

JAVA / LESSON 03 OF 40

Blocks Have Data Types, Too

First ask what the data is, and then decide how to save it

Java 11+ | 90-110 minutes



Make four Minecraft block data cards

The warehouse manager will not pour the potion into the grid containing the torch. Java's types do similar checks: integers, decimals, true and false, and literals have their own rules. In this lesson, we'll meet six common types, see why dividing 5 by 2 might result in 2, and create a set of information cards with clear fields for the four types of blocks.

Before you start: It is recommended to complete Lessons 1-2 first

Read the full lesson and copy the code:

<https://teacher-ma.de/minecraft-java-lesson.php?lesson=3&lang=en>

Long code lines may wrap at the > margin marker. The marker is not part of the code.

In this lesson

01. Type is an agreement
02. Six commonly used types, one data card
03. int and long: integers also have boundaries
04. double: decimals are useful and approximate
05. boolean: only true and false
06. char and String: Are the quotation marks correct?
07. Integer division: Where does the decimal part go?
08. Type conversion: first ask whether information will be lost
09. Type debugging: the left and right sides must be compatible
10. Design four block data cards
11. complete work: four cards are output one by one
12. Find reasons for your type selection

01 / 12

Type is an agreement



Before choosing a container, first figure out whether you need to save the quantity, status or name.

"Health value 20" and "player name Alex" both appear to be displayed on the screen, but they are not the same data. Numbers can be involved in calculations, and text can be searched and intercepted. Types let the compiler know what you are going to save and how you are going to use it.

- Full quantity: usually starts with int.
- Decimals possible: double is used in this lesson.
- True or false status: use boolean.
- Name or message: Use String.

The values in the data card are teaching examples used to practice types and outputs; they are not parameter comparison tables for each version of Minecraft. The actual game mechanics will also be affected by the version and scenario.

02 / 12

Six commonly used types, one data card

Find a suitable location for data

Type	Purpose of this lesson	Example
int	Ordinary integer	64
long	A larger range of integers	9_000_000_000L
double	Numerical values with decimals	1.5
boolean	True and false status	false
char	A UTF-16 unit	'A'
String	A paragraph of text	"Stone"

Complete program · 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 = "Stone";  
        System.out.println(blockName + ": " + hardness);  
        System.out.println(maxStack + " / " + worldBlocks);  
        System.out.println(falls + " / " + grade);  
    }  
}
```

Output

```
Stone: 1.5  
64 / 9000000000  
false / A
```

Java has eight basic types: byte, short, int, long, float, double, char, and boolean. String is very commonly used, but it is a reference type, not the ninth basic type.

03 / 12

int and long: integers also have boundaries

Larger capacity, still scope

Type	Representable integer range
int	-2,147,483,648 to 2,147,483,647
long	-9,223,372,036,854,775,808 To 9,223,372,036,854,775,807

Snippet: place inside main and run separately

```
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);
```

Output

```
369
9000000000
1000000
```

Underlines in numbers only aid human reading and do not enter the output. A capital L at the end of a large integer indicates a long literal. 9_000_000_000 exceeds the range of int literals. You cannot understand it as int first and then expect the long on the left to automatically save the situation.

long is not an "infinite integer". The scope of the variable, the type of the literal and whether the calculation overflows are three issues that need to be checked separately. Practice using the safe values in this lesson first, then explore the boundaries.

04 / 12

double: decimals are useful and approximate

Complete quantity and quantity with decimals

Element	Explanation
int	Number of items For example 4
double	Exercise coordinates or hardness For example 1.5
float	Another floating point type The literal can be written as 0.8F

Snippet: place inside main and run separately

```
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);
```

Output

```
1.5
-23.75
0.8
0.30000000000000004
```

The last line is not a computer casually typing a few more words. Many decimal fractions cannot be represented exactly in binary floating point numbers, and operations will leave approximation errors. Coordinate experiments can usually accept this small error, but accurate pricing requires the selection of another tool.

Fractional literals without F are usually double. Don't think of double as "a decimal-exact decimal box."

05 / 12

boolean: only true and false

The judgment result is not the third type of number

Expression	Result	Annotations
12 >= 8	true	There are at least 8
12 == 0	false	Not 0
!false	true	Flip the true from the false
!true	false	Flip again

Snippet: place inside main and run separately

```
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);
```

Output

```
true
false
true
```

boolean expresses the answer to a question. It does not accept yes, True, or the number 1 as true or false literals. The comparison operation returns a boolean; it does not itself choose the next action for the program.

! is "reverse". The same variable is updated from false to true as a result of executing !gateOpen; not all boolean variables flip on their own.

06 / 12

char and String: Are the quotation marks correct?



A piece of rune and a string of messages have different representations.

Single quotes and double quotes

Writing method	Type	Meaning
'A'	char	A UTF-16 unit
'钻'	char	The Chinese character used in this example
"Alex"	String	A string of text
"A"	String	A string of length 1

Snippet: place inside main and run separately

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

Output

```
A / 钻
Alex
```

char is not "anything that looks like a pattern". Some emojis require two UTF-16 units to be placed in a String. Assigning "A" to char, or assigning 'Alex' to String, does not comply with the type rules here.

07 / 12

Integer division: Where does the decimal part go?

First look at the types of calculations involved

Expression	Result	Calculate what happened
5 / 2	2	Division of two integers
5.0 / 2	2.5	One side is already double
(double) (5 / 2)	2.0	Calculate 2 first, then convert
(double) 5 / 2	2.5	Convert first, then divide

Snippet: place inside main and run separately

```
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);
```

Output

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

Integer division truncates the decimal part toward zero: 5 / 2 is 2, -5 / 2 is -2. Don't write it off simply as "always round down." When decimals are required, let at least one side be a float before division occurs.

Writing double result = 5 / 2; does not preserve 2.5. The right side completes integer division first, and the left side finally catches 2.0.

08 / 12

Type conversion: first ask whether information will be lost



The portal can change the representation, but it cannot retrieve the lost decimals out of thin air.

Two different conversion routes

Start	Conversion	End point
int 12	assigned to double	12.0
double 12.8	(int) forced conversion	12
double -12.8	(int) forced conversion	-12

Snippet: place inside main and run separately

```
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);
```

Output

```
12.0  
12  
-12
```

The int to double conversion in this example can preserve the integer value. Double to int here truncates decimals towards zero without rounding. There are other rules for out-of-range, NaN, and infinity; check the data before forcing, and don't think of forcing as a "safety verification".

09 / 12

Type debugging: the left and right sides must be compatible



Checking whether tags and content match is the first step in locating type problems.

Intentional typo: check four lines separately

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

- Decimal 1.5 should be caught by double.
- This example value of char is written as 'A', using single quotes.
- The name is a string of text, and it should be written "Steve".

- Add a capital L at the end of the large integer literal.

Snippet: place inside main and run separately

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

Output

```
641  
65
```

They appear to be very similar, but it does not mean that they are of the same type. Lesson 6 will parse numerical words into numerical values; now we can distinguish between "splicing" and "calculation".

10 / 12

Design four block data cards



The book-style data card allows different fields to be in place, and also facilitates horizontal comparison.

Practice data of this work

Block	Upper limit of each group	Practice Hardness	Gravity drop	Rating
Grass Block	64	0.6	false	C
sand	64	0.5	true	B
Obsidian	64	50.0	false	S
Diamond Ore	64	3.0	false	A

Each card contains five items: name, maximum number of each group, hardness, whether it fell due to gravity and rating. The reference program below arranges them in a row, with four rows corresponding to four cards. The numerical values are used in the exercise sheet on this page; the rating is our custom label for the work.

Want to change the card to multiple lines? Just use the `println` you have already learned to split the output. Make the data and type correct first, then carve the border and typesetting.

complete work: four cards are output one by one

Complete program · FourCards.java

```
public class FourCards {  
    public static void main(String[] args) {  
        String name = "Grass Block";  
        int maxStack = 64;  
        double hardness = 0.6;  
        boolean falls = false;  
        char grade = 'C';  
        System.out.println(name + " | " + maxStack + " | "  
            + hardness + " | " + falls + " | " + grade);  
        name = "sand";  
        hardness = 0.5;  
        falls = true;  
        grade = 'B';  
        System.out.println(name + " | " + maxStack + " | "  
            + hardness + " | " + falls + " | " + grade);  
        name = "Obsidian";  
        hardness = 50.0;  
        falls = false;  
        grade = 'S';  
        System.out.println(name + " | " + maxStack + " | "  
            + hardness + " | " + falls + " | " + grade);  
        name = "Diamond Ore";  
        hardness = 3.0;  
        falls = false;  
        grade = 'A';  
        System.out.println(name + " | " + maxStack + " | "  
            + hardness + " | " + falls + " | " + grade);  
    }  
}
```

Output

```
Grass Block | 64 | 0.6 | false | C  
sand | 64 | 0.5 | true | B  
Obsidian | 64 | 50.0 | false | S  
Diamond Ore | 64 | 3.0 | false | A
```

The same set of variables is intentionally reused here. When the output is complete and then updated, the old rows still retain the old snapshot. Check each box: Did you forget to update a field?

12 / 12

Find reasons for your type selection



The clearer the data description, the more reliable subsequent calculations and judgments will be.

The ability map of the third lesson

Step	Explanation
1. Distinguish	Numeric value or text
2. Options	Integer or decimal
3. Check	Quotes and suffixes
4. Conversion	Check information loss
5. Explanation	Prediction and re-verification

- Basics: Save a pet's name, life, number of items, adult status and rating.
- Advanced: Compare $9 / 4$, $9.0 / 4$, (double) $9 / 4$; the result is 2, 2.25, 2.25.
- Creation: Design a data card for the mysterious square, and add a boolean field for whether it emits light.

Self-test: Is String a basic type? (int) 12.8 will get 13? The answer is no. The next lesson, Miner's Calculator, will organize this data into arithmetic, comparison, and logical expressions.

Technical reference: Java SE 11 Language Specification, Chapter 4 Data Types, Chapter 5 Type Conversion.

Official documentation: docs.oracle.com/javase/specs/jls/se11/html/.