Java的内存划分为5个部分：

1.栈(Stack): 存放的都是方法中的局部变量。方法的运行一定要在栈当中。
局部变量：方法的参数，或者方法{}内部的变量。
作用域：一旦超出作用域，立刻从栈内存当中消失。

2.堆(Heap): 凡是new出来的东西，都在堆当中。
堆内存里面的东西都有一个地址值：16进制
堆内存里面的数据，都有一个默认值，规则：
如果是整数： 默认为0
如果是浮点数： 默认为0.0
如果是字符： 默认为==‘\u0000’==
如果是布尔: 默认为false
如果是引用类型: 默认为null

3.方法区(Method Area): 存储.class相关信息，包含方法的信息

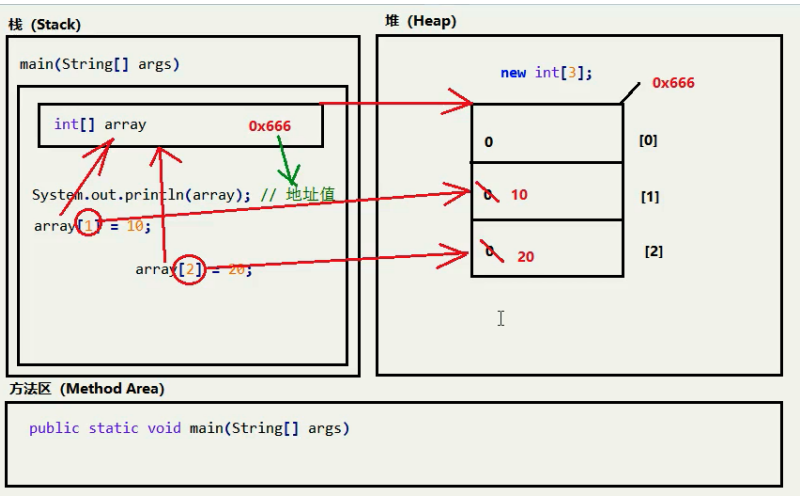
4.本地方法栈(Natave Method Stack): 与操作系统有关

5.寄存器(Pc Register): 与CPU相关。

数组内存图

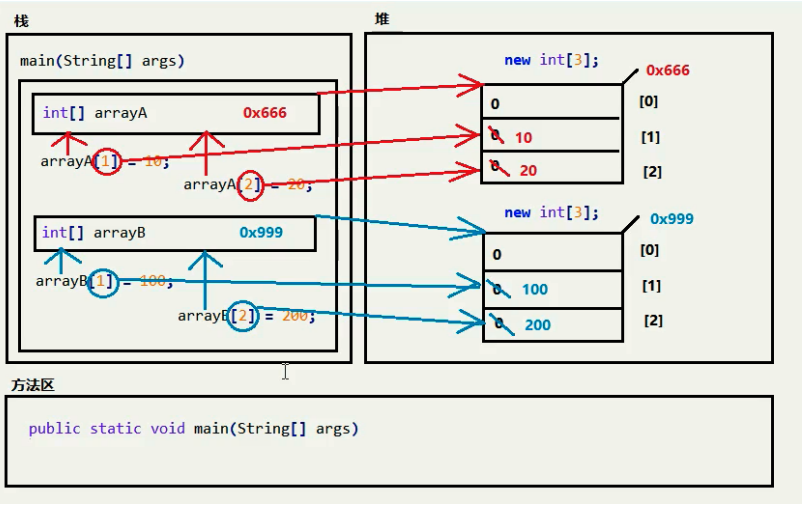
1.一个数组的内存图

```
public class Demo01ArrayOne {

    public static void main(String[] args) {
        int[] array = new int[3]; // 动态初始化
        System.out.println(array); // 地址值
        System.out.println(array[0]); // 0
        System.out.println(array[1]); // 0
        System.out.println(array[2]); // 0
        System.out.println("==============");

        // 改变数组当中元素的内容
        array[1] = 10;
        array[2] = 20;
        System.out.println(array); // 地址值
        System.out.println(array[0]); // 0
        System.out.println(array[1]); // 10
        System.out.println(array[2]); // 20
    }

}
```



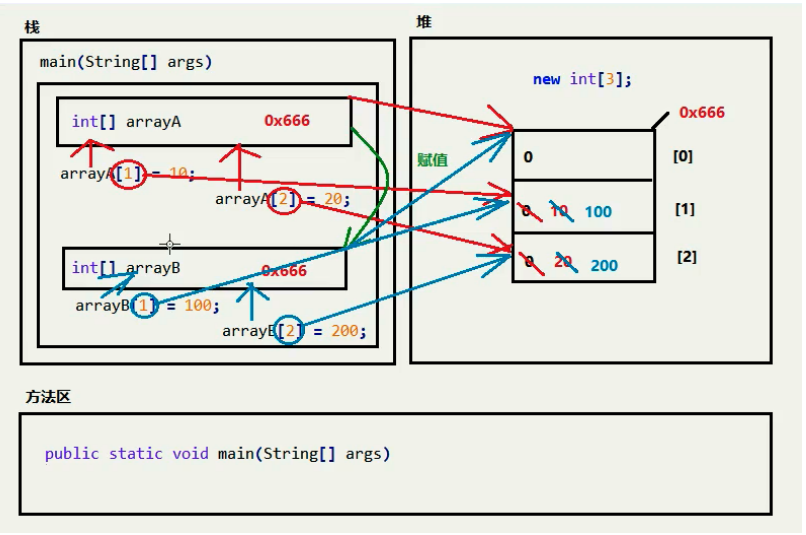
2.两个数组的内存图

```java
public class Demo02ArrayTwo {
    public static void main(String[] args) {
        int[] arrayA = new int[3];
        System.out.println(arrayA); // 地址值
        System.out.println(arrayA[0]); // 0
        System.out.println(arrayA[1]); // 0
        System.out.println(arrayA[2]); // 0
        System.out.println("==============");
        arrayA[1] = 10;
        arrayA[2] = 20;
        System.out.println(arrayA); // 地址值
        System.out.println(arrayA[0]); // 0
        System.out.println(arrayA[1]); // 10
        System.out.println(arrayA[2]); // 20
        System.out.println("==============");

        int[] arrayB = new int[3];
        System.out.println(arrayB); // 地址值
        System.out.println(arrayB[0]); // 0
        System.out.println(arrayB[1]); // 0
        System.out.println(arrayB[2]); // 0
        System.out.println("==============");
        arrayB[1] = 100;
        arrayB[2] = 200;
        System.out.println(arrayB); // 地址值
        System.out.println(arrayB[0]); // 0
```



3.两个引用指向同一个数组

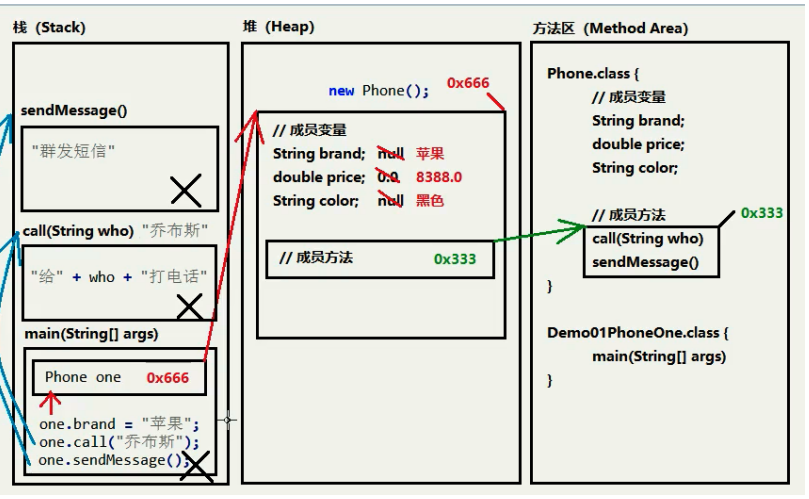
```java
public class Demo03ArraySame {
    public static void main(String[] args) {
        int[] arrayA = new int[3];
        System.out.println(arrayA); // 地址值
        System.out.println(arrayA[0]); // 0
        System.out.println(arrayA[1]); // 0
        System.out.println(arrayA[2]); // 0
        System.out.println("==============");
        arrayA[1] = 10;
        arrayA[2] = 20;
        System.out.println(arrayA); // 地址值
        System.out.println(arrayA[0]); // 0
        System.out.println(arrayA[1]); // 10
        System.out.println(arrayA[2]); // 20
        System.out.println("==============");

        int[] arrayB = arrayA;
        System.out.println(arrayB); // 地址值
        System.out.println(arrayB[0]); // 0
        System.out.println(arrayB[1]); // 10
        System.out.println(arrayB[2]); // 20
        System.out.println("==============");
        arrayB[1] = 100;
        arrayB[2] = 200;
        System.out.println(arrayB); // 地址值
        System.out.println(arrayB[0]); // 0
```



Java对象内存图

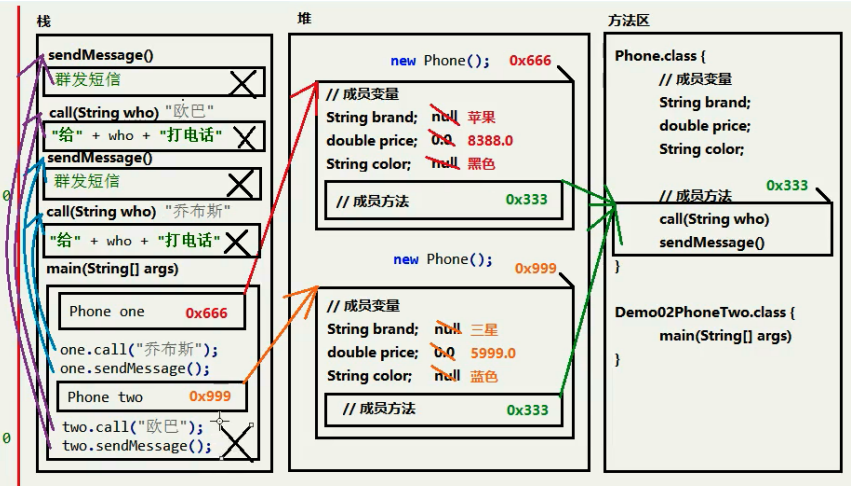
1.一个对象的内存图

```java
public class Phone {
    String brand; // 品牌
    double price; // 价格
    String color; // 颜色
    public void call(String who) {
        System.out.println("给" + who + "打电话");
    }
    public void sendMessage() {
        System.out.println("群发短信");
    }
}
public class Demo01PhoneOne {
    public static void main(String[] args) {
        Phone one = new Phone();
        System.out.println(one.brand); // null
        System.out.println(one.price); // 0.0
        System.out.println(one.color); // null
        one.brand = "苹果";
        one.price = 8388.0;
        one.color = "黑色";
        System.out.println(one.brand); // 苹果
        System.out.println(one.price); // 8388.0
        System.out.println(one.color); // 黑色
        one.call("乔布斯"); // 给乔布斯打电话
        one.sendMessage(); // 群发短信
    }
```



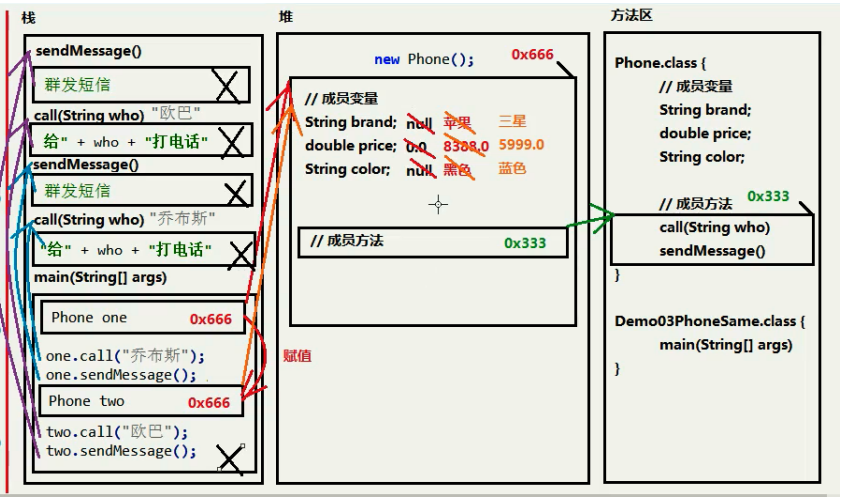
2.两个对象的内存图

```java
public class Demo02PhoneTwo {
    public static void main(String[] args) {
        Phone one = new Phone();
        System.out.println(one.brand); // null
        System.out.println(one.price); // 0.0
        System.out.println(one.color); // null
        one.brand = "苹果";
        one.price = 8388.0;
        one.color = "黑色";
        System.out.println(one.brand); // 苹果
        System.out.println(one.price); // 8388.0
        System.out.println(one.color); // 黑色
        one.call("乔布斯"); // 给乔布斯打电话
        one.sendMessage(); // 群发短信

        Phone two = new Phone();
        System.out.println(two.brand); // null
        System.out.println(two.price); // 0.0
        System.out.println(two.color); // null
        two.brand = "三星";
        two.price = 5999.0;
        two.color = "蓝色";
        System.out.println(two.brand); // 三星
        System.out.println(two.price); // 5999.0
        System.out.println(two.color); // 蓝色
        two.call("欧巴"); // 给欧巴打电话
```



3.两个引用指向同一个对象

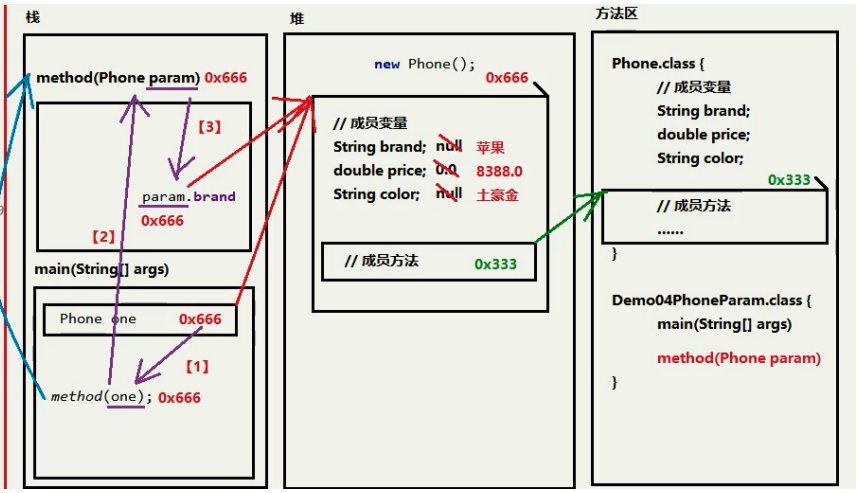
```java
public class Demo03PhoneSame {
    public static void main(String[] args) {
        Phone one = new Phone();
        System.out.println(one.brand); // null
        System.out.println(one.price); // 0.0
        System.out.println(one.color); // null
        one.brand = "苹果";
        one.price = 8388.0;
        one.color = "黑色";
        System.out.println(one.brand); // 苹果
        System.out.println(one.price); // 8388.0
        System.out.println(one.color); // 黑色
        one.call("乔布斯"); // 给乔布斯打电话
        one.sendMessage(); // 群发短信

        Phone two = one;
        System.out.println(two.brand); // 苹果
        System.out.println(two.price); // 8388.0
        System.out.println(two.color); // 黑色
        two.brand = "三星";
        two.price = 5999.0;
        two.color = "蓝色";
        System.out.println(two.brand); // 三星
        System.out.println(two.price); // 5999.0
        System.out.println(two.color); // 蓝色
        two.call("欧巴"); // 给欧巴打电话
```



4.使用对象类型作为方法参数

```java
public static void main(String[] args) {
    Phone one = new Phone();
    one.brand = "苹果";
    one.price = 8388.0;
    one.color = "土豪金";

    method(one); // 传递进去的参数其实就是地址值
}

public static void method(Phone param) {
    System.out.println(param.brand); // 苹果
    System.out.println(param.price); // 8388.0
    System.out.println(param.color); // 土豪金
}
```

当一个对象作为参数，传递到方法当中时，
实际上传递进去的是对象的地址值。



5.使用对象作为方法返回值

```java
public static void main(String[] args) {
    Phone two = getPhone();
    System.out.println(two.brand); // 苹果
    System.out.println(two.price); // 8388.0
    System.out.println(two.color); // 玫瑰金
}

public static Phone getPhone() {
    Phone one = new Phone();
    one.brand = "苹果";
    one.price = 8388.0;
    one.color = "玫瑰金";
    return one;
}
```

当使用一个对象类型作为方法的返回值时：
返回值其实就是对象的地址值。