JAVA 相关基础知识

1、面向对象的特征有哪些方面

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。

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。为了提高效率节省空间，我们应该用 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。

5、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。

6、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。

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。

10、&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）。

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（如果是 ArrayList：List lst = Collections.synchronizedList(new ArrayList());如果是 HashMap：Map map = Collections.synchronizedMap(new HashMap());）。

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?
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。

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。

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。

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?

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.

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

两个

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。

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();
}
}
}
}
```

29、Java 有没有 goto?

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()?

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。

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。
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。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。
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?

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。

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?

2 << 3

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?

不能，一个对象的一个 synchronized 方法只能由一个线程访问。

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: 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

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?

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

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

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。

59、JSP 的常用指令

<%@page language="java" contenType="text/html;charset=gb2312" session="true" buffer="64kb" autoFlush="true" isThreadSafe="true" info="text" errorPage="error.jsp" isErrorPage="true" isELIgnored="true"

pageEncoding=”gb2312” import=”java.sql.*”%>

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

<%@include file=”filename”%>

<%@taglib prefix=”c”uri=”http://……”%>

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

<%@ 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)
web 容器：给处于其中的应用程序组件（JSP，SERVLET）提供一个环境，使 JSP,SERVLET 直接更容器中的环境变量接口交互，不必关注其它系统问题。主要有 WEB 服务器来实现。例如：TOMCAT,WEBLOGIC,WEBSPHERE 等。该容器提供的接口严格遵守 J2EE 规范中的 WEB APPLICATION 标准。我们把遵守以上标准的 WEB 服务器就叫做 J2EE 中的 WEB 容器。
EJB 容器：Enterprise java bean 容器。更具有行业领域特色。他提供给运行在其中的组件 EJB 各种管理功能。只要满足 J2EE 规范的 EJB 放入该容器，马上就会被容器进行高效率的管理。并且可以通过现成的接口来获得系统级别的服务。例如邮件服务、事务管理。
JNDI：（Java Naming & Directory Interface）JAVA 命名目录服务。主要提供的功能是：提供一个目录系统，让其它各地的应用程序在其上面留下自己的索引，从而满足快速查找和定位分布式应用程序的功能。
JMS：（Java Message Service）JAVA 消息服务。主要实现各个应用程序之间的通讯。包括点对点和广播。
JTA：（Java Transaction API）JAVA 事务服务。提供各种分布式事务服务。应用程序只需调用其提供的接口即可。
JAF：（Java Action FrameWork）JAVA 安全认证框架。提供一些安全控制方面的框架。让开发者通过各种部署和自定义实现自己的个性安全控制策略。
RMI/IIOP:（Remote Method Invocation /internet 对象请求中介协议）他们主要用于通过远程调用服务。例如，远程有一台计算机上运行一个程序，它提供股票分析服务，我们可以在本地计算机上实现对其直接调用。当然这是要通过一定的规范才能在异构的系统之间进行通信。RMI 是 JAVA 特有的。
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。

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

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?

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)?

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 ServiceWeb 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 代码查错

1.

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

大侠们，这有何错误?

答案: 错。abstract method 必须以分号结尾，且不带花括号。

2.

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

有错吗?

答案: 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.

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

这好像没什么错吧?

答案: 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.

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

这个比较明显。

答案: 错。int x 被修饰成 final，意味着 x 不能在 addOne method 中被修改。

5.

```
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;
}
```

和上面的很相似，都是关于 final 的问题，这有错吗?

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.

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

有什么错呢? 看不出来啊。

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.

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

和上面一题只有一个地方不同，就是多了一个 final。这难道就错了吗?

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.

```
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 ...";
}
}
```

看上去很完美。

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 ...");
}
}
```

这个好像很明显。

答案: 正确。从来没有人说过 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。

10.

```
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.

```
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());
}
}
```

这个错误不容易发现。

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。

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;i<s.length;i++){
ii[i] =Integer.parseInt(s[i]);
}
Arrays.sort(ii);
//asc
for(int i=0;i<s.length;i++){
System.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);
cMoneyStringBuffer.insert(cMoneyStringBuffer.length() - 1, JIAO);
```

```
cMoneyStringBuffer.insert(cMoneyStringBuffer.length(), FEN);
if (cMoneyStringBuffer.indexOf("零角零分") != -1)//没有零头，加整
cMoneyStringBuffer.replace(cMoneyStringBuffer.indexOf("零角零分"),
cMoneyStringBuffer.length(), "整");
else
if (cMoneyStringBuffer.indexOf("零分") != -1)//没有零分，加整
cMoneyStringBuffer.replace(cMoneyStringBuffer.indexOf("零分"),
cMoneyStringBuffer.length(), "整");
else {
if(cMoneyStringBuffer.indexOf("零角")!=-1)
cMoneyStringBuffer.delete(cMoneyStringBuffer.indexOf("零角
"),cMoneyStringBuffer.indexOf("零角")+2);
// tmpBuffer.append("整");
}
result = cMoneyStringBuffer.toString();
return result;
}

private void checkPrecision(String moneyStr) {
int fractionDigits = moneyStr.length() - moneyStr.indexOf(DOT) - 1;
if (fractionDigits > 2)
throw new RuntimeException("金额" + moneyStr + "的小数位多于两位。"); //精度不
能比分低
}

public static void main(String args[]) {
System.out.println(getInstance().format(new Double(10010001.01)));
}
}
```

3、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
```
4、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：

一.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

二.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；return d1 出错。故 D 错

三.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

四.答案为 C、E

5、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:"+Remo
```

##

###【第二部分：难度比较大】###

##

某公司 Java 面试题及部分解答（难度较大）

1。请大概描述一下 Vector 和 ArrayList 的区别，Hashtable 和 HashMap 的区别。(5)

2。请问你在什么情况下会在你的 JAVA 代码中使用可序列化？(5)

为什么放到 HttpSession 中的对象必须要是可序列化的？(5)

3。为什么在重写了 equals()方法之后也必须重写 hashCode()方法？(10)

4。sleep()和 wait()有什么区别？(10)

5。编程题：用最有效率的方法算出 2 乘以 17 等于多少？(5)

6。JAVA 是不是没有内存泄漏问题？看下面的代码片段，并指出这些代码隐藏的问题。(10)

```
Object[] elements = new Object[10];
int size;
...
public Object pop() {
if (size == 0)
return null;
Object o = elements[--size];
return o;
}
```

7。请阐述一下你对 JAVA 多线程中“锁”的概念的理解。(10)

8。所有的递归实现都可以用循环的方式实现，请描述一下这两种实现方式各自的优劣。
并举例说明在什么情况下可以使用递归，而在什么情况下只能使用循环而不能使用递归？(5)

9。请简要讲一下你对测试驱动开发（TDD）的认识。(10)

10。请阐述一下你对“面向接口编程”的理解。(10)

11。在 J2EE 中有一个“容器（Container）”的概念，不管是 EJB、PICO 还是 Spring 都有他们
各自实现的容器，受容器管理的组件会具有有生命周期的特性，请问，为什么需要容器？
它的好处在哪里？它会带来什么样的问题？(15)

12。请阐述一下你对 IOC（Inversion of Control）的理解。（可以以 PICO 和 Spring 的 IOC 作为例子说明他们在实现上各自的特点）(10)

13。下面的代码在绝大部分时间内都运行得很正常，请问在什么情况下会出现问题？问题的根源在哪里？(10)

```
import java.util.LinkedList;

public class Stack {
```

```
LinkedList list = new LinkedList();

public synchronized void push(Object x) {
synchronized(list) {
list.addLast( x );
notify();
}
}

public synchronized Object pop()
throws Exception {
synchronized(list) {
if( list.size() <= 0 ) {
wait();
}
return list.removeLast();
}
}
}
```

解答：

。请大概描述一下 Vector 和 ArrayList 的区别，Hashtable 和 HashMap 的区别。(5)线程安全与否

2。请问你在什么情况下会在你的 JAVA 代码中使用可序列化？(5)cluster 中 session 复制,缓存 persist 与 reload
为什么放到 HttpSession 中的对象必须要是可序列化的？(5)没必须,不过 session 反序列化过程会导致对象不可用.

3。为什么在重写了 equals()方法之后也必须重写 hashCode()方法？(10)API 规范

4。sleep()和 wait()有什么区别？(10)前者占用 CPU,后者空闲 CPU

5。编程题：用最有效率的方法算出 2 乘以 17 等于多少？(5)17>>1

6。JAVA 是不是没有内存泄漏问题？看下面的代码片段，并指出这些代码隐藏的问题。
(10)不是
...
...没发现内存泄漏的问题

7。请阐述一下你对 JAVA 多线程中"锁"的概念的理解。(10)同步因子,在某段代码上增加同步因子,那么整个 JVM 内部只能最多有一个线程执行这段,其余的线程按 FIFO 方式等待执行.

8。所有的递归实现都可以用循环的方式实现，请描述一下这两种实现方式各自的优劣。
并举例说明在什么情况下可以使用递归，而在什么情况下只能使用循环而不能使用递归？
(5)没发现所有的递归都可以用循环实现的,尤其是那种不知道循环重数的递归算法.递归的优点是简炼,抽象性好;循环则更直观.递归一般用于处理一级事务能转化成更简的二级事务的操作.归纳不出二级事务或者二级事务更复杂的情况不能用.

9。请简要讲一下你对测试驱动开发（TDD）的认识。(10)不认识

10。请阐述一下你对"面向接口编程"的理解。(10)1,利于扩展;2,暴露更少的方法;

11。在 J2EE 中有一个"容器（Container）"的概念，不管是 EJB、PICO 还是 Spring 都有他们
各自实现的容器，受容器管理的组件会具有有生命周期的特性，请问，为什么需要容器？
它的好处在哪里？它会带来什么样的问题？(15)组件化,框架设计...

12。请阐述一下你对 IOC（Inversion of Control）的理解。（可以以 PICO 和 Spring 的 IOC 作为例子说明他们在实现上各自的特点）(10)不理解

13。下面的代码在绝大部分时间内都运行得很正常，请问在什么情况下会出现问题？问题的根源在哪里？(10)wait 和 notify 使用目的不能达到,wait()的 obj,自身不能 notify().出题人对 wait 和 notify 机制不够理解.

```
import java.util.LinkedList;

public class Stack {

LinkedList list = new LinkedList();

public synchronized void push(Object x) {
synchronized(list) {
```

```
list.addLast( x );
notify();
}
}

public synchronized Object pop()
throws Exception {
synchronized(list) {
if( list.size() <= 0 ) {
wait();
}
return list.removeLast();
}
}
}
```

你拿了多少分？

1。请大概描述一下 Vector 和 ArrayList 的区别，Hashtable 和 HashMap 的区别。(5)

// thread-safe or unsafe, could contain null values or not

2。请问你在什么情况下会在你的 JAVA 代码中使用可序列化？(5)
为什么放到 HttpSession 中的对象必须要是可序列化的？(5)
// save, communicate

3。为什么在重写了 equals()方法之后也必须重写 hashCode()方法？(10)

// implementations of dictionaries need hashCode() and equals()

4。sleep()和 wait()有什么区别？(10)

// threads communication: wait() and notifyAll()

5。编程题：用最有效率的方法算出 2 乘以 17 等于多少？(5)

// 2<<4+2

6。JAVA 是不是没有内存泄漏问题？看下面的代码片段，并指出这些代码隐藏的问题。(10)

```
...
Object[] elements = new Object[10];
int size;
...
public Object pop() {
if (size == 0)
return null;
Object o = elements[--size];
return o;
}
```

// elements[size] = null;

7。请阐述一下你对 JAVA 多线程中"锁"的概念的理解。(10)

// optimistic lock, pessimistic lock, signal, dead lock, starvation, synchronization

8。所有的递归实现都可以用循环的方式实现，请描述一下这两种实现方式各自的优劣。
并举例说明在什么情况下可以使用递归，而在什么情况下只能使用循环而不能使用递归？(5)

// recursive: when you need a stack and stack memory is enough
// non-recursive: when you need a queue

9。请简要讲一下你对测试驱动开发（TDD）的认识。(10)

// write unit testing code first

10。请阐述一下你对"面向接口编程"的理解。(10)

// adapter, listener, bridge, decorator, proxy... patterns

11。在 J2EE 中有一个"容器（Container）"的概念，不管是 EJB、PICO 还是 Spring 都有他们

各自实现的容器，受容器管理的组件会具有有生命周期的特性，请问，为什么需要容器？它的好处在哪里？它会带来什么样的问题？(15)

```
// encapsulation
```

12。请阐述一下你对 IOC（Inversion of Control）的理解。（可以以 PICO 和 Spring 的 IOC 作为例子说明他们在实现上各自的特点）(10)

```
// reduce classes' dependencies
```

13。下面的代码在绝大部分时间内都运行得很正常，请问在什么情况下会出现问题？问题的根源在哪里？(10)

```
import java.util.LinkedList;

public class Stack {

LinkedList list = new LinkedList();

public synchronized void push(Object x) {
synchronized(list) {
list.addLast( x );
notify();
}
}

public synchronized Object pop()
throws Exception {
synchronized(list) {
if( list.size() <= 0 ) {
wait();
}
return list.removeLast();
}
}
}

// dead lock, synchronized on both 'list' and 'this'
```