Module 7

Encapsulation des Données et des Méthodes

Sommaire

- Controller la Visibilité des Membres d’un Type
- Partager des Méthodes et des Données

Leçon 1: Controller la Visibilité des Membres d'un Type

- Qu'est-ce que l'Encapsulation?
- Comparaison de Membres Private et Public
- Comparaison des Types Internal et Public

Qu’est-ce que l’Encapsulation?

Définition

- Un principe essentiel de la programmation orientée objet
- Masque les algorithmes et les données internes
- Fournit une exploitation publique bien définie

Bénéfices

- Rend le code externe plus simple et plus cohérent
- Vous permet de modifier les détails d’implémentation plus tard

Comparaison de Membres Private et Public

private Niveau d'accès le moins permissif

```
class Sales
{
    private double monthlyProfit;
    private void SetMonthlyProfit(double monthlyProfit)
    {
        this.monthlyProfit = monthlyProfit;
    }
}
```

Uniquement accessible à partir de la classe **Sales**

public Niveau d'accès le plus permissif

```
class Sales
{
    private double monthlyProfit;
    public void SetMonthlyProfit(double monthlyProfit)
    {
        this.monthlyProfit = monthlyProfit;
    }
}
```

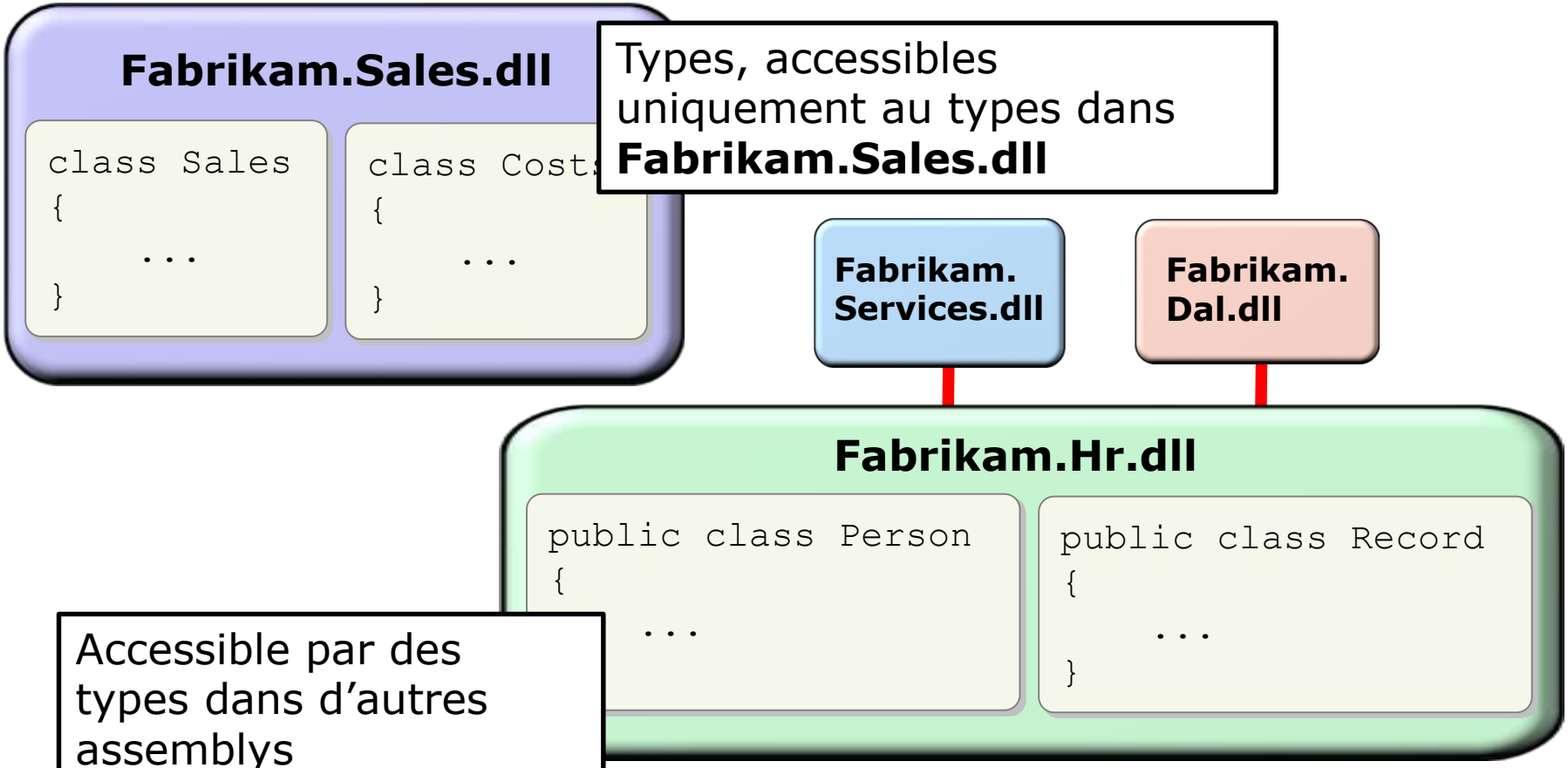
Accessible par n'importe quel autre type

Comparaison des Types Internal et Public

Internal est le modificateur d'accès par défaut pour les types

Internal permet l'accès uniquement à des types du même assembly

Public permet d'accéder à d'autres types dans d'autres assemblys



Leçon 2: Partager des Méthodes et des Données

- Créer et utiliser des Champs Static
- Créer et utiliser des Méthodes Static
- Créer des Types Static et Utiliser des Constructeurs Static
- Créer et Utiliser des Méthodes d’Extension

Créer et utiliser des Champs Static

Les Champs d'instance contiennent des données pertinentes à une instance spécifique d'un type

Les Champs statiques contiennent des données pertinentes au type-même

```
class Sales
{
    public static double salesTaxPercentage = 20;
}
```

Pour accéder à un champ statique, vous utilisez le nom de la classe, suivie d'une période, suivie du nom du champ

```
Sales.salesTaxPercentage = 32;
```

Créer et utiliser des Méthodes Statiques

Les méthodes statiques fournissent des fonctionnalités pertinentes au type lui-même et non à une instance du type

Vous pouvez utiliser les méthodes statiques pour implémenter les classes utilitaires, comme **File** dans l'espace de noms **System.IO**

```
class Sales
{
    public static double GetMonthlySalesTax(double
        monthlyProfit)
    {
        ...
    }
}
```

Vous n'avez pas besoin de créer une instance du type pour utiliser les méthodes statiques

```
double monthlySalesTax = Sales.GetMonthlySalesTax(34267);
```

Créer des Types Static et Utiliser des Constructeurs Statics

Vous déclarez des types statiques en utilisant le modificateur **static**

Les types statiques ne peuvent contenir que des membres statiques

```
static class Sales
{
}
```

Défini avec le modificateur **static**

Les types Statiques et d'instance peuvent avoir des constructeurs

Vous pouvez appeler des constructeurs d'instance avec le mot clé**new**

Les constructeurs statics sont appelés implicitement par le CLR

```
static class Sales
{
    static Sales()
    {
    }
}
```

Défini avec le modificateur **static**

Défini sans paramètres

Défini sans modificateur d'accès

Créer et Utiliser des Méthodes d'Extension

Ajoutez les méthodes d'extension à une classe existante sans :

- Avoir accès à la source de la classe existante
- Modifier ou recompiler la classe existante
- Dériver un nouveau type de la classe existante

```
namespace Fabrikam.Extensions
{
    static class IntExtension
    {
        internal static int NextRand(this int seed, int
            maxValue)
        {
            Random randomNumberGenerator = new Random(seed);
            return randomNumberGenerator.Next(maxValue);
        }
    }
}
```

Méthode Static

Type d'Extension

```
using Fabrikam.Extensions;
...
int i = 8;
int j = i.NextRand(20);
```

Pas de reference à la classe **IntExtension**

Atelier Pratique

- Exercice 1:
- Exercice 2:
- Exercice 3: