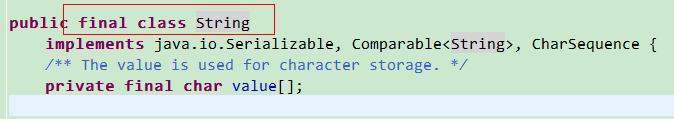
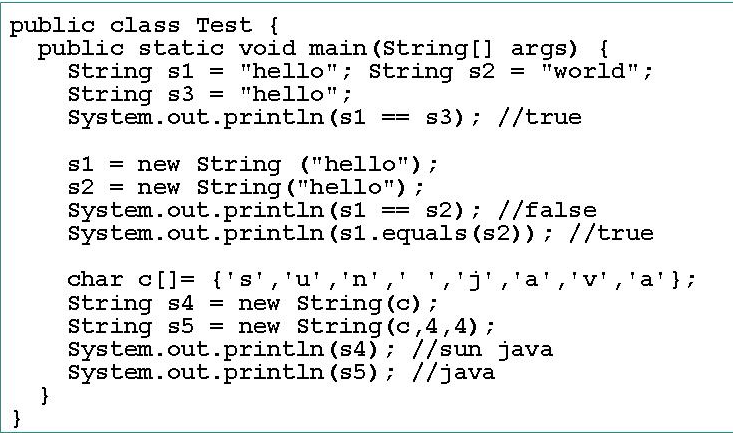
1. java.lang.String类代表**不可变**的字符序列。String类是final类，不可以被继承



1. “xxxx”为该类的一个对象。
2. String类的常见构造方法：(查api文档)

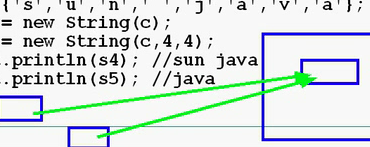


1. String类举例(1)

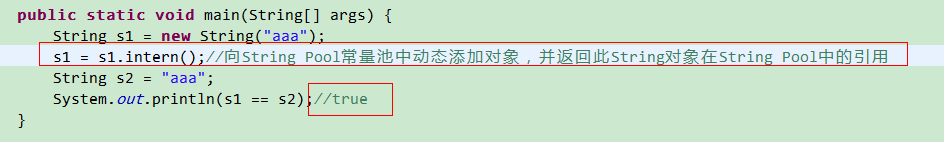


内存分析：

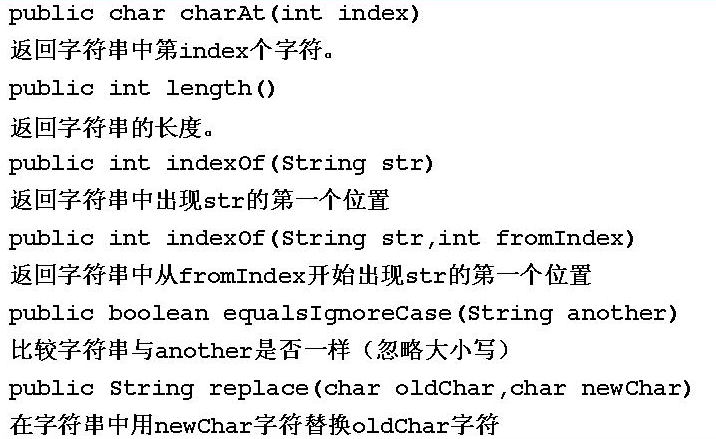
String s1 = “hello”; String s3 = “hello”;//字符串常量存于Data seg中，当编译器发现两个字符串常量相同时，会经过优化，不会再分配内存存储，所以内存格局如下：

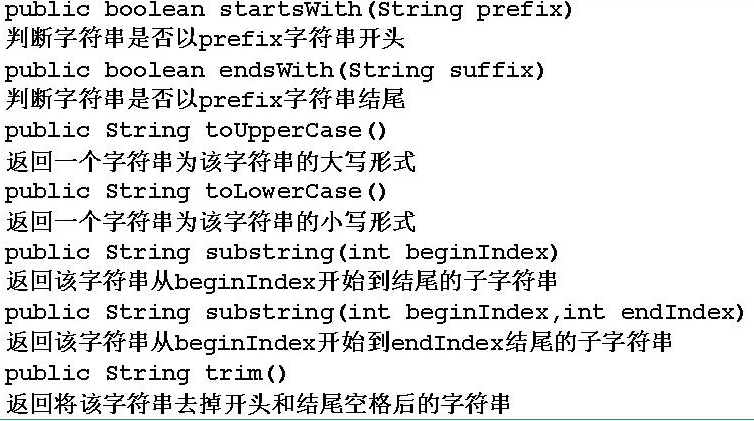
s1 == s3//true

String类中重写了Object类中的equals方法，比较的是字符串的内容而不是引用。

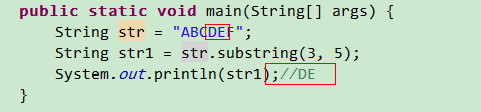


1. String类常用方法



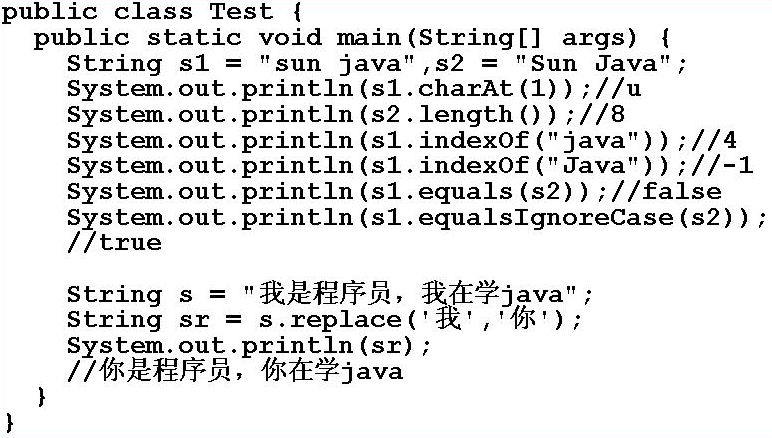


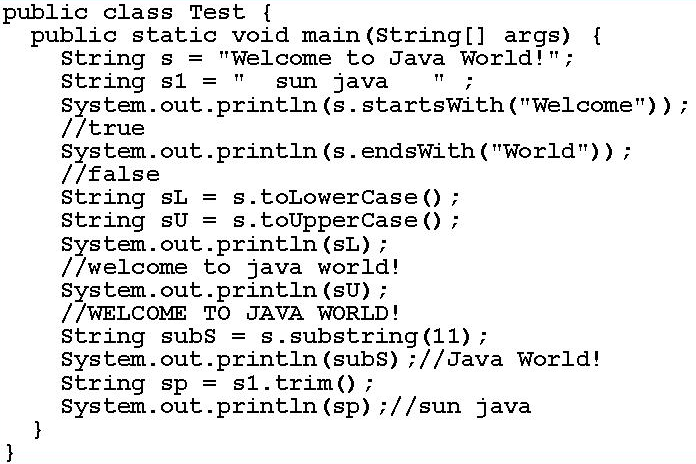
注：substring(int beginIndex, int endIndex);//前包括，后不包括。

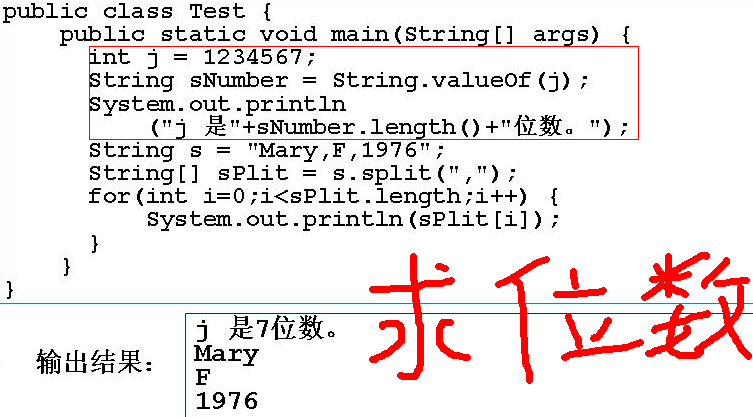




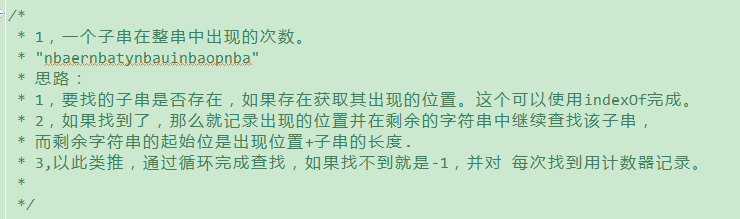
例：



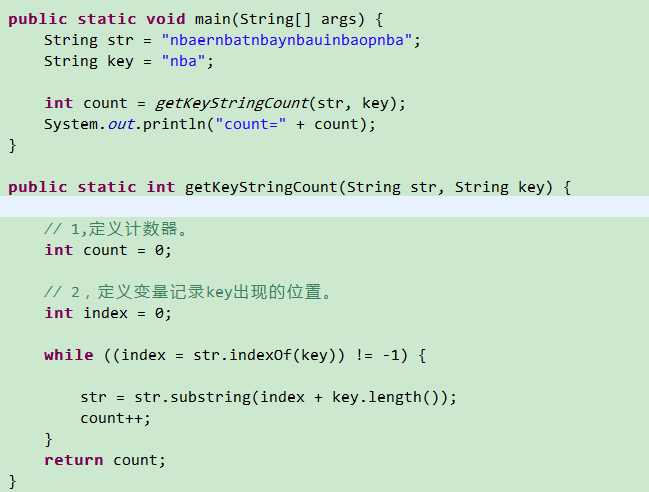




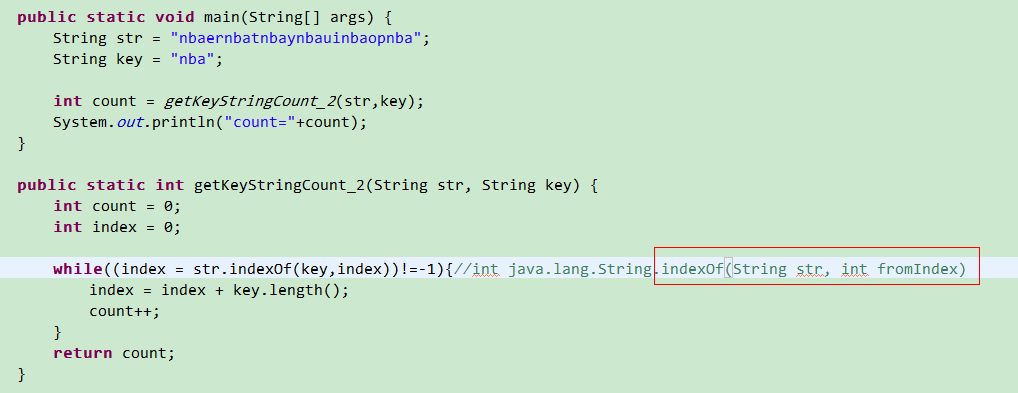
1. 字符串简单算法练习
   1. 子串的次数
      1. 思路



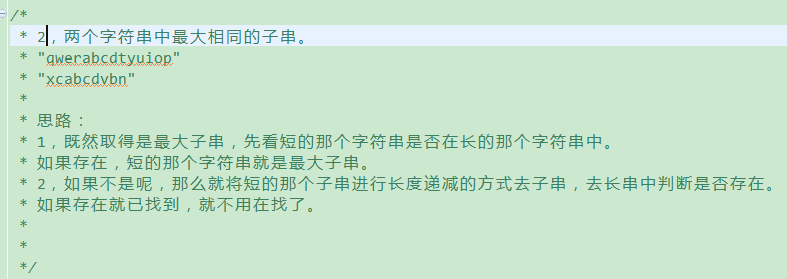
* + 1. 代码1(会产生很多的临时字符串)：



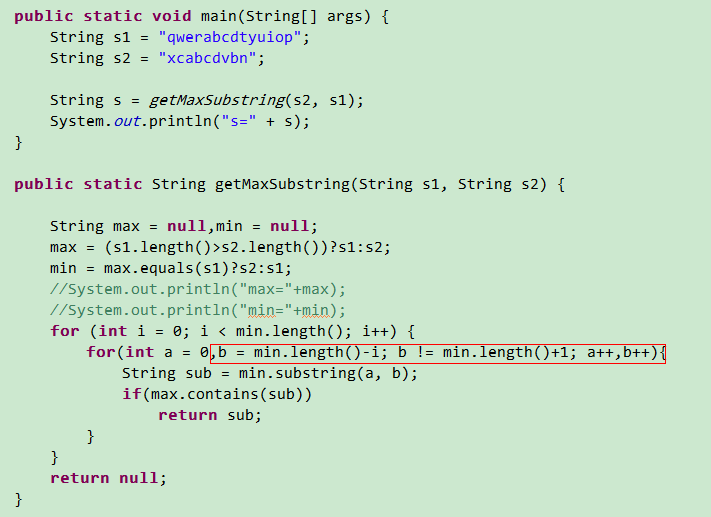
* + 1. 代码2：



* 1. 最大相同子串
     1. 思路



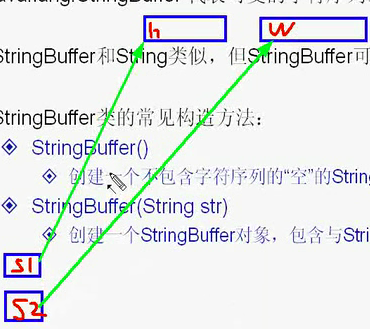
* + 1. 代码

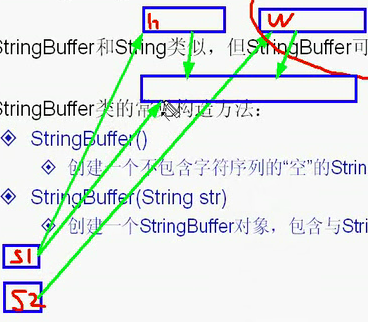


1. StringBuffer类：java.lang.StringBuffer代表**可变**的字符序列。StringBuffer和String类似，但StringBuffer可以对其字符串进行改变。

不可变的内存分析：

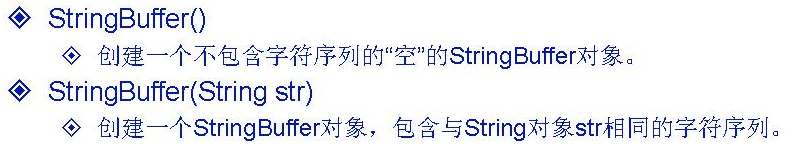
String s1 = “hello”; String s2 = “world”; s1 += s2;（不可变的String）



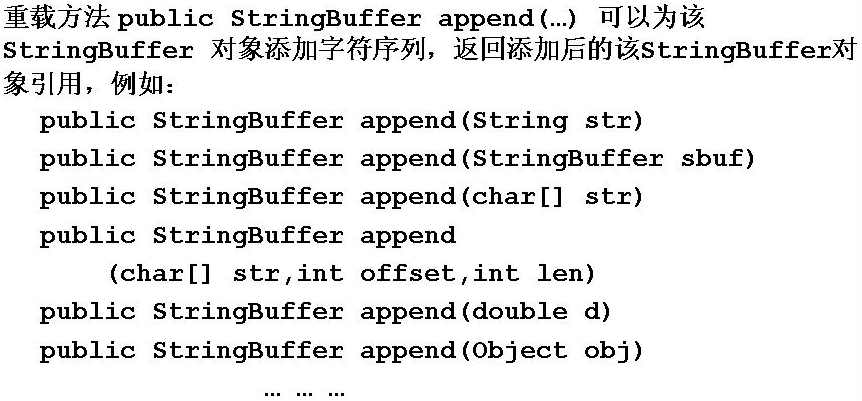


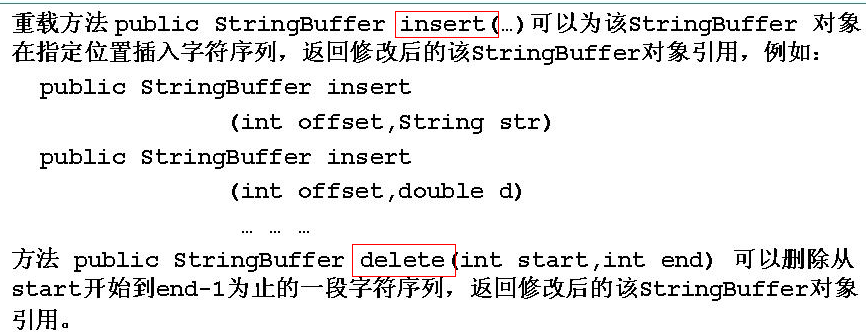
重新分配内存，两次copy，再改变s1引用(时空效率低)。

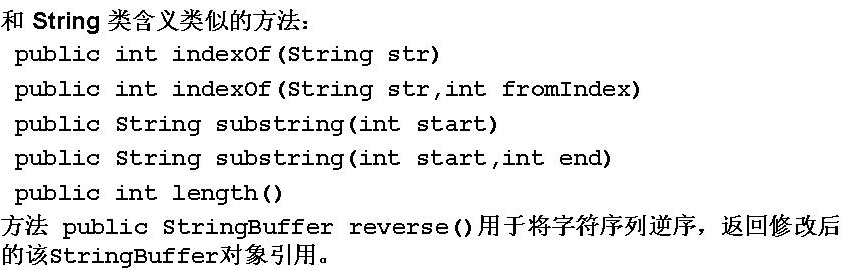
1. StringBuffer类常见的构造方法。



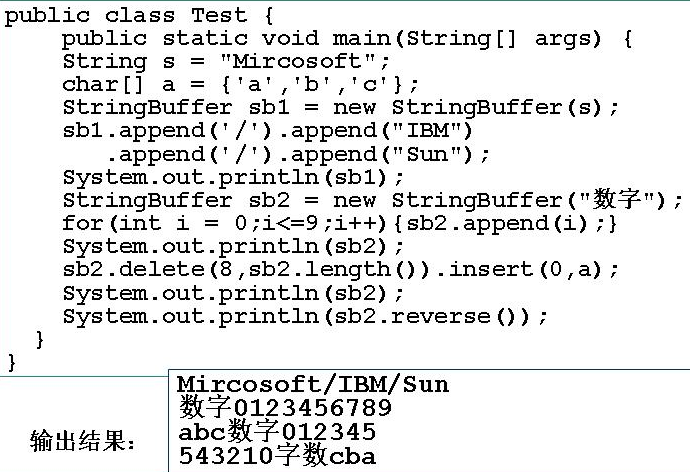
1. StringBuffer常用方法



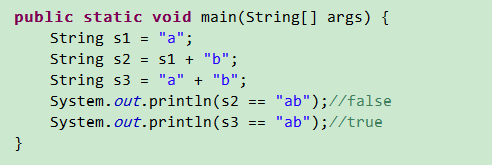




例子：



1. String，StringBuffer与StringBuilder的区别
   1. String 字符串常量
   2. StringBuffer 字符串变量（线程安全）
   3. StringBuilder 字符串变量（非线程安全）
   4. 在大部分情况下速度： StringBuffer > String
   5. 在大部分情况下速度： StringBuilder > StringBuffer
2. Javac编译可以对字符串常量直接相加的表达式进行优化，不必要等到运行期去进行加法运算处理，而是在编译时去掉其中的加号，直接将其编译成一个这些常量相连的结果



String s = “a” + “b” + “c” + “d”;该语句总共创建一个String对象。