1、overload和override的区别  
overload：  
方法的重载  
只在方法之间发生  
方法名相同，并且区分大小写  
参数列表不同，具体是指参数的类型，个数，顺序不同  
返回值类型没有要求，可以相同，也可以不同  
override：  
方法的覆盖  
发生在父子类之间  
方法名称，参数表，返回值都相同  
修饰符一样或者更宽  
静态方法只能被静态方法覆盖，没有多态  
子类不能抛出比父类更多的异常<指的是范围>

2、接口和抽象类的区别  
抽象类（abstract-class）:  
1).抽象类不能创建对象，只可以声明引用  
2).抽象类有构造函数  
3).抽象类里面可以有抽象方法，也可以没有  
4).如果一个类里面有抽象方法，则该类是抽象类  
5).如果一个类继承抽象类，则该类不想成为抽象类，则必须实现抽象类里面的所有抽象方法  
接口（interface）：  
是一个特殊的抽象类  
修饰属性：

3、string 和StringBuffer StringBuilder的区别  
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；线程安全；  
StringBuilder是从 JDK 5 开始，为StringBuffer该类补充了一个单个线程使用的等价类；  
通常应该优先使用 StringBuilder类，因为它支持所有相同的操作，但由于它不执行同步，所以速度更快。

4、arrayList linkedList vector 的区别  
ArrayList 和 Vector 都是使用数组方式存储数据，此数组元素数大于实际存储的数据以便增加和插入元素，它们都允许直接按序号索引元  
素，但是插入元素要涉及数组元素移动等内存操作，所以索引数据快而插入数据慢，Vector 由于使用了 synchronized 方法（线程安全），  
通常性能上较 ArrayList 差，而 LinkedList 使用双向链表实现存储，按序号索引数据需要进行前向或后向遍历，但是插入数据时只需要记  
录本项的前后项即可，所以插入速度较快。  
LinkedList 也是线程不安全的，LinkedList 提供了一些方法，使得  
LinkedList 可以被当作堆栈和队列来使用。

5、list和set 的区别  
list:有序，有下标，可重复，实现类有三种：ArrayList、Vector、LinkedList，遍历方式有三种：for循环遍历、迭代器遍历、foreach循环遍历  
ArrayList、Vector底层由数组实现，默认数组长度为0  
set：无序，无下标，不可重复，实现类为HashSet，遍历方式有两种：迭代器遍历、foreach循环遍历

6、写一个singleton类。  
单例模式 一个类只能有一个实例对象  
a 饿汉式  
b 懒汉式

package entity;  
public class Test {  
public static void main(String[] args) {  
// Student s= new Student();  
Student s1 = Student.getStudent();  
Student s2 = Student.getStudent();  
System.out.println(s1==s2);  
}  
}  
//饿汉式 类加载时就被创建对象  
//class Student{  
// private static Student s = new Student();  
// private Student(){}  
// public static Student getStudent(){  
// return s;  
// }  
//}  
//懒汉式 锁类对象  
class Student{  
private static Student s;  
private Student(){}  
public synchronized static Student getStudent(){  
if(s==null){  
s = new Student();  
}  
return s;  
}  
}

7、数据库连接池的工作机制  
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。实现方式，返回的 Connection是原始Connection的代理，代理Connection的close方法不是真正关连接，而是把它  
代理的 Connection 对象还回到连接池中。

8、说出servlet的生命周期  
servlet是单例的，servlet是当第一次发出请求时实例化一个servlet。  
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。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。

10、struts mvc的介绍  
M(Model)层,就相当于后台，V(View),相当于前台，C(Controller)层，相当于控制页面跳转；MVC 模式的目的就是实现 Web 系统的职能分工。  
Model层实现系统中的业务逻辑，通常可以用 JavaBean 或 EJB来实现。  
View 层用于与用户的交互，通常用 JSP 来实现。 Controller层是Model与View之间沟通的桥梁，它可以分派用户的请求并选择恰当的视图以用于显示，同时它也可以解释用户的输入并将它们映射为模型层可执行的操作。  
各个层的实现技术举例：  
M(Model)javaBean 中 DAO+Service 构成 Model 层一般由hibernate/mybatis 技术实现。模型是应用程序的主体部分。模型表示业务数据，或者业务逻辑.  
C(servlet/struts action/spring action) Struts/SpringMVC 等技术。控制器工作就是根据用户的输入，控制用户界面数据显示和更新 model对象状态。  
V(View) jsp/FreeMarker 等技术，视图是应用程序中用户界面相关的部分，是用户看到并与之交互的界面。

MVC的好处：分层开发，便于系统的升级

对于一个新的框架，如何学习？  
首先知道该框架是干嘛的，能解决什么问题。

11、介绍hibernate  
1)、hibernate 是一个开源的、采用面向对象的思想实现 ORM 映射框架，它对 jdbc 进行了一层封装，对于数据库的连接、关闭、数据的持久化（增删改查）、事务的管理都进行了封装，使得程序开发的时候可以用面向对象的思想来操作数据库。  
2)、hibernate 可以用于 jdbc 应用的任何场合。优点是，使用面向对象的思想操作数据库；对 jdbc 的封装，使得开发中不需要写大量的sql 语句，大大提高了开发效率；采用数据库方言等方式使得数据库移植十分方便，也使得在开发中不需要考虑数据之间的差异；提供缓存机制，提高查询效率等。  
3)、缺点是，hibernate 并不支持批量操作。对 jdbc 的封装失去了对sql 的控制，而如果执行本地 sql 则失去了 hibernate 的可移植性的优势。消耗内存较大。

12、spring 的主要作用。  
SSH框架的流程：Jsp页面——struts——Service（业务逻辑处理类）——Hibernate（左到右）struts负责控制Service的声明周期，这样层与层之间的依赖性强，属于耦合。这时，使用spring框架就起到了控制Action和Service类的作用，由于上层不再控制下层的生命周期，层与层之间实现了完全脱耦，使程序运行起来效果更高，维护起来也方便。

总的来说，spring的作用是解耦合（IOC—翻转控制或者DI—依赖注入）和事务的处理（即AOP应用）  
spring的IOC（控制反转和依赖注入）  
控制反转：就是由容器控制程序之间的（依赖）关系，而非传统实现中，由程序员代码直接操控（由new关键字来生成对象）  
依赖注入：组件之间的依赖关系由容器在运行期决定，由容器动态的将某种依赖注入到组件之中。

事务的处理： 在以往的JDBCTemplate中事务提交成功，异常处理都是通过Try/Catch来完成，而在Spring中，容器集成了TransactionTemplate，它封装了所有对事务处理的功能，包括异常时事务回滚，操作成功时数据提交等复杂业务功能。Hibernate中也有对事务的管理，Hibernate中事务管理是通过SessionFactory创建和维护session来完成，而Spring对SessionFactory也进行了整合，不需要再通过配置hibernate.cfg.xml来对SessionFactory进行设定。这样可以很好的利用Spring对事务管理的强大功能。避免了每次对数据操作都要获得Session实例来启动事务/提交/回滚事务还有繁琐的Try/Catch操作。这些也就是Spring中的AOP机制很好的应用。一方面使业务开发逻辑更清晰、专业分工更加容易进行。另一方面就是应用Spring-AOP隔离降低了程序的耦合性使我们可以在不同的应用中将各个切面结合起来使用，大大提高了代码重用度。

13、HashTable和HashMap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，在只有一个线程访问的情况下，效率要高于Hashtable。  
HashMap允许将null作为一个entry的key或者value，而Hashtable不允许。  
HashMap 把 Hashtable 的 contains 方法去掉了，改成 containsvalue和 containsKey。因为 contains 方法容易让人引起误解。  
Hashtable继承自Dictionary类，而HashMap是Java1.2引进的Map interface 的一个实现。  
最大的不同是，Hashtable 的方法是 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。  
就 HashMap 与 HashTable 主要从三方面来说：  
一.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  
二.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  
三.值：只有 HashMap 可以让你将空值作为一个表的条目的 key 或value

14.SERVLETAPI 中 forward() 与 redirect()的区别？  
forward：请求转发  
redirect：请求重定向  
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。

15.多态：编译时多态，运行时多态  
方法的重载没有多态，是指运行时多态

16.什么是hibernate懒加载？为什么要用懒加载？  
所谓懒加载，就是延时加载或者延迟加载。  
当我们访问的数据量过大时，明显用缓存不合适，因为内容容量有限，为了减少并发量，减少系统资源的消耗，我们让数据在需要的时候才进行加载，这时我们就用到了懒加载。

hibernate提供了对属性的延迟加载  
A.实体对象的延迟加载  
如果想对实体对象使用延迟加载，必须要在实体的映射配置文件中进行相应的配置，如下所示：  
<hibernate-mapping>  
< class name=”com.neusoft.entity.User” table=”user” lazy=”true”>  
……  
</class>  
< /hibernate-mapping>  
通过将class的lazy属性设置为true，来开启实体的延迟加载特性。如果运行下面的代码：User user = (User)session.load();(1)

17.如何理解HIbernate？  
属于DAO层，叫持久层；封装了JDBC  
使用HQL语句，屏蔽了底层数据库的差异  
hibernate自身的懒加载机制  
hibernate的三种状态：瞬时态(Transient)<>、 持久态(Persistent)、脱管态(Detached)