

一个程序就是一个世界，有很多事物(对象[属性，行为])

类与对象：

- 1) 类是抽象的，概念的，代表一类事物, 比如人类, 猫类..，即它是数据类型.
- 2) 对象是具体的，实际的，代表一个具体事物，即 是实例.
- 3) 类是对象的模板，对象是类的一个个体，对应一个实例

属性：

属性是类的一个组成部分，一般是基本数据类型，也可是引用类型(对象，数组)。

属性细节：

- 1) 属性的定义语法同变量，示例：访问修饰符 属性类型 属性名；

有四种访问修饰符 public, protected, 默认, private ,后面我会详细介绍

- 2) 属性的定义类型可以为任意类型，包含基本类型或引用类型
- 3) 属性如果不赋值，有默认值，规则和数组一致。具体说：int 0, short 0, byte 0, long 0, float 0.0, double 0.0, char \u0000, boolean false, String null

创建对象：

- 1) 先声明再创建

```
Cat cat ; //声明对象 cat
```

```
cat = new Cat(); //创建
```

- 2) 直接创建

```
Cat cat = new Cat();
```

访问属性：对象名. 属性名；

Java 的内存结构：

- 1) 栈： 一般存放基本数据类型(局部变量)
- 2) 堆： 存放对象(Cat cat , 数组等)
- 3) 方法区： 常量池(常量，比如字符串)， 类加载信息

成员方法：

在某些情况下，我们要需要定义成员方法(简称方法)。

访问修饰符 返回数据类型 方法名（形参列表..） { //方法体  
语句;

return 返回值;

}

- 1) 形参列表：表示成员方法输入
- 2) 返回数据类型：表示成员方法输出，void 表示没有返回值
- 3) 方法主体：表示为了实现某一功能代码块
- 4) return 语句不是必须的。

使用细节：

访问修饰符（作用是控制 方法使用的范围）如果不写默认访问，[有四种：  
public, protected, 默认, private]

返回数据类型

- 1) 一个方法最多有一个返回值 [思考，如何返回多个结果 返回数组 ]
- 2) 返回类型可以为任意类型，包含基本类型或引用类型(数组，对象)
- 3) 如果方法要求有返回数据类型，则方法体中最后的执行语句必须为 return 值；而且要求返回值类型必须和 return 的值类型一致或兼容
- 4) 如果方法是 void，则方法体中可以没有 return 语句，或者 只写 return ;

方法名

遵循驼峰命名法，最好见名知义，表达出该功能的意思即可。

#### ✓ 形参列表

1. 一个方法可以有0个参数，也可以有多个参数，中间用逗号隔开，比如 `getSum(int n1,int n2)`
2. 参数类型可以为任意类型，包含基本类型或引用类型，比如 `printArr(int[][] map)`
3. 调用带参数的方法时，一定对应着参数列表传入相同类型或兼容类型 的参数！【`getSum`】
4. 方法定义时的参数称为形式参数，简称形参；方法调用时的传入参数称为实际参数，简称实参，实参和形参的类型要一致或兼容、个数、顺序必须一致！[演示]

#### ✓ 方法体

里面写完成功能的具体的语句，可以为输入、输出、变量、运算、分支、循环、方法调用，但里面不能再定义方法！即：方法不能嵌套定义。[演示]

### MethodDetail02.java

#### ✓ 方法调用细节说明(!!!)

1. 同一个类中的方法调用：直接调用即可。比如 `print(参数)`;  
案例演示：A类 `sayOk` 调用 `print()`
2. 跨类中的方法A类调用B类方法：需要通过对象名调用。比如 `对象名.方法名(参数)`；案例演示：B类 `sayHello` 调用 `print()`
3. 特别说明一下：跨类的方法调用和方法的访问修饰符相关，先暂时这么提一下，后面我们讲到访问修饰符时，还要再细说。

方法的递归调用：

递归就是方法自己调用自己

1. 执行一个方法时，就创建一个新的受保护的独立空间(栈空间)
2. 方法的局部变量是独立的，不会相互影响，比如n变量
3. 如果方法中使用的是引用类型变量(比如数组，对象)，就会共享该引用类型的数据。
4. 递归必须向退出递归的条件逼近，否则就是无限递归,出现 `StackOverflowError`，死龟了:)
5. 当一个方法执行完毕，或者遇到return，就会返回，遵守谁调用，就将结果返回给谁，同时当方法执行完毕或者返回时，该方法也就执行完毕。

方法的重载：java 中允许同一个类中，多个同名方法的存在，但要求 形参列表不一致！

- 1) 方法名：必须相同
- 2) 形参列表：必须不同（形参类型或个数或顺序，至少有一样不同，参数名无要求）
- 3) 返回类型：无要求

可变参数：java 允许将同一个类中多个同名同功能但参数个数不同的方法，封装成一个方法。

访问修饰符 返回类型 方法名(数据类型... 形参名) {  
}

### VarParameterDetail.java

- 1) 可变参数的实参可以为0个或任意多个。
- 2) 可变参数的实参可以为数组。
- 3) 可变参数的本质就是数组。
- 4) 可变参数可以和普通类型的参数一起放在形参列表，但必须保证可变参数在最后
- 5) 一个形参列表中只能出现一个可变参数

作用域：

面向对象中，变量作用域是**非常重要**知识点，相对来说不是特别好理解，请大家注意听，认真思考，要求深刻掌握变量作用域。**VarScope.java**

1. 在java编程中，主要的变量就是属性(成员变量)和局部变量。
2. 我们说的局部变量一般是指在成员方法中定义的变量。【举例 Cat类: cry】
3. java中作用域的分类  
全局变量：也就是属性，作用域为整个类体 Cat类: cry eat 等方法使用属性  
【举例】  
局部变量：也就是除了属性之外的其他变量，作用域为定义它的代码块中！
4. 全局变量(属性)可以不赋值，直接使用，因为有默认值，局部变量必须赋值后，才能使用，因为没有默认值。【举例】

#### **VarScopeDetail.java**

1. 属性和局部变量可以重名，访问时遵循就近原则。
2. 在同一个作用域中，比如在同一个成员方法中，两个局部变量，不能重名。【举例】
3. **属性生命周期较长**，伴随着对象的创建而创建，伴随着对象的销毁而销毁。局部变量，**生命周期较短**，伴随着它的代码块的执行而创建，伴随着代码块的结束而销毁。即在一次方法调用过程中。

4. **作用域范围不同**  
全局变量/属性：可以被本类使用，或其他类使用（通过对象调用）  
局部变量：只能在本类中对应的方法中使用
5. **修饰符不同**  
全局变量/属性可以加修饰符  
局部变量不可以加修饰符

构造器：

[修饰符] 方法名(形参列表){

方法体；

}

- 1) 构造器的修饰符可以默认，也可以是 public protected private
- 2) 构造器没有返回值
- 3) 方法名 和类名字必须一样
- 4) 参数列表 和 成员方法一样的规则
- 5) 构造器的调用，由系统完成

构造方法又叫构造器(constructor)，是类的一种特殊的方法，它的主要作用是完成对新对象的初始化。它有几个特点：

- 1) 方法名和类名相同
- 2) 没有返回值



3) 在创建对象时，系统会自动的调用该类的构造器完成对象的初始化。

#### ConstructorDetail.java

1. 一个类可以定义多个不同的构造器，即构造器重载  
比如：我们可以再给Person类定义一个构造器,用来创建对象的时候,只指定人名,不需要指定年龄
2. 构造器名和类名要相同
3. 构造器没有返回值
4. 构造器是完成对象的初始化, 并不是创建对象
5. 在创建对象时,系统自动的调用该类的构造方法
6. 如果程序员没有定义构造器, 系统会自动给类生成一个默认无参构造器(也叫默认构造器), 比如 `Dog (){}` , 使用 **javap指令** 反编译看看
7. 一旦定义了自己的构造器,默认的构造器就覆盖了, 就不能再使用默认的了, 除非显式的定义一下,即: `Dog(){}`  写 **(这点很重要)**

This 关键字：

- 1) this 关键字可以用来访问本类的属性、方法、构造器
- 2) this 用于区分当前类的属性和局部变量
- 3) 访问成员方法的语法：this. 方法名(参数列表)；
- 4) 访问构造器语法：this(参数列表)；注意只能在构造器中使用(即只能在构造器中访问另外一个构造器，必须放在第一条语句)
- 5) this 不能在类定义的外部使用，只能在类定义的方法中使用。