

第03课

方块也有数据类型

先问数据是什么，再决定怎样保存



仓库管理员不会把药水倒进装火把的格子。Java 的类型也在做类似的检查：整数、小数、真假和文字各有规则。这一课，我们会遇见六种常用类型，看看5除以2为什么可能得到2，并为四种方块制作一组有清楚字段的资料卡。

阅读方式：先猜结果，再运行核对；每次只改一个条件。

使用 Java 11 或更新版本。故事与插画借用方块世界，本系列练习运行在普通 Java 控制台。

01 类型是一份约定



选择容器之前，先弄清你需要保存的是数量、状态还是名字。

“生命值20”和“玩家名Alex”看起来都能显示在屏幕上，却不是同一种数据。数字可以参与计算，文字可以被搜索和截取。类型让编译器知道：你准备保存什么，又打算怎样使用它。

- 完整数量：通常从 `int` 开始。
- 可能带小数：本课使用 `double`。
- 真假状态：使用 `boolean`。
- 名字或消息：使用 `String`。

资料卡中的数值是教学示例，用来练习类型和输出；它们不是各版本 Minecraft 的参数对照表。真实游戏机制还会受版本和场景影响。

02 六种常用类型，一张资料卡

给数据找合适的位置

类型	本课用途	例子
int	普通整数	64
long	更大范围的整数	9_000_000_000L
double	带小数的数值	1.5
boolean	真假状态	false
char	一个 UTF-16 单元	'A'
String	一段文字	"石头"

完整程序 • BlockData.java

```
public class BlockData {  
    public static void main(String[] args) {  
        int maxStack = 64;  
        long worldBlocks = 9_000_000_000L;  
        double hardness = 1.5;  
        boolean falls = false;  
        char grade = 'A';  
        String blockName = "石头";  
        System.out.println(blockName + ": " + hardness);  
        System.out.println(maxStack + " / " + worldBlocks);  
        System.out.println(falls + " / " + grade);  
    }  
}
```

运行核对

石头: 1.5
64 / 9000000000
false / A

Java 有八种基本类型: byte、short、int、long、float、double、char、boolean。String 很常用，但属于引用类型，不是第九种基本类型。

03 int 与 long：整数也有边界

容量更大，仍然有范围

类型	可表示的整数范围
int	-2,147,483,648 到 2,147,483,647
long	-9,223,372,036,854,775,808 到 9,223,372,036,854,775,807

片段：放在 main 中，单独运行

```
int stack = 64, height = 320, score = -15;
System.out.println(stack + height + score);
long exploredBlocks = 9_000_000_000L;
System.out.println(exploredBlocks);
int oneMillion = 1_000_000;
System.out.println(oneMillion);
```

运行核对

```
369
9000000000
1000000
```

数字中的下划线只帮助人读数，不会进入输出。大整数末尾的大写 L 表示 long 字面量。9_000_000_000 超过 int 字面量的范围，不能先按 int 理解，再指望左侧的 long 自动救场。

long 不是“无限大整数”。变量的范围、字面量的类型和计算是否溢出，是三个需要分别检查的问题。先用本课的安全数值练习，再探索边界。

04 double：小数有用，也有近似

完整数量与带小数的量

int

物品数量
例如 4

double

练习坐标或硬度
例如 1.5

float

另一种浮点类型
字面量可写 0.8F

片段：放在 main 中，单独运行

```
double hardness = 1.5;
double spawnX = -23.75;
float lightRatio = 0.8F;
System.out.println(hardness);
System.out.println(spawnX);
System.out.println(lightRatio);
System.out.println(0.1 + 0.2);
```

运行核对

```
1.5
-23.75
0.8
0.30000000000000004
```

最后一行不是电脑随便多打了几个字。很多十进制小数无法在二进制浮点数中精确表示，运算会留下近似误差。坐标实验通常可以接受这种微小误差，精确计价则需要另选工具。

不带 F 的小数字面量通常是 double。不要把 double 理解成“十进制完全精确的小数盒子”。

05 boolean：只有真和假

判断结果不是第三种数字

表达式	结果	说明
<code>12 >= 8</code>	<code>true</code>	至少有8
<code>12 == 0</code>	<code>false</code>	不是0
<code>!false</code>	<code>true</code>	把真假翻转
<code>!true</code>	<code>false</code>	再次翻转

片段：放在 main 中，单独运行

```
int torches = 12;
boolean enoughLight = torches >= 8;
boolean empty = torches == 0;
boolean gateOpen = false;
gateOpen = !gateOpen;
System.out.println(enoughLight);
System.out.println(empty);
System.out.println(gateOpen);
```

运行核对

```
true
false
true
```

boolean 表达的是一个问题的答案。它不接受 yes、True 或数字1作为真假字面量。比较运算会得到 boolean；它本身还不会替程序选择下一步行动。

! 是“取反”。同一个变量从 false 更新为 true，是执行 `!gateOpen` 的结果；不是所有 boolean 变量都会自行翻转。

06 char 与 String：引号戴对了吗？



一块符文与一串消息有不同的表示方式。

单引号与双引号

写法	类型	含义
'A'	char	一个 UTF-16 单元
'钻'	char	本例中文字符
"Alex"	String	一串文字
"A"	String	长度为1的字符串

片段：放在 main 中，单独运行

```
char grade = 'A';
char symbol = '钻';
String playerName = "Alex";
System.out.println(grade + " / " + symbol);
System.out.println(playerName);
```

运行核对

A / 钻
Alex

char 不是“任何看起来像一个图案的东西”。有些表情符号需要两个 UTF-16 单元，要放在 String 中。把 "A" 赋给 char，或者把 'Alex' 赋给 String，都不符合这里的类型规则。

07 整数除法：小数部分去哪了？

先看参与计算的类型

表达式	结果	计算发生了什么
5 / 2	2	两个整数做除法
5.0 / 2	2.5	一侧已经是 double
(double) (5 / 2)	2.0	先算出2，再转换
(double) 5 / 2	2.5	先转换，再做除法

片段：放在 main 中，单独运行

```
System.out.println(5 / 2);
System.out.println(5.0 / 2);
System.out.println((double) (5 / 2));
System.out.println((double) 5 / 2);
System.out.println(-5 / 2);
```

运行核对

```
2
2.5
2.0
2.5
-2
```

整数除法会向零截去小数部分：5 / 2 是2，-5 / 2 是-2。不要把它简单记成“总是向下取整”。需要小数时，让至少一侧在除法发生之前成为浮点数。

写 double result = 5 / 2; 并不能保住2.5。右侧先完成整数除法，左侧最终接住的是2.0。

08 类型转换：先问会不会丢信息



传送门可以改变表示方式，但不能凭空找回已经丢掉的小数。

两条不同的转换路线

起点	转换	终点
int 12	赋给 double	12.0
double 12.8	(int) 强制转换	12
double -12.8	(int) 强制转换	-12

片段：放在 main 中，单独运行

```
int level = 12;
double displayLevel = level;
double exactXp = 12.8;
int shownXp = (int) exactXp;
System.out.println(displayLevel);
System.out.println(shownXp);
System.out.println((int) -12.8);
```

运行核对

```
12.0
12
-12
```

本例的 int 转 double 能保留整数值。double 转 int 在这里向零截去小数，不会四舍五入。超出范围、NaN 和无穷大有另一些规则；强转前先检查数据，别把强转当成“安全验证”。

09 类型排错：左右两边要相容



检查标签和内容是否匹配，是定位类型问题的第一步。

故意写错：分别检查四行

```
int hardness = 1.5;
char grade = "A";
String name = 'Steve';
long blocks = 9_000_000_000;
```

- 小数1.5应由 double 接住。
- char 的本例值写成 'A'，使用单引号。
- 名字是一串文字，要写 "Steve"。
- 大整数字面量末尾补大写 L。

片段：放在 main 中，单独运行

```
String countText = "64";
int count = 64;
System.out.println(countText + 1);
System.out.println(count + 1);
```

运行核对

```
641
65
```

显示出来很像，不代表类型相同。第6课会把数字文字解析成数值；现在先能分清“拼接”与“计算”。

10 设计四张方块资料卡



图书式资料卡让不同字段各就各位，也方便横向比较。

本作品的练习数据

方块	每组上限	练习硬度	重力下落	评级
草方块	64	0.6	false	C
沙子	64	0.5	true	B
黑曜石	64	50.0	false	S
钻石矿	64	3.0	false	A

每张卡都包含名字、每组数量上限、硬度、是否因重力下落和评级五项。下面的参考程序把它们排成一行，四行对应四张卡。数值按本页练习表使用；评级是我们给作品自定的标签。

想把卡片改成多行？用已经学过的 `println` 拆开输出即可。先让数据与类型正确，再雕刻边框和排版。

11 完整作品：四张卡逐一输出

完整程序 • FourCards.java

```
public class FourCards {  
    public static void main(String[] args) {  
        String name = "草方块";  
        int maxStack = 64;  
        double hardness = 0.6;  
        boolean falls = false;  
        char grade = 'C';  
        System.out.println(name + " | " + maxStack + " | "  
            + hardness + " | " + falls + " | " + grade);  
        name = "沙子";  
        hardness = 0.5;  
        falls = true;  
        grade = 'B';  
        System.out.println(name + " | " + maxStack + " | "  
            + hardness + " | " + falls + " | " + grade);  
        name = "黑曜石";  
        hardness = 50.0;  
        falls = false;  
        grade = 'S';  
        System.out.println(name + " | " + maxStack + " | "  
            + hardness + " | " + falls + " | " + grade);  
        name = "钻石矿";  
        hardness = 3.0;  
        falls = false;  
        grade = 'A';  
        System.out.println(name + " | " + maxStack + " | "  
            + hardness + " | " + falls + " | " + grade);  
    }  
}
```

运行核对

```
草方块 | 64 | 0.6 | false | C  
沙子 | 64 | 0.5 | true | B  
黑曜石 | 64 | 50.0 | false | S  
钻石矿 | 64 | 3.0 | false | A
```

这里故意复用同一组变量。输出完成后再更新，旧行仍保留旧快照。检查每种方块：是否忘记更新某个字段？

12 为自己的类型选择找理由



数据说明得越清楚，后续的计算与判断越可靠。

第三课的能力地图



- 基础：给一只宠物保存名字、生命、物品数量、是否成年和评级。
- 进阶：比较 $9 / 4$ 、 $9.0 / 4$ 、 $(\text{double}) 9 / 4$ ；结果是2、2.25、2.25。
- 创造：给神秘方块设计资料卡，增加是否发光的 `boolean` 字段。

自测：String 是基本类型吗？ $(\text{int}) 12.8$ 会得到13吗？答案都是否。下一课《矿工计算器》会把这些数据组成算术、比较和逻辑表达式。

技术参考：Java SE 11 语言规范，第4章数据类型、第5章类型转换。官方文档：
docs.oracle.com/javase/specs/jls/se11/html/。