第5章

面向对象编程(二)

建议学时：2～4。

本章详细讲解面向对象的基本概念。针对面向对象的应用，详细讲解一些比较高级的概念。首先讲解静态变量、静态函数、静态代码块，然后讲解封装、包和访问控制修饰符，最后讲解类中类的使用。

5.1 静态变量和静态函数

5.1.1 静态变量

一个类可以实例化很多对象，各个对象分别占据自己的内存，示例代码如下。

StaticTest1.java

```
class Customer{
    String name;
}

public class StaticTest1 {
    public static void main(String[] args) {
        Customer zhangsan = new Customer();
        zhangsan.name = "张三";
        Customer lisi = new Customer();
        lisi.name = "李四";
    }
}
```

在 main 函数中定义了 zhangsan、lisi 两个对象，这两个对象具有不同的成员变量——name，内存示意图如图 5-1 所示。

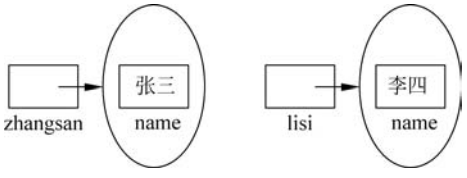

图 5-1 两个对象具有不同的成员变量

如果要保存 zhangsan 和 lisi 乃至 Customer 类中所有对象共有的信息，如该类用在某个银行系统中，要保存其所在的银行名称(如香港银行)，而每个对象的银行名称都一样，应如何实现？

此时,如果代码写成如下。

StaticTest2. java

```
class Customer{
    String name;
    String bankName;
}

public class StaticTest2 {
    public static void main(String[] args) {
        Customer zhangsan = new Customer();
        zhangsan.name = "张三";
        zhangsan.bankName = "香港银行";
        Customer lisi = new Customer();
        lisi.name = "李四";
        lisi.bankName = "香港银行";
    }
}
```

内存情况如图 5-2 所示。

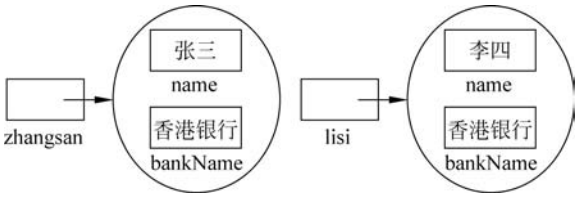

图 5-2　内存情况

同样的信息保存了两次,浪费空间。如果以后要改银行名称,需要一个一个地改,很麻烦。

在本例中能否让各对象共有的内容只用一个空间保存呢? 可以,只要将 bankName 定义成静态变量即可,方法是在其定义前加上关键字 static,代码改写如下。

StaticTest2. java

```
class Customer{
    String name;
    static String bankName;
}

public class StaticTest2 {
    public static void main(String[] args) {
        Customer zhangsan = new Customer();
        zhangsan.name = "张三";
        zhangsan.bankName = "香港银行";
        Customer lisi = new Customer();
        lisi.name = "李四";
        System.out.println("lisi.bankName = " + lisi.bankName);
```

```
    }
}
```

运行代码，控制台打印结果如图 5-3 所示。

在以上代码中，main 函数调用之后的内存情况如图 5-4 所示。

图 5-3　StaticTest2.java 的运行结果

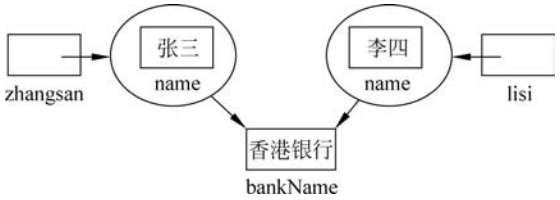

图 5-4　main 函数调用后的内存情况

注意

（1）静态变量可以通过“对象名.变量名”来访问，例如“zhangsan.bankName”，也可以通过“类名.变量名”来访问，例如“Customer.bankName”。一般情况下推荐用“类名.变量名”的方法访问，而非静态变量是不能用“类名.变量名”的方法访问的。

（2）从底层讲，静态变量在类被载入时创建，只要类存在，静态变量就存在，不管对象是否被实例化。

5.1.2　静态变量的常见应用

下面讲解静态变量的几个常见应用。

1. 保存跨对象信息

对象的通信是比较复杂的，例如，登录 QQ 时在登录界面中输入账号、密码，单击“登录”按钮，若登录成功到达聊天界面，如图 5-5 所示。那么聊天界面如何知道登录界面中输入的账号呢？

图 5-5　登录 QQ

有很多方法可以解决这个问题,其中有一种比较简单的方法,可以定义一个类,用静态变量保存登录账号,代码如下。

```
class Conf{
    static String loginAccount;
}
```

在登录界面中,如果登录成功,则将账号存入 Conf. loginAccount。在聊天界面中,访问 Conf. loginAccount 即可得到登录的账号。

2. 存储对象个数

有时候需要保存一个类已经实例化的对象个数。例如,某游戏是多人探险的游戏,在游戏的过程中有人阵亡,当存活的人数不足 3 人时屏幕上显示报警提示。那么,如何让系统知道当前存活的人数呢? 此时可以将当前存活的人数定义为静态变量,代码如下。

StaticTest3. java

```
class Person{
    String name;
    static int number = 0;
    Person(String name){
        this.name = name;
        System.out.println("创建了" + name);
        number++;
    }
    void die(){
        System.out.println(name + "阵亡");
        number -- ;
        if(number < 3){
            System.out.println("警告!不足 3 人");
        }
    }
}
public class StaticTest3 {
    public static void main(String[] args) {
        Person p1 = new Person("张三");
        Person p2 = new Person("李四");
        Person p3 = new Person("王强");
        Person p4 = new Person("赵海");
        p3.die();
        p1.die();
    }
}
```

运行代码,控制台打印结果如图 5-6 所示。

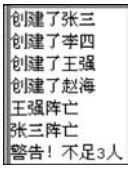

图 5-6　StaticTest3. java 的运行结果

5.1.3 静态函数

有静态变量就有静态函数，静态变量和静态函数统称为静态成员。静态函数就是在普通函数的定义前加上关键字 static，示例代码如下。

StaticTest4.java

```
class Customer{
    String name;
    static String bankName;
    static void setBankName(String bankName){
        Customer.bankName = bankName;
    }
}

public class StaticTest4 {
    public static void main(String[] args) {
        Customer zhangsan = new Customer();
        zhangsan.name = "张三";
        Customer.setBankName("香港银行");
        Customer lisi = new Customer();
        lisi.name = "李四";
        System.out.println("lisi.bankName = " + lisi.bankName);
    }
}
```

运行代码，控制台打印结果如图 5-7 所示。

```
lisi.bankName=香港银行
```

图 5-7 StaticTest4.java 的运行结果

静态函数可以通过“类名.函数名”来访问，也可以通过“对象名.函数名”来访问，推荐用“类名.函数名”来访问。

注意

在静态函数调用时对象还没有创建，因此在静态函数中不能直接访问类中的非静态成员变量和成员函数，也不能使用关键字 this。例如，下面的代码报错（Cannot make a static reference to the non-static field name）。

```
class Customer{
    String name;
    static String bankName;
    static void setBankName(String bankName){
        Customer.bankName = bankName;
        System.out.println(name);              //报错
    }
}
```

5.1.4　静态代码块

构造函数对于每个对象执行一次,对每个对象进行初始化。那么有没有对所有对象的共同信息进行初始化,并对所有对象只执行一次的机制呢? 有,它就是静态代码块(static block),示例代码如下。

StaticTest5. java

```
class Customer{
    String name;
    static String bankName;
    static{
        bankName = "香港银行";
        System.out.println("静态代码块执行");
    }
}

public class StaticTest5 {
    public static void main(String[] args) {
        Customer zhangsan = new Customer();
        Customer lisi = new Customer();
    }
}
```

运行代码,控制台打印结果如图 5-8 所示。

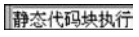

图 5-8　StaticTest5. java 的运行结果

当类被载入时静态代码块被执行,且只被执行一次,静态代码块经常用来进行类属性的初始化。

5.2　认 识 封 装

5.2.1　封装

封装(Encapsulation)是面向对象的基本特征之一。为了理解封装,观察下列代码。

EncTest1. java

```
class Customer {
    String name;
    String sex;
    int age;
}
public class EncTest1 {
    public static void main(String[] args) {
        Customer zhangsan = new Customer();
        zhangsan.age = 25;
```

```
        System.out.println("zhangsan.age = " + zhangsan.age);
    }
}
```

运行代码，控制台打印结果如图 5-9 所示。

在主函数中利用“zhangsan.age＝25;”进行了赋值。

但是这样有一个问题，Customer 类被使用，其对象中的 age 成员可以被任意赋值，例如将“zhangsan.age＝25;”改为“zhangsan.age＝－100;”。运行代码，控制台打印结果如图 5-10 所示。

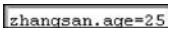

图 5-9　EncTest1.java 的运行结果

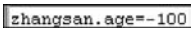

图 5-10　更改值后的运行结果

显然不符合实际情况。因此需要在赋值时进行判断，只有符合常识的 age 才能被赋值，否则会报错，或者赋默认值（如“0”）。

age 是 Customer 的一个成员，在给 age 赋值时应该在 Customer 内进行判断。这就如同使用手机发短信，短信是否已经成功发出是由手机来判断的，我们只需要知道结果就可以了。

此时可以使用封装来完善对象的使用。

5.2.2　实现封装

实现封装有以下两个步骤。

（1）将不能暴露的成员隐藏起来，例如 Customer 类中的 age，不能让其在类的外部被直接赋值。实现方法是将该成员定义为私有的，在成员定义前加上修饰符 private。

（2）用公共方法来暴露对该隐藏成员的访问，可以给函数加上修饰符 public，将该方法定义为公共的。

修改之后的代码如下。

EncTest2.java

```
class Customer {
    String name;
    String sex;
    private int age;
    public void setAge(int age){
        if(age < 0 || age > 100){
            System.out.println("age 无法赋值");
            return;
        }
        this.age = age;
    }
    public int getAge(){
        return this.age;
    }
}

public class EncTest2 {
    public static void main(String[] args) {
```

```
        Customer zhangsan = new Customer();
        zhangsan.setAge(25);
        System.out.println("zhangsan.age = " + zhangsan.getAge());
    }
}
```

运行代码,控制台打印结果如图 5-11 所示。

如果将“zhangsan.setAge(25);”改为“zhangsan.setAge(-100);”,控制台打印运行结果如图 5-12 所示。

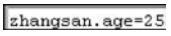

图 5-11　EncTest2.java 的运行结果

图 5-12　更改值后的运行结果

注意

(1) 私有成员只能在定义它的类的内部被访问,在类的外部不能被访问。例如,如果在主函数中调用“zhangsan.age=-100;”将会报错,如图 5-13 所示。

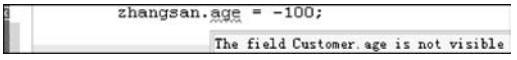

图 5-13　在外部访问报错

(2) 一般情况下,可以将成员变量定义为 private 的,通过 public 函数(方法)对其进行访问。例如要给一个成员赋值,可以使用 setter 函数,如上面的 setAge 函数;要获得该变量的值,可以使用 getter 函数,如上面的 getAge 函数。

(3) private 和 public 都是访问区分符,其他访问区分符将在后面章节讲解。

5.3　使　用　包

5.3.1　包

前面编写的代码,所有的类都写在一个.java 文件中,可能会使文件特别臃肿。在实际操作中,最好将类写在单独的文件中。

当系统庞大之后,类的个数很多,功能也分门别类,可以从操作系统管理文件的方法中得到启发。在操作系统中可能存在大量文件,将文件用文件夹进行管理,如图 5-14 所示。

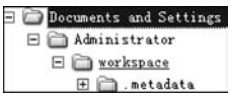

图 5-14　将文件用文件夹管理

在 Java 中使用类似的方法管理类,这就是包(Package)。

5.3.2　将类放在包中

如果定义了一个类,如何将其放在一个包中呢?

方法很简单,只要在类的定义文件头上加“package 包名;”即可。也可以在 Eclipse 中快速建立一个包,右击项目中的 src 目录,选择 New→Package 命令,如图 5-15 所示。

弹出如图 5-16 所示的对话框。

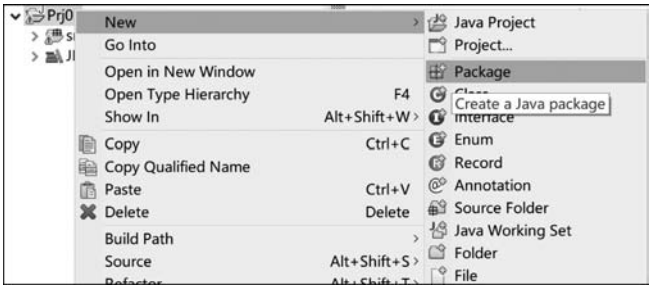

图 5-15　选择 Package 命令

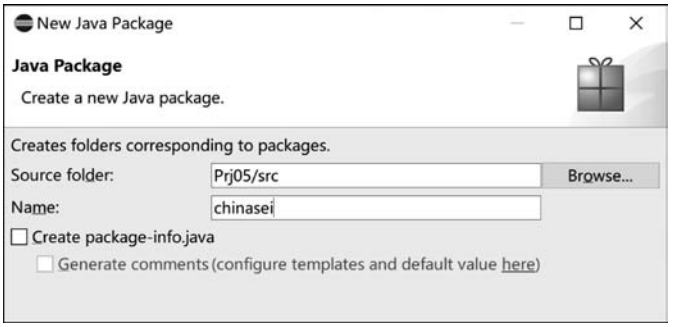

图 5-16　New Java Package 对话框

输入包的名称，单击 Finish 按钮即可，项目结构变为图 5-17 所示。

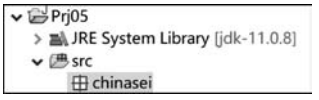

图 5-17　项目结构

可以在包里面建立一个类，代码如下。

Customer. java

```
package chinasei;
class Customer {
    String name;
    String sex;
    private int age;
    public void setAge(int age){
        if(age < 0||age > 100){
            System.out.println("age 无法赋值");
            return;
        }
        this.age = age;
    }
    public int getAge(){
        return this.age;
```

```
    }
}
```

这相当于将 Customer 类放在了 chinasei 包中。

注意

(1) 在源代码中,“package chinasei;”表示该源文件中的所有类都位于包 chinasei 中。package 语句必须放在源代码文件的最前面,也可以不指定 package 语句,相当于将类放在默认包中,不过指定包,使用更加方便、可靠。

(2) 在 Java 中,推荐包名字的字母小写,例如“chinasei”“bank”等,为了便于阅读,有时候还用“. ”隔开,例如“school. admin”“school. stu”等。

(3) 在将类放入某个包中之后,包将会用专门的文件夹来表示,例如上面的 Customer 类,编译出来的. class 文件路径如图 5-18 所示。

图 5-18　文件路径

如果在包名中用了“.”,例如 Teacher 类放在包 school. admin 中,如图 5-19 所示。

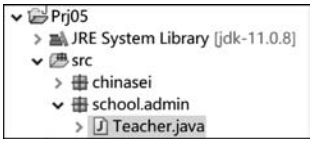

图 5-19　包名中用了“.”

编译出来的结果如图 5-20 所示。

图 5-20　编译结果

当遇到“.”时,系统会认为是要建立一个子文件夹。

(4) 如果要用命令行来运行某个包中的类,必须首先到达包目录所在的上一级目录,例如本例中的 bin 目录,使用以下命令。

```
java  包路径.类名
```

例如,运行 school. admin 中的 Teacher 类,首先必须到达 bin 目录,然后输入以下命令。

```
java school.admin.Teacher
```

这样即可运行其中的主函数。

(5) 使用命令行编译一个. java 文件,在默认情况下不会生成相应目录,例如将前面的

Customer.java 放在 C 盘根目录下，使用如图 5-21 所示的命令。

```
C:\>javac Customer.java
```

图 5-21　编译 Customer.java 文件

此时将在同一目录下生成.class 文件，如图 5-22 所示。

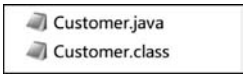

图 5-22　生成.class 文件

如果不将 Customer.class 文件放在相应包目录下则不能运行。
为了解决这个问题，可以用 javac 的-d 选项来生成相应的包目录，如图 5-23 所示。

```
C:\>javac -d . Customer.java
```

图 5-23　用-d 选项生成相应的包目录

编译，则可以生成相应的包目录，如图 5-24 所示。

图 5-24　生成的包目录

（6）编写一个类，编译成.class 文件之后任意放在一个目录下，这并不等于就将该类放在包中。包名必须在源代码中，通过 package 语句指定，而不是靠目录结构来确定。

5.3.3　访问包中的类

将类用包管理之后如何访问包中的类？要分以下几种情况考虑。

1. 在同一个包中直接用类名来访问，不用指定类所在的包

例如，在 chinasei 包中有 Customer 类和 CustomerTest 类，如图 5-25 所示。

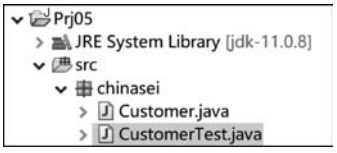

图 5-25　chinasei 包中的类

在 CustomerTest 类中访问 Customer 类，代码如下。

CustomerTest.java

```
package chinasei;
class CustomerTest {
    public static void main(String[] args) {
```

```
        Customer zhangsan = new Customer();
    }
}
```

运行代码不会报错,可以直接访问。

2. 两个类不在同一个包中的情况

在 chinasei 包中建立一个 TeacherTest 类,并在其中使用 school. admin 包中的 Teacher 类,如图 5-26 所示。

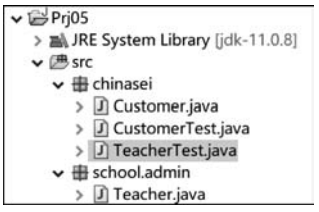

图 5-26　类不在同一个包中

Teacher 类的代码如下。

Teacher. java

```
package school. admin;
public class Teacher {}
```

TeacherTest 类的代码如下。

TeacherTest. java

```
package chinasei;
class TeacherTest {
    public static void main(String[] args) {
        Teacher teacher = new Teacher();
    }
}
```

运行代码则会报错,如图 5-27 所示。

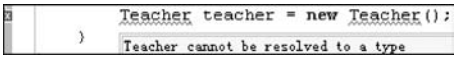

图 5-27　系统报错

解决这个问题的方法有以下两种。

(1) 在使用类时指定类的路径,代码如下。

TeacherTest. java

```
package chinasei;
class TeacherTest {
```

```
    public static void main(String[] args) {
        school.admin.Teacher teacher = new school.admin.Teacher();
    }
}
```

（2）用 import 语句导入该类，代码如下。

TeacherTest.java

```
package chinasei;
import school.admin.Teacher;
class TeacherTest {
    public static void main(String[] args) {
        Teacher teacher = new Teacher();
    }
}
```

注意

（1）如果一个包中的类很多，可以用“import 包名.*”导入该包中的所有类。

（2）在本例中，TeacherTest 类访问 Teacher 类，必须要保证 Teacher 是 public 类（定义时 class 前必须加关键字 public），这将在后面章节讲解。

（3）有时候，包名中有“.”，例如“school.admin”，这并不是说 school 包中包含了 admin 包，school.admin 仅是一个包名而已。因此，“import school.*;”只是导入了 school 包中的类，并没有导入 school.admin 包中的类，如果要导入 school.admin 包中的类，必须使用“import school.admin.*;”。

5.4 使用访问控制修饰符

5.4.1 访问控制修饰符

前文讲解了两个访问控制修饰符，分别是 private 和 public，但是没有对它们进行详细讲解，本节将结合包的相关知识对访问控制修饰符进行详细讲解。

5.4.2 类的访问控制修饰符

在定义类时，有时会在类的前面加上关键字 public，示例代码如下。

```
public class Customer {
    String name;
    String sex;
    int age;
}
```

写与不写关键字 public 有何区别？

在不写 public 的情况下属于默认访问修饰，此时该类只能被同一包中的所有类识别。

如果写了 public，该类则是一个公共类，可以被包内、包外的所有类识别。

注意

如果将一个类定义成 public 类，类名和文件名必须相同，因此在一个.java 文件中最多只能有一个 public 类。

5.4.3 成员的访问控制修饰符

对于成员来说，访问控制修饰符共有 4 个，分别是 private、default、protected、public，示例代码如下。

```
public class Customer {
    private String name;
    String sex;
    protected int age;
    public void display(){}
}
```

name 成员为 private 类型，sex 成员为 default 类型，age 成员为 protected 类型，display 成员为 public 类型。其中，default 类型的成员前面没有任何修饰符。

其特性如下。

(1) private 类型的成员只能在定义它的类的内部被访问。

(2) default 类型的成员可以在定义它的类的内部被访问，也可以被这个包中的其他类访问。

(3) protected 类型的成员可以在定义它的类的内部被访问，也可以被这个包中的其他类访问，还可以被包外的子类访问。关于子类，将在后面章节讲解。

(4) public 类型的成员可以在定义它的类的内部被访问，也可以被包内、包外的所有其他类访问。

很明显，从开放的程度上讲，private＜default＜protected＜public。

5.5 使用类中类

类中类，顾名思义是在类中定义了类，也称为内部类。

为什么要在类中定义类呢？这是由实际需要决定的。例如有两个类 A 和 B，B 中要用到 A 中的一些成员，A 又要实例化 B，两者的关系错综复杂，此时编写成类中类比较紧凑，示例代码如下。

Outer.java

```
class Outer{
    int a;
    void funOuter(){
        Inner inner = new Inner();
    }
    class Inner{
        int b;
        void fun(){
```

```
            a = 3;
            this.b = 5;
        }
    }
}
```

用命令行编译，如图 5-28 所示。

```
C:\>javac Outer.java
```

图 5-28　用命令行编译

得到的.class 文件如图 5-29 所示。

Outer$Inner.class	2017/2/17 10:57	CLASS 文件	1 KB
Outer.class	2017/2/17 10:57	CLASS 文件	1 KB

图 5-29　得到.class 文件

很显然，以上代码在 Outer 类中定义了 Inner 类，内部类可以访问外部类中的成员。类中类编译成的.class 文件的命名为“外部类 $ 内部类.class”。

注意

（1）内部类中的成员只在内部类范围内才能使用，外部类不能像使用自己的成员变量一样使用它们。

（2）如果在内部类中使用 this，仅代表内部类的对象，因此也只能引用内部类的成员。

本章习题

1. 编写一个 Customer 类，含有一个名为“编号”的成员变量。要求每实例化一个对象，对象的编号从“1”自动递增。

2. 我们经常使用 System.out.println()，思考 System 是什么？out 是什么？println 是什么？

3. 定义一个银行 Customer 类，含有 name 和 balance（余额）两个成员。余额不能随意赋值，必须通过存款或者取款活动才能进行变化。请编写存款和取款这两个函数。注意，取款时必须保证余额充足。

4. 定义一个“日期”类，含有“年”“月”“日”3 个成员变量，含有以下成员函数。

（1）输入年月日。保证月为 1～12，日要符合相应范围。否则报错。

（2）用“年-月-日”的形式打印日期。

（3）用“年/月/日”的形式打印日期。

（4）比较该日期是否在另一个日期的前面。

5. 将上题中的“日期”类放入 date 包。建立一个 main 包，包内放一个 TestDate 类，含有主函数，来测试日期类。

6. 定义一个“时间”类，含有“小时”“分钟”“秒钟”3 个成员变量，含有以下成员函数。

(1) 输入时、分、秒数据,所输入的数据要符合相应范围。

(2) 用“时:分:秒”的形式打印时间。

(3) 计算该时间和另一个时间相差的秒数。

7. 将上题中的“时间”类放入 time 包。在 main 包中编写一个 TestTime 类,含有主函数,来测试“时间”类。