复习：

1. 数组

定义格式

内存存储：数组是存储在堆内存中，在栈内存中存储堆内存中对应的地址。数组在存入堆内存的时候需要自动赋予默认值。--- 数组在赋值的时候给的也是地址

int[] arr1 = new int[3];

int[] arr2 = arr1; // 指向同一个空间

数组应用：

数组的遍历：通过下标遍历，增强for循环

元素的排序：冒泡排序（相邻两个元素两两比较），选择排序（选定一个元素和其他元素来依次比较）--- 时间复杂度O(n2),空间复杂度o(1)，是稳定的排序算法；Arrays.sort只能进行升序排序

数组的反转：首尾互换 - 时间复杂度O(n),空间复杂度o(1)

元素的查找：元素无序，循环遍历；元素有序，二分查找 - 时间复杂度O(logn)，空间复杂度o(1)

数组的复制：System.arraycopy - Arrays.copyOf

2. 二维数组

用于存储一维数组

定义格式

内存存储：二维数组的每一个位置上存储的是对应的一维数组的地址；如果没有指定一维数组的大小，那么二维数组的位置上存储得是null

## 方法

当在程序中需要重复使用某段逻辑或者功能，将这段逻辑提取出来形成一种新的形式---方法，也叫函数

### 定义格式

修饰符 返回值类型 方法名(参数列表){

方法体;

return 返回值;

}

// 写一个方法求1-n的和

// 明确结果：求1-n的和，那么和是一个整数，意味着结果是int类型，所以返回值类型也就是int类型

// 明确未知量：求1-n的和，n在方法中不能自动产生，需要使用方法的人来传入一个值，这个时候需要以参数形式来体现

// 定义方法的时候在()中定义了参数 --- 形式参数 - 形参

public static int sum(int n){

int sum = 0;

for(int i = 1; i <= n; i++){

sum += i;

}

return sum;

}

练习：

1. 哥德巴赫猜想：任何一个大于等于6的偶数都可以分解成两个质数之和。

16 = 3 + 13

16 = 5 + 11

输入大于等于6的偶数，然后输出它所有的分解形式

思路：在这个过程中要重复执行的代码是判断质数

2. 亲密数：如果A的所有因子（含1而不含本身）之和等于B，而且B的所有因子（含1而不含本身）之和等于A，A和B就是一对亲密数

16:1+2+4+8 = 15

15:1+3+5 = 9

打印5000以内所有的亲密数

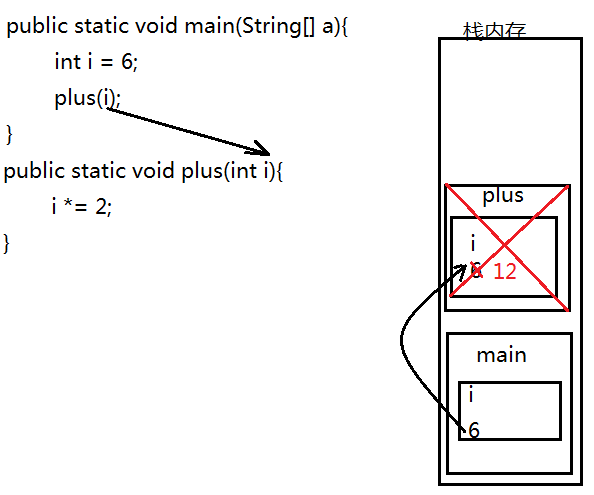
思路：需要重复执行的逻辑是获取一个数的所有因子之和

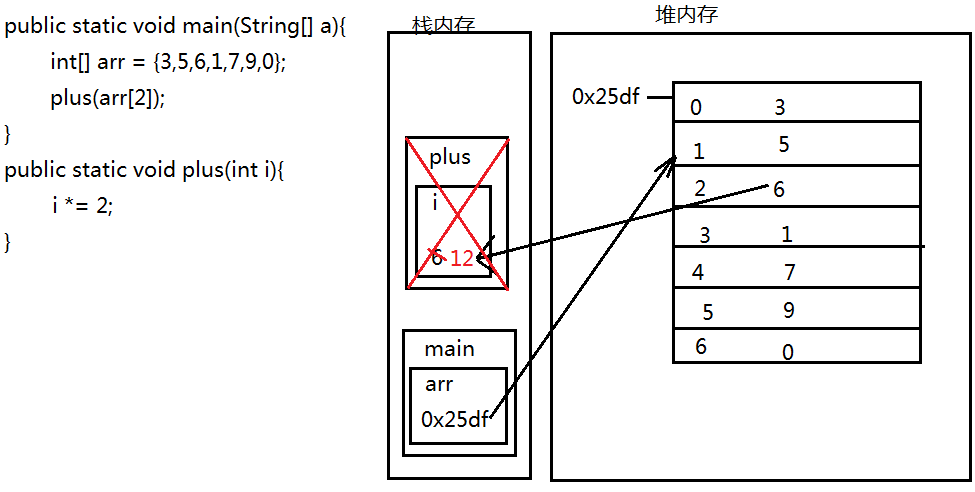
### 方法的重载

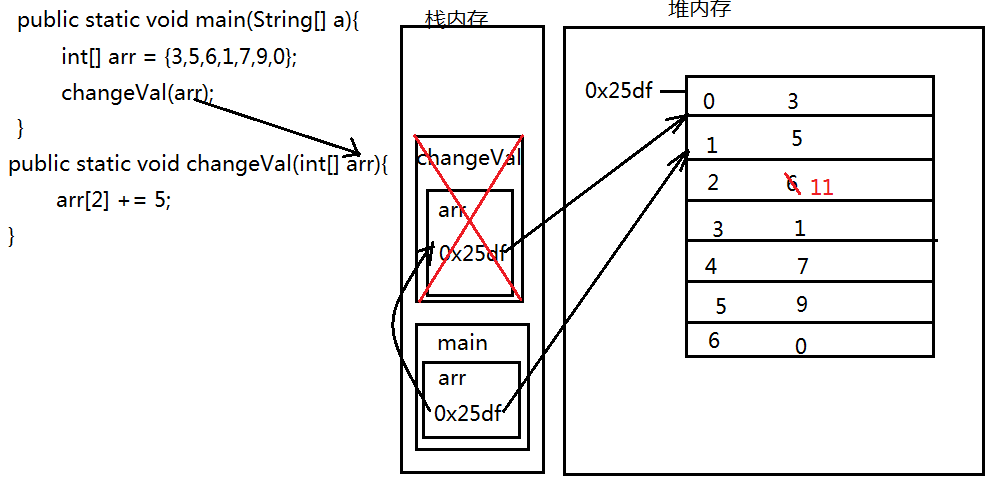
在同一个类中，存在了方法名一致而参数列表不同（参数个数不同或者是对应位置上的参数类型不同）的方法 --- 方法的重载 --- 依靠参数列表来区分调用的方法

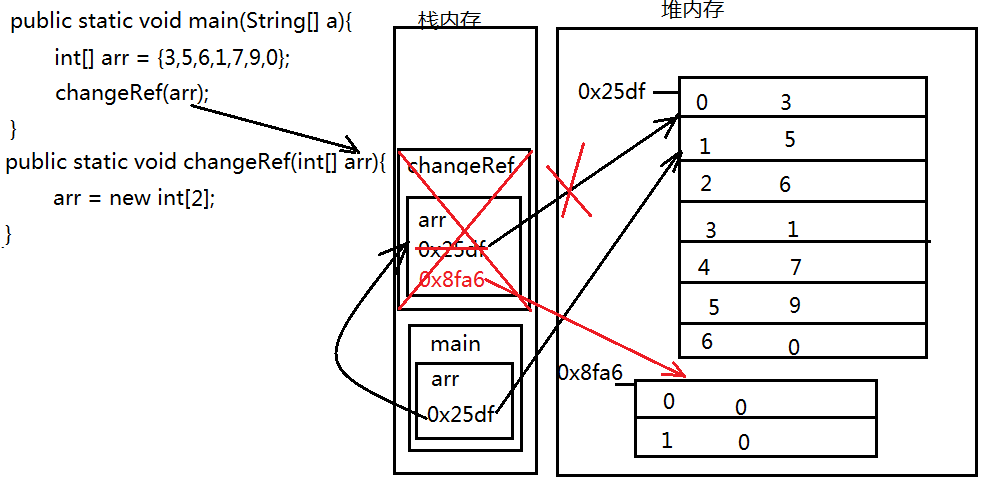
方法在调用的时候会进行模糊匹配---参数类型在没有最符合的情况下会自动提升，就会导致提升之后可能会有多个匹配

### 方法的传值









注意：

1. 方法是在栈内存中执行

2. 方法在传值的时候，基本类型传递的实际值，数组传递的是地址。

3. 传递地址之后，如果地址没有发生改变，则代码执行影响原来的数组；如果地址发生了改变，则改变之后的代码不会影响原来的数组

### 方法的递归

展现形式：方法调用了自己本身

练习：输入一个数字n，求n! = n(n-1)(n-2)...1

6! = 6\*5\*4\*3\*2\*1=720

注意：在递归的时候，当执行次数过多会产生StackOverflowError - 栈溢出错误

总结：在一些场景中，如果能确定这一项和前n项的关系，那么可以使用递归 ---> 逆推

IDE --- 智能开发工具

Eclipse(日食) --- 免费、基于插件、开源、绿色

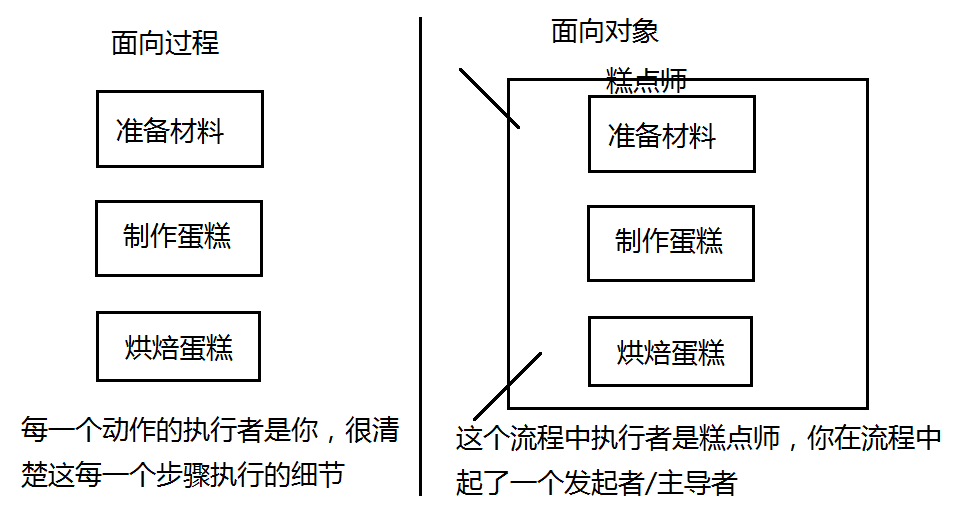
Kepler(开普勒) -> Luna(月神) -> Mars(火星) -> Neon(霓虹灯) -> Oxygen(氧气)

## 面向对象

面向对象是一种思维方式，相对于面向过程而言的。

面向过程在流程中关注动作执行的每一个细节 --- 自己动手做

面向对象重点找这个对象，只要找到了对象，那么这个对象所具有的功能就能够被使用 --- 找别人做

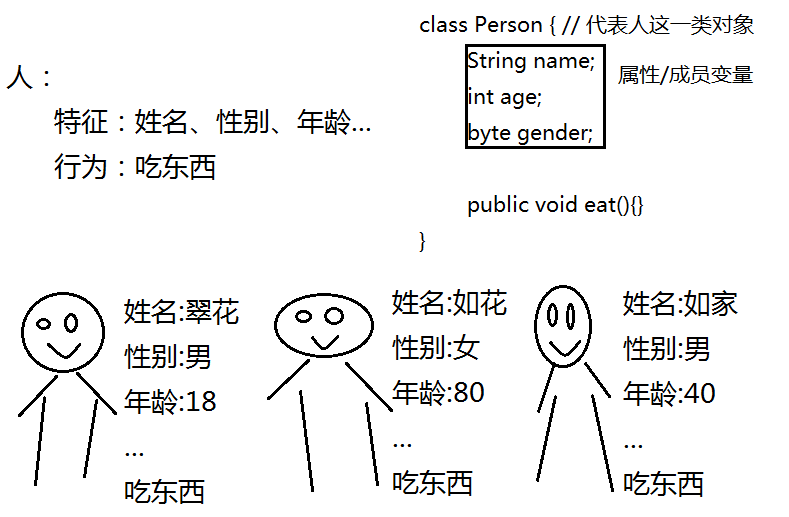


面向对象一定比面向过程好吗？--- 不一定 ---相对简单的事务建议使用面向过程，相对复杂的事务建议使用面向对象

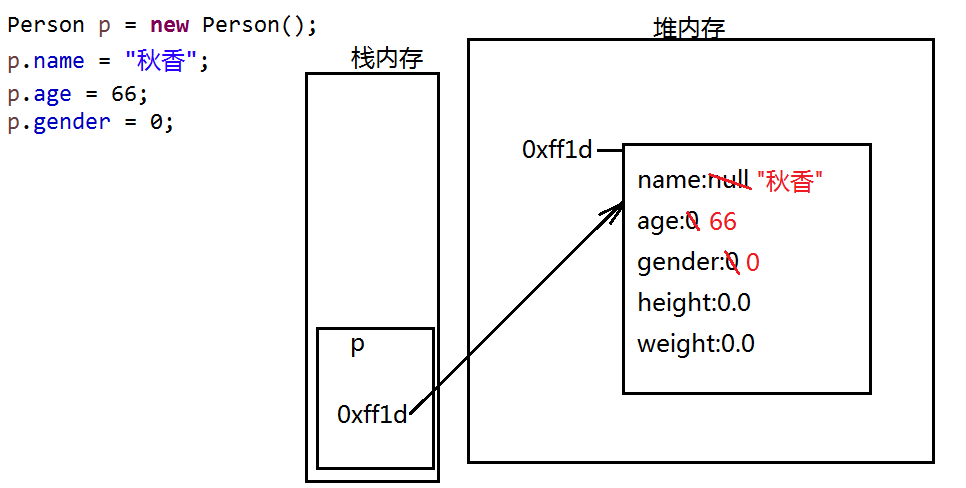
面向对象本身是基于面向过程的

### 类与对象的关系

根据一类对象进行抽取和总结，将这类对象的特征抽取成了属性，将这类对象的行为抽取成了方法，用类表示这一类对象 -> 类是对象的抽取和概括的过程



### 对象的内存存储



### 成员变量和局部变量

1. 定义位置：成员变量定义在类内方法外；局部变量是定义在方法或者是语句里面

2. 作用范围：成员变量作用在整个类内；局部变量是只能作用在定义它的方法或者语句中

3. 内存位置：成员变量随着对象的创建而存在了堆内存中并且在堆内存中赋予了默认值；局部变量在栈内存中存储

4. 生命周期：成员变量再对象创建的时候出现，在对象被回收的时候销毁；局部变量在方法或者语句执行的时候创建，方法或者语句执行完成之后就立即销毁