java.lang.String類的使用

1. **概述**

String:字符串，使用一對""引起來表示。

1.String聲明為final的，不可被繼承

2.String實現了Serializable接口：表示字符串是支持序列化的。

實現了Comparable接口：表示String可以比較大小

3.String內部定義了final char[] value用於存儲字符串數據

4.通過字面量的方式(區別於new）給一個字符串賦值，此時的字符串值聲明在字符串常量池中。)

5.字符串常量池中是不會存儲相同內容(使用String類的equals()比較,返回true)的字符串的。

1. **String的不可變性**
   1. 說明

1.當對字符串重新賦值時，需要重寫指定內存區域賦值，不能使用原有的value進行賦值。

2. 當對現有的字符串進行連接操作時，也需要重新指定內存區域賦值，不能使用原有的value進行賦值。

3. 當調用String的replace()方法修改指定字符或字符串時，也需要重新指定內存區域賦值，不能使用原有的value進行賦值。

* 1. 代碼舉例

**public** **void** test1(){

String s1 = "abc";//字面量的定義方式

String s2 = "abc";

s1 = "hello";

System.***out***.println(s1 == s2);//true 比較s1和s2的地址值

System.***out***.println(s1);//hello

System.***out***.println(s2);//abc

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

String s3 = "abc";

s3 += "def";

System.***out***.println(s3);//abcdef

System.***out***.println(s2);

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

String s4 = "abc";

String s5 = s4.replace('a', 'm');

System.***out***.println(s4);//abc

System.***out***.println(s5);//mbc

}

* 1. 圖示



1. String實例化的不同方式
   1. 方式說明

方式一：通過字面量定義的方式

方式二：通過new + 構造器的方式

* 1. 代碼舉例

//通過字面量定義的方式：此時的s1和s2的數據javaEE聲明在方法區中的字符串常量池中。

String s1 = "javaEE";

String s2 = "javaEE";

//通過new + 構造器的方式:此時的s3和s4保存的地址值，是數據在堆空間中開辟空間以後對應的地址值。

String s3 = **new** String("javaEE");

String s4 = **new** String("javaEE");

System.***out***.println(s1 == s2);//true

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

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

System.***out***.println(s3 == s4);//false

* 1. 面試題

String s = new String("abc");方式創建對象，在內存中創建了幾個對象？

ANS:兩個,一個是堆空間中new結構，另一個是char[]對應的常量池中的數據："abc"

* 1. 圖示



1. 字符串拼接方式賦值對比

4.1說明

1.常量與常量的拼接結果在常量池。且常量池中不會存在相同內容的常量。

2.只要其中有一個是變量，結果就在堆中。

3.如果拼接的結果調用intern()方法，返回值就在常量池中

4.2代碼舉例

String s1 = "javaEE";

String s2 = "hadoop";

String s3 = "javaEEhadoop";

String s4 = "javaEE" + "hadoop";

String s5 = s1 + "hadoop";

String s6 = "javaEE" + s2;

String s7 = s1 + s2;

System.***out***.println(s3 == s4);//true

System.***out***.println(s3 == s5);//false

System.***out***.println(s3 == s6);//false

System.***out***.println(s3 == s7);//false

System.***out***.println(s5 == s6);//false

System.***out***.println(s5 == s7);//false

System.***out***.println(s6 == s7);//false

String s8 = s6.intern();//返回值得到的s8使用的常量池中已經存在的

“javaEEhadoop”

System.***out***.println(s3 == s8);//true

String s1 = "javaEEhadoop";

String s2 = "javaEE";

String s3 = s2 + "hadoop";

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

**final** String s4 = "javaEE";//s4:常量

String s5 = s4 + "hadoop";

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

1. 常用方法

int length()：返回字符串的長度： return value.length

char charAt(int index)： 返回某索引處的字符return value[index]

boolean isEmpty()：判斷是否是空字符串：return value.length == 0

String toLowerCase()：使用默認語言環境，將 String 中的所有字符轉換為小寫

String toUpperCase()：使用默認語言環境，將 String 中的所有字符轉換為大寫

String trim()：返回字符串的副本，忽略前導空白和尾部空白

boolean equals(Object obj)：比較字符串的內容是否相同

boolean equalsIgnoreCase(String anotherString)：與equals方法類似，忽略大小寫

String concat(String str)：將指定字符串連接到此字符串的結尾。 等價於用“+”

int compareTo(String anotherString)：比較兩個字符串的大小

String substring(int beginIndex)：返回一個新的字符串，它是此字符串的從beginIndex開始截取到最後的一個子字符串。

String substring(int beginIndex, int endIndex) ：返回一個新字符串，它是此字符串從beginIndex開始截取到endIndex(不包含)的一個子字符串。

boolean endsWith(String suffix)：測試此字符串是否以指定的後綴結束

boolean startsWith(String prefix)：測試此字符串是否以指定的前綴開始

boolean startsWith(String prefix, int toffset)：測試此字符串從指定索引開始的子字符串是否以指定前綴開始

boolean contains(CharSequence s)：當且僅當此字符串包含指定的 char 值序列時，返回 true

int indexOf(String str)：返回指定子字符串在此字符串中第一次出現處的索引

int indexOf(String str, int fromIndex)：返回指定子字符串在此字符串中第一次出現處的索引，從指定的索引開始

int lastIndexOf(String str)：返回指定子字符串在此字符串中最右邊出現處的索引

int lastIndexOf(String str, int fromIndex)：返回指定子字符串在此字符串中最後一次出現處的索引，從指定的索引開始反向搜索

注：indexOf和lastIndexOf方法如果未找到都是返回-1

替換：

String replace(char oldChar, char newChar)：返回一個新的字符串，它是通過用 newChar 替換此字符串中出現的所有 oldChar 得到的。

String replace(CharSequence target, CharSequence replacement)：使用指定的字面值替換序列替換此字符串所有匹配字面值目標序列的子字符串。

String replaceAll(String regex, String replacement)：使用給定的 replacement 替換此字符串所有匹配給定的正則表達式的子字符串。

String replaceFirst(String regex, String replacement)：使用給定的 replacement 替換此字符串匹配給定的正則表達式的第一個子字符串。

匹配:

boolean matches(String regex)：告知此字符串是否匹配給定的正則表達式。

切片：

String[] split(String regex)：根據給定正則表達式的匹配拆分此字符串。

String[] split(String regex, int limit)：根據匹配給定的正則表達式來拆分此字符串，最多不超過limit個，如果超過了，剩下的全部都放到最後一個元素中。

1. String與其它結構的轉換
   1. 與基本數據類型、包裝類之間轉換

String --> 基本數據類型、包裝類：調用包裝類的靜態方法：parseXxx(str)

基本數據類型、包裝類 --> String:調用String重載的valueOf(xxx)

**public** **void** test1(){

String str1 = "123";

// int num = (int)str1;//錯誤的

**int** num = Integer.*valueOf*(str1);

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

String str2 = String.*valueOf*(num);//"123"

String str3 = num + "";

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

* 1. 與字符數組之間轉換

String --> char[]:調用String的toCharArray()

char[] --> String:調用String的構造器

**public** **void** test2(){

String str1 = "abc123"; //題目： a21cb3

**char**[] charArray = str1.toCharArray();

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

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

}

**char**[] arr = **new** **char**[]{'h','e','l','l','o'};

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

String copyValueOf = str2.*copyValueOf*(arr);

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

}

* 1. 與字節數組之間轉換

編碼：String --> byte[]:調用String的getBytes()

解碼：byte[] --> String:調用String的構造器

編碼：字符串 -->字節 (看得懂 --->看不懂的二進制數據)

解碼：編碼的逆過程，字節 --> 字符串 （看不懂的二進制數據 ---> 看得懂）

說明：解碼時，要求解碼使用的字符集必須與編碼時使用的字符集一致，否則會出現亂碼。

**public** **void** test3() **throws** UnsupportedEncodingException {

String str1 = "abc123中國";

**byte**[] bytes = str1.getBytes();//使用默認的字符集，進行編碼。

System.***out***.println(Arrays.*toString*(bytes));

**byte**[] gbks = str1.getBytes("gbk");//使用gbk字符集進行編碼。

System.***out***.println(Arrays.*toString*(gbks));

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

String str2 = **new** String(bytes);//使用默認的字符集，進行解碼。

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

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

System.***out***.println(str3);//出現亂碼。原因：編碼集和解碼集不一致！

String str4 = **new** String(gbks, "gbk");

System.***out***.println(str4);//沒有出現亂碼。原因：編碼集和解碼集一致！

}

* 1. 與StringBuffer、StringBuilder之間的轉換

String 🡪 StringBuffer、StringBuilder：調用StringBuffer、StringBuilder構造器

StringBuffer、StringBuilder --> String:1.調用String構造器 2. StringBuffer、StringBuilder的toString()

1. JVM中字符串常量池存放位置說明：

jdk1.6(jdk6.0,java6.0):字符串常量池儲存在堆空間

jd1.7:字符串常量池儲存在堆空間

jd1.8:字符串常量池儲存在方法趨(元空間)

1. 常見算法題目的考察：

1. 模擬一個trim方法,去除字符串兩端的空格。

2. 將一個字符串進行反轉。將字符串中指定部分進行反轉。比如“abcdefg”反轉為”abfedcg”

3. 獲取一個字符串在另一個字符串中出現的次數。

比如:獲取“ab”在“abkkcadkabkebfkabkskab”中出現的次數

4.獲取兩個字符串中最大相同子串。比如:str1 = "abcwerthelloyuiodef";str2 = "cvhellobnm"

提示:將短的那個串進行長度依次遞減的子串與較長的串比較。

5.對字符串中字符進行自然順序排序。

提示: 1)字符串變成字符數組。

2)對數組排序,選擇,冒泡,Arrays.sort();

3)將排序後的數組變成字符串。