

1 数据类型

1.1 整型

1.2 浮点类型

1.3 char类型

1.4 Unicode编码与char类型

1.5 boolean类型

2 变量与常量

2.1 变量

2.2 常量

1 数据类型

Java是一种强类型语言，必须为每个变量声明一种类型。在Java中，一共有8种基本类型，其中有4种整型、2种浮点类型、1种字符类型和1种 `boolean` 类型。

1.1 整型

类型	存储长度	取值范围
<code>int</code>	4字节	-2 147 483 648~2 147 483 647
<code>short</code>	2字节	-32768~32767
<code>long</code>	8字节	-9 223 372 036 854 775 808 ~9 223 372 036 854 775 807
<code>byte</code>	1字节	-128~127

长整型数值有一个后缀 `L` 或 `l`（如 `4000000000L`），十六进制数值有一个前缀 `0x` 或 `0X`（如 `0xCAFE`），八进制有一个前缀 `0`。八进制表示法比较容易混淆，所以最好不要使用八进制常数。

从Java 7开始，可以用前缀 `0b` 或 `0B` 表示二进制数。此外，从Java 7开始，还可以为数字字面量加下划线（如 `1_000_000`），使数字更易读，Java编译器会去除这些下划线，不会对程序产生影响。

1.2 浮点类型

类型	存储长度	取值范围
<code>float</code>	4字节	大约±3.402 823 47E+38F（有效位数为6~7位）
<code>double</code>	8字节	大约±1.797 693 134 862 315 70E+308（有效位数为15位）

`float` 类型的数值有一个后缀 `F` 或 `f`（如 `3.14F`），没有后缀 `F` 的浮点数值默认为 `double` 类型。也可以在 `double` 类型数值后面添加后缀 `D` 或 `d`。

可以使用十六进制表示浮点数值，尾数采用十六进制，指数采用十进制，指数的基数是2，尾数和指数之间用 `p` 分隔。例如， $0.125=2^{-3}$ ，在十六进制表示法中表示为 `0x1.0p-3`，其中 `0x` 是十六进制前缀，`1.0` 是尾数，`-3` 是指数。

1.3 char类型

`char` 类型的字面值要用单引号括起来，如 `'A'` 是一个字符常量，而 `"A"` 是一个字符串。

`char` 类型的值可以表示为十六进制值，其范围从 `\u0000` 到 `\uFFFF`。`\u` 是一个转义序列，后面跟四位十六进制数，表示一个字符。除此之外，还有一些用于表示特殊字符的转义序列，如下表所示。

转义序列	名称	Unicode值
<code>\b</code>	退格	<code>\u0008</code>
<code>\t</code>	制表	<code>\u0009</code>
<code>\n</code>	换行	<code>\u000a</code>
<code>\r</code>	回车	<code>\u000d</code>
<code>\"</code>	双引号	<code>\u0022</code>
<code>\'</code>	单引号	<code>\u0027</code>
<code>\\</code>	反斜杠	<code>\u005c</code>

所有这些转义序列都可以出现在加引号的字符字面量或字符串中，例如 `'\u2122'` 和 `"Hello\n"`。

转义序列 `\u` 还可以出现在加引号的字符常量或字符串之外，而其他所有转义序列不可以。例如：

```
public static void main(String\u005B\u005D args)
```

这种写法符合语法规则，`\u005B` 和 `\u005D` 分别是 `'['` 和 `']'` 的编码。

一定要当心注释中的 `\u`。例如：

```
// \u000A is a newline
错误，因为 \u000A 是换行符，之后的所有内容在下一行，不属于注释内容

// look inside c:\users
错误，因为 \u 后面没有跟着四位十六进制数。正确的写法为 c:\\users，\\是转义序列，表示反斜杠
```

1.4 Unicode编码与char类型

码点（code point）是指与一个编码表中的某个字符对应的代码值。在Unicode标准中，码点采用16进制书写，并加上前缀“U+”，例如“U+0041”是“A”的码点。Unicode的码点可分为17个**代码平面**（code plane）。第一个代码平面称为**基本多语言平面**，包括码点从U+0000到U+FFFF的“经典”Unicode代码；其余16个平面的码点从U+10000到U+10FFFF，包括**辅助字符**。

UTF-16编码采用不同长度的编码表示所有Unicode码点。在基本多语言平面中，每个字符用16位表示，通常称为**代码单元**（code unit）；而辅助字符编码为一对连续的代码单元，即32位。辅助字符的两个代码单元取值落入基本多语言平面中未用的2048个值范围内，通常称为**替代区域**。这样，对于任意给定的一个代码单元，可以迅速确定它是一个字符的编码，还是一个辅助字符的第一部分或第二部分。

在Java中，`char` 类型描述了UTF-16编码中的一个代码单元。对于基本多语言平面中的字符，只需要1个 `char` 值；而辅助字符则需要2个 `char` 值。对于一个单独的 `char` 值，当它代表辅助字符中的一个代码单元时，对它的解释将没有意义。因此，最好不要在程序中使用 `char` 类型，除非需要处理UTF-16代码单元。

1.5 boolean类型

`boolean` 类型有两个值：`false` 和 `true`，用来判定逻辑条件。**整型值和布尔值之间不能进行相互转换。**

2 变量与常量

2.1 变量

声明变量时，先指定变量的类型，然后是变量名。例如：

```
double salary;  
int vacationDays;  
long earthPopulation;  
boolean done;
```

变量名必须是一个以字母开头并由字母或数字构成的序列。变量名中所有的字符都有意义，并且大小写敏感。

声明一个变量之后，必须用赋值语句对变量进行显示初始化，**千万不要使用未初始化的变量的值。**

变量的声明尽可能地靠近变量第一次使用的地方，这是一种良好的代码风格。

从Java 10开始，对于局部变量，如果可以从变量的初始值推断出它的类型，就不再需要声明类型，只需要使用关键字 `var` 而无需指定类型。例如：

```
var vacationDays = 12;  
var greeting = "Hello";
```

2.2 常量

在Java中，使用关键字 `final` 指示常量。例如：

```
final double CM_PER_INCH = 2.54;
```

常量只能被赋值一次，一旦被赋值后，就不能再更改了。习惯上，常量名使用全大写。

可以在类中使用关键字 `static final` 设置类常量，这样就可以在类的多个方法中使用这个常量。例如：

```
public class Constant  
{  
    public static final double CM_PER_INCH = 2.54;  
}
```