Lesson 1 Review

Object is base type of all types.

All types derive from the base type.

Functions from Base Type:

ToString() 🡪 converts to a string

Equals() 🡪 brings back a bool

In c#, what identifies a reference type? A Class

A value type? Primitive Types and struct. All primitives derive from struct.

For Equality:

Value Types are compared by actual value types.

Reference Types are compared by reference (memory address). Must reference the same value (address) to be equal.

For Assignment:

Class = New keyword new T()

Value Types = value

Value types:

X = y: with assignment, the value of y will be passed over to the value of x. You cannot modify the value of a value type. Assignment always copies value types.

Reference types are mutable. They are designed to be changed.

X = y

Can reference types be null? Yes

Can value types be null? No, because a value type is stored directly on the call stack, and must have value.

**NULL Coalesce**

Exp(1) ?? Exp(2)

This picks the first non-null expression.

Same as

If (Exp(1) != null){

Exp(1)

}

Else

{

Exp(2)

}

If first expression is null, it will return the second expression, even if it is null. You can chain them.Chain: Exp(1) ?? Exp(2) ?? Exp(3) ?? Exp(4)

**NULL Conditional**

?.

Normally, we would check for null, then do something.

Like

If (Exp != null)

Do something

With ?., it evaluates the right side.

Exp ?. M

M is a function, not an expression

Designed to conditionally execute the right side if the left side is not null. It protects us from referencing null without having to write the != null check.

Int x = 10;

Int y = 3;

Var result = x/y;

Result will be 3. An int divided by an int yields an int.

Type casting will change the value type:

Var result = x/(double)y;

It doesn’t change the value of y. It only change how the compiler sees the variable.