Java程序逻辑控制

在程序开发的过程中，会存在着有三种程序逻辑：顺序结构，分支结构、循环结构，之前的大部分代码都是顺序结构的定义，即：所有的程序将按照定义的代码顺序依次执行。
1.IF分支结构
IF分支结构主要是针对关系表达式进行判断处理的分支操作，对于分支语句主要有三类使用形式，主要关键字就是：if、else。

IF判断：
```
if(布尔表达式) {
条件满足时执行 ;
}
```

if.else：
```
if(布尔表达式) {
条件满足时执行 ;
} else {
条件不满足时执行 ;
 }
```

多条件判断：
```
if(布尔表达式) {
条件满足时执行 ;
} else if (布尔表达式) {
条件满足时执行 ;
} else if (布尔表达式) {
条件满足时执行 ;
} [else {
条件不满足时执行 ;
}]
```

范例：观察一个If判断
```
public class JavaStudy {
   public static void main(String mage[]) {
      int age = 10 ;
      if (age >= 10 && age <= 11) {
         System.out.println("满足要求") ;
      }
      System.out.println("不满足要求") ;
   }
}
```
判断是超过10，如果是则继续运行，如果不是的话则弹出第二个 ”不满足要求“

范例：if..else (不满足条件时执行)
```
public class JavaStudy {
   public static void main(String mage[]) {
      double money = 20 ;
      if (money >= 19.9 ) {
         System.out.println("OK") ;
      } else {
         System.out.print("ON") ;
```

```
        }
        System.out.println("hELLO") ;
    }
}
```

如果参数大于19.9则显示OK之后继续。

```
public class JavaStudy {
    public static void main(String mage[]) {
        double money = 10 ;
        if (money >= 19.9 ) {
            System.out.println("OK") ;
        } else {
            System.out.print("ON") ;
        }
        System.out.println("hELLO") ;
    }
}
```

如果参数小于19.9则直接执行on并继续往下执行。

IF的特点就是可以进行若干个条件判断。

范例：多条件判断

```
public class JavaStudy {
    public static void main(String mage[]) {
        double score = 90 ;
        if (score >= 90.00 && score <=100) {
            System.out.println("A+优等生") ;
        } else if(score >=60 && score < 90 ) {
            System.out.println("B+中等生") ;
        } else {
            System.out.println("C差生") ;
        }
    }
}
```

在进行多条件判断的时候可以不写上else语句，但是如果要养成好的习惯，还是要加上else的

2.switch分支语句

switch是一个开关语句，它主要根据内容来进行判断， 但是需要注意的是switch中可以判断的只能够是数据（int、char、枚举、string）而不能使用逻辑判断

```
switch(数据) {
case 数值 ： {
数值满足时执行 ;
[break  ;]
}
case 数值 ：
数值满足时执行 ;
[break  ;]
[default;
所有判断数值不满足时执行 :
[break ;]
]
}
```

范例：观察switch语句

```
public class JavaStudy {
   public static void main(String mage[]) {
      int ch = 2 ;
      switch (ch) {
      case 2 :
         System.out.println("设置内容是2") ;
      case 1 : {
         System.out.println("设置内容是2") ;
      }
      default : {
         System.out.println("没内容满足") ;
         }

      }
   }
}
```

switch语句在设计的时候，如果在每一个case后面没有追加break语句，那么会在第一个匹配case之后继续执行，一直到全部的switch中后续代码执行完毕，或者遇见break

范例：使用break

```
 public class JavaStudy {
   public static void main(String mage[]) {
      int ch = 2 ;
      switch (ch) {
      case 2 :
         System.out.println("设置内容是2") ;
         break ;
      case 1 : {
         System.out.println("设置内容是2") ;
         break ;
      }
      default : {
         System.out.println("没内容满足") ;
         break ;
         }

      }
   }
}
```

从1.7开始，Oracle公司推出jdk1.7板板里面将开发者呼吁10年以上的功能加入到了字符串数据的判断。

范例：判断字符串

```
 public class JavaStudy {
   public static void main(String mage[]) {
      String str = "hello" ;
      switch (str) {
         case "Hello": {
            System.out.println("Hello") ;
            break ;
         }
```

```
        case "hello": {
            System.out.println("hello") ;
            break ;
        }
        default: {
            System.out.println("No") ;
            }
        }
    }
}
```
switch是一个时代的见证。

3.while循环结构
循环结构指的是某一段代码被重复执行处理的操作，在程序之中提供while，该语句有两段是使用形式；

while循环：
```
while(布尔表达式) {
条件满足时执行 ;
修改循环条件 ;
}
```

do...while循环：
```
do {
条件满足时执行 ;
修改循环条件;
} while(布尔表达式)
```

范例：实现 1 ~ 100的累加
```
public class JavaStudy {
    public static void main(String mage[]) {
        int sum = 0 ; //保存最终的计算总和
        int num = 1 ; //进行循环控制
        while (num <= 100) { //循环执行条件
            sum += num ; //累加
            num ++ ; //修改循环条件
        }
        System.out.println(sum) ;
    }
}
```

除了可以使用while循环，也可是使用 do..while循环

范例：使用do..while实现数字累加
```
public class JavaStudy {
    public static void main(String mage[]) {
        int sum = 0 ; //保存最终的计算总和
        int num = 1 ; //进行循环控制
        do { //循环执行条件
            sum += num ; //累加
            num ++ ; //修改循环条件
        } while (num <= 100) ;
        System.out.println(sum) ;
```

```
    }
}
```

while于do..while循环的区别：
while：先判断后执行
do..while：先执行一次候判断

4.for循环
for循环是常用的使用结构。
for (定义循环的初始化数值 ; 循环判断 ; 修改循环数据) {
循环语句执行
}

范例：使用for循环，实现 1 ~100累加

```
public class JavaStudy {
    public static void main(String mage[]) {
        int sum = 0 ; //保存最终的计算总和
        for (int x = 2 ; x <= 100 ; x ++) {
            sum += x ;//累加
        }
        System.out.println(sum) ;
    }
}
```

三个操作的定义，可以拆开处理

```
public class JavaStudy {
    public static void main(String mage[]) {
        int sum = 0 ; //保存最终的计算总和
        int x = 1 ;   //循环初始化
        for ( ; x <= 100 ; ) {
            sum += x ;//累加
             x ++ ; //修改循环条件
        }
        System.out.println(sum) ;
    }
}
```

对于while和for循环只有一个参考标准：
在明确确定循环次数的情况下for循环优先选择；
在不知道循次数的循环结束的情况下使用while循环

5.循环控制
循环控制语句中有两个控制语句：break、continue：
1.break主要的功能是退出整个循环结构；

```
public class JavaStudy {
    public static void main(String mage[]) {
        for (int x = 0 ; x < 10 ; x ++) {
            if (x == 3) {
                break ; //循环结束
            }
            System.out.println(x + "我好帅") ;
        }
    }
}
```

2.continue只是结束当前循环。

```
public class JavaStudy {
    public static void main(String mage[]) {
        for (int x = 0 ; x < 10 ; x ++) {
            if (x == 3) {
                continue ; //循环结束
            }
            System.out.println(x + "我好帅") ;
        }
    }
}
```

当执行到continue的时候就结束到了后续代码不在进行执行，而直接跳过此，从而继续往下执行。比如你跳坑对吧。比如下面会有一个生动形象的解释。

—执行——(这有一个坑：我是continue)—继续执行————————
你会去跳嘛

在C语言中里面有一个goto的指令，这个指令会直接造成代码的混乱。但是在Java里面可以利用continue实现部分goto的功能

```
public class JavaStudy {
    public static void main(String mage[]) {
        point: for (int x = 0 ; x < 10  ; x ++ ) {
            for (int y = 0 ; y < 3 ; y ++) {
                if (x == y) {
                    continue point ; //循环结束
                }
                System.out.print(x + "-") ;
            }
            System.out.println() ;
        }
    }
}
```

不建议在代码中出现此代码。，不到万不得已建议不要使用。

6.循环嵌套

在一个循环嵌套之中，嵌套其他的循环语句就称为循环嵌套处理，循环嵌套的层次越多，时间复杂度就越多

范例：打印乘法口诀

```
public class JavaStudy {
    public static void main(String mage[]) {
        for (int x = 1 ; x <= 9 ; x ++)  { // 口诀表最多9行
            for (int y = 1 ; y <= x ; y ++ ) {
                System.out.print(y + "*" + x + "=" + (x * y) + "\t") ;
            }
                System.out.println() ;
        }
    }
}
```

范例：打印一个三角形

```
public class JavaStudy {
    public static void main(String mage[]) {
        int line = 5 ;  //总体个数
```

```
        for (int x = 0 ; x < line ; x ++) {
            for (int y = 0 ; y < line - x ; y ++ ) {
                System.out.print(" ");
            }
            for (int y = 0 ; y <= x ; y ++) {
                System.out.print("* ");
            }
            System.out.println() ;
        }
    }
}
```