Day 05 方法(method)

今日内容

1.方法介绍
2.定义方法
3.使用方法
4.方法重载
5.递归
6.字符串
7.成员变量与局部变量区别

方法(函数)method

- 函数的定义
- 函数的特点
- 函数的应用
- 函数的重载

什么是函数?

我们编写程序其实就是不断实现功能,Java 中最小的功能单元就是函数

函数就是定义在类中的具有特定功能的一段独立小程序。

函数也称为方法。

1.1 函数的定义

函数的格式:

修饰符 返回值类型 函数名(参数类型 形式参数 1，参数类型 形式参数 2，)

{

执行语句;

return 返回值;

}

返回值类型：函数运行后的结果的数据类型。

参数类型：是形式参数的数据类型。

形式参数：是一个变量，用于存储调用函数时传递给函数的实际参数。

实际参数：传递给形式参数的具体数值。

return：用于结束函数。

返回值：该值会返回给调用者。

1.2 函数的特点

*定义函数可以将功能代码进行封装

*便于对该功能进行复用

* 函数只有被调用才会被执行

* 函数的出现提高了代码的复用性

*对于函数没有具体返回值的情况，返回值类型用关键

字 **void** 表示，那么该函数中的 **return** 语句如果在最后一行可以省略不写。

注意：

函数中只能调用函数，不可以在函数内部定义函数。

定义函数时，函数的结果应该返回给调用者，交由调用者处理。

1.3 函数的应用

两个明确

1.明确要定义的功能最后的结果是什么？

2.明确在定义该功能的过程中，是否需要未知内容参与运算

示例：

需求：定义一个功能，可以实现两个整数的加法运算。

分析：

该功能的运算结果是什么？两个数的和，也是一个整数**(int)**

在实现该功能的过程中是否有未知内容参与运算？加数和被加数是不确

定的。**(两个参数 int，int)**

代码：

```
int getSum(int x,int y)

{

return x+y;

}
```

1.4 函数的重载(overload)

重载的概念:在同一个类中，允许存在一个以上的同名函数，只要它们的参数个数或者参数类型不同即可。

重载的特点：与返回值类型无关，只看参数列表。

重载的好处：方便于阅读，优化了程序设计。

重载示例：

```
//返回两个整数的和

int add(int x,int y){return x+y;}

//返回三个整数的和

int add(int x,int y,int z){return x+y+z;}

//返回两个小数的和

double add(double x,double y){return x+y;}
```

方法重载的注意事项

1. 参数列表必须不同

2. 重载和参数变量名无关

3. 重载和返回值类型无关

4. 重载和修饰符无关

技巧: 重载看方法名和参数列表

1.5 递归结构

递归，指在当前方法内调用自己的这种现象

```
public void method(){
    System.out.println(“递归的演示”);
    //在当前方法内调用自己
    method();
}
```

递归分为两种，直接递归和间接递归。

直接递归称为方法自身调用自己。间接递归可以 A 方法调用 B 方法，B 方法调用 C 方法，C 方法调用 A 方法。

递归结构包括两个部分：

① 定义递归头。解答：什么时候不调用自身方法。如果没有头，将陷入死循环。

② 递归体。解答：什么时候需要调用自身方法。

递归的缺陷：

简单的程序是递归的优点之一。但是递归调用会占用大量的系统堆栈，内存耗用多，在递归调用层次多时速度要比循环慢的多。所以再使用时要慎重。

注意：任何可用递归解决的问题也能使用迭代解决。但递归方法可以更加自然地反映问题，并且易于理解和调试，并且不强调效率问题时，可以采用递归；

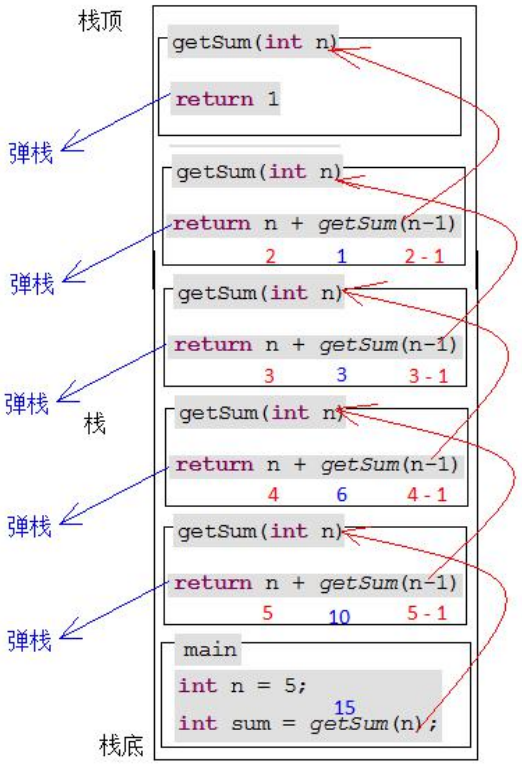
在要求高性能的情况下尽量避免使用递归，递归调用既花时间又耗内存。

- 递归的代码演示，计算 1-n 之间的和，使用递归完成

```
public class DiGuiDemo {

    public static void main(String[] args) {

            //计算 1~num 的和，使用递归完成

            int n = 5;

            int sum = getSum(n);

            System.out.println(sum);


    }

    public static int getSum(int n) {

            if(n == 1){

                    return 1;

            }

            return n + getSum(n-1);

    }

}
```

- **代码执行流程图解**

```
public class DiGuiDemo {
    public static void main(String[] args) {
        //计算1~num的和，使用递归完成
        int n = 5;
        int sum = getSum(n);
        System.out.println(sum);

    }
    public static int getSum(int n) {
        if(n == 1){
            return 1;
        }
        return n + getSum(n-1);
    }
}
```



注意：递归一定要有条件限定，保证递归能够停止下来，否则会发生栈内存溢出。

在递归中虽然有限定条件，但是递归次数不能太多。否则也会发生栈内存溢出。

1.6 String 常用方法

String 为不可变的字符序列，常用的方法为:

1.**length();** 长度

2.**charAt(索引);** 获取索引[0,length()]对应的单个字符

3.**equals(其他字符串):**比较两个字符串内容是否相等

equalsIgnoreCase(其他字符串):忽略大小写比较两个字符串内容是否相等

4.**substring([起始索引,结束索引])** :截取字符串

5.**indexOf(“字符串”,[起始索引]):**默认从 0 开始,查找子串第一次出现的位置

6.**replaceAll(“旧字符串”,”新字符串”):**替代所有的字符串为新的字符串

7.**+拼接:**不是追加

8.**trim():**-->去除左右所有空格-->String

1.7 成员变量与局部变量的区别

(1)在类中的位置不同

成员变量：类中方法外

局部变量：方法定义中或者方法声明上

(2)在内存中的位置不同

成员变量：在堆中

局部变量：在栈中

(3)生命周期不同

成员变量：随着对象的创建而存在，随着对象的消失而消失

局部变量：随着方法的调用而存在，随着方法的调用完毕而消失， 从定义开始，到离他最近的右大括号结束

(4)初始化值不同

成员变量有默认值，如下：

成员变量类型	取值
byte	0
short	0
int	0
long	0L
char	'\u0000'
float	0.0F
double	0.0D
boolean	false
所有引用类型	null

局部变量：没有默认值，必须定义，赋值，然后才能使用

Scanner 中 next()与 nextLine 区别

next 与 nextLine 的区别:

next:一定要读取到有效字符后才可以结束输入，对输入有效字符前遇到的空格键、Tab 键或 Enter 键等结束符，next()方法会自动将其去掉，只有在输入有效字符后，next()方法才将其后输入的空格键、Tab 键或 Enter 键视为分隔符或结束符。简单地说，next()查找 并返回来自扫描器的下一个完整标记。完整标记的前后是与分隔符模式匹配的输入信息，所以 next()方法不能得到带空格的字符串

nextLine()方法的结束只是 Enter 键，即 nextLine()方法返回的是 Enter 键之前的所有字符，它是可以得到带空格的字符串的

```
Scanner sc=new Scanner(System.in);
String s1,s2;
System.out.println("请输入第一个字符串:");
s1=sc.nextLine();
System.out.println("请输入第二个字符串:");
```

```
        s2=sc.next();
        System.out.println("输入的字符串是"+s1+" "+s2);
```

控制台输出：

```
请输入第一个字符串:
home
请输入第二个字符串:
work
输入的字符串是 home work
```

若把程序改一下:

```
s1=next();

s2=nextLint();
```

控制台输出：

```
请输入第一个字符串:
hello
请输入第二个字符串:
输入的字符串是 hello
```

注意：nextLine()自动读取了被 next()去掉的 Enter 作为他的结束符，所以 s2 无法从键盘输入值。其他的 next 的方法，如 nextDouble()、nextFloat()、nextInt()等与 nextLine()连用时都存在该问题，解决办法：在这些语句之后加一个 nextLine()语句，将被 next()去掉的 Enter 结束符过滤掉

```
        Scanner sc=new Scanner(System.in);
        String s1,s2;
        System.out.println("请输入第一个字符串:");
        s1=sc.next();
        sc.nextLine();
        System.out.println("请输入第二个字符串:");
        s2=sc.nextLine();
        System.out.println("输入的字符串是"+s1+" "+s2);
```

控制台输出：

```
请输入第一个字符串:
home
请输入第二个字符串:
work
```

输入的字符串是 home work

1.8 今日小结

要求：

掌握方法

掌握重载

理解递归

预习内容：

1、成员变量、局面变量

2、数组

3、面向对象

任务：

看视频

复习前面的内容、预习后面的内容