



Les délégués et les événements en C#

Mme Ben bouzid NOURHENE



Les délégués en C#

Mme Ben bouzid NOURHENE

Introduction:

- Un délégué est un type qui représente des références aux méthodes avec une liste de paramètres et un type de retour particuliers.
- Grâce aux délégués (delegates), vous pouvez transmettre une méthode comme paramètre : c'est un peu l'équivalent des pointeurs de fonction en C / C++ sauf que les délégués sont totalement orientés objet.
- **delegate** est aussi un mot clé du langage C# qui permet de déclarer un delegate.
- La classe **Delegate** est la classe de base pour les types délégué.
- une méthode doit avoir le même type de retour que le délégué: dans le contexte de surcharge de méthodes, la signature d'une méthode n'inclut pas la valeur de retour mais dans le contexte des délégués, la signature inclut la valeur de retour.

Déclaration des délégués

- Pour déclarer un delegate en CSharp:
 - Utiliser le mot clé delegate suivi de la signature de la méthode associée dans la classe.
 - Le nom de la méthode indiquée dans la signature sera le nom de votre delegate.
 - On peut ajouter un indicateur de visibilité au début.
- Une fois qu'un délégué est déclaré, son instance se réfère et appelle les méthodes dont le type de retour et la liste de paramètres correspondent à la déclaration de délégué.
- NB: Les types délégués sont **sealed**, c'est-à-dire qu'ils ne peuvent pas être faire l'objet d'une dérivation

Déclaration des délégués

➤ Exemple:

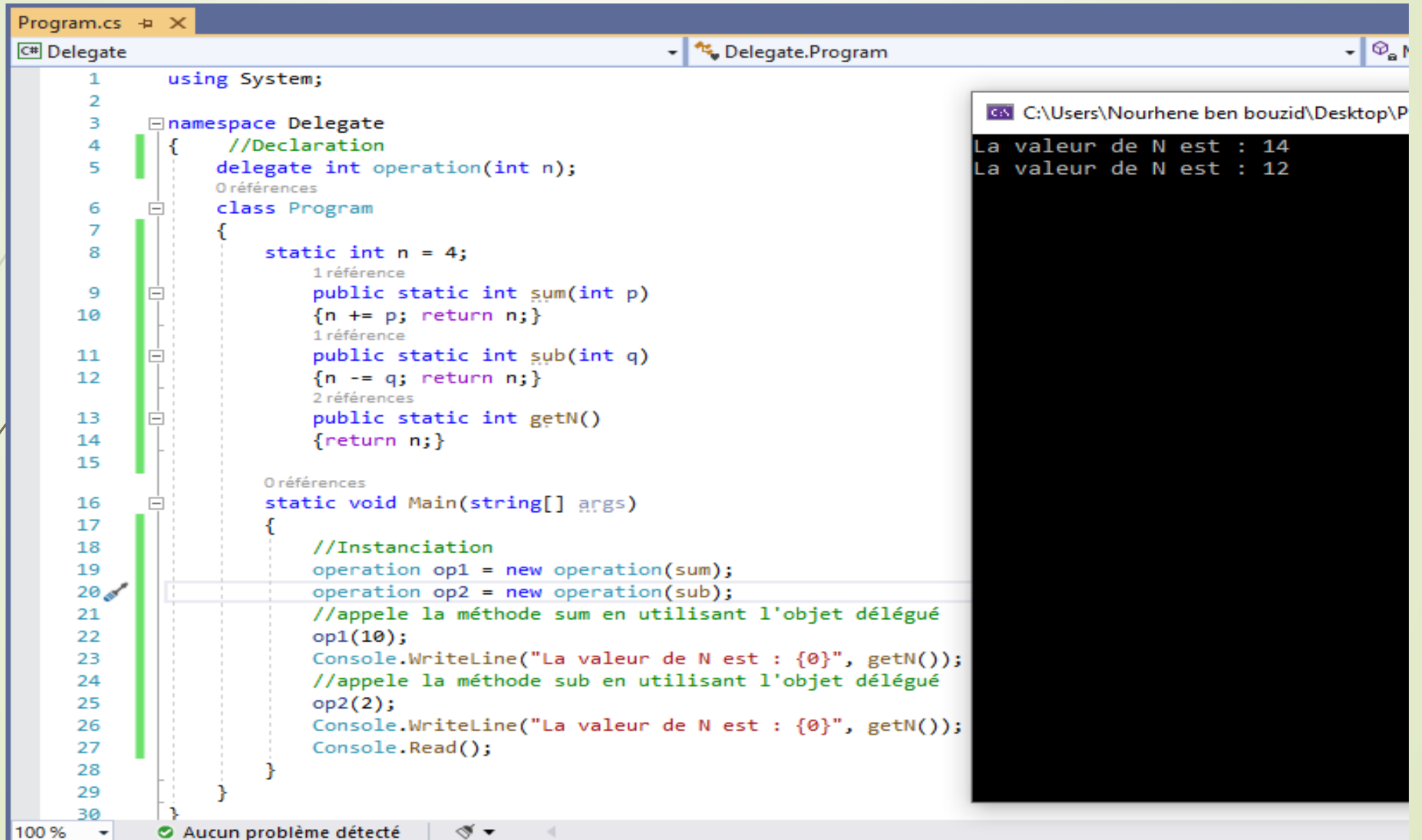
```
{  
    public delegate int MyDelegate(string str);  
    ...  
}
```

- Le délégué «MyDelegate» peut être utilisé pour référencer toute méthode comportant un seul paramètre de type string et renvoyant une variable de type Int.

Instanciación de delegados

- Une fois qu'un type de délégué est déclaré, un objet délégué doit être créé avec le mot-clé « **new** » et doit être associé à une méthode particulière.
- Lors de la création d'un délégué, l'argument transmis à l'expression « new » est le nom de la méthode sans indiquer les arguments.

Instanciación de delegados



```
Program.cs
C# Delegate Delegate.Program

1 using System;
2
3 namespace Delegate
4 {
5     //Declaration
6     delegate int operation(int n);
7     class Program
8     {
9         static int n = 4;
10        public static int sum(int p)
11        {n += p; return n;}
12        public static int sub(int q)
13        {n -= q; return n;}
14        public static int getN()
15        {return n;}
16
17        static void Main(string[] args)
18        {
19            //Instanciación
20            operation op1 = new operation(sum);
21            operation op2 = new operation(sub);
22            //apelle la méthode sum en utilisant l'objet délégué
23            op1(10);
24            Console.WriteLine("La valeur de N est : {0}", getN());
25            //apelle la méthode sub en utilisant l'objet délégué
26            op2(2);
27            Console.WriteLine("La valeur de N est : {0}", getN());
28            Console.Read();
29        }
30    }
}
```

C:\Users\Nourhene ben bouzid\Desktop\Program.cs

La valeur de N est : 14
La valeur de N est : 12

100 % Aucun problème détecté

Les événements en C#



Introduction:

- Les événements permettent à une classe ou à un objet de notifier d'autres classes ou objets quand quelque chose de significatif se produit.
- La classe qui déclenche l'événement est appelée **publisher** et les classes qui reçoivent ou gèrent l'événement sont appelées **subscriber**.
- Les événements sont déclarés et déclenchés dans une classe et associés aux gestionnaires d'événements à l'aide de délégués au sein de la même classe ou d'une autre classe.
- **Publisher** est un objet contenant la définition de l'événement et du délégué. L'association événement-délégué est également définie dans cet objet.
- **Subscriber** est un objet qui accepte l'événement et fournit un gestionnaire d'événements. Le délégué de la classe «Publisher » appelle la méthode de la classe « Subscriber ».

Déclaration des événements

- Pour déclarer un événement dans une classe, vous devez d'abord déclarer un type de délégué pour l'événement.
- Ensuite, l'événement est déclaré à l'aide du mot-clé **event**.

```
class Program
{
    delegate int operation(int n);
    private event operation Event_eg;
```

Déclaration des événements

```
//This delegate can be used to point to methods
//which return void and take a string.
public delegate void MyEventHandler(string foo);

//This event can cause any method which conforms
//to MyEventHandler to be called.
public event MyEventHandler SomethingHappened;

//Here is some code I want to be executed
//when SomethingHappened fires.
void HandleSomethingHappened(string foo)
{
    //Do some stuff
}

//I am creating a delegate (pointer) to HandleSomethingHappened
//and adding it to SomethingHappened's list of "Event Handlers".
myObj.SomethingHappened += new MyEventHandler(HandleSomethingHappened);

//To raise the event within a method.
SomethingHappened("bar");
```

Déclaration des événements

- Les événements ont les propriétés suivantes :
 - Le publieur détermine quand un événement est déclenché ; les abonnés déterminent l'action entreprise en réponse à l'événement.
 - Un événement peut avoir plusieurs abonnés. Un abonné peut gérer plusieurs événements provenant de plusieurs publieurs.
 - Les événements qui n'ont aucun abonné ne sont jamais déclenchés.
 - Les événements sont généralement utilisés pour signaler des actions de l'utilisateur, comme les clics de bouton ou les sélections de menu dans les interfaces utilisateur graphiques.

Exemple:

```
public delegate myDelegate();
```

```
event MyDelegate OnEvent;
```

```
class Account
{
    double credit;

    public delegate void DebitBalanceDelegate();

    public event DebitBalanceDelegate OnDebitBalance;

    void withdraw(double amount)
    {
        // on lève l'évènement lorsque le débit est négatif
        if(credit - amount < 0 && OnDebitBalance != null) OnDebitBalance();
        //...
    }
}
```

les méthodes Sms_DebitBalanceReached et Email_DebitBalanceReached seront exécutées lorsque l'évènement sera levé. La classe main est abonnée à cet évènement par ces méthodes → le code de ces deux méthodes sera exécuté.

```
static void Main(string[] args)
{
    Account acc = new Account();

    acc.OnDebitBalance += Sms_DebitBalanceReached;
    acc.OnDebitBalance += Email_DebitBalanceReached;
    acc.withdraw(100);
}

static void Sms_DebitBalanceReached()
{
    Console.WriteLine("Sms debit balance sent");
}

static void Email_DebitBalanceReached()
{
    Console.WriteLine("Email debit balance sent");
}
```



Merci pour votre attention