面向对象编程（基础）

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

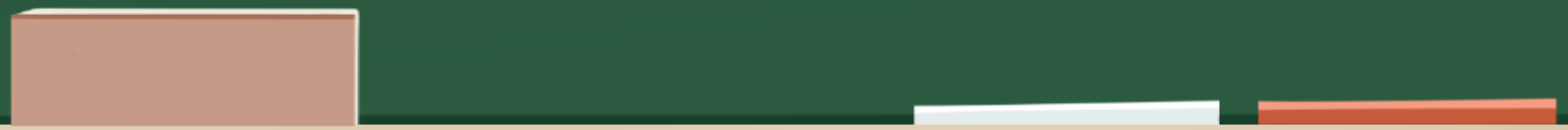
面向对象

面向对象介绍

- 并不是一个技术，而是一种编程指导思想。
- 把现实世界的具体事物全部看成一个一个的对象来解决问题。

为什么要用面向对象编程

- 生活中我们解决问题就是按照对象化的方式进行的。如果程序也能够按照生活的中的方式来解决问题，那么程序就更符合人类的思维习惯，代码看起来会更易理解、更简单。




对象


对象


对象


对象


我要买一部车！
对象
对象


对象

```
public class Test {
    public static void main(String[] args) {
            客户 customer = new 客户();
            销售 salesman = new 销售();
            钱 money= customer.掏钱();
            车 car =  salesman.收到(money);
            customer.获取(car);
    }
}
```

```
public class Test {
    public static void main(String[] args) {
            老师  teacher  = new 老师();
            学生 student = new 学生();
            试卷  exam = teacher.出题();
            student.考试(exam );
            teacher.批阅(exam);
    }
}
```

面向对象编程的指导思想、优点小结：

- 在程序中也把现实世界的具体事物全部看成一个一个的对象来解决问题。
- 按照面向对象编程来设计程序：程序代码符合人类思维习惯，更易理解、更简单。

获取已有对象并使用

```
public class Test {
    public static void main(String[] args) {
        // 1、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);

        // 2、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);
    }
}
```

面向对象的重点学习什么？

学习获取已有对象并使用

学习如何自己设计对象并使用

目录

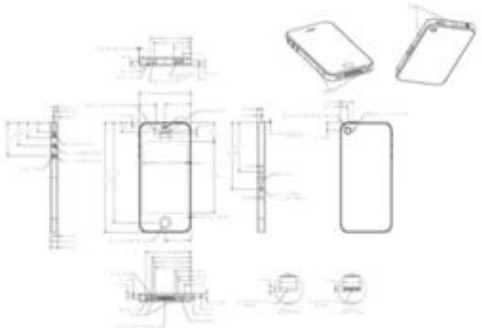
Contents

在Java中，必须先设计类，才能获得对象。

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。




设计图


类

```
public class Car {
        ...
        ...

}
```

new Car()

new Car()

如何定义类

```
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 （后面学习）
        4、代码块 （后面学习）
        5、内部类 （后面学习）
}
```

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

    }
    public void run(){

    }
}
```

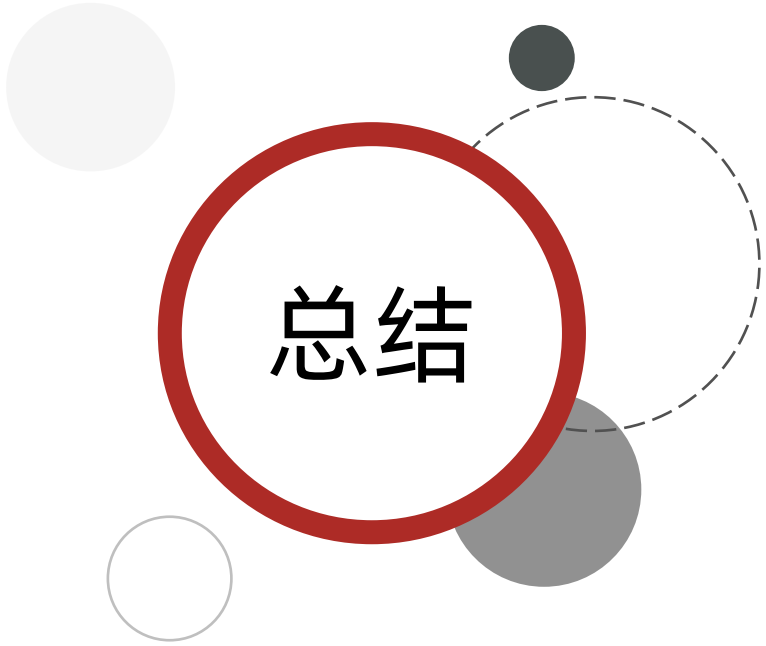
如何得到类的对象

```
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．如何得到对象？

```
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）
}
```

```
类名 对象名 = new 类名();
```

3．拿到对象后能做什么？

- **对象.成员变量；**
- **对象.成员方法(...)**

目录

Contents

定义类的补充注意事项

```
public class 类名{
        1、成员变量（代表属性）
        2、成员方法（代表行为）
}
```

```
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。
- 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，且只能一个类是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。

对象的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。**

目录

Contents

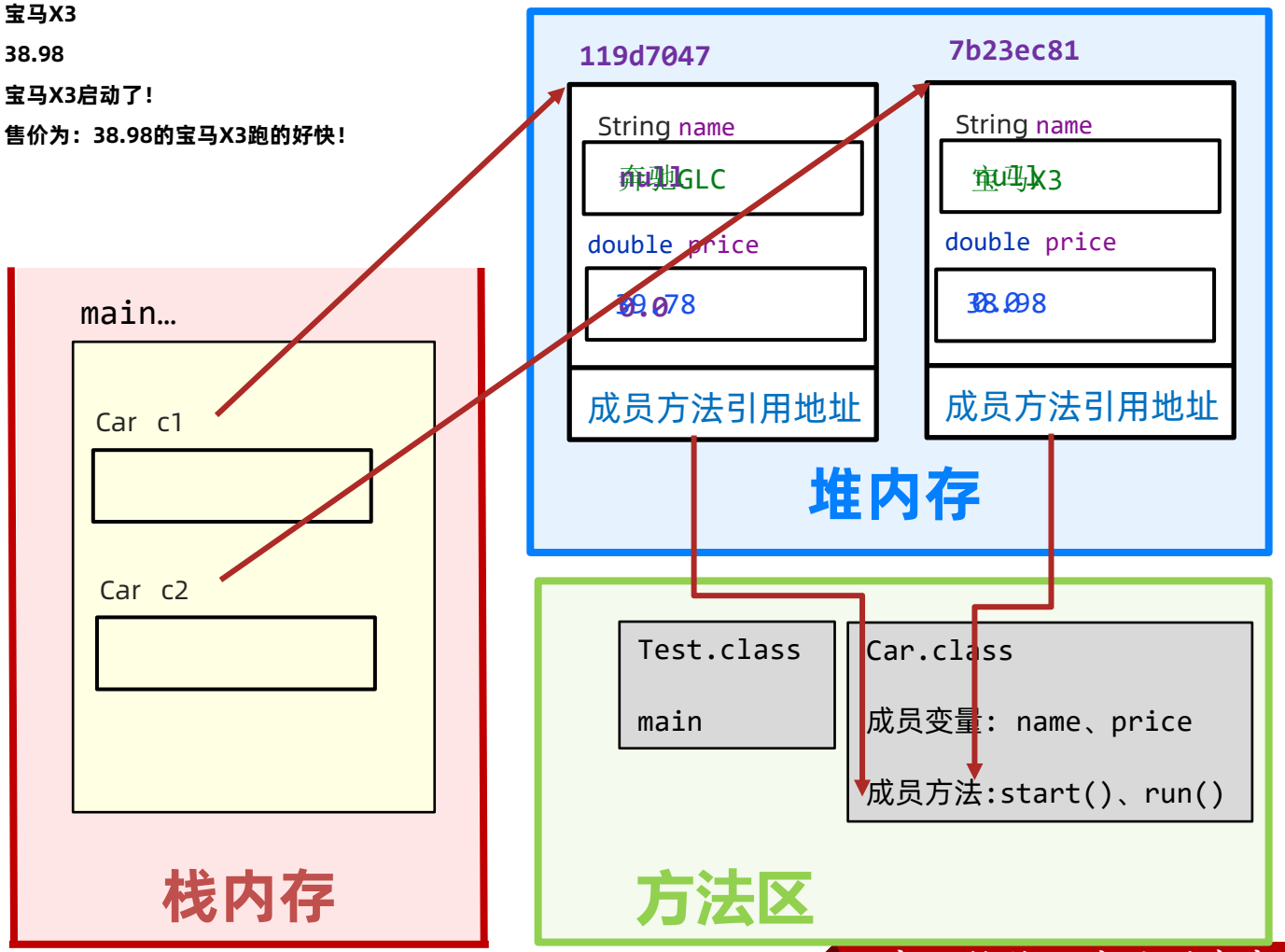
两个对象内存图

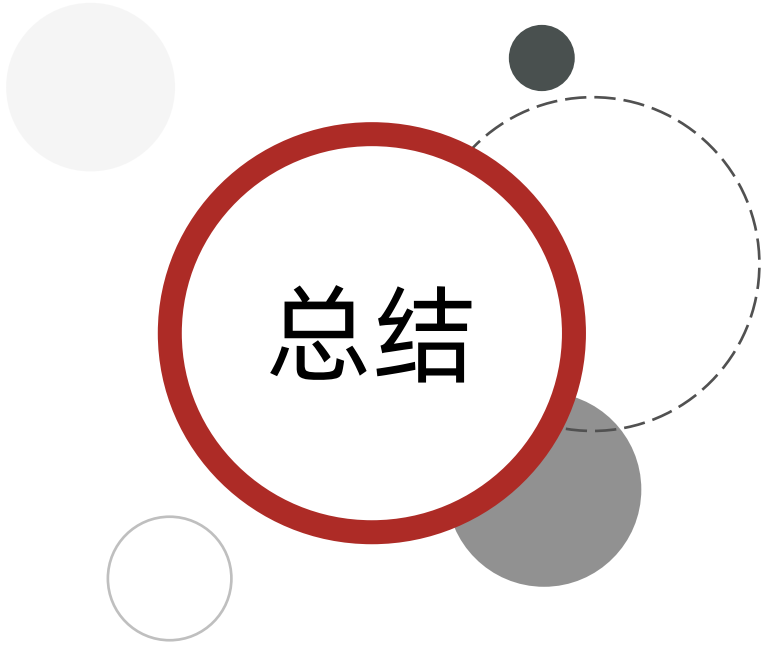
```
public class Car {
    // 成员变量(属性)
    String name;
    double price;
    // 方法(行为)
    public void start(){
        System.out.println(name+"启动了！");
    }
    public void run(){
        System.out.println("售价为：" + price +"的" + name+"跑的快！");
    }
}
```

```
public class Test {
    public static void main(String[] args) {
        Car c1 = new Car();
        c1.name = "奔驰GLC";
        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X3";
        c2.price = 38.98;
        System.out.println(c2.name);
        System.out.println(c2.price);
        c2.start();
        c2.run();
    }
}
```

```
奔驰GLC
39.78
奔驰GLC启动了！
售价为：39.78的奔驰GLC跑的好快！
宝马X3
38.98
宝马X3启动了！
售价为：38.98的宝马X3跑的好快！
```





1.对象放在哪个位置？

- **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();
    }
}
```

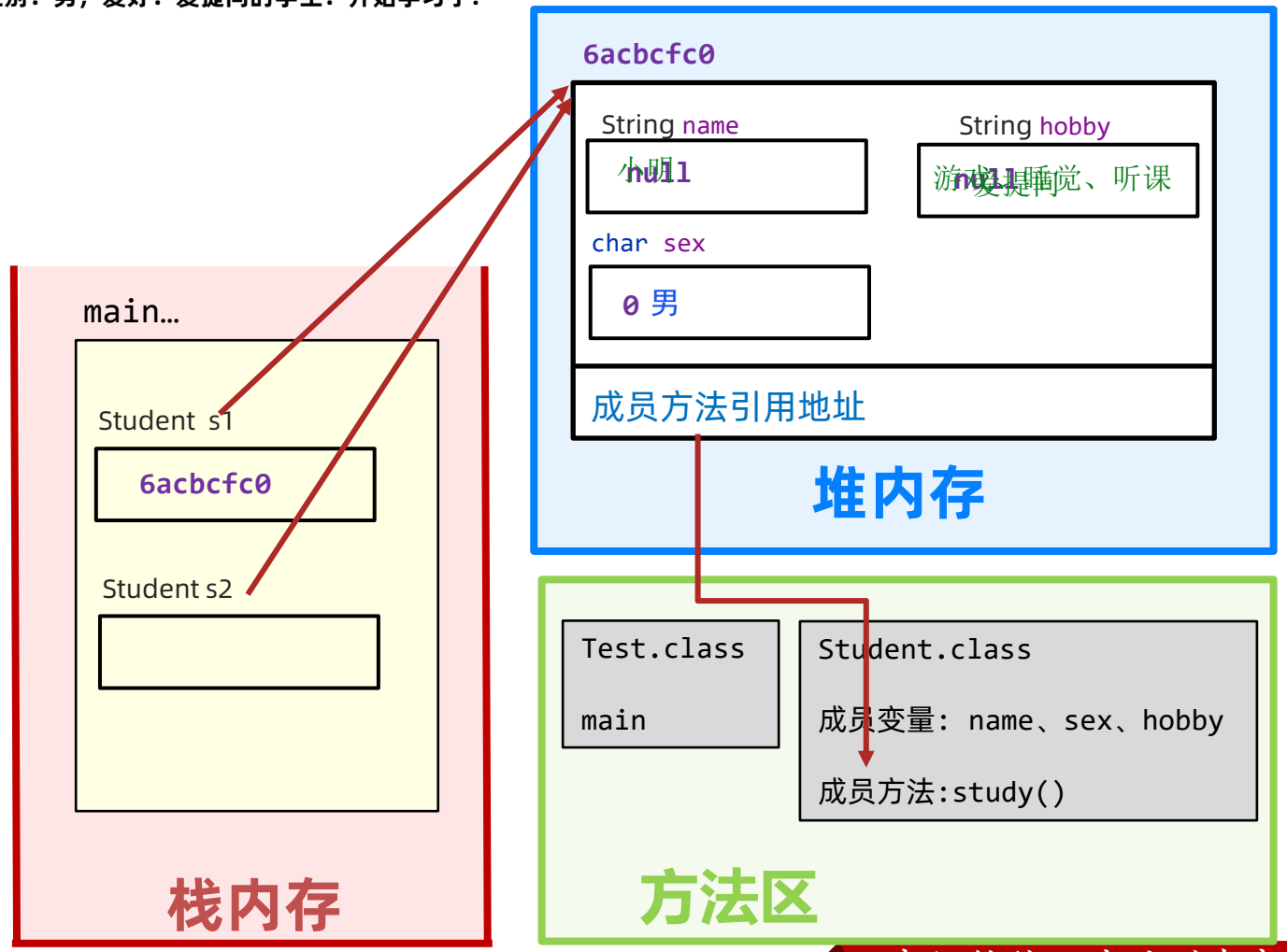
名称：小明，性别：男，爱好：游戏、睡觉、听课的学生：开始学习了！

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**类对象**或**数组对象**，没有被任何变量引用（指向）时，就会被判定为内存中的**“垃圾”**。
- Java存在自动垃圾回收器，会定期进行清理。

目录

Contents


对象


对象


对象


对象

案例

考试系统模拟

需求

- 使用面向对象编程模拟考试系统，可以实现学生考试，老师批阅试卷操作并打分出来。

分析

- 设计老师类（属性：名称、年龄、性别、行为：出题、批阅试卷）
- 设计试卷类（属性：模拟包含学生答案，正确答案即可）
- 设计学生类 （属性：学生名称、班级名称、行为：答卷）

目录

Contents

案例

考试系统模拟-功能实现

需求

- 实现一个老师，分发三张试卷，给3个学生进行考试，最终输出每个学生的考试结果。

分析

- 创建一个老师对象
- 调用老师对象的出题方法，依次得到3份试卷对象
- 创建3个学生对象
- 调用3个学生对象的考试方法并同时传入试卷对象
- 调用老师对象的批阅方法并传入试卷对象，输出考试的结果

目录

Contents


我要买一部车！
对象
对象


对象

案例

购车系统模拟-设计对象

需求

- 使用面向对象编程模拟购车，可以实现客户对象掏钱，销售对象交车功能。

分析

- 设计客户类（属性：名称、性别、电话　　行为：掏钱、取车）
- 设计钱类（余额、支付者）
- 设计销售类（属性：名称、性别、电话　　行为：卖车）
- 设计汽车类 （属性：名称、价格）

目录

Contents

案例

购车系统模拟-功能实现

需求

- 实现一个客户掏钱给销售，买一部车。

分析

- 创建一个客户对象。
- 调用客户对象的掏钱功能，得到钱对象。
- 创建一个销售对象。
- 调用销售对象的卖车功能同时注入钱对象，得到返回的车对象。
- 调用客户对象的取车功能同时注入车对象。

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完成的。

Get2：能够掌握为对象属性赋值的其他写法。

构造器的作用

- 用于初始化一个类的对象，并返回对象的地址。

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

构造器的定义格式

```
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, String b){
          ...
  }
}
```

初始化对象的格式

```
类型 变量名称 = new 构造器;
```

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

构造器的分类

- 无参数构造器（默认存在的）：初始化的对象时，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，同时可以为对象进行赋值。

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){

  }
}
```

总结

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：初始化的对象时，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

- 作用：出现在成员方法、构造器中代表当前对象的地址，用于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 b){
        name = n;
        price = b;
    }
}
```

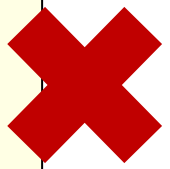

```
public class Car {
    String name;
    double price;

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

this出现在成员方法中的用法

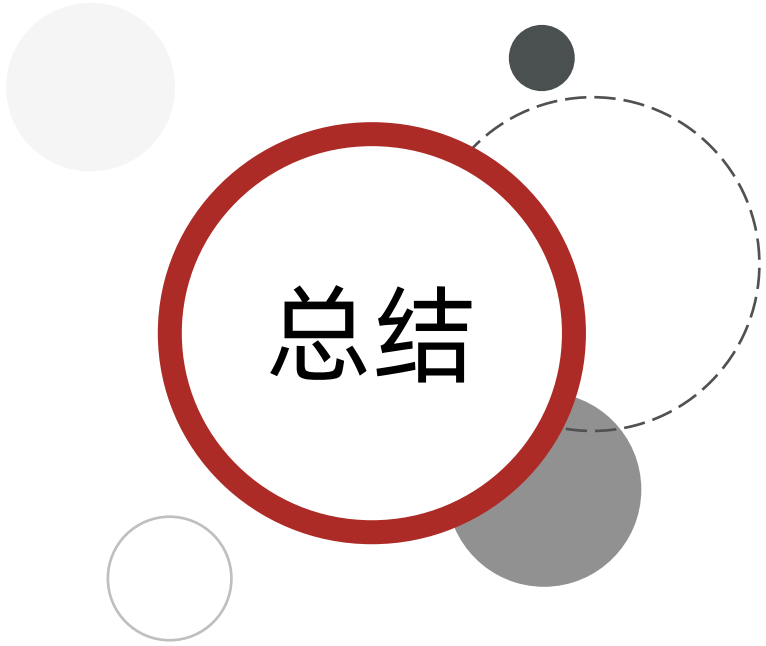
```
public class Car {
    String name;
    double price;

    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关键字的作用？

- **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？

- **可以用于访问当前对象的成员变量**

目录

Contents

封装

- 面向对象的三大特征：**封装，继承，多态**。
- 什么是封装？ 隐藏实现细节，暴露出合适的访问方式。（合理隐藏、合理暴露）

为什么要用封装？

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

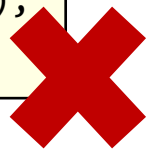
```
Student s = new Student();
s.age = -23;
```

封装的实现步骤

- 一般对成员变量使用private(私有)关键字修饰进行隐藏，private修饰后该成员变量就只能在当前类中访问。
- 提供public修饰的公开的getter、setter方法暴露其取值和赋值。

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

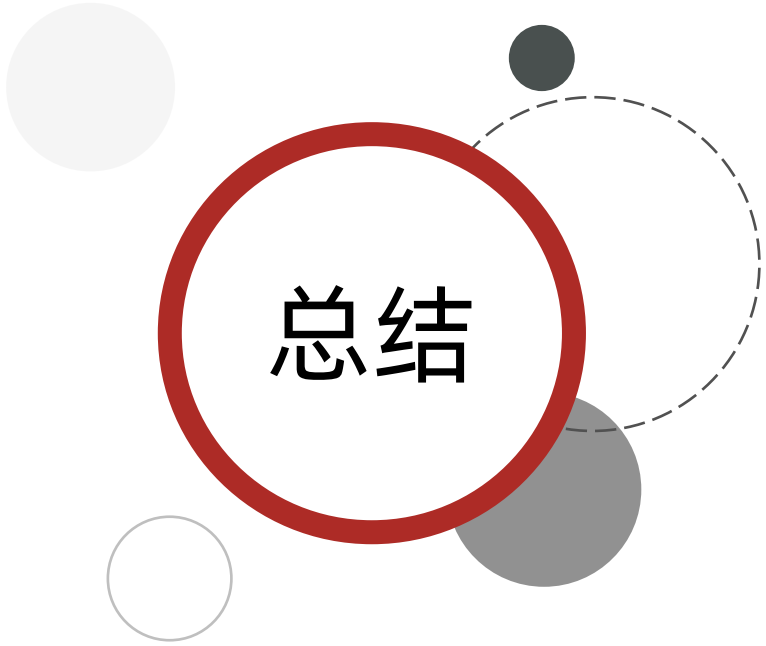
```
Student s = new Student();
s.age;
```



```
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");
        }
    }
}
```

封装的好处小结

- 加强了程序代码的安全性。
- 适当的封装可以提升开发效率，同时可以让程序更容易理解与维护。



1.封装是什么，一般封装怎么体现出来？

- 面向对象的三大特征之一，合理隐藏，合理暴露。
- 一般会把成员变量使用private隐藏起来。
- 通过getter和setter方法暴露其访问。

2.封装有什么好处？

- 加强了程序代码的安全性。
- 适当的封装可以提升开发效率，同时可以让程序更容易理解与维护。

目录

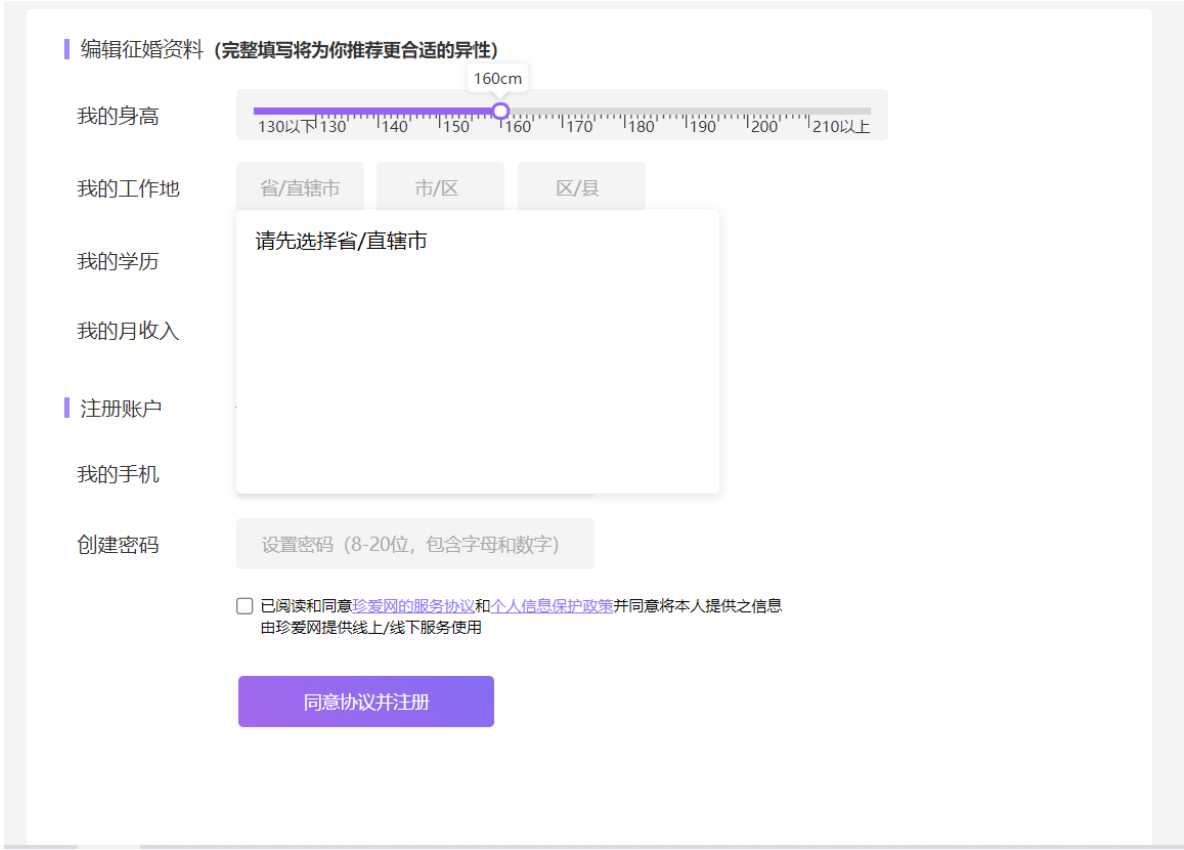
Contents

JavaBean

- 也可以理解成实体类，其对象可以用于在程序中封装数据。

标准JavaBean须满足如下要求：

- 成员变量使用 **private** 修饰。
- 提供每一个成员变量对应的 **setXxx() / getXxx()。**
- 必须提供一个**无参构造器。**



目录

Contents


我要买一部车！
对象
对象


对象

案例 购车系统模拟

```
public class Car {
    String name;   // 名称
    double price;  // 售价
}
```

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

    public String getName() {
        return name;
    }

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

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }
}
```

目录

Contents

成员变量和局部变量的区别

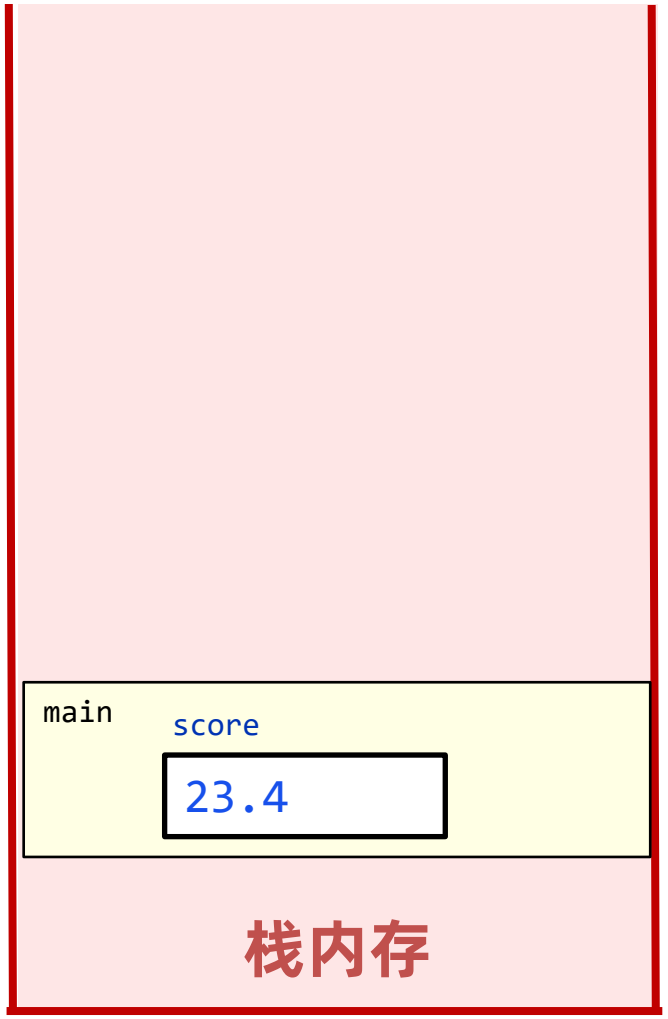
区别	成员变量	局部变量
类中位置不同	类中，方法外	常见于方法中
初始化值不同	有默认初始化值	没有，使用之前需要完成赋值
内存位置不同	堆内存	栈内存
生命周期不同	随着对象的创建而存在，随着对象的消失而消失	随着方法的调用而存在，随着方法的运行结束而消失
作用域		在所归属的大括号中

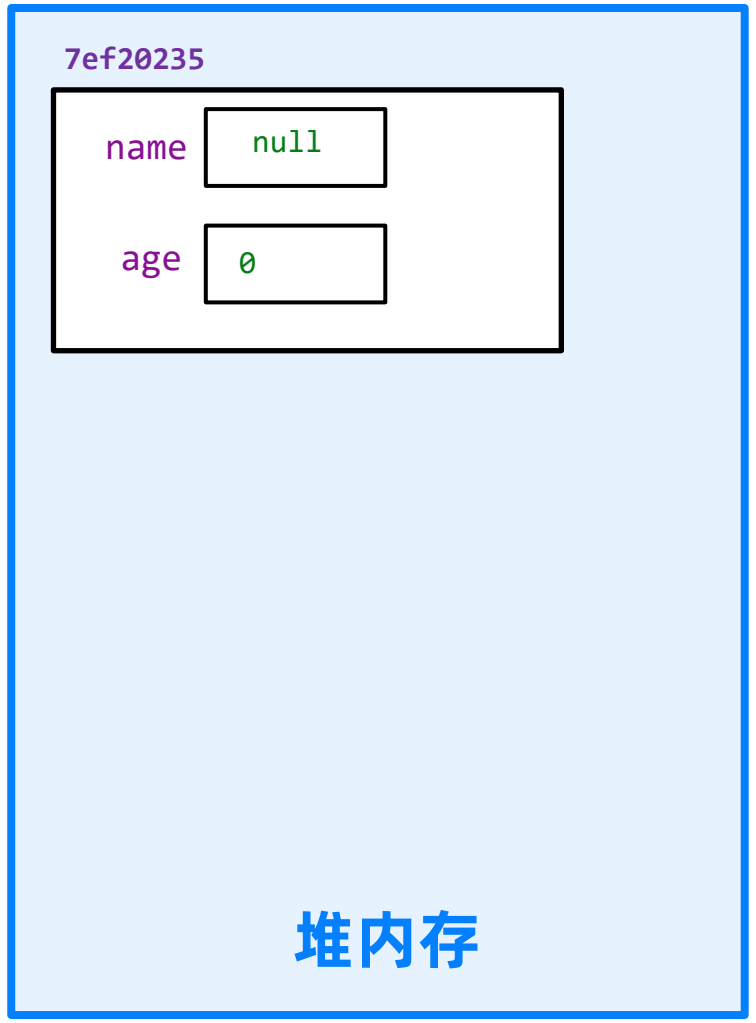
成员变量和局部变量的区别

```
public class Student {

    private String name;
    private int age;
}
```

```
public class Test {
    public static void main(String[] args){
        double score = 23.4;
        new Student();
    }
}
```





传智教育旗下高端IT教育品牌