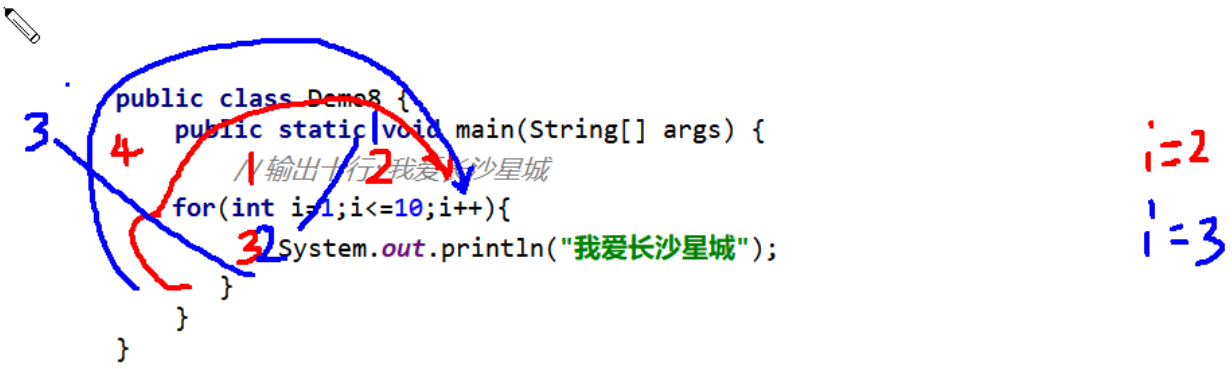
●Java 语言基础(循环结构概述和 for 语句的格式及其使用)

- 需求：要求控制台输出 10 行“我爱长沙星城”

```
public class Demo8 {
    public static void main(String[] args) {
        //输出十行我爱长沙星城
       System.out.println("我爱长沙星城");
         System.out.println("我爱长沙星城");
         System.out.println("我爱长沙星城");
         System.out.println("我爱长沙星城");
         System.out.println("我爱长沙星城");
         System.out.println("我爱长沙星城");
         System.out.println("我爱长沙星城");
         System.out.println("我爱长沙星城");
         System.out.println("我爱长沙星城");
         //输出一万行
    }
}
```

```
public class Demo8 {
    public static void main(String[] args) {
        //输出十行:我爱长沙星城
      for(int i=1;i<=10;i++){
            System.out.println("我爱长沙星城");
       }
    }
}
```



首次是按红色数字顺序运行，从第二次开始到结束一直按蓝色的数字顺序运行

- A:循环结构的分类

  for, while, do...while

  B:循环结构 for 语句的格式：

```
for(初始化表达式;条件表达式;循环后的操作表达式){
    循环体;
}
```

```
public class Demo8 {
```

```
    public static void main(String[] args) {
        //输出十行:我爱长沙星城
//        int i=1;//局部变量
      for(int i=1;i<=10;i++){  //在for循环里的变量i的有效范围只在for循环体中有效，当for循环运行结束，i就在内存中被及时释放，可以节约内存的使用
          System.out.println("我爱长沙星城");
       }
//        System.out.println("i="+i);//出了for循环变量i无法使用
   }
}
```

C 执行流程：
a:执行初始化语句
b:执行判断条件语句,看其返回值是 true 还是 false
　　如果是 true，就继续执行
　　如果是 false，就结束循环
c:执行循环体语句;
d:执行循环后的操作表达式
e:回到 for 继续。
D:案例演示
　　在控制台输出 10 次"山上有座庙，庙里有老和尚和小和尚，老和尚给小和尚讲故事，说"

```
public class Demo8 {
    public static void main(String[] args) {
       for(int i=0;i<10;i++){
           System.out.println("山上有座庙，庙里有老和尚和小和尚，老和尚给小和尚讲故事，说");
       }
    }
}
```

●循环结构 for 语句的练习之获取数据

A:案例演示
　　需求 1：请在控制台输出数据 1-10
　　需求 2：请在控制台输出数据 10-1

```
public class Demo8 {
    public static void main(String[] args) {
      //输出1~10
       for(int i=1;i<=10;i++){
           System.out.print(i +" ");
       }
        System.out.println();//输出换行
      //输出10~1
       for(int i=10;i>=1;i--){
            System.out.print(i+" ");
        }
    }
```

```
}
```

B:注意事项

判断条件语句无论简单还是复杂结果是 boolean 类型。

●循环结构 for 语句的练习之求和思想

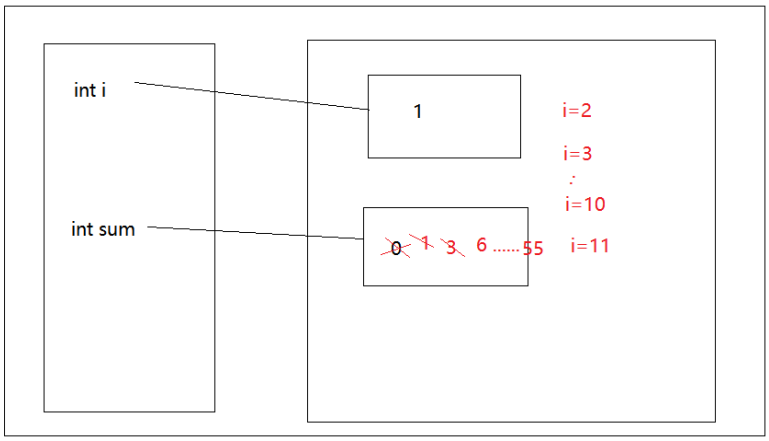
A:案例演示

需求：求出 1-10 之间数据之和

```
public class Demo8 {
    public static void main(String[] args) {
        //求 1~10 中所有数的和
      int sum=0;//声明一个变量用来存储求和的结果，这个 sum 变量所在的位置要放在 for 循环之外

      for(int i=1;i<=10;i++){
            sum += i;
        }
        System.out.println("求和的结果是："+sum);
    }
}
```

```
public class Demo8 {
    public static void main(String[] args) {
        //求1~10中所有数的和
     int sum=0;//声明一个变量用来存储求和
    for(int i=1;i<=10;i++){
//          System.out.print(i+" ");
        sum += i;
    }
        System.out.println("求和的结果是："+sum);
    }
}
```



B:学生

需求：求出 1-100 之间偶数和

练习：求出 1-100 之间奇数和

```
public class Demo1 {
    public static void main(String[] args) {
        //求出 1~100 之间的所有数的和
      //申明接收求和结果的变量
      int sum=0;
        int sum2= 0;//用来接收奇数的和
      for(int a=1;a<=100;a++){
            //求偶数的和
          if(a % 2 == 0) {  //能被 2 整除的是偶数
              sum += a;
```

```
            }else{
                //接收奇数的和
                sum2 += a;
            }
        }
        System.out.println("偶数的和："+sum);
        System.out.println("奇数的和："+sum2);
    }
}
```

●循环结构 for 语句的练习之水仙花

A:案例演示

需求：在控制台输出 100 ~ 999 所有的"水仙花数"

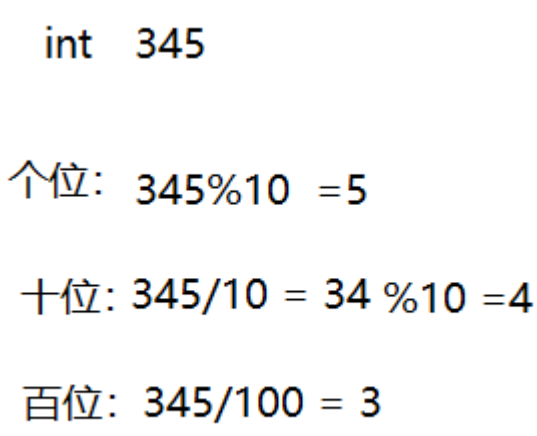
所谓的水仙花数是指一个三位数，其各位数字的立方和等于该数本身。

举例：153 就是一个水仙花数。

153 = 1*1*1 + 5*5*5 + 3*3*3 = 1 + 125 + 27 = 153

```
public class Demo2 {
    public static void main(String[] args) {
        for(int i=100;i<1000;i++){
            //获取个十百每一位上的数字
          int ge = i%10;
            int shi = i/10%10;
            int bai = i/100;

            if(ge*ge*ge + shi*shi*shi+bai*bai*bai == i){
                System.out.println(i);
            }
        }
    }
}
```



●循环结构 for 语句的练习之统计思想

A:案例演示

需求：统计 100 ~ 1000 所有的”水仙花数”共有多少个

```
public class Demo2 {
    public static void main(String[] args) {
        //申明一个变量来对水仙花进行计数，要写在循环外
      int num =0;

        for(int i=100;i<1000;i++){
            //获取个十百每一位上的数字
          int ge = i%10;
            int shi = i/10%10;
            int bai = i/100;

            if(ge*ge*ge + shi*shi*shi+bai*bai*bai == i){
                System.out.println(i);
//                num++;
               ++num;  //满足水仙花的条件就对 num 作 num+1 运算
          }
        }
        System.out.println("水仙花数是："+num);


    }
}
```

●while 语句的格式和基本使用

A:循环结构 while 语句的格式：

```
初始化语句;
while(判断条件语句) {
      循环体语句;
      控制条件语句;
}
```

B:执行流程：

1:执行初始化语句

2:执行判断条件语句,看其返回值是 true 还是 false

如果是 true，就继续执行

如果是 false，就结束循环

3:执行循环体语句;

4:执行控制条件语句

C:案例演示

需求：用 while 循环在控制台输出数据：1-10

```
public class Demo1 {
```

```
    public static void main(String[] args) {
        //输出1~10
       int i =1;
        while(i<=10){  //当i<=10
           System.out.print(i+" ");
            i++;//控制循环的条件
       }
    }
}
```

```
public class Demo1 {
    public static void main(String[] args) {
    //输出10~1
        int i =10;
        while(i>=1){
            System.out.print(i+" ");
            i--;//控制循环的条件
       }
    }
}
```

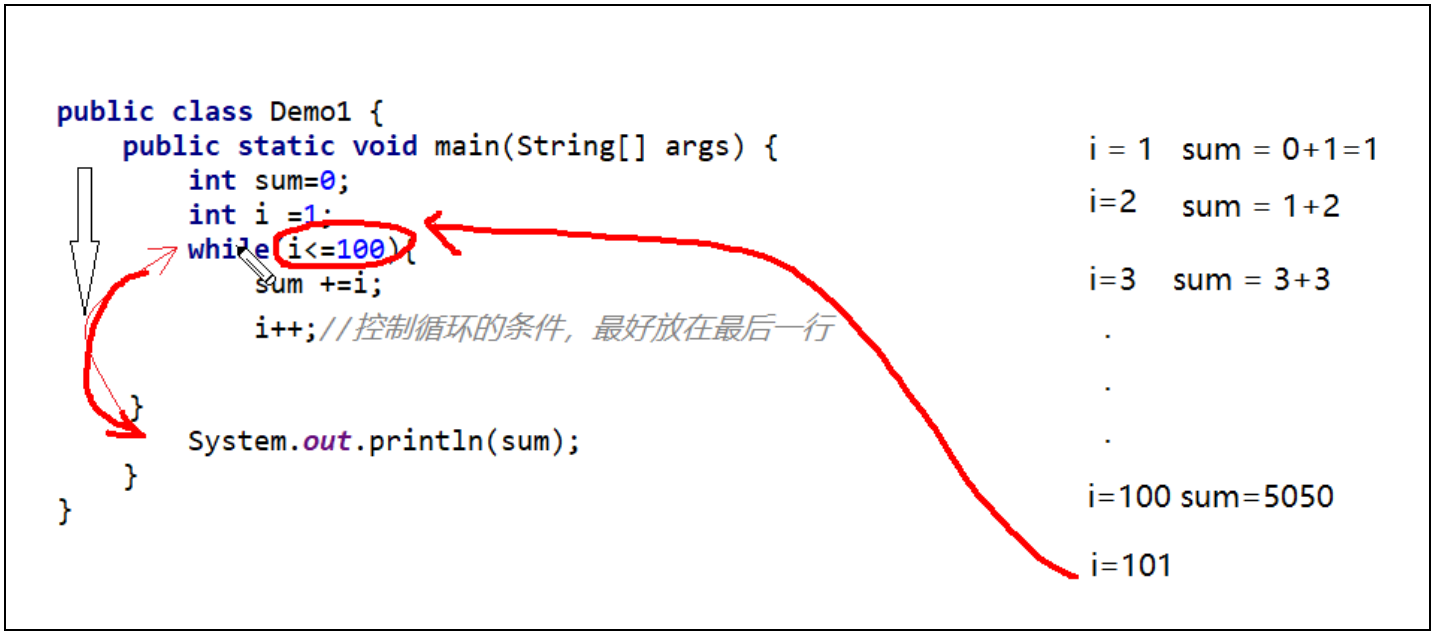
●while 语句的练习

A:求和思想

求 1-100 之和

```
public class Demo1 {
    public static void main(String[] args) {
        int sum=0;
        int i =1;
        while(i<=100){
            sum +=i;
            i++;//控制循环的条件，最好放在最后一行

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



B:统计思想

统计 100~1000”水仙花数”共有多少个

```
public class Demo1 {
    public static void main(String[] args) {
        //声明一个记录个数的变量
       int num = 0;
         int n = 100;
         while(n<1000){
             int ge = n%10;
             int shi = n/10%10;
             int bai = n/100%10;
             if(ge*ge*ge+shi*shi*shi+bai*bai*bai == n){
                 num++;
             }
             //控制循环的条件
           n++;
         }
         System.out.println("水仙花数是："+num);


         System.out.println("------------for 循环也可以这么写，了解即可，开发中不推荐这么写
--------------");
         int num2 = 0;
         int b=100;
         for(;b<1000;){
             int ge = b%10;
             int shi = b/10%10;
             int bai = b/100%10;
             if(ge*ge*ge+shi*shi*shi+bai*bai*bai == b){
                 num2++;
             }
             b++;
         }
```

```
        System.out.println(num2);
    }
}
```

●循环结构 do...while 语句的格式和基本使用

A:循环结构 do...while 语句的格式：

```
初始化语句;
do {
    循环体语句;
    控制条件语句;
}while(判断条件语句);
```

B:执行流程：

1:执行初始化语句

2:执行循环体语句;

3:执行控制条件语句

4:执行判断条件语句,看其返回值是 true 还是 false

如果是 true，就继续执行

如果是 false，就结束循环

C:案例演示

需求：请在控制台输出数据 1-10

```
public class Demo1 {
    public static void main(String[] args) {
        int n=19;
        do{   //先干了再说（先斩后奏）
          System.out.println(n);
             n++;
        }while (n<=10);
/*
do_while 无论是否满足条件，都会运行一次（将在外，军命有所不为）
 */

    }
}
```

特点：

无论是否满足条件都会输出一次结果

●循环结构三种循环语句的区别

案例演示

A:案例演示

三种循环语句的区别:

do...while 循环至少执行一次循环体。
而 for,while 循环必须先判断条件是否成立，然后决定是否执行循环体语句。

for 循环和 while 循环的区别：
A:如果你想在循环结束后，继续使用控制条件的那个变量，用 while 循环，否则用 for 循环。不知道用谁就用 for 循环。因为变量及早的从内存中消失，可以提高内存的使用效率。

```
import java.util.Scanner;

public class Demo1 {
    public static void main(String[] args) {
        for(int i=1;i<=10;i++){
            System.out.print(i+" ");
        }
//        System.out.println(i);//在for 循环之后，循环中的i 会被及时释放内存
       System.out.println();

        System.out.println("------------");
        int num=1;
        while(num<=10){
            System.out.print(num+",");
            num++;
        }
        System.out.println();
        System.out.println(num);//while 循环用过的num 是可以继续使用的,如果你有需要继续使用循环的初始化变量的时候，选择while 循环
    }
}
```

●注意事项：死循环

A:一定要注意控制条件语句控制的那个变量的问题，不要弄丢了，否则就容易死循环。
B:两种最简单的死循环格式
while(true){
}
for(;;){
}

```
public class Demo1 {
    public static void main(String[] args) {
        if(true) {
            System.out.print(1);
        }
//        for(;;){
//            System.out.println(2);
//        }

//        System.out.println("死循环之后的程序没法到达");//开发中一定要避免出现死循环。程序员开发的禁忌
```

```
    }
}
```

```
import java.util.Scanner;

public class Demo1 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a =sc.nextInt();//用户在操作的时候，输入的信息，构成了死循环
      if(a!=1000) {   //解决：避免用户输入1000
            while (a == 1000) {
                System.out.print(1);
            }
        }
//        for(;;){
//            System.out.println(2);
//        }

//        System.out.println("死循环之后的程序没法到达");//开发中一定要避免出现死循环。程序员开发的禁忌
    }
}
```

●循环结构循环嵌套输出 4 行 5 列的星星

A:案例演示

需求：请输出 4 行 5 列图案

```
*****
*****
*****
*****
```

B:结论：

外循环控制行数，内循环控制列数

```
public class Demo1 {
    public static void main(String[] args) {
        for(int j=0;j<5;j++) {   //外层循环，控制重复执行下面的五次

          //是输出一行五个*，把这里看作一个整体
          for (int i = 0; i < 5; i++) {
                System.out.print("*");
          }
          System.out.println();//输出换行

      }
    }
}
```

●循环结构循环嵌套输出正三角形

需求：请输出下列的形状

```
*
**
***
****
*****
```

方法一：

```
public class Demo1 {
    public static void main(String[] args) {
        int n=1;//控制每行输出的个数
      for(int j=0;j<5;j++) {   //外层循环，控制重复执行下面的五次

          //是输出一行五个*，把这里看作一个整体
          for (int i = 0; i < n; i++) {
                System.out.print("*");
            }

            System.out.println();//输出换行
          n++;//放在外层循环中

      }
    }
}
```

方法二

```
public class Demo1 {
    public static void main(String[] args) {
        for(int j=1;j<=5;j++) {   //外层循环，控制重复执行下面的五次

          //是输出一行五个*，把这里看作一个整体
          for (int i = 0; i <j; i++) {  //j 整合记录的行数，每行*的个数=行数
             System.out.print("*");
            }

            System.out.println();//输出换行
      }
    }
}
```

●循环结构九九乘法表

- A:案例演示
  - 需求：在控制台输出九九乘法表。

```
public class Demo1 {
    public static void main(String[] args) {
        for(int j=1;j<=9;j++) {   //外层循环，控制重复执行下面的五次

          //是输出一行五个*，把这里看作一个整体
          for (int i = 1; i <=j; i++) {  //j 整合记录的行数，每行*的个数=行数
              System.out.print(i+"*"+j+"="+i*j+"  ");
            }

            System.out.println();//输出换行
      }
    }
}
```

- B:代码优化
- 注意：
- '\x' x 表示任意，\是转义符号,这种做法叫转移字符。
- '\t' tab 键的位置 （单引号，双引号都可以）
- '\r' 回车
- '\n' 换行
- '\"'
- '\''

```
public class Demo2 {
    public static void main(String[] args) {
//        System.out.println("\"");  //在这里\是转义符，它的后面接要被转义的字符
//        System.out.print("你好");
//        System.out.print("\r");  //回车，用在 linux（作服务器用）系统中使用
//        System.out.print("我很好");
//        System.out.println("\n"); //回车，支持 windows 和 mac 系统
      for(int i=0;i<10;i++){
            System.out.print(i);
            System.out.print("\n");//回车
            //System.out.print("\n\r");//这样写可以支持 windows、mac、linux 等几乎所有的操作系统
      }
        System.out.println();//这也是换行
      for(int i=1;i<20;i++){
            System.out.print(i+"\t");  //  \t 代表制表符等同于 Tab 键
      }
    }
}
```

windows 里面换行\n 都支持，它来自于早期的键盘打字机
linux 系统换行支持\r mac 系统换行支持\n

●控制跳转语句 break 语句

- A:break 的使用场景
  - 只能在 switch 和循环(loop)中

```
public class Demo2 {
    public static void main(String[] args) {
        for(int i=1;i<=10;i++){
            if(i==5){
                break;//中止当前循环
          }
            System.out.println(i);
        }

        System.out.println("循环之后的程序");
    }
}
```

break 只是中止循环（跳出循环体），但整个程序会继续运行

●控制跳转语句标号（了解）

- 标号:标记某个循环对其控制
- 标号组成规则:其实就是合法的标识符

```
public class Demo3 {
    public static void main(String[] args) {
        a:for(int i=1;i<=5;i++){  //外层循环,a 可以随便取名只要满足标识符的要求
          b:for (int j=1;j<=5;j++){  //内层循环，b 可以随便取名只要满足标识符的要求
              System.out.print("j="+j+"\t");
                if(j==2){
                    break b;  //指定跳出的循环
              }
            }
            System.out.println();
            System.out.println("i="+i);
        }
    }
}
```

●控制跳转语句 continue 语句（了解）

- A:continue 的使用场景(中止本次循环，继续下一次循环)
  - 只能在循环(loop)中

```
public class Demo3 {
    public static void main(String[] args) {
       for(int i=0;i<10;i++){
           if(i==4){
               continue; //跳出本次（i=4）循环，继续下一次循环
```

```
        }
            System.out.print(i+"\t");
        }
    }
}
```

●控制调整语句练习

A:练习题（面试题）

要求：
1.在控制台输出 3 次:"Java 班"
2.在控制台输出 7 次:"Java 班"
3.在控制台输出 13 次:"Java 班"

```
    for(int x=1; x<=10; x++) {
        if(x%3==0) {
            //在这里写代码
        }
        System.out.println("Java 班");
    }
```

输出 3 次

```
public class Demo3 {
    public static void main(String[] args) {
        for(int x=1; x<=10; x++) {
            if(x%3==0) {
                //在这里写代码
                System.out.println("java 班");
                break;//break 后面的代码没有机会运行
            }
            System.out.println("Java 班");
        }
    }
}
```

输出 7 次

```
public class Demo3 {
    public static void main(String[] args) {
        for(int x=1; x<=10; x++) {
            if(x%3==0) {
                //在这里写代码
                continue;//跳出本次循环，当 i=3,6,9 的时候跳出循环，那么就输出的 10 次中少了 3 次
            }
            System.out.println("Java 班");
        }
    }
}
```

```
输出 13 次
public class Demo3 {
    public static void main(String[] args) {
        for(int x=1; x<=10; x++) {
            if(x%3==0) {
                //在这里写代码
             System.out.println("java 班");
            }
            System.out.println("Java 班");
        }
    }
}
```

●控制跳转语句 return 语句

A:return 的作用

返回

其实它的作用不是结束循环的，而是结束方法的。

案例演示

```
public class Demo3 {
    public static void main(String[] args) {
        for(int x=1; x<=10; x++) {
            System.out.println(x);
            if(x==5){
                return;//返回，结束的是整个所在的方法，return 后面不要有程序了，因为到达不了
          }
        }

        System.out.println("循环之后的程序");//这句话不会被执行了
    }
}
```

return 和 break 以及 continue 的区别?

return 是结束它所在的方法

break 是跳出循环（for,while,switch）

continue 是中止本次循环继续下次循环(for，while,do_while)

●Java 语言基础(方法)

A:为什么要有方法

提高代码的复用性

B:什么是方法

完成特定功能的代码块。

C:方法的格式

```
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...){
    方法体语句;
    return 返回值;
}
```

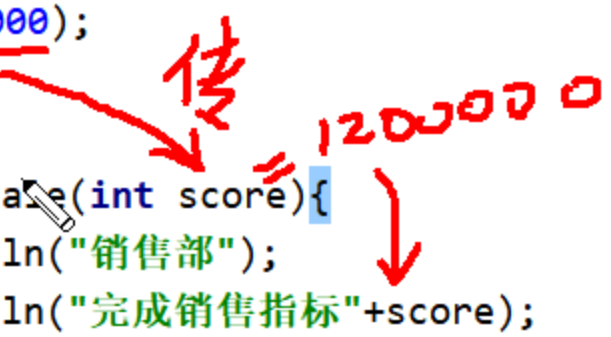
```java
public class Demo4 {
    public static void main(String[] args) {
        //调用方法
       add(50,60);
    }
//    修饰符      返回值类型   参数
    public static void add(int a,int b){
        //方法体
       int result = a+b;
        System.out.println(result);
    }
}
```

```java
public  class Company {  //假如类就是公司

    public static void main(String[] args) {   //总经办，核心
       //安排各部门工作，(调用方法)
        sale();
        finace();
        development();
    }
    //销售部
    public static void sale(){  //方法中的参数可以有，也可以没有
        System.out.println("销售部");
    }
    //财务部
    public static void finace(){
        System.out.println("财务部");
    }
    //开发部
    public static void development(){
        System.out.println("开发部");
    }
}
```

```java
public  class Company {  //加入类就是公司

    public static void main(String[] args) {   //总经办，核心
        //安排销售部工作，(调用方法)
        sale( score: 1200000);
    }
    //销售部
    public static void sale(int score){
        System.out.println("销售部");
        System.out.println("完成销售指标"+score);
    }
    //财务部
    public static void finace(){
        System.out.println("财务部");
    }
    //开发部
    public static void development(){
        System.out.println("开发部");
    }
}
```




```java
public  class Company {  //假如类就是公司

   public static void main(String[] args) {   //总经办，核心
      //安排销售部工作，(调用方法)
        saleDepatment(1200000);//1200000 是实际参数
   }
    //销售部
   public static void saleDepatment(int score){  //方法上的这个参数叫做形式参数（摆设），如果方法名只有一个参数全部小写，如果有多个那么从第二个单词开始首字母大写
       System.out.println("销售部");
        System.out.println("完成销售指标"+score);
    }
    //财务部
   public static void finace(){
        System.out.println("财务部");
    }
    //开发部
   public static void development(){
        System.out.println("开发部");
    }
```

```
}
```

D:方法的格式说明

修饰符：目前就用 public static。后面我们再详细的讲解其他的修饰符。

返回值类型：就是功能结果的数据类型。

方法名：符合命名规则即可。方便我们的调用。

参数：

实际参数：就是实际参与运算的。

形式参数：就是方法定义上的，用于接收实际参数的。

- 参数类型：就是参数的数据类型
- 参数名：就是变量名
- 方法体语句：就是完成功能的代码。
- return：结束方法的。
- 返回值：就是功能的结果，由 return 带给调用者。

●方法之求和案例及其调用

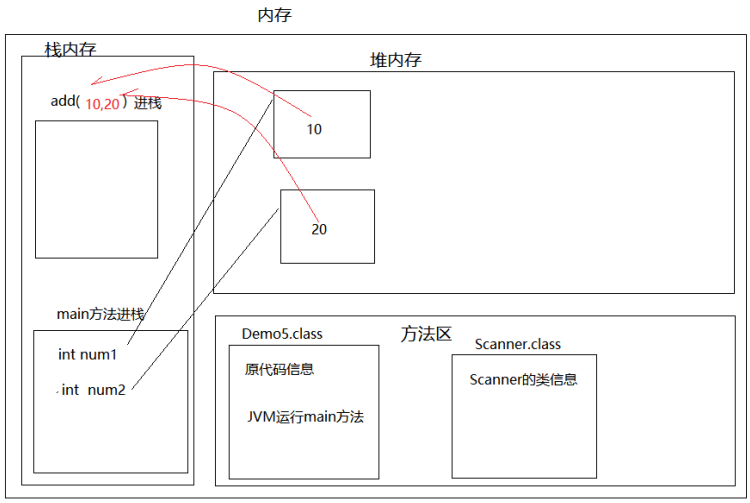
- A:如何写一个方法
  - 1,明确返回值类型
  - 2,明确参数列表
- B:案例演示
  - 需求：求两个数据之和的案例
- C:方法调用图解

```java
import java.util.Scanner;

public class Demo5 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("输入第一个数：");
        int num1 = sc.nextInt();
        System.out.println("输入第二个数：");
        int num2 = sc.nextInt();
        add(num1,num2);
    }
    public static void add(int x,int y){
        int result = x+y;
        System.out.println("两个数的和是："+result);
    }
}
```

```
import java.util.Scanner;

public class Demo5 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("输入第一个数: ");
        int num1 = sc.nextInt();
        System.out.println("输入第二个数: ");
        int num2 = sc.nextInt();
        add(num1,num2);
    }
    public static void add(int x,int y){
        int result = x+y;
        System.out.println("两个数的和是: "+result);
    }
}
```



●方法的注意事项

- A:方法调用(有具体返回值)
  - a:单独调用,一般来说没有意义，所以不推荐。
  - b:输出调用,但是不够好。因为我们可能需要针对结果进行进一步的操作。
  - c:赋值调用,推荐方案。
- B:案例演示
  - a:方法不调用不执行
  - b:方法与方法是平级关系，不能嵌套定义
  - c:方法定义的时候参数之间用逗号隔开
  - d:方法调用的时候不用再传递数据类型
  - e:如果方法有明确的返回值，一定要有 return 带回一个值

●方法的练习

案例演示 1

需求：键盘录入两个数据，返回两个数中的较大值

案例演示 2

需求：键盘录入两个数据，比较两个数是否相等

```
import java.util.Scanner;

public class Demo5 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("输入第一个数：");
        int num1 = sc.nextInt();
        System.out.println("输入第二个数：");
        int num2 = sc.nextInt();

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

        biJiao(num1,num2);
    }
```

```
    //比较两个数是否相等
    public static void biJiao(int a,int b){
        String result = a == b?"相等":"不相等";
        System.out.println(result);
    }
}
```

课堂练习：

1. 写一个方法完成三个数的加法运算

```
import java.util.Scanner;

public class Demo5 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("输入第一个数：");
        int num1 = sc.nextInt();
        System.out.println("输入第二个数：");
        int num2 = sc.nextInt();
        System.out.println("输入第三个数：");
        int num3 = sc.nextInt();
        int sum = add(num1,num2,num3); //声明一个变量 sum 类接收 add 方法返回的值
       System.out.println("sum="+sum);
        System.out.println(add(num1,num2,num3));//第二中输出方式
    }
    //求三个数的加法
    public static int add(int a,int b,int c){   //int 表示 add 方法运算完之后会给我们返回一个 int 类型的值
       int result = a+b+c;
        return result;  //return 表示把结果 result 返回
    }
}
```

2. 写一个方法完成两个数和的平均值计算

```
import java.util.Scanner;

public class Demo5 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("输入第一个数：");
        int num1 = sc.nextInt();
        System.out.println("输入第二个数：");
        int num2 = sc.nextInt();
        int result =avg(num1,num2);
        System.out.println("平均值是："+result);
    }
```

```
    //求两个数的和的平均值
   public static int avg(int a,int b){  //如果方法有申明了返回值的类型，那么这个方法的方法体内的最后一行一定要有 return 一个值
       int avgResult = (a+b)/2;
        return avgResult;
    }


}
```

总结：
方法中如果返回值是 void，不用 return
如果方法中申明了返回值类型，一定要有 return

●方法之输出星形及其调用

A:案例演示

需求：根据键盘录入的行数和列数，在控制台输出星形

例如：

```
******
******
******
```

```
import java.util.Scanner;

public class Demo9 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入行数：");
        int row = sc.nextInt();
        System.out.println("请输入列数：");
        int col = sc.nextInt();
        print(row,col);
    }
    //输出图案
   public static void print(int row,int col){
        for(int k=1;k<=row;k++) {  //输出多少行
          for (int i = 1; i <= col; i++) {   //输出一行 col 列
              System.out.print("*");
           }
           System.out.println();
        }
    }

}
```

B:方法调用：(无返回值,void)

单独调用
什么都不返回
打印输出(错误)
用来赋值(错误)

●方法重载概述和基本使用

A:方法重载概述
求和案例
2 个整数
3 个整数
4 个整数

```
public class Demo3 {
    public static void main(String[] args) {
        int result = add(2,9);
        System.out.println(result);
        int result2 = add(4,6,8);
        int reuslt3 = add(3,4,5,6);
        double result4 = add(3.4,10);//jvm 会根据传入的实际参数的类型取调用对应的重载方法
       System.out.println("result4="+result4);
    }
        /*
        求和
        2 个整数
        3 个整数
        4 个整数
         */
        //以下方法名同名，形式参数的个数或参数的类型不一样都构成重载,
        public static int add(int a,int b){
            return a+b;
        }
        //下面的方法与上面的add 方法不构成重载，重载和返回值类型没有关系
//        public static double add(int a,int b){
//            return a+b;
//        }
        public static int add(int a,int b,int c){
            return a+b+c;
        }

        public static int add(int a,int b,int c,int d){
            return a+b+c+d;
        }

//        public static int add(int b,int a){//同一个类中方法名相同，参数类型和个数也相同不构成重载，会报错
```

```
//        }
        public static double add(double a,int b){
            System.out.println("被调用了");
            return a+b;
        }
        public static double add(int b,double c){  //这个上上面的add从语法上构成重载，但是开发一般不会这么用
            return b+c;
        }
}
```

B:方法重载：

在同一个类中，方法名相同，参数列表不同。与返回值类型无关。

参数列表不同：

A:参数个数不同
B:参数类型不同
C:参数的顺序不同(算重载,但是在开发中不用)

●方法重载练习比较数据是否相等

A:案例演示
需求：比较两个数据是否相等。
参数类型分别为两个 int 类型，两个 double 类型，并在 main 方法中进行测试

```
public class Demo4 {
    public static void main(String[] args) {
        boolean flag1 = check(4,6);
        boolean flag2 = check(4.5,4.5);
        System.out.println(flag1);
        System.out.println(flag2);
    }
        //比较两个数据是否相等
    public static boolean check(double a,double b){
        boolean flag = a == b? true : false;
        return flag;
    }
    public static boolean check(int a,int b){
        if(a ==b){
            return true;
        }else{
            return false;
        }
    }
}
```

●方法之间的相互调用

设计一个注册公司的类，里面包含：核名、工商登记、银行开户、查看公司的信息，在查看公司信息中可以看到所有的公司信息（名称，工商登记的信息：纳税号），在工商登记中能看到公司的名称

```
public class Company {
    public static void main(String[] args) {
        showMsg();
    }
        /*
         设计一个注册公司的类，里面包含：核名、工商登记、银行开户、查看公司的信息，
       在查看公司信息中可以看到所有的公司信息（名称，工商登记的信息：纳税号），在工商登记中能看到公司的名称
         */
        //核名
   public static String checkName(String name){
        return name;
    }
    //工商登记
   public static String register(String no){
        return no;
    }
    //开户
   public static String account(){
        return "2341231234324235";
    }
    //查看公司的所有信息
   public static void showMsg(){   //类中除了main 方法，其他方法都是平级的，可以相互之间调用
      String name = checkName("钞票印刷公司");
        String no  =register("Nu89798e4r322w");
        String accountNo = account();
        System.out.println(name);
        System.out.println(no);
        System.out.println(accountNo);

    }
}
```

课堂练习：

1. 写一个方法计算一个数的立方
2. 写一个方法计算一个圆柱的体积
3. 写一个方法计算工作岗位的年薪

```
public class Demo5 {
    public static void main(String[] args) {
        liFang(10,60.5,70);
        areaValue(40,30);
        sal(12000);
    }
```

```
    //计算正方体体积
    public static void liFang(double w,double L,double h){
        System.out.println("立方是："+w*L*h);
    }
    //圆柱的体积
    public static void areaValue(double r,double h){
        System.out.println("圆柱的体积："+3.14*r*r*h);
    }
    //年薪
    public static void sal(double yue){
        System.out.println("年薪"+yue*14);
    }
}
```

●递归（了解）-开发中用的不多，几乎不用

初学递归会难以理解它的程序设计思想。递归思想之所以困难，原因在于它非常像是循环推理

案列（自己调用自己）

使用场景：当不知道要计算多少次的时候，使用递归

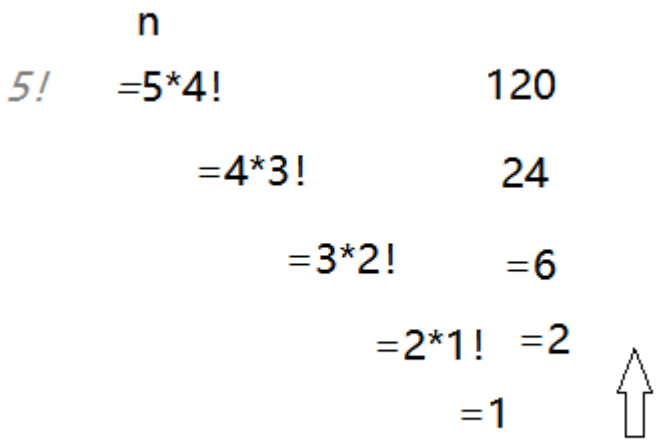
例如：

1. 用在搜索文件夹中的文件
2. 搜索引擎抓取网络上的数据

A:案例演示

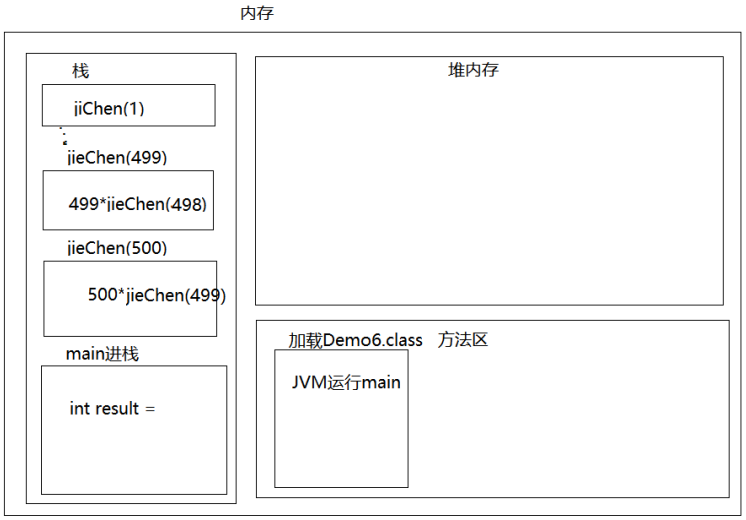
计算 5!

```
public class Demo6 {
    public static void main(String[] args) {
       long result = jieChen(500);
        System.out.println(result);
    }
    /*
    计算  5!  阶乘
       5！ = 5*4*3*2*1
       10！ = 10*9*8*7*6*5*4*3*2*1
     */
    public static long jieChen(int n){
        if(n >=1){
            if(n==1){
            return 1;
            }else{
                return n*jieChen(n-1);
            }
        }else{
            return -1; //表示输入有问题
      }
    }
}
```




```java
public class Demo6 {
    public static void main(String[] args) {
       long result = jieChen(500);
        System.out.println(result);
    }
    /*
    计算 5! 阶乘
       5! = 5*4*3*2*1
       10! = 10*9*8*7*6*5*4*3*2*1
     */
    public static long jieChen(int n){
        if(n >=1){
            if(n==1){
            return 1;
            }else{
                return n*jieChen(n-1);
            }
        }else{
            return -1; //表示输入有问题
    }
    }
}
```

递归结构（自己调用自己）



从这个内存图可以看出，直到jieChen(1)出现在内存中，才有结果往回计算，内存中要存储500个jieChen(int n)方法，会造成内存的崩溃 所以开发中谨慎使用递归

注意事项：

递归是很耗资源的，除非是没有替代的方法，用的时候慎用。

long d1=System.currentTimeMilis（）；返回当前的时刻毫秒数

递归占内存因为它打开的方法多

递归函数运行一般比循环函数慢，有时候甚至是一倍的差距

```java
public class Demo6 {
    public static void main(String[] args) {
        long time1 = System.currentTimeMillis();//获取当前时间的毫秒值，从1970年1月1日 0时0分0秒开始
       System.out.println(time1);
       long result = jieChen(23);
        System.out.println(result);
        long time2 = System.currentTimeMillis();
        System.out.println(time2);
        System.out.println("消耗的时间："+(time2 - time1)+"毫秒");
    }
    /*
```

```
    计算  5!  阶乘
       5! = 5*4*3*2*1
       10! = 10*9*8*7*6*5*4*3*2*1
     */
    public static long jieChen(int n){
        if(n >=1){
            if(n==1){
            return 1;
            }else{
                return n*jieChen(n-1);
            }
        }else{
            return -1; //表示输入有问题
       }
    }
}
```

B:案例演示

递归实现：

一列数的规则如下: 1、1、2、3、5、8、13、21、34 ，求第 30 位数是多少？

```
public class Demo7 {
    public static void main(String[] args) {
        int result = getNum(30);
        System.out.println(result);
    }
        /*
        1、1、2、3、5、8、13、21、34 ，求第30 位数是多少?
          从第三位开始每一位数是前面两位的和
         */
    public static int getNum(int n){
        if(n==1 || n==2){
            return 1;
        }else{
            return getNum(n-2)+getNum(n-1);
        }
    }
}
```

假设有一张足够大的纸，厚度为 1 毫米，珠穆朗玛峰高度为 8848 米（8848000 毫米），

求纸张至少折叠几次可以超过珠穆朗玛峰的高度

```
public class Demo8 {
    public static void main(String[] args) {
//        System.out.println(getNum(5));
        int num=0;
        int n =0;//获取第 n 位置上的数
       while(true){
```

```
            if(n <=8848000){
                n=getNum(++num);
                System.out.println("折叠第"+(num-1)+"次,达到"+n+"毫米");
            }
        }

    }
    //假设有一张足够大的纸，厚度为1 毫米，珠穆朗玛峰高度为8848 米（8848000 毫米），
    /*
      求纸张至少折叠几次可以超过珠穆朗玛峰的高度
        1,2,4,8,16.....8848000
        从第二位数开始，是前面数乘以2
        2*getNum(n-1) //n 代表第n 位上的数
      */
    //求得第n 位值上的数
    public static int getNum(int n){
        if(n==1){
            return 1;
        }else{
            return 2*getNum(n-1);
        }
    }

}
```