《Java 程序设计》实验报告五

学生姓名：李季鸿　　班级：2021 级计科本（3）班　　学号：20213002624

实验地点：线上线下结合　　指导教师：张春元

实验日期：2023-04-03　　共 2 学时

实验环境：Win10+JDK1.8+IntelliJ IDEA 2022.1.1

1. 实验目的

学习子类与继承，接口与实现，内部类，掌握

• 集成化开发工具工程的创建。

2. 实验内容

（1） 用集成化开发工具完成实验教材 P39 实验 1 内容。

（2） 用集成化开发工具完成实验教材 P43 实验 2 内容（请思考实验教材上这题写法是否符合面向对象编程的基本思想，如不符合，该如何重写这个题？）。

（3） 用集成化开发工具完成实验教材 P46 实验 3 内容。

3. 实验过程

报告撰写具体要求：截屏显示或直接写出实验 1 至实验 3 的源码和运行结果。

实验内容（1）：

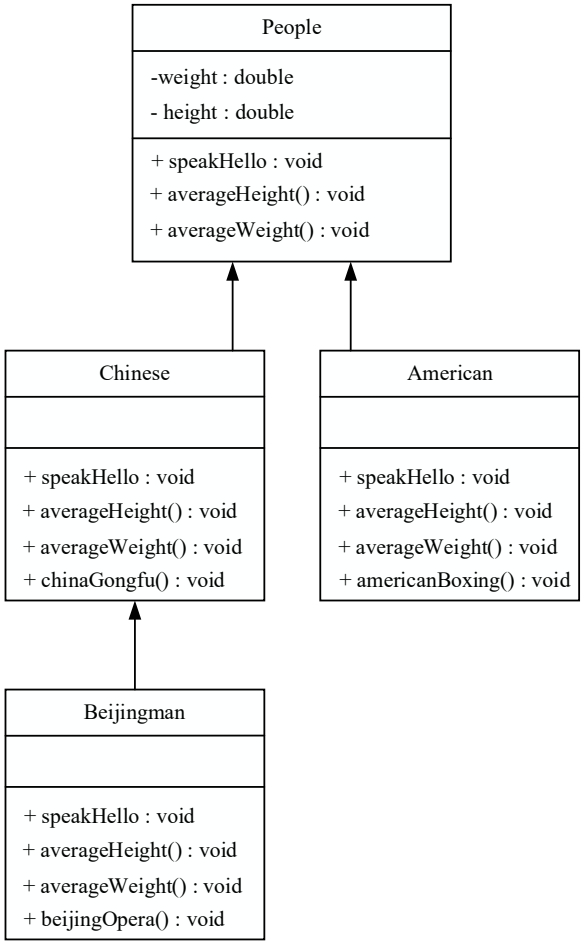


图 5.1 类的 UML 图

实验源代码：

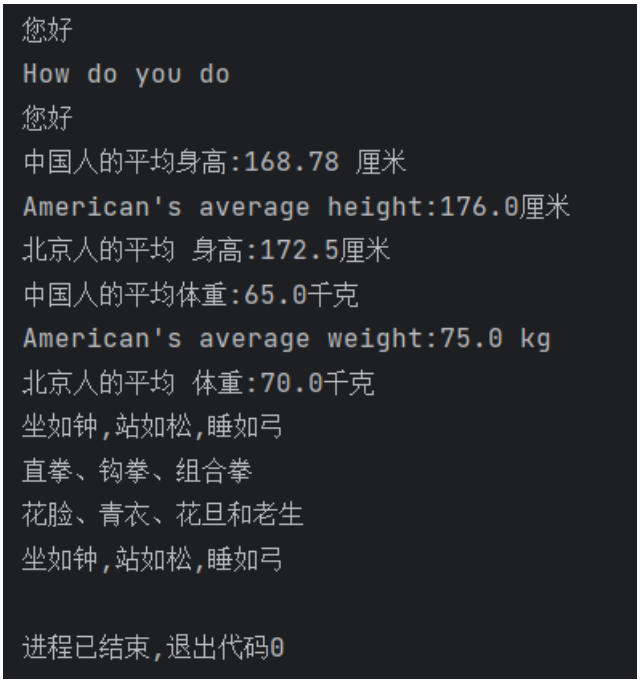
```
1.  package _5_;
2.
3.  /**
4.   * \* Created with IntelliJ IDEA.
5.   * \* User: lijihong
6.   * \* Date: 2023-04-03
7.   * \* Time: 13:29
8.   * \
9.   */
10. public class People {
11.     protected double weight,height;
12.     public void speakHello() {
13.         System.out.println("yayayaya");
14.     }
15.     public void averageHeight() {
16.         height=173;
17.         System.out.println("average height:"+height);
18.     }
19.     public void averageWeight() {
20.         weight=70;
21.         System.out.println("average weight:"+weight);
22.     }
23. }
24. package _5_;
25.
26. /**
27.  * \* Created with IntelliJ IDEA.
28.  * \* User: lijihong
29.  * \* Date: 2023-04-03
30.  * \* Time: 13:32
31.  * \
32.  */
33. public class Chinese extends People {
34.     public void speakHello() {
35.         System.out.println("您好");
36.     }
37.     public void averageHeight() {
38.         height = 168.78;
39.         System.out.println("中国人的平均身高:"+height+" 厘米");
40.     }
```

```
41.         public void averageWeight() {
42.             weight=65;
43.             System.out.println("中国人的平均体重:"+weight+"千克 ");
44.         }
45.         //重写 public void averageWeight()方法，输出:"中国人的平均体重:
65 公斤"
46.         public void chinaGongfu() {
47.             System.out.println("坐如钟,站如松,睡如弓");
48.         }
49.     }
50.     package _5_;
51.
52.     /**
53.      * \* Created with IntelliJ IDEA.
54.      * \* User: lijihong
55.      * \* Date: 2023-04-03
56.      * \* Time: 13:33
57.      * \
58.      */
59.     public class American extends People {
60.         public void speakHello() {
61.             System.out.println("How do you do");
62.         }
63.         //重写 public void speakHello()方法，输出"How do you do"
64.         public void averageHeight() {
65.             height=176;
66.             System.out.println("American's average height:"+height+"
厘米");
67.         }
68.         //重写 public void averageHeight()方法，输出
"American'saverage height:176 cm"
69.         public void averageWeight() {
70.             weight = 75;
71.             System.out.println("American's average weight:"+weight+"
kg");
72.         }
73.         public void americanBoxing() {
74.             System.out.println("直拳、钩拳、组合拳");
75.         }
76.     }
77.     package _5_;
78.
79.     /**
```

```
80.     * \* Created with IntelliJ IDEA.
81.     * \* User: lijihong
82.     * \* Date: 2023-04-03
83.     * \* Time: 13:33
84.     * \
85.     */
86.    public class Beijingman extends Chinese {
87.        public void averageHeight() {
88.            height=172.5;
89.            System.out.println("北京人的平均 身高:"+height+"厘米 ");
90.        }
91.        //重写 public void averageHeight()方法,输出:"北京人的平均身
高: 172.5 厘米"
92.        public void averageWeight() {
93.            weight=70;
94.            System.out.println("北京人的平均 体重:"+weight+"千克 ");
95.        }
96.        //重写 public void averageWeight()方法,输出:"北京人的平均体重:
70 公斤"
97.        public void beijingOpera() {
98.            System.out.println("花脸、青衣、花旦和老生");
99.        }
100.   }
101.   package _5_;
102.
103.   /**
104.    * \* Created with IntelliJ IDEA.
105.    * \* User: lijihong
106.    * \* Date: 2023-04-03
107.    * \* Time: 13:38
108.    * \
109.    */
110.   public class Example{
111.       public static void main(String args[]){
112.           Chinese chinaPeople=new Chinese();
113.           American americanPeople=new American();
114.           Beijingman beijingPeople=new Beijingman();
115.           chinaPeople.speakHello();
116.           americanPeople.speakHello();
117.           beijingPeople.speakHello();
118.           chinaPeople.averageHeight();
119.           americanPeople.averageHeight();
120.           beijingPeople.averageHeight();
```

```
121.            chinaPeople.averageWeight();
122.            americanPeople.averageWeight();
123.            beijingPeople.averageWeight();
124.            chinaPeople.chinaGongfu();
125.            americanPeople.americanBoxing();
126.            beijingPeople.beijingOpera();
127.            beijingPeople.chinaGongfu();
128.        }
129.    }
```

实验运行结果截图：



实验后的练习：

答案：可以。

方法重写，也有人叫方法覆盖，其实覆盖这个词更形象些。父类有通用方法，但在某些业务场景下这个通用方法可能不太适用所有子类。那么子类可以定义自己的方法，调用时执行自己的方法，而不使用父类的。这就是方法重写。即:子类重写了父类的方法（或者叫子类方法覆盖了父类的方法)。方法重写就是子类有一个方法，和父类的某个方法名称，返回类型，参数一样（哪怕是形参列表，也必须一样），那么我们就说子类的这个方法覆盖了父类的方法。

继承会根据查找关系就近原则，如果子类有，则执行子类的方法，不执行父类的方法（因为父类的方法被子类方法重写）。而子类方法一旦被注释掉，则会执行父类的公共方法。

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，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；重写方法既可以操作继承的，也可以操作子类新声明的成员变量和调用新定义的其他方法，但是无法操作被子类隐藏的成员变量和方法；子类在重写方法时，实例方法和类方法不能互换。

实验内容（2）：

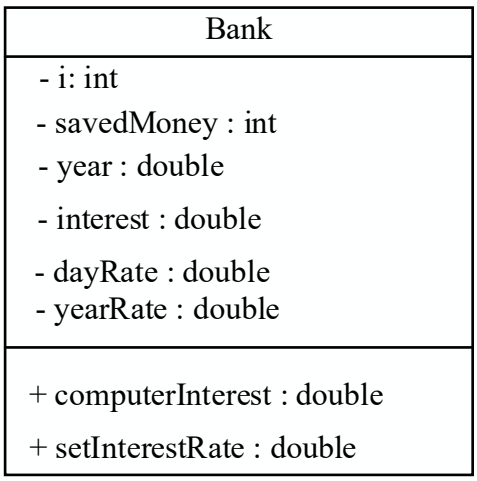


图 5.2 类 UML 图

实验源代码：

```
1. package _5_;
2.
3. /**
4.  * \* Created with IntelliJ IDEA.
5.  * \* User: lijihong
6.  * \* Date: 2023-04-03
7.  * \* Time: 14:12
8.  * \
9.  */
10.public class Bank {
11.    int i;
12.    int savedMoney;
13.    double year;
14.    double dayRate;
15.    double yearRate;
16.    double interest;
17.    public double computerInterest() {
18.        double r = year-(int)year;
19.        int day=(int)(r * 1000);
20.        double dayInterest = day * this.dayRate * savedMoney;
21.        double yearInterest = (int)year * this.yearRate * savedMo
   ney;
22.        interest = yearInterest + dayInterest;
23.        System.out.printf("%d 元存在银行%d %d 年零%d 天的利息:%f 元
   \n", savedMoney,i,(int)year,day,interest);
24.        return interest;
```

```
    }
    public Bank(double year, int i ,double dayRate, double yearRate) {
        this.i = i;
        this.year = year;
        this.dayRate = dayRate;
        this.yearRate = yearRate;
    }
}
package _5_;

/**
 * \* Created with IntelliJ IDEA.
 * \* User: lijihong
 * \* Date: 2023-04-03
 * \* Time: 14:14
 * \
 */
public class SaveMoney{
    public static void main(String arg[]){
        int amount = 8000;
        Bank bank_1 = new Bank(8.236, 1, 0.0001,  0.035);
        bank_1.savedMoney = amount;
        Bank bank_2 = new Bank(8.236, 2, 0.00012,  0.035);
        bank_2.savedMoney = amount;
        double interest1 = bank_1.computerInterest();
        double interest2 = bank_2.computerInterest();

        System.out.printf("两个银行利息相差%f 元\n",interest2 - interest1);
    }
}
```

实验运行结果：

```
8000 元存在银行1 8 年零236 天的利息:2428.800000 元
8000 元存在银行2 8 年零236 天的利息:2466.560000 元
两个银行利息相差37.760000元
```

实验后的练习：

```java
public class SaveMoney{
    新*
    public static void main(String arg[]){
        int amount = 8000;
        Bank bank_1 = new Bank( year: 8.236, i: 1, dayRate: 0.0001,  yearRate: 0.035);
        bank_1.savedMoney = amount;
        Bank bank_2 = new Bank( year: 8.236, i: 2, dayRate: 0.00012,  yearRate: 0.035);
        bank_2.savedMoney = amount;
        Bank bank_3 = new Bank( year: 8.236, i: 3, dayRate: 0.00013,  yearRate: 0.035);
        bank_3.savedMoney = amount;
        double interest1 = bank_1.computerInterest();
        double interest2 = bank_2.computerInterest();
        double interest3 = bank_3.computerInterest();

        System.out.printf("三个银行利息分别是%f元 %f元 %f元",interest1, interest2, interest3);
    }
}

```

```
8000 元存在银行1 8 年零236 天的利息:2428.800000 元
8000 元存在银行2 8 年零236 天的利息:2466.560000 元
8000 元存在银行3 8 年零236 天的利息:2485.440000 元
三个银行利息分别是2428.800000元 2466.560000元 2485.440000元
进程已结束,退出代码0
```

实验内容（3）：

实验源代码：

```java
package _5_;

/**
 * \* Created with IntelliJ IDEA.
 * \* User: lijihong
 * \* Date: 2023-04-03
 * \* Time: 16:45
 * \
 */
abstract class Employee {
    public abstract double earnings();
}
class YearWorker extends Employee {
    public double earnings() {
        return 12000;
    } //重写 earnings()方法
}
class MonthWorker extends Employee {
    public double earnings() {
```

```
            return 12*2300;
        } //重写 earnings()方法
    }
    class WeekWorker extends Employee {
        public double earnings() {
            return 52*780;
        } //重写 earnings()方法。
    }

    class Company {
        Employee[] employee;
        double salaries=0;
        Company(Employee[] employee) {
            this.employee=employee;
        }
        public double salariesPay() {
            salaries=0;
            for(int i=0;i<employee.length;i++) {
                salaries=salaries+employee[i].earnings();
            } //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++) { //雇员简单地分成三类
                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");
        }
    }
```

实验运行结果：

```
公司薪水总额:789600.0 元

进程已结束,退出代码0
```

实验后的练习：

（1）子类 YearWorker 如果不 重写 earnings（）方法，编译时则会提示 YearWorker 不是抽象的，并且未 覆盖 Employee 中的抽象方法 earnings（）。

（2）模仿上面的类，添加小时工:

```
1 个用法  新 *
class HourWorker extends Employee {
    1 个用法  新 *
    public double earnings() {
        return 3650*50;
    } //重写 earnings()方法。
}
```

```
    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++) { //雇员简单地分成三类
            if(i % 4 == 0)
                employee[i] = new WeekWorker();
            else if(i % 4 == 1)
                employee[i] = new MonthWorker();
            else if(i % 4 == 2)
                employee[i] = new YearWorker();
            else if(i % 4 == 3)
                employee[i] = new HourWorker();
        }
        Company company = new Company(employee);
        System.out.println("公司薪水总额:" + company.salariesPay() + " 元");
    }
}
```

运行结果：

```
公司薪水总额:1879180.0 元

进程已结束,退出代码0
```

4. 实验总结

写出实验中的心得体会。

1、什么是继承：

在软件开发中，通过继承机制，可以利用已有的数据类型来定义新的数据类型。所定义的新的数据类型不仅拥有新定义的成员，而且还同时拥有旧的成员。说白了，就是定义一个类，让它成为某个类(一般叫父类)的子类，那么它就会继承这个做父类的类里的部分属性和方法，因此，类的继承性使所建立的软件具有开放性、可扩充性，这是信息组织与分类的行之有效的方法，通过类的继承关系，使公共的特性能够共享，简化了对象、类的创建工作量，增加了代码的可重用性，复用性。

2、java 中继承使用关键字 extends ，语法如下：

[类修饰符] class 子类名 extends 父类名{

语句;

}

有关类的继承上需要我们注意的知识点：

1、在 java 中，java.lang.Object 类是所有 java 类的最高层父类，是唯一一个没有父类的类: 如果在类的声明中未使用 extends 关键字指明其父类，则默认父类为 Object 类。java 中的类的继承关系形成了以 Object 类为树根的树状层次结构。例：

```
public class Text{
…;
}
```

等价于

```
public class Text extends Object{
…;
}
```

2、如果两个类存在继承关系，则子类会自动继承父类的部分方法和变量，在子类中可以调用父类的方法和变量。在 java 中，只允许单继承，也就是说 一个类最多只能显示地继承于一个父类。但是一个类却可以被多个类继承，也就是说一个类可以拥有多个子类。

(1) 子类继承父类的成员变量

当子类继承了某个类之后，便可以使用父类中的成员变量，但是并不是完全继承父类的所有成员变量。具体的原则如下：

1）能够继承父类的 public 和 protected 成员变量；不能够继承父类的 private 成员变量；

2）对于子类可以继承的父类成员变量，如果在子类中出现了同名称的成员变量，则会发生隐藏现象，即子类的成员变量会屏蔽掉父类的同名成员变量。如果要在子类中访问父类中同名成员变量，需要使用 super 关键字来进行引用。

(2) 子类继承父类的方法

同样地，子类也并不是完全继承父类的所有方法。

1）能够继承父类的 public 和 protected 成员方法；不能够继承父类的 private 成员方法；

2）对于子类可以继承的父类成员方法，如果在子类中出现了同名称且同参数的成员方法(又叫子类重写父类的方法)，则称为覆盖，即子类的成员方法会覆盖掉父类的同名成员方法。如果要在子类中访问父类中同名成员方法，需要使用 super 关键字来进行引用。

注意：隐藏和覆盖是不同的。隐藏是针对成员变量和静态方法的，而覆盖是针对普通方法的。

3、子类是不能够继承父类的构造器，但是要注意的是，如果父类的构造器都是带有参数的，

则必须在子类的构造器中显示地通过 super 关键字调用父类的构造器并配以适当的参数列表。如果父类有无参构造器，则在子类的构造器中用 super 关键字调用父类构造器不是必须的，如果没有使用 super 关键字，系统会自动调用父类的无参构造器。做个例子就清楚了：

4.super 关键字

super 主要有两种用法：

1）super.成员变量/super.成员方法;

2）super(parameter1,parameter2…)

第一种用法主要用来在子类中调用父类的同名成员变量或者方法；

第二种主要用在子类的构造器中显示地调用父类的构造器，

要注意的是，如果是用在子类构造器中，则必须是子类构造器的第一个语句。