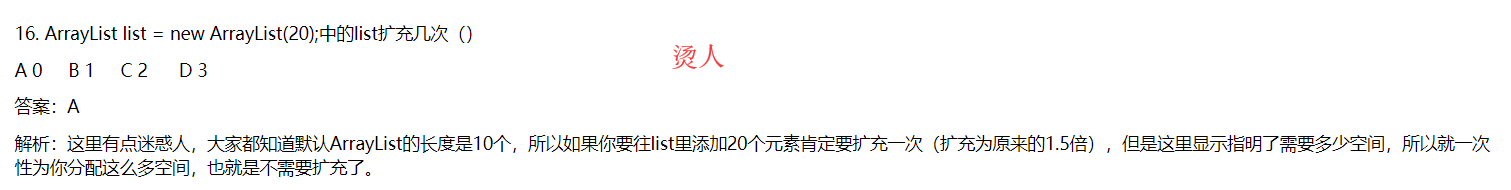
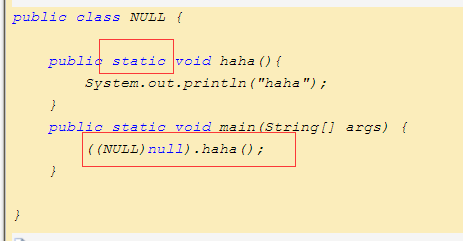
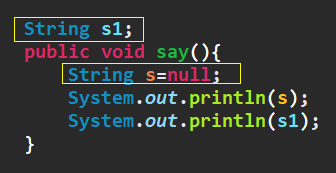


性能比较：new最快，其次，反射，clone，序列化





这样是能够输出的，这里必须是static方法，在java中可以将null强制转换为java类型，但是这里null任然为空，这里能输出结果的原因是static方法，这里并为创建对象



方法内部的String必须初始化，why：s1是成员变量，存在堆中，在分配堆内存时，JVM会对属性进行初始化操作，局部变量在栈中，声明周期短，如果由JVM初始化，增加开销，所以，局部变量必须手动初始化

接口和抽象类的区别：

抽象类的方法不能是private，这样子类将不能重写此方法，默认public

接口中的方法默认是public abstract，其他修饰符都不可以：private，protected，static，final

接口的定义就是不同类直接的相同行为，如果使用private修饰符就限制了接口的使用范围

接口中声明的属性是常量 static final

接口中不能包含静态代码块和静态方法

抽象类中可以有静态代码块和静态方法

java内存空间：程序计数器，java虚拟机栈，本地方法栈，堆，方法区

程序计数器：

用于记录(保存)当前线程正在执行的字节码的地址

程序计数器每个线程私有，这是一个唯一不会出现OutOfMemoryError的内存区域

java虚拟机栈：

java虚拟机栈使用描述java方法运行过程的内存模型

java 虚拟机栈会为每一个即将运行的 Java 方法创建一块叫做 “栈帧” 的区域

用于存储方法的局部变量表，基本数据类型，引用数据类型，返回值等

方法执行完毕，方法对应的栈帧将会出栈，并释放内存空间

java虚拟机栈线程私有

本地方法栈：

本地方法栈和java虚拟机栈类型，只不过本地方法栈是本地方法(native)的内存模型

堆：

用来存放对象

线程共享，java中只有一个堆

在JVM启动时创建，GC的主要场所

堆：先进先出

栈：先进后出

方法区：

存放被虚拟机加载的类信息，常量，静态变量

线程共享，方法区是堆的一个逻辑部分，整个JVM只有一个方法区

Class对象在方法区中

常量池：

一个类中的public static final变量，类被编译为class文件之后，类的所有

信息存储在class文件中，当class文件被加载到内存中，常量就会存放在常量池中

运行时可以向常量池中添加新常量

java中new一个对象的过程：

1，jvm将class文件加载到方法区中，

根据方法区中该类的信息，确定该对象的大小，一个类的所有对象大小都是一样的

JVM 在一个类被加载进方法区的时候就知道该类生产的每一个对象所需要的内存大小

2，在对内存中划分一块对应大小的内存给对象

3，为对象的成员变量进行赋初值(默认大小)

4，调用构造函数进行初始化

对象属性存放的位置：

局部变量(方法内部)：

当调用对象的方法时，JVM会创建一个方法栈，局部变量是基本数据类型时

变量名和值都保存在栈中，引用类型时，变量名保存在栈中，值保存在堆中

类变量：

基本数据变量名和值都保存在对象所在的堆中

引用类型变量名存放在栈中，对应的值存放在堆中

成员变量存在于堆内存中，随着对象的产生而存在，消失而消失。

局部变量存在于栈内存中，随着所属区域的运行而存在，结束而释放。

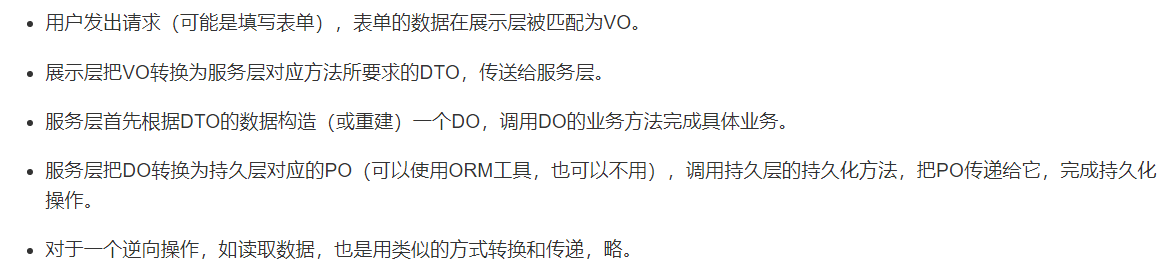
栈中存放的数据只有在执行方法的时候才会开辟，方法执行完被释放

VO：view Object，页面和Controller交换的对象

PO：和数据库交互的对象

DO：从现实世界中抽象出来的有形或无形的业务实体

DTO：Controller和Servlet交换数据的对象



对于绝大部分的应用场景来说，DTO和VO的属性值基本是一致的，而且他们通常都是POJO，因此没必要多此一举。但不要忘记这是实现层面的思维，对于设计层面来说，概念上还是应该存在VO和DTO，因为两者有着本质的区别，DTO代表服务层需要接收的数据和返回的数据，而VO代表展示层需要显示的数据。

网关：跳出当前网络(局域网)