**ava**

**J**

第 **5** 章

类和对象

面向对象程序设计（object-oriented programming，OOP）是一种基于对象概念的软件开发方法，目前已经成为一种主流的程序设计模式。类是数据及对数据操作的封装体，具有封装性。封装性是面向对象方法的基础。对象是类的具体实例，对象与类的关系恰如变量和数据类型的关系。

# 面向对象概述

面向对象程序设计方法是在结构化设计方法（如 C 语言即使用此方法）出现大量问题的情况下应运而生的。随着实际问题规模不断增加，应用系统越来越复杂，需求变更越来越频繁，结构化设计方法面临着维护和扩展的困难，甚



视 频 讲 解

至一个微小的变动，都会波及整个系统。这驱使人们寻求一种新的程序设计方法，以适应现代社会对软件开发的更高要求，面向对象程序设计方法由此产生。Java 正是一种纯面向对象的程序设计语言。

1. 面向对象程序设计思想

面向对象的思想是将所有问题都看作“对象”。现实世界中任何事物都可以作为“对象”，比如人、动物、汽车、手机等。每种事物都有自己的属性和行为，比如人，具有的属性有姓名、性别、年龄、身高、体重等，具有的行为有吃饭、睡觉、学习、打球等。正是因为每种事物都有自己的属性和行为，人们才能对现实中的“对象”进行分类。

面向对象程序设计是对现实世界的模拟，从人类考虑问题的角度出发，把人类解决 问题的思维过程转变为程序能够理解的过程。面向对象编程的主要任务是建立模拟问题领域的对象模型，并通过程序代码实现对象模型，然后把问题领域中每个具体的事物分别抽象为一个个对象。比如要处理学生数据，首先应刻画出一个学生模型，包括学生共有的属性和行为，然后根据学生模型生成张三、李四等具体的学生对象。

**Java** 程序设计

面向对象编程更加符合人的思维习惯，使编程者更容易写出易维护、易扩展、易复 用的程序。面向对象程序设计方法更利于复杂系统的分析、设计和编程实现，通过封装技术及消息机制可以像搭积木一样快速开发出一个全新的系统。

1. 面向对象的特性

面向对象编程主要有以下 3 个特性。

1. 封装性

封装是通过类来实现的。封装有两层含义：一是类中封装了该类型对象所具有的属 性和行为；二是隐蔽类的内部实现细节，与外部的联系只能通过外部接口实现。如图 5-1 所示。

ㆧ⮰჊⣜㏲㞮喏⩔ᝣ̹छ㻭

ㆧᄥโ⮰ᣑए喏⩔ᝣछ㻭

class Phone{

private String brand; private double price;

public void call(){…} public void message(){…} public void play(){…}

}

public void call() public void message() public void play()

图 5-1 类的封装性示例

通过将类中属性的访问权限设置为私有的（private），对象的属性值只能由该对象的行为进行读取和修改，阻止了对象以外的代码随意访问对象内部的属性，从而使程序数据更加安全；也有效地避免了外部错误对它的影响，大大减少了查错和排错的难度。

封装的信息隐蔽，使用户只关心类对外提供的接口，即它能做什么；而无须关心其内部实现细节，即如何提供这些服务。比如手机，人们只关心它对外提供哪些功能，如打电话、收发短信、播放音乐等，而不关心它的内部有哪些部件以及功能是如何实现的。

1. 继承性

继承指子类拥有父类的属性和行为。继承意味着“自动拥有”，即子类中不必重新定义父类中已定义过的属性和行为，它会自动地、隐含地拥有父类的属性与行为；同时子 类又可以新增自己的属性和行为。比如，“学生”类继承了“人”类的属性和行为，如姓名、性别、吃饭、睡觉等，同时又新增了自己新的属性和行为，如学号、专业、考试、上课等。

在软件开发过程中，继承实现了软件模块的可重用性，缩短了开发周期，提高了软 件开发效率，使软件更易于维护和修改。这是因为，要修改或增加某一属性或行为，只需要在相应的类中进行改动，由它派生的所有类就会自动地、隐含地做相应的改动。

第 5 章 类和对象

1. 多态性

多态的表现形式有两种。第一种是行为名称的多态，即有多个行为具有相同的名字， 但这些行为所接收的消息类型必须不同，对象会根据传递的消息产生一定的行为。例如 “求面积”的行为，当接收一个数据时求的是圆的面积，接收两个数据时求的是长方形的 面积。第二种是同一继承体系中，同一个操作被不同对象调用时产生不同的行为。例如 “动物”类都有“叫”的行为，子类“狗”和“猫”都继承了“动物”类的叫的行为，但 一个叫的声音是“汪汪……”，另一个叫的声音是“喵喵……”。

在 Java 中，行为名称的多态通过方法重载实现，子类继承的多态通过方法重写实现。

# 类和对象概述

任何面向对象程序中最主要的单元是对象。对象并不会凭空产生，它必须有一个可以依据的原型，而这个原型就是面向对象程序设计中所谓的类。类是一种用来具体描述对象属性与行为的数据类型。例如，汽车是一个类，参照这



视 频 讲 解

个汽车类的属性和行为造出来的每一辆汽车都被称为汽车类的对象。因此可以说，类是一个模型或蓝图，按照这个模型或蓝图所“生产”或“创建”出来的实例则被称为对象。

* + 1. 面向对象的基本概念

面向对象程序设计中最主要的两个概念是类和对象。

1. 类

在面向对象技术中，将客观世界中的一个事物作为一个对象看待，每个事物都有自 己的属性和行为。在面向对象的程序设计方法中，将属性和行为合起来定义为类。类是定义一组具有共同属性和行为的对象的模板。

从程序设计的角度看，事物的属性可以用变量描述，行为可以用方法描述。类中的 变量称为成员变量，类中的方法称为成员方法。成员变量反映类的状态和特征，成员方法表示类的行为能力。不同的类具有不同的属性和行为。

1. 对象

类是定义属性和行为的模板，但仅有类是不够的，还必须创建属于类的对象。对象 是类的具体实例，对象与类的关系恰如变量和数据类型的关系一样。例如，int 型变量 i 可以存放整数值 13，可以对 i 进行加、减、乘、除等操作；根据 Person 类创建 zhangSan 对象，用于存放“张三”的姓名、性别、身高等属性及阅读、开车、游泳等行为操作。

* + 1. 类的声明

类是组成 Java 程序的基本要素，每个程序中至少要包含一个类。类的定义包括两部分：类声明和类体。类体中包含了属性的定义和行为的定义，属性通过变量来刻画，行为

**Java** 程序设计

通过方法来实现。

在 Java 语言中，类的声明格式如下：

[ 访问权限修饰符 ] class 类名 {

成员变量 ;

成员方法 ;

}

其中，[] 表示该项为可选项，如果选择书写时要去掉 []。定义类的基本要求如下：

1. class 是声明类的关键字，class 后的类名的命名需要遵循标识符的命名规则，即类名应由字母、数字、下画线和美元符号组成，且首字母要大写，如 Person、Student。
2. 类的访问权限只有两种，即 public 和默认。

public 指明该类是公共类，可以被所有的类使用。一般将包含 main() 方法的类定义为public 类，保存的源程序文件名必须与 public 类的类名相同，而且该源程序文件中其他类不能再用 public 修饰。

默认即不给修饰符，该类只能被同一源程序文件中的类或同一个包中的其他类使用。

1. 类的具体定义包含在一对花括号内，包括成员变量和成员方法的声明。

① 成员变量

成员变量的声明格式如下：

[ 修饰符 ] 数据类型 变量名 [= 值 ];

其中，数据类型可以是 Java 中的任意类型，包括基本数据类型和引用数据类型。修饰符的内容会在后文中详细介绍。

② 成员方法

成员方法的声明格式如下：

[ 修饰符 ] 返回值数据类型 方法名 ([ 形参列表 ]){

方法体 ;

}

其中，返回值类型为方法返回值的数据类型，可以是任何数据类型。如果方法有返回值， 则方法体中至少有一条 return 语句，其格式为“return 表达式 ;”，而且 return 语句后的表达式类型要与返回值类型保持一致。如果方法没有返回值，则返回值类型为 void（无类型），方法体中不必使用 return 语句。

形参列表用于在调用方法时，接收实参传递过来的参数值。方法可以没有形参，也 可以有一个或多个形参。如果有多个形参，各形参声明时需要用逗号分隔，而且每个形参都必须包括数据类型和参数名。

【例 5-1】根据类的结构定义学生类 Student。

第 5 章 类和对象

class Student{ // 定义名为 Student 的类

String name; // 成员变量 - 姓名

int age; // 成员变量 - 年龄

void introduce(){ // 成员方法 - 自我介绍

System.out.println(" 姓名："+name+" 年龄："+age);

}

void haveClass(String course){ // 成员方法 - 上课

System.out.println(name+" 正在上 "+course+" 课，这门课真有趣！ ");

}

}

* + 1. 对象的创建和使用

类和对象的关系是模具和产品的关系，模具的作用是生成产品，定义类的目的是创 建具有相应属性和行为的对象。由类生成对象的过程，也称为类的实例化过程。一个类可以生成多个对象。

1. 对象的创建

对象的创建包括声明和初始化两项工作，可以通过两条语句分别完成声明和初始化 工作，也可以通过一条语句同时完成声明和初始化工作。

创建对象的格式如下：

类名 对象名；

对象名 =new 类名 ([ 实参列表 ]);

或

类名 对象名 =new 类名 ([ 实参列表 ]);

new 运算符的作用是初始化对象，为对象在内存中申请存储空间，并同时自动调用类

的构造方法为对象的属性赋初值。

例如，语句 Student liMing=new Student(); 创建了 Student 类的对象 liMing。

1. 对象的使用

创建一个对象后，该对象就具有了所属类的成员变量和成员方法。对象可以通过点 运算符“.”实现对成员变量的访问和成员方法的调用。

对象引用成员变量的格式如下：

对象名 . 成员变量名

对象调用成员方法的格式如下：

对象名 . 成员方法名 ([ 实参列表 ])

其中，实参是传递给该方法形参的值，实参可以是常量、变量或表达式。多个实参间用逗号分隔。实参的个数、顺序、类型和形参要一一对应。

【例 5-2】定义测试类 CH05\_02，创建并使用 Student 类的对象。

**Java** 程序设计

//CH05\_02.java class Student{

String name; int age;

void introduce(){

System.out.println(" 姓名："+name+" 年龄："+age);

}

void haveClass(String course){ //course 为形参变量

System.out.println(name+" 正在上 "+course+" 课，这门课真有趣！ ");

}

}

public class CH05\_02 { // 只能有一个 public 类

public static void main(String[] args){

Student zhangSan=new Student(); // 创建第一个学生对象Student liSi=new Student(); // 创建第二个学生对象zhangSan.name=" 张三 "; // 为对象的成员变量赋值zhangSan.age=18;

liSi.name=" 李四 "; liSi.age=20;

zhangSan.introduce(); // 调用对象的成员方法

liSi.haveClass("JAVA"); //"JAVA" 为实参

}

}

程序执行结果如下：



* + 1. 构造方法和对象的初始化

在例 5-2 中，创建学生对象后，都需要引用成员变量为姓名和年龄属性赋初值。如果在创建对象的同时即为属性赋初值，该如何实现呢？类的构造方法可以实现这种要求。构造方法是类的一种特殊的成员方法，作用是在创建对象时初始化对象，即给对象

的各成员变量赋初值。构造方法定义的格式如下：

构造方法名 ([ 形参列表 ]){

// 对象属性赋初值的语句

}

使用构造方法时，需要注意以下几方面：

① 构造方法名必须与类名相同。

② 构造方法没有返回值，前面不能有返回值类型，也不能有 void。

③ 构造方法是在用 new 运算符创建对象时，被系统自动调用的，不能像普通成员方法那样被显式地直接调用。

第 5 章 类和对象

④ 构造方法中除了初始化对象属性值的语句外，不建议包含其他语句。

【例 5-3】通过设计构造方法实现例 5-2。

//CH05\_03.java class Student1{

String name; int age;

Student1(String n,int a){ // 构造方法

name=n; // 为成员变量赋初值

age=a;

}

void introduce(){

System.out.println(" 姓名："+name+" 年龄："+age);

}

void haveClass(String course){

System.out.println(name+" 正在上 "+course+" 课，这门课真有趣！ ");

}

}

public class CH05\_03 {

public static void main(String[] args){

// 创建对象时自动调用构造方法初始化对象的属性值Student1 zhangSan=new Student1(" 张三 ",18); Student1 liSi=new Student1(" 李四 ",20); zhangSan.introduce(); liSi.haveClass("JAVA");

}

}

程序执行结果如下：



1. 默认构造方法

在例 5-2 的程序中，并没有显式地为 Student 类定义构造方法，那么在用 new 运算符创建对象时，系统自动调用的是哪一个构造方法呢？

在定义一个类时，如果没有显式地定义构造方法，系统将自动为该类创建一个默认 构造方法。默认的构造方法没有参数，方法体中没有任何代码。

如果例 5-2 中 Student 类没有显式定义构造方法，系统则会自动创建如下的构造方法：

public Student(){

}

在使用 new Student() 创建 Student 类的对象时，系统自动调用了这个默认构造方法。

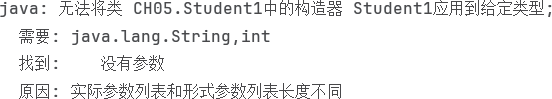
**Java** 程序设计

1. 带参数的构造方法

由于系统自动创建的默认构造方法往往不能满足使用要求，因此可以在设计类的时 候显式定义构造方法，如，例 5-3 Student1 类中增加了一个构造方法，通过构造方法初始化对象的成员变量。需要注意的是，如果类中定义了带参数的构造方法，系统将不再 自动创建默认构造方法。如，例 5-3 中只提供了带参数的构造方法，如果将“Student1 zhangSan=new Student1(" 张三 ",18); ”改为如下语句：

Student1 zhangSan=new Student1();

编译器将会提示如下错误：



1. 构造方法重载

如果要使用多种方式创建对象，则可以使用重载构造方法。重载构造方法在一个类 中声明多个构造方法，但各构造方法的参数个数和参数类型不能相同。在使用 new 运算符创建对象时，系统会根据给出的参数个数和类型调用对应的构造方法。

【例 5-4】构造方法重载示例。

//CH05\_04.java class Rectangle{

double length; double width;

Rectangle(){ // 无参数构造方法

length=2; width=5;

}

Rectangle(double len,double wid){ // 有参数构造方法

length=len; width=wid;

}

double area(){

return length\*width;

}

}

public class CH05\_04 {

public static void main(String[] args){ Rectangle r1=new Rectangle(); Rectangle r2=new Rectangle(10,20);

System.out.println(" 第一个长方形 r1 的面积 ="+r1.area());

第 5 章 类和对象

System.out.println(" 第一个长方形 r2 的面积 ="+r2.area());

}

}

程序执行结果如下：



* + 1. this 关键字的使用

this 是 Java 的一个关键字，表示当前对象。可以使用 this 引用当前对象的成员变量、成员方法和构造方法。注意，this 只能出现在非静态成员方法（即没有被 static 修饰的方法）中。

1. 使用 this 区分成员变量与方法形参

当成员方法中存在与类中成员变量同名的参数时，引用成员变量时其名称前必须加 上 this，因为成员方法中默认的变量是方法的参数变量。但如果成员方法中没有与成员变量同名的参数时，this 可以省略。通过 this 引用成员变量的格式如下：

this. 成员变量名

1. 使用 this 调用其他构造方法

如果一个类中有多个构造方法，有时会用某个构造方法完成初始化工作，其余的构 造方法再调用此构造方法，从而避免了重复编写相同的代码。可以利用 this 在一个构造方法中调用另一个构造方法。通过 this 调用构造方法的格式如下：

this. 构造方法 ([ 实参列表 ])

1. 使用 this 调用成员方法

this 表示当前对象，也就是调用成员方法的这个对象。可以利用 this 在一个成员方法中调用其他的成员方法。通过 this 调用成员方法的格式如下：

this. 成员方法名 ([ 实参列表 ])

其中，成员方法名前的 this 可以省略。

【例 5-5】this 使用示例。

//CH05\_05.java class Rectangle1{

double length; // 成员变量

double width;

Rectangle1(double length,double width){ this.length=length; // 区分成员变量与方法的形参this.width=width;

}

Rectangle1(){

**Java** 程序设计

this(4,5); // 调用上面带有 2 个参数的构造方法

}

double area(){

return length\*width;

}

void outArea(){

System.out.println(" 面积 ="+this.area()); //this 可省略

}

}

public class CH05\_05 {

public static void main(String[] args){ Rectangle1 r=new Rectangle1(); r.outArea();

}

}

程序执行结果如下：



* + 1. 成员方法的参数传递

在调用一个带有形参的方法时，必须为方法提供实参，完成实参与形参的结合，这 个过程称为参数传递。方法中形参的数据类型可以是基本数据类型，也可以是引用数据类型。基本数据类型的参数传递是以传值方式进行的，而引用数据类型的参数传递是以传地址方式进行的。

1. 基本数据类型参数传递

对于基本数据类型的形参，方法的形参接收实参的值。在方法中改变形参的值，并 不会影响到实参的值，因为实参和形参的传值是单向的，形参不会把值再传回给实参。

【例 5-6】基本数据类型参数传递示例，交换两个变量的值。

//CH05\_06.java class Swaping{

void swap(int m,int n){ //m 和 n 是形参

System.out.println(" 交换之前的数据，m="+m+" n="+n); int t=m; // 实现数据交换

m=n; n=t;

System.out.println(" 交换之后的数据，m="+m+" n="+n);

}

}

public class CH05\_06 {

public static void main(String[] args){ int a=10,b=20;

第 5 章 类和对象

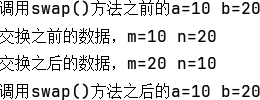
Swaping s=new Swaping();

System.out.println(" 调用 swap() 方法之前的 a="+a+" b="+b); s.swap(a,b); //a 和 b 是实参System.out.println(" 调用 swap() 方法之后的 a="+a+" b="+b);

}

}

程序执行结果如下：



1. 引用数据类型参数传递

如果实参是引用数据类型，并且已经指向一个对象，那么方法被调用时，传递给方 法形参的是实参所指向的对象在内存中的地址，即形参和实参指向同一个对象，如果在方法体中通过形参改变了对象，那么，实参所指向的对象也一样被改变了。

【例 5-7】数组参数的传递示例。

数组作为参数，要注意两点：在形参表中，数组名后的方括号必须给出，而且方括号个数和实参数组的维数相等，但不要给出数组的长度；在实参表中，数组名后不能给出方括号。

//CH05\_07.java

import java.util.Arrays; class ArraySort{

void sort(int b[]){ // 形参数组 b 后必须有 []

Arrays.sort(b);

}

}

public class CH05\_07 {

public static void main(String[] args){ int[] a={9,1,6,4,3,2};

ArraySort arr=new ArraySort();

System.out.println(" 调用 sort() 方法前数组 a 的值："+Arrays.toString(a)); arr.sort(a); // 实参数组 a 后不能有 []

System.out.println(" 调用 sort() 方法后数组 a 的值："+Arrays.toString(a));

}

}

程序执行结果如下：



【例 5-8】类对象参数的传递示例。

**Java** 程序设计

//CH05\_08.java class Student\_2 {

String name; double score;

Student\_2(String name, double score) { this.name = name;

this.score = score;

}

}

class Change{

void changeScore(Student\_2 s){ // 类对象 s 为形参

if(s.score>=95){ s.score=100;

}

else{

s.score=s.score+5;

}

}

}

public class CH05\_08 {

public static void main(String[] args){ Student\_2 student=new Student\_2(" 张三 ",85);

System.out.println(" 调用 changeScore() 方法前 "+student.name+ " 的成绩是："+student.score);

Change ch=new Change();

ch.changeScore(student); // 类对象 student 为实参

System.out.println(" 调用 changeScore() 方法后 "+student.name+ " 的成绩是："+student.score);

}

}

程序执行结果如下：



* + 1. 成员方法的递归调用

递归就是用自身的结构来描述自身，最典型的例子是阶乘运算的定义，阶乘运算的 定义如下：

n!=n×(n-1)!

{

1!=1

用阶乘本身来定义阶乘，这样的定义称为递归定义。

在 Java 中，不仅允许一个方法在其定义的内部调用其他方法，而且允许一个方法在

第 5 章 类和对象

自身定义的内部调用自己，这样的方法称为递归方法。在编写递归方法时，只要知道递归定义的公式和递归终止的条件，就能容易地写出相应的递归方法。

【例 5-9】采用递归算法求 n!。

根据阶乘的概念，写出求 n! 的递归定义： fac(n)=1, n=1

{

fac(n)=n×fac(n-1)， n>1 采用递归算法的程序如下：

//CH05\_09.java

import java.util.Scanner; class Factorial{

long fac(int n){ if(n==1){

return 1;

}

else{

return n\*fac(n-1); // 递归调用

}

}

}

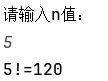
public class CH05\_09 {

public static void main(String[] args){ Factorial factorial=new Factorial(); Scanner in=new Scanner(System.in); System.out.println(" 请输入 n 值："); int n=in.nextInt();

long f=factorial.fac(n); System.out.println(n+"!="+f);

}

}

程序执行结果如下：

# 类的封装

封装性是面向对象的核心特征之一，它提供了一种信息隐藏技术。类的封装包含两层含义，一是将数据和对数据的操作组合起来构成类，类是一个不可



视 频 讲 解

分割的独立单位；二是类中既要提供与外部联系的接口，同时又要尽可能隐藏类的实现细节。本节主要介绍解决类命名冲突的包机制、类及其成员访问权限等内容。

**Java** 程序设计

* + 1. 包

Java 为了更好地管理类，引入了包（package）的概念。可以把不同的类放到不同的包中，进行分类管理，也可以把相同名字的类放到不同包中，避免同名问题。

1. 包的作用

一个 Java 源程序文件（即 .java 文件）包含了一个或多个类，经过编译后，文件中的每个类都将生成一个与类名一致的字节码文件（即 .class 文件）。例如，一个 Java 源程序文件定义了类 Student、类 Teacher、类 Course、类 Test，编译后，会在 Java 源程序文件所在的文件夹下生成 Student.class、Teacher.class、Course.class、Test.class 四个字节码文件。如果多个 Java 源程序文件中有同名的类，编译后生成的字节码文件会出现命名冲突的问题。为了解决类的命名冲突问题，Java 提供了包机制。

一个包实际上就是操作系统下的一个文件夹。一个包中不允许有同名的类存在，但 不同的包中允许有同名的类。包中还可以再建立多个子包，包含子包的包称为父包，父包和子包形成了一种层次结构。

包的作用主要有两个：

① 解决类的命名冲突问题。

② 将类按照类别放到不同的包中，方便对类的管理、维护和保护。

1. 创建包

通过关键字 package 创建包，其格式如下：

package 包名 ;

创建包时，需要注意以下几方面：

① 一个源程序文件中只能有一条 package 语句，并且必须是第一条语句，用于指明该源程序文件定义的类所在的包。

② 包的名字有层次关系，父包和子包之间以“.”分隔。

③ 包名由小写字母组成。在自己设定的包名之前最好加上唯一的前缀，通常使用组织倒置的网络域名，如 cn.edu.tsinghua。

例如：

package cn.edu.tsinghua; class Student{

…

}

以上代码表明 Student 类所在的包为 cn.edu.tsinghua， 即 Student 的命名为 cn.edu. tsinghua.Student。

第 5 章 类和对象

1. 导入包中的类

为了使用不在同一个包中的类，需要在源程序中使用 import 关键字导入这个类。导入的两种格式如下：

import 包名 . 类名 ; import 包名 .\*;

用 import 导入包中的类，需要注意以下两点：

①“\*”表示包中所有的类，即导入指定包中的所有类。

② 一个源程序文件可以包含多条 import 语句，但其必须位于其他类声明之前， package 语句之后。

import 不仅可以导入 Java 类库中的类，也可以导入编程者自己编写的类。

1. 导入 Java 类库中的类

Java 本身提供了许多类，这些类都放在不同的包中。如果编程者需要使用 Java 类库中的类，就必须用 import 语句将其导入。使用已经存在的类，避免一切从头做起，是面向对象编程的一个重要方面。

Java 提供的常用包如下：

① java.lang：该包是 Java 语言的核心包，系统自动为程序导入包中的类（如 System、String 等），不需要使用 import 导入即可使用。

② java.util：实用包。提供了各种实用功能的类，如 Scanner、Arrays 等。

③ javax.swing：轻量级窗口工具包。它是目前使用最广泛的 GUI 程序设计包。

④ java.io：输入输出包。提供了系统输入输出类。

⑤ java.sql：数据库连接包。提供了操作数据库的类。

⑥ java.net：网络方法包。提供了实现网络应用与开发的类。例如，导入 java.util 包中的所有类：

import java.util.\*;

使用 import 语句导入整个包中的类，可能会增加编译时间，但是不会影响程序的运行性能，因为程序运行时只加载真正使用的类的字节码文件。

1. 导入自定义包中的类

如果要导入编程者在其他包中声明的类，也需要使用 import 语句。同一包下的类不需要导入，可以直接使用。

例如，导入包 cn.edu.tsinghua 中的类 Student：

import cn.edu.tsinghua.Student;

1. 包的应用举例

在软件开发过程中，可以将有价值的类打包，形成“软件产品”，供其他软件开发者

**Java** 程序设计

使用。

【例 5-10】创建 Area 类，该类可以求圆、长方形或三角形的面积，将源程序文件Area.java 保存在 D:\CH05\mypackage 中。创建 CH05\_10 类，使用 import 语句导入 Area 类， 计算不同图形的面积，源程序文件 CH05\_10.java 保存在 D:\CH05 中。

//Area.java

package CH05.mypackage; // 创建自定义包

public class Area {

public double area(double r){ // 思考，public 是否可以省略

return 3.14\*r\*r;

}

public double area(double length,double width){ return length\*width;

}

public double area(double a,double b,double c){ double p=(a+b+c)/2;

return Math.sqrt(p\*(p-a)\*(p-b)\*(p-c));

}

}

//CH05\_10.java package CH05;

import CH05.mypackage.Area; // 导入其他包中的类

public class CH05\_10 {

public static void main(String[] args){ Area graph=new Area();

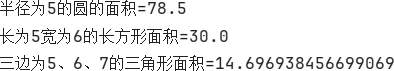
System.out.println(" 半径为 5 的圆的面积 ="+graph.area(5));

System.out.println(" 长为 5 宽为 6 的长方形面积 ="+graph.area(5,6)); System.out.println(" 三边为 5、6、7 的三角形面积 ="+graph.area(5,6,7));

}

}

程序执行结果如下：



* + 1. 访问权限

按照类的封装性原则，类的设计者既要提供类与外部的接口，又要尽可能地隐藏类 的实现细节。具体方法是为类及其成员变量和成员方法分别设置合理的访问权限，从而只向使用者提供接口，但隐藏了实现细节。

1. 类的访问权限

声明一个类可以使用的权限修饰符只有 public 和默认两种。public 指明该类是公共类，

第 5 章 类和对象

可以被同一个包中的类或不同包中的其他类使用。默认的类只能被同一源程序文件中的类或同一个包中的其他类使用。类的访问权限总结如表 5-1 所示。

表 5-1 类的访问权限

|  |  |  |
| --- | --- | --- |
| 权限修饰符 | 同一个包的类 | 不同包的类 |
| public | √ | √ |
| 默认 | √ | × |

虽然一个 Java 源程序文件中可以包含多个类，但只能有一个 public 类，而且该类的名字要与源程序文件的名字相同。

1. 类中成员的访问权限

类的成员包括成员变量和成员方法，成员的访问权限有 4 种，控制级别从强到弱依次为 public、protected、默认和 private。

1. 公共访问权限控制符 public

public 访问权限最宽松。如果类的成员变量或成员方法被 public 修饰，则任何类均可访问类中的 public 成员。

1. 保护访问权限控制符 protected

protected 设置了中间级的访问权限，被其修饰的成员变量或成员方法可以被同一个包中的类或不同包中的子类访问，但不能被不同包中的非子类访问。

1. 默认访问权限控制符

默认指不使用权限修饰符。不使用权限修饰符修饰的成员变量或成员方法，可以在 声明它的类中被访问，也可以被同一包中的类访问，但不能被其他包中的类访问。

1. 私有访问权限控制符 private

如果类的成员变量或成员方法被 private 修饰，则只能在声明它们的类中被访问，而其他任何类，包括该类的子类都不能访问。

类成员的 4 种访问权限总结如表 5-2 所示。

表 5-2 类的访问权限

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 权限修饰符 | 同一个类 | 同一个包中的类 | 不同包中的子类 | 不同包中的非子类 |
| public | √ | √ | √ | √ |
| protected | √ | √ | √ | × |
| 默认 | √ | √ | × | × |
| private | √ | × | × | × |

在声明类时，通常将成员变量声明为 private 权限，仅允许本类的方法访问成员变量， 而将成员方法声明为 public 权限，供其他类调用。其他类通过调用具有 public 权限的方

**Java** 程序设计

法，以其作为接口使用具有 private 权限的成员变量，从而实现了信息封装。

一个类可以供其他类使用，在创建该类的对象时，要调用构造方法，所以构造方法 都声明为 public 权限。

【例 5-11】使用封装性将属性私有化、方法公共化示例。

//CH05\_11.java class Cat{

private String name; // 成员变量为私有的

private int age;

public Cat(String name,int age){ // 构造方法为公有的

this.name=name; this.age=age;

}

public void showMessage(){ // 成员方法为公有的

System.out.println(name+age+" 岁了！ ");

}

}

public class CH05\_11 {

public static void main(String[] args){ Cat cat=new Cat(" 小黑 ",3); cat.showMessage();

}

}

程序执行结果如下：



* + 1. 实例成员和用 static 修饰的类成员

Java 的类中可以包含两种成员：实例成员和类成员。

实例成员是属于对象的，包括实例成员变量和实例成员方法。只有创建对象之后， 才能通过对象访问实例成员变量，调用实例成员方法。前面章节中讨论的类中的成员变量都是实例成员变量。

类成员是属于类的，需要使用 static 修饰，类成员也称为静态成员。类成员包括类成员变量和类成员方法。通过类名可以直接访问类成员变量，调用类成员方法。即使没有创建对象，也可以引用类成员。类成员也可以通过对象名引用。

1. 类变量和实例变量的区别

类变量和实例变量的区别如下：

① 在类中声明变量时，没有使用 static 修饰的变量为实例变量，使用 static 修饰的变量为类变量。

② 不同对象的实例变量互不相同，会被分配不同的内存空间。改变其中一个对象的

第 5 章 类和对象

实例变量，并不会影响其他对象的实例变量。

所有对象共享一个类变量，系统仅为类变量分配一个存储空间。在某个对象修改了 类变量的值后，所有对象都将使用修改了的类变量值。

③ 实例变量属于对象，只能通过对象引用；类变量属于类，既可以通过类名访问， 也可以通过对象名访问。

实例变量的访问格式如下：

对象名 . 实例变量名

类变量的访问格式如下：

类名 . 类变量名

或

对象名 . 类变量名

【例 5-12】类变量和实例变量的区别示例。

//CH05\_12.java class Student2{

private String name; // 实例变量

static int cn=0; // 类变量

public Student2(String name){ this.name=name;

cn=cn+1; // 创建一个学生对象，人数加 1

}

public String getName(){ return name;

}

}

public class CH05\_12 {

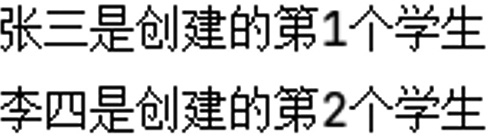
public static void main(String[] args){ Student2 s1=new Student2(" 张三 ");

System.out.println(s1.getName()+" 是创建的第 "+Student2.cn+" 个学生 "); Student2 s2=new Student2(" 李四 "); System.out.println(s2.getName()+" 是创建的第 "+s2.cn+" 个学生 ");

}

}

程序执行结果如下：



1. 类方法与实例方法的区别

类方法和实例方法的区别如下：

① 在类中声明方法时，没有使用 static 修饰的方法为实例方法，使用 static 修饰的方

**Java** 程序设计

法为类方法。

② 类方法不可以操作实例变量，也不可以调用实例方法，只能操作类变量和类方法。实例方法既可以操作实例变量和调用实例方法，也可以操作类变量和调用类方法。

③ 实例方法属于对象，只能通过对象引用；类方法属于类，既可以通过类名访问， 也可以通过对象名访问。

实例方法的访问格式如下：

对象名 . 实例方法名 ([ 参数列表 ])

类方法的访问格式如下：

类名 . 类方法名 ([ 参数列表 ])

或

对象名 . 类方法名 ([ 参数列表 ])

【例 5-13】类方法和实例方法的区别示例。

//CH05\_13.java

public class CH05\_13 {

public static int square(int x){ // 类方法，用于计算 x 的平方

return x\*x;

}

public static void main(String[] args){ int a=5;

//main() 是类方法，直接调用的 square() 方法必须为类方法

System.out.println(a+" 的平方 ="+square(a)); a=7;

System.out.println(a+" 的平方 ="+square(a));

}

}

程序执行结果如下：



【例 5-14】使用类成员统计学生考试成绩的平均分。

//CH05\_14.java class Course{

private String no; // 实例变量：课程编号

private double score; // 实例变量：成绩

static double s=0; // 类变量：总成绩

static int cn=0; // 类变量：成绩个数

public Course(String no,double score){ this.no=no;

this.score=score;

}

第 5 章 类和对象

public double getScore(){ // 实例方法

return score;

}

public static void sumScore(double grade){ // 类方法

s=s+grade;

}

public static void sumCount(){ // 类方法

cn=cn+1;

}

}

public class CH05\_14 {

public static void main(String[] args){ Course c1=new Course("C101",90);

Course.sumScore(c1.getScore()); // 类名调用类方法

Course.sumCount();

Course c2=new Course("C305",87); c2.sumScore(c2.getScore()); // 对象调用类方法c2.sumCount();

System.out.println(" 成绩的平均分 ="+(Course.s/Course.cn));

}

}

程序执行结果如下：



# 小结

本章详细介绍了类的结构，包括成员变量和成员方法；介绍了构造方法和对象的创 建，通过案例对类和对象的定义及使用进行详解。同时，介绍了 static 关键字、包的创建使用和访问权限等知识。通过本章的学习，读者应掌握面向对象设计的过程和类的结构， 构造方法的定义和对象的创建使用，封装的概念及使用，static 关键字，包和访问权限的控制。

习题五

一．选择题

1. 下面关于类的说法，不正确的是（ ）。

A. 类是同种对象的集合和抽象 B. 类属于 Java 语言中的引用数据类型

C. 类就是对 D. 对象是 Java 语言中的基本结构单位

1. 定义了一个猫类 Cat，包含的属性有名字 name、年龄 age、毛色 color，现在要在main() 方法中创建 Cat 类对象，在下列代码中，（ ）是正确的。

**Java** 程序设计

A. Cat cat=new Cat; B. Cat cat=new Cat(); cat.color=" 白色的 "; cat.color=" 白色的 ";

C. Cat cat; D. Cat cat=new Cat( 白色的 ); cat.color=" 白色的 ";

1. 构造方法的作用是（ ）。

A. 访问类的成员变量 B. 初始化成员变量

C. 描述类的特征 D. 保护成员变量

1. 构造方法在（ ）时被调用。

A. 类定义 B. 创建对象

C. 调用对象方法 D. 使用对象的变量

1. 下列构造方法的描述中，正确的是（ ）。
2. 一个类只能有一个构造方法
3. 构造方法没有返回值，应声明返回类型为 void
4. 一个类可包含多个构造方法
5. 构造方法可任意命名
6. 下列命题为真的是（ ）
7. 所有类都必须定义一个构造函数
8. 构造函数必须有返回值
9. 构造函数可以访问类的非静态成员
10. 构造函数必须初始化类的所有数据成员
11. 有一个类 B，下面为其构造方法的声明，正确的是（ ）。

A. void B(int x) { } B. B(int x) { } C. b(int x) { } D. void b(int x) { }

1. 下面关于类的定义，说法正确的是（ ）。

import java.util.\*; import java.io.\*; package com.abc.exam; Class A{ }

1. 该类定义错误，没有使用 public 修饰符修饰类 A
2. 该类定义错误，必须要导入 java.lang 包
3. 该类定义错误，package 声明应该是第一个有效语句
4. 该类定义正确
5. 访问修饰符作用范围由强到弱的是（ ）。

A. private- 默认 -protected-public B. public- 默认 -protected-private

C. private-protected- 默认 -public D. public-protected- 默认 -private

第 5 章 类和对象

1. 下面关于方法的说法，不正确的是（ ）。
   1. Java 中的构造方法名必须和类名相同
   2. 方法体是对方法的实现，包括变量声明和合法语句
   3. 如果一个类定义了构造方法，也可以用该类的默认构造方法
   4. 类的私有方法不能被其他类直接访问
2. 若一个无返回值无参数的方法 method() 是类方法，则该方法正确的定义形式为

（ ）。

* 1. public void method(){ } B. static void method(){ }

C. abstract void method(){ } D. void method(){ }

1. Java 语言中，只限子类或者同一包中类的方法能够访问的访问权限是（ ）。
   1. public B. private C. protected D. 默认
2. 如果一个类的成员变量只能在本类中使用，则该成员变量的修饰符必须是（ ）。
   1. public B. private C. protected D. static
3. 给出下面的程序代码：

public class Test{ private ﬂoat a;

public static void m ( ){ }

}

如何使成员变量 a 被方法 m( ) 访问？（ ）。

A. 将 private float a 改为 protected float a B. 将 private float a 改为 public float a

C. 将 private float a 改为 static float a D. 将 private float a 改为 float a

1. 假设类 A 有如下定义：

class A{

int i;

static String s; void method1(){ }

static void method2(){ }

}

设 a 是类 A 的一个对象，下列语句调用错误的是（ ）。

A. System.out.println(a.i); B. a.method1();

C. A.method1(); D. A.method2();

1. 给定一个 Java 程序代码如下：

public class Test { int count=9;

public void count(){ System.out.println("count="+count++);

**Java** 程序设计

}

public static void main(String[] args){ new Test().count();

new Test().count();

}

}

则编译运行后，输出结果是（ ）。

|  |  |  |  |
| --- | --- | --- | --- |
| A. count=9 | B. count=10 | C. count=10 | D. count=9 |
| count=9 | count=9 | count=10 | count=10 |

二．判断题

1. Java 类中不能存在同名的两个方法。
2. 无论 Java 源程序包含几个类的定义，若该源程序文件以 A.java 命名，编译后生成的都只有一个名为 A 的字节码文件。
3. Java 中类方法可以调用类变量，也可以调用实例变量。
4. Java 通过关键字 package 声明包，package 语句必须位于非空行和非注释行第一句。
5. 构造方法可以有返回值。

三．阅读程序题

1. 下列类 Test 的类体中代码 1~ 代码 5 哪些是错误的？

class Tom{

private int x=120; protected int y=20; int z=11;

private void f(){ x=200;

System.out.println(x);

}

void g(){

x=200;

System.out.println(x);

}

}

public class Test {

public static void main(String[] args){ Tom tom=new Tom();

tom.x=22; //【代码 1】

tom.y=33; //【代码 2】

tom.z=55; //【代码 3】

tom.f(); //【代码 4】

tom.g(); //【代码 5】

第 5 章 类和对象

}

}

1. 请给出程序输出结果。

class B{

int x=100,y=200;

public void setX(int x){ x=x;

}

public void setY(int y){ this.y=y;

}

public int getXYSum(){ return x+y;

}

}

public class Test {

public static void main(String[] args){ B b=new B();

b.setX(-100);

b.setY(-200); System.out.println("sum="+b.getXYSum());

}

}

1. 请给出程序输出结果。

class B{

int n;

static int sum=0; void setN(int n){ this.n=n;

}

int getSum(){

for(int i=1;i<=n;i++){ sum=sum+i;

}

return sum;

}

}

public class Test {

public static void main(String[] args){ B b1=new B(),b2=new B(); b1.setN(3);

b2.setN(5);

int s1=b1.getSum();

**Java** 程序设计

int s2=b2.getSum(); System.out.println("sum="+(s1+s2));

}

}

四．编程题

1. 编写一个类 Book，代表教材。
2. 具有属性：名称（title）、页数（pageNum）和类型（type）。
3. 具有方法 detail()：用来输出每本教材的名称、页数和类型。
4. 具有两个构造方法：无参构造方法（3 个属性用常量赋值）和包含 3 个参数的构造方法（对 3 个属性初始化）。

编写测试类 CH05\_P1，分别调用两个构造方法创建 Book 对象，并调用 detail() 方法进行功能测试。

1. 编写手机类（Phone），有以下属性和行为。
2. 具有属性：品牌（brand）、价格（price）、操作系统（os）和内存（memory）。
3. 具 有 功 能： 查 看 手 机 信 息（about())、 打 电 话（call(String no)） 和 玩 游 戏

（play(String game)）等。

编写主类 CH05\_P2，测试手机的各项功能。

1. 用类描述计算机中 CPU 的速度和内存容量。要求创建 4 个类，名字分别为Computer（计算机类）、CPU（CPU 类）、RAM（内存类）和 CH05\_P3（测试类），其中 CH05\_P3 为主类。

CPU 类 的 属 性： 速 度（speed）。 方 法：setSpeed(int speed) 用 于 设 置 CPU 速 度， getSpeed() 用于获取 CPU 速度。

RAM 类的属性：容量（capacity）。方法：setCapacity(int capacity) 用于设置 RAM 容量， getCapacity() 用于获取 RAM 容量。

Computer 类的属性：RAM（ram）、CPU（cpu）。方法：setRAM(RAM ram) 用于设置计算机的 RAM，setCPU(CPU cpu) 用于设置计算机的 CPU，show() 用于显示计算机的信息。

要求：在主类 CH05\_P3 的 main() 方法中完成如下操作。

1. 创建 CPU 对象，speed 设置为 2200。
2. 创建 RAM 对象，capacity 设置为 4。
3. 创建 Computer 对象，并把创建的 CPU 对象和 RAM 对象通过 set 方法传递给各自的属性。
4. Computer 对象调用 show() 方法，输出该计算机的 CPU 速度和 RAM 容量。

第 5 章 类和对象

五．Java 程序员面试题

* 1. 简述面向对象的基本特征。
  2. 下面的代码是否正确？如有误请改正。

public class Test { public Test(int a){

System.out.println("a="+a);

}

public static void main(String[] args){ new Test();

}

}