第2章 Java语言基础

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 标识符和关键字
- 基本数据类型、变量、常量
- 运算符
- 表达式的类型转换
- 流程控制
- 方法

2.1 标识符和关键字

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

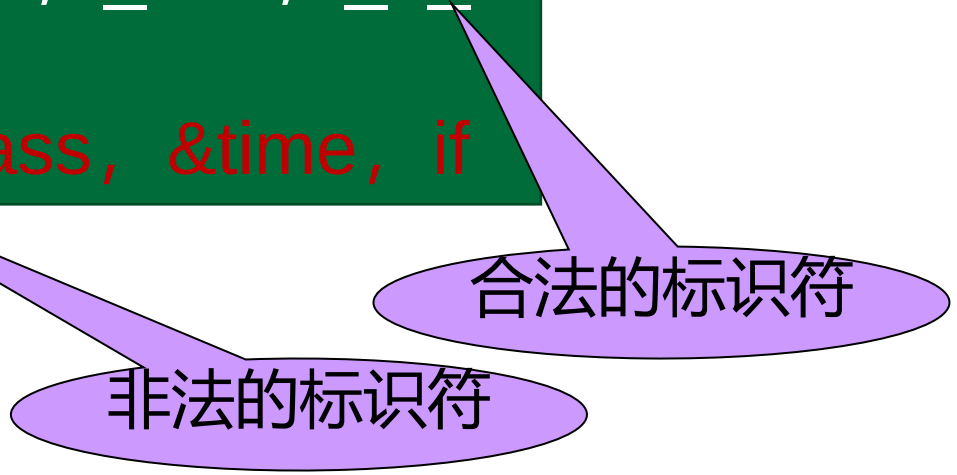
- **标识符**:程序员声明的单词，命名程序中的一些实体。如类名、对象名、方法名、变量名等。

- **Java标识符命名规则**
  - 以字母、下划线(_)或美元符$开始
  - 可以由大写字母、小写字母、下划线(_)、数字0~9组成
  - 不能是Java的关键字
  - 大写字母和小写字母代表不同的标识符
  - 标识符的长度是任意的
  - 不要使用系统预定义的符号，以免引起混淆

2.1 标识符和关键字

myName，My_name，Points，
$points，_sys_ta，OK，_23b，_3_

#name，25name，class，&time，if



2.1 标识符和关键字

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- Java中的命名规则
  - **包名：**
    统一使用小写字母；
    多层包之间用点进行分隔
    一般采用域名倒写的方式进行命名 ,如cn.edu.bit.*
  - **类名：**
    使用名词，
    驼峰命名法 ： 每个词的首字母大写， 如HelloWorld。
  - **接口名：**
    同类名
  - **抽象类：**
    符合类名的命名规范即可， 为了和接口做出区别，一般以“Abstract”作为前缀。

2.1 标识符和关键字

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- Java中的命名规则
  - **方法名：**

    使用动词；

    首字母小写，其余各词的首字母大写；

    不建议使用下划线作为连接；

    有返回值的方法一般加“get”前缀，设置的方法一般加对应的动词作为前缀（如：set、insert、update、delete）；

    查询的方法一般以“select”或“find”或“query”作为前缀；

    带有条件的方法一般在命名中使用“by”或“with”等字符；

    判断的方法一般以“is”作为前缀；

    测试方法一般以“test”作为前缀。

2.1 标识符和关键字

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- Java中的命名规则
  - 方法名：

```
1  public void show() {}
2  public String getName() {}
3  public void setName() {}
4  public int insertUser() {}
5  public int updateUser() {}
6  public int deleteUser() {}
7  public List<User> findUser() {}
8  public User queryUser()
9  public User selectUserById(int id) {}
10 public boolean isHas() {}
11 public void testUserServiceImpl() {}
```

2.1 标识符和关键字

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- Java中的命名规则
  - **变量名：**

    使用名词；

    首字母小写，其余各词的首字母大写；

    不建议使用下划线作为连接；

    如userID,userName
  - **常量名：**

    使用名词；

    全部大写字母；

    多个单词之间使用“_”进行分隔。

    如MAX_AGE；MIN_NUMBER

2.1 标识符和关键字

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 关键字:**Java预定义的单词。**
  - 数据类型：byte、short、int、long、char、float、double、boolean
  - 包引入和包声明：import、package
  - 类和接口的声明：class、extends、implement、interface
  - 流程控制：if、else、switch、case、break、default、while、for 、do、continue、return
  - 异常处理：try、catch、finally、throw、throws
  - 修饰符：abstract、final、private、protected、public、static、synchronized
  - 其他：new、instanceof、this、super、void、enum

Java 语言定义的关键字不能作为变量名、类名和方法名

2.2 基本数据类型与变量、常量

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- Java的数据类型
  - 基本数据类型:Java语言本身定义的数据类型。

基本类型。一种没有引用的对象，不用new来创建变量

  - 复合数据类型(引用类型):用户根据自己的需要定义的数据类型。

2.2 基本数据类型与变量、常量

2.2 基本数据类型与变量、常量

Java要确定每种基本类型所占存储空间的大小，他们的大小并不像其他语言那样随机器硬件架构的变化而变化。

数据类型	关键字	在内存中占用的位数	取值范围	成员默认值
字节型	byte	8	-128~127	(byte)0
短整型	short	16	-32768~32767	(short)0
整型	int	32	-2^{31}~$2^{31}-1$	0
长整型	long	64	-2^{63}~$2^{63}-1$	0L
字符型	char	16	0~65535	'\u0000'
单精度浮点型	float	32	1位符号,8位指数,23位尾数	0.0F
双精度浮点型	double	64	1位符号,11位指数,52位尾数	0.0D
布尔型	boolean	1	true，false	false

2.2.1 Java中的整数类型

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 整型常量按照所占用的内存大小分类
  - 整型(int)常量： 占用32位。
    如123，-34
  - 长整型(long)常量： 占用64位， 长整型常量的尾部有一个大写的L或小写的l。
    如-386L，017777l
  - 说明： **java中的整型常量默认为int**， 表示long型整数后面加后缀。
    byte a=200;(报错)
    Long b=2500000000(报错) 修改为2500000000L

Byte为字节型，取值范围-128-127

超出了int范围，但会被默认为int，因此要在数字后加后缀L

2.2.2 Java中的字符类型

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- Java中的字符采用Unicode字符集的编码方案，是16位的无符号整数，其前128个字符编码与ASCII码兼容。

2.2.2 Java中的字符类型

- **字符常量**：用一对单引号括起的单个字符。
  - 可见字符：'a'，'Z'，'8'，'#'
  - 转义字符
    - '\n'
    - '\t'
    - '\ddd'： 8八进制表示一个字符
    - '\uxxxx': 16进制无符号整数，表示Unicode码。

如: '\101'　用8进制表示一个字符'A'

'\u0041' 用Unicode码表示一个'A'

2.2.2 Java中的字符类型

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

转义字符	含义	对应Unicode码
' \b'	退格	' \u0008'
' \t'	水平制表符tab	' \u0009'
' \n '	换行	‘ \u000a'
' \f '	表格符	' \u000c'
' \r'	回车	' \u000d'
' \” '	双引号	' \u0022'
' \’ '	单引号	' \u0027 '
' \\ '	反斜线	' \u005c'
' \ddd '	三位8进制数表示的字符	
' \uxxxx '	四位16进制数表示的字符	

2.2.2 Java中的字符类型

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 字符串常量是用双引号括起的一串字符（可以0个）。

  例子："Hello"，

  "My \nJava"，

  "How old are you? 1234"，

  " "

  ""

  "My" + "name"

字符串常量是String类的对象

2.2.3 浮点类型

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 浮点型常量：表示可以含有小数部分的数值常量。
- 根据占用内存长度的不同分类
  - **单精度浮点常量float**：占用32位内存，用F、f表示。**如：19.4F，3.0513E3，8701.52f**
  - **双精度浮点常量double**：占用64位内存，用带D或d或不加后缀的数值表示，
    如：2.433E-5D，700041.273d，3.1415。

2.2.3 浮点类型

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 说明：在java中的实数型常量默认为double，所以写单精度的实数时要在数字后面写f，如3.14f。
  float a=3.4;(报错)float a=3.4f;

- 浮点型可能会有精度丢失，运算不够精确，不能对其进行精确的==运算

- 采用BigDecimal类进行精确运算，Java 提供了两个用于高精度计算的类: **BigInteger** and **BigDecimal**.必须以方法调用方式取代运算符方式来实现

2.2.4 布尔类型

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 布尔常量:true(真)和false(假)。
- 在流控制中经常用到布尔常量。

```
if (条件） 动作1
else       动作2
```

- 注意：Java不允许数值类型和布尔类型之间进行转换

```
int a=3;
if ( 0<a<1 ) ……
```

2.2.5 符号常量

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 在Java中必须用final关键字声明符号常量
- final关键字表示这个变量只能被赋值一次，一旦赋值后就不能够再更改。
- 声明格式
  - final 数据类型 常量名 = 缺省值;

    final int STUDENT_NUM = 10;

- 习惯上，符号常量名采用全部大写，词与词之间用下划线分隔。

变量

- 变量:在程序的运行过程中数值可变的数据，
  用来记录运算中间结果或保存数据。
- 变量的声明

数据类型　变量名1，变量名2，…变量n;

例如:

int num,total;

double d;

byte, short, int, long,
float, double, char, boolean
引用类型

- 变量初始化: 在变量声明时使用表达式初始化变量，也可以直接赋值。

```
class DynInit {
    public static void main(String[]  args){
        double  a = 3.0, b = 4.0;
        double  c = Math.sqrt(a * a + b * b);
        System.out.println("Hypotenuse is：  " + c);
    }
}
```

表达式初始化

变量

- 使用Var定义变量。
- 方式声明var会根据后面的值来推断变量的类型

```
var 变量名 = 初始值;
```

var是Java10中新增的局部类型变量推断

var根据初始化的值来推断变量的类型，必须要初始化，一但初始化，变量类型不可改变

var只能在方法内定义变量，不能用作方法参数

var每次只能定义一个变量，不能复合声明变量

变量

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 使用Var定义变量。

```
1  public class UseVar {
2      public void baseTypeVar() {
3          var a = 1;
4          var b = 255;
5          var c = 256;
6          var d = 10L;
7      }
8  }
```

2.3 运算符

```
int S=-a*x*x+b*x+c;
boolean l=a>b;
```

- 运算符: 指明对操作数的运算方式。
- 按操作数的个数分:单目运算符(如-a)，双目运算符(如a+b),三目运算符(如e1?e2:e3)。
- 按功能分类
  - 算术运算符： +，-，*，/，%，++，--
  - 关系运算符： >，<，>=，<=，==，!=
  - 逻辑运算符： !，&&，||，&，|
  - 赋值运算符： =，+=，—=，*=，/=等
  - 位运算符：<< ， >> ， >>>
  - 条件运算符： ?:
  - 其它：instanceof， () 等

2.3 运算符

北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

- **表达式**:由运算符、操作数(常量、变量、方法调用)和圆括号组成的式子。

2.3.1 算术运算符

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 算术运算符:对整型或实型数据的运算。

- 算术运算符分类
  - 双目运算符
  - 单目运算符

- 双目算术运算符

运算符	运算	例	功能
+	加	a + b	求a与b相加的和
-	减	a - b	求a与b相减的差
*	乘	a * b	求a与b相乘的积
/	除	a / b	求a除以b的商
%	取余	a % b	求a除以b所得的余数

- **注意：**

(1)与C不同的是，Java中的“%”两个运算对象可以是实数。

13%5=3

23.6%12=11.6

(2)两个整型的数据做“/”运算时，结果是截取商的整数部分，而小数部分被截断。

2/4=0

2.0/4=0.5

- 单目运算符：操作数只有一个。

运算符	运算	例	功能等价
++	自增	a++或++a	a = a + 1
--	自减	a--或--a	a = a - 1
-	求负数	-a	a = -a

例如：

```
int   x=-1;
```

```
int  y = (x++)*3;        int y=  (++x)  *3
```

- 前缀(++, --)：先执行单目运算，修改变量的值后用这个新值参与表达式的运算。
- 后缀(++, --)：先计算表达式的值，最后再修改变量的取值。
- 自增和自减的操作对象只能是变量。

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

前缀和后缀运算符举例

例如：

```
int     x = 5 ;
int     y = (--x) * 3;
```

x为4 y为12

x为4 y为15

```
int    y = (x--) * 3;   ? ?
```

例2-2

```
public static void main(String[] args) {
    int  i = 10, j = 8, m = 11, n = 20, k, g;

    System.out.println(i++);
    System.out.println(++j);

    System.out.println("i="+i);
    System.out.println("j="+j);

    k = m++;
    System.out.println("k="+k);
    System.out.println("m="+m);

    g = 3*(++n);
    System.out.println("g="+g);
    System.out.println("n="+n);
}
```

如果其操作数中有一个是字符串类型，则“+”功能为字符串的连接运算　　EX21.java

【例2-1】写出下面程序运行的结果。

```
public static void main(String[] args) {
        int a=10, b=20;
        System.out.println("a+b="+a+b);
        System.out.println("a+b="+(a+b));
}
```

2.3.2 关系运算符和逻辑运算符

运算符	运算	例
= =	等于	a==b
! =	不等于	a!=b
>	大于	a>b
<	小于	a<b
>=	大于等于	a>=b
<=	小于等于	a<=b

2.3.2 关系运算符和逻辑运算符

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 逻辑运算是对布尔型数据进行的运算，运算的结果仍然是布尔型。
- 常用的逻辑运算符

```
true ||false
(3>1) && (5>-4)
!false
```

运算符	运算	例	运算规则
!	逻辑取反	! x	x真时为假,x假时为真
||	逻辑或	x || y	x,y都假时结果才为假(短路)
&&	逻辑与	x && y	x,y都真时结果才为真(短路)
^	布尔逻辑异或	x ^ y	x,y同真同假时结果为假
&	布尔逻辑与	x & y	x,y都真时结果才为真
|	布尔逻辑或	x | y	x,y都假时结果才为假

短路：一旦可以明确确定逻辑表达式的值，余下的部分可以不用计算。

- 逻辑运算符与布尔逻辑运算符的区别 e1 && e2
  - 逻辑运算符： && ||

  利用它们做逻辑运算时，运算时右边的表达式有可能被忽略(短路)而不加执行。 e1 & e2

  - 布尔逻辑运算符：&、|、^

  利用它们做与、或、异或运算时，运算符左右两边的表达式总会被运算执行，然后再对两表达式的结果进行与、或运算。

例2-3

短路

```
例如：int  x = 3,  y = 5 ;
        boolean  b= x > y && x++ == y-- ;
       //运行后x为3，y为5，b为false
```

```
        boolean  b= x > y & x++ == y-- ;
       //运行后x为4，y为4，b为false
```

2.3.3 位运算符

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- **位运算**是对操作数以二进制比特位为单位进行的操作和运算，位运算的运算对象只能是整型和字符型，结果为整型。

【例2-5】航班计算问题。

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 设某航班周一、三、四、六飞行，当客户订票时如何根据客户的需求“星期几”获知该日是否有航班？

对应的星期		六	五	四	三	二	一	日
航班信息的二进制位flagFight byte flagFight=90	0	**1**	0	**1**	**1**	0	**1**	0
想获取哪位信息时 通过左移运算将“1”移至该位 并与flagFight进行位与运算	0	0	0	0	0	0	0	1
	0	0	0	0	0	0	1	0
	0	0	0	0	0	1	0	0
	.....							

```
( flagFight & (1<<n)) > 0
```

例2-5

- 设某航班周一、三、四、六飞行，当客户订票时如何根据客户的需求“星期几”获知该日是否有航班？

```
public static void main(String[] args) {
        byte  flagFight=90;  //1,3,4,6有航班

        //输入要查询的日期
        System.out.println("输入要查询的日期(星期几),星期日用0表示: ");
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();

        if( (flagFight&(1<<n))>0 ){
            System.out.println("该日有航班");
        }else{
            System.out.println("该日没有航班");
        }
    }
```

2.3.4 赋值运算符

- Java中赋值运算符：=、+=、-=、*=等。
- 赋值表达式:带有赋值运算符的表达式。
- 赋值表达式的含义:等号右边表达式的值赋给等号左边的变量。

例如，i=5　　//赋值表达式的值是5

操作运算符中的+ 和= 可以对引用类型操作

2.3.4 赋值运算符

```
i= 1;
i=2+(j=4);
i=(j=10)*(k=2);
```

//表达式值为1

//表达式值为6， j的值为4，i的值为6

//表达式值为20， j的值为10，k的值为2，i的值为20

2.3.4 赋值运算符

- 常用的复合赋值运算符

运算符	例子	等价于
+ =	x += a	x = x + a
- =	x -= a	x = x - a
*=	x *= a	x = x * a
/=	x /= a	x = x / a
%=	x %= a	x = x % a

- 例：a+=3　　等价于　a=a+3

  x*=y+8　　等价于　x=x*(y+8)

- 条件运算符与条件表达式

  e1? e2: e3

  - e1 为 boolean 类型
  - e2与e3的类型相同

- 执行顺序
  - 若e1的值为true，e2的值为最终结果
  - 若e1的值为false，e3的值为最终结果

2.3.5 运算符的优先级与结合性

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- **表达式的运算次序**:取决于表达式中各种运算符的优先级。优先级高的运算符先运算,优先级低的运算符后运算，同一行中的运算符的优先级相同。

- **运算符的结合性**:决定了并列的相同运算符的先后执行顺序。

2.3.5 运算符的优先级与结合性

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

优先级	描述	运算符	结合性
1	最高优先级	. [ ] ( )	左→右
2	单目运算	+(正号) –(负号) ++ -- ～ ！ 强制类型转换符	右→左
3	算术乘除运算	* / %	左→右
4	算术加减运算	+ -	左→右
5	移位运算	>> << >>>	左→右
6	关系运算	< <= > >=	左→右
7	相等关系运算	= = !=	左→右
8	按位与,布尔逻辑与	&	左→右
9	按位异或	^	左→右
10	按位或,布尔逻辑或	|	左→右
11	逻辑与	&&	左→右
12	逻辑或	||	左→右
13	三目条件运算	? :	右→左
14	赋值运算	= += -= *= /= %= <<= >>=	右→左

高

低

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

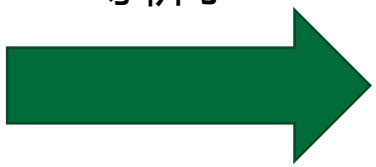
例

1、X > Y && ! Z

2、X + Y + Z

3、！ ！ X

等价于



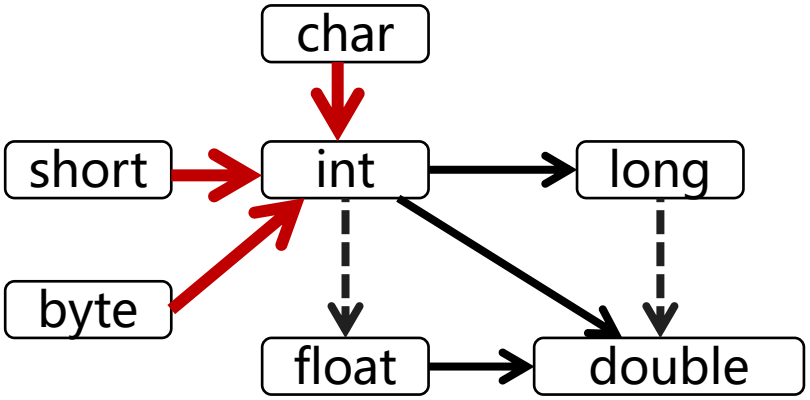
(X > Y) && (! Z)

(X + Y) + Z

! (! X)

不用死记，用好小括号即可

- 当表达式中出现了多种类型数据的混合运算时,需要进行类型转换。



boolean型数据不参与混合运算

只要计算，必须进行的转换

混合计算时发生的转换

转换时发生损失

从小类型可以自动转换为较大类型
表达式值结果的类型=表达式中最大的类型

2.4.1 数据类型自动转换的规则

【例2-6】分析下面的赋值出错的原因。

```
public static void main(String[] args) {
        int a = 1.2345;

        byte b = 1;
        b = b+1;

        float c = 1.5;
}
```

基本类型数据占有的内存宽度

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

数据类型	关键字	占用位数	取值范围
布尔型	boolean	8	true, false
字符型	char	16	‘\u 0000’～ ‘ \u FFFF’
字节型	byte	8	-128～127
短整型	short	16	-32768～32767
整型	int	32	-2147483648 ～ 2147483647
长整型	long	64	-2^{-63}～$2^{63}-1$
浮点型	float	32	1.40129846432481707e-45～ 3.40282346638528860e+38
双精度型	double	64	4.94065645841246544e-324～ 1.79769313486231570e+308d

2.4.2 强制类型转换

- 从较长的数据类型转换成较短的数据类型时，必须做强制类型转换。即将表达式的类型强制性地转换成某一数据类型。

- 强制类型转换的格式

  （数据类型）表达式

2.5 流程控制

北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

- 算法的基本控制结构
  - 顺序结构
  - 选择结构
  - 循环结构

2.5 流程控制

北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

- 与算法的基本控制结构相关的Java语句
  - 分支语句：if-else, switch
  - 循环语句：while, do-while, for
  - 与程序转移有关的其它语句：break, continue, return

2.5.1 if语句

(1) if (表达式) 语句

例：if (x <= 0) x = -x ;

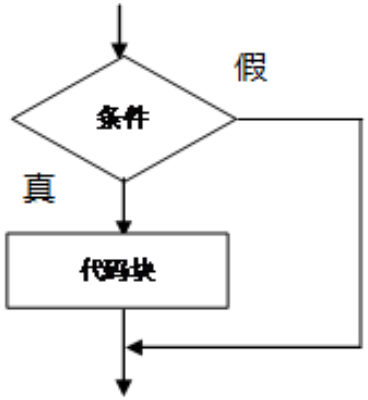
(2) if (表达式) 语句1 else 语句2

例：if (x>y) z=x;
else z=y;

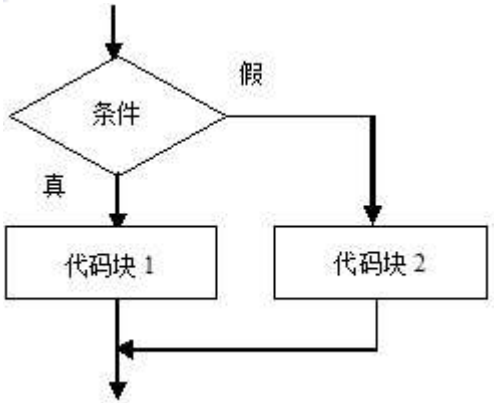
(3) if (表达式1) 语句1
else if (表达式2) 语句2
else if (表达式3) 语句3
…
else 语句 n

2.5.1 if语句

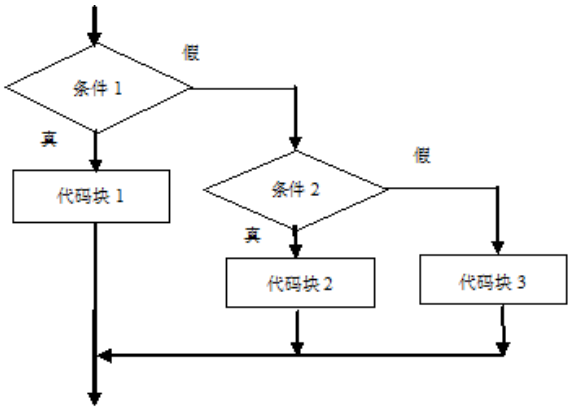
```
if(条件){
   条件成立时执行的代码
}
```



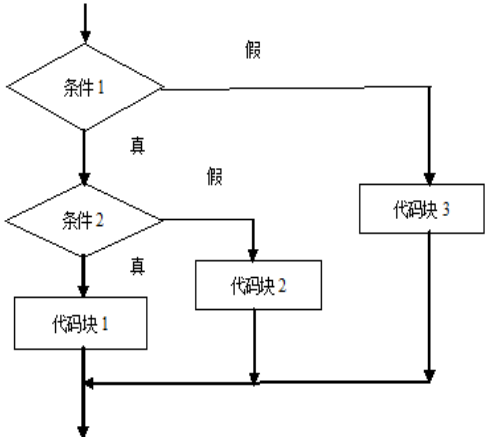

```
if(条件的布尔表达式){
   代码块1
}else{
   代码块2
}
```




```
if(条件1){
    代码块1
}else if(条件2){
    代码块2
}else{
    代码块3
}
```




```
if(条件1){
    if(条件2){
       代码块1
    }else{
       代码块2
    }
}else{
    代码块3
}
```



if-else语句或许是控制程序流程最基本的形式。
条件表达式必须产生一个布尔值

普通闰年：公历年份是4的倍数，且不是100的倍数的，为闰年（如2004年、2020年等就是闰年）。
世纪闰年：公历年份是整百数的，必须是400的倍数才是闰年（如1900年不是闰年，2000年是闰年）。

```
if(year%4==0 && year%100!=0 || year%400==0){
        System.out.println(year+"是闰年");
}else{
        System.out.println(year+"不是闰年");
}
```

【例2-7】根据输入的运算符（+、-、*、/）组织运算。

2.5.2 switch语句

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 一般形式

```
switch (表达式)
{ case 常量表达式 1: 语句1
  case 常量表达式 2: 语句2
     ┆
  case 常量表达式 n: 语句n
  default :          语句n+1
}
```

可以是整型、字符型

每个常量表达式的值不能相同，次序不影响执行结果。

- 执行顺序

以case中的常量表达式值为入口标号，由此开始顺序执行。
如果要跳出执行，每个case分支最后应该加break语句。

2.5.2 switch语句

```
char  myGrade= 'A';
```

```
switch (myGrade){
    case  'A' :    myScore = 5;
    case  'B' :    myScore = 4 ;
    case  'C' :    myScore = 3 ;
    default  :    myScore = 0 ;
}
```

myGrade的值为‘A’，执行完switch语句后，myScore的值被赋值为0

```
switch (myGrade){
 case 'A' :    myScore = 5 ;        break;
 case 'B' :    myScore = 4 ;        break;
 case 'C' :    myScore = 3 ;        break;
 default  :    myScore = 0 ;
}
```

myGrade的值为‘A’，执行完switch语句后，myScore的值被赋值为5

多个不同的case值可以执行一组相同的操作。

```
switch (myGrade)
{   case  'A' :
    case  'B' :
    case  'C' :   myScore = 1 ;    //及格
                  break ;
    default :     myScore = 0 ;   //不及格
}
```

2.5.2 switch表达式（java17 后）

使用switch标准方式编写代码太多的break造成代码冗余可读性不高，可以借助函数式接口和lambda表达式简化书写

```
1  switch (day) {
2      case MONDAY, FRIDAY, SUNDAY -> System.out.println(6);
3      case TUESDAY                -> System.out.println(7);
4      case THURSDAY, SATURDAY     -> System.out.println(8);
5      case WEDNESDAY              -> System.out.println(9);
6  }
```

case L ->表达式可以有返回值，如果某个case模块有多条语句，必须用{}，返回值前加yield关键字

每个 case 允许多个常量，用逗号分隔

不再需要break语句；

2.5.2 switch表达式（java17 后）

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

```
int j = switch (day) {
    case MONDAY  -> 0;
    case TUESDAY -> 1;
    default      -> {
        int k = day.toString().length();
        int result = f(k);
        yield result;
    }
};
```

```
int result = switch (s) {
    case "Foo":
        yield 1;
    case "Bar":
        yield 2;
    default:
        System.out.println("Neither Foo nor Bar, hmmm...");
        yield 0;
};
```

只要有返回值必须用yield

如果只有一行返回值可以
不用yield

2.5.3 while循环语句

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 循环结构是在**一定条件下**，反复执行某段程序的流程结构，被反复执行的程序被称为**循环体**。

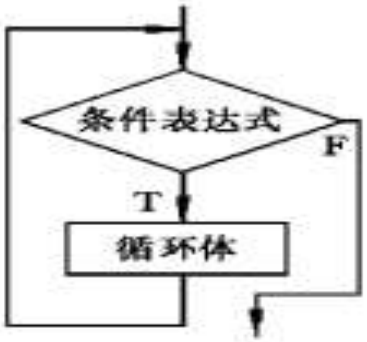
- Java的循环语句
  - while语句
  - do-while语句
  - for语句

2.5.3 while循环语句

- while 语句形式

  while（条件表达式）语句

- 执行顺序



循环体可以是复合语句，其中必须含有改变条件表达式值的语句。

2.5.3 while循环语句

【例2-9】随机生成一个整数（1~100之间），由用户进行猜数，每次给出大小的提示，并记录猜数的次数。

2.5.4 for循环语句

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 语法形式

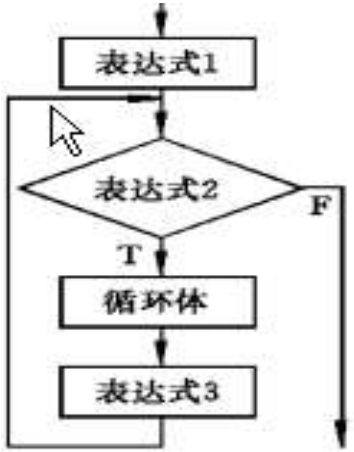
for (表达式1；表达式2；表达式3) 语句

循环前先求解，完成初始化循环变量和其他变量

为true时执行循环体

每次执行完循环体后求解，用于改变循环控制变量的值

例：for(i=1; i<=100; i++) sum+=i;



2.5.4 for循环语句

【例2-10】输入一个日期，包括年、月、日3个数字，计算该日期是该年中的第几天。

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

关于for语句的几点说明

（1）for语句的三个表达式可以为空（但分号不能省略）

```
for (; ;) 语句;
          //相当于  while (true)  语句;
```

（2）在表达式1和表达式3的位置上可包含多个语句

```
for(sum=0, int i=1;  i<=100;  i++)   sum+=i;
```

2.5.4 for循环语句

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

（3）多种表达方式

```
sum=0;
i=1       //在for语句之前给循环控制变量赋初值
for (; i<100; i++)  sum=sum+i;          //省略表达式1
```

```
i=1      //在for语句之前给循环控制变量赋初值
for (sum=0; i<100; i++)    //表达式1与循环控制变量无关
        sum=sum+i;
```

```
for (sum=0,  i=1;  i<100; ){       //省略表达式3
      sum=sum+i;  i++;  }          //在循环体中改变循环控制条件
```

```
for( i=0, j=10; i<j; i++, j--) {……}
                              //  表达式1和表达式3可以是逗号表达式
```

2.5.4 for循环语句

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 注意事项

```
sum=0;
for(int i=1; i<=100; i++)  //在for语句中声明循环控制变量并赋初值
    sum+=i;
```

```
System.out.println(i);     //!Error
```

2.5.5 do-while循环语句

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

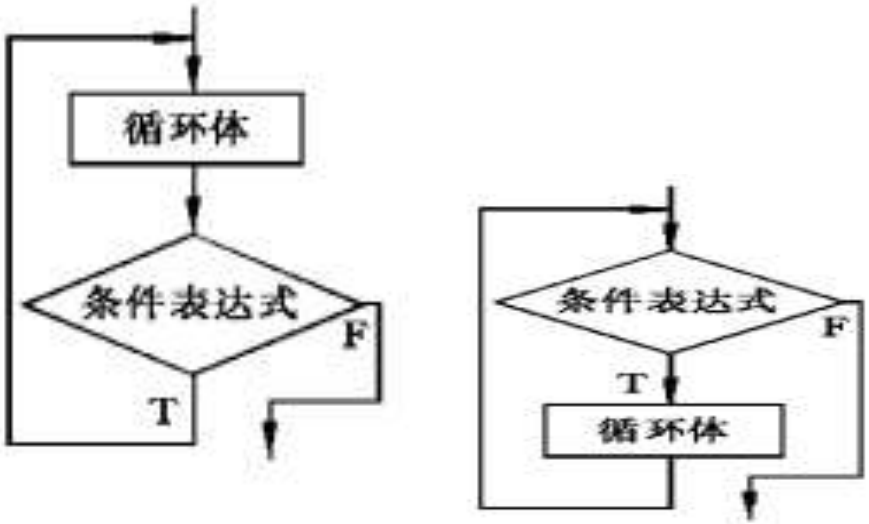
- 一般形式

do 语句
while (表达式)

循环体可以是复合语句，其中必须含有改变条件表达式值的语句。

- 与while语句的比较

while和do-while唯一的区别就是，即使表达式第一次计算结果为false，do-while语句至少会执行1次。
而在while循环中，如果判断条件第一次就为false，那么其中的语句根本不会被执行。



2.5.5 do-while循环语句

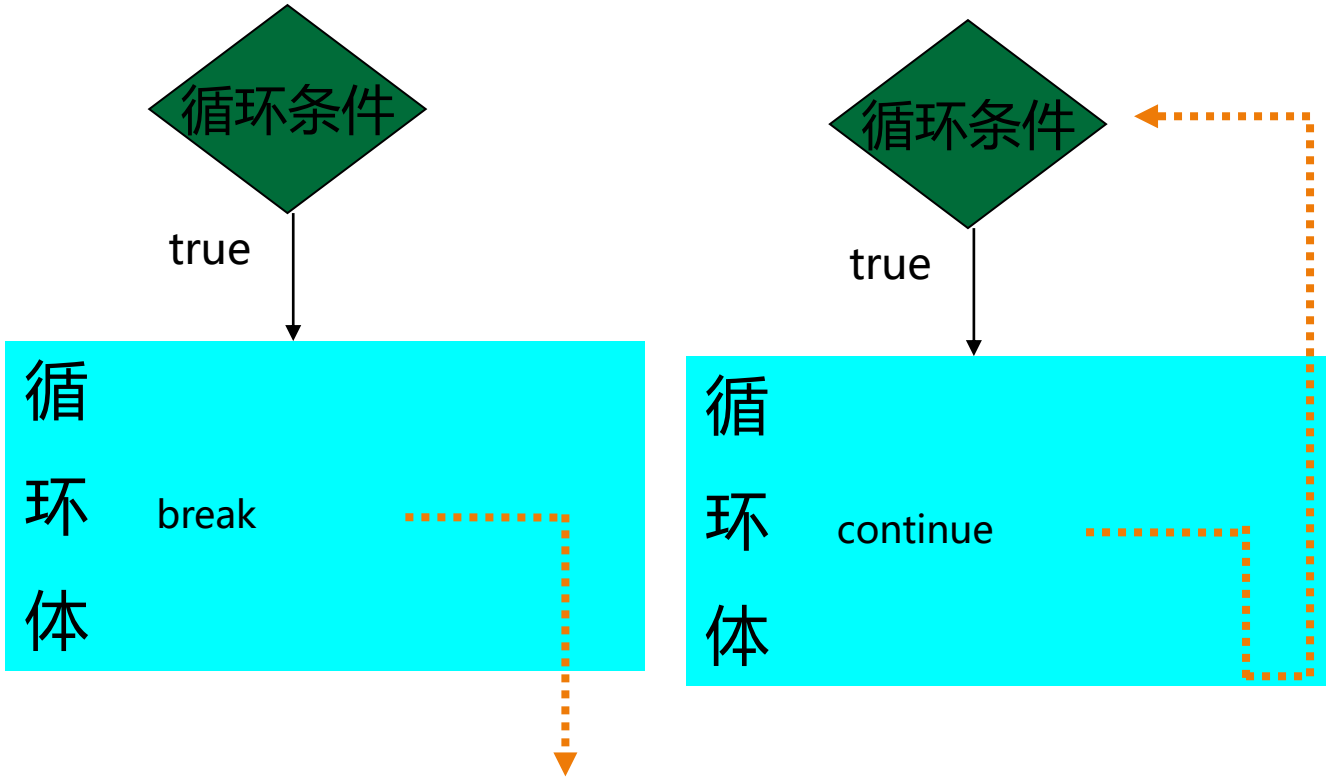
北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

【例2-11】输入两个正数，并利用欧几里德算法（辗转相除法）求它们的最大公约数。

- 用较大数除以较小数，再用出现的余数（第一余数）去除除数，再用出现的余数（第二余数）去除第一余数，如此反复，直到最后余数是0为止。如果是求两个数的最大公约数，那么最后的除数就是这两个数的最大公约数。
- step1: 将两数中大的那个数放在m中，小的放在n中。
- step2: 求出m被n除后的余数r。
- step3: 若余数为0则执行步骤(7)，否则执行步骤(4)。
- step4: 把除数作为新的被除数；把余数作为新的除数。
- step5: 求出新的余数r。
- step6: 重复步骤(3)到(5)。
- step7: 输出n，n即为最大公约数。

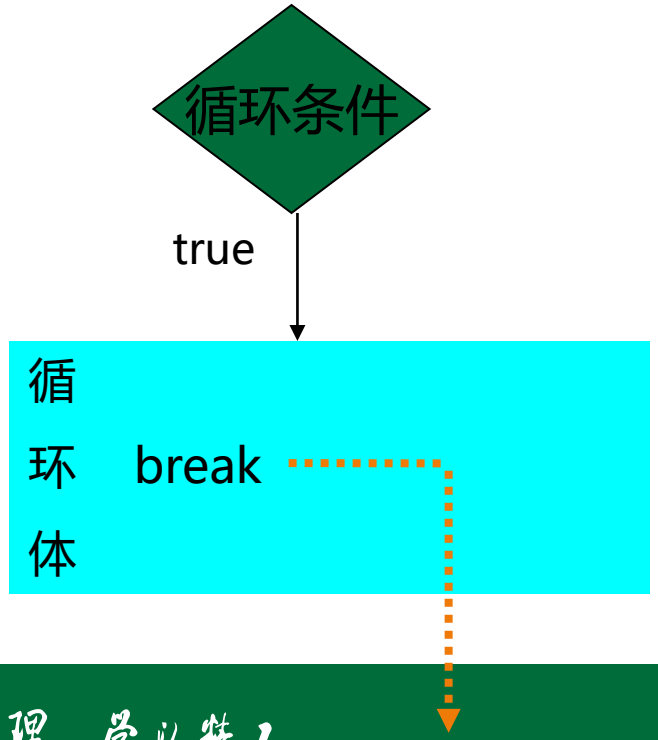
2.5.6 break语句

- 改变程序控制流语句
  - break
  - continue
  - return



2.5.6 break语句

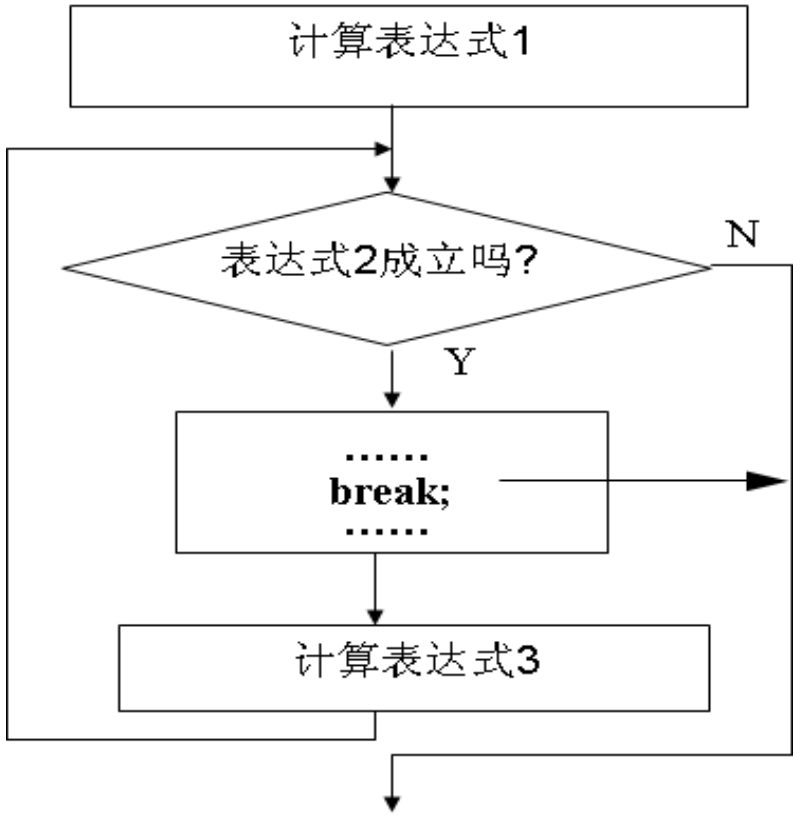
- break语句仅出现在switch语句或循环体中。
  - 作用：使程序的流程从一个语句块内部跳转出来，即从switch语句的分支中跳出，或从循环体内部跳出。



2.5.6 break语句

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- for循环结构中的break语句



2.5.6 break语句

【例2-12】判断某个数是否是素数。

素数又称质数，一个大于1的自然数，除了1和它自身外，不能被其他自然数整除的数叫做素数

- 素数判定法1：

遍历从2到n-1的所有数字，判断是否有可以被n整除的数，如果没有，则为素数。

- 优化法2：

判定的范围改为[2 -,n/2]。当 i>n/2 时，则判定为素数。

- 优化法3：

在Java中判定素数的范围也可以到sqrt(n),(对n开平方)。对应的函数为：Math.sqrt(n)

2.5.6 break语句

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- Java语言中break语句的特殊格式
  - break [标号];

```
stop:
for (int i=1; i<=10; i++) {
    for (int j=1; j<=5; j++) {
        if (i==5)    break stop;
        System.out.print( "*" );
    }
    System.out.println();
}
```

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

作用: 快速地从多重循环内部退出

2.5.7 循环的嵌套

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 循环的嵌套：一个循环体内又包含另一个完整的循环结构。

- 三种循环语句（while循环, do-while循环和for循环）它们可以相互嵌套使用。

2.5.7 循环的嵌套

【例2-13】打印一个指定大小的n×n的棋盘，用星号表示落棋的位置，棋盘位置的编号用0~9，a~z依次表示。

```
输入棋盘的大小:12
   0 1 2 3 4 5 6 7 8 9 a b
0  * * * * * * * * * * * *
1  * * * * * * * * * * * *
2  * * * * * * * * * * * *
3  * * * * * * * * * * * *
4  * * * * * * * * * * * *
5  * * * * * * * * * * * *
6  * * * * * * * * * * * *
7  * * * * * * * * * * * *
8  * * * * * * * * * * * *
9  * * * * * * * * * * * *
a  * * * * * * * * * * * *
b  * * * * * * * * * * * *
```

- 变量声明的作用域

（1）参数声明的作用域是声明方法所在的方法体。

（2）局部变量在方法或方法中的一块代码中声明，它的作用域为它所在的代码块(整个方法或方法中的某块代码)。

（3）在带标号的break和continue语句中，标号的作用域是带标号结构范围的语句（即带标号语句的主体）。

（4）出现在for结构头初始化部分的局部变量，其作用域是for结构体和结构体头中的其它表达式。

- 函数=方法=模块化设计
- Java中所有的方法都必须封装在类中，不能单独出现、使用。

2.6.1 方法的定义

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- Java中方法定义的基本格式为：

```
[修饰符] 返回值类型 方法名([形式参数列表]){
    [方法体]
}
```

- 修饰符：定义方法在类中的存在属性（如公有/私有、是否可以被重载等）
- 返回值类型：如果方法没有返回值则定义为void”
- 形式参数列表：定义方法需要接收的数据及相应数据类型，参数列表可缺省
- 方法体：由完成其逻辑功能的Java语句组成，可为空。

【例2-14】判断某数是否是素数的方法。

2.6.2 方法的重载

- 方法的重载：在一个类中定义多个同名的方法，但方法有不同类型的参数或参数个数。

注意依赖形参识别方法，不能依赖返回值。

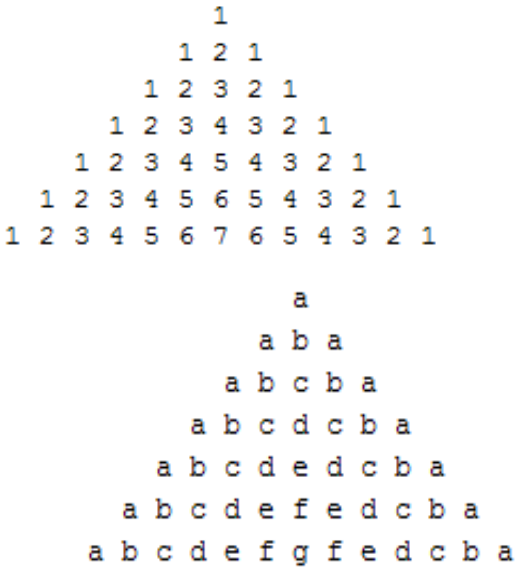
参数的类型不同

参数的个数不同

【例2-15】设计打印金字塔的方法printPyramid()，可以打印数字金字塔，也可以打印字母金字塔。

```
public void printPyramid(int n){
    //打印n行数字组成的金字塔
    ……
}
public void printPyramid(char c){
    //打印'a'~ch字母组成的金字塔
    ……
}
```

Sample : OverloadingOrder.java



用基本类型中的
int 和float,
BigInteger , BigDecimal
实现四则运算。