源码下载

# 上机实践5 子类与继承

## 实验1 子类和父类

**❶ 相关知识点**

由继承得到的类称为子类，被继承的类称为父类(超类)，Java 不支持多重继承，即子类只能有一个父类。人们习惯地称子类与父类的关系是“is-a”关系。

如果子类和父类在同一个包中，那么子类自然地继承了其父类中不是 private 的成员变量作为自己的成员变量，并且也自然地继承了父类中不是 private 的方法作为自己的方法，继承的成员变量或方法的访问权限保持不变。当子类和父类不在同一个包中时，父类中的 private 和友好访问权限的成员变量不会被子类继承，也就是说子类只继承父类中的 protected 和 public 访问权限的成员变量作为子类的成员变量；同样，子类只继承父类中的 protected 和 public 访问权限的方法作为子类的方法。

当子类声明的成员的变量的名字和从父类继承来的成员变量的名字相同时，将隐藏掉所继承的成员变量。方法重写是指子类中定义一个方法，这个方法的类型和父类的方法的类型一致或者是父类的方法的类型的子类型，并且这个方法的名字、参数个数、参数的类型和父类的方法完全相同。子类如此定义的方法称作子类重写的方法。

子类继承的方法所操作的成员变量一定是被子类继承或隐藏的成员变量。重写方法既可以操作继承的成员变量、调用继承的方法，也可以操作子类新声明的成员变量、调用新定义的其他方法，但无法操作被子类隐藏的成员变量和方法。

**❷ 实验目的**

本实验的目的是让学生巩固下列知识点：

(1) 子类的继承性。

(2) 子类对象的创建过程。

(3) 成员变量的继承与隐藏。

(4) 方法的继承与重写。

**❸ 实验要求**

编写程序模拟中国人、美国人是人，北京人是中国人。除主类外，该程序中还有 People、Chinese、American 和 Beijingman 4 个类。要求如下：

(1) People 类有权限是 protected 的 double 型成员变量 height 和 weight，以及 public void speakHello()、public void averageHeight()和 public void averageWeight()方法。

(2) Chinese 类是 People 的子类，新增了 public void chinaGongfu()方法，要求 Chinese 重写父类的 public void speakHello()、public void averageHeight()和 public void averageWeight()方法。

(3) American 类是 People 的子类，新增了 public void americanBoxing()方法，要求

American 重写父类的 public void speakHello()、public void averageHeight()和 public void averageWeight()方法。

(4) Beijingman 类是 Chinese 的子类，新增了 public void beijingOpera()方法，要求 Beijingman 重写父类的 public void speakHello()、public void averageHeight()和 public void averageWeight()方法。

People 类、Chinese 类、American 类和 Beijingman 类的 UML 图如图 5.1 所示。

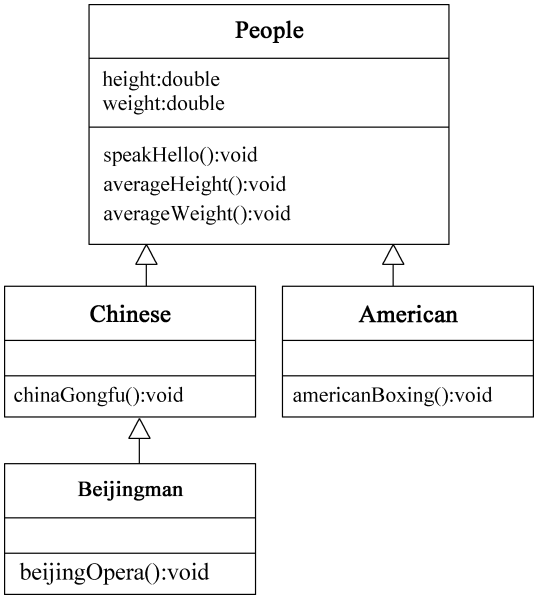


图 5.1　类的 UML 图

**❹ 程序运行效果**

程序运行效果如图 5.2 所示。

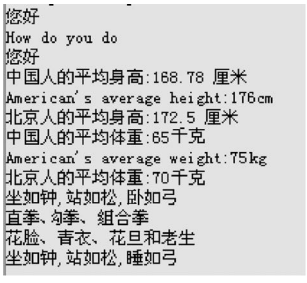


图 5.2　成员的继承与重写

**❺ 程序模板**

请按模板要求将【代码】替换为 Java 程序代码。

**People. java**

```
public class People {
    protected double weight,height;
    public void speakHello() {
      System.out.println("yayayaya");
    }
    public void averageHeight() {
      height = 173;
      System.out.println("average height:" + height);
    }
    public void averageWeight() {
      weight = 70;
      System.out.println("average weight:" + weight);
    }
}
```

**Chinese. java**

```
public class Chinese extends People {
    public void speakHello() {
      System.out.println("您好");
    }
    public void averageHeight() {
      height = 168.78;
      System.out.println("中国人的平均身高:" + height + " 厘米");
    }
    【代码 1】      //重写 public void averageWeight()方法,输出"中国人的平均体重:65 千克"
    public void chinaGongfu() {
      System.out.println("坐如钟,站如松,卧如弓");
    }
}
```

**American. java**

```
public class American extends People {
   【代码 2】      //重写 public void speakHello()方法,输出"How do you do"
   【代码 3】      //重写 public void averageHeight()方法,输出"American's average
                  //height:176 cm"
    public void averageWeight() {
       weight = 75;
       System.out.println("American's average weight:" + weight + " kg");
    }
    public void americanBoxing() {
       System.out.println("直拳、勾拳、组合拳");
    }
}
```

**Beijingman. java**

```
public class Beijingman extends Chinese {
   【代码 4】      //重写 public void averageHeight()方法,输出"北京人的平均身高:172.5 厘米"
   【代码 5】      //重写 public void averageWeight()方法,输出"北京人的平均体重:70 千克"
  public void beijingOpera() {
     System.out.println("花脸、青衣、花旦和老生");
  }
}
```

**Example. java**

```
public class Example {
  public static void main(String args[]) {
    Chinese chinaPeople = new Chinese();
    American americanPeople = new American();
    Beijingman beijingPeople = new Beijingman();
    chinaPeople.speakHello();
    americanPeople.speakHello();
    beijingPeople.speakHello();
    chinaPeople.averageHeight();
    americanPeople.averageHeight();
    beijingPeople.averageHeight();
    chinaPeople.averageWeight();
    americanPeople.averageWeight();
```

```
        beijingPeople.averageWeight();
        chinaPeople.chinaGongfu();
        americanPeople.americanBoxing();
        beijingPeople.beijingOpera();
        beijingPeople.chinaGongfu();
    }
}
```

**❻ 实验指导**

(1) 如果子类可以继承父类的方法,子类就有权利重写这个方法,子类通过重写父类的方法可以改变方法的具体行为。

(2) 方法重写时一定要保证方法的名字、类型、参数个数和类型与父类的某个方法完全相同,只有这样子类继承的这个方法才被隐藏。

(3) 子类在重写方法时不可以将实例方法更改为类方法,也不可以将类方法更改为实例方法,即如果重写的方法是 static 方法,static 关键字必须要保留;如果重写的方法是实例方法,在重写时不可以用 static 修饰该方法。

**❼ 实验后的练习**

People 类中的

```
public void speakHello()
public void averageHeight()
public void averageWeight()
```

3 个方法的方法体中的语句是否可以省略?

**❽ 填写实验报告**

实验编号:501　学生姓名:　　实验时间:　　教师签字:

| 实验效果评价 | A | B | C | D | E |
|---|---|---|---|---|---|
| 模板完成情况 | | | | | |
| **实验后的练习效果评价** | A | B | C | D | E |
| 练习完成情况 | | | | | |
| **总评** | | | | | |

# 实验 2　银行计算利息

**❶ 相关知识点**

子类一旦隐藏了继承的成员变量,那么子类创建的对象就不再拥有该变量,该变量将归关键字 super 所拥有;同样,子类一旦重写了继承的方法,就覆盖(隐藏)了继承的方法,那么子类创建的对象就不能调用被覆盖(隐藏)的方法,该方法的调用由关键字 super 负责。因此,如果在子类中想使用被子类隐藏的成员变量或覆盖的方法就需要使用关键字 super。比如 super.x、super.play()就是访问和调用被子类隐藏的成员变量 x 和方法 play()。

**❷ 实验目的**

本实验的目的是让学生掌握重写的目的以及怎样使用 super 关键字。

**❸ 实验要求**

假设银行 Bank 已经有了按整年 year 计算利息的一般方法,其中 year 只能取正整数。比如按整年计算的方法如下:

```
double computerInterest() {
    interest = year * 0.35 * savedMoney;
    return interest;
}
```

建设银行 ConstructionBank 是 Bank 的子类，准备隐藏继承的成员变量 year，并重写计算利息的方法，即自己声明一个 double 型的 year 变量，比如，当 year 取值为 5.216 时，表示要计算 5 年零 216 天的利息，但希望首先按银行 Bank 的方法 computerInterest()计算出 5 整年的利息，然后再自己计算 216 天的利息。那么，建设银行就必须把 5.216 的整数部分赋给隐藏的 year，并让 super 调用隐藏的、按整年计算利息的方法。

要求 ConstructionBank 类和 BankOfDalian 类是 Bank 类的子类，ConstructionBank 类和 BankOfDalian 类都使用 super 调用隐藏的成员变量和方法。

ConstructionBank 类、BankOfDalian 类和 Bank 类的 UML 图如图 5.3 所示。

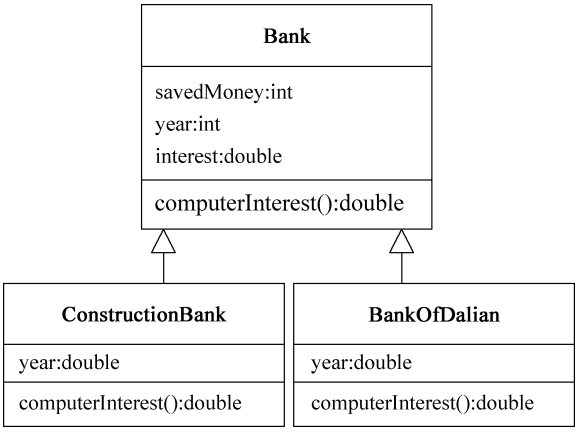


图 5.3　类的 UML 图

**❹ 程序运行效果**

程序运行效果如图 5.4 所示。

```
8000元存在建设银行8年零236天的利息:2428.800000元
8000元存在大连银行8年零236天的利息:2466.560000元
两个银行利息相差37.760000元
```

图 5.4　银行计算利息

**❺ 程序模板**

请按模板要求将【代码】替换为 Java 程序代码。

**Bank.java**

```
public class Bank {
  int savedMoney;
  int year;
  double interest;
  double interestRate = 0.29;
  public double computerInterest() {
    interest = year * interestRate * savedMoney;
    return interest;
  }
  public void setInterestRate(double rate) {
```

```
    interestRate = rate;
  }
}
```

**ConstructionBank. java**

```
public class ConstructionBank extends Bank {
  double year;
  public double computerInterest() {
    super. year = (int)year;
    double r = year - (int)year;
    int day = (int)(r * 1000);
    double yearInterest = 【代码 1】        //super 调用隐藏的 computerInterest()方法
    double dayInterest = day * 0. 0001 * savedMoney;
    interest = yearInterest + dayInterest;
    System. out. printf("%d 元存在建设银行 %d 年零 %d 天的利息: %f 元\n",
                    savedMoney, super. year, day, interest);
    return interest;
  }
}
```

**BankOfDalian. java**

```
public class BankOfDalian extends Bank {
  double year;
  public double computerInterest() {
     super. year = (int)year;
     double r = year - (int)year;
     int day = (int)(r * 1000);
     double yearInterest = 【代码 2】        //super 调用隐藏的 computerInterest()方法
     double dayInterest = day * 0. 00012 * savedMoney;
     interest = yearInterest + dayInterest;
     System. out. printf("%d 元存在大连银行 %d 年零 %d 天的利息: %f 元\n",
                         savedMoney, super. year, day, interest);
     return interest;
  }
}
```

**SaveMoney. java**

```
public class SaveMoney {
  public static void main(String args[]) {
     int amount = 8000;
     ConstructionBank bank1 = new ConstructionBank();
     bank1. savedMoney = amount;
     bank1. year = 8. 236;
     bank1. setInterestRate(0. 035);
     double interest1 = bank1. computerInterest();
     BankOfDalian bank2 = new BankOfDalian();
     bank2. savedMoney = amount;
     bank2. year = 8. 236;
     bank2. setInterestRate(0. 035);
     double interest2 = bank2. computerInterest();
     System. out. printf("两个银行利息相差 %f 元\n", interest2 - interest1);
  }
}
```

❻ 实验指导

(1) 当 super 调用被隐藏的方法时,该方法中出现的成员变量是被子类隐藏的成员变量或继承的成员变量。

(2) 子类不继承父类的构造方法,因此子类在其构造方法中需使用 super 来调用父类的构造方法,而且 super 必须是子类构造方法中的头一条语句,即如果在子类的构造方法中没有明显地写出 super 关键字来调用父类的某个构造方法,那么默认有"super();"。当类中定义了多个构造方法时,应当包括一个不带参数的构造方法,以防子类省略 super 时出现错误。

❼ 实验后的练习

参照建设银行或大连银行再编写一个商业银行,让程序输出 8000 元存在商业银行 8 年零 236 天的利息。

❽ 填写实验报告

实验编号:502　学生姓名:　　实验时间:　　教师签字:

| 实验效果评价 | A | B | C | D | E |
|---|---|---|---|---|---|
| 模板完成情况 | | | | | |
| **实验后的练习效果评价** | A | B | C | D | E |
| 练习完成情况 | | | | | |
| **总评** | | | | | |

# 实验 3　公司支出的总薪水

❶ 相关知识点

假设 B 是 A 的子类或间接子类,当用子类 B 创建一个对象,并把这个对象的引用放到 A 类声明的对象中时,比如:

```
A a;
a = new B();
```

或

```
A a;
B b = new B();
a = b;
```

那么就称对象 a 是子类对象 b 的上转型对象。上转型对象不能操作子类声明的成员变量(失掉了这部分属性),不能使用子类定义的方法(失掉了一些功能)。上转型对象可以操作子类继承的成员变量和隐藏的成员变量,也可以使用子类继承或重写的方法。上转型对象操作子类继承或重写的方法时就是通知对应的子类对象去调用这些方法。因此,如果子类重写了父类的某个方法,对象的上转型对象调用这个方法时一定是调用了这个重写的方法。上转型对象不能操作子类新增的方法和成员变量。可以将对象的上转型对象再强制转换为一个子类对象,这时该子类对象又具备了子类的所有属性和功能。

❷ 实验目的

本实验的目的是让学生掌握上转型对象的使用。在讲述继承与多态时通过子类对象的上转型体现了继承的多态性,即把子类创建的对象的引用放到一个父类的对象中,得到该对象的一个上转型对象,那么这个上转型对象在调用方法时就可能具有多种形态,不同对象的上转型

对象调用同一方法可能产生不同的行为。

**❸ 实验要求**

要求有一个 abstract 类，类名为 Employee。Employee 的子类有 YearWorker、MonthWorker、WeekWorker。YearWorker 对象按年领取薪水，MonthWorker 对象按月领取薪水，WeekWorker 对象按周领取薪水。Employee 类有一个 abstract 方法：

```
public abstract earnings();
```

子类必须重写父类的 earnings()方法，给出各自领取薪水的具体方式。

另外有一个 Company 类，该类用 Employee 对象数组作为成员，Employee 对象数组的单元可以是 YearWorker 对象的上转型对象、MonthWorker 对象的上转型对象或 WeekWorker 对象的上转型对象。程序能输出 Company 对象一年需要支付的薪水总额。

**❹ 程序运行效果**

程序运行效果如图 5.5 所示。

```
公司薪水总额:789600.0元
```

图 5.5　薪水总额

**❺ 程序模板**

请按模板要求将【代码】替换为 Java 程序代码。

**CompanySalary.java**

```
abstract class Employee {
  public abstract double earnings();
}
class YearWorker extends Employee {
  【代码 1】            //重写 earnings()方法
}
class MonthWorker extends Employee {
  【代码 2】            //重写 earnings()方法
}
class WeekWorker extends Employee {
  【代码 3】            //重写 earnings()方法
}
class Company {
  Employee[ ] employee;
  double salaries = 0;
  Company(Employee[ ] employee) {
    this.employee = employee;
  }
  public double salariesPay() {
    salaries = 0;
    【代码 4】          //计算 salaries
    return salaries;
  }
}
public class CompanySalary {
  public static void main(String args[ ]) {
    Employee [ ] employee = new Employee[29];          //公司有 29 名雇员
    for(int i = 0;i < employee.length;i++) {           //雇员简单地分成 3 类
        if(i % 3 == 0)
            employee[i] = new WeekWorker();
        else if(i % 3 == 1)
            employee[i] = new MonthWorker();
```

```
            else if(i % 3 == 2)
              employee[i] = new YearWorker();
        }
        Company company = new Company(employee);
        System.out.println("公司薪水总额:" + company.salariesPay() + "元");
      }
}
```

**❻ 实验指导**

(1) 对于【代码 2】,按一年 12 个月计算出雇员一年的年薪,比如:

```
public double earnings() {
  return 12 * 2300;
}
```

(2) 尽管 abstract 类不能创建对象,但 abstract 类声明的对象可以存放子类对象的引用,即成为子类对象的上转型对象。由于 abstract 类可以有 abstract 方法,这样就保证子类必须要重写这些 abstract 方法。由于数组 employee 的每个单元都是某个子类对象的上转型对象,实验中的【代码 4】可以通过循环语句让数组 employee 的每个单元调用 earnings()方法,并将该方法返回的值累加到 salaries,代码如下:

```
for(int i = 0;i < employee.length;i++) {
    salaries = salaries + employee[i].earnings();
}
```

**❼ 实验后的练习**

(1) 如果子类 YearWorker 不重写 earnings()方法,程序编译时将提示怎样的错误?

(2) 再增加一种雇员,并计算公司一年的总薪水。

**❽ 填写实验报告**

实验编号:503　学生姓名:　　实验时间:　　教师签字:

| 实验效果评价 | A | B | C | D | E |
|---|---|---|---|---|---|
| 模板完成情况 | | | | | |
| **实验后的练习效果评价** | A | B | C | D | E |
| 练习(1)完成情况 | | | | | |
| 练习(2)完成情况 | | | | | |
| **总评** | | | | | |

## 实验答案

实验 1:

【代码 1】

```
public void averageWeight() {
    weight = 65;
    System.out.println("中国人的平均体重:" + weight + " 千克");
}
```

【代码 2】

```
public void speakHello() {
```

```
        System.out.println("How do you do");
    }
```

【代码 3】

```
    public void averageHeight() {
        height = 176;
        System.out.println("American's average height:" + height + " cm");
    }
```

【代码 4】

```
    public void averageHeight() {
        height = 172.5;
        System.out.println("北京人的平均身高:" + height + " 厘米");
    }
```

【代码 5】

```
    public void averageWeight() {
        weight = 70;
        System.out.println("北京人的平均体重:" + weight + " 千克");
    }
```

实验 2：

【代码 1】super.computerInterest();

【代码 2】super.computerInterest();

实验 3：

【代码 1】

```
    public double earnings() {
        return 12000;
    }
```

【代码 2】

```
    public double earnings() {
        return 12 * 2300;
    }
```

【代码 3】

```
    public double earnings() {
        return 52 * 780;
    }
```

【代码 4】

```
    for(int i = 0;i < employee.length;i++) {
        salaries = salaries + employee[i].earnings();
    }
```

## 自 测 题

1. 下列叙述正确的是(　　)。

A. final 类不可以有子类