Day04

java课程-李彦伯

> 作业:
>
> 1. 找出数组的最大值

```
int [] arr = {34,15,8,19,45,25};
int max = arr[0];
for (int i = 1; i < arr.length;i++){
if (max < arr[i]){
    max = arr[i];
}
}
System.out.println(max);
```

1. 一维数组累加求和

```
int sum = 0;

for (int i = 0; i < arr.length;i++){
    sum += arr[i];
}
System.out.println(sum);
```

1. 数组的元素倒序

```
//如果在for中同时定义两个变量,用,隔开
for (int beginIndex = 0, endIndex = arr.l
ength - 1; beginIndex < endIndex; beginIn
dex++, endIndex--){

arr[beginIndex] = arr[beginIndex] ^ arr[e
ndIndex];
arr[endIndex] = arr[beginIndex] ^ arr[end
Index];
arr[beginIndex] = arr[beginIndex] ^ arr[e
ndIndex];

}

for (int i : arr){
System.out.println(i);
}
```

1. 数组的元素排序
   4.1 选择排序

```
//4
//选择排序
// 0 1 0 2 0 3
// 1 2 1 3
// 2 3
for(int i = 0; i < arr.length - 1; i++){
for (int j = i + 1; j < arr.length; j++){
    if(arr[i] > arr[j]){
        arr[i] = arr[i] + arr[j] - (ar
r[j] = arr[i]);
    }
}
}
```

4.2 冒泡排序

```
//冒泡排序
//4
// 0 1 1 2 2 3
// 0 1 1 2
// 0 1
for (int i = 0 ; i < arr.length - 1; i++)
{
    for(int j = 0; j < arr.length - 1 -i;
j++){
        if(arr[j] > arr[j+1]){
            arr[j] = arr[j] + arr[j + 1]
- (arr[j+1] = arr[j]);
        }
    }
}
```

方法

> 方法可以认为我们写程序中所要实现的某一个功能,方法中会包含很多条语句,流程控制,循环等,这些内容组合起来去处理一件事情

方法的声明

> 方法的声明位置必须写在类中,并且方法的内部不能再去声明另一个方法,一个类的方法与方法之间只能是并列关系,不能嵌套

- 方法声明的格式

```
修饰符 返回值类型 方法名(参数类型 参数名1,参数类型 参数名2,. . . . . ){
    执行语句
    ……
    return 返回值;
}
```

- 修饰符

> 方法的修饰符比较多，有对访问权限进行限定的，有静态修饰符static，还有最终修饰符final等

- 返回值类型

> 用于限定方法返回值的数据类型,可以是我们学过的所有类型,包括基本类型和引用类型,但是有一个返回值只能作为方法的返回值类型就是void,如果一个方法没有返回值要写成void

- 方法名

> 按照标识符的规则和规范即可,我们可以自行定义

- 参数类型

> 用于限定调用方法时传入参数的数据类型,可以是基本类型和引用类型

- 参数名称

> 用于是一个变量，用于接收调用方法时传入的数据,使我们自己定义的,如果多个参数需要使用”,”进行分隔,一个方法也可以没有参数,如果没有参数()中可以不写任何东西,参数的作用域是当前方法.

- {}

> 中的内容叫做方法体,用来限定的方法区域

- return

1. 有两个作用:a. 表示结束方法. b. 返回该方法指定类型的值
2. 如果一个方法的返回值是void,在方法结束的时候可以不写return,如果写了return,就只代表方法结束,return后不能跟任何内容格式为`return;`
3. 如果一个方法的返回值类型不是void,在方法结束的时候必须要写return,return后要写对应方法返回值类型的数据格式为`return 对应返回值类型的数据;`

- 返回值

> 被return语句返回的值，该值会返回给调用者,就代表执行过本方法后的结果是什么

- 练习

1. 定义一个无返回值无参数的方法,输出10个你好

```
public static void print(){
for (int i = 0 ; i < 10; i++){
System.out.println("你好");
    }
}
```

1. 定义一个无返回值有参数的方法,一个整型的参数,计算并输出这个整数的前n项和

```
public static void sum(int num){
    int sumNum = 0;
    for (int i = 1 ; i <= num; i++){
        sumNum+=i;
    }
    System.out.println(sumNum);
}
```

1. 定义一个有返回值无参数的方法,输出一个在控制台输入的字符串

```
public static String getStr(){

    Scanner sc = new Scanner(System.in);
    String str = sc.nextLine();
    System.out.println(str);
    return str;
}
```

1. 定义一个有返回值有参数的方法,计算并返回3个整型参数的平均数

```
public static int getAvg(int a,int b,int
c){

    return (a+b+c)/3;

}
```

方法的调用

方法声明过后,不会自动执行,需要我们调用执行,因为程序的入口是main,当执行完main方法后程序就结束,所以方法的调用应该放在main方法中

```
public static void main(String[] args) {
    //方法调用的格式是:方法名 (参数具体值)
    print();
    sum(100);
    sum(150);

    String st = getStr();

    System.out.println(st);



  System.out.println("结果是"+getAvg(1, 2,
3));

}
```

- 方法调用的格式为`方法名(参数1,参数2....);`,调用的时候参数的值必须要和定义方法的时候参数的类型相同
- 在main方法中依次调用上述的4个方法
- 注意返回值的使用

方法的重载

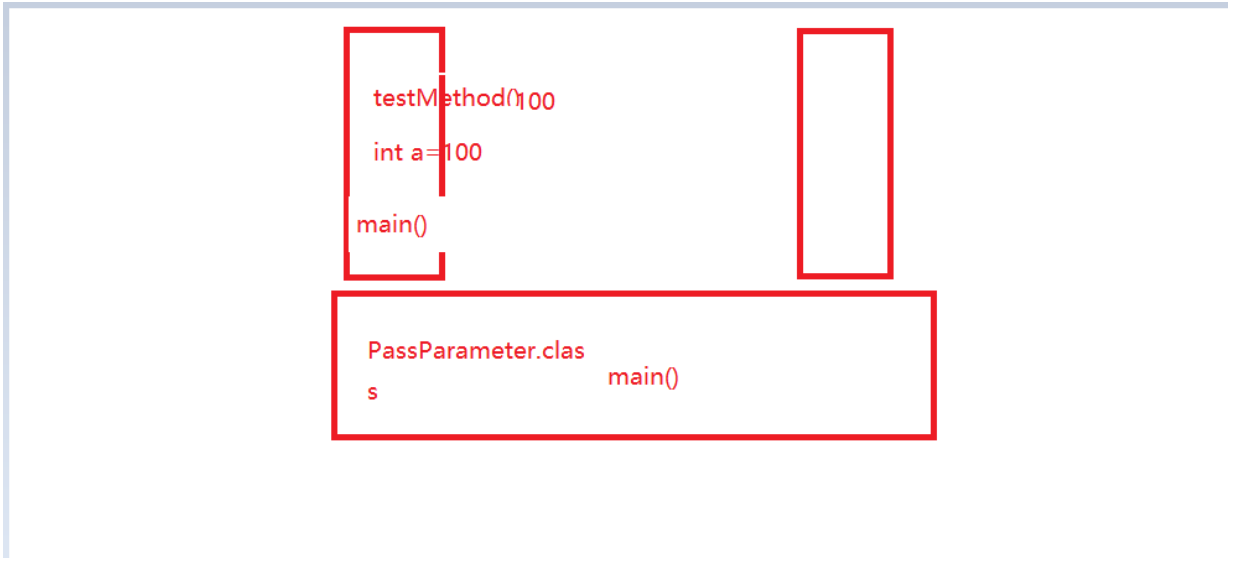
> 如果我们需要计算两个数的和,我们可以计算两个整数,两个小数,一个整数一个小数,按照上述方法我们需要定义三个求和方法,这就需要我们能够分别记住三个方法名称,这显然不是很方便.
> 对于以上这种情况我们可以将这三个方法的方法名称保持一致,只要保证,参数类型,参数个数,参数顺序这三者有一个不同,系统就不会将这三个方法当作同一个方法,这种形式叫做方法的重载.

- 注意方法的重载跟修饰符,返回值,参数的名称都没有关系,只和方法名称和参数类型,参数个数,参数顺序有关系.

```java
public class OverloadDemo {

    public static void main(String[] arg
s) {
        //方法的重载,解决仅仅参数不同的时候,我
们不用再去创建很多名称不一致的方法
        //1. 名称相同
        //2. 参数个数,参数类型,参数顺序3选
一,只要有一个不一致就行
        //方法的重载和返回值类型无关,和修饰符无
关

        sum(1,1.2f);
        sum(1,2);
        sum(1,2,3);
    }


    public static int sum(float a, int b)
{
        return (int)(a+b);
    }
    public static int sum(int a, float b)
{
        return (int)(a+b);
    }
    public static int sum(int a, int b){
        return a+b;
    }
```

```
    public static int sum(int a, int b,in
t c){
        return a+b+c;
    }
    public static int sum(int a, int b,in
t c,int d){
        return a+b+c+d;
    }



}
```

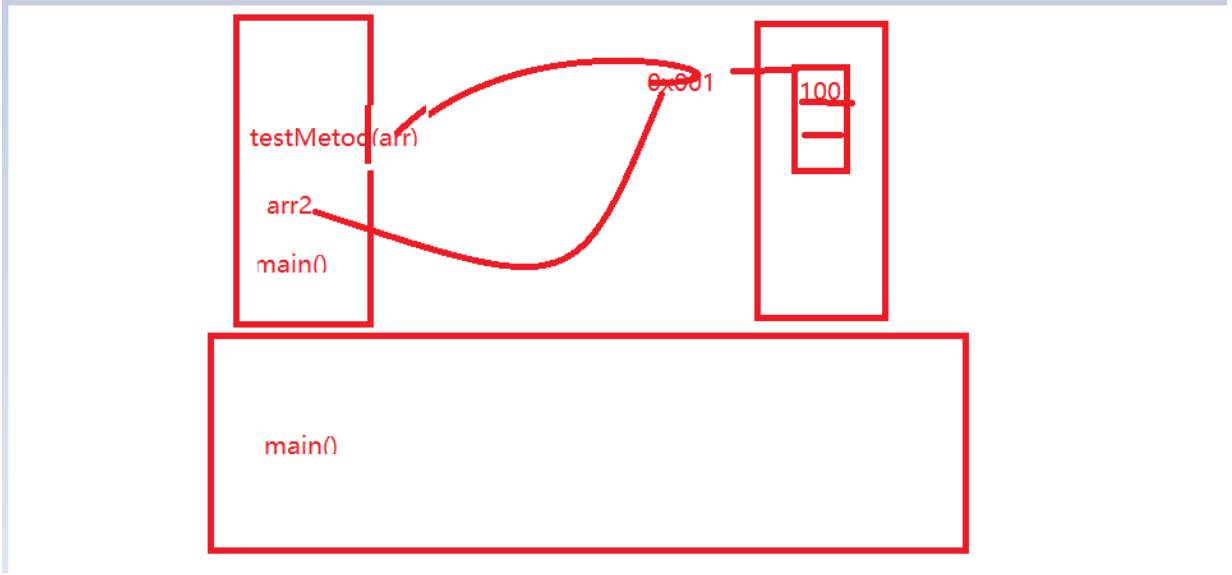
方法调用参数传递的内存原理



```
public static void main(String[] args) {
int a = 100;
testMethod(a);
System.out.println(a);
}

public static void testMethod(int a){
a = 101;
}
```

- 在main中定义的变量a的值是100
- 调用testMethod(a);此时在内存中testMethod压栈执行,并且将a的值作为参数赋值给方法参数
- 在testMethod中修改参数a的值,此时的参数仅仅和main中定义的变量a的值相同,所以修改参数的值,并不会修改main中a的值

```java
public static void main(String[] args) {

int [] arr2 = {1,2,3};
testMethod(arr2);
System.out.println(arr2[0]);

}

public static void testMethod(int [] arr)
{
arr[0] = 100;
//arr = new int []{100,100};
}
```



- 在main中定义的引用类型arr2,初始化内容是1,2,3
- 调用testMethod(int [] arr);此时在内存中testMethod压栈执行,并且将arr2的引用的堆中的地址赋值给参数arr
- arr和arr2两者只是对同一块堆中的地址有引用
- 在testMethod中将引用类型arr[0] = 100;实质是修改了

堆中那个对象的内容,所以arr2引用的那个堆中的内容就发生了改变
- 如果让arr引用新的堆中的内存,并不会影响arr2的引用

面向对象编程

- 面向对象程序设计（OOP Object Oriented Programming）是当前主流的程序设计架构,使我们编程过程更符合人们的日常生活习惯
- 面向过程程序设计,简称opp,c语言就是面向过程的编程语言
- 面向过程思维方式中更多的体现的是执行者（自己做事情），面向对象中更多的体现是指挥者（指挥对象做事情）

面向对象的三大特性

- 封装
  - 之前学习的方法,就是将某个功能写成方法,编写类的过程也是一种封装
  - 提高了代码的复用性,便于调用者的使用,提高了安全性

- 继承(后续讲)
- 多态(后续讲)

类和对象

- 类

  > 对某一类有相同特征的事物的抽象,例如建造房子时设计的模型

- 对象

  > 表示现实中该类事物的个体,就是类的具体实现,例如已经建造成的一个房子

- 属性

  > 该类事物的共有的特征,实际就是我们之前所学习的变量,只不过和之前学过的变量稍有区别

- 方法

  > 该类事物共有的功能

- 声明类的格式

```
public class 类名 {
    //一个类可以有多个成员变量
    数据类型 变量名称;
    数据类型 变量名称;
    //一个类可以有多个方法
    修饰符 返回值类型 方法名(参数类型 参数名称....){
        执行语句;
    }
}
```

- 创建对象的格式 `类名 对象名 = new 类名();`
- 调用成员变量格式:`对象名.成员变量`,如果在当前类中可以直接写成员变量名称
- 调用方法的格式:`对象名.方法名(具体的参数)`,如果在当前类中,可以直接写方法名
- 新建一个类,写main方法

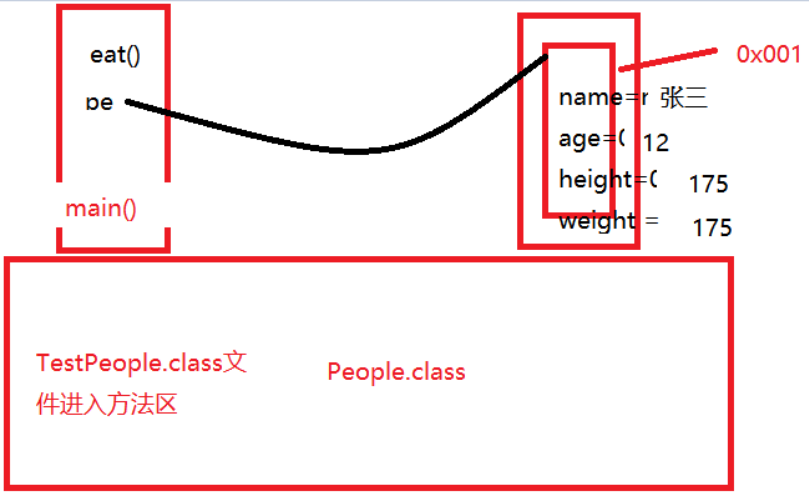
练习

1. 声明一个类是People,两个属性一个是姓名,一个是年龄,有一个方法叫做eat,没有参数,方法输出”吃饱就开心”
2. 思考如何调用?

- 注意:我们程序的入口仍然是main,所以要想对成员变量

赋值,以及调用方法,需要在main中写具体的赋值和调用的代码

调用过程的内存分析



1. class被加载到内存中的方法区,所以在方法区中就有类的所有信息
2. main方法压栈执行
3. People对象在堆中创建,初始化成员变量age为0,height为0.0,weight为0.0
4. 栈中的pe引用堆中的People的内存地址
5. 将堆中的People对象的age赋值为12,height赋值为175,weight为175
6. eat方法压栈执行,输出”吃饱就开心”
7. eat方法出栈
8. main方法出栈,方法结束

> 注意每次new对象,都会从堆内存中创建新的对象

成员变量和局部变量的分别

1. 定义的位置不同:成员变量定义在类中,局部变量定义在方法中
2. 内存中存储的位置不同:成员变量存储在堆中,局部变量存储在栈中
3. 生命周期不同:成员变量伴随这对象的存在而存在,局部变量当方法出栈后就不能使用
4. 初始化不同:成员变量会随着对象的创建进行默认初始化,局部变量如果不初始化就不能使用

set方法和get方法

private关键字

- private是修饰符,使用private修饰的成员变量或者方法只能在当前类中使用
- 所以使用private修饰后的成员变量就不能在其他类中访问

```java
public class People {
    //修饰符是private的话,就只能在当前类中进行使用
    //定义属性
    private int age;
    private double height;
    private double weight;
    private String name;
    //设置set方法
    public  void setAge(int age){
        if(age < 0){
            return;
        }
        this.age = age;
    }
    //this表示当前对象,谁调用的setAge,this就是谁

    public int getAge(){
        return age;
    }

    public void setHeight(double height){
        this.height = height;
    }
    public double getHeight(){
        return this.height;
    }
    public void setWeight(double weight){
```

```
        this.weight = weight;
    }
    public double getWeight(){
        return this.weight;
    }
    public void setName(String name){
        this.name = name;
    }
    public String getName(){
        return this.name;
    }


    //方法
    public  void eat(){
        System.out.println("吃饱了很开心");
        //在非静态方法中可以使用成员变量,因为方法需要对象调用,
        //而对象在创建爱你的时候就已经将成员变量初始化
        //People pe = new People();
        //System.out.println(age);
        weight += 10;
        if (weight > 180){
            //做运动
            sport();
        }

    }
```

```
    public void sport(){
        weight -= 5;
    }


}
```

```
public class TestPeople {
    //每吃一次,体重增加10
    //写sport方法,每运动一次体重减5
    //如果吃饭的时候体重超过180,做一次运动
    //修改所有成员变量变为private写set和get

    public static void main(String[] arg
s) {

        //创建对象,在堆中分配内存并且给成员变量
赋值默认值,基本类型为0,引用类型为null
        People pe = new People();
        pe.setName("张三");
        //pe.name = "张三";
        //pe.age = -12;

        pe.setAge(-12);
        pe.setHeight(175);
        pe.setWeight(175);
        //pe.height = 175;
        //pe.weight = 175;
        pe.eat();
        //pe.sport();
        System.out.println(pe.getAge()+"-
--"+pe.getHeight()+"----"+pe.getWeigh
t());


        People pe1 = new People();
```

```
        pe1.setName("李四") ;
        pe1.eat();
        //System.out.println(pe1.getAg
e()+"---"+pe1.getHeight()+"----"+pe1.getW
eight());



    }

}
```

> 注意以后我们声明类的时候,必须将成员变量用private修饰,然后生成set和get方法

this关键字

> this关键字出现在方法中,表示当前类的对象,一个类可以有多个对象,谁调用的方法,this指的就是谁

作业

定义一个People类,有name,height,weight三个属性,有一个方法判断是否肥胖,启动程序录入3个学生的信息,依次输入姓名,身高,体重,当输入完成后,显示”xx同学,你的身材标准|偏胖|偏瘦,录入成功”,当三个学生录入完成后将三个学生的姓名依次打印