链滴

java 源码探索系列 -01 String

作者：pleaseok
原文链接：https://ld246.com/article/1552292410269
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

这一系列的源码探索我打算结合一些常见的疑问深入探究一下。答案也许都知道，但是从源码角度来为什么会这样，我想还是有许多人是模糊状态的。所以一起来带着问题来看看java的设计吧～

String类中包含了许多方法，关于它的构造方法就有十来种。此外还有一些工具方法，比如字符串的较：equals()、contentEquals()、compareTo()、compareToIgnoreCase()等等，另外还有比如字串的长度length(),字符串的拼接concat()等等方法....

1. equals()、hashCode()、==的区别?

众所周知，equals() hashCode()在Object类中也有同样的方法:

```
//equals()：就是直接比较对象的内存地址（==）
//另外在看源码的过程中还注意到一个注释：当需要重写equals方法时，必须要重写HashCode方法因为相等的对象必须具有相等的哈希代码。
/** Note that it is generally necessary to override the {@code hashCode}
 * method whenever this method is overridden, so as to maintain the
 * general contract for the {@code hashCode} method, which states
 * that equal objects must have equal hash codes.
 */
public boolean equals(Object obj) {
    return (this == obj);
}
//hashCode()：返回对象的JVM内存地址
public native int hashCode();
```

下面再来看看String中重写的这两个方法又增加了什么内容吧～

```
public boolean equals(Object anObject) {
        //第一步，如果引用地址相同就直接返回true
        if (this == anObject) {
            return true;
```

```
        }
        //第二步，当地址不同的情况下，当都是Stirng类的实例的时候，然后再去挨个的比较它们的cha字符是否相等
        if (anObject instanceof String) {
            String anotherString = (String)anObject;
            int n = value.length;
            if (n == anotherString.value.length) {
                char v1[] = value;
                char v2[] = anotherString.value;
                int i = 0;
                while (n-- != 0) {
                    if (v1[i] != v2[i])
                        return false;
                    i++;
                }
                return true;
            }
        }
        return false;
    }

private int hash; // Default to 0
// final修饰，因此String的内容也是不变的
private final char value[];// The value is used for character storage.
public int hashCode() {
        int h = hash;
        if (h == 0 && value.length > 0) {
            char val[] = value;

            for (int i = 0; i < value.length; i++) {
                h = 31 * h + val[i];
            }
            hash = h;
        }
        return h;
    }
```

嗯嗯，代码看完可以直接得出这个问题的结论了。

1. String类中的equals()方法用来比较char内容，而Object类的equals()方法仅仅用来比较对象的用地址。

2. String类中的hashCode()方法是重新计算出的hashCode数值，而Object类的hashCode()方法回对象的JVM内存地址。

3. ==是用来比较对象的内存地址的

4. 重写equals()方法必须要重写hashCode()方法。(其实什么引用地址内存地址指的就是一个地址，是Object类中hashCode()32位JVM地址)

2. 创建String对象有哪些方式？它们的区别是什么？

第一个问题其实是非常清晰的：

1．我们常用到的Stirng s = "xxx"方式；

2．既然它是有构造函数的，我想它肯定也能通过构造函数来创建吧？话不多说直接看代码吧

```
// 这里只列出几个常用的构造，毕竟它有十几个构造，全列出来文章篇幅有点大...
private final char value[];
private int hash; // Default to 0
public String() {
        this.value = "".value;
    }

public String(String original) {
        this.value = original.value;
        this.hash = original.hash;
    }

public String(byte bytes[]) {
        this(bytes, 0, bytes.length);
    }

public String(StringBuffer buffer) {
        synchronized(buffer) {
            this.value = Arrays.copyOf(buffer.getValue(), buffer.length());
        }
    }

public String(StringBuilder builder) {
        this.value = Arrays.copyOf(builder.getValue(), builder.length());
    }
```

好啦，上面的代码块只是列举出了创建String的第二种方法。还是无法看出这两种创建方式的区别是么，那下面我们还是用一个例子来比较一下吧～

```
public static void main(String[] args) {
        String si = new String("ok");
        String ok = "ok";

        System.out.println(si==ok); // false
        System.out.println(ok==("o"+"k")); // true
    }
```

嗯？第一个为false我知道，但是为什么第二个为true呢？

其实这就引用到JVM堆内存里的常量池的定义了。**常量池(constant pool)指的是在编译期被确定，被保存在已编译的.class文件中的一些数据。包括类、方法中的一些常量**

那么，这里用new创建的显然是不属于常量池的咯。但是用new的方式如果常量池里面没有的话，它先也会在常量池里面创建。

是的，其实new就是创建的一个字符串对象。当然，可以使用这个方法intern()返回从字符串池中的符串对象的引用。

下面再来看一下这个问题

```
// 在没有创建相同字符串的情况下，下面会创建几个String对象？
String s1 = new String("Sean Blog Site: https://code666.top");
String s2 = new String("Sean Blog Site: https://code666.top");
```

两个？恭喜你答错了！答案是三个。

第一个是字符串常量池中Sean Blog Site: https://code666.top,另外两个就是堆内存创建的Stirng对略。所以应该是三个

还不懂？？？

其实它的执行顺序是这样的，首先在string池内找，如果找到就不创建string对象，否则创建，这样有了一个string对象，然后遇到new运算符号了，在内存上创建string对象，并将其返回给s1(s2)，又个对象。

还不懂？？？

那你的执行顺序应该要这样：搬上你的电脑主机爬上顶楼，然后朝向远处做抛物线运动...

3. String、StringBuilder、StringBuffer的区别?

关于这个经典面试题，我想大多数人都能回答出。

区别:

1. String创建的字符内容是不可变的，而StirngBuilder和StringBuffer是可变的
2. StringBuffer是线程安全的，StringBuilder是非线程安全，String也属于线程安全(字符内容是不可的)
3. 运行速度(字符拼接数据较多的情况)：StringBuilder运行速度最快，StringBuffer运行速度略快，Strng运行速度最慢

```
//第一个区别:
//String中char字段
private final char value[];

//StringBuffer
private transient char[] toStringCache;

//StringBuilder:直接使用的是父类AbstractStringBuilder的字段
char[] value;
```

由上可知，String类的字符内容是不可变的，因为是用的final修饰。而StringBuffer与StringBuilder建的字符内容都是可变的。transient是用来表示该字段在序列化与反序列化不变，

```
//第二个区别
//StringBuffer中方法基本都用`synchronized`修饰
@Override
public synchronized int length() {
    return count;
}

@Override
public synchronized int capacity() {
    return value.length;
}

@Override
public synchronized void ensureCapacity(int minimumCapacity) {
    super.ensureCapacity(minimumCapacity);
```

```
}

//而StringBuilder中的方法直接用public修饰，没有考虑到并发的情况
@Override
public StringBuilder deleteCharAt(int index) {
    super.deleteCharAt(index);
    return this;
}

@Override
public StringBuilder replace(int start, int end, String str) {
    super.replace(start, end, str);
    return this;
}
```

由上可知，StringBuffer是属于线程安全，就是因为它的方法都用关键词synchronized来修饰，而StringBuilder则没有。另外，String为什么也是线程安全呢？因为它是不可变的，不可变的肯定也是属于程安全啊。

第三个区别：

在数据量大的情况下为什么StringBuilder>StringBuffer>String呢？

其实从上面的代码可以得到答案，用synchronized修饰的方法多多少少也会影响到速度了。而String次都要创建一个新的对象，那肯定也会是最慢的了。