Le C#

Bonneton Marie
Patron Pauline

Plan de l'exposé

1. Présentation du C#
    A. Historique
    B. Présentation générale
2. Caractéristiques du C#
    A. L'anatomie générale d'une classe
    B. Particularités de C#
3. Conclusion

Plan de l'exposé

1. **Présentation du C#**
    A. Historique
    B. Présentation générale
2. Caractéristiques du C#
    A. L'anatomie générale d'une classe
    B. Particularités de C#
3. Conclusion

A. Historique

- Au début était :
  - le C (début 70) :
    - Dennis Ritchie & Brian Kernighan
    - portable et compact
  - puis vint le C++ (fin 80) :
    - Bjarne Stroustrup
    - évolution du C avec des classes
    - comprend de nouveaux concepts → héritage, polymorphisme, surcharge, ...
  - et enfin le C# (2001) : *Et les programmeurs dirent que cela était bien*
    - Anders Heljsberg (Delphi)
    - Langage de programmation orientée objet

B. Présentation générale

- Son nom :
  - C♯ *(sharp)* mais plus facile d'écrire C#
  - ♯ provient de la notation musicale : évolution du C++
  - *Le C# est au C++ ce que le C++ est au C.*
- But :
  - Développement applications sur la plateforme .Net
  - Lutter contre Java
- Vocabulaire du C++ avec grammaire du JAVA
- De nouvelles fonctionnalités :
  - garbage collector
  - surcharge des opérateurs
  - Type anonyme et typage dynamique des variables
  - **...**

Plan de l'exposé

1. Présentation du C#
    A. Historique
    B. Présentation générale
2. **Caractéristiques du C#**
    A. L'anatomie générale d'une classe
    B. Particularités de C#
3. Conclusion

A. L'anatomie générale d'une classe

```
using System ;

public class Voiture

{

        Private string  nom ;

        public Voiture()

        {

                nom = «Audi R8 » ;

        }

        public void AfficheNom()

        {

                Console.writeLine(nom) ;

        }

}
```

A. L'anatomie générale d'une classe

- Déclaration d'une classe : visibilite class NomClasse
  - Différentes visibilitées :
    - Public
    - Internal
    - Protected
    - Private
    - protected internal

- Déclaration de variable :
  - Champs : Description des données d'un objet
  - Propriété : Méthode publique d'accès aux champs privés

A. L'anatomie générale d'une classe

- Initialisation d'un objet
  - NomClasse NomObjet = new NomClasse()
- Destruction d'un objet
  - Ne pas appeler explicitement
  - Appelé automatiquement
- Accès à un champs ou une propriétés
  - NomObjet.NomChamp
    - Affichage de la valeur pour le champ
    - Appel de la méthode get pour une propriété

Exemple

```
class Classe1
{
    public int minute ; // champs
}
```

```
class Classe2
{
    private int minute;
    public int Heure
    {
        get{ return minute / 60;}
        set{ minute=value * 60 ;}
    }
```

```
Classe1 c = new Classe1();
c.minute = 10;
System.console.WriteLine(c.minute) ; // affiche le nombre

Classe2 t = new Classe2();
t.Heure = 10;
 System.console.WriteLine(t.Heure)   // affiche le nombre en heure
```

A. L'anatomie générale d'une classe

- Création d'une instance de la classe :
  - Constructeur : public NomClasse()
- Destruction de l'instance :
  - Destructeur : ~ NomClasse()
- Interaction avec la classe :
  - Méthode : Visibilité valeurDeRetour Nom(type Nparam1...)
- Type des paramètres:
  - Valeur : Stocke les données (mot clé ref :passage d'un type valeur par référence)
  - Référence : Stocke une référence sur l'objet (similaire aux pointeurs en C)

Exemple

```
Public Exemple{
public void Multiplie(int i)
{      i= i*4;
       System.console.WriteLine(i);
}
public void main(){
int j =4;
Multiplie(j);
System.console.WriteLine(j)
}
```

- Resultat : - 16
  - 4

```
Public Exemple2{
public void Multiplie(ref int i)
{      i= i*4;
       System.console.WriteLine(i);
}
public void main(){
int j =4;
Multiplie(ref j);
System.console.WriteLine(j)
}
```

- Resultat : - 16
  - 16

B. Particularités de C#

- **Héritage**
  - Permet d'avoir accès aux attributs et aux méthodes d'une classe.

    Public ClasseA : ClasseB
  - Le mot clé base:

    Permet l'accès aux méthodes et constructeur de la classe mere.

    base.nomMéthode();

- **Exemple**

```
public class Vehicule
    {
       int nbroue;

       public Vehicule()
       {
          nbroue = 4;
       }
    }
```

```
Public class Voiture : Vehicule
{
       public  string nom;
       Public Voiture()
       {
             base.Vehicule;
             nom= « Auto »;
       }
}
```

B. Particularités de C#

- Constructeur privé
  - Empêche l'instanciation d'une classe

    Private NomClasse()

- Ecriture dans la console :
  - System.Console.WriteLine(« texte »);
  - Pour afficher plusieurs elements
    - System.Console.WriteLine(« (0),(1) »,elem0,elem1);

- Lecture de la console
  - System.Console.ReadLine()

B. Particularités de C#

- Les indexeurs
  - Indexation des objets similaires à celles des tableaux
  - Similaire aux propriétés
  - Déclaration:

```
public classe1<Type>{
  Type [] nomAtt;

public type this [int index]
{
      Get
      {
            Return nomAtt[index];
      }
      Set
      {
            nomAtt[index] = value;
      }
}
```

```
Classe 1 <string> objet= new Classe1
<string>();

object[0] = « Hello World »;
system.Console.WriteLine(object[0]);
```

B. Particularités de C#

- Les énumérations
    - Valeurs possibles d'un attributs
    - Possède un type sous jacent, par défaut int
  - Déclaration :

    enum nomEnum { Item1, Item2,...}
  - Initialisation variable d'énumération:

    nomEnum nomVar = nomEnum.ItemChoisi;

- Le modificateur readonly
  - Il empêche la modification d'un champ après l'initialisation de la classe

    protected readonly int nombre;

B. Particularités de C#

- Delegate

  Similaire aux pointeurs de fonctions

  - Déclaration

    visibilite delegate type nomDeleg(param1,...)

  - Initialisation

    nomDeleg nom = NomMethode;

  - Exemple

```
delegate int Operation(int i, int j);
public class1{
      public static int addition(int i, int j){ return i+j;}
      public static int soustraction(int i, int j){ return i-j;}

      static void Main() {
       Operation d1 = addition;  Operation d2 = soustraction;
       Int i = 5 , int j = 2;
       d1(i,j);
       d2(i,j);
   }
```

B. Particularités de C#

- Évènements
  - Notification d'évènement aux « abonnés»
  - Utilisation du type delegate pour pouvoir déclarer des Events
  - Déclaration
    - public delegate type NomDeleg(param1,...);
    - public event NomDeleg NomEvent;
  - Gestionnaire d'évènement : procédure à réaliser lorsqu'un évènement survient
    - type NomGest(param1,...);
  - Association
    - NomObjet.NomEvenement += new NomDeleg(NomGest);

Exemple

```
class Program

{

static void Main(string[] args)

{

    Surveillant surveillant = new Surveillant();

    Samu samu = new Samu();

    surveillant.Accident += new
Surveillant.AccidentHandler(samu.onAccident);

    surveillant.Signaler("Chernobyl");

}

class AccidentEventArgs : EventArgs

{

public string Adresse;

public AccidentEventArgs(string a) {Adresse = a;}

}
```

```
class Surveillant
{
        public void Signaler(string adr)
        {
                AccidentEventArgs e = new
AccidentEventArgs(adr);
                if (Accident != null) Accident(this, e);
        }

        public delegate void AccidentHandler(object
sender,                         AccidentEventArgs acc);
        public event AccidentHandler Accident;
}


class Samu
{
public void onAccident(object sender,
AccidentEventArgs e)
{
Console.WriteLine("Appel reçu pour " + e.Adresse);
}
}
```

Plan de l'exposé

1. Présentation du C#
    A. Historique
    B. Présentation générale
2. Caractéristiques du C#
    A. L'anatomie générale d'une classe
    B. Particularités de C#
3. **Conclusion**

Conclusion

- Langage très utilisé
- Ressemblance forte avec Java
- Fortement couplé avec .Net
- Programmation évènementielle

Avez vous des questions ?