Java 入门之流程控制语句


作者 start筑梦 (/u/037b79d60790) + 关注
2016.12.10 22:13* 字数 1629 阅读 32 评论 0 喜欢 0
(/u/037b79d60790)

> 主要分为两种，Java **循环控制**和**条件判断。**

1. 循环控制

可能存在一种情况，当我们需要执行的代码块数次，通常被称为一个循环。
Java有非常灵活的三循环机制。可以使用以下三种循环之一：

- while 循环
- do...while 循环
- for 循环

while 循环 ，特点：先判断，后执行

while循环是一个控制结构，可以重复的特定任务次数。

while循环的语法是：

```
while(Boolean_expression)
{
   //Statements
}
```

在执行时，如果布尔表达式的结果为真，则循环中的动作将被执行。只要该表达式的结果为真，执行将继续下去。

在这里，while循环的关键点是循环可能不会永远运行。当表达式进行测试，结果为假，循环体将被跳过，在while循环之后的第一个语句将被执行。

```
4
5      /**
6       * while循环是一个控制结构，可以重复的特定任务次数
7       */
8      public static void main(String args[]) {
9            int x = 10;
10           while( x < 20 ) {
11              System.out.print("value of x : " + x );
12              x++;
13              System.out.print("\n");
14           }
15       }
16
17 }
18
```

while循环.png

这将产生以下结果:

```
value of x : 10
value of x : 11
value of x : 12
value of x : 13
value of x : 14
value of x : 15
value of x : 16
value of x : 17
value of x : 18
value of x : 19
```

do...while 循环 特点：先执行，后判断

do ... while循环类似于while循环，不同的是一个do ... while循环是保证至少执行一次。do...while循环的语法是：

```
do
{
  //循环操作
} while (Boolean_expression);
```

请注意，布尔表达式出现在循环的结尾，所以在循环中的语句执行前一次布尔测试。如果布尔表达式为真，控制流跳回，并且在循环中的语句再次执行。这个过程反复进行，直到布尔表达式为假。

```
public static void main(String[] args) {
    /**
     * do...while 至少会被执行一次
     */
    int sum=0; //保存偶数的和
    int num=2; //代表1-50之间的偶数
    do {
        sum =sum+num;   //累加求和
        num=num+2;    //每执行一次将变量值加2
    } while (num<=50);  //当变量小于或等于50时重复执行循环
    System.out.println("50以内的偶数之和为 ： "+sum);
}
```

运行结果如下:

```
50以内的偶数之和为 ： 650
```

for 循环

for循环是一个循环控制结构，可以有效地编写需要执行的特定次数的循环。知道一个任务要重复多少次的时候，for循环是有好处的。

for循环的语法是：

```
    // 循环变量初始化;循环条件;循环变量变化部分
for(initialization; Boolean_expression; update)
{
   //Statements
}
```

- 执行循环变量初始化部分，设置循环的初始状态，此部分在整个循环中只执行一次
- 进行循环条件的判断，如果条件为 true ，则执行循环体内代码；如果为 false ，则直接退出循环
- 执行循环变量变化部分，改变循环变量的值，以便进行下一次条件判断
- 依次重新执行< 2 >、< 3 >、< 4 >，直到退出循环

特点：相比 while 和 do...while 语句**结构更加简洁易读**

```
30
31      public static void main(String[] args) {
32          int sum = 0; // 保存不能被3整除的数之和
33
34          // 循环变量 i 初始值为 1 ,每执行一次对变量加 1，只要小于等于 100 就重复执行循环
35          for (int i = 1;i<=100;i++) {
36
37              // 变量 i 与 3 进行求模（取余），如果不等于 0，则表示不能被 3 整除
38              if (i % 3 != 0) {
39                  sum = sum + i; // 累加求和
40              }
41          }
42          System.out.println("1到100之间不能被3整除的数之和为：" + sum);
43      }
44  }
45
```

for循环.jpg

运行结果如下：

```
1到100之间不能被3整除的数之和为：3367
```

break 关键字

生活中，我们经常会因为某些原因中断既定的任务安排。如在参加 10000 米长跑时，才跑了 500 米就由于体力不支，需要退出比赛。在 Java 中，我们可以使用 break 语句退出指定的循环，直接执行循环后面的代码。

```
44
45      public static void main(String[] args) {
46          // 保存累加值
47          int sum = 0;
48
49          // 从1循环到10
50          for (int i = 1; i <= 10; i++) {
51
52              // 每次循环时累加求和
53              sum = sum + i;
54
55              // 判断累加值是否大于20，如果满足条件则退出循环
56              if ( sum > 20) {
57
58                  System.out.println("当前的累加值为:" + sum);
59
60                  //退出循环
61                     break;
62
63              }
64          }
65      }
```

break.png

运行结果如下：

```
当前的累加值为:21
```

continue 关键字

continue关键字可以在任一环的控制结构使用。它使循环立即跳转到循环的下一次迭代。

- 在for循环中，continue关键字会导致控制流立即跳转到更新语句。
- 在一个while循环或do/while循环，控制流立即跳转到布尔表达式。

```
65          }*/
66      public static void main(String[] args) {
67          int sum=0; //保存累加值
68          for (int i = 0; i <=10; i++) {
69              // 如果i为奇数,结束本次循环，进行下一次循环
70              if(i%2==1){
71                  continue;
72              }
73              sum=sum+i;
74          }
75          System.out.print("1到10之间的所有偶数的和为：" + sum);
76      }
77  }
```

continue.png

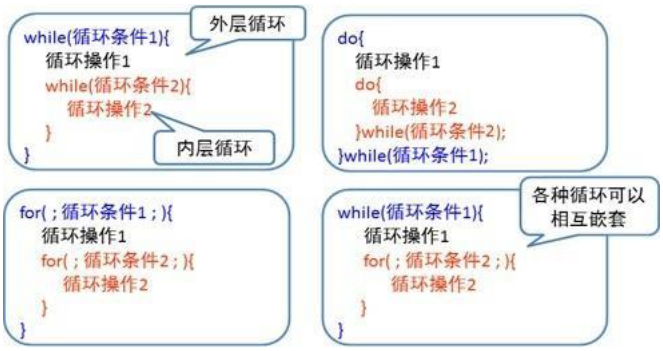
运行结果如下：

```
1到10之间的所有偶数的和为：30
```

Java 循环语句之多重循环

循环体中包含循环语句的结构称为**多重循环**。三种循环语句可以自身嵌套，也可以相互嵌套，最常见的就是**二重循环**。在二重循环中，外层循环每执行一次，内层循环要执行一圈。

如下所示：



例如：使用 * 打印长方形：

```
76
77          //外层循环控制打印的行数
78          for (int i = 0; i <=3; i++) {
79              //内层循环控制每行打印的 * 号数
80              for (int j = 0; j <=8; j++) {
81                  System.out.print("*");
82              }
83              //每行打印完毕换行
84              System.out.println();
85          }
86      }
87  }
```

长方形.png

运行结果如下：

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

执行流程：当 i = 1 时，外层循环条件成立，进入内层循环，开始打印第一行内容。此时，j 从 1 开始，循环 8 次，内层循环结束后换行，实现第一行 8 个 * 的输出。接下来返回外层循环 i 变为 2 ，准备打印下一行，依此类推，直到完成长方形的打印。

例如：使用 * 打印直角三角形：

```
        System.out.println("打印直角三角形");
        //外层循环控制打印的行数
        for (int i = 0; i <=3; i++) {
            //内层循环控制每行打印的 *  号数
            for (int j = 0; j <=i; j++) {
                System.out.print("*");
            }
            //每行打印完毕换行
            System.out.println();
        }
    }
```

直角三角形.png

运行结果如下：

```
打印直角三角形
*
**
***
****
```

Java 条件判断

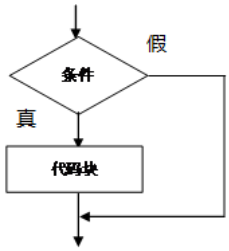
在 Java 中有两种类型的条件判断语句，它们分别是：

- if 语句
- switch 语句

生活中，我们经常需要先做判断，然后才决定是否要做某件事情。例如，如果考试成绩大于 90 分，则奖励一个 IPHONE 5S 。对于这种“需要先判断条件，条件满足后才执行的情况”，就可以使用**if 条件语句**实现。
语法：

```
if(条件){
        //条件成立时执行的代码
       }
```

执行过程：

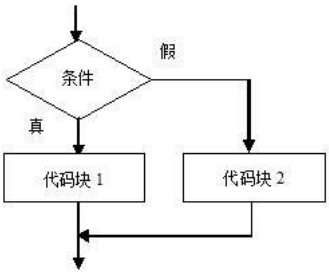


if...else 语句

任何 if 语句后面可以跟一个可选的 else 语句，当布尔表达式为 false，语句被执行。
语法：

```
if(条件){
         //条件成立时执行的代码
    }else{
        //条件不成立时执行的代码
          }
```

执行过程：



多重 if 语句

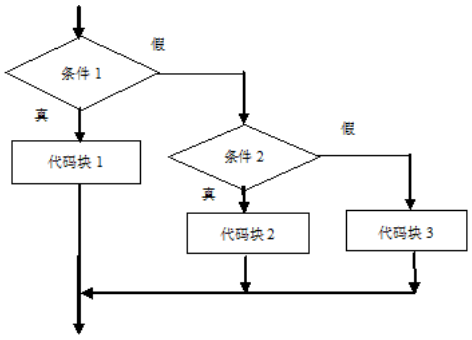
if 后面可以跟一个可选的 else if...else 语句，在测试不同条件下单一的 if 语句和 else if 语句是非常有用的。
当使用 if , else if , else 语句时有几点要牢记。

- 一个 if 语句可以有0个或一个 else 语句 且它必须在 else if 语句的之后。
- 一个 if 语句 可以有0个或多个 else if 语句且它们必须在 else 语句之前。
- 一旦 else if 语句成功, 余下 else if 语句或 else 语句都不会被测试执行。

语法：

```
if(条件1){
         //代码块1
    }else if(条件2){
        //代码块1
    }else{
       //代码块3
}
```

执行过程：
**



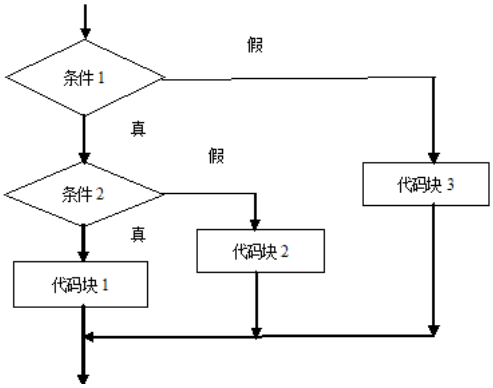
嵌套 if 语句

嵌套 if 语句，只有当外层 if 的条件成立时，才会判断内层 if 的条件。例如，活动计划的安排，如果今天是工作日，则去上班，如果今天是周末，则外出游玩；同时，如果周末天气晴朗，则去室外游乐场游玩，否则去室内游乐场游玩。

语法：

```
if(条件1){
    if(条件2){
      代码块1
    }else{
      代码块2
    }
}else{
   代码块3
}
```

执行过程：



switch 语句

当需要对选项进行**等值判断**时，使用 switch 语句更加简洁明了。
语法：

```
switch(表达式){
  case 值1:
    执行代码块1
    break ;
  case 值2:
    执行代码块2
    break ;
  case 值n:
    执行代码块n
    break ;
  default :
    默认执行的代码
}
```

执行过程：当 switch 后表达式的值和 case 语句后的值相同时，从该位置开始向下执行，直到遇到 break 语句或者 switch 语句块结束；如果没有匹配的 case 语句则执行 default 块的代码。

```
90
91          char grade = 'C';
92          switch(grade)
93          {
94             case 'A' :
95                System.out.println("Excellent!");
96                break;
97             case 'B' :
98             case 'C' :
99                System.out.println("Well done");
100               break;
101            case 'D' :
102               System.out.println("You passed");
103            case 'F' :
104               System.out.println("Better try again");
105               break;
106            default :
107               System.out.println("Invalid grade");
108         }
109         System.out.println("Your grade is " + grade);
110
111
112   }
```

switch.png

运行结果如下:

```
Well done
Your grade is C
```

重拾java (/nb/8247566)

start筑梦 (/u/037b79d60790)
写了 12014 字，被 9 人关注，获得了 14 个喜欢
(/u/037b79d60790)

+关注

在成长路上的小迷糊Android程序媛 爱看书，爱运动，爱旅行，爱唱歌，爱生活

被以下专题收入，发现更多相似内容

java入门基础

Java基础