Module 13:
Propriétés et indexeurs

Vue d'ensemble

- Utilisation des propriétés
- Utilisation indexeurs

Utilisation des propriétés

- Pourquoi utiliser des propriétés?
- Utilisation des accesseurs
- Comparaison entre les propriétés et les champs
- Comparaison entre les propriétés et les méthodes
- Types de propriété

Pourquoi utiliser des propriétés?

- Propriétés fournissent:
  - Un moyen utile pour encapsuler l'information dans une classe
  - Syntaxe concise
  - Flexibilité

Utilisation des accesseurs

- Les propriétés permenttent un accès semblable au champs (variables)
  - Utilisez **get** pour fournir un accès en lecture
  - Utilisez **set** pour fournir un accès en écriture

```
Bouton de classe
{
    public string Légende / / propriété
    {
        se {return légende;}
        set {caption = valeur;}
    }
    chaîne légende privé; / / Champ
}
```

technofutur TIC

WAVENET
Intelligent IT Solutions

Comparaison entre les propriétés et les champs

- Les propriétés sont des “champs logiques"
  - L’accesseur **get** peut retourner une valeur calculée
- Similitudes
  - La syntaxe pour l’affectation et l'utilisation est la même
- Différences
  - Propriétés ne sont pas des valeurs, elles n'ont pas d'adresse
  - Les propriétés ne peuvent pas être utilisés comme paramètres **ref** ou **out**

technofutur TIC

WAVENET
Intelligent IT Solutions

Comparaison entre les propriétés et les méthodes

- Similitudes
  - Les deux contiennent du code pour être exécuté
  - Les deux peuvent être utilisés pour masquer les détails d'implémentation
  - Les deux peuvent être virtual, abstract ou override
- Différences
  - Syntaxique : les propriétés n'utilisent pas entre parenthèses
  - Sémantique – les propriétés ne peuvent pas être **void** ou prendre des paramètres arbitrairerement

Types de propriété

- Propriétés en lecture/écriture
  - À la fois **get** et **set**
- Propriétés en lecture seule
  - Uniquement **get**
  - Ne sont pas constantes
- Propriétés en écriture seule - *utilisation très limitée*
  - Uniquement **set**
- Propriétés statiques
  - Appliqué à la classe et ne peut accéder qu'aux données statiques

Exemple de la propriété

```
public class Console
{
    public static TextReader In
    {
        get {
            if (reader == null) {
                reader = new StreamReader(...);
            }
            return reader;
        }
    }
    ...
    private static TextReader reader = null;
}
```


Utilisation indexeurs

- Qu'est-ce qu'un Indexer?
- Comparaison entre les indexeurs et les tableaux
- Comparaison entre les indexeurs et les propriétés
- Utilisation de paramètres pour définir des indexeurs
- Exemple de string
- Exemple de BitArray

Qu'est-ce qu'un Indexer?

- Un indexeur fournit un accès de type tableau à un objet
  - Utile si une propriété peut avoir plusieurs valeurs
- Pour définir un indexeur
  - Créer une propriété appelée *this*
  - Spécifiez le type d'index
- Pour utiliser un indexeur
  - Utilisez la notation de tableau pour lire ou écrire la propriété indexée

Comparaison indexeurs de tableaux

- Similitudes
  - Les deux utilisent la notation de tableau
- Différences
  - Les indexeurs peuvent utiliser des indices non entiers
  - Les indexeurs peuvent être surchargés, vous pouvez définir plusieurs indexeurs, chacun utilisant un type d'index différent
  - Les indexeurs ne sont pas des variables. Ils ne désignent pas des endroits de stockage. Vous ne pouvez pas passer un indexeur comme parametre **ref** ou **out**

Comparaison entre les indexeurs et les propriétés

- Similitudes
  - Utilisation des accesseurs **get** et **set**
  - Aucune adresse
  - Peuvent être vide
- Différences
  - Les indexeurs peuvent avoir plusieurs surcharges
  - Les indexeurs ne peuvent pas être statique

Utilisation de paramètres pour définir des indexeurs

- Lors de la définition des indexeurs
  - Indiquez au moins un paramètre d'indexation
  - Spécifiez une valeur pour chaque paramètre que vous spécifiez
  - Ne pas utiliser **ref** ou **out** modificateurs de paramètres

Exemple de chaîne

- La classe String
  - Est une classe immuable
  - Utilise un indexeur (**obtenir** accesseur mais pas **ensemble** accesseur)

```
classe String
{
    omble publique ce [int index]
    {
        get {
          if (index <0 | | index> = Longueur)
              throw new IndexOutOfRangeException ();
          ...
        }
    }
    ...
}
```

technofutur TIC

WAVENET Intelligent IT Solutions

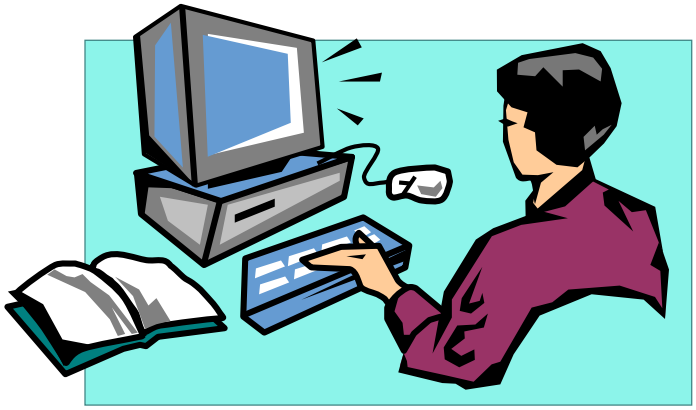
BitArray Exemple

```
classe BitArray
{
   public bool ce [int index]
   {
     get {
        BoundsCheck (indice);
        retourner (bits [index >> 5] et (1 << index)) = 0!;
     }
     mettre {
        BoundsCheck (indice);
        si (valeur) {
            les bits [index >> 5] | = (1 << index);
        } Else {
            les bits [index >> 5] & = ~ (1 << index);
        }
     }
   }
   private int [] les bits;
}
```

technofutur TIC

WAVENET Intelligent IT Solutions

Lab 13.1: Utilisation des propriétés et indexeurs



technofutur TIC

WAVENET
Intelligent IT Solutions

Examen

- Utilisation des propriétés
- Utilisation indexeurs

technofutur TIC

WAVENET
Intelligent IT Solutions