面向对象基础

今日内容

- 面向对象思想
- 面向对象特性之封装
- 面向对象之构造方法
- 类名作为形参和返回值案例

第1章 面向对象概述

1.1 面向对象思想

1.1.1 面向过程思想与面向对象思想

A:什么是面向过程

面向过程，其实就是面向着具体的每一个步骤和过程，把每一个步骤和过程完成，然后由这些功能方法相互调用，完成需求。

例如:吃煎饼果子利用面向过程的思想:

1.学习摊煎饼的技术

2.买材料鸡蛋,油,葱等等

3.开始摊

4.吃

5.收拾

B:什么是面向对象

面向对象思想就是不断的创建对象，使用对象，指挥对象做事情。（如果有对象，直接用对象，对我们直接提供服务）

例如:吃煎饼果子利用面向对象的思想

1. 找会摊煎饼的大妈(创建一个摊煎饼大妈的对象)
2. 调用其摊煎饼的技能(功能),传递进去钱参数
3. 返回给我们一个煎饼
4. 吃

1.1.2 面向过程特点与面向对象特点

A:面向过程：

强调的是过程，所有事情都需要自己完成

B:面向对象:

是一种更符合我们思想习惯的思想(懒人思想，我把事情自己不做，交给别人去做)

可以将复杂的事情简单化（对使用者来说简单了，对象里面还是很复杂的）

将我们从执行者变成了指挥者角色发生了转换

1.1.2.1 案例代码一:

```
package com.easthome_01;
/*
 * 面向对象思想：
 *        面向对象是基于面向过程的编程思想。
 *
```

```
     *        面向过程：强调的是每一个功能的步骤
     *        面向对象：强调的是对象，然后由对象去调用功能
     *
     * 面向对象的思想特点：
     *        A:是一种更符合我们思考习惯的思想
     *        B:可以将复杂的事情简单化
     *        C:将我们从执行者变成了指挥者
     *
     * 举例：
     *        买电脑：
     *            面向过程：我要买电脑--我要明确买电脑的意义--上网查对应的参数信息--去中关村买电脑--讨价还价--买回电脑
     *            面向对象：我要买电脑--班长去给我买电脑--买回电脑
     *        洗衣服：
     *            面向过程：把衣服脱下来--找一个盆--放点洗衣粉--加点水--浸泡 10 分钟--揉一揉--清洗衣服--拧干--晾起来
     *            面向对象：把衣服脱下来--打开全自动洗衣机--扔衣服--按钮--晾起来
     */
    public class Demo {

    }
```

1.2 类与对象及其使用:

1.2.1 类与对象概述:

A:我们学习编程是为了什么

为了把我们日常生活中实物用学习语言描述出来

B:我们如何描述现实世界事物

属性 就是该事物的描述信息(事物身上的名词)

行为 就是该事物能够做什么(事物身上的动词)

C:Java 中最基本的单位是类,Java 中用 class 描述事物也是如此

成员变量 就是事物的属性

成员方法 就是事物的行为

D:定义类其实就是定义类的成员(成员变量和成员方法)

a:成员变量　和以前定义变量是一样的，只不过位置发生了改变。在类中，方法外。

b:成员方法　和以前定义方法是一样的，只不过把 static 去掉，后面在详细讲解 static 的作用。

E:类和对象的概念

a:类：是一组相关的属性和行为的集合（我们班所有的同学都具备相同的属性和行为，比如：姓名，年龄，学习，这样就把所有的学生成为学生类）

b:对象：是该类事物的具体体现（说某个同学时，他都具备自己特有的属性和行为）

c:举例：

类　学生

对象　具体的某个学生就是一个对象

比如：车是一个类，具体的开的奔驰、宝马，就是对象

1.2.1.1 案例代码二:

```
package com.easthome_01;
/*
 * 我们学习编程语言，其实就是为了把现实世界的事物模拟出来，实现信息化。
 *
 * 我们是如何表示现实世界的事物的呢？
 *       A:属性　就是事物的描述信息
 *       B:行为　就是事物能够做什么
 *       举例：学生
 *
 * Java 语言最基本的单位是类，所以，我们在后面的学习过程中，是通过类来体现现实世界事物的。
 *
```

```
 * 类：是一组相关的属性和行为的集合
 * 对象：就是该事物的具体体现
 *       举例：
 *           类        学生
 *           对象      班长
 */
public class Demo2 {


}
```

1.2.2 类与对象案例:

1.2.2.1 学生类案例

1.2.2.1.1 案例代码三:

```
package com.easthome_02;
/*
 * 类的定义：
 *       类是用来描述现实世界的事物的
 *
 * 事物：
 *       属性 事物的描述信息
 *       行为 事物能够做什么
 *
 * 类是如何和事物进行对应的呢？
 *       类：
 *           成员变量
 *           成员方法
 *
 * 需求：写一个学生类
 *
 * 学生事物：
 *       属性：姓名，年龄...
 *       行为：学习，吃饭...
 *
 * 学生类：
 *       成员变量：姓名,年龄
 *       成员方法：学习,吃饭
 *
 * 成员变量：和我们前面学习过的变量的定义是一样的。
 *       位置不同：类中，方法外
```

```
 *       初始化值：不需要给初始化值
 * 成员方法：和我们前面学习过的方法的定义是一样的。
 *       去掉 static 关键字
 */
public class Student {
    //成员变量
    //姓名
    String name;
    //年龄
    int age;

    //成员方法
    //学习的方法
    public void study() {
        System.out.println("好好学习，天天向上");
    }

    //吃饭的方法
    public void eat() {
        System.out.println("学习饿了要吃饭");
    }
}
```

```
package com.easthome_02;
/*
 * Student 是一个学生事物描述类，main 方法不适合放在它里面。
 *
 * 使用一个类，其实就是使用该类的成员。(成员变量和成员方法)
 * 而我们要想使用一个类的成员，就必须首先拥有该类的对象。
 * 我们如何拥有一个类的对象呢？
 *       创建对象就可以了？
 * 我们如何创建对象呢？
 *       格式：类名 对象名 = new 类名();
 * 对象如何访问成员呢？
 *       成员变量：对象名.变量名
 *       成员方法：对象名.方法名(...)
 */
public class StudentDemo {
    public static void main(String[] args) {
        //格式：类名 对象名 = new 类名();
        Student s = new Student();
        //System.out.println("s:"+s); //com.easthome_02.Student@193c0cf
```

```
        //直接输出成员变量值
        System.out.println("姓名："+s.name); //null
        System.out.println("年龄："+s.age); //0
        System.out.println("----------");

        //给成员变量赋值
        s.name = "林青霞";
        s.age = 28;

        //再次输出成员变量的值
        System.out.println("姓名："+s.name); //林青霞
        System.out.println("年龄："+s.age); //28
        System.out.println("----------");

        //调用成员方法
        s.study();
        s.eat();
    }
}
```

1.2.2.2 手机类案例

1.2.2.2.1 案例代码四:

```
package com.easthome_02;
/*
 * 手机类：
 *      成员变量：品牌，价格，颜色...
 *      成员方法：打电话，发短信...
 */
public class Phone {
    //品牌
    String brand;
    //价格
    int price;
    //颜色
    String color;

    //打电话
    public void call(String name) {
        System.out.println("给"+name+"打电话");
    }
```

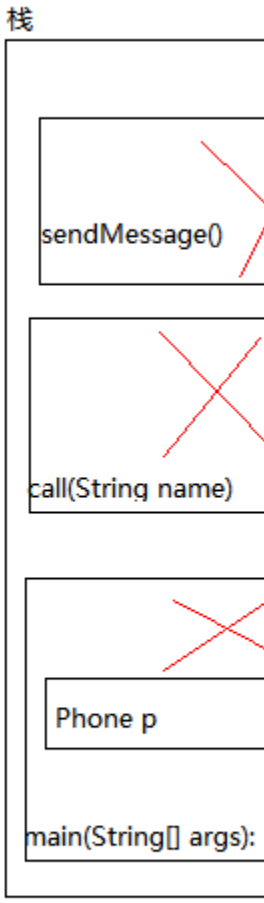
```
    //发短信
    public void sendMessage() {
        System.out.println("群发短信");
    }
}
```

```
package com.easthome_02;
/*
 * 手机类的测试类
 */
public class PhoneDemo {
    public static void main(String[] args) {
        //创建对象
        Phone p = new Phone();

        //输出成员变量值
        System.out.println("品牌："+p.brand);//null
        System.out.println("价格："+p.price);//0
        System.out.println("颜色："+p.color);//null
        System.out.println("------------");

        //给成员变量赋值
        p.brand = "锤子";
        p.price = 2999;
        p.color = "棕色";

        //再次输出成员变量值
        System.out.println("品牌："+p.brand);//锤子
        System.out.println("价格："+p.price);//2999
        System.out.println("颜色："+p.color);//棕色
        System.out.println("------------");

        //调用成员方法
        p.call("林青霞");
        p.sendMessage();
    }
}
```

1.3 对象的内存图

1.3.1 一个对象的内存图:

```
public class Phone {
        String brand;
        int price;
        String color;

        public void call(String name) {
                System.out.println("给"+name+"打电话");
        }

        public void sendMessage() {
                System.out.println("群发短信");
        }
}

public class PhoneDemo {
        public static void main(String[] args) {
                Phone p = new Phone();

                System.out.println(p.brand + "---" + p.price + "---" + p.color);
                                                null            0               null
                p.brand = "锤子";
                p.price = 2999;
                p.color = "棕色";

                System.out.println(p.brand + "---" + p.price + "---" + p.color);
                                                "锤子"          2999            "棕色"
                p.call("林青霞");
                p.sendMessage();
        }
}
```

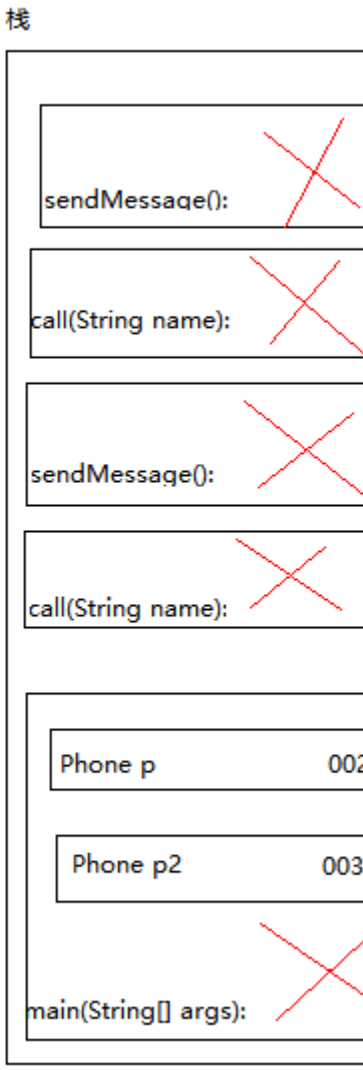


1.3.2 方法公用内存图:

1.3.2.1 案例代码五:

'

```
public class Phone {
        String brand;
        int price;
        String color;

        public void call(String name) {
                System.out.println("给"+name+"打电话");
        }

        public void sendMessage() {
                System.out.println("群发短信");
        }
}

public class PhoneDemo2 {
        public static void main(String[] args) {
                Phone p = new Phone();
                p.brand = "小米5s";
                p.price = 1999;
                p.color = "银色";
                System.out.println(p.brand+"---"+p.price+"---"+p.color);
                p.call("林青霞");          "小米5s"      1999      "银色";
                p.sendMessage();

                Phone p2 = new Phone();
                p2.brand = "IPhone7S";
                p2.price = 7999;
                p2.color = "土豪金";
                System.out.println(p2.brand+"---"+p2.price+"---"+p2.color);
                p2.call("张曼玉");       "IPhone7S"     7999;        "土豪金"
                p2.sendMessage();
        }
}
```

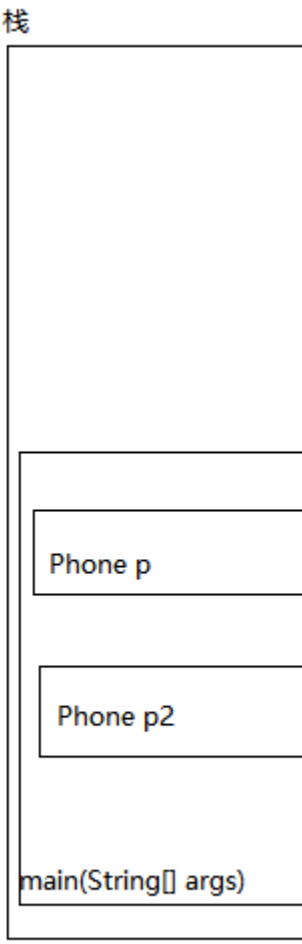



```java
package com.easthome_03;
/*
 * 手机类的测试类
 */
public class PhoneDemo2 {
    public static void main(String[] args) {
        Phone p = new Phone();
        p.brand = "小米 5s";
        p.price = 1999;
        p.color = "银色";
        System.out.println(p.brand+"---"+p.price+"---"+p.color);
        p.call("林青霞");
        p.sendMessage();
```

```
            Phone p2 = new Phone();
            p2.brand = "IPhone7S";
            p2.price = 7999;
            p2.color = "土豪金";
            System.out.println(p2.brand+"---"+p2.price+"---"+p2.color);
            p2.call("张曼玉");
            p2.sendMessage();
        }
    }
```

1.3.3 两个引用指向同一个对象内存图:

```
public class Phone {
        String brand;
        int price;
        String color;

        public void call(String name) {
                System.out.println("给"+name+"打电话");
        }

        public void sendMessage() {
                System.out.println("群发短信");
        }
}
public class PhoneDemo3 {
        public static void main(String[] args) {
                Phone p = new Phone();
                p.brand = "OPPO";
                p.price = 2999;
                p.color = "白色";
                System.out.println(p.brand+"---"+p.price+"---"+p.color);
                                        "OPPO"      2999      "白色"
                Phone p2 = p;
                p2.brand = "魅族";
                p2.price = 1999;
                p2.color = "蓝色";
                System.out.println(p.brand+"---"+p.price+"---"+p.color);
                System.out.println(p2.brand+"---"+p2.price+"---"+p2.color);
        }
                                        "魅族"      1999      "蓝色"
}
                                        "魅族"      1999      "蓝色"
```



1.3.3.1 案例代码六:

```
package com.easthome_03;
/*
 * 手机类的测试类
 */
public class PhoneDemo3 {
    public static void main(String[] args) {
        Phone p = new Phone();
        p.brand = "OPPO";
        p.price = 2999;
        p.color = "白色";
        System.out.println(p.brand+"---"+p.price+"---"+p.color);

        Phone p2 = p;
        p2.brand = "魅族";
        p2.price = 1999;
        p2.color = "蓝色";
        System.out.println(p.brand+"---"+p.price+"---"+p.color);
        System.out.println(p2.brand+"---"+p2.price+"---"+p2.color);
    }
}
```

1.4 成员变量和局部变量区别:

1.4.1 案例代码七:

```
package com.easthome_04;
/*
 * 成员变量和局部变量的区别:
 *      A:在类中的位置不同
 *          成员变量：类中，方法外
 *          局部变量：方法中或者方法声明上(形式参数)
 *      B:在内存中的位置不同
 *          成员变量：堆内存
 *          局部变量：栈内存
 *      C:生命周期不同
 *          成员变量：随着对象的创建而存在，随着对象的消失而消失
 *          局部变量：随着方法的调用而存在，随着方法的调用完毕而消失
 *      D:初始化值的问题
 *          成员变量：有默认值
 *          局部变量：没有默认值。必须先定义，赋值，最后使用
```

```
 */
public class Variable {
    int x;
    public void show() {
        int y = 0;

        System.out.println(x);
        System.out.println(y);
    }
}
```

第2章 面向对象特性之封装

2.1 封装与私有关键字

2.1.1 private 关键字

A:private 关键字：

a:是一个权限修饰符。

b:可以修饰成员(成员变量和成员方法)

c:被 private 修饰的成员只在本类中才能访问。

2.1.1.1 案例代码八:

```
package com.easthome_05;
/*
 * 学生类
 *
 * 通过对象直接访问成员变量，会存在数据安全问题
 * 这个时候，我们就想能不能不让外界的对象直接访问成员变量呢？
 * 能。
 * 如何实现呢？
 *      private 关键字
 *
 * private:
```

```
     *       是一个修饰符
     *       可以修饰成员变量，也可以修饰成员方法
     *       被 private 修饰的成员只能在本类中被访问
     *
     * 针对 private 修饰的成员变量，我们会相应的提供 getXxx()和 setXxx()用于获取和设置成员变量的值,方法用 public 修饰
     */
    public class Student {
        String name;
        //int age;
        private int age;

        public void setAge(int a) {
            if(a<0 || a>200) {
                System.out.println("你给的年龄有误");
            }else {
                age = a;
            }
        }

        public int getAge() {
            return age;
        }

        public void show() {
            System.out.println("姓名是："+name+",年龄是："+age);
        }
    }
```

```
    package com.easthome_05;
    /*
     * 学生类的测试类
     */
    public class StudentDemo {&
        public static void main(String[] args) {
            //创建学生对象
            Student s = new Student();
            s.show();

            s.name = "林青霞";
            //s.age = 28;
            //s.age = -28;
            //s.setAge(-28);
```

```
            s.setAge(28);
            s.show();
        }
    }
```

2.1.2 private 最常见应用

A:把成员变量用 private 修饰

B:提供对应的 getXxx()/setXxx()方法

2.1.2.1 案例代码九:

```
package com.easthome_06;
/*
 * 学生类
 */
public class Student {
    private String name;
    private int age;

    public void setName(String n) {
        name = n;
    }

    public String getName() {
        return name;
    }

    public void setAge(int a) {
        age = a;
    }

    public int getAge() {
        return age;
    }
}
```

```
package com.easthome_06;
/*
```

```
 * 学生类的测试类
 */
public class StudentDemo {
    public static void main(String[] args) {
        //创建对象
        Student s = new Student();
        System.out.println(s.getName()+"---"+s.getAge());

        s.setName("林青霞");
        s.setAge(28);
        System.out.println(s.getName()+"---"+s.getAge());
    }
}
```

2.1.3 封装的概述和好处:

A:封装概述

是面向对象三大特征之一

是面向对象编程语言对客观世界的模拟，客观世界里成员变量都是隐藏在对象内部的，外界无法直接操作和修改。就像刚才说的年龄。

B:封装原则：

将不需要对外提供的内容都隐藏起来。

把属性隐藏，提供公共方法对其访问。

成员变量 private，提供对应的 getXxx()/setXxx()方法

C:好处：

通过方法来控制成员变量的操作，提高了代码的安全性

把代码用方法进行封装，提高了代码的复用性

2.2 this 关键字

2.2.1 this 关键字由来和使用:

A:this:代表所在类的对象引用

方法被哪个对象调用，this 就代表那个对象

B:什么时候使用 this 呢

局部变量和成员变量重名

2.2.1.1 案例代码十:

```
package com.easthome_07;
/*
 * 学生类
 *
 * 起名字我们要求做到见名知意。
 * 而我们现在的代码中的 n 和 a 就没有做到见名知意，所以我要改进。
 *
 * 如果有局部变量名和成员变量名相同，在局部使用的时候，采用的是就近的原则。
 *
 * 我们有没有办法把局部变量的 name 赋值给成员变量的 name 呢?
 * 有。
 *
 * 什么办法呢?
 *      用 this 关键字就可以解决这个问题
 *
 * this:代表所在类的对象引用
 *      方法被哪个对象调用，this 就代表那个对象
 *
 * 使用场景:
 *      局部变量隐藏成员变量
 */
public class Student {
    private String name;
    private int age;

    public void setName(String name) { //"林青霞"
```

```
            //name = name;
            this.name = name;
        }

        public String getName() {
            return name;
        }

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

        public int getAge() {
            return age;
        }
    }
```

```
    package com.easthome_07;
    /*
     * 学生类的测试类
     */
    public class StudentDemo {
        public static void main(String[] args) {
            //创建对象
            Student s = new Student();
            System.out.println(s.getName()+"---"+s.getAge());

            s.setName("林青霞");
            s.setAge(28);
            System.out.println(s.getName()+"---"+s.getAge());
        }
    }
```

第3章 面向对象之构造方法

3.1 构造方法:

主要用来给对象的数据进行初始化

3.1.1 构造方法格式:

A:构造方法格式

a:方法名与类名相同

b:没有返回值类型，连 void 都没有

c:没有具体的返回值

3.1.1.1 案例代码十一:

```
package com.easthome_08;
/*
 * 构造方法：
 *       给对象的数据进行初始化
 *
 * 格式：
 *       方法名和类名相同
 *       没有返回值类型，连 void 都不能写
 *       没有具体的返回值
 *
 */
public class Student {
    public Student() {
        System.out.println("这是构造方法");
    }
}
```

```
package com.easthome_08;

public class StudentDemo {
    public static void main(String[] args) {
        //如何调用构造方法呢？
        //通过 new 关键字调用
        //格式：类名 对象名 = new 构造方法(...);
        Student s = new Student();
    }
}
```

3.1.2 构造方法注意事项与重载

如果你不提供构造方法，系统会给出默认构造方法

如果你提供了构造方法，系统将不再提供

构造方法也是可以重载的,重载条件和普通方法相同

3.1.2.1 案例代码十二:

```
package com.easthome_08;
/*
 * 构造方法：
 *       给对象的数据进行初始化
 *
 * 格式：
 *       方法名和类名相同
 *       没有返回值类型，连 void 都不能写
 *       没有具体的返回值
 *
 * 构造方法的注意事项：
 *       A:如果我们没有给出构造方法，系统将会提供一个默认的无参构造方法供我们使用。
 *       B:如果我们给出了构造方法，系统将不在提供默认的无参构造方法供我们使用。
 *        这个时候，如果我们想使用无参构造方法，就必须自己提供。
 *        推荐：自己给无参构造方法
 *      C:构造方法也是可以重载的
 *
 * 成员变量赋值：
 *       A:setXxx()方法
 *       B:带参构造方法
 */
public class Student {
    private String name;
    private int age;

    /*
    public Student() {
        System.out.println("这是构造方法");
    }
    */

    public Student() {}
```

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

    public Student(int age) {
        this.age = age;
    }

    public Student(String name,int age) {
        this.name = name;
        this.age = age;
    }

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

```
package com.easthome_08;

public class StudentDemo {
    public static void main(String[] args) {
        //如何调用构造方法呢?
        //通过 new 关键字调用
        //格式: 类名 对象名 = new 构造方法(...);
        Student s = new Student();
        s.show();

        //public Student(String name)
        Student s2 = new Student("林青霞");
        s2.show();

        //public Student(int age)
        Student s3 = new Student(28);
        s3.show();

        //public Student(String name,int age)
        Student s4 = new Student("林青霞",28);
        s4.show();
    }
}
```

3.1.3 包含 private,无参,有参构造的标准学生类代码:

A:类:

a:成员变量

b:构造方法

无参构造方法

带参构造方法

c:成员方法

getXxx()

setXxx()

B:给成员变量赋值的方式

a:无参构造方法+setXxx()

b:带参构造方法

3.1.3.1 案例代码十三:

```
package com.easthome_09;
/*
 * 学生类
 */
public class Student {
    //成员变量
    private String name;
    private int age;

    //构造方法
    public Student() {}

    public Student(String name,int age) {
        this.name = name;
        this.age = age;
    }
```

```
    //成员方法
    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

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

    public int getAge() {
        return age;
    }
}
```

```
package com.easthome_09;
/*
 * 学生类的测试类
 */
public class StudentDemo {
    public static void main(String[] args) {
        //无参+setXxx()
        Student s = new  Student();
        s.setName("林青霞");
        s.setAge(28);
        System.out.println(s.getName()+"---"+s.getAge());

        //带参构造
        Student s2 = new Student("林青霞",28);
        System.out.println(s2.getName()+"---"+s2.getAge());
    }
}
```

第4章 类名作为形参和返回值

4.1 类名作为方法的形式参数

4.1.1 案例代码十四:

```
package com.easthome_10;

public class Student {
    public void study() {
        System.out.println("好好学习,天天向上");
    }
}
```

```
package com.easthome_10;

public class Teacher {
    public void test(Student s) {//接收传递过来的 Student 对象的地址值
        s.study();
    }
}
```

```
package com.easthome_10;

//需求： 调用 Teacher 的 test 方法

//类名作为形式参数：其实这里需要的是该类对象。
public class Test {
    public static void main(String[] args) {
        Teacher t = new Teacher();
        Student s = new Student();
        t.test(s);
    }
}
```

4.2 类名作为返回值案例

4.2.1 案例代码十五:

```
package com.easthome_11;

public class Student {
    public void study() {
        System.out.println("好好学习,天天向上");
    }
}
```

```
package com.easthome_11;

public class Teacher {

    public Student getStudent() {
        Student s = new Student();
        return s;//返回的是 Student 对象的地址值
    }
}
```

```
package com.easthome_11;

//需求： 通过 Teacher 得到 Student 对象，然后调用 Student 类的方法
//如果方法的返回值是类名：其实返回的是该类的对象
public class Test {
    public static void main(String[] args) {
        Teacher t = new Teacher();
        Student s = t.getStudent();
        s.study();
    }
}
```