



河南中医药大学信息技术学院（智能医疗行业学院）智能医学工程专业《计算机程序设计》课程

第08章：流程控制

黄子杰

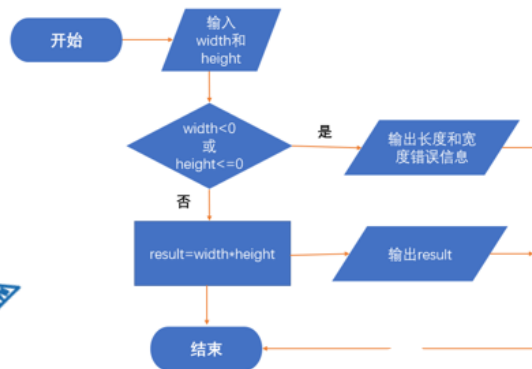
河南中医药大学信息技术学院（智能医疗行业学院）与河南方和信息科技有限公司 联合建设
河南中医药大学信息技术学院智能医疗教研室

<https://aitcm.hactcm.edu.cn>

2025/2/16

本章概要

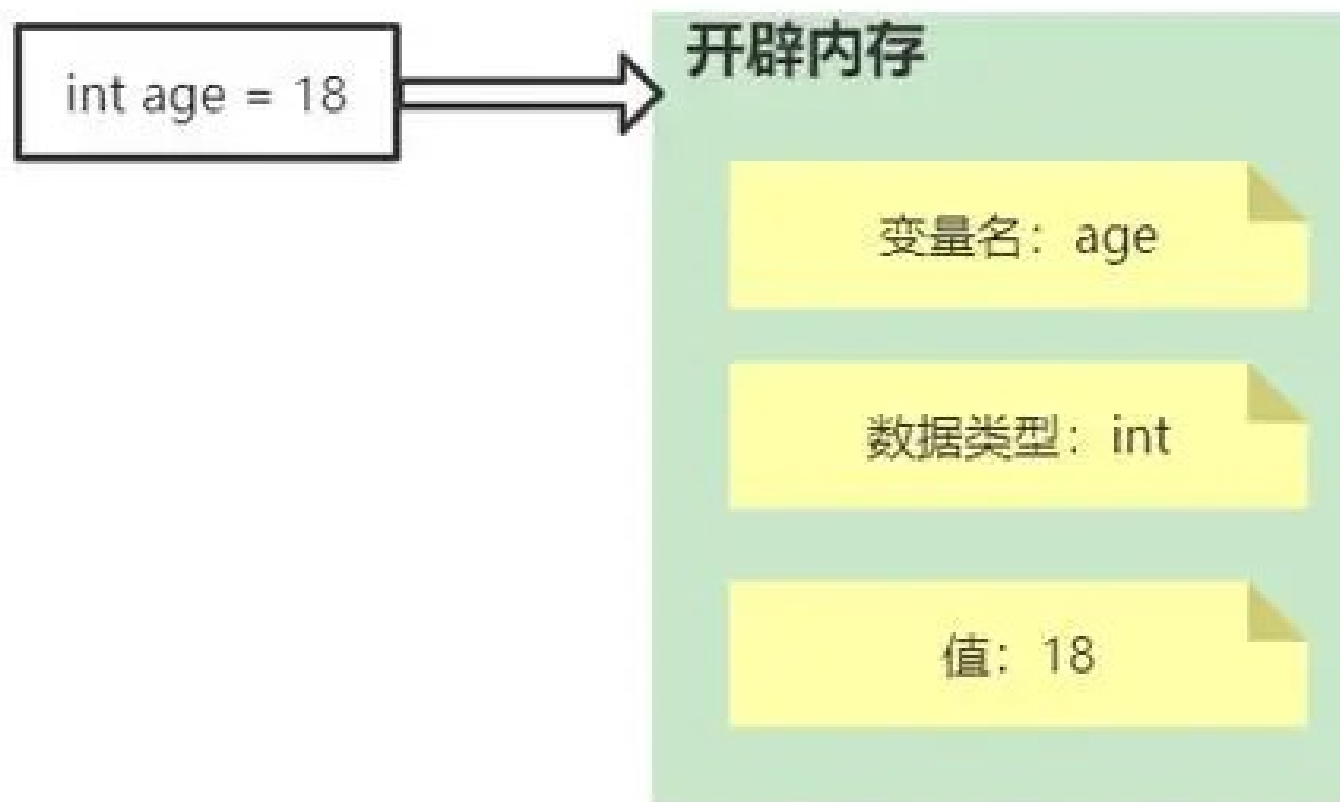
概述：在Java中，流程控制主要包括顺序、条件（分支）、循环三大结构。

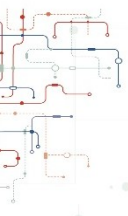


看的懂!



8.1 变量





8.1.1 什么是变量

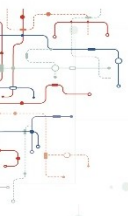
- 概述：

在Java编程中，**变量是存储数据的基本单元**，理解变量的概念、类型和使用方法是编写高效代码的基础。本章将详细介绍Java中的变量，包括变量的定义、类型、作用域和常见用法，并通过代码示例来帮助理解这些概念。

- 详细说明：

数据存储：变量用于临时存储程序中的数据。

类型定义：每个变量都有特定的数据类型（如int、double、String），定义了该变量可以存储的数据类型。



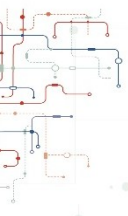
8.1.1 什么是变量

- 生活举例：

变量：

就像一个储物箱，你可以在里面放不同类型的物品（数据），但每个储物箱只能放一种类型的物品。

变量相当于内存中一个数据存储空间的表示，可以把变量看作门牌号，通过门牌号我们可找到房间，而通过变量我们可以找到值。



8.1.2 为什么需要变量

- 概述：

变量使得程序可以灵活地处理数据，通过变量，程序能够存储和操作用户输入、运算结果等动态数据。

- 详细说明：

灵活性：变量使程序可以动态处理数据，而不是依赖于固定的常量。

重复使用：通过变量，程序可以对数据进行多次操作，而不需要重新定义或硬编码。

8.1.2 为什么需要变量

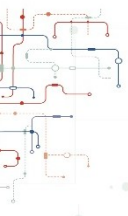
- 生活举例：

变量的必要性：就像在数学计算中使用 x 代表一个未知数，使得我们可以处理不同的值。

- 代码示例：



eg8.1.2.txt



8.1.3 变量的声明和赋值

- 在Java中，变量的定义和声明遵循以下格式：

`type variableName;`

详细说明：

声明：数据类型 变量名;例如：int number;

赋值：变量名 = 值;例如：number = 10;

初始化：声明变量时可以同时赋值，称为初始化。

8.1.3 变量的声明和赋值

1.1 变量类型

Java是一种强类型语言，每个变量在使用前必须声明其类型。常见的变量类型包括：

基本数据类型：byte, short, int, long, float, double, char, boolean

引用数据类型：类、接口、数组

1.2 变量声明和初始化

变量在声明后，可以进行初始化（即赋值）。变量可以在声明时初始化，也可以在之后的代码中赋值。

- 代码示例：



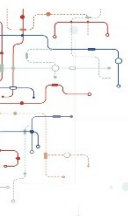
eg8.1.3.txt

8.1.3 变量的声明和赋值

- 生活举例：

声明和赋值：就像先建一个空的储物箱（声明），然后往里面放东西（赋值）。





8.1.4 变量命名规范

- 概述：

在Java中，变量的命名需要遵循一定的规范，以提高代码的可读性和维护性。

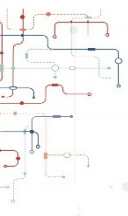
- 详细说明：

- 规则：

变量名必须以字母、下划线或美元符号开头。

变量名区分大小写，不能使用Java的关键字。

变量名应该具有描述性，以便于理解其用途。



8.1.4 变量命名规范

- 1. 变量命名规则
- 1.1 可以使用的字符

变量名可以包含字母、数字、美元符号 \$ 和下划线 _。

变量名必须以字母、美元符号 \$ 或下划线 _ 开始，不能以数字开头。

- 1.2 大小写敏感

Java是区分大小写的语言，因此变量名中的大小写字母是不同的。

- 1.3 关键字的限制

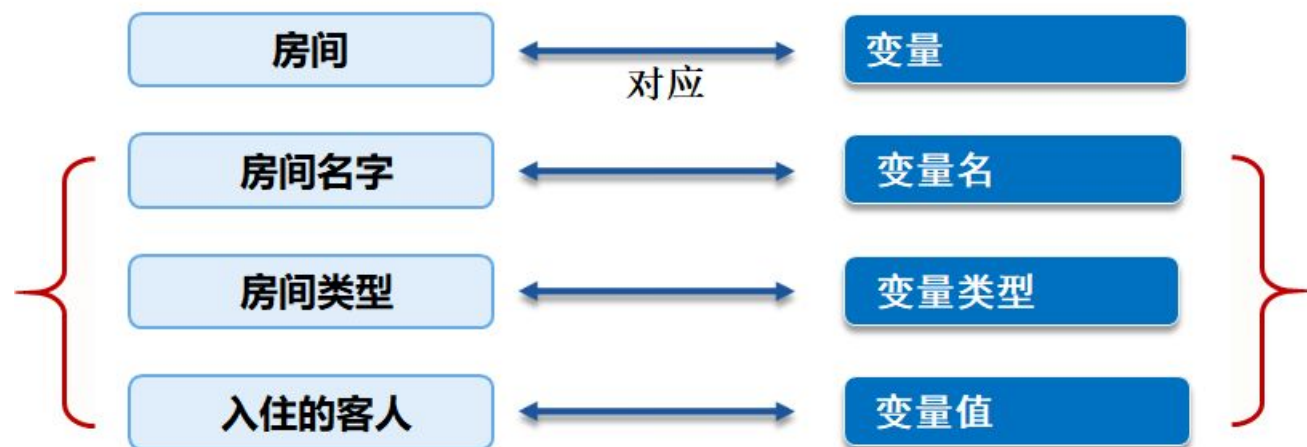
不能使用Java的保留关键字作为变量名。例如，class、int、void等都是Java的关键字，不能用作变量名。

- 1.4 规范推荐

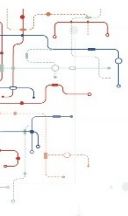
变量名使用驼峰命名法（Camel Case）：除第一个单词外，其他单词的首字母大写，例如myVariableName。

变量名应具有描述性，能够清晰地表达变量的用途和含义。

8.1.5 变量应用实例

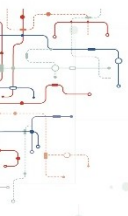


https://blog.csdn.net/GYL_Fight



8.1.5 变量应用实例

```
int a, b, c; // 声明三个int型整数：a、b、c。  
int d = 3, e, f = 5; // d声明三个整数并赋予初值。  
byte z = 22; // 声明并初始化z。  
double pi = 3.14159; // 声明了pi。  
char x = 'x'; // 变量x的值是字符'x'。
```



8.2 标识符与关键字

标识符与关键字的区别

标识符：

用于命名变量、方法、类、接口等。

可以由字母、数字、下划线 `_` 和美元符号 `$` 组成。

不能以数字开头，大小写敏感，没有长度限制。

例如：myVariable, sum, _count, \$amount

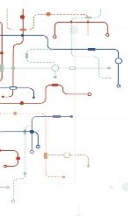
关键字：

是Java语言预定义的具有特殊含义的单词。

不能用作标识符。

全部为小写。

例如：class, public, static, if, else



8.2.1 Java 标识符命名规则

- 在Java中，标识符是用来给类、方法、变量和接口命名的。它必须遵循以下规则：

标识符必须以字母、下划线(_)或美元符号(\$)开头。

标识符可以包含字母、数字、下划线(_)或美元符号(\$)。

标识符不能包含空格、@、#、+、-、= 或其他特殊字符。

不能使用Java关键字作为标识符。



8.2.2 关键字

String 类

1、访问控制

- | | |
|-------------|-------|
| ● 关键字 | 说明 |
| ● private | 私有访问 |
| ● protected | 受保护访问 |
| ● public | 公共访问 |



8.2.2 关键字

String 类

2、类、方法和变量修饰符

- | | |
|----------------|-----------|
| ● 关键字 | 说明 |
| ● abstract | 抽象类或方法 |
| ● class | 类 |
| ● extends | 扩展（继承） |
| ● final | 最终类、方法或变量 |
| ● implements | 实现接口 |
| ● interface | 接口 |
| ● native | 本地方法 |
| ● new | 实例化对象 |
| ● static | 静态方法或变量 |
| ● strictfp | 严格浮点 |
| ● synchronized | 同步方法或块 |
| ● transient | 瞬态变量 |
| ● volatile | 易变变量 |
| ● enum | 枚举类型 |



8.2.2 关键字

3、程序控制

关键字	说明
• break	中断循环
• continue	继续循环
• return	返回值或退出
• do	执行循环
• while	循环
• if	条件语句
• else	条件语句
• for	循环
• isinstance	类型检查
• switch	选择语句
• case	选择分支
• default	默认分支
• assert	断言



8.2.2 关键字

4、错误处理

- | ● 关键字 | 说明 |
|-----------|-----------|
| ● try | 捕获异常块 |
| ● catch | 捕获异常 |
| ● throw | 抛出异常 |
| ● throws | 抛出异常声明 |
| ● finally | 异常处理完毕后执行 |

5、包相关

- | 关键字 | 说明 |
|---------|-----|
| import | 导入包 |
| package | 包声明 |



8.2.2 关键字

6、基本类型

- | ● 关键字 | 说明 |
|-----------|---------|
| ● boolean | 布尔类型 |
| ● byte | 字节类型 |
| ● char | 字符类型 |
| ● double | 双精度浮点类型 |
| ● float | 单精度浮点类型 |
| ● int | 整数类型 |
| ● long | 长整数类型 |
| ● short | 短整数类型 |



8.2.2 关键字

7、变量引用

- | ● 关键字 | 说明 |
|---------|--------|
| ● super | 父类引用 |
| ● this | 当前实例引用 |
| ● void | 无返回值 |

8、保留字

- | ● 关键字 | 说明 |
|---------|--------|
| ● goto | 保留，未使用 |
| ● const | 保留，未使用 |



8.2.2 关键字

- 关键字的作用：

在编写代码时，关键字具有特殊的含义，用来定义数据类型、控制结构等。

- 生活举例：

关键字：就像交通标志，规定了每个标志的特定含义，不能随意更改。



8.3 运算符

- 概述：

在Java中，运算符可以分为算术运算符、关系运算符、逻辑运算符、位运算符、赋值运算符和三目运算符等。



8.3.1 算术运算符

- 概述：

算术运算符用于执行基本的数学运算，如加、减、乘、除和取余。

- 常见运算符：

+: 加法

-: 减法

*: 乘法

/: 除法

?: 取余运算

8.3.1 算术运算符

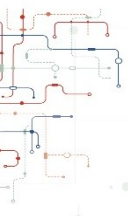
- 生活举例：

算术运算符：就像生活中的基本数学运算，比如购买物品时计算总价、找零等。

- 代码示例：

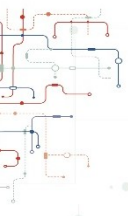


eg8.3.1.txt



8.3.1 算术运算符

- 1、加法运算符 (+)：可以用于两个数值类型的变量相加。
- 2、减法运算符 (-)：可以用于两个数值类型的变量相减。
- 3、乘法运算符 (*)：可以用于两个数值类型的变量相乘。
- 4、除法运算符 (/)：可以用于两个数值类型的变量相除。
- 5、取余运算符 (%)：可以用于获取两个数值类型的变量相除后的余数。
- 注意，在使用除法运算符 (/) 时，如果除数为0，会导致程序抛出“ArithmeticException”异常。因此，在使用除法运算符时，应该先判断除数是否为0。



8.3.2 赋值操作符

- 概述：

Java赋值运算符用于将值赋给变量。在Java中，赋值运算符使用“=”符号表示。

- 详细说明：

赋值操作： 可以将右侧的值或表达式结果赋给左侧的变量。

组合赋值： 如+=、-=、*=、/=、%=等，用于将运算结果赋值给变量。



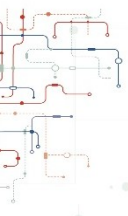
8.3.2 赋值操作符

符号	作用	说明
+=	加法赋值	$a+=b$ 等价于 $a=a+b$
-=	减法赋值	$a-=b$ 等价于 $a=a-b$
=	乘法赋值	$a=b$ 等价于 $a=a*b$
/=	除法赋值	$a/=b$ 等价于 $a=a/b$
%=	取余赋值	$a\%=b$ 等价于 $a=a\%b$

- 代码示例：



eg8.3.2.txt



8.3.3 关系操作符

- 概述：

关系运算符用于比较两个值之间的关系,通常返回布尔值（真或假）。这些运算符在条件语句、循环和其他需要比较的场景中经常使用。

- 详细说明：

常见关系操作符：

==：判断相等

!=：判断不相等

>：大于

<：小于

>=：大于或等于

<=：小于或等于

8.3.3 关系操作符

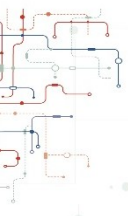
符号	说明
== (等于)	检查两个值是否相等
!= (不等于)	检查两个值是否不相等
> (大于)	检查左边的值是否大于右边的值
< (小于)	检查左边的值是否小于右边的值
>= (大于等于)	检查左边的值是否大于或等于右边的值
<= (小于等于)	检查左边的值是否大于或等于右边的值

- 代码示例：



eg8.3.3.txt





8.3.4 逻辑操作符

- 概述：

逻辑运算符用于组合或修改布尔表达式，通常用于控制程序流程和决策making。这些运算符处理布尔值（真/假）并返回布尔结果。

- 详细说明：

常见逻辑操作符：

&&（逻辑与）：所有条件为真，结果才为真。

||（逻辑或）：任意一个条件为真，结果即为真。

！（逻辑非）：将布尔值反转，true变为false，false变为true。

8.3.4 逻辑操作符

符号	作用	说明
&	逻辑与	当所有操作数都为真时返回真
	逻辑或	当至少一个操作数为真时返回真
^	逻辑异或	当操作数不同时返回真，相同时返回假
!	逻辑非	反转操作数的布尔值

- 代码示例：



eg8.3.4.txt





8.4 选择结构

- 什么是选择结构？

选择结构是当给定判断条件时，根据条件来判断是否满足某些条件，如果满足实行提前规定好的一段代码，反之执行另一代码的一种结构体。

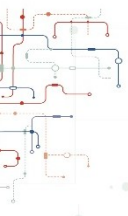
```
if(条件){
```

```
.....
```

```
}else{
```

```
.....
```

```
}
```



8.4.1 顺序语句

- 概述：

顺序语句是程序中最简单的控制结构，程序按顺序逐行执行代码。

- 详细说明：

顺序执行：代码从上到下依次执行，不存在跳转或分支。

常见场景：适用于程序的初始部分或不需要判断和循环的简单逻辑。

8.4.1 顺序语句

- 生活举例：

顺序语句：就像做饭时按照食谱的步骤逐步进行，从准备食材到烹饪。

- 代码示例：

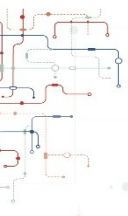


eg8.4.1.txt



8.4.2 选择条件语句 if, if else

- 1、if单支选择结构
- 2、if-else双分支选择结构
- 3、if-else-if多重选择结构
- 4、if嵌套



1.if语句

- 语法结构：

```
if(条件){  
    //代码块  
}
```

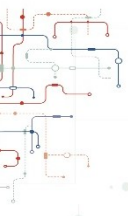
- 执行规律：

如果条件为true，执行{}里的代码块，执行完代码块之后，继续执行{}后面的代码

如果条件为false，则跳过if选择结构，执行{}后面的代码

- 注意事项：

条件不管写的多么的简单还是多么的复杂，最终结果应该是布尔值，要么为true要么为false



2.if else语句

- 语法结构：

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

- 执行规律：

如果条件为true，执行代码块1，执行完代码块1后，结束整个if-else结构，继续往后执行if-else结构后面的代码

如果条件为false，执行代码块2，执行完代码块2后，结束整个if-else结构，继续往后执行if-else结构后面的代码

- 注意事项：

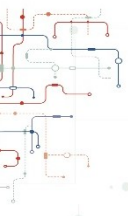
条件都为布尔值



3.if else if语句

- 语法结构:

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



3.if else if语句

- 执行规律:

判断条件1，如果条件1为true，则执行代码块1，执行完代码块1，结束整个if-else-if结构，执行if-else-if结构后面的代码

如果条件1为false，则继续往下判断条件2，如果条件2为true，则执行代码块2，执行完代码块2，结束整个if-else-if结构，执行if-else-if结构后面的代码

如果条件2为false，则继续往下判断条件3，如果条件3为true，则执行代码块3，执行完代码块3，结束整个if-else-if结构，执行if-else-if结构后面的代码

如果条件3为false，则继续.....

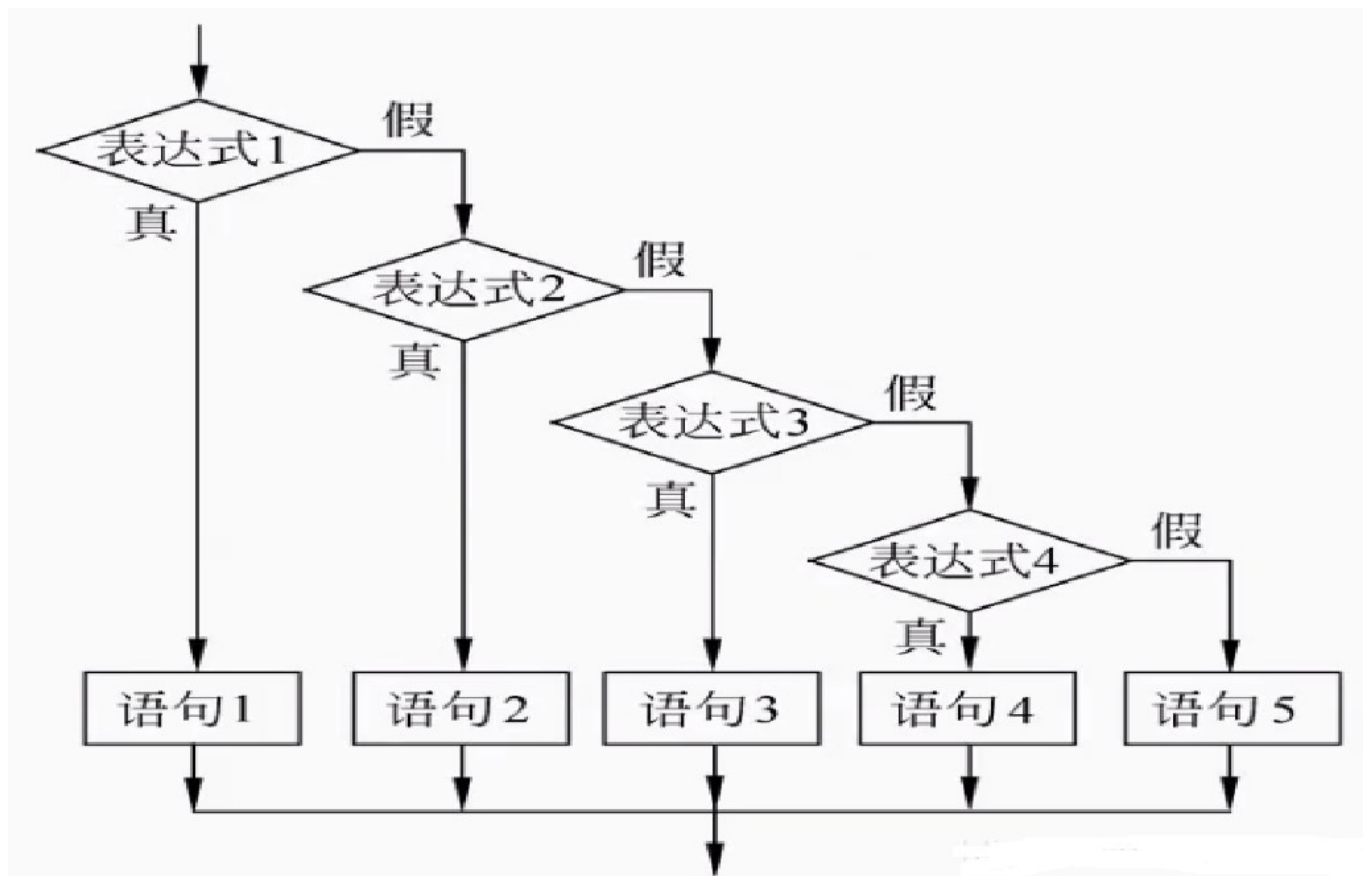
如果所有条件都不满足，则执行else{}里的代码块n

- 注意事项:

- (1) 所有的条件都为布尔值
- (2) else if可以有很多个
- (3) else是可以省略不写，当所有的条件都不满足，并且没有else的时候，则整个if-else-if结构里代码的都不会执行

4.if嵌套

if嵌套就是在一个if选择结构中，套用一个甚至多个完整的if结构，适用于一般处理条件比较复杂的类型。



```
int num = 4;
if(num == 1){
    System.out.println("num = 1");
}else if(num == 2){
    System.out.println("num = 2");
}else if(num == 3){
    System.out.println("num = 3");
}else if(num == 4){
    System.out.println("num = 4");
}else if(num == 5){
    System.out.println("num = 5");
}else{
    System.out.println("other");
}
```



8.4.3 switch结构

- 概述：

当需要对选项进行等值判断时，使用 switch 语句更加简洁明了。

- 问题：

根据考试分数，给予前四名不同的奖品。第一名，奖励笔记本一台；第二名，奖励 IPAD 2 一个；第三名，奖励移动电源一个；最后一名奖励 U 盘一个。

答案：

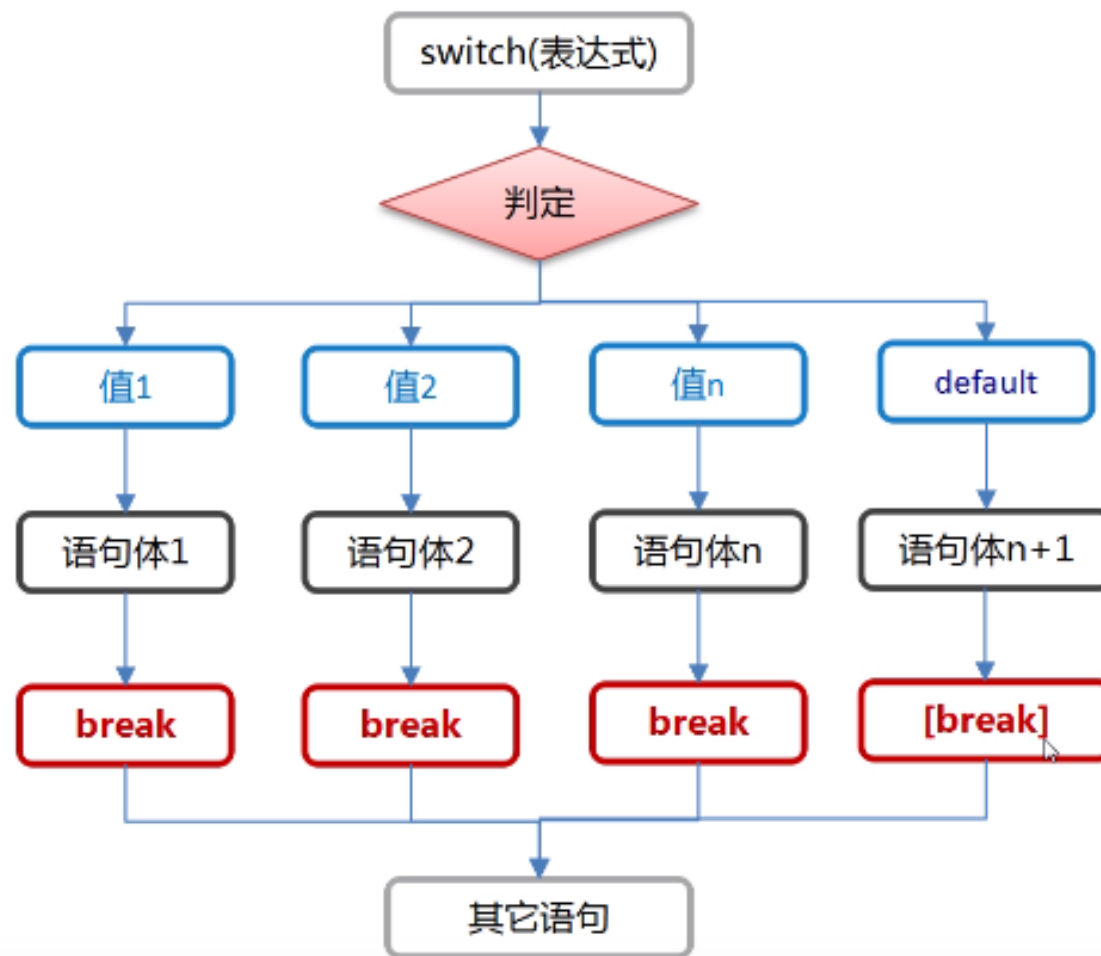


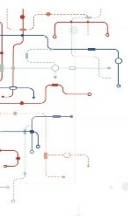
eg-switch.txt

8.4.3 switch结构

switch语句的执行流程

```
int num = 4;
switch (num){
    case 1:
        System.out.println("num = 1");
        break;
    case 2:
        System.out.println("num = 2");
        break;
    case 3:
        System.out.println("num = 3");
        break;
    case 4:
        System.out.println("num = 4");
        break;
    case 5:
        System.out.println("num = 5");
        break;
    default:
        System.out.println("other");
        break;
}
```





8.4.4 Switch和多重if结构比较

- if语句、if else if语句和switch case语句都属于流程控制语句。
- 在只需要判断一个条件的时候，自然是使用if语句方便有效；但是当判断条件很多的时候，我们可以使用多个if语句或者if...else if语句或者switch case语句。
- 对于这三者的选择，下面将做一些具体分析；对于后两者的选择，又将涉及到程序执行效率的问题。
- if...else if语句和多个if语句的区别还是很大的，**if...else if在任何一个环节满足条件的时候就会终止判断**，只处理一个满足条件的情况；而对于多个if语句，将会对每一个判断条件进行判断，自然而然会导致程序的执行效率降低。在多个判断条件的情况下，使用if...else if语句相对于使用多个if语句而言，可以减少程序的判断次数，提高效率。
- 在多个判断条件的情况下，不仅可以使if...else if语句，还可以使用switch case语句。一般情况下，当判断条件较多的情况下，使用switch case语句的效率会高于使用if...else if语句。switch...case与if...else if的根本区别在于，switch...case会生成一个跳转表来指示实际的case分支的地址，而这个跳转表的索引号与switch变量的值是相等的。从而，switch...case不用像if...else if那样遍历条件分支直到命中条件，而只需**访问对应索引号的表项从而到达定位分支的目的**。所以从效率上来说由于if...else if的遍历性，代码执行效率是不高的。
- 总结

if ...else逻辑判断时，每条if语句都独立需要加载，都要走一遍判断。这就是耗时的机制问题了。

switch...case 根据一个值进行多路分支，只做一次计算，然后将表达式的值与每个case的值比较，进而选择哪一个case语句块。

switch只能处理case为常量的情况，对不是常量的情况是无能为力的。例如 if (a > 1 && a < 100)属于关系逻辑，是无法使用switch...case来处理的。

优先级排序（效率最高的）：swachit、if else、多重if

8.4.4 Switch和多重if结构比较

- 比较switch和多重if 选择结构：

相同点：都是用来处理多分支条件的结构。

不同点：

switch选择结构：只能处理等值条件判断的情况

多重if选择结构：没有switch选择结构的限制，特别适合某个变量处于某个连续区间时的情况

- 代码示例：



eg8.4.4.txt



8.5 循环结构

- Java的循环结构主要有三种：

for循环

while循环

do-while循环

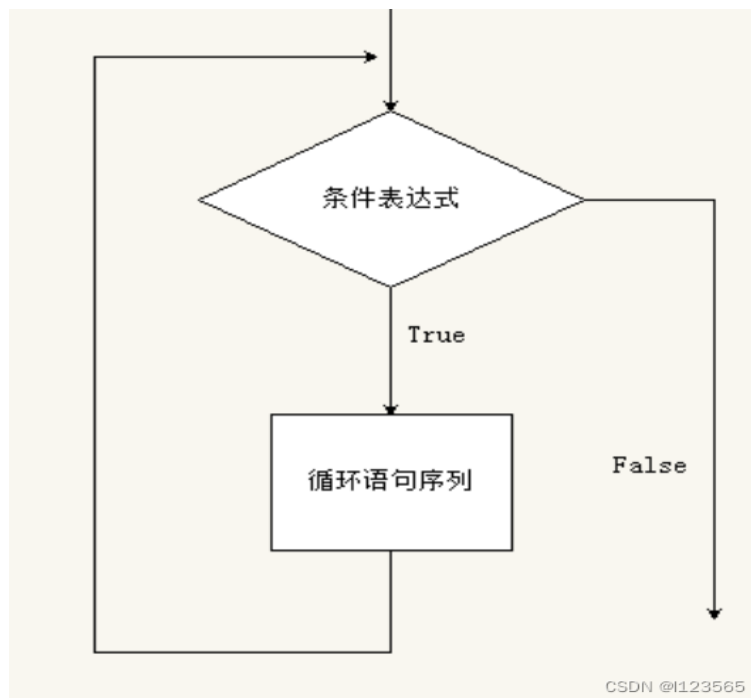
8.5.1 while 循环

- 概述：

while循环用于在条件为真时重复执行一段代码，适合处理不确定循环次数的情况。

- 基本语法：

```
while(布尔表达式) {  
    // 循环体  
}
```





8.5.1 while 循环

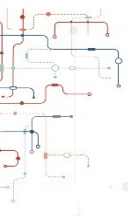
- while循环次数最少是0次，也就是一次都不运行。

- 用文字来描述它的运行过程就是：

先判断布尔的值，若结果为true则执行循环语句序列，再判断布尔的值，若结果为true则再执行循环语句序列，这样反复执行，直到布尔的值为false时，while循环结束。

- 生活举例：

while循环：就像不停地问小孩“你饿吗？”直到他回答“是”。



8.5.1 while 循环

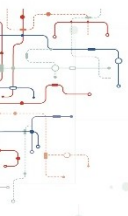
- 问题：

用while循环计算 $1+2+3+4+\dots+50+51+\dots+98+99+100$ 。

- 答案：



eg-while循环计算1-100的和.txt



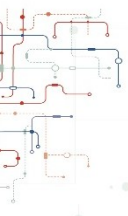
8.5.2 do...while 循环

- 概述：

do-while循环与while循环类似，但是它至少会执行一次循环体，然后再判断布尔表达式的结果，也就是说，就算布尔值为false(条件不满足)也会先执行一次。

- 基本语法：

```
do {  
    // 循环体  
} while(布尔表达式);
```



8.5.2 do...while 循环

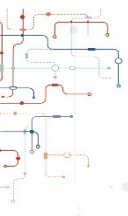
- 问题：

用do...while循环计算 $1+2+3+4+\cdot\cdot\cdot+100$ 。

- 答案：



eg-do-while循环计算1-100的和.txt



8.5.2 do...while 循环

- 生活举例：

do...while 循环：就像每次吃饭前先吃一口，尝一尝味道，然后决定是否继续吃。

- 补充

依旧是这道算术题，两种循环得出的结果一模一样，这就引出一个问题：

do-while循环与while循环是否可以互相转换？是可以的，但有一个条件，那就是while执行次数必须大于或等于1次，这里指的互换是指循环体内的语句基本一样，只是循环语句互换了而已。



8.5.3 for 循环

- 概述：

for循环通常用于已知循环次数的情况，使用该循环时，测试是否满足某个条件，如果满足条件，则进入下一次循环，否则，退出该循环。

- 详细说明：

基本结构：

for循环包含初始化语句、条件表达式和更新语句，分别在循环开始时、每次迭代前、和每次迭代后执行。

适用场景：

适合遍历数组、集合或执行固定次数的操作。

for循环的语法格式

循环开始前运行
次，可做赋值操
作

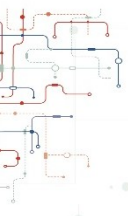
返回布尔值，
控制循环是否继续
执行

通常是赋值
表达式，在完
成一次循环
之后执行

```
for(初始表达式 1; 布尔表达式 2; 表达式 3){  
    //语句  
}
```

CSDN @1123565

```
for (初始化语句; 循环条件; 迭代语句) {  
    // 要重复执行的代码  
}
```



8.5.3 for 循环

- 生活举例：

for 循环：就像一周的计划，按顺序从星期一到星期天逐天进行。

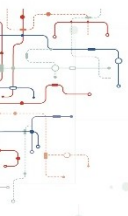
- 问题：

使用 for 循环打印从 1 到 10 的数字：

- 答案：



eg-for循环.txt



8.5.4 增强 for 循环

- 概述：

增强型 for 循环是 Java 5 中引入的一种简化版的 for 循环，专门用于遍历数组或集合。它的语法更简洁，但功能相对有限。

- 基本语法：

```
for (声明语句 : 表达式) {  
    // 要重复执行的代码  
}
```

8.5.4 增强 for 循环

- 问题:

使用增强型 for 循环遍历一个整数数组并打印每个元素。

- 答案:



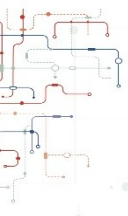
eg-增强for循环.txt



8.5 循环结构

- 总结

Java 的循环结构是编程中不可或缺的一部分，它们使得重复执行代码变得简单高效。了解和掌握这些循环结构对于编写高效的 Java 程序至关重要。每种循环结构都有其适用场景，选择合适的循环结构可以使代码更加清晰和易于维护。



8.5.5 break 语句

- 概述：

break语句用于立即终止最内层的循环或switch语句。

它是一种控制流语句，能够在满足特定条件时跳出循环或结束switch块的执行。

BREAK

1、在循环中使用

- 概述：

break语句可以用于for、while和do-while循环中。当在循环中遇到break语句时，循环会立即终止，程序控制流将跳转到循环之后的第一条语句。

- 问题：

随机生成 1-100 之间的整数，当生成出 66 这个数字时循环停止。

- 答案：



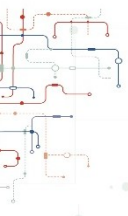
eg-循环break.txt



2、在 switch 中使用

- 概述：

break语句在switch语句中非常常见，用于终止特定的case块。如果没有break语句，程序会继续执行后续的case块（即使这些块的条件不满足），直到遇到break语句或switch语句结束为止。这种行为称为“贯穿”或“fall-through”。



8.5.6 continue 语句

- 概述：

在Java中，continue是一种流程控制语句，用于跳过当前循环中的剩余代码并开始下一次循环迭代。当程序执行到continue语句时，它会立即跳到循环的下一次迭代，而不执行剩余的循环体代码。

CONTiNUE..

8.5.6 continue 语句

- 问题：

使用for循环和continue语句，打印1，2，4，5

- 答案：

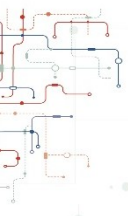


eg-循环continue.txt



本章重点

- 选择条件语句if,if else
- switch结构
- Switch和多重if结构比较
- 循环结构
- while 循环
- do…while 循环
- for 循环
- 增强 for 循环
- break语句
- continue语句



本章作业

- 作业一：标识符命名与基础语法练习
- 目的：加深理解Java中的标识符命名规则、关键字使用以及基本运算符的应用。
- 要求：
 - 1 编写一个Java程序，其中包含至少5个自定义变量，这些变量的命名需符合Java的标识符命名规则，并包含不同的数据类型（如int, double, boolean, String等）。
 - 2 在该程序中，使用算术运算符（+, -, *, /）和赋值操作符（=）对这些变量进行操作，展示基本的算术运算。
 - 3 使用关系操作符（>, <, ==, !=, >=, <=）和逻辑操作符（&&, ||, !）编写条件表达式，用于判断变量之间的关系，并输出判断结果。

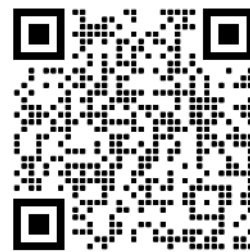


本章作业

- 作业二：控制结构实践
- 目的：掌握Java中的选择结构和循环结构，包括if-else语句、switch语句以及while、do-while、for循环的使用。
- 要求：
 - 1 编写一个程序，使用if-else语句判断用户的年龄是否成年（假设成年年龄为18岁），并输出相应的信息。
 - 2 扩展上述程序，增加使用switch语句的功能，根据用户的成绩等级（A, B, C, D, E）输出不同的评价。成绩等级通过用户输入的成绩范围确定（例如，90-100为A，80-89为B，以此类推）。
 - 3 编写一个使用while循环的程序，计算并输出1到100之间所有偶数的和。
 - 4 编写一个使用for循环的程序，打印一个9*9乘法表。
 - 5 尝试使用增强for循环（也称为for-each循环）遍历并打印一个字符串数组中的所有元素。

信创智能医疗系统研发课程体系

河南中医药大学信息技术学院（智能医疗行业学院）



河南中医药大学信息技术学院（智能医疗行业学院）智能医疗教研室

河南中医药大学医疗健康信息工程技术研究所