# 概述

## 什么是流程控制语句

流程控制语句：可以控制程序的执行流程。

## 分类



## 执行流程：

从上往下，依次执行。

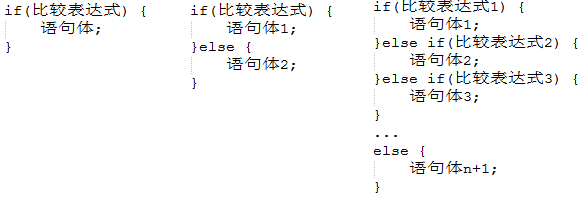
# 分支结构

## 分类

if语句

switch语句

## if-else语句



**public** **class** Test {

**public** **static** **void** main(String[] args) {

System.***out***.print("输入小明的期末成绩:");

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

**int** obj = s.nextInt();

**if** (obj > 90 && obj <= 100) {

System.***out***.println("奖励一辆BMW");

} **else** **if** (obj > 80 && obj <= 90) {

System.***out***.println("奖励一个台iphone5s");

} **else** **if** (obj >= 60 && obj <= 80) {

System.***out***.println("奖励一本参考书");

}**else** {

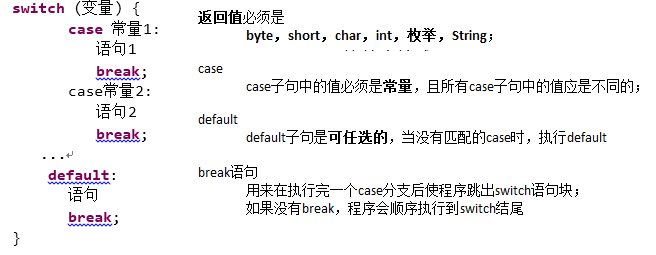
System.***out***.println("什么奖励也没有");

}

}

}

## switch语句



**public** **class** Test {

**public** **static** **void** main(String[] args) {

**int** i = 6;

**switch** (i) {

**case** 0:

System.***out***.println("zero");

**break**;

**case** 1:

System.***out***.println("one");

**break**;

**default**:

System.***out***.println("default");

**break**;

}

}

}

# 循环结构

## 基本概念

### 功能

在某些条件满足的情况下，反复执行特定代码的功能

### 四个组成部分

初始化部分（init\_statement）

循环条件部分（test\_exp）

循环体部分（body\_statement）

迭代部分（alter\_statement）

### 分类

for 循环

while 循环

do/while 循环

## for 循环语句



**public** **class** Test {

**public** **static** **void** main(String[] args) {

**int** result = 0;

**for**(**int** i=1; i<=100; i++) {

result += i;

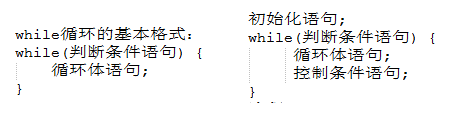
}

System.***out***.println("result=" + result);

}

}

## while 循环语句



**public** **class** Test {

**public** **static** **void** main(String[] args) {

**int** result = 0;

**int** i=1;

**while**(i<=100) {

result += i;

i++;

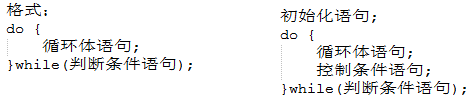
}

System.***out***.println("result=" + result);

}

}

## do-while 循环语句

****

**public** **class** Test {

**public** **static** **void** main(String[] args) {

**int** result = 0, i=1; //1

**do**{

result += i;

i++;

}**while**(i<=100);

System.***out***.println("result=" + result);

}

}

## 三种循环区别

do...while循环至少执行一次循环体。

而for,while循环必须先判断条件是否成立，然后决定是否执行循环体语句。

for循环和while循环的区别：

如果你想在循环结束后，继续使用控制条件的那个变量，用while循环，否则用for循环。不知道用谁就用for循环。因为变量及早的从内存中消失，可以提高内存的使用效率。

## 嵌套循环

将一个循环放在另一个循环体内，就形成了嵌套循环。其中，for ,while ,do…while均可以作为外层循环和内层循环。

实质上，嵌套循环就是把内层循环当成外层循环的循环体。当只有内层循环的循环条件为false时，才会完全跳出内层循环，才可结束外层的当次循环，开始下一次的循环。

设外层循环次数为m次，内层为n次，则内层循环体实际上需要执行m\*n=mn次。

九九乘法表：

**public** **class** Test {

**public** **static** **void** main(String[] args) {

**for** (**int** i = 1; i <= 9; i++) {

**for** (**int** j = 1; j <= i; j++) {

System.***out***.print(i +"\*"+ j +"="+ i\*j + "\t");//制表符

}

System.***out***.println();

}

}

}

# 控制跳转语句

## break 语句

使用在**switch语句**和**循环语句**

用于终止某个语句块的执行

break语句出现在多层嵌套的语句块中时，可以通过标签指明要终止的是哪一层语句块

**int** i;

/\*\*

\* break:1.break语句用于终止某个语句块的执行

\* 2.出现在多层嵌套的语句块中时，可以通过标签指明要终止的是哪一层语句块

\*/

**for**(i = 1 ;i <= 10 ; i++){

**if** (i % 4 == 0) {

System.***out***.print(i+" ");

**break**;

}

} //4

## contune 语句

只能用于**循环语句**中

用于跳过某个循环语句块的一次执行

contune 语句出现在多层嵌套的语句体中时，可以通过标签指明要跳过的是哪一层循环

**int** i;

/\*\*

\* continue1.用于跳过某个循环语句块的一次执行

\* 2.出现在多层嵌套的语句体中时，可以通过标签指明要跳过的是哪一层循环

\*/

**for**(i = 1 ;i <= 10 ; i++){

**if** (i % 4 == 0) {

System.***out***.print(i+" ");

**continue**;

}

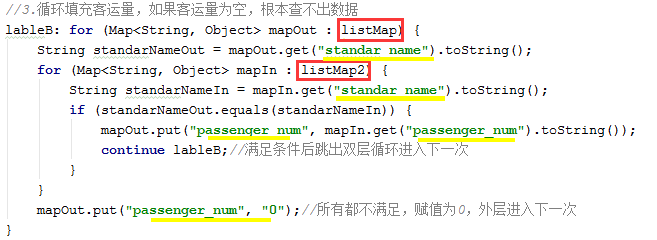
} //4 8

## return 语句

并非专门用于结束循环的，它的功能是结束一个方法。当一个方法执行到一个return语句时，这个方法将被结束。

## 跳出双层for循环

进入下一次双层循环



lableB: **for** (Map<String, Object> mapOut : listMap) {  
 String standarNameOut = mapOut.get(**"standar\_name"**).toString();  
 **for** (Map<String, Object> mapIn : listMap2) {  
 String standarNameIn = mapIn.get(**"standar\_name"**).toString();  
 **if** (standarNameOut.equals(standarNameIn)) {  
 mapOut.put(**"passenger\_num"**, mapIn.get(**"passenger\_num"**).toString());  
 **continue** lableB;*//满足条件后跳出双层循环进入下一次* }  
 }  
 mapOut.put(**"passenger\_num"**, **"0"**);*//所有都不满足，赋值为0，外层进入下一次*}