# Arithmetic Operators

Now that you've written a few programs, let's take a step back and discuss how to do basic arithmetic. The behavior of Java operators (+, -, *, /, %) depends on the type of data you have.

Manager: Recorder:

Presenter: Reflector:

## Content Learning Objectives

*After completing this activity, students should be able to:*

- Summarize the benefits of working as a team.
- Evaluate Java expressions that use the % operator with integers.
- Explain the difference between integer and floating-point division.
- Identify operator precedence for addition, division, and assignment.

## Process Skill Goals

*During the activity, students should make progress toward:*

- Recognizing mathematical operations based on tables. (Information Processing)

## Facilitation Notes

This activity focuses on integer vs floating-point division and operator precedence. Students should understand how to declare and assign variables before completing this activity. The meta activity at the beginning reinforces the idea of working together as a team.

#10 refers to the instructor hypothetically giving mints to teams to divide among themselves. Consider bringing real mints to build rapport with the class. You can use other small objects like paperclips. When reporting out, introduce the terms ***remainder operator***, ***modulo operation***, and ***modulus***. Explain that "x % y" is pronounced "x mod y".

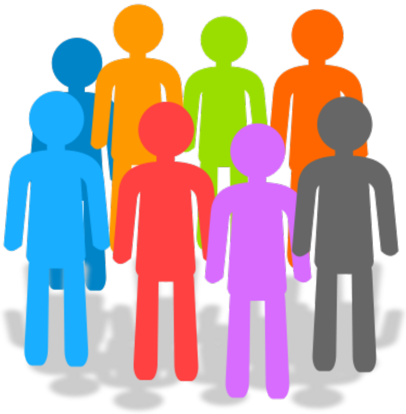
On Model 2, you may need to point out (after students work for a while) that it makes no difference whether a floating-point number ends with a zero. In other words, `4.` is the same value as `4.0`. For #22, have at least 2–3 teams share their answers with the class to encourage different ways of thinking about the concepts (and how to put them into words).

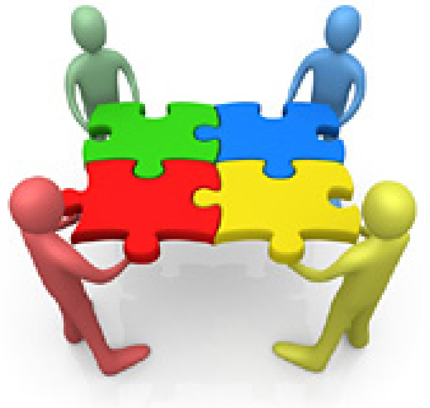
On Model 3, help students understand that assignment is an operation (in contrast to the equals sign in mathematics). At the end of the activity, invite teams to discuss what the ***assignment operation*** is in terms of what computer hardware does.

Key questions: #11, #22, #28

# Meta Activity: Group vs Team

Throughout the course, you will need to examine and process information, ask and answer questions, construct your own understanding, and develop new problem-solving skills.





## Questions (8 min)

**Start time:**

**1**. What are some advantages to working in groups/teams?

You get to know other people and make new friends. Different people have different backgrounds and skills. The responsibility is shared.

**2**. What are some disadvantages to working in groups/teams?

Some group members may decide not to contribute. One or two students may be absent. People may not always get along with each other.

**3**. Based on the images above, what is the difference between a group and a team? Come up with a precise answer.

A group is students who just sit by each other and turn in the same assignment. A team actually works together toward a common goal, drawing on each other's strengths.

**4**. How can working as a team help you accomplish the tasks described above? Give at least two specific examples.

Working as a team makes it easier to examine and process information, because different people have different perspectives. We can also develop new problem-solving skills by observing how each other approaches the problems.

## Model 1 The % Operator

| | | |
|---|---|---|
| 9 / 4 | *evaluates to* | 2 |
| 10 / 4 | *evaluates to* | 2 |
| 11 / 4 | *evaluates to* | 2 |
| 12 / 4 | *evaluates to* | 3 |
| 13 / 4 | *evaluates to* | 3 |
| 14 / 4 | *evaluates to* | 3 |
| 15 / 4 | *evaluates to* | 3 |
| 16 / 4 | *evaluates to* | 4 |

| | | |
|---|---|---|
| 9 % 4 | *evaluates to* | 1 |
| 10 % 4 | *evaluates to* | 2 |
| 11 % 4 | *evaluates to* | 3 |
| 12 % 4 | *evaluates to* | 0 |
| 13 % 4 | *evaluates to* | 1 |
| 14 % 4 | *evaluates to* | 2 |
| 15 % 4 | *evaluates to* | 3 |
| 16 % 4 | *evaluates to* | 0 |

### Questions (12 min) Start time:

**5.** Which numbers % 4 evaluate to 0 in the table above? If the table were extended to include more rows, which other numbers % 4 would evaluate to 0?

12 and 16. Other numbers include 0, 4, 8, 20, 24.

**6.** Look at the expressions in the second table that evaluate to 1. How do the left operands in these expressions (9, 13, 17) differ from those that evaluate to 0?

They differ by one.

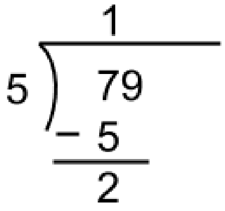
**7.** List three numbers % 5 that will evaluate to 0 and three numbers % 5 that will evaluate to 2.

0, 5, 10 and 2, 7, 12.

**8.** Evaluate the following Java expressions:

| | | | |
|---|---|---|---|
| 18 % 4 | 19 % 4 | 19 % 5 | 19 % 6 |
| = 2 | = 3 | = 4 | = 1 |

**9**. Consider how you were taught to do long division in elementary school. Finish solving for 79 ÷ 5. What is the answer?



We first bring the 9 down, then 5 goes into 29 five times, and so we subtract 25. The final answer is 15 remainder 4.

**10**. Imagine that you are given candy mints to divide evenly among your team members.

a) If your team receives 11 mints, how many mints are left over?

`11 % 4` is 3 or `11 % 5` is 1

b) If your team receives 2 mints, how many mints are left over?

`2 % 4` is 2 or `2 % 5` is 2

**11**. Describe what the % operator does. How are the / and % operators related?

Both operators divide two numbers: % returns the remainder, and / returns the quotient.

**12**. Would it make sense to apply the % operator to real numbers? Why or why not?

Not really, since the concept of remainder is based on integer division.
(BTW floating-point modulo is defined in IEEE 754, but its use is rare.)

## Model 2 Dividing Numbers

| | | |
|---|---|---|
| 9 / 4 | *evaluates to* | 2 |
| 10 / 4 | *evaluates to* | 2 |
| 11 / 4 | *evaluates to* | 2 |
| 12 / 4 | *evaluates to* | 3 |
| 13 / 4 | *evaluates to* | 3 |
| 14 / 4 | *evaluates to* | 3 |
| 15 / 4 | *evaluates to* | 3 |
| 16 / 4 | *evaluates to* | 4 |

| | | |
|---|---|---|
| 9 / 4.0 | *evaluates to* | 2.25 |
| 10 / 4. | *evaluates to* | 2.5 |
| 11. / 4 | *evaluates to* | 2.75 |
| 12 / 4.0 | *evaluates to* | 3.0 |
| 13 / 4. | *evaluates to* | 3.25 |
| 14.0 / 4 | *evaluates to* | 3.5 |
| 15 / 4.0 | *evaluates to* | 3.75 |
| 16 / 4. | *evaluates to* | 4.0 |

### Questions (15 min)

**Start time:**

**13**. In the first table, which number(s) divided by 4 evaluate to 3? What is significant about the number of answers you have written down?

12, 13, 14, and 15. There are four answers, because we divided by four.

**14**. How do the answers in the first table differ from the second table?

The second table is mathematically correct; the first table rounds down.

**15**. To the right of the second table, round each answer to the closest integer. How do those values compare to what you see in the first table?

The pattern is off by two rows. The 0.5 and 0.75 values round up, but in the first table they don't.

**16**. Carefully explain the difference between the number formats in the first and second tables.

All the numbers in the first table have no decimal points. At least one number in each row of the second table as a decimal point.

**17**. Complete the table:

| | | |
|---|---|---|
| 14. / 4. | *evaluates to* | 3.5 |
| 14. / 4 | *evaluates to* | 3.5 |
| 14 / 4. | *evaluates to* | 3.5 |
| 14 / 4 | *evaluates to* | 3 |

**18**. Dividing real numbers (also known as ***floating-point*** numbers) gives you different results from dividing integers. In the previous question:

a) Which rows evaluate to an integer? only the last row

b) Which rows evaluate to a real number? the first three rows

c) When will Java perform ***integer division***? when both numbers are integers

**19**. Imagine you are writing a Java program that requires division.

a) What must be true about the ***operands*** (the numbers before and after the operator) for you to get the mathematically correct answer?

At least one of them needs to be a floating-point number.

b) Does it need to be true for both operands? No

**20**. Consider what you know about addition (+). If you add two integers in Java, will the result always be mathematically correct? Justify your answer.

Yes, because adding two integers always results in an integer. (Unless the number gets too large and the arithmetic overflows.)

**21**. What about subtraction (-) and multiplication (*)? If you subtract or multiply two integers in Java, will the answer always be mathematically correct? Justify your answer.

Yes, because subtraction and multiplication can be rewritten in terms of addition. Division is the only special case, because it has both a quotient and a remainder.

**22**. Summarize what you have learned about the difference between integer division and floating-point division.

Integer division simply ignores the decimal part of the answer, whereas floating-point division gives you the mathematically correct answer.

## Model 3 Order of Operations

Java follows a specific order for math and other operations. For example, multiplication and division take ***precedence*** over addition and subtraction. The following table lists several Java operators from highest precedence to lowest precedence.

| | |
|---|---|
| Parenthesis | `( )` |
| Unary (positive or negative signs) | `+ -` |
| Multiplicative | `* / %` |
| Additive | `+ -` |
| Assignment | `=` |

For the following questions, assume you have these two variables:

```
int x;
double y;
```

### Questions (10 min) Start time:

**23.** What operator has the lowest precedence? Why do you think Java is designed that way?

Assignment has the lowest precedence so that all other operations happen first (before the final value is stored in memory).

**24.** What operator has the highest precedence? Why do you think Java is designed that way?

Parenthesis have the highest precedence so that you can impose a specific order. For example, `2 * (3 + 4)` will perform addition before multiplication.

**25.** The `+` and `-` operators show up twice in the table of operator precedence. For the Java statement `x = 5 * -3;` explain how you know whether the `-` operator is being used as an ***unary*** or ***binary*** operator.

It matters what is to the left or right of the operator. In this example, the `-` is preceded by a `*`, so it must be unary.

**26**. Determine the order of operations in the statement: `x = 5 * -3;`

a) First operator to be evaluated: -

b) Second operator: *

c) Third operator: =

**27**. Determine the order of operations in the statement: `y = 9 / 2;`

a) First operator to be evaluated: /

b) Second operator: =

**28**. Based on your answer to the previous question, explain why the variable `y` would be assigned 4.0 (as opposed to 4 or 4.5).

Since `y` is a floating-point variable, the integer result 4 would be stored as 4.0 in memory.

**29**. Rewrite the assignment for `y` so that it would be set correctly to 4.5. (Hint: you'll need to recall what you learned about division in Model 2.)

```
y = 9.0 / 2.0;
```