

day03——程序流程控制

各位同学，今天我们学习一个全新的知识——程序流程控制。什么是流程控制呢？说白了就是控制程序的执行顺序。

先给同学们介绍一下，程序有哪些流程控制、以及Java提供了哪些方案来控制程序的执行顺序？

程序的流程控制一般分为3种：**顺序结构、分支结构、循环结构**

- 顺序结构：就是不加任何控制，代码从main方法开始自上而下执行
- 分支结构：就是根据条件判断是true还是false，有选择性的执行哪些代码。在Java语言中提供了两个格式if、switch
- 循环结构：就是控制某一段代码重复执行。在Java语言中提供了三种格式，for、while、do-while

1661129154598

以上就是我们今天要学习的课程内容

一、分支结构


1.1 if分支

各位同学，接下来我们学习分支结构的第一种形式——if分支。

if它的作用，是用于对条件进行判断，判断的结果只可能有两个值true或者false，然后根据条件判断的结果来决定执行那段代码。

1. if分支的应用场景有哪些呢？

比如，在火车站、地铁站等公共场所，会对过往的旅客测体温。如果体温在37度以内，就属于正常的；如果体温在37度以上，测体温的装置就会报警。

1661130193692

再比如，你在使用微信付钱时，微信内部的程序会先判断你的余额是否足够，如果足够就可以支付成功；如果余额不足，就会提示支付失败。

1661133550463

2. if分支的格式

接下来，我们来看一看if分支在Java语言中长什么样子呢？在Java中if分支有三种格式。

1661131177976

接下来我们用一些实际案例给大家演示一下if语句的应用，以及每一种if语句的执行流程。

3. if 第一种形式

if 第一种形式的代码格式，和执行流程如下图所示

1661131910804

if 第一种形式执行流程如下：

- 如果 条件表达式 为**true**，就执行下面的语句体
- 如果 条件表达式 为**false**，就不执行

```
// 需求：测量用户体温，发现体温高于37度就报警。  
double t = 36.9;  
if(t > 37){  
    System.out.println("这个人的温度异常，把他赶紧带走~~");  
}
```

4. if 第二种形式

if 第二种形式的代码格式，和执行流程如下图所示



if 第二种形式执行流程如下：
如果 条件表达式 为true,就执行下面的语句体1
如果 条件表达式 为false,就执行else下面的语句体2

```
// 需求2：发红包，你的钱包余额是99元，现在要发出90元  
// 如果钱够触发发红包的动作，如果钱不够，则提示：余额不足。  
double money = 19;  
if(money >= 90){  
    System.out.println("发红包成功了~");  
}else {  
    System.out.println("余额不足~~");  
}
```

5. if 第三种形式

if 第三种形式的代码格式，和执行流程如下图所示



if 第三种形式执行流程如下：
如果 条件表达式1 为true,就执行下面的代码1；
如果 条件表达式1 为false,就继续判断条件表达式2；

如果 条件表达式2 为true,就执行下面的语句体；
如果 条件表达式2 为false,就继续判断条件语句体3；

如果 条件表达式3 为true,就执行下面的语句体3
如果 条件表达式3 为false,就继续判断后面的表达式；

....
如果前面所有条件表达式判断都为false,就执行最后的else语句中的代码

```
// 需求3: 某个公司有一个绩效系统, 根据员工的打分输出对应的绩效级别。[0,60) D [60,80) C
[80,90) B [90,100] A
int score = 298;
if(score >= 0 && score < 60) {
    System.out.println("您的绩效级别是: D");
}else if(score >= 60 && score < 80){
    System.out.println("您的绩效级别是: C");
}else if(score >= 80 && score < 90){
    System.out.println("您的绩效级别是: B");
}else if(score >= 90 && score <= 100){
    System.out.println("您的绩效级别是: A");
}else {
    System.out.println("您录入的分数有毛病~~");
}
```

6. if 使用的几个常见问题

同学们在第一次写if 代码时, 经常一不小心会出现一些问题。下面把同学们可能出现的问题给大家看一看, 以后大家要避免出现这些问题。

- 第1个问题: if的()后面不能写分号; 否则if下面的语句与if无关

1661132888600

- 第2个问题: if后面的大括号, 如果只有一行代码, 大括号可以省略

1661132851560

7. if 分支小结

关于if分支结构的几种格式, 以及各种格式的执行流程, 还有if在什么场景下使用我们就讲完了。下面我们总结一下

- if分支有什么作用? 举几个应用场景?

- if作用: if分支可以根据条件, 选择执行某段程序
- if应用场景
 - 比如1: 测量用户体温, 发现体温高于37度就报警
 - 比如2: 发红包, 你的钱包余额是99元, 现在要发出90元
 - 比如3: 根据员工的绩效打分输出对应的绩效级别

- if分支的格式有几种, 执行流程是什么样的?

1661133510341

1.2 switch分支

学完if 分支之后, 接下来我们来学习分支结构的第二种形式——switch分支。

1. switch分支的执行流程

switch 分支的作用, 是通过比较值来决定执行哪条分支代码。先看一下switch分支的格式和执行流程

下面通过案例来演示一下

```
/*  
需求：做个电子备忘录，在控制台分别输出周一到周五的工作安排  
周一：埋头苦干，解决bug  
周二： 请求大牛程序员帮忙  
周三：今晚啤酒、龙虾、小烧烤  
周四：主动帮助新来的女程序解决bug  
周五：今晚吃鸡  
周六：与王婆介绍的小芳相亲  
周日：郁郁寡欢、准备上班。  
*/
```

```
String week = "周三";  
switch (week){  
    case "周一":  
        System.out.println("埋头苦干，解决bug");  
        break;  
    case "周二":  
        System.out.println("请求大牛程序员帮忙");  
        break;  
    case "周三":  
        System.out.println("今晚啤酒、龙虾、小烧烤");  
        break;  
    case "周四":  
        System.out.println("主动帮助新来的女程序解决bug");  
        break;  
    case "周五":  
        System.out.println("今晚吃鸡");  
        break;  
    case "周六":  
        System.out.println("与王婆介绍的小芳相亲");  
        break;  
    case "周日":  
        System.out.println("郁郁寡欢、准备上班");  
        break;  
    default:  
        System.out.println("您输入的星期信息不存在~~~");  
}
```

2. if、switch如何选择

学习完switch 分支之后，有同学可能会想，已经有了if分支，为什么还有switch分支呢？感觉上面的案例用if分支也能做啊？ 那我们在具体应用场景下如何选择呢？

如果单从功能上来讲，if 分支 的功能是更加强大的，switch分支能做的事情if 分支都能做。但是具体用哪一种分支形式，也是有一些使用原则的

- 如果是对一个范围进行判断，建议使用if分支结构
- 如果是与一个一个的值比较的时候，建议使用switch分支结构

1.3 switch 注意事项

各位同学，接下来我们学习switch的注意事项。同学们掌握这些注意事项之后，就可以避免入坑了，也可以应对一些面试笔试题。

- 1. 表达式类型只能是byte、short、int、char
JDK5开始支持枚举，JDK7开始支持String
不支持double、float、double
- 2. case给出的值不允许重复，且只能是字面量，不能是变量。
- 3. 正常使用switch的时候，不要忘记写break，否则会出现穿透现象。

1. 演示switch语句匹配的数据类型

各位同学，如果下图所示，可以自己分别用变量a、b放在switch语句中匹配试一试，如果遇到不支持的写法，IDEA会报错的。



2. 演示case后面的值，只能是字面量不能是变量

各位同学，也可以自己试试，下图箭头指向的位置只能写字面量，不能写变量



3. 演示case穿透现象

当switch语句中没有遇到break，就会直接穿透到下一个case语句执行，直到遇到break为止。

这种语法设计也是有它的用处的，当多个case语句想要执行同一段代码时，可以利用case穿透现象，提高代码复用性。

比如：我们下面程序中，想要让周二、周三、周四，都请大牛程序员来写代码。



二、循环结构

各位同学，接下来我们学习循环结构。循环结构可以控制一段代码重复执行。循环结构有for循环、while循环、do-while循环。

2.1 for循环——格式和流程

这里首先来学习for循环，同学们重点掌握for循环的书写格式，并理解for循环的执行流程。

1. for循环的格式和流程

为了让大家更直观的理解for循环的执行流程，我们直接来看具体的案例代码。

比如：我们想要在控制台打印输出3个HelloWorld

```
//需求: 打印3行Hello world
for(int i = 0; i < 3; i++) {
    System.out.println("Hello world");
}
```

如下图所示, 是按照下面的① ② ③ ④, ② ③ ④... 的顺序来执行的;

当②条件为true时, 再依次执行③④代码, 然后回到②继续判断

当②条件为false时, 就结束循环

1661137599188

具体执行的每一步可以看下面的图解

1661138616082

通过上面的案例演示, 最后我们再总结一下for循环的格式

```
//for循环格式:
for (初始化语句; 循环条件; 迭代语句) {
    循环体语句(重复执行的代码);
}
```

初始化语句: 一般是定义一个变量, 并给初始值

循环条件: 一般是一个关系表达式, 结果必须是true或者false

迭代语句: 用于对条件进行控制, 一般是自增或者自减

循环语句体: 需要重复执行的代码

2. for循环有哪些应用场景

通过上面的学习, 我们已经知道了for循环怎么编写, 并且也知道了它的执行流程。

那么具体在哪些实际场景下使用呢? **其实只要是重复做的事情, 都可以用循环语句来做**

比如: 在京东的网页上展示100台手机信息, 我们只需要把展示数据的代码写一份, 重复执行就可以了。

1661139301013

再比如: 再我们教学管理系统中, 有很多班级需要展示在页面上, 我们没必要每一个班级都写一份展示数据代码, 只需要写一份重复执行就可以了。

1661139453144

2.2 for循环案例1——求和

学完for循环的格式和流程之后, 我们再通过案例来巩固一下。通过这个案例, 主要是让同学们掌握一种使用程序来求和的思想。

```
//1. 掌握使用for循环批量产生数据。
for (int i = 1; i <= 100; i++) {
    System.out.println(i);
}
```

求和的思路分析：

- 1) 首先需要定义一个求和变量，一般命名为`sum`
 - 2) 再遍历得到所有需要求和的数据(1~100之间的所有整数)
 - 3) 让需要求和的数据和`sum`累加，
- 结果：所有数据累加完之后最终`sum`就是所有数据的和

```
//2.需求：求1~100中所有整数的和
int sum = 0;
//定义一个循环，先产生1-100，这100个数
for (int i = 1; i <= 100; i++) {
    //每产生一个数据，就把这个数和sum累加
    sum += i; //sum = sum + i;
}
System.out.println("1-100的数据和: " + sum);
```

分析上面代码的执行过程：

```
i=1时: sum=0+1; sum=1;
i=2时: sum=1+2; sum=3;
i=3时: sum=3+3; sum=6;
i=4时: sum=6+4; sum=10;
...
i=100时: sum+=99; sum=5050
```

2.2 for循环案例2——求奇数和

需求：求1~100之间奇数的和

1. 代码写法一

求奇数和的思路（只是求和的数据变成了奇数，思路和前面没有变化）

- 1) 首先需要定义一个求和变量，这里命名为`sum1`
 - 2) 再遍历得到所有需要求和的数据(1~100之间的所有奇数)
 - 3) 让需要求和的数据和`sum1`累加，
- 结果：所有数据累加完之后最终`sum1`就是所有数据的和

```
//1)定义一个变量用于求和
int sum1 = 0;
//2)定义一个循环产生1-100之间的奇数
for (int i = 1; i < 100; i+=2) {
    // i = 1 3 5 7 ...
    //3)让需要求和的数据和sum1累加，
    sum1 += i;
}
System.out.println("1-100之间的奇数和: " + sum1);
```

以上代码的执行流程分析

```
初始化sum1=0;

当i=1时: sum1+=1; sum1=1;
当i=3时: sum1+=3; sum1=4;
当i=5时: sum1+=5; sum1=9;
...
当i=99时: sum1+=99; sum1=2500
```

2. 代码写法二

求奇数和的思路（只是求和的数据变成了奇数，思路和前面没有变化）

- 1) 首先需要定义一个求和变量，这里命名为sum2
 - 2) 再遍历得到所有要求和的数据（1~100之间的所有整数）
 - 3) 在求和之前先对数据判断，如果是奇数，才和sum1累加；否则什么也不干
- 结果：所有数据累加完之后最终sum1就是所有数据的和

```
//1) 首先需要定义一个求和变量，这里命名为sum2
int sum2 = 0;
//2) 再遍历得到所有要求和的数据(1~100之间的所有整数)
for (int i = 1; i <= 100; i++) {
    //i = 1 2 3 4 5 6 ... 99 100
    //3) 在求和之前先对数据判断，如果是奇数，才和sum1累加；否则什么也不干
    if(i % 2 == 1){
        // i = 1 3 5 7 9 ... 99
        sum2 += i;
    }
}
System.out.println("1-100之间的奇数和: " + sum2);
```

for循环小结

今天关于for循环，我们学习这几个案例就够了，重点还是掌握for循环的执行流程。在以后，我们还会经常用到for循环，用多了，你就会越来越熟悉了。但是在具体场景下，还是需要具体问题具体分析。

2.3 while循环——格式和流程

各位同学，接下来我们学习第二种循环结构——while循环。

我们先来认识一下while循环长什么样子，然后按照格式写一个while循环的基础案例

 1661141338265

```
// 需求：打印5行Hello world
int i = 0;
while (i < 5) {
    // i = 0 1 2 3 4
    System.out.println("Hello world");
    i++;
}
```


代码的执行流程如下图所示：按照① ②③④ ②③④ ... 的流程执行

如果②步骤为true，才循环执行③④步骤

如果②步骤为false，则循环结束



1661141996444



1661141867092

for、while如何选择

学到这里，细心的同学可能会发现while循环和for循环的执行流程是一样的。那他们是不是可以通用呢？

- 从功能来说：能够用for循环做的，都能用while循环做。
- 使用规范上来说：知道循环几次，建议使用for；不知道循环几次建议使用while

2.3 while循环案例——折纸案例

各位同学，上一节我们已经学习了while循环的基本使用。下面我们通过一个案例再将while循环的使用巩固一下，主要目的还是想让大家知道什么使用while循环来完成需求。

案例需求如下：

需求：世界最高山峰珠穆朗玛峰高度是：8848.86米=8848860毫米，假如我有一张足够大的它的厚度是0.1毫米。请问：该纸张折叠多少次，可以折成珠穆朗玛峰的高度？

我们来分析一下该怎么做

分析：首先由于不知道折叠多少次，我们可以选择用while循环

1) 纸张的初始化厚度为0.1毫米，珠峰的高度为8848860毫米

```
double peakHeight = 8848860;
double paperThickness = 0.1;
```

2) 每次折叠纸张的厚度为原来的两倍，这是需要循环执行的

```
while(纸张厚度<8848860){
    纸张厚度*=2;
}
```

3) 需要求折叠的次数，可以用一个变量来记录折叠的次数

```
int 次数 = 0;
while(纸张厚度<8848860){
    纸张厚度*=2;
    次数++; //每次折叠次数累加
}
```

结果：等循环结束之后，打印记录次数的值，就是折叠多少次了。

按照上面分析的思路把代码写出来

```
// 1、定义变量记住珠穆朗玛峰的高度和纸张的高度。
double peakHeight = 8848860;
double paperThickness = 0.1;

// 3、定义一个变量count用于记住纸张折叠了多少次
int count = 0;
```

```
// 2、定义while循环控制纸张开始折叠
while (paperThickness < peakHeight) {
    // 把纸张进行折叠，把纸张的厚度变成原来的2倍。
    paperThickness = paperThickness * 2;
    count++;
}
System.out.println("需要折叠多少次: " + count);
System.out.println("最终纸张的厚度是: " + paperThickness);
```

2.4 do-while循环——格式和流程

各位同学，接下来我们学习循环结构的第三种格式——do-while循环。

我们先来认识一下while循环长什么样子，然后按照格式写一个while循环的基础案例。



如下图所示：do-while循环的执行流程，是按照① ②③④ ②③④... 的顺序执行的。

我们会发现，do-while循环的特点是先执行，再判断的。即使条件不成立，也会先执行一次。



下面我们把三种循环的区别给同学们总结一下

1. **for**循环 和 **while**循环（先判断后执行）；
do...while （先执行后判断）
2. **for**循环和**while**循环的执行流程是一模一样的，
功能上无区别，**for**能做的**while**也能做，反之亦然。
如果已知循环次数建议使用**for**循环，如果不清楚要循环多少次建议使用**while**循环。
- 3 **for**循环中控制循环的变量只在循环中使用
while循环中，控制循环的变量在循环后还可以继续使用

2.6 死循环

同学们在写代码时，可能一不小心把代码写成了死循环，所谓死循环就是停不下来的循环。

接下来，带着同学们认识几种死循环的写法。然后再说一下死循环有什么用。

```
//for死循环
for ( ; ; ){
    System.out.println("Hello world1");
}

//while死循环
while (true) {
    System.out.println("Hello world2");
}

//do-while死循环
do {
```

```
System.out.println("Hello world3");  
}while (true);
```

死循环有什么应用场景呢？

最典型的是可以用死循环来做服务器程序，比如百度的服务器程序就是一直在执行的，你随时都可以通过浏览器去访问百度。如果哪一天百度的服务器停止了运行，就意味着所有的人都永不了百度提供的服务了。

对于这样的应用我们目前了解一下就可以了。对于目前来说我们只要知道代码格式该怎么写，能达到什么效果就行。

2.8 循环嵌套

各位同学，接下来我们学习一种在实际工作中很常用的循环形式——循环嵌套。

所谓循环嵌套，就是一个循环中又包含另一个循环（就是同学们常说的，套娃^_^），下面我们通过案例代码演示一下。



循环嵌套执行流程：外部循环每循环一次，内部循环会全部执行完一轮。

```
i=0时: i<3为true; 进入循环  
    //j的取值从0到5,执行一轮,打印5次"我爱你"  
    for(int j = 1; j <= 5; j++) {  
        System.out.println("我爱你: " + i);  
    }  
    内层循环执行完之后,执行外层的i++; i的值1  
  
i=1时: i<3为true; 进入循环  
    //j的取值从0到5,又执行一轮,打印5次"我爱你"  
    for(int j = 1; j <= 5; j++) {  
        System.out.println("我爱你: " + i);  
    }  
    内层循环执行完之后,执行外层的i++; i的值2  
  
i=2时: i<3为true; 进入循环  
    //j的取值从0到5,再执行一轮,打印5次"我爱你"  
    for(int j = 1; j <= 5; j++) {  
        System.out.println("我爱你: " + i);  
    }  
    内层循环执行完之后,执行外层的i++; i的值3  
  
i=3时: i<3为false; 外层循环结束
```

理解问循环嵌套的执行流程之后，我们再写一个案例来巩固一下

需求：在控制台使用 * 打印出4行5列的矩形

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

//1)先写一个循环用来在一行中打印5个"*"

```

for (int j = 1; j <= 5; j++) {
    System.out.print("*"); // 不换行
}
System.out.println(); //换行

System.out.println("-----");

//2)再将第一步的代码套一层循环，执行4次，就可以打印4行
for (int i = 1; i <= 4; i++) {
    for (int j = 1; j <= 5; j++) {
        System.out.print("*"); // 不换行
    }
    System.out.println(); //换行
}

```

总结一下，对于嵌套循环重点理解这句话：**外部循环每循环一次，内部循环会全部执行完一轮。**

2.9 跳转语句 break、continue

前面我们学习了循环结构，在中间我们还接触了死循环的一些形式，那么我想要在循环过程中提前跳出循环怎么做呢？

这里就需要用到跳转语句，需要用到**break**和**continue**两个关键字。我们先来认识一下这两个关键字的作用

- break作用：跳出并结束当前所在循环的执行
- continue作用：结束本次循环，进入下一次循环

案例1：演示break的使用，提前终止循环的执行

```

// 1、break:跳出并结束当前所在循环的执行。
// 场景：假如你又有老婆了，你犯错了，你老婆罚你说：5句我爱你
// 说到第三句的时候心软了，让你别再说了。
for (int i = 1; i <= 5; i++) {
    System.out.println("我爱你: " + i);
    if(i == 3){
        // 说明已经说完了第三句了，心软了。
        break; // 跳出并结束当前所在循环的执行。
    }
}

```

案例2：演示continue的使用，结束循环中的一次，继续下一次循环

```

// 2、continue:跳出当前循环的当次执行，直接进入循环的下一次执行。
// 场景：假如你有老婆，你犯错了，你老婆罚你洗碗5天。
// 第三天的时候，你表现很好，第三天不用洗碗，但是不解恨，第四天还是要继续的。
for (int i = 1; i <= 5; i++) {
    if(i == 3) {
        // 已经到了第三天，第三天不用洗的。
        continue;
    }
    System.out.println("洗碗: " + i);
}

```

需要注意的是**break**和**continue**不是任何地方都可以使用的

1661146324812

1661146405314

2.10 循环结构总结

到这里关于循环结构的所有内容就都已经学习完了，我们再把几种循环结构在什么场景下使用，再总结一下。

1. 什么是流程控制

答：流程控制是用来控制程序的执行顺序的

2. 分支结构**if**和**switch**，如何选择？

答：**if**分支：一般用于对一个范围进行判断

switch分支：对一个一个值进行匹配

3. **for**循环和**while**循环、**do-while**如何循环

答：知道循环次数用**for**、不知道循环次数用**while**

想要先执行，再判断，用**do-while**

三、生成随机数

各位同学，接下来我们再学习一个新的知识——生成随机数。

生成随机数其实在很多场景下都很实用，比如，在课堂上可以写一个随机点名器点同学起来回答问题；再比如公司年会可以随机抽奖等。

3.1 如何产生一个随机数

生成随机数的功能，其实 Java 已经给我们提供了，在 JDK 中提供了一个类叫做 Random，我们只需要调用 Random 这个类提供的功能就可以了。

1661147570538

```
// 目标：掌握使用Random生成随机数的步骤。
// 1、导包。import java.util.Random; (idea会自动完成)
import java.util.Random;
public class RandomDemo1 {
    public static void main(String[] args) {
        // 2、创建一个Random对象，用于生成随机数。
        Random r = new Random();
        // 3、调用Random提供的功能：nextInt得到随机数。
        for (int i = 1; i <= 20; i++) {
            int data = r.nextInt(10); // 0 - 9
            System.out.println(data);
        }
    }
}
```

3.2 猜数字小游戏

各位同学，接下来我们通过一个案例把前面的流程控制、跳转语句、随机数综合运用一下；

如果能把这个案例写出来，说明你对今天的知识点掌握得挺好了。

需求：

随机生成一个1-100之间的数据，提示用户猜测，猜大提示过大，猜小提示过小，直到猜中结束游戏

分析：

1. 先随机生成一个1-100之间的数据。
谁可以帮你生成随机数啊？是不是要用到Random？
2. 定义一个死循环让用户可以一直猜测。
用户猜的数据从哪里来啊？是不是要用到Scanner？
3. 在死循环里，每次让用户录入的数据和随机数进行比较
如果比随机数大：提示猜大了
如果比随机数小：提示猜小了
如果和随机数相同：提示恭喜你猜中了

```
import java.util.Random;
import java.util.Scanner;

public class RandomTest2 {
    public static void main(String[] args) {
        // 1、随机产生一个1-100之间的数据，做为中奖号码。
        Random r = new Random();
        int luckNumber = r.nextInt(100) + 1;

        // 2、定义一个死循环，让用户不断的猜测数据
        Scanner sc = new Scanner(System.in);
        while (true) {
            // 提示用户猜测
            System.out.println("请您输入您猜测的数据：");
            int guessNumber = sc.nextInt();

            // 3、判断用户猜测的数字与幸运号码的大小情况
            if(guessNumber > luckNumber){
                System.out.println("您猜测的数字过大~~");
            }else if(guessNumber < luckNumber){
                System.out.println("您猜测的数字过小~~");
            }else {
                System.out.println("恭喜您，猜测成功了，可以买单了~~");
                break; // 结束死循环
            }
        }
    }
}
```