Casting

Objectives

Explain how casting converts between different types and why you might need it.

Understand what an operator is and some examples

- You will learn what casting is, and how to use it to get the results you expect.
- To understand casting, let's first take a look at an example. Take the equation 3 / 2. When you do this in your head, the answer that you will come up with (should be) 1.5 (or 1 1/2). However, in Java, the answer will be 1. Why does this happen? To answer this, let's take a look at how Java handles types.

Types of Literals and Variables

- Java is a *strongly typed language*, which means that each variable has a type attached to it. The variable type tells the Java compiler how to interpret the variable, for example, whether a variable is a String, int, float, etc..
- For literals, e.g. 3.0, 2 or "hello world", which are values without a specified type, the Java compiler determines the type automatically. For example 3 is interpreted as an int, 1.2 is interpreted as a double, and "hello world" is interpreted as a String.

- Literal
  - (definition) A literal represents a fixed value, that can be taken "as-is" in code. Unlike variables, the values that literals take cannot be changed. For example, the number 0 will always have the value 0, and similarly a string like "hello" will always have the value "hello", and a boolean value of false will always have a value false.
- When you perform an operation (+, -, *, /, string operations, etc) on variables or literals, Java will set the type of the result, to be the same as the type of the operands. What this means is, given the operation 3 / 2, Java will determine that 3 and 2 are of type int, and thus the result will be an int. If you remember learning division in school and dealing with remainders, we can say that Java basically does the division and ignores the remainder, thus leaving us with an answer, 1.

Casting

- So, how would you go about getting an answer of 1.5 for the equation 3 / 2?
- Java has a few special rules for the arithmetic operators like +, * and / to decide what the type of the result is. If we multiply two numbers of different types, Java will give you the result in the most precise type. Since 2.0 is more precise than 5 (since it can include a fractional part), 5 / 2.0 will be 2.5, but 5 / 2 returns the int 2, since both are type int.
- Because of this, you can get 3 / 2 to return 1.5 by writing one of the operands using a decimal number, like this: 3.0 / 2 or 3 / 2.0. This would then cause Java to evaluate the operation using decimals.

- Another method, involves casting, which is when you force a variable to be of a certain type.
- For example, we can cast one of the operands to a primitive type, like float or double. Here are some examples:
  - (float) 3 / 2
  - 3 / (double) 2
- Casting
  - (definition) A casting is a way of telling Java that you want to convert a variable from one type to another. It is performed by placing the new type name in parentheses before the expression you want to convert to that type.

- For example, if we have
  int a = 1
  double b = 2.8,
- we would say
  a = (int)b
- to store the integer part of b into a.
- When we do this, a should end up holding 2
- (remember, Java doesn't round, it simply throws away the fractional part).

- The syntax for casting is as follows:
  (newType) expression or variable or literal
- cast an entire expression (such as (int) (3.0/2)) using *newType expression*
- cast a variable (such as (int) myNum using *newType variable*
- cast a literal (such as (int) 3) using *newType literal*

Learn by doing

- Let's use casting to do some temperature conversion. The equation for converting from Fahrenheit to Celsius is:
  - int fahrenheit = 27;
  - double celsius;
  - celsius = 5 / 9 * (fahrenheit – 32);
- how would you modify the code fragment above to give the correct result (with decimal places)?

Our answer

- T he literals are evaluated as integers. How would you change them to double or floating point variables?
- The answer involves casting the literals or variables to a double or floating point. The literals are evaluated as integers. There are several ways to do this, but one would be:

celsius = 5.0 / 9 * (double) (fahrenheit – 32);

Learn by doing

- Assume the following variables:
  double value;
  int colour = 3;
- Which of the following statements correctly assigns 2.25 to value?
  a. value = colour * 0.75
  b. value = colour * 3/4
  c. (double)value = colour * 0.75

Our answer

a. value = colour * 0.75
   - This will return 2.25. Remember that Java has special rules for arithmetic operators. It automatically will convert the value of color to a double because it is being multiplied by a double.

b. value = colour * ¾
   - This is incorrect. Because all of the variables involved are integers, the result will be an integer as well. The correct answer is value = color * 0.75.

c. (double)value = colour * 0.75
   - This is incorrect. Because all of the variables involved are integers, the result will be an integer as value that you are storing into a variable (the stuff on the right side of the equals sign), not the variable itself. The correct answer is value = color * 0.75.

Learn by doing

- Assume the following variables:
  double value;
  int a1 = 3;
  int a2 = 2;
- Which of the following statements correctly assigns a1 / a2 (i.e. 1.5) to value?
  a. (double) value = a1 / a2;
  b. value = (double) a1 / a2;
  c. value = a1 / a2;

Our answer

a. (double) value = a1 / a2;
   - You only need to cast one of the variables a1 and a2 to ensure that the division will result in a decimal (double).

b. value = (double) a1 / a2;
   - Correct.

c. value = a1 / a2;
   - You only need to cast one of the variables a1 and a2 to ensure that the division will result in a decimal (double).

Learn by doing

- Assume the following variables:
  int value;
  int a1 = 1;
  int a2 = 2;
- Which of the following statements correctly assigns a1 / a2 to value? Value should equal 0.5 + 0.5 = 1
  a. value = (int)(a1 / a2 + a1 / a2);
  b. value = (int)(a1 / a2 + (double)a1 / (double)a2);
  c. value = (int)((double)a1 / a2 + (double)a1 / a2);

Our answer

a. value = (int)(a1 / a2 + a1 / a2);
   - You only need to cast one of the variables a1 and a2 to ensure that the division will result in a decimal (double).

b. value = (int)(a1 / a2 + (double)a1 / (double)a2);
   - value = (double) a1 / a2 + (double) a1 / a2. You need to make sure that both the a1 / a2 values are converted to decimal so that they add to 1, which when casted to an int will lead to 1. Casting only the second half of the summation will yield 0 + 0.5 = 0.5, which when cast to an int yields 0. Casting none of the two a1/a2 expressiosns will lead to 0 + 0.

c. value = (int)((double)a1 / a2 + (double)a1 / a2);
   - Correct!

Summary

- In this module, we learned about how to cast variables and expressions to different variable types. The syntax for casting is:
- (*newType*) *expression or variable or literal*
- Casting happens automatically during some kinds of addition. Specifically, if you use a decimal and integer number as operands, the result will be decimal (double).