420-101-VA: Programming 1

WEEK 2: ARITHMETIC OPERATIONS

Arithmetic Operators

- There are many operators for manipulating numeric values and performing arithmetic operations
- An *expression* is a combination of one or more operators and operands

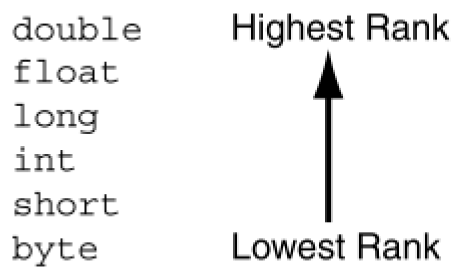
Operator	Meaning	Type	Example
+	Addition	Binary	`total = cost + tax;`
-	Subtraction	Binary	`cost = total - tax;`
*	Multiplication	Binary	`tax = cost * rate;`
/	Division	Binary	`salePrice = original / 2;`
%	Modulus	Binary	`remainder = value % 5;`

Arithmetic Operators

- The operators are called binary operators because they must have two operands.
- Each operator must have a left and right operator.
- It is an error to try to divide any number by zero.
- When working with two integer operands, the division operator requires special attention.
- If either or both operands are floating point values, then the result is a floating point value

Integer Division

- Division can be tricky. For example, in a Java program, what is the value of ½?
  - You might think the answer is 0.5… But that’s wrong. The answer is simply 0.
- Integer division will truncate any decimal remainder.
- Java cares about accuracy. In this case, if you have two numbers with different data types, then Java will automatically convert the number with lower accuracy to the data type with higher accuracy.
  - int + double equals double
  - int + long equals long
  - byte + int = int
  - char + char equals int
  - char + int equals int
  - char + float equals float



Division and Remainder

- If both operands to the division operator (`/`) are integers, the result is an integer (the fractional part is discarded)

`14 / 3` **equals** **4**

`8 / 12` **equals** **0**

- The remainder operator (%) returns the remainder after dividing the first operand by the second

`14 % 3` **equals** **2**

`8 % 12` **equals** **8**

Arithmetic Operation: %

Why we need to learn Modulus?

- Well, in computer science, you might need to use mod for many algorithms:

For example:

- If you have an int number, and you want to know whether it is even or odd, then you can modulus it by 2, if the remaining part is 0, then it is even, else if it is 1, then it is odd.

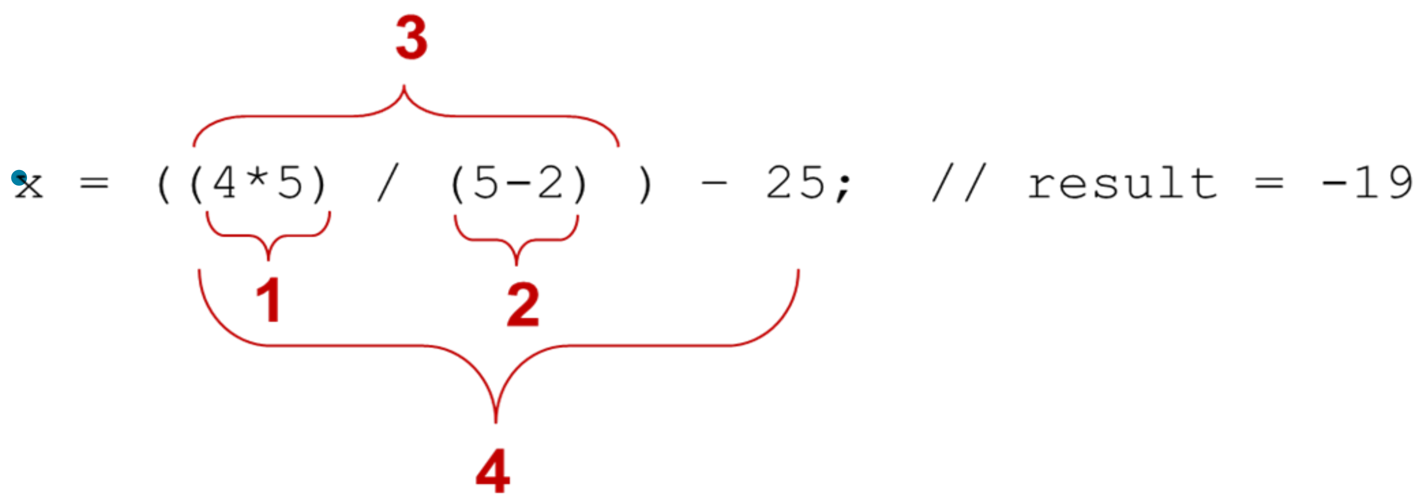
Operator Precedence

- Mathematical expressions can be very complex.
- There is a set order in which arithmetic operations will be carried out.

Operator	Associativity	Example	Result
`* / %`	Left to right	`x = -4 + 4 % 3 * 13 + 2;`	11 −1
+ −	Left to right	`x = 6 + 3 - 4 + 6 * 3;`	23

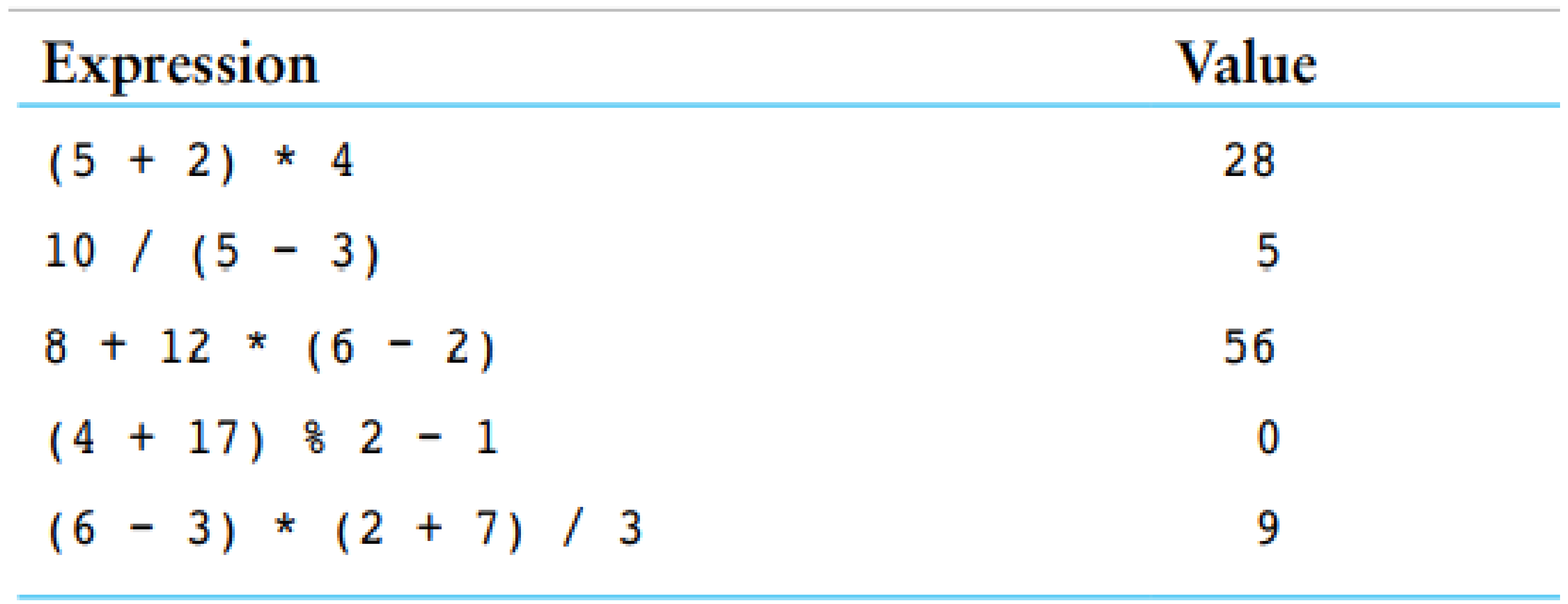
Grouping with Parenthesis

- When parenthesis are used in an expression, the inner most parenthesis are processed first.
- If two sets of parenthesis are at the same level, they are processed left to right.
- Java style guide suggests to add spaces around binary operators (operators with 2 numbers)



```
x = ((4*5) / (5-2) ) - 25;  // result = -19
```

Expressions and their values



Expression	Value
`(5 + 2) * 4`	28
`10 / (5 - 3)`	5
`8 + 12 * (6 - 2)`	56
`(4 + 17) % 2 - 1`	0
`(6 - 3) * (2 + 7) / 3`	9

Quick Check

In what order are the operators evaluated in the following expressions?

```
a + b + c + d + e
  1   2   3   4
```

```
a + b * c - d / e
  3   1   4   2
```

```
a / (b + c) - d % e
  2    1    4   3
```

```
a / (b * (c + (d - e)))
  4    3    2     1
```

Assignment Revisited

- The assignment operator has a lower precedence than the arithmetic operators

First the expression on the right hand side of the = operator is evaluated

```
answer  =  sum / 4 + MAX * lowest;
        4      1   3     2
```

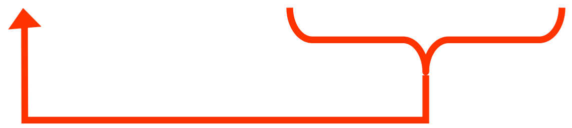
Then the result is stored in the variable on the left hand side

Assignment Revisited

- The right and left hand sides of an assignment statement can contain the same variable

First, one is added to the original value of `count`

```
count  =  count + 1;
```



Then the result is stored back into `count` (overwriting the original value)

Combined Assignment Operators (1 of 4)

- Java has some combined assignment operators.
- These operators allow the programmer to perform an arithmetic operation and assignment with a single operator.
- Often, we perform an operation on a variable, and then store the result back into that variable. The *assignment operators* simplify that process
- Although not required, these operators are popular since they shorten simple equations.

Combined Assignment Operators (2 of 4)

Operator	Example	Equivalent	Value of variable after operation
+=	`x += 5;`	`x = x + 5;`	The old value of **x** plus 5.
`-=`	`y -= 2;`	`y = y - 2;`	The old value of **y** minus 2
***=**	`z *= 10;`	`z = z * 10;`	The old value of **z** times 10
/=	`a /= b;`	`a = a / b;`	The old value of **a** divided by **b**.
%=	`c %= 3;`	`c = c % 3;`	The remainder of the division of the old value of **c** divided by 3.

Combined Assignment Operators (3 of 4)

- Example 1 : the salary of an employee’s increases 5% each year.

```
double salary = 3141.5;
double increaseRatio = 0.05;
salary *= 1 + increaseRatio;
// the same as: salary = salary * (1 + increaseRatio)
```

◈Notice: You should write 5% as 0.05, because % is modulo in Java.

Combined Assignment Operators (4 of 4)

- Example 2: Assume you are working in a museum, and your job is to check how many people visit the museum per day. So, you stand at the entrance, press the button of a counter in your hand each time when a visitor enters, so the number on the counter add one.

```
int counter = 0;            // no one at first
counter += 1;     //   read the value of counter, add 1, and save it back
```

- Example 3: a timer of 60 seconds decreases by 1 each second until it reach 0.

```
int timer = 60;                        // 60 at first
timer -= 1;                          // read the value of timer, minus 1, and save it
back
```

Increment and Decrement

- The increment (++) and decrement (−−) operators use only one operand
- The statement

```
count++;
```

is functionally equivalent to

```
count = count + 1;
```

Increment and Decrement

- The increment and decrement operators can be applied in *postfix form*:

```
count++
```

- or *prefix form*:

```
++count
```

- When used as part of a larger expression, the two forms can have different effects
- Because of their subtleties, the increment and decrement operators should be used with care

Increment and Decrement

- ++: read the value, add 1, and store it back
- --: read the value, minus1, and store it back

```
int counter = 0;
counter += 1;                     // read the value of counter, add 1, and save it back
counter++;                        // read the value of counter, add 1, and save it back
counter = counter + 1             // read the value of counter, add 1, and save it back
```

```
int timer = 60;                   // 60 at first
timer -= 1;                       // read the value of timer, minus 1, and save it back
timer--;                          // read the value of timer, minus 1, and save it back
timer = timer - 1                 // read the value of timer, minus 1, and save it back
```

- For ++ and --, the value 1 is fixed.

Increment and Decrement

- If there is an “=” in a statement, then there is a difference between the two.

- ++ after: first use the original value of a variable (calculate the equation without the ++), then increase the variable by 1.
- ++ before: first increase the variable by 1, then use the updated value to calculate the equation.

```
int num = 10;
int num2 = 3 + num++;

// num2 = 3 + num;          // num = 10, num2 = 13
// num = num + 1;           // num = 11, num2 = 13
```

```
int num = 3;
int num2 = 3 + ++num;

// num = num + 1;           // num = 4
// num2 = 3 + num;          // num = 4, num2 =  7
```

Hands-on

- Suppose you earn $6,000 per month and you are allowed to contribute a portion of your gross monthly pay to a retirement plan. You want to determine the amount of your pay that will go into the plan if you contribute 5 percent, 8 percent, or 10 percent of your gross wages.

Try it out

- Consider the following scenario: A retail business offers a product with a standard price of $59. The business intends to initiate a sale during which the price of the product will be discounted by 20 percent. Your task is to develop a program that computes the sale price of the product. This involves two key calculations:

  - Calculate the discount amount, equivalent to 20 percent of the regular price of the product.
  - Deduct the discount amount from the regular price to determine the final sale price.