狂神社群笔记资料，禁止外传，本人QQ：24736743

**方法**

1**、何谓方法？**

在前面几个章节中我们经常使用到 System.out.println()，那么它是什么呢？

println() 是一个方法。   
System 是系统类。

out 是标准输出对象。

这句话的用法是调用系统类 System 中的标准输出对象 out 中的方法 println()。   
**那么什么是方法呢？**

Java方法是语句的集合，它们在一起执行一个功能。

方法是解决一类问题的步骤的有序组合   
方法包含于类或对象中   
方法在程序中被创建，在其他地方被引用

设计方法的原则：方法的本意是功能块，就是实现某个功能的语句块的集合。我们设计方法的时候，最 好保持方法的原子性，就是一个方法只完成1个功能，这样利于我们后期的扩展。

**方法的优点**   
使程序变得更简短而清晰。

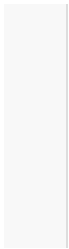
有利于程序维护。   
可以提高程序开发的效率。   
提高了代码的重用性。

**回顾：方法的命名规则？**

2**、方法的定义**

Java的方法类似于其它语言的函数，是一段用来完成特定功能的代码片段，一般情况下，定义一个方法 包含以下语法：

1

2

3

4

5

6

修饰符 返回值类型 方法名(参数类型 参数名){   ...

   方法体   
   ...

   return 返回值;

}

方法包含一个方法头和一个方法体。下面是一个方法的所有部分：   
**修饰符：**修饰符，这是可选的，告诉编译器如何调用该方法。定义了该方法的访问类型。

**返回值类型** **：**方法可能会返回值。returnValueType 是方法返回值的数据类型。有些方法执行所需 的操作，但没有返回值。在这种情况下，returnValueType 是关键字void。   
**方法名：**是方法的实际名称。方法名和参数表共同构成方法签名。   
**参数类型：**参数像是一个占位符。当方法被调用时，传递值给参数。这个值被称为实参或变量。参 数列表是指方法的参数类型、顺序和参数的个数。参数是可选的，方法可以不包含任何参数。

形式参数：在方法被调用时用于接收外界输入的数据。 实参：调用方法时实际传给方法的数据。

狂神社群笔记资料，禁止外传 狂神社群笔记资料，禁止外传，本人QQ：24736743

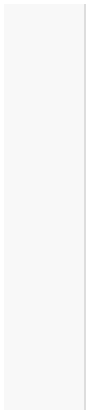
狂神社群笔记资料，禁止外传，本人QQ：24736743

**方法体：**方法体包含具体的语句，定义该方法的功能。 比如我们写一个比大小的方法：

【演示】下面的方法包含 2 个参数 num1 和 num2，它返回这两个参数的最大值。

1

2

3

4

5

6

7

8

9

10

/\*\* 返回两个整型变量数据的较大值 \*/

public static int max(int num1, int num2) {   int result;

  if (num1 > num2)

     result = num1;

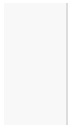
  else

     result = num2;

  return result;

}

【演示：加法】

1

2

3

public int add(int num1, int num2) {   return num1+num2;

}

3**、方法调用**

Java 支持两种调用方法的方式，根据方法是否返回值来选择。 当程序调用一个方法时，程序的控制权交给了被调用的方法。当被调用方法的返回语句执行或者到达方 法体闭括号时候交还控制权给程序。

当方法返回一个值的时候，方法调用通常被当做一个值。例如：

1 int larger = max(30, 40);

Java语言中使用下述形式调用方法：对象名.方法名(实参列表)   
如果方法返回值是void，方法调用一定是一条语句。例如，方法println返回void。下面的调用是个语 句：

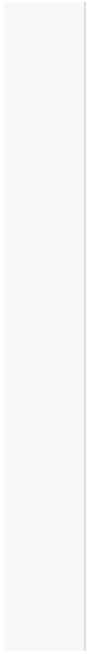
1 System.out.println("Hello,kuangshen!");

【演示：定义方法并且调用它】

1

2

3

4

5

6

7

8

9

10 11 12 13 14 15 16

public static void main(String[] args) {

   int i = 5;

   int j = 2;

   int k = max(i, j);

   System.out.println( i + " 和 " + j + " 比较，最大值是：" + k);

}

/\*\* 返回两个整数变量较大的值 \*/

public static int max(int num1, int num2) {    int result;

   if (num1 > num2)

       result = num1;

   else

       result = num2;

   return result;

}

狂神社群笔记资料，禁止外传 狂神社群笔记资料，禁止外传，本人QQ：24736743

狂神社群笔记资料，禁止外传，本人QQ：24736743



这个程序包含 main 方法和 max 方法。main 方法是被 JVM 调用的，除此之外，main 方法和其它方法 没什么区别。JAVA**中只有值传递！**

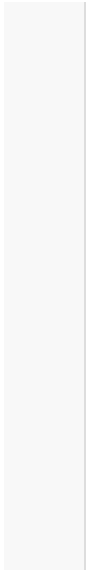
main 方法的头部是不变的，如例子所示，带修饰符 public 和 static,返回 void 类型值，方法名字是 main,此外带个一个 String[] 类型参数。String[] 表明参数是字符串数组。

4**、方法的重载**

上面使用的max方法仅仅适用于int型数据。但如果你想得到两个浮点类型数据的最大值呢？ 解决方法是创建另一个有相同名字但参数不同的方法，如下面代码所示：

1

2

3

4

5

6

7

8

9

10 11 12 13 14

public static double max(double num1, double num2) {  if (num1 > num2)

   return num1;

 else

   return num2;

}

public static int max(int num1, int num2) {   
    int result;

   if (num1 > num2)

       result = num1;

   else

       result = num2;

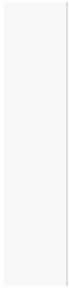
   return result;

}

如果你调用max方法时传递的是int型参数，则 int型参数的max方法就会被调用；   
如果传递的是double型参数，则double类型的max方法体会被调用，这叫做方法重载； 就是说一个类的两个方法拥有相同的名字，但是有不同的参数列表。   
Java编译器根据方法签名判断哪个方法应该被调用。 方法重载可以让程序更清晰易读。执行密切相关任务的方法应该使用相同的名字。 重载的方法必须拥有不同的参数列表。你不能仅仅依据修饰符或者返回类型的不同来重载方法。   
5**、拓展命令行传参**

有时候你希望运行一个程序时候再传递给它消息。这要靠传递命令行参数给main()函数实现。 命令行参数是在执行程序时候紧跟在程序名字后面的信息。   
【下面的程序打印所有的命令行参数】

1

2

3

4

5

6

7

public class CommandLine {

  public static void main(String args[]){

     for(int i=0; i<args.length; i++){   
         System.out.println("args[" + i + "]: " + args[i]);     }

  }

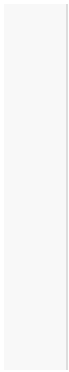
}

【命令行】

狂神社群笔记资料，禁止外传 狂神社群笔记资料，禁止外传，本人QQ：24736743

狂神社群笔记资料，禁止外传，本人QQ：24736743

1

2

3

4

5

6

7

8

9

$ javac CommandLine.java

$ java CommandLine this is a command line 200 -100 args[0]: this

args[1]: is

args[2]: a

args[3]: command

args[4]: line

args[5]: 200

args[6]: -100

【错误: 找不到或无法加载主类，解决方法】 在项目输出的项目目录下执行java命令，写完整路径即可。

1 $ java com.kuang.chapter3.Demo03 Hello World

6**、可变参数**

JDK 1.5 开始，Java支持传递同类型的可变参数给一个方法。   
方法的可变参数的声明如下所示：

1 typeName... parameterName

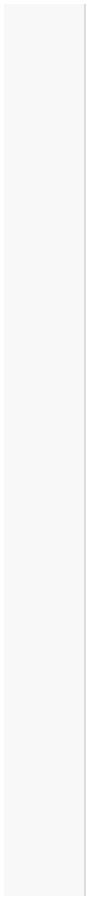
在方法声明中，在指定参数类型后加一个省略号(...) 。 一个方法中只能指定一个可变参数，它必须是方法的最后一个参数。任何普通的参数必须在它之前声 明。

1

2

3

4

5

6

7

8

9

10 11 12 13 14 15 16 17 18 19 20 21 22

public static void main(String args[]) {    // 调用可变参数的方法

   printMax(34, 3, 3, 2, 56.5);   
    printMax(new double[]{1, 2, 3});

}

public static void printMax( double... numbers) {   
    if (numbers.length == 0) {   
        System.out.println("No argument passed");        return;

  }

   double result = numbers[0];

   //排序！

   for (int i = 1; i <  numbers.length; i++){   
        if (numbers[i] >  result) {   
            result = numbers[i];

      }

  }

   System.out.println("The max value is " + result);

}

7**、递归**

A方法调用B方法，我们很容易理解！   
递归就是：A方法调用A方法！就是自己调用自己，因此我们在设计递归算法时，一定要指明什么时候自 己不调用自己。否则，就是个死循环！

狂神社群笔记资料，禁止外传 狂神社群笔记资料，禁止外传，本人QQ：24736743

狂神社群笔记资料，禁止外传，本人QQ：24736743

**递归算法重点：**   
递归是一种常见的解决问题的方法，即把问题逐渐简单化。递归的基本思想就是“自己调用自己”，一个 使用递归技术的方法将会直接或者间接的调用自己。

利用递归可以用简单的程序来解决一些复杂的问题。它通常把一个大型复杂的问题层层转化为一个与原 问题相似的规模较小的问题来求解，递归策略只需少量的程序就可描述出解题过程所需要的多次重复计 算，大大地减少了程序的代码量。递归的能力在于用有限的语句来定义对象的无限集合。

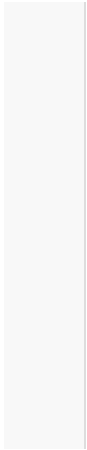
递归结构包括两个部分：

1. 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。   
【演示：利用代码计算5的乘阶！】

1

2

3

4

5

6

7

8

9

10 11

//5\*4\*3\*2\*1

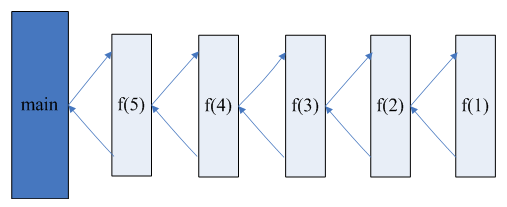
public static void main(String[] args) {    System.out.println(f(5));

}

public static int f(int n) {    if (1 == n)   
        return 1;

   else

       return n\*f(n-1);

} 

此题中，按照递归的三个条件来分析：   
(1)边界条件：阶乘，乘到最后一个数，即1的时候，返回1，程序执行到底；   
(2)递归前进段：当前的参数不等于1的时候，继续调用自身；   
(3)递归返回段：从最大的数开始乘，如果当前参数是5，那么就是5 4，即5 (5-1)，即n \* (n-1)

递归其实是方便了程序员难为了机器，递归可以通过数学公式很方便的转换为程序。其优点就是易理 解，容易编程。但递归是用栈机制实现的，每深入一层，都要占去一块栈数据区域，对嵌套层数深的一 些算法，递归会力不从心，空间上会以内存崩溃而告终，而且递归也带来了大量的函数调用，这也有许 多额外的时间开销。所以在深度大时，它的时空性就不好了。（会占用大量的内存空间） 而迭代虽然效率高，运行时间只因循环次数增加而增加，没什么额外开销，空间上也没有什么增加，但 缺点就是不容易理解，编写复杂问题时困难。

**能不用递归就不用递归，递归都可以用迭代来代替。**

**总结和作业**

**总结：**

狂神社群笔记资料，禁止外传 狂神社群笔记资料，禁止外传，本人QQ：24736743

狂神社群笔记资料，禁止外传，本人QQ：24736743

用户交互Scanner   
顺序结构   
选择结构   
循环结构

break & continue   
方法

**作业：** 写一个计算器，要求实现加减乘除功能，并且能够循环接收新的数据，通过用户交互实现。思路推荐：

1. 写4个方法：加减乘除

2. 利用循环+switch进行用户交互   
3. 传递需要操作的两个数

4. 输出结果

狂神社群笔记资料，禁止外传 狂神社群笔记资料，禁止外传，本人QQ：24736743