

面向对象

Java Platform Standard Edition

郑春光

课程目标

CONTENTS

ITEMS **1** 什么是对象

ITEMS **2** 什么是类

ITEMS **3** 类的组成

ITEMS **4** 方法重载

ITEMS **5** 构造方法

ITEMS **6** this关键字

什么是程序



即时聊天



路况、导航



便捷支付



美颜、修图

- 程序是为了模拟现实世界，解决现实问题而使用计算机语言编写的指令集合。

- 世界是由什么组成的？
 - 有人说：“世界是由无数原子组成的”。
 - 有人说：“世界是由无数事物组成的”。
 - 有人说：“世界是由无数物体组成的”。
 - 有人说：“世界是由一切有生命的和一切没有生命的组成的”。
 - 有人说：“你、我、他、大家组成的”。
- 所有回答都很抽象，没有特别明确的答案。
- 在程序员的眼里，世界的组成最为明确：“世界是由无数个对象组成的”。

什么是对象

- 面向对象思想 (Object Oriented Programming) :
 - 一切客观存在的事物都是对象，万物皆对象。
 - 任何对象，一定具有自己的特征和行为。

对象

特征：称为属性，一般为名词，代表对象有什么。

行为：称为方法，一般为动词，代表对象能做什么。

现实中的对象

- 请分析以下对象都有哪些属性和方法？

属性：

品牌：apple
颜色：银色
价格：5988.0
重量：138克
.....

方法：

收发短信
接打电话
.....



属性：

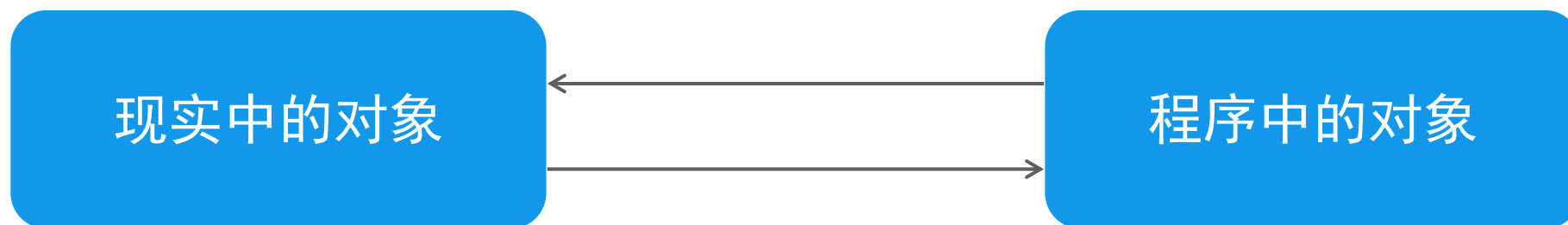
品牌：保时捷
颜色：白色
型号：911
产地：德国
.....

方法：

前进
后退
.....

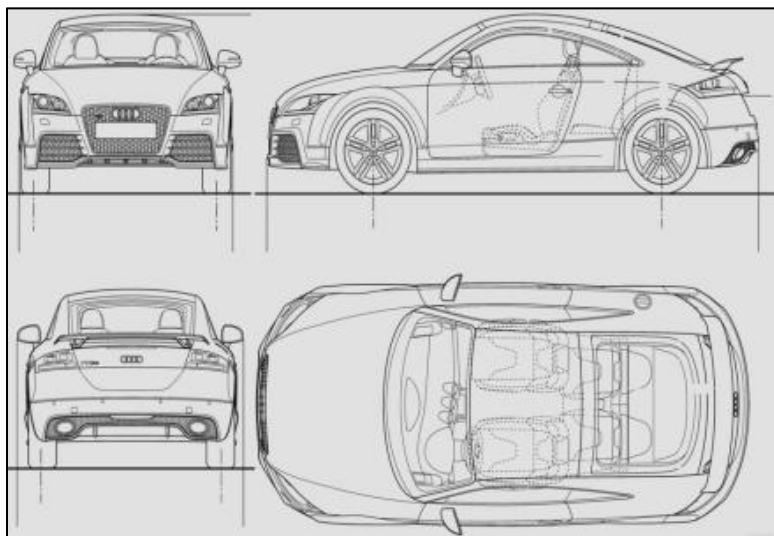
- 如何使用程序模拟现实世界，解决现实问题？
 - 首先，在程序当中，必须具有和现实中相同的对象，用以模拟现实世界。
 - 然后，使用程序中的对象代表现实中的对象，并执行操作，进而解决现实问题。

如何在程序中创造出与现实中一样的对象？



- 现实中的对象多数来自于“模板”，程序中的对象也应该具有“模板”。

什么是类



汽车设计图纸规定了该款汽车所有的组成部分，包括外观形状、内部结构、发动机型号、安全参数等具体的信息。这即为现实对象的模板。程序中的模板也有相同作用，称之为“类”。



按照设计图纸创造出来的汽车，才是真实存在、切实可用的实体，所以汽车实体被称为现实中的对象。而通过程序中的模板创造出来的实体，即为程序中的对象，称之为“对象”。

类的抽取

- 在一组相同或类似的对象中，抽取出共性的特征和行为，保留所关注的部分。



属性：

品种
年龄
性别
毛色
.....

方法：

吃
睡
.....

类的定义

```
public class Dog {
```

类名

```
String breed; //品种  
int age; //年龄  
String sex; //性别  
String furColor; //毛色
```

```
public void eat(){  
    System.out.println("eating...");  
}  
  
public void sleep(){  
    System.out.println("sleeping...");  
}
```

```
}
```

属性：通过变量表示，又称实例变量。

语法：数据类型 属性名；

位置：类的内部，方法的外部。

方法：通过函数表示，又称实例方法。

语法：

```
public 返回值类型 方法名(形参) {  
    //方法的主体  
}
```

注意：不再书写static，后续详解。

对象的创建

将对象保存在相同类型的myDog变量中，
myDog变量称为“对象名”或“引用名”

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

```
        Dog myDog = new Dog();
```

基于Dog类创建对象

```
        myDog.breed = "萨摩";  
        myDog.age = 2;  
        myDog.sex = "公";  
        myDog.furColor = "白色";
```

访问属性：对象名. 属性名 = 值； //赋值

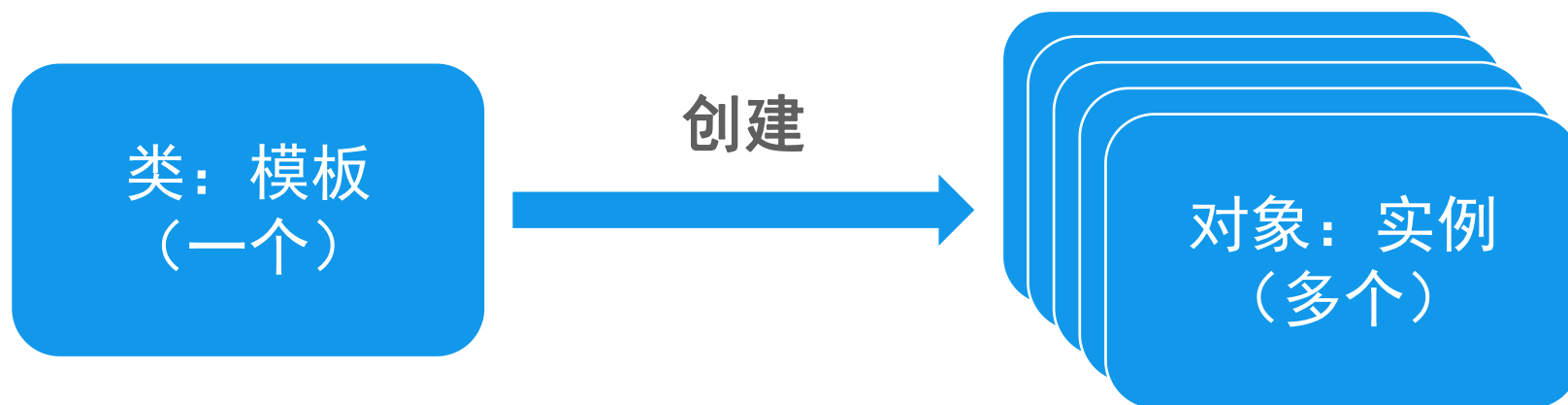
访问属性：对象名. 属性名； //取值

```
        System.out.println(myDog.breed + "\t" + myDog.age + "\t" + myDog.sex + "\t" + myDog.furColor);
```

```
        myDog.eat();  
        myDog.sleep();
```

调用方法：对象名. 方法名()；

```
    }  
}
```



- 类：定义了对对象应具有的特征和行为，类是对象的模板。
- 对象：拥有多个特征和行为的实体，对象是类的实例。

- 思考：之前学习局部变量时，要求必须先赋值再使用，否则编译错误。对于实例变量而言，未赋值并不会编译错误，能否直接访问？

```
public class TestCreateObject {  
    public static void main(String[] args) {  
  
        Dog myDog = new Dog();  
  
        //省略赋值语句  
  
        System.out.println(myDog.breed + "\t" + myDog.age + "\t" + myDog.sex + "\t" + myDog.furColor);  
    }  
}
```

实例变量的默认值：

整数：0

小数：0.0

字符：\u0000（空格）

布尔：false

其他：null

运行结果：null 0 null null

实例变量与局部变量的区别

	局部变量	成员变量
定义位置	方法或方法内的结构当中	类的内部，方法的外部
默认值	无默认值	字面值（与数组相同）
使用范围	从定义行到包含其结构结束	本类有效
命名冲突	不允许重名	可与局部变量重名，局部变量优先

- 对象的实例方法包含两部分：方法的声明和方法的实现。
 - 方法的声明：
 - 代表对象能做什么。
 - 组成：修饰符 返回值类型 方法名(形参列表)
 - 方法的实现：
 - 代表对象怎么做：即如何实现对应的功能。
 - 组成：{ }

- 定义学生类：
 - 属性：姓名(name)、年龄(age)、性别(sex)、分数(score)
 - 方法：打招呼(sayHi) //打印学生所有信息
- 创建多个学生对象，为其各个属性赋值，并调用方法。

方法重载概念

- 有些情况下，对象的同一种行为可能存在多种实现过程。
- 例如：人对象的吃行为，吃饭和吃药的过程就存在差异。

到底采用哪种实现形式，
需要取决于调用者给定的参数。

```
public class Person {  
    public void eat(食物 a){  
        //食物放入口中  
        //咀嚼  
        //咽下  
    }  
    public void eat(药物 b){  
        //药物放入口中  
        //喝水  
        //咽下  
    }  
    public void eat(口香糖 c){  
        //口香糖放入口中  
        //咀嚼  
        //吐出  
    }  
}
```

- 重载 (Overload) : 一个类中定义多个相同名称的方法。
- 要求:
 - 方法名称相同。
 - 参数列表不同 (类型、个数、顺序)。
 - 与访问修饰符、返回值类型无关。
- 调用带有重载的方法时, 需要根据传入的实参去找到与之匹配的方法。
- 好处: 屏蔽使用差异, 灵活、方便。

- 思考：以下方法是不是重载？

```
public void m(int a) {}
```

```
public void m(int b) {}
```

- 两个方法的方法名称和参数列表都相同，只有参数名称不一样，编译报错。
- 注意：只是参数名称不同，并不能构成方法的重载。

- 构造方法：类中的特殊方法，主要用于创建对象。
- 特点：
 - 名称与类名完全相同。
 - 没有返回值类型。
 - 创建对象时，触发构造方法的调用，不可通过句点手动调用。
- 注意：如果没有在类中显示定义构造方法，则编译器默认提供无参构造方法。

对象创建过程

```
public class TestConstructors {  
    public static void main(String[] args) {  
        Student s = new Student();  
    }  
}  
  
class Student{  
    String name;  
    int age;  
    String sex;  
    double score;  
  
    public Student(){  
        System.out.println("Student() Executed");  
    }  
}
```

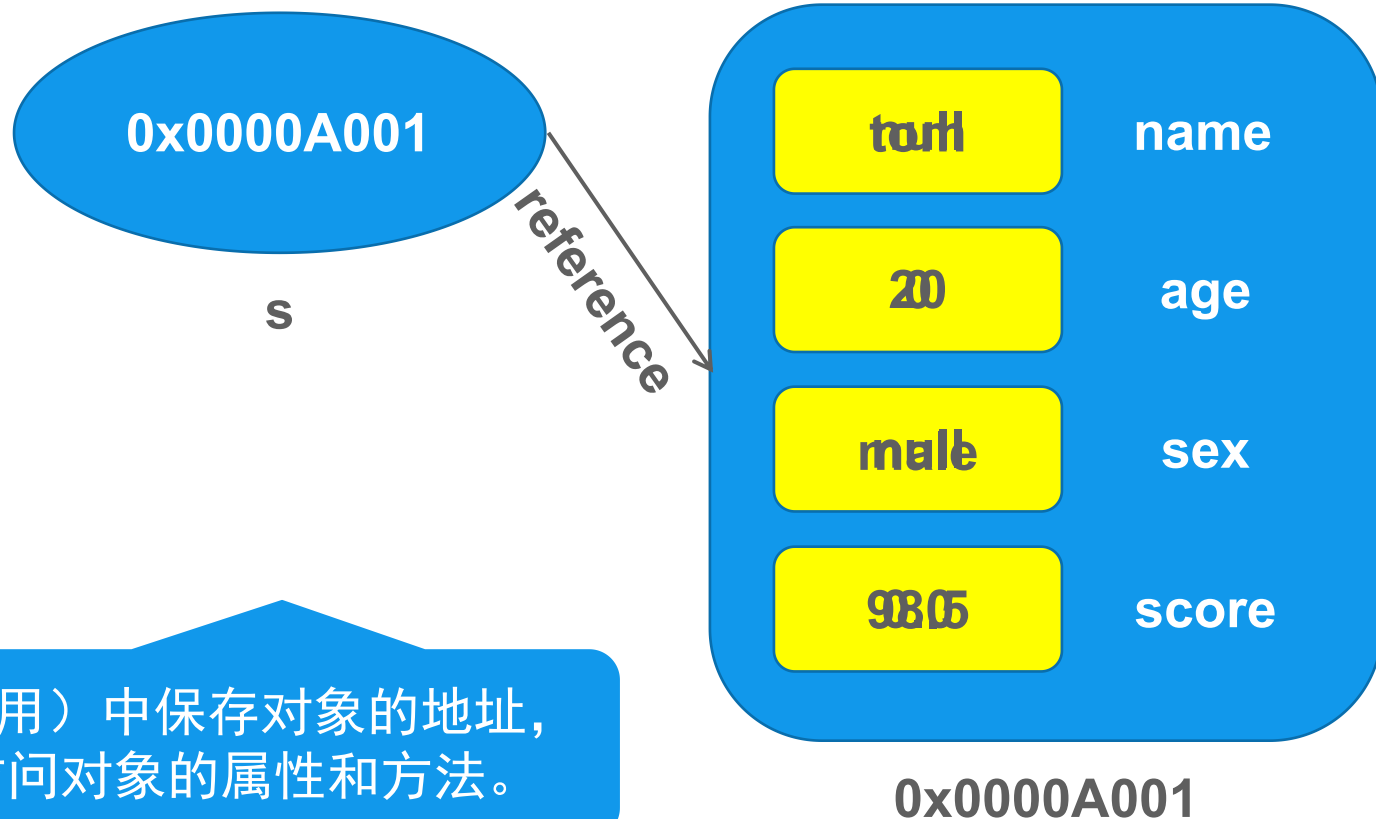
new Student(); 触发对象创建

- 对象的创建过程：
 - 内存中开辟对象空间
 - 为各个属性赋予初始值
 - 执行构造方法中的代码
 - [将对象的地址赋值给变量]

对象的内存分配

```
Student s = new Student();  
s.name = "tom";  
s.age = 20;  
s.sex = "male";  
s.score = 98.5;
```

内存



存储对象的变量s（引用）中保存对象的地址，通过变量中的地址访问对象的属性和方法。

构造方法重载

- 构造方法也可重载，遵循重载规则。

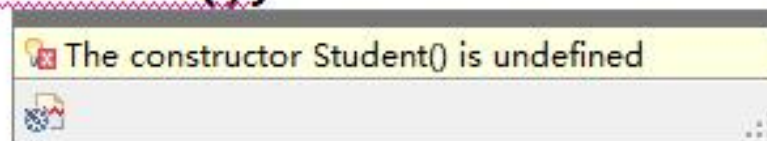
```
class Student{  
    String name;  
    int age;  
    String sex;  
    double score;  
  
    public Student(){  
        System.out.println("Student() Executed");  
    }  
  
    public Student(String name){  
        System.out.println("Student(String name) Executed");  
    }  
  
    public Student(String name , int age){  
        System.out.println("Student(String name , int age) Executed");  
    }  
}
```

```
public class TestConstructors {  
    public static void main(String[] args) {  
        new Student();  
  
        new Student("tom");  
  
        new Student("jack" , 20);  
    }  
}
```

创建对象时，根据传入参数，
匹配对应的构造方法。

默认构造方法

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



编译错误：无参构造方法未定义

```
class Student{  
    String name;  
    int age;  
    String sex;  
    double score;  
  
    public Student(String n, int a, String s, double sc) {  
  
    }  
}
```

在类中，如果没有显示定义构造方法，则编译器默认提供无参构造方法。

如已手动添加有参构造方法，则无参构造方法不再默认提供，可根据需求自行添加。

构造方法为属性赋值

```
public class TestConstructors {  
    public static void main(String[] args) {  
        Student s = new Student("tom", 20, "male", 98.5);  
        System.out.println(s.name + "\t" + s.age + "\t" + s.sex + "\t" + s.score);  
    }  
}  
  
class Student {  
    String name;  
    int age;  
    String sex;  
    double score;  
  
    public Student(String n, int a, String s, double sc) {  
        name = n;  
        age = a;  
        sex = s;  
        score = sc;  
    }  
}
```

创建对象的同时，将值传入构造方法

运行结果：tom 20 male 98.5

由构造方法为各个属性赋值

this关键字

```
public class TestThisKeyword {  
    public static void main(String[] args) {  
        Student s1 = new Student();//0x0000A001  
        s1.sayHi();  
  
        Student s2 = new Student();//0x0000B002  
        s2.sayHi();  
    }  
}  
  
class Student{  
    String name;  
    int age;  
    String sex;  
    double score;  
  
    public void sayHi(){  
        System.out.println(this.name);  
    }  
}
```

类是模板，可服务于此类的所有对象；
this是类中的默认引用，代表**当前实例**；
当类服务于某个对象时，**this**则指向这个对象。

当创建s1对象时，**this**指向0x0000A001，
访问的name属性即是0x0000A001地址中的name空间；
当创建s2对象时，**this**指向0x0000B002，
访问的name属性即是0x0000B002地址中的name空间；

this关键字

- this第一种用法：调用实例属性、实例方法。如：this.name、this.sayHi()

```
public class TestThisKeyword {  
    public static void main(String[] args) {  
        Student s1 = new Student();  
        s1.sayHi();  
    }  
}  
  
class Student{  
    String name = "tom";  
  
    public void sayHi(){  
        String name = "jack";  
        System.out.println(name);  
        System.out.println(this.name);  
    }  
}
```

当实例变量和局部变量重名时，优先访问局部变量；此时，如需访问实例变量，需要增加this. 前缀。不存在重名时，则可省略this.

运行结果：

jack
tom

this关键字

- this第二种用法：调用本类中的其他构造方法。如：this()、this(实参)

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

在构造方法中，调用本类的其他构造方法，即可复用构造方法中的逻辑代码。

包含多条冗余代码。

this()：调用无参构造
this(实参)：调用有参构造
注：必须在构造方法的首行

四参构造将接收到的实参直接传递给三参构造进行属性赋值。

- 什么是对象：
 - 一切客观存在的事物都是对象，万物皆对象。
- 什么是类：
 - 定义了对象应具有的特征和行为，类是对象的模板。
- 什么是方法重载：
 - 方法名相同、参数列表不同。
- 什么是构造方法：
 - 类中用于创建对象的特殊方法，名称与类名相同，没有返回值，不可通过句点调用。
- 什么是this关键字：
 - 代表当前实例，通过this. 访问实例成员；通过this()/this(xxx)访问本类中的其他构造方法。