

Java程 序 设 计

第2讲：数据类型、变量、常量与运算表达式

- 1 本讲目标
- 2 循环一:类型的地图
- 3 循环二:类型转换
- 4 循环三:运算符的三个坑
- 5 循环四:常量与换算(机动)
- 6 收束:自查与课后

本讲目标:下课前,你应该能做到

三件事

1. **会选**:给一个数据挑对类型,说得出 8 种基本类型的字节大小;
2. **会判**:看到类型转换和 `int` 除法,能预判结果;
3. **会解释**: $127 + 1 = -128$ 和 $0.1 + 0.2 \neq 0.3$ 这两件怪事。

课堂节奏

- 4 个循环,每个循环:讲 10 分钟 → 动手 5 分钟;
- 抢答先自己猜,再验证;实操限时完成,截图发课程群。

开场预测:这段代码输出 true 还是 false?

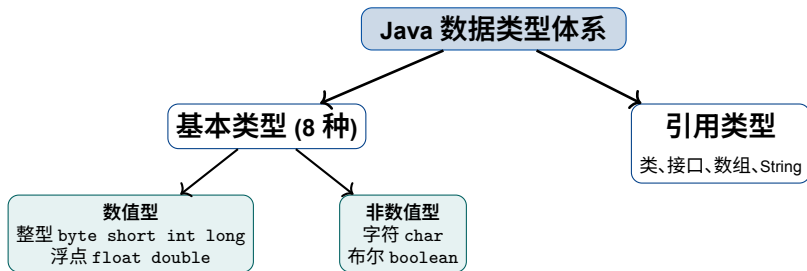
【随堂抢答·概念检测】先猜,30 秒后现场验证

```
System.out.println(0.1 + 0.2 == 0.3);
```

- A. true——数学上显然相等
- B. false——里面藏着坑

变量:内存里一块带名字格子

- 变量 = 内存中带名称的存储单元,先声明类型、后使用;
- Java 是**强类型**语言:变量一旦声明,类型终身不变;
- 语法:数据类型变量名 = 初始值;



整型四兄弟:字节越大,跑得越远

类型	字节	取值范围	字面量写法
byte	1	-128 ~ 127	(byte) 100
short	2	-32768 ~ 32767	(short) 1000
int	4	约 ± 21.47 亿	100 (整型默认)
long	8	$\pm 9.2 \times 10^{18}$	100000000000L (加 L)

记法: n 字节能表达 $-2^{8n-1} \sim 2^{8n-1} - 1$; 日常写 int, 超 21 亿才上 long。

其余四型,各有一条铁律

浮点:float / double

- float 4 字节,字面量必须加 **F**:3.14F;
- double 8 字节,小数默认就是它;
- 都是**近似值**——开场那道题的祸根。

浮点比较的正确姿势:`Math.abs(a - b) < 1e-9`,永远不用 `==`。

char 与 boolean

- char 2 字节,存一个字符:'A'、'中';
- boolean 只有 true / false;
- 铁律:boolean 与 0、1 **互不相通**,不能强转。

抢答 1:哪一行编译直接报错?

【随堂抢答·概念检测】4 行声明,找出报错行

```
long a = 100;           // 行 1  
long b = 2147483648;    // 行 2  
long c = 2147483648L;   // 行 3  
float d = 3.14;         // 行 4
```

- A. 仅行 2
- B. 仅行 4
- C. 行 2 和行 4
- D. 全部正常

抢答 1 解析:答案 c,都是“默认类型”惹的祸

【知识要点】字面量也有默认类型

- 行 2:整数字面量默认是 `int`, 2147483648 超 `int` 上限 → `integer number too large`;
- 行 3:加 `L` 才是 `long` 字面量,放得下;
- 行 4:3.14 默认是 `double` (8 字节),塞进 `float` (4 字节) 属窄化 → `lossy conversion`;
- 口诀:整数默认 `int`, 小数默认 `double`。

转换规则一张图：顺流自动，逆流强转



- 小 → 大(顺着箭头): **自动提升**, 绝不丢数据;
- 大 → 小(逆着箭头): 必须强转 (byte) x, 高位**直接截断**。

强转的代价:258 变 2

```
int x = 258;    // 二进制: ...0001 0000 0010
byte b = (byte) x;    // 只留最低 8 位: 0000 0010
System.out.println(b);    // 输出 2
```

- 强转 = 把高位二进制剪掉,只留目标类型的位数;
- 剪掉的部分**不报警、不还原**——强转是你自己签的责任书。

找茬 1:127 自增之后是 128 吗?

【随堂找茬·避坑挑错】输出是多少? 为什么?

```
byte b = 127;  
b++;  
System.out.println("b = " + b);
```

- A. b = 128
- B. b = -128
- C. b = 0
- D. 编译报错

【知识要点】溢出回绕(Overflow)

- 127 的补码:0111 1111,加 1 $\rightarrow$ 1000 0000;
- 最高位变成符号位:真值 = -128;
- 想象汽车里程表从 999999 跳回 000000——字节只有 8 位表盘;
- 通信里的校验和、时间戳溢出,全是这一族问题。

实操 1: 亲手把 258 掰成 2、把 127 掰成 -128

【随堂实操·动手练笔】TypeCastDemo ·限时 4 分钟

编写 TypeCastDemo.java:

1. `int x = 258` 强转 `byte` 并打印;
2. `byte b = 127` 自增后打印。

完成标志: 两行输出 2 和 -128 → 截图发课程群。

坑一:整数除法不四舍五入,直接扔小数

```
int a = 5, b = 2;  
System.out.println(a / b);           // 2    小数被扔掉!  
System.out.println((double) a / b); // 2.5  浮点除法  
System.out.println(1.0 / 2 * 10);  // 5.0
```

- 两个**整型**相除 = 整除,小数部分**直接丢弃**(不是四舍五入);
- 想要小数:把其中一个操作数变成 double。

&& 短路与

- 左边为 false,右边根本不执行;
- 用途:(arr != null) && (arr.length > 0) 先保命再判断。

|| 短路或

- 左边为 true,右边同样被跳过;
- 副作用(++y、赋值)放右边 = 可能不执行。

抢答 2:短路之后,y 是多少?

【随堂抢答·概念检测】执行后 x、y 的值分别是?

```
int x = 5, y = 10;  
boolean res = (x > 10) && (++y > 10);
```

- A. x=5, y=11
- B. x=5, y=10
- C. x=6, y=11
- D. 编译报错

【知识要点】短路执行轨迹

- 左边 $(x > 10) = \text{false}$, 短路与 **直接判死** 整个表达式;
- $++y$ **从未执行**, y 保持 10;
- 推论:把昂贵的判断放右边、保命的判断放左边,是性能也是安全。

坑三回收:浮点比较的正确姿势

```
System.out.println(0.1 + 0.2);           // 0.30000000000000004
System.out.println(0.1 + 0.2 == 0.3);    // false
System.out.println(Math.abs(0.1 + 0.2 - 0.3) < 1e-9); // true
```

- 浮点是**二进制近似值**,十进制小数大多存不准;
- 通信采样值、金额计算:**比较用误差阈值**,高精度用 BigDecimal。

final 常量:写死一次,全工程共享

```
public class ConstantsDemo {  
    public static final double PI = 3.141592653589793;  
    public static final int DEFAULT_BAUD_RATE = 115200;  
}
```

- 三要素:final 不可改 + static 全局一份 + 全大写加下划线;
- 好处:魔法数字收编成有名字的约定,改一处全局生效。

【随堂实操·动手练笔】TemperatureConverter • 限时 5 分钟

编写 TemperatureConverter.java:

1. 摄氏转华氏: $F = C \times 9.0/5.0 + 32$;
2. 注意用 9.0, 想想写成 9/5 会发生什么;
3. 输出 26.5°C 的换算结果。

完成标志: 输出 79.7 → 截图发课程群。

1. **会选**: 8 种类型各占几字节? 什么时候必须 long、什么时候加 F?
2. **会判**: $5 / 2$ 等于几? (byte) 258 等于几?
3. **会解释**: $127++$ 为什么是 -128? 浮点为什么不能 `==`?

没做到的那条, 就是课后任务的第一优先级。

工程视野

类型的边界就是系统的边界: 量化精度、溢出回绕, 差之毫厘谬以千里——严谨是对通信工程师的底线要求。

1. 练习:用三目运算符 ($a > b ? a : b$) 写三数取最大值;
2. AI 实战:让 AI 生成一组针对 byte 边界(127、-128、258)的测试用例,逐条人工核对预期输出。

下讲预告:控制台交互——`nextLine` 凭空吃掉一行输入之谜。

在线主页与课程资源

■ 个人主页:

<https://matnoble.top>

■ 课程专栏:

<https://matnoble.top/courses/>

■ GitHub 仓库:

<https://github.com/MatNoble>

自由交流与提问 (Q&A)

“学贵善疑,逻辑筑基”

感谢大家的专注聆听!

课件最新版本与补充资料将持续同步至课程主页。

现在进入自由提问与交流时间。