面向对象

啥是面向对象编程？

- Java面向对象编程（Object-Oriented Programming，简称OOP）是一种编程范式，它将**现实世界中的实体**抽象为**对象**，并通过对象之间的交互来设计和构建软件系统。

面向对象编程的例子

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

面向对象学习什么?

学习获取已有对象并使用

学习自己设计对象并使用

面向对象的语法

```
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

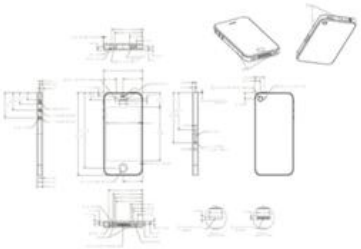
Contents

- 设计对象并使用
  - 类和对象
  - 定义类的几个补充注意事项
- 对象内存图
- 构造方法
- **this**关键字
- 封装
- 标准 **JavaBean**
- 补充知识：成员变量、局部变量区别
- 面向对象综合案例

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）

        ……

}
```

```
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

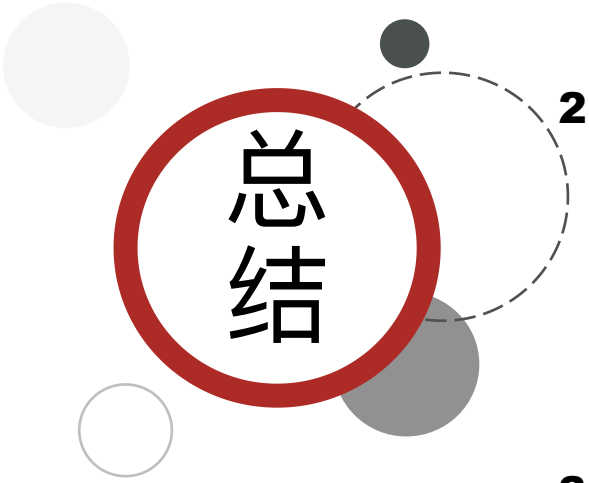
```
类名 对象名 = new 类名();

Car c = new Car();

Car c2 = new Car();
```

如何使用对象

- 访问属性: 对象名.成员变量
- c.name
- c2.price
- 访问行为: 对象名.方法名(…)
- c.start()
- c2.run()

```java
package com.mycar;
public class Car {
    //属性-成员变量
    String name;
    double price;
    //行为-成员方法
    public void start(){
        System.out.println(name+"启动了");
    }
    public void stop(){
        System.out.println(name+"停车了");
    }
}
```

```java
package com.mycar;
public class CarTest {
    public static void main(String[] args) {
        //类名 对象名 = new 类名（）；
        Car c = new Car();
        c.name ="奔驰";
        c.price = 39.98;
        System.out.println(c.name);
        System.out.println(c.price);
        c.start();
        c.stop();

        Car c2 = new Car();
        c2.name ="宝马";
        c2.price = 29.98;
        System.out.println(c2.name);
        System.out.println(c2.price);
        c2.start();
        c2.stop();
    }
}
```



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(...)**

目录

Contents

定义类的补充注意事项

- JavaBean 是一种 符合特定规范的 Java 类，通常用于 封装数据，而不是执行程序逻辑。它常用于表示某种“事物”或“实体”，比如：
  - 学生（Student）
  - 图书（Book）
  - 用户（User）
- Javabean类中，不写main方法

- 测试类是一个包含 main 方法的类，用于 运行程序 和 测试其他类（如 JavaBean）的功能。

- 我们可以在测试类中创建Javabean类的对象并进行赋值调用。

定义类的补充注意事项

```
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）
}
```

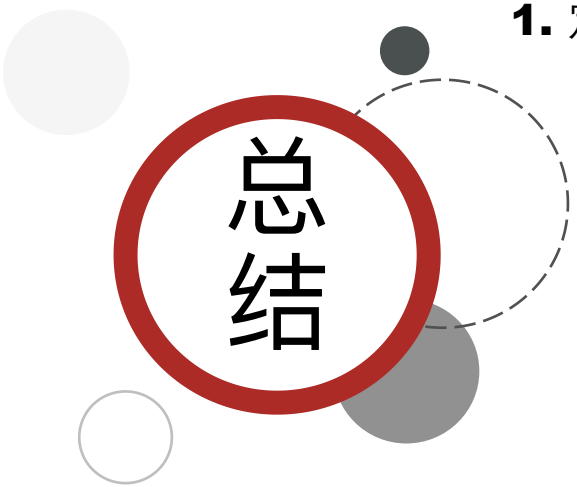
```
public class Student {
    // 属性 (成员变量)
    String name;
    double height;
    // 行为（方法）
    public void study(){

    }
    public void run(){

    }
}
```

- 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。

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**



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。


学生信息管理系统
查询信息 修改信息 修改管理员信息 退出系统
学生信息修改
学生信息修改
学号:
验证学号
姓名:
性别:
年龄:
学校:
确认修改
迎登陆!!!
-09-04 23:56:54

填空题 2分

设置

学生类中，需要写 [填空1] 个成员变量和 [填空2] 个成员方法。

作答

练习 练习时间

- 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。

主观题 10分

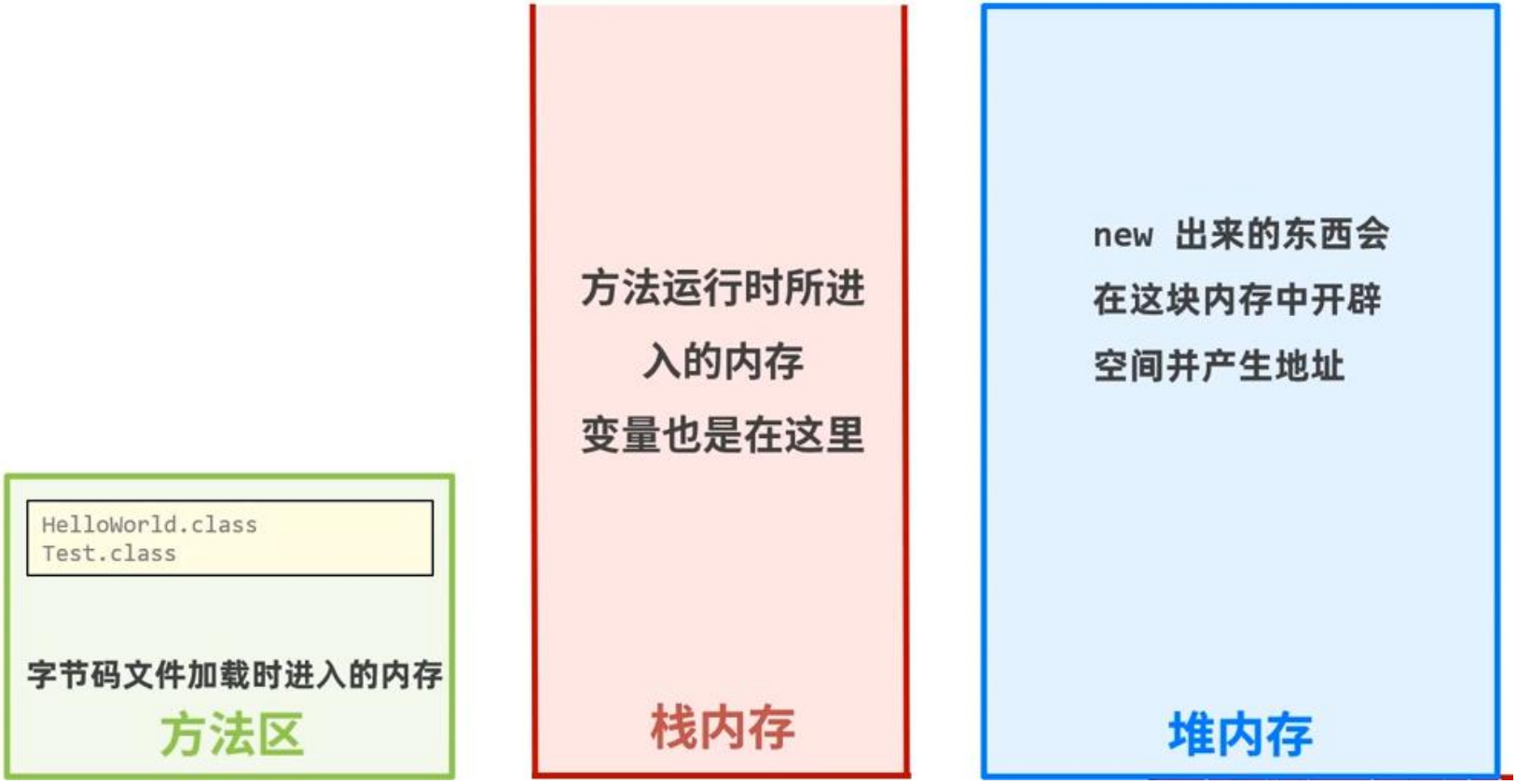
设置

- 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。

作答

- 设计对象并使用
- 对象在内存中的运行机制
  - 多个对象的内存图
  - 两个变量指向同一个对象内存图
- 构造方法
- **this**关键字
- 封装
- 标准 **JavaBean**
- 补充知识：成员变量、局部变量区别
- 面向对象综合案例

JAVA内存分配

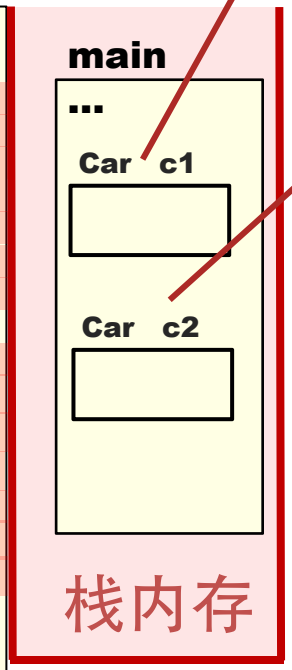

HelloWorld.class
Test.class
字节码文件加载时进入的内存
方法区
方法运行时所进
入的内存
变量也是在这里
栈内存
new 出来的东西会
在这块内存中开辟
空间并产生地址
堆内存

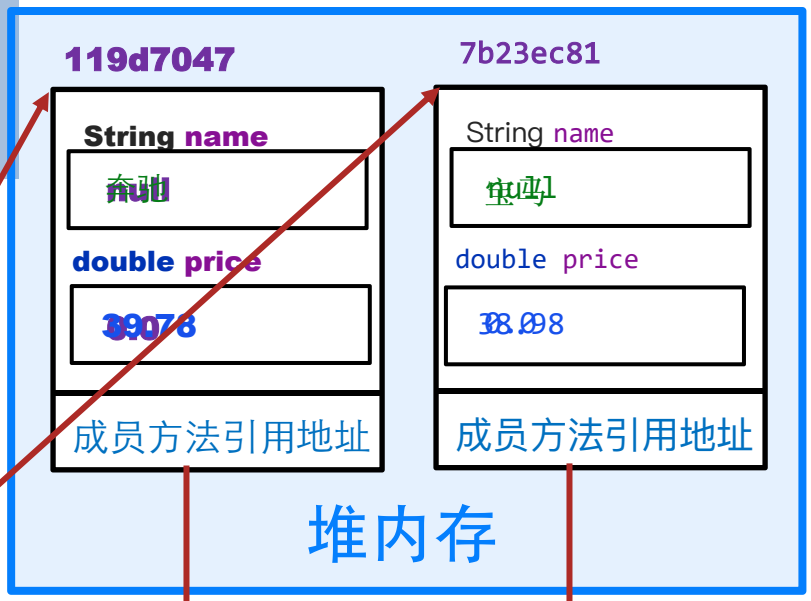
两个对象内存图

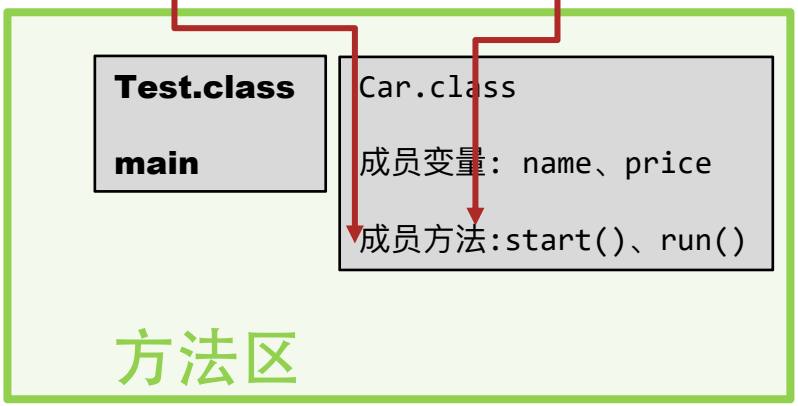
```
public class Car {
    // 成员变量(属性)
    String name;
    double price;
    // 方法(行为)
    public void start(){
        System.out.println(name+“启动了！”);
    }
    public void run(){
        System.out.println(“价格是：" + price +"的" + name+"跑的快！");
    }
}
```

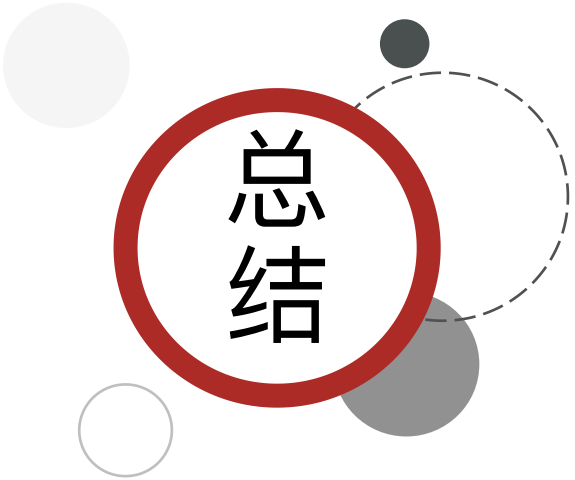
```
public class Test {
    public static void main(String[] args) {
        Car c1 = new Car();
        c1.name = "奔驰";
        c1.price = 39.78;
        System.out.println(c1.name);
        System.out.println(c1.price);
        c1.start();
        c1.run();

        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。

- 设计对象并使用
- 对象在内存中的运行机制
  - 多个对象的内存图
  - 两个变量指向同一个对象内存图
- 构造方法
- **this**关键字
- 封装
- 标准 **JavaBean**
- 补充知识：成员变量、局部变量区别
- 面向对象综合案例

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();
    }
}
```

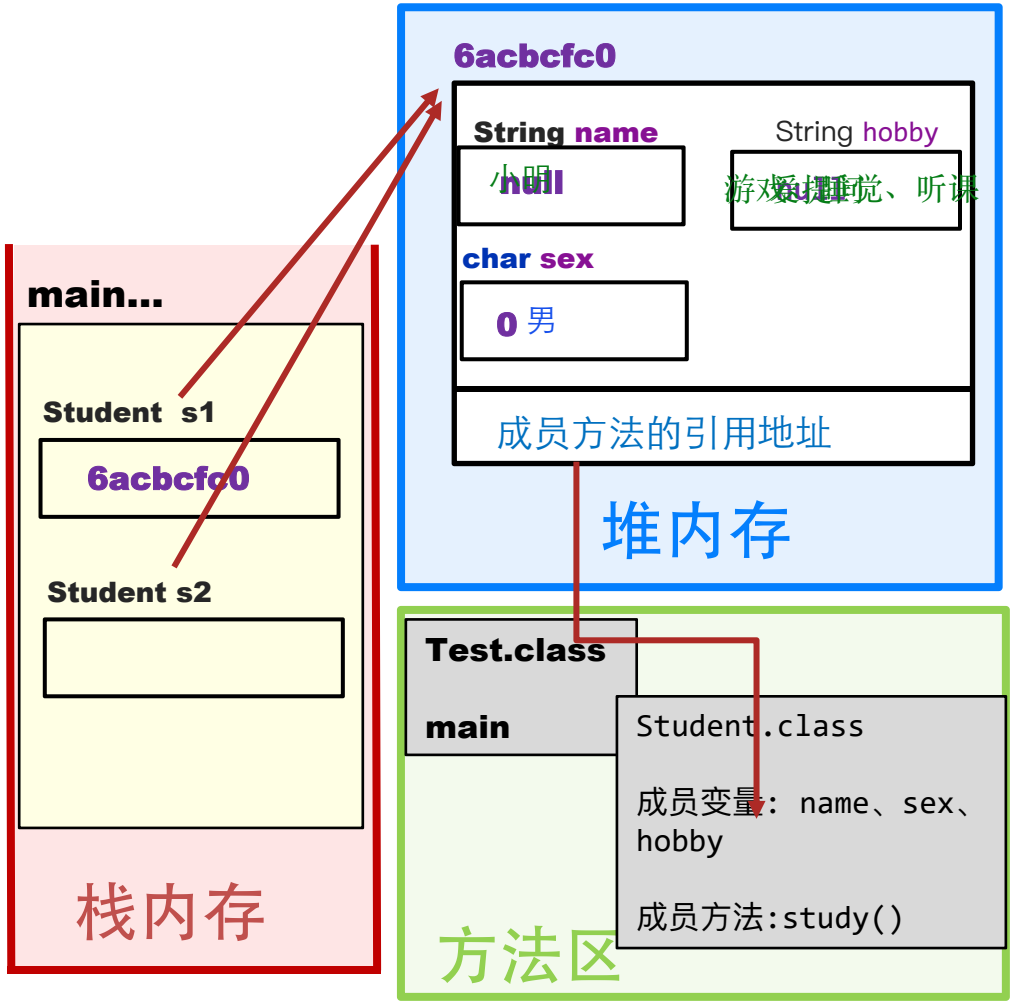
名称：小明，性别：男，爱好：游戏、睡觉、听课的学生：开始学习了！

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”。

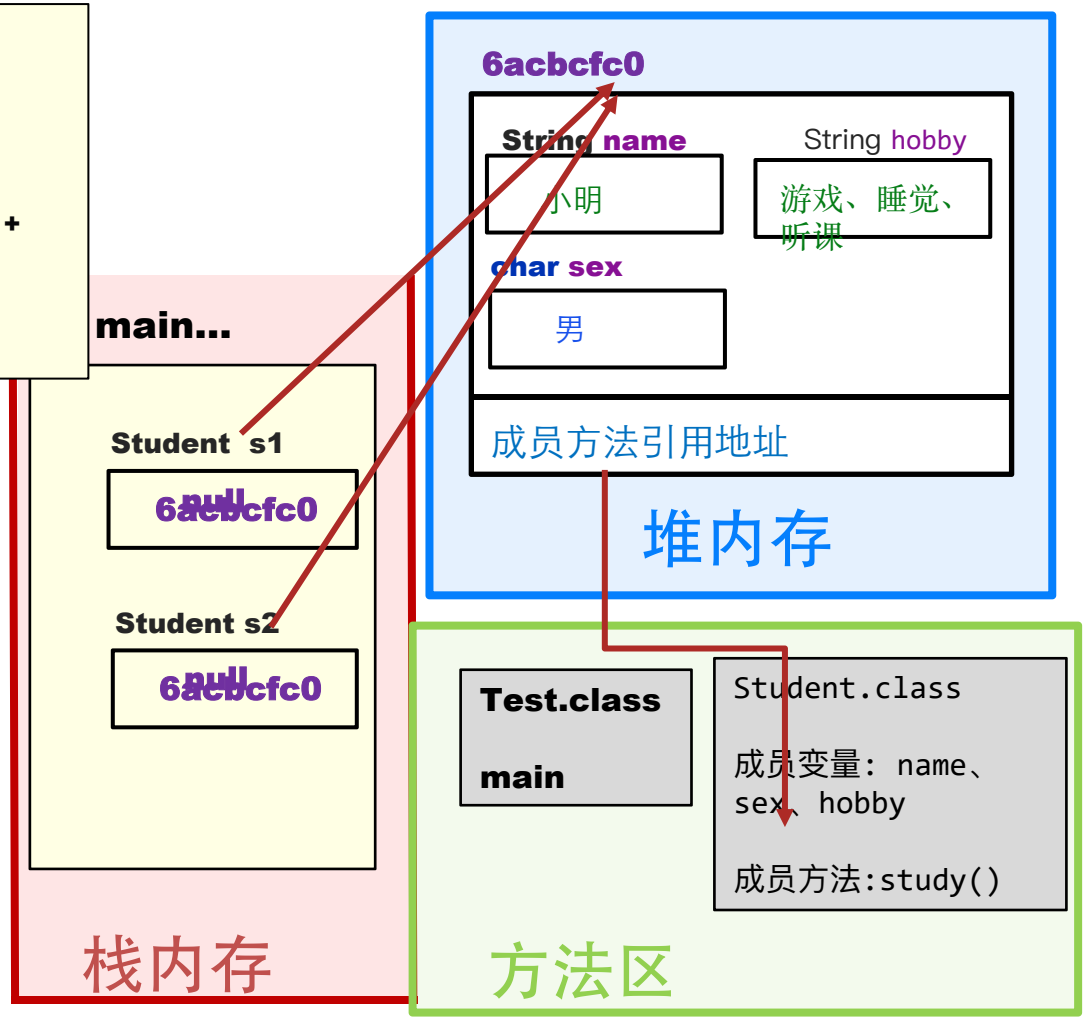
两个变量指向同一个对象内存图

Java存在自动垃圾回收器，会定期进行清理。

```
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;
```



- 设计对象并使用
- 对象内存图
- 构造方法
- **this**关键字
- 封装
- 标准 **JavaBean**
- 补充知识：成员变量、局部变量区别
- 面向对象综合案例

构造方法

- 在声明了一个对象引用后，要调用new运算符为新对象分配空间，其实就是要调用***构造方法***（constructor） 。
- 在Java中，使用构造方法是**生成实例对象**的**唯一方法**。
- 构造方法是一类比较**特殊的方法**，是通过new运算符调用，用来创建对象并进行初始化的方法。

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

类名 引用变量名 = new 构造方法(…) ;

构造方法

1．构造方法是一类特殊的方法：

- 方法名和类名相同，**大小写也要一致**；
- 没有返回值类型（连void也不没有）；
- 在创建对象实例时由new运算符自动调用；
- 同时为了创建对象实例时的方便，一个类可以有多个具有不同参数列表的构造方法，即**构造方法可以重载**。

构造方法的作用

- 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();
```

构造方法的格式

```
修饰符 类名(形参列表){
    ...
}
```

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

调用构造方法得到对象的格式

```
类 变量名称 = new 构造方法;
```

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

```
Car c1 = new Car("奔驰", 39.8);
```

构造方法的分类和作用

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

构造方法的注意事项

- 任何类定义出来，默认就自带了无参数构造方法，写不写都有。
- 一旦定义了有参数构造方法，那么无参数构造方法就没有了，如果还想用无参数构造方法，此时就需要自己手写一个无参数构造方法了。
- 推荐：无论是否使用，都手动书写无参数构造方法和带全部参数的构造方法。

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

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

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

  }
}
```

填空题 3分

设置

三个类的定义如下，请问类A有 [填空1] 个构造方法，类B有 [填空2] 个构造方法，类C有 [填空3] 个构造方法。

```
class A{
    int i;
}
```

```
class B{
    int i;
    public B(){
     }
}
```

```
class C{
    int i;
    public C(int i){
        this.i = i;
     }
}
```

作答

构造方法

```
class A{
    int i;
}
```

有1个构造方法
缺省的构造方法

```
class B{
    int i;
    public B(){
     }
}
```

有1个构造方法
自定义构造方法

```
class C{
    int i;
    public C(int i){
        this.i = i;
     }
}
```

有1个构造方法
自定义构造方法

总结

1.构造方法的作用?

- 初始化类的对象，并返回对象的地址。

2.构造方法有几种，各自的作用是什么?

- 无参数构造方法：初始化对象时，成员变量的数据均采用默认值。
- 有参数构造方法：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.构造方法有哪些注意事项?

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

- 设计对象并使用
- 对象内存图
- 构造方法
- **this关键字**
- 封装
- 标准 **JavaBean**
- 补充知识：成员变量、局部变量区别
- 面向对象综合案例

成员变量和局部变量

```
public class Student {
    String name;
    int age;
    public void friend(String n) {
        System.out.println(name+"的朋友是"+n);
    }
}
```

成员变量

局部变量

- 在方法外，类里面定义的变量称为**成员变量。**成员变量在**整个类内**有效。
- 在方法内定义的变量和方法的参数称为**局部变量。**局部变量仅在**方法内**有效。
- 若局部变量和成员变量同名，则成员变量被屏蔽

成员变量和局部变量

- 若局部变量和成员变量同名，则成员变量被屏蔽

```
public class Student {
    String name;
    int age;
    public void friend(String name) {
        System.out.println(name+"的朋友是"+name);
    }
}
```

成员变量

局部变量

```
public class StudentTest {
    public static void main(String[] args) {
        Student xiaoming = new Student();
        xiaoming.name = "小明";
        xiaoming.friend("小红");
    }
}
```

填空题 1分

设置

成员变量和局部变量

```
public class Student {
    String name;
    int age;
    public void friend(String name) {
        System.out.println(name+"的朋友是"+name);
    }
}
```

```
public class StudentTest {
    public static void main(String[] args) {
        Student xiaoming = new Student();
        xiaoming.name = "小明";
        xiaoming.friend("小红");
    }
}
```

[填空1]

作答

成员变量和局部变量

- 若局部变量和成员变量同名，则成员变量被屏蔽

```
public class Student {
    String name;
    int age;
    public void friend(String name) {
        System.out.println(name+"的朋友是"+name);
    }
}
```

成员变量

局部变量

```
public class StudentTest {
    public static void main(String[] args) {
        Student xiaoming = new Student();
        xiaoming.name = "小明";
        xiaoming.friend("小红");
    }
}
```

这里输出：小红的朋友是小红。而我们想要的是：小明的朋友是小红

this关键字是什么

- 可以出现在**构造方法**、**方法**中
- 代表**当前对象**的地址。

```
public class Car {
    public Car() {
        System.out.println("this在构造方法中：" + 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);
    }
}
```

```
"C:\Program Files\Java\jdk-2
this在构造方法中：Car@2f4d3709
this在方法中：Car@2f4d3709
Car@2f4d3709
```

this关键字是什么

- 可以出现在**构造方法**、**方法**中
- 代表**当前对象**的地址。

```
public class Student {
    String name;
    int age;
    public void friend(String name) {
        System.out.println(this.name+"的朋友是"+name);
    }
}
```

```
public class StudentTest {
    public static void main(String[] args) {
        Student xiaoming = new Student();
        xiaoming.name = "小明";
        xiaoming.friend("小红");
    }
}
```

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出现在有参数构造方法中的用法

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

n、p变量命名不规范

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

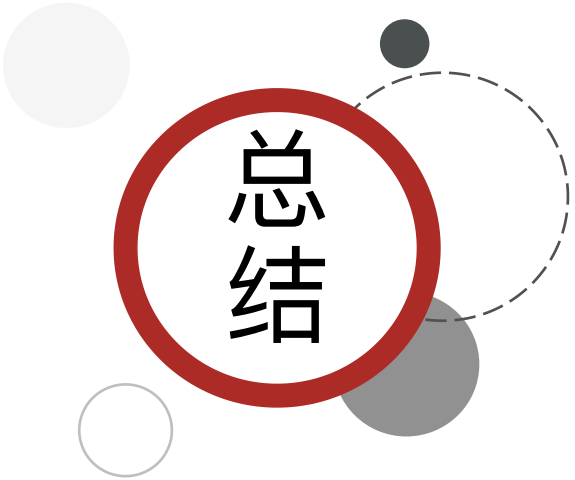
this关键字的作用

- 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?

   - 出现在构造方法和成员方法中，代表当前对象的地址。

2. **this**关键字在构造方法中、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。

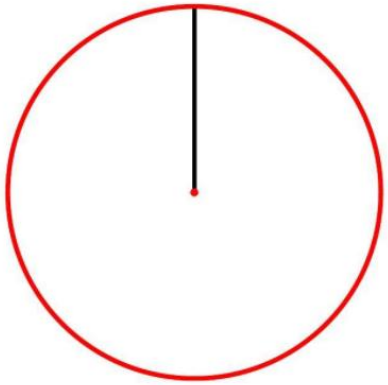
```
public class People {
    名称
    年龄
    吃饭
    睡觉
}
```

封装

- 封装：告诉我们，如何正确设计对象的属性和方法。

需求

- 人画圆，针对这个需求进行面向对象设计。



人画圆

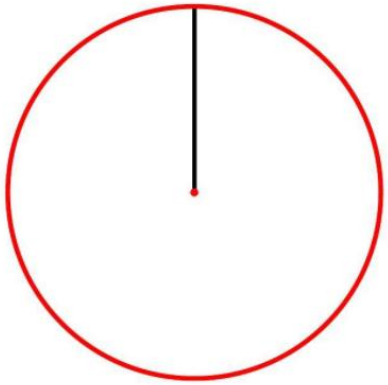
```
public class People {

    ...

}
```

```
public class Circle {



}
```

```
public void draw(){

}
```

封装

- 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。

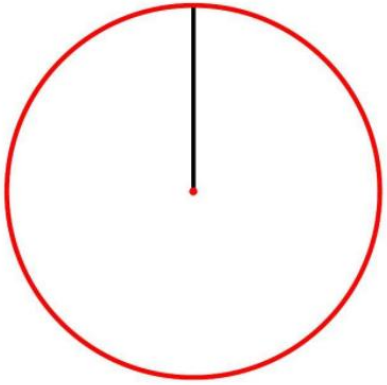


人画圆

```
public class Circle {
    double radius; // 半径
    public void draw(){
        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

- 门的状态数据有：开和关。那么开门和关门的方法就是属于门这个对象的。

理解封装思想有啥好处?

理解封装思想有啥好处?

String

代表字符串对象

拥有操作字符串的很多方法

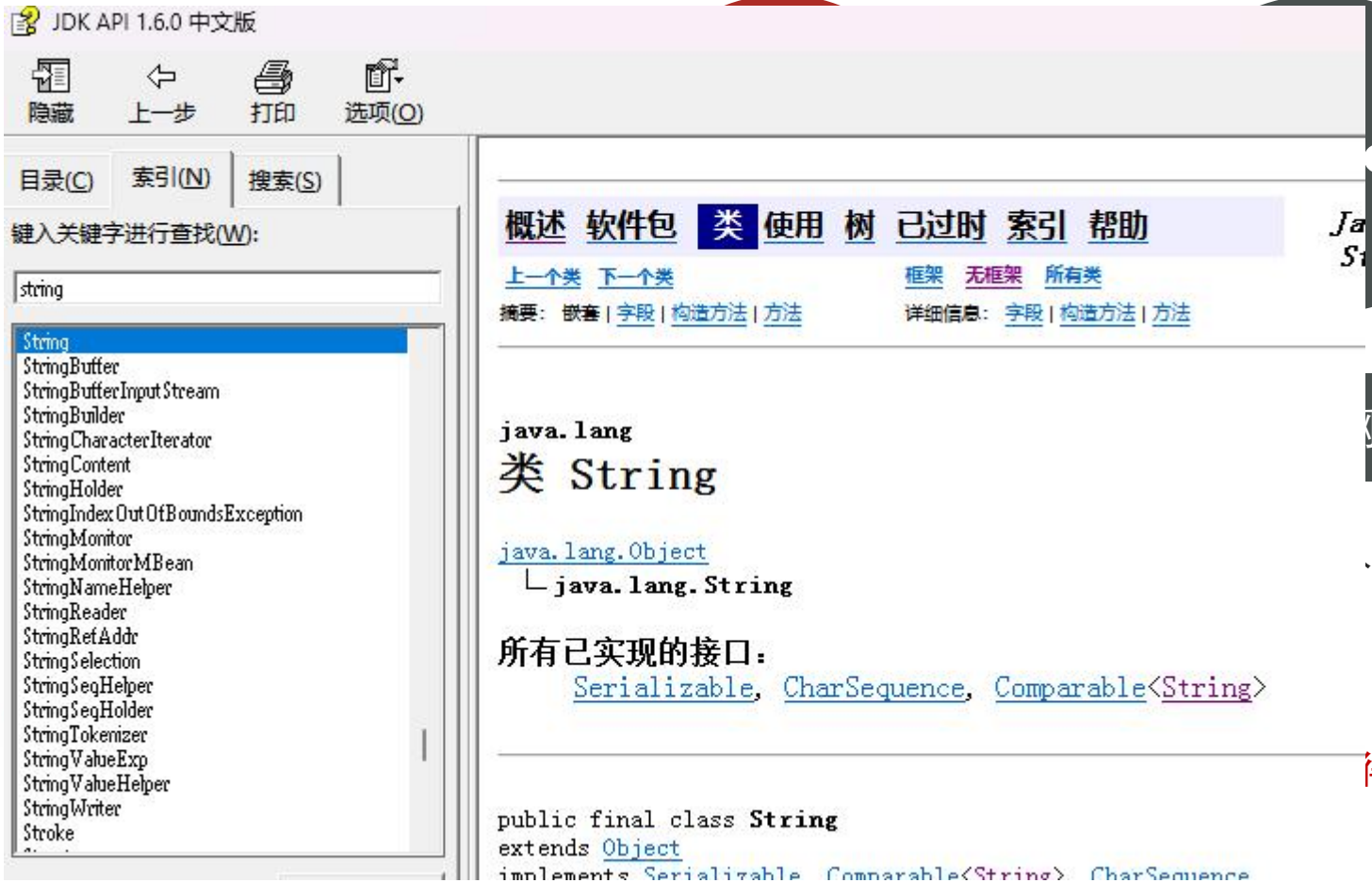
Socket

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。

理解封装思想有啥好处?



et

络连接

., 发消息,

得很简单。

理解封装思想有啥好处?

```
public class Test1 {
    public static void main(String[] args) {
        String s ="abcdef";
        int len=s.length();
        System.out.println(len);
        System.out.println(s.);

    }
}
```

toUpperCase()
toUpperCase(Locale locale)
length()
getBytes(StandardCharsets.UTF_8)
getBytes(String charsetName)
getBytes(Charset charset)
getBytes()
toLowerCase(Locale.ROOT)
toLowerCase(Locale locale)
toLowerCase()
toUpperCase(Locale.ROOT)
charAt(int index)

Press Enter to insert, Tab to replace Next Tip

理解封装思想有啥好处?

String

代表字符串对象

拥有操作字符串的很多方法

Socket

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。

总结

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（**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 Student {
    private 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 Test {
    public static void
main(String[] args) {
        Student s = new Student();
        s.age=20;
    }
}
```

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（**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 a) {
        if (a >= 18 && a <= 100) {
            age = a;
        } else {
            System.out.println("请检查年龄数值");
        }
    }
}
```

练习 练习时间

- **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。**
- 把成员变量使用**private**隐藏起来，提供**public**修饰的**getter**和**setter**方法暴露其取值和赋值。

主观题 10分

设置

- 定义一个学生类，随便定义2个属性，2个行为。并创建1个学生对象，访问属性和行为。
- 把成员变量使用private隐藏起来，提供public修饰的getter和setter方法暴露其取值和赋值。

作答

```java
// 定义学生类
public class Student {
    // 定义属性
    private int age;

   // 外部访问属性 'age' 的getter方法
    public int getAge() {
        return this.age;
    }
     // 外部修改属性 'age' 的setter方法
    public void setAge(int age) {
        this.age = age;
    }

    // 行为1: 学习
    public void study() {
        System.out.println( " 在学习");
    }
}
```

```java
// 主函数（调用示例）
public class StudentTest {

public static void main(String[] args) {
    // 创建学生对象
    Student student1 = new Student();

    //赋值年龄
    student1.setAge(18);
    //调用年龄
    System.out.println(student1.getAge());

     //调用方法
    student1.study();

    }

}
```

总结

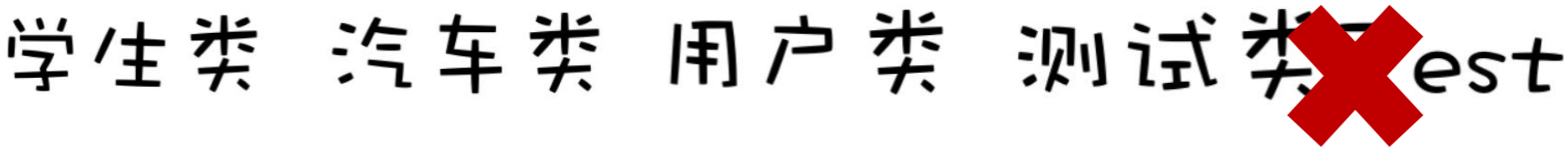
1. 如何进行更好的封装?

- 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。

- 设计对象并使用
- 对象的内存运行机制
- 构造方法
- **this**关键字
- 封装
- 标准 **JavaBean**
- 补充知识：成员变量、局部变量区别
- 面向对象综合案例

JavaBean

- 也可以称为实体类，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**无参构造方法；有参数构造方法是可写可不写的。**

- 设计对象并使用
- 对象内存图
- 构造方法
- **this**关键字
- 封装
- 标准 **JavaBean**
- 补充知识：成员变量、局部变量区别
- 面向对象综合案例

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	随着方法的调用而存在，随着方法的
作用域	整个类中	

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

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

成员变量和局部变量的区别

区别	成员变量	
类中位置不同		
初始化值不同		
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	整个类中	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 = "小明";
        System.out.println(name + ":" + score);
    }
}
```

- 设计对象并使用
- 对象内存图
- 构造方法
- **this**关键字
- 封装
- 标准 **JavaBean**
- 补充知识：成员变量、局部变量区别
- 面向对象综合案例

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.getActor());
      }
   }
}
```