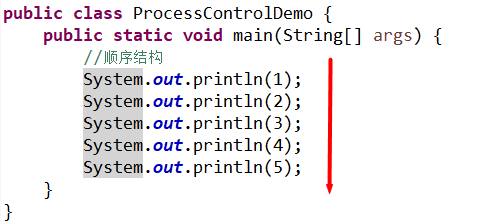
# 流程控制

## 概念

* 什么是流程控制

使用不同的关键字/语法来控制代码的执行流程/顺序

* 分类
  + 顺序结构---我们之前写的都是这种结构

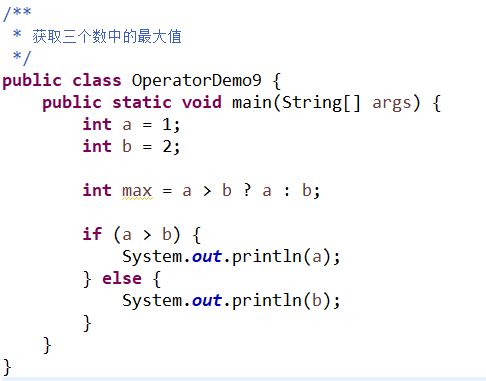


**代码从上向下依次执行**

* + 分支结构

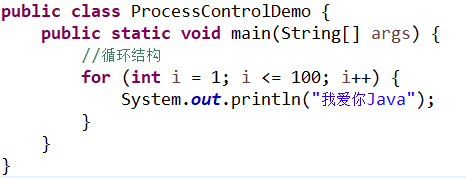
到达某个状态时，可能会有多种情况，每种情况都需要提前准备好，所以多条路的路口就是一个分支。代码写得多，程序运行的少。

之前例子中的三元运算符可以改写成选择结构



* + 循环结构

某段代码相同或者相似，使用相同的逻辑。使用一定的格式，来完成相同代码的**反复执行**



代码写的少，程序执行的多

## 分支结构

### if语句第一种格式： if

* 格式

if(关系表达式)｛

语句体;

｝

* 执行流程

首先判断关系表达式看其结果是true还是false

如果是true就执行语句体

如果是false就不执行语句体

* 示例代码

|  |
| --- |
| 例如：小毛参加考试，如果小毛考试成绩及格(>60分)，则输出小毛考试成绩及格。 |

* 练习

找出 int x = 12；

int y = 20；

int z = 8；

三个数中最大的数？

**假设修正法**

|  |
| --- |
|  |

### if语句第二种格式： if...else

* 格式

if(关系表达式) {

语句体1;

}else {

语句体2;

}

* 执行流程

首先判断关系表达式看其结果是true还是false

如果是true就执行语句体1

如果是false就执行语句体2

**if代码块和else代码块只会执行其中一个**

* 示例代码

|  |
| --- |
| 小邓参加考试，如果成绩>=60分，输出“成绩及格”。如果成绩<60分，输出“成绩不及格”。 |

### if语句第三种格式： if...else if ...else

* 格式

if (判断条件1) {

执行语句1;

} else if (判断条件2) {

执行语句2;

}

...

}else if (判断条件n) {

执行语句n;

} else {

执行语句n+1;

}

* 执行流程

首先判断关系表达式1看其结果是true还是false

如果是true就执行语句体1

如果是false就继续判断关系表达式2看其结果是true还是false

如果是true就执行语句体2

如果是false就继续判断关系表达式…看其结果是true还是false

…

如果没有任何关系表达式为true，就执行语句体n+1。

* 示例代码

|  |
| --- |
| 需求：根据键盘输入的数值不同,打印对应的星期数。  if 只能进行一层判断,if else 只能进行两层判断,那么需要多层判断时呢?星期可是有7个数的。如何设计代码？  **int** x = 2;  **if** (x == 1) {  System.*out*.println("星期一");  }  **if** (x == 2) {  System.*out*.println("星期二");  }  **if** (x == 3) {  System.*out*.println("星期三");  }  … |

缺点是什么？

|  |
| --- |
| **int** x = 4;  **if** (x == 1) {  System.*out*.println("星期一");  } **else** **if** (x == 2) {  System.*out*.println("星期二");  } **else** **if** (x == 3) {  System.*out*.println("星期三");  } **else** **if** (x == 4) {  System.*out*.println("星期四");  } **else** **if** (x == 5) {  System.*out*.println("星期五");  } **else** **if** (x == 6) {  System.*out*.println("星期六");  } **else** **if** (x == 7) {  System.*out*.println("星期日");  } **else** {  System.*out*.println("请输入数字1-7");  } |

### 练习

* 根据用户输入的不同成绩(0~100),给出相应的提示:优秀,良好,及格,不及格...



### switch

* 格式

switch (表达式){

case 目标值1:

执行语句1

break;

case 目标值2:

执行语句2

break;

．．．．．．

case 目标值n:

执行语句n

break;

default:

执行语句n+1

break;

}

* 说明

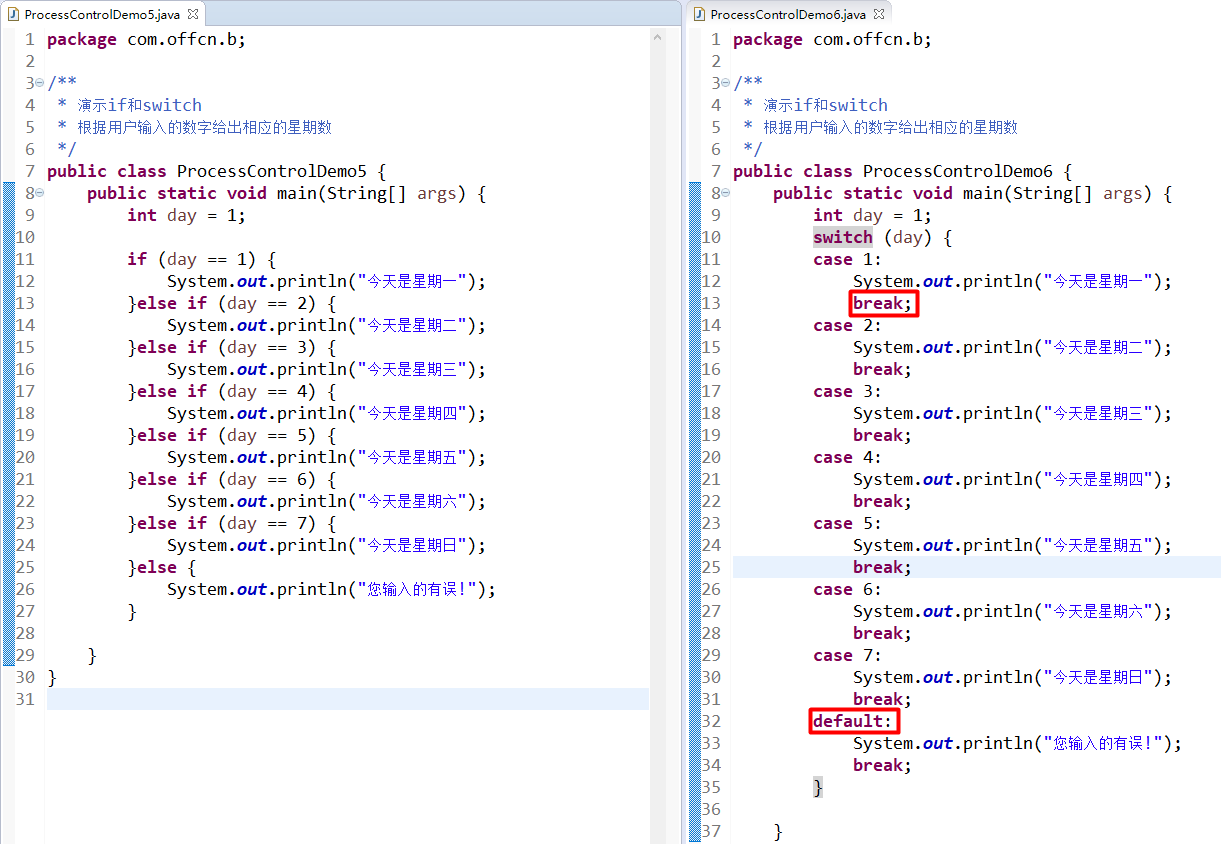
在上面的格式中，switch语句将表达式的值与每个case中的目标值进行匹配，如果找到了匹配的值，会执行对应case后的语句，如果没找到任何匹配的值，就会执行default后的语句。switch语句中的break关键字将在后面的做具体介绍，此处，我们只需要知道break的作用是跳出switch语句即可。

* 注意

switch语句中的表达式只能是byte、short、char、int类型的值，如果传入其它类型的值，程序会报错。

但是在JDK5.0中引入的新特性enum枚举也可以作为switch语句表达式的值，在JDK7.0中也引入了新特性，switch语句可以接收一个String类型的值。

* 示例代码



* 示例代码2

|  |
| --- |
| char类型 在switch中的应用  **int** x = 1, y = 2;  **char** ch = '\*';  **switch** (ch) {  **case** '+':  System.*out*.println("x+y=" + (x + y));  **break**;  **case** '-':  System.*out*.println("x-y="+(x-y));  **break**;  **case** '\*':  System.*out*.println("x\*y="+(x\*y));  **break**;  **case** '/':  System.*out*.println("x/y="+(x/y));  **break**;  **default**:  System.*out*.println("不靠谱");  } |

* case穿透（下坠）

在使用switch语句的过程中，如果多个case条件后面的执行语句是一样的，则该执行语句只需书写一次即可，这是一种简写的方式。

例如，要判断一周中的某一天是否为工作日，同样使用数字1~7来表示星期一到星期天，当输入的数字为1、2、3、4、5时就视为工作日，否则就视为休息日。接下来通过一个案例来实现上面描述的情况，如下所示

当变量week值为1、2、3、4、5中任意一个值时，处理方式相同，都会打印“今天是工作日”。同理，当变量week值为6、7中任意一个值时，打印“今天是休息日”。

|  |
| --- |
| **public** **class** SwitchDemo02 {  **public** **static** **void** main(String[] args) {  **int** week = 2;  **switch** (week) {  **case** 1:  **case** 2:  **case** 3:  **case** 4:  **case** 5:  // 当 week 满足值 1、2、3、4、5 中任意一个时，处理方式相同  System.*out*.println("今天是工作日");  **break**;  **case** 6:  **case** 7:  // 当 week 满足值 6、7 中任意一个时，处理方式相同  System.*out*.println("今天是休息日");  **break**;  }  }  } |

* 练习

键盘录入一个月份，输出这个月份的季节

345月是春季，678是夏季，9、10、11秋季，12、1、2冬季

其他数字，没有任何季节，录入错误

|  |
| --- |
| System.***out***.println("请输入月份：");  Scanner sc = **new** Scanner(System.***in***);  **int** month = sc.nextInt();  **switch**(month){  **case** 3:  **case** 4:  **case** 5:System.***out***.println("春季");**break**;  **case** 6:  **case** 7:  **case** 8:System.***out***.println("夏季");**break**;  **case** 9:  **case** 10:  **case** 11:System.***out***.println("秋季");**break**;  **case** 12:  **case** 1:  **case** 2:System.***out***.println("冬季");**break**;  **default**:System.***out***.println("月份输入错误");  } |

### if和switch的区别

1. 相同点：if语句和switch语句都可以表达分支结构
2. 不同点：

1、if语句格式相对灵活，更加强大，switch语句相对弱小。所有的switch语句都可以被if语句替代。

2、if语句更多的是做一些连续性的判断，而switch语句更多的是做一些离能1散性的匹配11

3、if语句每次都需要获取表达式的值，而switch只获取一次表达式的值，所以相对switch的效率高一些。

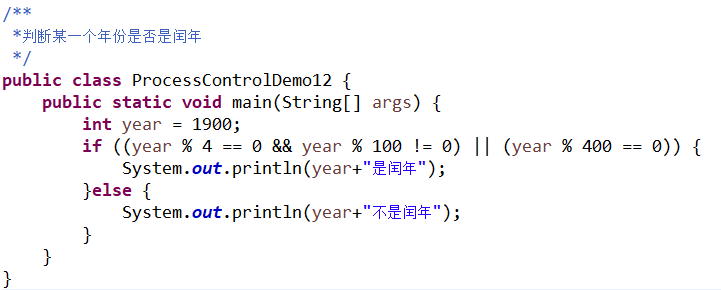
### if练习

**1.判断闰年**

* 需求分析



* 代码实现



**2.水仙花数**

说明:所谓的水仙花数是指一个三位数，其各个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

* 代码实现

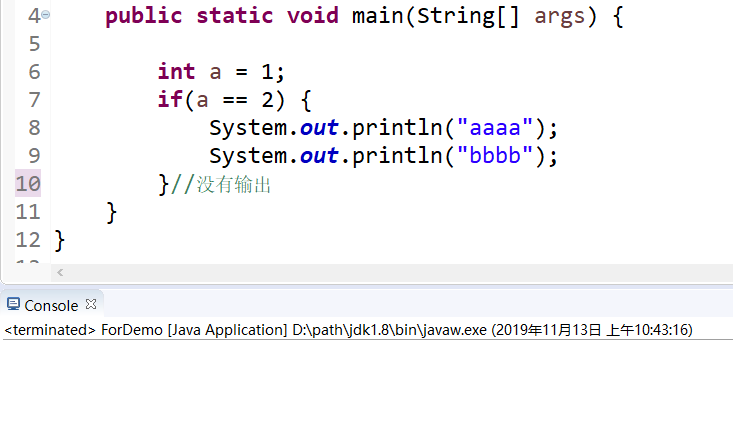
1. **判断三角形**

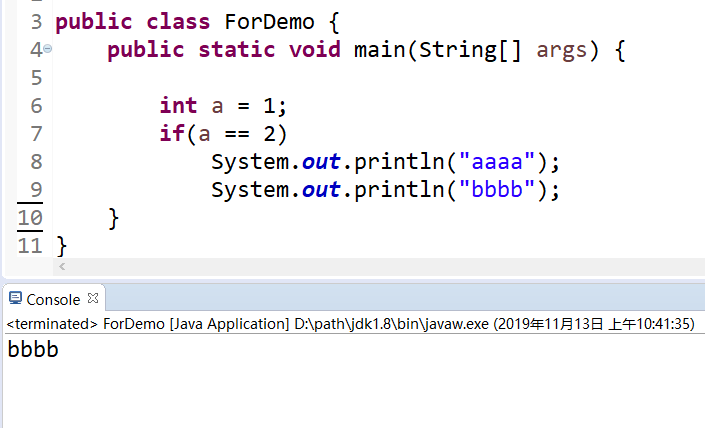
输入三位整数，判断以这三位整数为边长可否组成三角形。

### if注意

if的小括号后面可以没有大括号，但是有区别，例如：

1. 有大括号





**总结：**

**if后可以没有大括号，但是只可以限制离它最近的一条语句，所以为了靠谱尽量写上大括号。**

## 循环结构

### 问题引入

输出10000遍“好好学习，天天向上”该怎么做？

### 概述

循环语句可以在满足循环条件的情况下，**反复**执行某一段代码，这段被重复执行的代码被称为循环体语句，当反复执行这个循环体时，需要在一定的时候让循环判断条件值为false，从而结束循环，否则循环将一直执行下去，形成死循环。

Java中三种循环结构：

1 for循环

2 while循环

3 do while循环

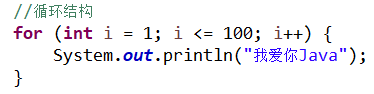
### for循环

* for循环语句格式：

for(初始化表达式①; 布尔表达式②; 步进表达式④){

循环体③

}



* 执行流程

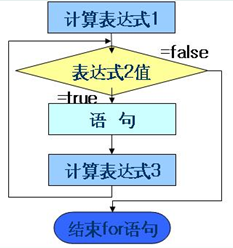
执行顺序：① ②③④>②③④>②③④…②不满足为止。

①负责完成循环变量初始化

②负责判断是否满足循环条件，不满足则跳出循环

③具体执行的语句

④循环后，循环条件所涉及变量的变化情况



* 示例代码

**案例1：使用for循环在控制台输出1到10**

|  |
| --- |
|  |

**案例2：在控制台输出10到1**

|  |
| --- |
| /\*  for(int i=1;i<=10;i++){  System.out.println(11-i);  }  \*/    **for**(**int** i=10;i>=1;i--){  System.***out***.println(i);  } |

**案例3：在控制台输出1,3,5,7,9**

|  |
| --- |
| /\*  for(int i=1;i<=10;i++){  if(i%2!=0){  System.out.println(i);  }  }\*/    **for**(**int** i=1;i<=10;i=i+2){  System.***out***.println(i);  } |

**案例4：求出1到100的和**

|  |
| --- |
|  |

**案例5: 键盘输入一个数n，求n的阶乘**

|  |
| --- |
| //5!= 5\*4\*3\*2\*1  System.***out***.println("请输出要求阶乘的正整数");    Scanner sc = **new** Scanner(System.***in***);    **int** n = sc.nextInt();    **if**(n<0){  System.***out***.println("请输入正确数字");  }**else** **if**(n==0){  System.***out***.println(n+"的阶乘是1");  }**else**{    **int** jiecheng = 1;    **for**(**int** i=1;i<=n;i++){    //jiecheng=jiecheng\*i;  jiecheng\*=i;  }    System.***out***.println(n+"的阶乘是"+jiecheng);    } |

**案例6：求出100----1000之间的水仙花数**

|  |
| --- |
| /\*  \* 求出所有水仙花数  \*  \* 思路：  \* 1 产生 100 到 999 之间的 所有整数  \*  \* 2 判断产生的数是否为水仙花数  \*  \* \*/    **public** **static** **void** main(String[] args) {    **for**(**int** i=100;i<1000;i++){    // 判断i是否为水仙花数  **int** ge = i%10;  **int** shi = i/10%10;  **int** bai = i/100;    **if**(i==ge\*ge\*ge+shi\*shi\*shi+bai\*bai\*bai){ // 满足条件，i就是水仙花数  System.***out***.println(i);  }      }  } |

**案例7：输出1到100之间能被3整除的数**

|  |
| --- |
| /\*  \* 输出1到100之间能被3整除的数  \* 思路：  \* 1 产生 1到100之间的整数  \* 2 判断这个数能否被3整除  \* \*/    **public** **static** **void** main(String[] args) {    **for**(**int** i=1;i<=100;i++){    **if**(i%3==0){  System.***out***.println(i);  }  }  } |

### 死循环

|  |
| --- |
| /\*  \* 什么是死循环  \*  \* 循环会永远执行下去，没有退出循环的条件  \*  \* \*/  **public** **static** **void** main(String[] args) {    **for**(;;){  System.***out***.println("111");  }    } |

### 跳出循环

* 概述

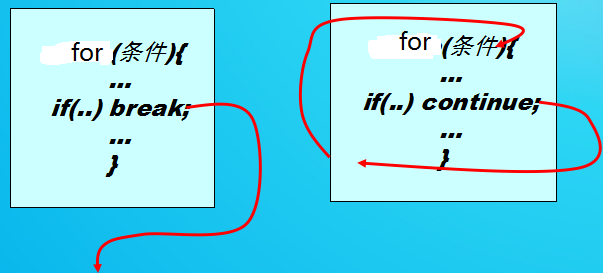
如果我们想让循环在某一步的时候结束或者跳过某些数据不要这些数据，该怎么办?

Java就提供了break，continue来实现控制语句的中断和跳转。

* break
  + break的使用场景：在选择结构switch语句中、在循环语句中
  + break的作用：跳出当前switch/循环语句
* continue
  + continue的使用场景：在循环语句中
  + continue的作用：结束本次循环,继续下一次循环
* 总结

break 结束整个循环

continue 结束本次循环,继续下一次循环



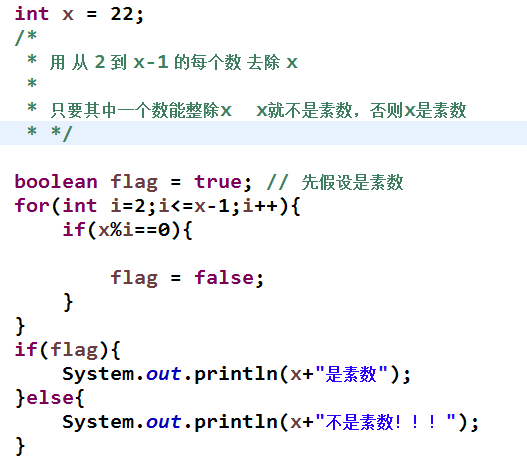
* 代码演示

|  |
| --- |
|  |

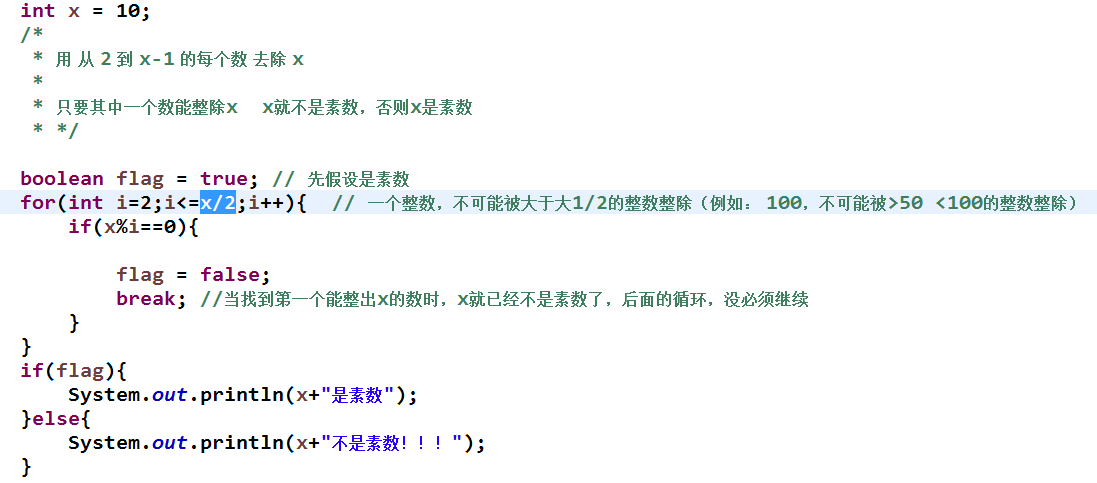
|  |
| --- |
|  |

## 案例

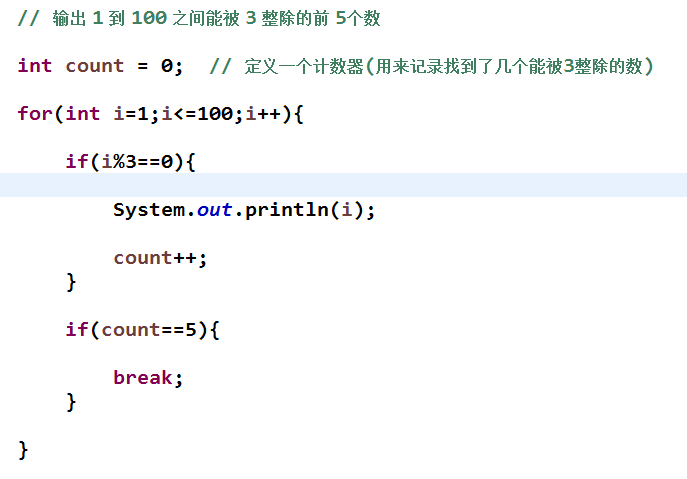
### 案例1：**判断一个数是不是素数（质数）**



**改进上面的代码**



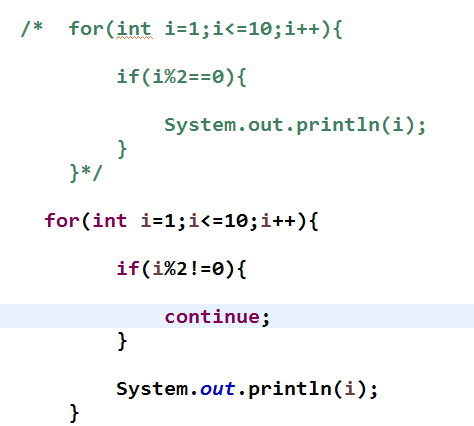
### 案例2：输出 1 到 100 之间能被 3 整除的前 5个数



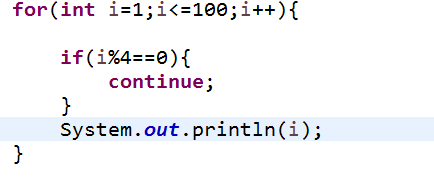
### 案例3：打印1到10个数，遇到4的倍数程序自动退出

|  |
| --- |
| **for**(**int** i=1;i<=10;i++){  **if**(i%4==0){  **break**;  }  System.***out***.println(i);  } |

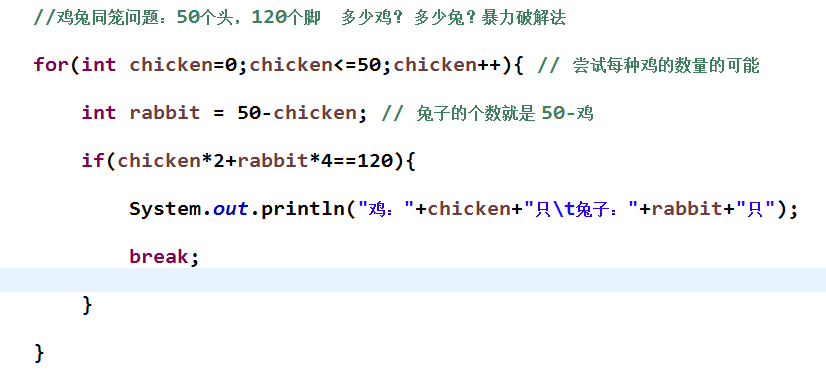
### 案例4：打印1到10中的偶数（使用continue）



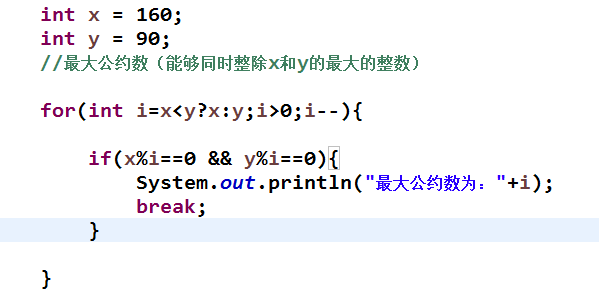
### 案例5：打印1到100中非4的倍数的数字（使用continue）



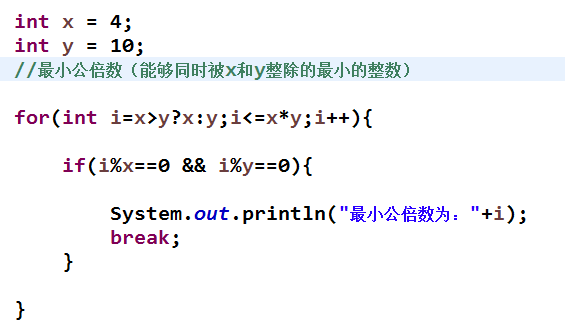
### **案例6：鸡兔同笼问题：50个头，120个脚 多少鸡？ 多少兔？暴力破解法**



### 案例7：最大公约数



### 案例8：最小公倍数



### 案例9：模拟登录

* 需求:

程序接收用户输入的用户名和密码并进行比较

如果用户名和密码为admin,123,就登录成功

否则登录失败,重新登录

并且只有三次机会,每次失败提示还有几次

* 代码实现

