String:

特性:

- 内容不会发生改变，它的对象在创建后不能被更改。如果进行修改则会创建一个新的字符串变量。
- 由于String对象不可变，所以它是线程安全的。

创建方法:

- 直接赋值:

```
String name="... ";
```

- 构造方法:

```
public String()
public String(String original)
public String(char[] chs)
public String (byte[] chs)
```

方法:

- 获取长度:

```
str.length()
```

- 查找子串:

```
//返回第一个指定字串的索引
str.indexOf (String)
//返回最后一个指定字串的索引
str.lastIndexOf(String)
```

- 转换大小写:

```
str.toLowerCase()
str.toUpperCase()
```

- 字符串比较:

```
//返回boolean类型，这个方法是重写了Object类中的equals方法
//（另两个sb没有重写，所以比较的不是内容而是引用）
str.equals(string)
//忽略大小写的比较（只能作用于String对象）
//如果调用者是null或其他类型对象，抛出NullPointerException异常
str.equalsIngoreCase(string)
```

- 按字典顺序比较两个字符串:

```
//返回值为int类型
//如果str在String前面，返回负整数。相同返回0，大于返回正整数
str.compareTo(String)
//忽略大小写差异
str.compareToIngoreCase(String)
```

- 获取索引为i的字符：

```
char c=str.charAt(i)
```

- 截取字符串内容：

```
//包头不包尾，包左不包右
str.substring (int beginIndex ,int endIndex)
//截取到末尾
str.substring (int beginIndex)
```

- 字符串替换

```
//替换字符或字符串子串
str.replace(char oldChar,char newChar)
str.replace(CharSequence target,CharSequence replacement)
//利用正则表达式替换
str.replaceAll(String regex,String replacement)
str.replaceFirst(String regex,String replacement)//替换第一个
```

- 去除开头和结尾空白：

```
str.trim()
```

- 分割成数组(在regex处将字符串切开变成字符串数组)

```
//这与limit为0相同
str.split(String regex)
//limit为正表示生成的数组最多有limit个元素
//limit为负表示保留所有可能的尾随空字符串
str.split(String regex,int limit)
```

- 将字符串转换成字符数组：

```
str.toCharArray()
```

StringTable

- StringTable在堆内存中
- 只有直接复制的字符串存在串池中，使用构造方法的不存
- 采用直接赋值的方式时，系统会在串池中搜索有没有要赋值的字符串，如果有，串池中不会创建新的字符串.

所以采用直接赋值的方法，代码简单且节约内存

StringBuilder

特性:

- 对象可变，允许在其生命周期内修改内容而不创建新的对象
- 线程不安全：没有使用同步机制
- 适合在单线程环境中需要频繁修改字符串的情况下使用

构造方法:

```
public StringBuilder()//创建一个空白可变字符串对象
public String Builder(String str)//根据字符串的内容来创建可变字符串对象
```

方法:

- 长度

```
sb.length();
```

- 拼接字符串

```
sb.append(str);
```

- 删除字符或子串

```
delete(int start,int end);//删除指定范围的子串（包左不包右）
deleteCharAt(int index);//删除指定索引的字符
```

- 插入字符串

```
sb.insert(int offset,String str)
//如果offset超出了最大索引，则在字符串的末尾插入
```

- 反转

```
sb.reverse();
```

- 按字典顺序比较字符串

```
//返回值为int类型
//如果sb1在sb2前面，返回负整数。相同返回0，大于返回正整数
sb1.compareTo(sb2)
```

- 转化成string:

```
sb.toStirng();
```

- 容量相关

```
//返回当前容量
sb.capacity();
//保证当前StringBuilder至少具有指定的容量
sb.ensureCapacity(int minimunCapacity);
```

StringBuffer

特性:

- `StringBuffer` 是可变的，允许在其生命周期内修改其内容而不创建新的对象。
- `StringBuffer` 的方法都是同步的（synchronized），因此它是线程安全的，适合用在多线程环境中。
- 由于同步机制的存在，在单线程环境下，`StringBuffer` 的性能可能不如 `StringBuilder`，但在多线程环境下，它提供了必要的线程安全性。
- 适合在多线程环境中需要频繁修改字符串的情况下使用。

构造方法/方法:

与StringBuilder基本无异,但StringBuffer的方法某些是同步的