Java基础整理

# **1. String**

## 1. 不可变String

String a = “abc”;

String b = “abc”;

System.out.println(a == b);

答案：true

首先会去字符串常量池查找是否存在“abc”，这里a已经创建,所以给b直接赋值

## 2. String s = new String(“abc”);

String s = new String(“abc”)

首先会去到字符串常量池中查找是否存在“abc”，不存在则创建一个；

其次因为有new关键字，那么会在堆中创建一个对象,一共创建两个对象

## 3. 重载“+”,和StringBuilder

String a = “abc” ;

String b = a + “def” + 22；

当虚拟机遇到字符串“+“号时,会创建一个StringBuilder对象,追加字符串,所以当存在大量字符串相加的情况时，直接使用StringBuilder效果要好一点。

## 4. 格式化

当需要对数字、时间、日期等进行格式化的时候，我们可以采用

Java.util.Formatter类进行格式化，String类中的format()方法其内部也是调用的Formatter实现

**Java格式化大致分两类：常规类型的格式化和时间日期类型的格式化。**

### 常规类型格式化固定格式

**%[argument\_index$] [flags] [width] [.percision] conversion**

1. **argument\_index ：可选，是一个十进制整数，用于表明参数在参数列表中的位置。第一个参数由“1$“引用,第二个参数由”2$”引用，以此内推。**
2. **flags： 可选，用来控制输出格式。**
3. **width：可选，是一个正整数，表示输出的最小长度。**
4. **precision：可选，用来限定输出的精度。**
5. **converison：必须，用来表示如何格式化参数的字符。**

#### Conversion

‘b’ : 将参数格式化为boolean输出，‘B’的效果相同。

‘h’ : 将参数格式化为散列输出，原理Integer.toHexString(arg.hashCode()) ‘H’ 效果相同。

‘s’：将参数格式话为字符串输出，如果参数实现了Formattable接口，则调用formatTo方法。‘S’效果相同。

‘c’：将参数格式化为Unicode字符，‘C’效果相同。

‘d’：将参数格式化为十进制整数。

‘o’：将参数格式化为八进制整数。

‘x’：将参数格式化为十六进制整数。

‘e’：将参数格式化为科学技术法的浮点数，‘E’效果相同。

‘f’：将参数格式化为十进制浮点数。

‘g‘：根据具体情况，自动选择用普通表示方式还是科学计数方法。’G’效果相同。

‘a‘：结果格式化为带有效位数和指数的十六进制浮点数。

‘t’ ：时间日期格式化前缀

‘%‘：输出%。%

‘n‘：平台独立的行分隔符

**补充**：

* 对于浮点转换符 'e' 、'E' 和 'f'，精度是小数点分隔符后的位数。如果转换符是 'g' 或 'G'，那么精度是舍入计算后所得数值的所有位数。如果转换是 'a' 或 'A'，则不必指定精度。
* 对于部分转换符，如 'b' 和 'B' ，两者转换效果是相似的，但是 'B' 会把输出中的字母都转换为大写，其它相似特性的转换符应该还有好多，大家可以去找一下O(∩\_∩)O哈哈~

#### Flags

**Flags是可选参数用于控制输出格式**

参数值：

‘-‘ ： 最小宽度左对齐,不可以与“0填充“同时使用

‘+’ ：总是包含一个符号

‘ ‘ ：正直前加空格，负值钱加负号

‘0’ ：结果用0填充

‘,‘ ： 每三位数字用‘,’隔开

‘(‘ ： 若参数是负数，则结果中不添加负号而是用圆括号把数据括起来

#### Width

**用于控制输出的宽度**

#### precision

**用于控制浮点型数值的精度。**

### 时间日期格式化固定格式

%[argument\_index$][flags][width]conversion

#### 常见的日期和时间组合

c ： 包括全部日期和时间信息：如 星期四 十一月 01 10:46:58 CST 2018

F ： “年-月-日”格式 ：如 2018-11-01

D ： 年/月/日’格式 : 如 2018/11/01

r ： “HH:MM:SS PM” 格式 ：如 11:10:07 上午

T ： “HH:MM:SS” 格式：如 11:22:47

#### 单个日期输出

b : 月份简称 ：如 十一月

a ：星期简称, A 效果相同 ：如 星期四

C ：年的前两位（不足前面补0） : 如 20

y ：年的后两位 （不足前面补0）: 如 18

j : 一年的第几天：如 305

m ：两位数字的月份 ： 如 11

d : 两位数字的日（不足两位前面补0） ： 如 01

#### 单个时间输出

H : 2位数字24小时制的小时 如： 15 (不足前面补0)

I : 2位数字的12小时制的小时 如： 03 (不足前面补0)

k : 2位数字24小时制的小时 如： 15 （不补0）

M ：2位数的分钟：如 22

S ：2位数的秒: 如 07

L ：3位数的秒: 如 015

N ：9位数字的毫秒数

p ： 上午或下午

## 5. 正则表达式

**正则表达式定义字符串的模式，可以用来搜索、编辑或处理文本。**

**从jdk1.4起Java提供了java.util.regex.\* 包支持正则表达式**

**元字符：**

\ ： 转移符，将特殊符号转为普通符号

^ ： 位于[]中表示取反，位于[]外面表示一行的开头

$ : 一行的结尾

\* : 0个或多个字符

+ ： 1个或多个字符

? : 0个或1个字符，当此字符跟在(\*、+、?、{n},{n,}, {n,m}) 之后时,匹配 模式是”非贪心的”。”非贪心”模式匹配的尽快能短的字符串，而默认的”贪心的” 模式匹配的是尽可能长的模式。例如 ：字符串 oooo 中 o+? 只匹配o 而o +则匹配oooo。

{n} : n是非负整数，正好匹配n个

{n,} ：n是非负整数，至少匹配n个

{n,m} ：至少匹配n个，至多m个

x|y : 匹配x或y

[ayz] : 匹配字符集中的任意一个字符

[^ayz] : 反向字符集

[^a-z] : 反正字符范围

\b :匹配字符边界,即字与空格间的位置

\B :非边界字符

\cx: 匹配 x 指示的控制字符。例如，\cM 匹配 Control-M 或回车符。x 的值必须在 A-Z 或 a-z 之间。如果不是这样，则假定 c 就是"c"字符本身。

\d : 数字字符匹配 。等效于[0-9]

\D : 非数字字符匹配。等效于[^0-9]

\f : 匹配换页符。等效于\x0c或者\cL

\n : 匹配换行符。等效于 \x0a 和 \cJ。

\r : 匹配一个回车符。等效于\x0d和\cM。

\s : 匹配任何空白字符，包括空格，制表符，换行符.与[\f\n\r\t\v]等效

\S ：匹配任何非空白字符，与[^\f\n\r\t\v]等效

\t ：匹配制表符.与\x09 和\cl等效

\v ：匹配垂直制表符,与 \x0b 和 \cK等效

\w : 匹配任何字符类，包括下划线，与[a-zA-Z0-9\_]等效

\W :匹配非任何字符类，等效于[^a-zA-Z0-9\_]

### NFA引擎匹配原理

#### 引擎分类

**正则引擎大致分两类：**

**DFA (Deterministic finite automaton) ： 确定型有穷自动机**

**NFA(Non-Deterministic finite automaton) : 非确定型有穷自动机**

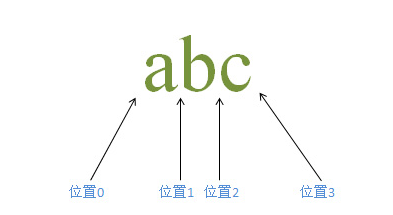
**DFA :**

**DFA从文本入手，从左到右，每个字符不会匹配两次，它的时间复杂度是多项式的，所以通常情况下它的速度更快，但支持的特性很少.**

**NFA :**

**NFA从正则表达式入手，不断读入字符，尝试是否匹配当前正则。不匹配则吐出字符重新匹配，通常速度较慢，最优时间是多项式。最差情况为指数级。但NFA支持更多的特性，因而绝大多数编程场景下(包括java, js)**

#### 字符串组成



**对于abc字符串而言，包含三个字符和四个位置。**

#### 占有字符和零宽度

正则表达式匹配过程中，如果子表达式匹配的是字符串，而非位置，并被保存到最终的匹配结果中，那么就认为这个子表达式是占有字符的；如果子表达式匹配的仅仅是位置，或者匹配的内容。