Chapter 1: C#2

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)

{

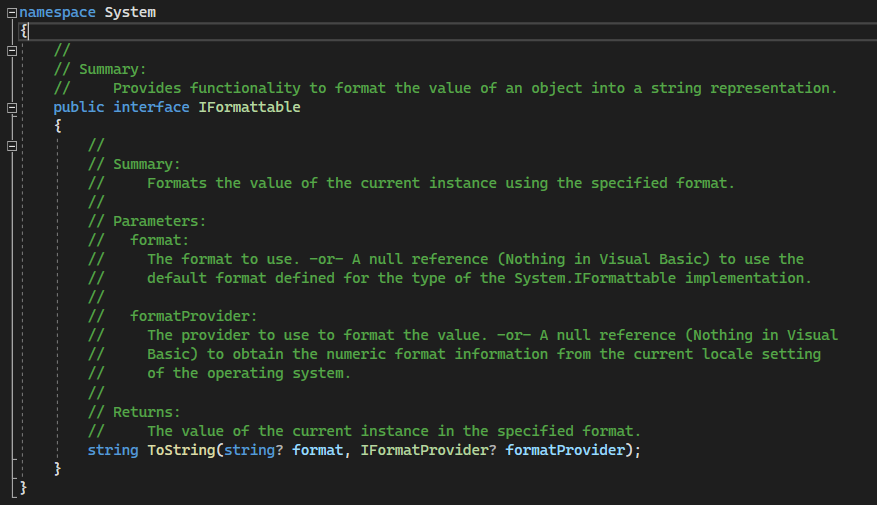
Console.WriteLine(**item.ToString(null, culture)**);

}

}

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.

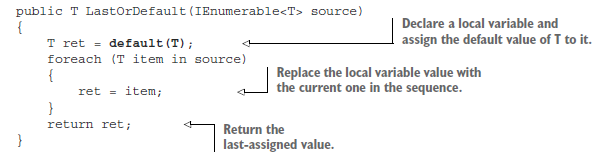


Type constraints k gioi han rieng cho interface, ma available with:

* 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

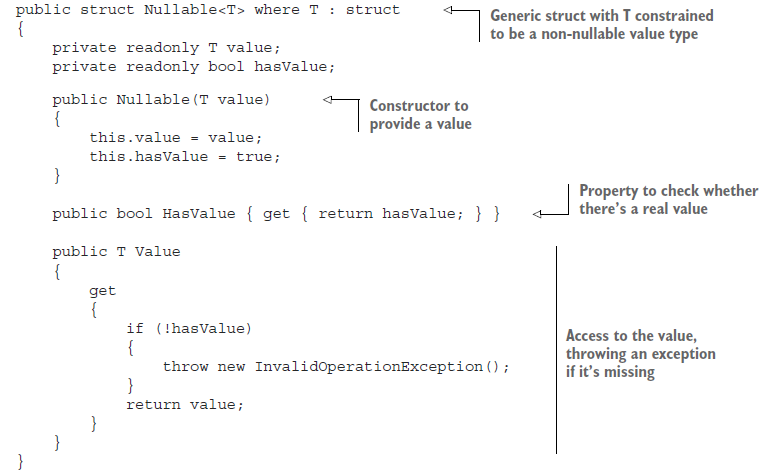
2.2 Nullable value types

2.2.1 Aim: Expressing an absence of information

->null value and nullable type

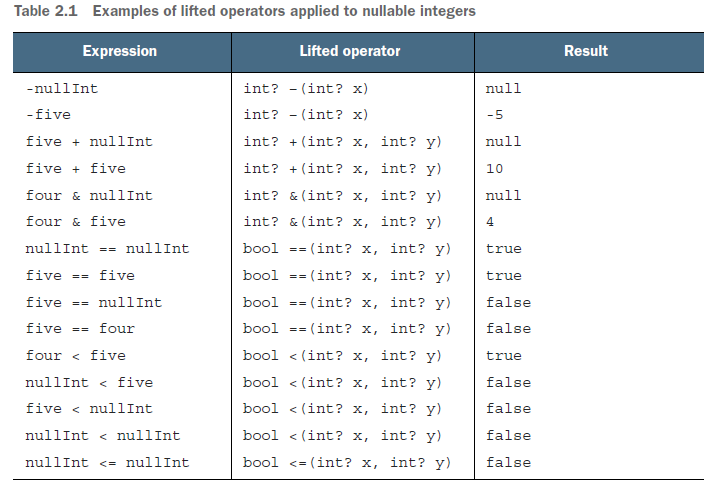
2.2.2 CLR and framework support: The Nullable<T> struct

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



2.2.3 Language support

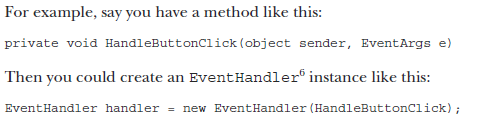
The ? type suffix



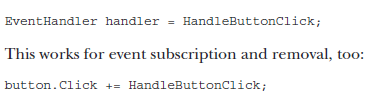
2.3 Simplifed delegate creation

2.3.1 Methods group conversions

- Method group gồm hoặc nhiều methods cùng tên. C#1: compler dựa vào số argument để phân biệt các methods và dùng delegate creation expression để tạo 1 delegate instance

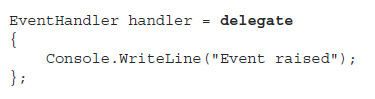


Với C#2, method group conversion giúp ngầm chuyển đổi các method trong method group thành đúng delegate mình mong muốn, theo số đối số đc truyền vào



2.3.2 Anonymous methods

Cho phép ta k cần tạo 1 method, chỉ cần viết code inline mô tả cách thức và vị trí tạo instance và sử dụng từ khóa delegate



(Tham khảo lamda expression ở C#3)

2.3.3 Delegate compatibility

2.4 Iterators (Vòng lặp)

Yield return

Yield break

2.5 Minor features

2.5.1 Partial types

Cho phép 1 class, interface, struct có thể được viết thành nhiều phần, chia ra nhiề source file khác nhau

C#3 partial methods

2.5.2 Static classes

Static class k thể khai báo các instance methods, properties, events hay constructors. Statice class chỉ bao gồm các static member ( static methods, const, )và có thể bao gồm các kiểu lồng nhau (nested types)

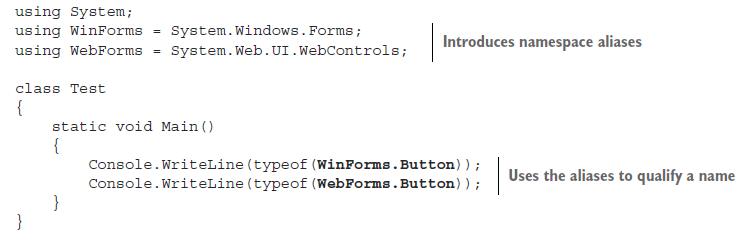
2.5.3 truy cập Get/set properties tách biệt

So với C#1 chỉ có 1 access modifier duy nhất dùng cho cả set và get giá trị cho thuộc tính

2.5.4 Namespace aliases

Gán biệt danh/tên gợi nhớ cho các namespace

Với C#1



Từ C#2 mở rộng hỗ trợ gán tên cho namespace ......

2.5.5 Pragma directive

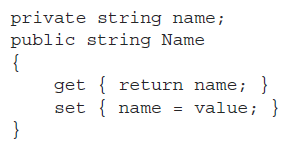
2.5.6 Fixed-size buffers

2.5.7 InternalsVisibleTo

Chapter 3: C#3

Khác với C#2 nơi các thay đổi và cập nhật gồm nhiều phần độc lập, C#3 xoay quanh LinQ

3.1Automatically implemented properties

Old: 

* public string Name { get; set; }

không cần tạo các filed (như name) để get/set properties nữa

3.2 Implicit typing

3.2.1 Typing terminology

Static/dynamic typing

Explicit/implicit typing

3.2.2 Implicitly typed local variables (var)

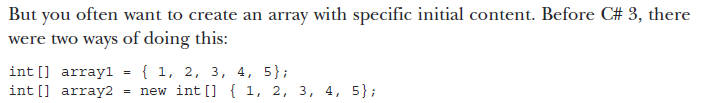


2 rules: biến phải được khởi tạo khi khai báo, và giá trị khởi tạo phải có kiểu xác định

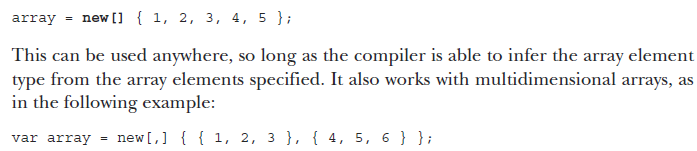
3.2.2 implicitly typed arrays

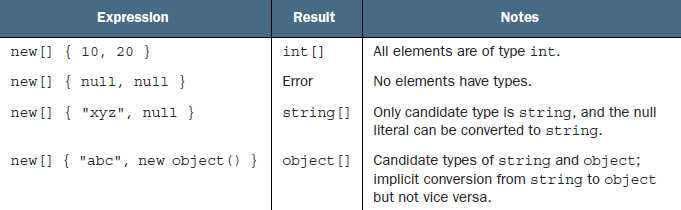
Với C#1

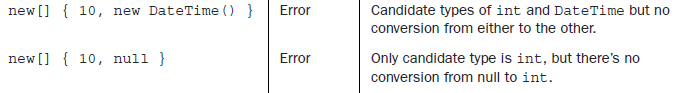




Với C#3:







3.3 Object and collection initializers

3.3.1 introduction to object and collection innitializers

3.3.2 Object Initializers

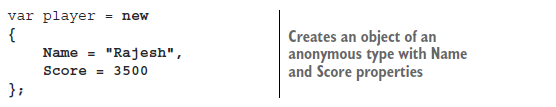
Object Initializers -> gồm 1 chuỗi các member initializer, mỗi member initializer cần mô tả khởi tạo tất cả properties của member đó

3.3.3 Collection Initializers

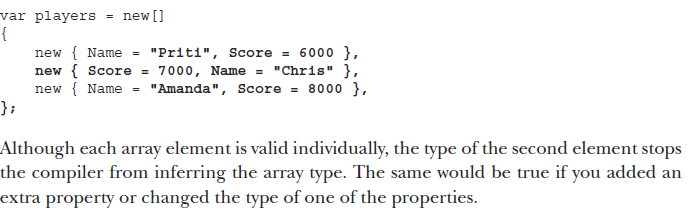
3.3.4 Benefits of single expression for initialization

3.4 Anonymous types

3.4.1 Syntax and basic behavior



3.4.2 The compiler-generated type



3.4.3 Limitations

3.5 Lamda Expressions

3.5.1 Lamda expression syntax