# 第 3 章　对象的属性——成员变量详述

**学习目标：**

- 掌握 Java 的常用数据类型；
- 掌握变量的声明和使用；
- 掌握 Java 数组的定义和使用；
- 掌握用户输入数据的接收和使用；
- 掌握对象的成员变量的赋值和使用。

对象的成员变量描述了对象的属性，成员变量用来存储数据，而数据有不同的类型。本章围绕对象的成员变量进行展开讨论，详细学习 Java 中的类型、变量、运算符、数组等内容。

学习完本章内容，可以编写出例 3-1 这样的程序。

**例 3-1**　对象中成员变量的灵活定义及使用。

模拟通讯录功能，定义 Person 类保存人员信息，能够对用户输入的姓名、性别、电话号码进行保存，还可以进行分类。Person 类的多个对象就是通讯录中的一条完整信息。

```
import java.util.Scanner;

class Person{

    String name;
    char sex;
    String phoneNumber;
    String category[]={"同学","同事","朋友","家人"};
    int i;

    void setInfo(){
        int n;
        Scanner reader=new Scanner(System.in);
        System.out.print("姓名:");
        name=reader.next();
        System.out.print("性别:");
        sex=reader.next().charAt(0);
```

```
            System.out.print("电话号码:");
            phoneNumber=reader.next();
            do{
                System.out.print("选择分类:1同学 2同事 3朋友 4家人  请选择:");
                i=reader.nextInt()-1;
            }while(i<0||i>3);
        }

        void showInfo(){
            System.out.println(name+"\t"+sex+"\t"+phoneNumber+"\t"+category[i]);
        }

    }

    public class PersonTest {

        public static void main(String[] args) {

            Person p[]=new Person[3];
            for(int i=0;i<3;i++){
                p[i]=new Person();
                System.out.println("------请输入第"+(i+1)+"个人的信息------");
                p[i].setInfo();
            }
            System.out.println("--------------------------------------");
            System.out.println("姓名\t性别\t电话号码\t分类");
            for(int i=0;i<3;i++)
                p[i].showInfo();

        }

    }
```

代码中使用了多种数据类型,还使用了数组、关系运算符、逻辑运算符、循环结构语句,同时实现了输入功能。程序运行结果如图 3-1 所示。


图 3-1　例 3-1 程序运行结果

## 3.1　变量与基本数据类型

对象的成员变量要使用不同类型的数据进行描述，在程序的执行过程中也经常要使用不同类型的变量来存储和访问数据。不同类型的数据在计算机中所占用的内存空间大小不同，能够进行的运算也不同。

在 Java 语言中，定义变量的格式为：

```
变量的类型 变量名称;
变量的类型 变量名称 1,变量名称 2,…;
变量的类型 变量名称=变量的值;
```

变量之间用逗号“,”隔开，语句的最后是一个分号“;”。Java 语言中变量只声明而不赋初值，会自动赋默认值。

Java 允许将变量的声明放在代码中的任何地方，良好的编程习惯是把变量的声明尽可能地放在变量第一次使用之前。

基本数据类型也称作简单数据类型。Java 共有 8 种基本数据类型，包括 4 个整数类型、2 个浮点类型、1 个字符类型及 1 个逻辑类型。

### 3.1.1　整数类型

整数类型用来描述整数，共有 4 种，分为 int 、byte 、short 和 long，如表 3-1 所示。

**表 3-1　整数类型**

| 关键字 | 名称 | 存储需求 | 定义变量举例 |
|---|---|---|---|
| int | 常整型 | 4 字节 | int x = −100; |
| byte | 字节型 | 1 字节 | byte a = 5; |
| short | 短整型 | 2 字节 | short m = 25; |
| long | 长整型 | 8 字节 | long n = −2000000000L; |

关于整数类型的说明：

(1) Java 语言规定，直接写出的整数被认为是 int 类型，如 5，−3 等。

(2) 描述 byte 类型的数据通常要使用强制类型转换，如(byte)10。

(3) 描述 short 类型的数据通常需要使用强制类型转换，如(short)15。

(4) 描述 long 型的整数常量要在数字后加上字母 L 或 l，如 128L。

(5) Java 中还可以描述八进制和十六进制的数。八进制数以 0 开头，十六进制数以 0x 或 0X 开头。如 010 表示八进制的 8，0x10 表示十六进制的 16。

(6) Java 中没有任何无符号 unsigned 类型。

### 3.1.2　浮点类型

浮点类型用来描述实数。实数可用小数表示，如 2.0、78.9 等；也可以用指数表示，如

3.5e8、15.7E－3 等(分别表示 $3.5\times10^{8}$、$15.7\times10^{-3}$，使用字母 E 或 e 均可，e 前必须有数字，e 后必须为整数)。

Java 中的浮点类型分为 float 和 double 两种，如表 3-2 所示。

**表 3-2　浮点类型**

| 关键字 | 名　　称 | 存储需求 | 定义变量举例 |
|---|---|---|---|
| float | 单精度浮点型 | 4 字节 | float x = 3.25F; |
| double | 双精度浮点型 | 8 字节 | double y = 37.4; |

关于浮点类型的说明：

(1) Java 语言规定，直接写出的浮点数被认为是 double 类型，如 37.4，－3.69 等。

(2) 可以通过在数字后加上字母 D 或 d 来表明当前数据是 double 型的实数常量，如 37.4D，－3.69d 等。对于 double 来说，不写后缀也是可以的。

(3) 描述 float 型的浮点数值必须在数字后加上字母 F 或 f，如 2.5F，0.7f。

(4) float 类型变量保留 6～7 位有效数字，double 类型变量保留 15 位有效数字，实际精度取决于具体数值。

### 3.1.3　字符类型

字符类型用来描述单个的字符。字符类型的关键字是 char 。

关于字符类型的说明：

(1) Java 语言中的字符采用 UNICODE 编码，一个字符在内存中占两个字节空间。这使得 Java 可以使用 char 类型描述更多种类的字符，包括英文字母、标点符号、汉字、日文单字、韩文单字等。

(2) 每个字符类型的数据必须用单引号括起来，一个字符类型的变量只能存放一个字符。如语句“char ch1＝'a', ch2＝'*', ch3＝'好';”。

(3) 字符类型变量的内存空间中实际存储的是字符编码，即 char 类型的变量可以与整数类型的变量通用。如语句“char ch＝97;”相当于给变量 ch 存储了字符编码为 97 的小写字母“a”这个字符。

(4) Java 中的字符类型变量可以存储转义字符。

常用转义字符如表 3-3 所示。

**表 3-3　常用转义字符**

| 转义字符 | 名　　称 | 转义字符 | 名　　称 |
|---|---|---|---|
| \n | 换行 | \\ | 反斜线 |
| \t | 制表位 | \' | 单引号 |
| \r | 回车 | \" | 双引号 |

### 3.1.4　逻辑类型

逻辑类型用来描述真与假。逻辑类型的关键字是 boolean ，每个 boolean 类型的变量

在内存中占 1 个字节的空间。boolean 类型数据的常量有 true 和 false，即 boolean 类型的变量只能存储这两个值之一，不能存储其他内容。在 Java 语言中逻辑数据不会转换成其他数值类型的数据。

使用 boolean 定义逻辑类型的变量并赋初值：

```
boolean b1=true, b2=false;
```

### 3.1.5　数据类型的转换

数值类型数据相互之间可以进行转换，类型转换分为自动转换（隐式）和强制转换（显式）。

**1. 自动类型转换**

自动类型的转换通常是在一个运算式中参加运算的各个变量的类型不一致，或者要给某种类型的变量赋一个不同类型的值时发生。自动类型转换由系统自动完成，以出现的最高级别的类型为标准进行转换。各类型间的转换方向如图 3-2 所示。

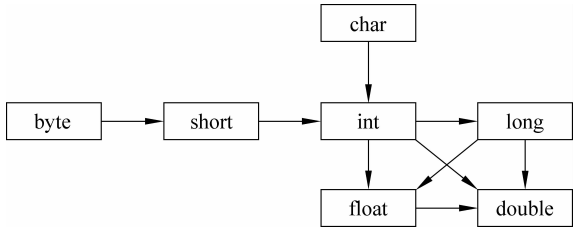


图 3-2　数据类型之间自动转换关系

例如，有混合运算 10＋'a'＋1.5＊3，其最终的结果为 double 类型的 111.5，因为运算量中出现了 1.5，是 double 类型，它的级别最高。

语句 float x＝3；执行后，变量 x 存储的数据是 3.0。

在自动类型转换中存在可能损失精度的转换。

**2. 强制类型转换**

自动类型转换是系统按照类型级别由低向高自动完成的转换，强制类型转换则是要使数据由高级别转向低级别，需要显式地写出来。

将一个数据或表达式强制转换成所需的更低类型，格式为：

**（类型名）要转换的数据**

例如：

```
double x=4.7;
int y=(int)x;  //强制类型转换
```

则变量 y 的值为 4。强制类型转换通过截断小数部分把一个浮点型数据转换为整型。

强制类型转换得到的是一个中间变量，原变量类型并不发生变化。上述语句执行后，

变量 x 的值仍为 4.7。

**注意**：boolean 类型不能与任何数值类型进行类型转换。

**例 3-2** 熟悉 Java 的基本数据类型。

```
public class BasicDataTypes{
    public static void main(String args[]){
        byte b=0x55;            //将十六进制数据赋值给 b
        short s=0x55ff;         //将十六进制数据赋值给 s
        int i=1000000;
        long l=0x10EF;          //将十六进制数据赋值给 l
        char c='*';
        float f=0.23f;
        double d=0.7E-3;        //将科学记数法表示的浮点型数据赋值给 d
        boolean bool=true;
        System.out.println("byte  b="+b);
        System.out.println("short  s="+s);
        System.out.println("int  i="+i);
        System.out.println("long  l="+l);
        System.out.println("char  c="+c);
        System.out.println("float  f="+f);
        System.out.println("double  d="+d);
        System.out.println("boolean bool="+bool);
    }
}
```

程序运行后在屏幕上显示：

```
byte  b=85
short  s=22015
int  i=1000000
long  l=4335
char  c=*
float  f=0.23
double  d=7.0E-4
boolean bool=true
```

上面的程序使用不同的基本数据类型分别定义了变量，并进行了赋值，然后输出了各变量的值。试着修改变量的值，如去掉 folat 类型数据的后缀 f 或 long 类型数据的 l，观察一下会出现什么情况。

**例 3-3** 熟悉 Java 中各类型的转换。

```
public class Conversion{
    public static void main(String args[]) {
        byte b;
        int i=257;
```

```
        double d=323.567;
        char ch;
        short s;
        b=(byte)i;        //将整型变量 i 的数据强制类型转换为 byte 类型赋值给 b
        System.out.println("int→byte 例:int 型"+i+"→byte 型"+b);
        i=(int)d;         //将双精度类型变量 d 的数据强制类型转换为 int 型赋值给 i
        System.out.println("double→int 例:double 型"+d+"→int 型"+i);
        b=(byte)d;        //将双精度类型变量 d 的数据强制类型转换为 byte 型赋值给 b
        System.out.println("double→byte 例:double 型"+d+"→int 型"+b);
        ch=(char)64;      //将 int 型数据 64 强制类型转换为 char 类型赋值给 ch
        System.out.println("64→char 例:int 型 64→char 型"+ch);
        s=(short)4000;  //将 int 型数据 4000 强制类型转换为 short 型赋值给 s
        System.out.println("4000→short 例:int 型 4000→short 型"+s);
    }
}
```

程序运行后在屏幕上显示：

```
int→byte 例: int 型 257→byte 型 1
double→int 例: double 型 323.567→int 型 323
double→byte 例: double 型 323.567→int 型 67
64→char 例: int 型 64→char 型@
4000→short 例: int 型 4000→short 型 4000
```

## 3.2　常　　量

在程序中其值不能被改变的量叫做常量，最常见的就是那些直接使用的数据。Java 共有 5 种类型的常量。

(1) 整型常量，如 12，12L；

(2) 浮点型常量，如 12.5F，3.15，12.0E2；

(3) 布尔型常量，如 true，false；

(4) 字符型常量，如'a'，'9'；

(5) 字符串常量，如"a"，"wonderful"，"你好"。

可以在程序中自己定义常量。自定义常量可以理解成是用文字代表的常量，是变量的特殊形式。自定义常量用关键字 final 修饰，要在定义时赋值，常量定义之后就不能再改变它的值。

例如，定义一个字符常量 ch 代表字符'#'，可写为：

```
final public char ch='#';
```

定义 double 类型常量 d 代表常数 1.234，可写为：

```
final double d=1.234;
```

整型常量可以用八进制、十进制、十六进制的数来赋值。

Java 另有 4 个系统定义的常量：NaN(非数值)、Inf(无穷大)、－Inf(负无穷大)、Null(空)。

## 3.3 字符串类型

在 Java 中可以很方便地对字符串进行操作，Java 提供了字符串类型 String ，实际上 String 是一个类，它不属于基本数据类型，但字符串被使用得实在太频繁了，所以 Java 针对它提供了更方便的使用方式。使用 String 声明字符串就像使用基本数据类型声明变量那样简单。字符串变量(确切地说应该是对象，因为 String 其实是一个类，但这里简单地声明为变量)可以像基本类型变量那样被赋值、访问。

例如，语句：

```
String s1;
String s2="Hello World!";
```

声明了一个字符串变量 s1，一个字符串变量 s2，并给 s2 赋初值为“Hello World!”。

**例 3-4** 字符串的方便使用。

```
public class StringTest{
    public static void main(String[] args){
        String s1="你好啊~!", s2="hello~^_^";
        System.out.println("字符串 s1 的内容是:"+s1);
        System.out.println("字符串 s2 的内容是:"+s2);
        s1=s2;
        System.out.println("将 s2 赋值给 s1 后,");
        System.out.println("字符串 s1 的内容是:"+s1);
        System.out.println("字符串 s2 的内容是:"+s2);
    }
}
```

程序运行结果如图 3-3 所示。


图 3-3 例 3-4 程序运行结果

字符串作为类使用的相关内容将在第 6 章详细学习。

## 3.4　运　算　符

Java 提供了丰富的运算符，如表 3-4 所示。

**表 3-4　Java 中的运算符**

| 各种运算符 | 说　明 |
|---|---|
| 算术运算符 | 双目运算符：+，－，*，/，% |
| | 单目运算符：++，－－，+，－ |
| 关系运算符 | ==，!=，<，>，<=，>= |
| 逻辑运算符 | !，^，&，\|，&&，\|\| |
| 赋值运算符 | = |
| 位运算符 | ~，&，\|，^，<<，>>，>>> |
| 对象运算符 | instancesof |
| 条件赋值运算符 | A =（条件？b:c），条件为真时 b 赋给 A，反之为 c |
| 广义赋值运算符 | +=，－=，*=，/=，%=，^=，&=，\|=，<<=，>>=，>>>= |
| 括号与方括号运算符 | ()，[] |
| 运算符的优先级 | .，[]，()的优先级最高，>>>=的优先级最低 |

本节将主要学习算术运算符、关系运算符、逻辑运算符与赋值运算符及相关的表达式。

### 3.4.1　算术运算符

Java 中的算术运算符有加、减、乘、除和求模 5 种（+、－、*、/、%）。使用算术运算符将数据连接起来就构成了算术表达式，如 3+4，5*a%7 等。

算术运算符的使用如表 3-5 所示。

**表 3-5　基本算术运算符的使用**

| 运算符 | 说　明 |
|---|---|
| + | 加法运算符，或正值运算符，如 3+5，+3 |
| － | 减法运算符，或负值运算符，如 5－2，－3 |
| * | 乘法运算符，如 3*5 |
| / | 除法运算符，如 5/3 |
| % | 模运算符（求余运算符），可对小数操作，如 7%4==3，8.5%3==2.5 |

算术运算符的使用注意事项：

（1）两个整数相除结果仍为整数，小数部分将被直接舍去。例如，1/2 的结果是 0，不是 0.5。所以在编写程序时对整数相除要特别加以注意，避免产生的误差影响到整个计

算结果。

(2) 参加混合运算的两个数中只要有一个是实数,则结果为 double 型。

(3) 乘、除、模运算符优先级相同,高于加、减运算符的优先级。进行混合运算时,先乘、除、模,后加、减。相同优先级的算术运算符连续出现时,从左向右依次计算即可。

(4) Java 语言重载了加法运算符+,可以对字符串进行+的运算,其作用是将字符串进行连接。将数值型数据与字符串进行加法运算时,系统会自动将数值型数据转换为字符串,最终得到一个连接后的完整字符串。例如,"abc"+123 的运行结果是字符串"abc123","abc"+(1+2)的运行结果是字符串"abc3"。

**例 3-5** 不同类型数据间的混合算术运算。

```
public class MixTest{
    public static void main(String[] args){
        int iNum1=10;
        double fNum2=25.5;
        char chNum3='a';
        String str1="结果是";
        System.out.println(iNum1+"%6"+"="+iNum1%6);
        System.out.println(iNum1+"/6"+"="+iNum1/6);
        System.out.println(fNum2+"%"+iNum1+"="+fNum2%iNum1);
        System.out.println(iNum1+"*"+fNum2+"%125"+"+"+chNum3+"="+
            (iNum1*fNum2%125+chNum3));
        System.out.println(iNum1+"+"+fNum2+str1+(iNum1+fNum2));
    }
}
```

程序运行结果如图 3-4 所示。


图 3-4 例 3-5 程序运行结果

**注意**:加法运算从左至右依次运算,中途一旦出现字符串,则后面的加法运算含义全部变为字符串连接。如果需要先计算再以字符串显示结果,则应该用括号()把运算括起来,就像本例中最后一个语句所做的那样。

### 3.4.2 自增、自减运算符

自增运算符(++)、自减运算符(--)的功能是使与它相临的变量的值增 1、减 1,可