第 2 章 Java 类的组成

学习第 1 章后，读者已经对 Java 语言有了初步的了解，本章将具体解析 Java 类的组成元素。任何一个 Java 类，都有 5 种基本组成元素：属性、方法、构造方法、块和内部类。其中属性、方法、构造方法是使用最多的元素，而块、内部类使用较少。本章将对各种元素进行学习。

Java 类基本结构

2.1 类

类的声明形式及作用

要学习 Java 类的组成，首先要了解 Java 类的声明形式。类的声明方式如下（【】代表可选项）：

```
【访问权限修饰符】【修饰符】class 类名{
类体
}
```

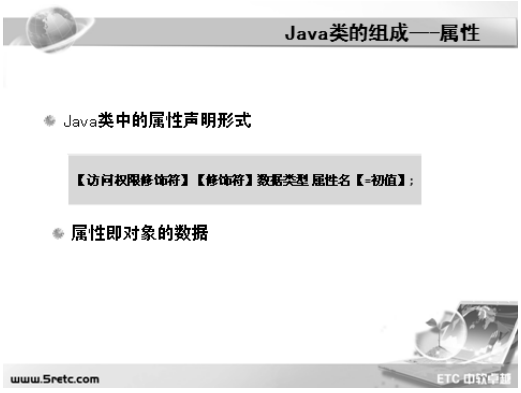
假设有一个员工类，类名为 Employee，则声明形式如下：

```
public class Employee{
}
```

其中 public 是访问权限修饰符，Employee 是类的名字。访问权限修饰符的相关知识将在后面章节详细学习。该类的类体目前为空，类体往往包括属性、方法、构造方法、块、内部类等元素，下面将详细学习。

Java 语言中的标识符有哪些规则？标识符可以由字母、数字、下画线、$符组成，但是不能以数字开头。类名的首字母要大写，属性、方法名首字母小写，第二个单词的首字母大写。标识符中可以包含关键字，但是不能直接使用关键字。如类名 Employee、方法名 getName 都是合法且规范的标识符。而 1test、class 都是不合法的标识符。

2.2　属性



属性的声明形式及作用

类是对象的模板，对象都会有不同的属性，所以类里应该声明该类所具有的属性。属性即对象的数据，如员工的姓名、薪水、数量，就是员工类的属性。属性的声明方式如下（【】代表可选项）：

```
【访问权限修饰符】【修饰符】数据类型 属性名【=初值】;
```

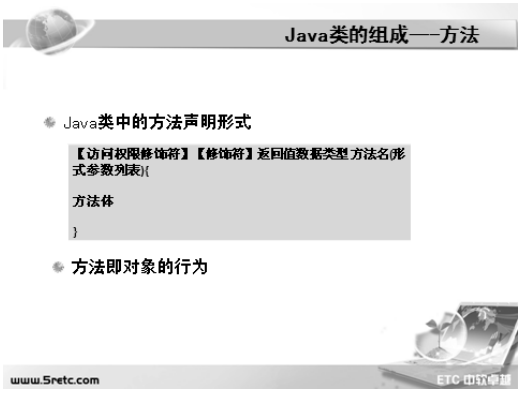
在 Employee 类中声明姓名、薪水、数量 3 个属性，代码如下：

```
private String name;
private double salary;
private static int count=0;
```

其中，private 是访问权限修饰符的一种，表示私有权限；static 是修饰符的一种，表示静态变量，二者都不是必须使用。String、double、int 均为数据类型，name、salary、count 均为属性名称。其中 count 的初始值为 0。权限修饰符、修饰符、数据类型，均在下面章节详细展开。

2.3 方法

2.3.1 方法的声明形式



方法的声明形式及作用

对象除了具有属性外，还有自己的行为，即方法。所以类里除了要声明属性外，还要声明类的所有方法。如员工对象可以修改姓名、修改薪水、查看姓名、查看薪水，这些就是 Employee 类应该声明的方法。方法的声明方式如下（【】代表可选项）：

```
【访问权限修饰符】【修饰符】返回值数据类型 方法名(形式参数列表){
方法体
}
```

在 Employee 类里声明修改姓名、修改薪水、查看姓名、查看薪水 4 个方法，代码如下：

```
public String getName() {
      return name;
}
public void setName(String name) {
      this.name = name;
}
public double getSalary() {
      return salary;
}
public void setSalary(double salary) {
      this.salary = salary;
}
```

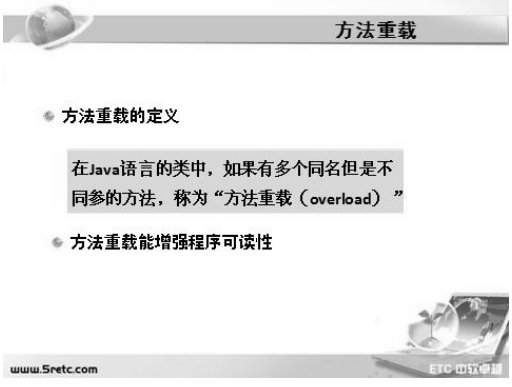
上述代码中声明了 4 个方法。其中 getSalary 是一个有返回值的方法，必须以一个带值的 return 语句返回。若方法没有返回值，则返回值类型用关键字 void 表示，如 setSalary 方法。如果方法需要参数，则需要声明形式参数，如 setName 方法的 String name 为该方法的形式参数列表，形式参数可以有多个，用逗号隔开即可。方法中的细节问题，请参见相关章节。

this 是什么意思？this 关键字，在此处代表的是当前对象的引用，由于方法的局部变量与类的属性同名，所以就必须借助 this 来区分。this.salary 的 salary 是类的属性，=后的 salary 是方法的形式参数。

至此，Employee 类的类体已经不再是空，而包含了 3 个属性、4 个方法，代码如下：

```
public class Employee {
    private String name;
    private double salary;
    private static int count;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public double getSalary() {
        return salary;
    }
    public void setSalary(double salary) {
        this.salary = salary;
    }
}
```

2.3.2 方法重载



方法重载

假设 Employee 的薪水调整还有另外一种方案，即根据员工级别，按照一定比例调整薪水，那么在 Employee 类中增加如下方法：

```
public void setSalary(char level){
    switch(level){
         case 'a':salary=salary*1.1;break;
         case 'b':salary=salary*1.2;break;
         case 'c':salary=salary*1.3;break;
    }
}
```

至此，在 Employee 类中，就有两个名字为 setSalary 的方法，只不过参数不同。在 Java 语言的类中，如果有多个同名但是不同参数的方法，称为“方法重载（overload）”。所谓不同参数，可能是形式参数的个数不同，也可能是类型不同。方法重载能够增强程序的可读性。如 Employee 类中的 setSalary(char level)方法，命名为 setSalary2 也可以，不会影响使用，但是可读性将降低。两个方法都跟薪资调整有关，不过是算法不同、参数不同，所以应该使用同样的名字，可读性将提高。

一个类中，可以出现两个同名同参但是返回值不同的方法吗？答案是否定的，这种情况会发生编译错误。因为调用方法时，是通过指定方法名和参数调用的，如果两个方法的名字和参数都相同，那么调用的时候就会混淆，所以方法的返回值是不具备区分性的。

目前，Employee 类已经完成，那么，如何使用该类的方法或属性？对于任何一个类，最终使用的时候都需要实例化该类的对象，通过对象调用相应方法，从而执行相应的功能。所以，要使用类，首先就需要掌握创建对象的方法。创建对象的语句如下：

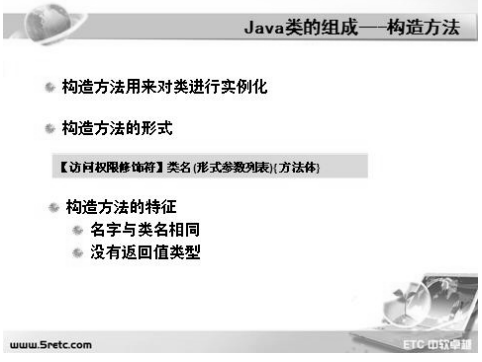
```
类名 对象名=new 类的构造方法;
```

要创建 Employee 类的对象，可由以下语句完成：

```
Employee e=new Employee();
```

其中 Employee 是要创建的对象的类型，即类名；e 是对象的名字，即引用；new 后面的 Employee()是 Employee 类的构造方法。构造方法的相关知识，参见 2.4 节。

2.4 构造方法



构造方法的声明形式及作用

构造方法重载

要想使用 Java 类，必须创建类的对象，即对类进行实例化。而创建对象就必须使用构造方法，因此，构造方法在 Java 类中有举足轻重的作用。构造方法的声明形式如下（【】代表可选项）：

```
【访问权限修饰符】类名 (形式参数列表){方法体}
```

构造方法有两个显著特征：①名字必须与类名相同，且大小写敏感；②没有返回值类型。Employee 类中无参的构造方法形式如下：

```
public Employee() {
}
```

值得注意的是，任何一个 Java 类都默认有一个如上所示的无参构造方法。也就是说，即使 Employee 类中没有声明无参的构造方法，照样可以直接使用。如对于 2.2 节展示的 Employee 类，其中没有任何构造方法的声明，但是却可以使用如下代码：

```
public static void main(String[] args) {
    Employee e=new Employee();
    e.setName("Gloria");
    e.setSalary(5000);
    System.out.println(e.getName()+e.getSalary());
}
```

上述代码首先使用 Employee 类的无参构造方法 Employee()创建一个对象，名字为 e。接下来，使用圆点调用了 e 的方法，其中 e.setName 对 e 指定了名字为 Gloria，e.setSalary 方法将 e 的薪水赋值为 5000，最后调用 getXXX 方法返回 e 的名字和薪水。打印输出结果如下所示：

```
Gloria 5000.0
```

如果希望在创建 Employee 对象时，为该对象直接指定名字，而不是使用 setName 方法指定名字，该如何实现？这种情况下，就可以在 Employee 类中声明带形式参数的构造方法。如下代码所示：

```
public Employee(String name) {
    this.name = name;
}
```

同理，也可以创建具有其他参数的构造方法，使得创建对象的同时，能够直接对名字、薪水赋值。如下代码所示：

```
public Employee(String name, double salary) {
    this.name = name;
    this.salary = salary;
}
```

需要注意的是，无参的构造方法，只有在该类没有声明任何构造方法的时候，才是默认存在的；只要声明了其他带参的构造方法，无参的构造方法将不会默认存在，而是必须声明才可以使用。至此，Employee 类的代码如下所示：

```
public class Employee {
    private String name;
    private double salary;
    private static int count;
    public Employee() {
    }
    public Employee(String name) {
        this.name = name;
    }
    public Employee(String name, double salary) {
        this.name = name;
        this.salary = salary;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public double getSalary() {
        return salary;
    }
```

```
    public void setSalary(double salary) {
        this.salary = salary;
    }
}
```

该类中声明了 3 个属性、3 个构造方法、4 个方法。可以使用如下代码测试该类：

```
public static void main(String[] args) {
    //使用无参构造方法创建对象
    Employee e=new Employee();
    e.setName("Gloria");
    e.setSalary(5000);
    System.out.println(e.getName()+" "+e.getSalary());
    //使用带一个参数的构造方法创建对象
    Employee e1=new Employee("Alice");
    System.out.println(e1.getName()+" "+el.getSalary());
    //使用带两个参数的构造方法创建对象
    Employee e2=new Employee("John",6000);
    System.out.println(e2.getName()+" "+e2.getSalary());
}
```

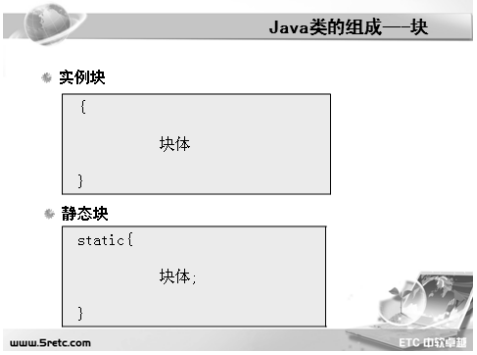
运行结果如下：

```
Gloria 5000.0
Alice 0.0
John 6000.0
```

Java 类的主要组成部分就是属性、方法、构造方法。属性用来表示对象的数据，方法用来表示对象具有的行为，构造方法用来创建对象，反之也成立，即构造方法只能用来创建对象。

类名、属性名、方法名有什么命名规范？Java 语言中的所有标识符都只能由字母、数字、下画线、$符组成，且数字不能打头。类名建议首字母大写，每个单词的首字母都大写，如 CustomerService。属性名建议第一个单词首字母小写，其他单词首字母大写，如 accountType。方法名与属性名的建议相同，如 getName。即使违反了命名规范，也不会发生编译运行错误，但是却严重影响代码的可读性。

2.5 块



块与内部类

块是在 Java 类中不太常见的一种元素。块的声明形式与方法体类似，分实例块和静态块（static 块）两种。static 的具体含义在相关章节将详细展开。

1. 实例块

实例块的声明形式如下：

```
{
        块体
}
```

在 Employee 类中增加实例块，如下所示：

```
{
        count++;
        System.out.println("创建了一个对象");
}
```

实例块不能直接调用，每次调用任何构造方法创建对象时，都会在调用构造方法前自动调用实例块的代码。所以实例块常常用来包含所有构造方法都需要执行的功能，而这些功能只在调用构造方法前执行。再次运行 2.4 节中的测试代码，结果如下：

```
创建了一个对象
Gloria 5000.0
创建了一个对象
Alice 0.0
创建了一个对象
John 6000.0
```

可见，不管使用哪个构造方法创建对象，在调用构造方法前，都默认执行了实例块代码。一个类中可以有多个实例块，将按照声明顺序被执行。

2. 静态块

静态块的声明形式如下：

```
static{
        块体;
}
```

静态块与实例块的区别在于，静态块只被运行一次，而实例块是每次创建对象都会运行。因此，静态块往往用来包含该类必须执行且只执行一次的代码。在 Employee 类中加入如下静态块：

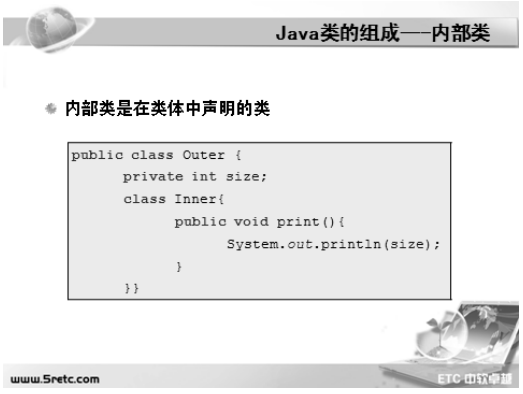
```
static{
      System.out.println("静态块");
}
```

再次运行 2.4 节的测试代码，结果如下：

```
静态块
创建了一个对象
Gloria 5000.0
创建了一个对象
Alice 0.0
创建了一个对象
John 6000.0
```

可见，虽然创建了 3 个对象，但是静态块却只运行了一次，而且在最开始处运行。

2.6 内部类



块与内部类

顾名思义，内部类（inner class）即在类体中声明的类，包含内部类的类往往被称为外部类（outer class）。内部类作为外部类的成员存在，可以像方法一样使用外部类的属性和方法。同时内部类也具有类的特征，可以声明属性、方法、构造方法等，内部类也需要创建对象才能使用类中的方法或属性等成员。

如下代码中的 Inner 类即 Outer 类的内部类：

```
Public class Outer {
    private int size;
    class Inner{
         public void print(){
              System.out.println(size);
         }
    }
}
```

内部类的具体应用，请参考第 17 章。

2.7 本章小结

Java 程序都由若干个类组成，所以先了解 Java 类的主要组成元素非常关键。本章主要学习了类中常见的 5 种元素，包括属性、方法、构造方法、块和内部类。学习本章后，读者能够对 Java 类有清晰认识，对各元素的声明形式、含义、作用也有所了解。但是 5 种元素中有很多通用的知识点，如类、属性、方法、构造方法都使用到了访问权限修饰符，属性、方法都涉及数据类型等。在第一部分接下来的章节中，将针对这些“通用”的知识点逐一展开学习。