Module 10: Héritage dans C

technofutur TIC
WAVENET
Intelligent IT Solutions

Vue d'ensemble

- Dériver des classes
- Mise en œuvre de méthodes
- Utilisation de classes scellées
- Utilisation des interfaces
- Utilisation de classes abstraites

technofutur TIC
WAVENET
Intelligent IT Solutions

Dériver des classes

- Extension des classes de base
- Accès aux membres base de la classe
- Appel aux constructeurs de base de la classe

Extension des classes de base

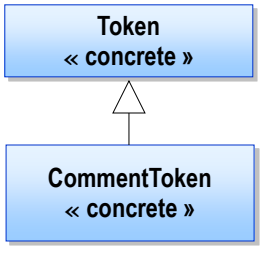
- Syntaxe pour dériver une classe à partir d'une classe de base

```
class Token
{
    ...
}
class CommentToken: Token
{
    ...
}
```

Derived class

Base class

Colon



- Une classe dérivée hérite de la plupart des éléments de sa classe de base
- Une classe dérivée ne peut pas être plus accessible que sa classe de base

Accès aux membres base de la classe

```
class Token
{   ...
    protected string name;
}
class CommentToken: Token
{   ...
    public string Name( )
    {
        return name;
    }
}
```

✓

```
class Outside
{
    void Fails(Token t)
    {
        ...
         t.name
        ...
    }
}
```

✘

- Membres protégées héritées sont implicitement protégés dans la classe dérivée
- Les méthodes d'une classe dérivée ne peuvent accédés que part leurs membres protégées héritées.
- Les modificateurs accès **protected ne peuvent pas** être utilisées dans une **struct**

technofutur TIC

WAVENET Intelligent IT Solutions

Appel classe de base Constructeurs

- Les déclarations de constructeur doivent utiliser le mot clé de base

```
class Token
{
    protected Token(string name) { ... }
    ...
}
class CommentToken: Token
{
    public CommentToken(string name) : base(name) { }
    ...
}
```

- Un constructeur de classe de base privée ne peut pas être accessible par une classe dérivée

technofutur TIC

WAVENET Intelligent IT Solutions

Mise en œuvre de méthodes

- Définition des méthodes virtuelles
- Travailler avec des méthodes virtuelles
- La substitution de méthodes
- Travailler avec des méthodes Override
- Utilisation du mot clé new pour masquer les méthodes
- Travailler avec le mot clé new
- Pratique: Méthodes d'exécution
- Quiz: Découvrir les Bugs

Définition des méthodes virtuelles

- Syntaxe: Déclarer en tant que virtual

```
class Token
{
    ...
    public int LineNumber( )
    { ...
    }
    public virtual string Name( )
    { ...
    }
}
```

- Les méthodes virtuelles sont polymorphes

Travailler avec des méthodes virtuelles

- Pour utiliser des méthodes virtuelles:
  - Vous ne pouvez pas déclarer des méthodes virtuelles statique
  - Vous ne pouvez pas déclarer des méthodes virtuelles privé

technofutur TIC
WAVENET Intelligent IT Solutions

L'override de méthodes

- Syntaxe: Utilisez le mot-clé override

```
class Token
{   ...
    public virtual string Name( ) { ... }
}
class CommentToken: Token
{   ...
    public override string Name( ) { ... }
}
```

technofutur TIC
WAVENET Intelligent IT Solutions

Travailler avec des méthodes Override

- Vous devez redéfinir les méthodes virtuelles héritées identiques

```
class Token
{   ...
    public int LineNumber( )  { ... }
    public virtual string Name( ) { ... }
}
class CommentToken: Token
{   ...
    public override int LineNumber( ) { ... }
    public override string Name( ) { ... }
}
```

- Vous devez faire correspondre une méthode de substitution avec sa méthode virtuelle associée
- Vous pouvez « override » une méthode déjà « overridée »
- Vous ne pouvez pas déclarer explicitement une méthode override virtual
- Vous ne pouvez pas déclarer une méthode statique ou privé override

technofuturTIC

WAVENET
Intelligent IT Solutions

Utilisation de new pour masquer les méthodes

- Syntaxe: Utilisez le mot-clé new pour masquer une méthode

```
class Token
{   ...
    public int LineNumber( )  { ... }
}
class CommentToken: Token
{   ...
    new public int LineNumber( )  { ... }

}
```

technofuturTIC

WAVENET
Intelligent IT Solutions

Travailler avec le mot-clé new

- Masquer les méthodes virtuelles et non virtuelles

```
class Token
{   ...
    public int LineNumber( )  { ... }
    public virtual string Name( ) { ... }
}
class CommentToken: Token
{   ...
    new public int LineNumber( )  { ... }
    public override string Name( ) { ... }
}
```

- Résous les conflits de noms dans le code
- Masque les méthodes qui ont des signatures identiques

technofutur TIC

WAVENET Intelligent IT Solutions

Pratique: Méthodes d'exécution

Que va-t-il se passer ?

```
class A {
   public virtual void M() { Console.Write("A"); }
}
class B: A {
   public override void M() { Console.Write("B"); }
}
class C: B {
   new public virtual void M() { Console.Write("C"); }
}
class D: C {
   public override void M() { Console.Write("D"); }
   static void Main() {
       D d = new D(); C c = d; B b = c; A a = b;
       d.M(); c.M(); b.M(); a.M();
   }
}
```

technofutur TIC

WAVENET Intelligent IT Solutions

Quiz: Découvrir les Bugs

```
class Base
{
    public void Alpha( ) { ... }
    public virtual void Beta( ) { ... }
    public virtual void Gamma(int i) { ... }
    public virtual void Delta( ) { ... }
    private virtual void Epsilon( ) { ... }
}
class Derived: Base
{
    public override void Alpha( ) { ... }
    protected override void Beta( ) { ... }
    public override void Gamma(double d) { ... }
    public override int Delta( ) { ... }
}
```

technofutur TIC

WAVENET Intelligent IT Solutions

Utilisation de classes sealed (scellées)

- Vous pouvez utiliser des classes **sealed** pour rendre vos classes inhéritables
- Plusieurs classes du Framework .NET sont **sealed**: *String, StringBuilder, et ainsi de suite*
- Syntaxe: Utilisez le mot-clé **sealed**

```
namespace System
{
    public sealed class String
    {
        ...
    }
}
namespace Mine
{
    class FancyString: String { ... }
}
```

technofutur TIC

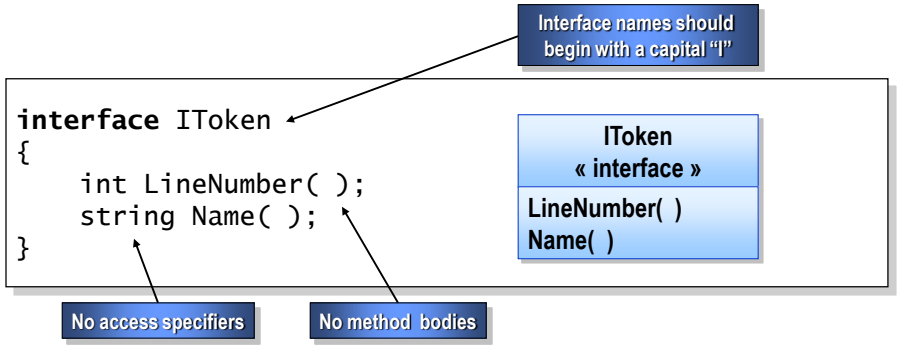
WAVENET Intelligent IT Solutions

Utilisation des interfaces

- Déclaration d'interfaces
- Implémentation d'interfaces multiples
- Implémentation de méthodes d'interface
- Implémentation explicite de méthodes d'interface
- Quiz: Découvrir les Bugs

Interfaces déclarant

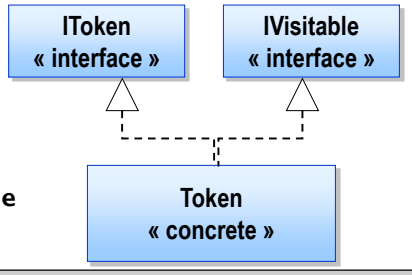
- Syntaxe:
  - Utilisez le mot-clé interface pour la déclarer
  - déclarez des méthodes propres à cette interface



Implémentation d'interfaces multiples

- Une **classe** peut implémenter zéro ou plusieurs **interfaces**

```
interface IToken
{
    string Name( );
}
interface IVisitable
{
    void Accept(IVisitor v);
}
class Token: IToken, IVisitable
{ ...
}
```



- Une **interface** peut étendre zéro ou plusieurs **interfaces**
- Une classe peut être plus accessible que ses interfaces de base
- Une interface ne peut pas être plus accessible que ses interfaces de base
- Une classe doit implémenter toutes les méthodes d'interface héritées

technofutur TIC

WAVENET
Intelligent IT Solutions

Implémentation des méthodes d'interface

- Les méthode doivent être Implémentatée avec la meme signature que celle dans l'interface
- Les méthode Implémentatée peuvent être virtual ou pas.

```
Jeton de classe: itoken, IVisitable
{
    public String virtuel Nom ()
    {...
    }
    public void Accepter (IVisitor v)
    {...
    }
}
```

Même accès
Type de retour même
Même nom
Mêmes paramètres

technofutur TIC

WAVENET
Intelligent IT Solutions

Implémentation de méthodes d'interface explicite

- Utilisez le nom de l'interface explicitement

```
class Token: IToken, IVisitable
{
    string IToken.Name( )
    { ...
    }
    void IVisitable.Accept(IVisitor v)
    { ...
    }
}
```

- Vous ne pouvez accéder aux méthodes que via l'interface
- Vous ne pouvez pas déclarer des méthodes virtual
- Vous ne pouvez pas spécifier un modificateur d'accès

technofutur TIC

WAVENET Intelligent IT Solutions

Quiz: Découvrir les Bugs

```
interface IToken
{
    string Name( );
    int LineNumber( ) { return 42; }
    string name;
}

class Token
{
    string IToken.Name( ) { ... }
    static void Main( )
    {
        IToken t = new IToken( );
    }
}
```

technofutur TIC

WAVENET Intelligent IT Solutions

Utilisation de classes abstraites

- Déclaration de classes abstraites
- Utilisation de classes abstraites dans une hiérarchie de classe
- Comparaison classes abstraites et interfaces
- Implémentation de méthodes abstraites
- Travailler avec des méthodes abstraites
- Quiz: Découvrir les Bugs

Déclarer classes abstraites

- Utilisez le mot-clé abstract

```
abstract class Token
{
    ...
}
class Test
{
    static void Main( )
    {
        new Token( );
    }
}
```

Token
{ abstract }

An abstract class cannot be instantiated

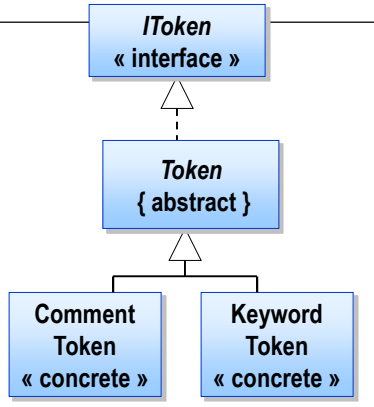
Utilisation de classes abstraites dans une hiérarchie de classe

- - Exemple 1

```
interface IToken
{
    string Name( );
}

abstract class Token: IToken
{
    string IToken.Name( )
    {   ...
    }
    ...
}

class CommentToken: Token
{   ...
}
class KeywordToken: Token
{   ...
}
```



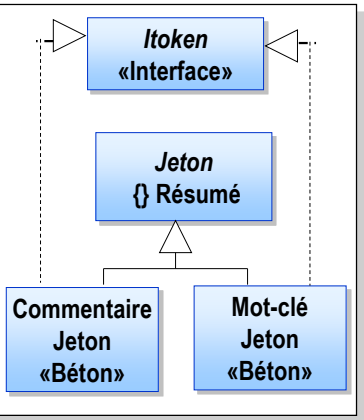
Utilisation de classes abstraites dans une hiérarchie de classe *(Suite)*

- - Exemple 2

```
Interface itoken
{
    string Name ();
}

résumé classe Token
{
    public String virtuel Nom ()
    {...
    }
    ...
}

classe CommentToken: Token, itoken
{...
}
classe KeywordToken: Token, itoken
{...
}
```



Comparaison classes abstraites aux interfaces

- Similitudes
  - Ne peut être instancié
  - Ne peut être sealed
- Différences
  - Interfaces ne peuvent pas contenir d'implémentation
  - Les interfaces ne peuvent pas déclarer de membres non publics
  - Les interfaces ne peuvent étendre que des interfaces

Mise en œuvre de méthodes abstraites

- Syntaxe: Utilisez le mot-clé abstract

```
abstract class Token
{
    public virtual string Name( ) { ... }
    public abstract int Length( );
}
class CommentToken: Token
{
    public override string Name( ) { ... }
    public override int Length( ) { ... }
}
```

- Seules les classes abstraites peuvent déclarer des méthodes abstraites
- Les méthodes abstraites ne peuvent pas contenir un corps de méthode

Travailler avec des méthodes abstraites

- Les méthodes abstraites sont virtuelles
- Les méthodes abstraites ne doivent pas forcément etre override dans les classes « filles » si celles-ci sont aussi abstraites
- Les méthodes abstraites peuvent remplacer les méthodes de la classe de base déclarés **virtual**
- Les méthodes abstraites peuvent remplacer les méthodes de la classe de base déclarés comme **override**

technofutur TIC

WAVENET
Intelligent IT Solutions

Quiz: Découvrir les Bugs

```
class First
{
    public abstract void Method( );
}
```

1

```
abstract class Second
{
    public abstract void Method( ) { }
}
```

2

```
interface IThird
{
    void Method( );
}
abstract class Third: IThird
{
}
```

3

technofutur TIC

WAVENET
Intelligent IT Solutions