Unit 1 → Primitive Types

Basics

- Syntax for a class and main method
  - `String [] args` → read the text as a string

```
public class MyClass
{
      public static void main (String [] args)
      {
            //code
      }
}
```

- Printing in the console
  - `System.out.println()`
    - Prints input *then* moves to the following line
  - `System.out.print()`
    - Prints input and *stays* on the *same* line
- Creating comments
  - `/* code here */`
    - Comments out a everything with in it
  - `/** code */`
    - Creates a bullet comment
  - `//code`
    - Comments out a singular line

Variables

- Variables
  - We name them using camelCase
    - Never starts with a number
    - Cannot contain any special characters *unless* it's an underscore (_)
  - Name associated with memory location in the computer
  - When you create a variable, you are declaring it
  - Stored as binary digits → 0 or 1
  - Using `final` means an `int` or `double` ***cannot*** be changed
  - Initializing a variable
    - FIrst mention of a variable
      - `int x; → x=0;`
      - `int x = 0;`
- Primitive data types
  - `int`
    - Stores integer values → {0,1,2,3,4…}
    - 32 bits → 2^{31}
  - `double`
    - Stores floating point numbers → {0, 1.1, 2.3, 3.14…}
    - Also known as a float
    - 64 bits
  - `boolean`
    - Stores true/false arguments → {true, false}
    - 1 bit
  - ***STRINGS ARE <u>NOT</u> PRIMITIVES***
- Strings
  - String literal
    - A string of text written with double quotes → " "
  - String concatenation
    - Use "+" to connect two or more strings

Variable declaration

- Assigning values
  - Variable being assigned a value *always* goes to the left side of the expression
  - Operators
    - Plus (+) → Adds variables together
    - Subtract (-) → Subtracts variables
    - Multiplication (*) → Multiplies variables
    - Division (/) → Divides variables
    - Modulus (%) → Takes the remainder of variables
    - Equals (=) → Sets a variable equation to an expression
    - Double equals (==) → Returns a boolean value depending on the expression
    - Not equal (!=) → Returns a boolean value depending if the expression is not equal to another
- Compound assignment operators

`+`	`-`	`*`	`/`	`%`
`x=x+1`	`x=x-1`	`x=x*1`	`x=x/1`	`x=x%1`
`x+=1`	`x-=1`	`x*=1`	`x/=1`	`x%=1`
`x++`	`x--`			

- Dividing with ints and doubles
  - int / int → truncates and cuts off the decimals
  - double / int → double
  - int / double → double
  - ((double) int/int) → double
  - (double) (int/int) → double but truncates because it divides ints first