

# Java基础语法

zcx

## 一、方法

### 1. 方法的完整定义格式及调用

修饰符 返回值类型 方法名 (参数列表) { }

如:

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

方法名为sum, a,b是形参, int代表返回值类型, c是返回值 public static 是修饰符

```
package com.cxz.method;  
  
public class method {  
    public static void main(String[] args) {  
        //目标:掌握方法的定义和调用  
        System.out.println(sum(10,20));          //方法的调用  
    }  
    //定义一个方法求任意两个整数的和  
    public static int sum(int a,int b){      //方法的定义  
        int result = a+b;  
        return result;  
    }  
}
```

### 2. 方法的其他形式

- i. 方法可以没有参数, 也可以有参数, 即可以不接收数据处理
- ii. 方法也可以不需要返回数据

```
//打印三行的Hello World  
public static void printlnHelloWorld(){  
    System.out.println("Hello World");  
    System.out.println("Hello World");  
    System.out.println("Hello World");  
}
```

这就是不需要接收数据, 故形参为空, 并且不需要返回值, 故返回值为void

```
//定义一个方法获取一个四位数验证码，需要接收验证码长度，返回验证码
public static String getCode(int length){
    String code = "";
    for (int i = 0; i < length; i++) {
        int num = (int)(Math.random()*10);
        code += num;
    }
    return code;
}
```

从主函数内调用时传递给length值，之后经过一系列变化赋值给code，最后返回code，得到验证码

### 3. 方法的注意事项

#### i. 方法可以重载

- 一个类中出现多个方法名称相同，但是他们的参数列表不同(顺序不同，数据类型不同，个数不同等)，这些方法就是重载(其他无所谓哦)

```
//掌握方法的重载
public static void printlnHelloWorld(int count){
    for (int i = 0; i < count; i++) {
        System.out.println("Hello World");
    }
}

public static void printlnHelloWorld(String name){
    System.out.println("Hello World, "+name);
}

public static void printlnHelloWorld(String name,int count){
    for (int i = 0; i < count; i++) {
        System.out.println("Hello World, "+name);
    }
}
```

#### ii. 无返回值的方法中可以使用单独的return;立即结束当前方法的执行

```
//掌握在无返回值的方法中单独使用return，设计一个除法到的功能
public static void divide(int a,int b){
    if (b==0) {
        System.out.println("除数不能为0");
        return;
    }
    System.out.println(a/b);
}
```

因为输入的除数若为0，则a/b会报错，故使用return结束当前方法的执行，并输出提示信息

## 二、类型转换

### 1. 自动类型转换

- 类型范围小的变量可以直接赋值给类型范围大的变量，称为自动类型转换

如：

byte -> short -> int -> long -> float -> double

char->int

```
package com.cxz.type;
public class type {
    public static void main(String[] args) {
        //认识自动类型转换
        byte a=12;
        type_demo1(a); //byte类型直接赋值给int类型
        int b=12;
        type_demo2(b); //int类型直接赋值给double类型
    }
    //认识自动类型转换
    public static void type_demo1(int a){
        System.out.println(a);
    }
    public static void type_demo2(double a){
        System.out.println(a);
    }
}
```

- 自动类型转换原理：范围大的会包容范围小的，多余的高位全为0即可

### 2. 强制类型转换

- 类型范围大的变量需要强制类型转换才能赋值给类型范围小的变量
- 格式：数据类型 变量名 = (数据类型)变量名；

数据类型都是小类型范围的，并且左边的变量名是需要重新声明一个新的

比如：

```
//强制类型转换
double c=12;
int d=(int)c;
type_demo3(d);
//写一个方法掌握强制类型转换
public static void type_demo3(int a){
```

```
System.out.println(a);
}
```

可见这是大转小，double强制转换为int，根据格式需要int d = (int)c;

- 强制转换如果数据超过了小范围的，会导致数据溢出，截取低位得出结果
- 若为double强转，则会丢掉小数，保留整数部分

### 3. 表达式的自动类型提升

- 表达式中的小范围的变量会自动转换为较大范围的类型，再参与运算

byte short char 是直接转换为int类型参与运算

表达式的最终结果类型由表达式中的最高类型决定

比如：

```
//目标：理解表达式中的自动类型提升
public static double calc(int a ,int b,double c,char r){
    return a+b+c+r;
}
```

发现a,b,c,r中的最高类型为double，故返回类型也应该为double

```
public static int calc(byte a ,byte b){
    return a+b;
}
```

同理，byte,short,char都是直接转换为int类型参与运算，故返回类型为int

当然，若只想返回byte，可以直接把结果先 **强转** 赋给一个byte类型的变量，在返回这个变量（或者在return的时候强转）：

```
public static byte calc(byte a ,byte b,byte d){
    byte c=(byte)(a+b+d);
    return c;
}
```

因为方法内自动转为int了故想要转回byte只能强转

当然，这种有风险，毕竟 **相加后可能会超byte的范围**，最好还是不要强转

## 三、输入、输出

- 输出：System.out.println();
  - 输入：Scanner程序，是JAVA提供的API，可直接调用
  - 下载API文档：<https://www.oracle.com/java/technologies/javase-jdk21-doc-downloads.html>
- 实例演示：

```

package com.cxz.scanner;

//1.导包
import java.util.Scanner; //告诉我的程序去jdk里面找包用scanner
//可以 选中scanner, 按alt+shift+回车, 自动导包
public class scanner {
    public static void main(String[] args) {
        print();
    }
    //写一个程序, 可以让用户键盘输入他的用户名和年龄
    public static void print() {
        //2.创建对象(直接 抄写这行代码, 得到一个scanner扫描器对象)
        Scanner sc = new Scanner(System.in);
        //3.调用方法
        System.out.println("请输入你的用户名：");
        String name = sc.next(); //让程序在这一行暂停, 等待用户输入一个字符串, 等到按了回车, 再往下走
        System.out.println("请输入你的年龄：");
        int age = sc.nextInt(); //让程序在这一行暂停, 等待用户输入一个整数, 等到按了回车, 再往下走
        System.out.println("你的用户名是：" + name + ", 你的年龄是：" + age);
    }
}

```

- 总之输入数据有三步：
  - 1.导包: import java.util.Scanner;
  - 2.创建对象: Scanner sc = new Scanner(**System.in**);
  - 3.调用方法: String name = sc.next();等方法, 得到用户输入的数据

## 四、运算符

### 1. 基本算数运算符

运算符	描述
+	加
-	减
*	乘
/	除
%	取余

```

package com.cxz.operator;
public class operator {
    public static void main(String[] args) {
        //目标：理解基本算数运算符
        operator_demo1();
    }
    //算数运算符
    public static void operator_demo1(){
        int a = 10;
        int b = 3;
        int c = a + b;
        System.out.println(c);
        System.out.println(a-b);
        System.out.println(a*b);
        System.out.println(a/b);
        System.out.println(a%b);
    }
}

```

a/b 因为都是int类型，结果的小数不会保留，但若是写成

System.out.println((double)a/b);或者System.out.println(1.0\*a/b);  
就可以得到小数了

## 2. +符号在Java中的特殊用途

- +符号可做连接符
- +符号与字符串运算，可做拼接符

比如:"abc"+5 -> "abc5"

比如：

```

public static void operator_demo2() {
    int a=5;
    System.out.println("abc"+a);
    System.out.println(a+5);
    System.out.println("fdsfee"+a+"abc");
    System.out.println(a+'a'+"abc");
}

```

输出得到

abc5

10

fdsfee5abc

102abc

可知, 能算的是先与单个字符运算, 可相加, 不能算的是与字符串运算, 只能连接

### 3. ++自增运算符和--自减运算符

运算符	描述
++	自增
--	自减

注意: ++和--在变量前后单独使用无区别

```
public static void operator_demo3(){
    int a=1;
    a++;
    System.out.println(a);
    int b=1;
    ++b;
    System.out.println(b);
}
```

++和--在表达式中或同时有其他操作则放前后有区别

```
public static void operator_demo3(){
    int a=1;
    int d=++a; //先加1再赋值
    System.out.println(d);
    int b=1;
    int e=b++; //先赋值再加1
    System.out.println(e);
    int c=1;
    int m=--c; //先减1再赋值
    System.out.println(m);
    int z=1;
    int n=z--; //先赋值再减1
    System.out.println(n);
} 结果为2,1,0,1
```

### 4. 赋值运算符

- 基本算数运算符 = (从右往左看)

int a=10; 就是把10赋值给a

- 扩展赋值运算符(隐含了强制类型转换, 转换为符号左边的类型)

运算符	用法	作用	底层代码形式
$+=$	$a += b$	加后赋值	$a = a + b$
$-=$	$a -= b$	减后赋值	$a = a - b$
$*=$	$a *= b$	乘后赋值	$a = a * b$
$/=$	$a /= b$	除后赋值	$a = a / b$
$\%=$	$a \%= b$	取余后赋值	$a = a \% b$

```
public static void operator_demo4(){
//收红包
byte a = 100;
byte b=10;
a +=b;
System.out.println(a);
}
```

$a=a+b$  并且其中暗含了强制类型转换，因为a与b都是byte类型，相加会自动转为int类型， $a+=b$ 可以省去自己写强转

## 5. 关系运算符、三元运算符

- 关系运算符

运算符	例子	作用	结果
$>$	$a > b$	判断a是否大于b	成立返回true，否则返回false
$<$	$a < b$	判断a是否小于b	成立返回true，否则返回false
$\geq$	$a \geq b$	判断a是否大于等于b	成立返回true，否则返回false
$\leq$	$a \leq b$	判断a是否小于等于b	成立返回true，否则返回false
$\==$	$a == b$	判断a是否等于b	成立返回true，否则返回false
$\!=$	$a != b$	判断a是否不等于b	成立返回true，否则返回false

```
public static void operator_demo5(){
int a=10;
int b=11;
System.out.println(a>b);
System.out.println(a<b);
System.out.println(a>=b);
System.out.println(a<=b);
System.out.println(a==b);
```

```
System.out.println(a!=b);
}
```

输出为

```
false
true
false
true
false
true
```

- 三元运算符

- 格式：条件表达式？真值：假值

条件表达式为真，返回真值，为假，返回假值

```
public static void operator_demo6(){
int a=10;
int b=40;
int max=a>b?a:b;
System.out.println(max);
}
```

若 $a > b$ ，则 $max = a$ ，若 $a < b$ ，则 $max = b$

可以嵌套

```
public static void operator_demo6(){
int a=10;
int b=40;
int c=50;
int max=a>b?(a>c?a:c):(b>c?b:c); //真真假值也可用条件表达式表示
System.out.println(max);
}
```

## 6. 逻辑运算符

运算符	叫法	例子	运算逻辑
&	逻辑与	$2>1 \& 3>2$	多个条件全为true，结果为true
	逻辑或	$2>1   3>2$	多个条件有一个为true，结果为true
!	逻辑非	$!(2>1)$	取反， !true为false,!false为true
^	逻辑异或	$2>1 ^ 3>2$	前后条件结果相同返回false，不同返回true

运算符	叫法	例子	运算逻辑
&&	短路与	2>1&&3>2	短路, 当第一个条件为false时, 第二个条件不再判断 <b>(不执行)</b> , 直接返回false
	短路或	2>1    3>2	短路, 当第一个条件为true时, 第二个条件不再判断 <b>(不执行)</b> , 直接返回true

```
public static boolean operator_demo7(int height,int weight,char sex){
    boolean result=height>170&&weight>60&&sex=='男';
    return result;
}
```

输出为true

## 五、实战小程序

- 请输入年龄、性别、体重、身高，计算BMI和BMR

```
package com.cxz.demo;
import java.util.Scanner; //导入Scanner类
public class all_test {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in); //创建Scanner对象
        System.out.println("请输入你的年龄： ");
        int age = sc.nextInt();
        System.out.println("请输入你的性别： ");
        String sex = sc.next(); //输入字符串
        System.out.println("请输入你的体重(kg)： ");
        double weight = sc.nextDouble();
        System.out.println("请输入你的身高(m)： ");
        double height = sc.nextDouble();
        System.out.println("你的BMI是：" + BMI(weight,height)); //+号表示连接
        System.out.println("你的BMR是：" + BMR(weight,height,age,sex));
    }
    public static double BMI(double weight,double height){
        double bmi=weight/(height*height);
        return bmi; //返回值
    }
    public static double BMR(double weight,double height,int age,String sex){
        double bmr=0.0;
        if("男".equals(sex)){ //equals()判断字符串是否相等
            bmr=66+(13.7*weight)+(5*height)-(6.76*age);
        }
        return bmr;
    }
}
```

```
bmr=88.362+(13.397weight)+(4.799100height)-(5.677age);  
}  
else if("女".equals(sex)){  
bmr=447.593+(9.247weight)+(3.098100height)-(4.330*age);  
}  
return bmr;  
}  
}
```