面向对象

万物皆对象

我们是怎么认识世界的?

人类从小就不断的接触到各种各类存在世界上的各种生物，然后通过事物的公共特性，将它们归类，所以以后就不会出现见到猫叫老虎。那么我们在现实生活中，是通过具体的某个事物归纳总结它们的公共特性然后产生类那么类就描述了该种事物的的共别。性，相当于造事物的图纸，我们可以根据这个图纸去做出具体的实体对象。

对象：在现实生活中存在具体的一个事物。;
类：实际就是对某种类型事物的共性属性与行为的抽取。
人类认识世界： 对象----→类。
在java中： 类 -----→对象。

使用计算机语言就是不断的在描述现实生活中的事物。
java中描述事物通过类的形式体现，类是具体事物的抽象，概念上的定义。
对象即是该类事物实实在在存在的个体。
类与对象的关系如图
SHAPE * MERGEFORMAT
可以理解为:
类就是图纸
汽车就是堆内存中的对象

面向对象的概述

“面向对象”(英语：Object Oriented,简称OO)是一种**以事物为中心**的编程思想。面向对象程序设计（英语：Object-oriented programming，缩写：OOP），是一种程序开发的方法。它将对象作为程序的基本单元，将程序和数据封装其中，以提高软件的重用性、灵活性和扩展性。

面向对象时相对于面向过程而已的（c则是一个典型的面向过程的语言），站在面向对象的角度去看问题，你则是对象的动作的指挥者。如果站在面向过程的角度去看问题，你则是动作的执行者。

面向对象与面向过程对比

“万物皆对象”。

1：买电脑
1：面向过程
1：查资料
2：电脑城砍价
3：被黑

4：痛苦归来
1：面向对象
1：找对象。老师
2：老师.砍价
3：老师.检测电脑
4：电脑成功购买
2：吃饭
1：面向过程
1：自己动手做
2：买菜
3：洗菜
4：煮饭炒菜
5：很难吃，浪费时间
2：面向对象
1：找专业对象
2：餐馆.点餐
3：餐馆，做饭
4：饭好吃，节约时间，精力
4：找对象
1：求介绍，相亲，找现成的对象。（面向对象的思想先找有的对象，直接拿来使用）
2：不满意，没有对象，自己造一个。（sun没有提供，自己造对象）

再例如：人开门，人开电视，人画园。

面向过程

强调的是功能行为，面向过程"是一种以过程为中心的编程思想。"面向过程"他们不支持丰富的"面向对象"特性（比如继承、多态），就是分析出解决问题所需要的步骤，然后用函数把这些步骤一步一步实现，使用的时候一个一个依次调用就可以了。面向过程在这一系列工作的执行中，强调的是工作的执行。

对象

对象（object）代表现实世界中可以明确标识的一个实体。例如：一个学生、一张桌子、一间教室，一台电脑都可以看做是一个对象。每个对象都有自己独特的状态标识和行为
对象的属性（attribute，或者状态（state）），学生有姓名和学号，该学生特有的姓名和学号就是该学生（对象）的属性。
对象的行为（behavior），是由方法定义，调用对象的一个方法，其实就是给对象发消息，要求对象完成一个动作。可以定义学生对象具备学习的行为。学生对象可以调用学习的方法，执行学习的动作

面向对象的特征

封装(encapsulation)
继承(inheritance)
多态(polymorphism)
开发的过程：其实就是不断的创建对象，使用对象，指挥对象做事情。
设计的过程：其实就是在管理和维护对象之间的关系.

使用java来描述事物

案例：通过Java语言定义一个汽车类，并生产出汽车，有颜色，轮胎个数， 有运行的功能。
分析：
如何描述现实世界中的事物，描述该事物的属性和行为，汽车具有颜色和轮胎数的属性，具备运行的行为。
如何使用Java语言进行转换?
根据对应关系：
属性：类中的成员变量
行为：类中的成员函数
那么定义Java类就是定义一个类的成员。汽车类具备的成员是：颜色，轮胎数，运行方法。
Car类定义流程：
使用class 关键字 定义类，
class空格 类名。类名就是标识符，命名规则，单词的首字母大写，多个单词的首字母大写。注意：不是规定，但是最好遵守
类名后紧跟一对{}表示类的开始和结束。
汽车有轮胎数 int num;
不需要给num初始化值，因为汽车轮胎数不确定，有4，有6，有8。
有颜色 String color
为什么使用String 例如定义颜色"红色"是字符串类型。
也不需要初始化值。
跑的行为(方法、函数) void run(){}
方法中执行输出语句。syso("跑啦。。。。");

```
public class Car {

    String color;// 成员变量
    int num; // 成员变量

    // 成员函数
    void run() {
        System.out.println(color + "的车，轮胎数：" + num + "个，跑起来了");
    }
}
```

对象的创建

创建Car对象

使用new关键词，就像new数组一样
需要给型的汽车起一个名字，car
变量都是有类型的，car属于什么类型，属于Car类型，叫做类类型
Car car=new Car();

图纸画好了，类定义成功了。如何根据图纸生产汽车，在Java中如何根据类来生产一个对象。

Java中生产汽车比较简单，通过一个关键字"new"， 通过 new Car(); 就在内存中产生了一个实体，汽车对象就生产出来了。

汽车对象生产出来后，还没有名字，为了方便使用，需要起一个名字。就用小写的c 来表示新车的名字。

Java中的变量都需要有类型。那么c 是什么类型呢？c 是Car类型，所以c也叫做类类型变量。

```
class CarDemo {
    public static void main(String[] args) {
        // java 中创建对象，使用new关键字。在堆内存中开辟了空间。产生了一个实体。
        Car c = new Car();
        //为了方便使用生产的汽车，就起了一个名字。
        //那么c是什么类型,是Car类型。叫做类类型变量。
        //注意 c是持有的引用，新生产的汽车并没有直接赋值给c，持有的只是一个引用。c就想电视遥控器一样。
        c.run();  //使用对象的功能。
    }
}
```

对象成员的调用

有了car对象，调用对象成员
成员变量
成员方法

```
public class CarDemo {
    public static void main(String[] args) {
        Car c = new Car();
        //对象名.成员变量名将返回成员变量中存储的数值
        int num=c.num;
        System.out.println(num);

        //对象名.成员变量名,也可以给成员变量赋值
        c.num = 4;
        c.color = "black";

        //对象名.成员方法();
        c.run();
    }
}
```

局部变量和成员变量

成员变量: 定义在类中变量
局部变量: 定义在方法中变量

成员变量与局部变量的区别:
应用范围
成员变量在整个类内都有效
局部变量只在其声明的方法内有效
生命周期
成员变量: 它属于对象，它随着对象的创建而创建，随着对象的消失而消失
局部变量: 使用完马上释放空间。

```
void show(int id){
    for(int i=0;i<10;i++){
        for(int j=0;j<10;j++){
            System.out.println(id);
        }
    }
}
```

这时候 id,i,j者是在方法内声明的，全是局部变量
j当里层for循环执行它的生命周期开始，当里层for结束，j消失
i当外层for循环执行它的生命周期开始，当外层for结束，j消失
id在方法被调用时开始，方法结束时，id消失.

存储位置　成员变量属于对象，它存储在堆内，堆内的实体，当没有引用指向其时，才垃圾回收清理 局部变量存在栈内存中，当不在使用时，马上就会被释放。
初始值
成员变量它存储在堆中，如果没有赋初值，它有默认值。

整数byte、short、int、long =0;
char='\uoooo';
boolean =flase;
String =null;
类类型 =null;
数组 =null;

局部变量，如果要想使用必须手动初始化.
方法中，参数列表中，语句中。
必须给初始化值，没有初始值，不能使用
在栈内存中

内存分析

案例一:

```
//汽车
class Car {
    //汽车应该具备的属性
    int num;
    //汽车具备的颜色
    String color;
    //汽车跑的行为
    public void run(){
```

```
            System.out.println(num+"轮子的汽车跑起来啦");
        }
    }
    public class CarDemo{

        public static void main(String[] args)
        {   //创建实体，并且给该实体起一个名字
            Car c = new Car();
            c.color = "red";
            c.num = 4;
            c.run();//指挥车进行运行。调用格式：对象.对象成员

        }
    }
```

(图1)

案例二分析:

```
    public static void main(String[] args)
    {   //创建实体，并且给该实体起一个名字
        Car c = new Car();
        Car c1 = new Car();
        c.color = "red";
        c1.num = 4;
        System.out.println(c1.color);
        c.run();//指挥车进行运行。调用格式：对象.对象成员

    }
```

内存图:

INCLUDEPICTURE "../../Local%20Settings/Temp/ksohtml/wps_clip_image-17487.png" * MERGEFORMAT

(图二)

案例三

```
    public static void main(String[] args)
    {   //创建实体，并且给该实体起一个名字
        Car c = new Car();
        Car c1 = c;
        c.color = "red";
        c1.num = 4;
        c1.color = "green";
        System.out.println(c1.color);
        c.run();//指挥车进行运行。调用格式：对象.对象成员

    }
```

内存图三

INCLUDEPICTURE "../../Local%20Settings/Temp/ksohtml/wps_clip_image-31144.png" * MERGEFORMAT

(图三)

面向对象练习

1：完成修理汽车的功能
    2：分析
        1：面向对象的思想思考需要什么对象
            1：汽车
                1：汽车有轮子
                2：有颜色
                3：有名字
                4：有运行的方法
            2：汽车修理厂
                1：有名字
                2：有地址
                3：有修理汽车的方法
    3：代码实现
        1：定义汽车类
        2：定义汽车修理厂类
    4：测试代码
        1：创建汽车对象
        2：汽车少了轮子无法运行。
        3：创建汽车修理厂
            1：设置厂名
            2：设置地址
        3：将汽车拖进修理厂
            1：运行汽车修理厂的修车方法，修理汽车
        4：取车
            1：开走汽车

```java
package oop01;

/*
 面向对象之练习
 完成修理汽车的功能

 汽车类
 汽车修理厂类
 名字，地址， 修理汽车的功能

 */
public class Demo1 {

    public static void main(String[] args) {
        SCar sc = new SCar();
        sc.run();
    //将汽车轮子改为3个
        sc.num = 3;
        sc.run();

        CarFactory cf = new CarFactory();
        cf.name = "幸福修理厂";
        cf.addr = "天河区棠东东路预付科贸园a栋206";

        cf.repairCar(sc);
        sc.run();

        System.out.println();
    }
}

class SCar {
    String name = "smart";
    String color = "red";
    int num = 4;

    void run() {
        if (num < 4) {
            System.out.println("汽车坏了，赶紧修理吧。。。。");
        } else {
            System.out.println(name + ":" + color + ":" + num + ":跑起来
了。。。。");
        }

    }

}

class CarFactory {
    String name;
    String addr;

    void repairCar(SCar sc) {
        sc.num = 4;
        System.out.println("汽车修好了。。。。");
```

匿名对象

2.1匿名对象：没有名字的实体，也就是该实体没有对应的变量名引用。

2.2匿名对象的用途

1，当对象对方法进行一次调用的时候，可以使用匿名对象对代码进行简化。

为什么只对方法，而不调用属性呢？因为匿名对象调用属性没意义。

如果对象要多成员进行多次调用，必须给对象起个名字。不能在使用匿名对象。

2，匿名对象可以实际参数进行传递。

2：匿名对象的简单演示

```
1：new Car().run();
```

3：内存结构图

```
1：new Car().num=5;
2：new Car().clor="blue";
```

两个new 是两个不同的对象，在堆内存中有不同的空间，相互不相互干扰。

4：匿名对象的使用

```
1：当只使用一次时可以使用匿名对象。执行完毕到;后该对象就变成了垃圾。
new Car().run();
```

2:执行方法时，可以将匿名对象作为实际参数，传递进去。

5:修黑车

1：需求

将小汽车改装成3个轮子的黑车。

1：汽车类。

2：汽车修理厂

```
/*
 匿名对象
 匿名信
 修黑车
 汽车类
 黑车厂类
 把汽车的改成黑色3轮车。

 */
class Car {
    String name = "smart";
    String color = "red";
    int num = 4;

    void run() {
        System.out.println(name + ":" + color + ":" + num + ":跑起来
了。。。。");
    }

}

class BlackCarFactory {
    String name;
    String addr;

    Car repairCar(Car c) {
        c.num = 3;
        c.color = "黑色";
        System.out.println("改装成功啦。。。");

    }
}

class Demo1 {

    public static void main(String[] args) {

        BlackCarFactory bcf = new BlackCarFactory();
        bcf.name = "幸福修理厂";
        bcf.addr = "天河区棠东东路御富科贸园a栋206";

        // 非匿名对象
        Car c = new Car();
        c.run();
        // 改装
        bcf.repairCar(c);
        // 取车
        c.run();

        // 匿名对象一,只使用一次:
        // 如下创建了2个对象
        /*
         * new Car().run();
         *
```

练习:
请问输出的name属性是什么? new Perosn == new Person()相等吗?

总结:

1. 匿名对象设置的属性永远无法获取? 没有引用变量指向那个对象。
2. 任何两个匿名对象使用==比较，永远返回false。
3. 匿名对象主要应用于实参。

封装

我们日常使用的电脑主机，把cpu、内存、主板等等都封装到机箱里面去。假如没有机箱的话的出现什么问题，主机、主板全部都散落在一处，然后开机没有开机按钮，那么需要我们直接操作接跳线才能把电脑开启。这样子的话假如操作不慎的话，会让机器损坏危险,那么假如用机箱封装起来的话，那么就不需要这样子做了。体现了封装的---安全特性。
你拿电脑去加内存，可以直接给电脑给维修的人，等他加好内存了之后。你拿到的还是那个机箱，里面发生了怎样子的变化你并不知道。封装的第二个好处-将变化隔离。

在机箱里面提供一个开机按钮，而不需要你直接使用跳线开机的话，体现了封装的—便于使用的特性。

只要机箱提供了一个开机的功能，然后无论这个机箱拿到哪里去，都可以使用这个开机的功能.体现了封装的---提供重复性的特性。

没有封装

模拟问题
描述`Employee`类。定义姓名，工号，性别的成员变量，和工作的方法。成员使用`public`修饰。
创建`Employee`对象，对象.成员的方式进行赋值。最后该对象调用工作方法。
总结：如果不使用封装，很容易赋值错误，并且任何人都可以更改，造成信息的 不安全。
问题解决：使用封装

```java
package oop01;

public class EmployeeDemo {
    public static void main(String[] args) {
        // 创建对象
        Employee jack = new Employee();

        // 进制通过类名.成员的形式调用成员。初始化实例变量
        jack.name = "jack";
        jack.id = "123456";
        jack.gender = "男";

        // 调用成员方法
        jack.work();
        System.out.println();

        // 传入非法的参数
        jack.gender = "不是男人";
        jack.work();

    }
}

class Employee {
    String name;
    String id;
    String gender;

    public void work() {
        System.out.println(id + ":" + name + ":" + gender + " 努力工作中！！！");
    }
}
```

封装的实现

1：设置类的属性为private(关键字)，不能使用对象名.属性名的方式直接访问对象的属性。

```java
package oop01;

public class EmployeeDemo {
    public static void main(String[] args) {
        // 创建对象
        Employee jack = new Employee();

        //编译报错
        jack.name = "jack";
        jack.id = "123456";
        jack.gender = "男";



        // 编译报错
        jack.gender = "不是男人";
        jack.work();

    }
}

class Employee {
    //使用了private修饰了成员变量
    private String name;
    private String id;
    private String gender;

    public void work() {
        System.out.println(id + ":" + name + ":" + gender + " 努力工作中！！！");
    }
}
```

问题：

1：为什么之前可以通过对象名.属性名的方式访问?

2：public 成员修饰符，公共的谁都可以访问。

3：private 成员修饰符，私有的，只有自己可以访问。

2：修改Employee类 性别的修饰符修改为private

1：编译不通过

2：private修饰的成员在自己所在的类中可以使用，在类外边不可以使用。

3：Employee类的gender的修饰符修改为private后，无法再类外调用，那么如何给gender设置值?

1：对外提供公开的用于设置对象属性的public方法

1：设置set

2：获取get

2：在set方法中加入逻辑判断，过滤掉非法数据。

3：将所有的成员变量封装加上private，提供get、set方法

```java
package oop01;

public class EmployeeDemo {
    public static void main(String[] args) {
        // 创建对象
        Employee jack = new Employee();

        // 调用公有方法，给成员变量赋值。
        jack.setId("007");
        jack.setName("jack");
        jack.setGender("男xx");

        // 获取实例变量的值
        System.out.println(jack.getGender());
        System.out.println(jack.getId());
        System.out.println(jack.getName());

        // 调用成员方法
        jack.work();

    }
}

class Employee {
    private String name;
    private String id;
    private String gender;

    // 提供公有的get set方法
    public String getName() {
        return name;
    }

    public void setName(String n) {
        name = n;
    }

    public String getId() {
        return id;
    }

    public void setId(String i) {
        id = i;
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gen) {
        if ("男".equals(gen) || "女".equals(gen)) {
            gender = gen;
        } else {
            System.out.println("请输入\"男\"或者\"女\"");
        }
```

封装的好处

1：隐藏了类的具体实现
2：操作简单
3：提高对象数据的安全性

封装练习

练习：描述一个计算器类

```java
/**
 Demo9描述一个计算器类。
 */
// 0. 使用词霸确定类名
class Calculator
{
    // 1. 查看具体的计算器对象抽取所有计算器具有的共同属性
    public String name = "我的计算器我做主";
     public double num1;
    public double num2;
    public char option;
    // 2. 查看具体的计算器对象抽取所有计算器具有的共同功能
     // 2.1  定义接受数据的功能函数
    public void init( double a , char op , double b  ){

        num1 = a;
        option  = op;
        num2 = b;
    }
    // 2.2 定义计算的功能
    public void calculate(){

       switch ( option )
       {
       case '+': System.out.println(  name + " : " + num1 + " + " +
num2 + " = " + ( num1 + num2 ) );
                 break;
        case '-': System.out.println(  name + " : " + num1 + " - " +
num2 + " = " + ( num1 - num2 ) );
                 break;
        case '*': System.out.println(  name + " : " + num1 + " * " +
num2 + " = " + ( num1 * num2 ) );
                 break;
       case '/': {
                 if( num2 != 0 )
                    System.out.println(  name + " : " + num1 + " / " +
num2 + " = " + ( num1 / num2 ) );
                else
                     System.out.println("除数不能为0!");
                 break;
                 }
        case '%': {
                 // 1.处理结果的符号问题，使得结果的符号满足数学的要求
                // 2.解决NaN的问题
                 System.out.println(  name + " : " + num1 + " % " +
num2 + " = " + ( num1 % num2 ) );
                 break;
                 }
        default : System.out.println("你在捣乱，我不理你，气死你.....");

       }

    }
}
class Demo9
```

构造方法

1.我们人出生的时候，有些人一出生之后再起名字的，但是有些人一旦出生就已经起好名字的。那么我们在java里面怎么在对象一旦创建就赋值呢?

构造方法的作用

构造方法作用：对对象进行初始化.

构造函数与普通的函数的区别

一般函数是用于定义对象应该具备的功能。而构造函数定义的是，对象在调用功能之前，在建立时，应该具备的一些内容。也就是对象的初始化内容。
构造函数是在对象建立时由jvm调用, 给对象初始化。一般函数是对象建立后，当对象调用该功能时才会执行。
普通函数可以使用对象多次调用，构造函数就在创建对象时调用。
构造函数的函数名要与类名一样，而普通的函数只要符合标识符的命名规则即可。
构造函数没有返回值类型。

构造函数注意的细节

1. 当类中没有定义构造函数时，系统会指定给该类加上一个空参数的构造函数。这个是类中默认的构造函数。当类中如果自定义了构造函数，这时默认的构造函数就没有了。
   备注：可以通过javap命令验证。
2.在一个类中可以定义多个构造函数，以进行不同的初始化。多个构造函数存在于类中，是以重载的形式体现的。因为构造函数的名称都相同。

```
class Perosn{
    private int id;

    private String name;

    private int age;

    public Perosn(){
        cry();
    }
    public Perosn(int id, String name, int age) {
        this.id = id;
        this.name = name;
        this.age = age;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }


    public void cry(){
        System.out.println("哇哇哇....");
    }

}
```

问题：要求每个小孩出生都会哭，这份代码有两个构造函数，如果需要每个小孩出生都要哭的话，那么就需要在不同的构造函数中都调用cry()函数，但是这样子的话造成了代码重复问题，那么怎么解决呢？构造代码块。

构造代码块

构造代码块作用：给所有的对象进行统一的初始化。

```
class Perosn{
    private int id;

    private String name;

    private int age;

    {
        cry();// 每个Person对象创建出来都会执行这里的代码
    }

    public Perosn(){
        cry();
    }
    public Perosn(int id, String name, int age) {
        this.id = id;
        this.name = name;
        this.age = age;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }


    public void cry(){
        System.out.println("哇哇哇....");
    }

}
```

2：作用

1：给对象进行初始化。对象一建立就运行并且优先于构造函数。

2：与构造函数区别

1：构造代码块和构造函数的区别，构造代码块是给所有对象进行统一初始化，　构造函数给对应的对象初始化。

2：构造代码块的作用：它的作用就是将所有构造方法中公共的信息进行抽取。

例如孩子一出生统一哭

```
class Boy {

    String name;
    int age;
    String gender;
      // 构造代码块,给所有对象进行初始化。
    {
        System.out.println("哭。。。");
    }

    Boy() {
        System.out.println("无参构造");
    }

    Boy(String n, int a, String g) {
        name = n;
        age = a;
        gender = g;
        System.out.println("有参构造");
    }

    void run() {
        System.out.println("跑...");
    }

}

class Demo9 {

    public static void main(String[] args) {

        System.out.println();
        Boy b = new Boy();

        Boy b2 = new Boy("jack", 1, "男");

    }
}
```

this关键字

疑问：创建的p对象为什么没有值。
解答：name与age在指定的构造函数里面已经存在，当name=name这个语句执行的时候，如果jvm在该方法内能寻找到该变量，则不会去寻找成员变量，那么要想指定给成员变量或对象的属性进行初始化赋值，那么必须指定name是成员属性。

this 的概述

this关键字代表是对象的引用。也就是this在指向一个对象，所指向的对象就是调用该函数

的对象引用。

1：没有this会出现什么问题

1：定义Person类

1：有姓名年龄成员变量，有说话的方法。

2：定义构造方法，无参的，多个有参的。都要实现。

```
class Person {

    String name;
    int age;
    //无参数构造函数
    Person() {
        System.out.println("这是无参的构造函数");
    }

    //有参数构造函数
    Person(int a) {
        age = a;
        System.out.println("有参构造1");
    }
    //有参数构造函数
    Person(String n) {
        name = n;
        System.out.println("有参构造2");
    }
    //有参数构造函数
    Person(int a, String n) {
        age = a;
        name = n;
        System.out.println("有参构造");
    }

    //普通函数
    void speak() {
        System.out.println("hah");
    }
}
```

2；假设定义40个成员变量，第一个有参构造初始化20个变量，第二个有参构造需要初始化40个变量。

1：第二个有参构造想要使用第一个有参构造。

2：成员函数相互之间可以调用。构造函数可以吗?

3：编译失败，那么构造函数之间应该存在相互调用的模式。this就可以完成这个工作。

```
class Person {
    String name;
    int age;

    Person() {

    }
    Person(String n){
        name=n;
    }
    Person(String n, int a) {
         //编译报错
        Person(n);
        age = a;
    }
}
```

3：总结：实际工作中，存在着构造函数之间的相互调用，但是构造函数不是普通的成员函数，不能通过函数名自己接调用

所以sun公司提供this关键字。

2：this是什么

1：在构造函数中打印this

2：创建对象，打印对象名p

3：this和p是一样的都是内存地址值。

4：this代表所在函数所属对象的引用。

```
class Student {
    String name;
    String gender;
    int age;

    Student() {

    }

    Student(String name) {
        this();
        this.name = name;
    }

    Student(String name, String gender, int age) {
        this(name);
        System.out.println(this); // Student@c17164

        this.gender = gender;
        this.age = age;
    }

    void speak() {
        run();
        System.out.println("姓名: " + name + " 性别: " + gender + " 年
龄: " + age
                + " 哈哈!!! ");
    }

    void run() {
        System.out.println("run.....");
    }

}

class Demo2 {

    public static void main(String[] args) {

        Student p = new Student("jack", "男", 20);
        System.out.println(p); // Student@c17164

        Student p2 = new Student("rose", "女", 18);
        System.out.println(p2);

        p.speak();

    }
}
```

3: 递归构造函数调用

1: 构造函数的相互调用

在编译时期会报错

```
class Student {
    String name;
    String gender;
    int age;
     //构造函数见相互调用
    Student() {
        this(null);
    }
     //构造函数见相互调用
    Student(String name) {
        this();
        this.name = name;
    }

    Student(String name, String gender, int age) {
        this(name);
        this.gender = gender;
        this.age = age;
    }

    void speak() {
        run();
        System.out.println("姓名: " + name + " 性别: " + gender + " 年
龄: " + age
                + " 哈哈!!! ");
    }

    void run() {
        System.out.println("run.....");
    }

}
```

4: this只能在非静态中（没有static修饰的）函数使用
5: 构造函数间相互调用必须放在构造函数的第一个语句中，否则编译错误
6: 可以解决构造函数中对象属性和函数形参的同名问题。

作业

面向对象的理解并举例?
类与对象之间的关系?
如何对类进行分析，如果创建自定义类对象，并如何指挥对象做事情?
对象的内存分布图?
成员变量和局部变量的区别?
私有的使用。
构造函数和一般函数的区别?
构造函数什么时候用?

构造代码块的作用?
this关键字的特点和使用以及应用场景?

汽车C

汽车B

汽车A

图纸