第 5 章

方法的定义和调用

单元概述

方法是程序的重要组成部分，在面向对象的程序设计语言中，方法是类的组成部分，用来实现程序功能的，通过本章的学习应该能够了解 Java 语言中方法编写的语法，掌握方法的调用和重载。

教学重点与难点

重点：

(1)方法的定义

(2)方法的调用

(3)方法的重载

难点：

(1)有参数的方法

(2)方法的重载

在前几章的学习中，已经应用了一些方法，例如 main 方法和 System. out. print()方法。方法是为了执行一个操作而组合在一起的语句组，方法要有名字。

5.1　方法的定义

方法是程序设计语言中重要的组成部分，可以说程序的功能都是通过方法和方法间的调用来实现的，在面向对象程序设计中，类的方法用来描述类的动作或活动，比如图书的增加、修改或删除。下面我们通过一个例子来看一下不使用方法的缺点。

例 5.1　打印多个矩形——不使用方法的编写方式

```
class Func_sample1 {
  public static void main(String args[]) {
    /*
     * 打印一个 3 行,3 列的矩形
     */
    for (int row = 0; row < 3; row++) {
      for (int col = 0; col < 3; col++) {
        System.out.print("*");
      }
      System.out.println();
    }
    System.out.println();
    /*
     * 打印一个 5 行,6 列的矩形
     */
    for (inti = 0; i< 5; i++) {
      for (int j = 0; j < 6; j++) {
        System.out.print("*");
      }
      System.out.println();
    }
    System.out.println();
    /*
     * 打印一个 7 行,8 列的矩形
     */
    for (inti = 0; i< 7; i++) {
      for (int j = 0; j < 8; j++) {
        System.out.print("*");
      }
      System.out.println();
    }
  }
}
```

运行结果如图 5.1 所示。

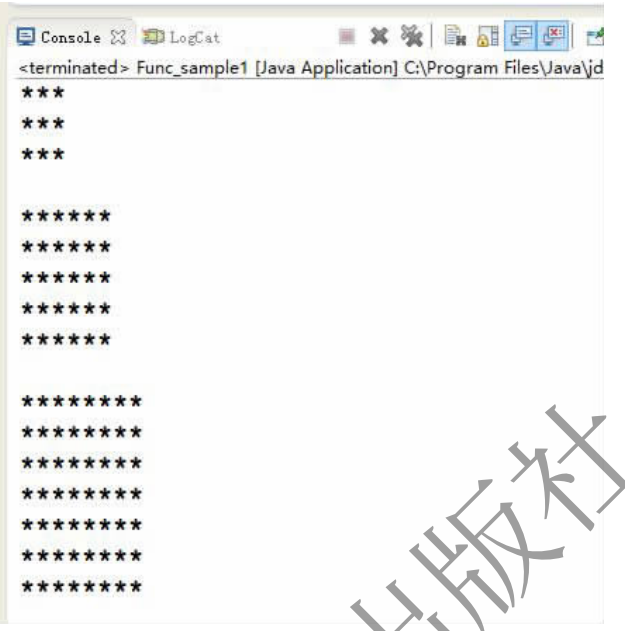

图 5.1　Func_sample1 的运行结果

上面的例子是打印多个矩形，可以看出打印不同行数和列数的矩形基本语句是相同的，只有行数和列数变量的取值不同，而代码却写了多遍，不够简化，那么有什么办法可以简化程序的编写呢，方法就可以解决这样的问题，下面我们来学习方法的定义。

定义方法就是编写一段有特定功能的代码，在程序中使用同样功能的地方，没有必要重复编写同样的代码，只要调用定义好的方法就可以，可以实现代码的重用。简化了程序的编写和维护工作。

方法声明的语法结构为：

```
[访问控制符][修饰符]返回值类型 方法名( 参数类型 形式参数,参数类型 形式参数,…)
{
        方法体
}
```

首先，我们来看例 5.2 的方法。

例 5.2　求两个数的最大值

```
public static intgetBigger(int num1, int num2){
    if (num1 > num2)
        return num1;
    else
        return num2;
}
```

这个方法的功能为求两个参数 num1 和 num2 的最大值，修饰符包含两个，分别为 public 和 static，返回值类型为 int，方法名为 getBigger，包含两个参数，都是 int 类型的，方法体为一组 if-else 语句。

下面我们对方法的结构进行详细介绍：

(1)方法头指定方法的修饰符、返回值类型、方法名和参数。

(2)修饰符：修饰符是可选的，它指定了方法的属性并且告诉编译器该方法可以如何调用，public、static 被称为修饰符(后续会详细讲解它们)。

(3)返回值类型：用来说明该方法运算结果的类型，也就是方法要返回的结果的数据类型。如果返回其他类型，编译就可能出错，方法可以返回一个值。返回值类型是方法要返回的值的数据类型，例如可以是 int，这时要用到 return 语句，例如 return num2。若方法不返回值，则返回值类型为关键字 void。除构造方法外，所有的方法都要求有返回值类型。方法在执行完毕后返回给调用它的程序的数据。

(4)方法名：它作为调用时引用方法的标识。

(5)参数列表：方法可以有一个参数列表，按方法的规范称为形式参数。当方法被调用时，形式参数用变量或数据替换，这些变量或数据称为实际参数。参数是可选的，方法的参数个数可以是 0 个到多个，每个参数前面要声明参数的数据类型；每个参数要用逗号分开。也可以一个参数都没有。

(6)方法体：它是一个语句块，执行特定的功能操作。对于有返回值类型的方法，方法体当中最后一个语句是 return 关键字，它的作用是把方法的执行(运算)结果返回到方法外部。

(7)return 表达式：这里，进一步分析，return 后面的表达式就是方法的返回值。需要注意表达式的类型，必须与方法头中声明的“返回类型”相匹配。

(8)形式参数：在方法被调用时用于接受外部传入的变量。

掌握了方法的基本语法，前面没有使用方法的例题就可以写成如下的方式。

例 5.3　打印多个矩形——使用方法的编写方式

```
class Func_sample2 {
  /*
   * 打印一个 n 行，m 列的矩形
   */
  public static void printRectangle(int row,int col){
    for (int i = 0; i< row; i++) {
      for (int j = 0; j < col; j++) {
        System.out.print("*");
      }
      System.out.println();
    }
  }
  public static void main(String args[]) {
    printRectangle(3,3);
    printRectangle(5, 6);
    printRectangle(6, 7);
    }
}
```

运行结果如图 5.2 所示。

图 5.2　Func_sample2 的运行结果

由上面的例题可见，打印矩形的代码只写了一遍，用一个方法封装起来，当需要使用的时候，用方法名调用即可，这样大大简化了程序。

例如计算某个数的平方：

```
public static int square( int x ) {
int y=x * x;
return y; //返回值
}
```

方法的编写和应用都比较灵活，主要是从参数和返回值两个方面去变化，那么可以对方法进行两种分类。

根据参数个数：

(1)无参方法

(2)有参方法

根据返回值类型：

(1)有返回值的方法

根据返回值的类型细分为：①基本数据类型；②引用数据类型。

(2)无返回值的方法

void

例 5.4　无返回值的方法

```
public class FunctionDemo{
    public void getBigger(int x, int y ){
      if( x>=y ) {
        System.out.println( x );
```

```
        }else{
          System.out.println( y );
        }
        return; //没有返回值,return 可以省略
      }
  }
```

对于无返回值类型的方法,它不向本方法外部返回任何值。定义这样的方法时,声明方法返回类型的位置不能省略不写,而应该用关键字 void 来代替,即“空”的意思。

下面例题中是一个有返回值有参方法,该方法具有如下的特点:

- 该方法体有两个 return 语句,但是只有一个 return 语句能被执行。
- 方法的返回类型可以是 Java 中的任何数据类型:基本数据类型(4 种整型、2 种浮点型、字符型、布尔型共 8 种)和引用数据类型(数组、类、接口)。

例 5.5　有返回值和参数的方法

```
public class FunctionDemo {
  public int absolut(int x){
    if(x>=0){
      return x;
    } else{
      return -x;
    }
  }
}
```

下面我们来看一下有返回值和无返回值的方法的对比,由图 5.3 可见 add 方法和 drawRect()方法的返回值分别是 int 和 void,那么返回值是 int 的方法在方法体中一定要有 return 语句,返回值是 void 则不需要有,方法的对比图如图 5.3 所示。

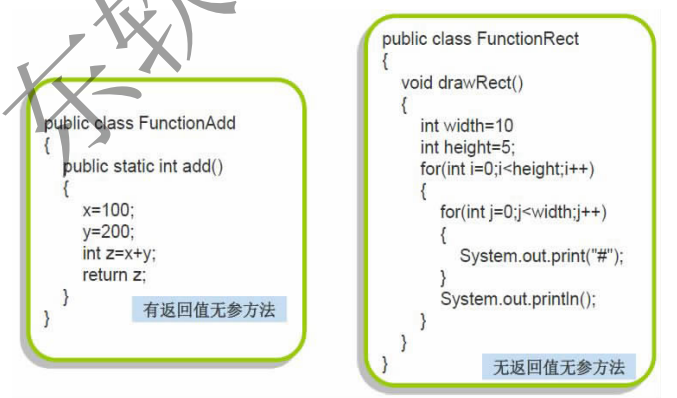

图 5.3　方法的对比

5.2　方法的调用

创建方法就是为了要使用的,那么如何使用一个方法呢,这就是调用。方法只有在被调

用后才生效，那么有返回值的方法和无返回值的方法调用方式是否相同呢，参数如何处理呢？下面就来学习方法的调用方式。

方法的调用方法：

(1)有返回值方法的调用

(2)无返回值方法的调用

调用方法的格式为：

方法名(实际参数表)；

根据方法是否有返回值，通常有两种途径调用方法：

①如果方法返回一个值，对方法的调用通常就当作一个值处理。

②如果方法返回 void，对方法的调用应是当作语句处理。

所谓调用方法，其实就是给方法的入口传入一些值(参数)，然后在出口得到方法执行的结果(返回值)。给方法传入参数的过程，称为“传参”。实际上，方法传参的过程就是把实参赋值给对应的形参的过程，并且实参和形参的数量、类型必须匹配。

- 形参必须注明数据类型
- 实参直接写，不需要类型声明
- return 只能返回一次
- 遇到 return 语句，方法结束执行，后续语句不执行
- 方法的返回值，必须与方法声明中的返回值类型匹配
- 方法定义，不能写在 main()中
- 方法是不能嵌套的

图 5.4 是参数传递的例子，在 FunctionDemo 中有 add 和 main 两个方法，add 方法有两个形式参数，在 main 方法中对 add 方法进行调用，调用时带两个实参，分别是 10 和 20 两个值传递给 x 和 y，又调用时带两个实参，分别是 100 和 200 两个值传递给 x 和 y，如图 5.4 所示。

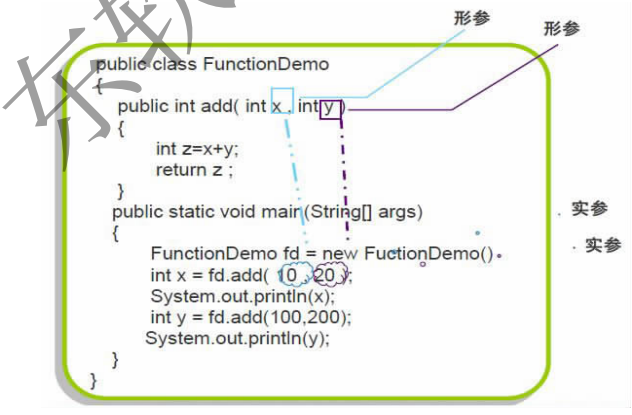

图 5.4　FunctionDemo

通过下面这道例题来学习当作值和当作语句调用的区别。

例 5.6　计算学生期末成绩。

```
public class Student {
  String name;
  int pingshi;
```

```
    int qimo;

    Student(String n, int p, int q) {
      name = n;
      pingshi = p;
      qimo = q;
    }

    void print() {
      System.out.print("姓名为:" + name + " 的同学 ");
    }

    double jisuan() {
      returnpingshi + qimo * 0.5;
    }

    public static void main(String[] args) {
      Student s1;
      s1 = new Student("王明", 30, 80);
      s1.print();//当作语句调用
      System.out.println("总成绩为 " + s1.jisuan());//当作值调用
    }
}
```

运行结果如图 5.5 所示。

图 5.5　Student 的运行结果

下面的例题中把有返回值的方法 public static int getSum(int end)当作了语句调用，这样应用是没有语法错误的，但是方法的返回值没有存储到变量中，因此不会被记录和保存下来，在后续的学习中用到 Java 自带的方法时，有的返回值是程序不需要使用的，就可以这样做。

例 5.7　有返回值并且当作语句调用的方法。

```
class FuncSum_1 {
    public static int  getSum(int end){
      int sum = 0;
      for(int i=1;i<=end;i++){
          sum+=i;
      }
```

```
        return sum;
    }
    public static void main(String args[]){
        getSum(10);
    }
}
```

学习完方法的调用，掌握了如何把简单类型参数传递给方法。也可将对象传递给方法，把对象类型的变量传递给方法和把简单类型传递给方法是有区别的，简单类型和数组类型做形参的对比如图 5.6 所示。

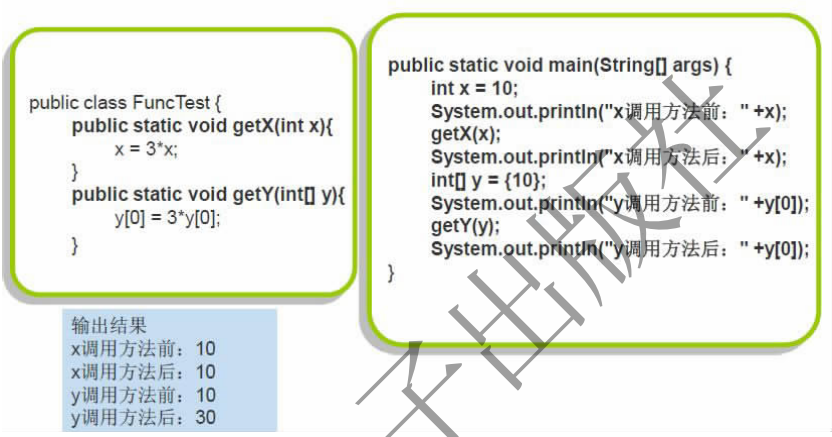

图 5.6　简单类型和数组类型做形参的对比

下面通过一个例子来看一下基本类型和引用类型变量的区别。

例 5.8　参数传递。

```
class A {
    int a;
    public A() {
        a = 1;
    }
    public void add(int m, A n) {
        m++;
        n.a++;
    }
}
public class TestPassObject{
    public static void main(String[] args) {
        int x = 5;
        A y = new A();
        System.out.println("调用前简单类型变量 x="+x);
        System.out.println("调用前引用类型变量 y 的属性 y.a="+y.a);
```

```
        y.add(x, y);
        System.out.println("调用后简单类型变量 x="+x);
        System.out.println("调用后引用类型变量 y 的属性 y.a="+y.a);
    }
}
```

运行结果如图 5.7 所示。

图 5.7　TestPassObject 的运行结果

从上例中可以看出，类 A 中有名为 add 的方法，方法的两个参数分别为简单类型的变量和引用类型的变量，在传递的过程中是有区别的，从结果可以看出简单类型变量 x 作为实参，把值传递给形参 m，m 的值改变后，并没有影响到 x，而引用类型变量 y 作为实参，形参 n 的改变影响了 y。

5.3　方法的重载

在前面的求两个数的最大值的例题中只能用于 int 类型，但是如果要求两个 double 类型的数的最大值，应该怎么做呢，解决办法就是创建另一个方法名相同但是参数不同的方法，代码如下：

例 5.9　求两个数的最大值。

```
public static double getBigger(double num1, double num2){
    if (num1 > num2)
        return num1;
    else
        return num2;
}
```

这种在同一个类中允许同时存在一个以上同名的方法的现象就叫做方法的重载，也就是说，两个或者两个以上的方法，具有相同的名称和不同的参数列表。Java 编译器根据方法名和参数情况共同决定调用哪个方法。

方法重载的规则：

(1)方法名称相同

(2)方法的参数必须不同，参数个数不同或参数类型不同

(3)方法的返回值类型可以相同，也可以不同

重载的例子如图 5.8 所示。

```
class SumFunc {

public int getSum(int end){
    int sum = 0;
    for(int i=1;i<=end;i++){
        sum+=i;
    }
    return sum;
}

public int getSum(int start,int end){
    int sum = 0;
    for(int i=start;i<end;i++){
        sum+=i;
    }
    return sum;
}

}
```

图 5.8　重载的例子

下面通过一道例题来看一下重载的应用。

例 5.10　计算参数之和。

```
class MethodAdd {
  /*
   * 计算两个整数的和
   */
  int add(int x, int y) {
    int sum;
    sum = x + y;
    return sum;
  }
  /*
   * 计算三个整数的和
   */
  int add(int x, int y, int z) {
    return x+y+z;
  }
  /*
   * 计算两个双精度浮点数的和
   */
  double add(double x, double y) {
    return x + y;
  }
}
```

调用如下：

```
public class TestMethodAdd {
  public static void main(String args[]) {
    MethodAdd ma = new MethodAdd();
    int sum1;
    sum1 = ma.add(135, 246);
    System.out.println(sum1);
    System.out.println(ma.add(123, 456, 789));
    double sum3;
    sum3 = ma.add(1.2, 4.5);
    System.out.println(sum3);
  }
}
```

运行结果如图 5.9 所示。

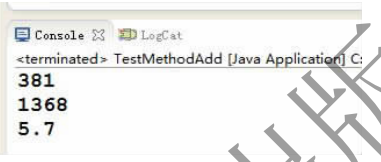

图 5.9　TestMethodAdd 的运行结果

在上面的例子中 MethodAdd 类中有三个 add 方法，但是带的参数各不相同，在 main 方法中调用时，会根据带的参数是两个还是三个，或者是整数还是浮点数共同决定调用和实参匹配的 add 方法。

【总结与提示】

1. 方法可以有返回值也可以没有，使用 return 语句返回值；
2. 方法没有返回值时，返回值类型为 void；
3. 定义方法时参数可以有多个，也可以没有；
4. 调用方法时，实参的个数和类型必须和形参匹配；
5. 方法可以重载。

5.4　课后作业

(一)选择题

1. 有以下方法的定义，请选择该方法的返回类型(　　)

```
ReturnType  method(byte x, float y)
{
  return  (short)x/y * 2;
}
```

A. byte　　B. short　　C. int　　D. float

2. 下列方法定义中，方法头不正确的是(　　)

A. double m(int m){ }　　B. void m(int m){ }

C. public int m(int m, int n){ }　　D. m(int h,int m,int n){ }

3. 在 Java 中，一个类可同时定义许多同名的方法，这些方法的形式参数的个数、类型或顺序各不相同，返回的值也可以不相同。这种特性称为(　　)

A. 隐藏　　B. 覆盖　　C. 重载　　D. Java 不支持此特性

4. 下列选项中，哪些可以与 `void setAge(int year,int month,int day)` 方法在同一个类中定义(　　)

A. public void setAge() { }　　B. void setAge(int age) { }

C. void setAge(int y,int m,int d){ }　　D. int setAge(Date d) { }

E. int setAge(int year,int month,int day){}

F. void setage(int year,int month,int dat){}

(二)编程题

1. 编写一个方法，求整数 n 的阶乘，例如 5 的阶乘是 1 * 2 * 3 * 4 * 5。

2. 编写一个方法，判断该年份是平年还是闰年。

3. 编写一个方法，输出大于 200 的最小的质数。

4. 写一个方法，功能：定义一个一维的 int 数组，长度任意，然后将它们按从小到大的顺序输出(使用冒泡排序)。