

| 今天学习目标

- * 理解面向对象概述
 - * 软件开发过程的方法论
 - * 封装，继承，多态
 - * OOA,OOD,OOP
- * 理解类与对象
 - * 类抽象
 - * 对象具体
- * 掌握无参的方法
- * 掌握有参的方法
- * 掌握为什么要使用包和包的命名规则
 - * import
- * 能够使用eclipse生成注释文档
 - * javadoc
- * 区别基本数据类型和引用数据类型作为方法参数（画堆栈图）

* 回顾

* 冒泡排序

* 从小到大排序，从前到后两两比较，假如前面大于后面，交换

```
* for(int x=0;x<data.length-1;x++){  
    for(int i=0;i<data.length-1-x;i++){  
        if(data[i]>data[i+1]){  
            int temp=data[i];  
            data[i]=data[i+1];  
            data[i+]=temp;  
        }  
    }  
}
```

```
}
```

* 选择排序

* 从当前中选出最小，与初始化位置交换（怎么样找最小下标）

```
* for(int x=0;x<data.length-1;x++){  
    int min=x;  
    for(int i=min+1;i<data.length;i++){  
        if(data[min]>data[i]){  
            min=i;  
        }  
    }  
    // 与初始化位置交换  
    int temp=data[x];  
    data[x]=data[min];  
    data[min]=temp;  
}
```

* 对比排序性能

* Arrays.sort>选择排序>冒泡排序

* 二分查找

* 前提排序好，拆半查找

```
* int from=0 ;  
    int to=data.length-1;  
    int mid=(from+to)/2;  
    while(from<=to){  
        if(data[mid]==key){  
            return min;  
        }else if(data[mid]>key){  
            // 左  
            to=mid-1;
```

```

        mid=(from+to)/2;

    }else if ( data[mid]<key ) {

        // 右

        from=mid+1;

        mid=(from+to)/2;

    }

}

return -1;

```

* 理解面向对象概述

* 面向对象是软件开发过程的方法论。

* 思维转换，概念，学习面向对象语法

* Java 编程思想

* 面向对象与面向过程区别

* 类的使用者和类编写者之间区别

* 业务场景（报销流程）



* 面向过程

```

1 public static void main(String[] args) {
2     System.out.println("添加写报销");
3     System.out.println("财务审核");
4     System.out.println("主管审核");
5     System.out.println("总经理审核");
6     System.out.println("财务发送报销款");

```

```
7     }
```

* 面向对象

小黑正在填写报销单
小黑提出了报销
小白审核报销
小黑的报销审核通过
小红审核报销
小黑的报销审核通过
小明审核报销
小黑的报销审核通过
给： 小黑报销

```
1 package com.lg.test;
2
3 public class Emp {
4     // 姓名
5     public String name;
6     // 性别
7     public char sex;
8     // 工号
9     public String empId;
10
11
12     // 报销
13     public void claimingExpenses() {
14         System.out.println(name+"正在填写报销单");
15         System.out.println(name+"提出了报销");
16     }
17 }
18
19 package com.lg.test;
20
21 /**
```

```
22  * @author xiaozhao
23  *   财务
24  */
25 public class Accounter extends Emp{
26
27     /**
28      * 审核报销: auditAndReimbursement
29      */
30     public void auditAndReimbursement(String name) {
31         System.out.println(this.name+"审核报销");
32         System.out.println(name+"的报销审核通过");
33     }
34
35     public void sendAndReimbursement(String name) {
36         System.out.println("给: "+name+"报销");
37     }
38 }
39
40 package com.lg.test;
41
42 /**
43  * @author xiaozhao
44  *   alt+shift+j
45  *   主管
46  */
47 public class Manager extends Emp {
48
49     /**
50      * 审核报销: auditAndReimbursement
51      */
52     public void auditAndReimbursement(String name) {
53         System.out.println(this.name+"审核报销");
54         System.out.println(name+"的报销审核通过");
55     }
56 }
57
58 package com.lg.test;
59
60 public class President extends Emp{
61     /**
```

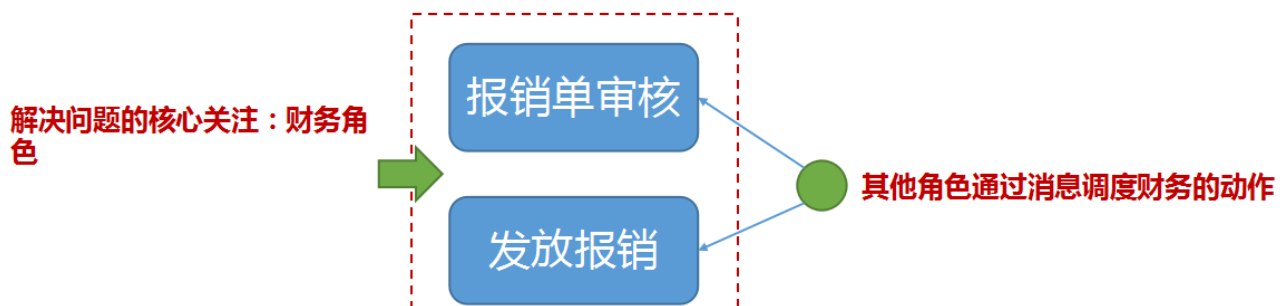
```
62     * 审核报销: auditAndReimbursement
63     */
64     public void auditAndReimbursement(String name) {
65         System.out.println(this.name+"审核报销");
66         System.out.println(name+"的报销审核通过");
67     }
68
69     // ...
70 }
71
72 package com.lg.test;
73
74 public class Test3 {
75     public static void main(String[] args) {
76         // 填写报销单
77         Emp emp=new Emp();
78         emp.empId="100001";
79         emp.sex='男';
80         emp.name="小黑";
81         emp.claimingExpenses();
82         // 财务审核
83         Accounter accounter=new Accounter();
84         accounter.empId="100002";
85         accounter.sex='女';
86         accounter.name="小白";
87         accounter.auditAndReimbursement(emp.name);
88         //主管审核
89         Manager manager=new Manager();
90         manager.empId="100003";
91         manager.sex='女';
92         manager.name="小红";
93         manager.auditAndReimbursement(emp.name);
94         // 总经理审核
95         President president=new President();
96         president.empId="100000";
97         president.sex='男';
98         president.name="小明";
99         president.auditAndReimbursement(emp.name);
100        // 财务发送报销款
101        accounter.sendAndReimbursement(emp.name);
```

```
102     }  
103 }  
104  
105
```

- * 可以发现有两个和财务相关的任务，
如果思考核心在过程，
那么这两个任务将被分散编写，
增加了代码的维护难度，不符合人类思考方式

* 面向对象程序设计是：

- * 将数据及对数据的操作**封装**在一起，成为一个不可分割的整体
- * 同时将具有相同特征的对象**抽象**成一种新的数据类型---类
- * 通过对象间的消息传递使整个系统运转，通过类的**继承**实现代码重用。



* 面向对象的方法学，就是使我们分析、设计和实现一个系统的方法

* OOA：面向对象的分析 (Object-Oriented Analysis)

* OOD：面向对象的设计 (Object-Oriented Design)

* OOP：面向对象的编程 (Object Oriendted Programming)

* 面向对象最重要的三大特征是：

* **封装**

* **继承**

* **多态**

* 美女是多态

* 理解类与对象

* 对象

* 万物皆对象



* 身边的对象

收银员

员工号—10008

姓名—小白

部门—财务部

操作:

收款

打印账单

顾客

姓名—小黑

年龄—22

体重—60kg

操作:

购买商品



* 对象的特征——属性，方法

* 属性——对象具有的各种特征

* 每个对象的每个属性都拥有特定值

* 例如：小黑和小白的年龄、姓名不一样

* 方法——对象执行的操作

* 总结：对象---->用来描述客观事物的一个实体，由一组属性和方法构成



* 类

* 具有相同属性和方法的一组对象的集合

* 类是对象的类型

* 不同于基本数据类型（int）：具有方法

* 类是抽象的概念，对象是具体实体

* 例如“人”是类，具体有小白，小黑，是对象

* Java程序的类都由class定义

* Java程序是由类构成的

```
1 package com.hx.test;  
2  
3 public class HelloWorld {  
4     public static void main(String[] args) {  
5         System.out.println("HelloWorld");  
6     }  
7 }  
8
```

定义类

类名：规则每个字母单词大写

* 类将现实世界中的概念模拟到计算机程序中

* public class 类名 {

//定义属性部分

属性1的类型 属性1;

属性2的类型 属性2;

...

属性n的类型 属性n;

//定义方法部分

方法1;

方法2;

...

方法m;

}

* 业务场景

* 1 编写学生类，输出学生相关信息

* 学生类

* 属性

* 姓名

* 年龄

* 班级

* 爱好

* 方法

* 显示学生信息

```

2
3 public class Student { → 1 定义类的名字
4     public String name; //姓名
5
6     public int age; //年龄
7
8     public String classNo; //班级 → 2 声明属性
9
10    public String hobby; //爱好
11
12
13    //显示学生信息
14    public void showInfo() {
15        System.out.println(name+"今年"+age+"岁在"+classNo+",他的爱好是"+hobby);
16    }
17 }
18

```

```

1 package com.lg.test;
2
3 public class Student {
4     public String name; //姓名
5
6     public int age; //年龄
7
8     public String classNo; //班级
9
10    public String hobby; //爱好
11
12
13    //显示学生信息
14    public void showInfo() {
15        System.out.println(name+"今年"+age+"岁在"+classNo+",他的爱好是"+hobby);
16    }
17 }
18

```

* 使用对象

```

2
3 public class Test1 {
4     public static void main(String[] args) {
5         Student s1=new Student();
6         s1.name="小黑";
7         s1.age=20;
8         s1.classNo="大圣一班";
9         s1.hobby="游泳";
10        s1.showInfo();
11    }
12 }
13

```

1 使用对象：类型 对象名称=new 类型();

2 使用对象属性或者方法：对象名称. 属性或者方法

```

1     public static void main(String[] args) {
2         Student s1=new Student();
3         s1.name="小黑";
4         s1.age=20;
5         s1.classNo="大圣一班";
6         s1.hobby="游泳";
7         s1.showInfo();
8     }

```

* 2 一个景区根据游人的年龄收取不同价格的门票。

请编写游人类，根据年龄段决定能够购买的门票价格并输出

年龄段：大于等于18岁小于等于60收15块，其余免费

游人类
姓名
年龄
显示姓名及门票价格

请输入姓名：小黑

请输入年龄：10

小黑的年龄为：10，免费

请输入姓名：小明
请输入年龄：20
小明年龄为20, 价格为15元

请输入姓名：小白
请输入年龄：65
小白的年龄为：65，免费

```
3 public class Visitor {  
4     public String name; //姓名  
5     public int age;      //年龄  
6     //显示信息方法  
7     public void show(){  
8         if(age>=18 && age<=60){ //判断年龄  
9             System.out.println(name+"年龄为"+age+", 价格为15元" );  
10        }else{  
11            System.out.println(name + "的年龄为: "+age+", 免费");  
12        }  
13    }  
14 }
```

```
1 public class Visitor {  
2     public String name; //姓名  
3     public int age;      //年龄  
4     //显示信息方法  
5     public void show(){  
6         if(age>=18 && age<=60){ //判断年龄  
7             System.out.println(name+"年龄为"+age+", 价格为15元" );  
8         }else{  
9             System.out.println(name + "的年龄为: "+age+", 免费");  
10        }  
11    }  
12 }
```

```

public class Test2 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        Visitor v = new Visitor();
        System.out.print("请输入姓名: ");
        v.name = input.next();
        System.out.print("请输入年龄: ");
        v.age = input.nextInt();
        v.show();
    }
}

```

```

1    public static void main(String[] args) {
2        Scanner input = new Scanner(System.in);
3        Visitor v = new Visitor();
4        System.out.print("请输入姓名: ");
5        v.name = input.next();
6        System.out.print("请输入年龄: ");
7        v.age = input.nextInt();
8        v.show();
9    }

```

* 掌握无参的方法

* 定义方法

```

public 返回值类型 方法名() {
    //这里编写方法的主体
}

```

* 返回值两种情况

*如果方法具有返回值，方法中必须使用关键字return返回该值，返回类型为该返回值的类型

* 如果方法没有返回值，返回类型为void

* 业务场景：写出机器人对象的“做饭”方法、“扫地”方法



机器人
属性：
颜色：白色

行为：
做饭
扫地
说话

```
public class Rabbit {  
    public String color; // 属性颜色  
    // 做饭  
    public void cook() {  
        System.out.println("I am cooking");  
    }  
    // 扫地  
    public void sweep() {  
        System.out.println("I am sweeping");  
    }  
    // 说话  
    public String speak() {  
        String value = "I can speak";  
        return value;  
    }  
}
```

方法名

没有返回值

有返回值

```
1 public class Rabbit {  
2     public String color; // 属性颜色  
3     // 做饭  
4     public void cook() {  
5         System.out.println("I am cooking");  
6     }  
7     // 扫地  
8     public void sweep() {  
9         System.out.println("I am sweeping");  
10    }
```

```

10     }
11     // 说话
12     public String speak() {
13         String value="I can speak";
14         return value;
15     }
16 }
17

```

```

3 public class Test4 {
4     public static void main(String[] args) {
5         Rabbit rabbit=new Rabbit();
6         rabbit.color="白色";
7         rabbit.cook();
8         rabbit.sweep();
9         String speak = rabbit.speak();
10        System.out.println(speak);
11    }
12 }
13

```

Problems @ Javadoc Declaration Console

<terminated> Test4 (7) [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (2018年8月5日 上午10:48:41)

```

I am cooking
I am sweeping
I can speak

```

```

1 public static void main(String[] args) {
2     Rabbit rabbit=new Rabbit();
3     rabbit.color="白色";
4     rabbit.cook();
5     rabbit.sweep();
6     String speak = rabbit.speak();
7     System.out.println(speak);
8 }

```

* 方法之间允许相互调用，不需要知道方法的具体实现，提高了效率


```
public static void main(String[] args) {
    Rabbit rabbit=new Rabbit();
    rabbit.color="白色";
    rabbit.cook();
    rabbit.sweep();
    String speak = rabbit.speak();
    System.out.println(speak);
}
```

方法之间是可以直接调用的

* 方法细节

```
public class Person {
    public void say() {
        System.out.println("say hello");
        return "hello";
    }
}
```

方法的返回类型为void，方法中不能有return返回值

```
public int sum() {
    int a=1;
    int b=2;
    return a,b;
}
```

方法不能返回多个值

```
public int sum() {
    int a=1;
    int b=2;
    int sum=a+b;
    return sum;
    public void say() {
        System.out.println("say hello");
    }
}
```

多个方法不能相互嵌套定义

```

2
3 public class Person {
4     int age=0;
5     if(age>20) {
6         System.out.println(age);
7     }
8
9     public void say() {
10        System.out.println("say hello");
11    }
12

```

不能在方法外部直接写程序逻辑代码



* 掌握有参的方法

* <访问修饰符> 返回类型 <方法名>(<形式参数列表>)

{

//方法的主体

}

* 为什么要用带参数的方法



* 业务需求

* 1编写榨汁机类 (Mill)

* 2 编写mill的方法 (可以根据根据放进去的水果，炸不同的汁)

* 3 编写测试类

```

2
3= /**
4  * @author xiaozhao
5  * 榨汁机
6  */
7 public class Mill {
8     public String mill(String fruit) {
9         System.out.println("开始搅拌....");
10        return fruit+"汁";
11    }
12 }

```

参数

```

1 public class Mill {
2     public String mill(String fruit) {
3         System.out.println("开始搅拌....");
4         return fruit+"汁";
5     }
6 }

```

```

public static void main(String[] args) {
    Mill m=new Mill();
    String mill = m.mill("苹果");
    System.out.println("获得"+mill);
}

```

实参

```

1     public static void main(String[] args) {
2         Mill m=new Mill();
3         String mill = m.mill("苹果");
4         System.out.println("获得"+mill);
5     }

```

* 业务需求

* 创建客户业务类，实现客户姓名的添加和显示

* 实现思路：

* 1、创建CustomerBiz类

- * 2、创建带参方法addName()
- * 3、创建方法showNames()
- * 4、创建测试类

请输入学生姓名：刘备
请输入学生姓名：张飞
请输入学生姓名：关羽
学生姓名：刘备 张飞 关羽

```
public class StudentsBiz {  
    String[] names = new String[30];  
  
    // 增加学生姓名  
    public void addName(String name) {  
        int index = 0;  
        for (int i = 0; i < names.length; i++) {  
            if (names[i] == null) {  
                index = i;  
                break;  
            }  
        }  
        names[index] = name;  
    }  
  
    // 显示全部学生姓名  
    public void showNames() {  
        System.out.print("学生姓名: ");  
        for (int i = 0; i < names.length; i++) {  
            if (names[i] != null) {  
                System.out.print(names[i] + " ");  
            }  
        }  
    }  
}
```

1 声明数组来存储名字

添加名字

加入入数组的元素为null，就可以存储值

显示名字

```
1 package com.lg.test;  
2  
3 public class StudentsBiz {  
4     String[] names = new String[30];  
5  
6     // 增加学生姓名  
7     public void addName(String name) {  
8         int index = 0;  
9         for (int i = 0; i < names.length; i++) {
```

```

10         if (names[i] == null) {
11             index = i;
12             break;
13         }
14
15     }
16     names[index] = name;
17 }
18
19 // 显示全部学生姓名
20 public void showNames() {
21     System.out.print("学生姓名: ");
22     for (int i = 0; i < names.length; i++) {
23         if (names[i] != null) {
24             System.out.print(names[i] + " ");
25         }
26     }
27 }
28 }
29

```

```

public static void main(String[] args) {
    StudentsBiz st = new StudentsBiz();
    Scanner input = new Scanner(System.in);
    for(int i=0;i<3;i++){
        System.out.print("请输入学生姓名: ");
        String newName = input.next();
        st.addName(newName);
    }
    st.showNames();
}

```

```

1 public static void main(String[] args) {
2     StudentsBiz st = new StudentsBiz();
3     Scanner input = new Scanner(System.in);
4     for(int i=0;i<3;i++){
5         System.out.print("请输入学生姓名: ");
6         String newName = input.next();
7         st.addName(newName);

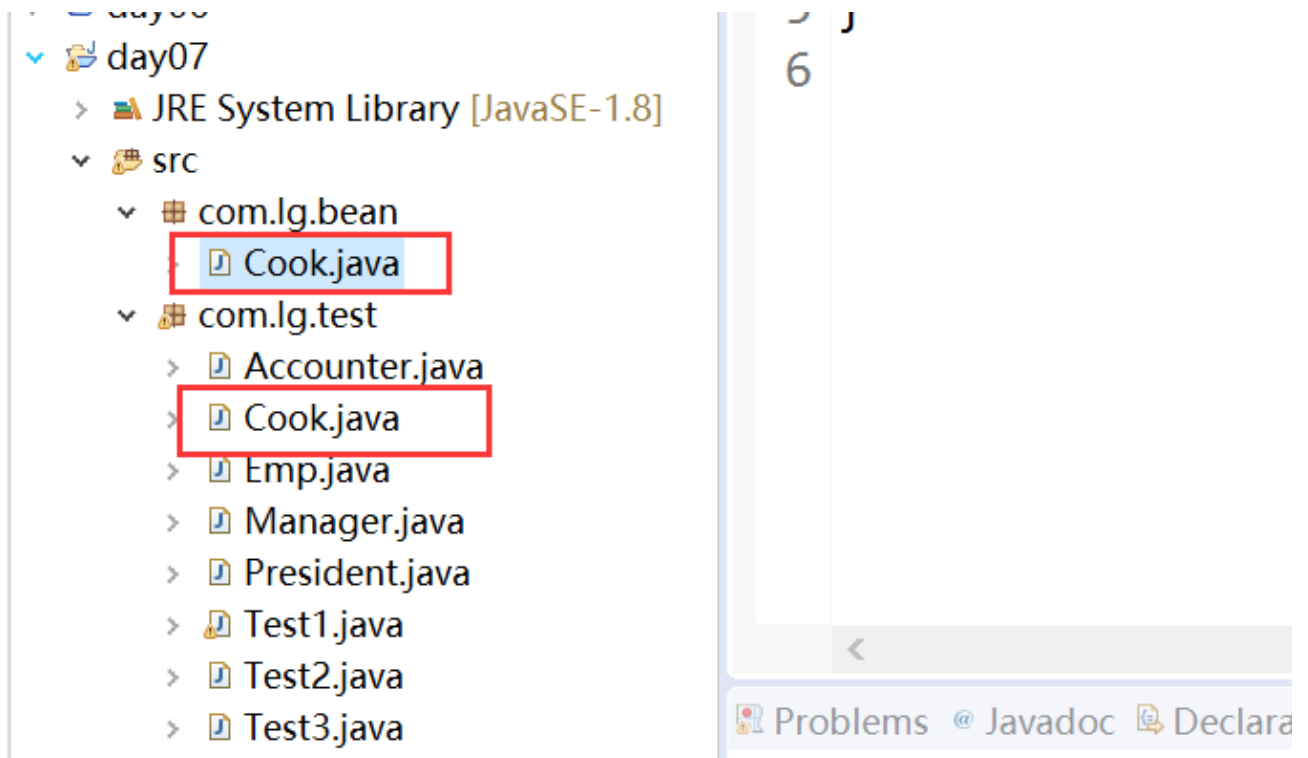
```

```
8      }
9      st.showNames();
10     }
```

* 掌握为什么要使用包和包的命名规则

* 为什么需要包

* 包——解决类的同名问题



* 为了使用不在同一包中的类，需要在Java程序中使用import关键字导入这个类

```

package com.lg.test;
import java.io.*;
import java.util.ArrayList;
import java.util.LinkedList;
public class Test4 {
    public static void main(String[] args) {
        // ctrl+shift+o
        ArrayList list=new ArrayList();
        LinkedList list2=new LinkedList<>();
        //java.lang.*:默认导入了
        String str=new String();
        StringBuffer sb=new StringBuffer();
        File file=new File("xx");
    }
}

```

* 能够使用eclipse生成注释文档

* 介绍eclipse注释

* 单行注释，多行注释，文档注释

* 演示通过eclipse生成文档注释

* 区别基本数据类型和引用数据类型作为方法参数（画堆栈图）

* 区别数据类型两个数的交换分析

交换前： a=5;b=3

交换后： a=5;b=3

```

1 public static void main(String[] args) {
2     int a=5;
3     int b=3;
4     System.out.println("交换前： a="+a+";b="+b);
5     // int temp=a;

```

```

6      //      a=b;
7      //      b=temp;
8      swap(a, b);
9      System.out.println("交换后: a="+a+";b="+b);
10     }
11
12     public static void swap(int a,int b) {
13         int temp=a;
14         a=b;
15         b=temp;
16         System.out.println("方法内: a="+a+";b="+b);
17     }

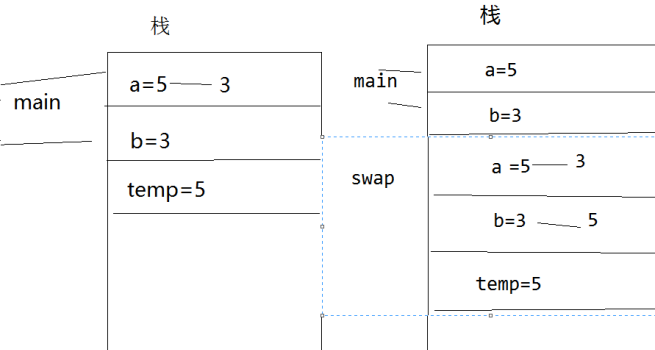
```

基本数据类型：存储在栈

```

main
int a=5;
int b=3;
System.out.println("交换
前: a="+a+";b="+b);
int temp=a;
a=b;
b=temp;
System.out.println("交换
后: a="+a+";b="+b);

```



```

main
int a=5;
int b=3;
System.out.println("交换前:
a="+a+";b="+b);
swap(a, b);
System.out.println("交换后:
a="+a+";b="+b);

swap
public static void swap(int a,int
b) {
    int temp=a;
    a=b;
    b=temp;
    System.out.println("方法
内: a="+a+";b="+b);
}

```

* 引用类型

```

1     public static void main(String[] args) {
2         Person p1=new Person();
3         p1.name="xiaohei";
4         p1.age=20;
5         Person p2=new Person();
6         p2.name="xiaobai";
7         p2.age=88;
8         System.out.println("交换前: "+p1.name+": "+p1.age+"; "+p2.name+": "+p2.age)
9         swap(p1, p2);
10        System.out.println("交换后: "+p1.name+": "+p1.age+"; "+p2.name+": "+p2.age)

```



```

11     }
12
13     public static void swap(Person p1, Person p2) {
14         int temp = p1.age;
15         p1.age = p2.age;
16         p2.age = temp;
17     }

```

```

public static void main(String[] args) {
    Person p1 = new Person();
    p1.name = "xiaohei";
    p1.age = 20;
    Person p2 = new Person();
    p2.name = "xiaobai";
    p2.age = 88;
    System.out.println("交换
前: " + p1.name + ": " + p1.age + "; " + p2.name + ": " + p2.
age);
    swap(p1, p2);
    System.out.println("交换
后: " + p1.name + ": " + p1.age + "; " + p2.name + ": " + p2.
age);
}

public static void swap(Person
p1, Person p2) {
    int temp = p1.age;
    p1.age = p2.age;
    p2.age = temp;
}

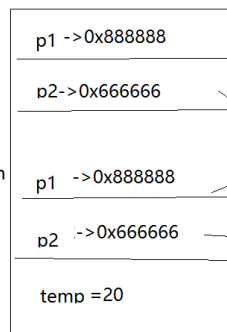
```

main: Person

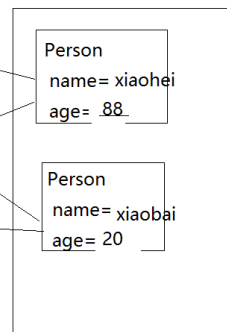
swap: Person

int

栈



堆



Person p1 = new Person();

null

20

: 0

: null

0

88

交换前: xiaohei:20;xiaobai:88
 交换后: xiaohei:88;xiaobai:20