第3章

类和对象

学习目标

通过本章的学习，理解面向对象的编程思想、类和对象的概念，学会使用面向对象的概念进行编程，能正确定义类及其中的成员变量、成员方法和构造方法，能创建相应的对象对类成员并进行调用。通过学习逐步掌握面向对象编程的方法，以达到使用面向对象技术编写 Java 程序的目的。

能力目标

- 全面了解面向对象的概念；
- 能正确定义类和对象；
- 能正确定义和使用类成员。

卡片大家都不陌生，打开钱包就会发现钱包里有很多不同功能的卡，如银行卡、电话卡、购物卡、积分卡、健身卡等，多种多样。这些卡一般都是用于终端机，如 ATM 机、电话机、POS 机等，但最终数据的处理是通过计算机来处理。现在也许会觉得很为难，但如果具备了面向对象编程的知识，或许会觉得："哦，原来就是这样啊！"

在众多卡片中，以电话卡为例，进行面向对象的初级编程。电话卡种类也众多，如校园的 201 卡、打长途的 IP 卡、能实现多种功能的一卡通等。人们拿到这些卡就可以或经济实惠或方便快捷地进行不同功能的通话了。它们的实现却也比较相似，下面看一个基本程序。

例 3-1 PhoneCard.java

```
class PhoneCard{                                    //电话卡类
    String cardNumber;                              //卡号
    private int password;                           //密码
    double balance;                                 //剩余金额
    String connectNumber;                           //连接号码
    boolean connected;                              //是否连通

    public PhoneCard(){                             //构造方法
```

```
    }
    public PhoneCard(String cn,int pw,double bl,String cnn){ //构造方法
        cardNumber = cn;
        password = pw;
        balance = bl;
        connectNumber = cnn;
        connected = false;
    }

    boolean performConnection(String cn,int pw){   //判断电话是否连通方法
        if(cn == cardNumber&&pw == password){
            connected = true;                      //电话已连通
            return true;
        }
        else{
            connected = false;                     //电话未连通
            return false;
        }
    }
    double getBalance(){                           //获取剩余金额
        if(connected)
            return balance;
        else
            return - 1;
    }
    double performDial(){                          //扣除通话费后金额
        if(connected)
            balance -= 0.5;
        return balance;
    }
    void outMessage(){                             //输出相关信息
        System.out.println("电话卡接入号码: " + connectNumber + "\n电话卡卡号: "
            + cardNumber + "\n电话卡密码: " + password + "\n剩余金额: " + balance);
    }
}

class UsePhoneCard{
    public static void main(String args[]){
        PhoneCard myCard = new PhoneCard("1234567890",1111,100.0,"201");
        myCard.outMessage();
    }
}
```

例 3-1 的运行结果如图 3-1 所示。

在例 3-1 所示的程序中，涵盖了类的定义、成员变量的定义、成员方法的定义、构造方法的定义、对象的定义和类成员的调用等。首先，第一行定义了一个用户类 PhoneCard，PhoneCard 封装了电话卡的一些静态的属性和动态的方法。例如卡号、密码、剩余金额等，都属于静态的属性，只有属性值发生变化。像 getBalance()等是属于动态的方法，是对类成员进行的操作。

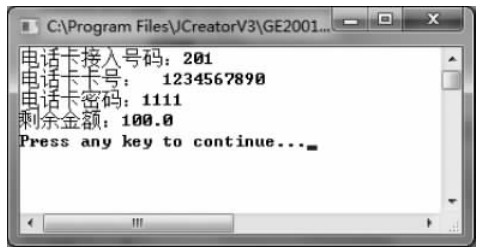

图 3-1 例 3-1 的运行结果

在 PhoneCard 类中，定义了 5 个成员变量、2 个构造方法和 4 个成员方法。在定义成员变量前，首先要对电话卡进行抽象，判断出电话卡应该有哪些成员。例如：

```
电话卡{
        卡号;
        密码;
        剩余金额;
        连接号码;
        是否接通;
        有效日期;
        ……

        连接();
        充值();
        查询余额();
        拨号();
        通话次数();
        信息查询();
        ……
}
```

根据抽象好的成员进行数据类型的判断，如卡号 cardNumber，这个卡号一般是固定的，由多个数字组成，人们不需要对卡号进行数学计算，这样卡号就可以定义为 String 型；剩余金额 balance，需要对卡中金额进行充值和消费，由于不完全是整数，所以 balance 可以定义为 float 或 double 类型；是否接通 connected 只有两个状态，即接通或未接通。这样，在定义这个变量时，就可以定义成 boolean 类型，只有 true 或 false 两个状态。其他成员变量类型的确定也可以这样分析。

在构造方法的设计中，构造方法目的是为成员变量赋值，这里写了两个构造方法：不带参数的构造方法是默认的构造方法，也就是说对成员变量赋默认的值；带参数的构造方法可以在通过对象生成具体卡片时，通过参数传递进行赋值，也就是对对象的初始化。

在成员方法的设计中，需要考虑方法应该实现的操作，如 performConnection()方法，主要是实现接入电话的操作，也就是插卡机器确认有效的工作，对此可以进行判断，如果电话卡内卡号和密码一致，则电话接通，如果任意一个不一致则不能接通。

在 UsePhoneCard 类中，只包含了一个 main()方法，main()方法是整个程序的入口，程序的运行是从 main()方法开始的。在使用面向对象编程的概念中，main()方法不再是

进行逻辑运算的方法了。在 main()方法中，主要用它来生成对象，然后使用对象来调用相应的成员方法。

3.1　类

本节主要介绍类的创建、成员变量的定义、成员方法的定义、构造方法的定义等基本概念和初级操作。

Java 程序设计就是定义类的过程。把众多的事物归纳，具有相同属性和行为方法的事物进行归类，形成一个整体，这就形成了一种事物的集合，在面向对象的编程语言中就形成了类。类是一个独立的程序单位，它应该有一个类名，并包括属性说明和行为说明两个主要部分。类与对象的关系就是整体与单一的关系，类是所有对象的集合，而对象是类的具体实例。

3.1.1　类的创建

类是具有相同属性和行为的一组对象的集合，封装了一类对象的状态和方法。类是现实世界中实体的抽象集合，封装了数据和数据操作的复杂抽象数据类型。类具有完成的功能和相对的独立性，可以包含更丰富的内涵、更好的安全性和更强大的功能。

1. 类的语法格式

定义类的语法格式如下：

```
[修饰符]  class  类名  [extends  父类名][implements  接口名]
{
        //类体
}
```

例如：

```
class Student{
//类体;
}
```

2. 类的说明

(1) []：可选项，根据类定义的需要增加相应内容。

(2) class：定义类的关键字。

(3) 修饰符：定义类可使用的修饰符有 public、final、abstract。

(4) 类名：首先要符合标识符的定义规则；其次命名要见名知意，能基本体现类的主要功能；最后类名应首字母大写，如 Person、UserPhoneCard。

(5) 类体：包含两部分——成员变量和成员方法，其中成员变量用来描述对象的属性，成员方法用来说明对象的行为。在实际使用中，可以只有成员方法，也可以只有成员变量。

(6) extends：继承关键字，用来继承父类。

(7) implements：实现关键字，用来实现接口。

提示

(1) public 修饰的类是公共的类,可以让任意类访问。

(2) final 修饰的类是最终类,不能被继承。

(3) abstract 修饰的类是抽象类,不能直接生成对象,需要通过继承才能使用。

3.1.2 成员变量

在类中定义的数据被称为成员变量或域变量。

1. 成员变量的语法格式

定义成员变量的语法格式如下:

```
[修饰符]  数据类型  变量名;
```

例如:

```
private  int  age;
```

2. 成员变量说明

(1) 变量名:命名满足标识符要求,按规则第一单词字母小写,如果由多个单词组成,则从第二单词起首字母大写。

(2) 修饰符:成员变量可使用的修饰符有 public、protected、private、final、static 等。

技巧

变量的命名一般都是由名词组成。

提示

(1) public:表示是公共的,可以让任意类访问。

(2) protected:表示是受保护的,只有同一个包中的类和其他包中的子类可以访问。

(3) private:表示是私有的,只有本类的成员才能访问,也不能被继承。

(4) final:表示是最终变量,即常量,值不能被改变。

(5) static:表示是静态变量,定义同时分配存储空间,可以类名直接调用该变量。

例如,定义一个人类,首先我们要对人类进行抽象,看看人类有哪些共有的属性和方法。

```
人{
    姓名;
    性别;
    年龄;
    身高;
    体重;
    家庭住址;
    ……
    学习();
    工作();
    运动();
    ……
}
```

人们在抽象时可以根据需求先列举出一些共有的属性和方法，再根据属性的取值不同，对相应的属性添加数据类型，如例 3-2 所示。

例 3-2　人.java

```
public class 人{
      String 姓名;
      String 性别;
      int 年龄;
      double 身高;
      double 体重;
}
```

根据编程需要，一般在程序中变量等命名尽量用英文单词或字母来表示，所以把例 3-2 再进行一下改编，如例 3-3 所示。

例 3-3　Person.java

```
public class Person{
      String name;
      String sex;
      int age;
      double height;
      double weight;
}
```

该类的类体中只有变量定义而无方法定义，即该类仅描述了人的属性而没有描述人的行为，下面看方法的定义。

3.1.3　成员方法

在类中对数据处理的代码被称为成员方法。

1. 成员方法的语法格式

定义成员方法的语法格式如下：

```
[修饰符]  返回值类型  方法名([参数列表])
{
    //方法体
}
```

例如：

```
public void setName(String n){
    name = n;
}
```

2. 成员方法说明

(1) 方法名：命名满足标识符要求，按规则第一个单词字母小写，如果由多个单词组成，则从第二个单词起首字母大写。

(2) 修饰符：成员方法可使用的修饰符有 public、protected、private、final、static 等。

(3) 返回值类型：无返回值类型使用的关键字是 void，其他有返回值的为返回值数据类型的关键字。

技巧

方法的命名一般第一个单词为动词，如 getName()、outMessage()等。

提示

(1) final：表示是最终方法，即方法不能被覆盖。

(2) static：表示是静态方法，方法中使用的成员变量必须是静态变量，可以用类名直接调用该方法。

例 3-4 Person1.java

```
public class Person1{
    String name;
    String sex;
    int age;
    double height;
    double weight;

    public void setName(String n){      //带 String 类型参数 n
        name = n;                       //使用参数 n 给成员变量 name 赋值
    }
    public String getName(){            //返回值类型为 String 型
        return name;                    //返回 name 的值，成员变量 name 的类型为 String 型
    }
    public void study(){
        System.out.println("好好学习，天天向上!");
    }
    public void outValue() {
        System.out.println("姓名:" + name);
        System.out.println("性别:" + sex);
        System.out.println("年龄:" + age);
        System.out.println("身高:" + height);
        System.out.println("体重:" + weight);
    }
}
```

在这部分程序片段中，void 表示无返回值，一般在给变量赋值或输出时，使用 void 定义方法。如果方法运行完，需要返回一个值，则返回值类型与 return 后面表达式的类型一致。

例 3-5 Circle.java(圆类)

```
class Circle
{
    double radius = 1.0;                //成员变量
    double CalculateArea()              //成员方法
    {
        return radius * radius * 3.14159;
    }
}
```

3.1.4　构造方法

构造方法是一种特殊的成员方法，方法名与类名一致，且构造方法无返回值类型。构造方法的目的是用类创建对象时为对象的各成员变量提供初值，也就是对对象进行初始化的一类方法。

1. 构造方法语法格式

定义构造方法的语法格式如下：

```
[修饰符]  方法名()
{
    //方法体
}
```

2. 构造方法说明

(1) 修饰符：可以使用 public、protected、private 和默认。

(2) 方法名：方法名与类名相同。

例 3-6　Circle1.java

```
public class Circle1{
    double radius;
    public Circle1 (double x){    //构造方法
        radius = x;
    }
    public double CalArea(){
        return 3.14 * radius * radius;
    }
}
```

程序的第 3 行定义了一个带参数的构造方法，通过外部参数向形参 x 赋值，再将 x 的值传递给 radius，实现对成员变量的赋值。

在没有定义构造方法的类中，有默认的构造方法，是对所有的成员变量赋默认的值。如果显式定义了构造方法，则默认的构造方法就取消了；如果想使用默认的构造方法，则需要显示定义一下。具体示例如下。

例 3-7　Circle2.java

```
public class Circle2{
    double radius;
    public Circle2 (){          //默认的构造方法显式定义
        radius = 0;
    }
    public Circle2 (double x){
        radius = x;
    }
    public double CalArea(){
        return 3.14f * radius * radius;
    }
}
```

在本例中定义了两个构造方法，一个是不带参数的，一个是带参数的。在方法中定义了多个同名不同参数的方法叫作方法的重载。

例 3-8 Person2.java

```
public class Person2{
    String name,sex;
    int age;
    double height,weight;

    Person2(){              //显式定义默认的构造方法
    }
    Person2(String n){      //带 1 个参数的构造方法
        name = n;
    }
    Person2(String n,String s,int a,double h,double w){   //带 5 个参数的构造方法
        name = n;
        sex = s;
        age = a;
        height = h;
        weight = w;
    }
    public void setName(String n){
        name = n;
    }
    public void outValue() {
        System.out.println("姓名:" + name + "\n 性别:" + sex
                           + "\n 年龄:" + age + "\n 身高:" + height
                           + "\n 体重:" + weight);
    }
}
```

本例中定义了 3 个构造方法，根据参数的不同，可以定义任意多的构造方法，但在一般情况下应至少定义两个构造方法，一个是不带参数的，一个是带全部参数的。

本例中出现的 setName()方法，方法体与构造方法一致，但这不是对构造方法的简单重复，构造方法的使用仅仅用于创建对象或给对象赋值时使用，而 setName()方法可在对象生成后由对象多次调用。

3.2 对象

在现实世界中，对象是指一个具体的个体，拥有某种特征和行为。任何事物都可以称为对象，如一个人、一辆车、一张卡、一个账号、一栋房子等。每一个对象都拥有自己的属性和行为。

在面向对象的程序设计中，对象是类的实例，类是对象的集合。对象与类的关系就像变量与数据类型的关系一样。对象使用数据和方法描述它的状态和行为。

3.2.1 对象的创建

在 Java 程序中，创建对象包括声明对象和为对象分配内存空间两部分。

1. 创建对象的语法格式

创建对象的语法格式如下：

```
类名　对象名;                                    //方法一
对象名 = new　类名([实际参数列表]);
类名　对象名 = new　类名([实际参数列表]);　//方法二
```

例如：

```
Circle2　c1;
C1 = new Circle2();　　//调用不带参数的构造方法
```

例如：

```
Person2　p1 = new Person("张三","男",18,175.0,58.5);
                        //调用带 5 个参数的构造方法
```

2. 创建对象说明

(1) 类名：已经存在的类，可以是系统类库中的类，也可以是用户自定义的类。

(2) 对象名：命名满足标识符要求，按规则第一单词字母小写，如果由多个单词组成，则从第二单词起首字母大写。

(3) new 关键字：使用 new 关键字为对象分配存储空间。

(4) 类名([实际参数列表])：在类中定义的构造方法，与使用 new 为对象分配存储空间时，系统自动调用该类的构造方法，实现对对象的初始化。实际参数列表是具体的值，要与构造方法中的参数类型一一对应。

注意

一个类可以创建无数个对象，每个对象占用不同的内存空间，它们之间是相互独立的，互不影响。

3.2.2　使用对象

创建对象的过程就是为对象分配内存空间的过程，对象一旦拥有了自己的内存空间，就可以调用所属类中的成员方法和成员变量，可以通过使用“.”运算符实现对变量和方法的访问。

访问对象的语法格式如下：

```
对象名.变量名
对象名.方法名([实际参数列表])
```

例 3-9　求半径为 2 和 5 的圆面积

```
public class Circle3{
    private double radius;
    final double PI = 3.14159;　　//常量 PI
    public Circle3(){
        radius = 0;
    }
```