Java

ver 1.0 fanfuli

关键字和保留字

关键字

定义：被Java语言赋予了特殊含义，用做专门用途的字符串（单词）

特点：关键字中所有字母都为小写

类别	关键字	说明
访问控制	private	私有的
	protected	受保护的
	public	公共的
	default	默认
类、方法和变量修饰符	abstract	声明抽象
	class	类
	extends	扩充,继承
	final	最终值,不可改变的
	implements	实现（接口）
	interface	接口
	native	本地，原生方法（非 Java 实现）
	new	新,创建
	static	静态
	strictfp	严格,精准
	synchronized	线程,同步
	transient	短暂
	volatile	易失

程序控制语句	break	跳出循环
	case	定义一个值以供 switch 选择
	continue	继续
	default	默认
	do	运行
	else	否则
	for	循环
	if	如果
	instanceof	实例
	return	返回
	switch	根据值选择执行
	while	循环
错误处理	assert	断言表达式是否为真
	catch	捕捉异常
	finally	有没有异常都执行
	throw	抛出一个异常对象
	throws	声明一个异常可能被抛出
	try	捕获异常
包相关	import	引入
	package	包
基本类型	boolean	布尔型
	byte	字节型
	char	字符型
	double	双精度浮点
	float	单精度浮点
	int	整型
	long	长整型
	short	短整型
变量引用	super	父类,超类
	this	本类
	void	无返回值
保留关键字	goto	是关键字，但不能使用
	const	是关键字，但不能使用
	null	空

这看起来会有很多，但是随着我们的使用，都会一一牢记，所以也不必太过担心。这里也不做过多解释。

标识符

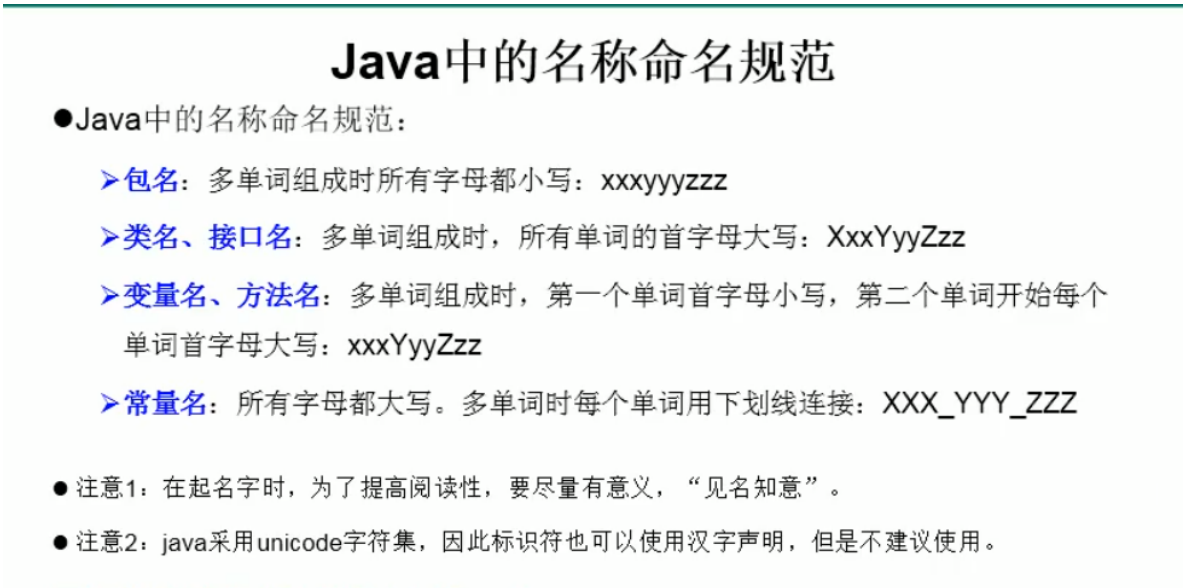
定义：Java对各种变量、方法和类等要素命名时使用的字符序列称为标识符

简单来说，凡是自己起名字的地方都是标识符

标识符规则如下

- 所有的标识符都应该以字母（A-Z 或者 a-z）,美元符（$）、或者下划线（_）开始
- 首字符之后可以是字母（A-Z 或者 a-z）,美元符（$）、下划线（_）或数字的任何字符组合
- 关键字不能用作标识符
- 标识符是大小写敏感的
- 合法标识符举例：age、$salary、_value、__1_value
- 非法标识符举例：123abc、-salary

当然程序员在日常的工作中，出于整洁性和可读性的考虑，也约定俗成了一种规范



常量

3.4.2 常量

在 Java 中，利用关键字 final 指示常量。例如：

```
public class Constants
{
   public static void main(String[] args)
   {
      final double CM_PER_INCH = 2.54;
      double paperWidth = 8.5;
      double paperHeight = 11;
      System.out.println("Paper size in centimeters: "
         + paperWidth * CM_PER_INCH + " by " + paperHeight * CM_PER_INCH);
   }
}
```

关键字 final 表示这个变量只能被赋值一次。一旦被赋值之后，就不能够再更改了。习惯上，常量名使用全大写。

在 Java 中，经常希望某个常量可以在一个类中的多个方法中使用，通常将这些常量称为类常量。可以使用关键字 static final 设置一个类常量。下面是使用类常量的示例：

```
public class Constants2
{
   public static final double CM_PER_INCH = 2.54;

   public static void main(String[] args)
   {
      double paperWidth = 8.5;
      double paperHeight = 11;
      System.out.println("Paper size in centimeters: "
         + paperWidth * CM_PER_INCH + " by " + paperHeight * CM_PER_INCH);
   }
}
```

需要注意，类常量的定义位于 main 方法的外部。因此，在同一个类的其他方法中也可以使用这个常量。而且，如果一个常量被声明为 public，那么其他类的方法也可以使用这个常量。在这个示例中，Constants2.CM_PER-INCH 就是这样一个常量。

C++ **注释**：const 是 Java 保留的关键字，但目前并没有使用。在 Java 中，必须使用 final 定义常量。

变量/数据类型

概念：变量是程序中最基本的存储单元，包含变量类型、变量名和存储的值。从内存上来看，变量是内存中的一个存储区域，该区域的数据可以在同意类型范围内不断变化

作用：在内存中保存数据

注意： 1、Java中每个变量必须先声明，后使用

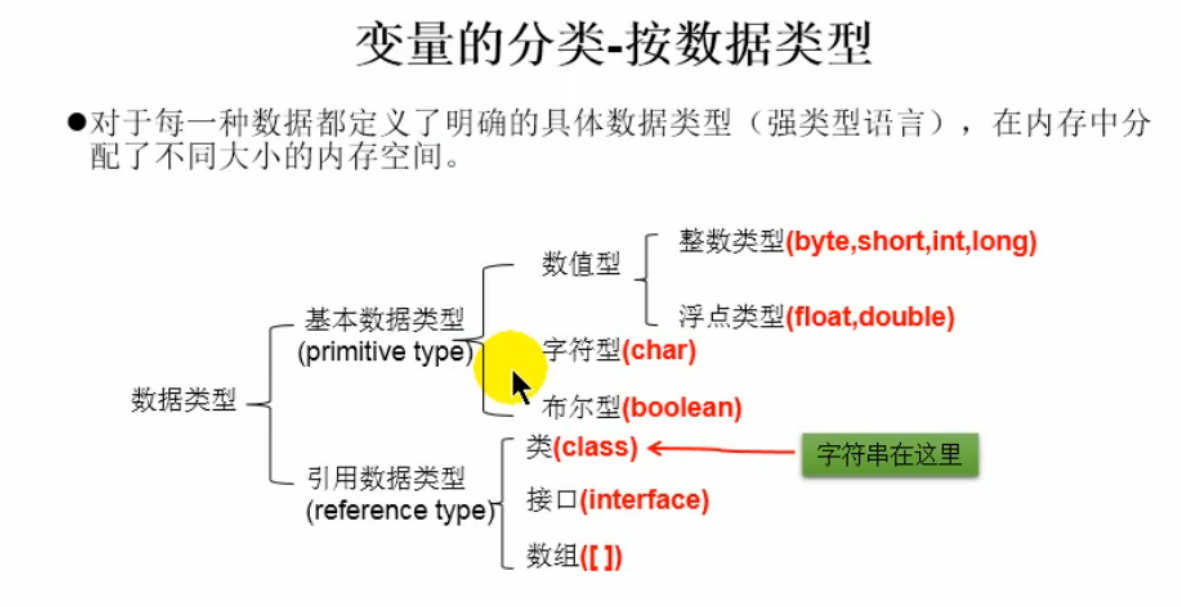
2、访问变量中的数据必须通过变量名

3、变量的作用域：其定义所在的一对{}内，变量只在其作用域中有效

```
{
    int x = 12;
    /* only x available */
    {
        int q = 96;
        /* both x & q available */
    }
    /* only x available */
    /* q "out of scope" */
}
```

X和q的作用域因为{}的限制而不同，里面的{}里能引用x，但是外面的{}里不能引用q

4、同一个作用域中不能定义重名的变量



byte:

- byte 数据类型是8位、有符号的，以二进制补码表示的整数；
- 最小值是 `-128（-2^7）`；
- 最大值是 `127（2^7-1）`；
- 默认值是 `0`；
- byte 类型用在大型数组中节约空间，主要代替整数，因为 byte 变量占用的空间只有 int 类型的四分之一；
- 例子：byte a = 100，byte b = -50。

short:

- short 数据类型是 16 位、有符号的以二进制补码表示的整数
- 最小值是 `-32768（-2^15）`；
- 最大值是 `32767（2^15 - 1）`；
- Short 数据类型也可以像 byte 那样节省空间。一个short变量是int型变量所占空间的二分之一；
- 默认值是 `0`；
- 例子：short s = 1000，short r = -20000。

int:

- int 数据类型是32位、有符号的以二进制补码表示的整数；
- 最小值是 `-2,147,483,648（-2^31）`；
- 最大值是 `2,147,483,647（2^31 - 1）`；
- 一般地整型变量默认为 int 类型；
- 默认值是 `0`；
- 例子：int a = 100000, int b = -200000。

long:

- long 数据类型是 64 位、有符号的以二进制补码表示的整数；
- 最小值是 `-9,223,372,036,854,775,808（-2^63）`；
- 最大值是 `9,223,372,036,854,775,807（2^63 -1）`；
- 这种类型主要使用在需要比较大整数的系统上；
- 默认值是 `0L`；
- 例子： long a = 100000L，Long b = -200000L。
  "L"理论上不分大小写，但是若写成"l"容易与数字"1"混淆，不容易分辨。所以最好大写。

float:

- float 数据类型是单精度、32位、符合IEEE 754标准的浮点数；
- float 在储存大型浮点数组的时候可节省内存空间；
- 默认值是 `0.0f`；
- 浮点数不能用来表示精确的值，如货币；
- 例子：float f1 = 234.5f。

double:

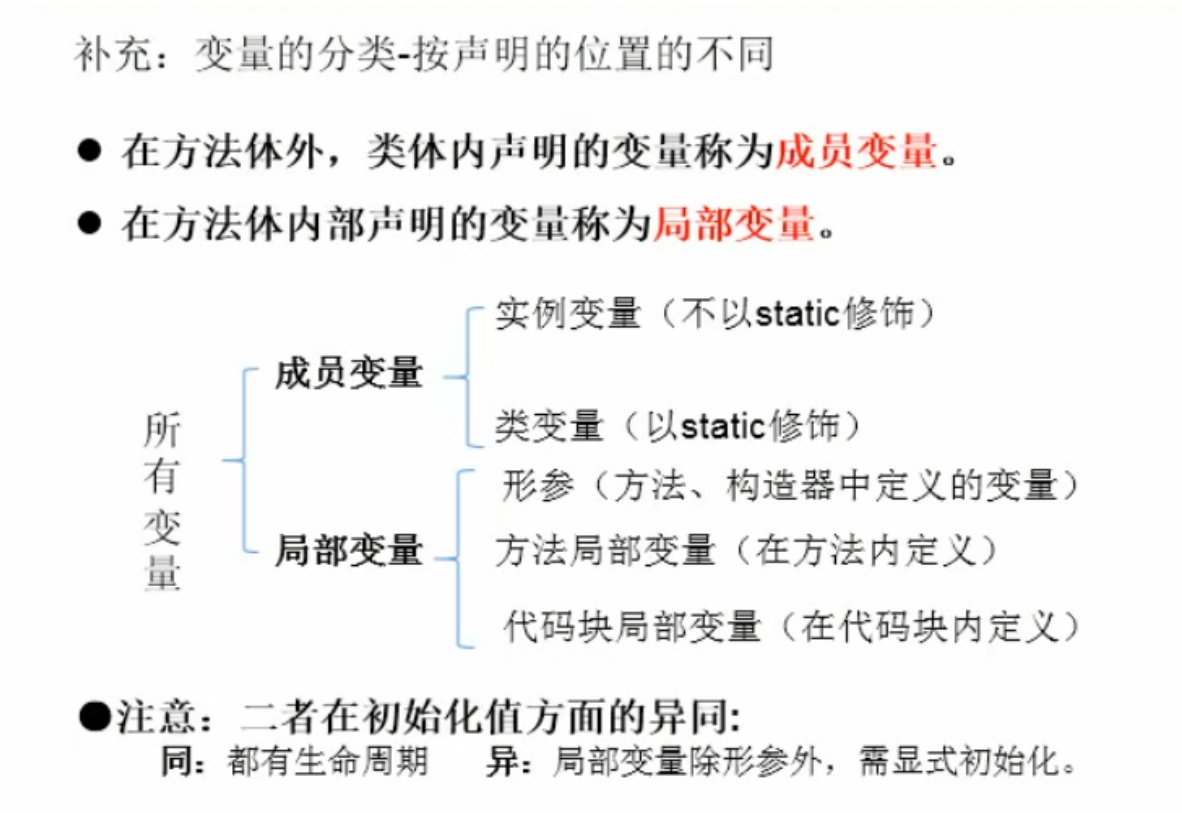
- double 数据类型是双精度、64 位、符合IEEE 754标准的浮点数；
- 浮点数的默认类型为double类型；
- double类型同样不能表示精确的值，如货币；
- 默认值是 `0.0d`；
- 例子：double d1 = 123.4。

boolean:

- boolean数据类型表示一位的信息；
- 只有两个取值：true 和 false；
- 这种类型只作为一种标志来记录 true/false 情况；
- 默认值是 `false`；
- 例子：boolean one = true。

char:

- char类型是一个单一的 16 位 Unicode 字符；
- 最小值是 `\u0000`（即为0）；
- 最大值是 `\uffff`（即为65,535）；
- char 数据类型可以储存任何字符；
- 例子：char letter = 'A';。



运算符/操作符

计算机的最基本用途之一就是执行数学运算，Java作为一门计算机语言，也提供了丰富的运算符来操作变量。

运算符一般分为以下几组：

- 算术运算符
- 关系运算符
- 位运算符
- 逻辑运算符
- 赋值运算符
- 其他运算符

算数运算符

操作符	描述	例子
+	加法 - 相加运算符两侧的值	A + B 等于 30
-	减法 - 左操作数减去右操作数	A – B 等于 -10
*	乘法 - 相乘操作符两侧的值	A * B等于200
/	除法 - 左操作数除以右操作数	B / A等于2
%	取余 - 左操作数除以右操作数的余数	B%A等于0
++	自增: 操作数的值增加1	B++ 或 ++B 等于 21（区别详见下文）
--	自减: 操作数的值减少1	B-- 或 --B 等于 19（区别详见下文）

关系运算符

运算符	描述	例子
==	检查如果两个操作数的值是否相等，如果相等则条件为真。	（A == B）为假。
!=	检查如果两个操作数的值是否相等，如果值不相等则条件为真。	(A != B) 为真。
>	检查左操作数的值是否大于右操作数的值，如果是那么条件为真。	（A> B）为假。
<	检查左操作数的值是否小于右操作数的值，如果是那么条件为真。	（A <B）为真。
>=	检查左操作数的值是否大于或等于右操作数的值，如果是那么条件为真。	（A> = B）为假。
<=	检查左操作数的值是否小于或等于右操作数的值，如果是那么条件为真。	（A <= B）为真。

位运算符

操作符	描述	例子
&	如果相对应位都是1，则结果为1，否则为0	（A&B），得到12，即0000 1100
|	如果相对应位都是 0，则结果为 0，否则为 1	（A | B）得到61，即 0011 1101
^	如果相对应位值相同，则结果为0，否则为1	（A ^ B）得到49，即 0011 0001
~	按位取反运算符翻转操作数的每一位，即0变成1，1变成0。	（~A）得到-61，即1100 0011
<<	按位左移运算符。左操作数按位左移右操作数指定的位数。	A << 2得到240，即 1111 0000
>>	按位右移运算符。左操作数按位右移右操作数指定的位数。	A >> 2得到15即 1111
>>>	按位右移补零操作符。左操作数的值按右操作数指定的位数右移，移动得到的空位以零填充。	A>>>2得到15即0000 1111

逻辑运算符

操作符	描述	例子
&&	称为逻辑与运算符。当且仅当两个操作数都为真，条件才为真。	（A && B）为假。
||	称为逻辑或操作符。如果任何两个操作数任何一个为真，条件为真。	（A || B）为真。
!	称为逻辑非运算符。用来反转操作数的逻辑状态。如果条件为true，则逻辑非运算符将得到false。	！（A && B）为真。

赋值运算符

操作符	描述	例子
=	简单的赋值运算符，将右操作数的值赋给左侧操作数	C = A + B将把A + B得到的值赋给C
+ =	加和赋值操作符，它把左操作数和右操作数相加赋值给左操作数	C + = A等价于C = C + A
- =	减和赋值操作符，它把左操作数和右操作数相减赋值给左操作数	C - = A等价于C = C - A
* =	乘和赋值操作符，它把左操作数和右操作数相乘赋值给左操作数	C * = A等价于C = C * A
/ =	除和赋值操作符，它把左操作数和右操作数相除赋值给左操作数	C / = A，C 与 A 同类型时等价于C = C / A
（%）=	取模和赋值操作符，它把左操作数和右操作数取模后赋值给左操作数	C%= A等价于C = C%A
<< =	左移位赋值运算符	C << = 2等价于C = C << 2
>> =	右移位赋值运算符	C >> = 2等价于C = C >> 2
&=	按位与赋值运算符	C&= 2等价于C = C&2
^ =	按位异或赋值操作符	C ^ = 2等价于C = C ^ 2
| =	按位或赋值操作符	C | = 2等价于C = C | 2

类与对象

类：类是一个模板，它描述一类对象的行为和状态

对象：对象是类的一个实例

例如 狗就是一个类 而哈巴狗就是类的一个实例

选择结构/循环结构

if

if 语句的语法如下：

```
if(布尔表达式)
{
   //如果布尔表达式为true将执行的语句
}
```

if...else

if 语句后面可以跟 else 语句，当 if 语句的布尔表达式值为 false 时，else 语句块会被执行。

语法

if...else 的用法如下：

```
if(布尔表达式){
   //如果布尔表达式的值为true
}else{
   //如果布尔表达式的值为false
}
```

if...else if...else

if 语句后面可以跟 else if...else 语句，这种语句可以检测到多种可能的情况。

使用 if，else if，else 语句的时候，需要注意下面几点：

- if 语句至多有 1 个 else 语句，else 语句在所有的 else if 语句之后。
- if 语句可以有若干个 else if 语句，它们必须在 else 语句之前。
- 一旦其中一个 else if 语句检测为 true，其他的 else if 以及 else 语句都将跳过执行。

语法

if...else 语法格式如下：

```
if(布尔表达式 1){
   //如果布尔表达式 1的值为true执行代码
}else if(布尔表达式 2){
   //如果布尔表达式 2的值为true执行代码
}else if(布尔表达式 3){
   //如果布尔表达式 3的值为true执行代码
}else {
   //如果以上布尔表达式都不为true执行代码
}
```

Java中主要存在3中循环结构

while循环

while是最基本的循环，它的结构为：

```
while( 布尔表达式 ) {
  //循环内容
}
```

只要布尔表达式为 true，循环就会一直执行下去。

do...while循环

对于 while 语句而言，如果不满足条件，则不能进入循环。但有时候我们需要即使不满足条件，也至少执行一次。

do...while 循环和 while 循环相似，不同的是，do...while 循环至少会执行一次。

```
do {
       //代码语句
}while(布尔表达式);
```

for循环

虽然所有循环结构都可以用 while 或者 do...while表示，但 Java 提供了另一种语句 —— for 循环，使一些循环结构变得更加简单。

for循环执行的次数是在执行前就确定的。语法格式如下：

```
for(初始化; 布尔表达式; 更新) {
    //代码语句
}
```

关于 for 循环有以下几点说明：

- 最先执行初始化步骤。可以声明一种类型，但可初始化一个或多个循环控制变量，也可以是空语句。
- 然后，检测布尔表达式的值。如果为 true，循环体被执行。如果为false，循环终止，开始执行循环体后面的语句。
- 执行一次循环后，更新循环控制变量。
- 再次检测布尔表达式。循环执行上面的过程。

Java增强for循环

Java 增强 for 循环语法格式如下:

```
for(声明语句 : 表达式)
{
   //代码句子
}
```

声明语句：声明新的局部变量，该变量的类型必须和数组元素的类型匹配。其作用域限定在循环语句块，其值与此时数组元素的值相等。

表达式：表达式是要访问的数组名，或者是返回值为数组的方法。

Break关键字

break 主要用在循环语句或者 switch 语句中，用来跳出整个语句块。

break 跳出最里层的循环，并且继续执行该循环下面的语句。

语法

break 的用法很简单，就是循环结构中的一条语句：

```
break;
```

continue关键字

continue 适用于任何循环控制结构中。作用是让程序立刻跳转到下一次循环的迭代。

在 for 循环中，continue 语句使程序立即跳转到更新语句。

在 while 或者 do...while 循环中，程序立即跳转到布尔表达式的判断语句。

语法

continue 就是循环体中一条简单的语句：

```
continue;
```

面对对象

常用api

接口与继承/多态

集合

异常

多线程

File类和IO流

JDBC

网络编程

泛型

反射

LAMBDA表达式

[^]: by fanfuli