

# Cursor and Calculator

Expressions and output



# Calculator vs. Java

PAUSE AND TRY

**Make a guess for each. Be ready to explain.**

## QUESTION 1

A calculator says  $7 / 2$  is 3.5. What do you think Java says?

## QUESTION 2

50 eggs go into cartons of 12. How many full cartons? How many eggs are left over?

## By the end of class you can...

- 1 Predict output from `print` and `println` by tracking the cursor.
- 2 Use string literals and the escape sequences `\`", `\\`", and `\n`.
- 3 Tell whether an arithmetic result is an `int` or a `double`, and find its value.
- 4 Use integer division and the remainder operator, `%`.
- 5 Apply operator precedence, and explain what dividing an `int` by `0` does.

A large white square with rounded corners, tilted slightly to the right, containing a large black number '1' in the center.

# 1

PART 1

# Output and the cursor

# println: display, then move to the next line

## INSIDE MAIN

```
System.out.println("Today's lunch:");  
System.out.println("Veggie tacos");  
System.out.println();  
System.out.println("Enjoy!");
```

## OUTPUT

```
Today's lunch:  
Veggie tacos  
  
Enjoy!
```



**System.out.println** displays information on the console, then moves the cursor to a new line. An empty **println()** just moves the cursor.

| After line | Cursor is at            |
|------------|-------------------------|
| 1          | start of console line 2 |
| 2          | start of console line 3 |
| 3          | start of console line 4 |
| 4          | start of console line 5 |

KEY IDEA

# print stays on the line

INSIDE MAIN

```
System.out.print("Score: ");
System.out.println(95);
System.out.print("Level:");
System.out.println(3);
```

OUTPUT

Score: 95  
Level:3

| Statement    | Displays? | Then a new line?     |
|--------------|-----------|----------------------|
| print(...)   | Yes       | No: cursor stays put |
| println(...) | Yes       | Yes                  |



The difference shows up in the **next** output, not the current one.

# Where is the cursor?

PAUSE AND TRY

**Write the exact output. Is there an empty line?**

```
System.out.print("X");  
System.out.println("Y");  
System.out.print("Z");  
System.out.println();  
System.out.println("W");
```

Track the cursor after every statement. An empty `println()` only makes an empty line if the cursor is already at the start of one.

A large, bold, teal-colored number '2' is centered within a white rounded square that has a slight 3D effect and a shadow.

PART 2

# Literals and escape sequences

# A literal is a fixed value in the code

## INSIDE MAIN

```
System.out.println(42);  
System.out.println("42");  
System.out.println(2.50);  
System.out.println(7.0);  
System.out.println("2.50");
```

## OUTPUT

```
42  
42  
2.5  
7.0  
2.50
```

| Literal | Kind                                         |
|---------|----------------------------------------------|
| 42      | int literal                                  |
| 2.50    | double literal                               |
| true    | boolean literal                              |
| "42"    | <b>string literal: text in double quotes</b> |

# Escape sequences start with a backslash

| Escape          | Displays       |
|-----------------|----------------|
| <code>\"</code> | a double quote |
| <code>\\</code> | one backslash  |
| <code>\n</code> | a new line     |

An **escape sequence** is two characters in a string that stand for one special character.

## INSIDE MAIN

```
System.out.println("She said \"hi\" to me.");  
System.out.println("A backslash: \\");  
System.out.println("Ready,\nset, go!");
```

## OUTPUT

```
She said "hi" to me.  
A backslash: \  
Ready,  
set, go!
```

# One backslash in a file path



Inside a string literal, every backslash starts an escape sequence. To display one backslash, write **two**.

## WRONG

```
System.out.println("C:\new");
```

## OUTPUT

```
C:  
ew
```

## FIXED

```
System.out.println("C:\\new");
```

## OUTPUT

```
C:\new
```

The n in "new" vanished: \n became a new line.

A large, bold, teal number '3' is centered within a white, rounded square that has a slight 3D effect and a shadow.

PART 3

# Arithmetic expressions

# The operands decide the result's type

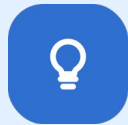
| Op | Meaning   |
|----|-----------|
| +  | add       |
| -  | subtract  |
| *  | multiply  |
| /  | divide    |
| %  | remainder |

## INSIDE MAIN

```
System.out.println(6 * 2);  
System.out.println(6 * 2.0);  
System.out.println(1.5 + 2.5);  
System.out.println(3 * 12);  
System.out.println("3 * 12");
```

## OUTPUT

```
12  
12.0  
4.0  
36  
3 * 12
```



Two **int** values give an **int**. At least one **double** gives a **double**.

# Whole groups and leftovers

## A 135-MINUTE MOVIE

```
System.out.println(135 / 60);  
System.out.println(135 % 60);  
System.out.println(135 / 60.0);
```

## OUTPUT

```
2  
15  
2.25
```

% is not percent.

## CHECK IT

$$135 = 60 * 2 + 15$$

/ with two ints: how many whole groups. No rounding.

% the remainder: what is left over after the whole groups.

**2 hours and 15 minutes.**

# Half a cup is not 1 / 2



Dividing two **int** values keeps only the integer part of the quotient. Nothing warns you.

## A RECIPE NEEDS HALF A CUP

```
System.out.println(1 / 2);
```

## OUTPUT

0

## ONE DOUBLE FIXES IT

```
System.out.println(1 / 2.0);
```

## OUTPUT

0.5

Integer division never rounds: 0.5 becomes 0, and 0.99 would be 0 too.

## Whole part, leftover, or full answer?

PAUSE AND TRY

Find each value. Is it an int or a double?

$47 / 10$

$47 \% 10$

$47 / 10.0$

$6 \% 8$

$6 / 8$

$10 \% 5$

A large, bold, teal number '4' is centered within a white square with rounded corners. The square is slightly tilted to the right.

# 4

## PART 4

# Precedence and dividing by zero

# Which operation goes first?

| Expression               | First                     | Then                | Result    |
|--------------------------|---------------------------|---------------------|-----------|
| <code>2 + 3 * 4</code>   | <code>3 * 4 is 12</code>  | <code>2 + 12</code> | <b>14</b> |
| <code>(2 + 3) * 4</code> | <code>2 + 3 is 5</code>   | <code>5 * 4</code>  | <b>20</b> |
| <code>20 - 6 - 4</code>  | <code>20 - 6 is 14</code> | <code>14 - 4</code> | <b>10</b> |
| <code>24 / 4 * 2</code>  | <code>24 / 4 is 6</code>  | <code>6 * 2</code>  | <b>12</b> |

## PRECEDENCE

1. `()` parentheses
2. `*` `/` `%`
3. `+` `-`

Same level: left to right

Java decides int or double one operation at a time.

# Dividing an int by 0

## INSIDE MAIN

```
int guests = 0;  
System.out.println("Splitting 30 slices...");  
System.out.println(30 / guests);  
System.out.println("Done!");
```



Dividing an **int** by the **int** zero throws an **ArithmeticException** while the program runs.

## OUTPUT

```
Splitting 30 slices...  
Exception in thread "main" java.lang.ArithmeticException: / by zero
```

# Precedence check

PAUSE AND TRY

Find each value. Which operation happens first?

$$10 - 2 * 3$$

$$(10 - 2) * 3$$

$$18 / 3 / 2$$

$$7 + 8 \% 3$$

# Words to know

## String literal

Characters in double quotes, displayed as written.

## Escape sequence

A backslash code in a string:  
`\"` `\\` `\n`

## Integer division

`int / int` keeps only the integer part. No rounding.

## Remainder %

What is left over after the whole groups.

## Precedence

`*` / `%` before `+` `-`. Same level: left to right.

## ArithmeticException

Thrown when an `int` is divided by the `int 0`.

## Back to the eggs



**Answer the warm-up in Java: 50 eggs, cartons of 12. Then explain why  $7 / 2$  is not 3.5.**

SENTENCE STARTER

$7 / 2$  is \_\_\_\_\_ because both operands are \_\_\_\_\_, so Java keeps only \_\_\_\_\_.

Next: the Cursor and Calculator puzzle worksheet.