第3章　类和对象

本章要点

- 类和对象的基本概念
- 对象及类的创建和使用
- 封装

3.1　类

在Java中，类是基本的构成要素。类是具有相同属性和行为的对象的集合，对象由类创建。在类中表示的对象或实体的特征称类的属性。对象执行的操作称为类的方法。

3.1.1　类和对象的区别

简单地说，类是一个抽象的概念，为对象定义了抽象的属性和行为，对象是真实的实体，是类的具体化、实例化。表3.1给出了类和对象的示例。

表3.1　类和对象的示例

类	对象
人	张三
	李四
水果	苹果
	香蕉
蔬菜	白菜
	萝卜

3.1.2　类的定义

在Java中类一般包括关键字class、类名、类的属性和类的方法。其中类的关键字class用小写格式，类名要符合Java编程规范。一般定义一个类有以下几个步骤。

1. 定义类名

在Java中，类名符合其编程规范即可，框架如下：

```
public class 类名 {
//类的属性
//类的方法
}
```

2. 编写类的属性

类的属性即类的数据成员，在类中通过定义成员变量来说明类的属性特征。

3. 编写类的方法

类的方法主要实现类的功能。简单地说，方法就是实现特定功能的一段程序代码。下面通过一个例子说明。

例3.1　类的定义示例。

```
public class School {
    //定义 school 的属性
```

```
    String schoolName;//学校的名称
    String schoolLocation;//学校的位置
    int stuNum;//在校生人数
    int collegeNum;//院系数目

    //定义 school 的方法
    public String toString() {
        return schoolName+"坐落于"+schoolLocation+"有"+collegeNum+"所院系"
                +"在校生"+stuNum;
    }
}
```

例 3.1 定义了一个 school 类，具有 schoolName、schoolLocation、stuNum、collegeNum 这 4 个成员变量和一个 toString()方法。该方法能够显示学校的基本信息。toString()方法是 Object 类中定义好的一个方法，可以返回一个字符串，在碰到 println 之类的输出方法时会自动调用，不用显示出来。

在 MyEclipse 中可以很方便地创建一个类。首先需要创建一个基本的项目；然后右击项目名称，在快捷菜单中选择 New|Class，然后在弹出的对话框中输入类名，单击 Finish 按钮完成，如图 3.1 所示。

图 3.1　新建类对话框

在对话框中输入相应的类名，完成之后或自动创建一个类的框架，在上述对话框中如果选中 public static void main(String[] args)复选框还可以自动创建一个 main()方法，如图 3.2 所示。

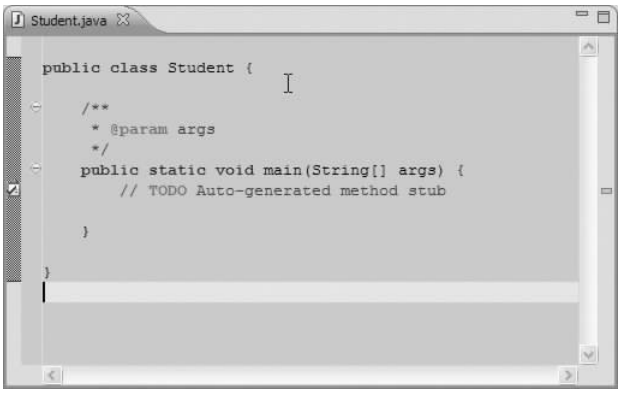

图 3.2　包含 main()方法的类框架

3.2　对　　象

面向对象（Object-Oriented,OO)的核心思想就是对象。对象就是现实中的实体,就有某种状态和行为,在 Java 中万事万物都可以作为对象,程序就是对象的组合。

在程序中,对象就是类的实例化,即类中所定义的变量和方法的集合。其中对象的特征,构成了对象属性的集合,对象执行的操作构成了对象的行为。

3.2.1　封装

对象有属性和方法,它们共同组成了对象的实体。将对象的属性和方法组合起来称为“封装”。封装可以实现信息的隐藏。

在使用对象时,只需要了解对象的外部特征,如功能等,而不需要关心其实现的细节。在程序设计中,对象就是方法和属性的集合。属性就是对象的状态,方法实现对象的行为。

3.2.2　对象的创建和使用

对象在使用之前首先要进行定义对象,为对象分配内存空间,初始化对象的成员变量的操作,即创建对象操作。

1. 对象的创建

创建一个对象需要定义、实例化和初始化三个步骤,格式如下:

```
类名　对象名=new 类名();
```

其中,由类生成对象称作类的实例化,一个类可以生成多个对象。类可以是 Java 类库中的系统类,如 String 类、Thread 类,也可以是用户自定义的类;创建类的对象用 new 关键字,它可以创建新的对象并且为对象分配内存空间;new 后面跟着类的构造方法为对象进行初始化。如果一个类没有显式声明任何构造方法,Java 平台自动提供一个没有参数的构造方法,这是一个默认的构造方法。实际上实例化的过程是用 new 调用对象的构造方法,返回一个对象的引用。在例 3.1 中已经定义好了一个 School 类,这时就可以由该类来创造对象。例如:

```
School sd=new School();
```

上述语句调用 School 类的构造方法实例化对象 sd，School()方法在本例中是默认的构造方法。

2. 对象的使用

对象的使用包括对象的成员变量和方法的调用，通过运算符“.”可以实现对变量的访问和方法的调用，使用格式如下：

```
对象名.变量;
对象名.方法名();
```

例如，对 sd 对象的变量赋初值，调用 sd 对象的 toString()方法语句如下：

```
sd.schoolName="聊城大学";                    //属性赋值初始化
sd.toString();                               //调用方法
```

下面来编写一个程序测试一下例 3.1 建立的 School 类。

例 3.2 对象的定义和使用。

```
public class SdSchool {
    /**
     * @param args
     */
    public static void main(String[] args) {
        //TODO Auto-generated method stub
        School sd=new School();//建立一个 School 的对象
        //对象初始化
        sd.schoolName="聊城大学";//学校名称
        sd.schoolLocation="山东省聊城市";//学校位置
        sd.collegeNum=28;//院系数目
        sd.stuNum=29000;//在校生人数
        System.out.println(sd);
      }
}
```

程序运行结果如图 3.3 所示。

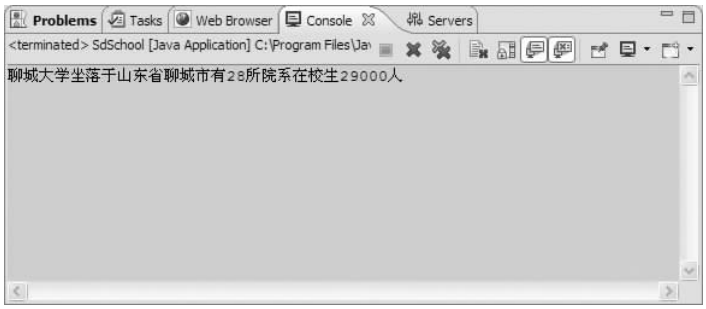

图 3.3 例 3.2 运行结果

3.3 类的方法

3.3.1 类方法的定义

类的方法实现了类的功能。当创建一个类的对象之后，就可以调用该类的方法实现相应的功能。方法一般必须有返回值类型、方法名和方法体三部分，格式如下：

```
修饰符 返回值类型 方法名(){
//方法体
}
```

其中的修饰符就是规定方法的特征，如访问控制 public、private 等。

1. 返回值类型

方法的返回值类型可以是简单变量也可以是对象，如果方法有返回类型就需要用 return 关键字返回值。用法如下：

```
return 表达式;
```

如果没有返回值，就用 void 表示。

2. 方法名

方法名要符合 Java 的命名规则，一般和变量的命名相同。方法名后的括号不能少。

3. 方法体

方法名后的大括号中的内容就是方法体。简单地说就是一段程序代码，包含变量的定义和执行语句。

3.3.2 类方法的调用

方法定义完就可以调用。调用格式和 3.2.2 节中的对象的方法调用一样，即当类的对象要实现相应的操作时就可以调用对象的方法实现。

例 3.3 类方法定义和调用示例。

```
//成绩计算的类代码
public class ScoreCal {
    int finalScore;                              //期末成绩
    int peacetimeScore;                          //平时成绩

    //计算综合成绩
    public double calGeneralScore() {
        double general=finalScore * 0.8+peacetimeScore * 0.2;
        return general;
    }

    //显示最后综合成绩
    public void showGeneralScore() {
```

```
        System.out.println("综合成绩是: "+calGeneralScore());
    }
}
//测试类代码
import java.util.Scanner;

public class TestScoreCal {

    /**
     * @param args
     */
    public static void main(String[] args) {
        //TODO Auto-generated method stub
        ScoreCal s=new ScoreCal();
        /*输入期末和平时成绩 */
        Scanner input=new Scanner(System.in);
        System.out.println("输入期末成绩: ");
        s.finalScore=input.nextInt();
        System.out.println("输入平时成绩: ");
        s.peacetimeScore=input.nextInt();
        /*计算并输出综合成绩 */
        s. showGeneralScore();
    }
}
```

程序运行结果如图 3.4 所示。

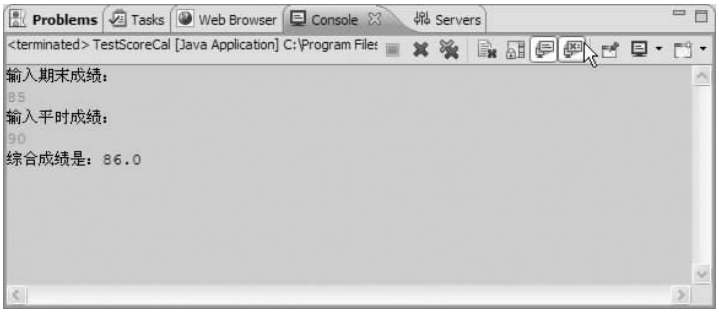

图 3.4　例 3.3 运行结果

程序说明:

(1) 在成绩计算类 ScoreCal 中定义了一个 calGeneralScore()方法和 showGeneralScore()方法,分别完成总成绩的计算和成绩显示功能。

(2) 在测试类 TestScoreCal 的测试类中调用 ScoreCal 类中的 showGeneralScore(),在使用该方法之前要先创建 ScoreCal 类的对象 s。

(3) 在一个类中允许方法的直接调用,在例 3.3 中 showGeneralScore()方法就调用了该类中的 calGeneralScore()方法。

3.4 包

在 Java 程序设计中经常需要调用，为了使程序结构清晰需要将源程序放到不同的文件夹中。在编写程序的过程中可能会由于类的命名等发生冲突，Java 为了解决这个问题就提出了“包”的概念。“包”就是区别类名空间的机制，提供了类的组织方法。Java 的类文件就可以存储在不同的包中。这样 Java 中的方法名和变量名就可以使用包含包名、类名来命名，从而减少命名冲突。简单地说 Java 中的包就相当于 Windows 的文件夹。

在 Java 中可以用包来保护类、方法和数据，是一种封装手段，限定了类中的方法和变量的作用域。同时有效地防止了命名的冲突，方便了类文件的使用和管理。

3.4.1 包的创建

在 Java 中创建包，只需把 package 命令作为 Java 源文件的第一条语句即可。语法格式如下：

```
package 包名;
```

在一个类文件中只允许有一条包声明语句。package 语句指明了存放 Java 类的命名空间。Java package 的命名习惯全部采用小写字母，通常使用网络组织域名的逆序。例如，域名为 lc.edu.cn，在程序中计划创建一个 simple 的程序库，则 package 的名称为：

```
package cn.edu.lc.simple;
```

在 MyEclipse 中可以方便地创建 Java 中的包。

(1) 在 Java 的项目文件上右击，在弹出的快捷菜单中依次选择 New|Package，然后在对话框中输入计划创建的包名，单击 Finish 按钮即可创建，如图 3.5 所示。

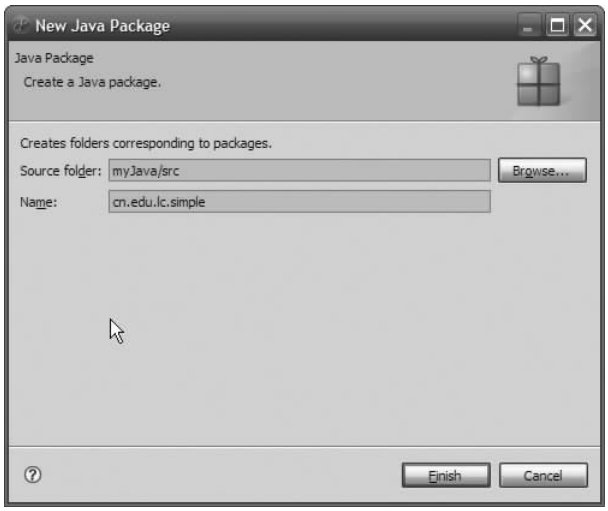

图 3.5 新建包

创建完包之后，就可以在包中创建类。

(2) 右击新建的包,然后在弹出的快捷菜单中选择 New|Class 命令,输入相应的类名,如 School 类,包名会自动添加。在包资源管理器中效果如图 3.6 所示。

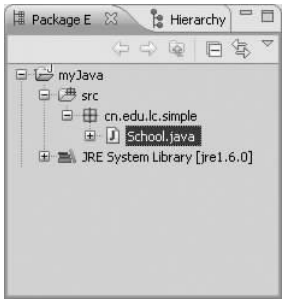

图 3.6 在包中创建类

此时 School 类就在 cn. edu. lc. simple 包中,在类文件中包的声明语句会自动加上。

(3) 在 MyEclipse 中支持在创建类的同时创建包。这时,只需先完成项目创建,然后创建类的同时输入包名即可,如图 3.7 所示。

之前在创建类时,使用的是默认的包,这时容易出现命名空间的冲突问题。因而在创建类时,尽量使用包来分类存放。

在 Java 中创建的包,实际就是一个目录结构。之前建立的包 cn. edu. lc. simple 实际就是文件系统中的 cn\edu\lc\simple。在 MyEclipse 的导航器中能很清楚地看见这个目录结构,如图 3.8 所示。

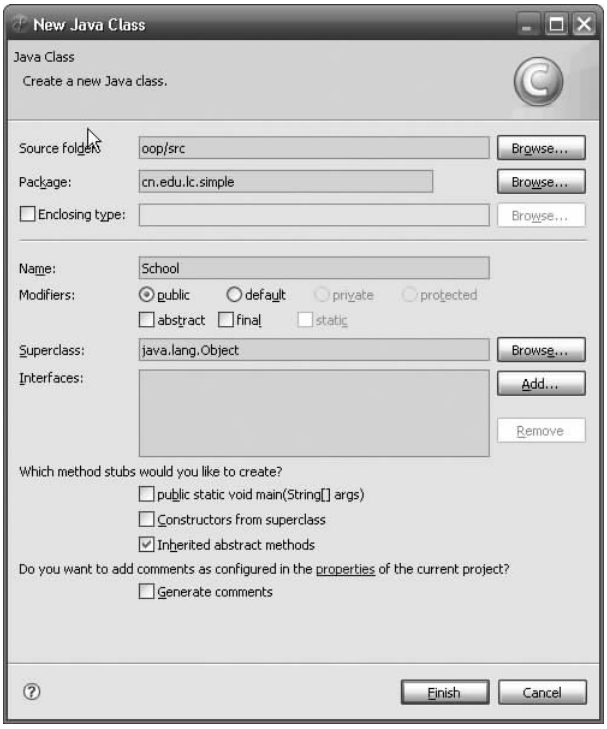

图 3.7 创建类同时创建包

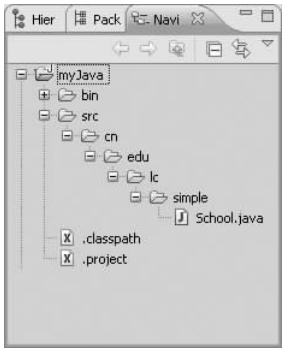

图 3.8 包的目录结构

3.4.2 包和类的导入

在 Java 中如果要使用不在同一个包中的类,就需要先将此类或包含此类的包导入。在 Java 中用 import 语句导入包或者类,形式如下:

```
import 包名.类名;
```

import 为 Java 关键字,多个包名和类名间用. 分隔。包可以是系统提供的包,也可以是

自己建的。如果用到一个包下的多个类，可以用 * 导入整个包，形式如下：

```
import 包名.*;
```

在 MyEclipse 中如果程序所需要包或者类没有导入，则会提示编译错误。此时可以使用菜单命令或者 Ctrl＋Shift＋O 键将相应的包或者类添加到程序中，同时也可以将多余不用的包移除。

3.5 封　　装

封装也称为信息隐藏，是指利用抽象数据类型将数据和基于数据的操作组合在一起，使其构成一个不可分割的独立实体，数据被保护在抽象数据类型的内部，只保留一些对外接口使之与外部发生联系。

用户无须知道其内部方法的实现细节，但可以根据类提供的方法取值和赋值。具体实现就是将类的属性私有化，限制其他成员对属性的直接访问，同时对每个私有属性提供一个公共的赋值和取值方法来访问。

例 3.4　类的封装性示例。

```
package cn.edu.lc.simple;

public class Stu {
    private String name;
    private int age;

    public String introduce() {
        return "我的名字是："+name+",今年"+age+"岁。";
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name=name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age=age;
    }
}
```

在例 3.4 中类 Stu 的属性 name 和 age 用关键字 private 修饰后，除了 Stu 类外，其他的类都不能访问。为 name 和 age 增加了一对公共的取值 get()和赋值 set()方法后，就可以使用这对方法的访问属性。

例 3.5 通过调用例 3.4 中的 set()方法实现赋值操作。

例 3.5 利用 set()方法对私有属性赋值示例。

```
package cn.edu.lc.simple;

public class StuTest {

    /**
     * @param args
     */
    public static void main(String[] args) {
        //TODO Auto-generated method stub
        Stu st=new Stu();
        st.setAge(20);
        st.setName("王华");
        System.out.println(st.introduce());
    }
}
```

图 3.9 例 3.5 运行结果

程序运行结果如图 3.9 所示。

在 Java 中封装主要是为了实现细节的隐藏，将类的数据和类的方法包装起来。在 Java 中实现封装的方法就是访问权限的控制，指明哪些客户端可用，哪些不可用。常用的方法就是将类的属性私有化，然后提供一个取值和赋值的方法对属性操作。在例 3.4 中对属性的取值和赋值没有限制，在实际使用中，会加上对取值和赋值方法的限制，从而降低类之间的耦合度，实现属性的私有化，提高安全性。例如在例 3.4 的基础上可以重写 setAge()方法，在其中可以加上判断、条件等过程。由于实现了信息的隐藏，因此不会影响其他类的调用。

例 3.6 重写对 set()方法示例。

```
package cn.edu.lc.simple;

public class Stu1 {
    private String name;
    private int age;

    public String introduce() {
        return "我的名字是:"+name+",今年"+age+"岁。";
    }

    public String getName() {
        return name;
    }
```