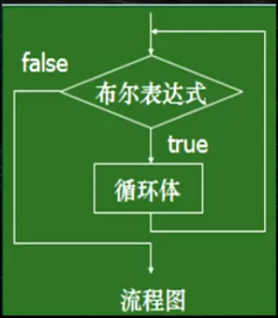
**循环**

* **While循环**

在循环刚开始时，会计算一次“布尔表达式”的值，若条件为真，执行循环体。而对于后来每一次额外的循环，都会在开始前重新计算一次。

语句中应有使循环趋向于结束的语句，否则会出现无限循环—死循环

****

循环结构由四个结构组成：

* 初始化：变量初始化
* 条件判断：必须要求返回true或false的值
* 循环体：具体的要执行的逻辑代码
* 迭代变量：促使此循环结束（如果出现死循环直接将JVM窗口关掉即可）

public class While{

public static void main(String [] args){

int i = 1;

while(i<=100){

System.out.println("第"+i+"遍输出");

i++;

}

}

}

以上循环会显示第1遍输出，第2遍输出 …… 第100遍输出

**课堂练习：**

* 求100以内的偶数和

int i = 1;

int sum = 0;

while(i<=100){

if (i%2 == 0){

sum += i;

}

i++;

}

System.out.println(sum);

* 购物清单



要求：

-循环输入商品编号和购买数量

-当输入n时结账

-结账时计算应付金额并找零

自己写的↓：

import java.util.Scanner;

public class While{

public static void main(String [] args){

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"+"\n"+

"请选择购买的商品编号："+"\n"+"1.T恤"+"\t"+"2.网球鞋"+"\t"+"3.网球拍"+

"\n"+"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"+"\r\n");

Scanner input = new Scanner(System.in);

String result = "y";

double totalPrice = 0;

while(result.equals("y")){

System.out.print("\n");

System.out.print("请输入商品编号：");

int identifier = Integer.valueOf(input.nextLine());

/\*int identifier = input.nextInt(); ←不可行

上面这个操作比如先输入1 然后有个回车（\n）的动作

nextInt后面不能和nextLine连用，因为nextInt只能读取数值，

不能读取空格/tab/和\n，，因此这一步就会剩下一个\n

\*/

System.out.print("请输入购买数量：");

int number = Integer.valueOf(input.nextLine());

/\*int number = input.nextInt();

上一个剩的\n ，加上这次输入的2和 \n ，又剩了\n

\*/

double priceT = 245.0;

double priceShoe = 570.0;

double priceRacket = 320.0;

if (identifier == 1){

System.out.print("T恤￥"+priceT);

System.out.print("\t数量 "+number);

double totalT = number\*priceT;

System.out.println("\t合计"+totalT);

totalPrice += totalT;

}else if(identifier == 2){

System.out.print("网球鞋￥"+priceShoe);

System.out.print("\t数量 "+number);

double totalShoe = number\*priceShoe;

System.out.println("\t合计 "+totalShoe);

totalPrice += totalShoe;

}else if(identifier == 3){

System.out.print("网球拍￥"+priceRacket);

System.out.print("\t数量 "+number);

double totalRacket = number\*priceRacket;

System.out.println("\t合计 "+totalRacket);

totalPrice += totalRacket;

}

System.out.print("是否继续(y/n)");

result = input.nextLine();

/\*

上面实行完剩的\n，因为nextLine读取\n并且以\n为结束符,

因此读了\n以后就退出了

\*/

}

System.out.print("\n");

System.out.println("折扣：0.8");

System.out.println("应付金额："+totalPrice);

System.out.print("实付金额：");

double pay = Double.valueOf(input.nextLine());

while (pay < totalPrice){

System.out.print("您输入的金额小于应付金额，请重新输入：");

pay = Double.valueOf(input.nextLine());

}

double change = pay - totalPrice;

System.out.print("找钱："+change);

}

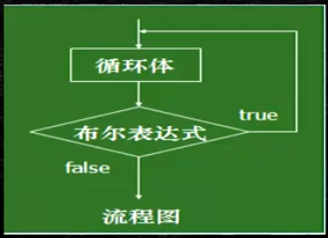
}

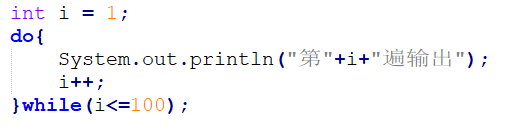
* **do -while循环**

do-while：先执行，后判断

while：先判断，后执行

dowhile总是保证循环体至少被执行一次，这是dowhile和while的主要差别



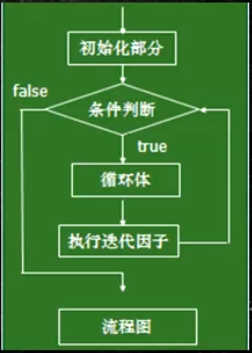


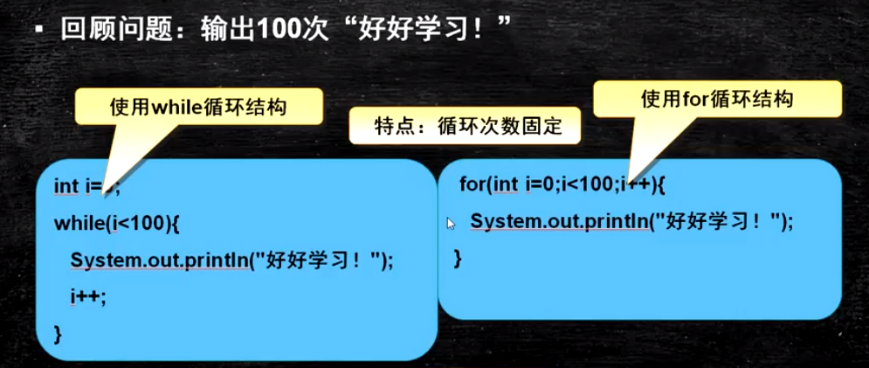
输出的是从1到100遍输出。在i=99的时候i++为100，能进入到循环里面；在i=100的时候i++为101，进不去循环里面，因此只能打印到100

* **for循环（使用最多）**

支持迭代的一种通用结构，是最有效，最灵活的循环结构

for循环和while循环实现的功能是一样的，但是for循环更简洁





**注意●：**

* -for循环在执行条件测试后，先执行程序部分，再执行步进

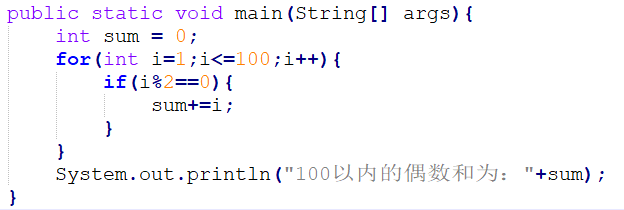
语法规则：

for（**1**初始表达式；**2**布尔表达式；**4**步进器）{

**3**循环体；

}

* -在for语句的初始化部分声明的变量，其作用域为整个for循环体



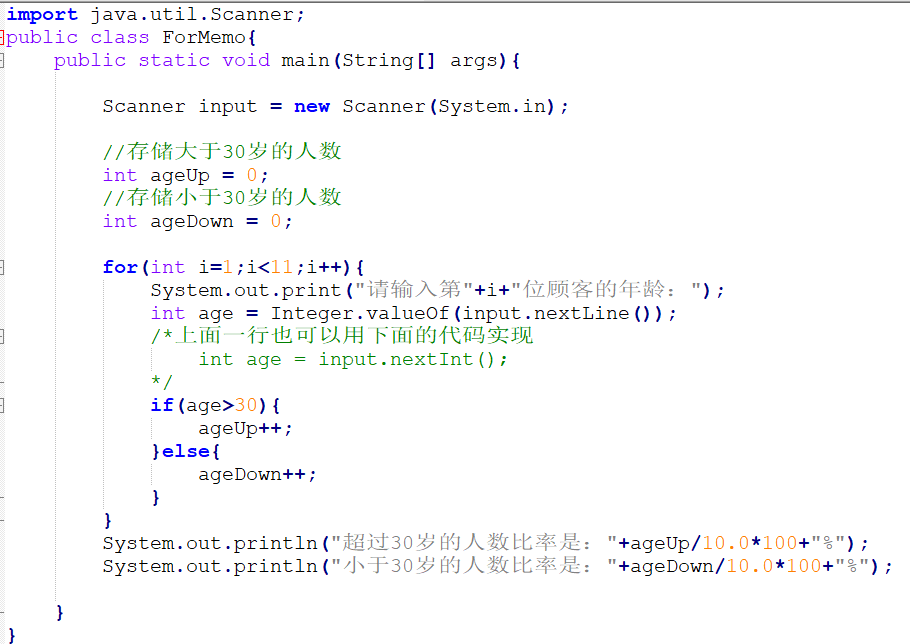
在这里假如我又定义了一个其他的for循环，for后面括号里面的变量也定义成i也OK，因为这个i的作用域只在这个循环里面，不涉及外面。而while的i的定义作用域是从定义变量开始到此方法结束，因此定义两个i及以上的时候会出问题

* “初始化”和“循环条件表达式”部分可以使用逗号来执行多个操作
* 如果三个部分都为空语句（分号不能省），相当于一个无限循环

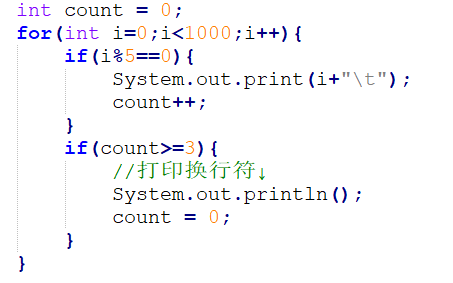
**课堂练习：**

* 计算顾客年龄比例





* 输出1-1000之间能被5整除的数，且每行输出3个



* 使用循环分别实现将10进制整数变成二进制数

逻辑：

比如将25变成二进制数

25/2 = 12 余1

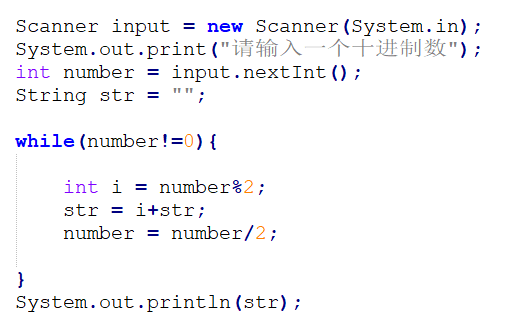
12/2 = 6 余0

6/2 = 3 余0

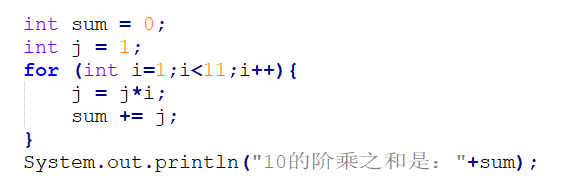
3/2 = 1 余1

1/2 = 0 余1

然后把余数反着写即为二进制数 11001 = 24+23+20=25



* 求1！+2！+ …… +10！



**跳转语句break和continue**

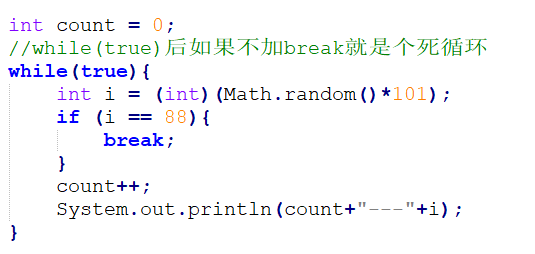
* 在任何循环语句的主体部分，均可用**break**控制循环的流程。break用于强行退出循环，不执行循环中剩余的语句（break语句还可用于多分支语句switch中）

break跳出循环，当包含多层循环的时候只能跳出内层循环，无法跳出外层循环

* **continue**语句用在循环语句体中，用于终止本次循环过程，即跳过循环体中尚未执行的语句，接着进行下一次是否执行循环的判定

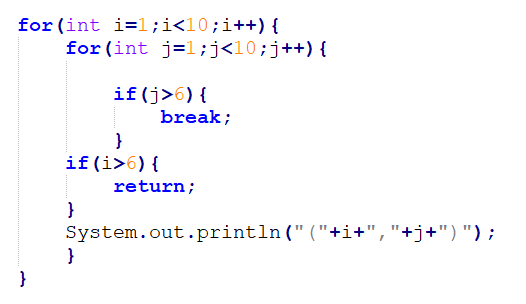
**课堂练习：**

* 生成0-100随机数，直到生成88为止，停止循环！（将第几次以及输出的数打印出来）



这时打印出来的内容里肯定没有i=88的，因为i=88时程序就跳出了

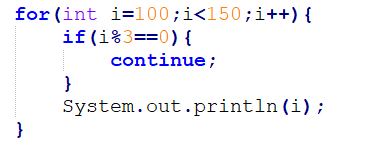
* 请打印输出(1,1)(1,2)(1,3)…直到输出(6,6)停止（前提是i和j的取值范围都是10）



**↑这种不可以**，因为break只能跳出内部循环（本次循环），它还是会继续外部循环

这时如果将break换成return就可以得到想要的结果了

* 把100~150之间不能被3整除的数输出



★对比break和continue★

-使用场合：

break可用于switch结构和循环结构中；continue只能用于循环结构中

-作用：

break语句**终止**某个循环，程序跳转到循环块外的下一条语句；continue**跳出**本次循环，

进入下一次循环

**跳转语句return**

* -return语句从当前方法退出，返回到调用该方法的语句处，并从该语句的下条语句处继续执行程序

-返回语句的两种格式

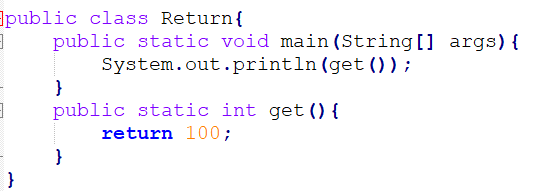
1.return expression

· 返回一个值给调用该方法的语句

· 返回值的数据类型必须和方法声明中返回值类型一致或时精度低于声明的数据类型

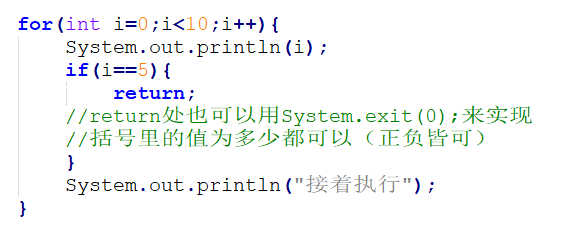
2.return

· 当方法声明中void声明返回类型为空时，应使用这种返回类型，它不返回任何值

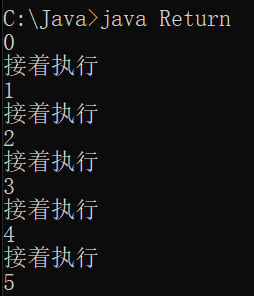


执行结果为 100

* 终止当前程序



执行结果为：



**死循环**

**while循环：**

while（）里的表达式为true，并且没有break终止就是死循环

**for循环：**

for(;;){

}

括号里必须写两个分号，不然编译不了

for(int i=1;;){

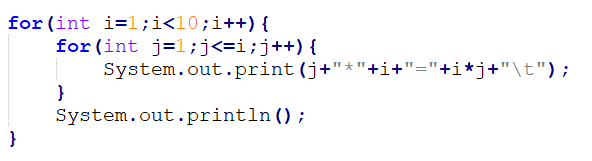
}

for(int i=1;;i++){

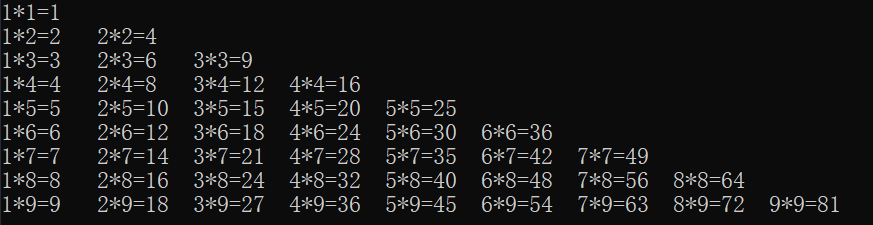
}

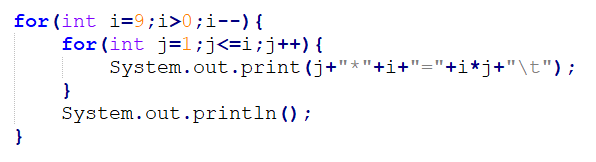
**多重循环**

* **输出九九乘法表**

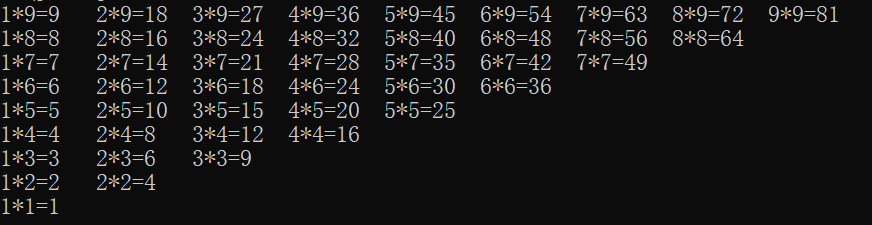


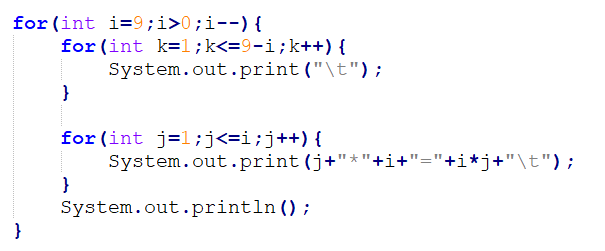
输出结果：



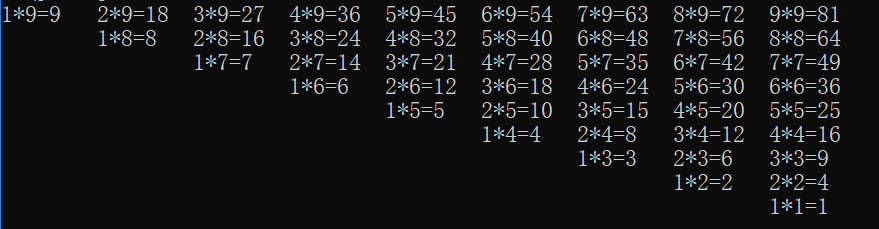


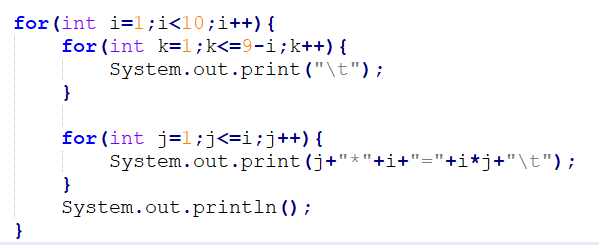
输出结果：

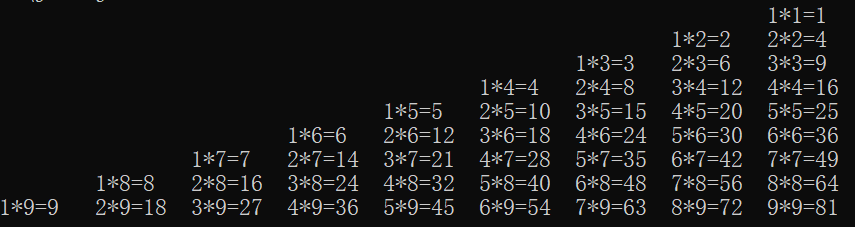




输出结果：

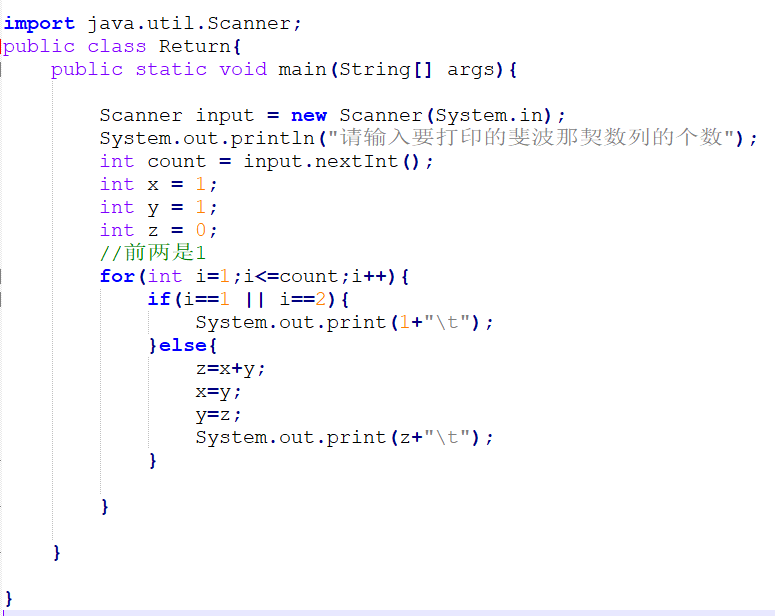




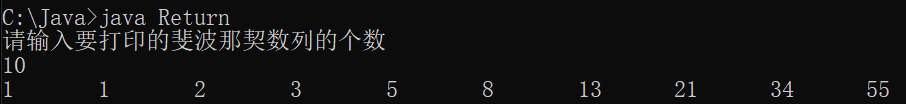
输出结果：

* **斐波那契数列**

1 1 2 3 5 8 13 21 34 55 ……

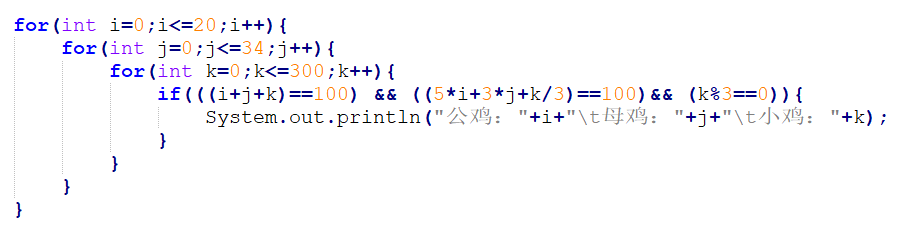


输出效果：



* **百钱买百鸡**

现在手头有100块钱，公鸡5元/只，母鸡3元/只，小鸡3只1元钱。要买100只鸡，里面分别有几只公鸡，几只母鸡，几只小鸡



执行结果：

