



第3章 方法

第3章 方法

- ◆ 定义方法
- ◆ 调用方法
- ◆ 方法实例
- ◆ 方法应用
- ◆ 方法重载

方法概述

- JAVA中有两种模块：**方法和类**。借助**Java API**和其他各种类库中可用的、“**预先打好包**”的**方法和类**，我们可以编写出新的方法和类。
- 在程序设计时，**可以将一个程序中完成特定功能的程序定义成方法**（有时在其他编程语言中称为函数或过程）。在需要使用这些功能时，**可调用相应的方法**，特别是在**某些功能多次被使用时**，采用方法可大大地提高程序代码的可重用性。

方法概述

- “方法”的目的：
 - 一是“分而治之”；
 - 二是软件的重用；
 - 三是避免在程序中重复代码。
- 常用的Math类的方法和String类的方法

Math.sqrt()//计算平方根

Math.cbrt()//计算立方根

Math.hypot(x,y)//计算 (x的平方+y的平方)的平方根

3.1 定义方法

方法包括：方法的声明以及方法体，语法如下：

```
返回值类型 方法名 (参数列表) {
```

```
    方法体
```

```
}
```

方法的声明

- 使用打印三个矩形的实例说明方法的声明
- 方法声明的一般格式为：

[修饰符1 修饰符2]返回值类型 函数名 (参数类型 形式参数1, 参数类型 形式参数2,)

{

程序代码

[return 返回值]

}

方法头

方法体

补充：声明方法时使用的参数称为**形式参数**（**形参**），调用方法时使用的参数称为**实际参数**（**实参**）。

方法的声明

■ 声明方法应注意:

- 在类体的“{ }”外定义方法，是语法错误
- 方法名可以是任何合法的标识符
- 返回值类型可以是任何基本数据类型、String 和 void(表示方法不返回值)，且返回值类型是不可省略的
- 如果指定的返回值类型不是 void，则方法中必须包含一条 return 语句，返回指定类型的值，同样在返回值类型为 void 的方法中返回一个值也是一种语法错误

方法的声明

■ 声明方法应注意（接上页）

- 方法头 “ () ” 中的形参用逗号分隔，且用 `add(int a,b)` 代替 `add(int a,int b)` 是一种错误
- 方法头 “ () ” 的右端不能放置 “ ; ”
- 在方法体中将形式参数重新声明是一种语法错误

方法的声明

- 方法按是否有参数可分为有参方法和无参方法：

例 定义计算平方的方法（有参方法）

```
static int square(int x)
```

```
{
```

```
    int s;
```

```
    s=x*x;
```

```
    return (s);
```

```
}
```

方法的声明

- 以下程序是否正确?
- **static void result(int x, int y);**
{
 int s;
 int x,y;
 s=x*y;
 return (s);
}

方法的声明

- 以下程序是否正确?
- `static void result(int x, int y);`
 {
 int s;
 int x,y;
 s=x*y;
 return (s);
 }

方法的声明

■ 例 求最大值的方法

```
int max3(int x,int y,int z)
{   int big;
    if(x>y)
        big=x;
    else
        big=y;
    if(big<z)   big=z;
    return(big);
}
```

方法的声明

■ 例 无参方法示例

```
public void my_print( )  
{  
    System.out.println("*");  
    System.out.println("* *");  
    System.out.println("* * *");  
    System.out.println("* * * *");  
}
```

方法的声明

- 按照是否有**static**修饰符，方法分为**类方法**(又称静态方法)和**对象方法**（又称实例方法）

例：类方法是有**static**修饰符的方法。

```
public static int max_method(int x,int y ){  
    int max;  
    if(x>max)  
        max=x;  
    else  
        max=y;  
    return max;  
}
```

3.2 调用方法

Java中，除了**main**方法是系统自动调用外，**其他方法必须明确被调用**。通过方法名和参数列表调用方法，形式如下：

方法名（**实际参数表**）；

3.2 调用方法

1、类方法调用

- 类方法可以**直接调用或通过类名.方法名([实参])**调用，不需要生成对象。
- 对于有返回值的类方法作为表达式或表达式的一部分来调用，其在表达式中出现的形式为：
方法名 ([实际参数表])

例 调用类方法

```
public class SquareC{  
    static int square(int x) { //类方法  
        int s;  
        s=x*x;  
        return (s);  
    }  
    public static void main(String[] args){  
        int n = 5;  
        int result;  
        result= square(n);//类方法调用  
        System.out.println(result);  
    }  
}
```

3.2 调用方法

对于没有返回值的类方法，以独立语句的方式调用方法。
例：

```
class AreaC{
    static void area(int a , int b ){
        int s;
        s = a * b;
        System.out.println(s);
    }
    public static void main(String[] args){
        int x = 5;
        int y=3;
        area(x, y); //方法语句，直接调用类方法
    }
}
```

3.2 调用方法

例 无参类方法调用

```
class SumC{  
    static void sum( ){  
        int i, j, s;  
        i=3;  
        j=6;  
        s=i+j;  
        System.out.println(s);  
    }  
    public static void main(String[] args){  
        sum( );  
    }  
}
```

3.2 调用方法

下列程序是否正确?

```
class SumC{  
    static void sum(int i, int j){  
        int s;  
        i=3;  
        j=6;  
        s=i+j;  
        return(s);  
    }  
    public static void main(String[] args){  
        int ss;  
        ss = sum(4,5);  
        System.out.println(ss);  
    }  
}
```

2、对象方法调用

对于对象方法，**必须先生成对象实例**，然后用如下形式调用方法：

对象名.方法名([实参])

```
class MyObject{
    static int x=10;
    static int y=20;

    void method(int k,int m){
        int sum;
        sum=k+m;
        System.out.println("sum="+sum);
    }
    public static void main(String args[]){
        MyObject obj=new MyObject(); //生成对象
        obj.method(x,y);           //调用对象方法
    }
}
```

//对象方法（实例方法）调用示例：

class **Car**

{

int speed; //变量定义部分。

float weight=12f,height=2f; //变量定义部分。

void changSpeed(int newSpeed) //对象方法定义。

{ speed=newSpeed;

}

float getWeight() //对象方法定义。

{ return weight;

}

float getHeight() //对象方法定义。

{ return height;

}

public static void **main**(String args[])

{ **Car** myCar=new **Car**();

myCar.changSpeed(120);

System.out.println("车速度为:"+myCar.speed);

System.out.println("车的重量为: "+myCar.getWeight());

System.out.println(" 高度为: "+myCar.getHeight());

}

}

//对象方法（实例方法）调用示例：注意x,y的值是否变化？文件名如何命名？

```
class Tom{
    void f(int x,double y)
    {
        x=x+1;
        y=y+1;
        System.out.println("参数x和y的值分别是:"+x+", "+y);
    }
}

public class Example{
    public static void main(String args[])
    {
        int x=10;
        double y=12.58;
        Tom tom=new Tom();
        tom.f(x,y);
        System.out.println("main方法中x和y的值仍然分别是:"+x+", "+y);
    }
}
```

【例3.1】方法声明和调用实例

//MyMethod.java

import java.util.*;

class MyMethod {

 //方法定义

 static void **displayMessage()** {

 // 方法体

 System.out.println("请注意保护个人密码。");

 System.out.println("请输入取款密码.");

 }

 public static void main(String[] args) {

 Scanner sc= new Scanner(System.in);

 String myPSW;

 // 调用displayMessage () 方法

displayMessage();

 myPSW = sc.next();

 }

}

3.3 方法实例

【例3.2】 闰年判断方法

如何判断是否为**闰年**？

- ◆ 能被 4 整除的**大多是闰年** (yearIn \% 4 == 0)
- ◆ 能被 100 整除而不能被 400 整除的年份**不是闰年** ($\text{yearIn \% 100 == 0} \ \&\& \ (\text{yearIn \% 400} \neq 0)$)
- ◆ 不能被 100 整除或能被 400 整除的年份是**闰年** ($(\text{yearIn \% 100} \neq 0) \ || \ (\text{yearIn \% 400} == 0)$)

3.3 方法实例

【例3.2】 闰年判断方法

```
static boolean isLeapYear ( int yearIn ){  
    boolean resultIn=false;  
    if((yearIn % 4 ==0)&& (yearIn % 100 !=0)  
        ||(yearIn % 400 == 0))  
        resultIn = true;  
    return resultIn;  
}
```

【例3.3】带参数和返回值的方法

```
//LeapYear.java
```

```
import java.util.*;
```

```
public class LeapYear {
```

```
    static boolean isLeapYear(int yearIn){
```

```
        boolean resultIn = false;
```

```
        if((yearIn % 4 ==0) && (yearIn % 100 !=0) || (yearIn % 400 == 0)){
```

```
            resultIn = true;
```

```
        return resultIn;
```

```
    }
```

```
    public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);
```

```
        int year;
```

```
        boolean result;
```

```
        year = sc.nextInt();    // 调用isLeapYear方法，传入实际参数year.
```

```
        result = isLeapYear(year);
```

```
        System.out.println(year + " is Leap Year? " + result);
```

```
    }
```

```
}
```

return语句

return语句有两个功能：

- ◆ 方法的结束语句
- ◆ 可以携带参数，将参数返回给调用者

方法的结束条件：

- ◆ 运行完方法体的最后一条语句
- ◆ 执行到return语句

return语句有两种形式：

- ◆ `return;`
- ◆ `return 表达式; 或 return(表达式);`

3.4 方法应用

在定义方法的过程中，一般要考虑四个问题：

- ◆ 方法名
- ◆ 方法的输入（即形参）
- ◆ 方法的输出（即返回值）
- ◆ 方法体（完成相应功能的语句）

方法名的定义要符合标识符规则，尽量做到见名知义。如果方法名由多个单词组成，一般第一个单词全部小写，后续单词的首字母要大写。

例如myMethod, isEven。

参数的传递

参数传递要点：虚实结合 “三一致” 原则即实参与虚参个数、类型、顺序一致。

例4-5 参数传递实例

```
public class Swaping{
    static void swap(int x , int y ){
        int temp ;
        temp = x; x = y; y = temp;
    }
    public static void main(String[] args){
        int u=23 , v=100;
        System.out.println("Before Calling");
        System.out.println("u= "+u+" v= "+v);
        swap(u, v);
        System.out.println("After Calling");
        System.out.println("u= "+u+" v= "+v);
    }
}
```

```
class Lader{
    double above,bottom,height;
    public void setAbove(double a){
        above=a;
    }
    public void setBottom(double b){
        bottom=b;
    }
    public void setHeight(double h){
        height=h;
    }
    double computeArea(){
        return (above+bottom)*height/2.0;
    }
}
```

```
public class Example
{
    public static void main(String
    args[]){
        double area=0;
        Lader laderOne=new Lader();
        laderOne.setAbove(10);
        laderOne.setBottom(40);
        laderOne.setHeight(2);
        area=laderOne.computeArea();
        System.out.println("laderOne
        的面积是:"+area);
    }
}
```

计算梯形的面积

1. 梯形的面积计算公式?

2. 有多少个参数?

3. 如何实现代码?

- 定义类
- 定义方法
- 调用方法

3.4.1 单个参数的方法

【例3.4】计算圆的面积

【编程思路】计算圆的面积，方法命名为 **circleArea**。要完成计算，必须知道圆的半径，因此方法需要一个输入参数：**半径**。计算完毕，需要把计算结果返回给调用方法 **main**，因此也需要一个返回值。计算面积的算法是： **$\pi * r^2$** ，方法体内按照公式计算得到结果。**用方法实现计算圆的面积：**

//MethodExp3.java

public class MethodExp3 {

static double circleArea(double radiusIn){

final double PI = 3.14; // 定义常量PI

//根据传入参数计算圆的面积，并将结果返回

return (PI * radiusIn * radiusIn);

}

public static void main(String[] args) {

double result=0.0;// 调用circleArea方法，

// 计算半径数值为2.4的圆面积

result = circleArea(2.4);

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

// 调用circleArea方法，计算半径数值为4.0的圆面积。

System.out.println(**circleArea(4.0)**);

}

}

3.4.2 多个参数的方法

【例4.5】 实现由任意符号组成的 $i * j$ 的矩形。

【编程思路】 方法的功能是绘制矩形，方法命名为drawRectangle。绘制时，要知道绘制符号，以及 $i * j$ 的值。因此，输入参数应该有3个，分别为绘制符号、矩形宽、矩形高。方法体实现矩形的输出显示，不需要返回结果，因此返回值类型是void。方法体内绘制算法，回顾第三章介绍过的for循环的实例，它实现了有 10×4 个“*”组成的矩形：

3.4.2 多个参数的方法

//外部循环，实现打印出多行。

```
for( i=0; i <4; i++ ){
```

```
    for( int j =0; j<9; j++){
```

```
        //内部循环实现打印出一行 “*”
```

```
        System.out.print(“*”);
```

```
    }
```

```
    System.out.println();    //每行结束后，换行
```

```
}
```

3.4.3 递归方法

递归的思想就是：

把问题分解成为**规模更小的**、具有**与原问题有着相同解法**的问题。

既然递归的思想是把问题分解成为规模更小且与原问题有着相同解法的问题，那么是不是这样的问题都能用递归来解决呢？

答案是否定的。并不是所有问题都能用递归来解决。那么什么样的问题可以用递归来解决呢？

3.4.3 递归方法

一般来讲，能用递归来解决的问题必须满足两个条件：

- ◆ 可以通过递归调用来缩小问题规模，且新问题与原问题有着相同的形式。
- ◆ 存在一种简单情境，可以使递归在简单情境下退出。

如果一个问题不满足以上两个条件，那么它就不能用递归来解决。

方法的递归调用有两种方式，一种是**直接递归**，即方法直接调用自己；另一种是**间接递归**，即方法调用其他方法，而其他方法又再调用这个方法。

【例3.6】求斐波那契 (Fibonacci) 数列第n项的方法fibo(n)。斐波那契数列为0、1、1、2、3、5、8、13、21……，即从第3项开始，每一项为前两项之和。

【编程思路】经过分析，斐波那契数列可表示为：

$\text{fibo}(0)=0;$

$\text{fibo}(1)=1;$

$\text{fibo}(n)=\text{fibo}(n-1)+\text{fibo}(n-2) \quad (n \geq 2)。$

也就是说，当形参n接收到0或1时，不再递归，而是将n返回。当形参接收的数大于等于2时将递归。

//MethodExp5.java

public class MethodExp5 {

static int **fibonacci**(int n){

if(n == 0 || n == 1)

//递归终止条件

return 1;

else

return fibonacci(n-1) + fibonacci(n-2);

//递归调用

}

public static void main(String[] args){

int n = 8;

//调用递归方法

System.out.println("fibonacci " + n + " =" + **fibonacci**(n));

}

}

//MethodExp5.java

public class MethodExp5 {

static int fibo(int n){

if(n == 0 || n == 1)

//递归终止条件

return 1;

else

return fibo(n-1) + fibo(n-2); //递归调用

}

public static void main(String[] args){

int n = 8;

//调用递归方法

System.out.println("fibo " + n + " =" + fibo(n));

}

}

运行结果: fibo 8=34

3.4.4 多个返回值的方法

前面介绍的方法返回值都只有一个。

方法中虽然可以有多个return语句，但程序只要运行到**第一个return语句**，**方法就结束**。因此，看似方法只能有一个返回值。

那么，方法是否可以有多个返回值呢？

方法可以有多个返回值。多个返回值的实现要借助含有多个数据项的数据类型（例如**数组**、**类**）**作为返回类型**。

3.5 方法重载

方法的重载 (Method overloading) 是指一个类中，允许存在一个以上的同名方法，只要它们的参数个数或者类型不同。

【例3.7】 add方法重载

```
//MethodExp5.java
public class MethodExp5 {
    static int add(int x, int y) {
        return x + y; // 两个整数相加
    }
    static int add(int x, int y, int z){
        return x + y + z; // 三个整数相加
    }
    static double add(double x, double y) {
        return x + y; // 两个双精度数相加
    }
    public static void main(String[] args) {
        int isum;
        double fsum;
        isum = add(3,5); //调用 int add(int x, int y) 方法
        isum = add(1,10, 3); //调用 int add(int x, int y, int z) 方法
        fsum = add(7.6, 8.5); //调用double add(double x, double y)方法
    }
}
```

习题

1. 判断下列说法是否正确

- (1) 方法的返回值只能返回一个值。
- (2) 方法的参数允许有多个。
- (3) 方法执行到return语句时，将立即退出。
- (4) 方法的形参和实参，名称必须相同。


习题

2. 语句

int (Math.random () * 6) + 1

的作用是什么（ ）。

- A 产生1 — 6之间的随机数
- B 产生100 — 600之间的随机数
- C 产生10 — 60之间的随机数
- D 产生1000 — 6000之间的随机数



3. 下面的方法头哪一个有效？如果无效，请解释原因。

(1) **public static one(int a, int b)**

(2) **public static int thisone(char x)**

(3) **public static char another(int a, b)**

(4) **public static double yetanother**

4. 如下方法，调用 **stery (5)** 时返回值是 ()。

```
public static int stery (int n) {  
    if (n == 0)  
        return 1;  
    else  
        return 3 * stery( n - 1);  
}
```

A.0

B.3

C.81

D.243

E.6561

5. 下面的程序段中，请指出方法头、方法体、形参、实参、方法调用和局部变量。

```
public class Exercise{  
    public static void main(String [] args){  
        int x=0;  
        double y=0;  
        char z="";  
        hello(x,y,z);  
        .....  
        hello( x + 2, y-2.4),'S');  
    }  
    public static void hello(int first, double second, char ch){  
        int num;  
        double y;  
        .....  
    }  
}
```

编程练习

1. (1) 编写提供三个选项的菜单驱动程序。

第一选项，当用户输入摄氏温度时显示出华氏温度；

第二选项，当用户输入华氏温度时显示出摄氏温度；

第三选项，用户退出。

用到的公式（C代表摄氏温度，F代表华氏温度）

$$F = 9C/5 + 32$$

$$C = 5(F - 32)/9$$

(2) 为了使程序正常运行，用户输入的温度不能低于绝对0度，也就是-273.15C，或者-459.67F.

2. 编写方法reverseDigit，将一个整数作为参数，并反向返回该数字。例如reverseDigit(123)的值是321.同时编写程序测试此方法。



3. 编写程序，一系列数的规律如下：1、1、2、3、5、8、13、21、34.....

求这列数的第30位数是多少。

4. 编写方法distance，计算笛卡尔平面中两点 $(x1,y1)$ 和 $(x2,y2)$ 间的距离。此方法将表示平面上两个点的四个数作为它的参数，并返回两点间的距离。

5. 编写程序，随机生产100个0到9之间的数，统计0到9之间每个数出现的概率。

习题1

1.面对对象的程序设计语言的特点是什么？

答：封装性、继承性和多态。

2.面向对象和面向过程的不同有哪些？

面向对象是一种新的程序设计范型，其基本思想是使用对象、类、继承、封装、消息等基本概念来进行程序设计。

面向过程的程序中数据和处理数据的过程分别存储于不同的地方，数据和过程之间没有逻辑或组织上的联系。

3.Java语言的特点有哪些？

(1) 跨平台 (2) 可移植性 (3) 简单 (4) 面向对象 (5) 分布式 (6) 解释性 (7) 健壮性 (8) 安全性 (9) 体系结构中立 (10) 高性能 (11) 多线程 (12) 动态性

4.开发与运行Java程序需要经过哪些主要步骤和过程？用到哪些工具？

下载、安装JDK、配置环境变量、安装开发工具软件。文字编辑辅助工具开始，例如UltraEdit（<http://www.ultraedit.com>）或EditPlus（<http://www.editplus.com>）。如使用Linux操作系统，可使用vi来编辑Java程序。

从开发效率的角度来看，选择一个好的集成开发环境（JCreator（<http://www.jcreator.com>）或BlueJ（<http://www.bluej.org/index.html>）。功能更齐全的IDE，JBuilder（<http://www.borland.com>）、NetBeans（<http://www.netbeans.org>）等都是功能齐全的IDE，免费使用而功能齐全的IDE，则建议使用Eclipse（<http://www.eclipse.org>）。

5.Java源程序编译成功后能得到什么文件？为什么一个Java文件编译后可以获得多个字节码文件？

Java源程序编译生成的字节码文件可以在任意机器平台上运行，前提是该机器上安装了相应的Java虚拟机（Java Virtual Machine, JVM）。

Java源程序经过编译后生成字节码文件，字节码是一种虚拟的机器指令代码，不针对特定的机器。运行时，Java解释器负责将字节码解释成本地机器指令代码，Java解释器包含在Java虚拟机中。

为了解决在不同平台间运行程序的问题，Java 的程序在进行编译时，并不直接编译为与平台相对应的原始机器语言，而是编译为与系统无关的“字节码”（Byte codes）。

6.什么是Java虚拟机？什么是Java平台？它包括几个部分？
JVM 等于是为Java程序所虚拟的环境。

Java平台划分成J2EE、J2SE、J2ME共3个平台，针对不同的市场目标和设备进行定位。

7.简述Java程序的运行机制？

Java虚拟机(Java Virtual Machine): Java虚拟机可以理解成个以字节码为机器指令的CPU;对于不同的运行平台，有不同的虚拟机；Java虚拟机机制屏蔽了底层运行平台的差别真正实现了“次编译，随处运行”。

Java垃圾回收 (Garbage Collection) : 不用使用的内存空间应该回收；在ac++等语言中，由程序员负责回收无用的内存；Java语言消除了程序员回收无用内存的职责，它提供一种系统级线程跟踪存贮空间的分配情况，并在JVM空闲的时候，检查并释放那些可以被释放的内存空间；垃圾收集在java程序运行过程中自动进行，程序员无法控制和干预。

8.编写一个Java应用程序，使其运行后能够在屏幕上显示如下图案：

*

```
publicclass print {  
    publicstaticvoid main(String args[]){  
        for(int i=0;i<5;i++){  
            for(int k=0;k<2*i+1;k++)  
                System.out.print ("*");  
            System.out.print ("\n");  
        }  
    }  
}
```


习题3

1.判断以下语句是否有误，如果有错，请指出。

(1) `if (a > b) then c=0;`

(2) `if a> b { c=0;}`

(3) `if (a= b) c=0;`

(4) `if (a >b) c=0 else b=0;`

答：

(1) F, `if (a > b){ c=0;}`

(2) F, `if (a > b){ c=0;}`

(3) T

(4) F, `if (a >b) c=0; else b=0;`

2.给出下列语句执行后j 的值是多少。假设i和 j都是整数。

(1) for (i=0, j =0; i<10; i++) j +=i;

(2) for (i=0, j =1; i<10; i++) j +=j;

(3) for (j =0; j<10; j++) j +=j;

(4) for (i=0, j =0; i<10; i++) j +=j++;

答：

(1) j=45

(2) j=1024

(3) j=15

(4) j=0

3.给出m 和n 的运行结果。

```
int n=12345;  
int m=0;  
while (n !=0){  
m=(10 * m) +(n % 10);  
n = n /10;  
}
```

答：

m=54321,n=0

4. 下列语句执行后，k的值是多少。

```
int i=4,j=5,k=9,m=5;
```

```
if(i>j||m<k)
```

```
    k--;
```

```
else
```

```
    k++;
```

k=8.

5.下面程序的输出结果是什么。

```
public class Mystery {  
    public static void main (String[]args ){  
        int y,x = 1,total =0;  
        while (x<=10) {  
            y = x*x;  
            System.out.println(y);  
            total+=y;  
            System.out.println (" Total is"+total );  
        }  
    }  
}
```

5.下面程序的输出结果是什么。

答：无限循环，Total加1。

```
{  
    1  
    Total is 11  
    Total is2  
    1  
    Total is3  
    .....  
}
```