Java核心API String,StringBuilder,StringBuffer

# String字符串

String类型介绍：字符串是有字符数组组成的, 或者说字符串封装了一个字符数组。 String类采用final修饰，String对象为不可变对象，一旦在内存中创建，内容不能发生变化，若要对字符串内容改变，那么就会创建新对象。String对象创建，有静态和动态之分。

String str1 = "dadsadad";

String str2 = new String("dasdasdadad");

注意：动态字符串（new,变量连接，方法返回），静态字符串（字面量，常量，字面量常量表达式）都会在常量池中创建对象。动态字符串还会在堆中创建对象，引用指向堆中对象。

String常量池： JVM对字符串有一个限制，让字符串作为不变对象，这样就可以做到重用。事实上，当我们通过字面量，常量来初始化一个字符串时，JVM首先会从字符串的常量池(一个JVM内部维护的内存区域，用来保存已经创建过的字符串对象)中查询用来保存该字符串的对象是否存在，若存在则直接引用，若不存在则创建该字符串对象并存入常量池，然后引用它。

## 字符串常用API

|  |
| --- |
| #构造器  String(byte[] bytes)  String(byte[] bytes, int offset, int length)  String(char[] value)  String(char[] value, int offset, int count)    int length()获取长度  char charAt(int index) ;返回下标对应的字符  #返回与字符串str或者代码点cp匹配的第一个子串的开始位置，这个位置从索引0或者fromIndex开始计算。不存在str,返回-1  int indexOf(int ch)  int indexOf(int ch, int fromIndex)  int indexOf(String str)  int indexOf(String str, int fromIndex)  #返回与字符串str或者代码点cp匹配的最后一个子串的开始位置，这个位置从索引0或者fromIndex开始计算。不存在str,返回-1  int lastIndexOf(int ch)  int lastIndexOf(int ch, int fromIndex)  int lastIndexOf(String str)  int lastIndexOf(String str, int fromIndex)  String substring(int beginIndex) ：返回一个新的字符串，从指定位置到结束位置  String substring(int beginIndex, int endIndex) 返回一个新字符串，不包含endIndex)  String concat(String str) 将指定字符串连接到此字符串的结尾  String[] split(String regex) 根据给定正则表达式的匹配拆分此字符串。  String[] split(String regex, int limit) 根据给定正则表达式的匹配拆分此字符串。    toCharArray() 将此字符串转换为一个新的字符数组。  getBytes() 使用平台的默认字符集将此 String 编码为 byte 序列，并将结果存储到一个新的 byte 数组中  getBytes(Charset charset) 使用给定的 charset 将此 String 编码到 byte 序列，并将结果存储到新的 byte 数组。    String toLowerCase() : String 中的所有字符都转换为小写。  String toUpperCase()： String 中的所有字符都转换为大写。  String trim() ：返回字符串的副本，忽略前导空白和尾部空白。  String replace(char oldChar, char newChar) 返回一个新的字符串，它是通过用 newChar 替换此字符串中出现的所有 oldChar 得到的  String replaceAll(String regex, String replacement) 使用给定的 replacement 替换此字符串所有匹配给定的正则表达式的子字符串。  boolean equals(Object other)： 返回字符串是否和other相等  boolean equalsIgnoreCase(String other) : 返回字符串是否和other相等(忽略大小写)  boolean contains(CharSequence s) ：当且仅当此字符串包含指定的 char 值序列时（char字符组成序列)，返回 true。  boolean startsWith(String prefix) ：字符串是否以指定的前缀开始。  boolean endsWith(String suffix) 测试此字符串是否以指定的后缀结束。  int compareTo(String anotherString) 按字典顺序比较两个字符串  int compareToIgnoreCase(String str) 按字典顺序比较两个字符串，不考虑大小写。  int a = 120;  System.out.println(a);//"120"  System.out.println(String.valueOf(a)); |

3）字符串的常用方法

# StringBuilder、StringBuffer类

String类我们已经得知，它是不变对象，那么每当对内容修改时都会引发新对象的创建。那么当我们有一个需求是需要频繁修改字符串时，这样不仅不能减少内存的开销，返回会增加内存的开销。为此java为我们提供了一个专门用于修改字符串内容的类StringBuilder类， 该类封装了可变的字符串，换句话说，当我们需要改变字符串内容时，并不会创建新对象，而是在原对象基础上进行修改。从而减小了内存的开销。StringBuilder类的常用方法有：

|  |
| --- |
| StringBuilder append(String str)：追加字符串；  StringBuilder insert (int index，String s)：从指定下标位置插入字符串；  StringBuilder delete(int start，int end)：从起始下标删除字符串到end结束下标，end要大于等于start，可以超出数组总长度；  StringBuilder replace(int start，int end，String str): 替换字符串；  StringBuilder reverse()：字符串反转。  String toString()：转化为字符串。 |
| #Java 的字符串连接的过程是利用 StringBuilder实现的  String s = "AB";  String s1 = s+"DE";  //String s1=new StringBuilder(s).append("DE").toString(); |

StringBuffer类，是JDK早期的版本，StringBuffer是线程安全的可变字符序列。虽然在任意时间点上它都包含某种特定的字符序列，但通过某些方法调用可以改变该序列的长度和内容。可将字符串缓冲区安全地用于多个线程。可以在必要时对这些方法进行同步，因此任意特定实例上的所有操作就好像是以串行顺序发生的，该顺序与所涉及的每个线程进行的方法调用顺序一致。StringBuffer 上的主要操作是 append 和 insert 方法，可重载这些方法，以接受任意类型的数据。每个方法都能有效地将给定的数据转换成字符串，然后将该字符串的字符追加或插入到字符串缓冲区中。

通常，如果 sb 引用 StringBuilder 的一个实例，则 sb.append(x) 和 sb.insert(sb.length(), x) 具有相同的效果。只要发生有关源序列（如在源序列中追加或插入）的操作，该类就只在执行此操作的字符串缓冲区上而不是在源上实现同步。每个字符串缓冲区都有一定的容量。只要字符串缓冲区所包含的字符序列的长度没有超出此容量，就无需分配新的内部缓冲区数组。如果内部缓冲区溢出，则此容量自动增大。

从 JDK 5 开始，为该类补充了一个单个线程使用的等价类，即 StringBuilder。与该类相比，通常应该优先使用 StringBuilder 类，因为它支持所有相同的操作，但由于它不执行同步，所以速度更快。 java.lang.StringBuilder一个可变的字符序列。此类提供一个与 StringBuffer 兼容的 API，但不保证同步。该类被设计用作 StringBuffer 的一个简易替换，用在字符串缓冲区被单个线程使用的时候（这种情况很普遍）。如果可能，建议优先采用该类，因为在大多数实现中，它比 StringBuffer 要快。

# 字符串总结

Java 中代表字符串的有: char[] String StringBuilder

* char[] 只是字符数据, 没有算法, 可以自己实现.。 如果优化的好, 性能最优! 开发效率低
* String 是不可变字符串, 是最常用的字符串, 界面文本数据。就是使用 String输出的. API经常返回新String对象, 性能稍差.

如: System.out.println(120)

实际上 System.out.println(String.valueOf(120));

* StringBuilder 是可变字符串, 字符串的内容计算, 建议采用StringBuilder实现. 性能好.