Java中的String总结

1. String 类是一个public 的**final类**，不可以被**继承**。字符串在Java中使用相当广泛。与字符串相关比较近的知识有：**String类、****StringBuffer类、StringBuilder类、正则表达式（regular expression即regex）。StringBuffer类、StringBuilder类也是public修饰的final类，不可以被继承。**
2. **字符串是一个常量，字符串String对象一旦初始化后，就不可以改变了，**两个或者多个字符串可以叠加到一个**新的字符串对象**中去，字符串用 + 号拼接。**（之所以可以在字符串后面进行添加，是因为创建了新的字符串对象，而原有的字符串对象没有改变）。**如果需要对字符串进行**增删**等操作，需要使用**StringBuffer或者StringBuilder**。
3. JAVA中的String的大小没有限制，只要是内存能够存下即可。
4. 例子：String str = new String("Shandong University of ");

str = str+"Science and Technology.";

解析：**str**只不过是一个String类类型引用变量，这个**类类型引用**首先指向了字符串"Shandong University of "，第二句等号右边创建了新的字符串"Shandong University of Science and Technology."，从而类类型引用str又指向了这个新的字符串对象。

1. **单引号和双引号的区别：**

**单引号只能包括一个字符（不能没有字符），而双引号可以包括任意多个字符，包括1个甚至没有也可以。**

**例子：System.out.println(****"a"+1); System.out.println('a'+1);**

**结果为： 第一句 输出 a1 第二句输出为 98**

**System.out.println(""+1); //双引号中可以什么都没有，但是单引号中必须有字符**

**System.out.println(' '+1);//注意区别空格字符与没有字符的区别，输出33（有空格）。**

1. **任何数据和字符串相拼接，都会转换成字符串**。因此如果想把其他数据类型的数据转换成字符串，最简单的方法是另其与**""相加即可，即xxx+""；""是一个空的字符串。另外，其他方法就是利用相应数据类型对象的转换方法如最常见的就是toString方法。**
2. **字符串的两种定义方法：**
   1. 第一种：String str = “abc”;
   2. 第二种：String str = new String(“abc”);//通过**各种构造方法**
3. 两种定义方法的区别：

（1） 字面值方式：

String s1 = "abcd"; String s2 = "abcd";

System.out.println(s1 == s2); //返回结果是 true

这种简单的定义方法，**只在字符串常量池（string pool）中创建唯一对象（字符串常量池不能存在相同字符串的多个对象，体现唯一性），String类类型引用指向字符串常量池中的对象。**在创建一个字符串时，首先判断**字符串池**中是否有此字符串，如果没有就在字符串常量池中创建一个字符串对象，如果已经存在，则直接把存在的字符串对象的地址赋给引用即可，故s1和s2的地址相同。（双等号 == 判断的是地址，equals方法对于字符串来说是判断内容的。）1. (由于在程序编写时，会大量使用字符串string，而对于重复的字符串不再分配存储空间，从而实现优化)

（2） 利用**new方式**(即利用**构造方法**的方式):

String s3 = **new** String("abcd"); String s4 = **new** String("abcd");

System.***out***.println(s3 == s4); //返回结果是false

这种方式是在**字符串常量池**和**堆内存**中都会创建对象，**String类类型引用**指向的是堆内存中的字符串对象。在字符串常量池中，**同一个字符串**不会存在多个对象，但是在堆内存中可以存在**相同字符串**的多个字符串对象。S3和s4就是指向了堆内存中的两个对象，故返回false。

采用new的方式，首先是在**字符串池中**查找有没有此字符串对象，如果有，就不在字符串池中创建此对象了，直接在**堆内存**中创建一个字符串对象，如果没有，则先在字符串池中创建这个字符串对象，然后**在堆内存中**再创建一个字符串对象，最后都是把**堆内存中的对象地址**返回来。注意：返回的是**堆内存中**的对象，所以new一次，就会在堆内存中新建一个对象，地址肯定不同，但是利用**字面赋值**的方法，返回的是字符串池中的地址，如果字符串内容相同，则就是同一个地址。

（3） 总结： 第一种方法是在**字符串常量池**中创建唯一的一个对象；第二种方法（**利用new**）是在**字符串池**和**堆内存**中都会创建字符串对象，**一个new对应一个字符串对象**。使用new方式，无论字符串内容是否相同，引用都不会相同，因为new创建的一定是在**堆内存**新建的对象。

1. **注意三者的区别：String s1= new String(); 与 String s2 = “”; 和 String s3 = null;**

前两者都创建了对象，只不过是**空字符**串，引用已经分配了**对象地址**，第一个是指向**堆内存中的地址**，第二个是**指向字符串常量池中的地址**；而第三个没有创建对象，只是定义了一个String类类型引用，**引用值为空**，不指向任何字符串对象。

例子：String str1 = "";

String str2 = "";//**把str1的地址直接传给str2**

String str3 = **new** String();

String str4 = **new** String();**//在堆内存中创建两个空字符串对象**

System.***out***.println(str1 == str2);

System.***out***.println(str1 == str3);

System.***out***.println(str3 == str4);**//在堆内存中创建两个空字符串对象**

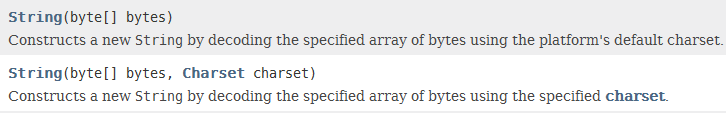
都是空字符串，且只有两个都是字符串池中的才是地址相同，指向同一个对象。

1. **利用new关键字创建字符串对象，就是通过调用String类的构造方法创建新的String对象，下面就介绍String类的构造方法（全部构造方法都在下面）：**
2. **空参数的构造方法：**



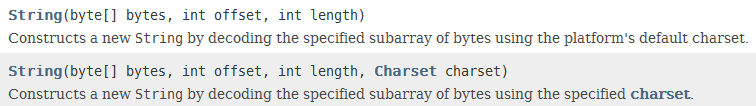
1. **利用字节数组 bytes[] 创建字符串对象：**

一个是利用默认的编码表，一个是可以指定编码表。





**对上面的进一步优化，可以指定字节数组的起始位置和长度：**

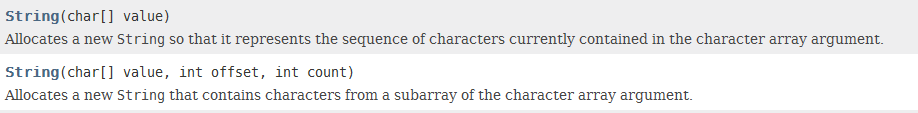




**遗留问题：Charset charset 与 String charsetName 的区别。**

1. **利用字符数组char[] 创建字符串对象：**

由于字符数组中存放的就是字符，不需要任何的编码表，故只有两个构造方法：

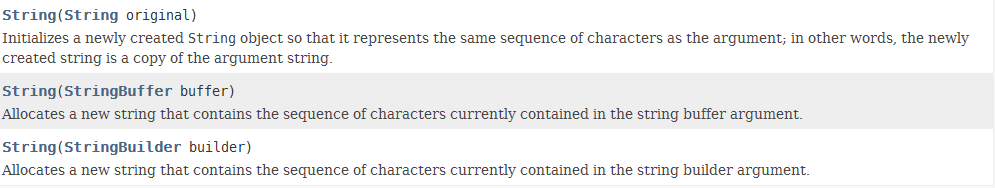


1. **利用int[] 数组创建字符串对象：**

**利用Unicode编码表对int 数据进行解码，只有这一个构造方法，对int数组的部分进行解码：**



1. **利用已存在的String对象、StringBuffer对象、StringBuilder对象创建新的字符串对象：**



1. **String的三个方法：①charAt（int index）用来判断index处的字符；**

**②indexOf（char ch）、indexOf（ ch，int fromIndex）indexOf（str）indexOf（str，fromIndex）用来判断某个字符或者字符串的索引号。**

**寻找字符或字符串出现的位置，对于indexOf是返回首次出现的位置，如果加上last就是对应最后一个出现此字符（串）的位置即lastIndexOf。**

注意：**charAt和indexOf方法是对应的**，但是valueOf与他们是没有关系的。

**③valueOf（Object object）（静态方法）可以把8种原生数据类型值、字符数组以及任意对象转换成字符串。**

**这个方法可以实现由基本数据类型向String转换，对于对象来说，就是调用的对象的toString方法。如：valueOf的源代码：**



1. **String类的一般方法进行分类介绍：**
2. **获取方法：**
   1. **length()方法：获取字符串的长度（字符的个数），返回int值；**

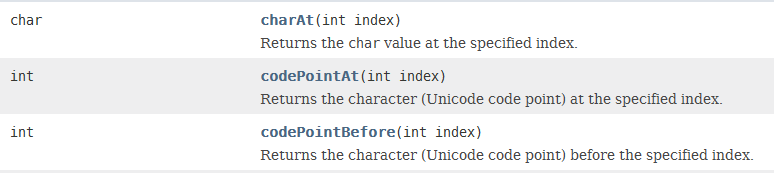
注意：字符串需要使用length（）方法获取，而**数组是利用它的属性length**得到长度值。

**②codePointCount**(int beginIndex,int endIndex)方法：获取指定字符串的一部分中Unicode编码的个数:



注：unicode 采用**双字节**对字符进行编码，统一的字符编码标准。

* 1. **根据索引值获取字符或者其Unicode值； char charAt(int index)**

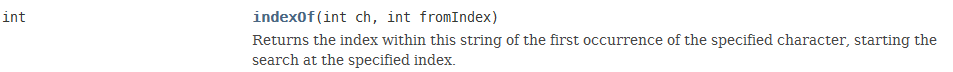


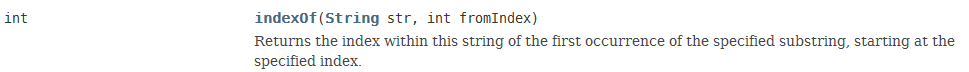
* 1. **indexOf（int ch）根据字符或子字符串获取第一次出现的位置:** 
     1. **获取指定字符或字符串第一次出现的位置：int值，如果没有，则返回-1.**





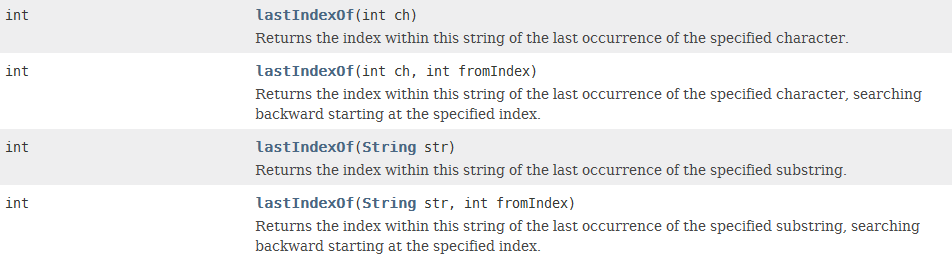
* + 1. **进一步指定，从哪个位置开始查找：返回值仍然是第一次出现的索引值，如果没有则返回-1：**







* 1. **相对应地，倒过来查找索引：同上述正着寻找索引。本质也是正着查找，只不过是把最后一个返回过来而已。**



* 1. **获取字符串中的一部分：**
     + 1. **subString（int beginIndex）方法：默认从指定起点到结束，返回一个新的字符串。**



* + - 1. **subString（int beginIndex，int endIndex）方法：指定起点和终点，注意是左闭右开区间，返回一个新的字符串对象。如果begin和end 相同，返回一个空字符串，如果begin大于end或者begin为负值，则会抛出异常**[IndexOutOfBoundsException](mk:@MSITStore:E:\JAVA\JAVA%20JIHE\API帮助文档\CHM帮助文档\jdk-8u60-docs-all_API.chm::/api/java/lang/IndexOutOfBoundsException.html)。



* + - 1. **charSequence（int beginIndex，int endIndex）方法：**

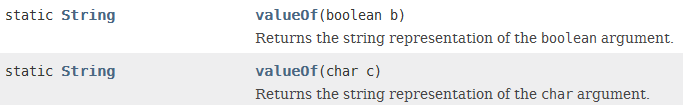


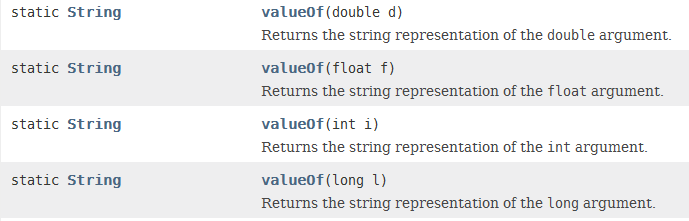
**注： CharSequence 是一个接口，String类实现了这个接口，父类接口引用可以指向子类的对象，所以参数可以是一个字符串对象。此外，直接实现此接口的类有CharBuffer, Segment, String, StringBuffer, StringBuilder，显然参数可以是这些类中的对象。**

1. **转换：**
   1. **静态方法valueOf（xxx）方法：这些方法都是静态方法，返回值都是字符串对象。**

**记住：将一般原型数据及对象转换成字符串，直接利用String.valueOf(xxx)方法即可实现。**

**将基本数据类型（除了byte和short之外）的数据转换成字符串对象。**

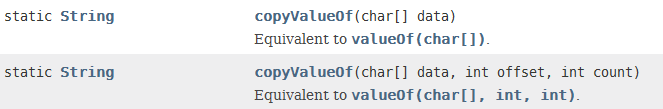




**将字符数组转换成字符串：返回一个字符串对象；**



**还有两个功能一样的方法：**



**此外，可以接收一切对象：返回一个该对象的字符串表示形式及调用对象的toString方法；**



* 1. **切割方法：将字符串转换成字符串数组：按照正则表达式对字符串进行切割，返回字符串数组。** [**regular expression**](mk:@MSITStore:G:\Android学习资料\CHM帮助文档\jdk-8u60-docs-all_API.chm::/api/java/util/regex/Pattern.html#sum)**.正则表达式regex**
     1. [**split**](mk:@MSITStore:G:\Android学习资料\CHM帮助文档\jdk-8u60-docs-all_API.chm::/api/java/lang/String.html#split-java.lang.String-)**(**[**String**](mk:@MSITStore:G:\Android学习资料\CHM帮助文档\jdk-8u60-docs-all_API.chm::/api/java/lang/String.html)**regex)方法：**



* + 1. [**split**](mk:@MSITStore:G:\Android学习资料\CHM帮助文档\jdk-8u60-docs-all_API.chm::/api/java/lang/String.html#split-java.lang.String-int-)**(**[**String**](mk:@MSITStore:G:\Android学习资料\CHM帮助文档\jdk-8u60-docs-all_API.chm::/api/java/lang/String.html)**regex, int limit)方法：指定最大切割片数，如果切割片数到了limit值，后面的就不再切割，算作一个。**



**例子：String str = new String("abc.sade.gaha.xixi");**

**String[] strarr = str.split("\\.");**

**String[] strarr2= str.split("\\.",2);**

**strarr数组4个元素，分别为abc、sade、gaha、xixi；而strarr2只有两个元素，为abc、sade.gaha.xixi。**

* 1. **将字符串转换成字符数组：**

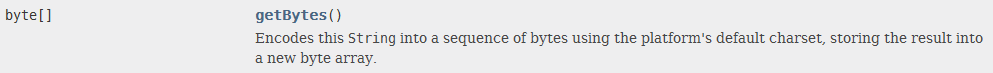
**①toCharArray（）方法：返回一个字符数组。**



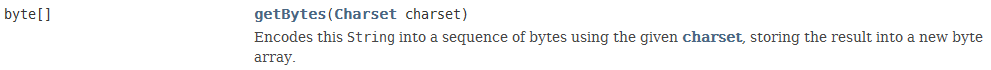
**② getChars(int,int,char[],int)方法**：注意返回值为空void，用于存储字符的数组作为参数穿进去，且需要传进去字符串的开始和结束位置及字符数值的开始存放的位置。

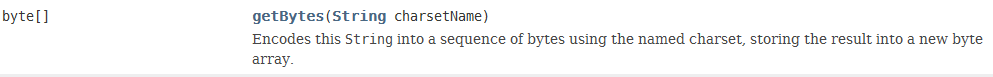


* 1. **将字符串转换成字节数组 getBytes（）方法：返回一个字节数组（按照默认的编码表）。注：每个中文占用两个字节。**



**进一步，可以指定编码表进行编码：**





* 1. **大小写转换：返回一个新的字符串对象，原字符串对象仍然存在。**





**指定Local：**

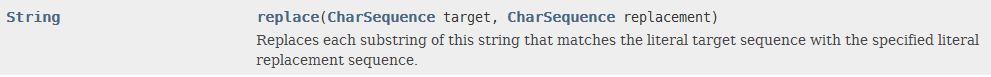




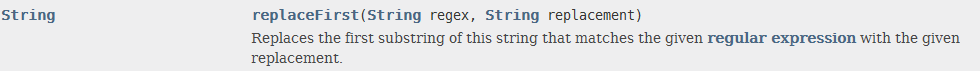
* 1. **将字符串中的内容进行替换replace方法：**
     1. [**replace**](mk:@MSITStore:G:\Android学习资料\CHM帮助文档\jdk-8u60-docs-all_API.chm::/api/java/lang/String.html#replace-char-char-)**(char oldChar, char newChar)方法，将所有旧字符替换成某个字符，返回一个新的字符串。**



**提升：参数可以是CharBuffer, Segment, String, StringBuffer, StringBuilder中任意的对象。**



* + 1. [**replace**](mk:@MSITStore:G:\Android学习资料\CHM帮助文档\jdk-8u60-docs-all_API.chm::/api/java/lang/String.html#replace-char-char-)**(String regex,String replacement) 方法：把第一个或者所有的满足指定正则表达式的字符串替换成新的字符串：**





1. **判断方法：返回值是boolean值：**
2. **isEmpty（） 方法：返回boolean值。**



1. **equals（String anotherString）方法：判断两个字符串的内容是否相同，是对Object类的equals方法的复写，返回boolean值。**



1. **忽略大小写的equals方法：equalsIgnoreCase(String anotherString)**



1. **matches（）方法：返回boolean值。**

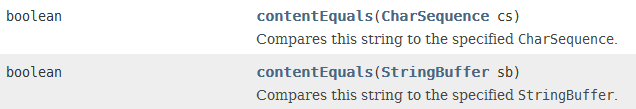


1. **contains（CharSequence s）方法：字符串中是否包含指定字符串。参数可以是String, StringBuffer, StringBuilder,CharBuffer, Segment中的任意一个对象。**



**注： CharSequence 是一个接口，String类实现了这个接口，父类接口引用可以指向子类的对象，所以参数可以是一个字符串对象。此外，直接实现此接口的类有CharBuffer, Segment, String, StringBuffer, StringBuilder，显然参数可以是这些类中的对象。**

1. **内容是否相等的方法：参数可以是String, StringBuffer, StringBuilder,CharBuffer, Segment中的任意一个对象。**



1. **startsWith（**[String](mk:@MSITStore:G:\Android学习资料\CHM帮助文档\jdk-8u60-docs-all_API.chm::/api/java/lang/String.html) prefix**） 方法：判断是否以某字符串开头的。**



（注：prefix ['priːfɪks] n. 前缀，suffix ['sʌfɪks] n. 后缀）

与此相对应的是，**endsWith（）方法**，判断是否以某字符串结尾的。



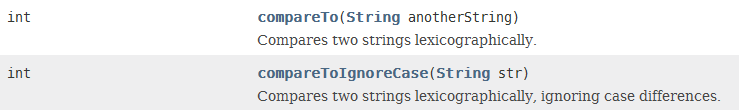
**进一步，判断开头的还有一个方法，可以指定开始位置（判断结束的没有哦）：**



1. **比较方法：**

**compareTo（String anotherString）与compareToIgnoreCase（String str）**

**返回值为int值，若为0，则相等，若为正数，则前面的大，若为负数，则后面的大。实际上，返回的int值就是字符串第一个不相等字符对应的编码值之差（前面字符串的字符减去后面的字符串的对应字符）。**



从前到后，按照字母的顺序比较：**如果前面的小于后面的，则返回负数；**

**如果前面的大于后面的，则返回正数；如果相等则返回0.**

**因为字符串本身就是一个对象，对象不能利用比较符号比较，必须通过方法进行比较，不能利用“abc”> “adc”；这是非法的。**

1. **intern（） 方法：返回一个字符串池中的字符串对象，即该字符串对象对应的字符串池中的字符串对象。（若此字符串本身就是字符串池中的，那么返回的就是自己本身，如果是堆内存中，则返回对应字符串池中的对象。证明见下面例子）**



**例如：String str1 = "abcdef";**

**System.out.println(str1.intern()==str1);//结果为true，说明….**

**String str2 = new String("abcdef");**

**System.out.println(str2.intern()==str2);//结果为false，说明…..**

**System.out.println(str1==str2);//结果为false，说明…..**

**System.out.println(str1==str2.intern());//结果为true，说明……**

注意：需要掌握利用字面值方式和new创建字符串对象的区别，见前面。

1. **concat（String str）方法：连接两个字符串，功能同+号，只不过显得更专业，返回的是一个新的字符串对象，原字符串仍然存在。**



1. **trim（）方法：去除字符串两端的空字符，返回一个新的字符串对象。**



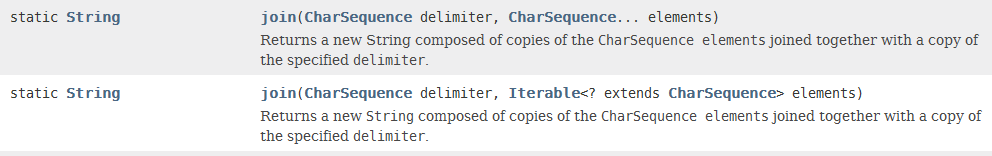
1. **hashCode（）方法：返回该字符串对象的哈希值，int类型。**

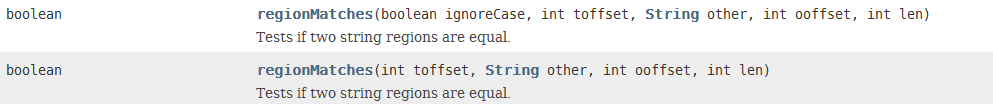


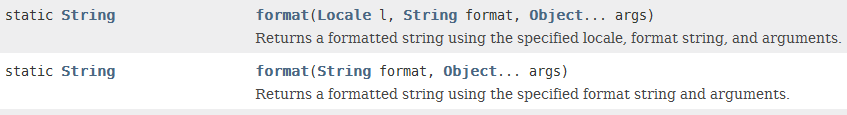
**toString（）方法：返回字符串内容。（打印输出字符串对象时，默认调用的就是其toString（）方法）**



1. **未知方法：**







1. **小总结：**
2. **字符串和字符数组之间的转换**
   1. **字符数组---🡪 字符串**

**利用构造方法，把字符数组做参数传进去，直接new出字符串对象即可；**

* 1. **字符串---🡪字符数组**

**利用toCharArray（）方法或者getChars(int,int,char[],int)方法。**

1. **字符串和字节数组之间的转换**
   1. **字节数组---🡪 字符串**

**利用构造方法，把字节数组做参数传进去，直接new出字符串对象即可；**

* 1. **字符串---🡪字节数组**

**利用getBytes（）方法。**

1. **字符串和基本数据类型之间的转换（都是静态方法）**
   1. **基本数据类型---🡪 字符串（valueOf方法是String的静态方法，可以利用类名调用）**

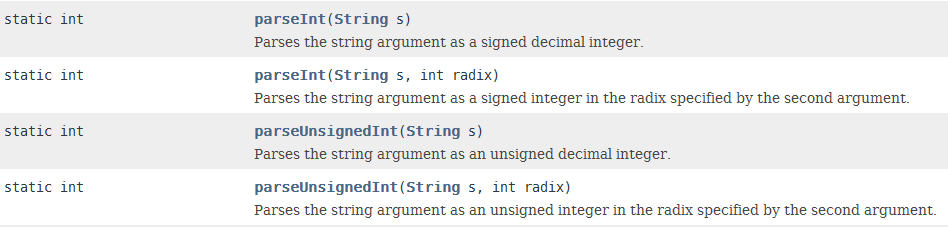
**利用valueOf（xxx）方法即可。**

**或者利用基本数据类型对象的toString（）方法。**

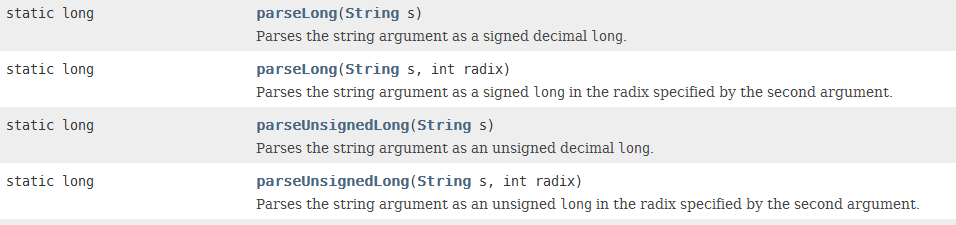
**按照面向对象的思想，要想把基本数据类型转换成字符串，那么基本数据类型的包装类对象自身最清楚，所以可以利用他们自己的toString方法，当然对于String对象来说，其他东西想变成它，它也可以提供方法，就是静态方法valueOf。**

* 1. **字符串--**🡪**基本数据类型-（parseXxx（String str）都是静态方法，利用相应的包装类即可） （除了char之外）**

**如int类型——>字符串利用parseInt（String str）方法：**



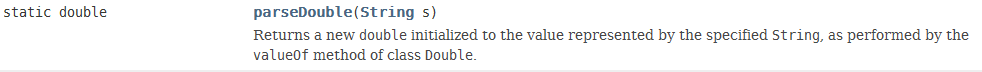
**long类型：**



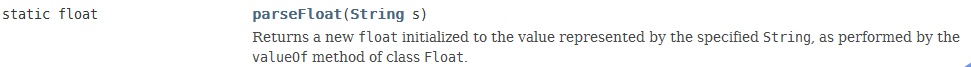
**boolean类型：**



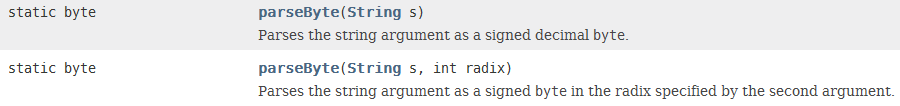
**double类型：**



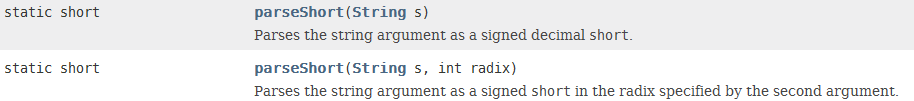
**float类型：**



**byte类型：**



**short类型：**



1. **练习题：**
2. **写出实现trim（）功能的代码：去除字符串两端的空格。**

**思路：a. 找出第一个非空字符的索引；b. 找出最后一个非空字符的索引；**

**c． 截取这段子字符串。**

**注意：需要健壮性判断，若字符串是空的，就返回空字符串；若只有一个空字符，那么就返回一个空字符的字符串。**

**private static String myTrim3(String str) {**

**if(str.equals("")) {**

**return "";**

**}**

**if(str.equals(" ")) {**

**return " ";**

**}**

**int indexmin = 0;**

**int indexmax = str.length()-1;**

**for(;str.charAt(indexmin) == ' ';indexmin++);**

**for(;str.charAt(indexmax)==' ';indexmax--);**

**str = str.substring(indexmin,indexmax+1);**

**return str;**

**}**

1. **判断两个字符串的最大相同子字符串。**

**思路：**

* + - 1. **既然是找最大的相同子字符串，那么最大的可能就是较短的字符串本身，把这个较短的子字符串的长度逐渐减少，判断长字符串中是否包含它即可；**
      2. **利用contains方法进行判断，长的字符串是否包含短字符串的子字符串；**
      3. **第一步：需要找出短的字符串，利用length（）方法比较即可；**
      4. **第二步：利用两次嵌套for循环，外循环控制子字符串的长度，内循环控制短字符串在固定长度下的子字符串的个数；**
      5. **第三步：循环体是利用contains方法判断，若为真，则直接return xx，结束；**
      6. **第四步：若不存在相同元素，则返回空的字符串或提示“不存在相同字符”。**

**private static String findMaxSame(String str1, String str2) {**

**if(str1.length() > str2.length()){**

**String temp = str1;**

**str1 = str2;**

**str2 = temp;**

**}//保证str1 是指向短的字符串**

**System.out.println(str1 +"\r\n"+str2);**

**String str3 = null;**

**for (int i = str1.length(); i > 0 ; i--) {**

**for (int j = 0; j <= str1.length()-i ; j++) {**

**str3 = str1.substring(j,j+i);**

**if(str2.contains(str3)) {**

**//contains方法可以利用str2.indexOf(str3) != -1替换**

**return str3;**

**}**

**}**

**}**

**return "不存在相同字符";**

**}**

**}**

1. **一个字符串中某个子字符串出现的次数。**

**分析：**

1. **利用切割的思想，把字符串当做regex正则表达式，切成字符串数组，然后根据数组的元素个数，判断出现的次数；由于切出的片数过多，不是很好；而且会带来很多问题，不存在regex仍然会切出一个来，而档regex做尾时，后面的不会有数组元素，造成数目混乱，这种是不可行的；**
2. **利用substring方法，找出该字符串的许多与此等长度的字符串，然后一一比较是否相等，从而计数，这样会切出很多子字符串，不是很好，且是计数混乱，如ababa中寻找aba出现次数，可能会计数出错。**
3. **利用indexOf（String str,int fromIndex）方法：利用while循环，返回-1就结束，每次获取出子字符串首次出现的位置，然后让其再利用指定起始位置从新查找，直到返回 -1结束。同时利用计数器计数，最后返回数目。**

**private static int numSubstring(String str, String str1) {**

**int count = 0;**

**int index = 0;**

**int len = str1.length();**

**while((index = str.indexOf(str1,index ))!=-1) {**

**count ++;**

**index += len;**

**if(index >= str.length()) {**

**break;**

**}**

**}**

**return count;**

**}**

**}**

1. **面试题：**

**public class Example {**

**String str = new String("good");**

**char[] ch = { 'a', 'b', 'c' };**

**public static void main(String args[]) {**

**Example ex = new Example();**

**ex.change(ex.str, ex.ch);**

**System.out.print(ex.str + " and ");**

**System.out.print(ex.ch);**

**}**

**public void change(String str, char ch[]) {**

**str = "test ok";**

**ch[0] = 'g';**

**}**

**}**

**请问这个程序为什么输出是什么？**

**答案：**

**引用分为三种：数组型引用、类类型引用、接口型引用，引用中存放的就是地址。String类的引用属于类类型引用。当String类类型引用作为实参时，把地址传给形参，形参和实参引用指向同一个字符串，形参变量的作用范围仅仅是方法体。由于字符串是常量，引用是一个变量，str = "test ok";这句语句使（局部）引用指向了新创建的字符串对象，但是不会影响方法体外的实参的指向。传入数组时，传入的也是地址，直接对数组进行修改，故会改变数组的值。**

**这个面试题归根到底考察的是字符串常量不可改变，但是字符串引用变量可以指向不同的字符串对象。**

对于引用来说，传递的都是地址，但是字符串一旦创建不可改变（字符串不可改变，但是字符串引用变量可以改变），如果字符串不同，就是说明创建了新的字符串，所以字符串的引用（即字符串形参）就变了指向；但是对于数组而言，数组引用没有发生改变，始终指向同一数组。

变式训练1.只改变方法小括号里**形参名字**，此时方法体中的str就是类的引用了，不再是局部引用，所以结果就不一样了。

2.如果小括号和方法体中的str都变为str11，结果还是不变。

3.对于ch数组名，变与不变，变一个或者两个都一样，都会修改数组元素。

public void change(String str11, char ch11[]) {

//str11 = "test ok";//此时是局部引用变量，方法一结束，此变量就会消失了，并不会改变对象的成员变量str的指向

str = "test ok";//此时直接调用的就是对象的成员变量了

ch11[0] = 'g';//无论数组名字如何改变，因为操作数组的地址没有改变，所以操作的始终是同一个数组

}

}

1. **考试题目：**

String s1 = "abcd";

String s2 = "abcf";

String s5 = "abcd";

String s3 = s2.replace('f','d');

String s4 = s5.replace('g','d');

System.***out***.println(s1 == s5);

System.***out***.println(s1 == s3);

System.***out***.println(s1 == s4);

请问：输出结果是什么？

true

false

true

解析：s1、s2、s5都是通过在字符串常量池中创建对象，显然第一行输出true；对于replace方法，如果有字符替换了，则就是有了新的字符串，那么就会等同于new一个String，在堆中创建一个新对象，再将地址赋予s3，显然字符串池中的对象引用s1与s3不同；**但是如果replace方法并没有完成替换，即找不到需要替换的字符时，相当于字符串没有发生任何变化，所以也就没有创建新的字符串，还是将旧的字符串的地址赋予引用s4；**显然s4与s1指向同一个字符串对象。

1. String s5 = **new** String("abcd");

String s4 = s5.replace('g','d');

System.***out***.println(s5 == s4);**//结果为true，找不到**

1. String s6 = "abcd";

String s7 = s6.replace('g','d');

System.***out***.println(s6 == s7);**//结果也为true**

**由A 、B 可以看出，replace方法如果没有改变字符串，则就没有创建新的字符串对象。**

1. String s1 = new String("abcd");

String s2 = "abcf";

String s3 = s2.replace('f','d');

System.***out***.println(s1 == s3);/**/结果为false**

1. String s11 = "abcd";

String s22 = "abcf";

String s33 = s2.replace('f','d');

System.***out***.println(s11 == s33);//**结果为false**

**从C、D可以看出，replace方法如果改变了字符串，则是通过new的方式创建一个新的字符串，当然新创建的字符串与原来的无论是字符串池的还是堆内存中的都不一样了。**

1. **交换练习：**

**package sdk.November25.am;**

**public class SwapStringTest1 {**

**public static void main(String[] args) {**

**String s1 = "djkjk";**

**String s2 = "abdcgde";**

**String[] strarr = {"aaaa","bbbb","cccc","dddd","eeee"};**

**System.out.println(s1 + " " +s2);**

**swap(s1,s2);**

**System.out.println(s1 + " " +s2);**

**System.out.println("&&&&&&&&&&&&&&&&&&&&&");**

**System.out.println(strarr[1] + " " + strarr[3]);**

**swapStrArray(strarr,1,3);**

**System.out.println(strarr[1] + " " +strarr[3]);**

**}**

**//形参和实参之间的传递只是单向的**

**private static void swapStrArray(String[] strarr, int i, int j) {**

**String temp = null;**

**temp = strarr[i];**

**strarr[i] = strarr[j];**

**strarr[j] = temp;**

**}**

**//利用数组可以实现交换的目的**

**private static void swap(String s1, String s2) {**

**String temp = null;**

**temp = s1; s1 = s2; s2 = temp;**

**}**

**}**

1. **字符串数组排序：**

**思路：**

1. **字符串大小比较利用compareTo方法；**
2. **排序利用选择法或者冒泡法；**
3. **交换两个字符串，利用字符串数组可以实现；**

**自定义一个交换两个字符串的方法，注意参数一定是数组名和需要交换的索引号，不能是数组元素（字符串），因为字符串作为参数，只是一种值传递的方式，并没有改变原来的字符串。实参字符串把内容传给形参字符串，相当于在交换方法中创建了新的字符串，所以没有改变实参的字符串的顺序。**

**package StringTest1;**

**public class SortStringArray {**

**public static void main(String[] args) {**

**String[] strarr = {"abcdd","msfdsg","hsjakjgkdsj","djkshhjdsah","jdhhjhsad","jsdhjah"};**

**sortStringArrayMethod(strarr);**

**for (int i = 0; i < strarr.length; i++) {**

**System.out.println(strarr[i]);**

**} }**

**private static void sortStringArrayMethod(String[] strarr) {**

**// 冒泡法**

**for (int i = 1; i < strarr.length; i++) {**

**for (int j = 0; j < strarr.length - i; j++) {**

**if(strarr[j].compareTo(strarr[j+1]) > 0) {**

**swapSring(strarr,j,j + 1);**

**} } } }**

**private static void swapSring(String[] strarr, int i, int j) {**

**String temp = null;**

**temp = strarr[i];**

**strarr[i] = strarr[j];**

**strarr[j] = temp;**

**} }**

**毕老师讲的方法：**

**package sdk.November25.am;**

**public class StringSort {**

**public static void main(String[] args) {**

**String[] str = {"jgdgds","djsj","sdjkhjkg","abc","aad","z","zzzz"};**

**printStringArray(str);**

**sortStringArray(str);**

**printStringArray(str);**

**}**

**private static void sortStringArray(String[] str) {**

**for(int i = 1;i < str.length ;i ++)**

**{**

**for(int j = 0;j < str.length - i;j++)**

**{**

**if(str[j].compareTo(str[j+1]) > 0)//字符串的比较大小利用compareTo方法**

//swapString(str[j],str[j+1]);//仍然是以值传递的方式传进去的，所以必须是传递数组名才可以

**swapString(str,j,j+1);**

**}**

**}**

**}**

**private static void swapString(String[] str,int i,int j) { //一定是把数组名传进//去private static void swapString(String string, String string2) {**

**//}这中字符串作为参数的方法是错误的，传进去后，就变成了局部变量**

**String temp = null;**

**temp = str[i];**

**str[i] = str[j];**

**str[j] = temp;**

**}**

**private static void printStringArray(String[] str) {**

**for(int i = 0; i < str.length; i++) {**

**if(i != str.length-1)**

**System.out.print(str[i] + " ,");**

**else**

**System.out.print(str[i]);**

**}**

**System.out.println();**

**}**

**}**