

2. 数据类型和运算符

本章概述

- 标识符
- 常量和变量
- 数据类型
 - 整型 浮点型 字符型 布尔型
- 运算符
 - 算术运算符 赋值运算符 关系运算符
 - 逻辑运算符 位运算符 条件运算符
- 基本数据类型的类型转换
 - 自动类型转换
 - 强制类型转换



本章技能点列表

技能点名称	难易程度	认知程度	重要程度
标识符	易	记忆	**
数据类型分类	易	记忆	***
常量和变量	易	记忆	***
整数数据类型	中	记忆	***
浮点数据类型	中	记忆	***
字符数据类型	中	记忆	**
布尔数据类型	易	记忆	***
final常量	中	应用	***
Scanner类	中	应用	***



本章技能点列表

技能点名称	难易程度	认知程度	重要程度
算术运算符	中	应用	***
赋值运算符	中	应用	***
关系运算符	中	应用	***
逻辑运算符	难	应用	***
位运算符	难	了解	*
条件运算符	中	应用	*
运算符的优先级别和结合性	中	理解	***
基本数据类型类型转换	中	应用	***



标识符Identifier

- 作用

- 常量、变量、方法、类和包等的名称

- 命名规则

- 必须以字母、_下划线、美元符\$开头。
- 其它部分可以是字母、下划线“_”、美元符“\$”和数字的任意组合
- 大小写敏感，长度无限制。
- 不可以是Java的关键字。

- 注意

- Java不采用通常语言使用的ASCII字符集
- Java采用unicode这样的标准的国际字符集。因此，这里的字母的含义：
英文、汉字等等。(不建议大家使用汉字来定义标识符！)

正确的：

name, Name, user_name, \$name, _name, publicName;

错误的：

9username, user name, public



关键字/保留字

- Java关键字是Java语言保留供内部使用的,如class用于定义类。关键字也可以称为保留字,它们的意思是一样的。
- 注意: 不要刻意去背! 后面会慢慢介绍每个关键字的用法

abstract↵	assert↵	<u>boolean</u> ↵	break↵	byte↵	case↵
catch↵	char(character)↵	class↵	const↵	continue↵	default ↵
do↵	double↵	else↵	extends↵	final↵	finally↵
float ↵	for↵	<u>goto</u> ↵	if↵	implements↵	import↵
<u>instanceof</u> ↵	<u>int</u> ↵	interface↵	long↵	native↵	new↵
null↵	package↵	private↵	protected↵	public↵	return↵
short↵	static↵	<u>strictfp</u> ↵	super↵	switch↵	synchronized↵
this↵	throw↵	throws↵	transient↵	try↵	void↵
volatile↵	while↵	↵	↵	↵	↵



字符集简介

- ASCII
 - 英文字符集 1个字节
- ISO8859-1
 - 西欧字符集 1个字节
- BIG5
 - 台湾的大五码，表示繁体汉字 2个字节
- GB2312
 - 大陆使用最早、最广的简体中文字符集 2个字节
- GBK
 - GB2312的扩展，可以表示繁体中文 2个字节
- GB18030
 - 最新GBK的扩展，可以表示汉字、维吾尔文、藏文等中华民族字符 2个字节
- Unicode
 - 国际通用字符集 2个字节



Java基本数据类型

- Java是一种强类型语言
 - 常量是有数据类型的
 - 变量都必须声明其数据类型





常量和变量

- 常量变量定义
 - 在程序中存在大量的数据来代表程序的状态，其中有些数据在程序的运行过程中值会发生改变，有些数据在程序运行过程中值不能发生改变，这些数据在程序中分别被叫做变量和常量。
- 变量举例：
 - 在2D游戏程序中，需要代表人物的位置，则需要2个变量，一个是x坐标，一个是y坐标，在程序运行过程中，这两个变量的值会发生改变
- 常量举例
 - 代表常数，便于程序的修改（例如：圆周率的值）
 - 增强程序的可读性（例如：常量UP、DOWN、LEFT和RIGHT分别代表上下左右，其数值分别是1、2、3和4）
- 在实际的程序中，可以根据数据在程序运行中是否发生改变，来选择应该是使用变量代表还是常量代表。



常量和变量

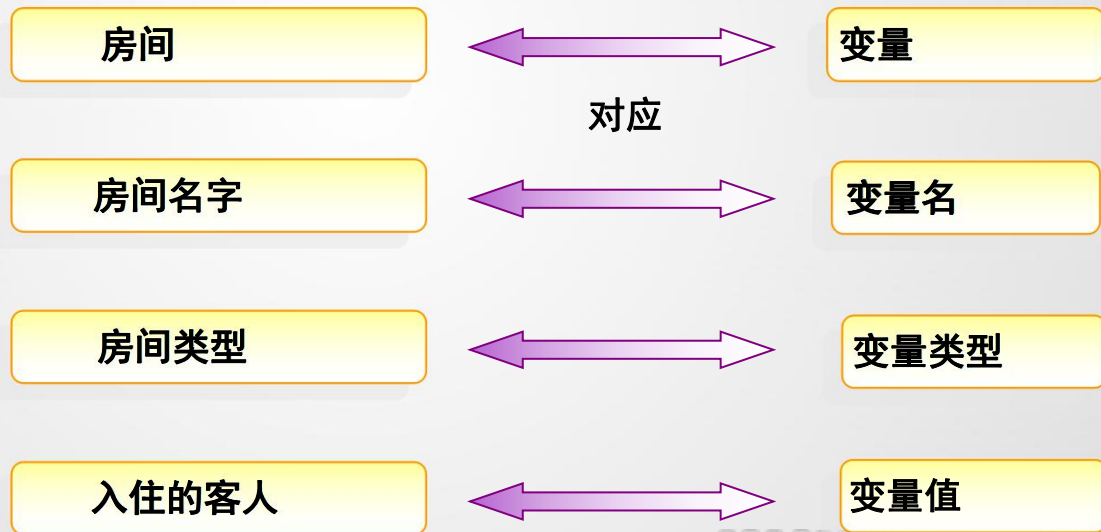
- Java是一种强类型语言，每个变量都必须声明其类型。
- Java变量是程序中最基本的存储单元，其要素包括变量名，变量类型和作用域。
- 变量在使用前必须对其声明, 只有在变量声明以后，才能为其分配相应长度的存储单元，声明格式为：
 - `type varName [=value] [{,varName[=value]}] ;`
- 注意事项：
 - 每个变量都有类型，类型可以是基本类型，也可以是引用类型。
 - 变量名必须是合法的标识符。



常量和变量

- 电脑使用内存来记忆计算时所使用的数据。人类采用旅馆来存储外出住宿的人们。
- 内存中变量好比旅馆的房间，内存中常量好比住旅馆房间的人

- `int age=20;`
- `age=21;`
- `String name="小明";`





变量声明

- 变量声明举例：
 - `double salary ;`
 - `boolean done;`
 - `long earthPopulation ;`
 - `int age ;`
- 可以在一行中声明多个变量：
 - `int i ,j; // both are integers`
- 不提倡这种风格，逐一声明每一个变量可以提高程序可读性。
- 可以将变量的声明和初始化放在同一行中，例如：
 - `int age = 18;`
 - `float e = 2.718281828f;`



整型数据类型

- 整型常量
 - 十进制整数，如：99, -500, 0
 - 八进制整数，要求以 0 开头，如：015
 - 十六进制数，要求 0x 或 0X 开头，如：0x15
- 整型变量

类 型	占用存储间	表数范围
byte	1 字节	-128 ~ 127
short	2 字节	$-2^{15} \sim 2^{15}-1$ (-32768~32767)
int (integer)	4 字节	$-2^{31} \sim 2^{31}-1$ (-2147483648~2147483647) 约 21 亿
long	8 字节	$-2^{63} \sim 2^{63}-1$

- 整型常量默认为int型，声明long型常量可以后加 ‘l’或 ‘L’(建议使用大写，小写容易误认为数字1)，如：
 - `long a = 55555555;` //不出错，在Int表示的范围内(21亿内)。
 - `long b = 555555555555;` //不加l出错，已经超过int表示的范围。



浮点型数据类型

• 浮点类型常量

- 十进制数形式，例如：

- 3.14 314.0 0.314

• 科学记数法形式，如

- 314e2 314E2 314E-2
- `double f = 314e2; //314*10^2-->31400.0`
- `double f2 = 314e-2; //314*10^(-2)-->3.14`

• 浮点型变量

- float类型：单精度类型，尾数可以精确到7位有效数字，在很多情况下，float类型的精度很难满足需求。
- double类型：双精度类型 精度是float类型的两倍，绝大部分应用程序都采用double类型。

• 注意

- 浮点常量默认为double。要变为float，需在后面增加F/f. 如： 3.14F
- 浮点数存在舍入误差，很多数字不能精确表示。如果需要进行不产生舍入误差的精确数字计算，需要使用BigDecimal类。

类 型 ↴	占用存储空间 ↴	表数范围 ↴
float ↴	4 字节 ↴	-3.403E38~3.403E38 ↴
double ↴	8 字节 ↴	-1.798E308~1.798E308 ↴



字符数据类型（2个字节）

- 单引号用来表示字符常量。例如 ‘A’是一个字符，它与“A”是不同的，“A”表示含有一个字符的字符串。
- char 类型用来表示在Unicode编码表中的字符。
- Unicode编码被设计用来处理各种语言的所有文字，它占2个字节，可允许有65536个字符；ASCII码占1个字节，可允许有128个字符（最高位是0），是Unicode编码表中前128个字符。
- Java 语言中还允许使用转义字符 ‘\’ 来将其后的字符转变为其它的含义，
char c2 = '\n'; //代表换行符
- char类型在内存中存储的是该字符的Unicode编码值，所以char类型可以当做int类型来处理

转义符↵	含义↵	Unicode 值↵
\b↵	退格（backspace）↵	\u0008↵
\n↵	换行↵	\u000a↵
\r↵	回车↵	\u000d↵
\t↵	制表符（tab）↵	\u0009↵
\"↵	双引号↵	\u0022↵
'\''↵	单引号↵	\u0027↵
\\↵	反斜杠↵	\u005c↵



布尔数据类型（一位，不是一个字节）

- boolean类型有两个值，true和false
- boolean 类型用来判断逻辑条件，一般用于程序流程控制
- 实践：
- 请不要这样写：if (is == true && done == false) ，只有新手才那么写。
对于任何程序员 if (whether && !done) 都不难理解吧。所以去掉所有的==false 和 ==true。



final 常量

- 使用final修饰的变量，只能被初始化一次，变成了常量。
- final常量是有名称的

```
public class Constants {  
    public static void main(String[] args) {  
        final double PI = 3.14;  
        // PI = 3.15; //error  
        double r = 4;  
        double area = PI * r * r;  
        double circle = 2 * PI * r;  
        System.out.println("area = " + area);  
        System.out.println("circle = " + circle);  
    }  
}
```



命名规则 (规范)

- 所有变量、方法、类名：见名知意
- 变量、方法名：
 - 首字母小写和驼峰原则
 - `run()`, `runRun()`, `age` `ageNew` `monthSalary`
- 常量：
 - 大写字母和下划线：`MAX_VALUE`
- 类名：
 - 首字母大写和驼峰原则：`Man`, `GoodMan`



运算符

- Java 语言支持如下运算符：
 - 算术运算符: +, -, *, /, %, ++, --
 - 赋值运算符 =
 - 扩展赋值运算符: +=, -=, *=, /=
 - 关系运算符: >, <, >=, <=, ==, !=
 - 逻辑运算符: &&, ||, !
 - 位运算符: &, |, ^, ~, >>, <<, >>> (了解!!!)
 - 条件运算符 ? :
- 相关概念辨析
 - + 运算符 操作符 Operator
 - 5+6 表达式 expression
 - 5 6 操作数 Operand
 - int m =5+6; 语句 Sentence



算术运算符

• 注意:

- 1: / 除 $6/4=1$ $6/4.0=1.5$
- 2: %取模(求余) $6\%4=2$
- 3: + $6+'a'=103$ $6+"a"=6^a$
- 4: ++ $a=2;b=++a+9;$ $a=2;b=a+++9;$
- +: 字符串相加, 只要有一个操作数是字符串, 自动变为字符串相连

算术运算符

运算符	运算	范例	结果
+	正号	+3	3
-	负号	b=4;-b;	-4
+	加	5+5	10
-	减	6-4	2
*	乘	3*4	12
/	除	5/5	1
%	取模	5%5	0
++	自增(前)	a=2;b=++a;	a=3;b=3
++	自增(后)	a=2;b=a++;	a=3;b=2
--	自减(前)	a=2;b=--a	a=1;b=1
--	自减(后)	a=2;b=a--	a=1;b=2
+	字符串相加	"He"+"llo"	"Hello"



算术运算符

- 二元运算符类型提升：
 - 整数运算：
 - 如果两个操作数有一个为Long, 则结果也为long
 - 没有long时, 结果为int。即使操作数全为short,byte, 结果也是int.
 - 浮点运算：
 - 如果两个操作数有一个为double, 则结果为double.
 - 只有两个操作数都是float, 则结果才为float.
- 一元运算符(++ , --):
 - `int a = 3;`
 - `int b = a++;` //执行完后,b=3。先给b赋值, 再自增。
 - `int c = ++a;` //执行完后,c=5。先自增,再给b赋值



赋值运算符

- 基本赋值运算符
- 扩展赋值运算符
- 算术运算符和赋值运算符结合

赋值运算符

运算符	运算	范例	结果
=	赋值	a=3;b=2;	a=3;b=2;
+=	加等于	a=3;b=2;a+=b;	a=5;b=2;
-=	减等于	a=3;b=2;a-=b;	a=1;b=2;
=	乘等于	a=3;b=2;a=b;	a=6;b=2;
/=	除等于	a=3;b=2;a/=b	a=1;b=2;
%=	模等于	a=3;b=2;a%=b	a=1;b=2;



关系运算符

- 关系运算符用来进行比较运算
- 关系运算的结果是布尔值：true/false
- 注意
 - 区分==和=
 - 不等于不是！=不是<>

运算符	含义	示例
==	等于	a==b
!=	不等于	a!=b
>	大于	a>b
<	小于	a=	大于或等于	a>=b
<=	小于或等于	a<=b



逻辑运算符

- 逻辑运算符与布尔操作数一起使用，组成逻辑表达式

- 逻辑表达式的结果是布尔值

- &和&&的区别

- “&”:无论任何情况，“&”两边的表达式都会参与计算。
- “&&”:当“&&”的左边为false，则将不会计算其右边的表达式。即左false则false

- “|”和“||”的区别与“&”和“&&”的区别类似。

运算符	含义	示例
&	逻辑与	A & B
	逻辑或	A B
^	逻辑异或	A ^ B
!	逻辑反	! A
	短路或	A B
&&	短路与	A && B



位运算符（了解）

- 位运算符是对操作数以二进制比特位为单位进行操作和运算，操作数和结果都是整型数。
- 如果操作的对象是char、byte、short，位移动作发生时其值会自动晋升为int，运算结果也为int

运算符	含义	示例
~	按位非（NOT）/取反	$b = \sim a$
&	按位与（AND）	$c = a \& b$
	按位或（OR）	$c = a b$
^	按位异或(相同为0相异为1)	$c = a ^ b$
>>	右移；左边空位补最高位即符号位	$b = a >> 2$
>>>	无符号右移，左边空位补0	$b = a >>> 2$
<<	左移；右边空位以补0	$b = a << 1$



条件运算符

- 语法格式
 - $x ? y : z$
 - 唯一的三目运算符
- 执行过程
 - 其中 x 为 `boolean` 类型表达式，先计算 x 的值，若为 `true`，则整个三目运算的结果为表达式 y 的值，否则整个运算结果为表达式 z 的值。
- 经常用来代替简单的 `if-else` 判断！

```
int a=3;
int b=5;
String str="";
if(a<b){
    str = "a<b";
}else{
    str = "a>b";
}
```

```
int a=3;
int b=5;
String str=
(a<b)?"a<b":"a>b";
```



运算符的优先级

- 不需要去刻意的记优先级关系
- 赋值<三目<逻辑<关系<算术<单目
- 理解运算符的结合性

优先级	运算符	类	结合性
1	()	括号运算符	由左至右
1	[]	方括号运算符	由左至右
2	!、+ (正号)、- (负号)	一元运算符	由右至左
2	~	位逻辑运算符	由右至左
2	++、--	递增与递减运算符	由右至左
3	*, /、%	算术运算符	由左至右
4	+, -	算术运算符	由左至右
5	<<、>>	位左移、右移运算符	由左至右
6	>、>=、<、<=	关系运算符	由左至右
7	==、!=	关系运算符	由左至右
8	& (位运算符 AND)	位逻辑运算符	由左至右
9	^ (位运算符 XOR)	位逻辑运算符	由左至右
10	(位运算符 OR)	位逻辑运算符	由左至右
11	&&	逻辑运算符	由左至右
12		逻辑运算符	由左至右
13	?:	条件运算符	由右至左
14	=	赋值运算符	由右至左



基本数据类型之间的转换

- 在赋值运算或算术运算时，要求数据类型相同，否则要进行类型转换
- 转换方式：
 - 自动转换
 - 强制转换
- 除boolean类型外，所有的基本数据类型因为各自的精度不同，赋值时都要考虑这个问题
- 除boolean类型外，所有的基本数据类型包括：整型，浮点型，字符型。



基本数据类型之间的转换

- 算术运算时的转换

- 整型,浮点型,字符型数据可以混合运算。
- 运算中相邻两个操作数先转化为同一类型,转换从低级到高级,然后进行运算;运算的结果再和下一个操作数据进行运算,规则相同
- $3+'A'+45L$
- $5+3.6+'A'+3.14f$

操作数1类型	操作数2类型	转换后类型
byte、short、char	int	int
byte、short、char、int	long	long
byte、short、char、int、long	float	float
byte、short、char、int、long、float	double	double



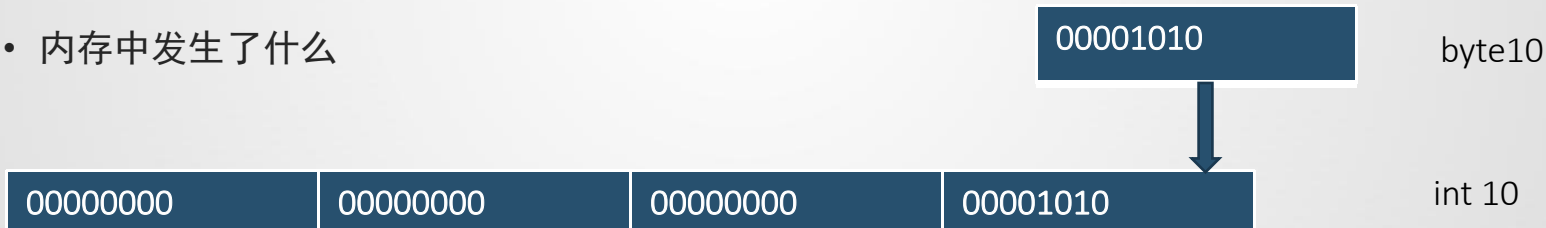
基本数据类型之间的转换

- 赋值运算时的转换
 - 自动类型转换（左>右）

```
byte b=10;  
System.out.println("b="+b);  
int n = b;  
System.out.println("n="+n);
```

```
C:\myjava>javac Test.java  
  
C:\myjava>java Test  
b=10  
n=10
```

- 内存中发生了什么



- 类型转换的方向
 - 低----->高
 - byte → short, char → int → long → float → double



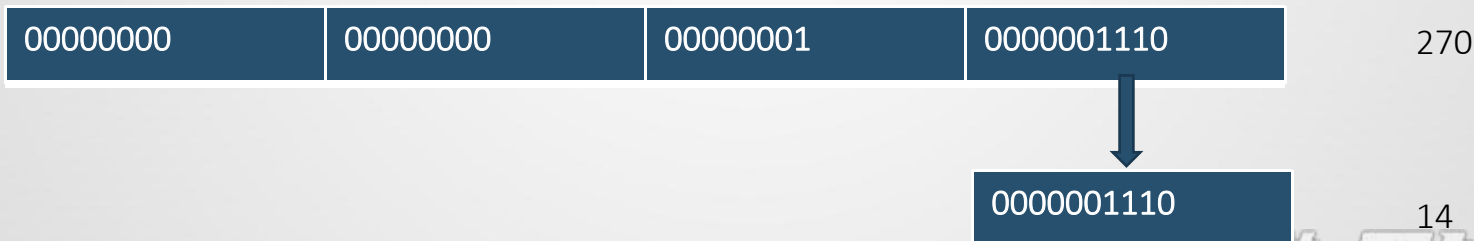
基本数据类型之间的转换

- 赋值运算时的转换
 - 强制类型转换（左<右）

```
int n=270;  
System.out.println("n="+n);  
byte b = (byte) n;  
System.out.println("b="+b);
```

```
C:\myjava>javac Test.java  
  
C:\myjava>java Test  
n=270  
b=14
```

- 强转时，当心丢失数据或失真
- 当将一种类型强制转换成另一种类型，而又超出了目标类型的表示范围，就会被截断成为一个完全不同的值。





基本数据类型之间的转换

- 总结

- =右侧：所有变量先转换为字节数最多的数据类型，再计算
- =两侧：左侧宽度>右侧宽度 自动转换
- 左侧宽度<右侧宽度 强制转换

- 特例

- 可以将整型常量直接赋值给byte, short, char等类型变量，而不需要进行强制类型转换，只要不超出其表数范围
- `short b = 12;` //合法
- `short b = 1234567;` //非法



总结

1.标识符

标识符的命名规则

2.关键字

3.数据类型：基本数据类型和引用数据类型

3.1整型数据类型 byte short int long

3.2浮点型数据类型 float double

3.3字符数据类型 char

3.4布尔数据类型 boolean

4.运算符

4.1算术运算符

难点 ++ -- 注意+的三个作用

4.2赋值运算符

= += -=...

sum+=6---->sum=sum+6

4.3关系运算符

> < == 关系运算的结果是boolean类型



总结

4.运算符

4.4逻辑运算符

操作数和结果都是boolean型

&和&&的区别和联系（是否短路）

4.5条件运算符

?: 唯一的三目运算符

```
int max = num1>num2?num1:num2;
```

4.6 位运算符 了解即可

4.7 运算符的优先级别

- 1.不用刻意背诵，多做练习
- 2.单目>算术>关系>逻辑>条件>赋值

5.基本数据类型的类型转换

- 1.自动类型转换 `double d=5;`
- 2.强制类型转换 `int n =(int)3.14;`

6.其他

- 6.1.字符集 Java采用Unicode字符集 常用字符集都兼容ASCII字符集
- 6.2.final常量 使用final修饰变量，变量变成常量，值不能再被改变
- 6.3.Scanner 可以接收来自键盘的输入