Module 3

Déclarer et Appeler des Méthodes

Sommaire

- Définir et Invoquer des Méthodes
- Spécifier des Paramètres Optionnels et des Paramètres de Sortie

Leçon 1: Définir et Invoquer des Méthodes

- Qu’est-ce qu’une méthode?
- Créer une Méthode
- Appeler une Méthode
- Créer et Appeler des Méthodes Surchargées
- Utiliser des Tableaux de Paramètres
- Refactorer du Code en Méthode
- Tester une Méthode

Qu'est-ce qu'une méthode?

Une méthode est un membre de classe qui contient un bloc de code qui représente une action:

- Tout le code exécutable appartient à une méthode
- Une application c# doit avoir au moins une méthode
- Les méthodes peuvent être définies pour un usage privé pour un type, ou pour un usage public par d'autres types

La méthode Main définit le point de départ d'une application

C# prend en charge les méthodes statiques et d'instances

Méthode d'Instance

```
int count = 99;
string strCount =
    count.ToString();
```

Méthode Statique

```
string strCount = "99";
count =
   Convert.ToInt32(strCount);
```

Créer une Méthode

Une méthode contient:

- Spécification de la méthode (accès, type de retour, et signature)
- Corps de la méthode (code)

Signature de la Méthode:

1. **Nom** (Pascal case)
2. **Paramètres** (camel case)

Chaque signature de méthode doit être unique dans un type

Exemple

```
bool LockReport(string userName)
{
    bool success = false;
    // Perform some processing here.
    return success;
}
```

Appeler une Méthode

Pour appeler une méthode, il faut:

- Spécifier le nom de la méthode
- Fournir un argument pour chaque paramètre
- Gérer la valeur de retour

Exemple de method

```
bool LockReport(string reportName, string userName)
{
    ...;
}
```

Appelle de la Méthode

```
bool isReportLocked =
    LockReport("Medical Report", "Don Hall");
```

Note: Les Arguments de la gauche vers la droite

Créer et Appeler des Méthodes Surchargées

Une méthode surchargée:

- Possède le même nom qu'une méthode existante
- *Doit* effectuer la même opération que la méthode existante
- Utilise des paramètres différents pour effectuer l'opération

Le compilateur invoque la version qui convient aux arguments spécifiés lorsque la méthode est appelée

Exemple

```
void Deposit(decimal amount)
{
    _balance += amount;
}

void Deposit(int dollars, int cents)
{
    _balance += dollars + (cents / 100.0m);
}
```

La signature de la méthode de Deposit est différente pour chaque implémentation

Utiliser des Tableaux de Paramètres

La surcharge peut ne pas être possible ou appropriée si une méthode peut prendre un nombre variable d'arguments

Exemple

```
int Add(params int[] data)
{
    int sum = 0;
    for (int i = 0; i < data.Length; i++)
    {
        sum += data[i];
    }
    return sum;
}
```

Appelle de la Méthode

```
int sum = Add(99, 2, 55, -26);
```

Refactoriser du Code en Méthode

1. Sélectionnez le code que vous souhaitez extraire d'une méthode, faites un clic droit, pointer sur **Refactoriser** et puis cliquez sur **Extraire la méthode**

2. Dans la boîte de dialogue **Extraire la méthode**, dans la zone de nom de méthode, tapez un nom pour la méthode et puis cliquez sur **OK**

Exemple de code refactorisé

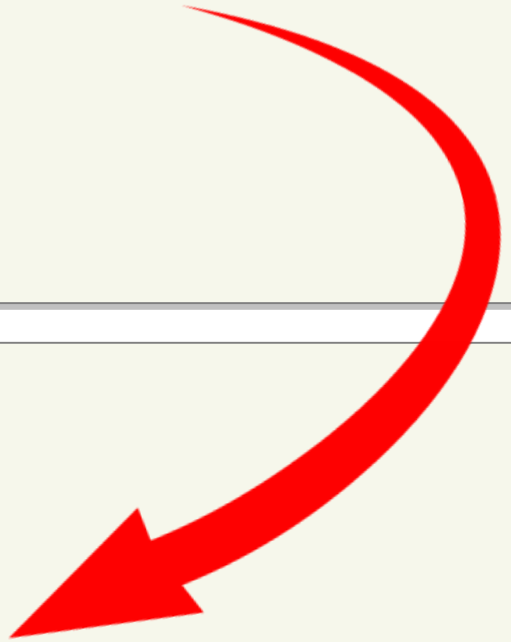
```
LogMessage(messageContents, filePath);
...
private void LogMessage(string messageContents, string filePath)
{
    ...
}
```

Tester une Méthode

```
int Calculate(int operandOne, int operandTwo)
{
    int result = 0;

    // Perform some calculation.

    return result;
}
```



```
[TestMethod()]
public void CalculateTest()
{
    Program target = new Program();
    int operandOne = 0;
    int operandTwo = 0;
    int expected = 0;
    int actual;
    actual = target.Calculate(operandOne, operandTwo);
    Assert.AreEqual(expected, actual);
    ...
}
```

Leçon 2: Spécifier des Paramètres Optionnels et des Paramètres de Sortie

- Qu’est-ce que les Paramètres Optionnels?
- Appeler une Méthode en Utilisant des Arguments Nommés
- Qu’est-ce que les Paramètres de Sortie?

Qu’est-ce que les Paramètres Optionnels?

Les paramètres optionnels vous permettent de définir une méthode et de fournir des valeurs par défaut pour les paramètres :

- Ils sont utiles pour interagir avec d’autres technologies qui prennent en charge les paramètres facultatifs
- Ils peuvent fournir une implémentation alternative à la surcharge si cette dernière n’est pas appropriée

Exemple

```
void MyMethod(
    int intData, float floatData, int moreIntData = 99)
{
    ...
}
```

```
// Arguments provided for all three parameters
MyMethod(1o, 123.45, 99);

// Arguments provided for 1st two parameters only
MyMethod(100, 54.321);
```

Appeler une Méthode en Utilisant des Arguments Nommés

En spécifiant des paramètres par leur nom:

- les Arguments peuvent être fournis dans n'importe quel ordre
- les Arguments omis se voient affecter des valeurs par défaut

si le paramètre correspondant est facultatif

Exemple

```
void MyMethod(
  int intData, float floatData = 101.1F, int moreIntData = 99)
{
    ...
}
```

```
MyMethod(moreIntData: 100, intData: 10);
// floatData is assigned default value
```

Qu’est-ce que les Paramètres de Sortie?

Les Paramètres de sortie vous permettent de passer une valeur de retour d’une méthode à un argument :

- Utilisez le mot clé **out** dans la liste des paramètres

```
void MyMethod(int first, double second, out int data)
{
    ...
    data = 99;
}
```

Appelez la méthode avec une variable spécifiée comme un argument out

```
int value;
MyMethod(10, 101.1F, out value);
// value = 99
```

Atelier Pratique

- Exercice 1: