java.lang.String

1、字符串，用于表示一切字符

在jdk1.8之前，字符串的底层是一个char[]字符数组

会将字符串数据存储到这个字符数组中

String s = "ab你好";

底层实际存储：

char[] value = {'a','b','你','好'};

在JDK1.9之后，将字符串底层的char[]字符数组变为了byte[]字节数组

char:在JDK1.8之前采用Unicode编码，固定的双字节编码。如果存的是字母的话会造成资源浪费，所以在1.9之后都使用byte。

1.9之后如果字符在0-255之间的话，会采用LATIN1编码（西欧编码，固定的单字节编码）如果不再0-255之间的话，会采用UTF16（固定的双字节编码），可以提高内存使用效率。

也就是，在1.9之后会判断字符串中每个字符。

2、字符串的特点

字符串不可变，是一个常量，一旦定义后，内容（字符，长度，大小）就不可改变。

如果字符串的内容变了，呢么一定是创建了新的对象

原因：字符串底层采用byte[]字节数组来存储数据，

而byte[]字节数组使用了private final 修饰符

并且没有对外界提供公开的set/get方法来修饰这个数组

3、字符串对象的创建

字符串常量池：

当时使用字符串字面值创建对象时，首先会检查字符串

常量池中是否有这个对象，如果没有就在池中创建一个

然后栈中引用指向池中这个对象，如果池中有，呢么就不再创建了

栈中引用直接指向这个对象，从而让对象得到复用，可以节省空间

2）使用构造方法创建对象，

会首先会检查字符串常量池中是否有这个对象，如果没有就像在常量池中创建一个，然后将这个对象赋值到堆中，如果池中有这个对象，就直接赋值到堆中，最后栈中的引用指向堆中的对象。

面试题：

例：String s3 = new String("def");

这一行代码总共创建了2个对象，pool池中和heap堆中各一个

String s3 = new String("def");

String s4 = new String("def");

这两行代码总共创建了3个对象，pool池中一个，heap堆中两个。

3、利用字符串拼接创建对象

字符串可以和其他任意类型之间使用+做拼接

拼接完成后的结构还是一个字符串

例子：String s1 = "a" + "b" + "c";

String s2 = "abc";

System.out.println(s1.equals(s2));//其结果为true

System.out.println(s1 == s2);//其结果为true

这个字符拼接在java编译的时候，会进行Java编译器优化：

如果右侧参与拼接的全部都是字符串字面值，也就是在编译期间就已经能确定他的最终结果，呢么Java编译器在编译期间会直接将右侧所以字符串字面值全部拼接在一起，然后再运行。

.Java源码 ==========> .class字节码

"a" + "b" + "c" "abc"

例子：String s1 = "a" + "b";

String s2 = s1 + "c";

System.out.println(s1.equals(s2));//其结果为true

System.out.println(s1 == s2);//其结果为false

因为拼接的s1是一个变量，所以在编译期间无法确定s2的值是什么，只有在运行期间才会把值从变量中取出来，再参与拼接。

所以Java编译器不会对其优化，而是直接在池中创捷一个新的对象。

例子：String s3 = s1 + "c";

System.out.println(s1.equals(s2));//其结果为true

System.out.println(s3 == s2);//其结果为false

因为s6是一个变量，中间操作可能会修改变量的值，所以会进行重写拼接来创建一个新的对象。

例子：String s9 = "123";

String s10 = "1" + 2 + 3;

System.out.println(s3 == s2);//其结果为true

同样因为编译器优化，这个1、2也是字面值。

4、字符串中常用的方法

1）获取字符串的长度： .length() 返回int类型

2)判断是否为空字符串： .isEmpty() 返回boolean类型

3)获取指定下标位置的字符： .charAt(int index) 返回char类型

字符串下标与数组相同，都是从左到右从0开始，最大下标长度-1

可以利用下标遍历字符串中所有字符

4）将字符串转为字符数组 .toCharArray() 返回字符数组

5）判断字符串是否以指定的字串开头

.startsWith(String s) 返回Boolean类型

6）判断字符串是否以指定的字串结尾

.endsWith(String s) 返回boolean类型

7）判断字符串中是否包含指定的字段

.contains(String s) 返回boolean类型

8）将字符串转为全大写 .toUpperCase() 返回一个新的字符串

9）将字符串转为全小写 .toLowerCase() 返回一个新的字符串

10）忽略大小写比较 .equalsIgnoreCase() 返回Boolean

11）获取指定字串的下标位置 .indexOf(String str) 返回下标

以第一个字符下标为主，如果字串不存在则返回-1。

如果有多个相同的字串，以第一次出现的字串为准。

12）从指定下标位置开始向后查找字串出现的下标位置。

.indexOf(String str, int fromIndex)

13）查找字串最后一次出现的下标位置

.lastIndexOf(String str)

14）截取字串，从指定下标位置开始截取到字符串末尾

.substring(int beginIndex)

15）截取字符串

.substring(int beginIndex,int endIndex) [beginIdex,endIndex)

16）去除字符串两端空格 .trim()返回一个新的字符串

17）将其他任意类型转为字符串类型

String.valueOf( ) 返回一个新的字符串

18)concat(String str) 将str拼接到调用对象的末尾

JDK11之后新增的字符串方法

18）判断字符串是否为空白(空格、换行、制表) .isBlank()

19）复制字符串 String.repeat(int num) String复制num次

20）判断字符串有几行

.lines().count() 统计字符串有几行 返回long类型

21）去除空格

21.1）去除头部空格

String.stripLeading()

21.2) 去除尾部空格

String.stripTrailing()

21.2) 去除两端空格

String.strip()

5、正则表达式Regex

作用：用于检测字符串是否符合某一特定规则的格式匹配工具

例：判断字符串是否为6位纯数字

指定一个规则：6位纯数字

在Java中正则表达式也是使用字符串来进行表示的

正则表达式基础语法：

. 代表任意的一个字符

\d 代表任意一个数字字符

\w 代表任意一个单词字符（字母、数字、下划线）

\s 代表任意一个空白字符（空格、换行\n、制表\t）

\D 代表任意一个非数字字符

\W 代表任意一个非单词字符

\S 代表任意一个非空白字符

表示个数

? 代表0个或1个

+ 代表1个或多个

* 代表0个到多个

表示具体有多少个

{6} 必须是6个

{8，14} 最少8个，最多14个

{8，} 最少8个，最多不限

{0，14} 最少不限，最多14个

[0-9] 表示任意一个数字字符 范围在0-9之内

[a-z] 表示任意一个小写英文字母

[A-Z] 表示任意的大写字母

[0-9a-zA-z_] 表示任意的一个数字、字母或下划线

[abc] 表示a或b或c中的任意一个字符

[^abc] 排除a,b,c之外其他字符

& 并且

| 或者

字符串中与API相关的方法

1)matches（String regex）

判断字符串是否能匹配上正则表达式指定的规则

2)split(String regex)

按照正则表达式指定的规则切分字符串

3)replaceAll(String regex, String replacement)

将字符串中能匹配上regex正则表达式的子串

都替换位replacement

String 与 StringBuilder和StringBuffer

与字符串相关的api

都在lang包中

StringBuilder (构建) 线程不安全

StringBuffer (缓存) 线程安全

1、可变字符串

作用：比较适合用于对字符串进行频繁修改，无论修改（字符，大小，长度）多少次都是在原来呢个对象基础上修改的，不会去再创建新的对象，可以节省系统开销。

2、构建对象

.append(所有类型都可以拼接)

在原有对象基础上追加内容时不会创建新对象

1>无参构造器方法

StringBuilder sb = new StringBuilder();

2>指定初始容量

StringBuilder sb = new StringBuilder(int capacity);

3>指定初始字符串

StringBuilder sb = new StringBuilder(String str);

3、主要方法

所有方法调用完成都是返回对象自身，不会去创建新的对象

1> .append(任意类型)

将任意类型转为字符串然后拼接到对象中

2> .reverse()

翻转字符串内容

3> .setCharAt(int index,char c)

修改指定下标位置字符

4> .deleteCharAt(int index)

删除指定下标位置字符

5> .delete(int start,int exd)

删除从[start,end)下标位置的字符

6> .insert(int index,任意类型);

向指定下标位置插入数据

4、String , StringBuilder , StringBuffer 三个之间有什么区别

1>String是不可变对象

StringBuilder , StringBuffer是可变对象

2>StringBuilder线程不安全

StringBuffer 线程安全