Java技术

第二章 Java语言基础

路　强

luqiang@hfut.edu.cn

合肥工业大学计算机与信息学院

本章学习提示

- 本章的重点是介绍Java的语言基础。

主要包括：

- 数据类型分类，讨论简单数据类型的实例化
- 变量与常量
- 数据类型的优先关系和相互转换规则
- 运算符和表达式
- 程序流程控制

目　录

Java语言的数据类型

- 数据类型
  - 基本数据类型
    - 数值类型
      - 整数类型：byte，short，int，long
      - 浮点数类型：float，double
    - 字符类型：char
    - 布尔类型：boolean
  - 复合数据类型
    - 类类型
    - 接口类型：interface

基本数据类型

- 基本数据类型也称作简单数据类型。

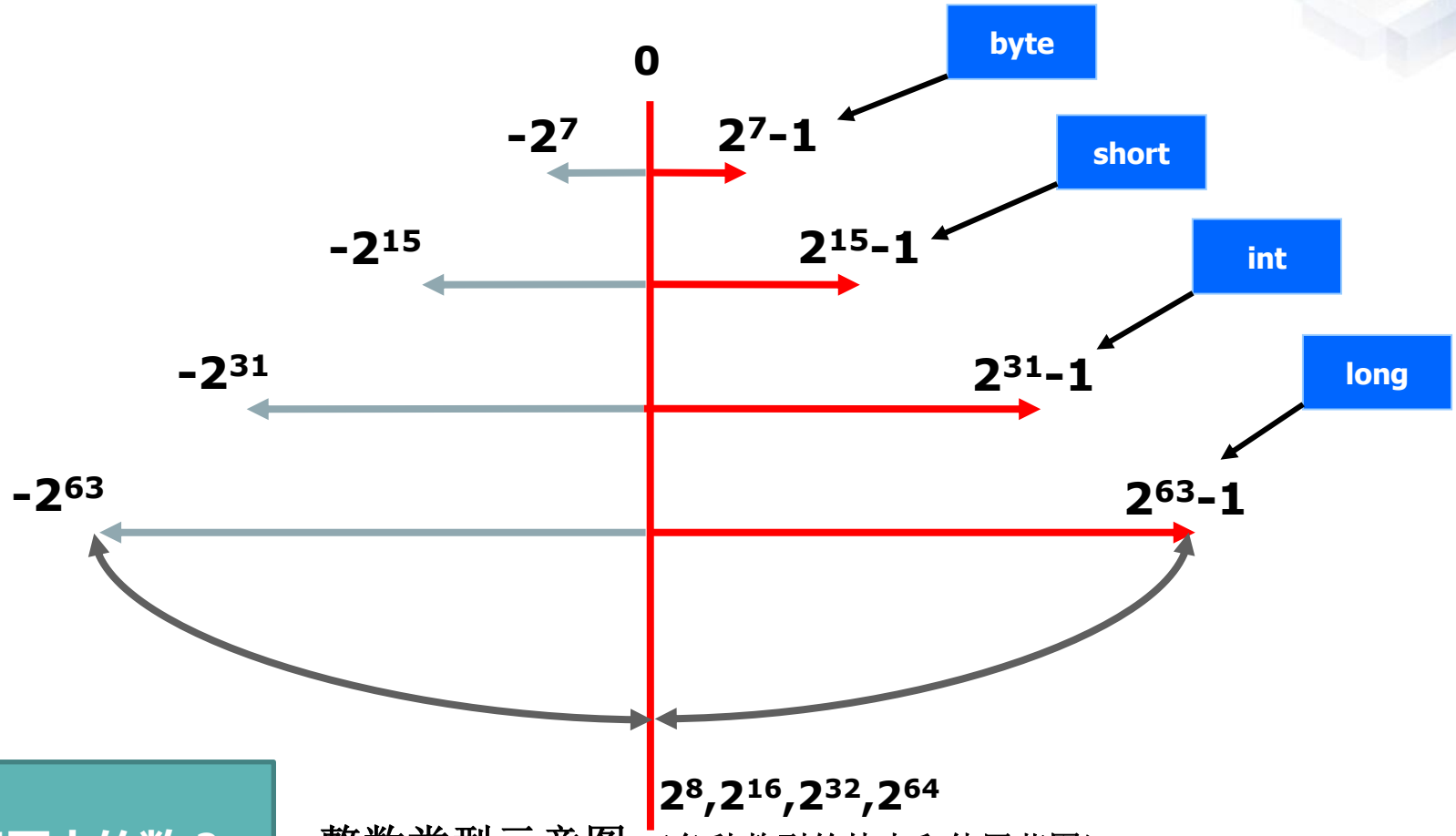
  Java语言有8种简单数据类型，分别是：

  byte 、short 、int、long、float、double、char 、 boolean

- 这8种数据类型习惯上可分为4大类型：
  - 整数类型：byte、short、int、long
  - 浮点类型：float、double
  - 字符类型：char
  - 逻辑类型：boolean

整数类型

- Java编程语言中的整数类型都是有符号整数，不存在无符号整数。(与C和C++不同)



整数类型示意图（各种整型的特点和使用范围）

思考：
如何表示更大的数？

整数类型

- 整型可用十进制、八进制或十六进制表示。 首位为“0”表示八进制的数值；首位为“0x”表示十六进制的数值。

  请看下面的例子：

  - 5　　表示十进制值5
  - 075　　表示八进制数值75（也就是十进制数61）
  - 0x9ABC　　表示十六进制的数值9ABC（也就是十进制数39612）

- 整型，如在其后有一个字母“L”表示一个long常量（也可以用小写“l”）

浮点类型

- Java语言中有两种浮点类型：float和double
  - 如果一个数值常量中包含小数点或指数部分，或者其后跟有字母F或f（float），D或d（double），则该数为浮点数。
  - 如果不明确指明浮点数的类型，浮点数缺省为double类型。

    3.14159　（double型浮点数）

    2.08E25　（double型浮点数）

    6.56f　　（float型浮点数）
  - float为32位（单精度），double为64位（双精度）

字符型

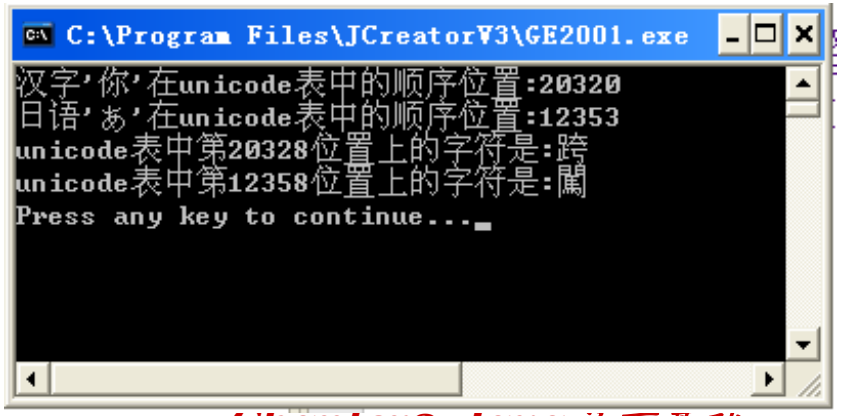
关于字符型

- Java中char是16位的**无符号型数据**
- 内存分配给2个字节，占16位，最高位不用来表示符号
- 字符必须用一对单引号括起来，如‘a’，‘B’等
- Unicode字符集采用双字节对字符进行编码，

  例如：‘A’、‘!’、‘9’、‘好’、‘\t’、‘您’、‘δ’
- 与C语言类似，Java也提供转义字符，以反斜杠（\）开头，将其后的字符转变为另外的含义。

字符型之例

```
1.    public class  example02_01 {
2.       public static void main (String[] args) {
3.          char chinaWord='你',
4.                  japanWord='ぁ';
5.          int  p1=36328,p2=38358;
6.          System.out.println("汉字\'你\'在unicode表中的顺序位置" + (int)chinaWord);
7.          System.out.println("日语\'ぁ\'在unicode表中的顺序位置" + (int)japanWord);
8.          System.out.println("unicode表中第20328位置上的字符是:" + (char)p1);
9.          System.out.println("unicode表中第12358位置上的字符是:" + (char)p2);
10.      }
11.   }
```



字符集 – 编码方式

- 在中国，大陆最常用的就是GBK18030编码，除此之外还有GBK，GB2312
  - GB2312：6763个汉字和682个其它符号
  - 95年GBK1.0：1886个符号
  - GBK18030：收录了27484个汉字，同时还收录了藏文、蒙文、维吾尔文等主要的少数民族文字
  - 按照GBK18030、GBK、GB2312的顺序， 3 种编码是向下兼容，同一个汉字在三个编码方案中是相同的编码

Unicode - 1

- 如果把各种文字编码形容为各地的方言，那么Unicode就是世界各国合作开发的一种语言。

  在这种语言环境下，不会再有语言的编码冲突，在同屏下，可以显示任何语言的内容，这是Unicode的最大好处。

- 那么Unicode是如何编码的呢？其实非常简单。

  就是将世界上所有的文字用 2 个字节统一进行编码。

? 2 个字节最多能够表示65536个编码，够吗？

Unicode - 2

- 韩国和日本的大部分汉字都是从中国传播过去的，字型是完全一样的。
  - 比如：“文”字，GBK和SJIS中都是同一个汉字，只是编码不同而已.
  - 这样统一编码， 2 个字节就已经足够容纳世界上所有的语言的大部分文字.
- 为了防止将来 2 个字节不够用开发了UCS-4，由原先的65536个编码扩展至将近100万编码

字符型 vs 字符串

- 值得注意的一点是，用双引号引用的文字，就是我们平时所说的字符串类型（String），它不是基本类型，而是一个对象类型，Java是将字符串作为对象实现的.
- 为char类型所规定的字符转义序列在字符串内同样适用，如“three\nlines”等.

逻辑类型

- 逻辑类型
  - 常量：**true ， false**
  - 变量的定义

    使用关键字boolean来定义逻辑变量,

    定义时也可以赋给初值：

    boolean x=true，tom=false，jiafei;
  - 小写（与C++不同）

基本数据 vs. 数据类型类

- 对于每一种基本数据类型，Java分别提供相应的类对其进行封装，以便提供更强大的数据处理能力。

```
int a = Integer.parseInt("3");

Float f = Float.MAX_VALUE;
```

基本 数据类型	数据 类型类
byte	Byte
short	Short
int	Integer
long	Long
char	Char
boolean	Boolean
float	Float
double	Double

复合数据类型

复合数据类型：

- 用户定义的、由一系列简单数据类型及其运算符合而成。
- （类和接口）——引用模型

目　录

类型转换

- 各类型所占用的位数从短到长依次为：

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

- 简单数据类型之间的转换又可以分为：
  - 低级到高级的自动类型转换
  - 高级到低级的强制类型转换

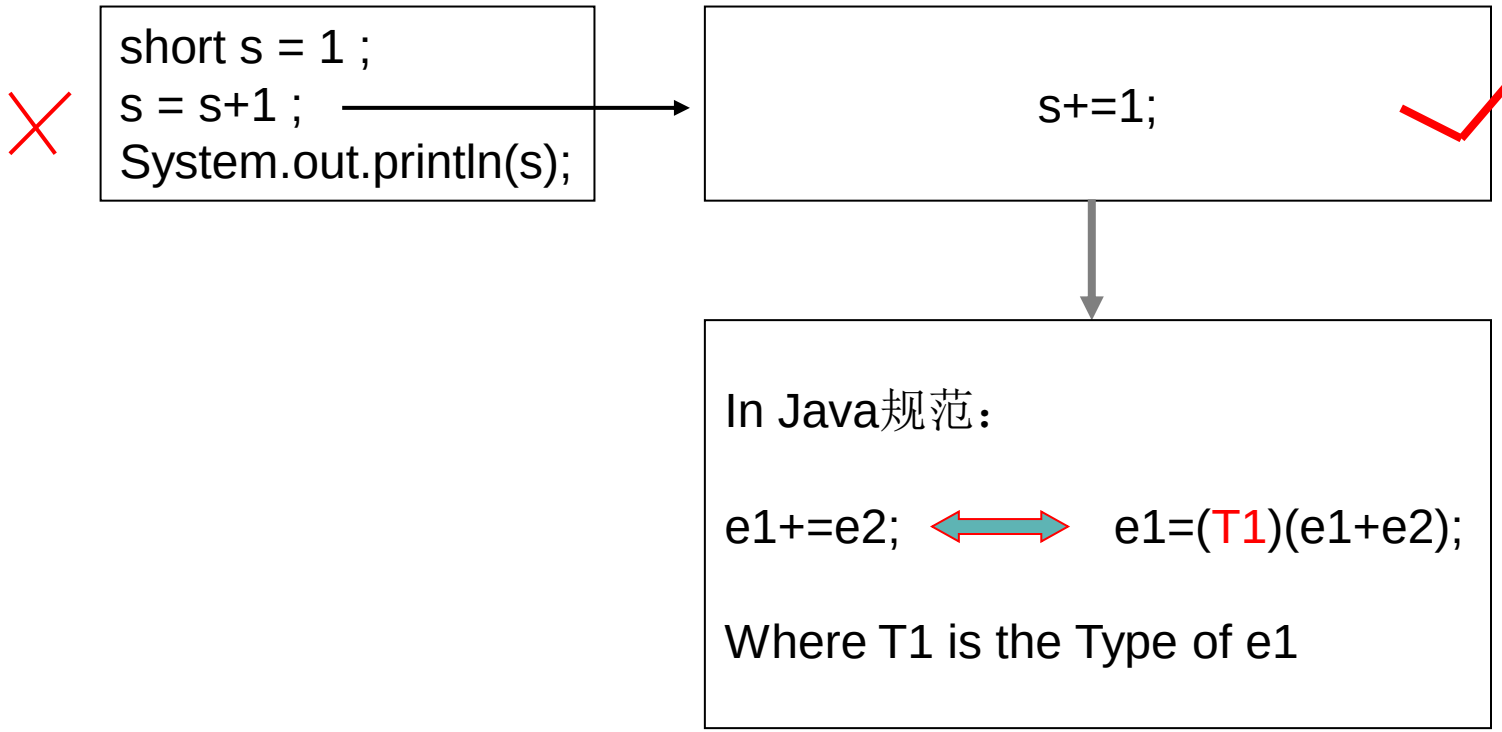
自动类型转换

- Java中整型、字符型和浮点型数据可以互相自动转换（低级向高级），混合运算。

操作数1类型	操作数2类型	计算结果
byte或short	int	int
byte或shart或int	long	long
byte或shart或int或long	float	float
byte或shart或int或long或float	double	double
char	int	int

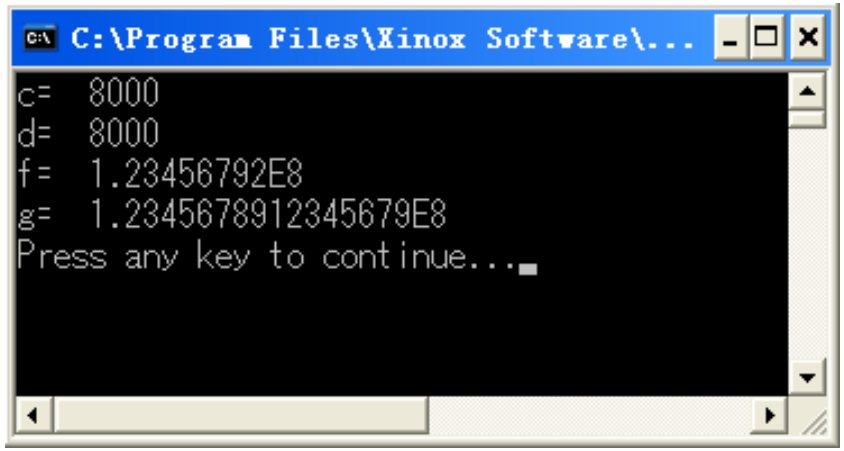
强制数据类型转换

- 由低级向高级可以自动转换，但是由高级向低级转换必须显示说明、进行强制转换。



高级变量转换为低级变量之例


```
1.  //简单类型强制转换
2.  public class example02_02{
3.    public static void main (String args[ ]){
4.      int c=2200;
5.      long d=8000;
6.      float f;
7.      double g=123456789.123456789;
8.      c=(int)d;
9.      f=(float)g;   //导致精度的损失.
10.     System.out.println("c=  "+c);
11.     System.out.println("d=  "+d);
12.     System.out.println("f=  "+f);
13.     System.out.println("g=  "+g);
14.   }
15. }
```



目　录

标识符

- 标识符（**identifier**）好像一个人生下来就要给他起个名字一样，作为一种识别记号。计算机中的变量、常量、方法、类等也都是用名字来加以识别的，这种名字就是标识符。
- 起名的方法有一种约定，即由一个字母或者一串以字母开头由字母、数字或符号组成的字符串。

1. 符号只允许**下划线_**和**美元$符号**
2. 名字长度不限
3. 注意英文字母大小写代表不同含义
4. 取名应遵循易于理解、便于记忆的原则

例如：合法标识符 **Lotus_1　wang$**

非法标识符 **A?2　3ab**

保留字

- 保留字(关键字)就是Java语言中已经被赋予特定意义的一些单词
- 不可以把这类词作为名字来用
- 注意
  - Java中**true、false、null**都是**小写**，C++中大写
  - **无 sizeof** 运算符
    Java所有数据类型的长度和表示是**固定**的，与平台无关，不是像在C/C++语言中那样数据类型的长度根据不同的平台而变化。这正是Java语言的一大特点.
  - **goto**和**const**是Java编程语言中保留的**没有意义**的关键字

保留字

abstract	boolean	break	byte	case
catch	char	class	const	continue
do	double	else	extends	false
finally	cast	default	final	finally
float	for	future	generic	goto
if	implements	import	inner	instanceof
int	interface	long	native	new
null	operator	outer	package	private
protected	public	rest	return	short
static	super	switch	synchronized	this
throw	throws	transient	true	try
var	void	volatile	while	

目　录

Java的运算符

- Java语言按运算符对数据的运算结果分类有算术运算符、逻辑运算符、关系运算符、赋值运算符和位运算符；按运算符运算的数据个数分类可分为一元运算符、二元运算符和三元运算符.
- 算术运算符有五种 加 + 减 – 乘 * 除 / 取余% 其中除减号可作为一元运算符外均为二元运算符.
  - 另外两个经常使用的一元运算符是
    - i++ 和 ++i
    - i-- 和 --i

int a=1, b=2;

a+++b = ?

语句与表达式

- 语句是构成程序的基本单位，语句具有独立完整的含义，可以对计算机发出操作命令，每一条语句都必须以分号；作为结束符
- 表达式是用运算符把操作数（变量、常量和方法）连接起来表达某种运算或含义的式子
- 一个表达式可以同时包括多个操作，而操作的顺序由各运算符的优先级及括号来决定。其中运算符是算术运算符称为算术表达式；运算符为逻辑运算符称为逻辑表达式

语句与表达式 - 1

- 语句和表达式的区别与联系
  - 语句是程序的组成部分；表达式是语句的组成部分
  - 语句有结束符；表达式没有结束符
  - 语句是针对程序而言的；表达式是数学上的计算概念
  - 算式的计算结果=号在右边；赋值语句=号在表达式左边

算术混合运算的精度

- 精度从“底”到“高”排列的顺序是：

  byte short int long float double

- Java将按运算符两边的操作元的最高精度保留结果的精度，例如：

  5/2的结果是2，要想得到2.5，必须写成5.0/2或5.0f/2。

- char型数据和整型数据运算结果的精度是int。

  例如：byte x=7;

  那么

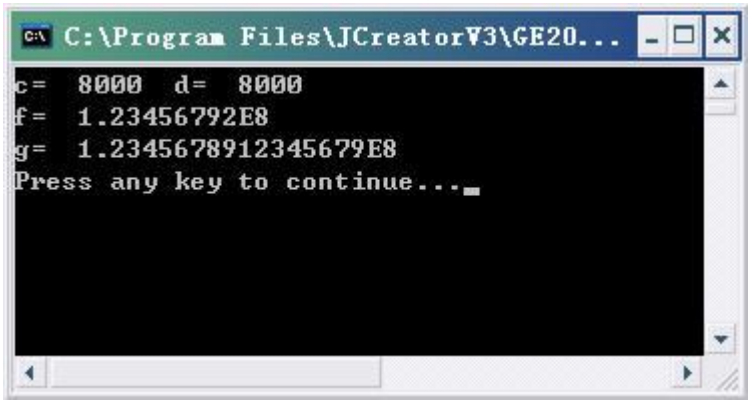
  ‘B’+x;

  的结果是int型.

运算的精度之例

```
1.  //算术运算测试
2.  public class example02_03 {
3.    public static void main(String args[]){
4.      int a=4;
5.      int b=10;
6.      int d=b*++a;
7.      int e=(a/(++b));
8.      System.out.println("a="+a);
9.      System.out.println("b="+b);
10.     System.out.println("d="+d);
11.     System.out.println("e="+e);
12.   }
13. }
```

? a,b,c和d分别输出什么



二元算术运算符、逻辑运算符和位运算符

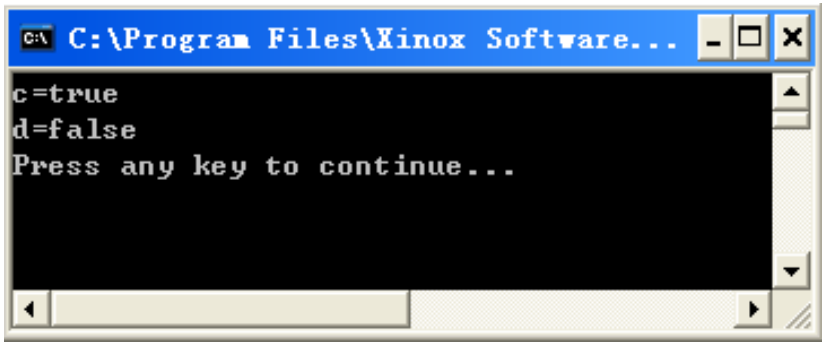
运算符	运算	举例	含义	运算符	运算	举例	含义
+=	加法	x +=y	x=x+y	&=	与	x &=y	x=x&y
-=	减法	x-=y	x=x-y	|=	或	x |=y	x=x|y
*=	乘法	x*=y	x=x*y	^=	异或	x ^=y	x=x^y
/=	除法	x/=y	x=x/y	<<=	左移	x <<=y	x=x<<y
%=	取余	x%=y	x=x%y	>>=	右移	x>>=y	x=x>>y
>>>=	不带符号右移					x >>>=y	x=x>>>y

运算符的优先级

运算符	描述	优先级	结合性
. [ ] ( )	域、数组、括号	1	从左至右
++ -- - ! ~	一元操作符	2	从右至左
* / %	乘、除、取余	3	从左至右
+ -	加、减	4	从左至右
<< >> >>>	位运算	5	从左至右
< <= > >=	逻辑运算	6	从左至右
== !=	逻辑运算	7	从左至右
&	按位与	8	从左至右
^	按位异或	9	从左至右
|	按位或	10	从左至右
&&	逻辑与	11	从左至右
||	逻辑或	12	从左至右
?:	条件运算符	13	从右至左
= *= /= %= += -= ， <<= >>= >>>= &= ^= |=	赋值运算符	14	从右至左

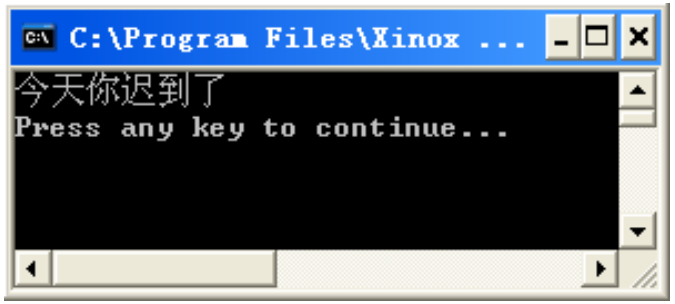
运算例程


```
1.  //通过对给定常数进行大小比较，将比较结果作为逻辑值输出
2.  public class example02_04 {
3.    public static void main(String args[]){
4.      int x=15;
5.      int y=7;
6.      boolean c=(x>y);
7.      boolean d=((float)x/y==(double)x/y);
8.      System.out.println("c="+c);
9.      System.out.println("d="+d);
10.   }
11. }
```



运算例程


```
1.  //通过比较字符串，将字符串比较结果输出
2.  public class example02_05 {
3.    public static void main(String args[]){
4.      int time=10;//上班时间
5.      String week="Monday";
6.      if(week=="Sunday"||week=="Saturday")
7.      System.out.println("今天我休息");
8.      else
9.      if(time>8 && time<12)
10.     System.out.println("今天你迟到了");
11.   }
12. }
```



位运算符

运算符	运算	举例	含 义
~	位反	~x	将x逐位取反
&	位与	x&y	x、y 逐位进行与操作
|	位或	x|y	x、y 逐位进行或操作
^	位异或	x^y	x、y 逐位进行 相同取0 相异取1
<<	左移	x<<y	x向左移动，位数是y
>>	右移	x>>y	x向右移动，位数是y
>>>	不带符号右移	x>>>y	x向右移动，位数是y，空位补0

位运算符

```
~  0 1 0 0 1 1 1 1
-----------------
   1 0 1 1 0 0 0 0
```

```
   0 0 1 0 1 1 0 1
&  0 1 0 0 1 1 1 1
-----------------
   0 0 0 0 1 1 0 1
```

```
   0 0 1 0 1 1 0 1
^  0 1 0 0 1 1 1 1
-----------------
   0 1 1 0 0 0 1 0
```

```
   0 0 1 0 1 1 0 1
|  0 1 0 0 1 1 1 1
-----------------
   0 1 1 0 1 1 1 1
```

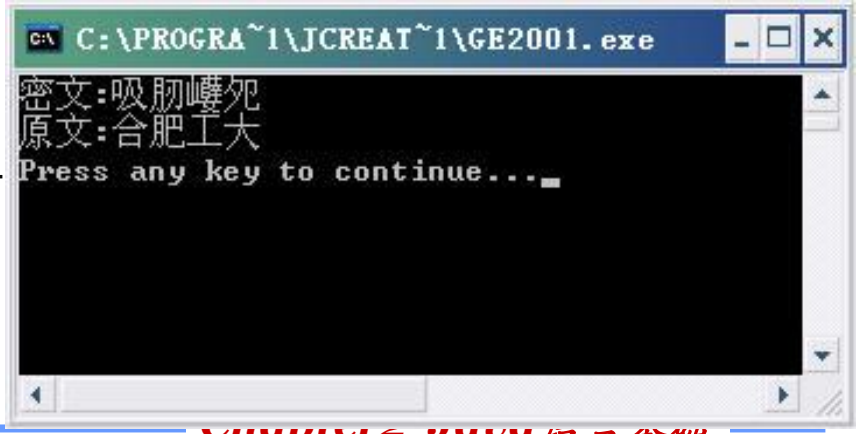
位运算应用

- “清零” —— 和“零”按位“与”
- “乘 2 ” —— 左移“<<”一位
- “除 2 ” —— 右移“>>”一位
- “交换变量” —— 通过按位异或“^”
- “加密”

1. a = 10100001, b = 00000110
2. a = a ^ b; //a = 10100111
3. b = a ^ b; //b = 10100001
4. a = a ^ b; //a = 00000110

位运算应用 – 加密


```
1.  public class example02_06 {
2.    public static void main(String [] args) {
3.      char a1='合',a2='肥',a3='工',a4='大';
4.      char secret='0';
5.      a1=(char)(a1^secret);        //21512 ^ 54
6.                  //(101010000001000 ^ 110110)
7.      a2=(char)(a2^secret);        //32933
8.      a3=(char)(a3^secret);        //24037
9.      a4=(char)(a4^secret);        //22823
10.     System.out.println("密文:"+a1+a2+a3+a4);
11.     a1=(char)(a1^secret);
12.     a2=(char)(a2^secret);
13.     a3=(char)(a3^secret);
14.     a4=(char)(a4^secret);
15.     System.out.println("原文:"+a1+a2+a3+
16.   }
17. }
```



目　录

基本控制结构

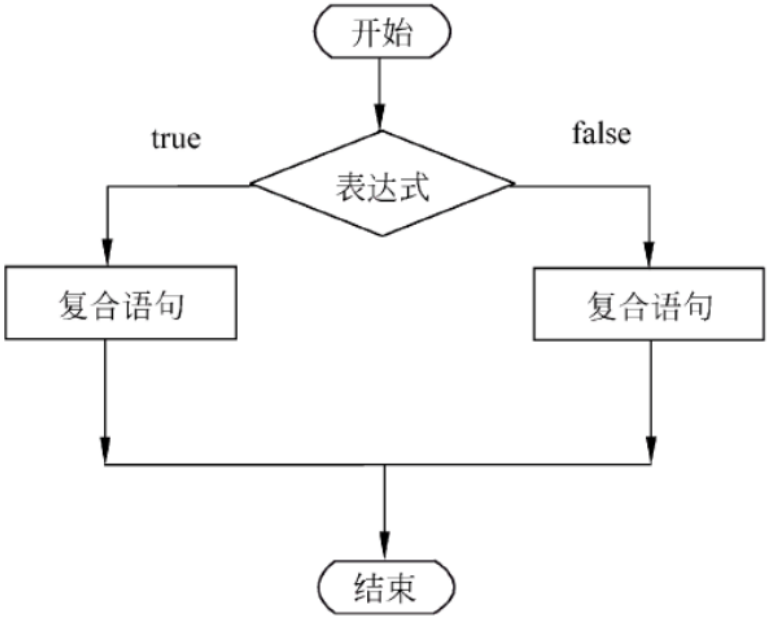
- 顺序结构 就是按照指令的先后顺序依次执行。
- 为实现分支结构程序设计，Java语言提供了条件分支语句 if 和多重分支语句switch，根据它们所包含的逻辑表达式的值决定程序执行的方向。
- 循环结构的程序可以对反复执行的程序段进行精炼，用较少的语句执行大量重复的工作。

  Java提供了for 、while和do-while三种循环语句

分支结构

- 分支结构，是在两条或多条（两条以上）执行路径中选择一条执行的控制结构
- if语句格式

```
1.  if（条件表达式）{
2.          语句组;
3.  }
4.  else {
5.          语句组;
6.  }
```



- 括号中的条件是逻辑表达式，其值为 true 执行语句块(左)，其值为 false 执行语句块(右)。两种情况在完成各自的任务之后，于if的下一条语句汇合。

分支结构


```
1.    //本程序从命令行输入两个数据进行比较，
2.    // 输出较大的那个数
3.    public class example02_07 {
4.        public static void main(String args[]) {
5.            int x,y=0;
6.        //读入命令行参数
7.        x=Integer.parseInt(args[0]);
8.        y=Integer.parseInt(args[1]);
9.        if(x>=y) System.out.println("x="+x);
10.           else System.out.println("y="+y);
11.       }
12.   }
```

该程序总结

- if 的用法
- String类型 args[ ] 的使用
- IDE环境与命令行环境（命令行参数）
- 程序的功能性、完善性、良好用户交互能力

例程 -求三个数中的最大者


```
1.  //给出任意三个数，通过使用if语句嵌套，将它们
2.  //按从小到大排序
3.  import javax.swing.JOptionPane;
4.   public class example02_08 {
5.    public static void main(String args[]){
6.     String str;
7.     double x,y,z,t;
8.     str=JOptionPane.showInputDialog("请输入第一个数");
9.     x=Double.parseDouble(str);
10.    str=JOptionPane.showInputDialog("请输入第二个数");
11.    y=Double.parseDouble(str);
12.    str=JOptionPane.showInputDialog("请输入第三个数");
13.    z=Double.parseDouble(str);
14.    if (x>y){
15.       t=x; x=y; y=t;
16.       if (x>z){
17.          t=x; x=z; z=t;
18.       }
19.    else
20.       if (y>z){
21.          t=y; y=z; z=t;}
22.       }
23.       else
24.          if (x>z) {
25.             t=x;x=z;z=t;
26.             if (y>z) {
27.                t=y;y=z;z=t;
28.             }
29.          }
30.    if (y>z) {
31.       t=y;y=z;z=t;
32.    }
33.    System.out.println("最小值＝"+x);
34.    System.out.println("中间值＝"+y);
35.    System.out.println("最大值＝"+z);
36.    System.exit(0);
37.   }
38.  }
```

switch 语句

```
1.    switch(表达式) {
2.        case 值1：  语句块1；
3.                    break；
4.        case 值2：  语句块2；
5.                    break；
6.        case 值3：  语句块3；
7.                    break；
8.        ……
9.        case 值n：  语句块n；
10.                   break；
11.       default:
12.                   语句块n+1;
13.   }
```

1. switch 语句中的表达式的数据类型可以是 byte、char、short、int 类型。
2. 根据表达式值与case语句后面的匹配情况决定程序执行的分支
3. 每个case 语句都要有break语句
4. 不匹配的情况执行 default 语句

switch 语句之例


```
1.    //通过键盘输入月份，输出相应月份的英文单词
2.    //注意月份是在命令行中给出的
3.    public class example {
4.      public static void main(String args[]){
5.        int month;
6.        month=Short.parseShort(args[0]);
7.        switch(month){
8.          case 1: System.out.println("January"); break;
9.          case 2: System.out.println("February"); break;
10.         case 3: System.out.println("March"); break;
11.         case 4: System.out.println("April"); break;
12.         case 5: System.out.println("May"); break;
13.         case 6: System.out.println("June"); break;
14.         case 7: System.out.println("July"); break;
15.         case 8: System.out.println("August"); break;
16.         case 9: System.out.println("September"); break;
17.         case 10: System.out.println("October"); break;
18.         case 11: System.out.println("Novenber"); break;
19.         case 12: System.out.println("December"); break;
20.         default: System.out.println("Input Number Error!!");break;}
21.       }
22.   }
```

下面代码会产生什么样的输出?

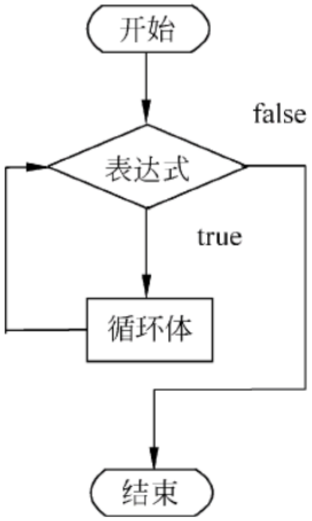

```
1.  public class testswitch {
2.      public static void main(String [] args){
3.              char letter = 'A' ;
4.              switch (letter){
5.                      case 'A':
6.                      case 'a':
7.                              System.out.println("Some kind of A.");
8.                      case 'B':
9.                      case 'b':
10.                             System.out.println("Some kind of B.");
11.                             break;
12.                     default:
13.                             System.out.println("Something else.");
14.                             break;
15.             }
16.     }
17. }
```

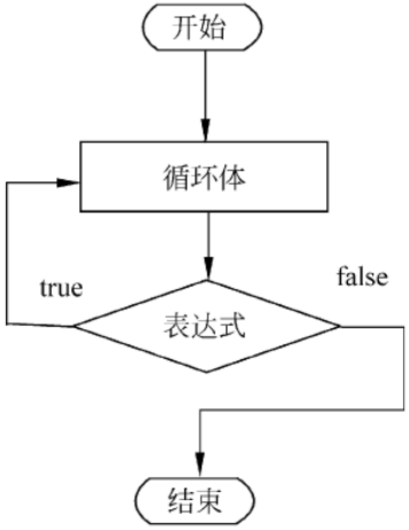
循环结构

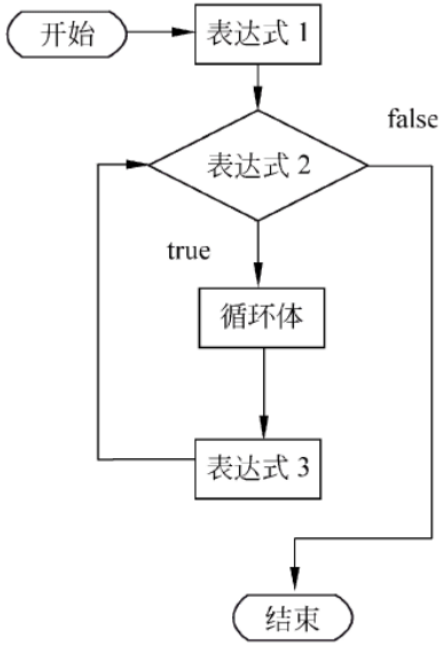
- 循环结构是程序中一种重要的基本结构，是指：

  在一定的条件下反复执行某段程序，被反复执行的这段程序称为“循环体”

- Java中有三种语句来实现循环结构，分别是while，do-while和 for 语句







while语句

- **while语句的格式如下：**

```
1.    while(条件表达式) {
2.        循环体语句;
3.    }
```

- **在循环刚开始时，会计算一次“条件表达式”的值。**
  - 当条件为假时，将不执行循环体，直接跳转到循环体外，执行循环体外的后续语句；
  - 当条件为真时，便执行循环体。每执行完一次循环体，都会重新计算一次条件表达式，当条件为真时，便继续执行循环体，直到条件为假才结束循环.

do-while语句

- do-while语句的格式如下：

```
1.    do {
2.            循环体语句;
3.    } while(条件表达式)
```

- do-while循环与while循环的不同在于：
  - 先执行循环中的语句，然后再判断条件是否为真，如果为真则继续循环
  - 如果为假，则终止循环
  - 因此，do-while循环至少要执行一次循环语句

for语句

- for语句是三个循环语句中功能最强，使用最广泛的一个。for语句的格式如下：

```
1.  for (表达式1; 表达式2; 表达式3) {
2.      循环体语句;
3.  }
```

  - 表达式1一般是一个赋值语句，它用来给循环控制变量赋初值；
  - 表达式2是一个布尔类型的表达式，它决定什么时候退出循环；
  - 表达式3一般用来修改循环变量，控制变量每循环一次后按什么方式变化。
  - 三个部分之间用“;”分开

for语句

- for语句的执行过程：

1. 在循环刚开始时，先计算表达式1，在这个过程中，一般完成的是初始化循环变量或其它变量。
2. 根据表达式2的值来决定是否执行循环体。表达式2是一个返回布尔值的表达式，若该值为假，将不执行循环体，并退出循环；若该值为真，将执行循环体。
3. 执行完一次循环体后，计算表达式3。
   在这个过程中一般会修改循环变量。
4. 转入第（2）步继续执行。

循环嵌套

- 循环嵌套是指在循环体中包含有循环语句的情况。
- 三种循环语句即可以自身进行嵌套，也可以相互进行嵌套构成多重循环。
- 多重循环自内向外展开，即先执行内循环，后执行外循环。多重循环不允许相互交叉。

```
1.  //分别输出1!、2!、3!、...6! 以及它们的和
2.  public class example02_09 {
3.    public static void main(String args[])  {
4.        long sum=0;
5.        for(int i=1;i<=6;i++) {
6.            long m=1;
7.            for(int j=1;j<=i;j++)
8.               m*=j;
9.            System.out.println(i+"!="+m);
10.           sum+=m;
11.       }
12.       System.out.println("1!+2!+3!+...+6!=" +sum);
13.     }
14. }
```

跳转语句

- 跳转语句用来实现循环执行过程中的流程转移。在switch语句中使用过的break语句就是一种跳转语句。在Java语言中，有两种跳转语句：**break语句**和**continue语句**.
- Java语言中，可用break和continue**控制循环的流程**
  - break用于强行退出循环，不执行循环中剩余的语句
  - continue则停止执行当前的循环，开始新的循环
- break语句和continue语句都有两种使用形式：
  - 不带标号的break语句和continue语句
  - 带标号的break语句和continue语句
  - 只使用break只能退出内循环；要想达到从内循环体中直接跳转出外循环的目的， 必需与label标号语句连用

break 和 continue 之例

- 不带标号的break语句和continue语句

  【example02_10】 使用break语句跳出循环体

  【example02_11】 使用continue语句终止当前循环

- 带标号的break语句和continue语句

  【example02_12】 使用带标号break语句退出循环嵌套

  【example02_13】 使用带标号continue语句继续外循环

本章总结

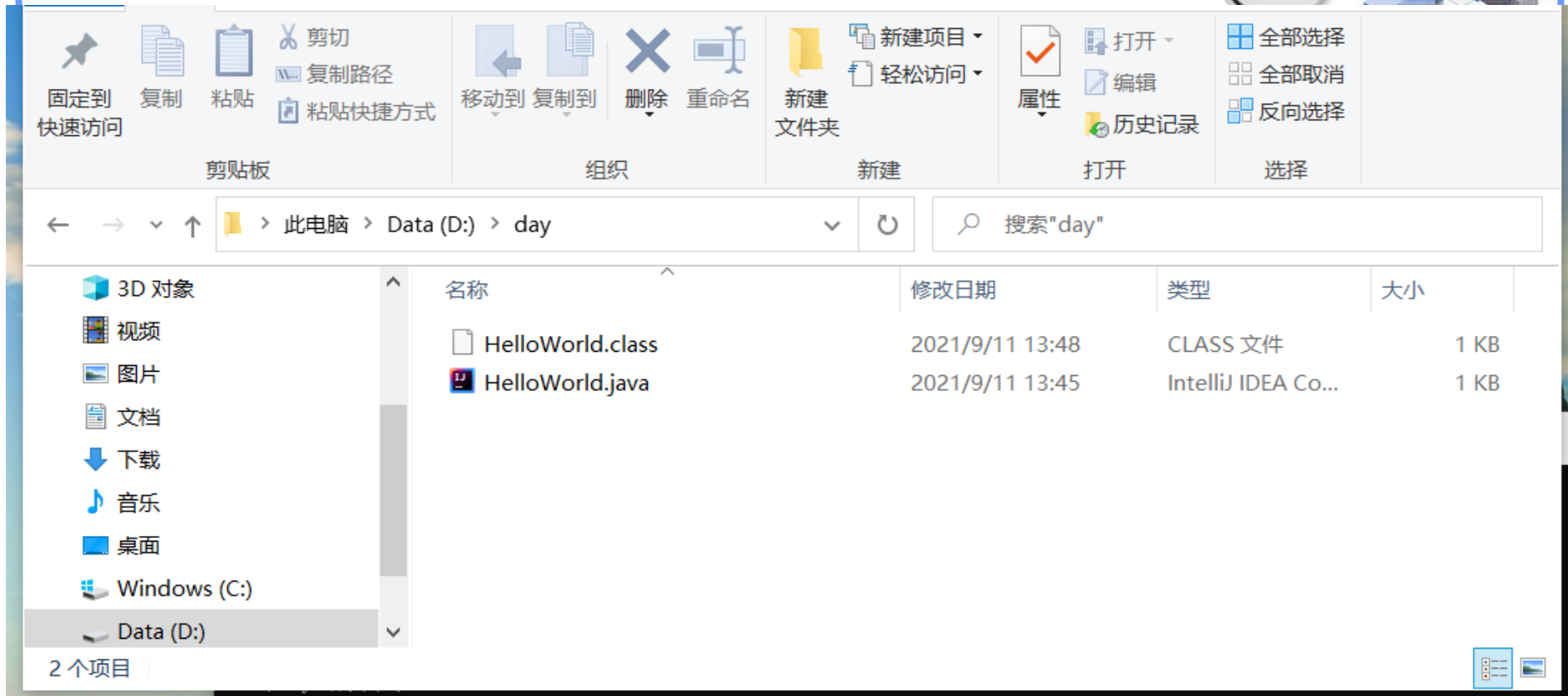
- 本章我们学习了
  - 基本数据类型
  - 字符集与标识符
  - Unicode
  - 运算符
  - 表达式
  - 程序流程控制

作业1

作业1：见作业文档。


Thank You !

课后讨论的问题1




```
2021/09/11  13:48    <DIR>          .
2021/09/11  13:48    <DIR>          ..
2021/09/11  13:48               425 HelloWorld.class
2021/09/11  13:45               124 HelloWorld.java
               2 个文件            549 字节
               2 个目录 319,622,475,776 可用字节

D:\day>java  HelloWorld
错误: 找不到或无法加载主类 HelloWorld

D:\day>
```

课后讨论的问题2

File View

Workspace '123' : 1 Project
- 123
  - src
    - 123.java
    - hello.java

123.java | hello.java

```
public class 123 {

    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Build Output

```
C:\Users\zZOMZz\Documents\JCreator Pro\MyProjects\123\src\hello.java:3: 错误: 需要<标识符>
public class 123 {
             ^
C:\Users\zZOMZz\Documents\JCreator Pro\MyProjects\123\src\hello.java:5: 错误: 非法的表达式开始
    public static void main(String[] args) {
    ^
C:\Users\zZOMZz\Documents\JCreator Pro\MyProjects\123\src\hello.java:5: 错误: 非法的表达式开始
    public static void main(String[] args) {
           ^
C:\Users\zZOMZz\Documents\JCreator Pro\MyProjects\123\src\hello.java:5: 错误: 需要';'
    public static void main(String[] args) {
                ^
C:\Users\zZOMZz\Documents\JCreator Pro\MyProjects\123\src\hello.java:5: 错误: 需要 '.class'
    public static void main(String[] args) {
                       ^
C:\Users\zZOMZz\Documents\JCreator Pro\MyProjects\123\src\hello.java:5: 错误: 需要';'
    public static void main(String[] args) {
                            ^
C:\Users\zZOMZz\Documents\JCreator Pro\MyProjects\123\src\hello.java:8: ?誥? 解析时已到达文件结尾
}
^
14 个错误

Process completed.
```

Task View

Build Report

Message	Folder	Location
Resource: 123.java		
错误: 需要<标识符>	C:\Users\zZOMZ...	line 3
错误: 非法的表达式开始	C:\Users\zZOMZ...	line 5
错误: 非法的表达式开始	C:\Users\zZOMZ...	line 5
错误: 需要';'	C:\Users\zZOMZ...	line 5
错误: 需要 '.class'	C:\Users\zZOMZ...	line 5
错误: 需要';'	C:\Users\zZOMZ...	line 5
错误: 解析时已到达文件结??	C:\Users\zZOMZ...	line 8
Resource: hello.java		
错误: 需要<标识符>	C:\Users\zZOMZ...	line 3
错误: 非法的表达式开始	C:\Users\zZOMZ...	line 5
错误: 非法的表达式开始	C:\Users\zZOMZ...	line 5
错误: 需要';'	C:\Users\zZOMZ...	line 5

课后讨论的问题3

test1.java | What's New in IntelliJ IDEA

```
1  package practice;
2
3
4  import java.util.Scanner;
5
6  public class test1 {
7      public static void main(String []args){
8          Scanner in =new Scanner(System.in);
9          System.out.println(in.nextline());
10     }
11 }
12
13
14
```

1 error 4 sec, 22 ms

```
D:\java\.idea\src\practice\test1.java:9:30
java: 找不到符号
  符号:   方法 nextline()
  位置: 类型为java.util.Scanner的变量 in
```