String在内存中如何存储

JDK1.8中JVM把String常量池移入了堆中，同时取消了“永久代”，改用元空间代替（Metaspace）

java中对String对象特殊对待，所以在heap区域分成了两块，一块是字符串常量池(String constant pool)，用于存储java字符串常量对象，另一块用于存储普通对象及字符串对象。

string的创建有两种方法：

```
public static void main(String[] args) {
   String a = "abc"; //第一种
   String b=new String("abc"); //第二种
   String c = "abc";
   System.out.println(a == b);//false
   System.out.println(a == c);//true
}
```

对于第一种，此创建方法会在String constant pool中创建对象。jvm会首先在String constant pool 中寻找是否已经存在"abc"常量，如果没有则创建该常量，并且将此常量的引用返回给String a；如果已有"abc" 常量，则直接返回String constant pool 中“abc”的引用给String a。

对于第二种，jvm会直接在非String constant pool 中创建字符串对象，然后把该对象引用返回给String b，并且不会把"abc” 加入到String constant pool中。new就是在堆中创建一个新的String对象，不管"abc"在内存中是否存在，都会在堆中开辟新空间。

虽然new String()方法并不会把"abc” 加入到String constant pool中，但是可以手动调用String.intern()，将new 出来的字符串对象加入到String constant pool中。

```
String s1 = new String("abc");
String s2 = "abc";
System.out.println(s1 == s2); //false
System.out.println(s1.intern() == s2); //true
```

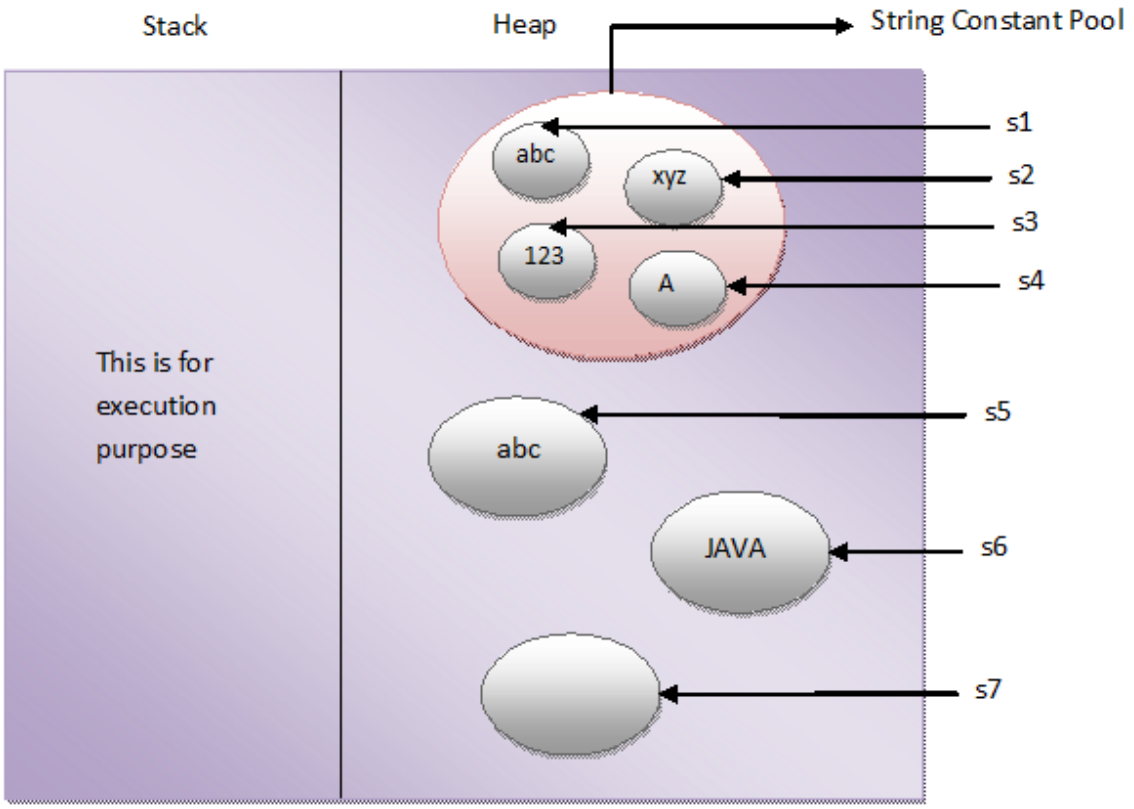
当一个String实例调用intern()方法时，会查找常量池中是否有相同的字符串常量，如果有，则返回其的引用，如果没有，则在常量池中增加一个等于str的字符串并返回它的引用，由于s2已经在常量池中，所以s1.intern()不会再创建，而是直接引用同一个"aaa"。

例：

```
public static void main(String[] args) {
      String s1 = "abc";//字符串常量池
      String s2 = "xyz";//字符串常量池
      String s3 = "123";//字符串常量池
      String s4 = "A";//字符串常量池
      String s5 = new String("abc"); //堆里
      char[] c = {'J','A','V','A'};
      String s6 = new String(c);//堆里
```

```
        String s7 = new String(new StringBuffer());//堆里
    }
```

字符串在内存中的存储情况如下图所示：



对于字符串：其对象的引用都是存储在栈中的，如果是【编译期已经创建好(直接用双引号定义的)的就存储在常量池中】，如果是【运行期（new出来的）才能确定的就存储在堆中】。对于equals相等的字符串，在常量池中永远只有一份，在堆中有多份。