# 封装性

## 类和对象

C++将对象抽象成类进行处理，由于目的的不同抽象出的信息也可能不同。

根据目的不同，决定哪些信息需要暴露出去，哪些信息需要隐藏起来，这些通过访问限定符（public、protected、private）。如果需要暴露信息使用public访问控制符，如果需要隐藏信息则使用private访问控制符。

## 对象实例化

类就是一个模板，所谓的对象实例化就是根据一个类的设定，制造多个对象的过程。

有两种实例化对象的方式：

1. 从栈中实例化：



从栈中实例化对象不需要手动释放内存，系统完成这部分工作，但是在堆中实例化则需要手动释放内存。

1. 从堆中实例化：

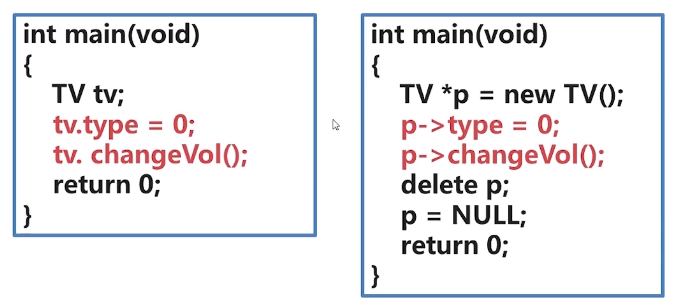


在堆中实例化对象需要手动释放内存，如果实例化的是一个对象，则采用delete p的形式，如果是实例化对象数组，那么采用delete []p的形式。

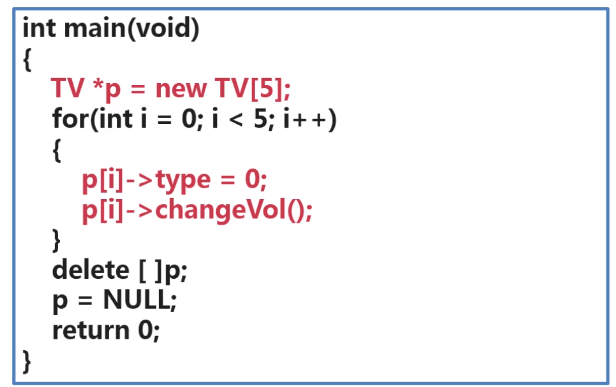
## 对象成员访问

实例化对象后需要访问对象的成员，根据栈和堆两种实例化方式的不同，访问成员的方式也有所不同。

针对单一对象而言访问方式：

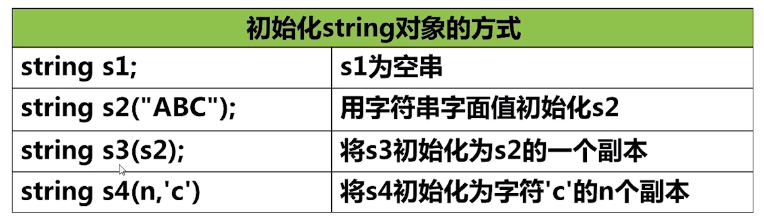


针对对象数组而言访问方式：

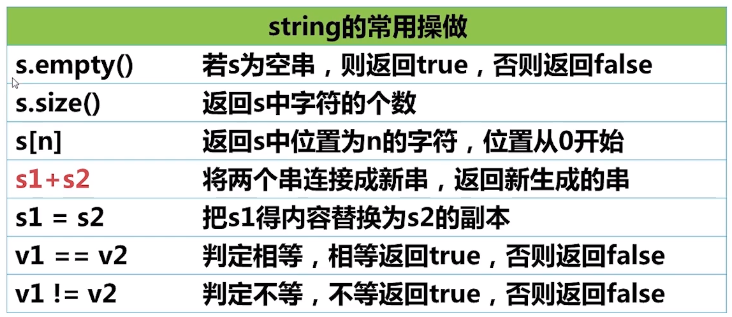


## String

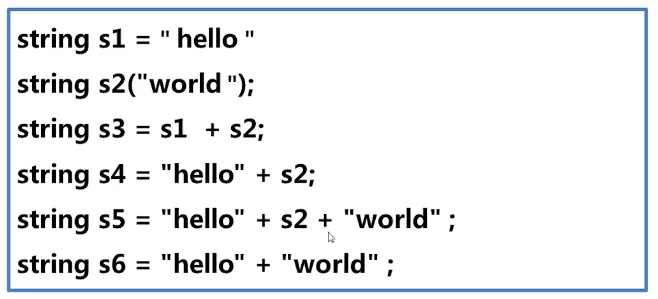
### 初始化



### 常用操作



有一种连接方式需要格外注意：



只有string变量与双引号“”的字符串结合的时候才是合法的，两个“”字符串结合是非法的，比如这里的s6。

拓展：为什么Java的String设置为不可变的呢？

Java中将String设计成不可变的是综合考虑到各种因素的结果，想要理解这个问题，需要综合内存，同步，数据结构以及安全等方面的考虑：

1、字符串常量池的需要

字符串常量池(String pool, String intern pool, String保留池) 是Java堆内存中一个特殊的存储区域，当创建一个String对象时，假如此字符串值已经存在于常量池中，则不会创建一个新的对象，而是引用已经存在的对象。

假若字符串对象允许改变，那么将会导致各种逻辑错误，比如改变一个对象会影响到另一个独立对象。严格来说，这种常量池的思想,是一种优化手段。

2、允许String对象缓存HashCode

Java中String对象的哈希码被频繁地使用，比如在hashMap 等容器中。

字符串不变性保证了hash码的唯一性，因此可以放心地进行缓存。这也是一种性能优化手段，意味着不必每次都去计算新的哈希码。

3、安全性

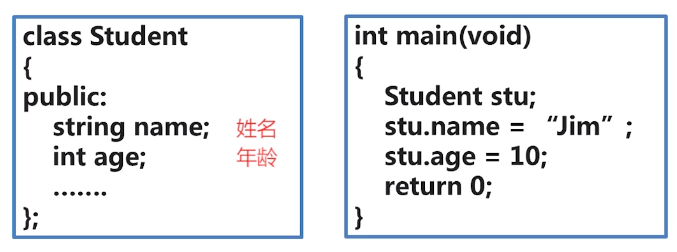
String被许多的Java类(库)用来当做参数。例如：网络连接地址URL，文件路径path，还有反射机制所需要的String参数等，假若String不是固定不变的，将会引起各种安全隐患。

## 数据的封装

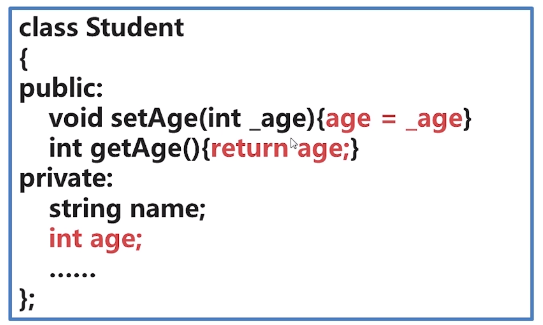
所谓数据的封装就是将类中成员变量的操作转换为成员函数的操作，这样才是封装性的体现，即不直接暴露成员变量本身，而是暴露出成员变量操作的函数。

面向对象的思想就是以谁做什么为基础，反映到代码层面上就是类的成员函数。

如下例子就是违背了这种思想：



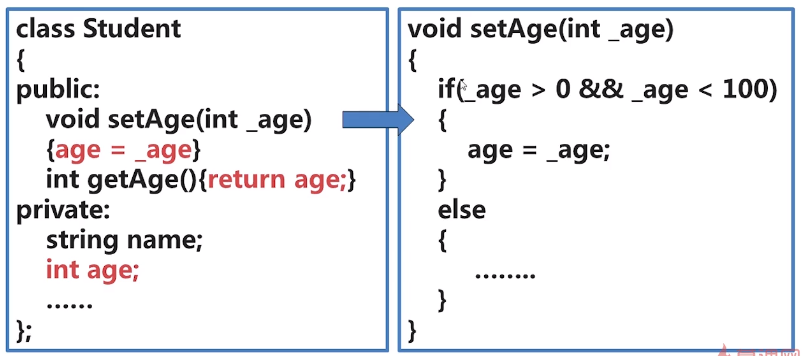
应该采用如下的方式：



这样具备两种好处：

1、更加符合面向对象的思想；

2、可以根据需要对参数进行校验：

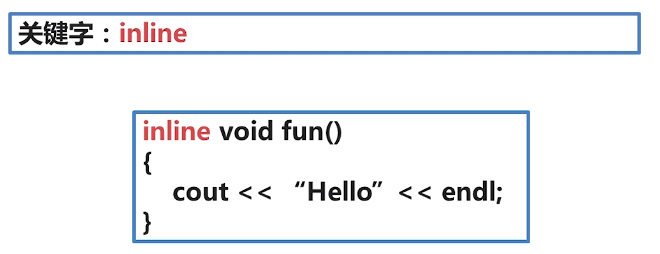


3、可以设置只读属性：

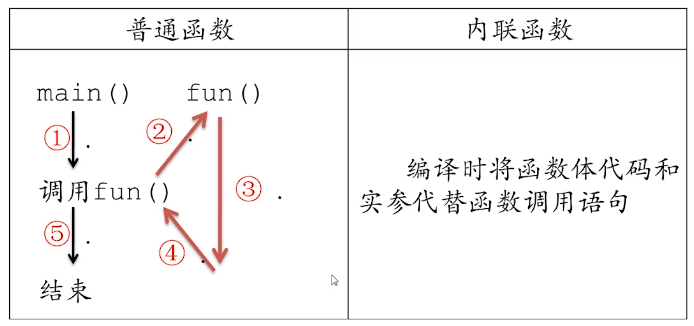
有些时候我们可能需要设置某些成员变量只能被读或者写操作，这样就单独设置仅有getter或者setter中的某一个操作函数即可。



## 内联函数



内联函数与普通函数的区别：



内联函数相当于省去了2、4调用和返回的过程，只有一个3函数本身执行的过程。这样效率更高，速度更快，但是仅仅适用于结构比较简单的函数，如果函数逻辑复杂，即使设置内联函数，编译器也会拒绝按照内联函数的方式处理。

# 继承性

# 多态性