

一，方法的重载：

要求：

- 1) 在同一个类中
- 2) 方法名必须相同
- 3) 方法的从参数类表不同（参数的个数，参数类型不一样，参数的顺序不一样）

方法的重载与方法的返回值类型没有关系

同一个类中的方法之间可以调用

二，面向对象个面向过程的区别

面向对象的编程关注于类的设计

- 1) 一个项目不管有多么庞大，一定是一个一个的类构成的，程序的基本单位
 - 2) 类是抽象的，就好比制造汽车的图纸，具体的一辆一辆的车是根据图纸制造的，实际上是类的实例化
 - 3) 完成一个项目（功能）的思路
- ①所要完成的功能对应的类的对象是否存在，若存在，则直接调用类中的属性和方法，若不存在，则需要创建类的对象，甚至说，类都不存在，则需要设计该类

面向对象的三条主线：

- 1) 类及类的构成成分（属性和方法，构造器，构造器，内部类）
- 2) 面向对象编程的特征：封装性，继承性，多态性（抽象性）
- 3) 其他的关键字（修饰的作用）：this super import static abstract interface...

三，类以及对象

1，关于类的设计

2，类的组成成分

- 1) 属性（成员变量，field）
- 2) 方法（成员方法，method）

2.1属性

相同点：1. 遵循变量声明的格式：数据类型 变量名 = 初始化值

2. 都有作用域

不同点：1. 声明的位置的不同：成员变量：声明在类里，方法外

局部变量：声明在方法内，方法的形参部分，代码块内

2. 成员变量的修饰符有四个：public private protected 缺省

局部变量没有修饰符，与所在的方法修饰符相同。

3. 初始化值：一定会有初始化值。

成员变量：如果在声明的时候，不显式的赋值，那么不同数据类型会有不同的默认值

byte short int long ==>0

float double ==>0.0

char ==>空格

boolean ==>>false

引用类型变量==>null

局部变量：一定要显式的赋值。（局部变量没有默认初始化值）

4. 二者在内存中存放的位置不同：成员变量存在于堆空间中；局部变量：栈空间中

总结：关于变量的分类：1) 按照数据类型的不同：基本数据类型（8种） & 引用数据类型

2) 按照声明的位置的不同：成员变量 & 局部变量

2.2 方法

一般情况都是对象来调用

2.2 方法：提供某种功能的实现

1) 实例：public void eat() { // 方法体 }

public String getName() { }

public void setName(String n) { }

格式：权限修饰符 返回值类型 (void: 无返回值/具体的返回值) 方法名(形参) { }

2) 关于返回值类型：void：表明此方法不需要返回值

有返回值的方法：在方法的最后一定有return + 返回值类型对应的变量

记忆：void 与return不可以同时出现在一个方法内。像一对“冤家”。

3) 方法内可以调用本类的其他方法或属性，但是不能在方法内再定义方法！

3. 面向对象编程的思想的落地法则1：

1) 设计并创建类及类的成分

2) 实例化类的对象

3) 通过”对象. 属性“或者”对象. 方法“的形式完成某项功能

4. 内存结构

1) 栈（局部变量，对象的引用，数组的引用）

2) 堆（new出来的东西（对象的实体，数组的实体）含成员变量）

3) 方法区（包含字符串常量）

4) 静态域（声明为static的变量）

以下四个方法构成重载：

```
//定义两个int型变量的和
public int getSum(int i,int j){
    return i + j;
}
//定义三个int型变量的和
public int getSum(int i,int j,int k){
    return i + j + k;
}
//定义两个double型数据的和
public double getSum(double d1,double d2){
    return d1 + d2;
}

//定义三个double型数组的和
public void getSum(double d1,double d2,double d3){
    System.out.println(d1 + d2 + d3);
}
```