基础[2]

一、基本内容

目录

1、标识符

标识符

1.1、定义

- 给包,类,方法,变量起名字的符号。

1.2、组成规则

- 标识符由字母、数字、下划线、美元符号组成。

1.3、命名原则

> 见名知意

- 包名：全部小写，多级包用.隔开。

```
com.yourtest
```

- 类、接口：一个单词首字母大写，多个单词每个单词的首字母大写。

```
Student,Car,HelloWorld
```

- 方法和变量：一个单词首字母小写，多个单词从第二个单词开始每个单词的首字母大写。

```
age,maxAge,show(),getAge()
```

- 常量：如果是一个单词,所有字母大写，如果是多个单词,所有的单词大写,用下划线区分每个单词。

```
DATE,MAX_AGE
```

- 项目名：全部用小写字母，多个单词之间用横杆-分割。

```
demo，spring-boot\n
```

1.4、注意事项

> - 不能以数字开头
> - 不能是Java中的关键字
> - Java标识符大小写敏感，长度无限制
> - 标识符不能包含空格

2、关键字

关键字

关键字	关键字	关键字	关键字	关键字
private	protected	public	abstract	class
enum	extends	final	implements	interface
native	new	static	strictfp	synchronized
transient	volatile	break	case	continue
default	do	else	for	if
instanceof	return	switch	while	assrt
catch	finally	throw	throws	try
import	package	boolean	byte	char
double	float	int	long	short
super	this	void	goto	const
default	exports	module	requires	var

3、分隔符

分隔符

3.1、定义

> 空格、逗号、分号以及行结束符都被称为分隔符，规定任意两个相邻标识符、数字、保留字或语句之间必须至少有一个分隔符，以便程序编译时能够识别。

3.2、分类

- `;`（分号） 用来终止一个语句
- `{}` （花括号、大括号） 用来包括自动初始化的数组的值，也用来定义程序块、类、方法以及局部范围
- `[]` （方括号、中括号） 用来声明数组的类型，也用来表示撤消对数组值的引用\n
- `,` （逗号）在变量声明中，区分变量说明的各个变量。在for控制语句中，用来将圆括号内的语句连接起来
- . （原点） 用来将软件包的名字与它的子包或类分隔。也用来将引用变量与变量或方法分隔\n
- `()` （圆括号） 在定义和调用方法时用来容纳参数表。在控制语句或强制类型转换组成的表达式中用来表示执行或计算的优先权

3.3、注意事项

> 必须都是半角下的英文符号。

4、访问修饰符

访问修饰符

4.1、定义

> `Java`中，可以使用访问**修饰符来保护对类、变量、方法和构造方法的访问**。Java 支持 4 种不同的访问权限。

4.2、分类

- `private`：**在同一类内可见**。使用对象：变量、方法。 **注意：不能修饰类（外部类）**
- `default` (即缺省，什么也不写，不使用任何关键字)：**在同一包内可见**，不使用任何修饰符。使用对象：类、接口、变量、方法。
- `protected`：**对同一包内的类和所有子类可见**。使用对象：变量、方法。 注意：不能修饰类（外部类）。
- `public`：**对所有类可见**。使用对象：类、接口、变量、方法

4.3、图解释

修饰符	当前类	同包	子类	其他包
private	✔	✘	✘	✘
default	✔	✔	✘	✘
protected	✔	✔	✔	✘
public	✔	✔	✔	✔

二、运算符

目录

1、定义

定义

> 运算符**指明对操作数的运算方式**

2、分类及用法

分类及用法

2.1、算术运算符

```
+ - + - * / % ++ -- +
```

- `/` 左右两端的类型需要一致；
- `%`最后的符号和被模数相同；
- **前++；先+1，后运**
- **后++；先运算，后+1**
- `+`：当String字符串与其他数据类型只能做**连接运算**；并且结果为String类型；

2.2、比较运算符

```
= += -= *= /= %=
```

运算符	作用
==	相等于
!=	不等于
<	小于
>	大于
<=	小于等于
>=	大于等于
instanceof	检查是否是累的对象

2.3、赋值运算符

```
= += -= *= /= %=
```

- 举例：

  `a+=20;`《——》`a=a+20;`

2.4、逻辑运算符

```
& | ^ ! && ||
```

运算符	含义
&	逻辑与
&&	短路与
|	逻辑或
||	短路或
!	逻辑非
^	逻辑异或

- `& 与 &&`以及|与||的区别：
  - &：左边无论真假，右边都会进行运算；
  - **&&：如果左边为假，则右边不进行运算；**
- `| 与 ||`的区别同上；在使用的时候建议使用&&和||；
- `(^)与或（|）`的不同之处是：当左右都为true时，结果为false。

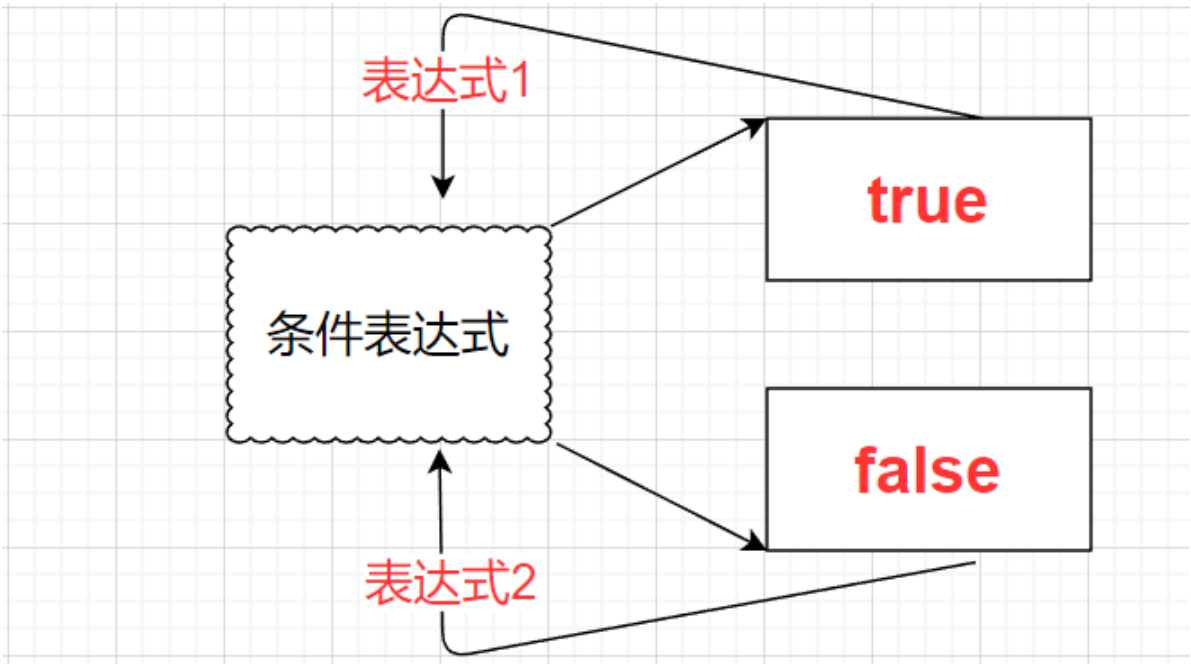
2.5、位运算符

运算符	运算	细节
<<	左移	高位丢弃，空位补0
>>	右移	最高位为0，右移后，空位补0。最高位为1，右移后，空位补1
>>>	无符号右移	右移无关最高位，空位都补0
&	与运算	与运算，`1&1=1` `1&0=0`
|	或运算	或运算，`0&0=0` `1&0=1`
^	异或运算	异或运算，`0&0=0` `1&0=1`
~	反码	各位取反码求值

2.6、三元运算符

`格式`

```
（条件表达式）？ 表达式1:表达式2
```



3、运算符优先级

运算符优先级

优先级	运算符
1	()
2	!、~、++、--
3	*、/、%
4	+、-
5	<<、>>、>>>
6	<、<=、>、>=、instanceof
7	==、!=
8	&
9	^
10	|
11	&&
12	||
13	?:
14	=、+=、-=、*=、/=、%=、&=、

输入

```
Scanner scanner = new Scanner(System.in);
String name = scanner.next();
System.out.println(name);
```

三、程序流程控制

目录

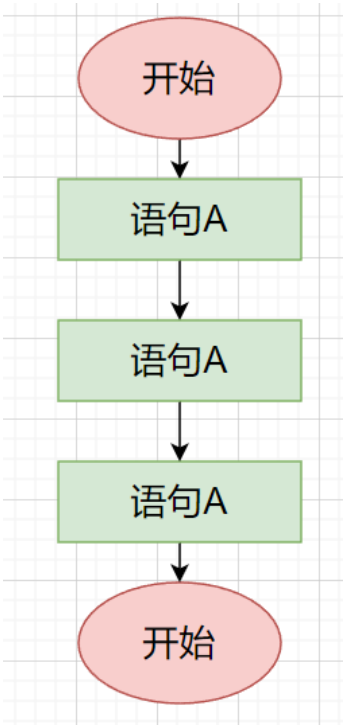
- 【1】顺序结构
- 【2】分支结构
- 【3】循环结构

> 流程是指程序运行时，各语句的执行顺序。流程控制语句就是用来控制程序中各语句执行的顺序。

1、顺序结构

顺序结构

> 按照代码的先后顺序，依次执行



2、分支结构

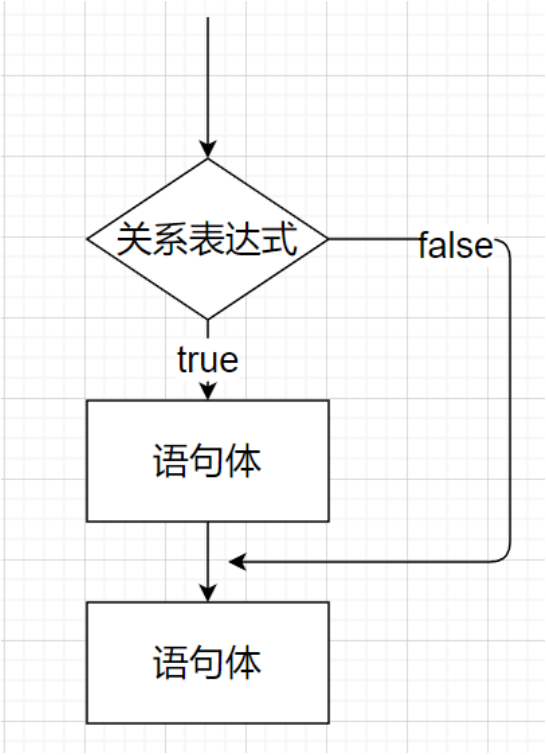
分支结构

> 条件语句可根据不同的条件执行不同的语句。包括if条件语句与switch多分支语句。

2.1、if分支

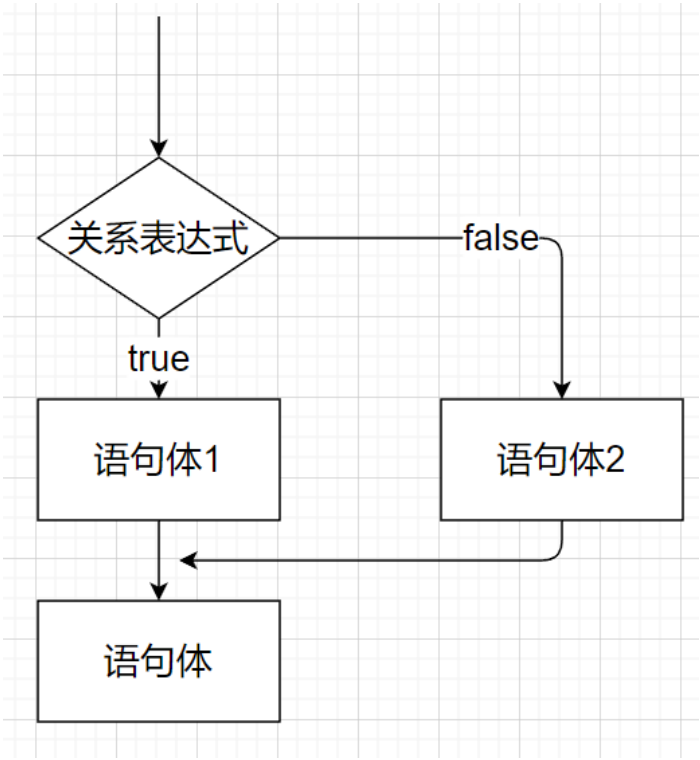
2.1.1、格式1

```
if(条件表达式){
    执行的代码块;
}
```



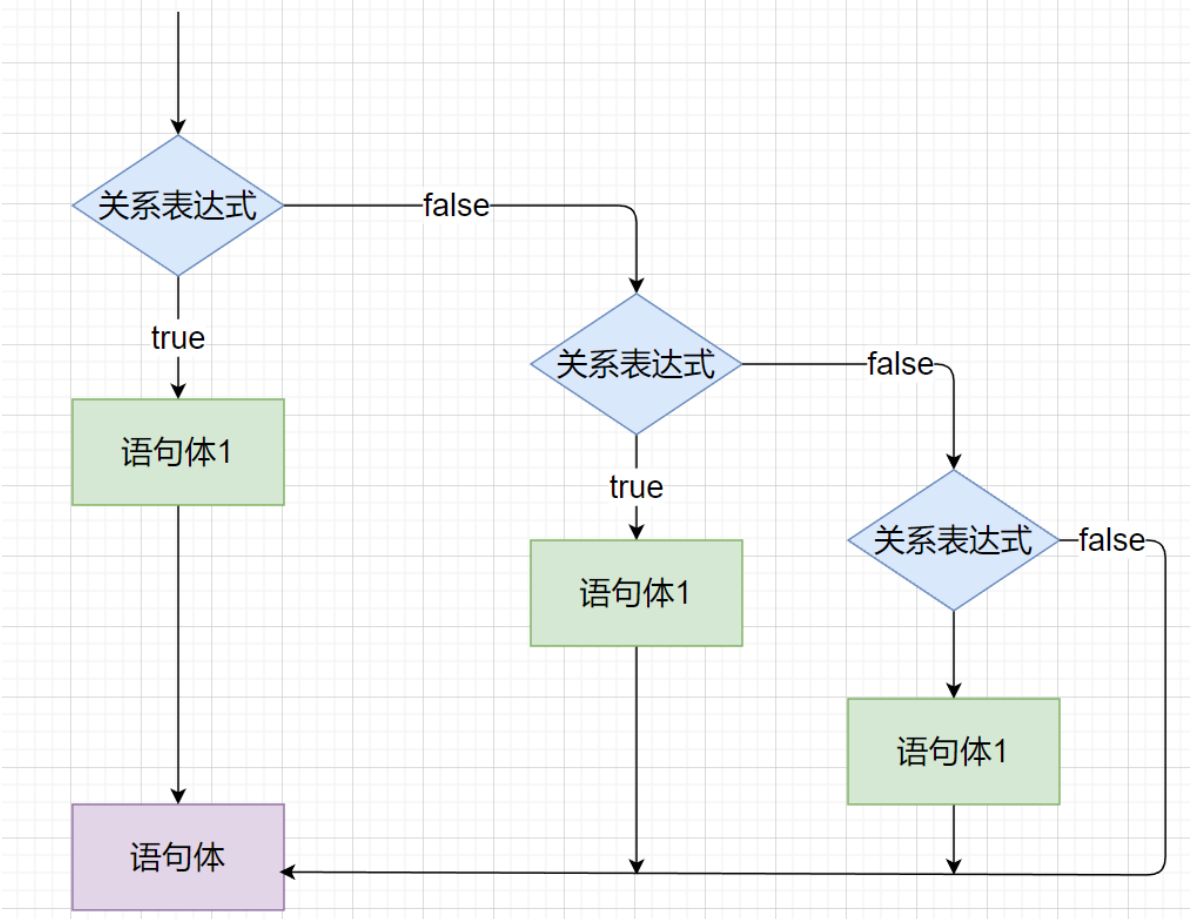
2.1.2、格式2

```
if(条件表达式){
    执行的代码块;
}else{
    执行的代码块;
}
```



2.1.3、格式3

```
if(条件表达式){
    执行的代码块;
}else if(条件表达式){
    执行的代码块;
}.......
else{
    执行代码块;
}
```



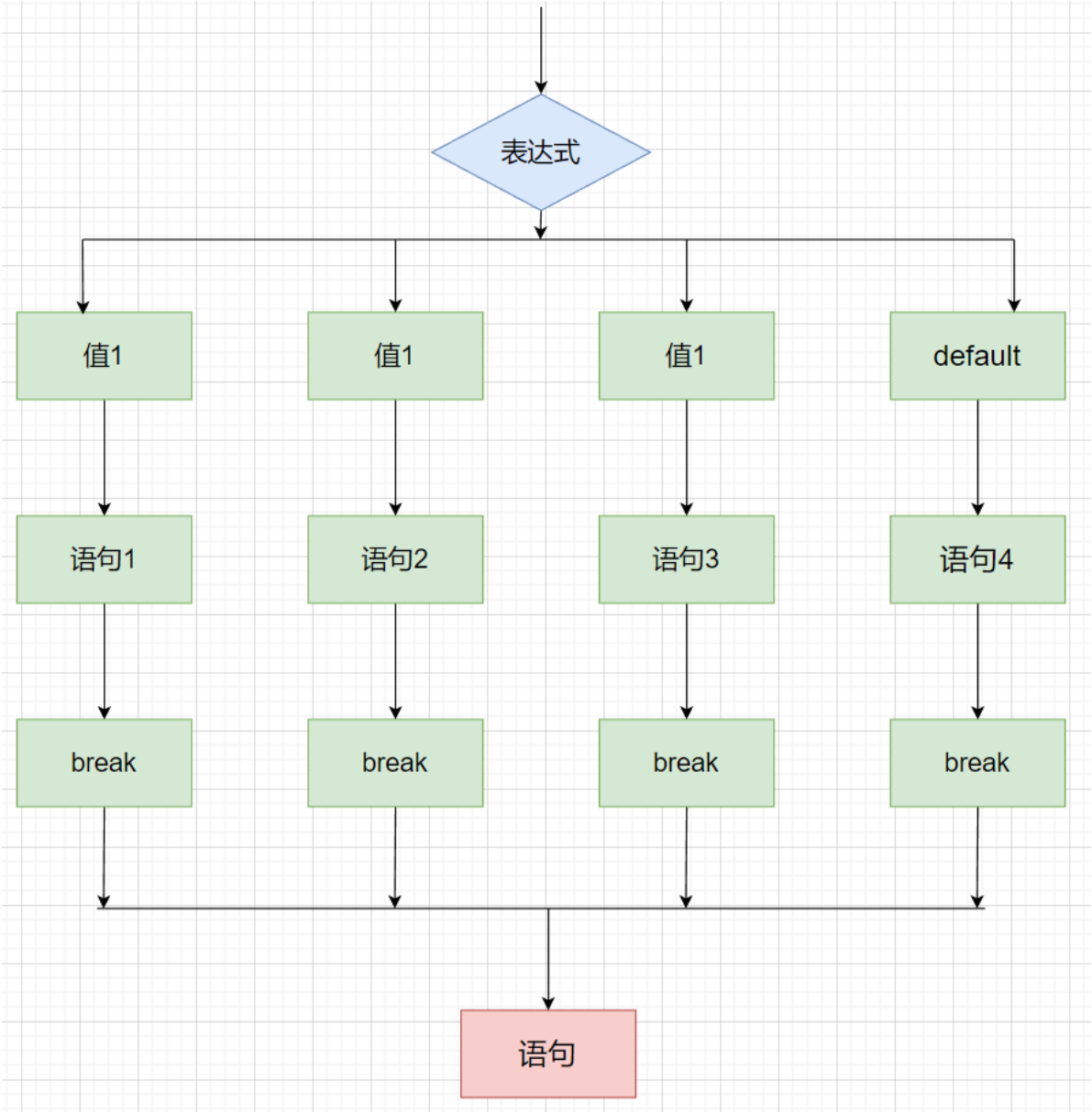
- 一旦满足某个条件表达式，**则进入其执行语句块执行，执行完毕后不会执行其一下的条件语句。**
- 如果多个条件表达式之间为“互斥”关系，多个语句之间可以上下调换顺序，一旦是包含关系，要求条件表达式范围小的写到范围大的上边；

2.2、switch分支

> - **首先计算出表达式的值**
> - 和case依次比较，一旦有对应的值，就会执行相应的语句，在执行的过程中，**遇到`break`就会结束。**
> - **如果所有的case都和表达式的值不匹配，就会执行default语句体部分**，然后程序结束掉。

```
switch (n) {
    case 1:
        语句1;
        break;
    case 2:
        语句2;
        break;
```

```
    case 3:
        语句3;
        break;
    case 4:
        语句4;
        break;
    default:
        默认语句
        break;
}
```



注意

- 表达式类型
  - 支持 `byte`、`short`、`int`、`char`、`String`、枚举
  - 不支持 ~~`double`~~、~~`float`~~、~~`long`~~
- case给出的值不允许重复，只能为字面量，不可为常量
- **注意 `break`，防止循环穿透**

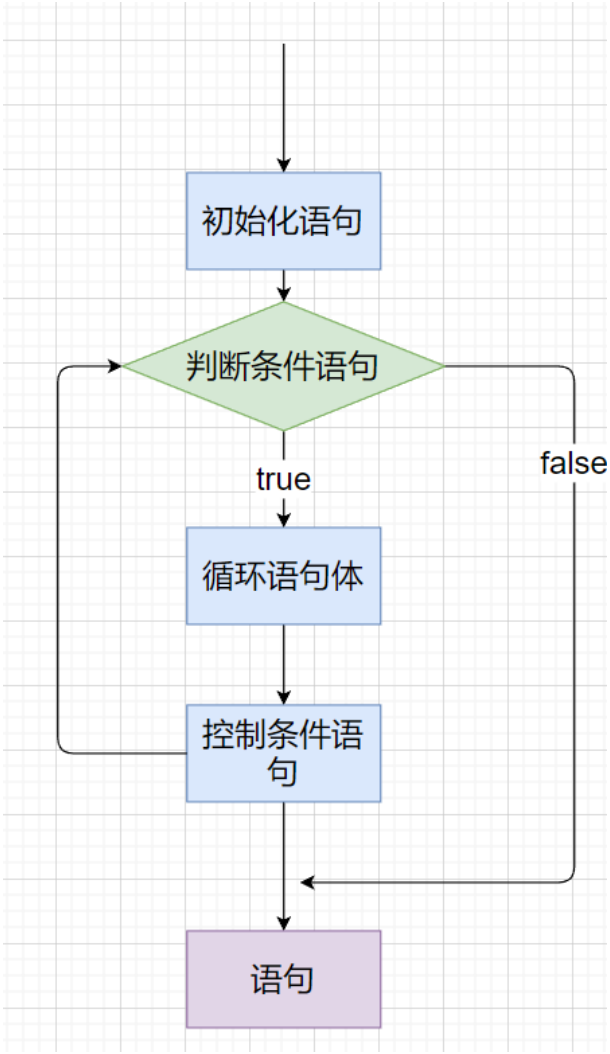
3、循环结构

循环结构

> 循环语句就是在满足一定条件的情况下反复执行某一个操作。包括while循环语句、do…while循环语句和for循环语句。

3.1、for

```
for(初始化语句;判断条件语句;控制条件语句) {
    循环体语句;
}
```



3.2、foreach

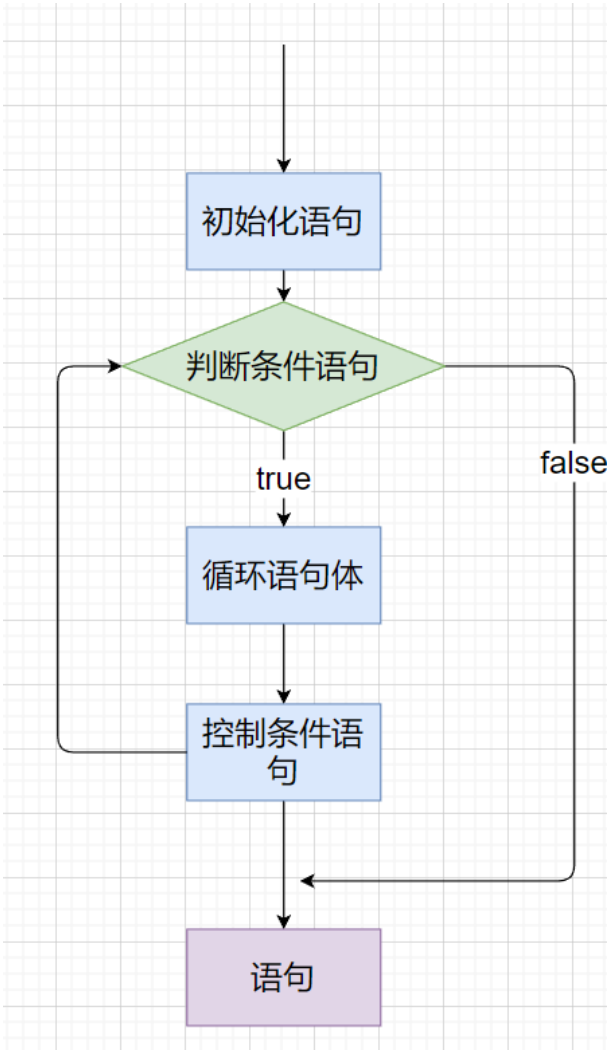
```
for (声明语句 : 表达式) {
代码句子
}
```

举例

```java
int[] numbers = { 10, 20, 30, 40, 50 };
for (int x : numbers) {
    System.out.print(x);
    System.out.print(",");
}
System.out.print("");
String[] names = { "James", "Larry", "Tom", "Lacy" };
for (String name : names) {
    System.out.print(name);
    System.out.print(",");
}
```

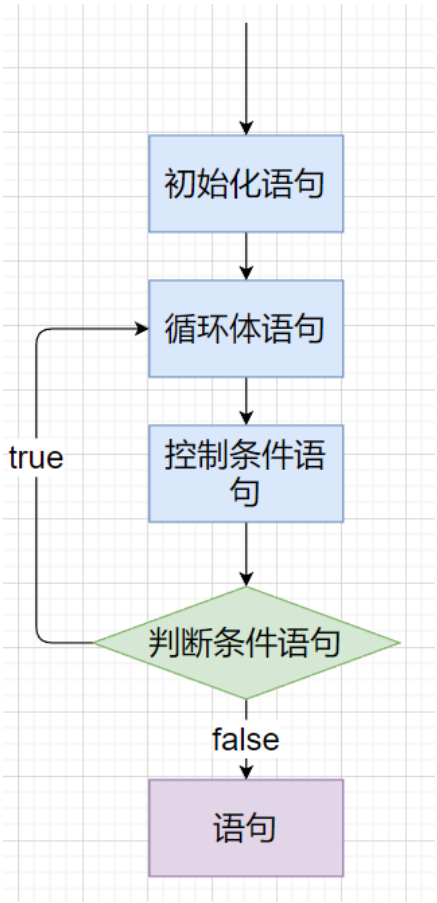
3.3、while

```
初始化语句;
while(判断条件语句) {
    循环体语句;
    控制条件语句;
}
```



3.4、do..while

```
初始化语句;
do {
循环体语句;
控制条件语句;
} while(判断条件语句);
```



3.5、区别

- 【1】`do…while`**循环至少会执行一次循环体。**
- 【2】**for循环和while循环只有在条件成立的时候才会去执行循环体**
- 【3】for循环语句和while循环语句的小区别：
  - **控制条件语句所控制的那个变量，在for循环结束后**，就不能再被访问到了，而while循环结束还可以继续使用，如果你想继续使用，就用while，否则推荐使用for。原因是for循环结束，该变量就从内存中消失，能够提高内存的使用效率。

3.6、跳转语句

`break`

> 在循环语句中，可以跳出单层循环

`continue`

> 在循环语句中，结束一次循环，继续下一次的循环

`return`

> 在循环中或者方法中，返回值，**直接结束循环或者方法**