用户交互Scanner

1、Scanner对象

之前我们学的基本语法中我们并没有实现程序和人的交互，但是Java给我们提供了这样一个工具类，我们可以获取用户的输入。java.util.Scanner 是 Java5 的新特征，我们可以通过 Scanner 类来获取用户的输入。

【都是固定格式，大家先不用理解代码的意思，先跟着学会操作，之后讲解面向对象时候就直接明白了这些代码的意思】

下面是创建 Scanner 对象的基本语法：

```
Scanner s = new Scanner(System.in);
```

接下来我们演示一个最简单的数据输入，并通过 Scanner 类的 next() 与 nextLine() 方法获取输入的字符串，在读取前我们一般需要 使用 hasNext() 与 hasNextLine() 判断是否还有输入的数据。

2、next & nextLine

我们使用next方式接收一下输入的数据！

```
public static void main(String[] args) {
    //创建一个扫描器对象，用于接收键盘数据
    Scanner scanner = new Scanner(System.in);

    //next方式接收字符串
    System.out.println("Next方式接收:");
    //判断用户还有没有输入字符
    if (scanner.hasNext()){
        String str = scanner.next();
        System.out.println("输入内容: "+str);
    }
    //凡是属于IO流的类如果不关闭会一直占用资源.要养成好习惯用完就关掉.就好像你接水完了要关水龙头一样.很多下载软件或者视频软件如果你不彻底关,都会自己上传下载从而占用资源,你就会觉得卡,这一个道理.
    scanner.close();
}
```

测试数据：Hello World！

结果：只输出了Hello。

接下来我们使用另一个方法来接收数据：nextLine()

狂神社群笔记资料，禁止外传，本人QQ：24736743

```java
public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    // 从键盘接收数据

    // nextLine方式接收字符串
    System.out.println("nextLine方式接收：");
    // 判断是否还有输入
    if (scan.hasNextLine()) {
        String str2 = scan.nextLine();
        System.out.println("输入内容：" + str2);
    }
    scan.close();
}
```

测试数据：Hello World!

结果：输出了Hello World!

两者区别：

next():

- 1、一定要读取到有效字符后才可以结束输入。
- 2、对输入有效字符之前遇到的空白，next() 方法会自动将其去掉。
- 3、只有输入有效字符后才将其后面输入的空白作为分隔符或者结束符。
- 4、next() 不能得到带有空格的字符串。

nextLine()：

- 1、以Enter为结束符,也就是说 nextLine()方法返回的是输入回车之前的所有字符。
- 2、可以获得空白。

3、其他方法

如果要输入 int 或 float 类型的数据，在 Scanner 类中也有支持，但是在输入之前最好先使用 hasNextXxx() 方法进行验证，再使用 nextXxx() 来读取：

【演示：IDEA中查看源码中的所有方法，并写出案例】

```java
public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    // 从键盘接收数据
    int i = 0;
    float f = 0.0f;
    System.out.print("输入整数：");
    if (scan.hasNextInt()) {
        // 判断输入的是否是整数
        i = scan.nextInt();
        // 接收整数
        System.out.println("整数数据：" + i);
    } else {
        // 输入错误的信息
        System.out.println("输入的不是整数！");
    }
    System.out.print("输入小数：");
    if (scan.hasNextFloat()) {
        // 判断输入的是否是小数
        f = scan.nextFloat();
        // 接收小数
```


狂神社群笔记资料，禁止外传，本人QQ：24736743

```
        System.out.println("小数数据: " + f);
    } else {
        // 输入错误的信息
        System.out.println("输入的不是小数! ");
    }
    scan.close();
}
```

以下实例我们可以输入多个数字，并求其总和与平均数，每输入一个数字用回车确认，通过输入非数字来结束输入并输出执行结果：

```
public static void main(String[] args) {
    //扫描器接收键盘数据
    Scanner scan = new Scanner(System.in);

    double sum = 0; //和
    int m = 0;  //输入了多少个数字

    //通过循环判断是否还有输入，并在里面对每一次进行求和和统计
    while (scan.hasNextDouble()) {
        double x = scan.nextDouble();
        m = m + 1;
        sum = sum + x;
    }

    System.out.println(m + "个数的和为" + sum);
    System.out.println(m + "个数的平均值是" + (sum / m));
    scan.close();
}
```

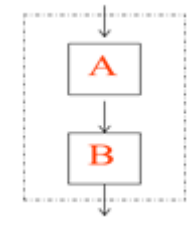
可能很多小伙伴到这里就看不懂写的什么东西了！这里我们使用了我们一会要学的流程控制语句，我们接下来就去学习这些语句的具体作用！

Java中的流程控制语句可以这样分类：顺序结构，选择结构，循环结构！这三种结构就足够解决所有的问题了！

顺序结构

JAVA的基本结构就是顺序结构，除非特别指明，否则就按照顺序一句一句执行。

顺序结构是最简单的算法结构。



语句与语句之间，框与框之间是按从上到下的顺序进行的，它是由若干个依次执行的处理步骤组成的，它是任何一个算法都离不开的一种基本算法结构。

顺序结构在程序流程图中的体现就是用流程线将程序框自上而地连接起来，按顺序执行算法步骤。

【演示】

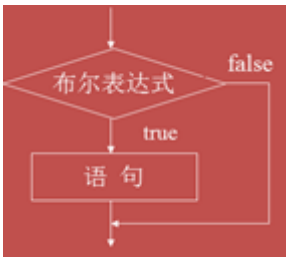
狂神社群笔记资料，禁止外传，本人QQ：24736743

```java
public static void main(String[] args) {
    System.out.println("Hello1");
    System.out.println("Hello2");
    System.out.println("Hello3");
    System.out.println("Hello4");
    System.out.println("Hello5");
}

//按照自上而下的顺序执行！依次输出。
```

选择结构

1、if单选择结构

我们很多时候需要去判断一个东西是否可行，然后我们才去执行，这样一个过程在程序中用if语句来表示：




```java
if(布尔表达式){
   //如果布尔表达式为true将执行的语句
}
```

意义：if语句对条件表达式进行一次测试，若测试为真，则执行下面的语句，否则跳过该语句。

【演示】比如我们来接收一个用户输入，判断输入的是否为Hello字符串：

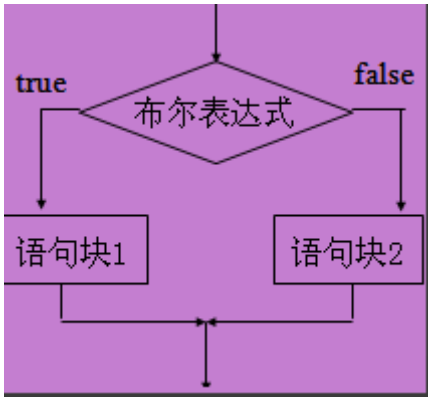
```java
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    //接收用户输入
    System.out.print("请输入内容：");
    String s = scanner.nextLine();

    if (s.equals("Hello")){
        System.out.println("输入的是："+s);
    }

    System.out.println("End");
    scanner.close();
}
```

【equals方法是用来进行字符串的比较的，之后会详解，这里大家只需要知道他是用来比较字符串是否一致的即可！和==是有区别的。】

2、if双选择结构

那现在有个需求，公司要收购一个软件，成功了，给人支付100万元，失败了，自己找人开发。这样的需求用一个if就搞不定了，我们需要有两个判断，需要一个双选择结构，所以就有了if-else结构。

狂神社群笔记资料，禁止外传，本人QQ：24736743




```
if(布尔表达式){
   //如果布尔表达式的值为true
}else{
   //如果布尔表达式的值为false
}
```

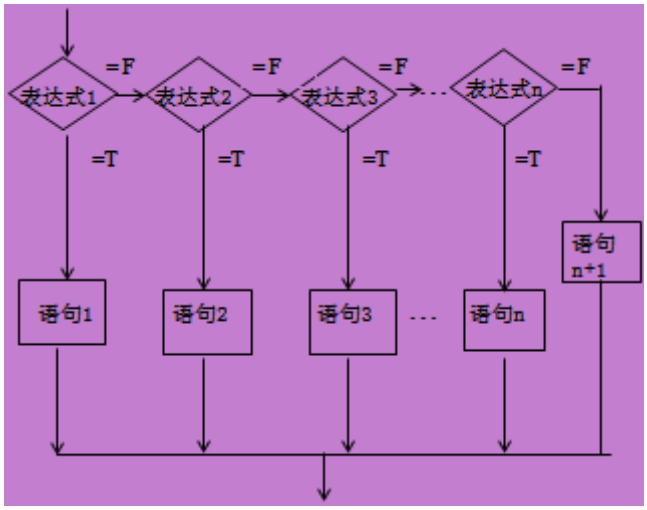
意义：当条件表达式为真时，执行语句块1，否则，执行语句块2。也就是else部分。

【演示】我们来写一个示例：考试分数大于60就是及格，小于60分就不及格。

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("请输入成绩：");
    int score = scanner.nextInt();

    if (score>60){
        System.out.println("及格");
    }else {
        System.out.println("不及格");
    }

    scanner.close();
}
```

3、if多选择结构

我们发现上面的示例不符合实际情况，真实的情况还可能存在ABCD，存在区间多级判断。比如90-100就是A，80-90 就是B..等等，在生活中我们很多时候的选择也不仅仅只有两个，所以我们需要一个多选择结构来处理这类问题！



狂神社群笔记资料，禁止外传，本人QQ：24736743

```
if(布尔表达式 1){
   //如果布尔表达式 1的值为true执行代码
}else if(布尔表达式 2){
   //如果布尔表达式 2的值为true执行代码
}else if(布尔表达式 3){
   //如果布尔表达式 3的值为true执行代码
}else {
   //如果以上布尔表达式都不为true执行代码
}
```

if 语句后面可以跟 else if...else 语句，这种语句可以检测到多种可能的情况。

使用 if，else if，else 语句的时候，需要注意下面几点：

- if 语句至多有 1 个 else 语句，else 语句在所有的 else if 语句之后。
- if 语句可以有若干个 else if 语句，它们必须在 else 语句之前。
- 一旦其中一个 else if 语句检测为 true，其他的 else if 以及 else 语句都将跳过执行。

【演示】我们来改造一下上面的成绩案例，学校根据分数区间分为ABCD四个等级！

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("请输入成绩：");
    int score = scanner.nextInt();

    if (score==100){
        System.out.println("恭喜满分");
    }else if (score<100 && score >=90){
        System.out.println("A级");
    }else if (score<90 && score >=80){
        System.out.println("B级");
    }else if (score<80 && score >=70){
        System.out.println("C级");
    }else if (score<70 && score >=60){
        System.out.println("D级");
    }else if (score<60 && score >=0){
        System.out.println("不及格！");
    }else {
        System.out.println("成绩输入不合法！");
    }

    scanner.close();
}
```

【我们平时写程序一定要严谨，不然之后修补Bug是一件十分头疼的事情，你要哦在编写代码的时候就把所有的问题都思考清除，再去一个个解决，这才是一个优秀的程序员应该做的事情，多思考，多犯错！】

4、嵌套的if结构

使用嵌套的 if...else 语句是合法的。也就是说你可以在另一个 if 或者 else if 语句中使用 if 或者 else if 语句。你可以像 if 语句一样嵌套 else if...else。

```
if(布尔表达式 1){
   ////如果布尔表达式 1的值为true执行代码
   if(布尔表达式 2){
      ////如果布尔表达式 2的值为true执行代码
   }
}
```

有时候我们在解决某些问题的时候，需要缩小查找范围，需要有层级条件判断，提高效率。比如：我们需要寻找一个数，在1-100之间，我们不知道这个数是多少的情况下，我们最笨的方式就是一个个去对比，看他到底是多少，这会花掉你大量的时间，如果可以利用if嵌套比较，我们可以节省大量的成本，如果你有这个思想，你已经很优秀了，因为很多大量的工程师就在寻找能够快速提高，查找和搜索效率的方式。为此提出了一系列的概念，我们生活在大数据时代，我们需要不断的去思考如何提高效率，或许哪一天，你们想出一个算法，能够将分析数据效率提高，或许你就可以在历史的长河中留下一些痕迹了，当然这是后话。

【记住一点就好，所有的流程控制语句都可以互相嵌套，互不影响！】

5、switch多选择结构

多选择结构还有一个实现方式就是switch case 语句。

switch case 语句判断一个变量与一系列值中某个值是否相等，每个值称为一个分支。

```
switch(expression){
    case value :
       //语句
       break; //可选
    case value :
       //语句
       break; //可选
    //你可以有任意数量的case语句
    default : //可选
       //语句
}
```

switch case 语句有如下规则：

- switch 语句中的变量类型可以是： byte、short、int 或者 char。从 Java SE 7 开始，switch 支持字符串 String 类型了，同时 case 标签必须为字符串常量或字面量。
- switch 语句可以拥有多个 case 语句。每个 case 后面跟一个要比较的值和冒号。
- case 语句中的值的数据类型必须与变量的数据类型相同，而且只能是常量或者字面常量。
- 当变量的值与 case 语句的值相等时，那么 case 语句之后的语句开始执行，直到 break 语句出现才会跳出 switch 语句。
- 当遇到 break 语句时，switch 语句终止。程序跳转到 switch 语句后面的语句执行。case 语句不必须要包含 break 语句。如果没有 break 语句出现，程序会继续执行下一条 case 语句，直到出现 break 语句。
- switch 语句可以包含一个 default 分支，该分支一般是 switch 语句的最后一个分支（可以在任何位置，但建议在最后一个）。default 在没有 case 语句的值和变量值相等的时候执行。default 分支不需要 break 语句。

switch case 执行时，一定会先进行匹配，匹配成功返回当前 case 的值，再根据是否有 break，判断是否继续输出，或是跳出判断。

```
public static void main(String args[]){
    //char grade = args[0].charAt(0);
    char grade = 'C';

```

```
    switch(grade)
    {
        case 'A' :
            System.out.println("优秀");
            break;
        case 'B' :
        case 'C' :
            System.out.println("良好");
            break;
        case 'D' :
            System.out.println("及格");
            break;
        case 'F' :
            System.out.println("你需要再努力努力");
            break;
        default :
            System.out.println("未知等级");
    }
    System.out.println("你的等级是 " + grade);
}
```

如果 case 语句块中没有 break 语句时，匹配成功后，从当前 case 开始，后续所有 case 的值都会输出。如果后续的 case 语句块有 break 语句则会跳出判断。【case穿透】

```
public static void main(String args[]){
    int i = 1;
    switch(i){
        case 0:
            System.out.println("0");
        case 1:
            System.out.println("1");
        case 2:
            System.out.println("2");
        case 3:
            System.out.println("3");
            break;
        default:
            System.out.println("default");
    }
}
```

输出：1，2，3。

【JDK7增加了字符串表达式】

```
public static void main(String[] args) {
    String  name = "狂神";

    switch (name) {   //JDK7的新特性，表达式结果可以是字符串！！！
        case "秦疆":
            System.out.println("输入的秦疆");
            break;
        case "狂神":
            System.out.println("输入的狂神");
            break;
        default:
            System.out.println("弄啥嘞！");
            break;
```


狂神社群笔记资料，禁止外传，本人QQ：24736743

```
    }
}
```

循环结构

上面选择结构中，我们始终无法让程序一直跑着，我们每次运行就停止了。这在真实环境中肯定是不行的嘛，比如网站服务器，肯定需要24小时全年无消息的跑着，我们需要规定一个程序运行多少次，运行多久，等等。所以按照我们编程是为了解决人的问题的思想，我们是不是得需要有一个结构来搞定这个事情！于是循环结构自然的诞生了！

顺序结构的程序语句只能被执行一次。如果您想要同样的操作执行多次，，就需要使用循环结构。

Java中有三种主要的循环结构：

- **while** 循环
- **do...while** 循环
- **for** 循环

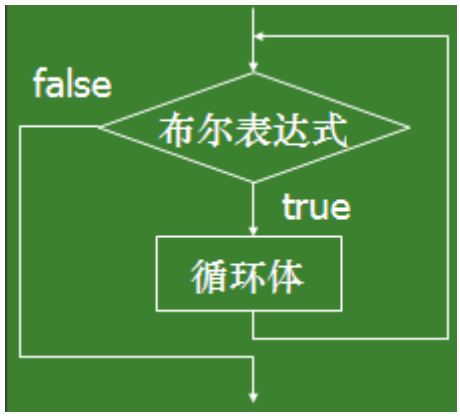
在Java5中引入了一种主要用于数组的增强型for循环。

1、while 循环

while是最基本的循环，它的结构为：

```
while( 布尔表达式 ) {
  //循环内容
}
```

只要布尔表达式为 true，循环就会一直执行下去。



【图解】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判断是否为真。直到条件不成立，则循环结束。

我们大多数情况是会让循环停止下来的，我们需要一个让表达式失效的方式来结束循环。

方式有：循环内部控制，外部设立标志位！等

```
public static void main(String[] args) {
    int i = 0;
    //i小于100就会一直循环
    while (i<100){
        i++;
        System.out.println(i);
    }
}
```


狂神社群笔记资料，禁止外传，本人QQ：24736743

少部分情况需要循环一直执行，比如服务器的请求响应监听等。

```
public static void main(String[] args) {
    while (true){
        //等待客户端连接
        //定时检查
        //......
    }
}
```

循环条件一直为true就会造成无限循环【死循环】，我们正常的业务编程中应该尽量避免死循环。会影响程序性能或者造成程序卡死奔溃！

【案例：计算1+2+3+...+100=?】

```
public static void main(String[] args) {
    int  i = 0;
    int  sum = 0;
    while (i <= 100) {
        sum = sum+i;
        i++;
    }
    System.out.println("Sum= " + sum);
}
```

【科普：**高斯的故事**】

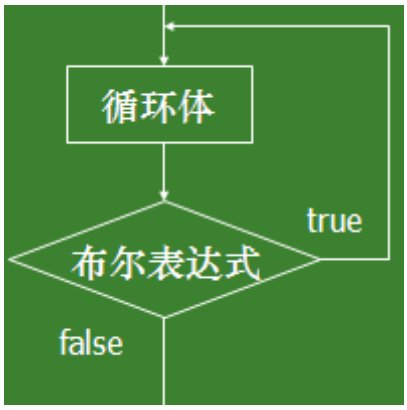
德国大数学家高斯（Gauss）：高斯是一对普通夫妇的儿子.他的母亲是一个贫穷石匠的女儿,虽然十分聪明,但却没有接受过教育,近似于文盲.在她成为高斯父亲的第二个妻子之前,她从事女佣工作.他的父亲曾做过园丁,工头,商人的助手和一个小保险公司的评估师.当高斯三岁时便能够纠正他父亲的借债账目的事情,已经成为一个轶事流传至今.他曾说,他在麦仙翁堆上学会计算.能够在头脑中进行复杂的计算,是上帝赐予他一生的天赋.
高斯用很短的时间计算出了小学老师布置的任务：对自然数从1到100的求和.他所使用的方法是：对50对构造成和101的数列求和（1 + 100,2 + 99,3 + 98.....）,同时得到结果：5050.这一年,高斯9岁.

2、do...while 循环

对于 while 语句而言，如果不满足条件，则不能进入循环。但有时候我们需要即使不满足条件，也至少执行一次。

do...while 循环和 while 循环相似，不同的是，do...while 循环至少会执行一次。

```
do {
        //代码语句
}while(布尔表达式);
```



注意：布尔表达式在循环体的后面，所以语句块在检测布尔表达式之前已经执行了。 如果布尔表达式的值为 true，则语句块一直执行，直到布尔表达式的值为 false。

我们用do...while改造一下上面的案例！

```
public static void main(String[] args) {
    int  i = 0;
    int  sum = 0;
    do {
        sum = sum+i;
        i++;
    }while (i <= 100);
    System.out.println("Sum= " + sum);
}
```

执行结果当然是一样的！

While和do-While的区别：

while先判断后执行。dowhile是先执行后判断！

Do...while总是保证循环体会被至少执行一次！这是他们的主要差别。

```
public static void main(String[] args) {
    int a = 0;
    while(a<0){
        System.out.println(a);
        a++;
    }
    System.out.println("-----");

    do{
        System.out.println(a);
        a++;
    } while (a<0);
}
```

3、For循环

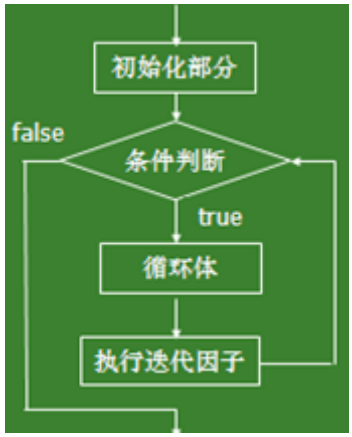
虽然所有循环结构都可以用 while 或者 do...while表示，但 Java 提供了另一种语句 —— for 循环，使一些循环结构变得更加简单。

for循环语句是支持迭代的一种通用结构，是最有效、最灵活的循环结构。

for循环执行的次数是在执行前就确定的。语法格式如下：

```
for(初始化; 布尔表达式; 更新) {
    //代码语句
}
```


狂神社群笔记资料，禁止外传，本人QQ：24736743



关于 for 循环有以下几点说明：

- 最先执行初始化步骤。可以声明一种类型，但可初始化一个或多个循环控制变量，也可以是空语句。
- 然后，检测布尔表达式的值。如果为 true，循环体被执行。如果为false，循环终止，开始执行循环体后面的语句。
- 执行一次循环后，更新循环控制变量(迭代因子控制循环变量的增减)。
- 再次检测布尔表达式。循环执行上面的过程。

【演示：while和for输出】

```
public static void main(String[] args) {
    int a = 1;    //初始化

    while(a<=100){  //条件判断
        System.out.println(a);    //循环体
        a+=2;  //迭代
    }
    System.out.println("while循环结束！");


    for(int i = 1;i<=100;i++){  //初始化//条件判断 //迭代
        System.out.println(i);    //循环体
    }
    System.out.println("while循环结束！");
}
```

我们发现，for循环在知道循环次数的情况下，简化了代码，提高了可读性。我们平时用到的最多的也是我们的for循环！

4、练习

【练习1：计算0到100之间的奇数和偶数的和】

```
public static void main(String[] args) {
    int oddSum = 0;  //用来保存奇数的和
    int evenSum = 0;  //用来存放偶数的和
    for(int i=0;i<=100;i++){
        if(i%2!=0){
            oddSum += i;
        }else{
            evenSum += i;
        }

    }
```


狂神社群笔记资料，禁止外传，本人QQ：24736743

```
    System.out.println("奇数的和："+oddSum);
    System.out.println("偶数的和："+evenSum);
}
```

【练习2：用while或for循环输出1-1000之间能被5整除的数，并且每行输出3个】

```
public static void main(String[] args) {
    for(int j = 1;j<=1000;j++){
        if(j%5==0){
            System.out.print(j+"\t");
        }
        if(j%(5*3)==0){
            System.out.println();
        }
    }
}
```

【练习3：打印九九乘法表】

```
1*1=1
1*2=2    2*2=4
1*3=3    2*3=6    3*3=9
1*4=4    2*4=8    3*4=12    4*4=16
1*5=5    2*5=10    3*5=15    4*5=20    5*5=25
1*6=6    2*6=12    3*6=18    4*6=24    5*6=30    6*6=36
1*7=7    2*7=14    3*7=21    4*7=28    5*7=35    6*7=42    7*7=49
1*8=8    2*8=16    3*8=24    4*8=32    5*8=40    6*8=48    7*8=56    8*8=64

1*9=9    2*9=18    3*9=27    4*9=36    5*9=45    6*9=54    7*9=63    8*9=72
 9*9=81
```

当然，成功的路不止一条，但是我们要追求最完美的一条，如果你做不到，不妨试试笨办法，依旧可以完成任务！比如一行行输出，也是可以搞定的。一定要多分析！

我们使用嵌套for循环就可以很轻松解决这个问题了！

第一步：我们先打印第一列，这个大家应该都会

```
for (int i = 1; i <= 9; i++) {
    System.out.println(1 + "*" + i + "=" + (1 * i));
}
```

第二步：我们把固定的1再用一个循环包起来

```
for (int i = 1; i <= 9 ; i++) {
    for (int j = 1; j <= 9; j++) {
        System.out.println(i + "*" + j + "=" + (i * j));
    }
}
```

第三步：去掉重复项，j<=i

```
for (int i = 1; i <= 9 ; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.println(j + "*" + i + "=" + (i * j));
    }
}
```

第四步：调整样式

```
for (int i = 1; i <= 9 ; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print(j + "*" + i + "=" + (i * j)+ "\t");
    }
    System.out.println();
}
```

通过本练习，大家要体会如何分析问题、如何切入问题！在我们以后写代码的过程中，一定要学会将一个大问题分解成若干小问题，然后，由易到难，各个击破！这也是我们以后开发项目时的基本思维过程。希望大家好好体会！

5、增强for循环

【这里我们先只是见一面，做个了解，之后数组我们重点使用】

Java5 引入了一种主要用于数组或集合的增强型 for 循环。

Java 增强 for 循环语法格式如下:

```
for(声明语句 : 表达式)
{
   //代码句子
}
```

声明语句：声明新的局部变量，该变量的类型必须和数组元素的类型匹配。其作用域限定在循环语句块，其值与此时数组元素的值相等。

表达式：表达式是要访问的数组名，或者是返回值为数组的方法。

【演示：增强for循环遍历输出数组元素】

```
public static void main(String[] args) {
    int [] numbers = {10, 20, 30, 40, 50};
    for(int x : numbers ){
        System.out.print( x );
        System.out.print(",");
    }

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

    String [] names ={"James", "Larry", "Tom", "Lacy"};
    for( String name : names ) {
        System.out.print( name );
        System.out.print(",");
    }
}
```

我们现在搞不懂这个没关系，就是拉出来和大家见一面，下章就讲解数组了！

break & continue

1、break 关键字

break 主要用在循环语句或者 switch 语句中，用来跳出整个语句块。

break 跳出最里层的循环，并且继续执行该循环下面的语句。

狂神社群笔记资料，禁止外传，本人QQ：24736743

【演示：跳出循环】

```
public static void main(String[] args) {
    int i=0;
    while (i<100){
        i++;
        System.out.println(i);
        if (i==30){
            break;
        }
    }
}
```

switch 语句中break在上面已经详细说明了，如果有疑惑可以回头看switch多选择结构小节；

2、continue 关键字

continue 适用于任何循环控制结构中。作用是让程序立刻跳转到下一次循环的迭代。

在 for 循环中，continue 语句使程序立即跳转到更新语句。

在 while 或者 do...while 循环中，程序立即跳转到布尔表达式的判断语句。

【演示】

```
public static void main(String[] args) {
    int i=0;
    while (i<100){
        i++;
        if (i%10==0){
            System.out.println();
            continue;
        }
        System.out.print(i);
    }
}
```

3、两者区别

break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语句也在switch语句中使用)

continue 语句用在循环语句体中，用于终止某次循环过程，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。

4、带标签的continue

【了解即可】

1. goto关键字很早就在程序设计语言中出现。尽管goto仍是Java的一个保留字，但并未在语言中得到正式使用；Java没有goto。然而，在break和continue这两个关键字的身上，我们仍然能看出一些goto的影子---带标签的break和continue。
2. “标签”是指后面跟一个冒号的标识符，例如：label:
3. 对Java来说唯一用到标签的地方是在循环语句之前。而在循环之前设置标签的唯一理由是：我们希望在其中嵌套另一个循环，由于break和continue关键字通常只中断当前循环，但若随同标签使用，它们就会中断到存在标签的地方。
4. 带标签的break和continue的例子：

【演示：打印101-150之间所有的质数】

```
public static void main(String[] args) {
    int count = 0;
    outer: for (int i = 101; i < 150; i ++) {
        for (int j = 2; j < i / 2; j++) {
            if (i % j == 0)
                continue outer;
        }
        System.out.print(i+ "  ");
    }
}
```

【看不懂没关系，只是了解一下即可，知道goto这个保留字和标签的写法】

狂神社群笔记资料，禁止外传，本人QQ：24736743