一、类与对象

1、类与对象的区别

```
    1.1、类是抽象的、概念的，代表一类事物，比如人类、猫类...，即数据类型
    1.2、对象是具体的、实际的，代表一个具体事务，即实例
    1.3、类是对象的模板，对象是类的个体，对饮一个实例
```

2、属性 / 成员变量

```
# 2.1、基本介绍
    - 从概念上看：成员变量 = 属性 = 字段(field)
    - 属性是类的一个组成部分，一般是基本数据类型，也可以是引用类型
```

```
# 2.2、细节说明
    - 属性的定义语法跟变量相同，示例：[访问修饰符] [属性类型] [属性名];
    - 属性的定义类型可以是任意类型，包括基本类型或引用类型
    - 属性如果不赋值，有默认值，和数组的情况一样
        int 0
        short 0
        byte 0
        long 0
        float 0.0
        double 0.0
        char \u0000
        boolean false
        String null
```

3、访问属性

```
    [对象名].[属性名];
```

4、类与对象的内存分配

```
# 4.1、JAVA内存结构
    - 栈：一般存放基础数据类型（局部变量）
    - 堆：存放对象（Cat cat,数组等）
    - 方法区：常量池（常量，如字符串）、类加载信息
```

```
# 4.2、创建对象流程
    - 先加载类信息
    - 在堆中分配空间，进行默认初始化
    - 把地址赋给p ,p就指向对象
    - 进行指定初始化
```

二、成员方法

1、简介

在某些情况下，除了属性外，我们还需要定义类似行为的成员方法，如说话、跑步等，此时就需要用到成员方法才能完成。

2、方法调用机制

2.1、当程序执行到方法时，就会开辟一个独立的空间（栈空间）
2.2、当方法执行完毕，或者执行到return语句，就会返回，并销毁这个栈空间
2.3、返回到一开始调用方法的地方
2.4、返回后，继续执行后面的代码
2.5、当main方法执行完毕后，也会销毁main方法的栈空间，整个程序退出

3、方法优点

1、提高代码的复用性
2、可以将实现的细节封装起来，当其他用户需要的时候直接调用即可

4、方法定义

```
访问修饰符 返回数据类型 方法名(形参) {
    // 方法体
    语句;
    return 返回值;
}
```

1、形参：表示成员方法输入 cal(int n), getSum(int num1, int num2)
2、返回数据类型：表示成员方法输出的数据类型，void表示没有return返回值
3、方法主体：表示为了实现某一功能的代码块
4、return：非必需。返回一个或一组数据

5、细节说明

```
# 访问修饰符（作用是控制方法使用的范围）
    可选[public, protected, 默认, private]
```

```
# 返回数据类型
    1、一个方法最多有一个返回值
    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;而且要求返回数据类型必须和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，或只写return ;
```

```
# 方法名
    遵循驼峰命名法，最好见名知义
```

形参列表

1、一个方法可以有0个或多个参数，中间用逗号隔开，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，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

4、方法定义时的参数称为形式参数，简称形参；方法调用时的参数称为实际参数，简称实参；实参和形参的数据类型要一致或兼容，个数、顺序必须一致

方法体

里面写完成功能的具体语句。

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，即不能嵌套定义

方法调用说明

1、同一个类中的方法调用：直接调用即可。比如 print(参数);

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);

3、跨类的方法调用和方法的访问修饰符相关

三、成员方法传参机制

1、简介

基本数据类型，传递的是值（值拷贝），形参的任何改变不影响实参。

引用数据类型，传递的是地址（地址拷贝），可以通过形参影响实参。

四、方法递归调用

1、简介

递归就是方法自己调用自己，每次调用时传入不同的变量。递归有助于编程者解决复杂的问题，同时可以让代码变得简洁。

2、应用场景

2.1、数学，如：8皇后、汉诺塔、阶乘、迷宫、球和篮子等

2.2、算法，如：快排、并归排序、二分查找、分治算法等

2.3、栈：递归代码更加简介

3、重要规则

3.1、执行一个方法时，就创建一个新的栈空间（受保护的独立空间）

3.2、方法的局部变量的是独立的，不会相互影响

3.3、如果方法中使用的是引用类型变量（如数组），就会共享该引用类型的数据

3.4、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

3.5、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

五、重载(overload)

1、简介

> 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。

2、优点

> 2.1、减轻了起名的麻烦
> 2.2、减轻了记名的麻烦

3、细节

> 3.1、方法名：必须相同
> 3.2、形参列表：必须不同
> 3.3、返回类型：无要求

六、可变参数

1、简介

> Java允许将同一个类种多个同名同功能但参数个数不同的方法封装成一个方法。

2、语法

```
访问修饰符 返回类型 方法名(数据类型... 形参名) {
    ...
}
```

3、细节

> 3.1、可变参数的实参可以为0个或多个
> 3.2、可变参数的实参可以为数组
> 3.3、可变参数的本质就是数组
> 3.4、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
> 3.5、一个形参列表种只能出现一个可变参数

七、作用域

1、简介

> 作用域可以理解为一个变量的适用范围。

2、特点

2.1、在Java中，主要的变量就是属性(全局变量)和局部变量
2.2、常说的局部变量一般是指在成员方法中定义的变量
2.3、全局变量/属性：作用域为整个类体
局部变量：也就是除了属性以外的其他变量，作用域为定义它的代码块中
2.4、全局变量可以不赋值，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

3、细节

3.1、属性和局部变量可以重名，访问时遵循就近原则
3.2、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
3.3、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
3.4、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

3.5、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

八、构造器

1、简介

构造方法又叫构造器（constructor），是类的一种特殊方法，它的主要作用是完成对新对象的初始化。

2、特点

2.1、方法名和类名相同
2.2、没有返回值
2.3、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

3、细节

3.1、一个类可以定义多个不同的构造器，即构造器重载
3.2、构造器名和类名要相同
3.3、构造器没有返回值
3.4、构造器是完成对象的初始化，并不是创建对象
3.5、在创建对象时，系统自动的调用该类的构造方法
3.6、如果程序员没有定义构造方法，系统会自动给类生成一个默认无参构造方法（也叫默认构造方法），比如Person(){}，可通过javap指令反编译class文件进行查看
3.7、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无参构造器，除非显式的定义一下，即Person(){}

九、this

1、简介

Java虚拟机会给每个对象分配this，代表当前对象。哪个对象调用，this就代表哪个对象。

2、细节

2.1、this关键字可以用来访问本类的属性、方法、构造器
2.2、this用于区分当前类的属性和局部变量
2.3、访问成员方法的语法：this.方法名(参数列表);
2.4、访问构造器语法：this(参数列表);只能在构造器中使用
2.5、this不能再类定义的外部使用，只能再类定义的方法中使用