Java 面向对象：

- 面向过程：

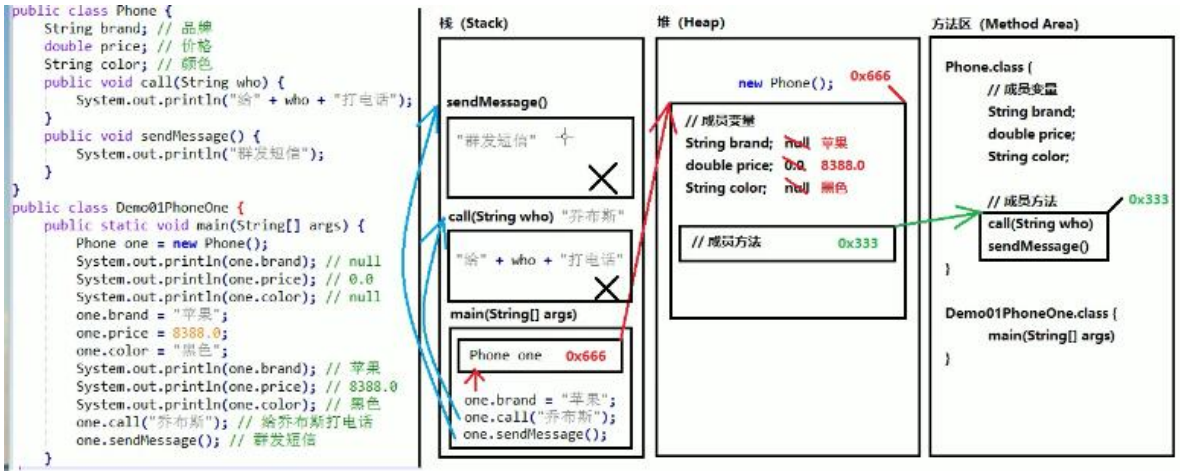
自己写代码实现功能

- 面向对象：

找一个对象来帮你做事

- 类：相关属性和行为的集合，是抽象的
- 对象：一类事物的实例，是具体的

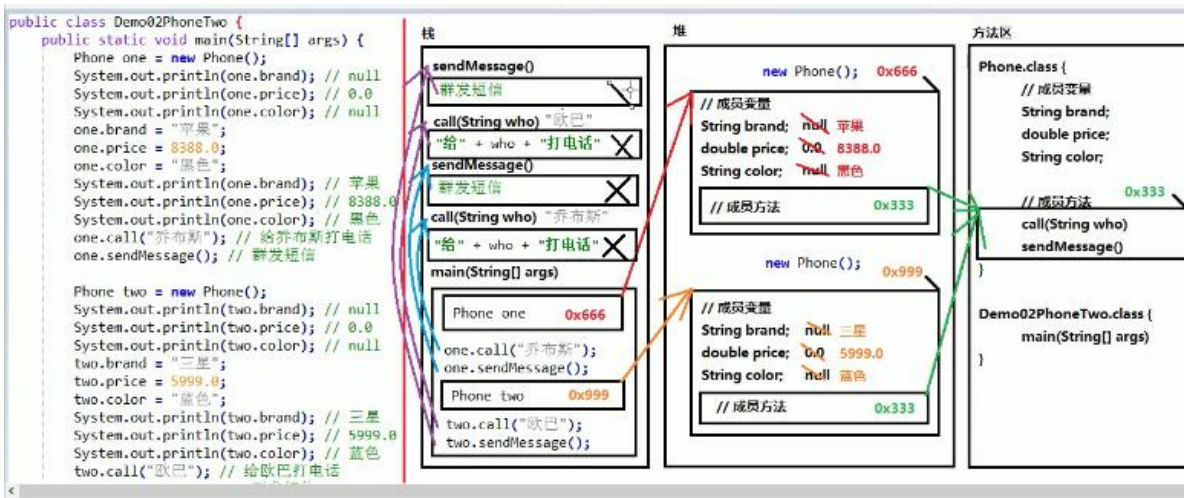
一个对象内存图：



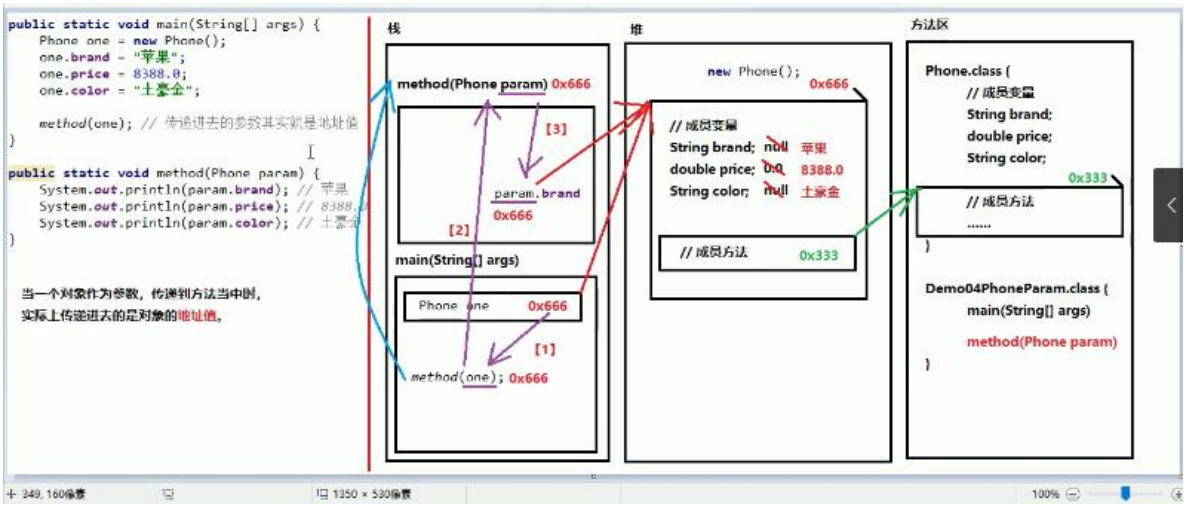
- 方法执行必须进栈
- 执行完出栈

shift +F6:统一更改名字

俩个对象使用同一个方法的内存



当一个对象做为参数，传进去的是对象的地址值：



局部变量和成员变量
1.定义的位置不一样
局部变量:定义在方法中
成员变量：直接定义在类中
2.作用的范围不一样
局部变量：只有方法当中可以使用
成员变量：整个类通用
3.默认值不一样
局部变量：没有默认值
成员变量：会有默认值
4.内存的位置不一样
局部变量：位于栈内存
成员变量：位于堆内存
5.生命周期不一样
局部变量：随着方法进栈而诞生，出栈而消失
成员变量：随着对象的创建而诞生，垃圾回收而消失

方法的参数是局部变量

面向对象之封装性：

面向对象三大特效：封装，继承，多态
1.方法就是一种封装
2.关键字`private` 也是一种封装
封装：将一些细节信息隐藏起来，对于外界不可见

当方法的局部变量和成员变量区分不开的时候，优先使用局部变量！this关键字可以解决这个问题。

通过谁调用，谁就是this。

常用的API(Application programming interface)

- 1.Scanner

```
public class DemoScanner {
    public static void main(String[] args) {
        Scanner scanner=new Scanner(System.in);

        System.out.println("请输入俩个字符：");
        int a=scanner.nextInt();
        int b=scanner.nextInt();

        int result=a+b;
        System.out.println("结果是："+result);
    }
}
```

- 2.佚名对象：

```
public class DemoAnoymous {
    public static void main(String[] args) {
        Person one =new Person();
       one.name="高圆圆";

       //匿名对象
        new Person().name="赵又提";
        new Person().show(); //显示空值
    }
}

public class Demo02Anoymous {
    public static void main(String[] args) {
//        Scanner sc=new Scanner();
//        int num=sc.nextInt();

        //匿名对象的使用方式
        int  num=new Scanner(System.in).nextInt();
        System.out.println("输入的是："+num);
    }
}
```

- 3.Random

```
2.创建
Random r=new Random();
3.使用
int num=r.nextInt();  范围是int的所有范围，有正负俩种
 */
import java.util.Random;

public class Demo01Random {
    public static void main(String[] args) {
        Random r=new Random();
        int num=r.nextInt();
        System.out.println(num);
    }

public class Demo03Random {
    public static void main(String[] args) {
        int num=5;
```

```
        Random r=new Random();
        for (int i = 0; i < 100; i++) {
            int result=r.nextInt(num)+1; //范围是 0-n-1   ->[1,n]
            System.out.println(result);
        }
    }
}
```

快捷键：100.fori ：for循环

- 4.Arraylist： 对象数组
  - 快捷键: sout -> System.out.println()

```
public class Demo02ArrayList {
    public static void main(String[] args) {
        //创建了一个ArrayList集合，集合名称是list ,里面装的全是String
        ArrayList<String> list =new ArrayList<>();
        String [] array=new String[5];
        System.out.println(list);
        //向集合当中添加数据 add方法
        list.add("赵丽颖");
        list.add("赵本山");
        list.add("古力娜扎");
        System.out.println(list);
        System.out.println(list.get(2));

    }
}
```

```
ArrayList常用方法:
public boolean  add(E e);向集合当中添加元素，参数的类型和泛型一致
public E  get(int index) ;从集合当中获取元素，参数是索引编号
public E remove(int index); 从集合当中删除元素，返回元素就是被删除的元素
public int size() :获取集合的尺寸长度，返回集合元素个数
对于ArrayList集合来说，添加动作一定是成功的（对于其他集合来说，add不一定成功，返回值有用）
```

ArrayList遍历快捷键：list.fori

```
public class Demo04ArraylistEach {
    public static void main(String[] args) {
        ArrayList<String> list=new ArrayList<>();
        list.add("迪丽热巴");
        list.add("古力娜扎");
        list.add("马尔真是");
        for (int i = 0; i < list.size(); i++) {
            System.out.println(list.get(i));
        }
    }
}
```

字符串String

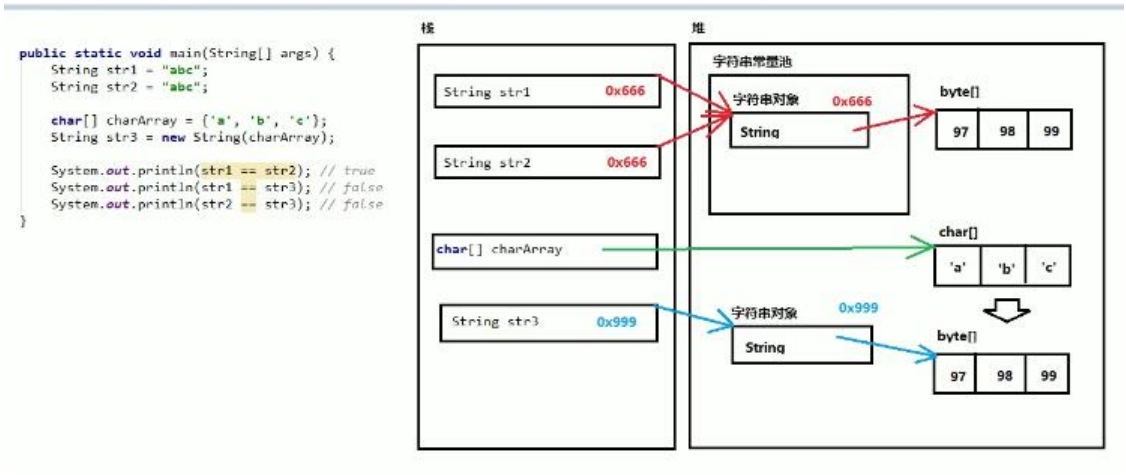
四种创建方式：

创建字符串的常见3+1种模式：
1.public String();创建一个空白字符串
2.public String(char[] array);根据字符数组的内容来创建对应的字符串
3.public String(Byte[] array);根据字节数组的内容，来创建对应的字符串。
4.一种直接创建: String str4="Hello"; //也是字符串的对象 */

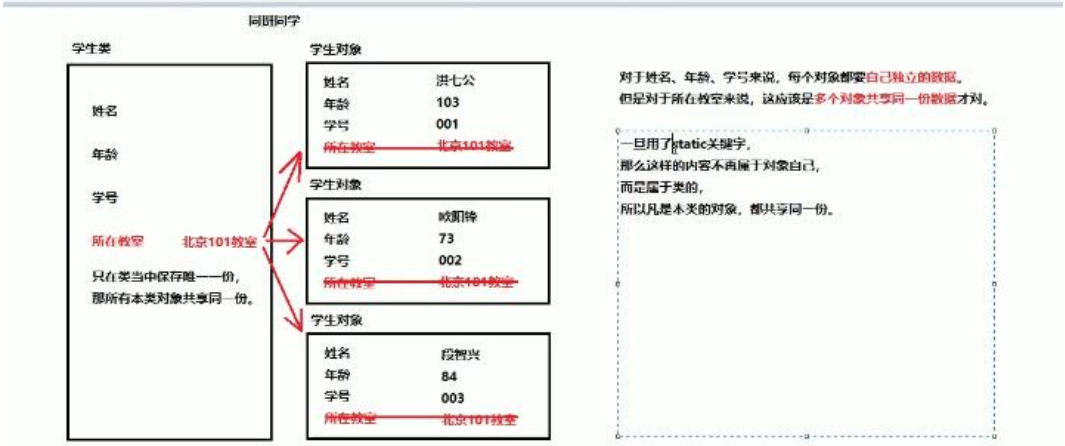
字符串常量池：

字符串常量池：程序当中直接写上双引号的字符串，就在字符串常量池中
对于基本类型来说：== 是进行数值的比较
对于引用类型来说：== 是【地址】值的比较

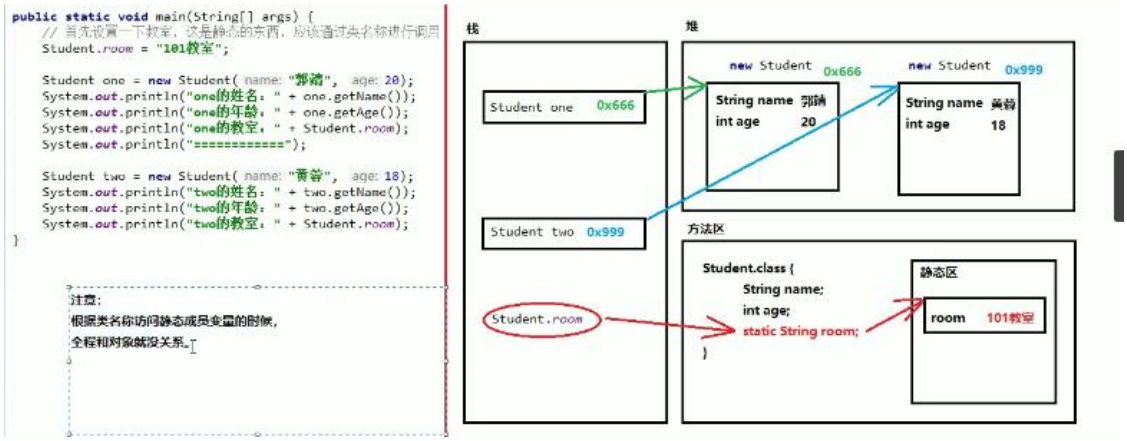
字符串常量池：



静态Static关键字



静态static内存图：




```
/*
题目：
使用Arrays将一个随机字符串进行升序排列，倒序打印
 */
public class Demo02Prac {
    public static void main(String[] args) {
        String str="fskjslkagk4rkfsj4534";
        //必须是数字才可以用Arrays
        char[] chars = str.toCharArray();
        Arrays.sort(chars);
        for (int i = chars.length - 1; i >= 0; i--) {
            System.out.print(chars[i]);
        }
    }

}
```

快捷键：
chars.forr ：逆序遍历chars

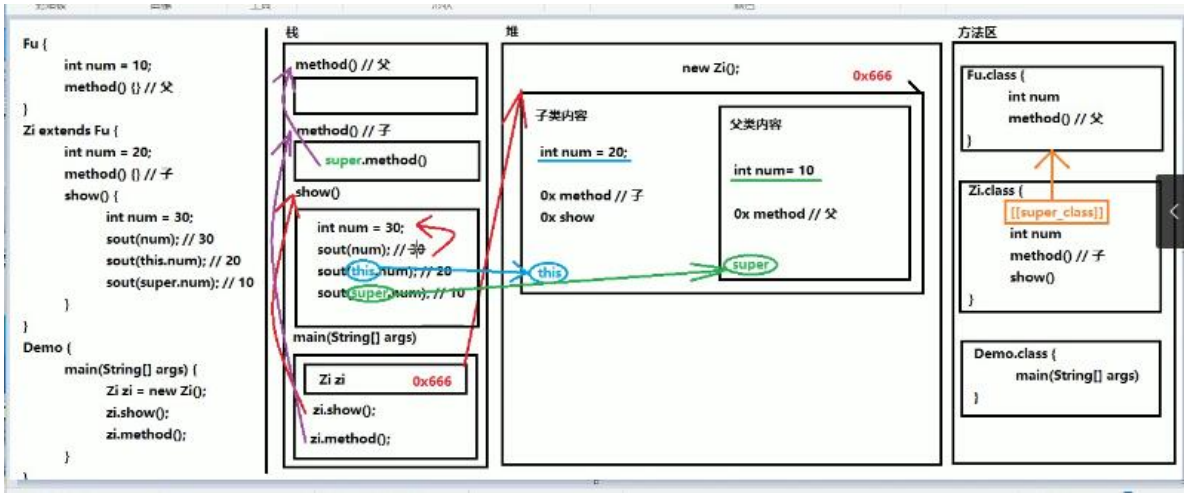
Math工具类：

```
public class Demo01Math {
    public static void main(String[] args) {
        //获取绝对值
        System.out.println(Math.abs(3.14));
        System.out.println(Math.abs(-6.999999));
        System.out.println("==================");
        //向上取整
        double mod=10.0;
        double num=21;
        System.out.println(Math.ceil(num/mod));
        System.out.println("==================");
        //向下取整
        System.out.println(Math.floor(30.9));
        //四舍五入
        System.out.println(Math.round(30.5));
        System.out.println(Math.round(30.4));
    }
}
```

继承性： 继承是多态的前提

继承主要解决的问题就是： 共性抽取

this和super 内存图



java继承的三个特点：