修改，要么全都执行，要么全都不执行。
一致性
事务在完成时，必须使所有的数据都保持一致状态。在相关数据库中，所有规则都必须应用于事务的修改，以保持所有数据的完整性。事务结束时，所有的内部数据结构（如 B 树索引或双向链表）都必须是正确的。
隔离性
由并发事务所作的修改必须与任何其它并发事务所作的修改隔离。事务查看数据时数据所处的状态，要么是另一并发事务修改它之前的状态，要么是另一事务修改它之后的状态，事务不会查看中间状态的数据。这称为可串行性，因为它能够重新装载起始数据，并且重播一系列事务，以使数据结束时的状态与原始事务执行的状态相同。
持久性
事务完成之后，它对于系统的影响是永久性的。该修改即使出现系统故障也将一直保持。
3. 游标的作用？如何知道游标已经到了最后？
游标用于定位结果集的行，通过判断全局变量@@FETCH_STATUS可以判断是否到了最后，通常此变量不等于0表示出错或到了最后。
4. 触发器分为事前触发和事后触发，这两种触发有和区别。语句级触发和行级触发有何区别。
事前触发器运行于触发事件发生之前，而事后触发器运行于触发事件发生之后。通常事前触发器可以获取事件之前和新的字段值。
语句级触发器可以在语句执行前或后执行，而行级触发在触发器所影响的每一行触发一次。
5.
何为“事务处理”，谈谈你对它的理解。事务处理是指一个单元的工作，这些工作要么全做，要么全部不做。作为一个逻辑单元，必须具备四个属性：自动性、一致性、独立性和持久性。自动性是指事务必须是一个自动的单元工作，要么执行全部数据的修改，要么全部数据的修改都不执行。一致性是指当事

务完成时，必须使所有数据都具有一致的状态。在关系型数据库中，所有的规则必须应用到事务的修改上，以便维护所有数据的完整性。所有的内部数据结构，在事务结束之后，必须保证正确。独立性是指并行事务的修改必须与其他并行事务的修改相互独立。一个事务看到的数据要么是另外一个事务修改这些事务之前的状态，要么是第二个事务已经修改完成的数据，但是这个事务不能看到正在修改的数据。

3. 常用的数据结构有哪些？请枚举一些。(不少于5个)链表、堆栈、二叉树、队列、图、堆，集合。

4. 什么是OOP？什么是类？请对比类和对象实例之间的关系。OOP是Object_oriented Programming(面向对象编程)的缩写。这主要是为了区别于以前的面向过程的程序设计！指的是用对象的观点来组织与构建系统，它综合了功能抽象和数据抽象，这样可以减少数据之间的耦合性和代码的出错几率。使用面向对象编程技术可以使得软件开发者按照现实世界里人们思考问题的模式编写代码,可以让软件开发者更好地利用代码直接表达现实中存在的对象,将问题空间直接映射到解空间!
类:即class 在面向对象的程序设计中，专门用“类”来表示用户定义的抽象数据类型(user_defined abstract
type)。它将具有相同状态、操作和访问机制的多个对象进行了抽象。类具有继承、数据隐藏和多态三种主要特性。利用类的这三种特性可以更好地表示现实世界中事物。类是同一类对象实例的共性的抽象，对象是类的实例化。对象通常作为计算机模拟思维，表示真实世界的抽象，一个对象就像一个软件模块，可以为用户提供一系列的服务---
可以改变对象的状态、测试、传递消息等。类定义了对象的实现细节或数据结构。类是静态的，对象是动态的，对象可以看作是运行中的类。类负责产生对象，可以将类当成生产对象的工厂(Object factory).

5. connection connconn.setAuto(false)//表示手动提交conn.commit// 提交conn.rollback();//事务回滚
-内联接use pubsselect a.au_fname, a.au_lname, p.pub_name from authors a inner join publishers p on a.city = p.city order by p.pub_name asc, a.au_lname asc, a.au_fname asc
--左外联接use pubs select a.au_fname, a.au_lname, p.pub_name from authors a left join publishers p on a.city = p.city order by p.pub_name asc, a.au_lname asc, a.au_fname asc
-使用子查询USE pubs GO select distinct pub_name FROM publishers WHERE pub_id IN (select pub_idFROM titlesWHERE type = 'business') GO
--如果平均价格少于 $30，WHILE 循环就将价格加倍，然后选择最高价。
--如果最高价少于或等于 $50，WHILE 循环重新启动并再次将价格加倍。
--该循环不断地将价格加倍直到最高价格超过 $50 USE pubs GO

```
WHILE (select AVG(price) FROM titles) < $30
BEGIN
  update titles
    SET price = price * 2
  select MAX(price) FROM titles
  IF (select MAX(price) FROM titles) > $50
    BREAK
  ELSE
    CONTINUE
END
```

---如果平均价格少于 $30，WHILE 循环就将价格加倍，然后选择最高价。
--如果最高价少于或等于 $50，WHILE 循环重新启动并再次将价格加倍。
--该循环不断地将价格加倍直到最高价格超过 $50

```
USE pubs
GO
```

```
WHILE (select AVG(price) FROM titles) < $30
BEGIN
  update titles
    SET price = price * 2
  select MAX(price) FROM titles
  IF (select MAX(price) FROM titles) > $50
    BREAK
  ELSE
    CONTINUE
END
CREATE PROCEDURE au_info
  @lastname varchar(40),
  @firstname varchar(20)
AS
select au_lname, au_fname, title, pub_name
  FROM authors a INNER JOIN titleauthor ta
    ON a.au_id = ta.au_id INNER JOIN titles t
    ON t.title_id = ta.title_id INNER JOIN publishers p
    ON t.pub_id = p.pub_id
  WHERE  au_fname = @firstname
    and au_lname = @lastname
GO
EXECUTE au_info 'Dull', 'Ann'--或者
EXECUTE au_info @lastname = 'Dull', @firstname = 'Ann'--创建存储过程CREATE PROCEDURE
titles_sum @TITLE varchar(40),@SUM money OUTPUT
AS
select @SUM = SUM(price)
FROM titles
WHERE title LIKE @TITLE
GO
DECLARE @TOTALCOST money
EXECUTE titles_sum 'The%', @TOTALCOST OUTPUT
select @TOTALCOST
go
CREATE PROCEDURE Oakland_authors
AS
select au_fname, au_lname, address, city, zip
FROM authors
WHERE city = 'Oakland'
and state = 'CA'
ORDER BY au_lname, au_fname
GO
--sp_helptext Oakland_authors
ALTER PROCEDURE Oakland_authors
AS
select au_fname, au_lname, address, city, zip
FROM authors
WHERE state = 'CA'
ORDER BY au_lname, au_fname
```

```
GO
--sp_helptext Oakland_authors
--提交事务后，所有书籍支付的版税增加 10%。
begin transaction MyTransaction
update roysched
set royalty = royalty * 1.10
commit transaction MyTransaction
--rollback transaction MyTransaction
select royalty from roysched
--select @@trancount
--1.创建试验实验表
create table temptrigger
( id_temp varchar(2) not null primary key,
  temp_name varchar(10) null,
  temp_age int null)go
insert temptrigger values('01','张三','10')
insert temptrigger values('02','李四','11')
insert temptrigger values('03','王五','12')
insert temptrigger values('04','赵六','11')
select * from temptrigger  go
--2.创建insert , update触发器
create trigger temptrigger_modify
on temptrigger
for insert,update
as
begin
  if (select temp_age from inserted) > 15
    begin
      rollback transaction
      print '年龄不能超过15岁！'
    end
end
--insert temptrigger values('04','大朋','17')
--insert temptrigger values('05','大朋','17')
--insert temptrigger values('05','大朋','14')
--update temptrigger set temp_age='18' where id_temp = '01'
--update temptrigger set temp_age='9' where id_temp = '01'
-3.创建delete 触发器--drop trigger temptrigger_delete
create trigger temptrigger_delete
on temptrigger
for delete
as
begin
  print @@rowcount
  if @@rowcount > 1
  begin
    rollback transaction
```

```
    print '一次删除记录不能多于1条'
  end
end
--delete from temptrigger
--delete from temptrigger where id_temp='01'
--创建聚集索引create clustered index clindx_titleid  on roysched(title_id)--sp_help roysched
--创建非聚集索引create nonclustered index unclindx_titleid  on roysched(title_id)--sp_help roysched
--查看索引统计dbcc show_statistics(roysched,titleidind)
--更新索引统计update statistics authors
--重建索引dbcc dbreindex('roysched',unclindx_titleid)
--删除索引drop index roysched.unclindx_titleid-sp_help roysched
1--创建ssn(社会保险号)的基于varchar的自定义数据类型。
--用于存储11位社会保险号(999-99-999)的列。该列不能
--为null。use pubs  exec sp_addtype ssn , 'varchar(11)' , 'NOT NULL'
--查看创建的数据类型--sp_help ssn
--使用创建的数据类型create table mytable( myid varchar(2) primary key, myssn ssn)
4-删除创建的数据类型--drop table mytable--exec sp_droptype ssn
?批是包含一个或多个 Transact-SQL 语句的组，从应用程序一次性地发送到 Microsoft SQL Server
执行。批作为一个整体执行，以GO命令结束。批处理是客户端作为一个单元发出的一个或多个 SQL
语句的集合。每个批处理编译为一个执行计划。
触发器?触发器是在对表进行插入、更新或删除操作时自动执行的存储过程?触发器通常用于强制业务
规则?触发器可以确保数据的完整性和一致性
事务是用户定义的一个操作序列，这些操作要么全做要么全不做，是一个不可分割的工作单位(构成
单一逻辑工作单元的操作集合)如果某一事务成功，则在该事务中进行的所有数据更改均会提交，成
为数据库中的永久组成部分。
如果事务遇到错误且必须取消或回滚，则所有数据更改均被清除
?锁 是在多用户环境中对数据访问的限制封锁就是事务 T
在对某个数据对象(如表、记录等)操作之前，先向系统发出请求，对其加锁。加锁后事务 T
就对该数据对象有了一定的控制，在事务T释放它的锁之前，其它的事务不能更新此数据对象。(锁蕴
含的基本概念是用户需要对表的排它访问)?从程序员的角度看:分为乐观锁和悲观锁。乐观锁:完全依
靠数据库来管理锁的工作。悲观锁:程序员自己管理数据或对象上的锁处理。
子查询:一个 select 语句嵌套在另一个 select 语句中。
—索引—
是一个数据库对象，它是某个表中一列或若干列值的集合和相应的指向表中物理标识这些值的数据
页的逻辑指针清单,然后根据指定的排序次序排列这些指针 —优点提高查询执行的速度。
强制实施数据的唯一性。 提高表之间联接的速度。缺点
存储索引要占用磁盘空间。数据修改需要更长的时间，因为索引也要更新。
?视图?是一种虚拟表，通常是作为来自一个或多个表
的行或列的子集创建的。?视图本质上讲，就是保存在数据库中的select查询?视图并不是数据库中存储
的数据值的集合。?对最终用户的好处– 结果更容易理解– 获得数据更容易
?对开发人员的好处– 限制数据检索更容易– 维护应用程序更方便
存储过程?使用一个名称存储的预编译T-
SQL语句和流程控制语句的集合?由数据库开发人员或数据库管理员编写
```

?用来执行管理任务或应用复杂的业务规则
优点?执行速度更快?首次运行时，进行优化和编译得到执行计划并将该计划存储在系统表中，以后直接运行。?实现多个程序共享应用程序逻辑?组件式编程?能够屏蔽数据库的结构，实现更高的安全性?减少网络流通量
数据库设计和建模必要性?好的数据库结构有利于:-节省数据的存储空间-能够保证数据的完整性-方便进行数据库应用系统的开发?设计不好的数据库结构将导致-数据冗余、存储空间浪费-内存空间浪费
不管数据库的大小和复杂程度如何，可以用下列基本步骤来设计数据库:–收集信息–标识对象–设计数据模型–标识每个对象 存储的信息类型–标识对象之间的关系
?数据模型是一种标识实体类型及其实体间联系的模型。典型的数据模型由网状模型、层次模型和关系模型。什么是规范化从关系数据库的表中，除去冗余数据的过程称为规范化。—精简数据库的结构—从表中删除冗余的列—标识所有依赖于其它数据的数据
三级范式第一范式的定义:
如果一个表中没有重复组(即行与列的交叉点上只有一个值，而不是一组值)，则这个表属于第一范式(常记成1NF)。简而言之:"每一字段只存储一个值"。例如:职工号，姓名，电话号码组成一个表(一个人可能有一个办公室电话 和一个家里电话号码)
第二范式的定义:如果一个表属于1NF，任何属性只依赖于关键字，则这个表属于第二范式(常记成2NF)。简而言之:必须先符合1NF的条件，且每一行都能被唯一的识别。
将1NF转换成2NF的方法是添加主键。学号,课程名,成绩第三范式的定义:如果一个表属于2NF，且不包含传递依赖性，则这个表是第三范式(常记成3NF)。满足3NF的表中不包含传递依赖。简而言之:没有一个非关键属性依赖于另一个非关键属性。学号，课程号，成绩，学分学号，姓名，所在系，系名称，系地址
什么是类与对象？
所谓对象就是真实世界中的实体，对象与实体是一一对应的，也就是说现实世界中每一个实体都是一个对象，它是一种具体的概念。
类是具备某些共同特征的实体的集合，它是一种抽象的概念，用程序设计的语言来说，类是一种抽象的数据类型，它是对所具有相同特征实体的抽象。
属性与方法？
不同对象具有相同特点，就可能抽象为一定的类，那么这些特点基本上可以分为两类，一类是描述对象静态状态的，就是对象的属性，在程序设计中，可以称之为变量;另一类是描述对象的动作，就是对象的方法，在程序设计中我们称之为函数。属性和方法是一个对象所具备的两大基本要素，也是我们后面编程工作的核心。
什么是封装？
只要有足够的方法，就没必要直接去操作对象属性，只要调用这些方法就可以实现要完成的任务，这种现象称为封装，它通过对象方法对其属性的操作把对象属性封装在一个对象内部，对象与外界打交道全部通过其自身的方法来实现，有效的把对象属性隐藏在对象内部。
编写 java文件的注意事项？
在记事本中编写java文件，在保存时一定要把文件名和扩展名用双引号括起来，否则将默认保存为文本文件，如果要保存的java
文件名为Program1.java,则在保存时在文件名文本框中一定要输入”Program1.java”。
如何编译java程序？

单击开始|运行命令，在命令行上输入cmd，按回车键（在window98中输入command，按回车键），即可打开一个命令窗口，将目录转换到编写java源程序所在的目录，输入javac filename.java

如何执行java程序？

同样在命令窗口中输入java filename,

基本数据类型？

Java的数据类型可以划分为4大类：整数，浮点数，字符型，布尔型。其中整数可以划分为：byte,short,int,long.浮点数可以划分为float,double.

c语言常见错误

C语言的最大特点是：功能强、使用方便灵活。C编译的程序对语法检查并不象其它高级语言那么严格，这就给编程人员留下“灵活的余地”，但还是由于这个灵活给程序的调试带来了许多不便，尤其对初学C语言的人来说，经常会出一些连自己都不知道错在哪里的错误。看着有错的程序，不知该如何改起，本人通过对C的学习，积累了一些C编程时常犯的错误，写给各位学员以供参考。

1.书写标识符时，忽略了大小写字母的区别。main() { int a=5; printf("%d",A); }

编译程序把a和A认为是两个不同的变量名，而显示出错信息。C认为大写字母和小写字母是两个不同的字符。习惯上，符号常量名用大写，变量名用小写表示，以增加可读性。

2.忽略了变量的类型，进行了不合法的运算。main() { float a,b; printf("%d",a%b); }

%是求余运算，得到a/b的整余数。整型变量a和b可以进行求余运算，而实型变量则不允许进行“求余”运算。

3.将字符常量与字符串常量混淆。char c; c="a";

在这里就混淆了字符常量与字符串常量，字符常量是由一对单引号括起来的单个字符，字符串常量是一对双引号括起来的字符序列。C规定以“”作字符串结束标志，它是由系统自动加上的，所以字符串“a”实际上包含两个字符：‘a'和‘'，而把它赋给一个字符变量是不行的。

4.忽略了“=”与“==”的区别。

在许多高级语言中，用“=”符号作为关系运算符“等于”。如在BASIC程序中可以写 if (a=3) then …

但C语言中，“=”是赋值运算符，“==”是关系运算符。如：if (a==3) a=b;

前者是进行比较，a是否和3相等，后者表示如果a和3相等，把b值赋给a。由于习惯问题，初学者往往会犯这样的错误。

5.忘记加分号。分号是C语句中不可缺少的一部分，语句末尾必须有分号。a=1 b=2

编译时，编译程序在“a=1”后面没发现分号，就把下一行“b=2”也作为上一行语句的一部分，这就会出现语法错误。改错时，有时在被指出有错的一行中未发现错误，就需要看一下上一行是否漏掉了分号。{ z=x+y; t=z/100; printf("%f",t); }

对于复合语句来说，最后一个语句中最后的分号不能忽略不写(这是和PASCAL不同的)。

6.多加分号。对于一个复合语句，如：{ z=x+y; t=z/100; printf("%f",t); };

复合语句的花括号后不应再加分号，否则将会画蛇添足。又如：if (a%3==0); I++;

本是如果3整除a，则I加1。但由于if (a%3==0)后多加了分号，则if语句到此结束，程序将执行I++语句，不论3是否整除a，I都将自动加1。再如：for (I=0;I<5;I++); {scanf("%d",&x); printf("%d",x);}

本意是先后输入5个数，每输入一个数后再将它输出。由于for()后多加了一个分号，使循环体变为空语句，此时只能输入一个数并输出它。

7.输入变量时忘记加地址运算符“&”。int a,b; scanf("%d%d",a,b);

这是不合法的。Scanf函数的作用是：按照a、b在内存的地址将a、b的值存进去。“&a”指a在内存中的地址。

8.输入数据的方式与要求不符。①scanf("%d%d",&a,&b);
输入时，不能用逗号作两个数据间的分隔符，如下面输入不合法：3，4
输入数据时，在两个数据之间以一个或多个空格间隔，也可用回车键，跳格键tab。
②scanf("%d,%d",&a,&b);
C规定：如果在“格式控制”字符串中除了格式说明以外还有其它字符，则在输入数据时应输入与这些字符相同的字符。下面输入是合法的：3，4 此时不用逗号而用空格或其它字符是不对的。3 4 3：4
又如：scanf("a=%d,b=%d",&a,&b); 输入应如以下形式：a=3,b=4
9.输入字符的格式与要求不一致。
在用“%c”格式输入字符时，“空格字符”和“转义字符”都作为有效字符输入。
scanf("%c%c%c",&c1,&c2,&c3); 如输入a b c 字符“a”送给c1，字符“
”送给c2，字符“b”送给c3，因为%c只要求读入一个字符，后面不需要用空格作为两个字符的间隔。
10.输入输出的数据类型与所用格式说明符不一致。例如，a已定义为整型，b定义为实型 a=3;b=4.5;
printf("%f%dn",a,b); 编译时不给出出错信息，但运行结果将与原意不符。这种错误尤其需要注意。
11.输入数据时，企图规定精度。scanf("%7.2f",&a); 这样做是不合法的，输入数据时不能规定精度。
12.switch语句中漏写break语句。例如：根据考试成绩的等级打印出百分制数段。switch(grade) { case 'A':printf("85~100n"); case 'B':printf("70~84n"); case 'C':printf("60~69n"); case 'D':printf("<60n"); default:printf("errorn");
由于漏写了break语句，case只起标号的作用，而不起判断作用。因此，当grade值为A时，printf函数在执行完第一个语句后接着执行第二、三、四、五个printf函数语句。正确写法应在每个分支后再加上“break;”。例如 case 'A':printf("85~100n");break;
13.忽视了while和do-while语句在细节上的区别。(1)main() {int a=0,I; scanf("%d",&I); while(I<=10) {a=a+I; I++; } printf("%d",a); } (2)main() {int a=0,I; scanf("%d",&I); do {a=a+I; I++; }while(I<=10); printf("%d",a); }
可以看到，当输入I的值小于或等于10时，二者得到的结果相同。而当I>10时，二者结果就不同了。因为while循环是先判断后执行，而do-while循环是先执行后判断。对于大于10的数while循环一次也不执行循环体，而do-while语句则要执行一次循环体。
14.定义数组时误用变量。int n; scanf("%d",&n); int a[n];
数组名后用方括号括起来的是常量表达式，可以包括常量和符号常量。即C不允许对数组的大小作动态定义。
15.在定义数组时，将定义的“元素个数”误认为是可使的最大下标值。main() {static int a[10]={1,2,3,4,5,6,7,8,9,10}; printf("%d",a[10]); }
C语言规定：定义时用a[10]，表示a数组有10个元素。其下标值由0开始，所以数组元素a[10]是不存在的。
16.初始化数组时，未使用静态存储。int a[3]={0,1,2};
这样初始化数组是不对的。C语言规定只有静态存储(static)数组和外部存储(exterm)数组才能初始化。
应改为：static int a[3]={0,1,2}; 17.在不应加地址运算符&的位置加了地址运算符。scanf("%s",&str);
C语言编译系统对数组名的处理是：数组名代表该数组的起始地址，且scanf函数中的输入项是字符数组名，不必要再加地址符&。应改为：scanf("%s",str);
18.同时定义了形参和函数中的局部变量。int max(x,y) int x,y,z; {z=x>y?x:y; return(z); }
形参应该在函数体外定义，而局部变量应该在函数体内定义。应改为：int max(x,y) int x,y; {int z; z=x>y?x:y; return(z);}

c语言数据类型

所谓数据类型是按被说明量的性质，表示形式，占据存储空间的多少，构造特点来划分的。在C语言中，数据类型可分为：基本数据类型，构造数据类型，指针类型，空类型四大类。

1.基本数据类型

基本数据类型最主要的特点是，其值不可以再分解为其它类型。也就是说，基本数据类型是自我说明的。

对于基本数据类型量，按其取值是否可改变又分为常量和变量两种。在程序执行过程中，其值不发生改变的量称为常量，取值可变的量称为变量。它们可与数据类型结合起来分类。例如，可分为整型常量、整型变量、浮点常量、浮点变量、字符常量、字符变量、枚举常量、枚举变量。在程序中，常量是可以不经说明而直接引用的，而变量则必须先说明后使用。

整型量 整型量包括整型常量、整型变量。

整型常量 整型常量就是整常数。在C语言中，使用的整常数有八进制、十六进制和十进制三种。

1.八进制整常数

八进制整常数必须以0开头，即以0作为八进制数的前缀。数码取值为0～7。八进制数通常是无符号数。以下各数是合法的八进制数：015(十进制为13)□0101(十进制为65)□0177777(十进制为65535)

以下各数不是合法的八进制数：256(无前缀0) 03A2(包含了非八进制数码)□-0127(出现了负号)

2.十六进制整常数

十六进制整常数的前缀为0X或0x。其数码取值为0~9，A~F或a~f。以下各数是合法的十六进制整常数：0X2A(十进制为42)□□0XA0 (十进制为160)□□0XFFFF (十进制为65535)

以下各数不是合法的十六进制整常数：5A (无前缀0X)□□0X3H (含有非十六进制数码)

3.十进制整常数　十进制整常数没有前缀。其数码为0～9。以下各数是合法的十进制整常数：237 -568 65535 1627 以下各数不是合法的十进制整常数：023 (不能有前导0)□23D (含有非十进制数码)

在程序中是根据前缀来区分各种进制数的。因此在书写常数时不要把前缀弄错造成结果不正确。

4.整型常数的后缀在16位字长的机器上，基本整型的长度也为16位，因此表示的数的范围也是有限定的。十进制无符号整常数的范围为0～65535，有符号数为-32768～+32767。八进制无符号数的表示范围为0～0177777。十六进制无符号数的表示范围为0X0～0XFFFF或0x0～0xFFFF。如果使用的数超过了上述范围，就必须用长整型数来表示。长整型数是用后缀“L”或“l”来表示的。

例如：十进制长整常数 158L (十进制为158) 358000L (十进制为-358000) 八进制长整常数 012L (十进制为10) 077L (十进制为63)□0200000L (十进制为65536) 十六进制长整常数 0X15L (十进制为21) 0XA5L (十进制为165)□0X10000L (十进制为65536) 长整数158L和基本整常数158在数值上并无区别。但对158L，因为是长整型量，C编译系统将为它分配4个字节存储空间。而对158，因为是基本整型，只分配2个字节的存储空间。因此在运算和输出格式上要予以注意，避免出错。无符号数也可用后缀表示，整型常数的无符号数的后缀为“U”或“u”。

例如：358u,0x38Au,235Lu均为无符号数。前缀，后缀可同时使用以表示各种类型的数。如0XA5Lu表示十六进制无符号长整数A5，其十进制为165。

整型变量 整型变量可分为以下几类：

1.基本型　类型说明符为int，在内存中占2个字节，其取值为基本整常数。

2.短整量　类型说明符为short int或short'C110F1。所占字节和取值范围均与基本型相同。

3.长整型　类型说明符为long int或long，在内存中占4个字节，其取值为长整常数。

4.无符号型　类型说明符为unsigned。无符号型又可与上述三种类型匹配而构成：(1)无符号基本型 类型说明符为unsigned int或unsigned。(2)无符号短整型 类型说明符为unsigned short (3)无符号长整型

类型说明符为unsigned long
各种无符号类型量所占的内存空间字节数与相应的有符号类型量相同。但由于省去了符号位，故不能表示负数。
在书写变量说明时，应注意以下几点：
1.允许在一个类型说明符后，说明多个相同类型的变量。各变量名之间用逗号间隔。类型说明符与变量名之间至少用一个空格间隔。
2.最后一个变量名之后必须以“;”号结尾。
3.变量说明必须放在变量使用之前。一般放在函数体的开头部分。
实型量 包括实型常量和实型变量。
实型常量 实型常量也称为实数或者浮点数。在C语言中，实数只采用十进制。它有二种形式：
十进制数形式指数形式
1.十进制数形式由数码0~ 9和小数点组成。例如：0.0，.25，5.789，0.13，5.0，300.，-267.8230等均为合法的实数。
2.指数形式由十进制数，加阶码标志“e”或“E”以及阶码（只能为整数，可以带符号）组成。其一般形式为a E n（a为十进制数，n为十进制整数）其值为 a*10,n□如：2.1E5 (等于2.1*10,5), 3.7E-2 (等于3.7*10,)-2*)□0.5E7 (等于0.5*10,7), -2.8E-2 (等于-2.8*10,)-2*)以下不是合法的实数 345 (无小数点)□E7 (阶码标志E之前无数字)□ -5 (无阶码标志)□53.-E3 (负号位置不对)□2.7E (无阶码)
标准C允许浮点数使用后缀。后缀为“f”或“F”即表示该数为浮点数。如356f和356.是等价的。下例说明了这种情况：

```
void main()
{ printf("%f\n%f\n",356.,356f); }
```

void 指明main不返回任何值□利用printf显示结果□结束
实型变量 实型变量分为两类：单精度型和双精度型.
其类型说明符为float 单精度说明符，double 双精度说明符。在Turbo C中单精度型占4个字节（32位）内存空间，其数值范围为3.4E-38～3.4E+38，只能提供七位有效数字。双精度型占8 个字节（64位）内存空间，其数值范围为1.7E-308～1.7E+308，可提供16位有效数字。实型变量说明的格式和书写规则与整型相同。
例如：float x,y; (x,y为单精度实型量) □□□
double a,b,c; (a,b,c为双精度实型量) 实型常数不分单、双精度，都按双精度double型处理。
字符型量 字符型量包括字符常量和字符变量。
字符常量
字符常量是用单引号括起来的一个字符。例如'a','b','=','+','?'都是合法字符常量。在C语言中，字符常量有以下特点：
1.字符常量只能用单引号括起来，不能用双引号或其它括号。
2.字符常量只能是单个字符，不能是字符串。
3.字符可以是字符集中任意字符。但数字被定义为字符型之后就不能参与数值运算。如'5'和5是不同的。'5'是字符常量，不能参与运算。
转义字符
转义字符也是一种特殊的字符常量。转义字符以反斜线"\"开头，后跟一个或几个字符。转义字符具有特定的含义，不同于字符原有的意义，故称“转义”字符。例如，在前面各例题printf函数的格式串中用到的“\n”就是一个转义字符，其意义是“回车换行”。转义字符主要用来表示那些用一般字符不便于表示的控制代码。常用的转义字符及其含义
\n □□□□□回车换行

\t □□□□□横向跳到下一制表位置
\v □□□□□竖向跳格
\b □□□□□退格
\r □□□□□回车
\f □□□□ □走纸换页
\\ □□□□□反斜线符"\"
\' □□□□□单引号符
\a □□□□□鸣铃
\ddd □□□□1～3位八进制数所代表的字符
\xhh □□□□1～2位十六进制数所代表的字符
广义地讲，C语言字符集中的任何一个字符均可用转义字符来表示。
字符变量
字符变量的取值是字符常量，即单个字符。字符变量的类型说明符是char。字符变量类型说明的格式和书写规则都与整型变量相同。
字符串常量是由一对双引号括起的字符序列。例如："CHINA"，"C program:"，"$12.5"
等都是合法的字符串常量。字符串常量和字符常量是不同的量。它们之间主要有以下区别：
1.字符常量由单引号括起来，字符串常量由双引号括起来。
2.字符常量只能是单个字符，字符串常量则可以含一个或多个字符。
3.可以把一个字符常量赋予一个字符变量，但不能把一个字符串常量赋予一个字符变量。在C语言中没有相应的字符串变量。这是与BASIC
语言不同的。但是可以用一个字符数组来存放一个字符串常量。在数组一章内予以介绍。
4.字符常量占一个字节的内存空间。字符串常量占的内存字节数等于字符串中字节数加1。增加的一个字节中存放字符"\0"(ASCII码为0)。这是字符串结束的标志。例如，字符串 "C
program"在内存中所占的字节为：C
program\0。字符常量'a'和字符串常量"a"虽然都只有一个字符，但在内存中的情况是不同的。
'a'在内存中占一个字节，可表示为：a
"a"在内存中占二个字节，可表示为：a\0符号常量
符号常量 在C语言中，可以用一个标识符来表示一个常量，称之为符号常量。
符号常量在使用之前必须先定义，其一般形式为：
#define 标识符 常量
其中#define也是一条预处理命令（预处理命令都?quot;#"开头），称为宏定义命令（在第九章预处理程序中将进一步介绍），其功能是把该标识符定义为其后的常量值。一经定义，以后在程序中所有出现该标识符的地方均代之以该常量值。习惯上符号常量的标识符用大写字母，变量标识符用小写字母，以示区别。

```
#define PI 3.14159
void main(){
float s,r;
r=5;
s=PI*r*r;
printf("s=%f\n",s);
}
```

由宏定义命令定义PI 为3.14159 s,r定义为实数□5->r□PI*r*r->s
显示程序结果 float s,r;□r=5;□s=PI*r*r; 本程序在主函数之前由宏定义命令定义PI
为3.14159，在程序中即以该值代替PI

。s=PI*r*r等效于s=3.14159*r*r。应该注意的是，符号常量不是变量，它所代表的值在整个作用域内不能再改变。也就是说，在程序中，不能再用赋值语句对它重新赋值。
2.构造数据类型
是根据已定义的一个或多个数据类型用构造的方法来定义的。也就是说，一个构造类型的值可以分解成若干个“成员”或“元素”。每个“成员”都是一个基本数据类型或又是一个构造类型。在C语言中，构造类型有以下几种：
·数组类型
·结构类型
·联合类型
3.指针类型
指针是一种特殊的，同时又是具有重要作用的数据类型。其值用来表示某个量在内存储器中的地址。虽然指针变量的取值类似于整型量，但这是两个类型完全不同的量，因此不能混为一谈。
4.空类型
在调用函数值时，通常应向调用者返回一个函数值。这个返回的函数值是具有一定的数据类型的，应在函数定义及函数说明中给以说明，但是，也有一类函数，调用后并不需要向调用者返回函数值，这种函数可以定义为“空类型”。其类型说明符为void。

多个参数遇到的问题
我在使用ajax中，涉及到一个url中多个参数处理的问题：
出现错误：
[Fatal Error] :1:135: The reference to entity "Type" must end with the ';' delimiter.
或
[Fatal Error] :1:131: The entity name must immediately follow the '&' in the entity reference.
原因：是SAX解析器出错，是XML的问题。
使用"&"代替"&"，原理和HTML中的转义相同，参照XML的语法规范。

常用log4j配置

开发75条（写的不错）选择自 churujianghu 的 Blog
开发75条：
1. 你们的项目组使用源代码管理工具了么？
应该用。VSS、CVS、PVCS、ClearCase、CCC/Harvest、FireFly都可以。我的选择是VSS。
2. 你们的项目组使用缺陷管理系统了么？
应该用。ClearQuest太复杂，我的推荐是BugZilla。
3. 你们的测试组还在用Word写测试用例么？
不要用Word写测试用例（Test Case）。应该用一个专门的系统，可以是Test Manager，也可以是自己开发一个ASP.NET的小网站。主要目的是Track和Browse。
4. 你们的项目组有没有建立一个门户网站？
要有一个门户网站，用来放Contact Info、Baselined Schedule、News等等。推荐Sharepoint Portal Server 2003来实现，15分钟就搞定。买不起SPS 2003可以用WSS (Windows Sharepoint Service)。
5. 你们的项目组用了你能买到最好的工具么？
应该用尽量好的工具来工作。比如，应该用VS.NET而不是Notepad来写C#。用Notepad写程序多半只是一种炫耀。但也要考虑到经费，所以说是“你能买到最好的”。

6. 你们的程序员工作在安静的环境里么？
需要安静环境。这点极端重要，而且要保证每个人的空间大于一定面积。
7.
你们的员工每个人都有一部电话么？需要每人一部电话。而且电话最好是带留言功能的。当然，上这么一套带留言电话系统开销不小。不过至少每人一部电话要有，千万别搞得经常有人站起来喊：“某某某电话”。《人件》里面就强烈谴责这种做法。
8. 你们每个人都知道出了问题应该找谁么？
应该知道。任何一个Feature至少都应该有一个Owner，当然，Owner可以继续Dispatch给其他人。
9. 你遇到过有人说“我以为…”么？
要消灭“我以为”。Never assume anything。
10. 你们的项目组中所有的人都坐在一起么？
需要。我反对Virtual Team，也反对Dev在美国、Test在中国这种开发方式。能坐在一起就最好坐在一起，好处多得不得了。
11. 你们的进度表是否反映最新开发进展情况？
应该反映。但是，应该用Baseline的方法来管理进度表：维护一份稳定的Schedule，再维护一份最新更改。Baseline的方法也应该用于其它的Spec。Baseline是变更管理里面的一个重要手段。
12. 你们的工作量是先由每个人自己估算的么？
应该让每个人自己估算。要从下而上估算工作量，而不是从上往下分派。除非有其他原因，比如政治任务工期固定等。
13. 你们的开发人员从项目一开始就加班么？
不要这样。不要一开始就搞疲劳战。从项目一开始就加班，只能说明项目进度不合理。当然，一些对日软件外包必须天天加班，那属于剥削的范畴。
14. 你们的项目计划中Buffer Time是加在每个小任务后面的么？
不要。Buffer Time加在每个小任务后面，很容易轻易的就被消耗掉。Buffer Time要整段的加在一个Milestone或者checkpoint前面。
15. 值得再多花一些时间，从95%做到100%好值得，非常值得。
尤其当项目后期人困马乏的时候，要坚持。这会给产品带来质的区别。
16. 登记新缺陷时，是否写清了重现步骤？
要。这属于Dev和Test之间的沟通手段。面对面沟通需要，详细填写Repro Steps也需要。
17.
写新代码前会把已知缺陷解决么？要。每个人的缺陷不能超过10个或15个，否则必须先解决老的bug才能继续写新代码。
18. 你们对缺陷的轻重缓急有事先的约定么？
必须有定义。Severity要分1、2、3，约定好：蓝屏和Data Lost算Sev 1，Function Error算Sev 2，界面上的算Sev 3。但这种约定可以根据产品质量现状适当进行调整。
19. 你们对意见不一的缺陷有三国会议么？必须要有。要有一个明确的决策过程。这类似于CCB (Change Control Board)的概念。
20. 所有的缺陷都是由登记的人最后关闭的么？
Bug应该由Opener关闭。Dev不能私自关闭Bug。
21. 你们的程序员厌恶修改老的代码么？
厌恶是正常的。解决方法是组织Code Review，单独留出时间来。XP也是一个方法。
22. 你们项目组有Team Morale Activity么？
每个月都要搞一次，吃饭、唱歌、Outing、打球、开卡丁车等等，一定要有。不要剩这些钱。

23. 你们项目组有自己的Logo么？
要有自己的Logo。至少应该有自己的Codename。
24. 你们的员工有印有公司Logo的T-Shirt么？
要有。能增强归属感。当然，T-Shirt要做的好看一些，最好用80支的棉来做。别没穿几次就破破烂烂的。
25. 总经理至少每月参加次项目组会议要的。
要让team member觉得高层关注这个项目。
26. 你们是给每个Dev开一个分支么？
反对。Branch的管理以及Merge的工作量太大，而且容易出错。
27. 有人长期不Check-In代码么？
不可以。对大部分项目来说，最多两三天就应该Check-In。
28. 在Check-In代码时都填写注释了么？
要写的，至少一两句话，比如“解决了Bug No.225”。如果往高处拔，这也算做“配置审计”的一部分。
29. 有没有设定每天Check-In的最后期限？
要的，要明确Check-In Deadline。否则会Build Break。
30. 你们能把所有源码一下子编译成安装文件吗？
要的。这是每日编译（Daily Build）的基础。而且必须要能够做成自动的。
31. 你们的项目组做每日编译么？
当然要做。有三样东西是软件项目/产品开发必备的：1. bug management; 2. source control; 3. daily build。
32. 你们公司有没有积累一个项目风险列表？
要。Risk Inventory。否则，下个项目开始的时候，又只能拍脑袋分析Risk了。
33. 设计越简单越好越简单越好。
设计时候多一句话，将来可能就带来无穷无尽的烦恼。应该从一开始就勇敢的砍。这叫scope management。
34.
尽量利用现有的产品、技术、代码千万别什么东西都自己Coding。BizTalk和Sharepoint就是最好的例子，有这两个作为基础，可以把起点提高很多。或者可以尽量多用现成的Control之类的。或者尽量用XML，而不是自己去Parse一个文本文件；尽量用RegExp，而不是自己从头操作字符串，等等等等。这就是“软件复用”的体现。
35. 你们会隔一段时间就停下来夯实代码么？
要。最好一个月左右一次。传言去年年初Windows组在Stevb的命令下停过一个月增强安全。Btw，“夯”这个字念“hang”，第一声。
36. 你们的项目组每个人都写Daily Report么？
要写。五分钟就够了，写10句话左右，告诉自己小组的人今天我干了什么。一则为了沟通，二则鞭策自己（要是游手好闲一天，自己都会不好意思写的）。
37. 你们的项目经理会发出Weekly Report么？
要。也是为了沟通。内容包括目前进度，可能的风险，质量状况，各种工作的进展等。
38. 你们项目组是否至少每周全体开会一次？
要。一定要开会。程序员讨厌开会，但每个礼拜开会时间加起来至少应该有4小时。包括team meeting, spec review meeting, bug triage meeting。千万别大家闷头写code。
39. 你们项目组的会议、讨论都有记录么？

会前发meeting request和agenda，会中有人负责主持和记录，会后有人负责发meeting minutes，这都是effective meeting的要点。而且，每个会议都要形成agreements和action items。

40. 其他部门知道你们项目组在干什么么？

要发一些Newsflash给整个大组织。Show your team's value。否则，当你坐在电梯里面，其他部门的人问："你们在干嘛"，你回答"ABC项目"的时候，别人全然不知，那种感觉不太好。

41. 通过Email进行所有正式沟通

Email的好处是免得抵赖。但也要避免矫枉过正，最好的方法是先用电话和当面说，然后Email来确认。

42. 为项目组建立多个Mailing Group

如果在AD+Exchange里面，就建Distribution List。比如，我会建ABC Project Core Team，ABC Project Dev Team，ABC Project All Testers，ABC Project Extended Team等等。这样发起Email来方便，而且能让该收到email的人都收到、不该收到不被骚扰。

43. 每个人都知道哪里可以找到全部的文档么？

应该每个人都知道。这叫做知识管理（Knowledge Management）。最方便的就是把文档放在一个集中的File Share，更好的方法是用Sharepoint。

44. 你做决定、做变化时，告诉大家原因了么？

要告诉大家原因。Empower team member的手段之一是提供足够的information，这是MSF一开篇的几个原则之一。的确如此，tell me why是人之常情，tell me why了才能有understanding。中国人做事喜欢搞限制，限制信息，似乎能够看到某一份文件的人就是有身份的人。大错特错。权威、权力，不在于是不是能access information/data，而在于是不是掌握资源。

45. Stay agile and expect change 要这样。

需求一定会变的，已经写好的代码一定会被要求修改的。做好心理准备，对change不要抗拒，而是expect change。

46. 你们有没有专职的软件测试人员？

要有专职测试。如果人手不够，可以peer test，交换了测试。千万别自己测试自己的。

47. 你们的测试有一份总的计划来规定做什么和怎么做么？这就是Test Plan。要不要做性能测试？要不要做Usability测试？什么时候开始测试性能？测试通过的标准是什么？用什么手段，自动的还是手动的？这些问题需要用Test Plan来回答。

48. 你是先写Test Case然后再测试的么？

应该如此。应该先设计再编程、先test case再测试。当然，事情是灵活的。我有时候在做第一遍测试的同时补上test case。至于先test case再开发，我不喜欢，因为不习惯，太麻烦，至于别人推荐，那试试看也无妨。

49. 你是否会为各种输入组合创建测试用例？

不要，不要搞边界条件组合。当心组合爆炸。有很多test case工具能够自动生成各种边界条件的组合——但要想清楚，你是否有时间去运行那么多test case。

50. 你们的程序员能看到测试用例么？

要。让Dev看到Test Case吧。我们都是为了同一个目的走到一起来的：提高质量。

51. 你们是否随便抓一些人来做易用性测试？

要这么做。自己看自己写的程序界面，怎么看都是顺眼的。这叫做审美疲劳——臭的看久了也就不臭了，不方便的永久了也就习惯了。

52. 你对自动测试的期望正确么？

别期望太高。依我看，除了性能测试以外，还是暂时先忘掉“自动测试”吧，忘掉WinRunner和LoadRunner吧。对于国内的软件测试的现状来说，只能“矫枉必须过正”了。

53. 你们的性能测试是等所有功能都开发完才做的么？

不能这样。性能测试不能被归到所谓的“系统测试”阶段。早测早改正，早死早升天。

54. 你注意到测试中的杀虫剂效应了么？

虫子有抗药性，Bug也有。发现的新Bug越来越少是正常的。这时候，最好大家交换一下测试的area，或者用用看其他工具和手法，就又会发现一些新bug了。

55. 你们项目组中有人能说出产品的当前整体质量情况么？

要有。当老板问起这个产品目前质量如何，Test Lead/Manager应该负责回答。

56. 你们有单元测试么？

单元测试要有的。不过没有单元测试也不是不可以，我做过没有单元测试的项目，也做成功了——可能是侥幸，可能是大家都是熟手的关系。还是那句话，软件工程是非常实践、非常工程、非常灵活的一套方法，某些方法在某些情况下会比另一些方法好，反之亦然。

57. 你们的程序员是写完代码就扔过墙的么？

大忌。写好一块程序以后，即便不做单元测试，也应该自己先跑一跑。虽然有了专门的测试人员，做开发的人也不可以一点测试都不做。微软还有Test Release Document的说法，程序太烂的话，测试有权踢回去。

58. 你们的程序中所有的函数都有输入检查么？

不要。虽然说做输入检查是write secure code的要点，但不要做太多的输入检查，有些内部函数之间的参数传递就不必检查输入了，省点功夫。同样的道理，未必要给所有的函数都写注释。写一部分主要的就够了。

59. 产品有统一的错误处理机制和报错界面么？

要有。最好能有统一的error message，然后每个error message都带一个error number。这样，用户可以自己根据error number到user manual里面去看看错误的具体描述和可能原因，就像SQL Server的错误那样。同样，ASP.NET也要有统一的Exception处理。可以参考有关的Application Block。

60. 你们有统一的代码书写规范么？

要有。Code Convention很多，搞一份来发给大家就可以了。当然，要是有FxCop这种工具来检查代码就更好了。

61. 你们的每个人都了解项目的商业意义么？

要。这是Vision的意思。别把项目只当成工作。有时候要想着自己是在为中国某某行业的信息化作先驱者，或者时不时的告诉team member，这个项目能够为某某某国家部门每年节省多少多少百万的纳税人的钱，这样就有动力了。平凡的事情也是可以有个崇高的目标的。

62. 产品各部分的界面和操作习惯一致么？

要这样。要让用户觉得整个程序好像是一个人写出来的那样。

63. 有可以作为宣传亮点的Cool Feature么？

要。这是增强团队凝聚力、信心的。而且，“一俊遮百丑”，有亮点就可以掩盖一些问题。这样，对于客户来说，会感觉产品从质量角度来说还是acceptable的。或者说，cool feature或者说亮点可以作为质量问题的一个事后弥补措施。

64. 尽可能缩短产品的启动时间要这样。

软件启动时间（Start-Up time）是客户对性能好坏的第一印象。

65.
不要过于注重内在品质而忽视了第一眼的外在印象程序员容易犯这个错误:太看重性能、稳定性、存储效率，但忽视了外在感受。而高层经理、客户正相反。这两方面要兼顾，协调这些是PM的工作。
66. 你们根据详细产品功能说明书做开发么？
要这样。要有设计才能开发，这是必须的。设计文档，应该说清楚这个产品会怎么运行，应该采取一些讲故事的方法。设计的时候千万别钻细节，别钻到数据库、代码等具体实现里面去，那些是后面的事情，一步步来不能着急。
67. 开始开发和测试之前每个人都仔细审阅功能设计么？
要做。Function Spec
review是用来统一思想的。而且，review过以后形成了一致意见，将来再也没有人可以说“你看，当初我就是反对这么设计的，现在吃苦头了吧”
68. 所有人都始终想着The Whole
Image么？要这样。项目里面每个人虽然都只是在制造一片叶子，但每个人都应该知道自己在制造的那片叶子所在的树是怎么样子的。我反对软件蓝领，反对过分的把软件制造看成流水线、车间。参见第61条。
69. Dev工作的划分是单纯纵向或横向的么？
不能单纯的根据功能模块分，或者单纯根据表现层、中间层、数据库层分。我推荐这么做:首先根据功能模块分，然后每个“层”都有一个Owner来Review所有人的设计和代码，保证consistency。
70. 你们的程序员写程序设计说明文档么？
要。不过我听说微软的程序员1999年以前也不写。所以说，写不写也不是绝对的，偷懒有时候也是可以的。71. 你在招人面试时让他写一段程序么？
要的。我最喜欢让人做字符串和链表一类的题目。这种题目有很多循环、判断、指针、递归等，既不偏向过于考算法，也不偏向过于考特定的API。
72. 你们有没有技术交流讲座？
要的。每一两个礼拜搞一次内部的Tech Talk或者Chalk
Talk吧。让组员之间分享技术心得，这笔花钱送到外面去培训划算。
73. 你们的程序员都能专注于一件事情么？
要让程序员专注一件事。例如说，一个部门有两个项目和10个人，一种方法是让10个人同时参加两个项目，每个项目上每个人都花50%时间;另一种方法是5个人去项目A，5个人去项目B，每个人都100%在某一个项目上。我一定选后面一种。这个道理很多人都懂，但很多领导实践起来就把属下当成可以任意拆分的资源了。
74. 你们的程序员会夸大完成某项工作所需要的时间么？
会的，这是常见的，尤其会在项目后期夸大做某个change所需要的时间，以次来抵制change。解决的方法是坐下来慢慢磨，磨掉程序员的逆反心理，一起分析，并把估算时间的颗粒度变小。
75. 尽量不要用Virtual Heads 最好不要用Virtual Heads。
Virtual heads意味着resource is not secure，shared
resource会降低resource的工作效率，容易增加出错的机会，会让一心二用的人没有太多时间去review spec、review
design。一个dedicated的人，要强过两个只能投入50%时间和精力的人。我是吃过亏的:7个part time的tester，发现的Bug和干的活，加起来还不如两个full-
time的。参见第73条。73条是针对程序员的，75条是针对Resource Manager的。
我现在做的项目是采用如下方法管理的:

BD:基础设计。在这个文档里，我们把程序的界面全部画出来；界面上的功能全部描述完整。如：一个查询界面的条件是什么，查询出来的结果如何显示等等。
FD:功能设计。在这个文档里，对BD阶段的各个页面里包含的数据逻辑处理做说明。如：查询时调用的数据处理函数该如何设计，入口参数，返回参数，关联的表等等各方面的说明。
因为程序的界面已经定了，数据处理逻辑也定了。所以，就开始编码阶段。当编码过程中发生什么问题，程序的整个功能还是必须满足BD和FD设计文档中的要求。程序中的各种疑问，都以BD和FD文档中的说明为准。
基本上我们一个小项目的开发周期为2个月，BD为2-3周，FD1周，PG（编程）2-3周，CT(测试)2周。
测试完毕后交出去的就是成品，基本上不会再有系统要求变更的问题了。如果有变更，且不在BD设计范围内，那就是新增需求。就是一个新项目了。
以上资料，仅供参考。
19. 说出在JSP页面里是怎么分页的?
页面需要保存以下参数：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。。
2 文件读写的基本类
答：File Reader 类和FileWriter类分别继承自Reader类和Writer类。FileReader类用于读取文件，File Writer类用于将数据写入文件，这两各类在使用前，都必须要调用其构造方法创建相应的对象，然后调用相应的read()或 write()方法。
11 forward和rederect的区别
答：redirect重定向到另外一个页面，重新开始一个请求forward跳转到另外一个页面，请求不断开
12 jsp的常用的命令
答：page, include, talib, forward,
14 servlet的init()方法和service()方法的区别
答：初始化时调用init()方法有请求到达时调用service()方法，service()根据请求的类型，调用doget()或depost()等方法
15、servlet的配置
<web-app><servlet><servlet-name>Admin</servlet-name><servlet-class>jb-aptech.adminservlet</servlet-class><init-param><param-name>email</param-name><param-value>admin@jb-aptech.com.cn</param-value></init-param> </servlet></web-app>
16 客服端口调用EJB对象的几个基本步骤
答;设置JNDI服务工厂以及JNDI服务地址系统属性，查找Home接口，从Home接口调用Create方法创建Remote接口通过Remote接口调用其业务方法
17、java的调试如何进行。
答：jdb 是java 的调试器，类似于 UNIX系统的调试器 dbx,jdb 使用
Java调试器应用程序接口来完成对本地或远程的Java调试器的调用工作。一般是在要测试的代码段想控制台打印消息。
18 java中对象之间的通讯采用什么方法。
答：直接调用另一对象方法来进行通讯以及数据的交换。
19 tcp/ip在连接是有几次握手？释放是有几次握手？答：建立连接是2次,释放是3次。
21 Java中线程间怎么通讯？什么叫僵死线程？答：线程之间可以通过管道进行通讯。
23、在java中如何进行socket编程。
答：Sockets有两种主要的操作方式:面向连接的和无连接的。

无连接的操作使用数据报协议.这个模式下的socket不需要连接一个目的的socket,它只是简单地投出数据报.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.一个这个模式下的socket必须在发送数据之前与目的地的socket取得一个连接.一旦连接建立了,sockets就可以使用一个流接口:打开-读-写-
关闭.所有的发送的信息都会在另一端以同样的顺序被接收.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.
在服务器,使用ServerSocket监听指定的端口,端口可以随意指定(由于1024以下的端口通常属于保留端口,在一些操作系统中不可以随意使用,所以建议使用大于1024的端口),等待客户连接请求,客户连接后,会话产生;在完成会话后,关闭连接。在客户端,使用Socket对网络上某一个服务器的某一个端口发出连接请求,一旦连接成功,打开会话;会话完成后,关闭Socket。客户端不需要指定打开的端口,通常临时的、动态的分配一个1024以上的端口。

24 用java怎样实现多线程?线程有那些状态?

答:Java 中实现多线程的方法有两种,一是继承 java.lang 包中的 Thread 类,二是用户自己的类实现 Runnable 接口。初始状态,就绪状态,阻塞状态,运行状态,死亡状态。

24、同时编译java两个类应带什么参数?答:CLASS PATH

30 在java中一个类被声明为final类型,表示了什么意思?

表示该类不能被继承,是顶级类。

32. 写一个方法,实现字符串的反转,如:输入abc,输出cba

```
public static String reverse(String s){int length=s.length();StringBuffer result=new
StringBuffer(length);for(int i=length-1;i>=0;i--)result.append(s.charAt(i));return result.toString();}
```

59在main(String[] args)方法内是否可以调用一个非静态方法?答案:不能

60一个文件里是否可以有两个public类?答案:不能

75. 类之间的继承关系

答:在类的创建过程中,新的类可以通过在原有类的斟础上增加新的成吊门烃量和成员方法来创建,这种创建方式称为继承。被继承的类称为父类,创建的新类称为子类。
如果基类的成员是私有的,那么派生类的成员不能访问这些成员。
如果基类的成员是保护的,那么派生类的成员能够访问这些成员,但保护成员不能在类的外部访问。
如果基类的成员是公有的,那么派生类的成员函数能够访问这些成员。

76. java有些什么集合类型,各自的区别

答,map,set,list
HashMap,HashTable,后者线程同步
set不允许重复元素
ArrayList,Vector,后者线程同步
ArrayList,LinkedList,前者读的速度快,后者修改速度快

77. 声明型异常和运行期异常的异同

答,都是Exception的子类
声明型必须catch,运行期异常继承自RuntimeException.

81 介绍在Jsp中如何使用JavaBeans。 1、使用动作元素〈use bean〉 2、在脚本中调用JavaBeans

82. 简单介绍JSP的标记库JSTL

答:从
JSP1.1规范开始,JSP就支持在JSP中使用自定义标签了,自定义标签的广使用造成了程序员的重定义,这样就促成了JSTL(JavaServer Pages Standard Tag Library)的诞生。JSTL椒一个不断完善的开放源泉代码的JSP标签库,是由Apache 的

Jakarta小组开发的。JSTL至少行在支持JSP1.2和Servlet2.3无偏见范的容器上，如Tomcat4.x。在JSP2.0最新标准中，JSTL作为标准支持，JSTL最新的版本是JSTL1.1。

JSTL的优点：

(1)在应用程序服务器之间提供了一致的接口，最大程度地提高了Web应用在各应用服务器之间的移植。

(2)简化了JSP和Web□应用程序的开发。

(3)以一种统一的方式减少了JSP中的Scriptlets代码数量，可以达到没有任何的IDE开发工具出现。

(4)允许JSP设计工具与Web应用程序开发的进一步集成，相信不久就会有支持JSTL的IDE开发工具出现

83 Jsp和Servlet中的请求转派发分别如何实现。

Jsp 实现转派发

1、在java脚本代码中使用 response.sendRedirect("favorite.jsp") 实现转派发

2、JSP标准动作：<jsp:forward> 例如：<jsp:forward page="forward2.jsp" />

Servlet 实现转派发

1、代码 response.sendRedirect("/abc.html")； 遇到该行代码，会转到abc.html页面。

2、ServletContext的getRequestDispatcher(String path)方法 返回与path相联系的RequestDispatcher对象

RequestDispatcher对象 调用 forward(ServletRequest request, ServletResponse response) 转向 path

Html 实现转派发

1、使用 "history.back()" 例如:<input type="button" name="Submit2" value="返回" onClick="history.back();">

2、使用 "javascript:history.go(-1);" 例如:返回！

3、使用 href 超链接 实现转派发 例如:返回！

4、使用 form 表单提交 实现转派发

5、使用 meta 例如:<meta http-equiv=refresh content='<%=aForwardInfo.getSecond()%>; url=<%=aForwardInfo.getUrl()%>'>

84 struts的优点，流程

答，优点：从mvc,BO VO方面回答

流程：client-->总控ActionServlet

总控调以下操作：

-->form收集数据

-->action调javbean处理业务(调ejb,DAO...)

-->转向相应的jsp

jsp从form中取得数据显示

85. 简单介绍所了解的XML。

xml 可扩展标记语言 ；

是一种存储数据标准格式，应用于数据交换、web服务、web集成、系统配置、可穿透防火墙；

J2EE初学者需要理解的问题

一、J2EE提出的背景

1、企业级应用框架的需求

在许多企业级应用中，例如数据库连接、邮件服务、事务处理等都是一些通用企业需求模块，这些模块如果每次再开发中都由开发人员来完成的话，将会造成开发周期长和代码可*性差等问题。于是许多大公司开发了自己的通用模块服务。这些服务性的软件系列统称为中间件。

2、为了通用必须要提出规范，不然无法达到通用

在上面的需求基础之上，许多公司都开发了自己的中间件，但其与用户的沟通都各有不同，从而导致用户无法将各个公司不同的中间件组装在一块为自己服务。从而产生瓶颈。于是提出标准的概念。其实J2EE就是基于JAVA技术的一系列标准。

注：中间件的解释中间件处在操作系统和更高一级应用程序之间。他充当的功能是：将应用程序运行环境与操作系统隔离，从而实现应用程序开发者不必为更多系统问题忧虑，而直接关注该应用程序在解决问题上的能力。我们后面说到的容器的概念就是中间件的一种。

二、相关名词解释

容器：充当中间件的角色

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。

WEB容器和EJB容器在原理上是大体相同的，更多的区别是被隔离的外界环境。WEB容器更多的是跟基于HTTP的请求打交道。而EJB容器不是。它是更多的跟数据库、其它服务打交道。但他们都是把与外界的交互实现从而减轻应用程序的负担。例如SERVLET不用关心HTTP的细节，直接引用环境变量session,request,response就行、EJB不用关心数据库连接速度、各种事务控制，直接由容器来完成。

RMI/IIOP:远程方法调用/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。

JNDI: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。

JAVAMAIL:JAVA邮件服务。提供邮件的存储、传输功能。他是JAVA编程中实现邮件功能的核心。相当MS中的EXCHANGE开发包。

JTA：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。

EAI:企业应用集成。是一种概念，从而牵涉到好多技术。J2EE技术是一种很好的集成实现。

三、J2EE的优越性

1、基于JAVA 技术，平台无关性表现突出

2、开放的标准，许多大型公司已经实现了对该规范支持的应用服务器。如BEA ,IBM,ORACLE等。

3、提供相当专业的通用软件服务。

4、提供了一个优秀的企业级应用程序框架，对快速高质量开发打下基础

四、现状

J2EE是由SUN 公司开发的一套企业级应用规范。现在最高版本是1.4。支持J2EE的应用服务器有IBM WEBSPHERE APPLICATION SERVER,BEA WEBLOGIC SERVER,JBOSS,ORACLE APPLICATION SERVER,SUN ONE APPLICATION SERVER 等。

学习Java的30个基本概念

Java概述:

目前Java主要应用于中间件的开发(middleware)---

处理客户机于服务器之间的通信技术,早期的实践证明,Java不适合pc应用程序的开发,其发展逐渐变成在开发手持设备,互联网信息站,及车载计算机的开发.Java于其他语言所不同的是程序运行时提供了平台的独立性,称许可以在windows,solaris,linux其他操作系统上使用完全相同的代码.Java的语法与C++语法类似,C++/C程序员很容易掌握,而且Java是完全的彻底的面向对象的,其中提出了很好的GC(Garbage Collector)垃圾处理机制,防止内存溢出.

Java的白皮书为我们提出了Java语言的11个关键特性.

(1)Easy:Java的语法比C++的相对简单,另一个方面就是Java能使软件在很小的机器上运行,基础解释其和类库的支持的大小约为40kb,增加基本的标准库和线程支持的内存需要增加125kb.

(2)分布式:Java带有很强大的TCP/IP协议族的例程库,Java应用程序能够通过URL来穿过网络来访问远程对象,由于servlet机制的出现,使Java编程非常的高效,现在许多的大的web server都支持servlet.

(3)OO:面向对象设计是把重点放在对象及对象的接口上的一个编程技术.其面向对象和C++有很多不同,在与多重继承的处理及Java的原类模型.

(4)健壮特性:Java采取了一个安全指针模型,能减小重写内存和数据崩溃的可能性。

(5)安全:Java用来设计网路和分布系统,这带来了新的安全问题,Java可以用来构建防病毒和防攻击的System.事实证明Java在防毒这一方面做的比较好.

(6)中立体系结构:Java编译其生成体系结构中立的目标文件格式可以在很多处理器上执行,编译器产生的指令字节码(Javabytecode)实现此特性,此字节码可以在任何机器上解释执行.

(7)可移植性:Java中对基本数据结构类型的大小和算法都有严格的规定所以可移植性很好.

(8)多线程:Java处理多线程的过程很简单,Java把多线程实现交给底下操作系统或线程程序完成.所以多线程是Java作为服务器端开发语言的流行原因之一

(9)Applet和servlet:能够在网页上执行的程序叫Applet,需要支持Java的浏览器很多,而applet支持动态的网页,这是很多其他语言所不能做到的.

基本概念:

1.OOP中唯一关系的是对象的接口是什么,就像计算机的销售商她不管电源内部结构是怎样的,他只关系能否给你提供电就行了,也就是只要知道can or not而不是how and why.所有的程序是由一定的属性和行为对象组成的,不同的对象的访问通过函数调用来完成,对象间所有的交流都是通过方法调用,通过对封装对象数据,很大限度上提高复用率.

2.OOP中最重要的思想是类,类是模板是蓝图,从类中构造一个对象,即创建了这个类的一个实例(instance)

3.封装:就是把数据和行为结合起在一个包中)并对对象使用者隐藏数据的实现过程,一个对象中的数据叫他的实例字段(instance field)

4.通过扩展一个类来获得一个新类叫继承(inheritance),而所有的类都是由Object根超类扩展而得,根超类下文会做介绍.

5.对象的3个主要特性

behavior---说明这个对象能做什么.

state---当对象施加方法时对象的反映.

identity---与其他相似行为对象的区分标志.

每个对象有唯一的indentity 而这3者之间相互影响.
6.类之间的关系:
use-a :依赖关系
has-a :聚合关系
is-a :继承关系--
例:A类继承了B类,此时A类不仅有了B类的方法,还有其自己的方法.(个性存在于共性中)
7.构造对象使用构造器:构造器的提出,构造器是一种特殊的方法,构造对象并对其初始化.
例:Data类的构造器叫Data
new Data()---构造一个新对象,且初始化当前时间.
Data happyday=new
Data()---
把一个对象赋值给一个变量happyday,从而使该对象能够多次使用,此处要声明的使变量与对象变量二者是不同的.new返回的值是一个引用.
构造器特点:构造器可以有0个,一个或多个参数
构造器和类有相同的名字
一个类可以有多个构造器
构造器没有返回值
构造器总是和new运算符一起使用.
8.重载:当多个方法具有相同的名字而含有不同的参数时,便发生重载.编译器必须挑选出调用哪个方法.
9.包(package)Java允许把一个或多个类收集在一起成为一组,称作包,以便于组织任务,标准Java库分为许多包.java.lang java.util java,net等,包是分层次的所有的java包都在java和javax包层次内.
10.继承思想:允许在已经存在的类的基础上构建新的类,当你继承一个已经存在的类时,那么你就复用了这个类的方法和字段,同时你可以在新类中添加新的方法和字段.
11.扩展类:扩展类充分体现了is-a的继承关系. 形式为:class (子类) extends (基类).
12.多态:在java中,对象变量是多态的.而java中不支持多重继承.
13.动态绑定:调用对象方法的机制.
(1)编译器检查对象声明的类型和方法名.
(2)编译器检查方法调用的参数类型.
(3)静态绑定:若方法类型为priavte static final 编译器会准确知道该调用哪个方法.
(4)当程序运行并且使用动态绑定来调用一个方法时,那么虚拟机必须调用x所指向的对象的实际类型相匹配的方法版本.
(5)动态绑定:是很重要的特性,它能使程序变得可扩展而不需要重编译已存代码.
14.final类:为防止他人从你的类上派生新类,此类是不可扩展的.
15.动态调用比静态调用花费的时间要长,
16.抽象类:规定一个或多个抽象方法的类本身必须定义为abstract例: public abstract string getDescripition
17.Java中的每一个类都是从Object类扩展而来的.
18.object类中的equal和toString方法.equal用于测试一个对象是否同另一个对象相等.toString返回一个代表该对象的字符串,几乎每一个类都会重载该方法,以便返回当前状态的正确表示.(toString方法是一个很重要的方法)
19.通用编程:任何类类型的所有值都可以同object类性的变量来代替.
20.数组列表:ArrayList动态数组列表,是一个类库,定义在java.uitl包中,可自动调节数组的大小.

21.class类
object类中的getclass方法返回ckass类型的一个实例,程序启动时包含在main方法的类会被加载,虚拟机要加载他需要的所有类,每一个加载的类都要加载它需要的类.
22.class类为编写可动态操纵java代码的程序提供了强大的功能反射,这项功能为JavaBeans特别有用,使用反射Java能支持VB程序员习惯使用的工具.能够分析类能力的程序叫反射器,Java中提供此功能的包叫Java.lang.reflect反射机制十分强大.
1.在运行时分析类的能力.
2.在运行时探察类的对象.
3.实现通用数组操纵代码.
4.提供方法对象.
而此机制主要针对是工具者而不是应用及程序.
反射机制中的最重要的部分是允许你检查类的结构.用到的API有:
java.lang.reflect.Field 返回字段.
java.reflect.Method 返回方法.
java.lang.reflect.Constructor 返回参数.
方法指针:java没有方法指针,把一个方法的地址传给另一个方法,可以在后面调用它,而接口是更好的解决方案.
23.接口(Interface)说明类该做什么而不指定如何去做,一个类可以实现一个或多个interface.
24.接口不是一个类,而是对符合接口要求的类的一套规范.若实现一个接口需要2个步骤:
1.声明类需要实现的指定接口.
2.提供接口中的所有方法的定义.
声明一个类实现一个接口需要使用implements 关键字class actionB implements Comparable
其actionb需要提供CompareTo方法,接口不是类,不能用new实例化一个接口.
25.一个类只有一个超类,但一个类能实现多个接口.Java中的一个重要接口Cloneable
26.接口和回调.编程一个常用的模式是回调模式,在这种模式中你可以指定当一个特定时间发生时回调对象上的方法.例:ActionListener 接口监听.
类似的API有:java.swing.JOptionPane
java.swing.Timer
java.awt.Tookit
27.对象clone:clone方法是object一个保护方法,这意味着你的代码不能简单的调用它.
28.内部类:一个内部类的定义是定义在另一个内部的类
原因是:1.一个内部类的对象能够访问创建它的对象的实现,包括私有数据
2.对于同一个包中的其他类来说,内部类能够隐藏起来.
3.匿名内部类可以很方便的定义回调.
4.使用内部类可以非常方便的编写事件驱动程序.
29.代理类(proxy):1.指定接口要求所有代码 2.object类定义的所有的方法(toString equals)
30.数据类型:Java是强调类型的语言,每个变量都必须先申明它都类型,java中总共有8个基本类型.4种是整型,2种是浮点型,一种是字符型,被用于Unicode编码中的字符,布尔型.
名词解释
a) JMS(Java消息服务),做消息处理。
b) DOM(文档对象模型),用来解析XML。
c) MVC(Model—View—Controller),分离表现逻辑、业务逻辑和数据。
d) JNDI(Java命名和目录接口),提供名字服务。

e) LDAP(轻量级目录访问协议)，提供数据的存储方式。
f) EJB(企业级Java Bean)，作为Model，可以封装数据(实体Bean)，也可以表示业务功能(会话Bean)，作为Model，可以处理消息(MDB)。
g) UML(统一建模语言)，提供画图的规范。Rose，Visio。
h) DTD(文档类型定义)，规定XML文件的格式。
i) XML(可扩展的标记语言)。
j) JAXP(解析XML的Java API).
k) RMI(远程方法调用)，在一个机器上使用另一个机器上创建的对象。
l) SOAP(简单对象访问协议)，用在WebService中。

EJB必须掌握的概念

1、从J2EE整体架构上看EJB

EJB是J2EE中用于构建分布式对象模型的一种技术，
采用OO设计，基于CORBA标准，使用RMI低层技术完成分布式运算
EJB是一个行业标准，必须部署到支持EJB的容器之中

2、与JavaBean的区别

EJB运行在服务器上，而JAVABEAN运行在客户端上
WEB端在J2EE中实际上是一个“客户端”程序。

3、EJB的分类及主要使用

分类	子类型	保存周期	作用
会话Bean (Session Bean)	无状态(stateless)	短暂，容器一关闭就消失	1□ 被远程客户端所访问，完成业务逻辑 2□ 调用实体Bean
	有状态(statefull)		
实体Bean (Entity Bean)	CMP(容器管理持久)	永久，数据保存在数据库中	1□ ORM的一种实现数据查询 2□ 操作数据库 3□ 永远实现本地接口，远程端无法直接操作实体Bean
	BMP(Bean管理持久)		
消息驱动Bean(MDB) (Message Driver Bean)	QUEUE(队列)	---	发送PTP消息
	TOPIC(主题)		发送主题消息

从EJB发展角度观察；
1、 EJB 1.1之前只有:SessionBean、EntityBean
2、EJB 2.0开始就包含了新的组件:MDB，如果使用MDB则需要消息队列中间件支持，MDB是基于JMS(Java消息服务)的一种应用。

4、三种EJB的关系

J2EE中提供两种应用
A、 远程客户 → MDB → Session Bean → Entity Bean → Database
B、 远程客户 → Session Bean → MDB → Entity Bean → Database
但在实际应用中往往只使用Session Bean + Entity Bean

5、会话与实体Bean中的主要方法说明

A、会话Bean

· ejbActivate()：从实例池中取出已有的对象继续使用
· ejbPassivate()：将暂时不用的对象保存在实例池中等待再次被使用
· 远程方法必须在远程接口（视图）中定义后才可以被远程客户端看见

B、实体Bean

本地接口方法	对应SQL类型	Bean中的方法	备注
create（local home）	insert	ejbCreate(属性)	Insert通过此方法完成
		ejbPostCreate(属性)	插入完成之后的收尾工作，一般为空
setter（local）	update	ejbStore()	一旦调用setter方法之后，则自动调用ejbStore()方法进行数据库更新
remove（EJBLocalObject）	delete	ejbRemove()	执行删除
findById	select id	ejbLoad()	实体Bean必须提供一个按ID查找的方法，此方法调用时不用单独编写EJB QL（EJB查询语言）
findByXxx	select	--	需要单独编写EJB QL进行操作

6、BMP与CMP的区别

	CMP	BMP
概念	容器管理持久，所有操作都由EJB容器完成	Bean管理持久，所有的代码必须开发人员手工编写
代码形式	抽象类及抽象方法的集合 所有的属性都在部署文件中配置	具体的一个操作类，所有方法不能为抽象，在相应的地方将方法体补充完整
执行效率	效率低	效率高
可移植性	可移植性高	可移植性低
总则	无论是BMP还是CMP最终全部都是ORM的一种应用，其实际的理论价值要比实现价值高	
	在实际开发中，EJB使用并不常见，而通过Hibernate去替代CMP操作	

7、EJB的方法调用过程

· 初始化JNDI上下文
　　|- Context ctx = new InitialContext() ;
· 查询远程HOME接口
　　|- 远程Home接口 home = (远程Home接口)ctx.lookup("JNDI名称") ;
· 通过HOME接口产生REMOTE接口的实例化对象
　　|- 远程接口 remote = home.create() ;
· 调用远程方法
　　|- remote.远程方法()

8、EJB代码的组成

· 容器自动生成的代码
· EJB API提供的代码
· 开发人员自己编写的代码

9、EJB中的六种角色

EJB服务器提供商、EJB容器提供商、EJB组件提供商、EJB组装人员、部署人员、系统管理员