1. 面向对象思想的体现一：类和对象的创建和执行操作有哪三步？
    1.提供java类：创建的；api提供的
    2.类的实例化（或创建类的对象）
    3.通过对象调用相关结构。“对象.属性” “对象.方法”

比如：Math.sqrt() / Math.random() / Arrays.sort() /...

3. 定义4个方法，要求：
①体现声明方法需要的几个部分。
②体现有返回值和无返回值的情况
③体现有参数和无参数的情况

```
public 返回值类型 方法名(参数类型1 形参1,参数类型2 形参2){


    return
}
```

4. 成员变量和局部变量在声明的位置上、是否有默认初始化值上、是否能有权限修饰符修饰上、内存分配的位置上有何不同？

5. 谈谈return关键字的使用

5. 谈谈return关键字的使用

1. 返回指定类型的变量或常量值的作用
2. 结束方法的功能

冒泡排序的实现：--需要能手写出来

```
int[] arr = new int[]{23,445,32,65,-76,-43,0,2,99,445};

        //使用冒泡排序实现数组元素从小到大的顺序排列
        for(int i = 0;i < arr.length - 1;i++){

            for(int j = 0;j < arr.length - 1 - i;j++){
                if(arr[j] > arr[j + 1]){
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }


        }


        for(int i = 0;i < arr.length;i++){
            System.out.print(arr[i] + "\t");
        }
```

就业面试时：需要能手写冒泡排序、快速排序。

就业面试时：需要能手写冒泡排序O(n^2)、快速排序O(nlogn)。
需要理解排序思想的：堆排序、归并排序。
熟悉：二分法搜索

1.面向对象学习的三条主线：

```
* 1. java中类及类的成员：成员变量（或属性、方法（函数、构造器（或构造方法；代码块（或初始化块、内部类
* 2. 面向对象的特征：封装性、继承性、多态性、(抽象性)
* 3. 其它关键字的使用：this、super、abstract、interface、package、import、. . .
```

2.面向对象与面向过程（理解）

```
* 举例：人把大象装进冰箱
*
* 面向过程：强调的是功能行为，以函数为最小单位，考虑怎么做。
*
* 1.把冰箱门打开
* 2.把大象抬起来，塞进冰箱
* 3.把门关上
*
* 面向对象：将功能封装进对象，强调具备了功能的对象，以类/对象为最小单位，考虑谁来做。
```

3.完成一个项目（或功能）的思路：

① 是否具有完成此功能现成的对象可以使用。如果没有，则考虑创建对象

② 是否已声明了具有此功能对象的类。如果有直接调用。如果没有，需要我们创建具有这样功能的类。

4.面向对象中两个重要的概念：类、对象(或实例)

类：对一类事物的描述、抽象的、概念上的定义
对象：由类所派生出来的，真实存在的一个结构、实例。

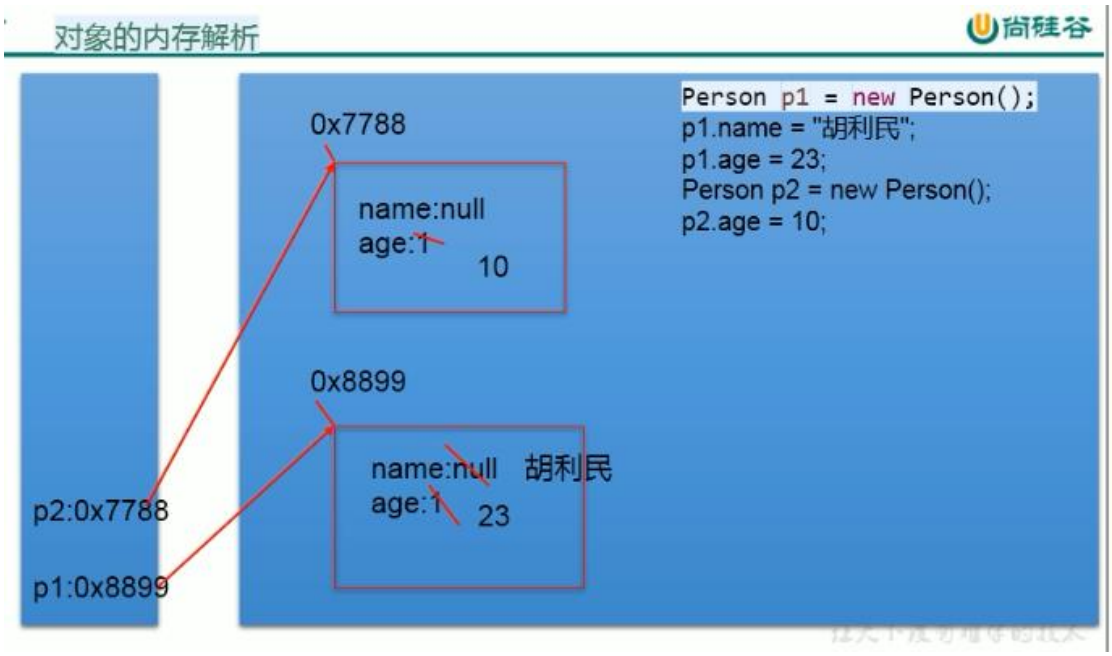
二者的关系：对象，是由类派生(new)出来的。

5.面向对象思想落地实现的规则一

```
* 1. 创建类及提供类的成员：属性、方法
* 2. 类的实例化（创建类的对象
* 3. 调用对象的相关结构:"对象.属性" 或 "对象.方法"
```

补充：几个概念的使用说明

```
* 属性 - 成员变量 - field - （域 - 字段）
* 方法 - 成员方法 - 函数 - method
* 创建类的对象 - 实例化类 - 类的实例化
```




```
* 一、匿名对象的使用
*   因为匿名对象，在创建以后，没有显式的赋一个变量名，所以我们后续不能再次通过变量名的方式，调用此对象：匿名对象只能调用
*
*
```

```
* 二、方法的使用（补充）
* 1. 方法的内可以调用当前类的属性
* 2. 方法内可以调用当前类的方法
*       特别的，方法a内调用方法a: 递归方法
* 3. 方法内不能定义方法
*/
```

练 习 1

编写教师类和学生类，并通过测试类创建对象进行测试

Student类
属性： name:String age:int major:String interests:String
方法：say() 返回学生的个人信息

Teacher类
属性： name:String age:int teachAge:int course:String
方法：say() 输出教师的个人信息

练习2

1.创建一个Person类，其定义如下：

Person
name:String age:int sex:int
+study():void +showAge():void +addAge(int i):int

要求：(1)创建Person类的对象，设置该对象的name、age和sex属性，调用study方法，输出字符串“studying”，调用showAge()方法显示age值，调用addAge()方法给对象的age属性值增加2岁。

(2)创建第二个对象，执行上述操作，体会同一个类的不同对象之间的关系。

2.利用面向对象的编程方法，设计类Circle计算圆的面积。

练习2

3.1 编写程序，声明一个method方法，在方法中打印一个10*8 的*型矩形，在main方法中调用该方法。

3.2 修改上一个程序，在method方法中，除打印一个10*8的*型矩形外，再计算该矩形的面积，并将其作为方法返回值。在main方法中调用该方法，接收返回的面积值并打印。

3.3 修改上一个程序，在method方法提供m和n两个参数，方法中打印一个m*n的*型矩形，并计算该矩形的面积， 将其作为方法返回值。在main方法中调用该方法，接收返回的面积值并打印。

练习2

4.定义类Student，包含三个属性：学号number(int)，年级state(int)，成绩score(int)。 创建20个学生对象，学号为1到20，年级和成绩都由随机数确定，打印出3年级(state值为3）的学生信息。

提示：
1) 生成随机数：Math.random()，返回值类型double;
2) 四舍五入取整：Math.round(double d)，返回值类型long。

4.6.2 方法的重载(overload)

重载的概念

在同一个类中，允许存在一个以上的同名方法，只要它们的参数个数或者参数类型不同即可。

重载的特点：

与返回值类型无关，只看参数列表，且参数列表必须不同。(参数个数或参数类型)。调用时，根据方法参数列表的不同来区别。

重载示例：

```
//返回两个整数的和
  int add(int x,int y){return x+y;}
//返回三个整数的和
  int add(int x,int y,int z){return x+y+z;}
//返回两个小数的和
  double add(double x,double y){return x+y;}
```

```
3  * 方法的重载（overload）
4  *
5  *
6  *
7  * 面试题：区分方法的重载与重写
8  *
9  *
10 * Collection / Collections
11 * finally / final / finalize
12 * throw / throws
13 * ....
14 *
15 */
```

```
3  * 方法的重载（overload）
4  * 1. 判别标准："两同一不同"：同一个类中，相同方法名，且方法的参数列表不同的方法之间，构成方法的重载。
5  *    参数列表不同：参数的个数不同，参数的类型不同
6  *
7  * 2. 方法之间是否构成重载，与方法是否有返回值，权限修饰符是否不同，没有关系！
8  *
9  *
10 *
```

```
11 * 补充1：不允许在一个类中定义相同的方法。何为相同的方法？
12 *      编译器会把同一个类中，相同方法名，且参数列表相同的方法，认为是相同的。跟权限修饰符、返回值类型无关！
13 *
14 *
15 * 补充2：如何确定调用的是某个确定的方法？通过方法名 ----> 通过形参列表
16 *
```

方法的重载

- 使用重载方法，可以为编程带来方便。
- 例如，System.out.println()方法就是典型的重载方法，其内部的声明形式如下：

```
public void println(byte x)
public void println(short x)
public void println(int x)
public void println(long x)
public void println(float x)
public void println(double x)
public void println(char x)
public void println(double x)
public void println()
……
```

练习3

1.判 断：

与void show(int a,char b,double c){}构成重载的有：

a) void show(int x,char y,double z){} // no
b) int show(int a,double c,char b){} // yes
c) void show(int a,double c,char b){} // yes
d) boolean show(int c,char b){} // yes
e) void show(double c){} // yes
f) double show(int x,char y,double z){} // no
g) void shows(){double c} // no

4.6.3 体会可变个数的形参

//下面采用数组形参来定义方法

public static void test(int a ,String[] books);

//以可变个数形参来定义方法

public static void test(int a ,String...books);

说明：

1. 可变参数：方法参数部分指定类型的参数个数是可变多个
2. 声明方式：方法名（参数的类型名...参数名）
3. 可变参数方法的使用与方法参数部分使用数组是一致的
4. 方法的参数部分有可变形参，需要放在形参声明的最后

```
/*
 * 新特性：可变个数形参的方法
 * 1.可变个数形参的格式：数据类型 ... 变量名
 * 2.当调用可变个数形参的方法时，可变个数的形参赋值时，可以赋值的变量个数为：0个，1个，或多个。
 * 3.可变个数形参的方法与同一个类中的其他同名的方法之间构成重载
 * 4.可变个数形参的方法与同类型参数的数组结构，不可同时出现在类声明中。
 * 5.规定：如果方法中存在可变个数形参，要求可变个数形参声明在方法形参的最后。
 * 6.推论：一个方法中最多只能有一个可变个数的形参
 *
 *
```

☆ 4.6.4 方法的参数传递

●方法，必须有其所在类或对象调用才有意义。若方法含有参数：

➢形参：方法声明时的参数

➢实参：方法调用时实际传给形参的参数值

● Java的实参值如何传入方法呢？

Java里方法的参数传递方式只有一种：值传递。 即将实际参数值的副本（复制品）传入方法内，而参数本身不受影响。

4.6.3 体会可变个数的形参

//下面采用数组形参来定义方法

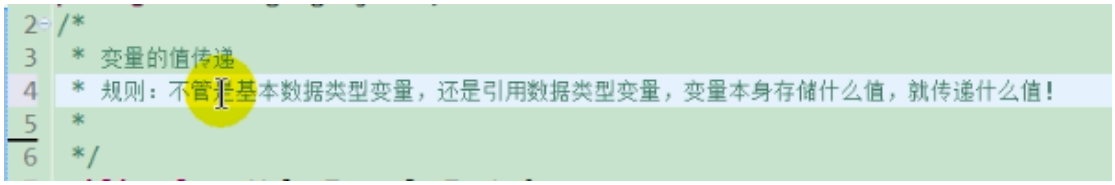
public static void test(int a ,String[] books);

//以可变个数形参来定义方法

public static void test(int a ,String...books);

说明：

1. 可变参数：方法参数部分指定类型的参数个数是可变多个
2. 声明方式：方法名（参数的类型名...参数名）
3. 可变参数方法的使用与方法参数部分使用数组是一致的
4. 方法的参数部分有可变形参，需要放在形参声明的最后



微软：

定义一个int型的数组：int[] arr = new int[]{12,3,3,34,56,77,432};
让数组的每个位置上的值去除以首位置的元素，得到的结果，作为该位置上的新值。遍历新的数组。

答案:

```
//错误写法
for(int i= 0 ; i < arr.length;i++){
    arr[i] = arr[i] / arr[0];
}
```

```
//正确写法1
for(int i = arr.length – 1;i >= 0;i--){
   arr[i] = arr[i] / arr[0];
}
//正确写法2
int temp = arr[0];
for(int i= 0 ; i < arr.length;i++){
    arr[i] = arr[i] / temp;
}
```

4.6.5 递归(recursion)方法

- **递归方法：一个方法体内调用它自身。**
- 方法递归包含了一种隐式的循环，它会重复执行某段代码，但这种重复执行无须循环控制。
- 递归一定要向已知方向递归，否则这种递归就变成了无穷递归，类似于死循环。

```
//计算1-100之间所有自然数的和
public int sum(int num){
        if(num == 1){
                return 1;
        }else{
                return num + sum(num - 1);
        }
}
```