第05天 java基础知识

今日内容介绍

* 方法的概述
* 方法的练习
* 方法的重载
* 方法参数类型详解

# 方法的概述

## 方法的概述和定义格式

### 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

### 方法定义及格式

**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

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…) {

函数体;

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

/\*

\* 方法：其实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：

\* 修饰符： 目前记住这里使用public static即可

\* 返回值类型： 用于限定返回值的数据类型

\* 方法名： 方便我们调用方法

\* 参数类型： 用于限定调用方法时传入的数据的数据类型

\* 参数名： 用于接收调用方法时传入的数据的变量

\* 方法体： 完成特定功能的代码

\* return 返回值：结束方法，并且把返回值带给调用者

\*

\*/

**public** **class** MethodDemo {

**public** **static** **void** main(String[] args) {

}

}

## 方法案例之定义求和方法

### 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
  + 求两个数据之和的案例

### 案例代码二

/\*

\* 方法：其实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：

\* 修饰符： 目前记住这里使用public static即可

\* 返回值类型： 用于限定返回值的数据类型

\* 方法名： 方便我们调用方法

\* 参数类型： 用于限定调用方法时传入的数据的数据类型

\* 参数名： 用于接收调用方法时传入的数据的变量

\* 方法体： 完成特定功能的代码

\* return 返回值：结束方法，并且把返回值带给调用者

\*

\* 需求：求两个数据之和的案例

\*

\* public static ??? sum(???) {...}

\*

\* 两个明确：

\* 返回值类型：其实就是要我们明确方法结果的数据类型

\* 参数列表：其实就是要我们明确有几个参数参加，并且每个参数都是什么数据类型的

\*

\* 注意事项：方法与方法是平级关系，不能嵌套定义

\*/

**public** **class** MethodDemo {

**public** **static** **void** main(String[] args) {

}

/\*

\* 需求：求两个数据之和的案例

\*

\* 两个明确：

\* 返回值类型：int

\* 参数列表：int a,int b

\*/

**public** **static** **int** sum(**int** a,**int** b) {

**int** c = a + b;

**return** c;

}

}

## 方法调用之有明确返回值的方法调用

* 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.itheima\_01;

/\*

\* 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** MethodDemo2 {

// 求和的方法

**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;

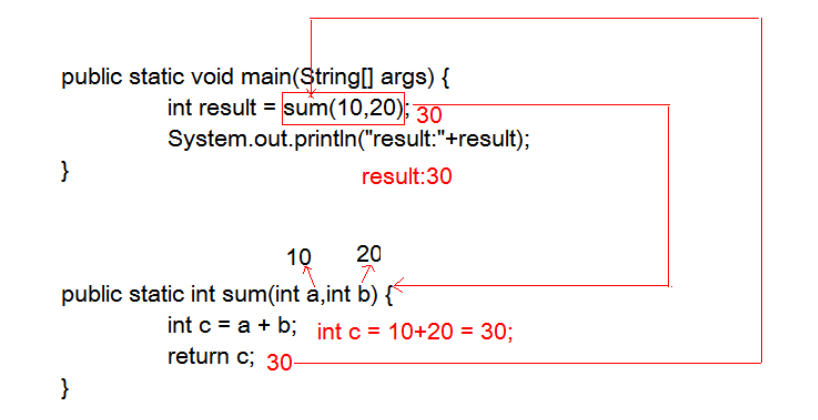
System.***out***.println("s:"+s);

}

}

## 方法调用之方法调用图解

### 方法调用流程图



# 方法的练习

## 方法的练习

### 方法练习之求两个数据较大值

### 案例代码四

**package** com.itheima\_01;

**import** java.util.Scanner;

/\*

\* 需求：键盘录入两个数据，返回两个数中的较大值

\*

\* 思考：

\* 键盘录入两个数据的动作放在哪里比较合适呢?

\* 假如键盘录入数据的动作放在了返回两个数中的较大值的方法里面，那么，我如果不想返回键盘录入

\* 的数据的较大值，我就想返回两个固定值的较大值，我们是不是又得修改代码了。这个时候发现

\* 是不合理的，所以我们应该把键盘录入数据的动作放在main方法中。

\*

\* 命名习惯问题：

\* 类：

\* 如果是一个单词，首字母大写。Student

\* 如果是多个单词，每个单词的首字母大写。HelloWorld

\*

\* 方法：和变量的命名是一样的规则。

\* 如果是一个单词，首字母小写。sum()

\* 如果是多个单词，从第二个单词开始，每个单词首字母大写。getMax()

\*/

**public** **class** MethodTest {

**public** **static** **void** main(String[] args) {

//创建键盘录入对象

Scanner sc = **new** Scanner(System.***in***);

//获取数据

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** max = *getMax*(a,b);

//输出结果

System.***out***.println("max:"+max);

}

/\*

\* 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** **static** **int** getMax(**int** a,**int** b) {

**if**(a > b) {

**return** a;

}**else** {

**return** b;

}

}

}

### 方法练习之比较两个数据是否相等

### 案例代码五

**package** com.itheima\_01;

**import** java.util.Scanner;

/\*

\* 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 {

//比较两个数是否相等

**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);

}

}

### 方法调用之void修饰的方法调用

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

### 案例代码六

**package** com.itheima\_02;

/\*

\* 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：没有参数

\*

\* 如果一个方法没有明确的返回值类型，也不能把返回值类型的地方空出来。

\* 这里应该写void表示该方法无返回值类型。

\*

\* 方法调用：(void修饰的方法)

\* 只能单独调用

\*/

**public** **class** MethodDemo {

**public** **static** **void** main(String[] args) {

//单独调用

*printHelloWorld*();

//输出调用

//System.out.println(printHelloWorld());

//赋值调用

//void v = printHelloWorld();

}

/\*

\* 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：

\*/

**public** **static** **void** printHelloWorld() {

/\*

for(int x=1; x<=10; x++) {

System.out.println("HelloWorld");

}

\*/

**for**(**int** x=0; x<10; x++) {

System.***out***.println("HelloWorld");

}

}

}

### 方法练习之在控制台打印所有的水仙花数

### 案例代码七

**package** com.itheima\_02;

/\*

\* 需求：写一个方法，把所有的水仙花数打印在控制台

\*/

**public** **class** MethodTest {

**public** **static** **void** main(String[] args) {

//调用方法

*printFlower*();

}

/\*

\* 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：

\*/

**public** **static** **void** printFlower() {

**for**(**int** x=100; x<1000; x++) {

**int** ge = x%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);

}

}

}

}

### 方法练习之在控制台打印1到n的数据

### 案例代码八

**package** com.itheima\_02;

/\*

\* 需求：写一个方法，打印1到n之间的数据(n是调用的时候传递过来的)

\* 隐含：n>=1

\*/

**public** **class** MethodTest2 {

**public** **static** **void** main(String[] args) {

//调用方法

*printNumber*(3);

System.***out***.println("------------");

*printNumber*(5);

}

/\*

\* 打印1到n之间的数据(n是调用的时候传递过来的)

\*

\* 两个明确：

\* 返回值类型：void

\* 参数列表：int

\*/

**public** **static** **void** printNumber(**int** n) {

**for**(**int** x=1; x<=n; x++) {

System.***out***.println(x);

}

}

}

# 方法的重载

## 方法重载的概述和特点

* 方法重载概述
  + 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

### 案例代码九

**package** com.itheima\_03;

/\*

\* 方法重载：在同一个类中，出现了方法名相同的情况。

\*

\* 方法重载的特点：

\* 方法名相同，参数列表不同。方法重载与返回值无关。

\*

\* 参数列表不同：

\* 参数的个数不同。

\* 参数对应的数据类型不同。

\*

\* 注意事项：

\* 在调用方法的时候，java虚拟机会根据方法名及参数列表的不同来区分方法。

\*/

**public** **class** MethodDemo {

**public** **static** **void** main(String[] args) {

//定义两个变量

**int** a = 10;

**int** b = 20;

//求和

**int** result = *sum*(a,b);

System.***out***.println("result:"+result);

//定义变量

**int** c = 30;

//int result2 = sum2(a,b,c);

**int** result2 = *sum*(a,b,c);

System.***out***.println("result2:"+result2);

}

/\*

public static int sum(int x,int y) {

return x + y;

}

\*/

**public** **static** **float** sum(**float** a,**float** b) {

**return** a + b;

}

//求三个数据的和

/\*

public static int sum2(int a,int b,int c) {

return a + b + c;

}

\*/

**public** **static** **int** sum(**int** a,**int** b,**int** c) {

**return** a + b + c;

}

//求两个数据的和

**public** **static** **int** sum(**int** a,**int** b) {

/\*

int c = a + b;

return c;

\*/

**return** a + b;

}

}

## 方法重载练习之比较两个数据是否相等

### 代码案例十

**package** com.itheima\_03;

/\*

\* 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) {

//System.out.println(compare(10, 20));

//System.out.println(compare((byte)10, (byte)20));

//System.out.println(compare((short)10, (short)20));

System.***out***.println(*compare*(10L, 20L));

}

/\*

\* 两个byte类型

\*

\* 两个明确：

\* 返回值类型：boolean

\* 参数列表：byte a,byte b

\*/

**public** **static** **boolean** compare(**byte** a,**byte** b){

/\*

if(a == b) {

return true;

}else {

return false;

}

\*/

System.***out***.println("byte");

**return** a == b;

}

//两个short类型

**public** **static** **boolean** compare(**short** a,**short** b){

System.***out***.println("short");

**return** a == b;

}

//两个int类型

**public** **static** **boolean** compare(**int** a,**int** b){

System.***out***.println("int");

**return** a == b;

}

//两个long类型

**public** **static** **boolean** compare(**long** a,**long** b){

System.***out***.println("long");

**return** a == b;

}

}

# 方法参数类型详解

## 方法中参数传递

### 方法参数是基本类型的情况和图解

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

#### 代码案例十一

**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

}

**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;

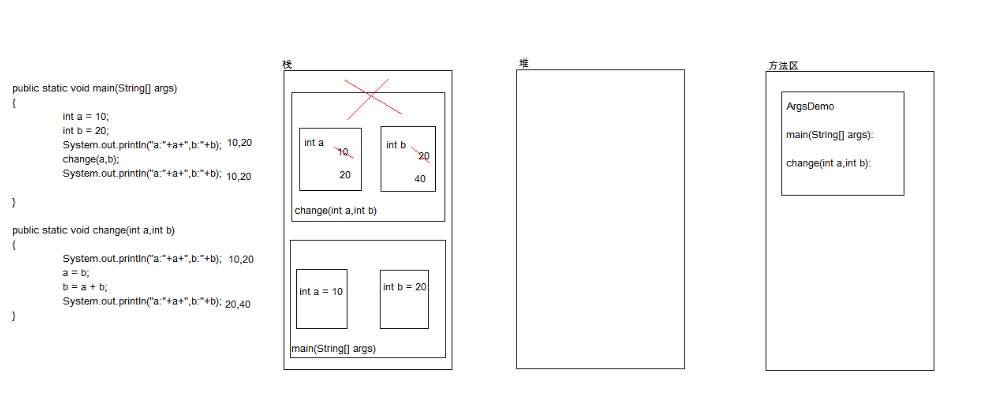
b = a + b;// b=40;

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



### 方法参数是引用类型的情况和图解

#### 代码案例十二

**package** com.itheima;

/\*

\* 基本数据类型作为参数传递，形式参数的改变不影响实际参数

\* 引用数据类型作为参数传递，形式参数的改变直接影响实际参数

\*/

**public** **class** ArgsDemo2 {

**public** **static** **void** main(String[] args) {

**int**[] arr = { 1, 2, 3, 4, 5 };

**for** (**int** x = 0; x < arr.length; x++) {

System.***out***.println(arr[x]);//1,2,3,4,5

}

*change*(arr);

**for** (**int** x = 0; x < arr.length; x++) {

System.***out***.println(arr[x]);//1,4,3,8,5

}

}

**public** **static** **void** change(**int**[] arr) {

**for** (**int** x = 0; x < arr.length; x++) {

//如果数组元素是偶数，值就变成了以前的2倍

**if** (arr[x] % 2 == 0) {

arr[x] \*= 2;

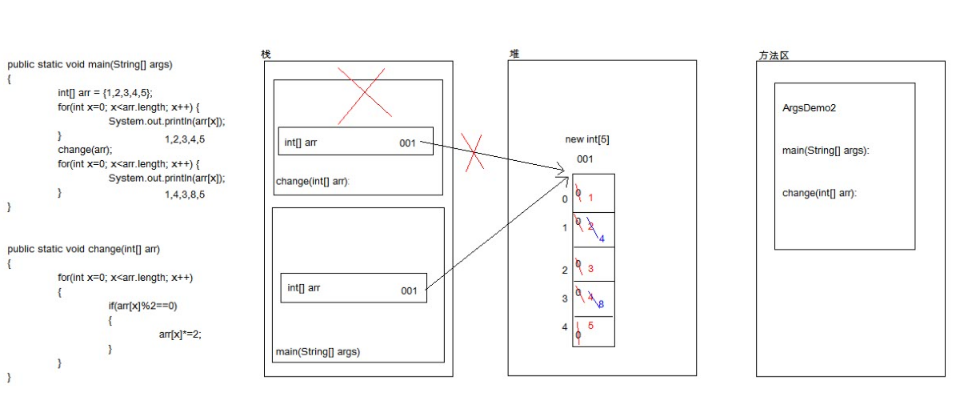
}

}

}

}

#### 方法的形式参数是引用类型图



## 方法操作数组练习

### 方法练习之数组遍历

需求：把遍历数组改进为方法实现，并调用方法

### 代码案例十三

**package** com.itheima;

/\*

\* 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};

//调用方法

//printArray(arr);

//我们已经实现了需求，但是我觉得这个效果不好看，我想要如下的效果可以吗

//[元素1, 元素2, 元素3, ...]

//我们的结果应该是这样的：[11, 22, 33, 44, 55]

*printArray*(arr);

}

/\*

\* 遍历数组的方法。

\*

\* 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]);

}

}

\*/

**public** **static** **void** printArray(**int**[] arr) {

System.***out***.print("[");

**for**(**int** x=0; x<arr.length; x++) {

**if**(x == arr.length-1) {

//判断是否是最后一个元素

System.***out***.print(arr[x]);

}**else** {

System.***out***.print(arr[x]+", ");

}

}

System.***out***.println("]");

}

}

### 方法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

### 代码案例十四

**package** com.itheima;

/\*

\* 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 = {24,36,90,75,81};

//调用方法

**int** max = *getMax*(arr);

//输出结果

System.***out***.println("max:"+max);

}

/\*

\* 数组获取最值的方法

\*

\* 两个明确：

\* 返回值类型：int

\* 参数列表：int[] arr

\*/

**public** **static** **int** getMax(**int**[] arr) {

//定义参照物

**int** max = arr[0];

//遍历，获取元素，进行比较

**for**(**int** x=1; x<arr.length; x++) {

**if**(arr[x] > max) {

max = arr[x];

}

}

**return** max;

}

}