

# Expressions

zyBook Chap 2.10, 2.11, 2.12, 2.13, 2.14

# Expression

A simple value or set of operations that produces a value

- **Operator** → indicates the operation to be performed
- **Operand** → value in the expression
- E.g.
  - **(3 + 29) - (4 \* 5)**

# Arithmetic Operators

- Addition Operator: +
- Subtraction Operator: -
- Multiplication Operator: \*
- Division Operator: /
- Remainder Operator: %

# Division & Remainder – int

$$\text{Dividend} = \text{Divisor} \times \text{Quotient} + \text{Remainder}$$

Q: Find the resulting value of ...

- $1 / 4 = 0$  ( $1 = 4 \times 0 + 1$ )
- $1 \% 4 = 1$  ( $1 = 4 \times 0 + 1$ )
- $0 / 4 = 0$
- $0 \% 4 = 0$
- $101 / 4 = 25$
- $101 \% 4 = 1$

# Division & Remainder – double

Q: Find the resulting value of ...

- $0.77 / 0.25 = 3.08$  (  $0.77 = 0.25 \times 3.08$  )
- $0.77 \% 0.25 = 0.02$  (  $0.77 = 0.25 \times 3 + 0.02$  )

With the **remainder** operator, Java will try to find how many times the dividend **completely (whole number)** goes into the divisor; and then generates the **remaining value**.

# Precedence

- Precedence:
  - The binding power of an operator, which determines how to group parts of an expression. That is, the order of evaluating the operations
- **Evaluate left to right.** Therefore, if two operations are at the same precedence order, evaluation from left to right, and
  1. Parentheses: ()
  2. Unary operators: +, -
  3. Multiplicative operators: \*, /, %
  4. Additive operators: +, -

Precedence:

1. Parentheses: ()
2. Unary operators: +, -
3. Multiplicative operators: \*, /, %
4. Additive operators: +, -

Q:  $50 - 7 * 5 \% 2 + (13 / 6)$

Diagram illustrating the evaluation of the expression  $50 - 7 * 5 \% 2 + (13 / 6)$  using operator precedence:

- Step 1: Evaluate the multiplication  $7 * 5$  to get 35.
- Step 2: Evaluate the modulo operation  $35 \% 2$  to get 1.
- Step 3: Evaluate the division  $13 / 6$  to get 2.
- Step 4: Evaluate the final expression  $50 - 1 + 2$  to get 51.

The final result is  $50 - 1 + 2 = 51$ .

# Mixing Types – Promotion/Coercion

## Promotion

- A **widening** primitive **conversion** that **does not lose** information about the value
  - E.g., converting an integer 4 to a double 4.0 does not lose any information
- **Occurs automatically** to the integers operands whenever there is at least one operand that is **double**
  - E.g., **23.0** / **4**  $\rightarrow$  23.0 / **4.0** = 5.75



Precedence:

1. Parentheses: ()
2. Unary operators: +, -
3. Multiplicative operators: \*, /, %
4. Additive operators: +, -

Q:  $5.0 / ( 6 - 4 \% 6 )$

Diagram illustrating the evaluation of the expression  $5.0 / ( 6 - 4 \% 6 )$  using operator precedence:

- The innermost operation is  $4 \% 6$ , which evaluates to 4.
- The next operation is  $6 - 4$ , which evaluates to 2.
- The final operation is  $5.0 / 2.0$ , which evaluates to 2.5.

Q:  $7 / 3 * 1.2 + 3 / 2$

Diagram illustrating the evaluation of the expression  $7 / 3 * 1.2 + 3 / 2$  using operator precedence:

- The first operation is  $7 / 3$ , which evaluates to 2.0.
- The next operation is  $2.0 * 1.2$ , which evaluates to 2.4.
- The second division is  $3 / 2$ , which evaluates to 1.0.
- The final operation is  $2.4 + 1.0$ , which evaluates to 3.4.

# Mixing Types – Casting

## Casting

- A **narrowing** primitive **conversion** that **may lose** information about the value (**truncating**)
  - E.g., converting a double 4.1 to an integer 4 loses the information after the decimal point
- **Requires cast** via the syntax of **(target type) <value>**
  - **(int)** 4.16 = 4
  - **(int)** 4.75 = 4

# Mixing Types – Casting

## Casting

- Only casts value **immediately following cast**
  - $23 / 2 = 11$
  - **(double) 23 / 2**
    - **23.0 / 2** (23 is cast to double, that is, 23.0)
    - **23.0 / 2.0 = 11.5** (2 is automatically promoted to 2.0 since there's a double 23.0 in the expression)
  - **(double) (23 / 2)**
    - **(double) 11 = 11.0** (parentheses have the highest precedence)

**Q:** Assuming there are books that are 0.15 feet wide, write an expression that evaluates the number of books that will fit on a bookshelf that is 2.5 feet wide.

`(int) (2.5 / 0.15) = 16`

# General Rule

- When the arithmetic operators are performed on two integers, the result will be an integer.
- When an arithmetic operation is performed on **at least** one real number (double), the result will be a real number (double).