上机实验 5
继承和多态

实验目的：

- ◆ 掌握继承的概念和实现。
- ◆ 掌握方法的重写。
- ◆ 掌握抽象类和抽象方法的概念和实现。
- ◆ 掌握接口的概念和实现。
- ◆ 掌握多态的概念，能够使用多态思想解决具体问题。

5.1 类的继承

继承是面向对象的另一大特征，它用于描述类的所属关系，多个类通过继承形成一个关系体系。继承是在原有类的基础上扩展新的功能，实现了代码的复用。

实验 5-1 学生类

【内容】

设计父类 People，People 类具有以下成员。

(1) 包含 3 个属性：姓名(name)、年龄(age)和性别(sex)。

(2) 两个构造方法。一个是无参数构造方法，用于输出“***创建了父类对象***”另一个构造方法需要传递一个参数，为姓名(name)赋值，最后输出变量 name 的值及“创建了父类对象”。

(3) 两个实例方法。eat()方法输出“已经吃饭了”；work()方法输出“开始工作了”。

设计子类 Student，Student 类是 People 的子类，Student 类具有以下成员。

(1) 增加两个属性：学号(id)和年级(grade)。

(2) 含有一个参数的构造方法，为姓名(name)赋值，输出变量 name 的值及“创建了子类对象”。

(3) work()方法输出“开始上学了”。

(4) exam()方法输出“我要考个好成绩”。

【思路】

(1) 父类 People 三个属性的数据类型分别为：姓名(name)和性别(sex)是 String 类

型，年龄(age)是 int 类型。

(2) 父类 People 中第一个无参数的构造方法，使用 System.out.println()方法输出“***创建了父类对象***”。第二个构造方法需要传递一个参数，这个参数的值赋给姓名(name)，然后使用 System.out.println()方法输出 name 的值＋“创建了父类对象”。

(3) eat()方法和 work()方法使用 System.out.println()方法分别输出“已经吃饭了”和“开始工作了”。

(4) 子类 Student 中新增学号(id)和年级(grade)两个属性的数据类型都是 String 类型。

(5) 子类 Student 中的一个参数的构造方法，需要传递一个参数，这个参数的值赋给姓名(name)，然后使用 System.out.println()方法输出 name 的值＋“创建了子类对象”。

(6) 在子类 Student 中重写父类的成员方法 work()，使用 System.out.println()方法输出“开始上学了”。定义 exam()方法，使用 System.out.println()方法输出“我要考个好成绩”。

【代码】

```
package five;
public class StudentInheritDemo {
    public static void main(String[] args) {
        People p=new People("王鹏");
        p.eat();
        p.work();
        System.out.println("----------------------");
        Student s=new Student("张丽");
        s.eat();
        s.work();
        s.exam();
    }

}
class People{
    String name;
    int age;
    String sex;
    People(){
        System.out.println("***创建了父类对象***");
    }
    People(String name){
        this.name=name;
        System.out.println(name+"***创建了父类对象***");
    }
    public void eat(){
        System.out.println("已经吃饭了");
    }
```

```
    public void work(){
        System.out.println("开始工作了");
    }
}
class Student extends People{
    String id;
    String grade;
    Student(String name){
        this.name=name;
        System.out.println(name+"***创建了子类对象***");
    }
    public void work(){
        System.out.println("开始学习了");
    }
    public void exam(){
        System.out.println("我要考个好成绩");
    }
}
```

【运行结果】

如图5-1所示，在测试类StudentInheritDemo中，使用父类创建对象p，调用一个参数的构造方法，输出“王鹏***创建了父类对象***”，使用p.eat()输出“已经吃饭了”，使用p.work()输出“开始工作了”。使用子类创建对象s，调用一个参数的构造方法，子类继承父类，子类的构造方法会自动调用父类无参数的构造方法，因此先输出“***创建了父类对象***”，然后输出“张丽***创建了子类对象***”。子类重写了父类的work()方法，使用s.work()时，会调用子类重写的work()方法，输出“开始学习了”。

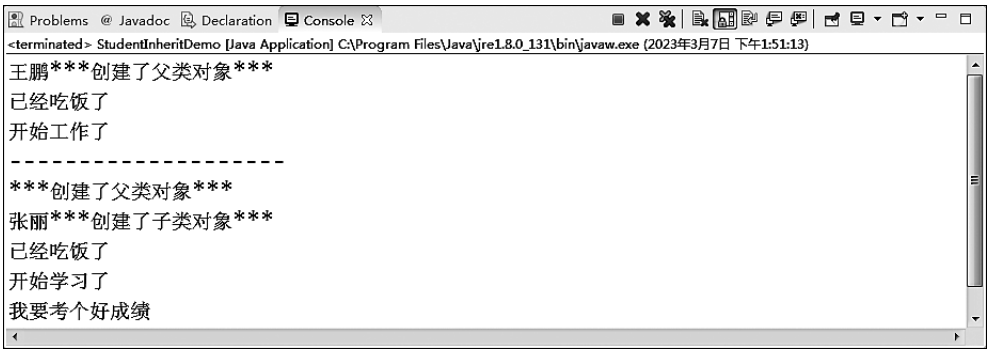

图5-1　学生类

实验5-2　员工类

【内容】

设计员工类Employee，员工类具有以下成员。

(1) 包括两个属性：姓名(name)和基本工资(salary)。

(2) 两个构造方法。一个是无参数的构造方法;一个是包括两个参数的构造方法。

(3) 两个实例方法。raiseSalary()方法用于调整员工的基本工资;toString()方法输出员工的基本信息。

设计经理类 Manager,员工有一类人员是经理。经理除了有普通员工的基本工资外,还有额外的奖金,因此经理的总工资是基本工资和奖金的总和。经理类具有以下成员。

(1) 增加一个属性:奖金(bonus)。

(2) 两个构造方法。一个是无参数的构造方法;一个是包括三个参数的构造方法。

(3) 增加一个实例方法。getSalary()方法用于获得经理的总工资。

【思路】

(1) 员工类 Employee 两个属性的数据类型分别为:姓名(name)是 String 类型,基本工资(salary)是 double 类型。

(2) 员工类 Employee 的无参数的构造方法的方法体为空;两个参数的构造方法可以为姓名(name)和基本工资(salary)赋值。

(3) raiseSalary()方法需要传递一个 double 类型的参数,此方法按百分比为员工调整工资;toString()方法输出员工的姓名和基本工资。

(4) 经理类 Manager 是员工类 Employee 的子类,添加了一个属性奖金(bonus),数据类型是 double 类型。

(5) 经理类 Manager 的无参数的构造方法的方法体为空;三个参数的构造方法可以为姓名(name)、基本工资(salary)和奖金(bonus)赋值,通过 super 关键字调用父类的构造方法。

(6) getSalary()方法是获得经理的总工资,是基本工资(salary)和奖金(bonus)的和。

(7) 重写了父类的 toString()方法,输出经理的姓名、基本工资、奖金和总工资。

【代码】

```
package five;
public class EmployeeDemo {
    public static void main(String[] args) {
        Employee e=new Employee("王丽",3500);
        e.raiseSalary(20);
        System.out.println(e);
        Manager m=new Manager("刘宇",5000,1000);
        System.out.println(m);
    }
}
class Employee{
    String name;
    double salary;
    public Employee(){
    }
    public Employee(String name,double salary){
        this.name=name;
        this.salary=salary;
```

```
    }
    public void raiseSalary(double percent){
        salary=salary+salary*percent/100;
    }
    @ Override
    public String toString() {
        return "Employee [name=" +name +", basicsalary=" +salary +"]";
    }
}
class Manager extends Employee{
    private double bonus;
    public Manager(){
    }
    public Manager(String name,double salary,double bonus){
        super(name,salary);
        this.bonus=bonus;
    }
    public double getSalary(){
        return salary+bonus;
    }
    @ Override
    public String toString() {
        return "Manager [name=" +name +", basicsalary="
              +salary +",bonus=" +bonus +",totalsalary="
              +this.getSalary()+"]";
    }
}
```

【运行结果】

如图5-2所示，在测试类EmployeeDemo中，使用Employee创建对象e，调用两个参数的构造方法对属性name和salary赋值，通过raiseSalary()方法为员工调整工资，然后调用toString()方法输出员工的基本信息。使用子类Manager创建对象m，调用构造方法为属性赋值，通过toString()方法输出经理的基本信息，在toString()方法中调用了getSalary()方法获得经理的总工资。

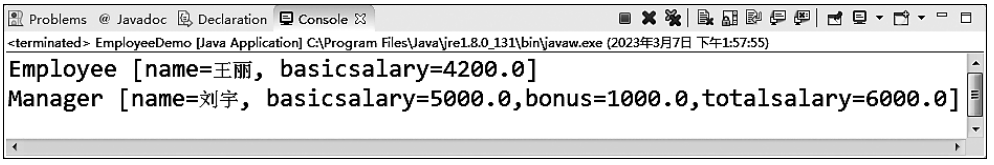

图5-2　员工类

5.2　方法的重写

在继承关系中，子类从父类中继承了可访问的方法，但有时从父类继承下来的方法不能

完全满足子类需要，这时就需要在子类方法中修改父类方法，即子类重新定义从父类中继承的成员方法，这个过程称为方法重写或覆盖。

实验 5-3　等边三角形类

【内容】

三角形具有三条边长，根据三条边可以计算周长和面积。等边三角形是一种特殊的三角形，特点是三条边相等。设计三角形类和等边三角形类，计算面积、周长以及输出三角形的信息。

【思路】

(1) 设计三角形类，包括三个属性，表示三条边，数据类型都为 int。

(2) 三角形类包括两个构造方法，一个是无参数的构造方法；一个是包括三个参数的构造方法，可以为三条边赋值。

(3) 定义三个实例方法，分别计算周长、面积和输出三角形的信息。

(4) 设计等边三角形，等边三角形是三角形的子类，包括两个构造方法，一个是无参数的构造方法；一个是包括一个参数的构造方法，此构造方法通过 super 关键字调用父类的构造方法为三条边赋值。

(5) 由于等边三角形三条边相等，需要重写父类中的三个实例方法。

【代码】

```
package five;
public class TriangleDemo {
    public static void main(String[] args) {
        EquilateralTriangle et=new EquilateralTriangle(5);
        System.out.println(et);
        System.out.println("Perimeter is:"+et.getPerimeter());
        System.out.println("area is:"+et.getArea());
    }
}
class Triangle{
    int a,b,c;
    public Triangle(){
    }
    public Triangle(int a,int b,int c){
        this.a=a;
        this.b=b;
        this.c=c;
    }
    public int getPerimeter(){
        return a+b+c;
    }
    public double getArea(){
        double s=(double)(a+b+c)/2;
```

```
        double area=Math.sqrt(s * (s-a) * (s-b) * (s-c));
        return area;
    }
    public String toString() {
        return "Triangle [a=" +a +", b=" +b +", c=" +c +"]";
    }
}
class EquilateralTriangle extends Triangle{
    public EquilateralTriangle(){
    }
    public EquilateralTriangle(int l){
        super(l,l,l);
    }
    public int getPerimeter(){
        return 3 * a;
    }
    public double getArea(){
        return Math.sqrt(3) * a * a/4;
    }
    public String toString() {
        return "EquilateralTriangle [a=b=c="+a+"]";
    }
}
```

【运行结果】

如图 5-3 所示，创建边长等于 5 的等边三角形，通过重写父类的计算面积、计算周长和输出三角形信息的三个方法，在控制台中显示这个等边三角形的相关信息。

图 5-3　等边三角形

实验 5-4　动物类

【内容】

不同的动物会发出不同的叫声，小猫喵喵叫，小狗汪汪叫。设计不同的类，并通过重写方法实现不同动物的叫声。

【思路】

(1) 设计动物类 Animal，包括 cry()方法，输出内容“animal cry…”。

(2) 设计小猫类 Cat，小猫类继承了动物类 Animal，并重写了 cry()方法，输出内容“cat miaomiaomiao…”。

(3) 设计小狗类 Dog,小狗类继承了动物类 Animal,并重写了 cry()方法,输出内容“dog wangwangwang…”。

【代码】

```
package five;
public class AnimalDemo {
    public static void main(String[] args) {
        Animal a=new Animal();
        a.cry();
        Cat c=new Cat();
        c.cry();
        Dog d=new Dog();
        d.cry();
    }
}
class Animal{
    public void cry(){
        System.out.println("animal cry…");
    }
}
class Cat extends Animal{
    public void cry(){
        System.out.println("cat miaomiaomiao…");
    }
}
class Dog extends Animal{
    public void cry(){
        System.out.println("dog wangwangwang…");
    }
}
```

【运行结果】

如图 5-4 所示,在测试类中创建动物类对象,调用 cry()方法输出“animal cry…”,创建子类小猫类的对象,调用重写后的 cry()方法输出“cat miaomiaomiao…”,创建子类小狗类的对象,调用重写后的 cry()方法输出“dog wangwangwang…”。

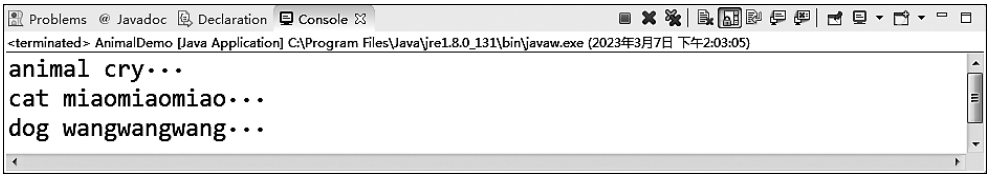

图 5-4 动物类

5.3　抽象类和接口

对于面向对象编程来说，抽象是它的特征之一。在 Java 中，可以通过两种形式来体现面向对象的抽象特点：接口和抽象类。

实验 5-5　打印机类

【内容】

设计打印机类 Printer，具有方法 printInfo()，用来输出打印机的类型。打印机有针式打印机、激光打印机和喷墨打印机，设计三个类 NeedlePrinter、LaserPrinter、InkjetPrinter，继承 Printer 类，具体实现 printInfo()方法。

【思路】

(1) Printer 是父类，父类中具有 printInfo()方法，但父类并没有打印机类型的信息，因此 printInfo()方法应为抽象方法，父类 Printer 是抽象类。

(2) NeedlePrinter、LaserPrinter、InkjetPrinter 是子类，继承了父类 Printer，因此需要重写父类的抽象方法 printInfo()，用于输出打印机的类型信息。

【代码】

```
package five;
public class PrinterDemo {
    public static void main(String[] args) {
        NeedlePrinter np=new NeedlePrinter();
        np.printInfo();
        LaserPrinter lp=new LaserPrinter();
        lp.printInfo();
        InkjetPrinter ip=new InkjetPrinter();
        ip.printInfo();
    }
}
abstract class Printer{
    public abstract void printInfo();
}
class NeedlePrinter extends Printer{
    public void printInfo() {
        System.out.println("针式打印机");
    }
}
class LaserPrinter extends Printer{
    public void printInfo() {
        System.out.println("激光打印机");
    }
}
```

```
class InkjetPrinter extends Printer{
    public void printInfo() {
        System.out.println("喷墨打印机");
    }
}
```

【运行结果】

如图 5-5 所示，在测试类中分别创建针式打印机、激光打印机和喷墨打印机的对象，分别调用 printInfo()方法显示打印机的类型信息。

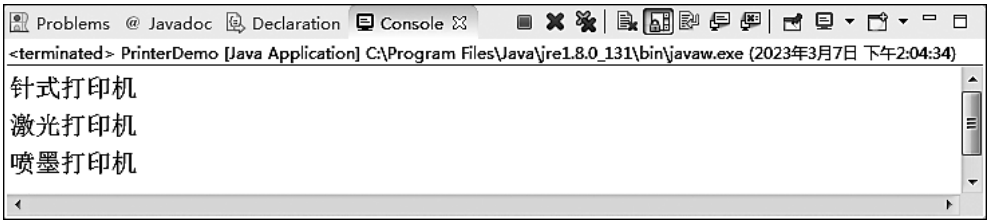

图 5-5　打印机类

实验 5-6　操作系统类

【内容】

设计操作系统类 OperatingSystem，具有方法 printSystemInfo ()，用来输出操作系统的类型。操作系统有 Windows 操作系统、Linux 操作系统等，设计两个类 WindowsOperatingSystem 和 LinuxOperatingSystem，继承 OperatingSystem 类，具体实现 printSystemInfo()方法。

【思路】

(1) OperatingSystem 是父类，父类中具有 printSystemInfo()方法，但父类并没有操作系统类型的信息，因此 printSystemInfo()方法应为抽象方法，父类 OperatingSystem 是抽象类。

(2) WindowsOperatingSystem 和 LinuxOperatingSystem 是子类，继承了父类 OperatingSystem，因此需要重写父类的抽象方法 printSystemInfo ()，用于输出操作系统的类型信息。

【代码】

```
package five;
public class OperatingSystemDemo {
    public static void main(String[] args) {
        WindowsOperatingSystem wop=new WindowsOperatingSystem();
        wop.printSystemInfo();
        LinuxOperatingSystem lop=new LinuxOperatingSystem();
        lop.printSystemInfo();
    }
}
abstract class OperatingSystem{
    public abstract void printSystemInfo();
```