第5章

继承与多态

本章学习目标

- 掌握子类定义的方法。
- 理解子类构造方法的执行过程。
- 理解关键字 super 的作用，掌握其使用方法。
- 理解方法重写和多态性的概念。
- 掌握关键字 final 的使用方法。
- 掌握包的定义和导入的方法。

在第 4 章，重点介绍了面向对象程序设计的基本要素——类和对象，可使读者掌握最基本的面向对象程序设计的方法。在此基础上，本章将进一步深入讨论面向对象中的继承和多态机制。首先，本章将介绍继承的基本概念、子类的定义方法，并讨论不同场景下的类成员的访问权限。接着，介绍方法重写的概念，并以此引出类的多态性讨论。最后，本章还将介绍 Java 中的包机制，并进一步讨论与包相关的访问权限。通过本章的学习，读者可以比较全面地掌握面向对象程序设计的方法。

5.1 继承的基本概念

继承是面向对象程序设计的重要特征之一，其含义是以已有类为基础，创建出新类，新类除了具备已有类的成员之外，还可以增加新的成员以满足解决问题的需求。在继承中，已有类称为父类（基类或超类），新类称为子类（派生类）。Java 中支持单继承机制，即每个类只有唯一的父类，C++ 中采用了多继承机制，即每个类可以有多个父类，这是二者在继承上的重要区别。事实上，使用单继承机制的 Java 中，所有的类都继承自 java.lang 包中的 Object 类，这样能够保证所有的对象都拥有某些功能。此外，单继承机制也使垃圾收集器的实现更加便捷。

Java 中的继承遵循层次化的树状结构，如图 5.1 所示，舰艇、客轮和货轮均继承自轮船，而舰艇又派生出 3 个子类，分别是驱逐舰、巡洋舰和航空母舰。这

表示舰艇、客轮和货轮都具有轮船所具备的共同属性，例如，都有长度、宽度、排水量等属性；同时，它们又有各自特有的特征，例如，舰艇具有舰员配置、武器系统、动力装置等属性，客轮具有客舱、娱乐设施、救生设施等属性，货轮有载货量、起重机、升降平台等属性。驱逐舰、巡洋舰和航空母舰都具有舰艇的共同属性（当然也具有轮船的共同属性），此外还拥有各自独有的属性，如航空母舰有舰载机等。需要说明的是，如果定义类时没有明确指明该类的父类，则它默认继承自 java.lang.Object 类。Object 类是 Java 中所有类的共同父类。

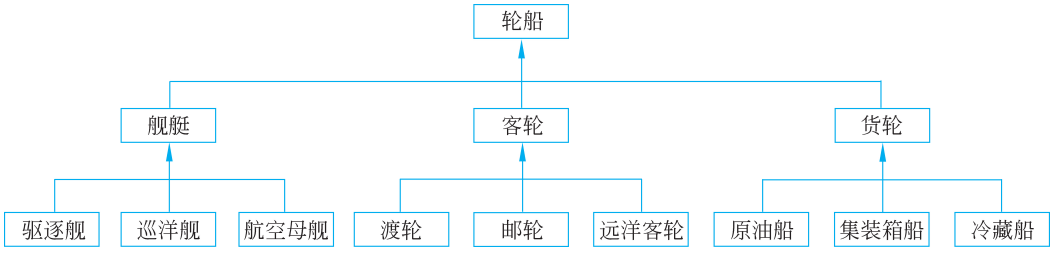

图 5.1　类的层次继承树状图

5.2　子类的定义

在 Java 中，继承是通过定义子类来实现的。一般来说，类中公共的属性放到父类中，而特殊的内容放到子类定义，这样做的明显优势是能够提高代码的复用性。

5.2.1　子类的定义格式

在 Java 中，定义一个子类的语法格式如下。

```
[public] class 子类名 extends 父类名
{
    类体
}
```

事实上，在第 4 章中已经学习过类定义的完整格式，此处只介绍和子类定义相关的内容。关键字 extends 是定义子类最重要的元素，extends 前的类名表示子类，其后的类名表示父类。继承的特性如下。

- 子类可以继承父类中所有非 private 权限的成员。
- 类的继承不改变类成员的访问权限，即子类继承的成员的访问权限和父类中一致。
- 父类的构造方法不能被子类所继承。

下面给出一个子类定义的具体实例。

【例 5-1】　子类定义实例。


ex5_1

```
0001 class NavalVessel
0002 {
```

```
0003    String name;
0004    float length, width;
0005    int displacement;
0006    int maxSpeed;
0007    int numberOfCrews;
0008    void showInfo()
0009    {
0010        System.out.println("【舰艇信息】");
0011        System.out.println("名称: " + name);
0012        System.out.println("长度(米): " + length);
0013        System.out.println("宽度(米): " + width);
0014        System.out.println("排水量(吨): " + displacement);
0015        System.out.println("最大航速(节): " + maxSpeed);
0016        System.out.println("舰员数量: " + numberOfCrews);
0017    }
0018 }
0019 class AircraftCarrier extends NavalVessel
0020 {
0021    int numberOfAircrafts;
0022    int numberOfPoilts;
0023    String takeoffMode;
0024    AircraftCarrier(String n, float len, float wid, int dis, int ms,
0025            int noc, int noa, int nop, String tm)
0026    {
0027        name = n;
0028        length = len;
0029        width = wid;
0030        displacement = dis;
0031        maxSpeed = ms;
0032        numberOfCrews = noc;
0033        numberOfAircrafts = noa;
0034        numberOfPoilts = nop;
0035        takeoffMode = tm;
0036    }
0037    void showInfo()
0038    {
0039        System.out.println("【航空母舰信息】");
0040        System.out.println("名称: " + name);
0041        System.out.println("长度(米): " + length);
0042        System.out.println("宽度(米): " + width);
0043        System.out.println("排水量(吨): " + displacement);
0044        System.out.println("最大航速(节): " + maxSpeed);
0045        System.out.println("舰员数量: " + numberOfCrews);
```

```
0046        System.out.println("舰载机数量: " + numberOfAircrafts);
0047        System.out.println("飞行员数量: " + numberOfPoilts);
0048        System.out.println("舰载机起飞方式: " + takeoffMode);
0049    }
0050 }
0051 public class Example5_1
0052 {
0053    public static void main(String[] args)
0054    {
0055        AircraftCarrier ac = new AircraftCarrier("辽宁号", 304.5f, 75f,
0056                67500, 32, 1960, 36, 626, "滑跃起飞");
0057        ac.showInfo();
0058    }
0059 }
```

【运行结果】

程序运行结果如图 5.2 所示。

图 5.2　子类定义实例输出

【程序说明】

例 5-1 定义了父类 NavalVessel 和子类 AircraftCarrier。父类 NavalVessel 拥有成员 name、length、width、displacement、maxSpeed 和 numberOfCrews，子类 AircraftCarrier 在继承这些成员的基础上又增加了成员 numberOfAircrafts(舰载机数量)、numberOfPoilts(飞行员数量)和 takeoffMode(舰载机起飞模式)，因此，类 AircraftCarrier 一共有 9 个成员变量。程序第 24～36 行定义了 AircraftCarrier 类的构造方法，其作用是利用形参对其所属的 9 个成员变量进行赋值。程序第 37～49 行定义了 showInfo()的方法，该方法与父类中的 showInfo()方法重名，且参数列表也相同(均为空)，这种现象称为方法重写(overriding)。

5.2.2　子类构造方法

当使用 new 运算符创建子类对象时，Java 虚拟机所执行的步骤如下。

第一步：调用父类构造方法，如存在多级继承，则递归地调用上级父类的构造方法。
第二步：根据子类中各个成员的声明顺序，调用各个成员的初始值设定。
第三步：调用子类自身的构造方法。


ex5_2

【例 5-2】 子类构造方法执行顺序实例。

```
0001 class Ship
0002 {
0003    Ship()
0004    {
0005        System.out.println("【调用 Ship 类构造方法,开始建造轮船…】");
0006    }
0007 }
0008 class NavalVessel extends Ship
0009 {
0010    String name;
0011    float length, width;
0012    int displacement;
0013    int maxSpeed;
0014    int numberOfCrews;
0015    {
0016        name = "山东号";
0017        length = 315;
0018        width = 75;
0019        displacement = 65000;
0020        System.out.println("舰艇名称: " + name);
0021        System.out.println("舰艇长度(米): " + length);
0022        System.out.println("舰艇宽度(米): " + width);
0023        System.out.println("排水量(吨): " + displacement);
0024    }
0025    NavalVessel()
0026    {
0027        System.out.println("【调用 NavalVessel 类构造方法,开始建造舰艇…】");
0028    }
0029 }
0030 class AircraftCarrier extends NavalVessel
0031 {
0032    int numberOfAircrafts;
0033    String takeoffMode;
0034    {
0035        numberOfAircrafts = 48;
0036        takeoffMode = "滑跃起飞";
0037        System.out.println("舰载机数量: " + numberOfAircrafts);
0038        System.out.println("舰载机起飞方式: " + takeoffMode);
```

```
0039    }
0040    AircraftCarrier()
0041    {
0042        System.out.println("【调用 AircraftCarrier 类构造方法,开始建造航空母
            舰…】");
0043    }
0044 }
0045 public class Example5_2
0046 {
0047    public static void main(String[] args)
0048    {
0049        AircraftCarrier ac = new AircraftCarrier();
0050    }
0051 }
```

【运行结果】

程序运行结果如图 5.3 所示。

图 5.3　子类构造方法执行顺序实例输出

【程序说明】

程序定义了 3 个类，Ship 是父类，NavalVessel 是 Ship 的子类，AircraftCarrier 是 NavalVessel 的子类。程序第 49 行创建了类 AircraftCarrier 的对象 ac。从图 5.3 的输出可以分析得出，欲创建子类的对象，必先递归地调用其父类的构造方法，然后执行子类的成员初始化（例如程序第 15～24 行、第 34～39 行），最后再执行子类自身的构造方法。

5.2.3　super 关键字

继续分析例 5-2，在类 AircraftCarrier 的构造方法中，并未看到任何调用其父类 NavalVessel 类构造方法的语句。事实上，如果在子类构造方法中没有显式地调用父类构造方法，则系统会自动在子类构造方法第一句调用父类的无参构造方法。

由于构造方法是不能被继承的，在子类中调用父类的构造方法需要使用到关键字 super，其语法格式为：

```
super(形参列表);
```

接下来,对例 5-1 的代码进行修改。


ex5_3

【例 5-3】 super 调用父类构造方法实例。

```
0001 class NavalVessel
0002 {
0003    String name;
0004    float length, width;
0005    int displacement;
0006    int maxSpeed;
0007    int numberOfCrews;
0008    NavalVessel(String name, float length, float width, int displacement,
0009            int maxSpeed, int numberOfCrews)
0010    {
0011        this.name = name;
0012        this.length = length;
0013        this.width = width;
0014        this.displacement = displacement;
0015        this.maxSpeed = maxSpeed;
0016        this.numberOfCrews = numberOfCrews;
0017    }
0018    void showInfo()
0019    {
0020        System.out.println("【舰艇信息】");
0021        System.out.println("名称: " + name);
0022        System.out.println("长度(米): " + length);
0023        System.out.println("宽度(米): " + width);
0024        System.out.println("排水量(吨): " + displacement);
0025        System.out.println("最大航速(节): " + maxSpeed);
0026        System.out.println("舰员数量: " + numberOfCrews);
0027    }
0028 }
0029 class AircraftCarrier extends NavalVessel
0030 {
0031    int numberOfAircrafts;
0032    int numberOfPoilts;
0033    String takeoffMode;
0034    AircraftCarrier(String name, float length, float width, int displacement,
0035            int maxSpeed, int numberOfCrews, int numberOfAircrafts,
                int numberOfPoilts, String takeoffMode)
0036    {
0037        super(name, length, width, displacement, maxSpeed, numberOfCrews);
```

```
0038        this.numberOfAircrafts = numberOfAircrafts;
0039        this.numberOfPoilts = numberOfPoilts;
0040        this.takeoffMode = takeoffMode;
0041    }
0042    void showInfo()
0043    {
0044        super.showInfo();
0045        System.out.println("【航空母舰信息】");
0046        System.out.println("舰载机数量: " + numberOfAircrafts);
0047        System.out.println("飞行员数量: " + numberOfPoilts);
0048        System.out.println("舰载机起飞方式: " + takeoffMode);
0049    }
0050 }
0051 public class Example5_3
0052 {
0053    public static void main(String[] args)
0054    {
0055        AircraftCarrier ac = new AircraftCarrier("辽宁号", 304.5f, 75f,
0056                67500, 32, 1960, 36, 626, "滑跃起飞");
0057        ac.showInfo();
0058    }
0059 }
```

【运行结果】

程序运行结果如图 5.4 所示。

图 5.4　super 调用父类构造方法实例输出

【程序说明】

程序第 37 行是子类显式地调用父类的构造方法，通过它完成相关成员变量（name、length、width、displacement、maxSpeed 和 numberOfCrews）的初始化；程序第 38～40 行，完成其余 3 个成员变量 numberOfAircrafts、numberOfPoilts 和 takeoffMode 的赋值，一

定程度上减少了冗余代码;程序第 44 行关键字 super 的作用并非是调用父类构造方法,而是访问父类的其他成员方法。

下面来介绍关键字 super 的其他功能。关键字 super 除了可以调用父类构造方法外,还可以访问父类中的成员变量和成员方法,其语法格式分别为:

```
super.父类成员变量名
```

和

```
super.父类成员方法名
```


ex5_4

【例 5-4】 super 访问父类成员实例。

```
0001 class NavalVessel
0002 {
0003     String name = "南昌号";
0004     void showInfo()
0005     {
0006         System.out.println("【舰艇信息】");
0007         System.out.println("本舰艇名称: " + name);
0008     }
0009 }
0010 class AircraftCarrier extends NavalVessel
0011 {
0012     String name = "山东号";
0013     void showInfo()
0014     {
0015         super.showInfo();
0016         System.out.println("【航空母舰信息】");
0017         System.out.println("舰艇名称: " + super.name);
0018         System.out.println("航空母舰名称: " + this.name);
0019     }
0020 }
0021 public class Example5_4
0022 {
0023     public static void main(String[] args)
0024     {
0025         AircraftCarrier ac = new AircraftCarrier();
0026         ac.showInfo();
0027     }
0028 }
```

【运行结果】

程序运行结果如图 5.5 所示。

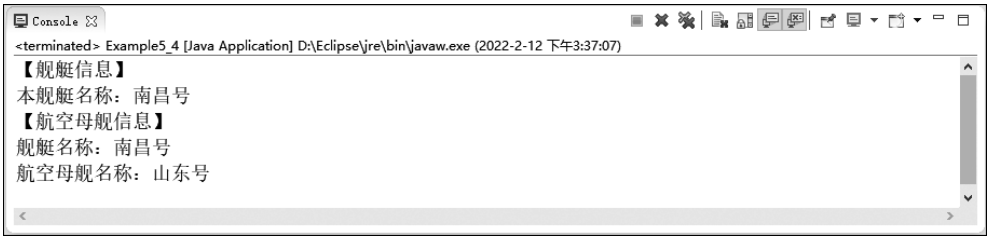

图 5.5　super 访问父类成员实例输出

【程序说明】

程序定义了两个类，分别是父类 NavalVessel 和子类 AircraftCarrier，两个类中都有成员变量 name。当子类和父类拥有相同的成员变量时，子类会隐藏从父类继承的成员变量。NavalVessel 类对象默认的 name 为“南昌号”，AircraftCarrier 类对象默认的 name 为“山东号”。程序第 15 行，子类通过关键字 super 访问父类的 showInfo()方法，因此输出“本舰艇名称：南昌号”；程序第 17 行，super.name 访问父类对象被隐藏的成员变量 name，因此输出“舰艇名称：南昌号”；程序第 18 行，this.name 访问的是子类对象的 name，因此输出“航空母舰名称：山东号”。

【注意】

- 使用 super 可以访问父类的构造方法，且必须放置在子类构造方法的第一条语句。
- 使用 super 可以访问父类中被子类所隐藏的成员变量和成员方法。

5.3　继承的访问权限

在前面章节中已经介绍了访问权限修饰符，并通过表格的形式给出了不同的访问权限修饰符在各种情形下的可访问性。下面将结合继承，进一步探讨成员变量（成员方法同理）在子类和在同一包中其他类的访问权限。包的概念在后续章节才会学习到，此处只需知道同一个源文件中的类必然属于同一个包，至于不同包中的类的成员的访问权限，将在后续章节继续讨论。

【例 5-5】　访问权限实例。

ex5_5

```
0001 class A
0002 {
0003     public int a;
0004     private int b;
0005     protected int c;
0006     int d;
0007     A(int a, int b, int c, int d)
0008     {
0009         this.a = a;
```