如何创建对象

1) 先声明再创建

```
Cat cat ;    //声明对象 cat   ──> 在栈中有 cat 类 内容为 [null]
cat = new Cat();    //创建   ──> 在堆中开辟对应空间，将地址填在栈中的内容.
```

2) 直接创建

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

如何访问属性

对象名.属性名;　　例：cat.name = "小白";

类和对象的内存分配机制 ☆

```
(base) MacBook-Pro:0508 songfei$ java Object04
小明
200
Exception in thread "main" java.lang.NullPointerException
        at Object04.main(Object04.java:16)
(base) MacBook-Pro:0508 songfei$
```

Object04.java

```
1  public class Object04{
2      // 编写一个main方法
3      public static void main(String[] args){
4          /* 查看以下输出内容
5          */
6
7          Person a = new Person();
8          a.age = 10;
9          a.name = "小明";
10         Person b;
11         b = a;
12         System.out.println(b.name);
13         b.age = 200;
14         b = null;
15         // b的地址为null，即栈b中的地址与堆中的对象空间之间的联系断了，所以访问出现问题；
16         // 但是这属于异常，编译仍会通过
17         System.out.println(a.age);
18         System.out.println(b.name);
19     }
20 }
```

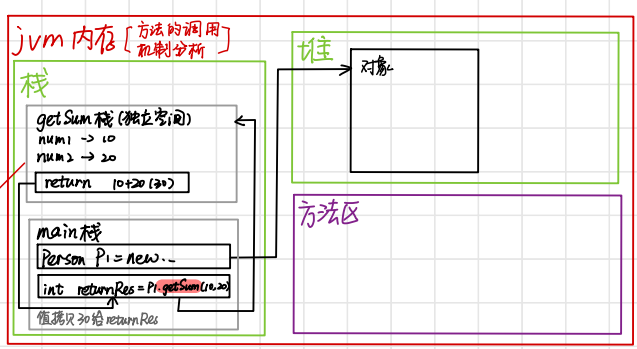
2021.5.9

P202 - P214　Java_成员方法

成员方法，简称方法，人类除了有一些属性外，还有一些行为（说话、跑步、学习...）

```
Person p1 = new Person();
int returnRes = p1.getSum(10, 20);
System.out.println("getSum的值为" + returnRes);
```

```
public int cal01(int num1, int num2){
    return num1 + num2;
}
```



方法调用小结

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) 当方法执行完毕，return语句返回调用方法执行的地方
3) 返回后，继续执行后面代码，此时方法的独立空间被销毁，空间释放

成员方法的好处

1) 提高代码的复用性

2) 可以将实现的细节封装起来，然后供其他用户来调用即可.

方法的定义

```
public 返回数据类型 方法名(形参列表) { //方法体
    语句;
    return 返回值;    // 不是必须的
}
```

访问修饰符(控制方法使用范围)

成员方法的注意事项

- 访问修饰符(作用是控制方法的使用范围)
- 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，包含基本类型和引用类型(数组，对象)
  3) 如果方法的返回数据类型不为 void，则方法体中必须有 return 语句，且返回值类型必须与方法返回数据类型一致或兼容.
  4) 若方法返回值类型为 void，则方法体中不应有 return 或只写 return;
- 方法名

  遵循驼峰命名法，且能基本表达功能意思.
- 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.
  2) 参数类型可以为任意类型，包括基本类型和引用类型
  3) 调用参数的方法时，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；方法调用时的传入参数称为实际参数，简称实参；实参与形参的类型要一致或兼容、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，即方法不能嵌套定义

```
26
27 class AA {
28
29     //细节: 方法不能嵌套定义
30     public void f4() {
31         //错误
32         // public void f5() {
33
34         // }
35     }
36
```

成员方法的使用细节

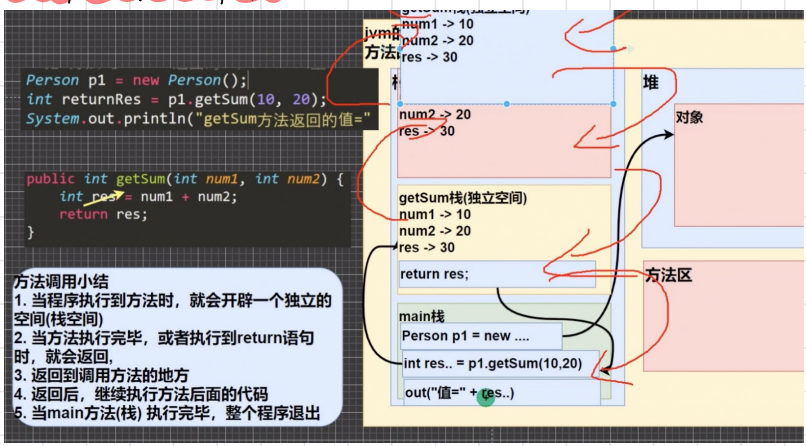
- 方法调用：

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) 特别说明：跨类的方法调用和方法的访问修饰符有关，详细见后文.

方法调用空间分配机制



不断开辟栈的独立空间再不断返回.

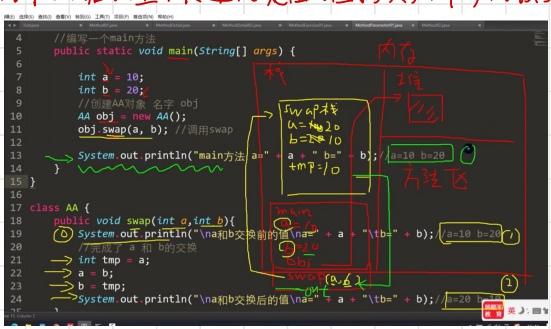
成员方法传参机制

- 基本数据类型的传参机制. Method Parameter 01.java

```
public void swap(int a,int b){
        int tmp = a;
        a = b;
        b = tmp;
        System.out.println("a="+a+"\tb="+b);
}
```

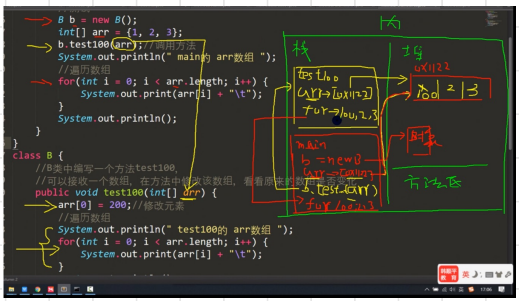
a=10;
b=20;
swap(a,b);
方法内输出: a=20, b=10; 形参
方法外输出: a=10, b=20; 实参.

基本数据类型，传递的是值(值拷贝)，形参的改变不影响实参！

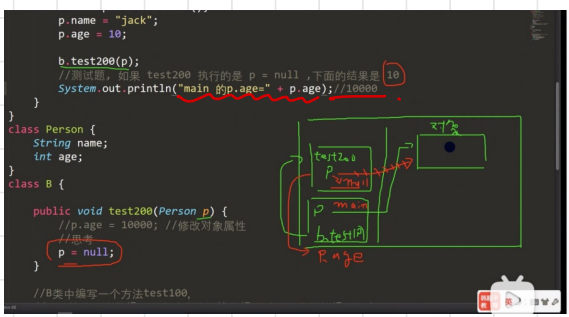


- 引用数据类型的传参机制 Method Parameter02.java.

引用数据类型，传递的是地址（传递的值是地址），可以通过形参影响实参

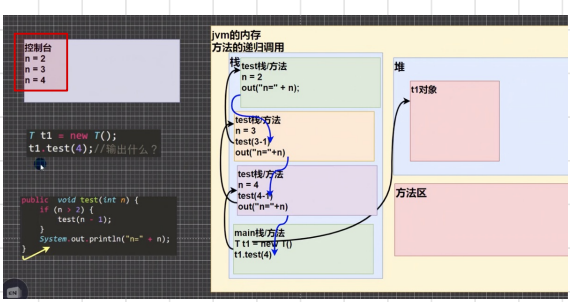


！提问：若在方法中 p=null 则会如何？



只是方法中 p 指向堆空间中的联系断开了，但 main 方法中的指向仍存在。
因为 p 传递的是地址的值，地址被 null 替换，并不会将堆空间中的对象内容修改。

！提问：下方方例代码输出什么？



会输出 n=2;
n=3;
n=4;

方法的递归调用流程。