面向对象

写程序的套路

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

啥是面向对象编程?

- 面向：找、拿。
- 对象：东西。
- 面向对象编程：找或拿东西过来编程。

面向对象编程的例子

```
public class Test {
    public static void main(String[] args) {
        // 1、创建一个扫描器对象，用于接收用户输入的数据
        Scanner sc = new Scanner(System.in);
        System.out.println("请您输入您的年龄：");
        int age = sc.nextInt();
        System.out.println(age);

        // 2、得到一个随机数对象，用于得到随机数
        Random r = new Random();
        int data = r.nextInt(10) + 1 ; // 生成 1-10之间的随机数
        System.out.println(data);

    }
}
```

这就是面向对象编程？？

为什么用面向对象编程？

面向对象编程学什么？？

面向对象编程的好处

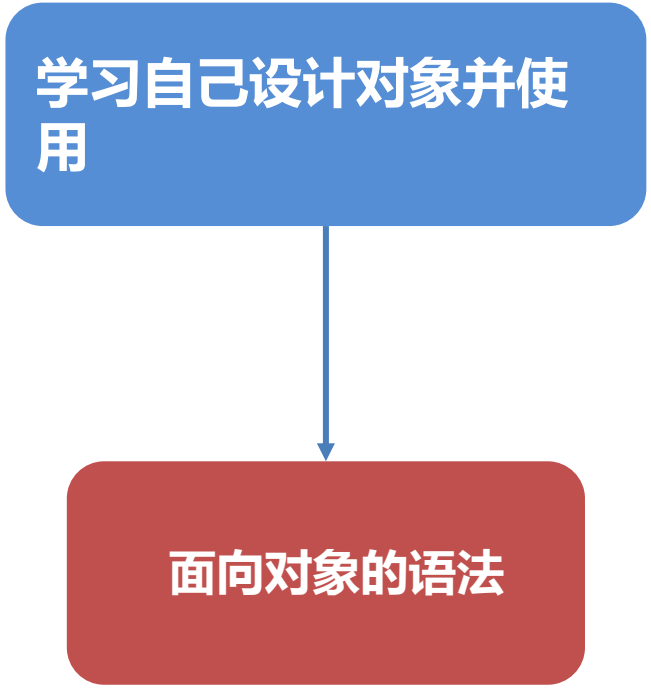
```
public class Test {
    public static void main(String[] args) {
        // 1、创建一个扫描器对象，用于接收用户输入的数据
        Scanner sc = new Scanner(System.in);
        System.out.println("请您输入您的年龄：");
        int age = sc.nextInt();
        System.out.println(age);

        // 2、得到一个随机数对象，用于得到随机数
        Random r = new Random();
        // 生成 1-10之间的随机数
        int data = r.nextInt(10) + 1 ;
        System.out.println(data);
    }
}
```

符合人类思维习惯，编程更简单、更好理解

为什么用面向对象编程？　面向对象编程学什么？？

面向对象学习什么？

```
public class Test {
    public static void main(String[] args) {
        // 1、创建一个扫描器对象，用于接收用户输入的数据
        Scanner sc = new Scanner(System.in);
        System.out.println("请您输入您的年龄：");
        int age = sc.nextInt();
        System.out.println(age);

        // 2、得到一个随机数对象，用于得到随机数
        Random r = new Random();
        // 生成 1-10之间的随机数
        int data = r.nextInt(10) + 1 ;
        System.out.println(data);
    }
}
```



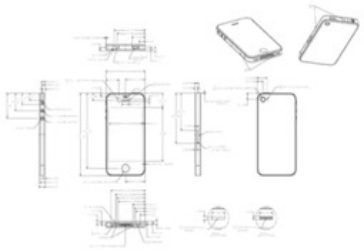
目录

Contents

类是什么？






设计图

类(设计图)：是对象共同特征的描述；

对象：是真实存在的具体实例。

结论：在Java中，必须先设计类，才能创建对象并使用。

如何设计类

```
public class 类名 {
        1、成员变量（代表属性,一般是名词）
        2、成员方法（代表行为,一般是动词）
        3、构造器 （后几节再说）
        4、代码块 （后面再学）
        5、内部类 （后面再学）
}
```

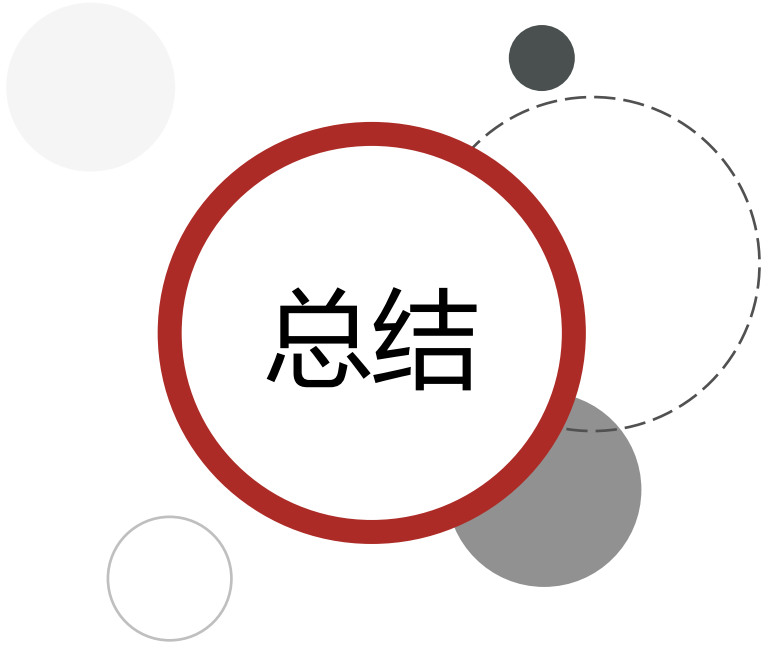
```
public class Car {
    // 属性（成员变量）
    String name;
    double price;

    // 行为（方法）
    public void start(){

    }
    public void run(){

    }
}
```

如何得到类的对象

```
类名 对象名 = new 类名();
Car c = new Car();
```

如何使用对象

- 访问属性： 对象名.成员变量
- 访问行为： 对象名.方法名(...)



1．类和对象是什么？

- **类：是共同特征的描述(设计图)；对象：是真实存在的具体实例。**

2．如何设计类？

```
public class 类名 {
    1、成员变量（代表属性的,一般是名词）
    2、成员方法（代表行为的,一般是动词）
}
```

3．如何创建对象？

```
类名 对象名 = new 类名();
```

4．拿到对象后怎么访问对象的信息？

- **对象.成员变量；**
- **对象.成员方法(…)**

练习 练习时间（10分钟）

- 请同学们模仿汽车类，自己定义一个学生类
- 随便定义2个属性，2个行为。
- 并创建2个学生对象，分别访问属性和行为。

目录

Contents

定义类的补充注意事项

- 类名首字母建议大写，且有意义，满足“驼峰模式”。
- 一个Java文件中可以定义多个class类，但只能一个类是public修饰，而且public修饰的类名必须成为代码文件名。

  实际开发中建议还是一个文件定义一个class类。
- 成员变量的完整定义格式是：修饰符 数据类型 变量名称 = 初始化值； 一般无需指定初始化值，存在默认值。

```
public class Student {
    //属性(成员变量)
    String name;
    double height;
    //行为(方法)
    public void study(){

    }
    public void run(){

    }
}
```

成员变量的默认值规则

数据类型	明细	默认值
基本类型	byte、short、char、int、long	0
	float、double	0.0
	boolean	false
引用类型	类、接口、数组、String	null

总结

1. 定义类有哪些建议，有什么需要注意的？

- **类名首字母建议大写、英文、有意义，满足驼峰模式，不能用关键字，满足标志符规定**
- **一个代码文件中可以定义多个类，但是只能一个类是public修饰的，public修饰的类名必须是Java代码的文件名称。**

2. 成员变量的格式是什么样的，有什么特点？

- **成员变量的完整格式是：修饰符 数据类型 变量名称 = 初始化值；**
- **一般无需为成员变量指定初始化值，存在默认值。**

目录

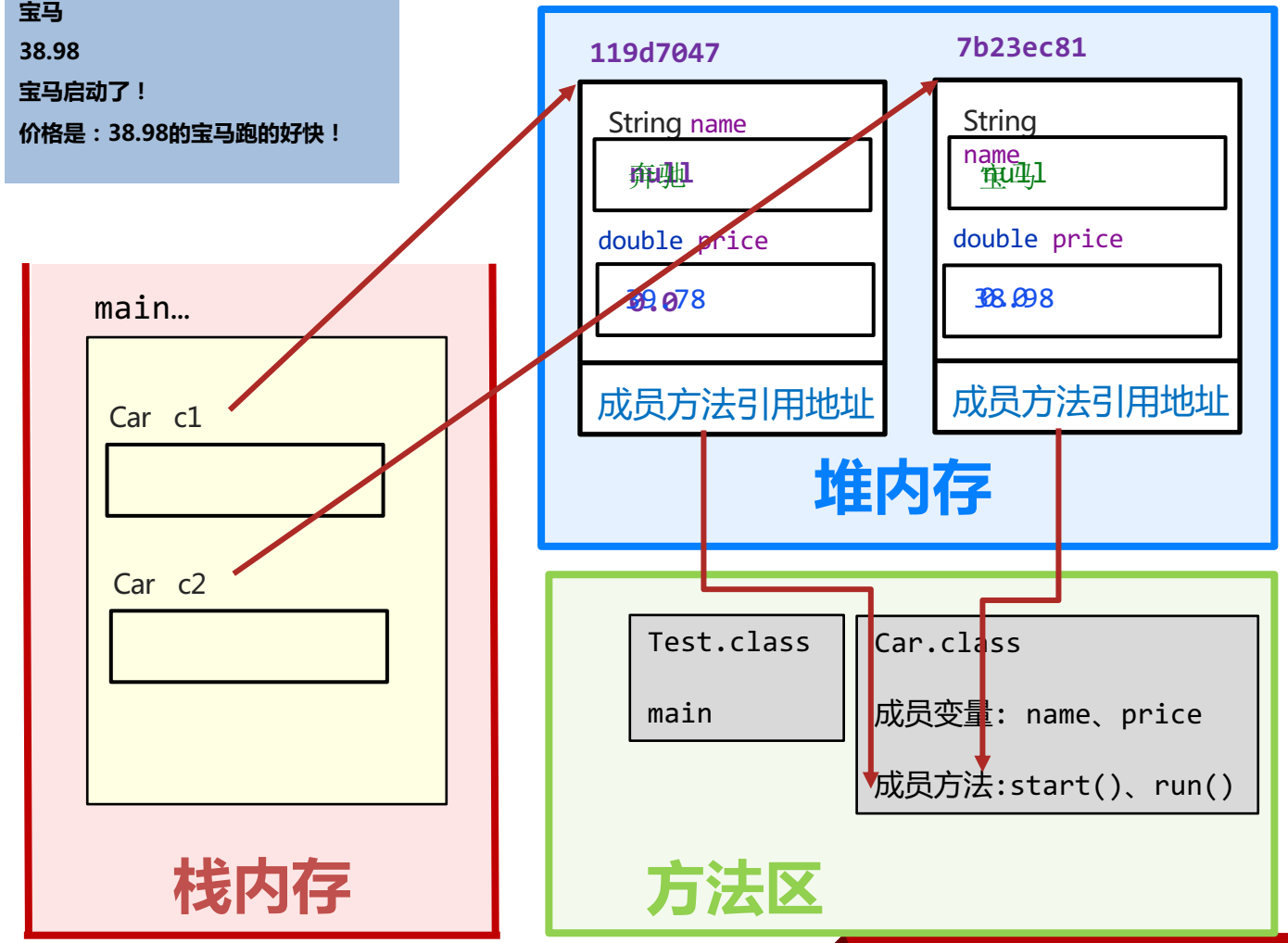
Contents

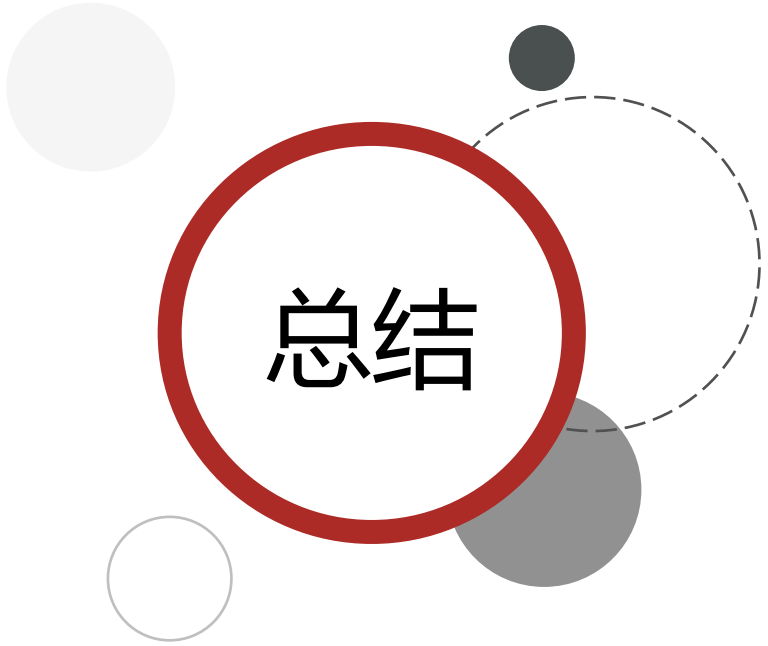
两个对象内存图

```
public class Car {
    // 成员变量(属性)
    String name;
    double price;
    // 方法(行为)
    public void start(){
        System.out.println(name+“启动了！”);
    }
    public void run(){
        System.out.println(“价格是：” + price +"的" + name+"跑的快！");
    }
}
```

```
public class Test {
    public static void main(String[] args) {
        Car c1 = new Car();
        c1.name = "奔驰";
        c1.price = 39.78;
        System.out.println(c1.name);
        System.out.println(c1.price);
        c1.start();
        c1.run();

        Car c2 = new Car();
        c2.name = "宝马";
        c2.price = 38.98;
        System.out.println(c2.name);
        System.out.println(c2.price);
        c2.start();
        c2.run();
    }
}
```

```
奔驰
39.78
奔驰启动了！
价格为：39.78的奔驰跑的好快！
宝马
38.98
宝马启动了！
价格是：38.98的宝马跑的好快！
```





1. 对象到底是放在哪个位置的？

- **堆内存中**

2. Car c = new Car(); c变量名中存储的是什么？

- **存储的是对象在堆内存中的地址。**

3. 成员变量（name、price）的数据放在哪里，存在于哪个位置？

- **对象中，存在于堆内存中。**

目录

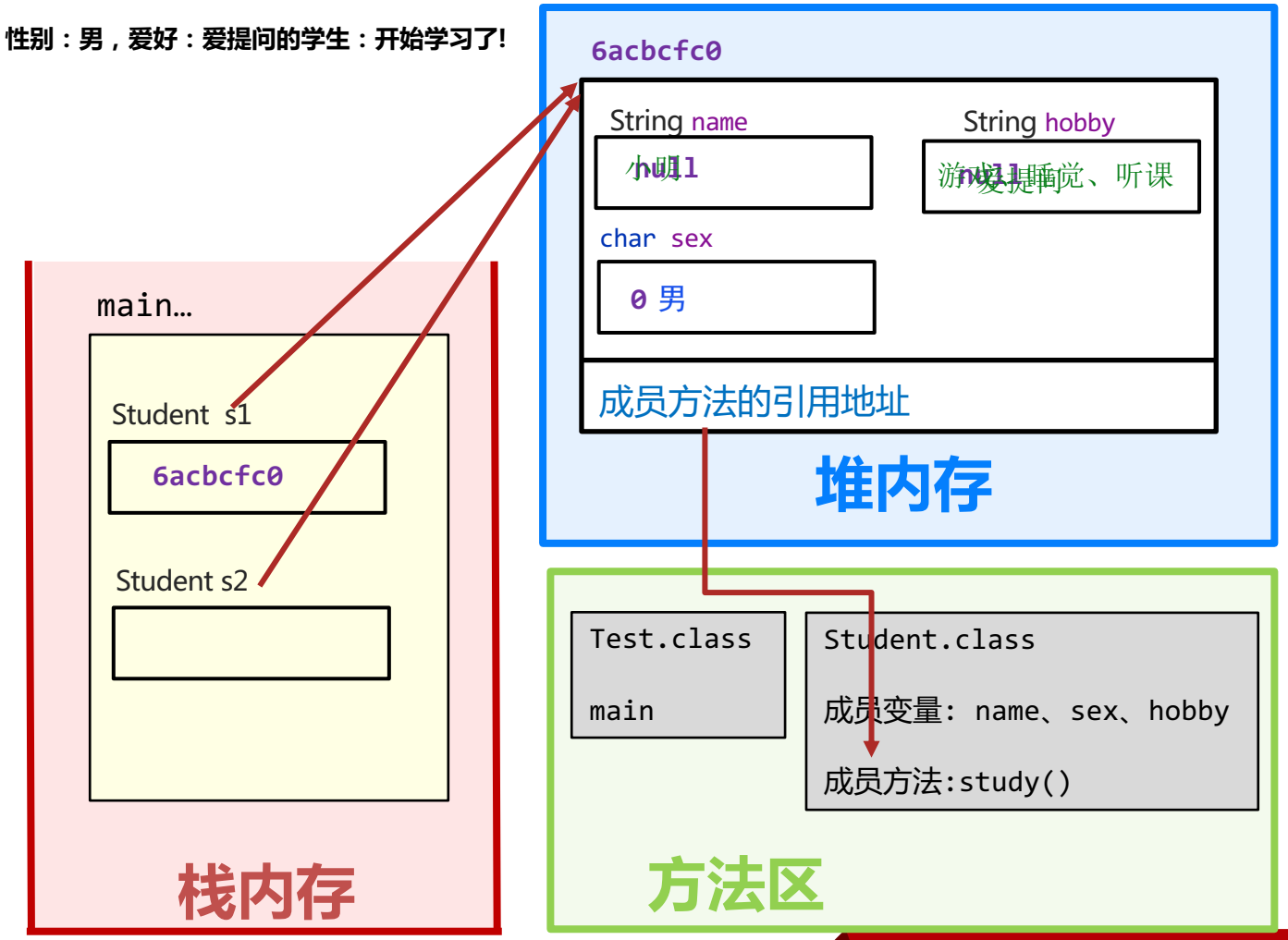
Contents

两个变量指向同一个对象内存图

```
public class Student {
    String name;
    char sex;
    String hobby; // 爱好

    public void study(){
        System.out.println("名称：" + name + "，性别：" + sex
+ "，爱好：" + hobby + "的学生：开始学习了！");
    }
}
```

```
public class Test {
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.name = "小明";
        s1.sex = '男';
        s1.hobby = "游戏、睡觉、听课";
        s1.study();

        //把学生类型的s1变量赋值给学生类型的s2变量
        Student s2 = s1;
        s2.hobby = "爱提问";

        System.out.println(s2.name);
        System.out.println(s2.sex);
        System.out.println(s1.hobby);
        s2.study();
    }
}
```

名称：小明，性别：男，爱好：游戏、睡觉、听课的学生：开始学习了！
小明
男
爱提问
名称：小明，性别：男，爱好：爱提问的学生：开始学习了!



垃圾回收

- 注意：当堆内存中的**对象**，没有被任何变量引用（指向）时，就会被判定为内存中的**“垃圾”**。

两个变量指向同一个对象内存图

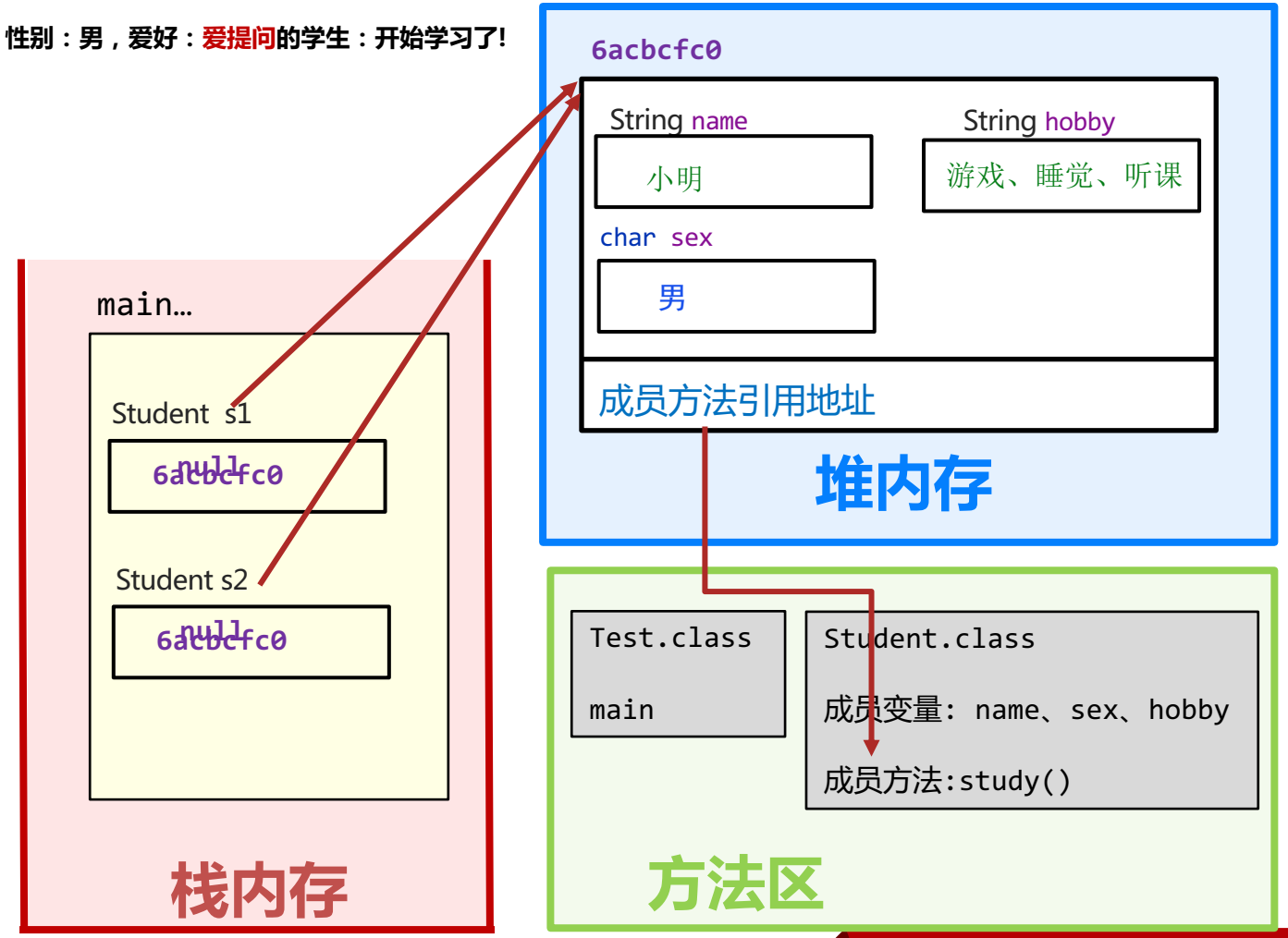
```
public class Student {
    String name;
    char sex;
    String hobby; // 爱好

    public void study(){
        System.out.println("名称：" + name + "，性别：" + sex
+ "，爱好：" + hobby + "的学生：开始学习了！");
    }
}
```

```
public class Test {
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.name = "小明";
        s1.sex = '男';
        s1.hobby = "游戏、睡觉、听课";
        s1.study();

        //把学生类型的s1变量赋值给学生类型的s2变量
        Student s2 = s1;
        s2.hobby = "爱提问";

        System.out.println(s2.name);
        System.out.println(s2.sex);
        System.out.println(s1.hobby);
        s2.study();
    }
}
```

```
s1 = null;
s2 = null;
```

名称：小明，性别：男，爱好：游戏、睡觉、听课的学生：开始学习了！
小明
男
爱提问
名称：小明，性别：男，爱好：爱提问的学生：开始学习了!

Java存在自动垃圾回收器，会定期进行清理。



目录

Contents

学构造器的目的？

Get1	真正知道对象具体是通过调用什么代码得到的。
Get2	能够掌握为对象赋值的其他简便写法。
Get3	为以后学习面向对象编程的其他内容做支撑。

构造器的作用

- 定义在类中的，可以用于初始化一个类的对象，并返回对象的地址。

```
Car c = new Car();
```

构造器的格式

```
修饰符 类名(形参列表){
    ...
}
```

```
public class Car {
        ...
        // 无参数构造器
        public Car(){
                ...
        }
        // 有参数构造器
        public Car(String n, double p){
                ...
        }
}
```

调用构造器得到对象的格式

```
类 变量名称 = new 构造器；
```

```
Car c = new Car();
```

```
Car c1 = new Car("奔驰", 39.8);
```

构造器的分类和作用

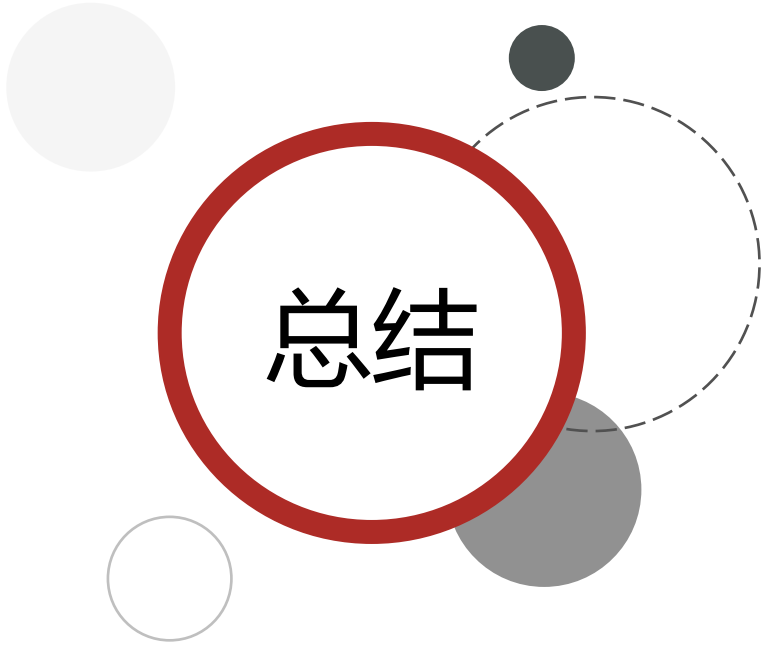
- 无参数构造器（默认存在的）：初始化对象时，成员变量的数据均采用默认值。
- 有参数构造器：在初始化对象的时候，同时可以接收参数为对象进行赋值。

构造器的注意事项

- 任何类定义出来，默认就自带了无参数构造器，写不写都有。
- 一旦定义了有参数构造器，那么无参数构造器就没有了，如果还想用无参数构造器，此时就需要自己手写一个无参数构造器了。

```
public class Car {
    ...
  // 无参数构造器（默认存在的）
}
```

```
public class Car {
    ...
  // 无参数构造器（需要写出来了）
  public Car(){

  }
  // 有参数构造器
  public Car(String n, String b){

  }
}
```



1.构造器的作用？

- **初始化类的对象，并返回对象的地址。**

2.构造器有几种，各自的作用是什么？

- **无参数构造器：初始化对象时，成员变量的数据均采用默认值。**
- **有参数构造器：在初始化对象的时候，同时可以接收参数为对象进行赋值。**

3.构造器有哪些注意事项？

- **任何类定义出来，默认就自带了无参数构造器，写不写都有。**
- **一旦定义了有参数构造器，无参数构造器就没有了，此时就需要自己写无参数构造器了。**

目录

Contents

this关键字是什么

- 可以出现在构造器、方法中
- 代表当前对象的地址。

```
public class Car {
    public Car() {
        System.out.println("this在构造器中：" + this);
    }

    public void run() {
        System.out.println("this在方法中：" + this);
    }
}
```

```
public class Test2{
    public static void main(String[] args) {
        Car c = new Car();
        c.run();
        System.out.println(c);
    }
}
```

this关键字的作用

- 可以用于指定访问当前对象的成员变量、成员方法。

this出现在有参数构造器中的用法

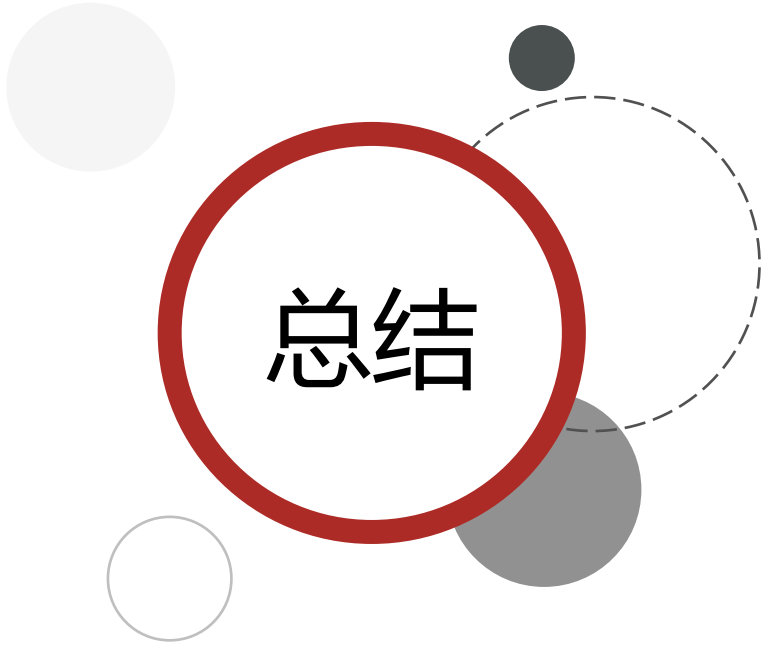
```
public class Car {
    String name;
    double price;

    public Car(String n , double p){
        name = n;
        price = p;
    }
}
```


```
public class Car {
    String name;
    double price;

    public Car(String name , double price){
              name = name;
              price = price;
    }
}
```

this出现在成员方法中的用法

```
public class Car {
    String name;
    double price;

    public void goWith(String name){
        System.out.println(name + "正在和" + name + "一起比赛！！");
    }
}
```

```
public class Car {
    String name;
    double price;

    public void goWith(String name){
        System.out.println(this.name + "正在和" + name + "一起比赛！！");
    }
}
```



1. this关键字是什么?

- **出现在构造器和成员方法中，代表当前对象的地址。**

2. this关键字在构造器中、成员方法中可以做什么?

- **可以用于指定访问当前对象的成员。**

目录

Contents

封装

- 面向对象的三大特征：**封装，继承，多态**。
- 封装：告诉我们，如何正确设计对象的属性和方法。

需求

- 请设计一个人对象，且要求这个对象有

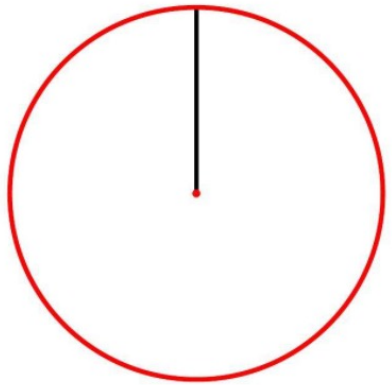
  名称、年龄，能吃饭、睡觉。

```
public class People {
    名称
    年龄
    吃饭
    睡觉
}
```

封装

- 面向对象的三大特征：**封装，继承，多态。**
- 封装：告诉我们，如何正确设计对象的属性和方法。
- 封装的原则：对象代表什么，就得封装对应的数据，并提供数据对应的行为。

```
public void draw(){


}
```



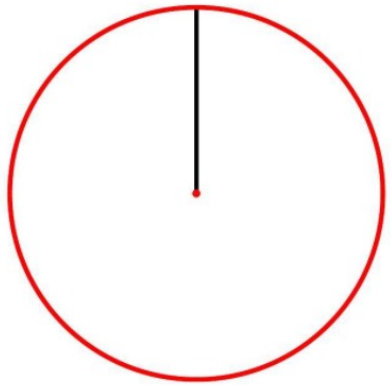
人画圆

```
public class People {

    ...


}
```

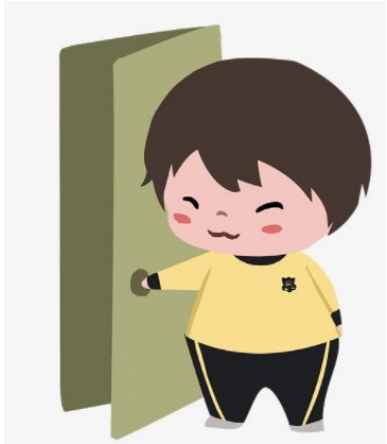
```
public class Circle {

    double radius; // 半径


    System.out.println("按照半径"
            + radius +"画了一个圆");
}
```

封装

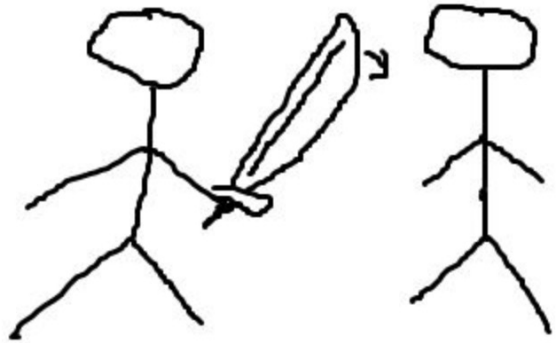
- 面向对象的三大特征：**封装，继承，多态**。
- 封装：告诉我们，如何正确设计对象的属性和方法。
- 封装的原则：对象代表什么，就得封装对应的数据，并提供数据对应的行为。



人画圆



人关门

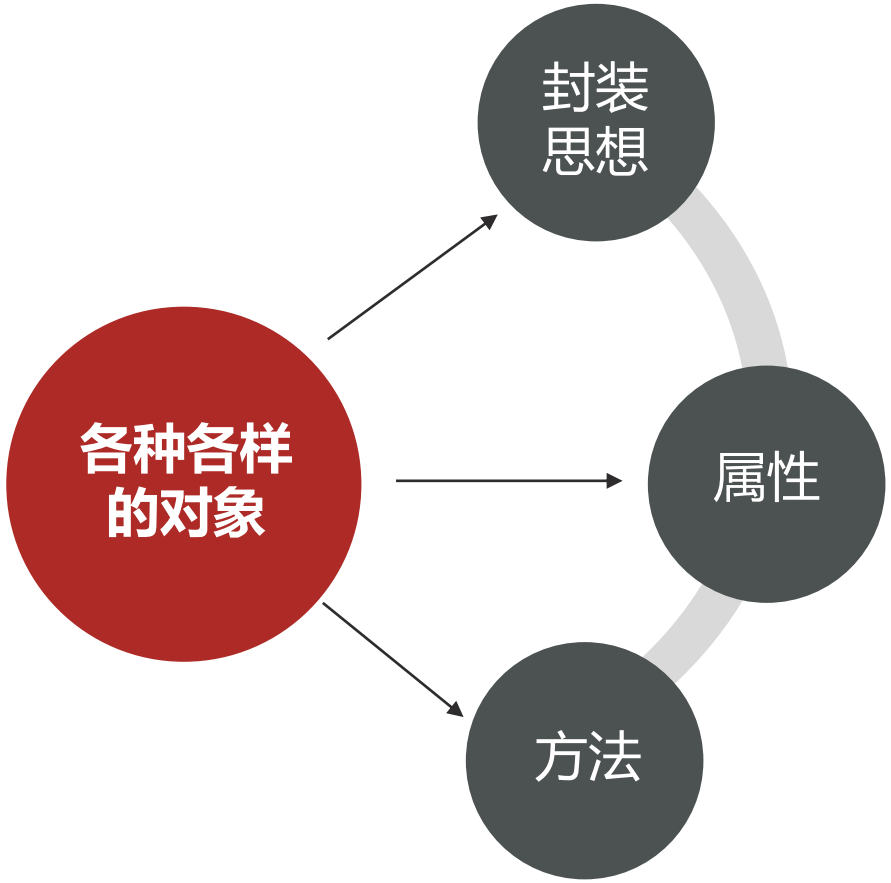


张三砍死了李四

xx人持刀行凶，造成了xx死亡的后果

理解封装思想有啥好处？

理解封装思想有啥好处？

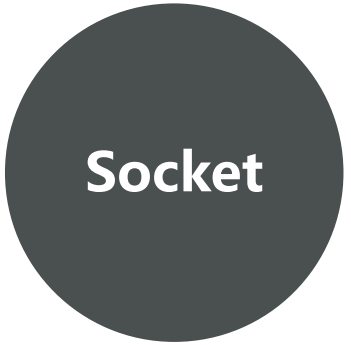


- **有什么事，找对象，调方法就行，编程变得很简单。**

理解封装思想有啥好处？

代表字符串对象

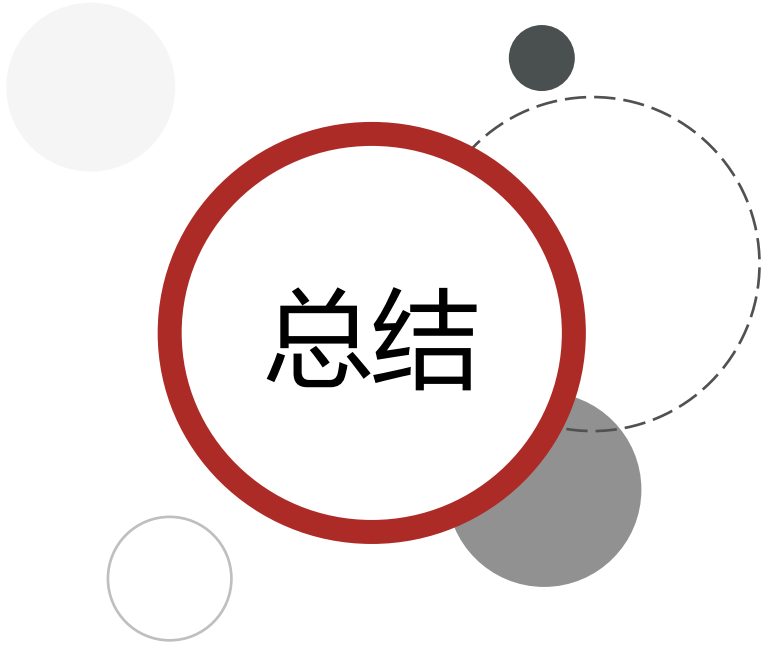
拥有操作字符串的很多方法



代表一个网络连接

可以连接别人，发消息，收消息

- **有什么事，找对象，调方法就行，编程变得很简单。**
- **降低我们的学习成本，可以少学、少记。**



1.什么是封装啊？

- **告诉我们，如何正确设计对象的属性和方法。**
- **原则：对象代表什么，就得封装对应的数据，并提供数据对应的行为。**

2. 理解封装思想有什么好处？

- **让编程变得很简单，有什么事，找对象，调方法就行。**
- **降低我们的学习成本，可以少学、少记，或者说压根不用学，**

  不用记对象的那么多方法，有需要时去找就行。

目录

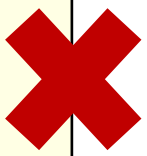
Contents

如何进行封装更好？

- 一般建议对成员变量使用private(私有、隐藏)关键字修饰进（private修饰的成员只能在当前类中访问）。
- 为每个成员变量提供配套public修饰的的getter、setter方法暴露其取值和赋值。

```
public class Student {
    int age;
}
```

```
public class Test {
    public static void main(String[] args) {
        Student s = new Student();
        s.age = -121;
        System.out.println(s.age); // -121
    }
}
```



```
public class Student {
    private int age;
}
```

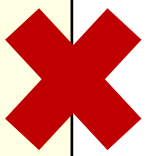
```
public class Test {
    public static void main(String[] args) {
        Student s = new Student();
        s.age;
    }
}
```

如何进行封装更好？

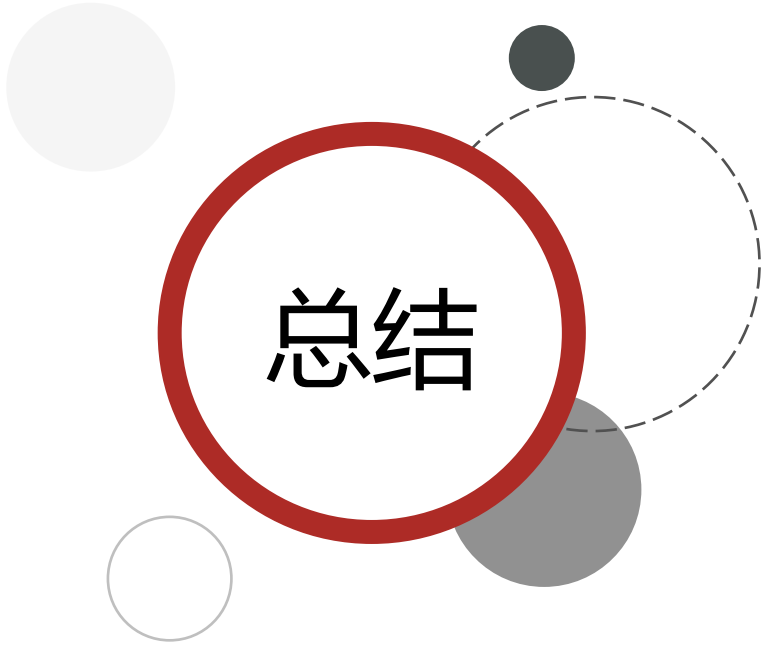
- 一般建议对成员变量使用private(私有、隐藏)关键字修饰进（private修饰的成员只能在当前类中访问）。
- 为每个成员变量提供配套public修饰的的getter、setter方法暴露其取值和赋值。

```
public class Student {
    int age;
}
```

```
public class Test {
    public static void main(String[] args) {
        Student s = new Student();
        s.age = -121;
        System.out.println(s.age); // -121
    }
}
```



```
public class Student {
    private int age;

    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        if (age >= 0 && age <= 200) {
            this.age = age;
        } else {
            System.out.println("请检查年龄数值!!");
        }
    }
}
```



1. 如何进行更好的封装？

- **一般会把成员变量使用private隐藏起来，对外就不能直接访问了。**
- **提供public修饰的getter和setter方法暴露其取值和赋值。**

目录

Contents

JavaBean

- 也可以称为实体类，其对象可以用于在程序中封装数据。

学生类 汽车类 用户类 测试类Test

标准JavaBean须满足如下书写要求：

- 成员变量使用 **private** 修饰。
- 提供成员变量对应的 **setXxx() / getXxx()方法。**
- 必须提供一个**无参构造器；有参数构造器是可写可不写的。**

目录

Contents

成员变量和局部变量的区别

区别	成员变量	局部变量
类中位置不同	类中，方法外	常见于方法中
初始化值不同		
内存位置不同		
生命周期不同	随着对象的创建而存在，随着对象的消失而消失	随
作用域		

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

```
public class Test {
    public static void main(String[] args){
        double score = 99.9;
        String name = "黑马程序员";
        System.out.println(name + ":" + score);
    }
}
```

成员变量和局部变量的区别

区别	成员变量	局部变量
类中位置不同	类中，方法外	常见于方法中
初始化值不同	有默认值,无需初始化	没有默认值，使用之前需要完成赋值
内存位置不同		
生命周期不同	随着对象	随
作用域		

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

```
public class Test {
    public static void main(String[] args){
        double score = 99.9;
        String name = "黑马程序员";
        System.out.println(name + ":" + score);
    }
}
```

成员变量和局部变量的区别

区别	成员变量	局部变量
类中位置不同	类中，方法外	常见于方法中
初始化值不同	有默认值,无需初始化	没有默认值，使用之前需要完成赋值
内存位置不同	堆内存	栈内存
生命周期不同	随着对象	随
作用域		

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

```
public class Test {
    public static void main(String[] args){
        double score = 99.9;
        String name = "黑马程序员";
        System.out.println(name + ":" + score);
    }
}
```

成员变量和局部变量的区别

区别	成员变量	局部变量
类中位置不同	类中，方法外	常见于方法中
初始化值不同	有默认值,无需初始化	没有默认值，使用之前需要完成赋值
内存位置不同	堆内存	栈内存
生命周期不同	随着对象的创建而存在，随着对象的消失而消失	随着方法的调用而存在，随着方法的运行结束而消失
作用域		

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

```
public class Test {
    public static void main(String[] args){
        double score = 99.9;
        String name = "黑马程序员";
        System.out.println(name + ":" + score);
    }
```

成员变量和局部变量的区别

区别	成员变量	局部变量
类中位置不同	类中，方法外	常见于方法中
初始化值不同	有默认值,无需初始化	没有默认值，使用之前需要完成赋值
内存位置不同	堆内存	栈内存
生命周期不同	随着对象的创建而存在，随着对象的消失而消失	随着方法的调用而存在，随着方法的运行结束而消失
作用域		在所归属的大括号中

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

```
public class Test {
    public static void main(String[] args){
        double score = 99.9;
        String name = "黑马程序员";
```

成员变量和局部变量的区别

区别	成员变量	局部变量
类中位置不同	类中，方法外	常见于方法中
初始化值不同	有默认值,无需初始化	
内存位置不同		
生命周期不同	随着	肖
作用域		在所归属的大括号中

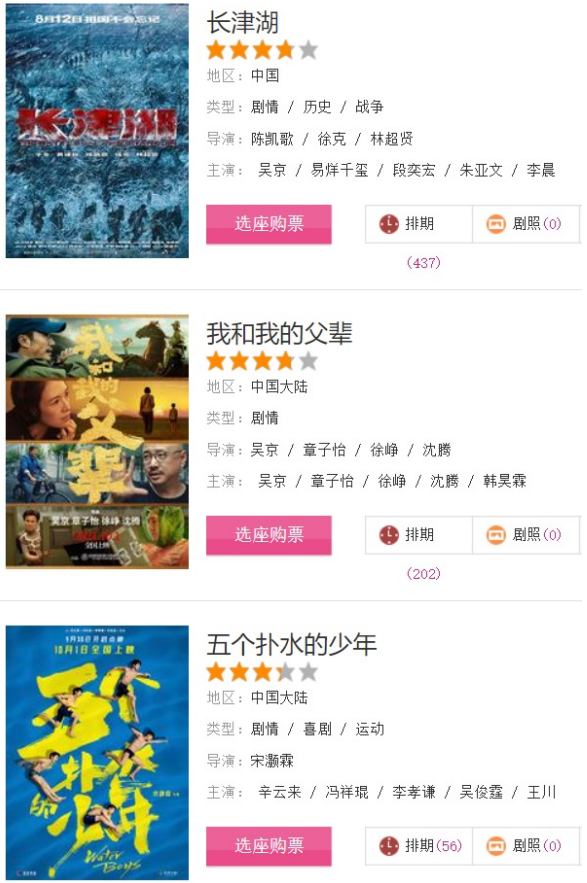
```
public class Student {
    private String name;
    private int age;
}
```

```
public class Test {
    public static void main(String[] args){
        double score = 99.9;
        String name = "黑马程序员";
        System.out.println(name + ":" + score);
    }
}
```

目录

Contents

案例 面向对象综合案例-模仿电影信息展示



需求

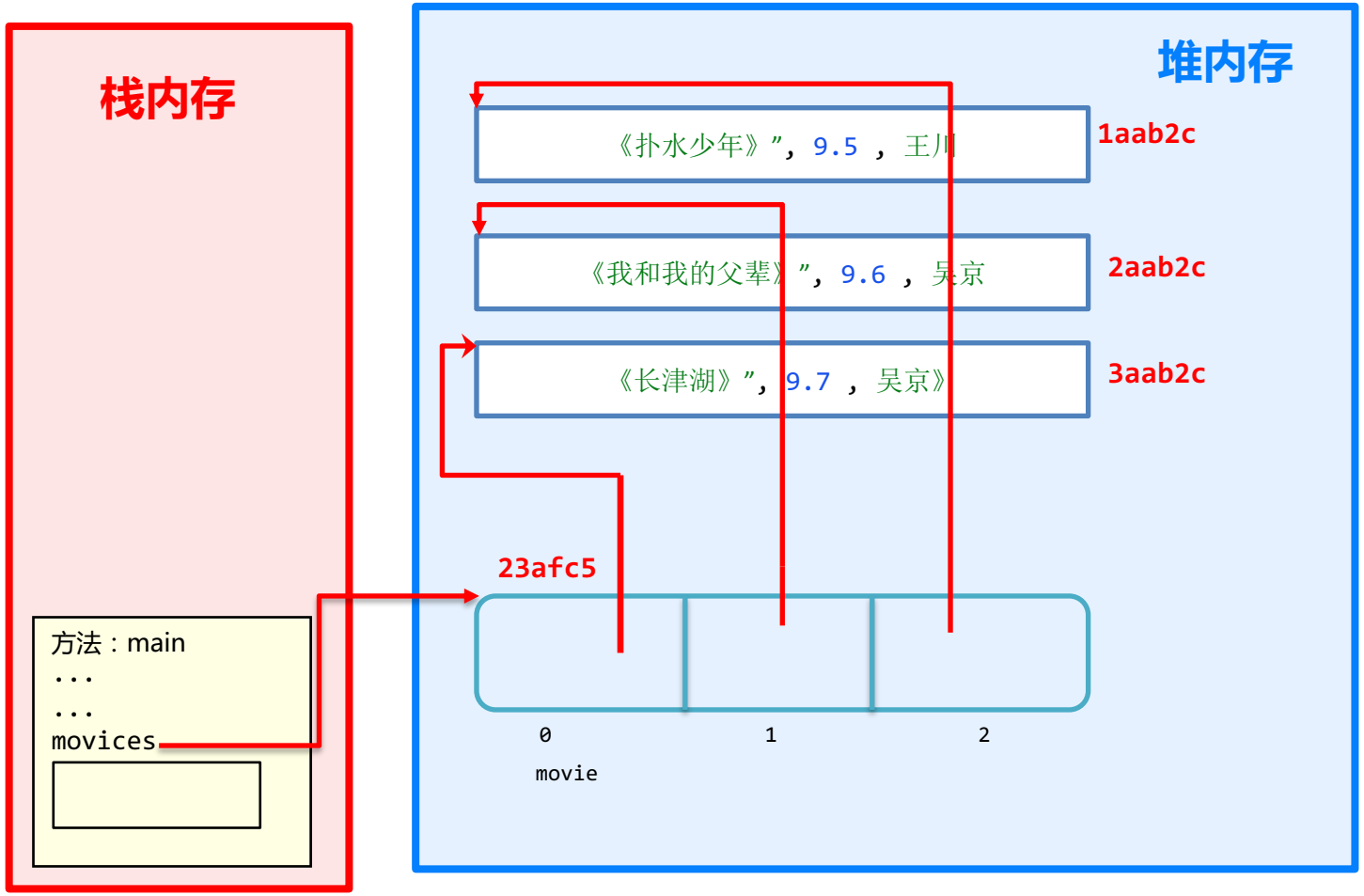
- 使用面向对象编程，模仿电影信息的展示。

分析

- 一部电影是一个Java对象，需要先设计电影类，再创建电影对象。
- 三部电影对象可以采用数组存储起来。
- 依次遍历数组中的每个电影对象，取出其信息进行展示。

```java
public class Movie {
    private String name;
    private double score;
    private String acotr;

    public Movie(String name, double score, String acotr) {
        this.name = name;
        this.score = score;
        this.acotr = acotr;
    }
    // ... getter + setter
}
```

```java
public class SystemDemo {
    public static void main(String[] args) {
        Movie[]  movies = new Movie[3];
        movies[0] = new Movie("《长津湖》", 9.7 ,  "吴京");
        movies[1] = new Movie("《我和我的父辈》", 9.6 ,  "吴京");
        movies[2] = new Movie("《扑水少年》", 9.5 ,  "王川 ");


        for (int i = 0; i < movies.length; i++ ) {
            Movie movie = movies[i];
            System.out.println("片名: " + movie.getName());
            System.out.println("评分: " + movie.getScore());
            System.out.println("主演: " + movie.getAcotr());
        }
    }
}
```



结论：数组中存储的元素并不是对象本身，而是对象的地址。

传智教育旗下高端IT教育品牌