# 7.6 可变参数

## 7.6.1 基本概念

在 Java 编程中，我们有时会遇到这样的情况：需要在同一个类中编写多个同名且功能相同，但参数个数不同的方法。

例如，编写一个计算数字总和的方法，可能需要计算两个数的和、三个数的和、四个数的和等等。按照常规做法，我们可能会像这样定义多个方法：

```
public int sum(int n1, int n2) {
    return n1 + n2;
}
public int sum(int n1, int n2, int n3) {
    return n1 + n2 + n3;
}
public int sum(int n1, int n2, int n3, int n4) {
    return n1 + n2 + n3 + n4;
}
```

然而，随着参数个数的不断增加，代码会变得冗长且繁琐。Java 提供了可变参数机制，允许我们将这些功能相同、参数个数不同的方法，封装成一个方法，从而简化代码编写。

## 7.6.2 基本语法

可变参数的基本语法形式如下：

```
访问修饰符 返回类型 方法名(数据类型... 形参名) {
    // 方法体
}
```

这里的`数据类型...`表示该参数是可变参数，它可以接受零个或多个指定数据类型的参数值。在方法体内部，可变参数可以当作数组来使用。例如，`int... nums`表示`nums`是一个可以存储多个`int`类型值的可变参数，在方法中可以通过`nums.length`获取传入参数的个数，通过`nums[i]`访问具体的参数值。

## 7.6.3 快速入门案例（VarParameter01.java）

下面通过一个具体的案例来深入理解可变参数的使用。我们定义一个`HspMethod`类，其中有一个`sum`方法，该方法能够计算任意个数整数的和。

```
public class VarParameter01 {
    //编写一个main方法
    public static void main(String[] args) {
        HspMethod m = new HspMethod();
        System.out.println(m.sum(1, 5, 100)); // 输出106
        System.out.println(m.sum(1, 19)); // 输出20
    }
}
```

```
class HspMethod {
    // 使用可变参数优化计算和的方法
    public int sum(int... nums) {
        // System.out.println("接收的参数个数=" + nums.length);
        int res = 0;
        for (int i = 0; i < nums.length; i++) {
            res += nums[i];
        }
        return res;
    }
}
```

在上述代码中：

1. `public int sum(int... nums)`定义了一个名为`sum`的方法，它接受可变参数`nums`，类型为`int`。这意味着该方法可以接受任意数量的整数作为参数。
2. 在方法体内部，通过`for`循环遍历`nums`数组，将所有参数值累加起来，最后返回累加的结果。
3. 在`main`方法中，我们分别调用`sum`方法传入不同个数的参数，验证其能够正确计算总和。

# 7.6.4 注意事项和使用细节

## 细节一：可变参数的实参可以为数组

可变参数在调用时，实参既可以像普通参数那样逐个传入，也可以直接传入一个与可变参数类型一致的数组。例如：

```
public class VarParameterDetail {
    //编写一个main方法
    public static void main(String[] args) {
        int[] arr = {1, 2, 3};
        T t1 = new T();
        t1.f1(arr);
    }
}

class T {
    public void f1(int... nums) {
        System.out.println("长度=" + nums.length);
    }
}
```

在上述代码中，`main`方法中定义了一个`int`类型数组`arr`，然后将其作为实参传递给`f1`方法。在`f1`方法中，可变参数`nums`能够正确接收数组中的元素，并且可以像操作普通数组一样获取其长度。

## 细节二：可变参数与普通参数的位置关系

可变参数可以和普通类型的参数一起放在形参列表中，但必须保证可变参数在最后。例如：

```
class T {
    public void f2(String str, double... nums) {
        // 方法体
    }
}
```

在`f2`方法中，`str`是普通参数，`nums`是可变参数。这种写法是正确的，因为可变参数位于形参列表的最后。如果将可变参数放在普通参数之前，例如`public void f2(double... nums, String str)`，则会导致编译错误。

### 细节三：一个形参列表中只能有一个可变参数

在一个方法的形参列表中，只能出现一个可变参数。如果定义多个可变参数，例如`public void f3(int... nums1, double... nums2)`，会导致编译错误。这是因为编译器在解析方法调用时，无法确定传入的参数应该对应哪个可变参数。

## 7.6.5 课堂练习

### 题目

定义一个`HspMethod`类，要求编写一个方法，该方法能够实现返回姓名和不同门数课程成绩的总分。具体来说，有三个功能需求：返回姓名和两门课成绩 (总分)，返回姓名和三门课成绩 (总分)，返回姓名和五门课成绩（总分）。将这些功能封装成一个可变参数的方法。

# 课后练习

## 一、选择题

1. 关于 Java 中可变参数的说法，正确的是（ ）
   A. 可变参数可以放在方法参数列表的任意位置
   B. 一个方法中可以有多个可变参数
   C. 可变参数本质上就是一个数组
   D. 可变参数只能接收基本数据类型
2. 以下方法定义中，使用可变参数正确的是（ ）
   A. `public void print(int... nums, String str) {... }`
   B. `public void print(String str, int... nums) {... }`
   C. `public void print(int... nums, double... doubles) {... }`
   D. `public void print(int num1, int num2,... nums) {... }`
3. 已知方法定义`public void show(String... names)`，以下调用该方法正确的是（ ）
   A. `show("Tom");`
   B. `show();`
   C. `String[] arr = {"Tom", "Jerry"}; show(arr);`
   D. 以上调用都正确
4. 若有方法`public int calculate(int... nums)`，在方法内部访问可变参数的元素，以下正确的是（ ）
   A. `nums(0)`
   B. `nums{0}`
   C. `nums[0]`
   D. `nums.get(0)`
5. 可变参数在方法调用时，实参可以是（ ）
   A. 一个与可变参数类型一致的数组
   B. 多个与可变参数类型一致的值
   C. 零个与可变参数类型一致的值
   D. 以上都可以
6. 定义方法`public void test(int a, double... b)`，调用该方法时，以下实参传递正确的是（ ）

A. `test(1);`

B. `test(1, 2.5);`

C. `test(1, 2.5, 3.5);`

D. B 和 C 都正确

7. 下列关于可变参数和方法重载的关系，说法错误的是（ ）

A. 可变参数可以用于方法重载

B. 可变参数方法和普通方法可以构成方法重载

C. 两个可变参数方法，只要参数类型不同，就能构成方法重载

D. 方法重载时，可变参数方法的参数个数可以不同

8. 方法`public void f(String... strs)`与方法（ ）不能构成方法重载。

A. `public void f(String str)`

B. `public void f(int... ints)`

C. `public void f(String str1, String str2)`

D. `public void f()`

9. 以下代码中，编译会出错的是（ ）

A. `public void m(int... nums) {... }`

B. `public void m(String str, int... nums) {... }`

C. `public void m(int... nums, String str) {... }`

D. `public void m(int num1, int num2, int... nums) {... }`

10. 假设类`A`中有方法`public void process(int... values)`，以下能在类`A`中合法定义的方法是（ ）

A. `public void process(int value)`

B. `public void process(double... values)`

C. `public int process(int... values)`

D. `public void PROCESS(int... values)`

## 二、填空题

1. Java 中可变参数的语法格式为`数据类型... 形参名`，在方法体中，可变参数可以当作__来使用。
2. 一个方法的形参列表中只能有__个可变参数。
3. 可变参数必须放在方法参数列表的__位置。
4. 已知方法定义`public double average(double... nums)`，在方法体中计算参数平均值，首先要获取参数个数，可通过__实现。
5. 若有方法`public String combine(String... words)`，要将传入的所有字符串用逗号连接起来返回，在方法体中可以使用__类来拼接字符串。
6. 调用可变参数方法时，实参可以是一个与可变参数类型一致的__，也可以是多个该类型的值。
7. 定义方法`public void printInfo(int id, String... messages)`，在调用该方法时，第一个实参的类型是__。
8. 可变参数方法`public int sum(int... nums)`，如果调用`sum(1, 2, 3)`，方法内部`nums.length`的值为__。
9. 方法重载时，可变参数方法与其他方法构成重载的依据是__不同。
10. 若有方法`public void display(boolean... flags)`，在方法体中遍历可变参数的循环条件可以是`for(int i = 0; i < ______; i++)`。

## 三、判断题

1. 可变参数只能用于实例方法，不能用于静态方法。（ ）
2. 可变参数方法在调用时，必须传入至少一个参数。（ ）
3. 两个方法，一个是可变参数方法，一个是普通方法，只要方法名相同，就构成方法重载。（ ）
4. 在方法体中，对可变参数进行操作和对普通数组的操作方式类似。（ ）
5. 定义可变参数方法时，数据类型和省略号之间不能有空格。（ ）
6. 一个类中不能同时存在普通方法和可变参数方法。（ ）
7. 可变参数方法可以被其他类继承和重写。（ ）
8. 若有方法`public void m(int... nums)`，则`public void m(int num1, int num2)`与之构成方法重载。（ ）
9. 可变参数在编译时会被转换为数组形式。（ ）
10. 调用可变参数方法时，实参的类型必须与可变参数声明的类型完全一致。（ ）

## 四、简答题

1. 简述 Java 中可变参数的概念及作用。
2. 可变参数的基本语法是什么？请举例说明。
3. 可变参数在方法调用时，实参有哪些传递方式？
4. 可变参数与普通参数混合使用时，需要注意什么？
5. 说明可变参数方法与普通方法构成方法重载的条件。
6. 假设你要编写一个方法，实现计算不同个数整数的乘积，使用可变参数实现该方法，并说明方法的设计思路。
7. 解释为什么一个方法的形参列表中只能有一个可变参数。
8. 在方法体中，如何对可变参数进行遍历和操作？请结合代码示例说明。
9. 可变参数方法在实际编程中有哪些应用场景？请举例说明。
10. 可变参数方法与数组作为参数的方法有什么区别和联系？

## 五、编程题

1. 编写一个 Java 类`Calculator`，包含一个可变参数方法`multiply`，该方法能够计算任意个数整数的乘积并返回结果。在`main`方法中调用`multiply`方法，传入不同个数的整数进行测试。
2. 设计一个类`StringUtil`，包含一个可变参数方法`join`，该方法接收多个字符串参数，将这些字符串用特定的分隔符（例如空格）连接成一个字符串并返回。在`main`方法中调用`join`方法，传入不同个数的字符串进行测试。
3. 编写一个类`AverageCalculator`，包含一个可变参数方法`calculateAverage`，该方法能够计算任意个数浮点数的平均值并返回结果。在`main`方法中调用`calculateAverage`方法，传入不同个数的浮点数进行测试，注意处理传入参数个数为 0 的情况（可以返回一个特定值，如 - 1 表示错误）。
4. 定义一个类`Student`，包含一个可变参数方法`printScores`，该方法接收学生姓名和多个成绩作为参数，输出学生姓名以及所有成绩的总和和平均值。在`main`方法中创建`Student`类的对象，调用`printScores`方法，传入不同学生的姓名和成绩进行测试。
5. 编写一个 Java 类`FileHandler`，包含一个可变参数方法`readFiles`，该方法接收多个文件路径作为参数，尝试读取每个文件的内容（可以简单返回文件路径字符串来模拟读取操作），并将所有文件内容拼接成一个字符串返回。在`main`方法中调用`readFiles`方法，传入不同个数的文件路径进行测试。
6. 设计一个类`MathOperations`，包含两个可变参数方法：`sum`方法用于计算任意个数整数的和，`product`方法用于计算任意个数整数的乘积。在`main`方法中调用这两个方法，传入不同个数的整数进行测试。

7. 编写一个类`MessagePrinter`，包含一个可变参数方法`printMessages`，该方法接收一个前缀字符串和多个消息字符串作为参数，在每个消息字符串前加上前缀后输出。在`main`方法中调用`printMessages`方法，传入不同的前缀和消息字符串进行测试。
8. 定义一个类`ShapeAreaCalculator`，包含一个可变参数方法`calculateTotalArea`，该方法接收多个形状的面积（假设形状面积为`double`类型）作为参数，计算并返回所有形状面积的总和。在`main`方法中调用`calculateTotalArea`方法，传入不同个数的形状面积进行测试。
9. 编写一个 Java 类`ShoppingCart`，包含一个可变参数方法`calculateTotalPrice`，该方法接收多个商品价格（假设商品价格为`double`类型）作为参数，计算并返回购物车中所有商品的总价。同时，添加一个方法`applyDiscount`，该方法接收总价和折扣率作为参数，计算并返回折扣后的价格。在`main`方法中调用这两个方法，传入不同个数的商品价格和折扣率进行测试。
10. 设计一个类`DataProcessor`，包含一个可变参数方法`processData`，该方法接收多个整数参数，对这些参数进行排序（可以使用 Java 内置的排序方法），并将排序后的结果以字符串形式返回（例如：“[1, 2, 3, 4]”）。在`main`方法中调用`processData`方法，传入不同个数的整数进行测试。