一，字符串简介

JDK1.9之后针对字符串的形式定义变多了

JDK1.9之前String用的是private final char[] value

JDK1.9之后String用的是private final byte[] value

从上面可以看出，String就是对数组的一种特殊包装应用

所有一旦赋值就无法改变

String类可以 1，直接赋值

也可以通过 2，构造方法进行实例化

两种方法的效果是不一样的

二，String的两种创建方式

==是数值比较，比较的是对象时，判断的是两个对象的内存地址是否相同(两个引用是否指向同一个对象)

equals() 判断的是两个对象的内容是否相同

首先我们要先了解Java中的字符串常量池

JVM中有常量池，常量池位于堆中，字符串常量池是常量池的一种，所以字符串常量池也位于堆中

1，String直接赋值

字符串常量其实是String类的隐匿对象

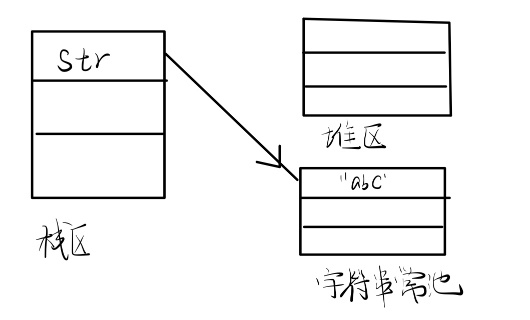
String str="abc"

去字符串常量池中查看字符串"abc"是否在字符串常量池中

如果不在————在字符串常量池中创建一个String对象("abc")，然后str保存这个对象的内存地址(指向这个对象)

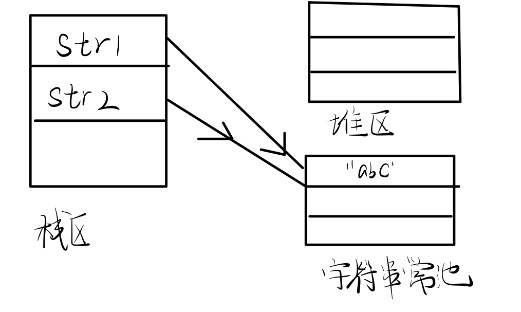
如果在————str保存字符串常量池中String对象("abc")的地址

无论以后用这种方式创建多少个值为”abc”的字符串对象，始终只有一个内存地址被分配。



String str1="abc";

String str2="abc";



str1==str2;————————true，因为str1和str2都保存着 字符串常量池中String对象("abc")的地址

2，new String创建字符串

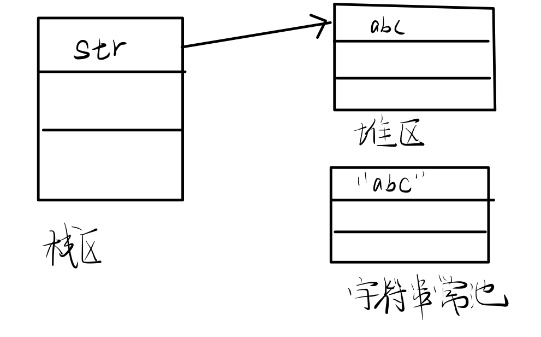
String str=new String("abc");

字符串常量池会检测"abc"是否在常量池中

如果不在————在字符串常量池中创建一个String对象("abc")，然后再堆中也创建一个String对象("abc")

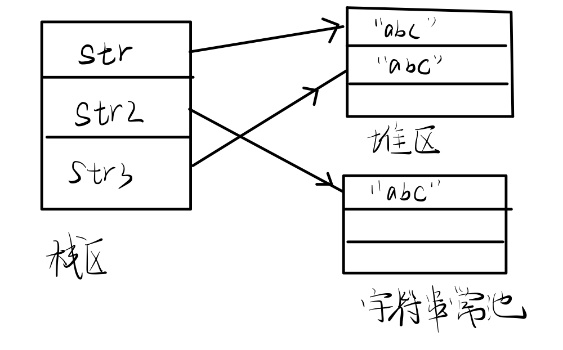
也就是会创建2个String对象，str保存堆中String对象的地址

如果在—————在堆中创建一个String对象，然后str保存堆中String对象的地址



String str2="abc"—————直接指向字符串常量池中的String对象"abc"

String str3=new String("abc")—————在堆中再创建一个String对象("abc")，str3保存该对象的地址



str==str2——————false

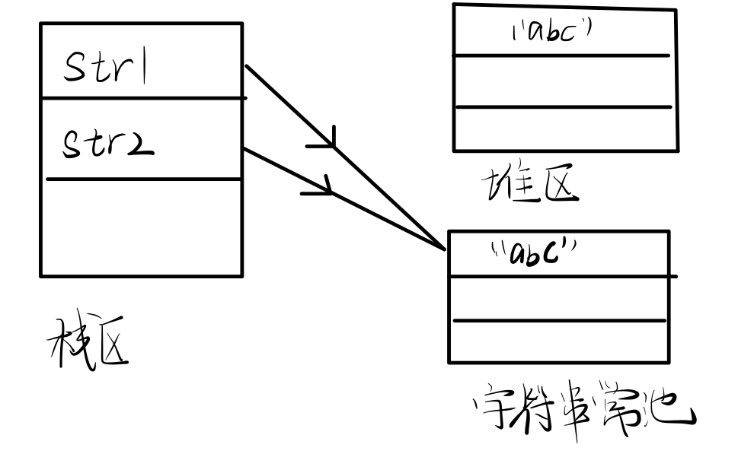
str==str3——————false

str2==str3——————false

3,String类有个intern()方法，可以实现String对象的入池操作

String str1=“abc”;

String str2=new String(“abc”).intern();



str1==str2————true

三，String是不可变对象

String str=“abc”;————在字符串常量池中创建“abc”对象，str保存该对象的地址

str=str+”efg”;————原来str指向的“abc”对象并没有消失，仍然保存在字符串常量池中

而是创建了一个新的String对象”abcdefg”,str指向了“abcdefg”

由此也可以看出，频繁的对String对象进行修改，会造成很大的内存开销。此时应该用StringBuffer或StringBuilder来代替String。

而new String（）更加不适合，因为每一次创建对象都会调用构造器在堆中产生新的对象，性能低下且内存更加浪费。