数据类型与运算符

> ## 本节目标
>
> 1. Java标识符与关键字
> 2. Java数据类型划分
> 3. Java运算符

我们先来看一段神奇的代码

```
public class 测试{
    public staic void main(String[] args){
        int 变量1 = 10;
        int 变量2 = 10;
        int 计算结果 = 变量1 * 变量2;
        System.out.println(计算结果);
    }
}
```

1. Java标识符与关键字

1.1 Java注释

在同学们以后的编码生涯中，切记，加上注释！

Java中的注释主要分为以下三种

> - 单行注释：// 注释内容（用的最多）
> - 多行注释：/* 注释内容*/（不推荐）
> - 文档注释： /** 文档注释 */（常见于方法和类之上描述方法和类的作用），可用来自动生成文档

1.2 Java标识符

Java语言中，对于变量，常量，函数，语句块也有名字，我们统统称之为Java标识符。

对于Java标识符，有以下三点要求：

> - 标识符由字母、数字、_、$所组成，其中不能以数字开头，不能用Java中的保留字（关键字）
> - 标识符采用有意义的简单命名
> - “$”不要在代码中出现

对于类和变量的命名，Java的标准命名规范为“驼峰”命名法

对于类名，类名是以大写字母开头的名词；如果类名由多个单词组成，则每个单词的首字母大写。eg：FirstSample。并且，源文件的文件名必须与公共类的名字相同。

对于变量，如果只含有一个单词，则全部小写；如果变量名由多个单词组成，则从第二个单词开始，每个单词的首字母大写。

eg:int myName = 10;

阿里编码规范：

> 1. 代码中的命名均不能以下划线或美元符号开始，也不能以下划线和美元符号结束。
> 2. 代码命名严禁使用拼音与英文混合的方式，更不允许直接使用中文。
> 3. 类名使用UpperCamelCase风格，方法名、参数名、成员变量、局部变量统一使用lowerCamelCase风格。
> 4. 常量命名全部大写，单词间用下划线隔开。

1.3 Java关键字

关键字：在所有程序中有特殊意义的文字标记，Java中关键字如下：

用于定义访问权限修饰符的关键字				
private	protected	public		
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			
其他修饰符关键字				
native	strictfp	transient	volatile	assert

关于Java关键字有以下几点说明：

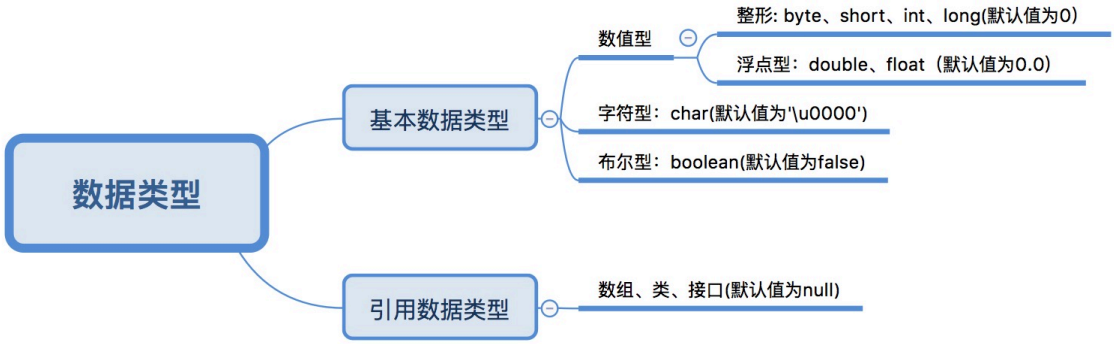
> - Java中有两个未使用的保留字：goto、const
> - Java中有三个特殊含义的单词：null、true、false
> - JDK1.4后追加了 assert关键字；JDK1.5以后追加了enum关键字

2.Java数据类型划分

程序就是一场数字游戏

Java数据类型划分以及数据类型操作，所有同学必须掌握

Java是一种强类型语言。这就意味着必须为每一个变量声明一种类型



本节课的重点在于基本数据类型，对于基本数据类型，就肯定有其保存范围

基本类型	默认值	存储	对应的包装类	数据范围
boolean	false	1位	java.lang.Boolean	true/false
byte	0	1个字节	java.lang.Byte	[-128~127] -2^7-2^7-1
short	0	2个字节	java.lang.Short	[-32768~32767] -2^15-2^15-1
char	'\u0000'	2个字节	java.lang.Character	[0~65535] 0-2^16-1
int	0	4个字节	java.lang.Integer	[-2147483648~2147483647] -2^31-2^31-1
float	0.0F或0.0f	4个字节	java.lang.Float	32位IEEE 754单精度范围
double	0.0或0.0D(d)	8个字节	java.lang.Double	64位IEEE 754双精度范围
long	0L或0l	8个字节	java.lang.Long	[-9223372036854774808~9223372036854774807] -2^63-2^63-1

对byte和int型的范围有个印象即可。

对于上述基本数据类型的选择，老师建议

> - 在程序开发之中，整数就用int，描述小数用double。
> - long一般用于描述日期、时间、内存或文件大小（字节）
> - 如果要进行编码转换或者进行二进制流的操作，使用byte（-127~128）
> - char一般在描述中文中会用到（基本忽略）

2.1 整型

在Java程序中，任何一个整型常量都是int类型

范例：定义一个整型变量

```
int num = 10 ; // 定义一个整型变量
System.out.println(num * 2) ;
```

但是在进行整型数据操作的过程之中也会存在数据溢出问题，指的是当已经达到了整型的最大最小值继续进行数学而可能产生的错误数据问题。

来看一段代码：

```
int maxValue = Integer.MAX_VALUE;
int minValue = Integer.MIN_VALUE;
System.out.println(maxValue+1);
System.out.println(minValue-1);
```

int型既然存在数据溢出问题，解决方案只有一个：更换数据类型

使用long来解决数据溢出问题：

```
int maxValue = Integer.MAX_VALUE;
long num = maxValue+1;
System.out.println(num);
```

所有程序的执行顺序是从右边到左边

解决方法如下：

> - long num = maxValue + 1L;
> - long num = 2147483648L("l"或者"L")都可以

（滴滴笔试题） System.out.println(11+1l);

大的数据类型转为小的类型，必须强转，但有时会丢失内容

```
long num = 2147483648L;
int result = (int) num;
```

> 1. 范围小的数据类型可以自动变为数据范围大的数据类型（在数学计算时）
> 2. 数据范围大的数据类型只有强制转换才能转为数据类型小的数据类型

重要说明：*关于数据默认值，默认值在主方法中无效*

范例：观察一段错误代码

```
public static void main(String[] args){
    int num ; // 定义一个整型变量但并未赋值
    System.out.println(num) ;
}
```

这个时候有两种解决方案，一种是在使用前进行赋值，另一种是在定义变量时赋值。

各个数据类型的默认值的使用，需要结合类才能观察到

范例：观察变量默认值

```
class Test {
    public int a ;
}
public class IntDemo {
    public static void main(String[] args) {
        Test test = new Test() ;
        System.out.println(test.a);
    }
}
```

既然说到了整型数据，就不得不提到byte。byte类型保存的范围是：-128~127

范例：声明byte变量

```
// 10是int，int的范围大于byte范围，理论上要进行强转，可是这里没有强转，因为数据在byte内。
byte data = 10 ; // 10是int类型，在byte范围内
System.out.println(data) ;
```

同时需要注意的是，如果要将int变量赋值给byte类型，必须强转

```
int num = 10 ; // int型变量
byte data = (byte) num ;
```

观察下面代码的输出，为什么?

```
byte data = (byte)300
```

2.2 浮点型

描述小数默认为double类型

范例：double定义小数

```
double num = 10.2 // 这是一个小数
System.out.printlb(num * 2 ) ; // num是double类型，2是int类型
```

需要注意的是，double可以保存的数字是最大的。

float保存范围小于double，如果要使用float，必须强制类型转换，在小数后加"F"或者"f"

范例：定义float变量

```
float f1 = 1.1F ;
```

神奇的代码：

```
System.out.println(1.1*1.1);
```

范例：观察计算结果

```
int numA = 10 ;
int numB = 4 ;
// int/int型还是int型
System.out.println(numA/numB) ;
// 如果要想准确计算，必须将其中一个变量变为double类型
System.out.println(numA/(double) numB) ;
```

2.3 字符型

字符型使用char来表示，并且使用' '来表示字符变量内容，并且可以与int相互转换

范例：观察char与int型转换

```
char c = 'A' ;
int num = c ;
System.out.println(num) ;
```

> - 大写字母（A~Z):65(A)~90(Z)
> - 小写字母（a~z):97(a)~122(z)
> - int型转为char型需要强制类型转换

范例：实现大写字母变小写字母（差32个长度）

```
char c = 'A'
int num = c + 32 ; // char+int=int ;
char x = (char) num ; // int转char需要强转
System.out.println(x) ;
```

注意：字符型数字（'0'~'9'）与int数字（0~9）不同 if('0'== 0)一定是false

Java中使用Unicode编码（一般其他语言采用ASC II编码），使用16进制编码，可以保存任何文字信息（包含中文），因此Java对字符的特性支持的比较好。

2.4 布尔型

在其他语言中，没有提供布尔类型，因此用'0'表示false，非0表示true。但在java中没有这样的概念

2.5 初见String类

String类是Java中非常重要的类，后续会对String类做详解。

char可以描述的只是单一的字符，现在要想描述多个字符，就必须利用字符串这一概念。需要说明的是String并不是基本数据类型，String是一个引用数据类型。

范例：观察String型变量

```
String name = "刘益铭" ;
String note = "Java老师" ;
```

String可以使用"+"表示两个字符串的连接。

范例：观察“+”问题

```
int numA = 10 ;
double numB = 10.5 ;
String result = "计算结果:" +numA + numB ; // 此时“+”不是数学运算而是字符连接。
System.out.println(result) ;
```

通过上述代码我们发现，只要使用了“+”出现了字符串，则所有的数据类型（包含引用类型）都变为String。如果要想得到正确的运算结果，必须使用"()"来解决问题.

```
String result = "计算结果:" + (numA + numB ) ;
```

在程序里还存在转义字符的概念，有如下几个常用转义字符。

换行：\n 制表符(tab)：\t 双引号：" 单引号：' (\)

范例：观察转义字符

```
System.out.println("比特科技\n\t第一节\"Java\"课开班啦");
```

> 1. 现阶段所考虑的数据类型就是int、double、boolean
> 2. 使用""声明的为字符串，字符串使用String来定义。

3. Java运算符

3.1 基础运算符

所有运算符是有优先级的，使用（）括起来，简单性原则

自增和自减运算（x++和++x)

```
int x = 3;
int y = 4;
int result = x++ * y;
System.out.println(x);
System.out.println(result);
```

总结：**x++ 先运算后自增；++x 先自增后运算**

3.2 三目运算符（重点）

三目是一种赋值运算，根据条件来判断赋哪个值。语法如下：

数据类型 变量 = 布尔表达式？满足时赋值：不满足时赋值

```
int x = 3;
int y = 4;
int z = 3>4?x+y:x-y;
System.out.println(z);
```

使用三目运算符可以简化代码，在后续JavaEE课程中会多次用到

3.3 关系运算符

关系运算符`>、<、>=、<=、==`返回类型均为布尔型(true,false)可以与逻辑判断语句一起使用

可以直接将字符和数字判断

```
char a = '哈';
int num = 10;
System.out.println(a>num);
```

3.4 逻辑运算符（重点）

逻辑运算符为：与(&、&&)、或(|、||)、非(!)，最麻烦的就是与和或

```
if(1<2|(10/0==0)){
 System.out.println("条件满足");
}
```

```
if(1<2||(10/0==0)){
 System.out.println("条件满足");
}
```

通过以上代码我们可以总结如下：

> - 当使用"&"时，明确的表示多个条件都判断了，如果在多个表达式中有条件返回了false，剩余的条件也要判断；而如果使用"&&"(短路与)，只要有条件返回false，剩余条件不再判断，返回false。
> - 当使用"|"时，明确的表示多个条件都判断了，如果在多个表达式中有条件返回了true，剩余的条件也要判断；而如果使用"||"(短路非)，只要有条件返回true，剩余条件不再判断，返回true。

因此，在同学们以后开发中，逻辑判断就用短路与和短路非

3.5 位运算符（了解）

`&、|、^、~`主要用于密码学，开发中使用不多，

位运算的关键在于二进制与十进制的转换。 使用&（有一个0就是0）、使用|（有一个1就是1）

```
System.out.println(2&8);
System.out.println(2|8);
```

课后习题

- 说说 & 和 && 的区别。
- 存在使 i + 1 < i的数吗?
- 0.6332 的数据类型是（）
- System.out.println("5" + 2); 的输出结果应该是（）
- float f=3.4;是否正确?