java基础知识 第3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
16. **【应用】Eclipse断点调试**
17. 【记忆】能够阐述Eclipse断点调试的作用及步骤
18. 【应用】能够独立调试循环求和的代码
19. 【应用】能够独立调试方法调用执行流程的代码
20. 【应用】能够独立调试参数传递的代码
21. **【应用】Java基础语法练习一**
22. 【应用】能够独立写出根据月份输出对应季节的两种实现方式
23. 【应用】能够独立写出求回文数的案例
24. 【应用】能够独立写出求不死神兔的案例
25. 【理解】能够独立写出求出满足要求的数组元素和的案例
26. **【应用】Java基础语法练习二**
27. 【应用】能够独立写出评委打分案例
28. 【应用】能够独立写出数组元素反转案例
29. 【应用】能够独立写出数组基本查找案例
30. 【应用】能够独立写出数据加密案例

# 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 {  } |

## 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

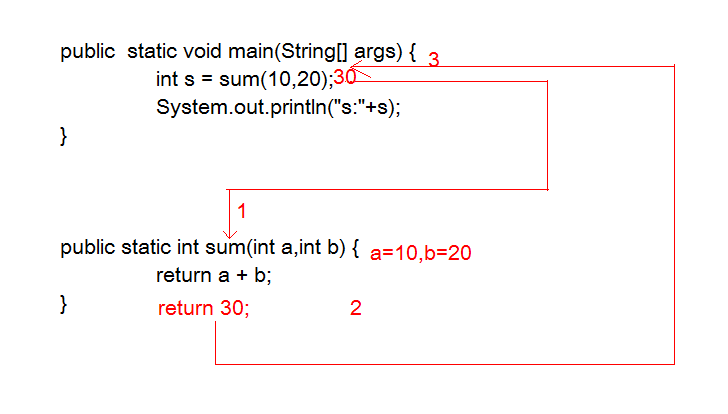
赋值调用，推荐方式

### 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。

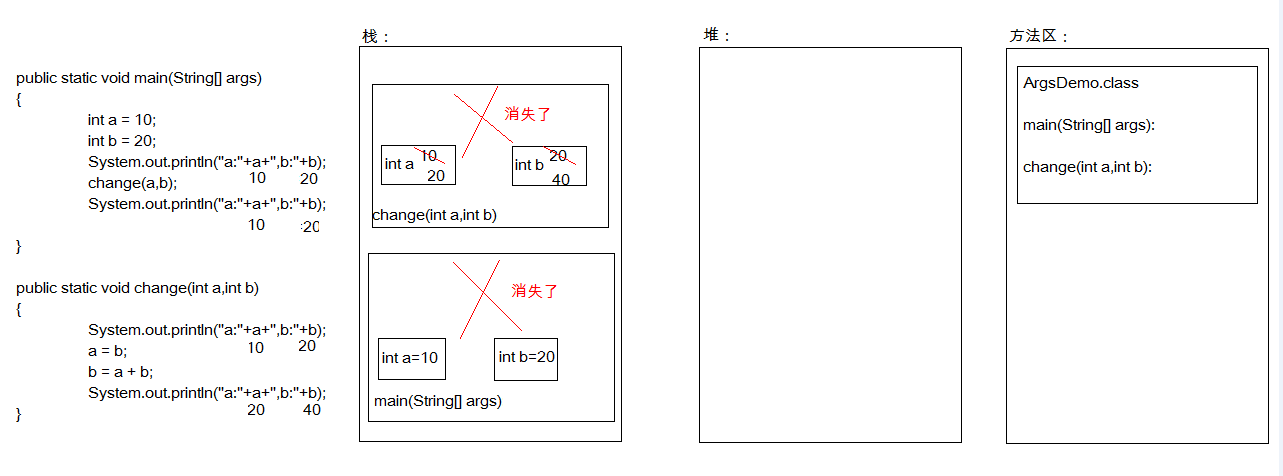
形式参数：用于接收实际数据的变量

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;  }  } |

# Eclipse断点调试

## Eclipse断点调试概述

Eclipse的断点调试可以查看程序的执行流程和解决程序中的bug

## Eclipse断点调试常用操作:

A:什么是断点：

就是一个标记，从哪里开始。

B:如何设置断点：

你想看哪里的程序，你就在那个有效程序的左边双击即可。

C:在哪里设置断点：

哪里不会点哪里。

目前：我们就在每个方法的第一条有效语句上都加。

D:如何运行设置断点后的程序：

右键 -- Debug as -- Java Application

E:看哪些地方：

Debug：断点测试的地方

在这个地方，记住F6，或者点击也可以。一次看一行的执行过程。

Variables：查看程序的变量变化

ForDemo：被查看的源文件

Console：控制台

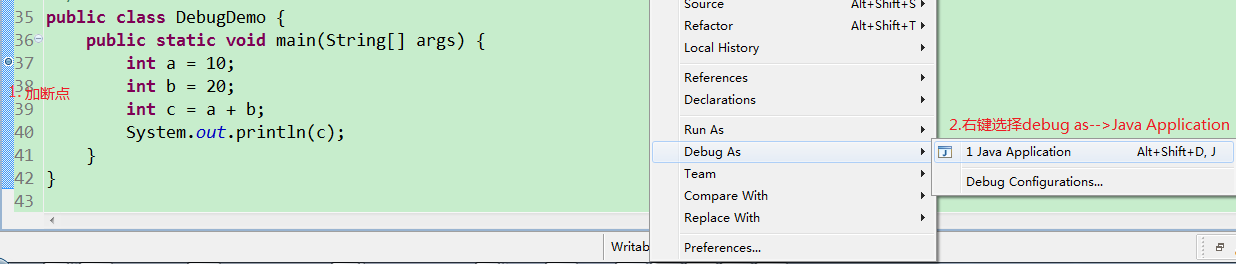
F:如何去断点：

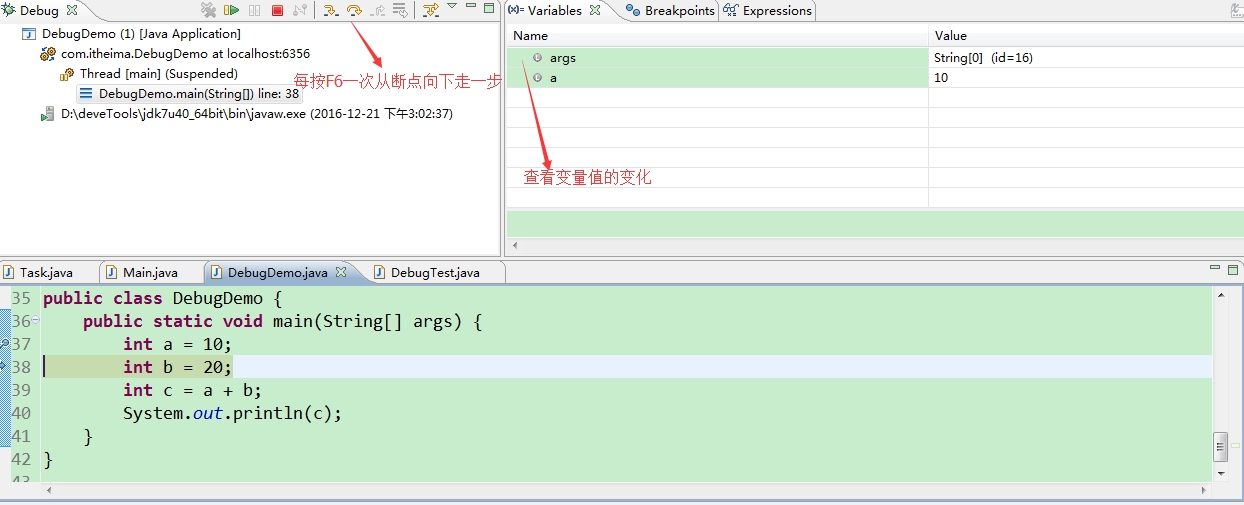
再次双击即可

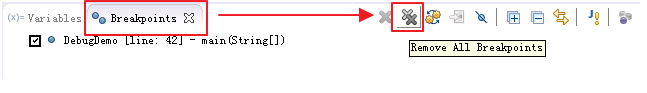
找到Debug视图，Variables界面，找到Breakpoints，并点击，然后看到所有的断点，最后点击那个双叉。

### 案例代码十七

|  |
| --- |
| **package** com.igeek\_02;  /\*\*  \* **@ClassName**: DebugDemo  \* **@Description**: 断点调试  \* **@date** 2017年10月13日 下午4:36:41  \* Company www.igeekhome.com  \*  \* 断点调试：  \* A:查看程序的执行流程  \* B:调试程序  \*  \* 断点：  \* 其实就是一个标记  \*  \* 在哪里加呢?  \* 想加哪里就加哪里，一般是加在我们看不懂的地方  \*  \* 如何加呢?  \* 在代码区域的最左边双击即可  \*  \* 如何运行加断点的程序呢?  \* 代码区域 -- 右键 -- Debug as -- Java Application  \* 会弹出一个页面让我们选择是否进入debug模式，选择yes。  \*  \* 如何让程序往下执行呢?  \* Step Over 执行下一步  \* F6  \*  \* 看那些区域呢?  \* 代码区域：看程序的执行步骤  \* Debug区域：看程序的执行步骤  \* Variables：看变量的创建，赋值，销毁等  \* Console：看程序的输入和输出  \*  \* 如何去除断点：  \* A:把加断点的动作再来一遍  \* B:在debug视图中，找到Breakpoints，选中断点，点击双x即可  \*/  **public** **class** DebugDemo {  **public** **static** **void** main(String[] args) {  **int** a = 10;  **int** b = 20;  **int** c = a + b;  System.***out***.println(c);  }  } |



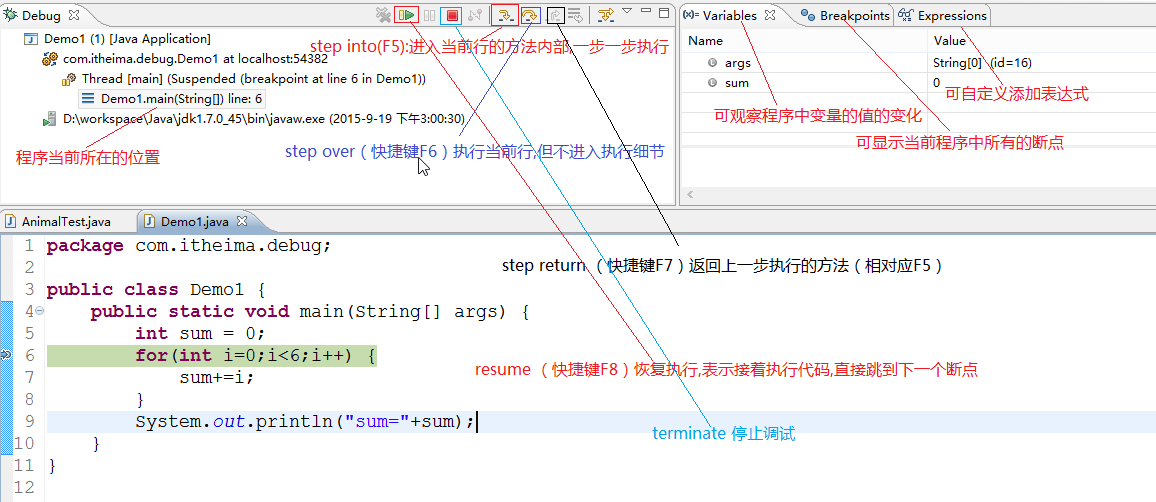




## 断点调试练习

### 案例代码十八

|  |
| --- |
| **package** com.igeek\_03;  /\*\*  \* **@ClassName**: DebugTest  \* **@Description**: 需求：看循环的执行流程(1-5求和案例)  \* **@date** 2017年10月13日 下午4:44:16  \* Company www.igeekhome.com  \*  \* 需求：看循环的执行流程(1-5求和案例)  \*/  **public** **class** DebugTest {  **public** **static** **void** main(String[] args) {  // 定义求和变量  **int** sum = 0;  // 循环获取每一个数据  **for** (**int** x = 1; x <= 5; x++) {  sum += x;  }  System.***out***.println("sum:" + sum);  }  } |



### 案例代码十九

|  |
| --- |
| **package** com.igeek\_03;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: DebugTest2  \* **@Description**: 看方法的调用流程  \* **@date** 2017年10月13日 下午4:51:01  \* Company www.igeekhome.com  \*  \* 需求：看方法的调用流程  \*  \* 有方法调用的时候，要想看到完整的流程，每个方法都要加断点，建议方法进入的第一条有效语句加断点  \*/  **public** **class** DebugTest2 {  **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  **int** result = *sum*(a, b);  // 输出结果  System.***out***.println("result:" + result);  }  /\*\*  \* **@Title**: sum  \* **@Description**: 求和方法  \* **@param** a  \* **@param** b  \* **@return**  \*/  **public** **static** **int** sum(**int** a, **int** b) {  **return** a + b;  }  } |

### 案例代码二十

|  |
| --- |
| **package** com.igeek\_03;  /\*\*  \* **@ClassName**: DebugTest3  \* **@Description**: 形式参数的改变不影响实际参数。  \* **@date** 2017年10月13日 下午4:53:17  \* Company www.igeekhome.com  \*  \* 参数是基本数据类型：  \* 形式参数的改变不影响实际参数。  \*/  **public** **class** DebugTest3 {  **public** **static** **void** main(String[] args) {  **int** a = 10;  **int** b = 20;  System.***out***.println("a:" + a + ",b:" + b);  *change*(a, b);  System.***out***.println("a:" + a + ",b:" + b);  }  /\*\*  \* **@Title**: change  \* **@Description**: 方法内部改变形参的值  \* **@param** a  \* **@param** b  \*/  **public** **static** **void** change(**int** a, **int** b) {  System.***out***.println("a:" + a + ",b:" + b);  a = b;  b = a + b;  System.***out***.println("a:" + a + ",b:" + b);  }  } |

### 案例代码二十一

|  |
| --- |
| **package** com.igeek\_03;  /\*\*  \* **@ClassName**: DebugTest4  \* **@Description**: 形式参数的改变直接影响实际参数  \* **@date** 2017年10月13日 下午5:01:08  \* 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** DebugTest4 {  **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;  }  }  }  } |

# 基础语法的练习

## 循环,if和switch练习

### 键盘录入月份，输出对应的季节(if…else或switch实现)

#### 案例代码二十二

|  |
| --- |
| **package** com.igeek\_01;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: Test  \* **@Description**: 键盘录入一个月份，输出该月份对应的季节。  \* **@date** 2017年10月13日 下午5:11:51  \* Company www.igeekhome.com  \*  \* 需求：键盘录入一个月份，输出该月份对应的季节。  \* 一年有四季  \* 3,4,5 春季  \* 6,7,8 夏季  \* 9,10,11 秋季  \* 12,1,2 冬季  \*  \* 分析：  \* A:键盘录入一个月份,用Scanner实现  \* B:判断该月份是几月，根据月份输出对应的季节  \* if  \* switch  \*/  **public** **class** Test {  **public** **static** **void** main(String[] args) {  // 键盘录入一个月份,用Scanner实现  Scanner sc = **new** Scanner(System.***in***);  // 接收数据  System.***out***.println("请输入一个月份(1-12)：");  **int** month = sc.nextInt();  // 判断该月份是几月，根据月份输出对应的季节  /\*  if (month == 1) {  System.out.println("冬季");  } else if (month == 2) {  System.out.println("冬季");  } else if (month == 3) {  System.out.println("春季");  } else if (month == 4) {  System.out.println("春季");  } else if (month == 5) {  System.out.println("春季");  } else if (month == 6) {  System.out.println("夏季");  } else if (month == 7) {  System.out.println("夏季");  } else if (month == 8) {  System.out.println("夏季");  } else if (month == 9) {  System.out.println("秋季");  } else if (month == 10) {  System.out.println("秋季");  } else if (month == 11) {  System.out.println("秋季");  } else if (month == 12) {  System.out.println("冬季");  } else {  System.out.println("你输入的月份有误");  }  \*/    //代码太长了，能不能简单一些呢?  //能，如何简单一些呢?  //我们可以把相同季节的月份放到一起来判断  //(month==3 || month==4 || month==5)  **if**(month==1 || month==2 || month==12) {  System.***out***.println("冬季");  }**else** **if**(month==3 || month==4 || month==5) {  System.***out***.println("春季");  }**else** **if**(month==6 || month==7|| month==8) {  System.***out***.println("夏季");  }**else** **if**(month==9 || month==10 || month==11) {  System.***out***.println("秋季");  }**else** {  System.***out***.println("你输入的月份有误");  }  }  } |

#### 案例代码二十三

|  |
| --- |
| **package** com.igeek\_01;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: Test2  \* **@Description**: 键盘录入一个月份，输出该月份对应的季节  \* **@date** 2017年10月13日 下午5:15:29  \* Company www.igeekhome.com  \*  \* 需求：键盘录入一个月份，输出该月份对应的季节。  \* 一年有四季  \* 3,4,5 春季  \* 6,7,8 夏季  \* 9,10,11 秋季  \* 12,1,2 冬季  \*  \* 分析：  \* A:键盘录入一个月份,用Scanner实现  \* B:判断该月份是几月，根据月份输出对应的季节  \* if  \* switch  \* case穿透  \*/  **public** **class** Test2 {  **public** **static** **void** main(String[] args) {  // 键盘录入一个月份,用Scanner实现  Scanner sc = **new** Scanner(System.***in***);  // 接收数据  System.***out***.println("请输入月份(1-12)：");  **int** month = sc.nextInt();  // 用switch语句实现  /\*  switch (month) {  case 1:  System.out.println("冬季");  break;  case 2:  System.out.println("冬季");  break;  case 3:  System.out.println("春季");  break;  case 4:  System.out.println("春季");  break;  case 5:  System.out.println("春季");  break;  case 6:  System.out.println("夏季");  break;  case 7:  System.out.println("夏季");  break;  case 8:  System.out.println("夏季");  break;  case 9:  System.out.println("秋季");  break;  case 10:  System.out.println("秋季");  break;  case 11:  System.out.println("秋季");  break;  case 12:  System.out.println("冬季");  break;  default:  System.out.println("你输入的月份有误");  break;  }  \*/    //case 穿透  /\*  switch(month) {  case 1:  System.out.println("hello");  //break;  case 2:  System.out.println("world");  break;  default:  System.out.println("over");  break;  }  \*/    //通过case穿透现象改进代码  **switch**(month) {  **case** 1:  **case** 2:  **case** 12:  System.***out***.println("冬季");  **break**;  **case** 3:  **case** 4:  **case** 5:  System.***out***.println("春季");  **break**;  **case** 6:  **case** 7:  **case** 8:  System.***out***.println("夏季");  **break**;  **case** 9:  **case** 10:  **case** 11:  System.***out***.println("秋季");  **break**;  **default**:  System.***out***.println("你输入的月份有误");  **break**;  }  }  } |

### 打印5位数中所有的回文数

#### 案例代码二十四

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: Test3  \* **@Description**: 打印5位数中的所有回文数  \* **@date** 2017年10月13日 下午5:17:53  \* Company www.igeekhome.com  \*  \* 需求：打印5位数中的所有回文数。  \* 什么是回文数呢?举例：12321是回文数，个位与万位相同，十位与千位相同。  \*  \* 分析：  \* A:5位数告诉了我们数据的范围，用for循环实现  \* B:获取每一个5位数，然后得到它的个位，十位，千位，万位  \* 假设x是一个5位数：  \* 个位：x%10  \* 十位：x/10%10  \* 千位：x/10/10/10%10  \* 万位：x/10/10/10/10%10  \* C:把满足条件的数据输出即可  \*/  **public** **class** Test3 {  **public** **static** **void** main(String[] args) {  //5位数告诉了我们数据的范围，用for循环实现  **for**(**int** x=10000; x<100000; x++) {  //获取每一个5位数，然后得到它的个位，十位，千位，万位  **int** ge = x%10;  **int** shi = x/10%10;  **int** qian = x/10/10/10%10;  **int** wan = x/10/10/10/10%10;    //把满足条件的数据输出即可  **if**((ge==wan) && (shi==qian)) {  System.***out***.println(x);  }  }  }  } |

## 数组练习

### 不死神兔问题

#### 案例代码二十五

|  |
| --- |
| **package** com.igeek\_02;  /\*\*  \* **@ClassName**: Test  \* **@Description**: 不死神兔问题  \* **@date** 2017年10月13日 下午5:22:15  \* Company www.igeekhome.com  \*  \* 需求：  \* 有一对兔子，从出生后第3个月起每个月都生一对兔子，小兔子长到第三个月后每个月又生一对兔子，  \* 假如兔子都不死，问第二十个月的兔子对数为多少？  \*  \* 规律：  \* 第一个月：1  \* 第二个月：1  \* 第三个月：2  \* 第四个月：3  \* 第五个月：5  \* ...  \*  \* 规律：从第三个月开始，每个月的兔子对数是前两个月的兔子对数之和。  \* 第一个月和第二个月的兔子对数是1  \* 分析：  \* int[] arr = new int[20];  \*  \* arr[0] = 1;  \* arr[1] = 1;  \*  \* arr[2] = arr[0] + arr[1];  \* arr[3] = arr[1] + arr[2];  \* arr[4] = arr[2] + arr[3];  \* ...  \*/  **public** **class** Test {  **public** **static** **void** main(String[] args) {  //定义数组  **int**[] arr = **new** **int**[20];    //初始化第一个月和第二个月的兔子对数c  arr[0] = 1;  arr[1] = 1;    **for**(**int** x=2; x<arr.length; x++) {  arr[x] = arr[x-2] + arr[x-1];  }    System.***out***.println("第二十个月的时候的兔子对数是："+arr[19]);  }  } |

### 求数组中满足要求的元素和

#### 案例代码二十六

|  |
| --- |
| **package** com.igeek\_02;  /\*\*  \* **@ClassName**: Test2  \* **@Description**: 求出该数组中满足要求的元素和  \* **@date** 2017年10月13日 下午5:24:33  \* Company www.igeekhome.com  \*  \* 需求：  \* (1)定义一个int类型的一维数组，内容为{171,72,19,16,118,51,210,7,18}  \* (2)求出该数组中满足要求的元素和。  \* 要求：求和的元素的个位和十位不能包含7,并且只能为偶数。  \*  \* 分析：  \* A:定义一个int类型的一维数组  \* B:定义一个求和变量  \* C:遍历数组，获取到数组中的每一个元素  \* D:判断该元素是否满足如下要求，如果是就累加，否则，不搭理它  \* x%2==0  \* x%10 != 7  \* x/10%10 !=7  \* E:输出结果  \*/  **public** **class** Test2 {  **public** **static** **void** main(String[] args) {  //定义一个int类型的一维数组  **int**[] arr = {171,72,19,16,118,51,210,7,18};    //定义一个求和变量  **int** sum = 0;    //遍历数组，获取到数组中的每一个元素  **for**(**int** x=0; x<arr.length; x++) {  **if**((arr[x]%10 != 7) && (arr[x]/10%10 != 7) && (arr[x]%2 == 0)) {  sum += arr[x];  }  }    System.***out***.println("sum:"+sum);  }  } |

### 裁判评分

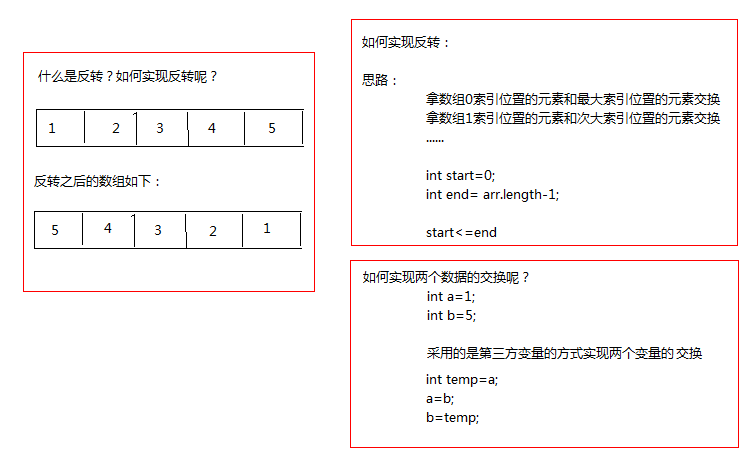
#### 案例代码二十七

|  |
| --- |
| **package** com.igeek\_02;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: Test3  \* **@Description**: 裁判评分  \* **@date** 2017年10月13日 下午5:28:53  \* Company www.igeekhome.com  \*  \* 需求：在编程竞赛中，有6个评委为参赛的选手打分，分数为0-100的整数分。  \* 选手的最后得分为：去掉一个最高分和一个最低分后 其余4个评委打分的平均值。  \* 请写代码实现。(不考虑小数部分)  \*  \* 分析：  \* A:定义一个长度为6的数组。  \* B:通过键盘录入的方式给出评委的分数  \* C:写方法实现获取数组中的最大值，最小值  \* D:写方法实现数组元素的求和  \* E:平均分： (和-最高分-最低分)/(arr.length-2)  \* F:输出分数即可  \*/  **public** **class** Test3 {  **public** **static** **void** main(String[] args) {  //定义一个长度为6的数组  **int**[] arr = **new** **int**[6];    //通过键盘录入的方式给出评委的分数  Scanner sc = **new** Scanner(System.***in***);  **for**(**int** x=0; x<arr.length; x++) {  // arr[x] = sc.nextInt();  System.***out***.println("请给出第"+(x+1)+"个评委的分数(0-100)：");  **int** number = sc.nextInt();  arr[x] = number;  }    //写方法实现获取数组中的最大值，最小值  **int** max = *getMax*(arr);  **int** min = *getMin*(arr);    //写方法实现数组元素的求和  **int** sum = *sum*(arr);    // (和-最高分-最低分)/(arr.length-2)  **int** avg = (sum-max-min)/(arr.length-2);    //输出分数即可  System.***out***.println("该选手的最终得分是："+avg);  }    /\*\*  \* **@Title**: sum  \* **@Description**: 数组元素求和  \* **@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;  }    /\*\*  \* **@Title**: getMin  \* **@Description**: 数组中的最小值  \* **@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**: 数组中的最大值  \* **@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;  }  } |

### 数组反转

#### 案例代码二十八

|  |
| --- |
| **package** com.igeek\_02;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: Test4  \* **@Description**: arr数组中的内容反转  \* **@date** 2017年10月13日 下午5:33:55  \* Company www.igeekhome.com  \*  \* 需求：  \* (1)键盘录入5个int类型的数据存储数组arr中  \* (2)定义方法将arr数组中的内容反转  \* (3)定义方法对反转后的数组进行遍历  \*  \* 分析：  \* A:定义一个长度为5的数组  \* B:通过键盘录入数据给数组中的元素赋值  \* C:定义方法将arr数组中的内容反转  \* 什么是反转?如何反转?  \* D:定义方法遍历数组  \*/  **public** **class** Test4 {  **public** **static** **void** main(String[] args) {  // 定义一个长度为5的数组  **int**[] arr = **new** **int**[5];  // 通过键盘录入数据给数组中的元素赋值  Scanner sc = **new** Scanner(System.***in***);  **for** (**int** x = 0; x < arr.length; x++) {  System.***out***.println("请给出第" + (x + 1) + "个元素");  arr[x] = sc.nextInt();  }    System.***out***.println("反转前的数组元素：");  *printArray*(arr);  // 定义方法将arr数组中的内容反转  *reverse*(arr);    System.***out***.println("反转后的数组元素：");  //定义方法遍历数组  *printArray*(arr);  }    /\*\*  \* **@Title**: printArray  \* **@Description**: 遍历数组  \* **@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**: reverse  \* **@Description**: 反转数组中的内容  \* **@param** arr  \*/  **public** **static** **void** reverse(**int**[] arr) {  **for**(**int** startIndex=0,endIndex=arr.length-1;startIndex<=endIndex;startIndex++,endIndex--) {  **int** temp = arr[startIndex];  arr[startIndex] = arr[endIndex];  arr[endIndex] = temp;  }  }  } |



### 数组基本查找

#### 案例代码二十九

|  |
| --- |
| **package** com.igeek\_02;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: Test5  \* **@Description**: 查找指定元素第一次在数组中出现的索引  \* **@date** 2017年10月13日 下午5:38:12  \* Company www.igeekhome.com  \*  \*需求：数组元素查找(查找指定元素第一次在数组中出现的索引)  \*(1)给定数组int[] arr = {5,7,3,2,5};  \*(2)要查询的元素通过键盘录入的方式确定  \*(3)定义一个查找数组元素第一次出现位置的方法(注,要查找的元素就是键盘录入的数据)  \*  \*分析：  \* A:给定数组int[] arr = {5,7,3,2,5};  \* B:要查询的元素通过键盘录入的方式确定  \* C:定义一个查找数组元素第一次出现位置的方法  \* 遍历数组，获取到每一个元素，进行比较，如果相等，就直接把该处的索引返回。  \* D:调用方法，输出结果  \*/  **public** **class** Test5 {  **public** **static** **void** main(String[] args) {  // 给定数组int[] arr = {5,7,3,2,5};  **int**[] arr = { 5, 7, 3, 2, 5 };  //要查询的元素通过键盘录入的方式确定  Scanner sc = **new** Scanner(System.***in***);    System.***out***.println("请输入要查找的元素：");  **int** number = sc.nextInt();    //定义一个查找数组元素第一次出现位置的方法  //调用方法  **int** index =*getIndex*(arr, number);  System.***out***.println("index:"+index);  }    /\*\*  \* **@Title**: getIndex  \* **@Description**: 查找指定元素第一次在数组中出现的索引  \* **@param** arr  \* **@param** value  \* **@return**  \*/  **public** **static** **int** getIndex(**int**[] arr,**int** value) {  //遍历数组，获取到每一个元素，进行比较，如果相等，就直接把该处的索引返回。  /\*  for(int x=0; x<arr.length; x++) {  if(arr[x] == value) {  return x;  }  }    return -1;  \*/    **int** index = -1;    **for**(**int** x=0; x<arr.length; x++) {  **if**(arr[x] == value) {  index = x;  **break**;  }  }    **return** index;  }  } |

### 数据加密

#### 案例代码三十

|  |
| --- |
| **package** com.igeek\_02;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: Test6  \* **@Description**: 对数据进行加密  \* **@date** 2017年10月13日 下午5:42:54  \* Company www.igeekhome.com  \*  \* 需求：键盘录入数据,要求数据是四位的整数,现需要对数据进行加密,加密规则如下:  \* 每位数字都加上5,然后除以10的余数代替该数字,  \* 再将第一位和第四位交换,第二位和第三位交换,  \* 请把加密后的数据输出到控制台  \*  \* 分析：  \* A:键盘录入一个四位数  \* B:对数据进行加密  \* 举例：  \* 4567  \* 把这个四位数分成个，十，百，千存储到数组中  \* int[] arr = {4,5,6,7};  \* 每位数字都加上5：  \* arr[x] += 5; {9,10,11,12}  \* 然后除以10的余数代替该数字：  \* arr[x] %= 10; {9,0,1,2}  \* 再将第一位和第四位交换,第二位和第三位交换：  \* {9,0,1,2} {2,1,0,9}  \* C:输出加密后的数据  \*/  **public** **class** Test6 {  **public** **static** **void** main(String[] args) {  //键盘录入一个四位数  Scanner sc = **new** Scanner(System.***in***);    //接收数据  System.***out***.println("请输入一个四位数：");  **int** number = sc.nextInt();    //分别得到该数据的每一个位上的数据  **int** ge = number%10;  **int** shi = number/10%10;  **int** bai = number/10/10%10;  **int** qian = number/10/10/10%10;    //定义一个数组  **int**[] arr = **new** **int**[4];  arr[0] = qian;  arr[1] = bai;  arr[2] = shi;  arr[3] = ge;    //加密规则  //每位数字都加上5,然后除以10的余数代替该数字  **for**(**int** x=0; x<arr.length; x++) {  arr[x] += 5;  arr[x] %= 10;  }    //再将第一位和第四位交换,第二位和第三位交换  **int** temp = arr[0];  arr[0] = arr[3];  arr[3] = temp;    **int** temp2 = arr[1];  arr[1] = arr[2];  arr[2] = temp2;    //输出加密后的数据  **for**(**int** x=0; x<arr.length; x++) {  System.***out***.print(arr[x]);  }  System.***out***.println();  }  } |

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

5、基础语法的练习：条件语句、循环语句、数组的具体使用