String：  
属性：  
1. private final char value[] //被final修饰，不能改变value数组的指向，String不可变，只能将String引用指向新的String对象，原对象舍弃。 value默认大小为0，且数组大小和字符串长度相等，也就是说value数组中没有多余空间。

构造方法：  
1. public String(byte bytes[]) //每个byte扩充到2字节(一个字符)

1. public String(char value[]) //复制一份字符数组
2. public String(String original) //两个String指向同一个value
3. public String(StringBuffer buffer)//拷贝一份buffer的value数组，加syn锁
4. public String(StringBuilder builder) //拷贝一份builder的value数组

成员方法：  
1. public int compareTo(String anotherString)

1. public int compareToIgnoreCase(String str) //忽略大小写比较
2. public String concat(String str) //新创建一个String 拼接(数组拷贝)后赋给新String并返回。原String对象不变。
3. public native String intern()
4. public String replace(char oldChar, char newChar) //创建新的String对象 字符串中的old全部换成new 然后返回 如果old==new 则不用创建新对象，直接返回this
5. public String substring(int beginIndex, int endIndex)//截取旧数组 创建新String对象返回
6. public String trim() //底层用substring 返回新String对象

StringBuilder：

属性：

1. char[] value; //这个数组不是final，可以修改指向。 数组中可能有多余空间 默认大小为16
2. int count; //记录了字符串的长度

构造方法：

1. public StringBuilder() //采用默认容量16 立即初始化数组
2. public StringBuilder(int capacity) //按照指定的容量创建数组
3. public StringBuilder(String str) //创建容量为str.len+16的数组，数组空闲元素为16，然后将str 调用append方法添加到数组中

成员方法：

1. private void ensureCapacityInternal(int minimumCapacity) //数组最小需求大小为

minimumCapacity，如果不够则扩容。

1. private int newCapacity(int minCapacity) //扩容 数组最小需求大小为 minimumCapacity

扩容为原来数组大小的2倍+2，如果不够直接指定为minimumCapacity，没超过上限则指定为上限，超过上限指定为minimumCapacity，超过int最大值抛异常

1. public AbstractStringBuilder append(String str) //将str添加到value数组尾部，在之前需要判断是否要扩容。
2. public AbstractStringBuilder delete(int start, int end) //value数组后面部分拷贝到前面覆盖掉删除的部分。
3. public AbstractStringBuilder insert(int offset, String str) //指定位置插入元素，之后的元素后移，如果str为空，则插入字符串”null”。
4. public String substring(int start, int end) //创建一个String对象并返回
5. public String toString()//创建一个String对象并返回
6. public void trimToSize()//原数组拷贝到大小正合适的新数组
7. 没有重写equals方法和hashCode方法。可以通过toString来比较。

StringBuffer：

属性:  
1. char[] value; //可变

2.private transient char[] toStringCache; //调用toString时，如果cache不是null则直接返回一个String对象，这个String对象的value直接指向cache，如果cache是null，会将value拷贝到cache中，作为返回的String的value，这样做不用在每次toString时都拷贝value数组，直接让value指向这个cache即可，因为cache是不变的，如果buffer改变了，chche会指向null，不会改变之前返回的cache。因为buffer是线程安全的，这样做为了更加节省时间。Builder中的toString是每次都拷贝一份新的value作为返回的String的参数。

1. int count;

构造方法和builder一样

成员方法：  
1.public synchronized String toString() //cache数组不为空，直接作为String的value返回。如果为空，则拷贝一份buffer的value作为cache，然后再作为String的value返回。

1. 方法都加了syn锁，且如果value有变动，则cache置为null。其他都和builder一样