Beuscart Benjamin

Lupa Valentin

Malbranque Louis

Verpoort Alexia

***Introduction au C# :***

C# : Microsoft 2002

Partie de Frameworks.net. Combinaison des frameworks et du langage permet la cr éation d’une application.

Par rapport au C : permet la compatibilité car transformation en langage CIL ou MSIL plutôt qu’en binaire.

Par rapport au C++ : plus simple

Par rapport au java : plus de features.

C# est un langage orienté objet et orienté composant. -> EXPLICATION

Type-safe, gestion des exceptions, garbage collection

Compilation C# : empaquement dans assemblys qui implémentent des bibliothèques (.dll) ou des applications (.exe)

***Les bases du C# :***

Types :

La déclaration d’une variable s’effectue en déclarant le type de la variable directement (float, int…) ou en utilisant var qui permet au compilateur de déduire. Une variable déclarée ne peut pas être assigné à une variable d’un autre type sans utiliser le type-casting qui est une conversion explicite d’un type. Certains cas entrainent cependant une conversion implicite s’il n’y a pas de perte de donnée et cela est effectué par le compilateur.

Exemple de cast : int a=10 ;

double b ;

b=(double)a ;

Types unifiés donc partage d’opération commune. Type-safe permet de vérifier que les opérations sur les variables sont bien utilisées.

Prise en charge des types références et valeurs.

L’utilisateur peut créer ses propres types à l’aide des constructions struct, classe, interface, enum.

Type valeur : une variable de type valeur contient directement la donnée. Une opération sur une variable n’influence pas une autre variable de même type. C’est un type scellé, il ne prend donc pas en charge l’héritage, c’est-à-dire qu’aucune classe ne peut hériter d’un struct par exemple. Le type valeur regroupe les struct et les enum.

En bref un struct est un conteneur de variables liées. Une enum se compose de constantes.

Type référence : une variable de type référence contient la référence à leur donnée qui est appelé objet. Une opération sur un variable peut influencer une autre variable de même type. Les types références sont des classes, interfaces, délégué ou tableaux. Ce type est initialisé à null et nécessite la création grâce à l’objet new.

Classe et struct :

Structure de données encapsulant d’autres données ayant un lien entre elles.

Les classes et structs sont constitués de membres : leur donnée et leur comportement.

A COMPLETER

Interface :

Ressemble à une classe de type abstraite. Elle ne peut être instanciée directement, c’est-à-dire qu’on ne peut créer un objet directement du type de l’interface. Les interfaces ne contiennent pas d’implémentation des méthodes.

Une classe eut implémenter plusieurs interfaces.

Tableaux :

Comme en C, indexés sur zéro, avec n éléments allant de 0 à n-1

Les tableaux sont de tout types y compris du type tableau

Les valeurs par défaut des éléments de tableau sont définies sur zéro et les éléments de référence sont définis sur Null.

Un tableau peut être unidimensionnel, multidimensionnel comme en C ou encore en escalier.

La particularité de l’escalier est que les éléments du tableau principale n’ont pas les mêmes dimensions entre eux :

Exemple de cast : jaggedArray[0] = new int[] { 1, 3, 5, 7, 9 };

jaggedArray[1] = new int[] { 0, 2, 4, 6 };

jaggedArray[2] = new int[] { 11, 22 };

Boucle FOREACH

***Concepts Clés du C# :***

Evenements :

Permet à un objet qui est appelé publieur d’avertir à d’autres classes, les abonnés lorsqu’un événement important se produit.

Le choix du déclenchement est effectué par le publieur, c’est-à-dire que le publieur choisit quelle action génère un événement et les abonnés choisissent que faire à la réception de l’événement.

Un évènement nécessite n abonné pour pourvoir être déclenché.

EXEMPLE PRATIQUE

Délégués :

Les délégués, déclarés avec delegate, permettent la mise en place du principe d’ouvert/fermé, autrement dit, ils permettent de passer des méthodes comme argument lors des appels. Ils sont en quelque sorte un moyen de modifier le comportement d’une application sans modifier son code source.

Leur effet est lié à celui de la catégorie de classes abstract

Les délégués peuvent être déclarés dans la partie namespace ou alors au sein même d’une classe.

EXEMPLE PRATIQUE :

namespace UsingDelegate

{

delegate string ActionOnString(string text);

class Program

{

//Regular method that matches signature

static string normalPrint(string text)

{

return text;

}

static void Main(string[] args)

{

//Instantiate delegate with named method

ActionOnString test1 = normalPrint;

//Invoke delegate test1:

//Normal print

string result1 = test1("Première phrase de test.");

Console.WriteLine(result1 + "\n");

//Instantiate delegate with anonymous method

//All in uppercase

ActionOnString test2 = delegate (string text)

{

string tmp = text.ToUpper();

return tmp;

};

string result2 = test2("Deuxième phrase de test.");

Console.WriteLine(result2 + "\n");

//Instantiate delegate with lambda expression

//concatenate 2 times the same string

ActionOnString test3 = s => s + s;

string result3 = test3("Troisième phrase de test.");

Console.WriteLine(result3 + "\n");

// Keep the console window open in debug mode.

Console.WriteLine("Press any key to exit.");

Console.ReadKey();

}

}

Attributs :

A la base une entité peut être qualifée de private, public, protected ou internal selon son degré d’accessibilité en dehors et dans la classe.

La majeure différence en C# est de pouvoir créer des attributs qui permettent eux aussi de qualifier l’entité.

Pour créer un attribut, il faut créer une classe héritant de la classe Attribute.

Une fois la classe créée, il suffit de placer l’attribut entre crochet juste avant la déclaration de classe par exemple ou encore de méthode ou de variable.

<https://docs.microsoft.com/fr-fr/dotnet/csharp/language-reference/language-specification/introduction>

https://openclassrooms.com/fr/courses/1526901-apprenez-a-developper-en-c/2866796-tableaux-listes-et-enumerations