* A variable has three parts: A name, type, and value

Example: A variable to store the number of students

Name: ‘numStudents’

Type: Integer (int), to hold whole numbers

Value: 25

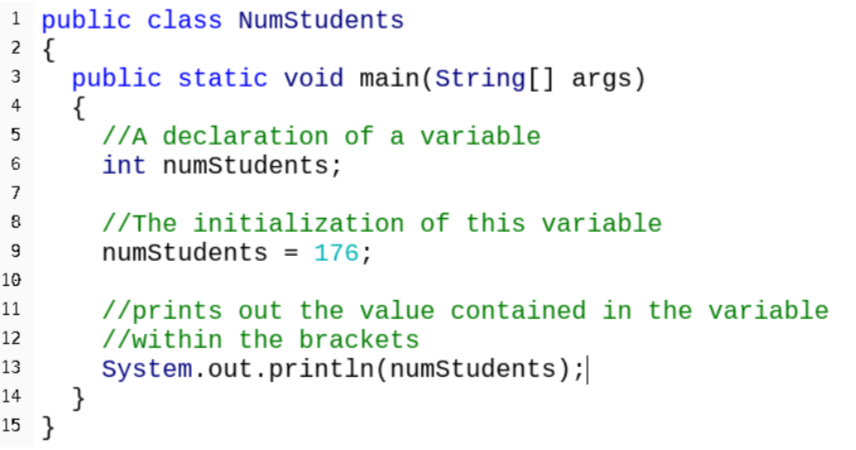
* declaration ：declaring to Java that we want a variable named ‘numStudents’

Write the statement *int numStudents;*

* initialization ：initializing the variable named ‘numStudents’ with the value 25

Write the next statement *numStudents = 25;*

* The statement *System.out.println(numStudents);* will print out the value of this variable.



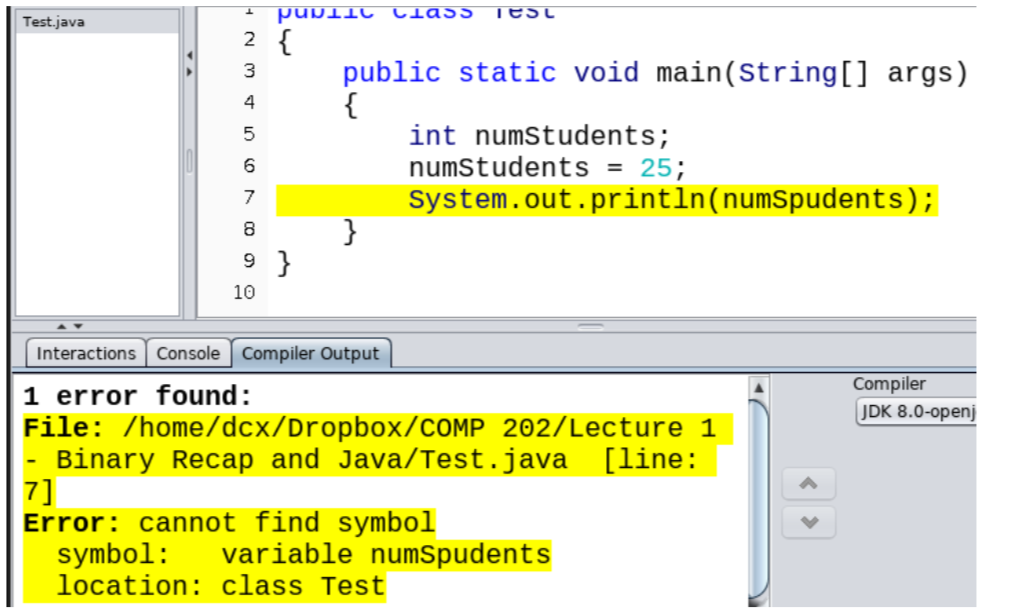
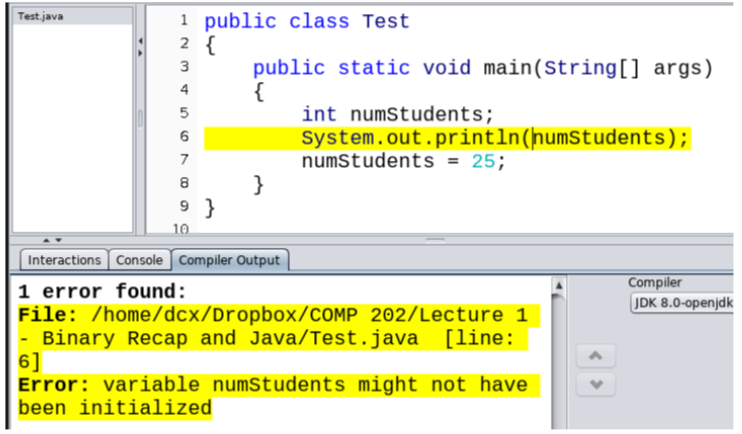
* Common errors:

1. without a value

Best way to solve this is to declare and initialize at same time

*int numStudents = 25;*

1. misspell the variable

****

* variable types

int - Integers (Whole numbers)

float (less precise) or double - Non-whole numbers (like 1.4)

String - Collection of characters (Double quotation marks around **String value/literal**)

boolean - True/False values

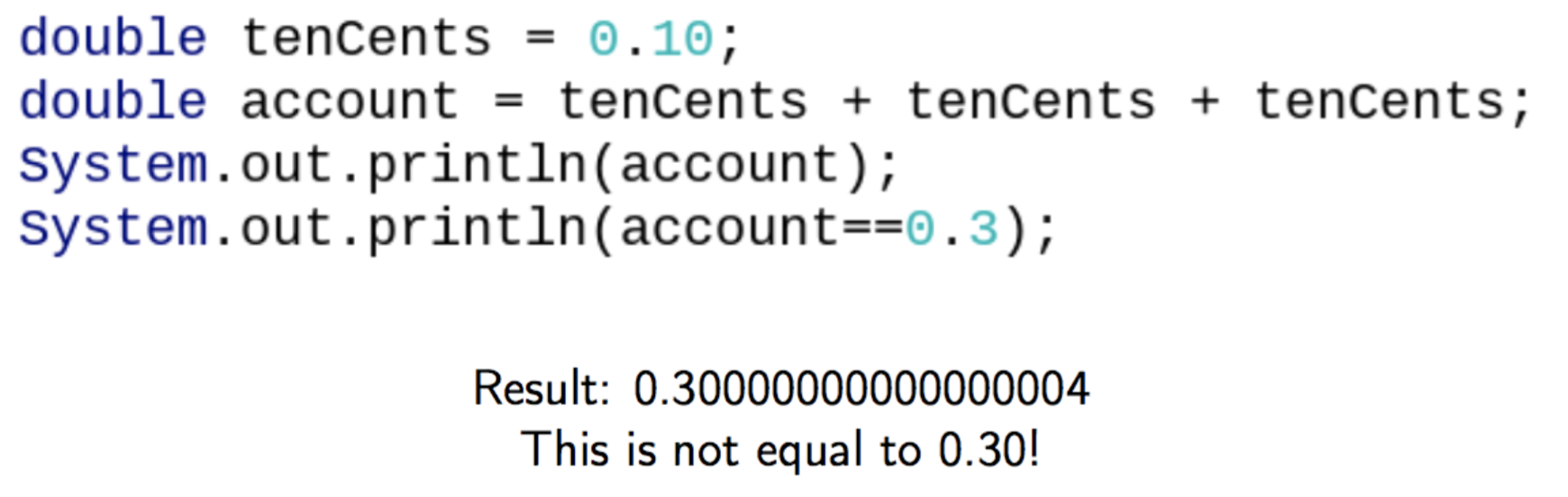
char – character (a symbol)

* integers vs doubles

About 40% slower to use doubles in math operations.

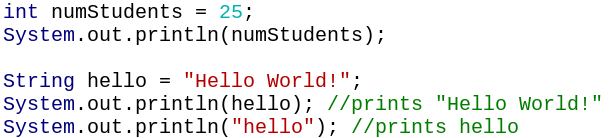
Doesn’t make sense to store age of a person in a decimal number

Store money in cents instead of 0.01 dollars. (not precise )



* system.out.println();

This is a **method**, which takes whatever is within the brackets and prints it.

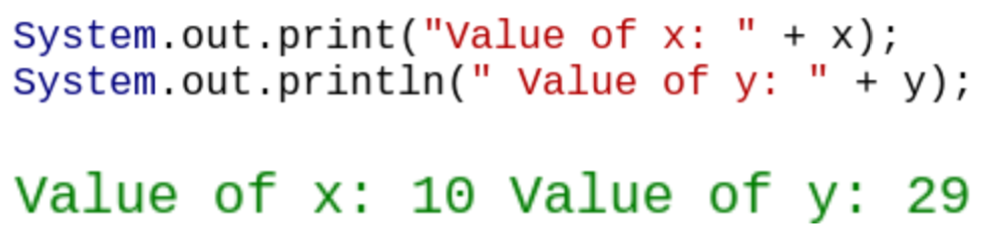


The System.out.println(); statement is evaluating what’s between the brackets first.

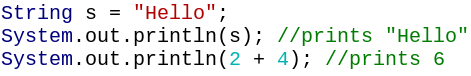
* System.out.println() prints out, then starts a new line

**System.out.print()** prints out, then doesn’t start a new line



* **Evaluation** means that System.out.println(); is determining the value of whatever is within its brackets.



* Concatenation

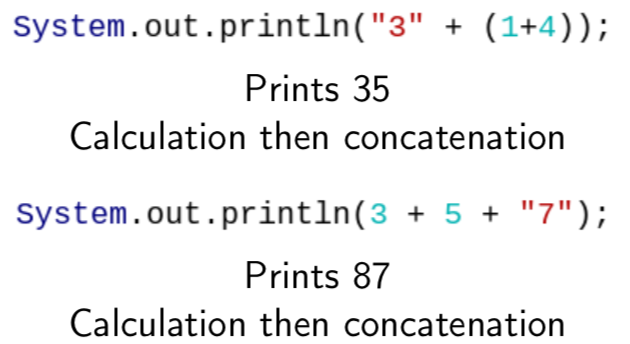
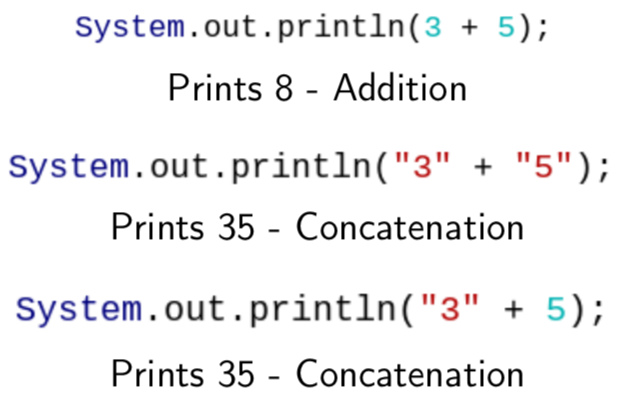
1. add together two Strings



1. print out some nice text with a variable’s value



Concatenation vs addition

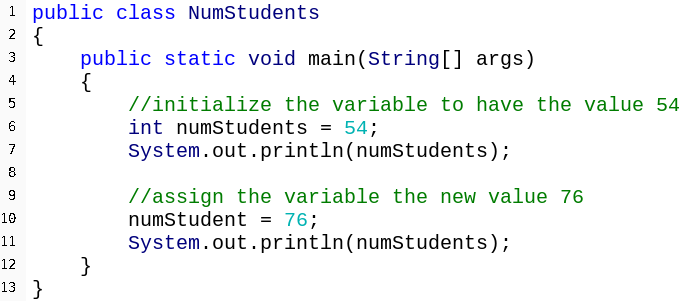


If there’s a String on either side of the + sign, then it’s concatenation.

The order of evaluation is left-to-right.

* Assignment

When performing calculations, we place the result in a variable. We call this **assigning** a new value to a variable.



The variable on the left is assigned the value on the right.

76 = numStudent does NOT work in programming.

* Types matching

Java is strongly typed. We need to assign correct types.

Typing int x; means that the variable can only store integer numbers.

int x = 3.5 --------Error: incompatible types.

int values can be stored in double variables

double a = 4

We can assign results of calculations to a variable.

int x= 4\*9 // x is set to 36

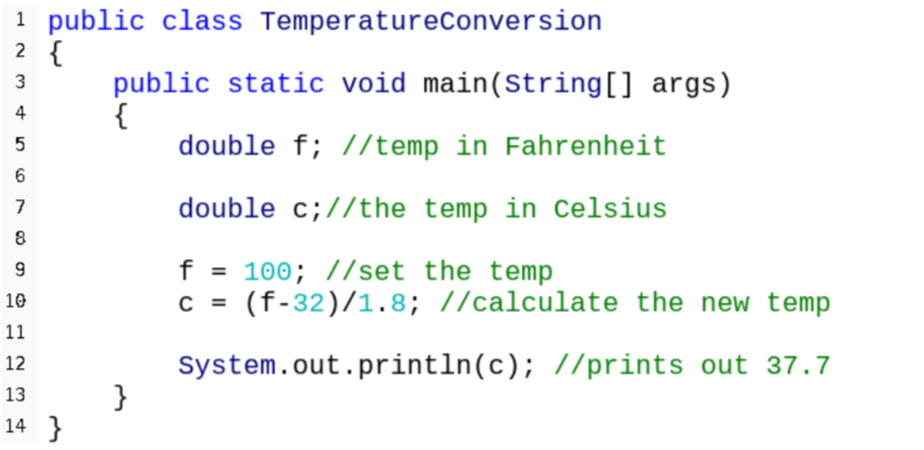
* Math operator

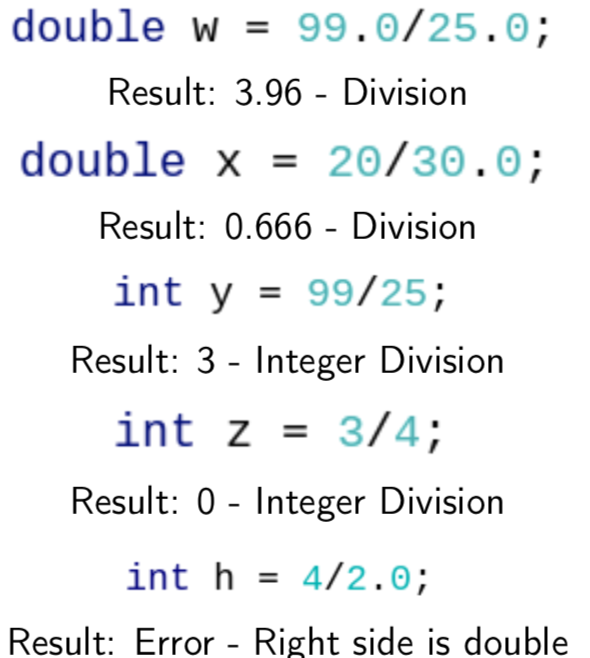
Between two int values, the result is an int.

If there’s a double value, then the result is a double.

BEDMAS: Brackets - Exponents - Division/Multiplication - Addition/Subtraction

* Temperature Conversion: (Fahrenheit------Celsius)





* Division vs integer division

When it is between two int values, the result is an int.

When there’s a double value, the result is a double.

*double w = 99/25;*

result: 3 – integer division

* Multiple variables

Java executes statements line-by-line.

The result depends on the variable’s values **at that time**.

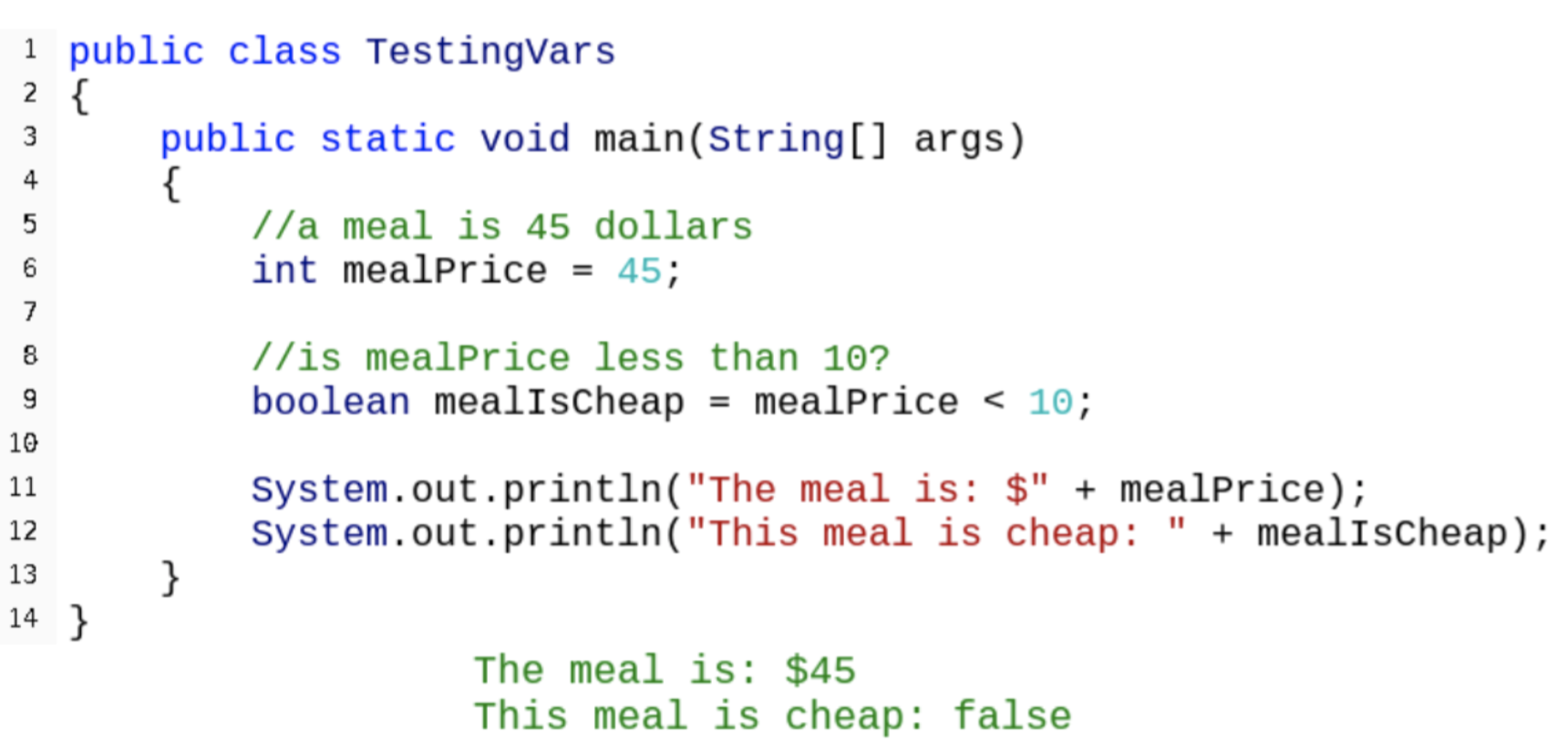


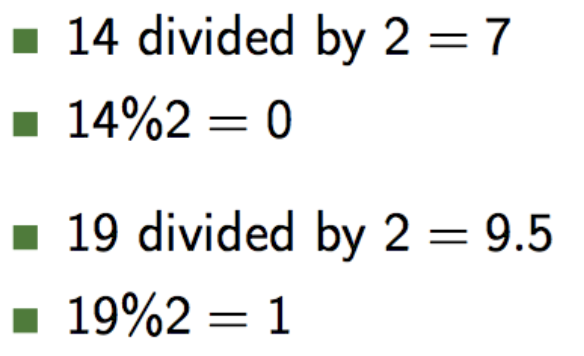


If we change x and make the same calculation for z as for y, z has different value than y.

* Booleans

We often need to make comparisons and tests that give true or false answer. In programming, these values are called **Boolean values** and the calculation is called a **Boolean expression.** Therefore, they are stored in **Boolean variables.** (the type of the variable is Boolean (lower-case b))

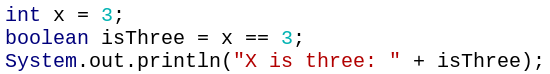


* Mod Operator

The remainder operator is called ‘mod’

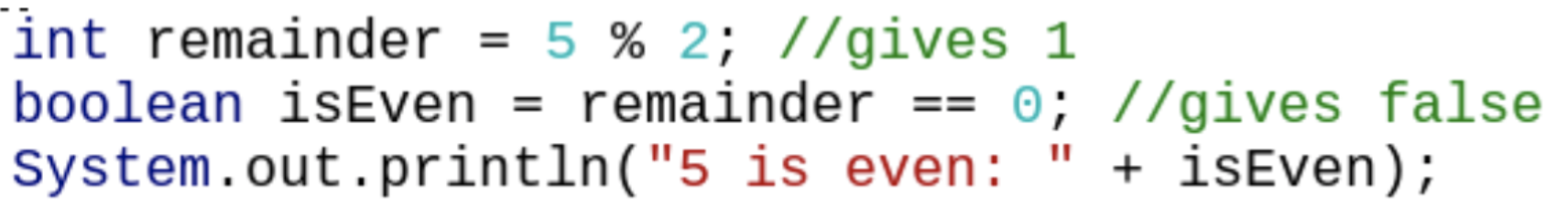
Its symbol is the number sign: %

* Equality

**== test if equal ! = test if unequal**

Note that = is assignment ; == is testing.

* Odd/even test



* Comparing different types of variables 