2.1.5 Type constraints (Ràng buộc kiểu dữ liệu)

VD: Method

~~static void PrintItems(List<IFormattable> items)~~

List<decimal> isn’t convertible to List<IFormattable>, or can’t call T.ToString() method

* Do like this:

static void PrintItems<T>(List<T> items) where T : IFormattable

{

CultureInfo culture = CultureInfo.InvariantCulture;

foreach (T item in items)

{

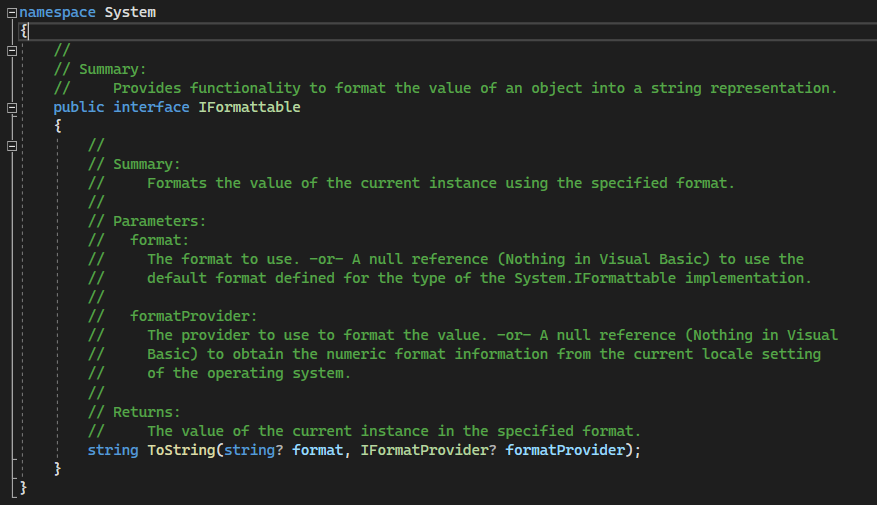
Console.WriteLine(**item.ToString(null, culture)**);

}

}

Constraint T help compiler knows that T implements IFormatable, so it allows the

IFormattable.ToString(string, IFormatProvider) method to be called on any T value.

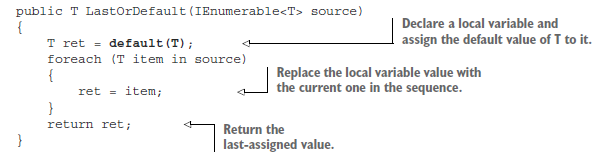


Type constraints k gioi han rieng cho interface, ma available with:

* Reference type constraint: where T : class/delegates/interface… T la kieu tham chieu
* Value type constraint: where T : struct T la 1 kieu du lieu non-nullable
* Constructor constraint: where T : new() Kieu du lieu T co mot pulic constructor
* Conversion constraint: where T : SomeType T co the la class, Control, IFormatable, T2 type…

2.1.6 The default and typeof operators

- Defaultlt: return result is default value for operand

VD: Thuong su dung voi operand (toan hang) vs generic type parameters, do voi cac type parameter khac thi co the de dang chi dinh gia tri mac dinh

- Typeof: return a Type value

2.1.7 Generic type initialization and state

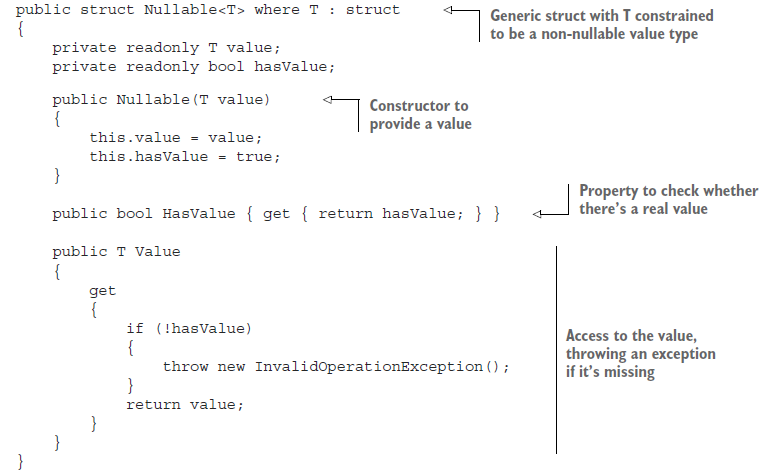
2.2 Nullable value types

2.2.1 Aim: Expressing an absence of information

->null value and nullable type

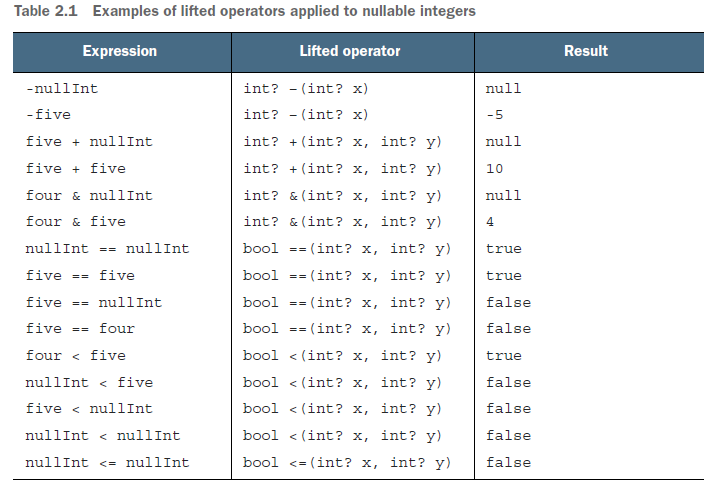
2.2.2 CLR and framework support: The Nullable<T> struct

The core of nullable value type support is the Nullable<T> struct



2.2.3 Language support

The ? type suffix



2.3 Simplifed delegate creation