# 1.正则表达式---Pattern class

在java中正则表达式所对应的类是pattern class，所在的包是Regex。

本质上使用与指定匹配或者筛选规则的一系列表达式。

每个字符在匹配的时候只能跟一个数量词。---用于筛选

#### 1.1正则类型

##### 字符型

###### [abc]

a、b或c。[]括号中的字母可以按照自己的使用要求任意改换。

//字符串由三个字母组成， abc/def/opq

System.out.println(str.matches("[cab][def][opq]"));

###### [a-zA-Z]

a 到z或A到Z，两头的字母包含在内。[]括号中的内容可以按照自己的使用要求任意替换成一段连续的字符。

// 判断字符串是否是由一个字母组成

System.out.println(str.matches("[a-zA-Z]"));

###### [^abc]

任何字符，除了a、b或c。[]括号中的字母可以按照自己的使用要求任意改换。

// 字符串由一个字符组成，字符不是a/d/h/l/p

System.out.println(str.matches("[^adhlp]"));

##### 通配符/预定义字符类

###### .

表示任意字符。

// 匹配字母a开头的由两个字符组成的字符串

// . 表示任意字符

System.out.println(str.startsWith("a") && str.length() == 2);

System.out.println(str.matches("a."));

###### \\[.](file:///\\.)

表示匹配点。

// 匹配 "."

System.out.println(str.matches("\\."));

###### \\\\

匹配反斜杠。

// 匹配 [\](\\)

// "[\\\\](\\) " -> "<\\>" -> [\](\\)

// 路径名

System.out.println(str.matches("[\\\\](\\)"));

###### [\\d](\\\\d)

表示所有数字。

// 任意一个数字

System.out.println(str.matches("[0-9]"));

System.out.println(str.matches("\\d"));

###### \\D

表示非数字。

###### \\s

表示空白符。

###### \\S

表示非空白。

###### \\w

表示单词字符。

###### \\W

非单词字符。

##### 数量词

###### +

表示至少出现一次。>=1。

// 匹配字符a开头的至少由两个字符组成的字符串

// System.out.println(str.startsWith("a") && str.length() >= 2);

// + 表示至少出现一次

// "a+" 至少出现1个a

System.out.println(str.matches("a.+"));

###### ?

表示至多出现一次。<=1。

// 匹配字符a开头的至多由2个字符组成的字符串

// ？表示至多一次

System.out.println(str.matches("a.?"));

###### \*

表示可有可无，有的话不限制次数。>=0。

// 匹配字符串"ab"/"abc"/ abcc

// \* 表示可有可无 ---如果有的话，不限制次数

System.out.println(str.matches("abc\*"));

###### {n}

表示有n个字符组成。==n。

// 匹配由5个字符组成的字符串

System.out.println(str.length() == 5);

// {n} 表示恰好由n个字符组成

System.out.println(str.matches(".{5}"));

###### {n,}

至少有n个组成。>=n。

// 匹配至少由5个字符a组成的字符串

// {n,} 至少由n个组成

// + -> {1,}

System.out.println(str.matches("a{5,}"));

###### {n,m}

数量在n到m之间。

// 匹配由8-12个字符组成的字符串

System.out.println(str.matches(".{8,12}"));

###### 注意

1. 每个字符至多只能跟一个数量词。
2. 在[]内一切数量词失效。

###### 练习：

匹配小数。

输入一个字符串，判断这个字符串是否是一个小数。

package cn.tedu.regex;

import java.util.Scanner;

public class PatternExer1 {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

String str = s.nextLine();

s.close();

// 3.21 10.53

System.out.println(str.trim().matches("0\\.\\d+|[1-9]\\d+\\.\\d+"));

}

}

匹配密码

长度8-12位，至少由字母数字空格至少有两种组成。

public class PatternExer5 {

public static void main(String[] args) {

String password = "";

System.out.println(checkPwd(password));

}

private static boolean checkPwd(String password) {

// 判断密码是否为空

if (password == null) {

return false;

}

// 判断密码的位数是否符合

if (!password.matches(".{8,12}")) {

return false;

}

// 记录出现的字符的种类

int i = 0;

// 判断是否含有字母

if (password.matches(".\*[a-zA-Z].\*")) {

i++;

}

// 判断是否包含数字

if (password.matches(".\*\\d.\*")) {

i++;

}

// 判断是否有符号

if (password.matches(".\*\\W.\*")) {

i++;

} else if (password.matches(".\*\_.\*")) {

i++;

}

return i >= 2;

}

}

##### 捕获组

###### (字符)

捕获组。

// (字符) --- 捕获组

// System.out.println(str.matches(".\*(abc).\*"));

###### [\\n](\\\\n)

表示引用之前对应编号为n的捕获组。

捕获组的编号是从“(”第一次出现的位置开始计算的，java会对捕获组进行自动的编号，编号从1开始。

// 匹配至少含有两个"abc"的字符串

// \\n 表示引用之前对应编号为n的捕获组---捕获组的编号是从1开始的

// System.out.println(str.matches(".\*(abc).\*\\1.\*"));

匹配邮箱（典型练习）

package cn.tedu.regex;

public class PatternExer4 {

public static void main(String[] args) {

String email = "langang@tarena.com.cn";

System.out.println(checkEmail(email));

}

private static boolean checkEmail(String email) {

return email == null ? false : email.matches("\\w+@[0-9A-Za-z]+(\\.com)|\\w+@[0-9A-Za-z]+(\\.com)?(\\.cn)");

}

}

#### 1.2.包含正则的重要方法

#### matches(String regex);

是否匹配参数。

#### replaceAll(String regex,String replacement);

将一类东西替换。后一个参数想引用前一个参数的捕获组，改用$符。$这个符号，仅限于此方法使用。

// String str = "adag35bknl2nkld08aadb";

// 将数字替换为\*

// System.out.println(str.replaceAll("\\d", "\*"));

// 消除所有的数字

// System.out.println(str.replaceAll("\\d", ""));

String str = "Amy Tom Sam David Grace";

// Tom和David调换顺序

System.out.println(str.replaceAll("(.\*)(Tom)(.\*)(David)(.\*)", "$1$4$3$2$5"));

###### 

统计每个字符出现的个数。

从控制台输入一个字符串，统计每个字符出现的次数。

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

String str = s.next();

s.close();

while (str.length() > 0) {

// 记录字符串的原始长度

int len = str.length();

// 获取这个字符串的首字母

char c = str.charAt(0);

// 判断数量词

if (c == '+' || c == '?' || c == '\*') {

str = str.replaceAll("\\"+c + "", "");

}else

// 去除掉这个字符串中的这个字母

str = str.replaceAll(c + "", "");

System.out.println(c + ":" + (len - str.length()));

}

}

###### 叠字换单字

我我我我我我爱爱爱爱学学学学学学学学学学学学编编编程程程程程程 -> 我爱学编程

package cn.tedu.regex;

public class PatternExer3 {

public static void main(String[] args) {

String str = "我我我我我我爱爱爱爱学学学学学学学学学学学学编编编程程程程程程";

System.out.println(str.replaceAll("(.)\\1+", "$1"));

}

}

#### replaceFirst(String regex,string newstr);

将字符串的首个reges替换为newstr。

练习

字符串碎片的平均长度。

String str = "aasdddfdg";

// 记录字符串的长度

int len = str.length();

// 记录碎片的个数

double i = 0;

while (str.length() > 0) {

// 去掉首个叠字

str = str.replaceFirst("(.)\\1\*", "");

// 记录+1

i++;

}

System.out.println(len/i);

#### split(String regex);

以参数regex作为切割符将字符串分开，切完之后，作为切割符的字符就被切除掉了。如果两个切割符相连，这两个切割符之间会切出一个空字符串（“”）;如果切割符在末尾，直接切除。如果在开头，会切出一个空字符串（“”）。

String str = "3aga2dha48nsl9dfg0";

// 以数字作为切割符将字符串分开

// 切完之后，作为切割符的符号就没切除掉了

// 如果两个切割符相连，这两个切割符之间会切出一个 ""

// 如果切割符在末尾，则会直接切除

String[] arr = str.split("\\d");

System.out.println(Arrays.toString(arr));

# 2.包装类（Package class）

## 2.1定义

给每种基本类型提供一个对应的类形式这些类我们称之为包装类。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte | short | int | long | float | double | char | boolean |
| Byte | Short | **Integer** | Long | Float | Double | **Character** | Boolean |

int和char对应的包装类发生变化。

数值类的顶级父类是Number class。

## 2.2特点

1.包装类底层对toString()方法做过重写，直接打印出数值。

2.包装类底层将字符串转化为一个字符数组，首先判断数组的首位是否是+、-，再依次判断每一位是否是一个数字字符。

## 2.3自动封箱

### 1.定义

将一个基本类型的变量转化为了对应的引用类型对象的过程叫封箱，也叫装箱。

将一个基本类型的变量来直接赋值给对应的引用类型对象的过程叫自动封箱，也叫自动装箱。自动封箱实际上是调用了对应类的valueOf方法。

int i = 0x15;

// 将一个基本类型的变量转化为了对应的引用类型的对象---封箱/装箱

Integer in = new Integer(i);

// 说明底层对toString方法做过重写

System.out.println(in);

// int i = 5;

// 将一个基本类型的变量直接赋值给对应的引用类型的对象--自动封箱

// 自动封箱在底层实际上调用了valueOf方法

// Integer in = Integer.valueOf(i);

// Integer in = i;

// System.out.println(in);

由于自动封箱的问题，所以值在-128到127之间的时候会进行判断，如果值在-128到127之间，会直接指向常量池。如果不在这个范围内，会在底层new integer(value)实际上指向了堆内存。

## 2.4自动拆箱

### 1.定义

将一个引用类型的对象来直接赋值给对应的基本类型变量的过程叫自动拆箱。自动拆箱实际上是调用了对应对象的\*\*\*Value方法。

Integer in = new Integer(120);

// 将一个引用类型的对象直接赋值给对应的基本类型的变量---自动拆箱

// 自动拆箱在底层实际上调用了intValue方法

// int i = in.intValue();

int i = in;

System.out.println(i);

自动封箱/拆箱是JDK1.5的特性之一。

Integer in1 = 150; // new Integer(150)

Integer in2 = 150; // new Integer(150)

System.out.println(in1 == in2);

如果是基本类型和引用类型进行运算，引用类型会进行自动拆箱。

Integer i1 =1000;

Int i = 1000;

System.out.println(i1 == i);//结果为true

## 2.5构造函数

### 1.Package(package p);

**Byte(byte value);**

**Short(short value);**

**Integer(int value);**

**Long(long value);**

**Double(double value);**

**Float(double value); Float(float balue);**

**Character (char value);**

**Boolean(boolean value);**

如果字符串不是“true”那么结果就是false。

### Package(String s);

**Byte(String s);**

**Short(String s);**

**Integer(String s);**

**Long(String s);**

**Double(String s);**

**Float(String s);**

**Boolean(String s);**

将一个字符串转化为一个包装类。底层调用的是pares\*\*\*方法。

包装类底层将字符串转化为一个字符数组，首先判断数组的首位是否是+、-，再依次判断每一位是否是一个数字字符。

Character没有提供字符串转化，所以可以用下面的方法转化：

“a” -> ‘a’ “a”.charAt(0) -> ‘a’ “a”.toCharArray()[0]

#### 底层过程

1. 调用来parse\*\*\*方法。
2. 获取这个字符串的第一个字符，判断是否是+/-。
3. 遍历这个字符串的每一位，依次判断每一位是否是一个数字字符。
4. 如果不是数字字符，则报错：NumberFormatException。

## 2.6重要方法

### parse基本数据类型(String s);

**parseByte(String s);**

**parseShort(String s);**

**parseInt(String s);**

**parseLing(String s);**

**ParseFloat(String s);**

**parseDouble(String s)**

**parseBoolean(String s);**

将一个字符串转化为一个包装类。

### hashCode();

Integer类型的哈希码是其实际值。包括baty，short，int。

Long类型的哈希码是其实际值(int)((value)^(value>>>32))。

Double 类型的哈希码也是固定的。

小数的哈希码需要经过计算；

字符的哈希码是其对应的编码；

布尔类型的哈希码是返回一个特殊值---true：1231 false:1237

包装类对象的哈希码都是固定的。

八种基本类型的哈希码都是固定值。

# 3.数学类

## 1.final Math class

Math class是一个最终类，构造函数是私有的，提供了一系列的静态方法来操作数据。是一个完全的工具类。提供了基本的数学运算，运算的是基本类型。

### 字段摘要

#### E

自然底数。比任何其他值都更接近 e（即自然对数的底数）的 double 值。

#### PI

圆周率。比任何其他值都更接近 pi（即圆的周长与直径之比）的 double 值。

### 重要方法

#### abs(int/long/double/float i);

绝对值。

#### cbrt();

立方根。

#### ceil(double d);

向上取整。结果是doulble类型。

#### floor(double d);

向下取整。结果是double类型。

#### max(int/long/float/double m,int/long/float/double m);

返回两个数中的最大数。

#### min(int/long/float/double m,int/long/float/double m);

返回两个数中的最小数。

#### round(double d);

四舍五入。结果是long类型。

#### sqrt(double d);

求平方根。

#### pow(double d , double a);

求d的a次方。结果是double类型。

#### random()

返回一个[0,1)伪随机数。

public static void main(String[] args) {

// System.out.println(Math.E);

// 向上取整

// System.out.println(Math.ceil(3.1));

// 向下取整

// System.out.println(Math.floor(-3.999));

// 四舍五入

// System.out.println(Math.round(3.45));

// 平方根

// System.out.println(Math.sqrt(3));

// 幂 ---pow(double a, double b) -> a^b

// System.out.println(Math.pow(-1, 0.5));

// 返回一个[0,1)的伪随机数

// System.out.println(Math.random());

for (int i = 0; i < 10000; i++)

if (Math.random() \* Math.random() > 0.95) {

System.out.println("亲，你中奖了~~~");

}

}

# 4.BigDecimal class

可以精确表示任意一个小数。

用于精确运算小数的类。不能使用以前的运算符进行计算，而是调用方法，在使用的时候需要将参数以字符串形式传入。

System.out.println(3 - 2.89);

BigDecimal bd1 = new BigDecimal("3");

BigDecimal bd2 = new BigDecimal("2.89");

System.out.println(bd1.subtract(bd2));

### 构造方法

#### Bigdecimal(int/long/double/String o);

### 重要方法

#### abs();

绝对值。

#### add(BigDecimal dibisor);

加法。

#### divide();

除法。

#### multiply();

乘法。

# 5.日期类

## Date class

这个类中大部分方法已经过时。

deprecation 表示此方法已过时。

### 构造函数

#### Date();

获取当前系统时间。

Thu Nov 16 10:28:31 CST 2017

星期 月份 日期 时间 时区 年份

China standard timezone 中国标准时区。

从打印对象就能看出，Date class底层对toString做来重写。

#### Date(2000,2,1);

在1900年1月来累加参数，指定时间。

Thu Mar 01 00:00:00 CST 3900

## \*DateFormat class

是一个抽象类。

### SimpleDateFormat class

继承了DateFormat class。

#### 日期标识符

|  |  |  |  |
| --- | --- | --- | --- |
| **字母** | **日期或时间元素** | **表示** | **示例** |
| G | Era 标志符 | [Text](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#text) | AD |
| y | 年 | [Year](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#year) | 1996; 96 |
| M | 年中的月份 | [Month](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#month) | July; Jul; 07 |
| w | 年中的周数 | [Number](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 27 |
| W | 月份中的周数 | [Number](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 2 |
| D | 年中的天数 | [Number](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 189 |
| d | 月份中的天数 | [Number](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 10 |
| F | 月份中的星期 | [Number](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 2 |
| E | 星期中的天数 | [Text](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#text) | Tuesday; Tue |
| a | Am/pm 标记 | [Text](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#text) | PM |
| H | 一天中的小时数（0-23） | [Number](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 0 |
| k | 一天中的小时数（1-24） | [Number](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 24 |
| K | am/pm 中的小时数（0-11） | [Number](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 0 |
| h | am/pm 中的小时数（1-12） | [Number](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 12 |
| m | 小时中的分钟数 | [Number](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 30 |
| s | 分钟中的秒数 | [Number](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 55 |
| S | 毫秒数 | [Number](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 978 |
| z | 时区 | [General time zone](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#timezone) | Pacific Standard Time; PST; GMT-08:00 |
| Z | 时区 | [RFC 822 time zone](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#rfc822timezone) | -0800 |

#### 构造函数

##### SimpleDateFormat();

用默认的模式和默认语言环境的日期格式符号构造 SimpleDateFormat。

##### SimpleDateFormat(String pattern);

#### 重要方法

##### Format([Date](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/../../java/util/Date.html) date, [StringBuffer](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/../../java/lang/StringBuffer.html) toAppendTo, [FieldPosition](mk:@MSITStore:D:\学习资料\JDK_API_1_6_zh_CN.CHM::/java/text/../../java/text/FieldPosition.html) pos);

将给定的 Date 格式化为日期/时间字符串，并将结果添加到给定的 StringBuffer。

##### parse(String text,ParsePosition pos);

解析字符串的文本，生成 Date。

// 将字符串转化为日期对象

Date date = new

SimpleDateFormat("yyyy-MM-dd HH:mm:ss").parse("2000-02-01 12:42:15");

// XX年XX月XX日

// 将日期对象转化为字符串

String str = new SimpleDateFormat("yyyy年MM月dd日").format(date);

System.out.println(str);

## Calendar class（日历类）

也是一个抽象类。有一个子类：GregorianCalendar。

Calendar c = Calendar.getInstance();

Calendar c = Calendar.getInstance();

System.out.println(c);