Lab3 面向对象

学习目标：

1.理解面向对象的概念，理解类与对象的区别

2.掌握类的声明和实例化方法

3.掌握方法以及构造方法的定义

4.理解方法的重载

一、面向对象简介

面向对象不仅是一项具体的软件开发技术，也是一种符合人类思维习惯的编程思想。现实生活中存在各种形态不同的事物，这些事物之间存在着各种各样的联系。面向对象编程（Object-Oriented Programming，OOP）就是利用对象建模技术来分析目标问题，抽象出相关对象的共性，并对共性进行分类及分析各类之间的关系，同时使用类来描述同一类问题。

面向对象中类的定义充分体现了抽象数据类型的思想，基于类的体系结构可以把程序的修改局部化，特别是一旦系统功能需要修改时，只要修改类中间的某些操作，而类所代表的对象基本不变，保持整个系统仍然稳定。

二、类与对象

面向对象的编程思想力图使程序对事物的描述与该事物在现实中的形态保持一致，为了做到这一点，在面向对象的思想中提出了两个概念，即类和对象。

类是对某一类事物的抽象描述，对象是该类事物的某一个实体，对象会被分配物理内存。如图 3-1 所示，以人类为例，每个人都可以看作是一个对象。类是描述多个对象的共同特征，是具有相同特性（属性）和行为（方法）的一组对象的集合，例如人类需要描述的特征和行为包括：名字、年龄、身高以及说话、唱歌等。对象用于描述现实中的实体，对象是类的实例化，例如“李雷”是一个具体的人，是一个对象，应该具有人类的属性和方法等。

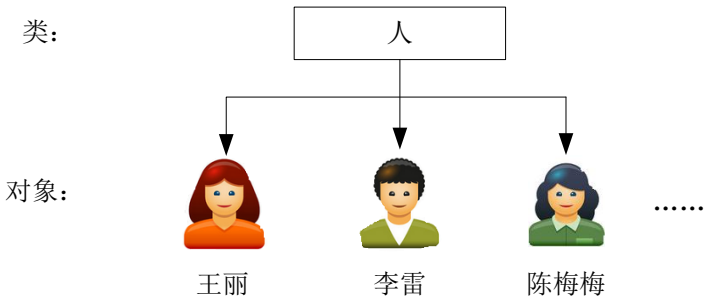


图 3-1 类与对象

三、类的声明

为了在程序中创建对象，首先需要声明一个类，用于描述一组对象的特征和行为。类中可以定义字段、属性和方法等成员。定义在类中的变量称为字段，字段用于在类中存储数据，属性用于描述对象的特征，而方法用于描述对象的行为。

【注意】如果类的声明中没有指定字段的初始值，使用对象时也没有给字段赋值，则编译时会自动赋予其类型的默认值，并发出警告。声明类的语法格式如下：

[访问修饰符] class 类名称 [：基类或接口]

{

类成员定义

}

【实例】声明一个 Person 类。

```
class Person
{
    private string _name;
    public string Name
    {
        get{return this._name;}
        set{this._name = value;}
    }
    public void Speak()
    {
        Console.WriteLine("大家好，我是" + this._name);
    }
}
```

【分析】本实例中，Person 是类名，_name 是字段，Name 是属性，Name 属性封装了_name 字段，Speak()是方法。

四、对象的创建与使用

1.对象的创建

在 C#中可以使用 new 关键字来创建对象，创建对象的语法格式如下：

类名 对象名称 = new 类名();

例如创建 Person 类的实例，代码如下：

Person p= new Person ();

其中“Person p”是声明了一个 Person 类型的变量 p，而“new Person()”用于创建 Person 类的一个实例对象，中间的等号用于将 Person 对象在内存中的地址赋值给变量 p，这样变量 p 就有了 Person 对象的引用。

2.对象的使用

在创建 Person 对象后，可以通过对象的引用来访问对象所有的成员，语法格式如下：

对象引用.对象成员

例如：p.Speak();

五、方法

在类中自定义的“函数”称为“方法”。方法是表示实现类功能而执行的计算或操作。

1.方法的定义与调用

每个方法都有一个名称和一个主体。方法名应该是一个有意义的标识符，描述出方法的用途；方法主体包含了调用方法时实际执行的语句。定义方法的语法格式如下：

[访问修饰符] 返回值类型 方法名（参数列表）

{

方法体

[return 返回值;]

}

上述语法格式中，需要注意以下几点：

（1）方法的返回类型是指调用方法后返回的值的类型，如果没有返回值，则为 void。如果有返回值，则在语句序列中必须使用 return 语句返回一个值，这个值的类型必须与返回类型一致。

（2）参数可有可无，如果有参数，每个参数都应指定数据类型和参数名，参数之间用逗号隔开。方法定义时参数列表中定义的参数称为形式参数（形参），而实际调用时传递的参数

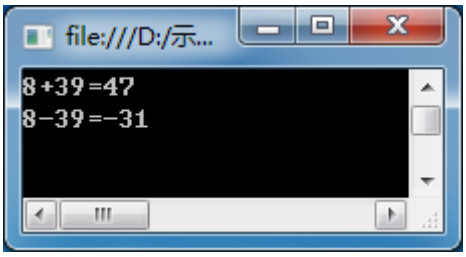
称为实际参数（实参）。形参和实参一一对应，个数和数据类型必须一致。

（3）方法调用有 3 种方式：①在同一个类中，方法可以直接调用；②在其他类中的方法，需要通过类的实例进行调用；③静态方法需要通过类名进行调用。

【实例】通过定义方法 Add()和 Subtract ()实现加减法运算。

程序代码如下：

```
namespace Method
{
    class AddAndSubtract
    {
        static void Main(string[] args)
        {
            int num1 = 8;
            int num2 = 39;
            Console.WriteLine("{0}+{1}={2}", num1, num2, Add(num1, num2));
            Console.WriteLine("{0}-{1}={2}", num1, num2, Subtract(num1,num2));
            Console.ReadLine();
        }
        //定义方法（加法）
        static int Add(int x, int y)
        {
            return x + y;
        }
        //定义方法（减法）
        static int Subtract(int x, int y)
        {
            return x - y;
        }
    }
}
```



【分析】本实例中，定义了 Add()方法和 Subtract ()方法实现两个整型参数的加法和减法运算，返回值均为整型。在同一个类的方法中，直接调用 Add()方法和 Subtract()方法，传入实参 num1 和 num2 的值，求出返回值并输出。

2.方法的重载

方法重载是一种操作性多态。当需要在多个不同的实现中对不同的数据执行相同的逻辑操作时，就可以使用重载，例如，Console 类的 WriteLine()方法具有 19 个重载。

方法重载指在同一个类中创建多个同名的方法，但这些方法的参数互不相同，可以是参数类型不同，也可以是参数个数不同。决定方法是否构成重载的三个条件如下：

（1）在同一个类中；

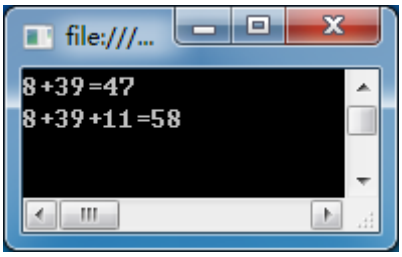
（2）方法名相同；

（3）参数列表不同。

【实例】通过方法重载分别实现对两个整数和三个整数的加法程序。

程序代码如下：

```
namespace Method
{
    class Overload
    {
        static void Main(string[] args)
        {
            int num1 = 8;
            int num2 = 39;
            int num3 = 11;
            Console.WriteLine("{0}+{1}={2}", num1, num2, Add(num1, num2));
            Console.WriteLine("{0}+{1}+{2}={3}", num1, num2, num3,Add(num1, num2,num3 ));
            Console.ReadLine();
        }
        //实现两个整数相加
        static int Add(int x, int y)
        {
            return x + y;
        }
        //实现三个整数相加
        static int Add(int x,int y,int z)
        {
            return x+y+z;
        }
    }
}
```



【分析】在本实例中，在 Overload 类中定义了两个求和方法，方法名都是 Add，但一个 Add()方法有 2 个参数，一个 Add()方法有 3 个参数，即参数列表个数不同，因此实现了方法的重载。在调用 Add()方法时，通过输入不同的参数，执行不同的方法体。

3.构造方法

构造方法是类的一个特殊成员，它会在类实例化对象时自动调用，为对象开辟内存空间，并对类中的成员进行初始化。

在 C#中的每个类至少有一个构造方法，当声明一个类时，如果没有定义构造方法，系统会自动添加一个默认的没有参数的构造方法，在其方法体中没有任何代码，即什么也不做。反之，如果声明类时定义了构造方法，系统将不会自动添加默认的构造方法。

在一个类中定义构造方法，必须满足以下 3 个条件：

（1）方法名与类名相同；

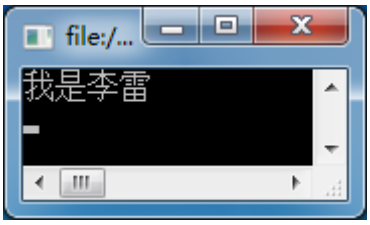
（2）在方法名的前面没有返回值类型的声明；

（3）在方法中不能使用 return 语句返回一个值。

【实例】在类中定义一个有参数的构造方法。

程序代码如下：

```
namespace ConstructionMethod
{
    class Program
    {
        static void Main(string[] args)
        {
            Person p = new Person("李雷");
            p.Speak();
            Console.ReadLine();
        }
    }
    public class Person
    {
        private string _name;
        public string Name
        {
            get{return _name;}
            set{_name = value;}
        }
        //定义有参数的构造方法
        public Person(string name)
        {
            Name = name;
        }
        public void Speak()
        {
            Console.WriteLine("我是" +_name);
        }
    }
}
```



【分析】在本实例中，在定义 Person 类时声明了一个有参数的构造方法，参数是一个 string 类型变量，因此，在主方法中实例化 Person 时，需要传入一个 string 类型的实参。