# Java学习笔记

目录

[Java学习笔记 1](#_Toc124087458)

[Day01 3](#_Toc124087459)

[关键字 3](#_Toc124087460)

[标识符 4](#_Toc124087461)

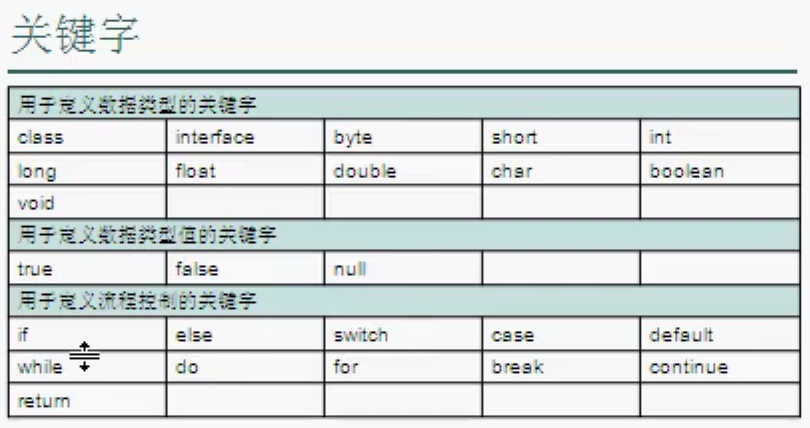
[注释 5](#_Toc124087462)

[常量 6](#_Toc124087463)

## Day01

### 关键字

1. 关键字概述：被java语言赋予特定含义的单词
2. 关键字特点：组成关键字的字母全部小写
3. 关键字注意事项
   1. goto和const作为保留字存在，目前并不使用
   2. 针对关键字有特别的颜色标记，非常直观





|  |
| --- |
| package Day01;  /\*  2023年1月8日11点54分  关键字测试  结果：所有关键字都会有颜色高亮  \*/  public class KeyWordDemo {  public static void main(String[] args) {  System.out.println();  }  } |

### 标识符

1. 就是给类，接口，方法，变量等起名字时使用的字符序列
2. 组成规则
   1. 英文大小写字母
   2. 数字字符
   3. $和\_
3. 注意事项
   1. 不能以数字开头
   2. 不能是java中的关键字
   3. 区分大小写
4. 常见命名规则：见名知意
   1. 包：其实就是文件夹，用于解决相同类名问题

单级：小写

多级：多个小写单词用.隔开

* 1. 类或接口

一个单词：单词的首字母必须大写 Student, Dog

多个单词：每个单词的首字母必须大写 HelloWorld, GoodMoring

* 1. 方法和变量

一个单词：单词的首字母小写 main

多个单词：从第二个单词开始，每个单词的首字母大写 studentAge, showAllNames()

* 1. 常量

一个单词：全部大写 PI

多个单词：每个字母都大写，用\_隔开 STUDENT\_MAX\_AGE

|  |
| --- |
| package Day01;  /\*  2023年1月8日  标识符测试  \*/  public class Identifier {  public static void main(String[] args) {  //正确做法  int x = 100;  //不能以数字开头  //int 1y = 100;  //不能是java中的关键字  //int public = 100;  }  } |

### 注释

1. 用于解释说明程序的文字
2. java中注释说明程序的文字
3. Java中注释分类 格式

单行注释：//注释文字

多行注释：/\*注释文字\*/

文档注释：/\*\*注释文字\*/

1. 注意
   1. 多行不可以嵌套使用，而单行可以
2. 注释的作用
   1. 解释说明程序，提高程序的阅读性
   2. 可以帮助我们调试程序

|  |
| --- |
| package Day01.注释;  /\*  2023年1月8日  测试注释  \*/  //这是我的注释案例  public class Comment {  public static void main(String[] args) {  System.out.println("好好学习，天天向上");  }  } |
| package Day01.注释;  /\*  需求：我准备写一个Java程序，把“HelloWorld”这句话输出在控制台  分析：  A：要写一个Java程序，必须定义类  B：把数据能够输出，说明我们的程序是可以独立运行的，而程序要独立运行，必须定义main方法  C：把数据输出在控制台，必须使用输出语句  实现：  A：java语言提供了一个关键字：class用来定义类，后面跟的是类名  B：main方法的格式是固定的  public static void main(String[] args){  }  C: 输出语句的格式是固定的  System.out.println("helloworld");  "helloworld"这个内容是可以改变的  \*/  //这是我的HelloWorld案例  public class HelloWorld {  /\*  为了程序能够独立运行，定义main方法  main方法是程序的入口  被jvm自动调用  \*/  public static void main(String[] args) {  //为了把数据显示在控制台，我们就使用了输出语句  System.out.println("HelloWorld");  }  } |

### 常量

1. 常量概述：在程序执行的过程中其值不可以发生改变
2. Java中常量分类
   1. 字面值常量

字符串常量：用双引号括起来的内容 “hello” “world”

整数常量：所有的整数

小数常量：所有的小数

字符常量：用单引号括起来的单个字符

布尔常量：true false

空常量：null

* 1. 自定义常量

|  |
| --- |
| package Day01.常量;  /\*  2023年1月8日  测试常量  \*/  public class Constant {  public static void main(String[] args) {  //字符串常量：用双引号括起来的内容 “hello” “world”  System.out.println("hello");  //整数常量：所有的整数  System.out.println(100);  //小数常量：所有的小数  System.out.println(100.10);  //字符常量：用单引号括起来的单个字符  System.out.println('a');  System.out.println('A');  System.out.println('0');  //这个是有问题的  //System.out.println('ab');  //布尔常量：true false  System.out.println(true);  System.out.println(false);  //空常量：null  }  } |

1. 针对整数提供了四种表现形式
   1. 二进制：以0b开头
   2. 八进制：以0开头
   3. 十进制：默认是十进制的
   4. 十六进制：以0x开头

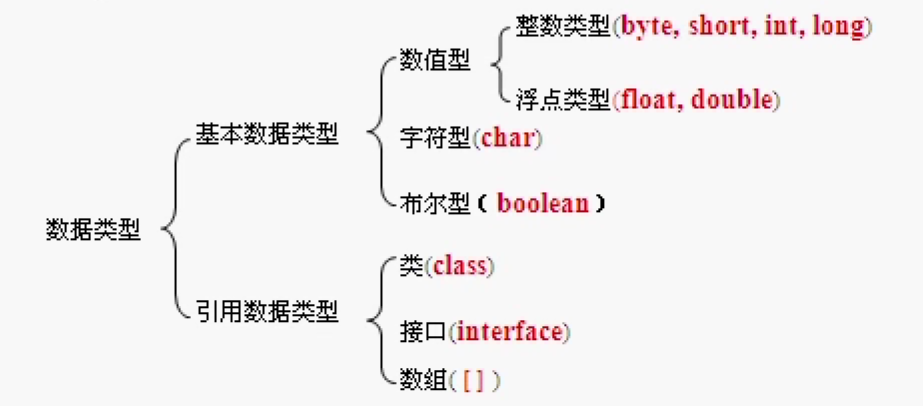
|  |
| --- |
| package Day01.常量;  /\*  测试进制  \*/  public class Constant2 {  public static void main(String[] args) {  //默认十进制  System.out.println(100);  //二进制  System.out.println(0b100);  //八进制  System.out.println(0100);  //十六进制  System.out.println(0x100);  }  } |

### 变量

1. 在程序执行的过程中，在某个范围内其值可以发生改变的量
2. 定义格式
   1. 数据类型 变量名 = 初始化值;
   2. 数据类型 变量名;

变量名 = 值;

1. 变量的组成规则
   1. 必须用对其进行限定（数据类型）
   2. 我们在运算的时候，不可能是拿着这个空间去运算，我们真正运算时使用的是该空间的值，我们就给该空间起了一个名字（变量名）
   3. 即使你有数据类型了，你有变量名了，但是如果没有值，这个空间是一个垃圾空间，没有任何意义（初始化值）
2. 数据类型：java是强类型语言，对于每一种数据都定义了明确的具体数据类型，在内存中分配了不同大小的内存空间

  
数据类型 占用字节数

byte 1

short 2

int 4

long 8

float 4

double 8

char 2

Boolean 1

整数默认int

小数默认double

长整型后缀用L/l标记，建议使用L

单精度负担书用F/f标记，建议用F

byte short在定义的时候，他们接收的其实是一个int类型的值。这个是自己做了一个数据转换

|  |
| --- |
| package Day01.变量和数据类型;  /\*  2023年1月8日  测试变量  \*/  public class DataTypeDemo {  public static void main(String[] args) {  //定义一个字节变量  byte b = 10;  System.out.println(10);  System.out.println(b);  //定义一个短整型变量  short s = 100;  System.out.println(s);  //定义一个整型变量  int i = 1000;  System.out.println(i);  //超过了int的范围  //int j = 100000000000000;//超过了int的范围  long j = 100000000000000L;  System.out.println(j);  //定义浮点数据变量  float f = 12.345f;  System.out.println(f);  double d = 12.345;  System.out.println(d);  //定义布尔变量  boolean flag = true;  System.out.println(flag);  }  } |

1. 使用变量的注意事项
   1. 作用域：变量定义在哪一级大括号中，哪个大括号的范围就是这个变量的作用域。相同的作用域中不能定义两个同名变量
   2. 初始化值：没有初始化值不能直接使用，推荐在定义的时候就给值
   3. 在一行上建议只定义一个变量

|  |
| --- |
| package Day01.变量和数据类型;  /\*  2023年1月8日  使用变量时的注意事项  \*/  public class DataTypeDemo2 {  public static void main(String[] args) {  //相同的作用域中不能定义两个同名变量  int x = 100;  //int x = 200;  //没有初始化值的变量不能直接使用  int y;//使用变量之前必须给值  //System.out.println(y);  int z;  z = 100;    //在一行上定义多个变量  //int a = 10; int b = 20; int c = 30;  int a = 10;  int b = 20;  int c = 30;    int d, e;  d = 40;  e = 50;    //int f, int g; //错误  int h; int i;  }  } |

1. 格式转换
   1. 在运算的时候，要求参与运算的数据类型必须一致
   2. Boolean类型不能转换为其他的数据类型
   3. 默认转换（从小到大）

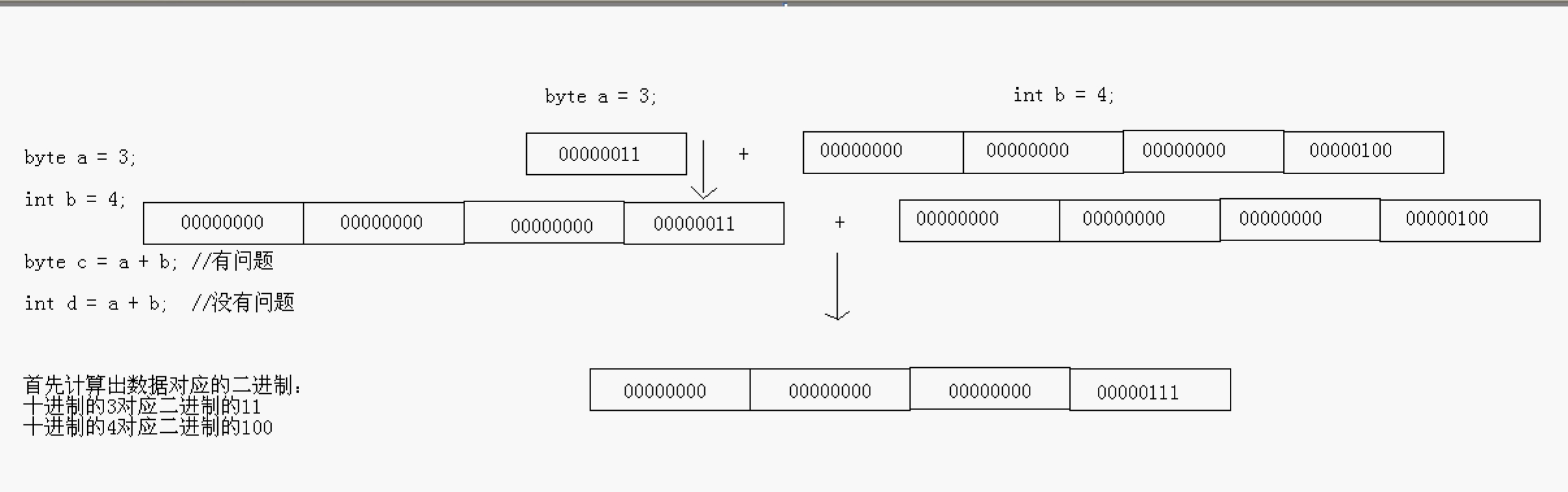
byte, short, char-int-long-float-double

byte, short, char相互之间不转换，他们参与运算首先转换为int类型

long和float底层的存储结构不同，float表示的数据范围比long的范围要大

java中的char类型可以存储一个中文汉字，因为char类型占两个字节，采用unicode编码

|  |
| --- |
| package Day01.变量和数据类型;  /\*  2023年1月8日  默认转换  \*/  public class DataTypeDemo2 {  public static void main(String[] args) {  //相同的作用域中不能定义两个同名变量  int x = 100;  //int x = 200;  //没有初始化值的变量不能直接使用  int y;//使用变量之前必须给值  //System.out.println(y);  int z;  z = 100;  //在一行上定义多个变量  //int a = 10; int b = 20; int c = 30;  int a = 10;  int b = 20;  int c = 30;  int d, e;  d = 40;  e = 50;  //int f, int g; //错误  int h; int i;  }  } |



1. 强制转换：从大的数据类型到小的数据类型
   1. 格式：目标数据类型 变量 = （目标数据类型）（被转换的数据）；
   2. 不要随意的去使用强制转换，因为它隐含了精度损失问题

|  |
| --- |
| package Day01.变量和数据类型;  /\*  2023年1月10日  强制转换  \*/  public class DataTypeDemo4 {  public static void main(String[] args) {  byte a = 3;  int b = 4;  //这个肯定没有问题  //int c = a + b;  //byte c = 7;  //这个是有问题的  //byte c = a + b;  //用强制类型转换  byte c = (byte) (a + b);  }  } |
| package Day01.变量和数据类型;  /\*  2023年1月10日  思考题1：请问下面这个有没有问题  double d = 12.345;  float f = d;  可能损失精度  思考题2：看看下面两个定义有没有区别呢？  float f1 = (float)12.345;  float f2 = 12.345f;  f1其实是一个通过double类型专注拿过来的  f2本身就是一个float类型  \*/  public class DataTypeDemo5 {  public static void main(String[] args) {  //把double赋值给float，加了强制类型转换  double d = 12.345;  float f = (float)d;  //看看下面两个定义有没有什么区别呢  float f1 = (float)12.345;  float f2 = 12.345F;  }  } |
| package Day01.变量和数据类型;  /\*  面试题：  byte b1 = 3, b2 = 4, b;  b = b1 + b2;  b = 3+4;  哪句是编译失败的呢？为什么呢？  b = b1 + b2;是有问题的  因为变量相加，会首先看类型问题，最终把结果赋值的时候也会考虑类型问题  常量相加，会首先做加法，然后看结果是否在赋值的数据类型范围内，如果不是，才报错  \*/  public class DataTypeDemo6 {  public static void main(String[] args) {  byte b1 = 3, b2 = 4, b;  //b = b1 + b2;//这个是类型提升，所以有问题  b = 3 + 4;//常量：先把结果计算出来，然后看是否在byte范围内，如果在就不报错    }  } |
| package Day01.变量和数据类型;  /\*  byte b = 130; 有没有问题？如果我想让赋值正确，可以怎么做？结果是多少呢？  练习：byte b = 300;  因为byte的范围是：-128到127  而130不在范围内，所以报错  \*/  public class DataTypeDemo7 {  public static void main(String[] args) {  //我们可以使用强制类型转换  byte b = (byte) 130;  System.out.println(b);  }  } |

1. 字符参加运算：查找ASCII里面的值

‘a’ 97

‘A’ 65

‘0’ 48

|  |
| --- |
| package Day01.变量和数据类型;  /\*  看程序写结果  通过字符和一个整数相加，我们给出一张表：ASCII码表  通过看完这张表以后，我们要记住三个值  'a' 97  'A' 65  '0' 48  \*/  public class DataTypeDemo8 {  public static void main(String[] args) {  //直接输出一个字符  System.out.println('a');  //输出一个字符和一个整数做加法  System.out.println('a' + 1);  }  } |

1. 字符串参加运算：这里其实是字符串的连接

|  |
| --- |
| package Day01.变量和数据类型;  /\*  2023年1月10日  字符串加法  \*/  public class DataTypeDemo9 {  public static void main(String[] args) {  System.out.println("hello" + 'a' + 1);  System.out.println('a' + 1 + "hello");  System.out.println("5+5=" + 5 + 5);  System.out.println(5 + 5 + "=5+5");  }  } |

## Day03

### 运算符

1. 就是对常量和变量进行操作的符号
2. 算数运算符：+ - \* / % ++ --
   1. 整数相除只能得到整数。如果想得到小数，必须把数据变化为浮点数类型
   2. /获取的是操作1商，%获取的是除法操作的余数

|  |
| --- |
| package Day03.算术运算符;  /\*  2023年1月11日  测试+-&/和%  \*/  public class OperatorDemo {  public static void main(String[] args) {  //定义变量  int x = 3;//把3赋值给int类型的变量x  int y = 4;  System.out.println(x + y);  System.out.println(x - y);  System.out.println(x \* y);  System.out.println(x / y);//整数相除只能得到整数  //我就想得到小数  //只需要把操作的数据中任意的一个数据变为浮点数  System.out.println(x\*1.0/y);  //%的应用  System.out.println(x % y);//得到的余数  }  } |

* 1. ++，--操作符的使用

作用：就是对变量进行自增1或者自减1

单独使用：放在操作数的前面和后面效果一样

参与运算使用：

放在操作数的前面，先自增或者自减，然后再参与运算

放在操作数的后面，先参与运算，再自增或者自减

|  |
| --- |
| package Day03.算术运算符;  /\*  2023年1月11日  测试自增自减运算符  \*/  public class OperatorDemo2 {  public static void main(String[] args) {  //定义两个变量  int x = 3;  int y = 4;  System.out.println("x:" + x);  System.out.println("y:" + y);  //单独使用  x++;  y--;  System.out.println("x:" + x + ",y:" + y);  //意外类型，常量是不可以这样做的  //System.out.println(10++);  System.out.println("------------------");  int a = 3;  int b = 4;  int c = a++;  int d = b--;  System.out.println("a:"+a); //4  System.out.println("b:"+b); //3  System.out.println("c:"+c); //3  System.out.println("d:"+d); //4  c = ++a;  d = ++b;  System.out.println("a:"+a); //4  System.out.println("b:"+b); //3  System.out.println("c:"+c); //4  System.out.println("d:"+d); //3  }  } |
| package Day03.算术运算符;  /\*  练习：算术运算符++--  \*/  public class OperatorTest {  public static void main(String[] args) {  //第一题  int a = 10;  int b = 10;  int c = 10;  a = b++;//a = 10 b = 11 c = 10  c = --a;//a = 9 b = 11 c = 9  b = ++a;//a = 10 b = 10 c = 9  a = c--;//a = 9 b = 10 c = 8  System.out.println("a:" + a);  System.out.println("b:" + b);  System.out.println("c:" + c);  //第二题  int x = 4;  int y = (x++) + (++x) + (x \* 10);  //x = 6, y = 70  }  } |

* 1. +的用法

加法

正号

字符串连接符

|  |
| --- |
| package Day03.算术运算符;  /\*  2023年1月11日  测试+的三种用法  \*/  public class OperatorDemo3 {  public static void main(String[] args) {  //加法  System.out.println(3 + 4);  System.out.println('a');  System.out.println('a'+1);  //正号  System.out.println(+4);  //字符串连接符  System.out.println("hello" + 'a' + 1);  System.out.println('a' + 1 + "hello");  }  } |

1. 赋值运算符
   1. 基本赋值运算符：=

把右边的数据赋值给左边

* 1. 扩展赋值运算符：+= -= \*= /= %=

+=把左边和右边做加法，然后赋值给左边

-=把左边和右边做减法，然后赋值给左边

\*=把左边和右边做乘法，然后赋值给左边

/=把左边和右边做除法，然后赋值给左边

|  |
| --- |
| package Day03.赋值运算符;  public class OperatorDemo {  public static void main(String[] args) {  //基本赋值运算符  int a, b;  a = b = 10;  System.out.println(a);  System.out.println(b);  System.out.println("--------");  //扩展赋值运算符  int y = 10;  y += 20;  System.out.println(y);  }  } |
| package Day03.赋值运算符;  /\*  2023年1月11日  扩展的赋值运算符和普通运算符的区别  \*/  public class OperatorTest {  public static void main(String[] args) {  short s = 1;  //s = s+1;报错  s += 1;//扩展的赋值运算符其实隐含了一个强制类型转换，等价于s = (s的数据类型)(s+1)  System.out.println(s);  }  } |

1. 比较运算符
   1. == != > >= < <=
   2. 无论操作是简单还是复杂，操作的结果是boolean
   3. ==不能写成=

|  |
| --- |
| package Day03.比较运算符;  /\*  2023年1月11日  测试关系运算符  \*/  public class OperatorDemo {  public static void main(String[] args) {  int x = 3;  int y = 4;  int z = 3;  System.out.println(x == y);//false  System.out.println(x == z);//true  System.out.println((x + y) == (x + z));//false  System.out.println(x != y);  System.out.println(x > y);  System.out.println(x >= y);  System.out.println(x < y);  System.out.println(x <= y);  }  } |

1. 逻辑运算符
   1. 用于连接布尔型表达式或值
   2. 逻辑与&
   3. 逻辑或|
   4. 逻辑异或^
   5. 逻辑非! ：偶数个不改变本身
   6. 短路与&&

和&的区别是：最终结果一样，&&和||具有短路效果

|  |
| --- |
| package Day03.逻辑运算符;  /\*  2023年1月11日  测试&与&&的区别  \*/  public class OperatorDemo2 {  public static void main(String[] args) {  int x = 3;  int y = 4;  //boolean b1 = (x++ == 3 & y++ ==4);//x = 4, y = 5  boolean b1 = (++x == 3 && ++y ==4);  System.out.println("x:" + x);  System.out.println("y:" + y);  System.out.println(b1);  }  } |

* 1. 短路或||

|  |
| --- |
| package Day03.逻辑运算符;  /\*  2023年1月11日  测试逻辑运算符  \*/  public class OperatorDemo {  public static void main(String[] args) {  int a = 3;  int b = 4;  int c = 5;  //&  System.out.println((a > b) & (a > c)); //false & false = false  System.out.println((a > b) & (a < c)); //false & false = false  System.out.println((a < b) & (a > c)); //true & false = false  System.out.println((a < b) & (a < c)); //true & true = true  System.out.println("------------------");  //逻辑或  System.out.println((a > b) | (a > c)); //false | false = false  System.out.println((a > b) | (a < c)); //false | false = false  System.out.println((a < b) | (a > c)); //true | false = true  System.out.println((a < b) | (a < c)); //true | true = true  System.out.println("------------------");  //逻辑异或  System.out.println((a > b) ^ (a > c)); //false ^ false = false  System.out.println((a > b) ^ (a < c)); //false ^ false = true  System.out.println((a < b) ^ (a > c)); //true ^ false = true  System.out.println((a < b) ^ (a < c)); //true ^ true = false  System.out.println("------------------");  //逻辑非  System.out.println(!(a > b));//true  System.out.println(!!(a < b));//true  System.out.println(!!!(a < b));//false  }  } |

1. 位运算符
   1. 注意：要做位运算，首先要把数据转换为二进制
   2. <<左移：左边最高位丢弃，右边补齐
   3. >>右移：最高位是0，左边补0，最高位是1，左边补1
   4. >>>无符号右移：无论最高位是0还是1，左边补齐0

|  |
| --- |
| package Day03.位运算符;  /\*  请用最有效率的方式写出计算2\*8的结果  2\*8 = 2<<3  \*/  public class OperatorDemo3 {  public static void main(String[] args) {  //<< 把<<左边的数据\*2的移动次幂  System.out.println(3 << 2); //3\*2^2 = 3 \* 4 = 12  //>> 把<<左边的数据除以2的移动次幂  System.out.println(24 >> 2); //24/2^2 = 24/4 = 6  System.out.println(24 >>> 2);  System.out.println(-24 >> 2);  System.out.println(-24 >>> 2);  }  }  /\*  计算出3的二进制：11  00000000 00000000 00000000 00000011  (00)00000000 00000000 00000000 00001100  计算出24的二进制：11000  10000000 00000000 00000000 00011000(原码)  11111111 11111111 11111111 11100111(反码)  11111111 11111111 11111111 11101000(补码)  11111111 11111111 11111111 11101000  1111111111 11111111 11111111 111010(00) (补码)  补码：1111111111 11111111 11111111 111010  反码：1111111111 11111111 11111111 111001  原码：0000000000 00000000 00000000 000110  \*/ |

* 1. &与
  2. |或
  3. ^异或

一个数据对另一个数据位异或两次该数本身不变

|  |
| --- |
| package Day03.位运算符;  public class OperatorDemo2 {  public static void main(String[] args) {  int a = 10;  int b = 20;  a = a ^ b ^ b;  b = b ^ b ^ b;  System.out.println(a);  }  } |
| package Day03.位运算符;  /\*  实现两个数的交换  \*/  public class OperatorTest {  public static void main(String[] args) {  int a = 10;  int b = 20;  System.out.println("a:" + a + ", b:" + b);  //方式1：使用第三方变量  int c = a;  a = b;  b = c;  System.out.println("a:" + a + ",b:" + b);  System.out.println("------------------");  //方式2：用位异或实现  //左边：a,b,c  //右边：a ^ b  a = a ^ b;  b = a ^ b;  a = a ^ b;  System.out.println("a:" + a + ",b:" + b);  //方式3：用变量相加的方法  a = a + b;  b = a - b;  a = a - b;  System.out.println("a:" + a + ", b" + b);  //方式4：一句话搞定  b = (a + b) - (a = b);  System.out.println("a:" + a + ", b" + b);  }  } |

* 1. ～取反

& | ^ ~两边是数据做的是位运算，两边是布尔值做的是逻辑运算

|  |
| --- |
| package Day03.位运算符;  /\*  2023年1月12日  测试位运算  \*/  public class OperatorDemo {  public static void main(String[] args) {  int a = 3;  int b = 4;  System.out.println(3 & 4);  System.out.println(3 | 4);  System.out.println(3 ^ 4);  System.out.println(~3);  /\*  分析：因为是位运算，所以我们必须先把数据换算成二进制补码  3的二进制：11  4的二进制：100  00000000 00000000 00000000 00000011  00000000 00000000 00000000 00000100  &位与运算：有0则0  00000000 00000000 00000000 00000011  &00000000 00000000 00000000 00000100  ------------------------------------  00000000 00000000 00000000 00000000  结果是0    |位或运算：有1则1  00000000 00000000 00000000 00000011  |00000000 00000000 00000000 00000100  ------------------------------------  00000000 00000000 00000000 00000111  结果是7  ^位异或运算：相同则0，不同则1  00000000 00000000 00000000 00000011  ^00000000 00000000 00000000 00000100  ------------------------------------  00000000 00000000 00000000 00000111  ~按位取反运算符：0变1，1变0  ~00000000 00000000 00000000 00000011  ------------------------------------  11111111 11111111 11111111 11111100（补码）  10000000 00000000 00000000 00000100（原码）  结果是-4  \*/  }  } |

1. 三目运算符（唯一）
   1. (关系表达式)？表达式1：表达式2
   2. 执行流程

根据比较表达式的计算返回一个true或者false

如果条件为true，运算后的结果是表达式1

如果条件为false，运算后的结果是表达式2

|  |
| --- |
| package Day03.三目运算符;  /\*  2023年1月12日  测试三目运算符  \*/  public class OperatorDemo {  public static void main(String[] args) {  int x = 100;  int y = 200;  //int z = x > y ? x : y;  //int z = x < y ? x : y;  int z = (x == y) ? x : y;  System.out.println("z:" + z);  }  } |
| package Day03.三目运算符;  public class OperatorTest {  public static void main(String[] args) {  //获取两个整数中的最大值  int x = 100;  int y = 200;  int max = x > y ? x : y;  System.out.println("max:" + max);  System.out.println("-----------");  //获取三个整数中的最大值  int a = 10;  int b = 30;  int c = 20;  //分两步  //先比较a,b的最大值  //拿a,b的最大值和c进行比较  int temp = ((a > b) ? a : b);  System.out.println(temp);  max = ((max > c) ? max : c);  System.out.println(max);  //一步搞定  int max2 = (a > b) ? ((a > c) ? a : c) : ((b > c) ? b : c);  //比较两个整数是否相同  int m = 100;  int n = 200;  boolean flag = (m == n) ? true : false;  System.out.println(flag);  }  } |

### 键盘录入

1. 为了让程序的数据更符合开发的数据，我们就加入了键盘录入让程序更灵活一下
2. 那么，我们如何实现键盘数据的录入呢
   1. A：导包 import java.util.Scanner;

在class上面

* 1. B：创建键盘录入对象

Scanner sc = new Scanner(System.in);

* 1. C：通过对象获取数据

int x = sc.nextInt();

|  |
| --- |
| package Day03.键盘录入;  /\*  测试键盘录入  \*/  import java.util.Scanner;  public class ScannerDemo {  public static void main(String[] args) {  //创建键盘录入数据对象  Scanner sc = new Scanner(System.in);  System.out.println("请你输入一个数据：");  int x = sc.nextInt();  System.out.println("你输入的数据是" + x);  }  } |

### 流程控制语句

1. 可以控制程序的执行流程
2. 顺序结构：是程序中最简单最基本的流程控制，没有特定的语法结构，按照代码的先后顺序，一次执行，程序中大多数的代码都是这样执行的

写在前面的先执行，写在后面的后执行

|  |
| --- |
| package Day03.顺序结构;  public class SequentialDemo {  public static void main(String[] args) {  System.out.println("程序开始了");  System.out.println("哈哈哈");  System.out.println("程序结束了");  }  } |

1. 选择结构（分支结构）
   1. 选择结构有特定的语法规则，代码要执行具体的逻辑运算进行判断，逻辑运算的结果有两个所以产生选择，按照不同的选择执行不同的代码
   2. if语句
      1. if(关系表达式){

语句体

}

执行流程：

首先判断关系表达式看其结果是true还是false

如果是true就执行语句体

如果是false就不执行语句体

|  |
| --- |
| package Day03.选择结构;  public class IfDemo {  public static void main(String[] args) {  int x = 10;  if (x == 10) {  System.out.println("x 等于 10");  }  if (x == 20) {  System.out.println("x 等于 20");  }  System.out.println("over");  }  } |

注意事项：

关系表达式无论简单还是复杂，结果必须是boolean类型

if语句控制的语句体如果是一条语句，大括号可以省略；如果是多条语句，就不能省略。建议永远不要省略

有左大括号就没有分号，有分号就没有左大括号

|  |
| --- |
| package Day03.选择结构;  public class IfDemo2 {  public static void main(String[] args) {  //关系表达式无论简单还是复杂，结果必须是boolean类型  int x = 10;  if (x == 10) {  System.out.println("x等于10");  }  if (x > 5 || x == 10) {  System.out.println("x大于或等于10");  }  System.out.println("------------------------------");  //if语句控制的语句体如果是一条语句，大括号可以省略；如果是多条语句，就不能省略。建议永远不要省略  int a = 100;  if (a == 100) {  System.out.println("a的值是100");  }  }  } |

* + 1. if (关系表达式){

语句体1;

} else {

语句体2;

}

执行流程：

首先判断关系表达式看其结果是true还是false

如果是true就执行语句体1

如果是false就执行语句体2

注意：else后面是没有比较表达式的，只有if后面有

|  |
| --- |
| package Day03.选择结构;  public class IfDemo3 {  public static void main(String[] args) {  //判断两个数据是否相等  int a = 10;  int b = 20;  if (a == b) {  System.out.println("a等于b");  } else {  System.out.println("a不等于b");  }  }  } |
| package Day03.选择结构;  /\*  if语句格式2的练习  获取两个数据中较大的值  判断一个数据是奇数还是偶数  \*/  import java.util.Scanner;  public class IfTest {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  //获取两个数据中较大的值  System.out.println("请输入第一个数据：");  int a = sc.nextInt();  System.out.println("请输入第二个数据：");  int b = sc.nextInt();  //定义一个变量接收最大值  int max;  if (a > b) {  max = a;  } else {  max = b;  }  System.out.println("max:" + max);  System.out.println("--------------");  //判断一个数据是奇数还是偶数  System.out.println("请输入你要判断的数据");  int x = sc.nextInt();  if (x % 2 == 0) {  System.out.println(x + "这个数据是偶数");  } else {  System.out.println(x + "这个数据是奇数");  }  }  } |

三元运算符实现的，都可以用if语句实现，反之不成立

什么时候if语句实现不能用三元改进呢？

当if语句控制的操作是一个输出语句的时候就不能

为什么呢？因为三元运算符是一个运算符，运算符操作完毕就应该有一个结果，而不是一个输出

* + 1. if(关系表达式1){

语句体1;

} else if(关系表达式2) {

语句体2;

} else {

语句体3;

}

执行流程

首先判断关系表达式1看其结果是true还是false

如果是true就执行语句体1

如果是false就继续判断关系表达式2看其结果是true还是false

如果是true就执行语句体2

如果是false就继续判断关系表达式...看其结果是true还是false

如果没有任何关系表达式为true,就执行语句体n+1。

|  |
| --- |
| package Day03.选择结构;  /\*  2023年1月24日  测试第三种if语句的格式  需求：键盘录入一个成绩，判断并输出成绩的等级  \*/  import java.util.Scanner;  public class IfDemo5 {  public static void main(String[] args) {  //需求：键盘录入一个成绩，判断并输出成绩的等级  /\*  90-100优秀  80-90好  70-80良好  60-70及格  0-60不及格  \*/  Scanner sc = new Scanner(System.in);  //录入数据  System.out.println("请输入你的考试成绩：");  int score = sc.nextInt();  if (score >= 90 && score < 100) {  System.out.println("优秀");  } else if (score >= 80 && score < 90) {  System.out.println("好");  } else if (score >= 70 && score < 80) {  System.out.println("良");  } else if (score >= 60 && score < 70) {  System.out.println("及格");  } else {  System.out.println("不及格");  }  //这样写已经满足我的基本要求，但是可能别人在使用的时候，不会按照你要求的数据给出了  //在左一个程序的基本测试的时候，一定要考虑这样的几个问题  //正确数据，错误数据，边界数据  //而我们刚才写的数据并没有处理错误数据，这个恒旭不是很好，要改进  System.out.println("请输入你的考试成绩：");  score = sc.nextInt();  if (score >= 90 && score <= 100) {  System.out.println("优秀");  } else if (score >= 80 && score < 90) {  System.out.println("好");  } else if (score >= 70 && score < 80) {  System.out.println("良");  } else if (score >= 60 && score < 70) {  System.out.println("及格");  } else if (score >= 0 && score < 60) {  System.out.println("不及格");  } else {  System.out.println("你输入的成绩有误");  }  }  } |
| package Day03.选择结构;  /\*  三种if语句分别适合做什么事情呢？  格式1：适合做单个判断  格式2：适合做两个判断  格式3：适合做多个判断  需求：键盘录入x的值，计算出y的值并输出  x>=3 y=2x+1;  -1<=x<3 y=2x;  x<=-1 y=2x-1;  分析：  A：由于数据要键盘录入，所以必须使用Scanner  B：由于是三种判断，所以我们选择if语句格式3  \*/  import java.util.Scanner;  public class IfTest2 {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  System.out.println("请输入x的值：");  int x= sc.nextInt();  int y;  //用if语句格式3进行判断  if (x >= 3) {  y = (2 \* x) + 1;  } else if (x >= -1 && x < 3) {  y = 2 \* x;  } else {  y = (2 \* x) - 1;  }  System.out.println("y:" + y);  }  } |
| package Day03.选择结构;  /\*  键盘录入月份的值，输出对应的季节  春：3，4，5  夏：6，7，8  秋：9，10，11  冬：12，1，2  分析  A：键盘录入月份的值，所以我们要使用Scanner  B：我们应该判断这个月份在哪个季节，而这个判断情况较多，所以用if语句格式3  \*/  import java.util.Scanner;  public class IfTest3 {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  System.out.println("请输入一个月份");  int month = sc.nextInt();  //第三种格式实现即可  if (month < 1 || month > 12) {  System.out.println("你输入的月份有误");  } else if (month == 1) {  System.out.printf("冬季");  }else if (month == 2) {  System.out.printf("冬季");  }else if (month == 3) {  System.out.printf("春季");  }else if (month == 4) {  System.out.printf("春季");  }else if (month == 5) {  System.out.printf("春季");  }else if (month == 6) {  System.out.printf("夏季");  }else if (month == 7) {  System.out.printf("夏季");  }else if (month == 8) {  System.out.printf("夏季");  }else if (month == 9) {  System.out.printf("秋季");  }else if (month == 10) {  System.out.printf("秋季");  }else if (month == 11) {  System.out.printf("秋季");  }else if (month == 12) {  System.out.printf("冬季");  }  System.out.println("---------------------");  //这个程序确实是符合了我们的需求，但是看起来比较麻烦  //可以用逻辑运算符给他们连接起来改进  if (month < 1 || month > 12) {  System.out.println("宁输入的月份有误");  } else if (month == 3 || month == 4 || month == 5) {  System.out.println("春季");  } else if (month == 6 || month == 7 || month == 8) {  System.out.println("夏季");  } else if (month == 9 || month == 10 || month == 11) {  System.out.println("秋季");  } else {  System.out.println("冬季");  }  //继续改进  if (month < 1 || month > 12) {  System.out.println("宁输入的月份有误");  } else if (month >= 3 && month <= 5) {  System.out.println("春季");  } else if (month >= 6 && month <= 8) {  System.out.println("夏季");  } else if (month >= 9 && month <= 11) {  System.out.printf("秋季");  } else {  System.out.println("冬季");  }  System.out.println("------------");  }  } |

if语句的使用场景：

针对表达式时一个boolean类型的判断

针对一个范围的判断

|  |
| --- |
| package Day03.选择结构;  /\*  2023年1月24日  if语句的嵌套使用  \*/  public class IfTest4 {  public static void main(String[] args) {  int a = 10;  int b = 30;  int c = 20;  //三元实现  int temp = (a > b) ? a : b;  int max = (temp > c) ? temp : c;  System.out.println("max:" + max);  //用if实现  if (a > b) {  if (a > c) {  max = a;  } else {  max = c;  }  } else {  if (b > c) {  max = b;  } else {  max = c;  }  }  }  } |

switch语句