C#; cours 3

Value Types:

* Enum
* Struct
  + System types : Int, float

Passed by value (copy) in parameters

“Ref” modifier : can be modified, must be pre-initialized

“Out” modifier: can be modified, may be uninitialized

“Const” modifier: cannot be modified, non allocated on the Stack or Heap

Cannot be null

“?” operator: int? A = null; put the int to null, it is “nullable”

**a ?? b**

a is non null → returns a

a is null → returns b

Reference Types (pointer : reference to the value):

* Pointer
* Interface
* Classes
  + Arrays
  + Boxed values
  + Delegate
  + String

“Readonly”: cannot be modified, may be initialized in the declaration or in the constructors

Cast:

**(MyType) obj**

**obj as MyType**

Cannot be applied on value types

String is ummutable. If it is modified then it creates a new reference

Boxing:

**int x;**

**object o = x;**

Unboxing:

**int y = (int) o; int y = o as int;**

Arrays:

[] near the type declaration : int[] blabla

Single Dimension array

Multiple Dimension array

Array of array

Interface:

Contract : define a generic method with in parameter a type object

Implementer : implement it in a class

Can be used inside a class

Collection:

Use “foreach”

Use generics

Name space for all types: definitions, default implementation, abstract classes

Implementation:

* Count
* Access
* Add
* Remove
* Browse

Collection.generic:

Allow strongly type construct: avoid boxing, better performances, error protection

* Declaration
* Add, Remove, Sort, Value
* Browse