第 9 天 面向对象

今日内容介绍

- 面向对象
- 封装

第 1 章 面向对象

1.1理解什么是面向过程、面向对象

面向过程与面向对象都是我们编程中，编写程序的一种思维方式。

- 面向过程的程序设计方式，是遇到一件事时，思考“我该怎么做”，然后一步步实现的过程。

  例如：公司打扫卫生（擦玻璃、扫地、拖地、倒垃圾等），按照面向过程的程序设计方式会思考“打扫卫生我该怎么做，然后一件件的完成”，最后把公司卫生打扫干净了。

- 面向对象的程序设计方式，是遇到一件事时，思考“我该让谁来做”，然后那个“谁”就是对象，他要怎么做这件事是他自己的事，反正最后一群对象合力能把事就好就行了。

  例如，公司打扫卫生（擦玻璃、扫地、拖地、倒垃圾等），按照面向对象的程序设计方式会思考“我该让谁来做，如小明擦玻璃、让小丽扫地、让小郭拖地、让小强倒垃圾等”,这里的“小明、小丽、小郭、小强”就是对象，他们要打扫卫生，怎么打扫是他们自己的事，反正最后一群对象合力把公司卫生打扫干净了。

1.2面向对象举例

- 买电脑（组装机）

先使用面向过程说明买电脑这件事：假如我们需要买组装电脑，这时首先会在网上查询具体每一个硬件的参数和报价。然后会去电脑城进行多家询价，接着询价结束后回家根据具体的结果分析出自己比较满意的哪家报价，接着会到这家店里进行组装，组装时还需要进行现场监督，组装完成安装相应的系统，然后电脑抱回家。

分析上述整个过程大体分一下几步：上网查询参数和报价、电脑城询价、现场安装和监督、抱电脑回家。在整个过程中我们参与了每一个细节，并且会感觉相当累。

使用面向对象说明买电脑这件事：假如我们需要买组装机，这时应该找一个懂电脑硬件的人，让他帮我们查看参数和报价，并进行询价和杀价，以及现场组装监督。而我们自己并不需要亲历亲为具体怎么做，只要告诉这个人我们想要的具体需求即可。

分析上述整个过程，发现瞬间变的十分轻松，只要找到懂电脑硬件的这个人，我们的问题都可以解决。并且在这个过程中我们不用那么辛苦。

1.3面向对象思维方式的好处

通过生活中的真实场景使用面向对象分析完之后，我们开始分析面向过程和面向对象的差异做出总结：

- 面向对象思维方式是一种更符合人们思考习惯的思想
- 面向过程思维方式中更多的体现的是执行者（自己做事情），面向对象中更多的体现是指挥者（指挥对象做事情）。
- 面向对象思维方式将复杂的问题简单化。

第 2 章 类与对象

2.1对象在需求中的使用

对面向对象有了了解之后，我们来说说在具体问题中如何使用面向对象去分析问题，和如何使用面向对象。

我们把大象装冰箱为例进行分析。

在针对具体的需求，可以使用名词提炼的办法进行分析，寻找具体的对象。

需求：把大象装冰箱里

对象：大象、冰箱

分三步：

1、打开冰箱门

2、将大象装进去

3、关闭冰箱门

分析发现打开、装、关闭都是冰箱的功能。即冰箱对象具备如下功能：

冰箱打开

冰箱存储

冰箱关闭

用伪代码描述，上述需求中有两个具体的事物 大象 和 冰箱

描述大象：

```
class 大象
{
}
```

描述冰箱

```
class 冰箱
{
    void 打开(){}
    void 存储(大象){}
    void 关闭(){}
}
```

当把具体的事物描述清楚之后，需要使用这些具体的事物，Java 使用具体的事物，需要通过 new 关键字来创建这个事物的具体实例。

使用对象：

1、创建冰箱的对象

```
冰箱 bx = new 冰箱();
```

2、调用冰箱的功能

```
对象.功能() ;
bx.打开();
bx.存储(new 大象());
bx.关闭();
```

- 总结：

1、先按照名词提炼问题领域中的对象

2、对对象进行描述，其实就是在明确对象中应该具备的属性和功能

3、通过 new 的方式就可以创建该事物的具体对象

4、通过该对象调用它以后的功能。

2.2对象在代码中的体现

在分析现实生活中的事物时发现，这些事物都有其具体的特点和功能，这些特点和功能就组成了这个特殊的事物。

描述小汽车。

分析：

事物的特点（属性）：

颜色。

轮胎个数。

事物的(功能)：

运行。

发现：事物其实就是由特点（属性）和行为（功能）组成的。

可以简单理解：属性就是数值，其实就是变量；行为就是功能，就是方法。

小汽车 {

颜色；

轮胎个数；

运行() { }

}

通过计算机语言 Java 来描述这个事物。

- 定义类的格式

```
public class 类名 {
    //可编写 0 至 n 个属性
    数据类型 变量名 1；
    数据类型 变量名 2；

    //可编写 0 至 n 个方法
    修饰符 返回值类型 方法名(参数){
        执行语句;
    }
}
```

- 汽车类

```
public class Car {
    String color;
```

```
        int number;

        void run() {
            System.out.println(color + ":" + number);
        }
    }
```

通过代码的描述，知道类的真正意义就是在描述事物。属性和功能统称为事物中的成员。

事物的成员分为两种：成员属性和成员功能。

成员属性在代码中的体现就是成员变量

成员功能在代码中的体现就是成员方法

把写好的代码测试一下。需要一个可以独立运行类。

- 创建对象的格式：

```
类名 对象名 = new 类名();
```

- 测试类

```
public class CarDemo {
    public static void main(String[] args) {
        /*
         *  测试：Car 类中的 run 方法。
         */
        // 1,创建 Car 的对象。给对象起个名字。
        Car c = new Car();// c 是类类型的变量。c 指向了一个具体的 Car 类型的对象。
        // 2,通过已有的对象调用该对象的功能。格式：对象.对象成员;
        // 3,可以该对象的属性赋值。
        c.color = "red";
        c.number = 4;
        c.run();
    }
}
```

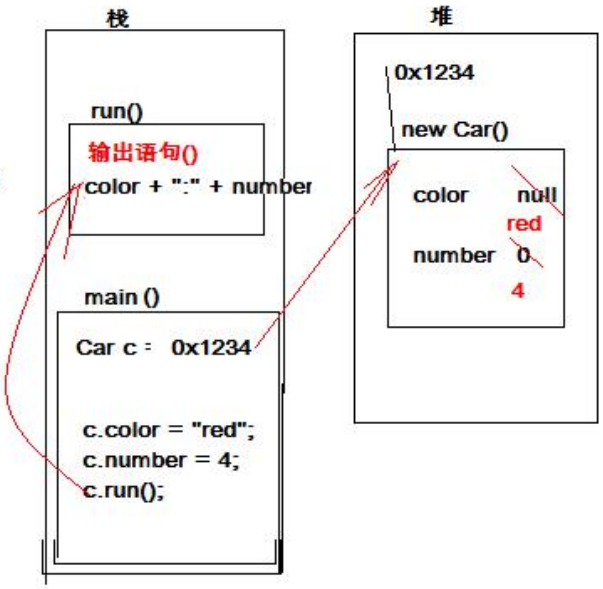
2.3对象的内存图解

经过上面对小汽车的描述，和 Java 代码测试，我们虽然可以将生活中的事物使用 Java 代码描述出来，但是这些代码在内存中是如何执行的，接下来我们需要研究下对象在内存的图解。

接下来就是分析对象在内存中的分配情况。这里需要画图一步一步演示，严格按照画图流程讲

解内存对象创建过程。

```
public class Car {
    String color;
    int number;

    void run() {
        System.out.println(color + ":" + number);
    }
}

public class CarDemo {
    public static void main(String[] args) {

        Car c = new Car();

        c.color = "red";
        c.number = 4;
        c.run();

    }
}
```



2.4类和对象的区别

面向对象的编程思想力图在程序中对事物的描述与该事物在现实中的形态保持一致。为了做到这一点，面向对象的思想中提出两个概念，即类和对象。其中，类是对某一类事物的抽象描述，而对象用于表示现实中该类事物的个体。接下来通过一个图例来抽象描述类与对象的关系，如下图所示。

图 1-1 类与对象

在上图中，可以将玩具模型看作是一个类，将一个个玩具看作对象，从玩具模型和玩具之间的

关系便可以看出类与对象之间的关系。类用于描述多个对象的共同特征，它是对象的模板。对象用于描述现实中的个体，它是类的实例。从上图中可以明显看出对象是根据类创建的，并且一个类可以对应多个对象，接下来分别讲解什么是类和对象。

经过前面几个知识点的学习，基本上掌握了类是用于描述事物的，类中可以定义事物的属性和行为。而对象是通过描述的这个类，使用 new 关键字创建出来，通过对象就可以调用该对象具体的属性和功能了。

2.5局部变量和成员变量区别

理解清楚了类和对象之后，结合前 5 天的学习知识，发现在描述类的属性和前面学习定义变量差别不大，唯一区别就是位置发生了改变，那么类中定义的变量，和在方法定义的变量有啥差别呢？

回忆以前学习时变量的定义方式，和位置，以及现在定义类中属性的特点。总结下面几点异同

区别一：定义的位置不同

定义在类中的变量是成员变量

定义在方法中或者{}语句里面的变量是局部变量

区别二：在内存中的位置不同

成员变量存储在对内存的对象中

局部变量存储在栈内存的方法中

区别三：声明周期不同

成员变量随着对象的出现而出现在堆中，随着对象的消失而从堆中消失

局部变量随着方法的运行而出现在栈中，随着方法的弹栈而消失

区别四：初始化不同

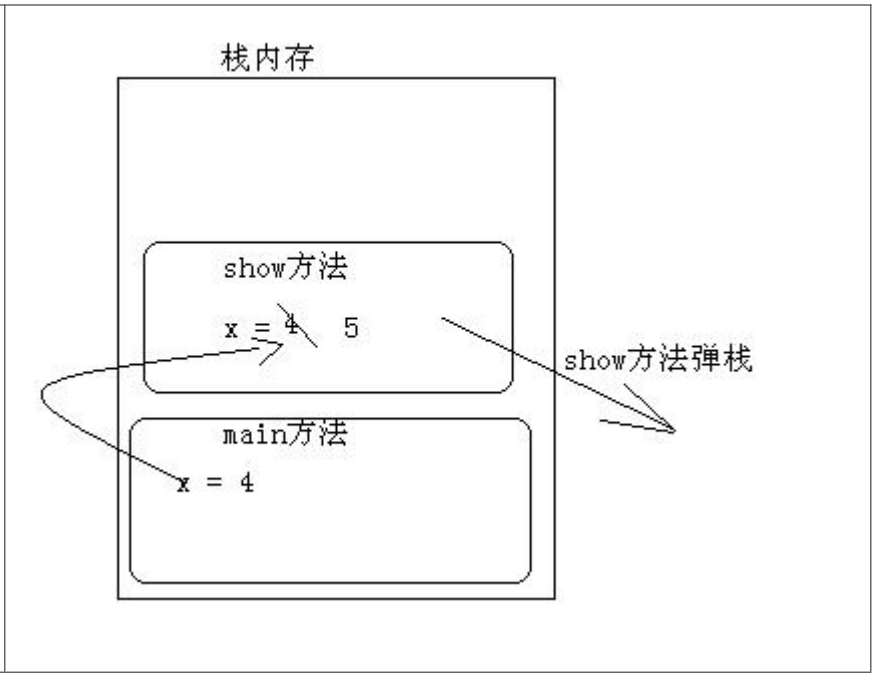
成员变量因为在堆内存中，所有默认的初始化值

局部变量没有默认的初始化值，必须手动的给其赋值才可以使用。

2.6基本类型和引用类型作为参数传递

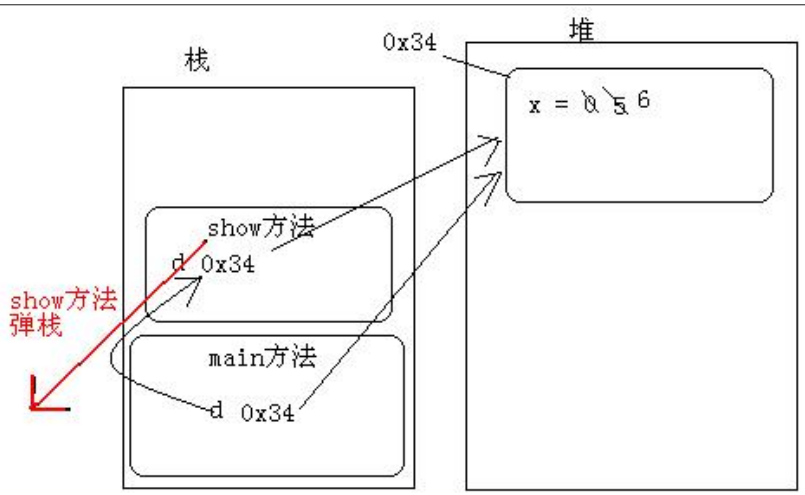
引用类型数据和基本类型数据作为参数传递有没有差别呢？我们用如下代码进行说明，并配合图解让大家更加清晰

```
class Demo
{
    public static void main(String[] args)
    {
        int x = 4;
        show(x);

    System.out.println("x="+x);
    }
    public static void show(int x)
    {
        x = 5;

    }
}
```



基本类型作为参数传递时，其实就是将基本类型变量 x 空间中的值复制了一份传递给调用的方法 show()，当在 show()方法中 x 接受到了复制的值，再在 show()方法中对 x 变量进行操作，这时只会影响到 show 中的 x。当 show 方法执行完成，弹栈后，程序又回到 main 方法执行，main 方法中的 x 值还是原来的值。

```
class Demo
{
    int x ;
    public static void main(String[] args)
    {

        Demo d = new Demo();
        d.x = 5;
        show(d);

    System.out.println("x="+d.x);
    }
    public static void show(Demo d)
    {
        d.x = 6;
    }
}
```



当引用变量作为参数传递时，这时其实是将引用变量空间中的内存地址(引用)复制了一份传递给了 show 方法的 d 引用变量。这时会有两个引用同时指向堆中的同一个对象。当执行 show 方法中的 d.x=6 时，会根据 d 所持有的引用找到堆中的对象，并将其 x 属性的值改为 6.show 方法弹栈。

由于是两个引用指向同一个对象，不管是哪一个引用改变了引用的所指向的对象的中的值，其他引用再次使用都是改变后的值。

第 3 章 封装

3.1封装概述

提起封装，大家并不陌生。前面我们学习方法时，就提起过，将具体功能封装到方法中，学习对象时，也提过将方法封装在类中，其实这些都是封装。

封装，它也是面向对象思想的特征之一。面向对象共有三个特征：封装，继承，多态。接下来我们具体学习封装。

- 封装表现：
  - 1、方法就是一个最基本封装体。
  - 2、类其实也是一个封装体。
- 从以上两点得出结论，封装的好处：
  - 1、提高了代码的复用性。
  - 2、隐藏了实现细节，还要对外提供可以访问的方式。便于调用者的使用。这是核心之一，也可以理解为就是封装的概念。
  - 3、提高了安全性。

3.2封装举例

机箱：

一台电脑，它是由 CPU、主板、显卡、内存、硬盘、电源等部件组长，其实我们将这些部件组装在一起就可以使用电脑了，但是发现这些部件都散落在外面，很容造成不安全因素，于是，使用机箱壳子，把这些部件都装在里面，并在机箱壳上留下一些插口等，若不留插口，大家想想会是什么情况。

总结：机箱其实就是隐藏了办卡设备的细节，对外提供了插口以及开关等访问内部细节的方式。

3.3私有 private

了解到封装在生活的体现之后，又要回到 Java 中，细说封装的在 Java 代码中的体现，先从描述 Person 说起。

描述人。Person

属性：年龄。

行为：说话：说出自己的年龄。

```
class Person {
    int age;
    String name;

    public void show() {
        System.out.println("age=" + age + ",name" + name);
    }
}

public class PersonDemo {
    public static void main(String[] args) {
        // 创建 Person 对象
        Person p = new Person();
        p.age = -20; // 给 Person 对象赋值
        p.name = "人妖";
        p.show(); // 调用 Person 的 show 方法
    }
}
```

通过上述代码发现，虽然我们用 Java 代码把 Person 描述清楚了，但有个严重的问题，就是 Person 中的属性的行为可以任意访问和使用。这明显不符合实际需求。

可是怎么才能不让访问呢？需要使用一个 Java 中的关键字也是一个修饰符 private(私有，权限修饰符)。只要将 Person 的属性和行为私有起来，这样就无法直接访问。

```
class Person {
    private int age;
    private String name;

    public void show() {
        System.out.println("age=" + age + ",name" + name);
    }
}
```

年龄已被私有，错误的值无法赋值，可是正确的值也赋值不了，这样还是不行，那肿么办呢？按照之前所学习的封装的原理，隐藏后，还需要提供访问方式。只要对外提供可以访问的方法，让

其他程序访问这些方法。同时在方法中可以对数据进行验证。

一般对成员属性的访问动作：赋值(设置 set)，取值(获取 get)，因此对私有的变量访问的方式可以提供对应的 setXxx 或者 getXxx 的方法。

```
class Person {
    // 私有成员变量
    private int age;
    private String name;

    // 对外提供设置成员变量的方法
    public void setAge(int a) {
        // 由于是设置成员变量的值，这里可以加入数据的验证
        if (a < 0 || a > 130) {
            System.out.println(a + "不符合年龄的数据范围");
            return;
        }
        age = a;
    }

    // 对外提供访问成员变量的方法
    public void getAge() {
        return age;
    }
}
```

- 总结：

  类中不需要对外提供的内容都私有化，包括属性和方法。

  以后再描述事物，属性都私有化，并提供 setXxx getXxx 方法对其进行访问。

- 注意：私有仅仅是封装的体现形式而已。

3.4this 关键字

3.4.1 成员变量和局部变量同名问题

当在方法中出现了局部变量和成员变量同名的时候，那么在方法中怎么区别局部变量成员变量呢？可以在成员变量名前面加上 this.来区别成员变量和局部变量

```
class Person {
    private int age;
    private String name;

    public void speak() {
        this.name = "小强";
        this.age = 18;
        System.out.println("name=" + this.name + ",age=" + this.age);
    }
}

class PersonDemo {
    public static void main(String[] args) {
        Person p = new Person();
        p.speak();
    }
}
```

3.4.2 对象的内存解释

我们已经学习了如何把生活中的事物使用 Java 代码描述 ,接下来我们分析对象在内存中的分配情况。这里需要画图一步一步演示，严格按照画图流程讲解内存对象创建使用过程。

```
class Person {
    private int age;
    public int getAge() {
        return this.age;
    }
    public void setAge(int age) {
        this.age = age;
    }
}
public class PersonDemo {
    public static void main(String[] args) {
        Person p = new Person();
        p.setAge(30);
        System.out.println("大家好，今年我" + p.getAge() + "岁");
    }
}
```

下图为程序中内存对象的创建使用过程。

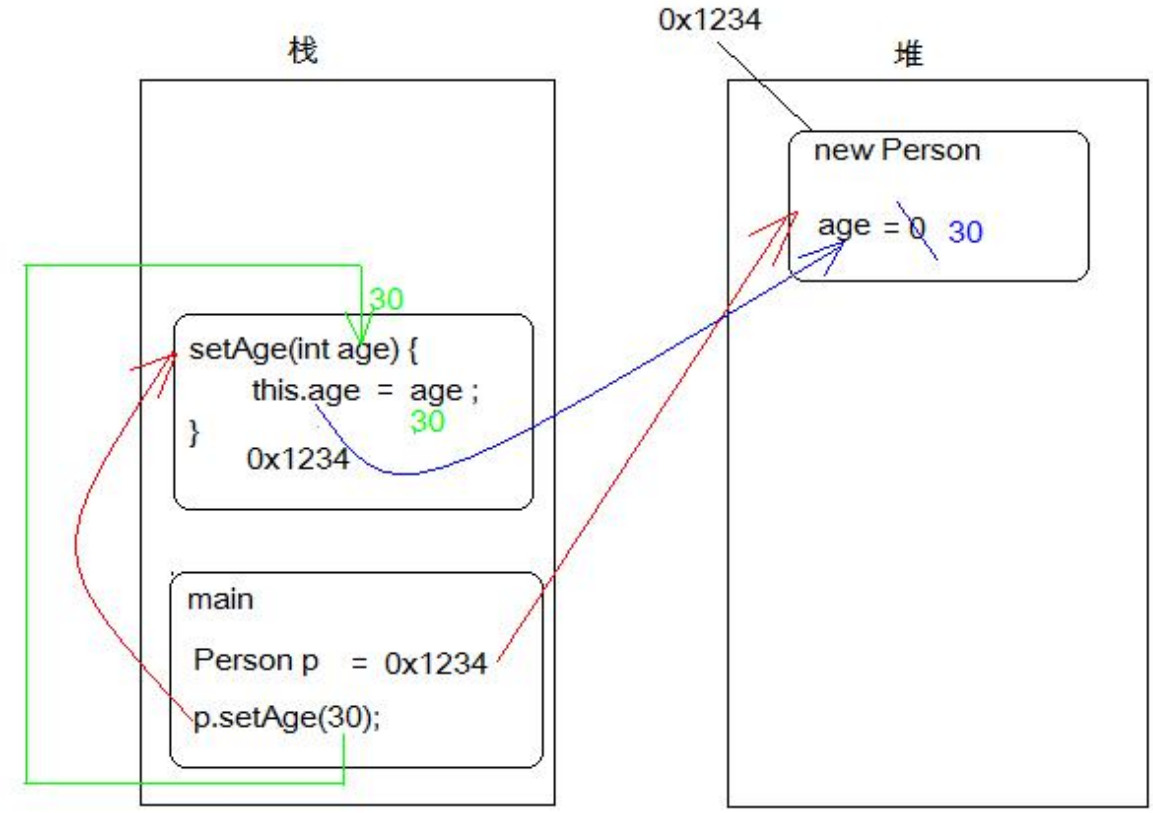


图 1-2 内存对象创建使用过程

程序执行流程说明：

1、 先执行 main 方法（压栈），执行其中的 Person p = new Person()；

2、 在堆内存中开辟空间，并为其分配内存地址 0x1234，紧接着成员变量默认初始化(age = 0)；将内存地址 0x1234 赋值给栈内中的 Person p 变量

3、 继续执行 p.setAge(30)语句，这时会调用 setAge(int age)方法，将 30 赋值为 setAge 方法中的“age”变量；执行 this.age = age 语句，将 age 变量值 30 赋值给成员变量 this.age 为 30；

4、 setAge()方法执行完毕后（弹栈），回到 main()方法，执行输出语句 System.out.println()，控制台打印 p 对象中的 age 年龄值。

- 注意：

- this 到底代表什么呢？this 代表的是对象，具体代表哪个对象呢？哪个对象调用了 this 所在的方法，this 就代表哪个对象。
- 上述代码中的 p.setAge(30)语句中，setAge(int age)方法中的 this 代表的就是 p 对象。

3.4.3 this 的应用

学习 this 的用法之后，现在做个小小的练习。

需求：在 Person 类中定义功能，判断两个人是否是同龄人

```
class Person {
    private int age;
    private String name;

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void speak() {
        System.out.println("name=" + this.name + ",age=" + this.age);
    }

    // 判断是否为同龄人
    public boolean equalsAge(Person p) {
        // 使用当前调用该 equalsAge 方法对象的 age 和传递进来 p 的 age 进行比较
        // 由于无法确定具体是哪一个对象调用 equalsAge 方法，这里就可以使用 this 来代替
        /*
         * if(this.age == p.age) { return true; } return false;
         */
```

```
            return this.age == p.age;
        }
    }
```

第 4 章 综合案例---随机点名案例重构

4.1案例介绍

随机点名器，即在全班同学中随机的找出一名同学，打印这名同学的个人信息。

此案例在我们前几天课程学习中，已经介绍，现在我们要做的是对原有的案例进行升级，使用新的方式来实现。

我们来完成随机点名器，它具备以下 3 个内容：

- 存储所有同学姓名
- 总览全班同学姓名
- 随机点名其中一人，打印到控制台

4.2案例分析

全班同学中随机的找出一名同学，打印这名同学的个人信息。

我们对本案例进行分析，得出如下分析结果：

1.存储全班同学信息（姓名、年龄）

2.打印全班同学每一个人的信息（姓名、年龄）

3.在班级总人数范围内，随机产生一个随机数，查找该随机数所对应的同学信息（姓名、年龄）并打印

随机点名器明确地分为了三个功能。如果将多个独立功能的代码写到一起，则代码相对冗长，我们可以针对不同的功能可以将其封装到一个方法中，将完整独立的功能分离出来。

而在存储同学姓名时，如果对每一个同学都定义一个变量进行姓名存储，则会出现过多孤立的变量，很难一次性将全部数据持有。此时，我们采用 ArrayList 集合来解决多个学生信息的存储问题。

4.3重构内容分析

将原来使用的简单 Student 类，封装为包装属性和方法的相对完整的 Student 类，并将所有访问属性的地方改为通过 get/set 方法访问。

重构部分已使用红色样色字体表示

4.4实现代码步骤

每名学生都拥有多项个人信息，为了方便管理每个人的信息，我们对学生信息进行封装，编写 Student.java 文件

```
/**
 * 学生信息类
 */
public class Student {
    private String name; // 姓名
    private int age; // 年龄

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
```

```
        }
    }
```

上述代码中，对学生信息（姓名、年龄）进行了封装。这样做的好处在于，以后只要找到这名学生，就能够知道他的每项个人信息了。

接下来我们编写 CallName.java 文件，完成程序的编写。

- main 方法中调用三个独立方法

```
        public static void main(String[] args) {
            ArrayList<Student> list = new ArrayList<Student>(); //1.1 创建一个可以存储多个同学名字的容器

            /*
             * 1.存储全班同学信息
             */
            addStudent(list);
            /*
             * 2.打印全班同学每一个人的信息（姓名、年龄）
             */
            printStudent(list);
            /*
             * 3.随机对学生点名，打印学生信息
             */
            randomStudent(list);
        }
```

- 存储所有学生的个人信息

```
        /**
         * 1.存储全班同学名字
         */
        public static void addStudent(ArrayList<Student> list) {
            //键盘输入多个同学名字存储到容器中
            Scanner sc = new Scanner(System.in);
            for (int i = 0; i < 3; i++) {
                //创建学生
                Student s = new Student();
                System.out.println("存储第"+i+"个学生姓名：");
                String name = sc.next();
                s.setName(name);
                System.out.println("存储第"+i+"个学生年龄：");
                int age = sc.nextInt();
```

```
            s.setAge(age);
            //添加学生到集合
            list.add(s);
        }
    }
```

上述方法中，方法参数 list 中用来表示已存储所有学生。通过 Scanner，完成新学生信息（姓名，年龄）的录入，并将学生添加到集合中。

- 打印全班同学每一个人的信息

```
    /**
     * 2.打印全班同学每一个人的信息（姓名、年龄）
     */
    public static void printStudent (ArrayList<Student> list) {
        for (int i = 0; i < list.size(); i++) {
            Student s = list.get(i);
            System.out.println("姓名："+s.getName() +",年龄："+s.getAge());
        }
    }
```

上述方法中，方法参数 list 中用来表示已存储所有学生。通过遍历集合中的每个元素，得到每个同学信息，并输出打印。

- 随机对学生点名，打印学生信息

```
    /**
     * 3.随机对学生点名，打印学生信息
     */
    public static void randomStudent (ArrayList<Student> list) {
        //在班级总人数范围内，随机产生一个随机数
        int index = new Random().nextInt(list.size());
        //在容器（ArrayList 集合）中，查找该随机数所对应的同学信息（姓名、年龄）
        Student s = list.get(index);
        System.out.println("被随机点名的同学："+s.getName() + "，年龄:" + s.getAge());
    }
```

上述方法中，通过随机数类 Random 产生一个从 0 到集合长度的随机索引。使用该索引获取 ArrayList 集合中对应的值，便得到了全班同学的随机学生信息并打印。

第 5 章 总结

5.1知识点总结

- 类与对象
  - 类，用于描述多个对象的共同特征，它是对象的模板。
  - 对象，用于描述现实中的个体，它是类的实例。
  - 类的定义：使用关键字 class 来定义 java 中的类
    - 格式：

```
class 类名 {
    //属性
    数据类型 变量名;
    …
    //方法
    修饰符 返回值类型 方法名(参数){    }
    …
}
```

  - 创建对象：
    - 格式：

```
类名 对象名 = new 类名();
```

- 封装（private 关键字）
  - 封装，把对象的属性与方法的实现细节隐藏，仅对外提供一些公共的访问方式
  - 封装的体现：
    - 变量:使用 private 修饰，这就是变量的封装
    - 方法:也是一种封装，封装了多条代码
    - 类： 也是一种封装，封装了多个方法

- private 关键字，私有的意思
  - 它可以用来修饰类中的成员(成员变量，成员方法)
  - private 的特点：
    - private 修饰的成员只能在当前类中访问，其他类中无法直接访问

- this 关键字
  - this 关键字，本类对象的引用
    - this 是在方法中使用的，哪个对象调用了该方法，那么，this 就代表调用该方法的对象引用
    - this 什么时候存在的？当创建对象的时候，this 存在的
    - this 的作用：用来区别同名的成员变量与局部变量（this.成员变量）

```
public void setName(String name) {
    this.name = name;
}
```