Note taking: C# Tutorial for Beginners: Learn C# from scratch with Mosh

(<https://www.youtube.com/watch?v=gfkTfcpWqAY>)

|  |  |
| --- | --- |
| **Function** | **단축키** |
| Run | [ctrl] + [F5] |
| Build | [ctrl] + [shift] + B |
| Console.WriteLine(); | cw + [Tab] |
| Delete a line | [ctrl] + x |

**C# vs .NET**

* **C#** is a programming language
* **.NET** is a framework for building applications on Windows
* **CLR (Common Language Run-time)**

C# code

**CLR** translates IL into Native code

This process is called ‘**Just-in-time compilation**’ (**JIT**)

-> IL (intermediate Language, Byte code) Code

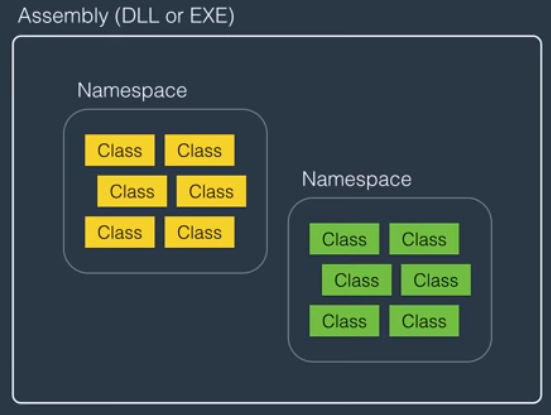
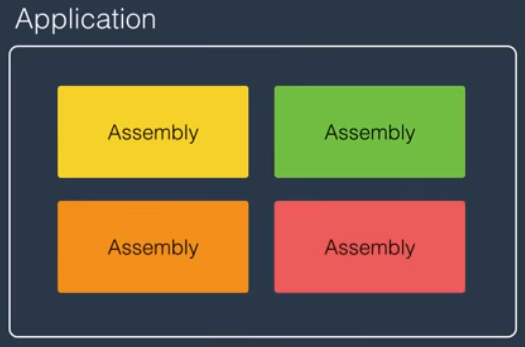
-> Native Code (machine code)

* Class Library

|  |
| --- |
| Class |
| Data (attributes, state of application) |
| Methods (behavior) |

|  |
| --- |
| Car |
| Make, Model, Color |
| Start(), Move() |

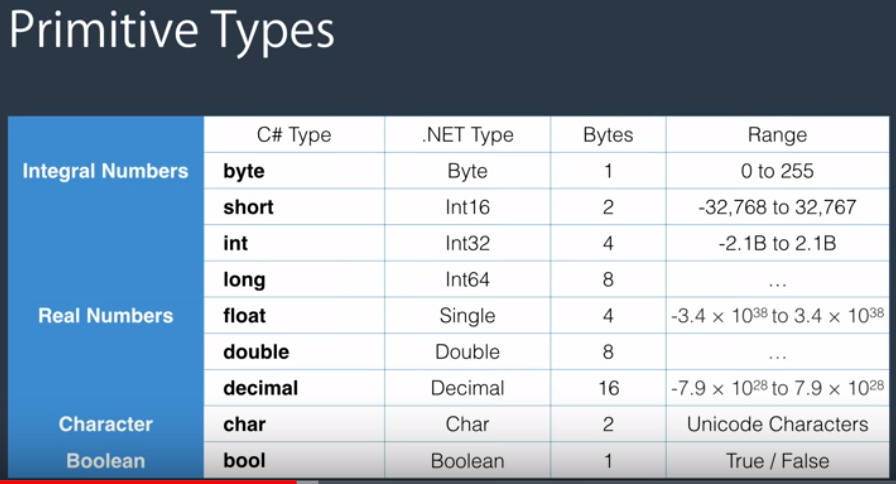
**Architecture of NET Applications**

DLL(Dynamically Linked Library)

**Variables & Constants**

* Variable: a name given to a storage location in memory
* Constant: an immutable value (used for safety)
* case-sensitive
* cannot start with a number
* cannot include a whitespace
* cannot be a reserved keyword (e.g) int
* Naming Conventions:
* Camel Case: firstName
* Pascal Case: FirstName
* ~~Hungarian Notation: strFirstName (not used in C#)~~



C# keyword are always lower case.

**Non-Primitive Types**

: String, Array, Enum, Class

**Overflowing**

checked // if we use **checked,** the program won’t overflow. It will throw an exception instead.

{

byte number = 255; //255 is the largest number byte can store

number = number + 1; // 0

}

**Scope**

Where a variable / constant has meaning and is accessible

{

byte a = 1;

{

byte b = 2;

{

byte c = 3;

}

}

}

**Implicit type conversion**

byte: 1 byte, integer: 4 bytes

byte b = 1; // 00000001

int i = b; // 00000000 00000000 00000000 00000001

* There is no data loss.

**Explicit type conversion (casting)**

int i = 300;

byte b = i; // won’t compile => There is data loss.

We are aware of data loss, but still want to data conversion!

int i = 300;

byte b = (casting)i;

float f = 1.0f;

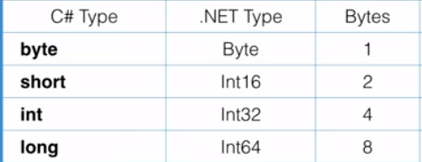
int i = (int)f;

**Conversion between non-compatible types**

string s = “1”;

int i = **Convert**.ToInt32(s); // we can we Convert class and ToInt32 which is .NET framework type

int j = int.**Parse**(s); // Or we can use parse() method



Convert

* ToByte(), ToInt16(), ToInt32(), ToInt64()

**Operators**

* Arithmetic: + - \* / % ++ -- (a++: postfix / ++a: prefix)
* Comparison: == != > >= < <=
* Assignment: = += -= \*= /=
* Logical: && || !
* Bitwise: & |

**Comments**

When to use: To explain whys, hows, constrains, etc. NOT whats.

* // single-line comment
* /\* \*/ multi-line comment