实验四 继承与多态

实验目的

1. 理解继承的概念及意义。
2. 掌握父类、子类中实例变量与方法的继承关系。
3. 理解多态的含义，及多态在继承中的应用。
4. 能够在编程中合理的设计并应用继承关系。

实验预习

1. 继承的意义

前面出现过的题目：在图形接口画出 Square、Circle 与 Triangle。当用户点选图形时，图形需要以中心点为轴，进行 360° 顺时针旋转，并依据形状的不同播放不同的 AIF 音效文件。我们编写了大致如图 4.1 的程序，对照图 3.1 的中两种编程思想，Java 的语言看起来要复杂很多。

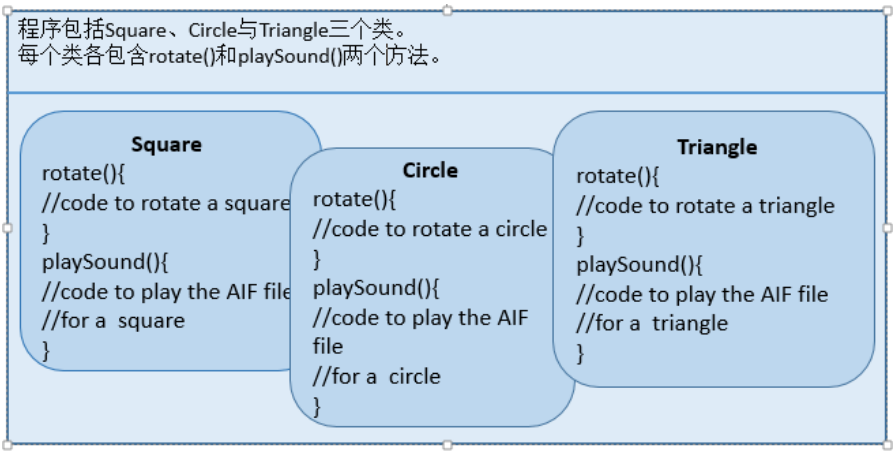

图 4.1 无继承关系的三个类

但如果我们引入继承的概念以后，解决问题会变得容易很多。不论是 Square、Circle 与 Triangle，它们都包含 rotate()方法和 playSound()方法，而且都可以被称为 Shape。那么我们提取出一个名为 Shape 的类，并为这个类编写 rotate()方法和 playSound()方法，再将 Square、Circle 与 Triangle 三个类以继承的关系连接到 Shape 这个类。如图 4.2 所示，Square、Circle 与 Triangle 三个类可以直接使用父类 Shape 的 rotate()方法和 playSound()方法，避免了代码的重复编写。

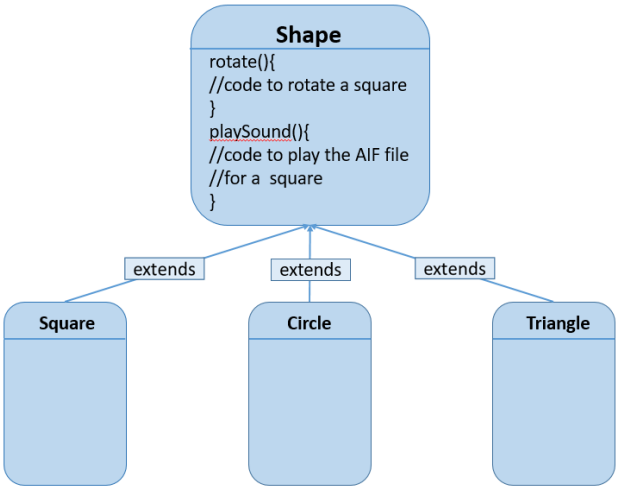

图 4.2 继承 Shape 类的三个类

2. 继承的规则

子类会继承父类所有 public 和 protected 修饰的实例变量和方法，但不会继承父类 private 修饰的变量和方法。实例变量不能被覆盖掉，但是方法可以被子类重写的方法覆盖，当某个方法在子类中被覆盖过，调用子类对象的这个方法时会调用覆盖过的版本。如果在子类的对象的方法中还需调用被覆盖过的父类方法，则需使用 super 关键字（super 关键字的使用在附录 B 中有详细说明）。

方法覆盖（重写）需要遵守两项规则：

1. 重写的方法参数必须要一样，且返回值类型必须要兼容。

即不论父类使用了哪种参数，覆盖此方法的子类一定要使用相同的参数；不论父类声明的返回类型是什么，子类必须要声明返回一样的类型或该类型的子类。

2. 不能降低方法的存取权限。

这代表存取权必须相同，或者更为开放。举例来说，你不能覆盖掉一个公有的方法并将它标记为私有。目前我们只看到过 public 与 private 两种权限，除此之外还有 protected 与 default 权限。详细说明请见表 4.1

表 4.1 存储权限说明

修饰符	可修饰成员	备注
public	类、实例变量、构造方法、方法	被public修饰的成员，可以在任何一个类中被调用。
protected	实例变量、构造方法、方法	不能修饰类（此处指外部类，不考虑内部类），被protected修饰的成员，能在定义它们的类中、同包的类中被调用。
default（默认权限）	类、实例变量、构造方法、方法	默认权限可以不写任何关键字，同包权限，成员能在定义它们的类中、同包的类中被调用。
private	实例变量、构造方法、方法	不能修饰类（此处指外部类，不考虑内部类），被private修饰的成员只能在定义它的类中使用，在其他类中不能被调用。

用 final 修饰符修饰的类，表示它是不能被继承，即它不能有子类。如果用 final 修饰符修饰了某个方法，即表示此方法不能被子类覆盖。

3. 继承关系的建立

建立合理继承关系的步骤：

1. 找出具有共同属性和行为的对象。
2. 设计代表共同状态和行为的类。
3. 决定子类是否需要让某项行为（方法的实现）有特定不同的运作方式。
4. 通过寻找使用共同行为的子类来找出更多抽象化的机会。
5. 完成类的继承层次。

在建立继承关系后，我们通过 IS-A 测试来检验继承层次的合理性。当一个类“D”继承自另外一个类“C”时，我们会说是子类“D”继承父类“C”。那么若要知道某个类是否应该要继承另一个类的时候，我们看“D IS A C”是否成立。例如三角形是一个多边形（成立），口腔科医生是一个医生（成立），键盘是一个鼠标（不成立）。

4. 多态

当我们使用父类声明引用变量，使用子类实例化时，如 Animal myDog = new Dog();，通过父类的引用变量调用方法时，实际调用的是具体子类覆盖的方法，即一个 Animal 类型的变量会在不同场合下表现为不同的行为，这种特征叫多态。参考以下例子，着重理解注释部分。

```
Animal[] animals = new Animal[5];
animals[0] = new Dog();
animals[1] = new Cat();
animals[2] = new Wolf();
animals[3] = new Hippo();
animals[4] = new Lion();
for(int i = 0; i < animals.length; i++){
  animals[i].eat(); //当 i 为 0 时，会调用 Dog 的 eat()
  animals[i].roam(); //当 i 为 1 时，会调用 Cat 的 roam()
}
class Vet{
  public void giveShot(Animal a){
  a.makeNoise();
  /*a 参数可以用任何 Animal 的类型对象来当传入。执行 makeNoise()的时候，不管它引
    用的对象是什么，该对象都会执行 makeNoise()*/
  }
}
class PetOwner{
  public void start(){
    Vet v = new Vet();
    Dog d = new Dog();
    Hippo h = new Hippo();
    v.giveShot(d); //执行 Dog 的 makeNoise()
    v.giveShot(h); //执行 Hippo 的 makeNoise()
  }
}
```

5. 概念理解练习

1. 设计继承关系图

根据图 4.3 给出的动物特点，依照上文中建立继承关系的步骤，在空白处给这些动物画出一个合理的继承关系图（树状图）。上文中的五步在此处可具体表现为：

a. 找出 6 种动物的共同点，不仅包括动物的状态（实例变量，如大小、食物类型等），还包括动物的行为（方法，如叫声、睡眠、获取食物方式等）。

b. 这些对象都是动物，我们可以用 Animal 作为父类的名称，然后将所需要的实例变量和方法加进图中（参考图 4.2）。

c. 观察 Animal 这个类，我们会发现各个子类，在方法上会有不同，思考哪些父类的方法在子类中需要被覆盖，或者需要加入哪些新方法及实例变量。

d. 通过 c 步骤后，总结一下子类中被覆盖的方法、新加入的方法和实例变量，找出共同点，为这些有共同特点的子类再抽象出一个父类，此时的父类应该为 Animal 类的子类。

e. 经过以上步骤后，画出一个清晰的继承关系的树形结构图，并用“IS A”来测试一下，继承结构是否合理。

图 4.3 设计继承关系

2. 写出实例变量及方法

根据上图设计好的树形继承关系，填写表 4.2，为每个类写出所包含的实例变量和方法，所有子类只需写出覆盖后的方法、新增实例变量和方法。

表 4.2 说明继承关系的类表

类名称	父类名称	子类名称	实例变量	方法

基础练习

1. 程序填空

将以下给出的类名、方法名、关键字、变量等片段填入程序空缺处（每个片段只能用一次），使程序能正确运行出：drift drift hoist sail。

片段：extends extends Boat Boat Boat void void void void static length length public public Sailboat Sailboat move move move setLength return drift b2 b3 hoist sail int len

程序：

```
public class TestBoats{
  ______ ______ ______ main(String[] args){
    ______ b1 = new Boat();
    Sailboat b2 = new ______();
    Rowboat ______ = new Rowboat();
    b2.setLength(32);
    b1.______();
    b3.______();
    ______.move();
  }
}
public class ______{
  private int ______;
  ______ void ______(______){
    length = len;
  }
  public int getLength(){
    ______ ______;
  }
  public ______ move(){
    System.out.print("______");
  }
}
public class Rowboat _______ _______{
  public ______ rowTheBoat(){
    System.out.print("stroke natasha");
  }
}
  public class _______ ________ Boat{
    public _____ ______(){
      System.out.print("________");
  }
}
```

2. 补充程序

读程序，并在 TestInterest 类中填写语句，使程序能正确输出：

8000 元存 5 年零 216 天

利息是 1572.800 元

程序：

```
public class Bank{
  int saveMoney;
  int year;
  double interest;
  public double computerInterest(){
    interest = year * 0.035 * saveMoney;
    return interest;
  }
}
public class ConstructionBank extends Bank{
  double year;
  public double computerInterest(){
    super.year = (int)year;
    double remainNumber = year - (int)year;
    int day = (int)(remainNumber * 1000);
    interest = super.computerInterest() + day * 0.0001 * saveMoney;
    System.out.printf("%d 元存%d 年零%d 天\n",saveMoney,super.year,day);
    return interest;
  }
}
public class TestInterest{
  public static void main(String args[]){
    int amount = 8000;
    bank.year = 5.216;
    ____________________________________
    ____________________________________
    ____________________________________
    ____________________________________
  }
}
```

3. 编写程序

现有一套木质七巧板需要用赤、橙、黄、绿、青、蓝、紫 7 种颜色进行油漆，七块板拼成的正方形边长为 10 米，厚度为 1 米，颜色分布如图 4.4 所示。每平方米面积使用油漆一小桶，编写程序计算出油漆一套七巧板需用 7 种油漆各多少桶。

要求：程序中需运用到继承与多态的概念。

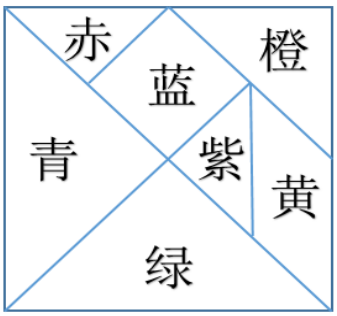

图 4.4 七巧板