**Java经典面试题及答案解析-100题**

目录

[1. 作用域public,private,protected,以及不写时的区别 4](#_Toc476585157)

[2. String是最基本的数据类型吗? 4](#_Toc476585158)

[3. int 和 Integer 有什么区别 4](#_Toc476585159)

[4. String 和StringBuffer的区别 4](#_Toc476585160)

[5. 运行时异常与一般异常有何异同？ 4](#_Toc476585161)

[6. 说出Servlet的生命周期，并说出Servlet和CGI的区别。 4](#_Toc476585162)

[7. 说出ArrayList,Vector, LinkedList的存储性能和特性 5](#_Toc476585163)

[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。 5](#_Toc476585164)

[9. Collection 和 Collections的区别。 5](#_Toc476585165)

[10. &和&&的区别。 5](#_Toc476585166)

[11. HashMap和Hashtable的区别。 5](#_Toc476585167)

[12. final, finally, finalize的区别。 6](#_Toc476585168)

[13. sleep() 和 wait() 有什么区别? 6](#_Toc476585169)

[14. Overload和Override的区别。Overloaded的方法是否可以改变返回值的类型? 6](#_Toc476585170)

[15. error和exception有什么区别? 6](#_Toc476585171)

[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。 6](#_Toc476585172)

[17. abstract class和interface有什么区别? 6](#_Toc476585173)

[18. heap和stack有什么区别。 7](#_Toc476585174)

[19. forward 和redirect的区别 7](#_Toc476585175)

[20. EJB与JAVA BEAN的区别？ 7](#_Toc476585176)

[21. Static Nested Class 和 Inner Class的不同。 7](#_Toc476585177)

[22. JSP中动态INCLUDE与静态INCLUDE的区别？ 7](#_Toc476585178)

[23. 什么时候用assert。 8](#_Toc476585179)

[24. GC是什么? 为什么要有GC? 8](#_Toc476585180)

[25. short s1 = 1; s1 = s1 + 1;有什么错? short s1 = 1; s1 += 1;有什么错? 8](#_Toc476585181)

[26. Math.round(11.5)等於多少? Math.round(-11.5)等於多少? 8](#_Toc476585182)

[27. String s = new String("xyz");创建了几个String Object? 8](#_Toc476585183)

[28. 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。 8](#_Toc476585184)

[29. Java有没有goto? 9](#_Toc476585185)

[30. 启动一个线程是用run()还是start()? 9](#_Toc476585186)

[31. EJB包括（SessionBean,EntityBean）说出他们的生命周期，及如何管理事务的？ 9](#_Toc476585187)

[32. 应用服务器有那些？ 10](#_Toc476585188)

[33. 给我一个你最常见到的runtime exception。 10](#_Toc476585189)

[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)? 10](#_Toc476585190)

[35. List, Set, Map是否继承自Collection接口? 10](#_Toc476585191)

[36. 说出数据连接池的工作机制是什么? 10](#_Toc476585192)

[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? 10](#_Toc476585193)

[38. 数组有没有length()这个方法? String有没有length()这个方法？ 10](#_Toc476585194)

[39. Set里的元素是不能重复的，那么用什么方法来区分重复与否呢? 是用==还是equals()? 它们有何区别? 10](#_Toc476585195)

[40. 构造器Constructor是否可被override? 11](#_Toc476585196)

[41. 是否可以继承String类? 11](#_Toc476585197)

[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? 11](#_Toc476585198)

[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? 11](#_Toc476585199)

[44. 编程题: 用最有效率的方法算出2乘以8等於几? 11](#_Toc476585200)

[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? 11](#_Toc476585201)

[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? 11](#_Toc476585202)

[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? 11](#_Toc476585203)

[48. 编程题: 写一个Singleton出来。 11](#_Toc476585204)

[49. Java的接口和C++的虚类的相同和不同处。 12](#_Toc476585205)

[50. Java中的异常处理机制的简单原理和应用。 12](#_Toc476585206)

[51. 垃圾回收的优点和原理。并考虑2种回收机制。 12](#_Toc476585207)

[52. 请说出你所知道的线程同步的方法。 13](#_Toc476585208)

[53. 你所知道的集合类都有哪些？主要方法？ 13](#_Toc476585209)

[54. 描述一下JVM加载class文件的原理机制? 13](#_Toc476585210)

[55. char型变量中能不能存贮一个中文汉字?为什么? 13](#_Toc476585211)

[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? 13](#_Toc476585212)

[57. JSP的内置对象及方法。 13](#_Toc476585213)

[58. 线程的基本概念、线程的基本状态以及状态之间的关系 14](#_Toc476585214)

[59. JSP的常用指令 14](#_Toc476585215)

[60. 什么情况下调用doGet()和doPost()？ 14](#_Toc476585216)

[61. servlet的生命周期 14](#_Toc476585217)

[62. 如何现实servlet的单线程模式？ 14](#_Toc476585218)

[63. 页面间对象传递的方法 15](#_Toc476585219)

[64. JSP和Servlet有哪些相同点和不同点，他们之间的联系是什么？ 16](#_Toc476585220)

[65. 四种会话跟踪技术 16](#_Toc476585221)

[66. Request对象的主要方法： 16](#_Toc476585222)

[67. J2EE是技术还是平台还是框架？ 17](#_Toc476585223)

[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？ 17](#_Toc476585224)

[69. 简述逻辑操作(&,|,^)与条件操作(&&,||)的区别。 17](#_Toc476585225)

[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？ 17](#_Toc476585226)

[71. 简述synchronized和 18](#_Toc476585227)

[java.util.concurrent.locks.Lock的异同 ？ 18](#_Toc476585228)

[72. EJB的角色和三个对象 18](#_Toc476585229)

[73. EJB容器提供的服务 18](#_Toc476585230)

[74. EJB规范规定EJB中禁止的操作有哪些？ 18](#_Toc476585231)

[75. remote接口和home接口主要作用 18](#_Toc476585232)

[76. bean 实例的生命周期 18](#_Toc476585233)

[77. EJB的激活机制 18](#_Toc476585234)

[78. EJB的几种类型 18](#_Toc476585235)

[79. 客服端调用EJB对象的几个基本步骤 19](#_Toc476585236)

[80. 如何给weblogic指定大小的内存? 19](#_Toc476585237)

[81. 如何设定的weblogic的热启动模式(开发模式)与产品发布模式? 19](#_Toc476585238)

[82. 如何启动时不需输入用户名与密码? 19](#_Toc476585239)

[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? 19](#_Toc476585240)

[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? 19](#_Toc476585241)

[85. 在weblogic中发布ejb需涉及到哪些配置文件 19](#_Toc476585242)

[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 19](#_Toc476585243)

[87. 如何查看在weblogic中已经发布的EJB? 20](#_Toc476585244)

[88. CORBA是什么?用途是什么? 20](#_Toc476585245)

[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 20](#_Toc476585246)

[90. 说说在weblogic中开发消息Bean时的persistent与non-persisten的差别 20](#_Toc476585247)

[91. Servlet执行时一般实现哪几个方法？ 20](#_Toc476585248)

[92. j2ee常用的设计模式？说明工厂模式。 21](#_Toc476585249)

[93. EJB需直接实现它的业务接口或Home接口吗，请简述理由。 21](#_Toc476585250)

[94. 排序都有哪几种方法？请列举。用JAVA实现一个快速排序。 25](#_Toc476585251)

[95. 请对以下在J2EE中常用的名词进行解释(或简单描述) 25](#_Toc476585252)

[96. StringBuffer是线程安全的，而StringBuilder是非线程安全的。 26](#_Toc476585253)

[97. Vector,ArrayList, LinkedList的区别是什么？ 26](#_Toc476585254)

[98. 什么是队列？ 26](#_Toc476585255)

[99. List 与 Set 的区别？ 27](#_Toc476585256)

[100. 数据库的三大范式？ 27](#_Toc476585257)

[101. java 实现非递归二分法算法。 27](#_Toc476585258)

[102. Java 写一个冒泡程序 28](#_Toc476585259)

[103. 多线程 29](#_Toc476585260)

[104. java 如何实现高并发？ 30](#_Toc476585261)

[105. HashMap、 TreeMap 区别，以及 TreeMap 原理 30](#_Toc476585262)

# 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 √ × × ×

不写时默认为friendly

# String是最基本的数据类型吗?

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

# 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。

# 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。

# 运行时异常与一般异常有何异同？

异常表示程序运行过程中可能出现的非正常状态，运行时异常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。

# 说出Servlet的生命周期，并说出Servlet和CGI的区别。

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

与cgi的区别在于servlet处于服务器进程中，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。

# 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。

# 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。

# 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。

# &和&&的区别。

&是位运算符，表示按位与运算，&&是逻辑运算符，表示逻辑与（and）。

# 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。

# 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。

# sleep() 和 wait() 有什么区别?

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。

# 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。

# 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。

# 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。

# 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。

# 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

# 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。

# 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。

# Static Nested Class 和 Inner Class的不同。

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

# JSP中动态INCLUDE与静态INCLUDE的区别？

动态INCLUDE用jsp:include动作实现 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

# 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。

# 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。

# 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）

# Math.round(11.5)等於多少? Math.round(-11.5)等於多少?

Math.round(11.5)==12

Math.round(-11.5)==-11

round方法返回与参数最接近的长整数，参数加1/2后求其floor.

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

两个

# 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();

}

}

}

}

# Java有没有goto?

java中的保留字，现在没有在java中使用。

# 启动一个线程是用run()还是start()?

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。

# 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。

# 应用服务器有那些？

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

# 给我一个你最常见到的runtime exception。

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

# 接口是否可继承接口? 抽象类是否可实现(implements)接口? 抽象类是否可继承实体类(concrete class)?

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

# List, Set, Map是否继承自Collection接口?

List，Set是，Map不是

# 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。

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

都不能

# 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。

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

Set里的元素是不能重复的，那么用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。

# 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。

# 是否可以继承String类?

String类是final类故不可以继承。

# 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。

# 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。

# 编程题: 用最有效率的方法算出2乘以8等於几?

2 << 3

# 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。

# 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。

# 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?

不能，一个对象的一个synchronized方法只能由一个线程访问。

# 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: 定义一个类，它的构造函数为private的，它有一个static的private的该类变量，在类初始化时实例话，通过一个public的getInstance方法获取对它的引用,继而调用其中的方法。

public class Singleton {

private Singleton(){}

　　 //在自己内部定义自己一个实例，是不是很奇怪？

　　 //注意这是private 只供内部调用

　　 private static Singleton instance = new Singleton();

　　 //这里提供了一个供外部访问本class的静态方法，可以直接访问

　　 public static Singleton getInstance() {

　　　　 return instance;

　　 }

}

第二种形式:

public class Singleton {

　　private static Singleton instance = null;

　　public static synchronized Singleton getInstance() {

　　//这个方法比上面有所改进，不用每次都进行生成对象，只是第一次

　　//使用时生成实例，提高了效率！

　　if (instance==null)

　　　　instance＝new Singleton();

return instance; 　　}

}

其他形式:

定义一个类，它的构造函数为private的，所有方法为static的。

一般认为第一种形式要更加安全些

# 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。

# 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。

# 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。

# 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。

# 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。

# 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。

# char型变量中能不能存贮一个中文汉字?为什么?

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

# 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

# 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

# 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。

# JSP的常用指令

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

" target="\_blank">http://......"%>

# 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()。

# 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。

# 如何现实servlet的单线程模式？

实现servlet的单线程的jsp命令是： <%@ page isThreadSafe=”false”%>

<%@ page isThreadSafe="true|false" %>

默认值为true

isThreadSafe=false模式表示它是以Singleton模式运行。

该模式implements了接口SingleThreadMode,

该模式同一时刻只有一个实例，不会出现信息同步与否的概念。

若多个用户同时访问一个这种模式的页面，

那么先访问者完全执行完该页面后，后访问者才开始执行。

isThreadSafe=true模式表示它以多线程方式运行。

该模式的信息同步，需访问同步方法(用synchronized标记的)来实现。

一般格式如下：

public synchronized void syncmethod(...){

while(...) {

this.wait();

}

this.notifyAll();

}

那我们看看他编译后的servlet类和以前的servlet有什么不同？没有加入这条指令的页面中servlet类定义是：

public final class index\_jsp extends

org.apache.jasper.runtime.HttpJspBase

implements org.apache.jasper.runtime.JspSourceDependent {

………………………………//内部实现咱们大可以不看他。

}

而加入这个命令之后的类定义是：

public final class index\_jsp extends

org.apache.jasper.runtime.HttpJspBase

implements org.apache.jasper.runtime.JspSourceDependent,

SingleThreadModel {

…………………………//同样省略实现细节

}

我想大家应该很清楚的发现：加入<@page isThreadSafe="false">之后servlet是实现了SingleThreadModel接口的。有兴趣的朋友可以查下相关资料，其实这个接口在servlet2.4之后就废除了。那么为什么废除？实现了这个接口查看代码内容的时候却发现什么方法都没添加，为什么？

其实这个接口只是一个标识，让容器改变servlet生成策略的。

　　默认的，servlet容器是只创建servlet単实例，为请求提供线程池，我们为了能保证servlet的线程安全问题而不在servlet中添加实例属性。

但是如果一个servlet实现了SingleThreadModel接口那容器在创建实例池，为每个求情分配实例，当用户请求结束时将实例归还实例池。

这样做真的安全吗？非也！因为每个请求都会分发一个servlet实例，对于同用户下分发的不同的servlet来说很可能用到同一个session中的属性数据，这样当然出现了线程同步的问题，是不安全的！而且不同用户也可能享用同一个context中的数据，也是不安全的。于是在servlet2.4中建议不再使用这样的设置。

　　这样一来我们知道：如果非要实现servlet单线程那就要在servlet中实现SingleThreadModel接口，在jsp中添加<@ page isThreadSafe="false">命令。但是这是不被提倡的。

　　也有网友疑问：让servlet单线程而多实例本来不是为了线程安全的吗？为什么<@ page isThreadSafe=?>？的答案却是false？

　　我觉得这是个误区，这样设置真的是单线程吗？那肯定不是了，其实这个设置只是要容器保证同一时刻只有一个线程能在servlet的service方法中执行，因为默认的容器实例化servlet时时単实例，如果同时只有一个线程去管理这个唯一的servlet实例，那性能将是极其慢的，为了解决这个问题，容器采取了servlet实例池的方式。每个请求一个线程，而一个线程一个servlet。

因为这种方式并不是线程安全的，所以答案是false。

　　终上所述：其实这道题目是指如果让一个servlet实例对应一个线程，而不是一个servlet实例对应多个线程的默认方式。

# 页面间对象传递的方法

request，session，application，cookie等

# 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。

# 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

# 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

# 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。

# 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;

}

# 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

# 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)

xml文档有两种定义方法：

dtd：数据类型定义（data type definition），用以描述XML文档的文档结构，是早期的XML文档定义形式。

schema：其本身是基于XML语言编写的，在类型和语法上的限定能力比dtd强，处理也比较方便，因为此正逐渐代替dtd成为新的模式定义语言。

# 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。

# 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

# 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。

# 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。

# 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

# 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。

# 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。

# 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

# 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。

# 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

# 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。

# 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.

# 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。

# 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。

# 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

# 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。

# 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

# 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。

# 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。

# 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。

既然没有标准答案，就根据自己的所了解的，补充修正一下好了

# 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()

init ()方法在servlet的生命周期中仅执行一次，在服务器装载servlet时执行。缺省的init()方法通常是符合要求的，不过也可以根据需要进行 override，比如管理服务器端资源，一次性装入GIF图像，初始化数据库连接等，缺省的inti()方法设置了servlet的初始化参数，并用它 的ServeltConfig对象参数来启动配置，所以覆盖init()方法时，应调用super.init()以确保仍然执行这些任务。

service ()方法是servlet的核心，在调用service()方法之前，应确保已完成init()方法。对于HttpServlet，每当客户请求一个 HttpServlet对象，该对象的service()方法就要被调用，HttpServlet缺省的service()方法的服务功能就是调用与 HTTP请求的方法相应的do功能，doPost()和doGet()，所以对于HttpServlet，一般都是重写doPost()和doGet() 方法。

destroy()方法在servlet的生命周期中也仅执行一次，即在服务器停止卸载servlet时执行，把servlet作为 服务器进程的一部分关闭。缺省的destroy()方法通常是符合要求的，但也可以override，比如在卸载servlet时将统计数字保存在文件 中，或是关闭数据库连接。

getServletConfig()方法返回一个servletConfig对象，该对象用来返回初始化参数和servletContext。servletContext接口提供有关servlet的环境信息。

getServletInfo()方法提供有关servlet的信息，如作者，版本，版权。

# 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。

# 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。

其实一直都不是很明白EJB的remote接口，home接口，Bean类究竟是如何使用的，或许应该进一步了解EJB的原理吧，查到了一个原创文章，那就说说EJB调用的原理吧。其实在这个问题上，最需要理解的是RMI机制原理。

一个远程对象至少要包括4个class文件：远程对象、远程对象接口、实现远程接口的对象的stub、对象的skeleton。

而在EJB中则至少要包括10个class:

Bean类，特定App Server的Bean实现类

Bean的remote接口，特定App Server的remote接口实现类，特定App Server的remote接口的实现类的stub类和skeleton类。

Bean的home接口，特定App Server的home接口实现类，特定App Server的home接口的实现类的stub类和skeleton类。

和RMI不同的是，EJB中这10个class真正需要用户写的只有3个，Bean类，remote接口，home接口，其它的7个究竟怎么生成，被打包在哪里，是否需要更多的类文件，否根据不同的App Server表现出较大的差异。

Weblogic：

home接口和remote接口的weblogic的实现类的stub类和skeleton类是在EJB被部署到weblogic的时候，由weblogic动态生成stub类和skeleton类的字节码，所以看不到这4个类文件。

对 于一次客户端远程调用EJB，要经过两个远程对象的多次RMI循环。首先是通过JNDI查找Home接口，获得Home接口的实现类，这个过程其实相当复 杂，首先是找到Home接口的Weblogic实现类，然后创建一个Home接口的Weblogic实现类的stub类的对象实例，将它序列化传送给客户 端（注意stub类的实例是在第1次RMI循环中，由服务器动态发送给客户端的，因此不需要客户端保存Home接口的Weblogic实现类的stub 类），最后客户端获得该stub类的对象实例（普通的RMI需要在客户端保存stub类，而EJB不需要，因为服务器会把stub类的对象实例发送给客户 端）。

客户端拿到服务器给它的Home接口的Weblogic实现类的stub类对象实例以后，调用stub类的create方法， (在代码上就是home.create()，但是后台要做很多事情),于是经过第2次RMI循环，在服务器端，Home接口的Weblogic实现类的 skeleton类收到stub类的调用信息后，由它再去调用Home接口的Weblogic实现类的create方法。

在服务端，Home 接口的Weblogic实现类的create方法再去调用Bean类的Weblogic实现类的ejbCreate方法，在服务端创建或者分配一个EJB 实例，然后将这个EJB实例的远程接口的Weblogic实现类的stub类对象实例序列化发送给客户端。

客户端收到remote接 口的Weblogic实现类的stub类的对象实例，对该对象实例的方法调用（在客户端代码中实际上就是对remote接口的调用），将传送给服务器端 remote接口的Weblogic实现类的skeleton类对象，而skeleton类对象再调用相应的remote接口的Weblogic实现类， 然后remote接口的Weblogic实现类再去调用Bean类的Weblogic实现类，如此就完成一次EJB对象的远程调用。

先拿普通RMI来说，有4个class，分别是远程对象，对象的接口，对象的stub类和skeleton类。而对象本身和对象的stub类同时都实现了接口类。而我们在客户端代码调用远程对象的时候，虽然在代码中操纵接口，实质上是在操纵stub类，例如：

接口类：Hello

远程对象：Hello\_Server

stub类：Hello\_Stub

skeleton类：Hello\_Skeleton

客户端代码要这样写：

Hello h = new Hello\_Stub();

h.getString();

我们不会这些写：

Hello\_Stub h = new Hello\_Stub();

h.getString();

因 为使用接口适用性更广，就算更换了接口实现类，也不需要更改代码。因此客户端需要Hello.class和Hello\_Stub.class这两个文件。 但是对于EJB来说，就不需要Hello\_Stub.class，因为服务器会发送给它，但是Hello.class文件客户端是省不了的，必须有。表面 上我们的客户端代码在操纵Hello，但别忘记了Hello只是一个接口，抽象的，实质上是在操纵Hello\_Stub。

拿Weblogic上的EJB举例子，10个class分别是：

Bean类：HelloBean （用户编写）

Bean类的Weblogic实现类：HelloBean\_Impl （EJBC生成）

Home接口：HelloHome （用户编写）

Home接口的Weblogic实现类 HelloBean\_HomeImpl（EJBC生成）

Home接口的Weblogic实现类的stub类 HelloBean\_HomeImpl\_WLStub（部署的时候动态生成字节码）

Home接口的Weblogic实现类的skeleton类 HelloBean\_HomeImpl\_WLSkeleton（部署的时候动态生成字节码）

Remote接口： Hello （用户编写）

Remote接口的Weblogic实现类 HelloBean\_EOImpl（EJBC生成）

Remote接口的Weblogic实现类的stub类 HelloBean\_EOImpl\_WLStub（部署的时候动态生成字节码）

Remote接口的Weblogic实现类的skeleton类 HelloBean\_EOImpl\_WLSkeleton（部署的时候动态生成字节码）

客户端只需要Hello.class和HelloHome.class这两个文件。

HelloHome home = (Home) PortableRemoteObject.narrow(ctx.lookup("Hello"), HelloHome.class);

这一行代码是从JNDI获得Home接口，但是请记住！接口是抽象的，那么home这个对象到底是什么类的对象实例呢？很简单，用toString()输出看一下就明白了，下面一行是输出结果：

HelloBean\_HomeImpl\_WLStub@18c458

这表明home这个通过从服务器的JNDI树上查找获得的对象实际上是HelloBean\_HomeImpl\_WLStub类的一个实例。

接下来客户端代码：

Hello h = home.create()

同样Hello只是一个抽象的接口，那么h对象是什么东西呢？打印一下：

HelloBean\_EOImpl\_WLStub@8fa0d1

原来是HelloBean\_EOImpl\_WLStub的一个对象实例。

用这个例子来简述一遍EJB调用过程：

首先客户端JNDI查询，服务端JNDI树上Hello这个名字实际上绑定的对象是HelloBean\_HomeImpl\_WLStub，所以服务端将创建HelloBean\_HomeImpl\_WLStub的一个对象实例，序列化返回给客户端。

于 是客户端得到home对象，表面上是得到HelloHome接口的实例，实际上是进行了一次远程调用得到了 HelloBean\_HomeImpl\_WLStub类的对象实例，别忘记了HelloBean\_HomeImpl\_WLStub也实现了 HelloHome接口。

然后home.create()实质上就是 HelloBean\_HomeImpl\_WLStub.create()，该方法将发送信息给 HelloBean\_HomeImpl\_WLSkeleton，而HelloBean\_HomeImpl\_WLSkeleton接受到信息后，再去调用 HelloBean\_HomeImpl的create方法，至此完成第1次完整的RMI循环。

注意在这次RMI循环过程中，远程对象 是HelloBean\_HomeImpl，远程对象的接口是HelloHome，对象的stub是HelloBean\_HomeImpl\_WLStub， 对象的skeleton是HelloBean\_HomeImpl\_WLSkeleton。

然后HelloBean\_HomeImpl 再去调用HelloBean\_Impl的ejbCreate方法，而HelloBean\_Impl的ejbCreate方法将负责创建或者分配一个 Bean实例，并且创建一个HelloBean\_EOImpl\_WLStub的对象实例。

这一步比较有趣的是，在前一步RMI循环 中，远程对象HelloBean\_HomeImpl在客户端有一个代理类HelloBean\_HomeImpl\_WLStub，但在这一步， HelloBean\_HomeImpl自己却充当了HelloBean\_Impl的代理类，只不过HelloBean\_HomeImpl不在客户端，而是 在服务端，因此不进行RMI。

然后HelloBean\_EOImpl\_WLStub的对象实例序列化返回给客户端，这一步也很有趣， 上次RMI过程，主角是HelloBean\_HomeImpl和它的代理类HelloBean\_HomeImpl\_WLStub，但这这一次换成了 HelloBean\_EOImpl和它的代理类HelloBean\_EOImpl\_WLStub来玩了。

Hello h = home.create();h.helloWorld();

假设Hello接口有一个helloWorld远程方法，那么表面上是在调用Hello接口的helloWorld方法，实际上是在调用HelloBean\_EOImpl\_WLStub的helloWorld方法。

然 后HelloBean\_EOImpl\_WLStub的helloWorld方法将发送信息给服务器上的 HelloBean\_EOImpl\_WLSkeleton，而HelloBean\_EOImpl\_WLSkeleton收到信息以后，再去调用 HelloBean\_EOImpl的helloWorld方法。至此，完成第2次完整的RMI循环过程。

在刚才 HelloBean\_EOImpl是作为远程对象被调用的，它的代理类是HelloBean\_EOImpl\_WLStub，但现在 HelloBean\_EOImpl要作为HelloBean\_Impl的代理类了。现在HelloBean\_EOImpl去调用 HelloBean\_Impl的helloWorld方法。注意！HelloBean\_Impl继承了HelloBean，而HelloBean中的 helloWorld方法是我们亲自编写的代码，现在终于调用到了我们编写的代码了！

至此，一次EJB调用过程终于完成。在整个过程 中，服务端主要要调用的类是HelloBean\_Impl， HelloBean\_HomeImpl，HelloBean\_HomeImpl\_WLSkeleton，HelloBean\_EOImpl， HelloBean\_EOImpl\_WLSkeleton。客户端主要调用的类是HelloBean\_HomeImpl\_WLStub， HelloBean\_EOImpl\_WLStub，这两个类在客户端代码中并不会直接出现，出现在代码中的类是他们的接口HelloHome和 Hello，因此客户端需要这两个接口文件，而Stub是服务器传送给他们的。

http://www.pbase.com/nobo123/image/27229257

http://forum.javaeye.com/viewtopic.php?t=3832&postdays=0&postorder=asc&start=0

# 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

# 请对以下在J2EE中常用的名词进行解释(或简单描述)

web 容器：给处于其中的应用程序组件（JSP，SERVLET）提供一个环境，使JSP,SERVLET直接和容器中的环境变量接接口互，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。

Web container：实现J2EE体系结构中Web组件协议的容器。这个协议规定了一个Web组件运行时的环境，包括安全，一致性，生命周期管理，事务， 配置和其它的服务。一个提供和JSP和J2EE平台APIs界面相同服务的容器。一个Web container 由Web服务器或者J2EE服务器提供。

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。

一个实现了J2EE体系结构中EJB组件规范的容器。

这个规范指定了一个Enterprise bean的运行时环境，包括安全，一致性，生命周期，事务，

配置，和其他的服务。

JNDI：（Java Naming & Directory Interface）JAVA命名目录服务。主要提供的功能是：提供一个目录系统，让其它各地的应用程序在其上面留下自己的索引，从而满足快速查找和定位分布式应用程序的功能。

JMS：（Java Message Service）JAVA消息服务。主要实现各个应用程序之间的通讯。包括点对点和广播。

JTA：（Java Transaction API）JAVA事务服务。提供各种分布式事务服务。应用程序只需调用其提供的接口即可。

JAF：（Java Action FrameWork）JAVA安全认证框架。提供一些安全控制方面的框架。让开发者通过各种部署和自定义实现自己的个性安全控制策略。

RMI/IIOP: （Remote Method Invocation /internet对象请求中介协议）他们主要用于通

# StringBuffer是线程安全的，而StringBuilder是非线程安全的。

ps：线程安全会带来额外的系统开销，所以StringBuilder的效率比StringBuffer高。如果对系统中的线程是否安全很掌握，可用StringBuffer，在线程不安全处加上关键字Synchronize。

# Vector,ArrayList, LinkedList的区别是什么？

答：1、Vector、ArrayList都是以类似数组的形式存储在内存中，LinkedList则以链表的形式进行存储。

2、List中的元素有序、允许有重复的元素，Set中的元素无序、不允许有重复元素。

3、Vector线程同步，ArrayList、LinkedList线程不同步。

4、LinkedList适合指定位置插入、删除操作，不适合查找；ArrayList、Vector适合查找，不适合指定位置的插入、删除操作。

5、ArrayList在元素填满容器时会自动扩充容器大小的50%，而Vector则是100%，因此ArrayList更节省空间。

# 什么是队列？

队列（Queue）是一种特殊的线性表，它按照先进先出的原则存储数据。

在java5中新增加了java.util.Queue接口，用以支持队列的常见操作。Queue接口与List、Set同一级别，都是继承了Collection接口。  
Queue使用时要尽量避免Collection的add()和remove()方法，而是要使用offer()来加入元素，使用poll()来获取并移出元素。它们的优  
点是通过返回值可以判断成功与否，add()和remove()方法在失败的时候会抛出异常。 如果要使用前端而不移出该元素，使用  
element()或者peek()方法。  
值得注意的是LinkedList类实现了Queue接口，因此我们可以把LinkedList当成Queue来用。

LinkedList实现了Queue接口。Queue接口窄化了对LinkedList的方法的访问权限（即在方法中的参数类型如果是Queue时，就完全只能访问Queue接口所定义的方法 了，而不能直接访问 LinkedList的非Queue的方法），以使得只有恰当的方法才可以使用。BlockingQueue 继承了Queue接口

# List 与 Set 的区别？

（1）、List,Set都是继承自Collection接口  
（2）、List特点：元素有放入顺序，元素可重复 ，Set特点：元素无放入顺序，元素不可重复（注意：元素虽然无放入顺序，但是元素在set中的位置是有该元素的HashCode决定的，其位置其实是固定的）   
（3）、List接口有三个实现类：LinkedList，ArrayList，Vector ，Set接口有两个实现类：HashSet(底层由HashMap实现)，LinkedHashSet

# 数据库的三大范式？

第一范式(1NF)：字段具有原子性,不可再分。所有关系型数据库系统都满足第一范式。

数据库表中的字段都是单一属性的，不可再分。例如，姓名字段，其中的姓和名必须作为一个整体，无法区分哪部分是姓，哪部分是名，如果要区分出姓和名，必须设计成两个独立的字段。

第二范式(2NF)是在第一范式(1NF)的基础上建立起来的，即满足第二范式(2NF)必须先满足第一范式(1NF)。

要求数据库表中的每个实例或行必须可以被惟一地区分。通常需要为表加上一个列，以存储各个实例的惟一标识。这个惟一属性列被称为主关键字或主键。

第二范式(2NF)要求实体的属性完全依赖于主关键字。所谓完全依赖是指不能存在仅依赖主关键字一部分的属性，如果存在，那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体，新实体与原实体之间是一对多的关系。为实现区分通常需要为表加上一个列，以存储各个实例的惟一标识。简而言之，第二范式就是非主属性非部分依赖于主关键字。

第三范式的要求如下：

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

所以第三范式具有如下特征：

1. 每一列只有一个值
2. 每一行都能区分。
3. 每一个表都不包含其他表已经包含的非主关键字信息。

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

# java 实现非递归二分法算法。

（1）、前提：二分查找的前提是需要查找的数组必须是已排序的，我们这里的实现默认为升序

（2）、原理：将数组分为三部分，依次是中值（所谓的中值就是数组中间位置的那个值）前，中值，中值后；将要查找的值和数组的中值进行比较，若小于中值 则在中值前面找，若大于中值则在中值后面找，等于中值时直接返回。然后依次是一个递归过程，将前半部分或者后半部分继续分解为三部分。可能描述得不是很清 楚，若是不理解可以去网上找。从描述上就可以看出这个算法适合用递归来实现，可以用递归的都可以用循环来实现。所以我们的实现分为递归和循环两种，可以根据代码来理解算法

（3）、实现：代码如下

/\*\*

\* 循环二分查找，返回第一次出现该值的位置

\* **@param** sortedData 已排序的数组

\* **@param** findValue 需要找的值

\* **@return** 值在数组中的位置，从0开始。找不到返回-1

\*/

**public** **int** searchLoop(**int**[] sortedData,**int** findValue)

{

**int** start=0;

**int** end=sortedData.length-1;

**while**(start<=end)

{

//中间位置

**int** middle=(start+end)>>1; //相当于(start+end)/2

//中值

**int** middleValue=sortedData[middle];

**if**(findValue==middleValue)

{

//等于中值直接返回

**return** middle;

}

**else** **if**(findValue<middleValue)

{

//小于中值时在中值前面找

end=middle-1;

}

**else**

{

//大于中值在中值后面找

start=middle+1;

}

}

//找不到

**return** -1;

}

# Java 写一个冒泡程序

比较相邻的元素。如果第一个比第二个大（小），就交换他们两个。对每一对相邻元素作同样的工作，从开始第一对到结尾的最后一对。在这一点，最后的元素应该会是最大的数。针对所有的元素重复以上的步骤，除了最后一个。持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。

**public** **void** bubbleSort() {

**int** score[] = { 67, 69, 75, 87, 89, 90, 99, 100 };

**for** (**int** i = 0; i < score.length - 1; i++) { // 最多做n-1趟排序

**for** (**int** j = 0; j < score.length - i - 1; j++) { // 对当前无序区间score[0......length-i-1]进行排序(j的范围很关键，这个范围是在逐步缩小的)

**if** (score[j] < score[j + 1]) { // 把小的值交换到后面

**int** temp = score[j];

score[j] = score[j + 1];

score[j + 1] = temp;

}

}

System.*out*.print("第" + (i + 1) + "次排序结果：");

**for** (**int** a = 0; a < score.length; a++) {

System.*out*.print(score[a] + "\t");

}

System.*out*.println("");

}

System.*out*.print("最终排序结果：");

**for** (**int** a = 0; a < score.length; a++) {

System.*out*.print(score[a] + "\t");

}

}

# 多线程

（1）、先理解这几概念：

a1.程序:指令和数据的byte序列,eg:qq.exe;

a2.进程:正在运行的程序(如QQ);

a3.一个进程中可能有一到多个线程.

线程的概念：Thread 每个正在系统上运行的程序都是一个进程。每个进程包含一到多个线程。进程也可能是整个程序或者是部分程序的动态执行。线程是一组指令的集合，或者是程序的特殊段，它可以在程序里独立执行。也可以把它理解为代码运行的上下文。所以线程基本上是轻量级的进程，它负责在单个程序里执行多任务。通常由操作系统负责多个线程的调度和执行。

多线程的概念：多线程是为了同步完成多项任务，不是为了提高运行效率，而是为了提高资源使用效率来提高系统的效率。多线程是在同一时间需要完成多项任务的时候实现的。多线程的优点：使用线程可以把占据长时间的程序中的任务放到后台去处理。

（2）、在java中要想实现多线程，有两种手段，一种是继续Thread类，另外一种是实现Runable接口。

对于直接继承Thread的类来说，代码大致框架是：

class 类名 extends Thread{

方法1;

方法2；

…

public void run(){

// other code…

}

属性1；

属性2；

…

}

运行时，调用的是start（）方法。

通过实现Runnable接口，大致框架是：

class 类名 implements Runnable{

方法1;

方法2；

…

public void run(){

// other code…

}

属性1；

属性2；

…

}

总结：实现Runnable接口比继承Thread类所具有的优势：

1）：适合多个相同的程序代码的线程去处理同一个资源

2）：可以避免java中的单继承的限制

3）：增加程序的健壮性，代码可以被多个线程共享，代码和数据独立。

# java 如何实现高并发？

尽量使用缓存，包括用户缓存，信息缓存等，多花点内存来做缓存，可以大量减少与数据库的交互，提高性能。

用jprofiler等工具找出性能瓶颈，减少额外的开销。

优化数据库查询语句，减少直接使用hibernate等工具的直接生成语句（仅耗时较长的查询做优化）。

优化数据库结构，多做索引，提高查询效率。

统计的功能尽量做缓存，或按每天一统计或定时统计相关报表，避免需要时进行统计的功能。

能使用静态页面的地方尽量使用，减少容器的解析（尽量将动态内容生成静态html来显示）。

解决以上问题后，使用服务器集群来解决单台的瓶颈问题。

基本上以上述问题解决后，达到系统最优。

# HashMap、 TreeMap 区别，以及 TreeMap 原理

区别：

（1）、HashMap：基于哈希表实现。使用HashMap要求添加的键类明确定义了hashCode()和equals()[可以重写hashCode()和equals()]，为了优化HashMap空间的使用，您可以调优初始容量和负载因子。

TreeMap：基于红黑树实现。TreeMap没有调优选项，因为该树总处于平衡状态。

（2）、HashMap：适用于在Map中插入、删除和定位元素。Treemap：适用于按自然顺序或自定义顺序遍历键(key)。

HashMap通过hashcode对其内容进行快速查找，而 TreeMap中所有的元素都保持着某种固定的顺序，如果你需要得到一个有序的结果你就应该使用TreeMap（HashMap中元素的排列顺序是不固定的）。 HashMap通常比TreeMap快一点(树和哈希表的数据结构使然)，建议多使用HashMap，在需要排序的Map时候才用TreeMap。

（3）、HashMap 非线程安全 TreeMap 非线程安全。