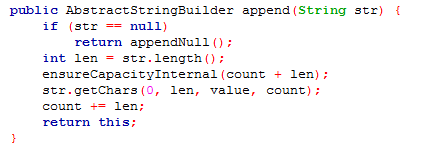
StringBuffer继承自AbstractStringBuilder，在AbstractStringBuilder里即实现了append()方法。

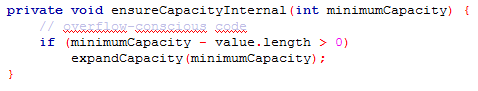
代码如下：

以String参数为例，其余的重载大同小异



该方法即表示在append的时候，直接把数组扩容，然后将str字符串直接接到数组后面。

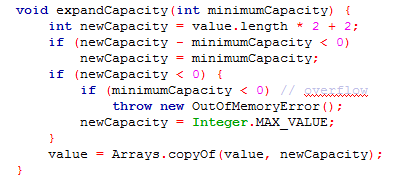
扩容的代码如下：



首先看初始容量大小，查看构造函数，

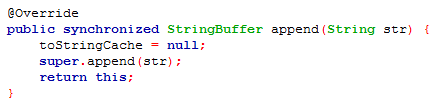


当组合的新串容量（即最小容量）大于原数组的最大值时，调用扩容方法。

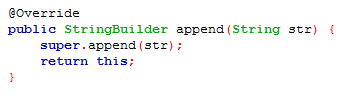


将原数组乘2加2后与新串比较，如果还是小于新串，那么直接将新串的长度赋给原数组。反之将原数组乘2加2式扩容。

在StringBuffer中，append是直接继承的父类的实现。



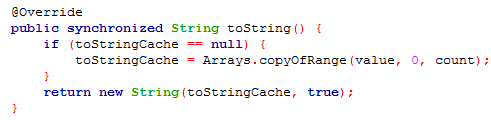
附带一提的是，StringBuilder和StringBuffer一样，在实现append方法时，都是直接沿用的父类实现。

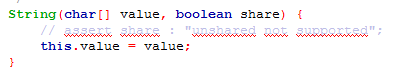


不过，在StringBuffer的append方法中出现了一个特殊的变量toStringCache。它的定义如下：



该变量是一个不可被序列化的字符数组，仔细读StringBuffer的源码，发现这个变量在很多地方出现过，全部都是在append，insert等操作数组的方法里，全部都是讲toStringCache赋空值。继续搜索，发现还有一个特殊的方法，toString()，



该方法表示，当toStringCache为空的时候，把value数组整体复制给toStringCache，所以就是缓存，正如它的字面解释。否则返回一个新字符串，查询一个String的构造函数，

发现这个boolean类型的true并没有起什么卵用，只是单纯地返回一个串，并且居然还用旧串指向新串的内存地址，这么做应该是为了节省复制新元素的时间吧。

综上，通过读StringBuffer和StringBuilder的一些代码，我发现，

首先，二者作为可变字符串，StringBuffer是线程安全的（基本所有操作串的方法都用了sychronized修饰），而StringBuilder是线程不安全的。

其次，StringBuilder和StringBuffer的大部分方法都是调用的父类AbstractStringBuilder的实现，其扩容机制首先是把容量变为原来容量的2倍加2。最大容量是Integer.MAX\_VALUE。

StringBuilder和StringBuffer的默认容量都是16，最好预先估计好字符串的大小避免扩容带来的时间消耗。