Code Academy
أكاديميةالبرمجة

Cracking The Basics

Eng : yousif mohamed

Table of Content

Consoling our DATA

As we have said in the previous Part That that console is a built in code that .net Programmers Provided for us . so Console is A pre-Built function that allow us to use the Cli (black window) to interact with our code . like displaying message .Let’s display our data

```
0 references
public static void Main()
{
    Console.WriteLine("hello my name is joe  , " +
        " my age is 26 ,job is software engineer  " +
        ",i am happy to participate in building 2040 dream for sultant oman ");

}
```

Console.Writeline Print a text message for us on command line interface that's why we call it Base Class Library .

—----------

Put into Consideration that the Text message is always Displayed between “ ” (double quote) also c# compiler know that you finished your operation when he detect (;)

What is Comments :

Compiler consider anything written in the code as something he should translate to assembly code so sometimes we need to leave for ourself or for our teammates a message to illustrate our work or the algorithm that we have used . to maintain and debug our code easily .

So for that reason Comments is Found

We have to types of comments

Single line comments

Multi line comments

```
// this is a single line Comment that will comment anything in this line
/*
    this
    is
    multi-line
    comment that will comment anything between the comment Block

 */
```

Variables

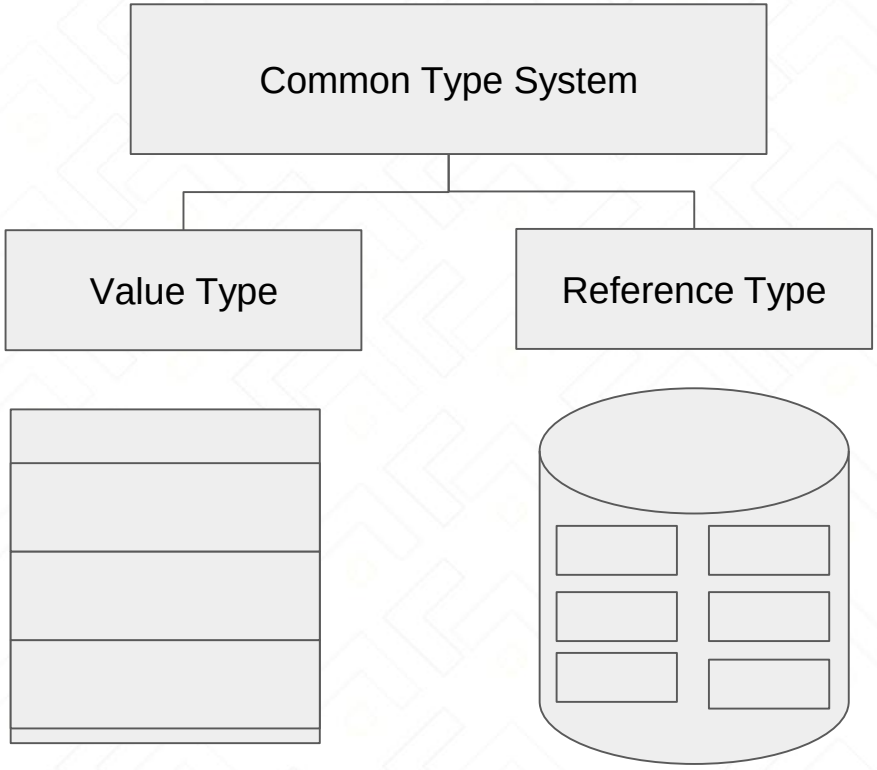
As we saw above we print a message by writing it between our “ ” but what if i want to store the Value of the message instead of writing it over and over so that’s the Variable Definition its a box that save our values in the memory to access it easily with typing it over and over . so how we can declare a variable i will tell you general Syntax then we will dive into Data Types that i can hold inside my Variable

```
// Variable Declaration : [ <Datatypes> <Identifier(varname)>  ; ]
```

Like us as humans we have text , numbers , yes or no and several Types of Data also we have tought the machine alot of Data-types in C# we deal with **Ram** to save our variables and that would be happened in a very specific way so we take into our Consideration space that the Variable Would Take so we handled this in C# with something Called Common Type System

Common Type System

The CTS defines a set of rules and guidelines for how types are defined, used, and represented in the .NET runtime environment. It provides a standard way to represent and manipulate data types, ensuring that code written in different programming languages can interoperate seamlessly within the .NET framework.



www.ca-oman.com

Common Type System

Value Type

- bool: represents a Boolean value (true or false)
- byte: represents an unsigned 8-bit integer
- char: represents a Unicode 16 character 2 byte
- decimal: represents a decimal number
- double: represents a double-precision floating-point number
- float: represents a single-precision floating-point number
- int: represents a signed 32-bit integer
- long: represents a signed 64-bit integer
- short: represents a signed 16-bit integer
- struct: a custom value type defined by the programmer

Reference Type

- string: represents a sequence of Unicode characters
- object: the base class for all types in .NET
- class: a custom reference type defined by the programmer
- interface: a reference type that defines a contract for a class to implement

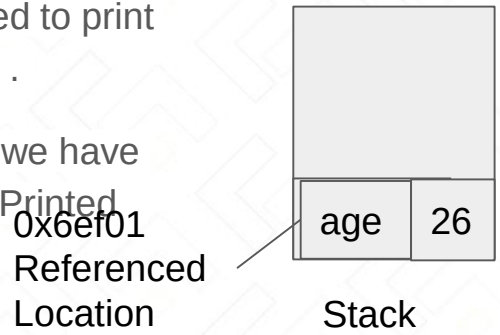
Declaration of Value Type Under the Hood :

As we Talked about Declaration of a Variable its through declaring what is the Datatype and here we Are talking about about Value Type that is stored in **Stack** and we will touch the Main Difference Between them in the Few Coming Slides . but what happened with the memory by Typing → int age ; The memory reserve 4 bytes in memory with name age but we didn't assign any Value so it will raise error if you tried to print the Value so how we can assign Value .

age = 26 ; now we have assigned the value and its ready to be Printed

```
// <Datatypes> <Identifier(varname)>  ;
    int              age      ;

// Variable Declaration :
// <Datatypes> <Identifier(varname)>  ;
int age;
//assignment a value to a variable
age = 25;
Console.WriteLine(age);
```



Declaration of Reference Type under the Hood

As we have discussed in the cts differences we knew that String is the group of Character and its a reference data type that is stored in heap .

Let’s see the Declaration and Assignment in another Form —-->

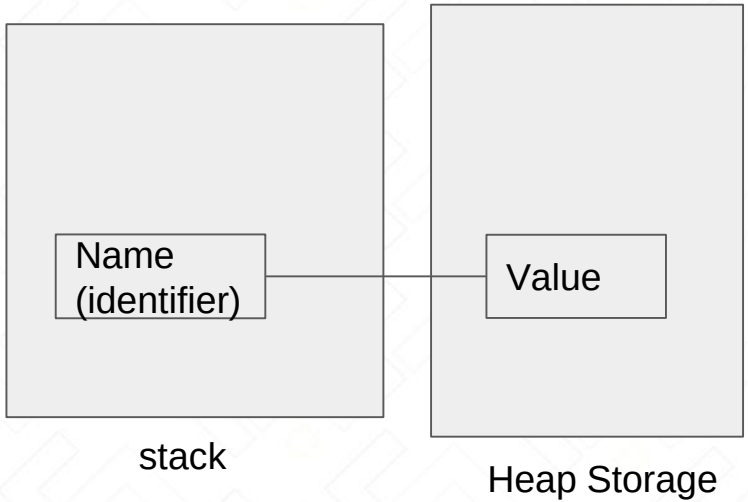
string name = “my name is joonguini ”;

Why String is Different and it is Special in any Programming Language??

When you say int x you will reserve in the memory only 4 bytes so here we know the maximum size length of Values that this Location Can Held . But when we say String str ;

We Don’t Know how Much we will reserve in the Memory so it is like Dynamic allocation (Stretch)

```
//String Declaration and assignment
String name = "my name is joonguini ";
Console.WriteLine(name);
```



Why String is Different ?

Why String is Different and it is Special in any Programming Language??

When you say int x you will reserve in the memory only 4 bytes so here we know the maximum size length of Values that this Location Can Held . But when we say String str ;

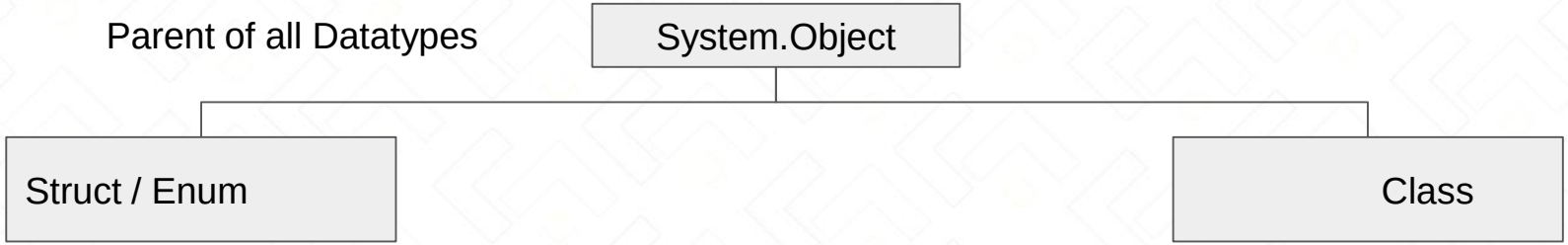
We Don’t Know how Much we will reserve in the Memory so it is like Dynamic allocation (Stretch)

Actually String is a Class Datatype so if you want to declare any copy (object) From Class we can Type :
string str = new String()

String class has a lot of prepared function that Can help us a lot and save Time for us

```
string str = new String("hello my name is joe  , " +
    " my age is 26 ,job is software engineer  ," +
    "i am happy to participate in building 2040 dream for sultant oman ");
Console.WriteLine(str);
```

Cts Under the Hood



The Shape above Describe all the Datatypes how the C# Prammers Team created Our Variables with i just illustrated it to discuss that there is set of behaviours in all Datatypes so the collect all that Behaviours and put them in System.Object :

- ToString() : predicted to return string representation to any object State
- GetHashCode() : return Object Identity its like a UNIQUE identifier for any object Created in C# if 2 objects returned same hashcode so they are same Object (References in memory to same Location but with 2 name)but with to different name
- Equals(obejct2) : For equality if i have 2 objects and i want to check if they equal each others or not
- GetType() : return object Type

Let’s dive into Reference Type under the hood (Object):

We will Declare 2 objects OBJECT 1 and Object 2 : let’s Think Together how that will be happened :

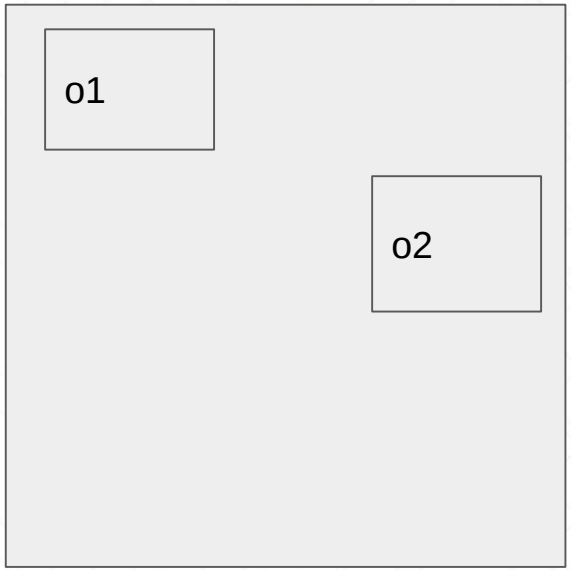
Object o1 ; → 1

o1 = new Object() → 2

Object o2 = new Object() →3

1- declaration but Nothing happened in memory just a name
2-using new Keyword will allocate space in heap for o1 and we will discuss what new do in next slide

3- declaration and allocation in memory



Heap

new Keyword :

When you type new then data-type keyword (object)

Its like you are Creating a child from the Parent a new child with different name but has your Characteristics and he can manage it in his own way .

New under hood

1.Allocate Require # of Bytes in heap

(object size + overhead Variables):

2-initialize allocated bytes

3-assgn reference to newly allocated object

Call user defined constructor if exists (OOP topic)

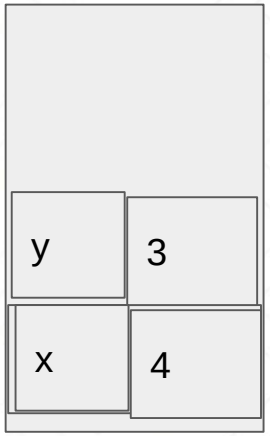
Code Academy
أكاديميةالبرمجة

Let’s CodeinPractice

Let’s Discuss the Difference that we Saw in CodeinPractice

Value Type

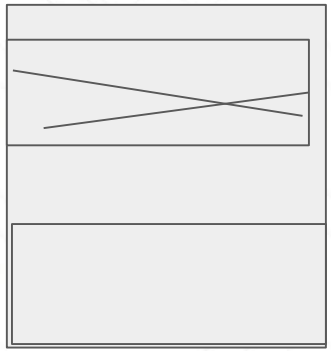
When we assign x = y in the stack we take a copy of the y value to to replace x value with it and f we changed one of them its changed each one as separate value



Stack

Reference Type

When we assign obj2 = obj1 in the heap we take a reference of obj2 address and replace it with obj1 and now its unreachable location (Garbage collector job)



Obj2
Unreached location

obj1

obj2

Built-in Datatypes in depth :

C# Type	.NET Framework type
bool	System.Boolean
byte	System.Byte
sbyte	System.SByte
char	System.Char
decimal	System.Decimal
double	System.Double
float	System.Single
int	System.Int32
uint	System.UInt32
long	System.Int64
ulong	System.UInt64
object	System.Object
short	System.Int16
ushort	System.UInt16
string	System.String

Data-types Limit :

Type	Range	Size
sbyte	-128 to 127	Signed 8-bit integer
byte	0 to 255	Unsigned 8-bit integer
char	U+0000 to U+ffff	Unicode 16-bit character
short	-32,768 to 32,767	Signed 16-bit integer
ushort	0 to 65,535	Unsigned 16-bit integer
int	-2,147,483,648 to 2,147,483,647	Signed 32-bit integer
uint	0 to 4,294,967,295	Unsigned 32-bit integer
long	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Signed 64-bit integer
ulong	0 to 18,446,744,073,709,551,615	Unsigned 64-bit integer

Float Difference :

Type	Approximate range	Precision
float	$\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$	7 digits
double	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$	15-16 digits
decimal	$\pm 1.0 \times 10^{-28}$ to $\pm 7.9 \times 1028$	28-29 significant digits

Range illustrated in code

```
Console.WriteLine($"sbyte: [{sbyte.MinValue} → {sbyte.MaxValue}]");  // [-128  → 127]
Console.WriteLine($"int: [{byte.MinValue} → {byte.MaxValue}]");  // [0 → 255]
Console.WriteLine($"int: [{short.MinValue} → {short.MaxValue}]");  // [-32,768 → 32,767]
Console.WriteLine($"ushort: [{ushort.MinValue} → {ushort.MaxValue}]");  // [0 → 65,535]
Console.WriteLine($"int: [{int.MinValue} → {int.MaxValue}]");  // [-2,147,483,648 → 2,147,483,647]
Console.WriteLine($"uint: [{uint.MinValue} → {uint.MaxValue}]");  // [0 → 4,294,967,295]
Console.WriteLine($"long: [{long.MinValue} → {long.MaxValue}]");  // [-9,223,372,036,854,775,808 → 9,223,372,036,854,775,807]
Console.WriteLine($"ulong: [{ulong.MinValue} → {ulong.MaxValue}]");  // [0 → 18,446,744,073,709,551,615]

Console.WriteLine($"float: [{float.MinValue} → {float.MaxValue}]");       // [±1.5 x 10-45 → ±3.4 x 1038]
Console.WriteLine($"double: [{double.MinValue} → {double.MaxValue}]");    // [±5.0 x 10-324 → ±1.7 x 10308]
Console.WriteLine($"decimal: [{decimal.MinValue} → {decimal.MaxValue}]"); // [±1.0 x 10-28 → ±7.9228 x 1028]
```

String Concatenation

String Concatenation used to concat variables together to get one Value .

Regular Concatenation(plus sign) :

String firstName = “youssif” ;

String lastName = “mohamed” ;

String fullname = firstname + “ ” + lastname ;

String interpolation :

String fullname = $”{firstname} {lastname} ;”

```
//String Concatination

string firstName = "yousif";
string lastName = "ebrahim";
//regular concatination
Console.WriteLine("your full name is " + firstName + " " + lastName);

//string interpolation
Console.WriteLine($"your full name is {firstName} {lastName}");
```

Var vs Dynamic Keyword :

Var

Var is used for automatic detection for the Variable Data Type but we will used Specially with LINQ (language integrated query) in Advanced Topic .

The type of the variable is determined at compile time, and cannot be changed later

Dynamic

Dynamic also is used for automatic detection for the Variable Data Type but we will used Specially with LINQ (language integrated query) in Advanced Topic .

The type of the variable is determined at runtime

Var Keyword Example :

```
//var keyword    // suffix literal
//-------------------------------------
var s = "joe";  // compilier detect that its a string

var f = 0f;     //compiler detect that it is a float

var d = 0d;     // compiler detect that it is a Double

var m = 0m;     // compiler detect that it is a decimal

var u = 0u;     // compiler detect that it is unsigned interger

var l = 0l;     //compiler detect that it is long integer

var ul = 0ul;   // compiler detect that it is unsigend long integer
```

Dynamic Keyword Example :

That's the main difference that we can change the data-type of the Variable after deceleration with dynamic , but with var if you declared you can't change its datatype

```
//Dynamic Keyword
dynamic test = "hello there"; //declared as String
test = 3;
test = 0f;  //compiler detect that it is a float
test = 0d;  // compiler detect that it is a Double
test = 0m;  // compiler detect that it is a decimal
test = 0u;  // compiler detect that it is unsigned interger
test = 0l;  //compiler detect that it is long integer
test = 0ul;  // compiler detect that it is unsigend long integer
```

Session Recap

Any Questions ?

Thank You

Email : youssef.mohamed@amit-learning.com

Linked-in : https://www.linkedin.com/in/youssif-mohamed-450795157/