java基础知识 第5天

**【学习目标】理解、了解、应用、记忆**

通过今天的学习，参训学员能够：（解释的时候说出二级目标的掌握程度）

1. **【理解】能够独立编写代码定义方法并调用方法**
2. 【理解】阐述方法的定义格式
3. 【应用】独立编写代码,完成求和方法的定义和调用
4. 【理解】能够画图阐述方法调用的执行流程
5. **【应用】能够独立编写代码完成方法的相关练习**
6. 【应用】定义方法完成练习：获取两个数据中较大值、比较两个数据是否相等、获取三个数据中的较大值
7. 【应用】阐述返回值为void的方法与其他返回值类型方法有什么不同
8. 【应用】定义方法完成打印1到n之间的数据练习、打印所有的水仙花数练习
9. **【理解】理解方法重载及不同类型参数在方法中如何传递**
10. 【理解】阐述什么是方法重载及方法重载的特点
11. 【应用】独立编写代码完成方法重载练习之比较数据是否相等的练习
12. 【理解】阐述基本数据类型作为参数传递的特点
13. 【理解】阐述引用数据类型作为参数传递的特点
14. **【应用】独立编写代码完成方法操作数组的练习**
15. 【应用】独立编写代码，定义方法完成: 数组遍历练习、数组获取最值练习、数组元素求和练习

# 方法的概述及基本使用

## 方法定义格式及格式解释

### 方法的概述

假设有一个游戏程序，程序在运行过程中，要不断地发射炮弹(植物大战僵尸)。发射炮弹的动作需要编写100行的代码，在每次实现发射炮弹的地方都需要重复地编写这100行代码，这样程序会变得很臃肿，可读性也非常差。为了解决代码重复编写的问题，可以将发射炮弹的代码提取出来放在一个{}中，并为这段代码起个名字，这样在每次发射炮弹的地方通过这个名字来调用发射炮弹的代码就可以了。上述过程中，所提取出来的代码可以被看作是程序中定义的一个方法，程序在需要发射炮弹时调用该方法即可

**简单的说：方法就是完成特定功能的代码块**

在很多语言里面都有函数的定义 , 函数在Java中被称为方法

### 方法格式

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…) {

方法体;

return 返回值;

}

### 方法格式解释

修饰符 目前记住 public static

返回值类型 用于限定返回值的数据类型

方法名 一个名字，为了方便我们调用方法

参数类型 用于接收调用方法时传入的数据的类型

参数名 用于接收调用方法时传入的数据的变量

方法体 完成功能的代码

return 结束方法，把返回值带给调用者

### 案例代码一

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: MethodDemo  \* **@Description**: 方法：就是完成特定功能的代码块。  \* **@date** 2017年10月13日 下午1:46:12  \* Company www.igeekhome.com  \*  \* 方法：就是完成特定功能的代码块。  \*  \* 定义格式：  \* 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...) {  \* 方法体;  \* return 返回值;  \* }  \* 格式解释：  \* A:修饰符 目前记住 public static  \* B:返回值类型 用于限定返回值的数据类型  \* C:方法名 一个名字，为了方便我们调用方法  \* D:参数类型 用于接收调用方法时传入的数据的类型  \* E:参数名 用于接收调用方法时传入的数据的变量  \* F:方法体 完成功能的代码  \* G:return 结束方法，把返回值带给调用者  \*  \*/  **public** **class** MethodDemo {  public static int main(String[] arg){    return 0;  }  } |
|  |

## 求和方法的编写

### 定义方法注意事项

写一个方法首先有两点需要明确

返回值类型 明确功能结果的数据类型

参数列表 明确有几个参数，以及参数的类型

按照格式和两个明确来完成如下功能

### 案例代码二

|  |
| --- |
| **package** com.igeek\_02;  /\*\*  \* **@ClassName**: MethodDemo  \* **@Description**: 方法的两个明确:A:返回值类型;B:参数列表。  \* **@date** 2017年10月13日 下午1:46:12  \* Company www.igeekhome.com  \*  \* 方法的两个明确：  \* A:返回值类型 明确功能结果的数据类型  \* B:参数列表 明确有几个参数，以及参数的类型  \*  \* 案例：  \* 求两个数和的案例。  \*  \*/  **public** **class** MethodDemo {  /\*\*  \* **@Title**: sum  \* **@Description**: 写一个方法，用于求和。 两个明确： 返回值类型 int 参数列表 int a,int b  \* **@param** a  \* **@param** b  \* **@return**  \*/  **public** **static** **int** sum(**int** a, **int** b) {  // int c = a + b;  // return c;  **return** a + b;  }  } |

## 求和方法的调用

有明确返回值的方法调用：

单独调用，没有意义

输出调用，有意义，但是不够好，因为我不一定非要把结果输出

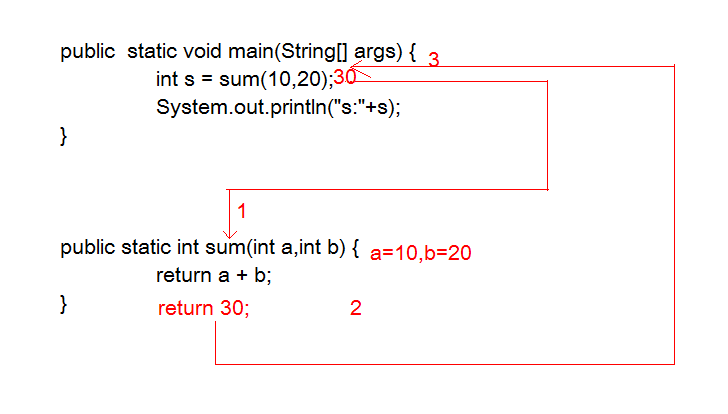
赋值调用，推荐方式

### 案例代码三

|  |
| --- |
| **package** com.igeek\_03;  /\*\*  \* **@ClassName**: MethodDemo  \* **@Description**: 有明确返回值的方法的调用  \* **@date** 2017年10月13日 下午2:01:50  \* Company www.igeekhome.com  \*  \* 有明确返回值的方法的调用：  \* A:单独调用,没有什么意义。  \* B:输出调用,有意义,但是不够好，因为我可能需要对求和的结果进行进一步的操作  \* C:赋值调用  \*/  **public** **class** MethodDemo {    /\*\*  \* **@Title**: sum  \* **@Description**: 求和的方法  \* **@param** a  \* **@param** b  \* **@return**  \*/  **public** **static** **int** sum(**int** a, **int** b) {  **return** a + b;  }  **public** **static** **void** main(String[] args) {  // 单独调用  // sum(10,20);  // 输出调用  // System.out.println(sum(10,20));  // 赋值调用  **int** s = *sum*(10, 20);  // s+=100;  System.***out***.println("s:"+s);  }  } |

## 求和方法的调用图解

### 方法调用流程图



# 方法的练习及注意事项

## 方法的练习1

### 方法练习之获取两个数据中的较大

### 案例代码四

|  |
| --- |
| **package** com.igeek\_01;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: MethodTest  \* **@Description**: 键盘录入两个数据，返回两个数中的较大值  \* **@date** 2017年10月13日 下午2:10:36  \* Company www.igeekhome.com  \*  \* 需求：键盘录入两个数据，返回两个数中的较大值  \*  \* 两个明确：  \* 返回值类型：int  \* 参数列表：int a,int b  \*/  **public** **class** MethodTest {    /\*\*  \* **@Title**: getMax  \* **@Description**: 返回两个数中的较大值  \* **@param** a  \* **@param** b  \* **@return**  \*/  **public** **static** **int** getMax(**int** a, **int** b) {  **if** (a > b) {  **return** a;  } **else** {  **return** b;  }  }    **public** **static** **void** main(String[] args) {  //创建对象  Scanner sc = **new** Scanner(System.***in***);    //接收数据  System.***out***.println("请输入第一个数据：");  **int** x = sc.nextInt();    System.***out***.println("请输入第二个数据：");  **int** y = sc.nextInt();    //调用方法  **int** max = *getMax*(x,y);  System.***out***.println("max:"+max);  }  } |

### 方法练习之比较两个数据是否相等

### 案例代码五

|  |
| --- |
| **package** com.igeek\_01;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: MethodTest2  \* **@Description**: 键盘录入两个数据，比较两个数是否相等  \* **@date** 2017年10月13日 下午2:14:30  \* Company www.igeekhome.com  \*  \* 需求：键盘录入两个数据，比较两个数是否相等  \*  \* 两个明确：  \* 返回值类型：boolean  \* 参数列表：int a,int b  \*/  **public** **class** MethodTest2 {    /\*\*  \* **@Title**: compare  \* **@Description**: 比较两个数是否相等  \* **@param** a  \* **@param** b  \* **@return**  \*/  **public** **static** **boolean** compare(**int** a,**int** b){  **if**(a==b){  **return** **true**;  }**else** {  **return** **false**;  }  }    **public** **static** **void** main(String[] args) {  //创建对象  Scanner sc = **new** Scanner(System.***in***);    //接收数据  System.***out***.println("请输入第一个数据:");  **int** a = sc.nextInt();    System.***out***.println("请输入第二个数据:");  **int** b = sc.nextInt();    //调用方法  **boolean** flag = *compare*(a,b);  System.***out***.println("flag:"+flag);  }  } |

### 方法练习之获取三个数据中的较大值

### 案例代码六

|  |
| --- |
| **package** com.igeek\_01;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: MethodTest3  \* **@Description**: 键盘录入三个数据，返回三个数中的最大值  \* **@date** 2017年10月13日 下午2:17:58  \* Company www.igeekhome.com  \*  \* 需求：键盘录入三个数据，返回三个数中的最大值  \*  \* 两个明确：  \* 返回值类型：int  \* 参数列表：int a,int b,int c  \*/  **public** **class** MethodTest3 {  /\*\*  \* **@Title**: getMax  \* **@Description**: 返回三个数中的最大值  \* **@param** a  \* **@param** b  \* **@param** c  \* **@return**  \*/  **public** **static** **int** getMax(**int** a, **int** b, **int** c) {  **int** max=a;  **if**(b>max){  max=b;  }  **if**(c>max){  max=c;  }  **return** max;  }    **public** **static** **void** main(String[] args) {  //创建对象  Scanner sc = **new** Scanner(System.***in***);    //接收数据  System.***out***.println("请输入第一个数据：");  **int** a = sc.nextInt();    System.***out***.println("请输入第二个数据：");  **int** b = sc.nextInt();    System.***out***.println("请输入第三个数据：");  **int** c = sc.nextInt();    //调用方法  **int** max = *getMax*(a,b,c);  System.***out***.println("max:"+max);  }  } |

## 方法的练习2

### void修饰的方法的调用

写一个方法，在控制台输出10次HelloWorld案例。

没有明确返回值的函数调用：

其实就是void类型方法的调用

只能单独调用

### 案例代码七

|  |
| --- |
| **package** com.igeek\_02;  /\*\*  \* **@ClassName**: MethodDemo  \* **@Description**: 写一个方法，在控制台输出10次HelloWorld案例  \* **@date** 2017年10月13日 下午2:24:23  \* Company www.igeekhome.com  \*  \* 需求：写一个方法，在控制台输出10次HelloWorld案例。  \*  \* 两个明确：  \* 返回值类型：void  \* 参数列表：无参数  \*  \* 如果一个方法没有明确的返回值类型，java提供了void进行修饰。  \*  \* void修饰的方法的调用：  \* A:单独调用  \*/  **public** **class** MethodDemo {      /\*\*  \* **@Title**: printHelloWorld  \* **@Description**: 在控制台输出10次HelloWorld案例。  \*/  **public** **static** **void** printHelloWorld() {  **for**(**int** x=1; x<=10; x++) {  System.***out***.println("HelloWorld");  }  }    **public** **static** **void** main(String[] args) {  //单独调用  *printHelloWorld*();    //输出调用，不可以  //System.out.println(printHelloWorld());  //System.out.println(void);    //赋值调用，不可以  //void v = printHelloWorld();  }  } |

### 打印1到n之间的数据

### 案例代码八

|  |
| --- |
| **package** com.igeek\_02;  /\*\*  \* **@ClassName**: MethodTest  \* **@Description**:写一个方法，传递一个整数(大于1)，在控制台打印1到该数据的值  \* **@date** 2017年10月13日 下午2:40:10  \* Company www.igeekhome.com  \*  \* 需求：写一个方法，传递一个整数(大于1)，在控制台打印1到该数据的值。  \*  \* 两个明确：  \* 返回值类型：void  \* 参数列表：int n  \*/  **public** **class** MethodTest {  /\*\*  \* **@Title**: printNumber  \* **@Description**: 在控制台打印1到该数据n的值  \* **@param** n  \*/  **public** **static** **void** printNumber(**int** n) {  **for**(**int** x=1; x<=n; x++) {  System.***out***.println(x);  }  }    **public** **static** **void** main(String[] args) {  *printNumber*(10);  System.***out***.println("-------------------");  *printNumber*(100);  }  } |

### 打印所有的水仙花数

### 案例代码九

|  |
| --- |
| **package** com.igeek\_02;  /\*\*  \* **@ClassName**: MethodTest2  \* **@Description**: 写一个方法，把所有的水仙花数打印在控制台  \* **@date** 2017年10月13日 下午2:44:40  \* Company www.igeekhome.com  \*  \* 写一个方法，把所有的水仙花数打印在控制台  \*  \* 两个明确：  \* 返回值类型：void  \* 参数列表：无参数  \*/  **public** **class** MethodTest2 {    /\*\*  \* **@Title**: printFlower  \* **@Description**: 把所有的水仙花数打印在控制台  \*/  **public** **static** **void** printFlower() {  **for**(**int** x=100; x<1000; x++) {  **int** ge = x/1%10;  **int** shi = x/10%10;  **int** bai = x/10/10%10;    **if**((ge\*ge\*ge+shi\*shi\*shi+bai\*bai\*bai) == x){  System.***out***.println(x);  }  }  }    **public** **static** **void** main(String[] args) {  *printFlower*();  }  } |

# 方法的重载及参数传递

## 方法重载的概述和基本使用

在同一个类中，允许存在一个以上的同名方法，只要它们的参数个数或者参数类型不同即可。

方法重载特点：

与返回值类型无关，只看方法名和参数列表

在调用时，虚拟机通过参数列表的不同来区分同名方法

### 案例代码十

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: MethodDemo  \* **@Description**: 方法重载：在同一个类中，出现了方法名相同的方法，这就是方法重载。  \* **@date** 2017年10月13日 下午2:52:45  \* Company www.igeekhome.com  \*  \* 方法重载：在同一个类中，出现了方法名相同的方法，这就是方法重载。  \* 方法重载特点：  \* 方法名相同，参数列表不同。与返回值无关。  \* 参数列表不同：  \* 参数的个数不同。  \* 参数对应的类型不同。  \* 注意：  \* 在调用方法的时候，java虚拟机会通过参数列表的不同来区分同名的方法。  \*/  **public** **class** MethodDemo {  **public** **static** **void** main(String[] args) {  **int** a = 10;  **int** b = 20;  // 求和  **int** result = *sum*(a, b);  System.***out***.println("result:" + result);  **int** c = 30;  // 求和  //int result2 = sum2(a,b,c);  //System.out.println("result2:"+result2);  result = *sum*(a,b,c);  System.***out***.println("result:"+result);  }    /\*\*  \* **@Title**: sum  \* **@Description**: 两个float类型的数据求和  \* **@param** a  \* **@param** b  \* **@return**  \*/  **public** **static** **float** sum(**float** a,**float** b) {  **return** a + b;  }  /\*\*  \* **@Title**: sum  \* **@Description**: 三个整数的求和  \* **@param** a  \* **@param** b  \* **@param** c  \* **@return**  \*/  **public** **static** **int** sum(**int** a,**int** b,**int** c) {  **return** a + b + c;  }    /\*\*  \* **@Title**: sum2  \* **@Description**: 三个整数的求和  \* **@param** a  \* **@param** b  \* **@param** c  \* **@return**  \*/  /\*  public static int sum2(int a, int b, int c) {  return a + b + c;  }\*/    /\*\*  \* **@Title**: sum  \* **@Description**: 两个整数的求和  \* **@param** a  \* **@param** b  \* **@return**  \*/  **public** **static** **int** sum(**int** a, **int** b) {  **return** a + b;  }  } |

## 方法重载练习

### 方法重载练习之比较数据是否相等

### 代码案例十一

|  |
| --- |
| **package** com.igeek\_02;  /\*\*  \* **@ClassName**: MethodTest  \* **@Description**: 需求：比较两个数据是否相等。  \* **@date** 2017年10月13日 下午3:01:43  \* Company www.igeekhome.com  \*  \* 需求：比较两个数据是否相等。  \* 参数类型分别为两个byte类型，两个short类型，两个int类型，两个long类型，  \* 并在main方法中进行测试  \*/  **public** **class** MethodTest {  **public** **static** **void** main(String[] args) {  // 调用  System.***out***.println(*compare*(10, 20));  System.***out***.println("-------------");  System.***out***.println(*compare*((**byte**)10, (**byte**)20));  System.***out***.println("-------------");  System.***out***.println(*compare*((**short**)10, (**short**)20));  System.***out***.println("-------------");  //System.out.println(compare((long)10, (long)20));  System.***out***.println(*compare*(10L, 20L));  }  // 两个byte类型的  **public** **static** **boolean** compare(**byte** a, **byte** b) {  System.***out***.println("byte");  // 第一种写法  // boolean flag = a==b?true:false;  // return flag;  // 第二种写法  // boolean flag = a == b;  // return flag;  // 第三种写法  **return** a == b;  }  // 两个short类型的  **public** **static** **boolean** compare(**short** a, **short** b) {  System.***out***.println("short");  **return** a == b;  }  // 两个int类型的  **public** **static** **boolean** compare(**int** a, **int** b) {  System.***out***.println("int");  **return** a == b;  }  // 两个long类型的  **public** **static** **boolean** compare(**long** a, **long** b) {  System.***out***.println("long");  **return** a == b;  }  } |

## 方法中参数传递

### 方法的形式参数为基本数据类型

方法的参数是基本类型的时候：

形式参数的改变不影响实际参数。

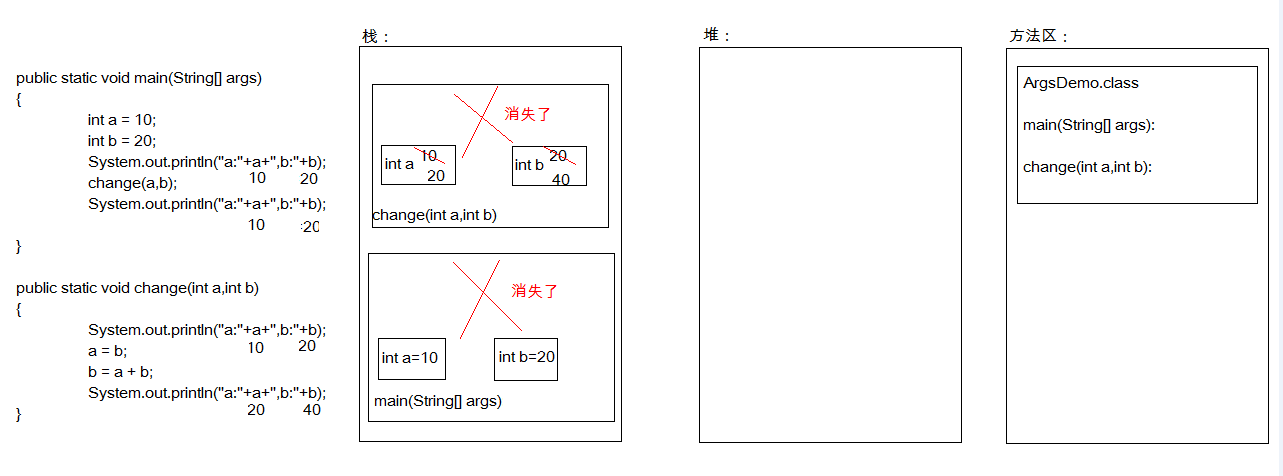
形式参数：用于接收实际数据的变量

实际参数：实际参与运算的变量

### 代码案例十二

|  |
| --- |
| **package** com.igeek\_03;  /\*\*  \* **@ClassName**: ArgsDemo  \* **@Description**: 方法的参数是基本类型的时候：形式参数的改变不影响实际参数。  \* **@date** 2017年10月13日 下午3:30:38  \* Company www.igeekhome.com  \*  \* 方法的参数是基本类型的时候：  \* 形式参数的改变不影响实际参数。  \* 形式参数：用于接收实际数据的变量  \* 实际参数：实际参与运算的变量  \*  \*/  **public** **class** ArgsDemo {    **public** **static** **void** main(String[] args) {  // 定义变量  **int** a = 10;  **int** b = 20;  System.***out***.println("a:" + a + ",b:" + b);// a:10,b:20  *change*(a, b);  System.***out***.println("a:" + a + ",b:" + b);// a:10,b:20  }  /\*\*  \* **@Title**: change  \* **@Description**: 改变形参的值  \* **@param** a  \* **@param** b  \*/  **public** **static** **void** change(**int** a, **int** b) { // a=10,b=20  System.***out***.println("a:" + a + ",b:" + b);// a:10,b:20  a = b; // a=20;  b = a + b; // b=40;  System.***out***.println("a:" + a + ",b:" + b);// a:20,b:40  }  } |

### 方法的形式参数是基本类型图解



### 方法的形式参数为引用数据类型

### 代码案例十三

|  |
| --- |
| **package** com.igeek\_03;  /\*\*  \* **@ClassName**: ArgsDemo2  \* **@Description**: 方法的参数是引用类型： 形式参数的改变直接影响实际参数  \* **@date** 2017年10月13日 下午3:41:01  \* Company www.igeekhome.com  \*  \* 方法的参数是引用类型：  \* 形式参数的改变直接影响实际参数  \*/  **public** **class** ArgsDemo2 {  **public** **static** **void** main(String[] args) {  // 定义数组  **int**[] arr = { 1, 2, 3, 4, 5 };  // 遍历数组  **for** (**int** x = 0; x < arr.length; x++) {  System.***out***.println(arr[x]);  }  System.***out***.println("----------------");  *change*(arr);  **for** (**int** x = 0; x < arr.length; x++) {  System.***out***.println(arr[x]);  }  }  /\*\*  \* **@Title**: change  \* **@Description**: 如果元素是偶数，值就变为以前的2倍  \* **@param** arr  \*/  **public** **static** **void** change(**int**[] arr) {  **for** (**int** x = 0; x < arr.length; x++) {  // 如果元素是偶数，值就变为以前的2倍  **if** (arr[x] % 2 == 0) {  arr[x] \*= 2;  }  }  }  } |

### 方法的形式参数是引用类型图

# 方法操作数组的练习

## 方法操作数组练习

### 方法的练习之数组遍历

需求：把遍历数组改进为方法实现，并调用方法

### 代码案例十四

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: MethodTest  \* **@Description**: 把遍历数组改进为方法实现，并调用方法  \* **@date** 2017年10月13日 下午3:44:10  \* Company www.igeekhome.com  \*  \* 需求：把遍历数组改进为方法实现，并调用方法  \*/  **public** **class** MethodTest {  **public** **static** **void** main(String[] args) {  // 定义数组  **int**[] arr = { 11, 22, 33, 44, 55 };  // 遍历  // for (int x = 0; x < arr.length; x++) {  // System.out.println(arr[x]);  // }    //用方法改进  //printArray(arr);    //这一次虽然可以，但是我觉得格式不好看，能不能打印成下面的格式呢?  //[元素1, 元素2, 元素3, ...]  *printArray*(arr);  }    /\*\*  \* **@Title**: printArray  \* **@Description**: 格式化打印数组内部数据  \* **@param** arr  \*/  **public** **static** **void** printArray(**int**[] arr) {  System.***out***.print("[");  **for**(**int** x=0; x<arr.length; x++){  **if**(x==arr.length-1){  System.***out***.println(arr[x]+"]");  }**else** {  System.***out***.print(arr[x]+", ");  }  }  }    /\*\*  \* **@Title**: printArray  \* **@Description**: 打印数组内部数据  \* **@param** arr  \*  \* 两个明确：  \* 返回值类型：void  \* 参数列表：int[] arr  \*/  // public static void printArray(int[] arr) {  // for(int x=0; x<arr.length; x++){  // System.out.println(arr[x]);  // }  // }  } |

### 方法的练习之数组获取最值

需求：把获取数组最值改进为方法实现，并调用方法

### 代码案例十五

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: MethodTest2  \* **@Description**: 把获取数组最值改进为方法实现，并调用方法  \* **@date** 2017年10月13日 下午3:51:53  \* Company www.igeekhome.com  \*  \* 需求：把获取数组最值改进为方法实现，并调用方法  \*/  **public** **class** MethodTest2 {  **public** **static** **void** main(String[] args) {  // 定义数组  **int**[] arr = { 34, 67, 10, 28, 59 };    //获取数组中的最大值  // //定义参照物  // int max = arr[0];  // //遍历，依次比较，大的留下来  // for(int x=1; x<arr.length; x++) {  // if(arr[x] > max) {  // max = arr[x];  // }  // }    //用方法改进  **int** max = *getMax*(arr);  System.***out***.println("max:"+max);    //获取数组中的最小值，用方法实现  **int** min = *getMin*(arr);  System.***out***.println("min:"+min);    }    /\*\*  \* **@Title**: getMin  \* **@Description**: 获取数组中的最小值的方法  \* **@param** arr  \* **@return**  \*/  **public** **static** **int** getMin(**int**[] arr) {  **int** min = arr[0];    **for**(**int** x=1; x<arr.length; x++) {  **if**(arr[x] < min) {  min = arr[x];  }  }    **return** min;  }    /\*\*  \* **@Title**: getMax  \* **@Description**: 获取数组中的最大值的方法  \* **@param** arr  \* **@return**  \*/  **public** **static** **int** getMax(**int**[] arr) {  **int** max = arr[0];    **for**(**int** x=1; x<arr.length; x++) {  **if**(arr[x] > max) {  max = arr[x];  }  }    **return** max;  }  } |

### 方法的练习之数组元素求和

需求：写一个方法，用于对数组进行求和，并调用方法。

### 代码案例十六

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: MethodTest3  \* **@Description**: 写一个方法，用于对数组进行求和，并调用方法。  \* **@date** 2017年10月13日 下午3:57:35  \* Company www.igeekhome.com  \*  \* 需求：写一个方法，用于对数组进行求和，并调用方法。  \*/  **public** **class** MethodTest3 {  **public** **static** **void** main(String[] args) {  // 定义数组  **int**[] arr = { 1, 2, 3, 4, 5 };    // //定义求和变量  // int sum = 0;  // //获取数组中的每一个元素  // for(int x=0; x<arr.length; x++) {  // sum += arr[x];  // }    //用方法改进  **int** sum = *sum*(arr);    System.***out***.println("sum:"+sum);  }    /\*\*  \* **@Title**: sum  \* **@Description**: 对数组进行求和  \* **@param** arr  \* **@return**  \*/  **public** **static** **int** sum(**int**[] arr) {  **int** sum = 0;    **for**(**int** x=0; x<arr.length; x++) {  sum += arr[x];  }    **return** sum;  }  } |

重点和总结

1、方法的概述

2、方法的编写及基本使用

3、方法的重载及参数传递

4、方法的操作数组