第 3 章

Java 语言基础

（视频讲解：58 分钟）

很多人认为学习 Java 语言之前必须要学习 C++语言，其实并非如此，这种错误认识是因为很多人在学习 Java 语言之前都学过 C++语言，事实上 Java 语言要比 C++语言更容易掌握。要掌握并熟练应用 Java 语言，就需要对 Java 语言的基础进行充分的了解。本章对 Java 语言基础进行了比较详细的介绍，对于初学者来说应该对本章的各个小节进行详细的阅读、思考，才能达到事半功倍的效果。

通过阅读本章，您可以：

- 了解 Java 程序的基本结构
- 了解 Java 中的标识符和关键字
- 了解 Java 语言中的基本数据类型
- 理解 Java 语言中的常量与变量
- 掌握 Java 语言中运算符的使用
- 理解 Java 语言中数据类型的转换
- 了解 Java 语言中的代码注释与编码规范

3.1 Java 程序的基本结构

视频讲解：光盘\TM\lx\3\Java 程序的基本结构.exe

要学习 Java 程序，首先应该了解程序的基本结构，这样有利于更进一步学习 Java 语言。一个 Java 程序的基本结构大体可以分为包、类、main()主方法、标识符、关键字、语句和注释等，如图 3.1 所示。

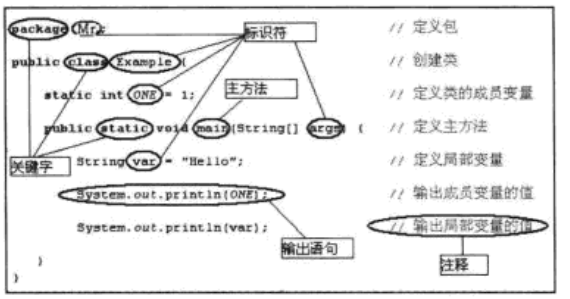

图 3.1 Java 程序的基本结构

第 1 条语句“package Mr;”定义了 Java 程序中类所在的包是 Mr，其中 Mr 是一个标识符，由程序员自己定义；package 是定义包的关键字，在 Java 程序中要定义包就必须使用 package 关键字。

说明 在 Java 语言中，标识符和关键字是区分大小写的，否则将导致程序无法正常执行。例如，如果将关键字 package 写成 Package，程序就会出错而无法正常执行，并且每条语句必须以分号结尾。

第 2 条语句“public class Example”是创建类的语句，其中 public 是 Java 程序的关键字，这里用于修饰类；class 是用于创建类的关键字，class 后的 Example 就是所创建类的名称，该类名由程序员自己定义。

第 3 条语句“static int ONE = 1;”定义了类的成员变量，其中 static 是 Java 程序的关键字，这里用于修饰成员变量；int 也是一个关键字，是 Java 程序中的整数数据类型，用于定义常量和变量，这里定义的是类的成员变量；ONE 是一个标识符，是该类的成员变量，其名称由程序员自己定义，该成员变量的数据类型是整数类型。

第 4 条语句“public static void main(String[] args)”是类的主方法，是 Java 应用程序的入口点，Java 程序是从该方法开始执行的，其中 main 是主方法的名称，程序员不可以更改；public 是 Java 程序的关键字，这里用于修饰主方法；static 是 Java 程序的关键字，这里用于修饰主方法；void 也是一个关键字，用于修饰方法，表示方法没有返回值；String 也是一个类，用于创建字符串对象，这里用于修饰主方法的形参，在 String 后跟上一对方括号“[”和“]”表示主方法的形参是一个字符串数组；args 是一个标识符，是主方法的形参数组，其数据类型是 String 类型。

说明 在 Java 语言中，主方法的写法是固定的，除了主方法的形参“String[] args”可以修改为“String args[]”以外，不可以改变第 4 条语句中的任何部分，否则 Java 程序都将无法运行。

第 5 条语句“String var = "Hello";”在主方法内定义了一个局部变量，其中 String 是一个类，用于

创建字符串对象，这里创建了一个局部变量；var 是局部变量的名称，是程序自己定义的一个标识符，"Hello"是局部变量 var 的值，是一个字符串常量；等号“=”是赋值运算符，用于将等号右边的字符串常量赋值给等号左边的变量 var，这样变量 var 的值就是 Hello 了。

说明 在 Java 程序中，除了字符串中常用的标点符号以外，代码中的所有标点符号必须是半角的或在英文输入法下输入（如逗号、分号、双引号等），否则程序会出错，这也是初学者容易犯的错误。

第 6 条语句“System.out.println(ONE);”是一个输出语句，可以在命令行或控制台输出信息，其中“System.out.println();”是输出语句的固定写法，其中 System 是一个系统类的名称，其第一个字母必须大写，out 是 System 类提供的一个标准输出流，println()是标准输出流 out 提供的方法，用于输出信息；println()方法内部的 ONE 是要输出的内容，这里的 ONE 是类的一个成员变量，其值是 1，所以执行该语句将输出 1。

最后一条语句“System.out.println(var);”是一个输出语句，其含义同第 6 条语句，这里的 var 是主方法内定义的一个局部变量，其值是"Hello"，所以执行该语句将输出 Hello。

3.2　标识符和关键字

视频讲解：光盘\TM\lx\3\标识符和关键字.exe

标识符是 Java 程序重要的组成部分，在程序中必须使用；关键字是 Java 中一些具有特殊意义的字符，不可以使用关键字对标识符进行命名。

3.2.1　标识符

标识符是 Java 程序中必须使用的，但也不是随便使用的，有一定的规则。本节将介绍什么是标识符和标识符的命名规则。

1．何为标识符

标识符可以简单地理解为一个名字，用来标识类名、变量名、方法名、数组名、文件名的有效字符序列。

【例 3.1】　定义变量 i 并赋值为 100。

```
int i = 100;
System.out.println(i);                    //在控制台输出信息
```

变量名 i 就是标识符，由程序员所起，但并不是随便取的，也有一定的规则。下面将介绍标识符的命名规则。

2．标识符的命名规则

标识符就是一个名字，对于所要表示的内容，用什么名字并不重要，但要通过标识符看出所写内

容即可。就好比人的名字，你叫什么名字并不重要，在别人叫这个名字时知道是你就可以了。标识符虽然可以任意取名，但是也要遵循以下规则：

- ☑ Java 语言的标识符由字母、数字、下划线和美元符号组成，第一个字符不能为数字。非法的标识符如 7word、5fox；合法的标识符如 tb_user、_u88。
- ☑ Java 语言使用 Unicode 标准字符集，最多可以识别 65535 个字符。因此，Java 语言中的字母可以是 Unicode 字符集中的任何字符，包括拉丁字母、汉字、日文和其他许多语言中的字符。合法的标识符如价格、User。
- ☑ 标识符不能是 Java 的关键字和保留字。非法的标识符如 this、goto。
- ☑ 在 Java 语言中标识符是区分大小写的，如果两个标识符的字母相同但是大小写不同，就是不同的标识符。good 和 Good 就是两个不同的标识符。

说明 在程序开发中，虽然可以使用汉字、日文等作为标识符，但为了避免出现错误，尽量不要使用，最好连下划线和数字也不要使用，而只用英文进行命名，且首字母用大写字母书写。

3.2.2 关键字

关键字是 Java 语言中已经被赋予特定意义的一些单词，不可以把这些字作为标识符来使用。Java 中的关键字如表 3.1 所示。

表 3.1 Java 关键字

int	public	this	finally	boolean	abstract
continue	float	long	short	throw	throws
return	break	for	static	new	interface
if	double	default	byte	do	case
strictfp	package	super	void	try	switch
else	catch	implements	private	final	class
extends	volatile	while	synchronized	instanceof	char
protecte	importd	transient	implements	dafaule	

说明 在命名标识符时，虽然 const 和 goto 不是 Java 的关键字，但也不可以使用，这两个词可能会在以后的升级版本中得以使用。

3.3 基本数据类型

视频讲解：光盘\TM\lx\3\基本数据类型.exe

在 Java 中有 8 种基本数据类型，其中 6 种是数值类型，另外两种分别是字符类型和布尔类型，而

6 种数值类型中有 4 种是整数类型，另外两种是浮点类型，如图 3.2 所示。

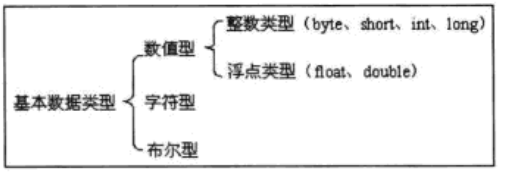

图 3.2　Java 基本数据类型

3.3.1　整数类型

整数类型用来存储整数数值，即没有小数部分的数值，可以是正数、负数，也可以是零。根据所占内存的大小不同，可以分为 byte、short、int 和 long 4 种类型。它们所占的内存和取值范围如表 3.2 所示。

表 3.2　整数类型

数 据 类 型	内存空间（8 位等于 1 字节）	取 值 范 围
byte	8 位（1 个字节）	-128～127
short	16 位（2 个字节）	-32768～32767
int	32 位（4 个字节）	-2147483648～2147483647
long	64 位（8 个字节）	-9223372036854775808～9223372036854775807

1．byte 型

使用 byte 关键字来定义 byte 型变量，可以一次定义多个变量并对其进行赋值，也可以不进行赋值。byte 型是整型中所分配的内存空间最少的，只分配 1 个字节；取值范围也是最小的，只在-128～127 之间，在使用时一定要注意，以免数据溢出产生错误。

【例 3.2】　定义 byte 型变量。

```
byte x = 48,y = -108,z;                    //定义 byte 型变量 x、y、z，并赋初值给 x、y
```

2．short 型

short 型即短整型，使用 short 关键字来定义 short 型变量，可以一次定义多个变量并对其进行赋值，也可以不进行赋值。系统给 short 型分配 2 个字节的内存，取值范围也比 byte 型大了很多，在-32768～32767 之间，虽然取值范围变大，但还是要注意数据溢出。

3．int 型

int 型即整型，使用 int 关键字来定义 int 型变量，可以一次定义多个变量并对其进行赋值，也可以不进行赋值。int 型变量取值范围很大，在-2147483648～2147483647 之间，足够一般情况下使用，所以是整型变量中应用最广泛的。

【例 3.3】　定义 int 型变量。

```
int x = 450,y = -462,z;                    //定义 int 型变量 x、y、z，并赋初值给 x、y
```

4．long 型

long 型即长整型，使用 long 关键字来定义 long 型变量，可以一次定义多个变量并对其进行赋值，也可以不进行赋值。而在对 long 型变量赋值时结尾必须加上“L”或者“l”，否则将不被认为是 long 型。当数值过大，超出 int 型范围时就使用 long 型，系统分配给 long 型变量 8 个字节，取值范围则更大，在-9223372036854775808～9223372036854775807 之间。

【例 3.4】 定义 long 型变量。

```
long x = 4556824L,y = -462447716l,z;    //定义 long 型变量 x、y、z，并赋初值给 x、y
```

说明

在定义 long 型变量时最好在结尾处加“L”，因为“l”非常容易和数字“1”弄混。

【例 3.5】 在项目中创建类 Number，在主方法中创建不同的整数类型变量，并将这些变量相加，将结果输出。（**实例位置：光盘\TM\sl\3\1**）

```
public class Number {                                          //创建类
    public static void main(String[] args) {                   //主方法
        byte mybyte = 124;                                     //声明 byte 型变量并赋值
        short myshort = 32564;                                 //声明 short 型变量并赋值
        int myint = 45784612;                                  //声明 int 型变量并赋值
        long mylong = 46789451L;                               //声明 long 型变量并赋值
        long result = mybyte+myshort+myint+mylong;             //获得各数相加后的结果
        System.out.println("四种类型相加的结果为："+result);    //将以上变量相加的结果输出
    }
}
```

运行结果如图 3.3 所示。

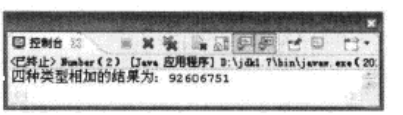

图 3.3 使用整数类型变量

上面的 4 种整数类型在 Java 程序中有 3 种表示形式，分别为十进制表示法、八进制表示法和十六进制表示法。

- ☑ 十进制表示法。十进制的表现形式大家都很熟悉，即逢十进一，每位上的数字最大是 9，如 120、0、-127 都是十进制数。
- ☑ 八进制表示法。八进制即逢八进一，每位上的数字最大是 7，且必须以 0 开头。例如，0123（转换成十进制数为 83）、-0123（转换成十进制数为-83）都是八进制数。
- ☑ 十六进制表示法。中国古代使用的就是十六进制数，所谓半斤八两就是如此，逢十六进一，每位上最大数字是 f（15），且必须以 0X 或 0x 开头。例如，0x25（转换成十进制数为 37）、0Xb01e（转换成十进制数为 45086）都是十六进制数。

3.3.2 浮点类型

浮点类型表示有小数部分的数字。在 Java 语言中，浮点类型分为单精度浮点类型（float）和双精

度浮点类型（double），它们具有不同的取值范围，如表 3.3 所示。

表 3.3　浮点类型

数 据 类 型	内存空间（8 位等于 1 字节）	取 值 范 围
float	32 位（4 个字节）	1.4E-45～3.4028235-E38
double	64 位（8 个字节）	4.9E-324～1.7976931348623157E-308

1．float 型

float 型即单精度浮点型，使用 float 关键字来定义 float 型变量，可以一次定义多个变量并对其进行赋值，也可以不进行赋值。在对 float 型进行赋值时在结尾必须添加“F”或者“f”，如果不加，系统自动将其定义为 double 型变量。float 型变量的取值范围在 1.4E-45 和 3.4028235E-38 之间。

【例 3.6】　定义 float 型变量。

```
float x = 12.521F,y = -5.264f,z;        //定义 float 型变量 x、y、z，并赋初值给 x、y
```

2．double 型

double 型即双精度浮点型，使用 double 关键字来定义 double 型变量，可以一次定义多个变量并对其进行赋值，也可以不进行赋值。在给 double 型赋值时，可以使用后缀“D”或“d”明确表明这是一个 double 类型数据，但加不加并没有硬性规定，可以加也可以不加。double 型变量的取值范围在 4.9E-324 和 1.7976931348623157E-308 之间。

【例 3.7】　定义 double 型变量。

```
double x = 12.521D,y = -5.264d,z=5.115,p;          //定义 double 型变量 x、y、z、p，并赋初值给 x、y、z
```

【例 3.8】　在项目中创建 SumNumber 类，在类的主方法中声明字符型变量，然后进行加法运算，并将运行结果输出。（**实例位置：光盘\TM\sl\3\2**）

```
public class SumNumber {                                          //创建类
    public static void main(String[] args) {                     //主方法
        float f1 = 13.23f;                                        //定义 float 型变量
        double d1 = 4562.12d;                                     //定义 double 型变量
        double d2 = 45678.1564;                                   //定义 double 型变量
        double result = f1+d1+d2;                                 //获得各数相加后的结果
        System.out.println("浮点型相加达到结果为："+result);          //将以上变量相加的结果输出
    }
}
```

运行结果如图 3.4 所示。

图 3.4　定义浮点类型

运行结果中输出了一个 16 位的数，这是因为 float 型变量是 8 位的，在转换成 16 位时自动在后面

补了一些数，这将在后面数据类型转换中详细介绍。

3.3.3 字符类型

char 型即字符类型，使用 char 关键字进行声明，用于存储单个字符，系统分配两个字节的内存空间。在定义字符型变量时，要用单引号括起来。例如，‘s’表示一个字符，且单引号中只能有一个字符，多了就不是字符类型了，而是字符串类型，需要用双引号进行声明。

【例 3.9】 声明 char 型变量。

```
char x = 'a';
```

由于字符 a 在 unicode 表中的排序位置是 97，因此允许将上面的语句写成：

```
char x = 97;
```

说明 与 C、C++语言一样，Java 语言也可以把字符作为整数对待。由于 Unicode 编码采用无符号编码，可以存储 65536 个字符（0x0000～0xffff），所以 Java 中的字符几乎可以处理所有国家的语言文字。若想得到一个 0～65536 之间的数所代表的 Unicode 表中相应位置上的字符，也必须使用 char 型显式转换。

【例 3.10】 在项目中创建类 Export，实现将 Unicode 表中某些位置上的字符以及一些字符在 Unicode 表中的位置在控制台上输出。

```
public class Export {                                          //定义类
    public static void main(String[] args) {                   //主方法
        int i = 'd';                                           //定义整型变量
        char c = 97;                                           //定义 char 型变量
        System.out.println("字符 d 的 Unicode 码是：" +i);
        System.out.println("Unicode 码 97 代表的字符是：" +c);
    }
}
```

运行结果如图 3.5 所示。

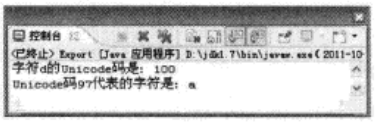

图 3.5 输出字符变量

在字符类型中有一种特殊的字符，以反斜线“\”开头，后跟一个或多个字符，具有特定的含义，不同于字符原有的意义，叫做转义字符。例如，“\n”就是一个转义字符，意思是“回车换行”。Java 中的转义字符如表 3.4 所示。

表 3.4 转义字符

转义字符	含义
\ddd	1～3 位八进制数据所表示的字符，如\456
\dxxxx	4 位十六进制所表示的字符，如\0052
\'	单引号字符

续表

转义字符	含　义
\\	反斜杠字符
\t	垂直制表符，将光标移到下一个制表符的位置
\r	回车
\n	换行
\b	退格
\f	换页

说明 转义字符本身也是字符，所以将转义字符赋值给字符变量时，与其他字符常量值一样需要使用单引号。

【例 3.11】　在项目中创建类 Gess，在类的主方法中使用转义字符定义字符型变量反斜杠和五角星，并在控制台上将转义字符输出。

```
public class Gess {                                          //定义类
    public static void main(String[] args) {                //主方法
        char char1 = '\\';                                   //将转义字符“\\”赋值给变量 char1
        char char2 = '\u2605';                               //将转义字符“\u2605”赋值给变量 char2
        System.out.println("输出反斜杠："char1);              //输出结果\
        System.out.println("输出五角星："char2);              //输出结果★
    }
}
```

运行结果如图 3.6 所示。

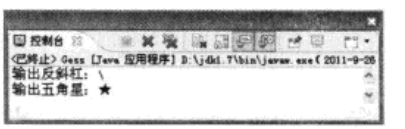

图 3.6　输出转义字符

3.3.4　布尔类型

布尔类型又称逻辑类型，只有 true 和 false 两个值，分别代表布尔逻辑中的“真”和“假”。使用 boolean 关键字声明布尔类型变量，通常被用在流程控制中作为判断条件。

【例 3.12】　声明 boolean 型变量。

```
boolean b1 = true,b2 = false,b3;          //定义布尔型变量 b1、b2、b3，并给 b1、b2 赋初值
```

3.4　变量与常量

视频讲解：光盘\TM\lx\3\变量与常量.exe

在程序执行过程中，其值不能改变的量称为常量，其值能被改变的量称为变量。变量与常量的声

明都必须使用合法的标识符，所有变量与常量只有在声明之后才能使用。

3.4.1 声明变量

在程序设计中，变量的使用是一个十分重要的环节。定义一个变量，就是要告诉编译器（compiler）这个变量属于哪一种数据类型，这样编译器才知道需要配置多少空间，以及能存放什么样的数据。变量都有一个变量名，变量名必须是合法的标识符，内存空间内的值就是变量值。在声明变量时可以不给予赋值，也可以直接赋给初值。

【例 3.13】 声明变量。

```
int age;                          //声明 int 型变量
char char1 = 'r';                 //声明 char 型变量并赋值
```

以上程序代码的内存状态如图 3.7 所示。

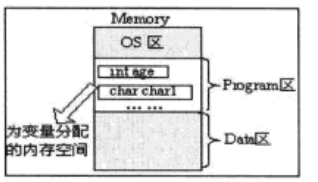

图 3.7 变量占用的内存空间

由图 3.7 可知，系统的内存可大略分为 3 个区域，即系统区（OS）、程序区（Program）和数据区（Data）。当程序执行时，程序代码会加载到内存中的程序区，数据暂时存储在数据区中。

变量虽然是由程序员所命名的，但是变量的命名并不是任意的，需要遵循一定的规则。Java 中变量的命名规则如下：

- ☑ 变量名必须是一个有效的标识符。变量名必须使用 Java 语言中合法的标识符，即以字母、数字和下划线组成，且首字母不能是数字，还不可以使用 Java 中的关键字。
- ☑ 变量名不能重复。如果两个变量具有同样的变量名，系统在对其进行使用时就不知道调用哪个变量，运行结果就会出现错误。
- ☑ 应选择有意义的单词作为变量名。在命名变量名时，最好能通过变量名看出变量的内容，这样既方便读者对程序的理解，增加可读性；又方便程序的维护，减轻程序维护人员的工作负担。

说明 在 Java 语言中允许使用汉字或其他语言文字作为变量名，如“int 年龄 =21”，在程序运行时并不出现什么错误，但建议读者尽量不要使用这些语言文字作为变量名。

【例 3.14】 在项目中创建类 Gess，声明整型变量 i 并赋给初值 10，之后再将 100 赋给 i，最后将 i 的值输出。

```
public class Gess {                                    //定义类
    public static void main(String[] args) {           //主方法
        int i = 10;                                    //定义变量 i，并赋给初值 10
        System.out.println("i 的初值是："+i);           //输出 i 的值
        i = 100;                                       //将 100 赋给 i
        System.out.println("i 现在的值是："+i);         //输出 i 的值
    }
}
```

运行结果如图 3.8 所示。

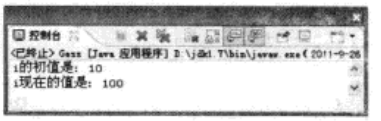

图 3.8　输出变量的值

3.4.2　声明常量

在程序运行过程中一直不会改变的量称为常量（constant），通常也被称为“final 变量”。常量在整个程序中只能被赋值一次。在为所有对象共享值时，常量是非常有用的。

在 Java 语言中声明一个常量，除了要指定数据类型外，还需要通过 final 关键字进行限定。声明常量的标准语法格式如下：

```
final 数据类型 常量名称[=值]
```

常量名通常使用大写字母，但这并不是必需的，很多 Java 程序员使用大写字母表示常量，常常是为了清楚地表明正在使用常量。

【例 3.15】　声明常量。

```
final double PI = 3.1415926F;                    //声明 double 型常量 PI 并赋值
final boolean BOOL = true;                       //声明 boolean 型常量 BOOL 并赋值
```

说明　当定义的常量如果属于“成员变量”，则必须在定义时就赋给初值，否则将会在编译时出现错误。“成员变量”将在 3.4.3 节中讲解。

【例 3.16】　在项目中创建类 Part，在类体中创建变量 age 与常量 PI。在主方法中分别给变量与常量赋值，通过输出信息可测试变量与常量的有效范围。（**实例位置：光盘\TM\sl\3\3**）

```
public class Part {                                                    //新建类 Part
    static final double PI = 3.14;                                     //声明常量 PI
    static int age = 23;                                               //声明 int 型变量 age 并进行赋值
    public static void main(String[] args) {                           //主方法
        final int number;                                              //声明 int 型常量 number
        number = 1235;                                                 //对常量进行赋值
        age = 22;                                                      //再次对变量进行赋值
        System.out.println("常量 PI 的值为：" + PI);                    //将 PI 的值输出
        System.out.println("赋值后 number 的值为:" + number);           //将 number 的值输出
        System.out.println("int 型变量 age 的值为：" + age);            //将 age 的值输出
    }
}
```

运行结果如图 3.9 所示。

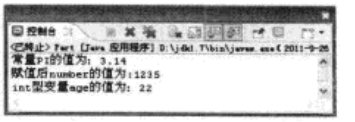

图 3.9　使用常量和变量

3.4.3　变量的有效范围

变量的有效范围是指程序代码能够访问该变量的区域，若超出变量所在区域访问变量则编译时会

出现错误。在程序中，一般会根据变量能够访问的区域将变量分为“成员变量”和“局部变量”。

1. 成员变量

在类体中定义的变量被称为成员变量，成员变量在整个类中都有效。类的成员变量又可分为静态变量和实例变量两种。

【例 3.17】 声明静态变量和实例变量。

```
class var {
    int x = 45;                                    //定义实例变量
    static int y = 90;                             //定义静态变量
}
```

其中，x 是实例变量，y 为静态变量（也称类变量）。如果成员变量的类型前面加上关键字 static，这样的成员变量称为静态变量。静态变量的有效范围可以跨类，甚至可达到整个应用程序之内。对于静态变量，除了能在定义它的类内存取，还能直接以“类名.静态变量”的方式在其他类内使用。

2. 局部变量

在类的方法体中定义的变量（方法内部定义，“{”与“}”之间的代码中声明的变量）称之为局部变量。局部变量只在当前代码块中有效，通俗地理解就是在其所定义的大括号内有效，出了这个大括号就没有效了，在其他类体中不能调用该变量。

局部变量的生命周期取决于方法，当方法被调用时，Java 虚拟机为方法中的局部变量分配内存空间，当该方法的调用结束后，则会释放方法中局部变量占用的内存空间，局部变量也随即销毁。

【例 3.18】 在项目中创建类 Val，分别定义名称相同的局部变量与成员变量，最后在控制台输出变量，观察输出的结果。

```
public class Val {                                 //新建类
    static int times = 3;                          //定义成员变量 times
    public static void main(String[] args) {       //主方法
        int times = 4;                             //定义局部变量 times
        System.out.println("times 的值为：" + times);      //将 times 的值输出
        System.out.println("times 的值为：" + Val.times);  //输出静态变量
    }
}
```

运行结果如图 3.10 所示。

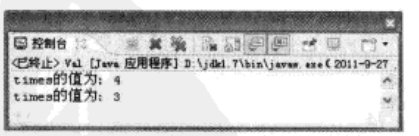

图 3.10 输出变量的值

说明 局部变量可与成员变量的名字相同，此时成员变量将被隐藏。即这个成员变量在此方法中暂时失效。如果想要调用成员变量，需要使用“类名.静态变量”调用。

成员变量和局部变量都有各自的有效范围，如图 3.11 所示。

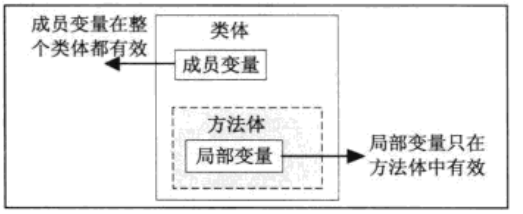

图 3.11　变量的有效范围

3.5 运　算　符

视频讲解：光盘\TM\lx\3\运算符.exe

运算符是一些特殊的符号，主要用于数学函数、一些类型的赋值语句和逻辑比较方面。Java 中提供了丰富的运算符，如赋值运算符、算术运算符、比较运算符等，本节将向读者介绍这些运算符。

3.5.1 赋值运算符

赋值运算符即“=”，是一个二元运算符（即对两个操作数进行处理），其功能是将右方操作数所含的值赋值给左方的操作数。语法格式如下：

```
变量类型 变量名 = 所赋的值;
```

左方必须是一个变量，而右边所赋的值可以是任何数值或表达式，包括变量（如 a、number）、常量（如 123、‘book’）或有效的表达式（如 45*12）。

【例 3.19】　使用赋值运算符为变量赋值。

```
int a = 10;                                    //声明 int 型变量 a
int b = 5;                                     //声明 int 型变量 b
int c = a+b+2;                                 //将变量 a、b 和 2 进行运算后的结果赋值给 c
```

遵循赋值运算符的运算规则，可知系统将先计算 a+b+2 的值，结果为 17。然后将 17 赋值给变量 c，因此运算后“c=17”。

【例 3.20】　在项目中创建 Eval 类，在类的主方法中定义变量，使用赋值运算符为变量赋值。

```
int a, b, c;                                   //声明 int 型变量 a、b、c
a = 15;                                        //将 15 赋值给变量 a
c = b = a +4;                                  //将 a+4 的值赋给变量 b、c
System.out.println("c 值为：" + c);             //将变量 c 的值输出
System.out.println("b 值为：" + b);             //将变量 b 的值输出
```

运行结果 b 和 c 的值都等于 19。

说明 在 Java 中可以把赋值运算符连在一起使用。例如：

```
x = y = z = 5;
```

在这个语句中，变量 x、y、z 得到同样的值 5。赋值运算符使用的顺序是先将 a+4 的值赋给 b，再将 b 的值赋给 c，从右向左运算。不过，在程序开发中不建议使用这种赋值语法。

3.5.2 算术运算符

Java 中的算术运算符主要有+（加号）、-（减号）、*（乘号）、/（除号）和%（求余），它们都是二元运算符。Java 中算术运算符的功能及使用方式如表 3.5 所示。

表 3.5 算术运算符

运 算 符	说 明	实 例	结 果
+	加	12.45f+15	31.45
-	减	4.56-0.16	4.4
*	乘	5L*12.45f	62.25
/	除	7/2	3
%	取余数	1%2	2

其中“+”和“-”运算符还可以作为数据的正负符号，如+5、-7。

注意 除法运算时，要记住 0 不可以作除数。例如，int a = 5/0;系统会报出 ArithmeticException 的异常。

【例 3.21】 在项目中创建 Arith 类，在类的主方法中定义变量，使用算术运算符对变量进行计算，并在控制台将计算结果输出。（实例位置：光盘\TM\sl\3\4）

```
public class Arith {                                                    //创建类
    public static void main(String[] args) {                            //主方法
        float number1 = 45.56f;                                         //声明 float 型变量并赋值
        int number2 = 152;                                              //声明 int 型变量并赋值
        System.out.println("45.56f 和 152 的和为：" + number1 + number2);   //将变量相加之和输出
        System.out.println("45.56f 和 152 的差为：" + (number2 - number1));  //将变量相减之差输出
        System.out.println("45.56f 和 152 的积为：" + number1 * number2);   //将变量相乘的积输出
        System.out.println("45.56f 和 152 的商为：" + number1 / number2);   //将变量相除的商输出
    }
}
```

运行结果如图 3.12 所示。

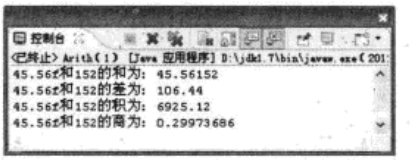

图 3.12　使用算术运算符运算

3.5.3　自增和自减运算符

自增、自减运算符是单目运算符，可以放在操作元之前，也可以放在操作元之后。操作元必须是一个整型或浮点型变量。放在操作元前面的自增、自减运算符，会先将变量的值加 1（减 1），然后再使该变量参与表达式的运算；放在操作元后面的自增、自减运算符，会先使变量参与表达式的运算，然后再将该变量加 1（减 1）。示例代码如下：

```
++a(--a)                    //表示在使用变量 a 之前，先使 a 的值加（减）1
a++(a--)                    //表示在使用变量 a 之后，使 a 的值加（减）1
```

【例 3.22】　粗略地分析++a 与 a++的作用都相当于 a = a+1，赋值 a = 4，计算 a++和++a 的值。

```
int a = 4;                  //声明变量 a，并赋值
b = ++a;                    //将 a 自加赋给 b
b = a++;                    //将 a 自加赋给 b
```

第 2 条语句是先将 a 自加，把 a 自加后的值赋给 b，自加后 a 的值是 5，b 的值也是 5；第 3 条语句是先将 a 的值赋给 b，然后再将 a 自加，自加后 a 的值是 5，而 b 的值是 4。

3.5.4　比较运算符

比较运算符属于二元运算符，用于程序中的变量和变量之间、变量和常量之间以及其他类型的信息之间的比较。比较运算符的运算结果是 boolean 型，当运算符对应的关系成立时，运算结果是 true，否则结果是 false。比较运算符通常用在条件语句中来作为判断的依据。比较运算符的种类和用法如表 3.6 所示。

表 3.6　比较运算符

运　算　符	作　　用	举　　例	操 作 数 据	结　　果
>	比较左方是否大于右方	'a'>'b'	整型、浮点型、字符型	false
<	比较左方是否小于右方	156 < 456	整型、浮点型、字符型	false
==	比较左方是否等于右方	'c'=='c'	基本数据类型、引用型	true
>=	比较左方是否大于等于右方	479>=426	整型、浮点型、字符型	true
<=	比较左方是否小于等于右方	12.45<=45.5	整型、浮点型、字符型	false
!=	比较左方是否不等于右方	'y'!= 't'	基本数据类型、引用型	true

【例 3.23】 在项目中创建 Compare 类，在类的主方法中创建整型变量，使用比较运算符对变量进行比较运算，将运算后的结果输出。（**实例位置：光盘\TM\sl\3\5**）

```
public class Compare {                                          //创建类
    public static void main(String[] args) {
        int number1 = 4;                                        //声明 int 型变量 number1
        int number2 = 5;                                        //声明 int 型变量 number2
        //依次将变量 number1 与变量 number2 的比较结果输出
        System.out.println("4>5 是否成立：" + (number1 > number2));
        System.out.println("4<5 是否成立：" + (number1 < number2));
        System.out.println("4==5 是否成立：" + (number1 == number2));
        System.out.println("4!=5 是否成立：" + (number1 != number2));
        System.out.println("4>=5 是否成立：" + (number1 >= number2));
        System.out.println("4<=5 是否成立：" + (number1 <= number2));
    }
}
```

运行结果如图 3.13 所示。

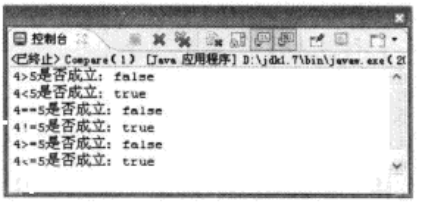

图 3.13 使用比较运算符比较

3.5.5 逻辑运算符

逻辑运算符包括&&（&）（逻辑与）、||（|）（逻辑或）和!（逻辑非），返回值为布尔类型的表达式，操作元也必须是 boolean 型数据。与比较运算符相比，逻辑运算符可以表示更加复杂的条件，如连接几个关系表达式进行判断。在逻辑运算符中，除了“!”是一元运算符之外，其余的都是二元运算符，其用法和含义如表 3.7 所示。

表 3.7 逻辑运算符

运 算 符	含 义	用 法	结 合 方 向
&&、&	逻辑与	op1&&op2	左到右
||、|	逻辑或	op1||op2	左到右
!	逻辑非	! op	右到左

用逻辑运算符进行逻辑判断时，不同的逻辑运算符与不同的操作元进行操作时，运行结果也不相同，运行结果如表 3.8 所示。

表 3.8 使用逻辑运算符进行逻辑运算

表达式 1	表达式 2	表达式 1&&表达式 2	表达式 1||表达式 2	! 表达式 1
true	true	true	true	false
true	false	false	true	false
false	false	false	false	true
false	true	false	true	true

说明 在 Java 中，逻辑运算符“&&”与“&”都表示“逻辑与”，那么它们之间的区别在哪里呢？从表 3.8 中可以看出，当两个表达式都为 true 时，逻辑与的结果才会是 true。使用逻辑运算符“&”会判断两个表达式；而逻辑运算符“&&”则是针对 boolean 类型进行判断，当第一个表达式为 false 时则不去判断第二个表达式，直接输出结果。使用“&&”可节省计算机判断的次数。通常将这种在逻辑表达式中从左端的表达式可推断出整个表达式的值称为“短路”，而那些始终执行逻辑运算符两边的表达式称为“非短路”。“&&”属于“短路”运算符，而“&”则属于“非短路”运算符。“||”和“|”也是如此。

【例 3.24】　在项目中创建 Calculation 类，在类的主方法中创建整型变量，使用逻辑运算符对变量进行运算，并将运算结果输出。

```
public class Calculation {                                          //创建类
    public static void main(String[] args) {
        int a = 2;                                                  //声明 int 型变量 a
        int b = 5;                                                  //声明 int 型变量 b
        boolean result1 = ((a > b) && (a != b));                    //声明布尔型变量
        boolean result2 = ((a > b) || (a != b));                    //声明布尔型变量
        System.out.println("(a > b) && (a != b)的值是："+result1);    //将变量 result1 输出
        System.out.println(" (a > b) || (a != b)的值是："+result2);    //将变量 result2 输出
    }
}
```

运行结果如图 3.14 所示。

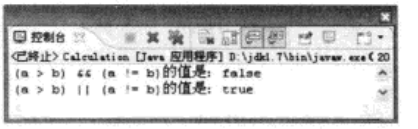

图 3.14　使用逻辑运算符判断

3.5.6　位运算符

位运算符用于处理整型和字符型的操作数，对其内存进行操作，数据在内存中以二进制的形式表示。例如，int 型变量 7 的二进制表示是 00000000 00000000 00000000 00000111，-8 的二进制表示是 11111111 11111111 11111111 11111000，最高位是符号位，0 表示正数，1 表示负数。Java 语言提供的位运算符如表 3.9 所示。

表 3.9　位运算符

运　算　符	含　义	用　法	运 算 分 类
~	按位取反	~op1	按位运算
&	按位与	op1 & op2	
|	按位或	op1 | op2	
^	按位异或	op1 ^ op2	
<<	左移	op1 << op2	移位运算符
>>	右移	op1 >> op2	
>>>	无符号右移	op1 >>> op2	

1. “按位与”运算

“按位与”运算的运算符为“&”，是双目运算符。其运算的法则是：如果两个操作数对应位都是1，则结果位才是 1，否则为 0。如果两个操作数的精度不同，则结果的精度与精度高的操作数相同，如图 3.15 所示。

2. “按位或”运算

“按位或”运算的运算符为“|”，是双目运算符。其运算法则是：如果两个操作数对应位都是 0，则结果位才是 0，否则为 1。如果两个操作数的精度不同，则结果的精度与精度高的操作数相同，如图 3.16 所示。

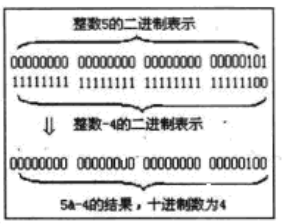

图 3.15　5&-4 的运算过程

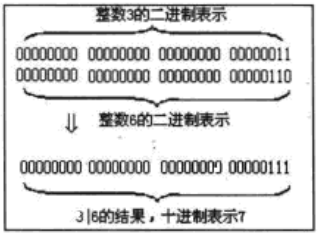

图 3.16　3|6 的运算过程

3. “按位非”运算

“按位非”运算也称“按位取反”运算，运算符为“～”，是单目运算符。其运算法则是：将操作数二进制中的 1 全部修改为 0，0 全部修改为 1，如图 3.17 所示。

4. “按位异或”运算

“按位异或”运算的运算符是“^”，是双目运算符。其运算法则是：当两个操作数的二进制表示相同（同时为 0 或同时为 1）时，结果为 0，否则为 1。若两个操作数的精度不同，则结果数的精度与精度高的操作数相同，如图 3.18 所示。

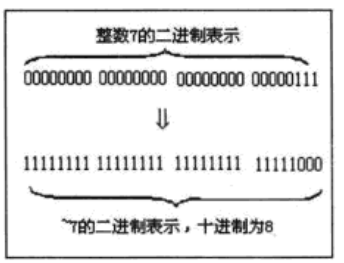

图 3.17　～7 的运算过程

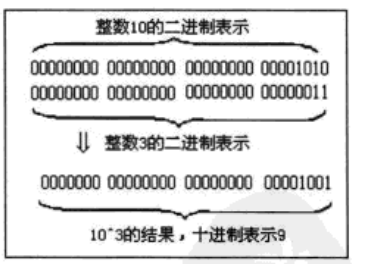

图 3.18　10^3 的运算过程

5. 移位运算符

Java 语言中的移位运算符有 3 种，其操作的数据类型只有 byte、short、char、int 和 long 5 种。

☑ 左移运算符“<<”。所谓左移运算符，就是将左边的操作数在内存中的二进制数据左移右边操作数指定的位数，左边移空的部分补 0。示例代码如下：

```
48 << 1;                    //将 48 的二进制数向左移 1 位
```

将 48 的二进制数向左移 1 位，移位后的结果是 96。

☑ 右移运算符“>>”。右移则复杂一些，当使用“>>”符号时，如果最高位是 0，左移空的位就填入 0；如果最高位是 1，右移空的位就填入 1，使用方法与左移类似。示例代码如下：

```
48 >> 1;                    //将 48 的二进制数向右移 1 位
```

将 48 的二进制数向右移 1 位，移位后的结果是 24。移位过程如图 3.19 所示。

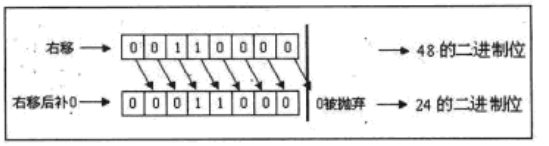

图 3.19　使用移位运算符移位

☑ 无符号右移运算符“>>>”。Java 还提供了无符号右移运算符“>>>”，不管最高位是 0 还是 1，左移空的高位都填入 0。

技巧　移位能让用户实现整数除以或乘以 2 的 n 次方的效果。例如，y<<2 与 y*4 的结果相同；y>>1 的结果与 y/2 的结果相同。总之，一个数左移 n 位，就是将这个数乘以 2 的 n 次方；一个数右移 n 位，就是将这个数除以 2 的 n 次方。

【例 3.25】　在项目中创建 Calculation 类，在类的主方法中创建整型变量，使用逻辑运算符对变量进行运算，将运算结果输出。

```
public class Calculation {                                  //声明类
    public static void main(String[] args) {                //主方法
        int i = 46;                                         //定义变量
        int j = 97;
        char c = 'a';
        System.out.println("46&97 的值是："+(i&j));          //输出 i 和 j 按位与的结果
        System.out.println("46^a 的值是："+(i^c));           //输出 i 和 j 按位异或的结果
        System.out.println("46>>1 的值是："+(i>>1));         //将 i 按位右移 1 位
    }
}
```

运行结果如图 3.20 所示。

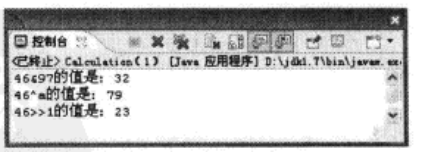

图 3.20　使用逻辑运算符判断

3.5.7　三元运算符

三元运算符是 Java 中唯一一个三目运算符，其操作元有 3 个，第一个是条件表达式，其余的是两个值，条件表达式成立时运算取第一个值，不成立时取第二个值。示例代码如下：

```
boolean b = 20 < 45 ? true : false;
```

三元运算符用于判断，等价的 if…else 语句如下：

```
boolean a;                          //声明 boolean 变量
if(20<45)                           //将 20<45 作为判断条件
    a = true;                       //条件成立将 true 赋值给 a
else
    a = false;                      //条件不成立将 false 赋值给 a
```

当表达式“20<45”的运算结果返回真时，则 boolean 型变量 a 取值 true；当表达式“20<45”返回假时，则 boolean 型变量 a 取值 false，此例的结果是 true。

【例 3.26】 在项目中创建 Calculation 类，在类的主方法中创建整型变量，使用三目运算符，并将运算结果输出。

```
public class Calculation {                                    //声明类
    public static void main(String[] args) {                  //主方法
        int i = 46;                                           //局部变量
        int j = 97;
        int z = i > j ? 100 : 200;                            //使用三目运算符
        System.out.println("i>j?100:200 的值是："+z);          //输出 z
    }
}
```

运行结果如图 3.21 所示。

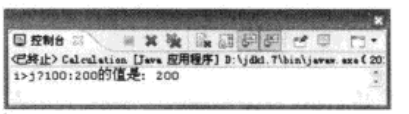

图 3.21 使用三目运算符

3.5.8 运算符优先级

Java 中的表达式就是使用运算符连接起来的符合 Java 规则的式子，运算符的优先级决定了表达式中运算执行的先后顺序。各运算符的优先级由高到低如图 3.22 所示。

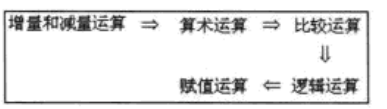

图 3.22 运算符的优先级

各运算符之间大致的优先级由图 3.22 列出，但如果两个符号属于同一运算符又怎么区分优先级呢？Java 中各符号的优先级如表 3.10 所示。

表 3.10 运算符的优先级

优 先 级	描 述	运 算 符
1	括号	( )
2	正负号	+、-
3	一元运算符	++、--、!
4	乘除	*、/、%
5	加减	+、-
6	移位运算	>>、>>>、<<
7	比较大小	<、>、>=、<=
8	比较是否相等	==、!=

续表

优　先　级	描　　述	运　算　符
9	按位与运算	&
10	按位异或运算	^
11	按位或运算	|
12	逻辑与运算	&&
13	逻辑或运算	||
14	三元运算符	? :
15	赋值运算符	=

说明　如果两个运算有相同的优先级，那么左边的表达式要比右边的表达式先被处理。在编写程序时尽量使用括号运算符来限定运算次序，以免产生错误的运算顺序。

3.5.9　范例 1：不用其他变量实现两变量互换

在对变量进行互换时，将创建一个临时变量来共同完成互换，临时变量的创建增加了系统资源的消耗。如果需要交换的是两个整数类型的变量，则可以使用更高效的方法，使用异或运算符进行互换，而不使用第三变量。运行结果如图 3.23 所示。（**实例位置：光盘\TM\sl\3\6**）

图 3.23　实现变量互换

在项目中创建 VariableExchange 类，在类中创建扫描器对象接收用户输入的两个变量值，然后通过位运算符中的异或运算符“^”实现两个变量的互换。代码如下：

```
import java.util.Scanner;
public class VariableExchange {                                    //声明类
    public static void main(String[] args) {                      //主方法
        Scanner scan = new Scanner(System.in);                     //创建扫描器
        System.out.println("请输入变量 A 的值");
        long A = scan.nextLong();                                  //接收第一个变量值
        System.out.println("请输入变量 B 的值");
        long B = scan.nextLong();                                  //接收第二个变量值
        System.out.println("A=" + A + "\tB=" + B);
        System.out.println("执行变量互换...");
        A = A ^ B;                                                 //执行变量互换
        B = B ^ A;
        A = A ^ B;
        System.out.println("A=" + A + "\tB=" + B);                 //输出交换后的结果
    }
}
```

3.5.10 范例 2：判断数字的奇偶性

根据数字被 2 除的余数来判断一个数的奇偶性，如果余数是 0 该数是偶数，否则是奇数。本范例使用三元运算符进行判断。运行结果如图 3.24 所示。（**实例位置：光盘\TM\sl\3\7**）

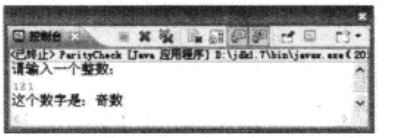

图 3.24 判断数字的奇偶性

在项目中创建 ParityCheck 类，在该类的主方法中创建扫描器对象接受用户输入的数字，通过三元运算符判断该数的奇偶性，并在控制台输出。代码如下：

```
import java.util.Scanner;
public class ParityCheck {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);                    //创建输入流扫描器
        System.out.println("请输入一个整数：");
        long number = scan.nextLong();                            //获取用户输入的整数
        String check = (number % 2 == 0) ? "这个数字是：偶数" : "这个数字是：奇数";
        System.out.println(check);
    }
}
```

3.6 类 型 转 换

类型转换是将变量从一种类型更改为另一种类型的过程。例如，可以将 String 类型数据 456 转换为一个 int 整型变量 456。Java 对数据类型的转换有严格的规定，数据从占用存储空间较小的类型转换为占用存储空间较大的数据类型时，则做自动类型转换（隐式类型转换）；反之则必须做强制类型转换（显式类型转换）。

3.6.1 自动类型转换

Java 中 8 种基本类型可以进行混合运算，不同类型的数据在运算过程中首先会自动转换为同一类型，再进行运算。数据类型根据占用存储空间的大小分为高低不同的级别，占用空间小的级别低，占用空间大的级别高，自动类型转换遵循低级到高级转换的规则。

【例 3.27】 定义 float 型变量 number1 并赋值 45，int 型变量 number2 并赋值 152，在控制台将 number1+number2 的值输出。

```
float number1 = 45f;                                              //声明 float 型变量并赋值
int number2 = 152;                                                //声明 int 型变量并赋值
System.out.println(number1 + number2);
```

系统先将 int 型变量转换成 float 变量之后相加，此时执行输出语句，在控制台上输出 197.0。

自动类型转换要遵循一定的规则，那么在运算时各类型间将怎么转换呢？各种情况下数据类型间转换的一般规则如表 3.11 所示。

表 3.11　隐式类型转换规则

操作数 1 的数据类型	操作数 2 的数据类型	转换后的数据类型
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

【例 3.28】　在项目中创建 Conver 类，在类的主方法中创建不同数值型的变量进行运算，在控制台上将运行结果输出。（实例位置：光盘\TM\sl\3\8）

```
public class Conver{                                          //创建类
    public static void main(String[] args) {                  //主方法
        byte mybyte = 127;                                    //定义 byte 型变量 mybyte，并赋值 127
        int myint = 150;                                      //定义 int 型变量 myint，并赋值 150
        float myfloat = 452.12f;                              //定义 float 型变量 myfloat，并赋值
        char mychar = 10;                                     //定义 char 型变量 mychar，并赋值
        double mydouble = 45.46546;                           //定义 double 型变量，并赋值
        //将运算结果输出
        System.out.println("127 与 452.12 相加的和是：" + (mybyte + myfloat));
        System.out.println("127 与 150 相乘的积是：" + mybyte * myint);
        System.out.println("127 被 10 除的商是：" + mybyte / mychar);
        System.out.println("45.46546 和 10 相加的和是：" + (mydouble + mychar));
    }
}
```

运行结果如图 3.25 所示。

图 3.25　变量的自动类型转换

3.6.2　强制类型转换

当把高精度的变量的值赋给低精度的变量时，必须使用显式类型转换运算（又称强制类型转换）。语法格式如下：

```
(类型名)要转换的值
```

【例 3.29】　将不同的数据类型进行显式类型转换。

```
int a = (int)45.23;                                           //此时输出 a 的值为 45
long y = (long)456.6F;                                        //此时输出 y 的值为 456
int b = (int)'d';                                             //此时输出 b 的值为 100
```

当把整数赋值给一个 byte、short、int、long 型变量时，不可超出这些变量的取值范围，否则就会发生数据溢出。示例代码如下：

```
Short s = 516;
byte b = (byte)s;
```

由于 byte 型变量的最大值是 127，516 已经超过了其取值范围，因此发生数据溢出。数据转换的过程如图 3.26 所示。

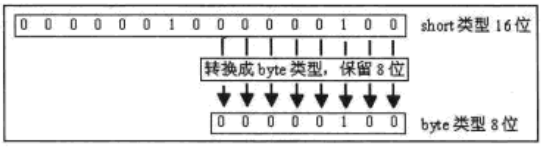

图 3.26　short 型转换成 byte 类型过程

此时就造成了数据丢失，所以在使用强制数据类型转换时，一定要加倍小心，不要超出变量的取值范围，否则就得不到想要的结果。

注意

boolean 型的值不能被转换为其他数据类型，反之亦然。

3.6.3　范例 3：类型转换实战

不管是自动类型转换还是强制类型转换都是程序设计中经常用到的，下面针对数据类型转换展开实战，运行结果如图 3.27 所示。（**实例位置：光盘\TM\sl\3\9**）

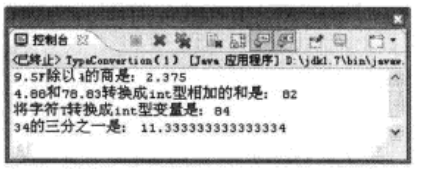

图 3.27　数据类型转换

在项目中创建 TypeConvertion 类，在类的主方法中定义不同数据类型的变量，然后在不同数据类型之间进行转换，并在控制台输出。代码如下：

```
public class TypeConvertion {                                          //声明类
    public static void main(String[]args){                             //主方法
        int intNum = 4;                                                //定义变量
        float floatNum = 9.5F;

        floatNum /= intNum;                                            //自动类型转换成 float 型
        System.out.println("9.5F 除以 4 的商是："+floatNum);

        double numX = 4.88;
        double numY = 78.83;
        int numZ = (int)numX+(int)numY;                                //将 double 型强制转换成 int 型
        System.out.println("4.88 和 78.83 转换成 int 型相加的和是：  "+numZ);

        char charVar = 'T';
        int intVar = (int)charVar;                                     //将 char 型强制转换成 int 型
        System.out.println("将字符 T 转换成 int 型变量是："+intVar);
```

```
        int num1 = 34;
        double num2 = (double)num1/3;                    //将 int 型强制转换成 double 型
        System.out.println("34 的三分之一是：  "+num2);
    }
}
```

3.7　代码注释和编码规范

视频讲解：光盘\TM\lx\3\代码注释和编码规范.exe

在程序代码中适当地添加注释可以提高程序的可读性、可维护性；好的编码规范可以使程序更易阅读和理解。本节将向读者介绍 Java 中的几种代码注释以及应该注意的编码规范。

3.7.1　代码注释

通过在程序代码中添加注释可提高程序的可读性，注释中包含了程序的信息，可以帮助程序员更好地阅读和理解程序。在 Java 源程序文件的任意位置都可添加注释语句，注释中的文字 Java 编译器并不进行编译，所有代码中的注释文字并不对程序产生任何影响。Java 语言提供了 3 种添加注释的方法，分别为单行注释、多行注释和文档注释。

1．单行注释

“//”为单行注释标记，从符号“//”开始直到换行为止的所有内容均作为注释而被编译器忽略。语法格式如下：

```
//注释内容
```

声明 int 型变量 age，并用单行注释加以注释。示例代码如下：

```
int age ;              //定义 int 型变量用于保存年龄信息
```

2．多行注释

“/* */”为多行注释标记，符号“/*”与“*/”之间的所有内容均为注释内容。注释中的内容可以换行。语法格式如下：

```
/*
    注释内容 1
    注释内容 2
    ……
*/
```

在多行注释中可嵌套单行注释。示例代码如下：

```
/*
    程序名称：Hello word  //开发时间:2008-03-05
*/
```

但在多行注释中不可嵌套多行注释。非法代码如下：

```
/*
    程序名称：Hello word
    /*开发时间:2008-03-05
    作者：张先生
    */
*/
```

3. 文档注释

“/** */”为文档注释标记。符号“/**”与“*/”之间的内容均为文档注释内容。当文档注释出现在任何声明（如类的声明、类的成员变量的声明、类的成员方法的声明等）之前时，会被 Javadoc 文档工具读取作为 Javadoc 文档内容。文档注释的格式与多行注释的格式相同。对于初学者而言，文档注释并不是很重要，了解即可。语法格式如下：

```
/**
    *程序名称：Hello word
    *开发时间:2008-03-05
    *作者：张先生
*/
```

说明 一定要养成良好的编程风格。软件编码规范中提到“可读性第一，效率第二”，所以程序员必须要在程序中添加适量的注释来提高程序的可读性和可维护性。程序中注释要占程序代码总量的 20%～50%。

3.7.2 编码规范

在学习开发的过程中要养成良好的编码规范，因为规整的代码格式会给程序的开发与日后的维护提供很大方便。总结的编码规范如下：

- ☑ 每条语句要单独占一行。虽然 Java 语言中可以在一行当中写几条语句，但为了程序看起来更加规范，且便于维护，要养成每行只写一条语句的好的编码规范。
- ☑ 每条命令都要以分号结束。语句要以分号结尾，程序代码中的分号必须为英文状态下的，初学者经常会将“;”写成中文状态下的“；”，此时编译器会报出 illegal character（非法字符）这样的错误信息。
- ☑ 声明变量时要分行声明。即使是相同的数据类型也要将其放置在单独的一行上，这样有助于添加注释。
- ☑ Java 语句中多个空格看成一个。在 Java 代码中，关键字与关键字间如果有多个空格，这些空

格均被视作一个。例如：

```
public  static  void  main(String args[])  ⟹ 等价于  public static void main(String args[])
```

- ☑ 不要使用技术性很高、难懂、易混淆判断的语句。由于程序的开发与维护不能是同一个人，为了程序日后的维护方便，应尽量使用简单的技术完成程序需要的功能。
- ☑ 对于关键的方法要多加注释。多加注释会增加程序的可读性，有助于阅读者很快地了解代码结构。

3.8　经 典 范 例

3.8.1　经典范例 1：判断某一年是否是闰年

视频讲解：光盘\TM\lx\3\判断某一年是否是闰年.exe

为了弥补人类历法的年度天数和地球公转实际周期的时间差，设立了有 366 天的闰年，闰年的二月份有 29 天。闰年的判定规则是：如果该年能被 4 整除且不能被 100 整除或者能被 400 整除，则该年是闰年，否则不是。运行结果如图 3.28 所示。（**实例位置：光盘\TM\sl\3\10**）

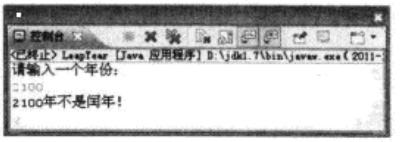

图 3.28　判断是否是闰年

在项目中创建 LeapYear 类，在类的主方法中创建扫描器对象接收输入的年份，然后判断该年份是否是闰年，最后在控制台输出。代码如下：

```
import java.util.Scanner;                                          //声明包
public class LeapYear {                                            //声明类
    public static void main(String[] args) {                      //主方法
        Scanner scan = new Scanner(System.in);                     //扫描器
        System.out.println("请输入一个年份：");
        long year = scan.nextLong();                               //接收用户输入
        if (year % 4 == 0 && year % 100 != 0 || year % 400 == 0) { //判断是否是闰年
            System.out.print(year + "年是闰年！ ");                 //输出是闰年
        } else {
            System.out.print(year + "年不是闰年！ ");               //输出不是闰年
        }
    }
}
```

3.8.2　经典范例 2：求球形的体积

视频讲解：光盘\TM\lx\3\求球形的体积.exe

在数学运算中，经常会计算球形的体积，这也是空间解析几何中必修的。通过计算机计算球形的

体积是很简单的，使用算术运算符即可实现。运行结果如图 3.29 所示。**（实例位置：光盘\TM\sl\3\11）**

图 3.29　计算球形的体积

在项目中创建 Volume 类，在类的主方法中创建扫描器对象接收输入的半径，然后根据球形体积公式计算球形的体积，并在控制台上输出。代码如下：

```
import java.util.Scanner;                                          //声明包
public class Volume {                                              //声明类
    public static void main(String[] args) {                      //主方法
        Scanner scan = new Scanner(System.in);                     //扫描器
        System.out.println("请输入球形的半径：");
        double r = scan.nextDouble();                              //接收数据
        final double PI = 3.1415926;                               //定义常量 PI
        double volume = 4.0/3.0 * PI * r * r * r;                  //求球形的体积
        System.out.println("球形的半径是：" + r);                   //输出球形的半径
        System.out.println("圆周率是：" + PI);                      //输出圆周率
        System.out.println("球形的体积是：" + volume);              //输出球形的体积
    }
}
```

3.9　本 章 小 结

本章向读者介绍的是 Java 语言基础，其中需要读者重点掌握的是 Java 语言的基本数据类型、变量与常量以及运算符等三大知识点。初学者经常会将 String 类型认为是 Java 语言的基本数据类型，在此提醒读者 Java 语言的基本数据类型中并没有 String 类型。另外，对数据类型之间的转换也要有一定的了解。在使用变量时，需要读者注意的是变量的有效范围，否则在使用时会出现编译错误或浪费内存资源。此外，Java 中的各种运算符也是 Java 基础中的重点，正确使用这些运算符，才能得到预期的结果。

3.10　实 战 练 习

1．编写 Java 程序，不使用乘号“*”，而只使用移位运算符计算出 21*16 的值。**（答案位置：光盘\TM\sl\3\12）**

2．编写 Java 程序，声明两个 int 型变量，运用三元运算符判断两个变量是否相等，若不相等，求出两个数中较大的。**（答案位置：光盘\TM\sl\3\13）**

3．编写 Java 程序，声明两个变量并赋值 38.9 和 27.2 作为矩形的长和宽，求出该矩形的面积。**（答案位置：光盘\TM\sl\3\14）**