# Java.lang

标准包：

Java.lang 包含java最基础的类，可以直接在程序中使用这些类不需要引入包

Java.io 包含支持输入/输出操作的类

Java.awt 支持java图形化用户界面的类（GUI）

Java.swing 提供支持GUI组件的类，推荐使用

Java.util 包含一系列标准操作以管理数据集、访问日期以及分析字符串的类

Java.util.zip 包含支持生成jar文件的类

Java.sql 包含支持使用标准sql的数据库访问功能的类

## 包装类

  Java语言是一个面向对象的语言，但是Java中的基本数据类型却是不面向对象的，这在实际使用时存在很多的不便，为了解决这个不足，在设计类时为每个基本数据类型设计了一个对应的类进行代表，这样八个和基本数据类型对应的类统称为包装类(Wrapper Class)，有些地方也翻译为外覆类或数据类型类。

         包装类均位于java.lang包，包装类和基本数据类型的对应关系如下表所示：

|  |  |
| --- | --- |
| 原始数据类型 | 包装类 |
| byte(字节) | Byte |
| char(字符) | Character |
| int(整型) | Integer |
| long(长整型) | Long |
| float(浮点型) | Float |
| double(双精度) | Double |
| boolean(布尔) | Boolean |
| short(短整型) | Short |

  在这八个类名中，除了Integer和Character类以后，其它六个类的类名和基本数据类型一直，只是类名的第一个字母大写即可。

         对于包装类说，这些类的用途主要包含两种：

                   a、作为和基本数据类型对应的类类型存在，方便涉及到对象的操作。

                   b、包含每种基本数据类型的相关属性如最大值、最小值等，以及相关的操作方法。

         由于八个包装类的使用比较类似，下面以最常用的Integer类为例子介绍包装类的实际使用。

### 1、实现int和Integer类之间的转换

在实际转换时，使用Integer类的构造方法和Integer类内部的intValue方法实现这些类型之间的相互转换，实现的代码如下：

                   int n = 10;

                   Integer in = new Integer(100);

                   //将int类型转换为Integer类型

                   Integer in1 = new Integer(n);

                   //将Integer类型的对象转换为int类型

                   int m = in.intValue();

### 2、Integer类内部的常用方法

                  在Integer类内部包含了一些和int操作有关的方法，下面介绍一些比较常用的方法：

                   a、parseInt方法

                            public static int parseInt(String s)

该方法的作用是将数字字符串转换为int数值。在以后的界面编程中，将字符串转换为对应的int数字是一种比较常见的操作。使用示例如下：

                                     String s = “123”;

                                     int n = Integer.parseInt(s);

则int变量n的值是123，该方法实际上实现了字符串和int之间的转换，如果字符串都包含的不是都是数字字符，则程序执行将出现异常。(说明：异常的概念将在下一章进行讲述)

另外一个parseInt方法：

         public static int parseInt(String s, int radix)

则实现将字符串按照参数radix指定的进制转换为int，使用示例如下：

         //将字符串”120”按照十进制转换为int，则结果为120

         int n = Integer.parseInt(“120”,10);

         //将字符串”12”按照十六进制转换为int，则结果为18

         int n = Integer.parseInt(“12”,16);

         //将字符串”ff”按照十六进制转换为int，则结果为255

         int n = Integer.parseInt(“ff”,16);

这样可以实现更灵活的转换。

                   b、toString方法

                            public static String toString(int i)

                            该方法的作用是将int类型转换为对应的String类型。

                            使用示例代码如下：

                                     int m = 1000;

                                     String s = Integer.toString(m);

                            则字符串s的值是”1000”。

                            另外一个toString方法则实现将int值转换为特定进制的字符串：

                                     public static int parseInt(String s, int radix)

                            使用示例代码如下：

                                     int m = 20;

                                     String s = Integer.toString(m);

                            则字符串s的值是”14”。

         其实，JDK自从1.5(5.0)版本以后，就引入了自动拆装箱的语法，也就是在进行基本数据类型和对应的包装类转换时，系统将自动进行，这将大大方便程序员的代码书写。使用示例代码如下：

                   //int类型会自动转换为Integer类型

                   int m = 12;

                   Integer in = m;

                   //Integer类型会自动转换为int类型

                   int n = in;

         所以在实际使用时的类型转换将变得很简单，系统将自动实现对应的转换。

### 装箱与拆箱

包装类型 实例名 = new 包装类型（基类型数据）

实例名.基类型value()

示例：

**int** num = 5;

**int** num2;

Integer integer = **new** Integer(num);

num2 = integer.intValue();

System.***out***.println(num2);

### 数字型常用方法

包装类型 实例名 = 包装类型.valueOf(String)

包装类型.parse基本类型(String)

实例名.toString()

### 字符型Character常用方法

|  |  |
| --- | --- |
| 方法 | 说明 |
| isDigit() | 确定字符是否为0至9之间的数字 |
| isLetter() | 确定字符是否为字母 |
| isLowerCase() | 确定字符是否为小写形式 |
| isUpperCase() | 确定字符是否为大写形式 |
| isSpace() | 确定字符是否为空格或换行符 |

## Java String 类

字符串广泛应用 在Java 编程中，在 Java 中字符串属于对象，Java 提供了 String 类来创建和操作字符串。

### 创建字符串

创建字符串最简单的方式如下:

String greeting = "字符串";

在代码中遇到字符串常量时，这里的值是 "**菜鸟教程**""，编译器会使用该值创建一个 String 对象。

和其它对象一样，可以使用关键字和构造方法来创建 String 对象。

String 类有 11 种构造方法，这些方法提供不同的参数来初始化字符串，比如提供一个字符数组参数:

public class Test {

public static void main(String[] args) {

char[] helloArray = { 'r', 'u', 'n', 'o', 'o', 'b' };

String helloString = new String(helloArray);

System.*out*.println(helloString);

}

}

**注意:**String 类是不可改变的，所以你一旦创建了 String 对象，那它的值就无法改变了如果需要对字符串做很多修改，那么应该选择使用 [StringBuffer & StringBuilder 类](http://www.runoob.com/java/java-stringbuffer.html)。

### 字符串长度

用于获取有关对象的信息的方法称为访问器方法。

String 类的一个访问器方法是 length() 方法，它返回字符串对象包含的字符数。

String site = "www.baidu.com";

**int** len = site.length();

System.***out***.println( "百度网址长度 : " + len );

### 连接字符串

String 类提供了连接两个字符串的方法：

string1.concat(string2);

返回 string2 连接 string1 的新字符串。也可以对字符串常量使用 concat() 方法，如：

"我的名字是 ".concat("Runoob");

更常用的是使用'+'操作符来连接字符串，如：

"Hello," + " runoob" + "!"

### 创建格式化字符串

我们知道输出格式化数字可以使用 printf() 和 format() 方法。

String 类使用静态方法 format() 返回一个String 对象而不是 PrintStream 对象。

String 类的静态方法 format() 能用来创建可复用的格式化字符串，而不仅仅是用于一次打印输出。

如下所示：

System.out.printf("浮点型变量的值为 " + "%f, 整型变量的值为 " + " %d, 字符串变量的值为 " + "is %s", floatVar, intVar, stringVar);

你也可以这样写

String fs;

fs = String.format("浮点型变量的值为 " + "%f, 整型变量的值为 " + " %d, 字符串变量的值为 " + " %s", floatVar, intVar, stringVar);

### String 方法

下面是 String 类支持的方法，更多详细，参看 [Java String API](https://www.runoob.com/manual/jdk1.6/java/lang/String.html) 文档:

|  |  |
| --- | --- |
| **SN(序号)** | **方法描述** |
| 1 | [char charAt(int index)](http://www.runoob.com/java/java-string-charat.html) 返回指定索引处的 char 值。 |
| 2 | [int compareTo(Object o)](http://www.runoob.com/java/java-string-compareto.html) 把这个字符串和另一个对象比较。 |
| 3 | [int compareTo(String anotherString)](http://www.runoob.com/java/java-string-compareto.html) 按字典顺序比较两个字符串。 |
| 4 | [int compareToIgnoreCase(String str)](http://www.runoob.com/java/java-string-comparetoignorecase.html) 按字典顺序比较两个字符串，不考虑大小写。 |
| 5 | [String concat(String str)](http://www.runoob.com/java/java-string-concat.html) 将指定字符串连接到此字符串的结尾。 |
| 6 | [boolean contentEquals(StringBuffer sb)](http://www.runoob.com/java/java-string-contentequals.html) 当且仅当字符串与指定的StringBuffer有相同顺序的字符时候返回真。 |
| 7 | [static String copyValueOf(char[] data)](http://www.runoob.com/java/java-string-copyvalueof.html) 返回指定数组中表示该字符序列的 String。 |
| 8 | [static String copyValueOf(char[] data, int offset, int count)](http://www.runoob.com/java/java-string-copyvalueof.html) 返回指定数组中表示该字符序列的 String。 |
| 9 | [boolean endsWith(String suffix)](http://www.runoob.com/java/java-string-endswith.html) 测试此字符串是否以指定的后缀结束。 |
| 10 | [boolean equals(Object anObject)](http://www.runoob.com/java/java-string-equals.html) 将此字符串与指定的对象比较。 |
| 11 | [boolean equalsIgnoreCase(String anotherString)](http://www.runoob.com/java/java-string-equalsignorecase.html) 将此 String 与另一个 String 比较，不考虑大小写。 |
| 12 | [byte[] getBytes()](http://www.runoob.com/java/java-string-getbytes.html)  使用平台的默认字符集将此 String 编码为 byte 序列，并将结果存储到一个新的 byte 数组中。 |
| 13 | [byte[] getBytes(String charsetName)](http://www.runoob.com/java/java-string-getbytes.html) 使用指定的字符集将此 String 编码为 byte 序列，并将结果存储到一个新的 byte 数组中。 |
| 14 | [void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)](http://www.runoob.com/java/java-string-getchars.html) 将字符从此字符串复制到目标字符数组。 |
| 15 | [int hashCode()](http://www.runoob.com/java/java-string-hashcode.html) 返回此字符串的哈希码。 |
| 16 | [int indexOf(int ch)](http://www.runoob.com/java/java-string-indexof.html) 返回指定字符在此字符串中第一次出现处的索引。 |
| 17 | [int indexOf(int ch, int fromIndex)](http://www.runoob.com/java/java-string-indexof.html) 返回在此字符串中第一次出现指定字符处的索引，从指定的索引开始搜索。 |
| 18 | [int indexOf(String str)](http://www.runoob.com/java/java-string-indexof.html)  返回指定子字符串在此字符串中第一次出现处的索引。 |
| 19 | [int indexOf(String str, int fromIndex)](http://www.runoob.com/java/java-string-indexof.html) 返回指定子字符串在此字符串中第一次出现处的索引，从指定的索引开始。 |
| 20 | [String intern()](http://www.runoob.com/java/java-string-intern.html)  返回字符串对象的规范化表示形式。 |
| 21 | [int lastIndexOf(int ch)](http://www.runoob.com/java/java-string-lastindexof.html)  返回指定字符在此字符串中最后一次出现处的索引。 |
| 22 | [int lastIndexOf(int ch, int fromIndex)](http://www.runoob.com/java/java-string-lastindexof.html) 返回指定字符在此字符串中最后一次出现处的索引，从指定的索引处开始进行反向搜索。 |
| 23 | [int lastIndexOf(String str)](http://www.runoob.com/java/java-string-lastindexof.html) 返回指定子字符串在此字符串中最右边出现处的索引。 |
| 24 | [int lastIndexOf(String str, int fromIndex)](http://www.runoob.com/java/java-string-lastindexof.html)  返回指定子字符串在此字符串中最后一次出现处的索引，从指定的索引开始反向搜索。 |
| 25 | [int length()](http://www.runoob.com/java/java-string-length.html) 返回此字符串的长度。 |
| 26 | [boolean matches(String regex)](http://www.runoob.com/java/java-string-matches.html) 告知此字符串是否匹配给定的正则表达式。 |
| 27 | [boolean regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len)](http://www.runoob.com/java/java-string-regionmatches.html) 测试两个字符串区域是否相等。 |
| 28 | [boolean regionMatches(int toffset, String other, int ooffset, int len)](http://www.runoob.com/java/java-string-regionmatches.html) 测试两个字符串区域是否相等。 |
| 29 | [String replace(char oldChar, char newChar)](http://www.runoob.com/java/java-string-replace.html) 返回一个新的字符串，它是通过用 newChar 替换此字符串中出现的所有 oldChar 得到的。 |
| 30 | [String replaceAll(String regex, String replacement)](http://www.runoob.com/java/java-string-replaceall.html) 使用给定的 replacement 替换此字符串所有匹配给定的正则表达式的子字符串。 |
| 31 | [String replaceFirst(String regex, String replacement)](http://www.runoob.com/java/java-string-replacefirst.html)  使用给定的 replacement 替换此字符串匹配给定的正则表达式的第一个子字符串。 |
| 32 | [String[] split(String regex)](http://www.runoob.com/java/java-string-split.html) 根据给定正则表达式的匹配拆分此字符串。 |
| 33 | [String[] split(String regex, int limit)](http://www.runoob.com/java/java-string-split.html) 根据匹配给定的正则表达式来拆分此字符串。 |
| 34 | [boolean startsWith(String prefix)](http://www.runoob.com/java/java-string-startswith.html) 测试此字符串是否以指定的前缀开始。 |
| 35 | [boolean startsWith(String prefix, int toffset)](http://www.runoob.com/java/java-string-startswith.html) 测试此字符串从指定索引开始的子字符串是否以指定前缀开始。 |
| 36 | [CharSequence subSequence(int beginIndex, int endIndex)](http://www.runoob.com/java/java-string-subsequence.html)  返回一个新的字符序列，它是此序列的一个子序列。 |
| 37 | [String substring(int beginIndex)](http://www.runoob.com/java/java-string-substring.html) 返回一个新的字符串，它是此字符串的一个子字符串。 |
| 38 | [String substring(int beginIndex, int endIndex)](http://www.runoob.com/java/java-string-substring.html) 返回一个新字符串，它是此字符串的一个子字符串。 |
| 39 | [char[] toCharArray()](http://www.runoob.com/java/java-string-tochararray.html) 将此字符串转换为一个新的字符数组。 |
| 40 | [String toLowerCase()](http://www.runoob.com/java/java-string-tolowercase.html) 使用默认语言环境的规则将此 String 中的所有字符都转换为小写。 |
| 41 | [String toLowerCase(Locale locale)](http://www.runoob.com/java/java-string-tolowercase.html)  使用给定 Locale 的规则将此 String 中的所有字符都转换为小写。 |
| 42 | [String toString()](http://www.runoob.com/java/java-string-tostring.html)  返回此对象本身（它已经是一个字符串！）。 |
| 43 | [String toUpperCase()](http://www.runoob.com/java/java-string-touppercase.html) 使用默认语言环境的规则将此 String 中的所有字符都转换为大写。 |
| 44 | [String toUpperCase(Locale locale)](http://www.runoob.com/java/java-string-touppercase.html) 使用给定 Locale 的规则将此 String 中的所有字符都转换为大写。 |
| 45 | [String trim()](http://www.runoob.com/java/java-string-trim.html) 返回字符串的副本，忽略前导空白和尾部空白。 |
| 46 | [static String valueOf(primitive data type x)](http://www.runoob.com/java/java-string-valueof.html) 返回给定data type类型x参数的字符串表示形式。 |

## Java StringBuffer 和 StringBuilder 类

当对字符串进行修改的时候，需要使用 StringBuffer 和 StringBuilder 类。

和 String 类不同的是，StringBuffer 和 StringBuilder 类的对象能够被多次的修改，并且不产生新的未使用对象。

StringBuilder 类在 Java 5 中被提出，它和 StringBuffer 之间的最大不同在于 StringBuilder 的方法不是线程安全的（不能同步访问）。

由于 StringBuilder 相较于 StringBuffer 有速度优势，所以多数情况下建议使用 StringBuilder 类。然而在应用程序要求线程安全的情况下，则必须使用 StringBuffer 类。

StringBuffer sBuffer = **new** StringBuffer("百度：");

sBuffer.append("www");

sBuffer.append(".baidu");

sBuffer.append(".com");

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

### StringBuffer 方法

以下是 StringBuffer 类支持的主要方法：

|  |  |
| --- | --- |
| **序号** | **方法描述** |
| 1 | public StringBuffer append(String s) 将指定的字符串追加到此字符序列。 |
| 2 | public StringBuffer reverse()  将此字符序列用其反转形式取代。 |
| 3 | public delete(int start, int end) 移除此序列的子字符串中的字符。 |
| 4 | public insert(int offset, int i) 将 int 参数的字符串表示形式插入此序列中。 |
| 5 | replace(int start, int end, String str) 使用给定 String 中的字符替换此序列的子字符串中的字符。 |

下面的列表里的方法和 String 类的方法类似：

|  |  |
| --- | --- |
| **序号** | **方法描述** |
| 1 | int capacity() 返回当前容量。 |
| 2 | char charAt(int index) 返回此序列中指定索引处的 char 值。 |
| 3 | void ensureCapacity(int minimumCapacity) 确保容量至少等于指定的最小值。 |
| 4 | void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin) 将字符从此序列复制到目标字符数组 dst。 |
| 5 | int indexOf(String str) 返回第一次出现的指定子字符串在该字符串中的索引。 |
| 6 | int indexOf(String str, int fromIndex) 从指定的索引处开始，返回第一次出现的指定子字符串在该字符串中的索引。 |
| 7 | int lastIndexOf(String str) 返回最右边出现的指定子字符串在此字符串中的索引。 |
| 8 | int lastIndexOf(String str, int fromIndex) 返回 String 对象中子字符串最后出现的位置。 |
| 9 | int length()  返回长度（字符数）。 |
| 10 | void setCharAt(int index, char ch) 将给定索引处的字符设置为 ch。 |
| 11 | void setLength(int newLength) 设置字符序列的长度。 |
| 12 | CharSequence subSequence(int start, int end) 返回一个新的字符序列，该字符序列是此序列的子序列。 |
| 13 | String substring(int start) 返回一个新的 String，它包含此字符序列当前所包含的字符子序列。 |
| 14 | String substring(int start, int end) 返回一个新的 String，它包含此序列当前所包含的字符子序列。 |
| 15 | String toString() 返回此序列中数据的字符串表示形式。 |

## Java Number & Math 类

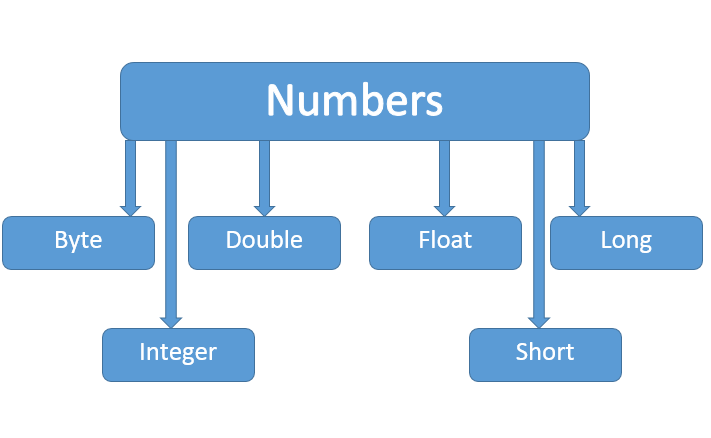
一般地，当需要使用数字的时候，我们通常使用内置数据类型，如：**byte、int、long、double** 等。

### 实例

int a = 5000; float b = 13.65f; byte c = 0x4a;

然而，在实际开发过程中，我们经常会遇到需要使用对象，而不是内置数据类型的情形。为了解决这个问题，Java 语言为每一个内置数据类型提供了对应的包装类。

所有的包装类**（Integer、Long、Byte、Double、Float、Short）**都是抽象类 Number 的子类。



这种由编译器特别支持的包装称为装箱，所以当内置数据类型被当作对象使用的时候，编译器会把内置类型装箱为包装类。相似的，编译器也可以把一个对象拆箱为内置类型。Number 类属于 java.lang 包。

下面是一个使用 Integer 对象的实例：

Test.java 文件代码：

public class Test{ public static void main(String args[]){ Integer x = 5; x = x + 10; System.out.println(x); } }

以上实例编译运行结果如下：

15

当 x 被赋为整型值时，由于x是一个对象，所以编译器要对x进行装箱。然后，为了使x能进行加运算，所以要对x进行拆箱。

## Java Math 类

Java 的 Math 包含了用于执行基本数学运算的属性和方法，如初等指数、对数、平方根和三角函数。

Math 的方法都被定义为 static 形式，通过 Math 类可以在主函数中直接调用。

Test.java 文件代码：

public class Test { public static void main (String []args) { System.out.println("90 度的正弦值：" + Math.sin(Math.PI/2)); System.out.println("0度的余弦值：" + Math.cos(0)); System.out.println("60度的正切值：" + Math.tan(Math.PI/3)); System.out.println("1的反正切值： " + Math.atan(1)); System.out.println("π/2的角度值：" + Math.toDegrees(Math.PI/2)); System.out.println(Math.PI); } }

以上实例编译运行结果如下：

90 度的正弦值：1.0

0度的余弦值：1.0

60度的正切值：1.7320508075688767

1的反正切值： 0.7853981633974483

π/2的角度值：90.0

3.141592653589793

### Number & Math 类方法

下面的表中列出的是 Number & Math 类常用的一些方法：

|  |  |
| --- | --- |
| **序号** | **方法与描述** |
| 1 | [xxxValue()](http://www.runoob.com/java/number-xxxvalue.html) 将 Number 对象转换为xxx数据类型的值并返回。 |
| 2 | [compareTo()](http://www.runoob.com/java/number-compareto.html) 将number对象与参数比较。 |
| 3 | [equals()](http://www.runoob.com/java/number-equals.html) 判断number对象是否与参数相等。 |
| 4 | [valueOf()](http://www.runoob.com/java/number-valueof.html) 返回一个 Number 对象指定的内置数据类型 |
| 5 | [toString()](http://www.runoob.com/java/number-tostring.html) 以字符串形式返回值。 |
| 6 | [parseInt()](http://www.runoob.com/java/number-parseInt.html) 将字符串解析为int类型。 |
| 7 | [abs()](http://www.runoob.com/java/number-abs.html) 返回参数的绝对值。 |
| 8 | [ceil()](http://www.runoob.com/java/number-ceil.html) 返回大于等于( >= )给定参数的的最小整数。 |
| 9 | [floor()](http://www.runoob.com/java/number-floor.html) 返回小于等于（<=）给定参数的最大整数 。 |
| 10 | [rint()](http://www.runoob.com/java/number-rint.html) 返回与参数最接近的整数。返回类型为double。 |
| 11 | [round()](http://www.runoob.com/java/number-round.html) 它表示**四舍五入**，算法为 **Math.floor(x+0.5)**，即将原来的数字加上 0.5 后再向下取整，所以，Math.round(11.5) 的结果为12，Math.round(-11.5) 的结果为-11。 |
| 12 | [min()](http://www.runoob.com/java/number-min.html) 返回两个参数中的最小值。 |
| 13 | [max()](http://www.runoob.com/java/number-max.html) 返回两个参数中的最大值。 |
| 14 | [exp()](http://www.runoob.com/java/number-exp.html) 返回自然数底数e的参数次方。 |
| 15 | [log()](http://www.runoob.com/java/number-log.html) 返回参数的自然数底数的对数值。 |
| 16 | [pow()](http://www.runoob.com/java/number-pow.html) 返回第一个参数的第二个参数次方。 |
| 17 | [sqrt()](http://www.runoob.com/java/number-sqrt.html) 求参数的算术平方根。 |
| 18 | [sin()](http://www.runoob.com/java/number-sin.html) 求指定double类型参数的正弦值。 |
| 19 | [cos()](http://www.runoob.com/java/number-cos.html) 求指定double类型参数的余弦值。 |
| 20 | [tan()](http://www.runoob.com/java/number-tan.html) 求指定double类型参数的正切值。 |
| 21 | [asin()](http://www.runoob.com/java/number-asin.html) 求指定double类型参数的反正弦值。 |
| 22 | [acos()](http://www.runoob.com/java/number-acos.html) 求指定double类型参数的反余弦值。 |
| 23 | [atan()](http://www.runoob.com/java/number-atan.html) 求指定double类型参数的反正切值。 |
| 24 | [atan2()](http://www.runoob.com/java/number-atan2.html) 将笛卡尔坐标转换为极坐标，并返回极坐标的角度值。 |
| 25 | [toDegrees()](http://www.runoob.com/java/number-todegrees.html) 将参数转化为角度。 |
| 26 | [toRadians()](http://www.runoob.com/java/number-toradians.html) 将角度转换为弧度。 |
| 27 | [random()](http://www.runoob.com/java/number-random.html) 返回一个随机数。 |

### Math 的 floor,round 和 ceil 方法实例比较

|  |  |  |  |
| --- | --- | --- | --- |
| **参数** | **Math.floor** | **Math.round** | **Math.ceil** |
| 1.4 | 1 | 1 | 2 |
| 1.5 | 1 | 2 | 2 |
| 1.6 | 1 | 2 | 2 |
| -1.4 | -2 | -1 | -1 |
| -1.5 | -2 | -1 | -1 |
| -1.6 | -2 | -2 | -1 |

### floor,round 和 ceil 实例：

public class Test {

public static void main(String[] args) {

double[] nums = { 1.4, 1.5, 1.6, -1.4, -1.5, -1.6 };

for (double num : nums) {

*test*(num);

}

}

private static void test(double num) {

System.*out*.println("Math.floor(" + num + ")=" + Math.*floor*(num));

System.*out*.println("Math.round(" + num + ")=" + Math.*round*(num));

System.*out*.println("Math.ceil(" + num + ")=" + Math.*ceil*(num));

}

}

## Class类

Class称作类的描述器，类的类

Class包装正在运行类的运行状态及信息，可用来动态加载类，得到类的实例