Chapter 2:
Variables / Assignments (and Expressions)

Variables and Assignments

- In programming, a *variable* refers to a memory location that allows us to store a value
- We can
  - Assign a value to the memory location
  - Change the value assigned to the location
- Note that we cannot erase a value; some value is always there

Identifiers and Variable naming rules

- Variable names are called identifiers
- Choosing variable names
  - Use *meaningful* names that represent data to be stored
  - First character must be a letter, dollar sign, or underscore character
  - Remaining characters must be

    letters numbers underscore character dollar sign
- Which of the following are valid variable names?

  ounces _ounces ounces* 1_ounce
  ounces-per-gallon ounces_per_gallon

Keywords

- Keywords (also called reserved words)
  - Already have special meaning in Java and must be used in this way
  - Cannot be used as identifiers
- Can you give an example of a keyword?

Declaring Variables

- Before a variable can be used, it must be declared
- Declaration syntax:
  - Type_name Variable_1 , Variable_2, . . . ;
- Declaration Examples:
  - double average, m_score, total_score;
  - double moon_distance;
  - int age, num_students;
  - int cars_waiting;

Some variable types

Type	Description	Example	Input

int	An integer between ~ +/- 2 billion	int num;
		int sum = 0;
			scnr.nextInt()

double	A floating point number	double num;
		double val = 2.3;
			scnr.nextDouble()

boolean	A true/false value	boolean happy = true;	scnr.nextBoolean()
char	A single character from the keyboard	char letter = 'a';	scnr.next().charAt(0)

String*	A sequence of characters	String class = "CSC-210";
		String name = "Will Foster"
			scnr.next()
			scnr.nextLine()

*Technically, a String is an *object* and not a variable. An object stores data and can also contain methods that act on the object, as in the code below:

```
String name = “Will Foster”;
int len = name.length();   // assigns the length of the string to variable len
```

Assignment Statements

- An assignment statement changes the value of a variable
  - total_weight = one_weight + number_of_bars;
    - total_weight is set to the sum one_weight + number_of_bars

  - The single variable to be changed is **always on the left** of the assignment operator ‘=‘

  - On the right of the assignment operator, we can have a/an
    - Literal: age = 21;
    - Variable: my_cost = your_cost;
    - Expression: circumference = diameter * 3.14159;

Assignment Statements and Algebra

- The ‘=‘ operator in Java is an *assignment* operator, and *not* an equal sign
- The statement

```
x =  x  +  3;
```

  - means the new value of *x* is the previous value of *x* plus 3 (i.e., *x* is increased by 3)
  - But is not true algebraically

2.4. Arithmetic Expressions

Arithmetic

- Arithmetic is performed with operators
  - + for addition
  - - for subtraction
  - * for multiplication
  - / for division

- Example: storing a product in the variable distance

```
distance  =  rate * time;
```

Arithmetic Expressions

- Use spacing to make expressions readable
  - Which is easier to read?

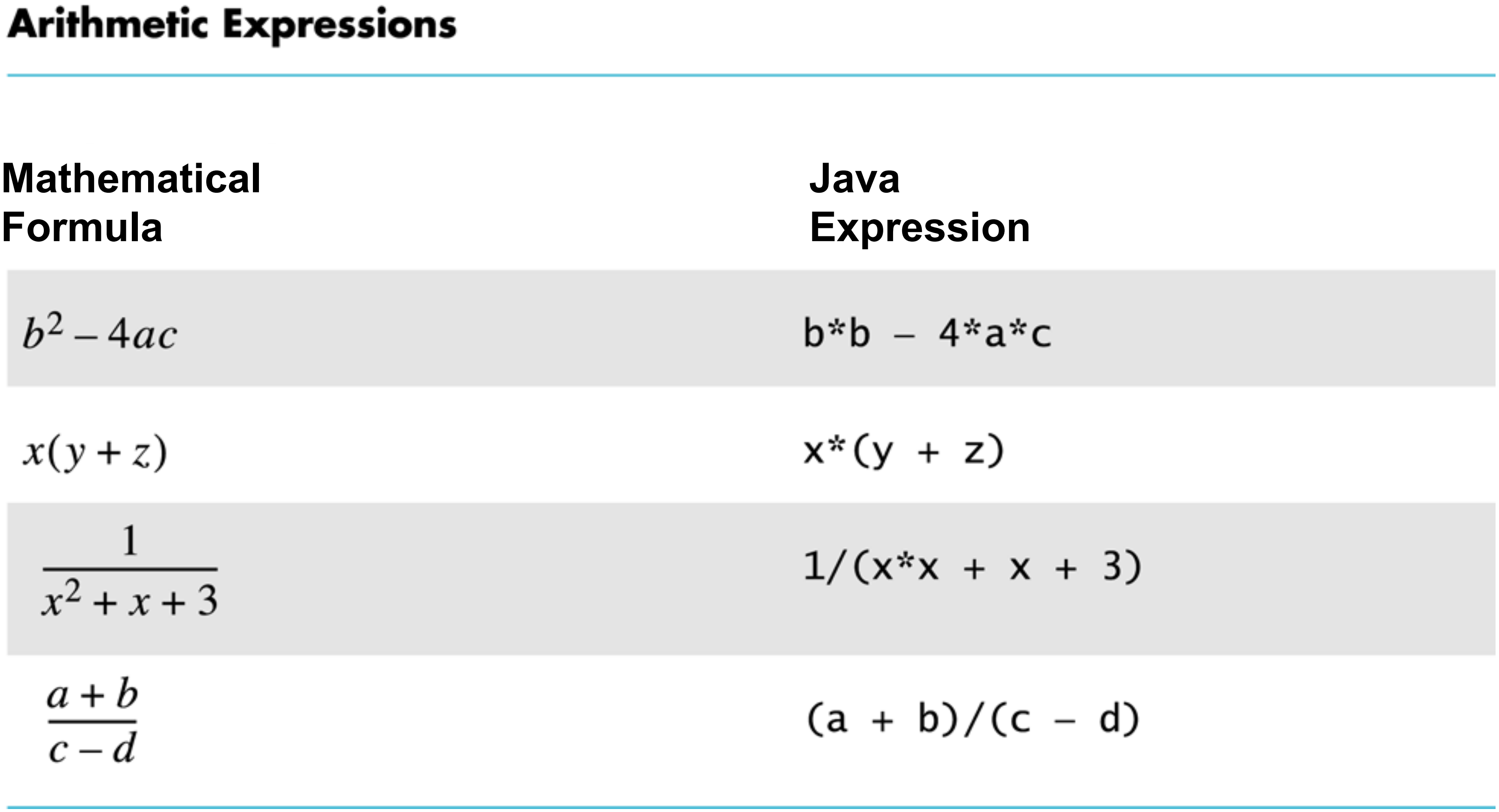
x+y*z   or   x + y * z

- Precedence rules for operators are the same as used in your algebra classes
- Use parentheses to alter the order of operations

  x + y * z   (y is multiplied by z first)

  (x + y) * z   (x and y are added first)

Arithmetic Expressions



Arithmetic Expressions

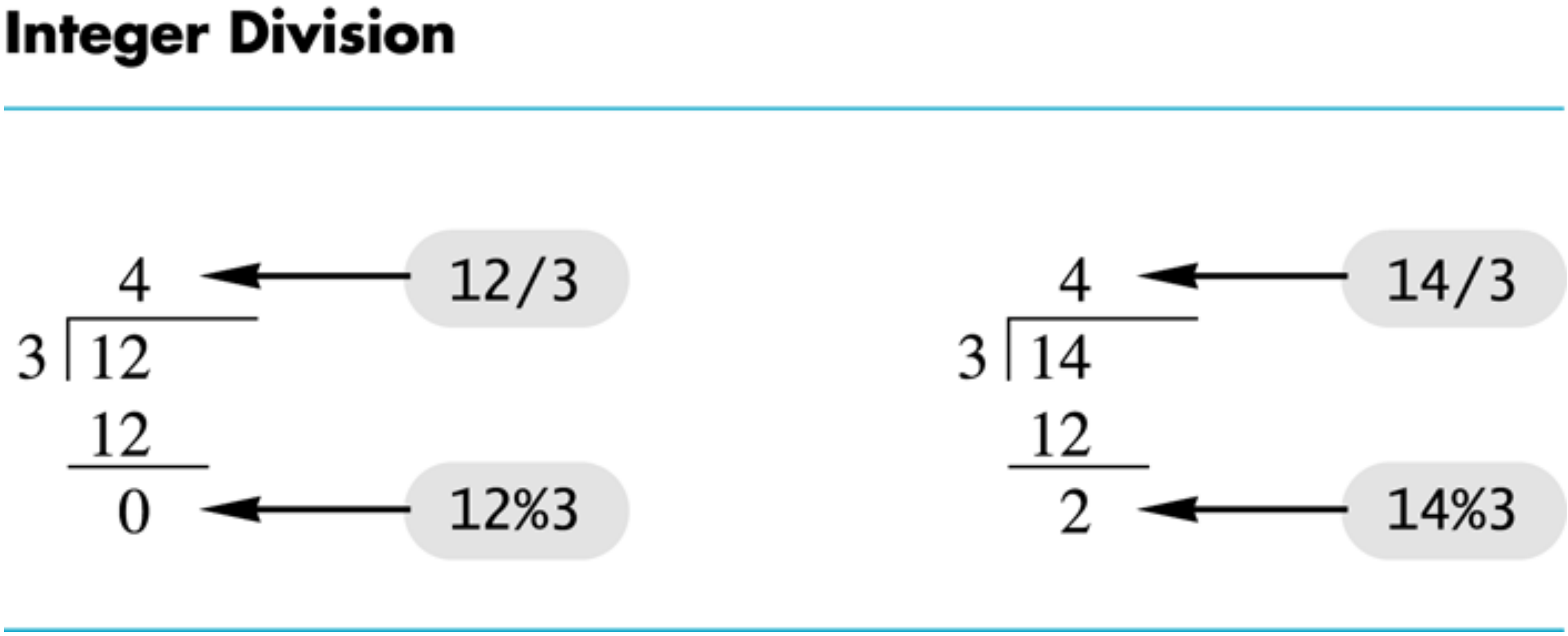
Mathematical Formula	Java Expression
$b^2 - 4ac$	`b*b - 4*a*c`
$x(y + z)$	`x*(y + z)`
$\frac{1}{x^2 + x + 3}$	`1/(x*x + x + 3)`
$\frac{a + b}{c - d}$	`(a + b)/(c - d)`

Results of Operators

- Arithmetic operators can be used with any numeric type
- An operand is a number or variable used by the operator
- Result of an operator depends on the types of operands (see *integerDivision.java* example)
    - **If both operands are of type *int*, the result is an *int***
    - If one or both operands are of type *double*, the result a *double*

Integer Division and Remainders

- The modulus operator (%) gives the remainder from integer division



Integer Division

Compound Operators (Shortcut Expressions)

- Some expressions occur so often that Java contains shorthand operators for them

  - count += 2; // same as count = count + 2;
  - count -= 2; // same as count = count – 2;
  - num *= 2; // same as num = num *2;
  - num /= 2; // same as num = num / 2;
  - num %= 2; // same as num = num % 2;

  - count++; // same as count = count + 1;
  - count--; // same as count = count – 1;

Constants

- final is the keyword to declare a constant
- Example:

  final int WINDOW_COUNT = 10;

  declares a constant named WINDOW_COUNT
  - Its value cannot be changed by the program like a variable
  - It is common to name constants with all capitals