**3. String 是基本数据类型吗？我可不可以写个类继承于**

**String？**

不是，Strng 是引用类型；String 是 final 的类，是不可以被继承的。

**10. 静态变量和实例变量的区别？**

静态变量也称为类变量，归全类共有，它不依赖于某个对象，可通过类名直接访问；而

实例变量必须依存于某一实例，只能通过对象才能访问到它。

**11. 面向对象的基本特征是什么？**

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：子类拥有父类一切非私有的属性和方法。

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：同一种事物的不同种表现形式。

**13. Overload 和 Override 的区别。**

（Overload）重载：发生在同一个类之中，方法名相同、参数列表不同，与返回值无关、与final 无关、与修饰符无关、与异常无关。

（Override）重写：发生在子类和父类之间，方法名相同、参数列表相同、返回值相同、不

能是 final 的方法、重写的方法不能有比父类方法更为严格的修饰符权限、重写的方法所抛

出的异常不能比父类的更大。

如果父类私有的方法，子类拥有方法签名相同的方法，子类不属于重写父类的方法，该方法

属于子类的新方法。

1.HashMap和HashTable区别

1. HashTable的⽅法前⾯都有synchronized来同步，是线程安全的；HashMap未经同步，是⾮线程安全的。
2. HashTable不允许null值(key和value都不可以) ；HashMap允许null值(key和value都可以)。
3. HashTable有⼀个contains(Object value)功能和containsValue(Object value)功能⼀样。
4. HashTable使⽤Enumeration进⾏遍历；HashMap使⽤Iterator进⾏遍历。
5. HashTable中hash数组默认⼤⼩是11，增加的⽅式是old\*2+1；HashMap中hash数组的默认⼤⼩是16，⽽且⼀定是2的指数。
6. 哈希值的使⽤不同，HashTable直接使⽤对象的hashCode； HashMap重新计算hash

2.hashCode和equals⽅法的关系

equals相等，hashcode必相等；hashcode相等，equals可能不相等。

3.ArrayList和LinkedList区别及使⽤场景区别

ArrayList底层是⽤数组实现的，可以认为ArrayList是⼀个可改变⼤⼩的数组。随着越来

越多的元素被添加到ArrayList中，其规模是动态增加的。

LinkedList底层是通过双向链表实现的， LinkedList和ArrayList相⽐，增删的速度较快。

但是查询和修改值的速度较慢。同时，LinkedList还实现了Queue接⼝，所以他还提供了

offer(),

peek(), poll()等⽅法。

使⽤场景

LinkedList更适合从中间插⼊或者删除（链表的特性）。

ArrayList更适合检索和在末尾插⼊或删除（数组的特性）。

4.多线程的实现⽅式

继承Thread类、实现Runnable接⼝、使⽤ExecutorService、Callable、Future实现有返

回结果的多线程。

27.如何保证线程安全

对⾮安全的代码进⾏加锁控制；

使⽤线程安全的类；

多线程并发情况下，线程共享的变量改为⽅法级的局部变量。

31.sleep和wait的区别(考察的⽅向是是否会释放锁)

sleep()⽅法是Thread类中⽅法，⽽wait()⽅法是Object类中的⽅法。

sleep()⽅法导致了程序暂停执⾏指定的时间，让出cpu该其他线程，但是他的监控状态

依然保持者，当指定的时间到了⼜会⾃动恢复运⾏状态，在调⽤sleep()⽅法的过程中，

线程不会释放对象锁。⽽当调⽤wait()⽅法的时候，线程会放弃对象锁，进⼊等待此对象

的等待锁定池，只有针对此对象调⽤notify()后本线程才进⼊对象锁定池准备。

33.如何才能产⽣死锁

产⽣死锁的四个必要条件：

⼀.互斥条件：所谓互斥就是进程在某⼀时间内独占资源。

⼆.请求与保持条件：⼀个进程因请求资源⽽阻塞时，对已获得的资源保持不放。

三.不剥夺条件:进程已获得资源，在末使⽤完之前，不能强⾏剥夺。

四.循环等待条件:

39.Java中的NIO，BIO，AIO分别是什么

BIO:同步并阻塞，服务器实现模式为⼀个连接⼀个线程，即客户端有连接请求时服务器

端就需要启动⼀个线程进⾏处理，如果这个连接不做任何事情会造成不必要的线程开

销，当然可以通过线程池机制改善。BIO⽅式适⽤于连接数⽬⽐较⼩且固定的架构，这种

⽅式对服务器资源要求⽐较⾼，并发局限于应⽤中，JDK1.4以前的唯⼀选择，但程序直

观简单易理解。

NIO:同步⾮阻塞，服务器实现模式为⼀个请求⼀个线程，即客户端发送的连接请求都会

注册到多路复⽤器上，多路复⽤器轮询到连接有I/O请求时才启动⼀个线程进⾏处理。

NIO⽅式适⽤于连接数⽬多且连接⽐较短（轻操作）的架构，⽐如聊天服务器，并发局限

于应⽤中，编程⽐较复杂，JDK1.4开始⽀持。

AIO:异步⾮阻塞，服务器实现模式为⼀个有效请求⼀个线程，客户端的I/O请求都是由OS

先完成了再通知服务器应⽤去启动线程进⾏处理.AIO⽅式使⽤于连接数⽬多且连接⽐较

⻓（重操作）的架构，⽐如相册服务器，充分调⽤OS参与并发操作，编程⽐较复杂，

JDK7开始⽀持。

cookie 和session 的区别：

1、cookie数据存放在客户的浏览器上，session数据放在服务器上。

2、cookie不是很安全，别人可以分析存放在本地的COOKIE并进行COOKIE欺骗考虑到安全应当使用session。

3、session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能考虑到减轻服务器性能方面，应当使用COOKIE。

4、单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie。

5、所以个人建议：  
   将登陆信息等重要信息存放为SESSION；其他信息如果需要保留，可以放在COOKIE中

1）继承：继承是从已有类得到继承信息创建新类的过程。提供继承信息的类被称为父类（超类、基类）；得到继承信息的类被称为子类（派生类）。继承让变化中的软件系统有了一定的延续性，同时继承也是封装程序中可变因素的重

要手段。

多态性：多态性是指允许不同子类型的对象对同一消息作出不同的响应。简单的说就是用同样的对象引用调

用同样的方法但是做了不同的事情。多态性分为编译时的多态性和运行时的多态性。如果将对象的方法视为对象向外

界提供的服务，那么运行时的多态性可以解释为：当 A 系统访问 B 系统提供的服务时，B 系统有多种提供服务的方式，但

一切对 A 系统来说都是透明的。方法重载（overload）实现的是编译时的多态性（也称为前绑定），而方法重写

（override）实现的是运行时的多态性（也称为后绑定）。运行时的多态是面向对象最精髓的东西，要实现多态需要做

两件事：1. 方法重写（子类继承父类并重写父类中已有的或抽象的方法）；2. 对象造型（用父类型引用引用子类型对

象，这样同样的引用调用同样的方法就会根据子类对象的不同而表现出不同的行为）。

1. 重载（Overload）和重写（Override）的区别。重载的方法能否根据返回类型进行区分？  
   答：方法的重载和重写都是实现多态的方式，区别在于前者实现的是编译时的多态性，而后者实现的是运行时的多态性。重载发生在一个类中，同名的方法如果有不同的参数列表（参数类型不同、参数个数不同或者二者都不同）则视为重载；重写发生在子类与父类之间，重写要求子类被重写方法与父类被重写方法有相同的参数列表，有兼容的返回类型，比父类被重写方法更好访问，不能比父类被重写方法声明更多的异常（里氏代换原则）。重载对返回类型没有特殊的要求，不能根据返回类型进行区分。

包装类型与基本数据类型的区别:

　　1. 包装类型允许为null,基本数据类型不可以;

　　2. 包装类型存放在堆中,基本数据类型存放在栈中;

　　3. 从效率上将,基本数据类型效率高于包装类型;

4. 包装类型可以做泛型，基本数据类型不可以；

过滤器，是在java web中，你传入的request,response提前过滤掉一些信息，或者提前设置一些参数，然后再传入servlet或者struts的 action进行业务逻辑，比如过滤掉非法url（不是login.do的地址请求，如果用户没有登陆都过滤掉）,或者在传入servlet或者 struts的action前统一设置字符集，或者去除掉一些非法字符

拦截器，是在面向切面编程的就是在你的service或者一个方法，前调用一个方法，或者在方法后调用一个方法比如动态代理就是拦截器的简单实现，在你调用方法前打印出字符串（或者做其它业务逻辑的操作），也可以在你调用方法后打印出字符串，甚至在你抛出异常的时候做业务逻辑的操作。

过滤器和拦截器的区别

1.拦截器是基于Java的反射机制的，而过滤器是基于函数回调。

2.拦截器不依赖与servlet容器，过滤器依赖与servlet容器。

3.拦截器只能对action请求起作用，而过滤器则可以对几乎所有的请求起作用。

4.拦截器可以访问action上下文、值栈里的对象，而过滤器不能访问。

5.在action的生命周期中，拦截器可以多次被调用，而过滤器只能在容器初始化时被调用一次。