Java语言与系统设计

第2讲 基本语法

- 关键字
- 标识符
- 变量
- 数据类型
- 运算符

1. 关键字

- 关键字：通常对应一个英文单词或者其缩写，其英文含义就体现了它所具有的语法作用和功能。

用于定义数据类型的关键字				
class	interface	enum	byte	short
int	long	float	double	char
boolean	void			
用于定义流程控制的关键字				
if	else	switch	case	default
while	do	for	break	continue
return				
用于定义访问权限修饰符的关键字				
private	protected	public		

1. 关键字

- 关键字：通常对应一个英文单词或者其缩写，其英文含义就体现了它所具有的语法作用和功能。

用于定义类，函数，变量修饰符的关键字				
abstract	final	static	synchronized	
用于定义类与类之间关系的关键字				
extends	implements			
用于定义建立实例及引用实例，判断实例的关键字				
new	this	super	instanceof	
用于异常处理的关键字				
try	catch	finally	throw	throws
用于包的关键字				
package	import			

2. 标识符

- 标识符：是编程者（用户）根据编写程序的需要自行定义的，用英文字母、下划线字符和数字字符组成的字串，并且第1个字符不能是数字字符。
- 如x2、_data、Time、sum、a、maxWage、x_y1等都是符合语法规定的标识符。

2. 标识符

标识符要注意以下几点:

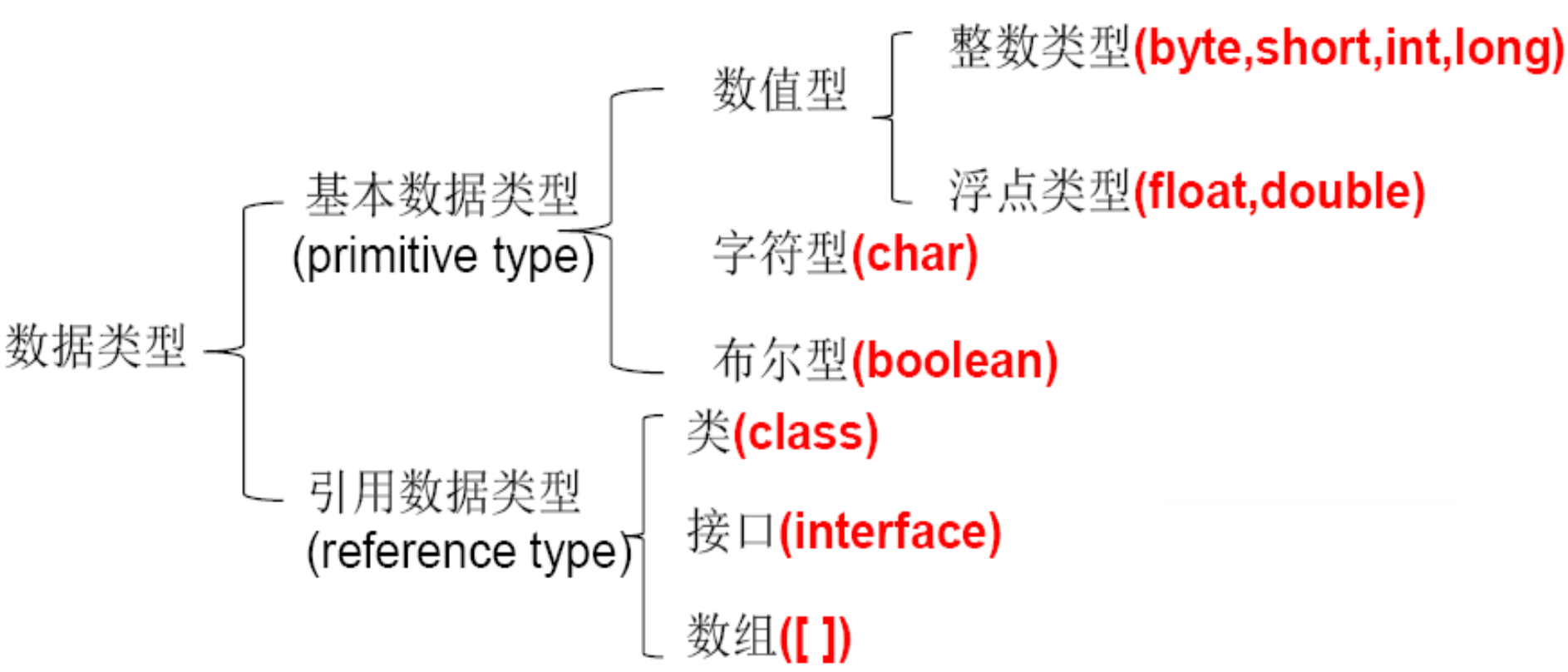
- 标识符要方便记忆
- 标识符中包含多个英文单词
- 关键字不能作为标识符
- 严格区分大小写

3. 变量

变量：变量是程序中最基本的存储单元，是内存中的一个存储区域，用于保存同一类型范围内不断变化的数据。

- 变量包含变量类型、变量名和存储的值
- Java中每个变量必须先声明，后使用。使用变量名来访问这块区域的数据
- 声明变量：int a;
- 赋值变量：a=10;
- 声明和赋值： int a=10;

4. 数据类型


数据类型
基本数据类型
(primitive type)
数值型
整数类型(byte,short,int,long)
浮点类型(float,double)
字符型(char)
布尔型(boolean)
引用数据类型
(reference type)
类(class)
接口(interface)
数组([ ])

4. 数据类型

整数类型：

- int表示标准整型，32个二进制位；
- short表示短整型，16个二进制位；
- byte表示字节整型，8个二进制位；
- long表示长整型，64个二进制位；

♦整型常量默认为int型，声明long型常量须后加‘l’或‘L’

类　型	占用存储空间	表数范围
byte	1字节=8bit位	-128 ~ 127
short	2字节	-2^{15} ~$2^{15}-1$
int	4字节	-2^{31} ~ $2^{31}-1$ (约21亿)
long	8字节	-2^{63} ~ $2^{63}-1$

4. 数据类型

浮点类型：

- float表示单精度实数类型，32个二进制位；
- double表示双精度实数类型，64个二进制位

♦浮点型常量默认为double型，声明float型常量，须后加‘f’或‘F’。

类 型	占用存储空间	表数范围
单精度float	4字节	-3.403E38 ~ 3.403E38
双精度double	8字节	-1.798E308 ~ 1.798E308

4. 数据类型

字符类型：char表示字符类型，每个值占用2个字节，16个二进制位，可以对字符类型的数据进行整数运算，此时字符类型的数据的值就是它的编码值

- 字符常量是用单引号(‘’)括起来的单个字符。例如：char c1 = 'a'; char c2 = '中'; char c3 = '9';
- 允许使用转义字符‘\’来将其后的字符转变为特殊字符型常量。例如：char c3 = ‘\n’; //'\n'表示换行符
- 直接使用Unicode值来表示字符型常量：‘\uXXXX’。XXXX代表一个十六进制整数。如：\u000a 表示\n。

4. 数据类型

字符类型：char表示字符类型，每个值占用2个字节，16个二进制位，可以对字符类型的数据进行整数运算，此时字符类型的数据的值就是它的编码值

- 字符常量是用单引号(‘’)括起来的单个字符。例如：char c1 = 'a'; char c2 = '中'; char c3 = '9';
- 允许使用转义字符‘\’来将其后的字符转变为特殊字符型常量。例如：char c3 = ‘\n’; //'\n'表示换行符
- 直接使用Unicode值来表示字符型常量：‘\uXXXX’。XXXX代表一个十六进制整数。如：\u000a 表示\n。

4. 数据类型

字符编码：

- ASCII码：上个世纪60年代，美国对英语字符与二进制位之间的关系，做了统一规定。这被称为ASCII码。ASCII码一共规定了128个字符的编码，表示字符有限。
- Unicode：将世界上所有的符号都纳入其中。每一个符号都给予一个独一无二的编码。但每个符号都用固定字节数表示，浪费存储空间。
- TF-8 是在互联网上使用最广的一种Unicode 的实现方式， 是一种变长的编码方式，使用1-6 个字节表示一个符号。

4. 数据类型

逻辑类型：

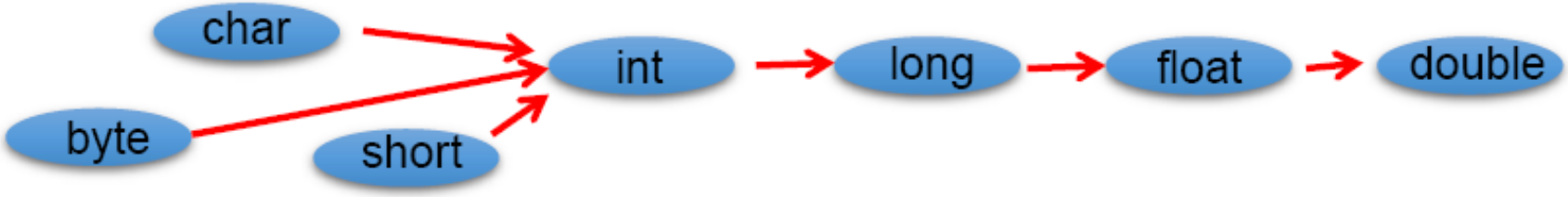
- boolean表示标准逻辑（布尔）类型，每个值占用1个字节，8个二进制位，它只包含有2个值：true和false，其中true表示逻辑真，false表示逻辑假；
- 注意：不可以使用0或非0 的整数替代false和true，这点和C语言不同。

无类型：

- void表示无类型，没有任何值，不占用存储空间，用于函数（方法）的返回值类型，表示该方法不需要返回值。

4. 数据类型

自动数据类型转换： 容量小的类型自动转换为容量大的数据类型。



- 有多种类型的数据混合运算时，系统首先自动将所有数据转换成容量最大的那种数据类型，然后再进行计算。
- byte,short,char之间不会相互转换，他们三者在计算时首先转换为int类型。boolean类型不能与其它数据类型运算。
- 当把任何基本数据类型的值和字符串(String)进行连接运算时(+)，基本数据类型的值将自动转化为字符串(String)类型。

4. 数据类型

字符串类型：

- String不是基本数据类型，属于引用数据类型；
- 使用方式与基本数据类型一致。

  例如：String str= “abcd”;
- 一个字符串可以串接另一个字符串，也可以直接串接其他类型的数据。例如：

```
str= str+ “xyz” ;
int n = 100;
str= str + n;
```

4. 数据类型

字符串类型：

- String str1 = 4;
- String str2 = 3.5f + “”;
- System.out.println(str2);
- System.out.println(3+4+“Hello!”);
- System.out.println(“Hello!”+3+4);
- System.out.println(‘a’+1+“Hello!”);
- System.out.println(“Hello”+‘a’+1);

4. 数据类型

强制数据类型转换：容量大的类型转换为容量小数据类型。

- 自动类型转换的逆过程，将容量大的数据类型转换为容量小的数据类型。使用时要加上强制转换符：()，但可能造成精度降低或溢出,格外要注意。
- 通常，字符串不能直接转换为基本类型，但通过基本类型对应的包装类则可以实现把字符串转换成基本类型。

4. 数据类型

字符串类型和其他类型转换:

- 字符串转换为基本数据类型

  Integer.parseInt(字符串)

  Float.parseFloat(字符串)

  Double.parseDouble(字符串)

- 基本数据类型转换为字符串

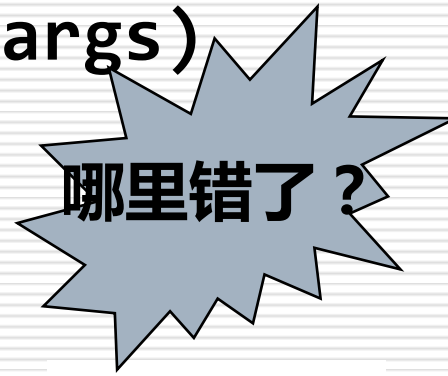
  String.valueof(基本数据类型)

实例：

```
class Example_3 {
    public static void main(String[] args) {
        int x1 = 5;
         String s1=String.valueOf(x1);
        System.out.println("x1的值是" + s1);
        String s2="10";
        int x2=Integer.parseInt(s2);
        System.out.println("s2的值是" +x2); }
    }
```

实例：

```
class Example_1
{
   public static void main(String[] args)
   {
    byte x=128;
    float f=12.58;
    System.out.println("x="+x+" f="+f );
   }
 }
```



实例：

```
class Example_1
{
   public static void main(String[] args)
   {
    byte x=127;
    float f=12.58f;
    System.out.println("x="+x+" f="+f );
   }
 }
```

实例：

```
class Example_2 {
 public static void main(String[] args) {
   char c1 = 'C';
   int ic1 = c1;
   System.out.println("ic1=" + ic1);
   int i1 = 65;
   char ci1 = (char)i1;
   System.out.println("ci1=" + ci1);
     }
    }
```

（char）？
第4行又为什么
不转换类型？

4. 数据类型

判断是否能通过编译：

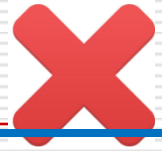
1) short s = 5;
s = s-2;

2) char c = ‘a’;
int i = 5;
float d = 0.314F;
c+i+d;

byte、short、char之间不会相互转换，
三者在计算时首先转换为int类型

3) byte b = 3;
b = b + 4;
b = (byte)(b+4);

4) byte b = 5;
short s = 3;
short t = s + b;



5. 运算符

算术运算符：有单目和双目两类，

- 单目的有++、--等，
- 双目的有+、-、*、/、%(取余)等。

5. 运算符

```
class Example_5 {
   public static void main(String[] args) {
        int a = 7;
         int b = 2;
         System.out.println(a+b);
        System.out.println(a/b);
        System.out.println(a%b);
        System.out.println(a++);
        System.out.println(a);
        System.out.println(""+a+b);
        }
}
```

5. 运算符

赋值运算符：分为一般赋值和复合赋值两类

- 一般赋值采用等号
- 复合赋值是算术运算符或者位运算符和等号的结合：

+=, -=, *=, /=, %=

如b=5为一般赋值，把5赋给b；

b+=3为复合赋值，相当于b=b+3，b的结果值为8。

5. 运算符

```
short s = 3;
s = s+2;   ①
s += 2;    ②
```

①和②有什么区别?

②不会改变变量s本身的数据类型

如果实现变量加2的操作:

```
int a=10;
a=a+2;
a+=2;（推荐）
a++++;//???
```

如果实现变量加1的操作:

```
a=a+1;
a+=1;
a++;（推荐）
```

5. 运算符

```
1) int m = 2;
   int n = 3;
   n *= m++;
   System.out.println("m=" + m);
   System.out.println("n=" + n);
   //第三行看作：n=n*m++；答案： m=3;n=6
```

5. 运算符

2）
```
int i = 1;
i *= 0.1;
System.out.println(i);
i++;
System.out.println(i);
```

//第2行不改变i的类型；答案：0;1

5. 运算符

3）
```
int n = 10;
n += (n++) + (++n);
System.out.println(n);
```

```
// 第2行看作：n =n+ (n++) + (++n);
// 代入n：          10   10       12
// 答案：32
```

5. 运算符

关系运算符：>、>=、<、<=、==、!=。它们都是双目运算符，运算结果为逻辑值真或假，即true和false。如5>4为真，3<=2为假。在这6个运算符中存在着3对相反的运算符，其中>与<=相反，<与>=相反，==与!=相反。

关系式和相反式：由关系运算构成的式子称为关系式，由它的相反运算符构成的式子称为原关系式的相反式。关系式和它相反式的值互为相反。如若x==1为真，则相反式x!=1为假；又如若x>y为真，则相反式x<=y为假。

5. 运算符

逻辑运算符： 与(&&)、或(||)、非(!)，其中与和或是双目运算符，非（取反）是单目运算符，运算对象是关系式或者是逻辑表达式，运算结果是逻辑值真或假。如y>=1 && y<=4就是一个逻辑“与”表达式，当y的值大于等于1、同时又小于等于4时，该表达式的值为真，否则为假。

逻辑运算的等价关系： !!a同a等价、!(a && b)同!a||!b等价、!(a||b)同!a&&!b等价，其中a和b表示逻辑值。如!(x>3 && x<8)同(x<=3 ||x>=8)等价。

5. 运算符

a	b	a&b	a&&b	a|b	a||b	!a	a^b
true	true	true	true	true	true	false	false
true	false	false	false	true	true	false	true
false	true	false	false	true	true	true	true
false	false	false	false	false	false	true	false

- “&”和“&&”的区别：单&时，左边无论真假，右边都进行运算；双&时，如果左边为真，右边参与运算，如果左边为假，那么右边不参与运算。“|”和“||”的区别同理。
- 逻辑运算符用于连接布尔型表达式，在Java中不可以写成3<x<6，应该写成x>3 & x<6 。

```
int x = 1;
int y=1;

if(x++==2 & ++y==2){
            x =7;
}
System.out.println
```

x=2 y=2

```
int x = 1,y = 1;

if(x++==2 && ++y==2){
            x =7;
}
System.out.print
```

x=2 y=1

```
int x = 1,y = 1;

if(x++==1 | ++y==1){
            x =7;
}
System.out.println
```

x=7 y=2

```
int x = 1,y = 1;

if(x++==1 || ++y==1){
            x =7;
}
System.out.print
```

x=7 y=1

5. 运算符

条件运算符：是一个3目运算符，包含有两个符号和3个运算对象，格式为：<逻辑表达式>？<表达式1>:<表达式2>。

如：x>y？z=x：z=y;构成的一条表达式语句，若x>y则把x的值赋给z，否则把y的值赋给z。。

5. 运算符

位运算符：

位运算符		
运算符	运算	范例
<<	左移	3 << 2 = 12 --> 3*2*2=12
>>	右移	3 >> 1 = 1 --> 3/2=1
>>>	无符号右移	3 >>> 1 = 1 --> 3/2=1
&	与运算	6 & 3 = 2
|	或运算	6 | 3 = 7
^	异或运算	6 ^ 3 = 5
~	取反运算	~6 = -7

5. 运算符

位运算符：

	位运算符的细节
<<	空位补0，被移除的高位丢弃，空缺位补0。
>>	被移位的二进制最高位是0，右移后，空缺位补0； 最高位是1，空缺位补1。
>>>	被移位二进制最高位无论是0或者是1，空缺位都用0补。
&	二进制位进行&运算，只有1&1时结果是1，否则是0;
|	二进制位进行 | 运算，只有0 | 0时结果是0，否则是1;
^	相同二进制位进行 ^ 运算，结果是0；1^1=0 , 0^0=0 不相同二进制位 ^ 运算结果是1。1^0=1 , 0^1=1
~	正数取反，各二进制码按补码各位取反 负数取反，各二进制码按补码各位取反

5. 运算符

优先级：

- 只有单目运算符、三元运算符、赋值运算符是从右向左运算的。

	. () {} ; ,
R—>L	++ -- ~ !(data type)
L—>R	* / %
L—>R	+ -
L—>R	<< >> >>>
L—>R	< > <= >= instanceof
L—>R	== !=
L—>R	&
L—>R	^
L—>R	|
L—>R	&&
L—>R	||
R—>L	? :
R—>L	= *= /= %=
	+= -= <<= >>=
	>>>= &= ^= |=

练习题：

1. 下列（　　）是不能通过编译的语句。

A. double d = 545.0;　　B. char a1 = “c”;

C. int i = 321;　　D. float f1 =45.0f;

2. 执行下列程序代码的输出结果是（　　）。

```
int a = 10;
int i, j;
i = ++a;
j = a--;
System.out.printf("%d,%d,%d", a, i, j);
```

已知：我们可以通过Math.random()来获取一个0-1之间的double随机数。要求：

(1) 生成一个0-100之间的整型随机数。

(2) 生成一个50-100之间的整型随机数。

```
public class test2
{ public static void main(String[] args)
{ int i1 = (int)(Math.random() * 100);
int i2 = (int)(Math.random() * 50) + 50;
System.out.println("0-100之间的随机数是:" + i1);
System.out.println("50-100之间的随机数是:" + i2);
}
}
```