Java编程思想_面向对象之封装

课程概要

学习目标

理解面向对象思想和面向过程思想做事方式的不同

能够举例说明类和对象的区别和联系

能够将现实事物定义成类并正确使用

理解封装的概念，能够说出封装的好处

理解private关键字的作用

理解this关键字的作用

能够定义标准JavaBean类并使用

面向对象思想概述

什么是面向对象?

面向:

基本义：面对、朝向 引申义：注重、关注

比如，面向农村，面向未来，指的是关注农村、关注未来

对象：

世间一切的事物

面向对象思想：

面向对象思想就是把关注点放在一件事或一个活动中涉及到的人或事物（也就是对象）上的思想（或思维方式）

面向过程思想：

面向过程思想就是把关注点放在一件事或一个活动中涉及到的步骤（也就是过程）上的思想（或思维方式）

面向对象和面向过程的区别主要在于做事方式的不同，一个关注事物（对象），另一个过程（步骤）

面向过程举例

面向过程关键字：步骤、过程

洗衣服： 面向过程的做法：打水 -> 放衣服 -> 放洗衣粉 -> 揉搓 -> 晾晒

面向对象举例

面向对象关键字：事物、对象

洗衣服：

面向对象的做法：1.用洗衣机洗；2.让洗衣店洗；

面向对象思想特点

- 是一种更符合人们思考习惯的思想
- 把复杂的事情简单化
- 把人们从执行者变成了指挥者

面向对象的程序开发

就是不断的找对象、使用对象、指挥对象做事情的过程 没有对象？创建一个！

面向对象思想特征

1. 封装（encapsulation）
2. 继承（inheritance）
3. 多态（polymorphism）

类与对象

现实中如何描述一个事物?

这是一部iPhone X手机，它能打电话，发短信，玩游戏... **iPhone**：品牌 **X**： 型号 **手机**： 名称 **打电话**： 功能 **发短信**： 功能 **玩游戏**： 功能

描述事物从两方面着手

属性：如品牌、型号、名称等事物的状态（外在特征） **行为**：如打电话、发短信、玩游戏等事物的功能

面向对象的编程语言与现实生活是紧密联系在一起的， Java是一门面向对象的编程语言，所以，学习Java就要学会如何用编程语言描述现实事物

Java中如何描述一个事物?

Java中通过“类”来描述事物，类主要由属性和行为构成。

类的概念

即归类，分类，是一系列具有相同属性和行为的事物的统称

属性：品牌、型号、名称...

行为：打电话、发短信、玩游戏...

抽象

把一系列相关事物共同的属性和行为提取出来的过程

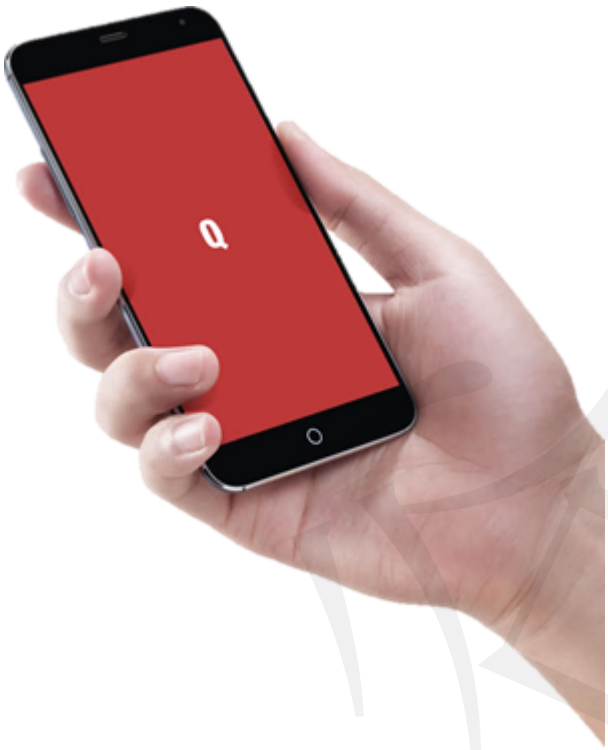
抽象是面向对象思想的第四大特征 “手机”是指一类事物的统称，是一个抽象概念，并不是某个具体事物

什么是对象?

某一类事物的某个具体存在

类和对象的关系

类：属性和行为的集合，是一个抽象概念 **对象**：是该类事物的具体体现，是一种具体存在

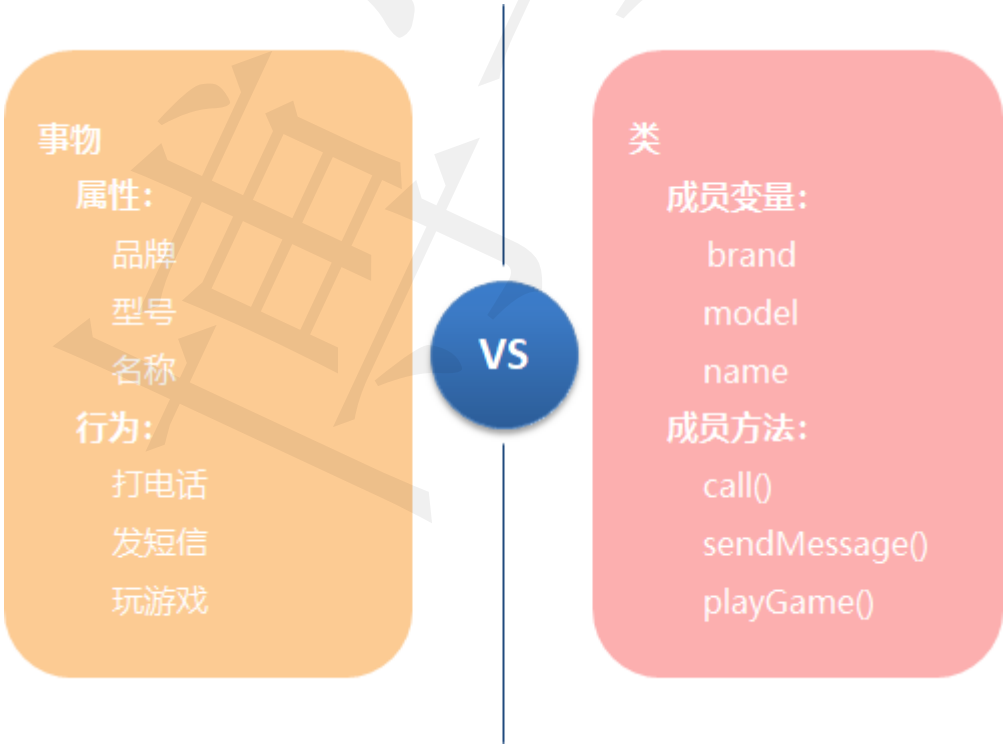


举例:

手机 -> 类

手中的这部魅族手机 -> 对象

类的定义和使用



如何定义一个类?

定义类的过程，就是把一系列相关事物共同的属性和行为抽取出来的过程 事物的属性，在类中叫**成员变量** 事物的行为，在类中叫**成员方法**

如何创建一个对象?

```
类名  对象名 = new  类名();
```

如何使用一个对象?

```
对象名.变量名
对象名.方法名(…)
```

手机类的定义和使用

定义

类名：Phone

成员变量：

```
    String brand; // 品牌
    String model; // 型号
    String name; // 名称
```

成员方法：

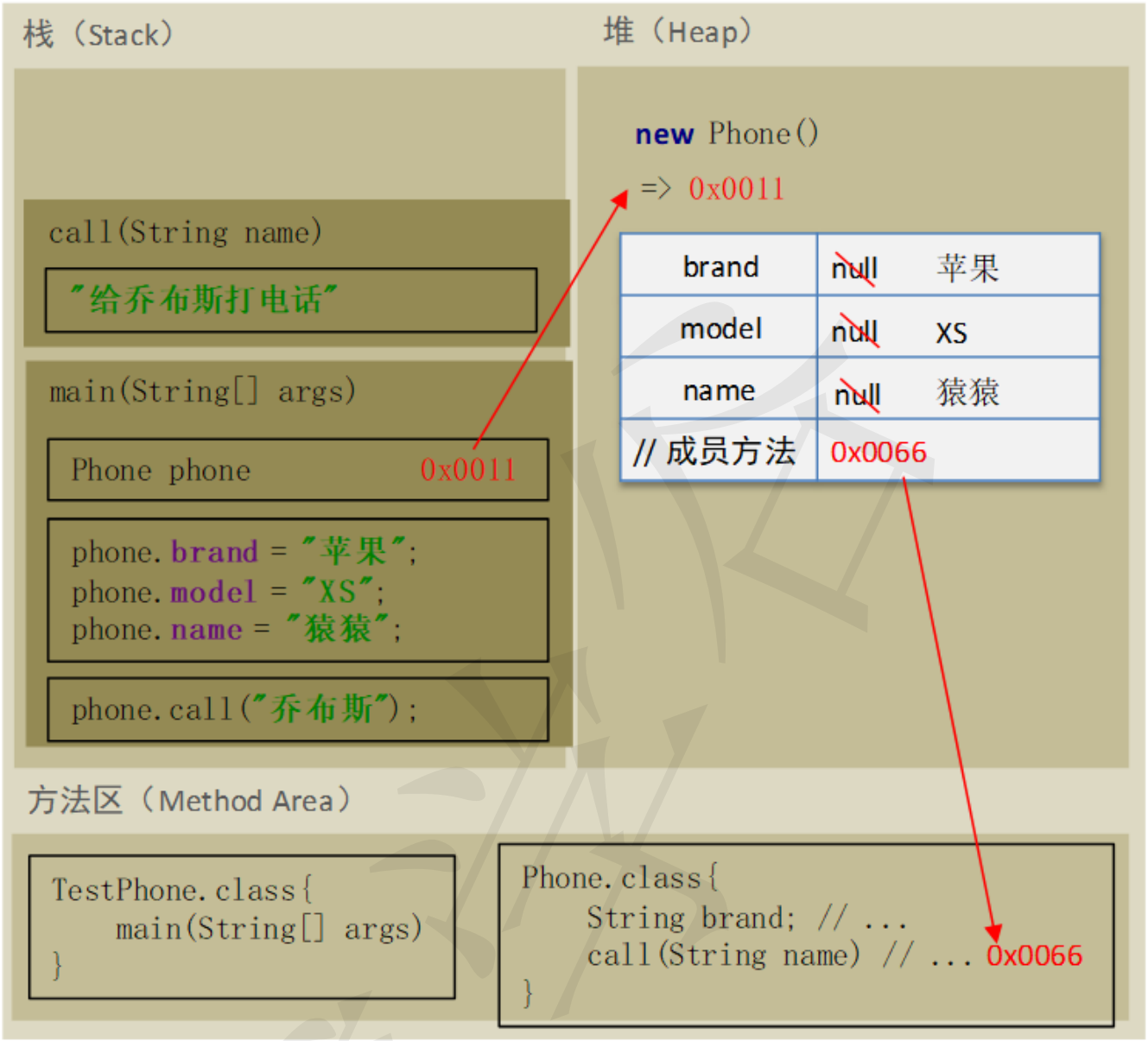
```
    call(); // 打电话
    sendMessage(); // 发短信
    playGame(); // 玩游戏
```

使用

```
    Phone stu = new Phone();
```

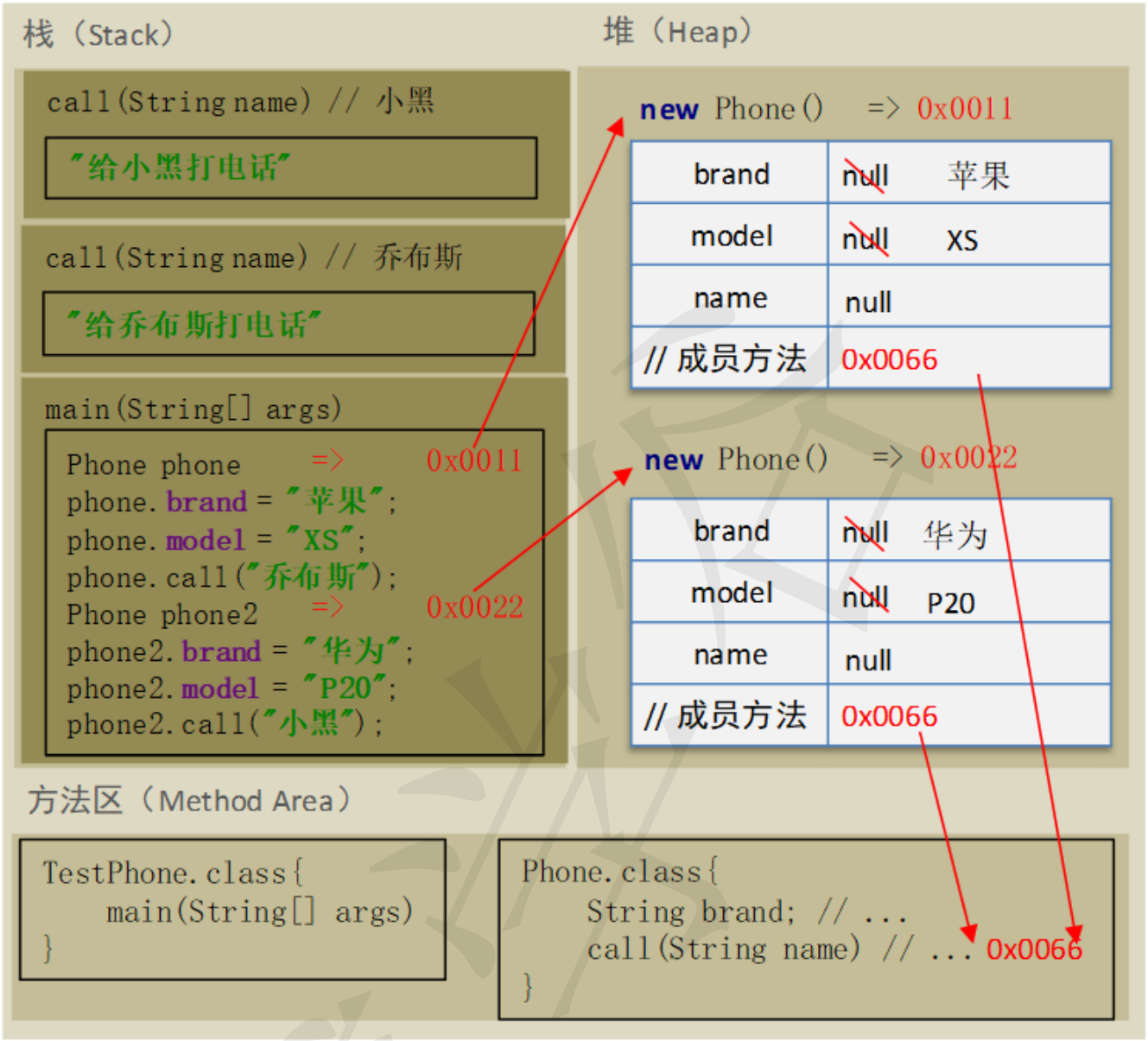
一个对象创建过程的内存图解

```
public class TestPhone {
    public static void main(String[] args) {
        Phone phone = new Phone();
        System.out.println(phone.brand);
        System.out.println(phone.model);
        System.out.println(phone.name);
        phone.brand = "苹果";
        phone.model = "XS";
        phone.name = "猿猿";
        phone.call("乔布斯");
    }
}
public class Phone {
    String brand;
    String model;
    String name;
    public void call(String name) {
        System.out.println("给" + name + "打电话");
    }
    public void sendMessage() {...}
    public void playGame() {...}
}
```



两个对象创建过程的内存图解

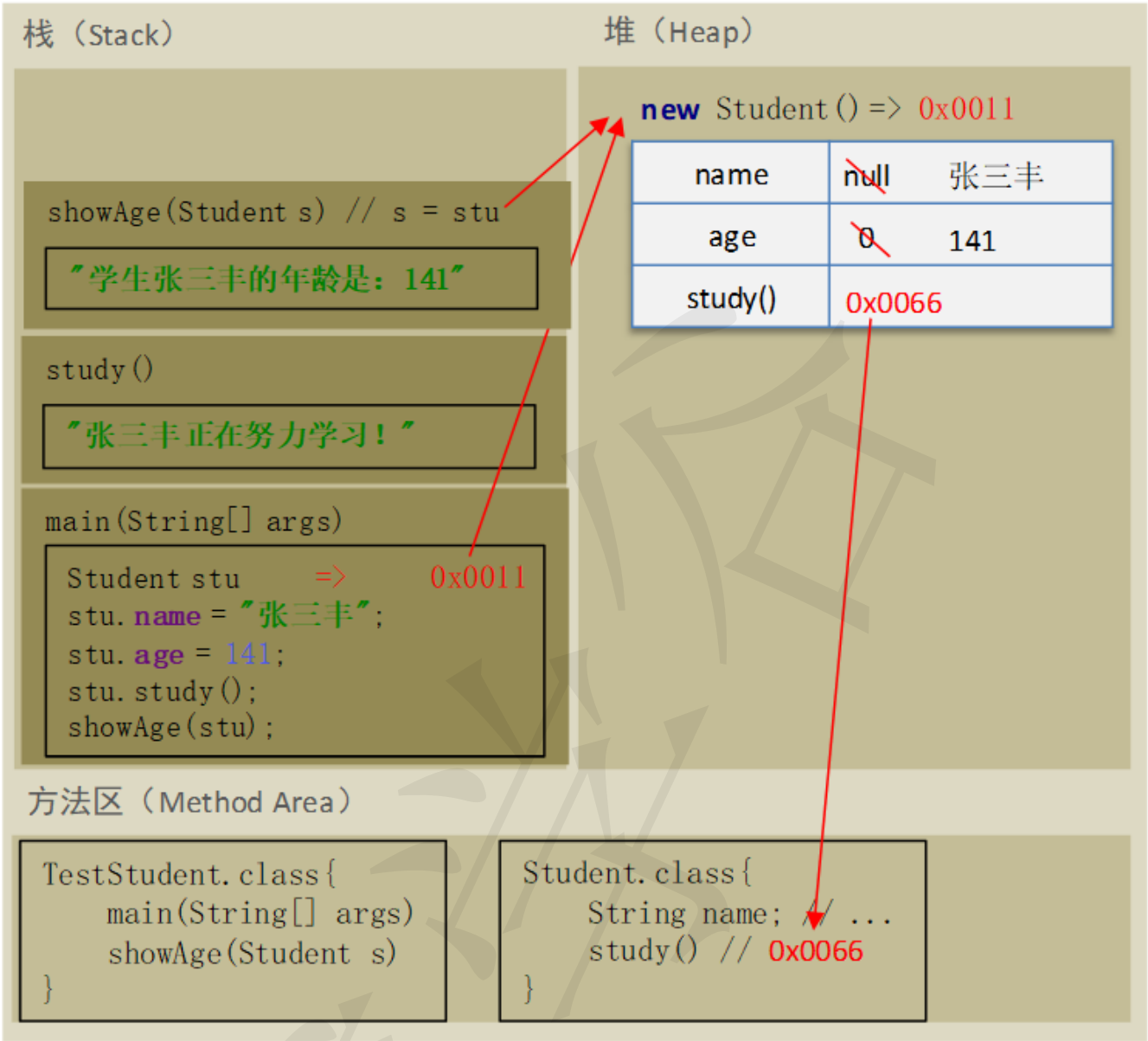
```
public class TestPhone {
    public static void main(String[] args) {
        Phone phone = new Phone();
        phone.brand = "苹果";
        phone.model = "XS";
        phone.call("乔布斯");
        Phone phone2 = new Phone();
        phone2.brand = "华为";
        phone2.model = "P20";
        phone2.call("小黑");
    }
}
public class Phone {
    String brand;
    String model;
    String name;
    public void call(String name) {
        System.out.println("给" + name + "打电话");
    }
}
```



对象作为参数的内存图解

对比数组，自定义对象作为方法的参数，和数组作为方法参数的道理是一样的

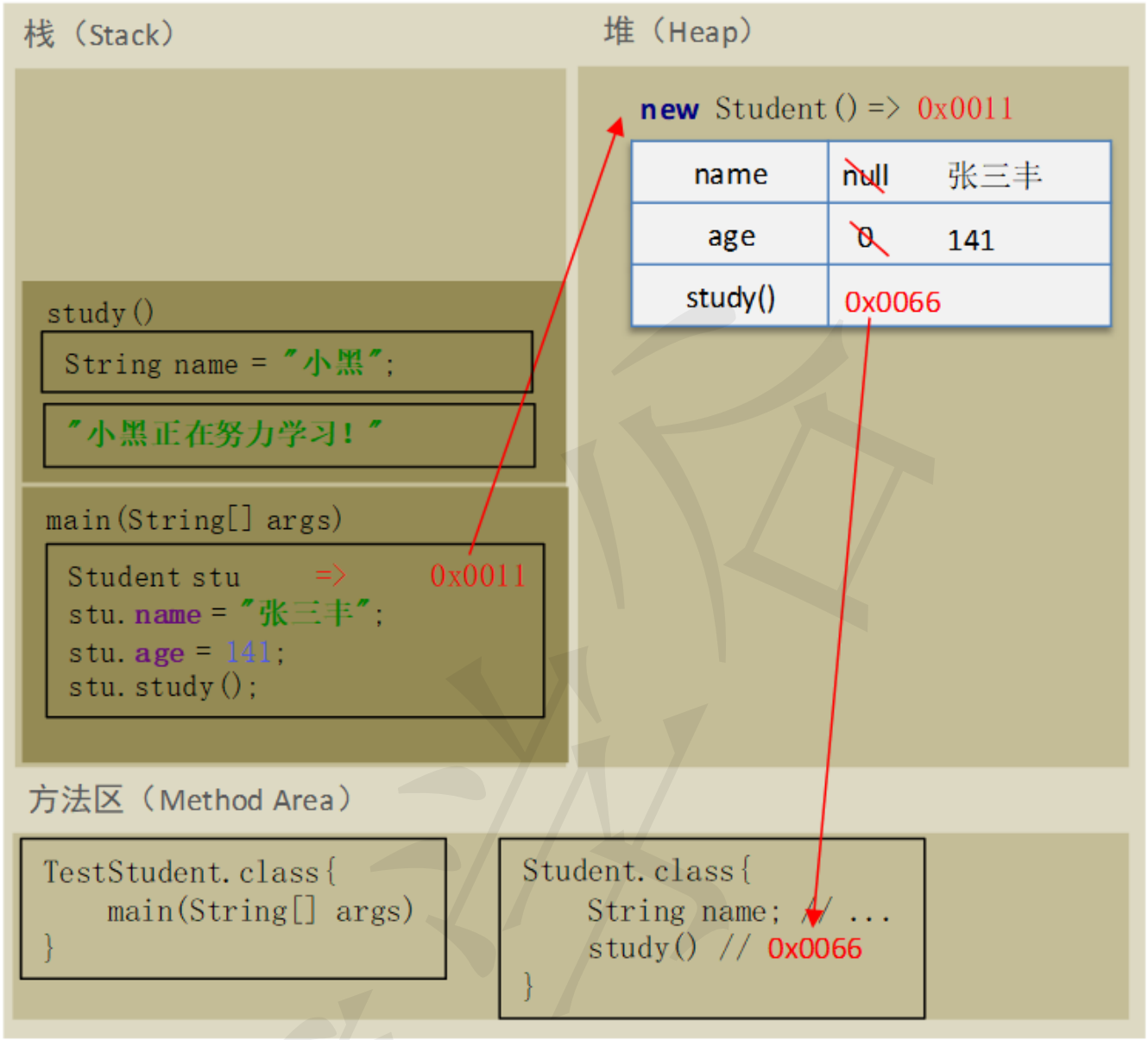
```
public class TestStudent {
    public static void main(String[] args) {
        Student stu = new Student();
        stu.name = "张三丰";
        stu.age = 141;
        stu.study();
        showAge(stu);
    }
    public static void showAge(Student s) {
        System.out.println("学生" + s.name
                + "的年龄是：" + s.age);
    }
}
public class Student {
    String name;
    int age;
    public void study() {
        System.out.println(name + "正在努力学习！");
    }
}
```



成员变量和局部变量的区别

当方法中定义和类的属性名相同的变量，如何调用呢?

```
public class TestStudent {
    public static void main(String[] args) {
        Student stu = new Student();
        stu.name = "张三丰";
        stu.age = 141;
        stu.study();
    }
}
public class Student {
    String name;
    int age;
    public void study() {
        String name = "小黑";
        System.out.println(name + "正在努力学习！");
    }
}
```



定义位置

成员变量：类中，方法外 局部变量：方法中，或形式参数

初始化值

成员变量：有默认初始化值 局部变量：无默认初始化值，必须先赋值再使用

作用范围

成员变量：在类中 局部变量：在方法中

内存中的位置

成员变量：堆内存 局部变量：栈内存

生命周期

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，随着方法调用完毕而消失

注意事项

局部变量和成员变量重名时，采用就近原则

封装概述

什么是封装?

基本义

把物体打包装到箱子里，然后封起来

引申义

把一系列功能打包到一台设备里，提供使用这些功能的界面

常见的封装体

汽车、电脑、洗衣机...

封装的好处

提高安全性

提高复用性

将复杂的事情简单化

思考：Java中哪些内容体现了封装?

方法、类...

方法的封装性：将繁多的代码整合在一起，以一个方法的形式呈现，所以方法是一个封装体 类的封装性：现实事物的属性和行为都包含在了类中，所以类也是一个封装体

Java中的封装体

方法 安全性：调用者不知道方法的具体实现 复用性：方法可以被重复使用 简单化：将繁多的代码以一个方法的方式呈现，仅通过调用方法就可以实现功能；代码维护也变得简单 类 安全性：？复用性：类的对象可以被重复使用 简单化：类的对象包含了更多的功能，使用起来也更方便

```
public class Student {
    String name;
    int age;
    public void study() {
        String name = "小黑";
        System.out.println(name + "正在努力学习！");
    }
    // 其它方法
}
```

private关键字

private的基本概念

私有的，一种访问权限修饰符，用来修饰类的成员

特点

被修饰的成员只能在本类中访问

用法

private 数据类型 变量名;

private 返回值类型 方法名(参数列表) { }

扩展

public，公共的，访问权限修饰符，用来修饰类、成员变量、成员方法等，被修饰的内容可以在任意类中访问

案例：private修饰成员变量

需求:

给Student类的成员变量用private修饰，然后在测试类中正确使用该成员变量

分析:

A：给成员变量添加private修饰后，测试类中将不能直接访问

B：由于private的特性，需要在Student类中添加访问该属性的方法，供其它类调用

C：属性的操作一般都是取值和赋值，所以添加对应的公共方法：

getXxx() setXxx(参数)

D：在测试类中通过getXxx()和setXxx(参数)方法来实现属性的访问

this关键字

this的基本概念

这，这个，表示本类对象的引用，本质是一个对象

特点

每一个普通方法都有一个this，谁调用该方法，this就指向谁

用法

this.属性名; this.方法名(参数);

案例：使用this完善set方法

需求：Student类中的set方法的参数名应按规范命名

分析：

A：将set方法参数名按规范改写后，会与成员变量重名，使用this关键字进行区分： this.name = name; this.age = age; B：在测试类中通过setXxx(参数)方法来赋值

```
public class Student {
    private String name;
    private int age;

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

标准代码：JavaBean

构造方法的基本概念

构建、创造，也叫构造器，用来帮助创建对象的方法，准确的说，构造方法的作用是初始化对象。

谁来创建对象?

new关键字。Java中通过new关键字创建对象，并在内存中开辟空间，然后使用构造方法（构造器）完成对象的初始化工作。

构造方法的定义

格式

```
修饰符 构造方法名(参数列表){
    // 方法体
}
```

要求

- 方法名必须与类名相同
- 没有返回值
- 没有返回值类型

注意事项

- 若未提供任何构造方法，系统会给出默认无参构造
- 若已提供任何构造方法，系统不再提供无参构造
- 构造方法可以重载

```
public class Student {
    public Student() {}

    private String name;
    private int age;

    // ...
}
```

标准代码：JavaBean

Java语言编写类的标准规范。符合JavaBean标准的类，必须是具体的、公共的，并且具有无参数的构造方法，提供用来操作成员变量的set 和get 方法

```
public class Student {
    public Student() {}

    private String name;

    private int age;

    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;
    }
}
```

Java中封装的概念

将一系列相关事物共同的属性和行为提取出来，放到一个类中，隐藏对象的属性和实现细节，仅对外提供公共的访问方式。

隐藏对象数据的实现细节，意味着一个类可以全面的改变存储数据的方式，只要使用同样的方法操作数据，

其它对象就不会知道或介意所发生的变化。

封装的关键

绝对不能让类中的方法直接访问其它类的数据（属性），程序仅通过对象的方法与对象的数据进行交互。