# 1 Math类

Math类封装了数学运算方面的功能。

Math类提供了两个静态常量PI和E，分别是圆周率和自然常熟e，即：

|  |
| --- |
| **public static final double *PI*** = 3.14159265358979323846;  **public static final double *E*** = 2.7182818284590452354;  （实际计算时，精度只到小数点后15位） |

Math是工具类，其中所有的方法都是静态的，常用的有（我省写了public修饰符）：

（1）static int abs(int a)：绝对值

（2）static double ceil(double a)：向上取整

（3）static double floor(double a)：向下取整

（4）static int max(int a,int b)：最大值，min是最小值

（5）static double pow(double a, double b)：返回a的b次幂

（6）static double random()：返回[0.0,1.0)之间的随机数，当前时间的毫秒值作为种子。

（7）static int round(double a)：四舍五入

（8）static double sqrt(double a)：求正平方根。

另外还有log()、sin()、cos()和tan()等。sin()等传递的是弧度，可以利用Math.PI。

# 2 Random类

Random类用于产生随机数。首先要知道什么是随机数种子：随机数是根据种子得到一系列随机数的，如果初始给的种子相同，那么每次运行程序得到的随机数顺序都相同。是伪随机。该类在java.utils包中，需要导入。

构造方法：

（1）public Random()：默认使用当前时间的毫秒值作为随机数种子。这样每次运行程序时，时间不同，达到了随机的效果。

（2）public Random(long seed)：可指定随机数种子。这样每次运行程序，随机数顺序都是一样的。

得到随机数用到的实例方法：

（1）int nextInt()：返回int范围内的随机整数；

（2）int nextInt(int n):返回[0,n)范围内的随机整数。

当然还有nextBoolean()、nextFloat()等方法。

例子：

|  |
| --- |
| **package** com.zhang.test;  **import** java.util.Random;  **public class** Demo {  **public static void** main(String[] args) {  Random r = **new** Random(1000); *// 固定种子  // 由于使用了固定的种子，下面的结果每次运行都是一样的* **for**(**int** i = 0; i < 10; i++) {  System.***out***.println(r.nextInt(100));  }  } } |

有时，直接使用Math.random()方法更方便。下面就是利用Math.random()封装了一个getRandomRange()方法，可以得到指定范围的随机整数，实现如下：

|  |
| --- |
| **package** com.zhang.test;  **import** java.util.Random;  **public class** Demo {  **public static void** main(String[] args) {  **for**(**int** i = 0; i < 10; i++) {  *// 取十次值* System.***out***.println(*getRandomRange*(50, 56));  }  }   *// 得到的数的范围是[from, to]* **public static int** getRandomRange(**int** from, **int** to) {  **return** (**int**)(Math.*random*() \* (to - from + 1)) + from;  } } |

# 3 System类

提供和系统相关的方法。该类中方法也全部是静态的，此类不能实例化。

（1）static void gc()：手动运行垃圾回收器，回收空引用对象并调用对象的finalize()方法。

（2）static void exit(int status)：终止当前运行的 JVM。参数用作状态码，根据惯例，非 0 的状态码表示异常终止，一般就使用0来结束程序。

（3）static long currentTimeMillis()：返回当前时间的毫秒值。

（4）static void arraycopy(Object src,int srcPos,Object dest,int destPos,int length) 从指定源数组中复制一个数组，复制从指定的位置开始，到目标数组的指定位置，并且指定长度。

（5）static Properties getProperties()：此方法返回系统的多种属性，并封装在Properties对象中，以后学了Properties就能获取信息了。

# 4 Date类

Date类用于处理日期和时间，精确到毫秒。Date类的构造方法有：

public Date()和public Date(long date)；第一个构造是根据当前时间的毫秒值创建Date对象，而第二个构造是根据指定的毫秒值创建Date对象。

常用的成员方法有：

public long getTime()：获取Date对象对应的毫秒值；

public void setTime(long date)：设置Date对象的毫秒值；

public boolean after(Date when)：测试该日期是否在指定的日期之后；

public boolean before(Date when)：测试该日期是否在指定的日期之前。

## 4.1 Date的格式化和解析

在实际中，经常用到下面两种情况：

（1）将Date对象格式化成友好的字符串输出，便于用户查看；

（2）将包含日期信息的字符串解析成为Java的Date对象保存起来，以便编程时使用。

上述就是格式化和解析。

可使用DateFormat类来格式化和解析。但是该类是抽象类，因此我们使用它的具体子类SimpleDateFormat，这是JDK提供的。

SimpleDateFormat的构造方法有：

public SimpleDateFormat()和public SimpleDateFormat(String pattern)。构造1是用默认的匹配模式来格式化和解析，构造2是用指定的模式来格式化和解析。

查看API，可找到对应的模式：

年 y 月 M 日 d 时 H 分 m 秒 s（区分大小写）

DateFormat类提供了下面两个方法：

（1）public String format(Date date)：将Date对象格式化成字符串

（2）public Date parse(String source)：将字符串解析成Date对象。

例子：（注意该Date类在java.util包下，不到导成java.sql包下的Date类）

|  |
| --- |
| **package** com.zhang.test;  **import** java.text.ParseException; **import** java.text.SimpleDateFormat; **import** java.util.Date;  **public class** Demo {  **public static void** main(String[] args) **throws** ParseException {  Date date = **new** Date();  *// 创建SimpleDateFormat对象，指定模式为：2016年12月25日 14:16:28* SimpleDateFormat sdf = **new** SimpleDateFormat(**"yyyy年MM月dd日 HH:mm:ss"**);  String dateStr = sdf.format(date); *// 将当前时间按照匹配模式格式化* System.***out***.println(dateStr);   *// 用parse方法将字符串变为Date对象，还是使用上面的匹配模式。  // 一个SimpleDateFormat对象只能有一个匹配模式  // 转换失败抛出ParseException异常* Date d = sdf.parse(**"2016年12月25日 9:23:20"**);  System.***out***.println(d.getTime());  } } |

为了方便使用，可以写一个工具类，提供格式化和解析的方法，参考代码：

|  |
| --- |
| **package** com.zhang.test;  **import** java.text.ParseException; **import** java.text.SimpleDateFormat; **import** java.util.Date;  */\*\*  \* 时间格式化和解析工具类  \*/* **public class** DateFormatTool {  **private** DateFormatTool() {}   */\*\*  \* 格式化日期对象  \** ***@param date*** *日期对象  \** ***@param pattern*** *匹配模式  \** ***@return*** *格式化字符串  \*/* **public static** String format(Date date, String pattern) {  **return new** SimpleDateFormat(pattern).format(date);  }   */\*\*  \* 解析日期字符串  \** ***@param source*** *日期字符串  \** ***@param pattern*** *匹配模式  \** ***@return*** *日期对象  \** ***@throws*** *ParseException 解析异常  \*/* **public static** Date parse(String source, String pattern) **throws** ParseException {  **return new** SimpleDateFormat(pattern).parse(source);  } } |

# 5 Calendar类

Calendar类也封装了系统的当前时间，用于获得年月日时分秒等信息，因为Date中获得这些信息的方法已经过时了（Deprecated），不推荐再使用了。

Calendar没有构造方法，如果想创建一个Calendar对象，则使用Calendar类的静态方法getInstance()，此方法返回一个当前Calendar对象。即：

Calendar canlendar = Calendar.getInstance();

常用的成员方法：

（1）public final void set(int year,int month,int date)：设置日历对象的年月日。

（2）public int get(int field)：返回指定日历字段的值。参数里面是int类型，传递的是Calendar里面定义的常量更为直观，比如：calendar.get(Calendar.YEAR);

（3）public void add(int field, int amount)：给指定的日历字段加上指定的时间。比如加一个月：calendar.add(Calendar.MONTH, 1)。

例子：

|  |
| --- |
| **package** com.zhang.test;  **import** java.util.Calendar;  **public class** Demo {  **public static void** main(String[] args) {  Calendar calendar = Calendar.*getInstance*();  *// 获取年月日* **int** year = calendar.get(Calendar.***YEAR***);  **int** month = calendar.get(Calendar.***MONTH***) + 1; *// get()得到的月份是从0开始的，因此要加1* **int** day = calendar.get(Calendar.***DAY\_OF\_MONTH***);  *// 输出* System.***out***.println(year + **"年"** + month + **"月"** + day + **"日"**);   *// 加减Calendar对象中的年月日等，可以是负值，表示往前推算时间，这样不用自己去推算了  // 向前56天* calendar.add(Calendar.***DAY\_OF\_MONTH***, -56);  *// 再输出calendar的月、日信息* System.***out***.println(calendar.get(Calendar.***MONTH***) + 1 + **"月"** + calendar.get(Calendar.***DAY\_OF\_MONTH***) + **"日"**);  } } |

# 6 正则表达式类

# 6.1 什么是正则表达式

正则表达式（Regular Expression）就是符合一定规则的字符串，它可以匹配特定的字符串，方便进行字符串的处理。

正则表达式的规则：

（1）字符：一个普通字符就代表匹配一个字符。也支持转义字符，比如\\ 表示一个反斜线字符，\n 是换行符，\r是回车符。

（2）中括号：代表匹配中括号中的任意一个字符。比如[0-9]匹配一个数字；[abc]表示匹配一个abc字母中的一个；[a-zA-Z]匹配一个英文字母。中括号里的^表示“取反”，即不匹配里面的字母，比如[^abc]匹配一个除了abc的字符。

（3）预定义字符：

1. “.”匹配一个任何字符，如果使用“\.”就表示匹配“.”字符本身。

2. “\d”匹配一个数字，相当于[0-9]。

3. “\w”匹配一个单词字符，相当于[a-zA-Z0-9\_]。

4. “\D”匹配除了数字，相当于[^0-9]。

5. “\W”匹配除了单词字符，相当于[^a-zA-Z0-9\_]。

（4）边界匹配器

1. “^”表示开头；“$”表示结尾。

2. “\b”表示单词边界，就是不是单词字符的地方。举例：hello world?haha;xixi

（5）Greedy数量词

1. X? ：X出现0次或1次

2. X\* ：X出现0次或多次

3. X+ ：X出现一次或多次

4. X{n} ：X出现恰好 n 次

5. X{n,} ：X出现至少 n 次

6. X{n,m} ：X出现n-m次

## 6.2 应用举例

### 6.2.1 匹配功能

匹配就是判断一个字符串是否能和指定的字符串匹配上。此功能使用String类的实例方法matches(String regex)，具体的原型是：

public boolean matches(String regex); 传递的参数就是正则表达式，返回布尔值。

也可以用Pattern类的静态方法matches()，即Pattern.matches(String regex, String str)。

例子：

|  |
| --- |
| **package** com.zhang.test;  **import** java.util.regex.Pattern;  **public class** Demo {  **public static void** main(String[] args) {  *// 写个正则表达式匹配任意的6位数字* String regex = **"[0-9]{6}"**;  *// 这是待测试的数据* String eg = **"123456"**;  *// 调用方法判断* **boolean** isMatch = eg.matches(regex);  System.***out***.println(isMatch); *// 结果是true   // 使用静态方法* **boolean** isMatch2 = Pattern.*matches*(regex, **"123"**);  System.***out***.println(isMatch2); *//这个就是false* } } |

### 6.2.2 字符串分割(split)

String类中有实例方法split()方法，一个原型是：

public String[] split(String regex)，就是根据正则，把字符串分割成数组返回。比如原来的字符串是foo:hoo:hei，按正则表达式“:”分割，那么分割返回的数组是：[“foo”, “hoo”, “hei”]。

例子：

|  |
| --- |
| **package** com.zhang.test;  **public class** Demo {  **public static void** main(String[] args) {  String str = **"aa.bb.cc"**;  *// 要求用分割“.”。由于.在正则中是特殊字符，因此需要转义使用“\.”，但是\在Java中又是特殊字符，还要加个\转义。* String[] strArr = str.split(**"\\."**);  *// 遍历数组，输出的就是aa bb cc了* **for**(**int** i = 0; i < strArr.**length**; i++) {  System.***out***.println(strArr[i]);  }  } } |

### 6.2.3 替换功能

替换就是把字符串中和指定正则匹配的子串替换为指定的字符串，再返回。使用String类中的replaceAll()方法，方法的原型为：

public String replaceAll(String regex, String replacement);

还可以使用Pattern中的静态方法：

Pattern.compile(regex).matcher(str).replaceAll(repl)。

例子：

|  |
| --- |
| **package** com.zhang.test;  **import** java.util.regex.Pattern;  **public class** Demo {  **public static void** main(String[] args) {  String src = **"hello2013hello77810show"**;  *// 目标：将src中每个数字换成让符号“\*”* String regex = **"\\d"**; *// 用\\d匹配一个数字，然后调用方法替换* String result1 = src.replaceAll(regex, **"\*"**);  System.***out***.println(result1);   *// 如果将连续的一串数字替换成一个\*，那么应该这样写regex：* regex = **"\\d+"**;  String result2 = Pattern.*compile*(regex).matcher(src).replaceAll(**"\*"**);  System.***out***.println(result2);  } } |

### 6.2.4 获取功能

获取就是在字符串中获取匹配指定的正则的子串。这里需要用到Pattern类，Pattern是正则表达式的编译表示形式。然后用Pattern对象和字符串对象获得Matcher对象，通过Matcher对象找到所有匹配的子串。

例子：

|  |
| --- |
| **package** com.zhang.test;  **import** java.util.regex.Matcher; **import** java.util.regex.Pattern;  **public class** Demo {  **public static void** main(String[] args) {  String src = **"Hello everyone, we are family. -- da jia hao, wo men shi yi jia ren."**;  *// 目标：将上面的字符串中，所有的3个字母的“单词”选择出来   //1. 先将正则编译成Pattern对象* Pattern pattern = Pattern.*compile*(**"\\b\\w{3}\\b"**); *// \b表示单词边界  //2. 由pattern和字符串获得Matcher对象* Matcher matcher = pattern.matcher(src);  *//3. 循环输出所有的匹配项，需要用find和group方法* **while**(matcher.find()) {  System.***out***.println(matcher.group());  }  *// 更深入的功能自行研究* } } |