Model 1 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:

1. 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.

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

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

3. 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.

4. 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.

5. 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

6. 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

7. 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

8. 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.)

9. 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.

10. 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.