# 概述

**不可变的字符序列**

1. String声明为final的，不可被继承  
   2. String实现了Serializable接口：表示字符串是支持序列化的。  
    实现了Comparable接口：表示String可以比较大小  
   3. String内部定义了**final char[] value**用于存储字符串数据

Value 数组声明为final不能改变，所以改变string 值（拼接之类的）的时候会指向新的字符串。  
4.String:代表不可变的字符序列。简称：不可变性。  
 体现：1.当对字符串重新赋值时，需要重写指定内存区域赋值，不能使用原有的value进行赋值。  
 2. 当对现有的字符串进行连接操作时，也需要重新指定内存区域赋值，不能使用原有的value进行赋值。  
 3. 当调用String的replace()方法修改指定字符或字符串时，也需要重新指定内存区域赋值，不能使用原有的value进行赋值。  
5.通过**字面量**的方式（区别于new）给一个字符串赋值，此时的字符串值声明在字符串常量池中。  
6.字符串常量池中是不会存储相同内容的字符串的。

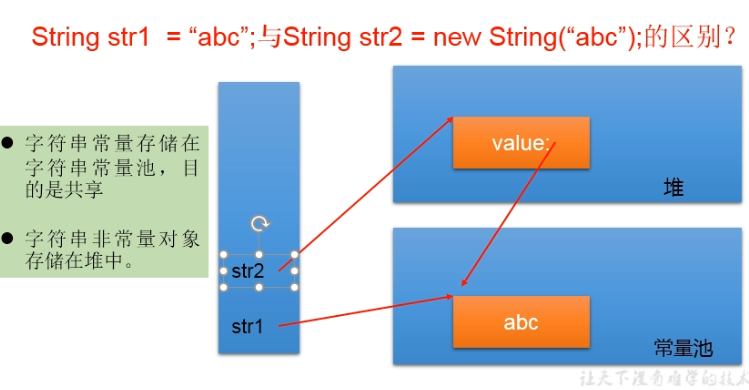
# 两种实例化的方式

## 方式一：通过字面量定义的方式



## 方式二：通过new + 构造器的方式

**两种方式的区别**

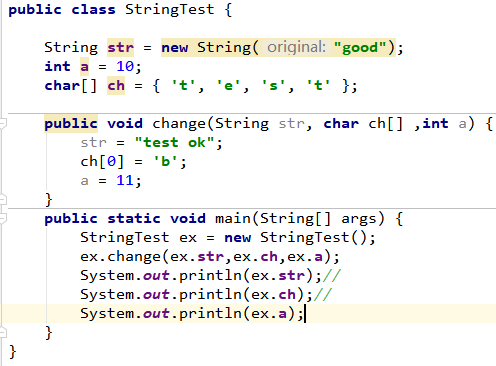


**~** 面试题:String s = new String("abc");方式创建对象,在内存中创建了几个对 象？  
 两个:一个是堆空间中new结构，另一个是char[]对应的常量池中的数据："abc"

# 不同拼接方式的对比

1.常量与常量的拼接结果在常量池。且常量池中不会存在相同内容的常量。  
2.只要其中有一个是变量，结果就在堆中。  
3.如果拼接的结果调用intern()方法，返回值就在常量池中

# 四、值传递和引用传递



# 五、String、StringBuffer、StringBuilder三者的异同？

String:不可变的字符序列；底层使用char[]存储  
StringBuffer:可变的字符序列；线程安全的，效率低；底层使用char[]存储  
StringBuilder:可变的字符序列；jdk5.0新增的，线程不安全的，效率高；底层使用char[]存储

# StringBuffer和StringBuilder源码分析

**String初始化**  
String str = new String();//char[] value = new char[0];  
String str1 = new String("abc");//char[] value = new char[]{'a','b','c'};

**StringBuffer初始化**  
StringBuffer sb1 = new StringBuffer();//char[] value = new char[16];底层创建了一个长度是16的数组。  
sb1.append('a');//value[0] = 'a';  
sb1.append('b');//value[1] = 'b';  
  
StringBuffer sb2 = new StringBuffer("abc");//char[] value = new char["abc".length() + 16];创建一个长度为形参长度+16的字符数组

# 七、Stringbuffer扩容

如果要添加的数据底层数组盛不下了，那就需要扩容底层的数组。  
默认情况下，扩容为原来容量的**2倍 + 2**，同时将原有数组中的元素复制到新的数组中。如果（添加的数据长度+原始长度）超过了原来长度的2倍+2 则扩容为（添加的数据长度+原始长度）  
  
指导意义：开发中建议大家使用：StringBuffer(int capacity) 或 StringBuilder(int capacity) 可以防止扩容产生的效率问题