7.4 方法重载（OverLoad）

7.4.1 基本介绍

1.定义

在 Java 中，允许在同一个类里存在多个同名的方法，但前提是这些方法的形参列表不一致，要么类型不一致要么个数不一致。

例如，我们常用的`System.out.println();`，其中的`out`属于`PrintStream`类型，`println`方法就存在多个重载版本，以适应不同类型数据的输出需求，如`System.out.println(100);`用于输出整数，`System.out.println("hello,world");`用于输出字符串等。

下面通过具体代码示例`OverLoad01.java`来进一步理解：

```
public class OverLoad01 {
    //编写一个main方法
    public static void main(String[] args) {
        // 以下展示了System.out.println方法针对不同类型参数的调用
        System.out.println(100);
        System.out.println("hello,world");
        System.out.println('h');
        System.out.println(1.1);
        System.out.println(true);

        MyCalculator mc = new MyCalculator();
        System.out.println(mc.calculate(1, 2));
        System.out.println(mc.calculate(1.1, 2));
        System.out.println(mc.calculate(1, 2.1));
    }
}

class MyCalculator {
    //下面的四个calculate方法构成了重载
    //两个整数的和
    public int calculate(int n1, int n2) {
        System.out.println("calculate(int n1, int n2)被调用");
        return n1 + n2;
    }

    // 注意：以下方法定义是错误的，因为它与已有的calculate(int n1, int n2)方法构成重复定义，并非重载
    // public void calculate(int n1, int n2) {
    //     System.out.println("calculate(int n1, int n2)被调用");
    //     int res = n1 + n2;
    // }

    // 以下方法同样是错误的，属于方法的重复定义，并非重载
    // public int calculate(int a1, int a2) {
    //     System.out.println("calculate(int n1, int n2)被调用");
    //     return a1 + a2;
    // }
```

```
    //一个整数，一个double的和
    public double calculate(int n1, double n2) {
        return n1 + n2;
    }

    //一个double,一个Int和
    public double calculate(double n1, int n2) {
        System.out.println("calculate(double n1, int n2)被调用..");
        return n1 + n2;
    }

    //三个int的和
    public int calculate(int n1, int n2, int n3) {
        return n1 + n2 + n3;
    }
}
```

在上述代码中，`MyCalculator`类里定义了四个名为`calculate`的方法，它们的参数列表各不相同，从而构成了方法重载。

7.4.2 重载的好处

减轻起名的麻烦

在编程过程中，如果没有方法重载机制，对于功能类似但参数不同的方法，我们可能需要为每个方法取不同的名字。例如，在`MyCalculator`类中，如果不能使用重载，对于计算两个整数和的方法可能取名为`sumOfTwoIntegers`，计算一个整数和一个双精度浮点数和的方法可能取名为`sumOfIntegerAndDouble`等，这样会使方法名变得冗长且难以记忆。而通过方法重载，我们可以统一使用`calculate`作为方法名，使代码更加简洁易读。

减轻记名的麻烦

从开发者的角度来看，记忆多个不同功能但名字相似的方法是一项负担。使用方法重载，只需记住一个具有代表性的方法名，通过参数的不同来区分具体的功能。例如，在调用`MyCalculator`类的计算方法时，开发者只需要记住`calculate`方法，根据实际传入的参数类型和数量来实现不同的计算需求，大大减轻了记忆的负担，提高了开发效率。

7.4.3 快速入门案例

我们通过`OverLoad01.java`案例来深入理解方法重载的实际应用。在这个案例中，定义了一个`MyCalculator`类，其中包含多个`calculate`方法。

1. `calculate(int n1, int n2)`：用于计算两个整数的和，返回值类型为`int`。在方法体中，先输出该方法被调用的提示信息，然后返回两个整数相加的结果。
2. `calculate(int n1, double n2)`：该方法接收一个`int`类型参数和一个`double`类型参数，计算它们的和并返回`double`类型的结果。此方法体现了参数类型不同构成的方法重载。
3. `calculate(double n1, int n2)`：与上一个方法类似，只是参数顺序不同，同样计算并返回一个`double`类型的和，这展示了参数类型顺序不同也能构成方法重载。
4. `calculate(int n1, int n2, int n3)`：用于计算三个整数的和，返回值类型为`int`。通过增加参数个数，与前面的`calculate`方法构成重载。

在`main`方法中，创建了`MyCalculator`类的对象`mc`，并分别调用不同版本的`calculate`方法，传入不同类型和数量的参数，验证了方法重载的效果。

7.4.4 注意事项和使用细节

1. **方法签名的唯一性**：方法签名由方法名和参数列表组成，在同一个类中，每个重载方法的方法签名必须是唯一的。这意味着不仅参数列表要不同，方法名也必须相同。例如，不能同时存在两个`calculate(int n1, int n2)`方法，即使它们的返回值类型不同也不行，因为方法签名完全一致。
2. **返回值类型无关**：方法重载与方法的返回值类型无关。也就是说，仅仅返回值类型不同，而方法名和参数列表相同的方法，不能构成方法重载。例如，以下两个方法定义是错误的：

```
public int calculate(int n1, int n2) {
    return n1 + n2;
}
public double calculate(int n1, int n2) {
    return n1 + n2;
}
```

这两个方法虽然返回值类型不同，但方法名和参数列表完全一样，会导致编译错误。
\3. **参数类型的兼容性**：在调用重载方法时，Java 编译器会根据传入参数的类型和数量来匹配最合适的方法。如果传入的参数类型与某个重载方法的参数类型不完全匹配，但可以通过自动类型转换进行兼容，那么也能正确调用该方法。例如，在`MyCalculator`类中，如果有一个方法`calculate(double n1, double n2)`，当调用`mc.calculate(1, 2);`时，由于`int`类型可以自动转换为`double`类型，所以实际上调用的是`calculate(double n1, double n2)`方法。
\4. **方法重载的层次**：方法重载可以在一个类中多层嵌套。例如，一个类中可以同时存在参数个数不同、参数类型不同以及参数类型顺序不同的多种重载方法，只要它们的方法签名是唯一的即可。

7.4.5 课堂练习题

题目 1

编写程序，类`Methods`中定义三个重载方法并调用。方法名为`m`。

1. 第一个方法接收一个`int`参数，执行平方运算并输出结果。
2. 第二个方法接收两个`int`参数，执行相乘运算并输出结果。
3. 第三个方法接收一个字符串参数，输出字符串信息。
   在主类的`main()`方法中分别用参数区别调用这三个方法。

题目 2

定义三个重载方法`max()`。

1. 第一个方法返回两个`int`值中的最大值。
2. 第二个方法返回两个`double`值中的最大值。
3. 第三个方法返回三个`double`值中的最大值。
   在主类的`main()`方法中分别调用这三个方法。

课后练习

一、选择题

1. 以下关于 Java 方法重载的说法，正确的是（ ）
   A. 方法重载要求方法名相同，参数列表和返回值类型都必须不同
   B. 只要方法名相同，就是方法重载
   C. 方法重载中，方法名相同，参数列表必须不同
   D. 方法重载只适用于成员方法，不适用于静态方法
2. 下列代码中，能与`public void test(int a, double b)`构成方法重载的是（ ）
   A. `public void test(int a, double b) {... }`
   B. `public int test(int a, double b) {... }`
   C. `public void test(double a, int b) {... }`
   D. `public void test(int a) {... }`
3. 在 Java 中，以下哪组方法构成了方法重载？（ ）
   A. `public void m1(int a);`和`public void m2(int a);`
   B. `public int m(int a, int b);`和`public int m(int x, int y);`
   C. `public void m(int a);`和`public void m(int a, int b);`
   D. `public void m(int a);`和`public int m(int a);`
4. 假设类`A`中有方法`public void f(int x)`，以下能在类`A`中合法定义的重载方法是（ ）
   A. `public void f(int y)`
   B. `public void f(int x, int y)`
   C. `public int f(int x)`
   D. `public void F(int x)`
5. 方法重载的主要作用是（ ）
   A. 提高代码的运行效率
   B. 使程序更美观
   C. 减少代码中方法名的数量，提高代码可读性
   D. 实现多态性
6. 下列关于方法重载的描述，错误的是（ ）
   A. 方法重载时，参数的类型可以不同
   B. 方法重载时，参数的个数可以不同
   C. 方法重载时，参数的顺序可以不同
   D. 方法重载时，方法的返回值类型必须不同
7. 已知类`Test`中有方法`public void print(int a)`，以下方法中，（ ）不能与该方法构成重载。
   A. `public void print(double a)`
   B. `public void print(int a, int b)`
   C. `public void print(int x)`
   D. `public void print(String a)`
8. 以下代码中，有（ ）处方法重载错误。

```
public class ErrorTest {
    public void m(int a) {}
    public int m(int a) {}
    public void m(double a) {}
    public void M(int a) {}
}
```

A. 1
B. 2
C. 3
D. 4
\9. 方法重载与方法重写的区别之一是（ ）
A. 方法重载发生在父子类之间，方法重写发生在同一类中
B. 方法重载方法名必须相同，方法重写方法名可以不同
C. 方法重载参数列表必须不同，方法重写参数列表必须相同
D. 方法重载返回值类型必须不同，方法重写返回值类型必须相同
\10. 类`Calculator`中有方法`public int add(int a, int b)`，要实现计算三个整数相加的功能，以下重载方法定义正确的是（ ）
A. `public int add(int a, int b, int c) {... }`
B. `public double add(int a, int b, int c) {... }`
C. `public int ADD(int a, int b, int c) {... }`
D. `public int add(int x, int y, int z) {... }`

二、填空题

1. 在 Java 中，方法重载要求方法名__，参数列表__。
2. 方法重载时，参数列表的不同可以体现在参数的__、__或__不同。
3. 若有方法`public void show(int num)`，则能与它构成重载的方法可以是`public void show(_____ num)`（填写一种参数类型）。
4. 方法重载与方法的__类型无关。
5. 假设类`MathUtil`中有方法`public int multiply(int a, int b)`，要添加一个计算两个浮点数相乘的重载方法，方法定义为`public _____ multiply(_____ a, _____ b)`。
6. 在同一个类中，多个重载方法的__必须是唯一的。
7. 若类`A`中有方法`public void print(String s)`，要定义一个接收整数参数并打印的重载方法，方法声明为`public void print(_____)`。
8. 方法重载可以使代码更具__性和可维护性。
9. 已知方法`public double calculate(double a, double b)`，要实现计算一个整数和一个双精度浮点数相除的重载方法，方法定义为`public double calculate(_____, double b)`。
10. 类`MyClass`中已经有方法`public void process(int x)`，要添加一个接收两个整数参数的重载方法，方法声明为`public void process(_____, _____)`。

三、判断题

1. 方法重载时，只要方法名相同，参数个数和类型可以随意变化。（ ）
2. 方法重载与方法的返回值类型没有关系。（ ）
3. 在一个类中，不能有两个方法签名相同的方法（包括方法名和参数列表）。（ ）
4. 方法重载只能用于实例方法，不能用于静态方法。（ ）
5. 若有方法`public void f(int a)`，则`public int f(int a)`能与之构成方法重载。（ ）
6. 方法重载可以提高代码的可读性，因为可以用相同的方法名表示相似的功能。（ ）
7. 方法重载时，参数顺序不同不能构成重载。（ ）
8. 在 Java 中，构造方法不能被重载。（ ）
9. 方法重载时，方法的访问修饰符必须相同。（ ）
10. 一个类中，如果有多个同名方法，一定构成方法重载。（ ）

四、简答题

1. 简述 Java 中方法重载的概念。
2. 方法重载的好处有哪些？请至少列举两点。
3. 方法重载与方法重写有什么区别？（从定义位置、方法签名、返回值类型等方面回答）
4. 请说明方法重载时，参数列表不同的具体表现形式有哪些。
5. 假设你正在设计一个数学运算类`MathOperation`，已经有一个计算两个整数相加的方法`public int add(int a, int b)`，现在要添加一个计算三个整数相加和一个计算两个浮点数相加的方法，分别写出这两个重载方法的定义。
6. 在 Java 中，判断以下两个方法是否构成方法重载，并说明理由。

```
public void m(int a, double b);
public void m(double a, int b);
```

1. 方法重载在实际编程中有哪些应用场景？请举例说明。
2. 若在一个类中定义了方法`public void print(int num)`，现在想定义一个接收字符串参数并打印的重载方法，在方法体中，除了参数类型不同外，方法实现逻辑与原方法完全不同，这样做是否符合方法重载的要求？为什么？
3. 解释为什么方法重载与返回值类型无关。如果仅返回值类型不同，而方法名和参数列表相同，会出现什么情况？
4. 简述在调用重载方法时，Java 编译器是如何确定调用哪个方法的。

五、编程题

1. 编写一个 Java 类

   `Calculator`

   ，包含三个重载方法：

   - `add(int a, int b)`：计算两个整数的和并返回。
   - `add(double a, double b)`：计算两个双精度浮点数的和并返回。
   - `add(int a, int b, int c)`：计算三个整数的和并返回。
     在`main`方法中调用这三个方法并输出结果。
2. 设计一个类

   `Shape`

   ，其中包含计算面积的重载方法：

   - `area(int radius)`：计算半径为`radius`的圆的面积（假设`π`取值为 3.14）。
   - `area(int length, int width)`：计算长为`length`，宽为`width`的矩形的面积。
     在`main`方法中创建`Shape`类的对象，分别调用这两个方法计算并输出圆和矩形的面积（圆半径设为 5，矩形长设为 4，宽设为 3）。
3. 编写一个类

   `StringUtil`

   ，包含以下重载方法：

   - `concat(String s1, String s2)`：将两个字符串连接起来并返回。

- `concat(String[] arr)`：将字符串数组中的所有字符串连接成一个字符串并返回。
  在`main`方法中调用这两个方法，分别传入两个字符串和一个字符串数组，输出连接后的结果。

4. 定义一个类

```
NumberProcessor
```

，包含以下重载方法：

- `max(int a, int b)`：返回两个整数中的较大值。
- `max(double a, double b)`：返回两个双精度浮点数中的较大值。
- `max(int a, int b, int c)`：返回三个整数中的最大值。
  在`main`方法中调用这三个方法，分别传入不同的参数，输出最大值。

5. 编写一个 Java 类

```
FileHandler
```

，包含两个重载方法：

- `read(String filePath)`：从指定路径的文件中读取内容并返回字符串（假设文件内容为纯文本，简单返回一个固定字符串 “文件内容” 来模拟读取操作）。
- `read(String filePath, int lineNumber)`：从指定路径的文件中读取指定行号的内容并返回字符串（同样简单返回一个固定字符串 “第 X 行内容”，X 为传入的行号来模拟读取操作）。
  在`main`方法中调用这两个方法，传入不同参数，输出读取结果。

6. 设计一个类

```
TemperatureConverter
```

，包含以下重载方法：

- `convertToFahrenheit(double celsius)`：将摄氏温度转换为华氏温度（转换公式：华氏温度 = 摄氏温度 * 1.8 + 32）。
- `convertToCelsius(double fahrenheit)`：将华氏温度转换为摄氏温度（转换公式：摄氏温度 = (华氏温度 - 32) / 1.8）。
  在`main`方法中创建`TemperatureConverter`类的对象，分别调用这两个方法，将 30 摄氏度转换为华氏温度，将 86 华氏温度转换为摄氏温度，并输出结果。

7. 编写一个类

```
MathFunctions
```

，包含三个重载方法：

- `power(int base, int exponent)`：计算`base`的`exponent`次幂（使用循环实现）。
- `power(double base, int exponent)`：计算`double`类型的`base`的`exponent`次幂。
- `power(int base, double exponent)`：计算`int`类型的`base`的`double`类型的`exponent`次幂（使用`Math.pow`方法）。
  在`main`方法中调用这三个方法，传入不同参数，输出幂运算结果。

8. 定义一个类

```
Student
```

，包含以下重载方法：

- `printInfo(String name)`：打印学生姓名。
- `printInfo(String name, int age)`：打印学生姓名和年龄。
- `printInfo(String name, int age, String major)`：打印学生姓名、年龄和专业。
  在`main`方法中创建`Student`类的对象，分别调用这三个方法，传入不同参数，输出学生信息。

9. 编写一个类

```
ArrayUtil
```

，包含以下重载方法：

- `sum(int[] arr)`：计算整数数组中所有元素的和并返回。
- `sum(double[] arr)`：计算双精度浮点数数组中所有元素的和并返回。
- `sum(int[][] multiArr)`：计算二维整数数组中所有元素的和并返回。
  在`main`方法中创建不同类型的数组，调用相应的`sum`方法，输出数组元素的和。

10. 设计一个类

```
Operation
```

，包含以下重载方法：

- `calculate(int a, int b, String operator)`：根据传入的运算符（“+”、“-”、“*”、“/”），对两个整数进行相应的运算并返回结果（注意除法运算时要处理除数为 0 的情况，返回一个特殊值，如 -1 表示错误）。
- `calculate(double a, double b, String operator)`：根据传入的运算符，对两个双精度浮点数进行相应的运算并返回结果（同样处理除法运算除数为 0 的情况）。
  在`main`方法中调用这两个方法，传入不同参数，输出运算结果。