

C#.Net

Eventos

Repaso Delegados - Ejercicio

- Vamos a codificar una clase **Cuenta5** que utilizaremos para contar en forma regresiva 5 segundos.
- Vamos a escribir los segundos que faltan en la consola cada vez que transcurra un segundo.
- En principio todo el trabajo lo va a realizar la clase **Cuenta5**
- Luego iremos transformando el código hasta implementar un mecanismo de eventos. La clase **Cuenta5** lanzará el evento **SegCumplido** al que podrán suscribirse otras clases para manejarlo en forma conveniente, en este caso simplemente imprimiendo en la consola los segundos que faltan.

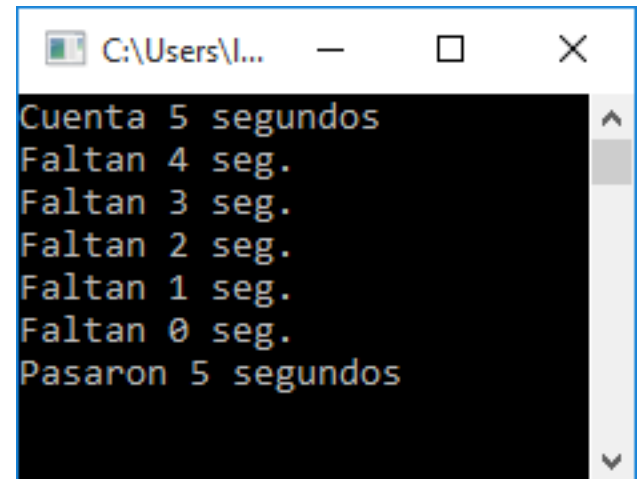
Repaso Delegados - Ejercicio

```
using System;
class Cuenta5
{
    public void Run()
    {
        int contador=5;
        while (contador-- > 0){
            System.Threading.Thread.Sleep(1000);
            Console.WriteLine("Faltan {0} seg.",contador);
        }
    }
}
```

Repaso Delegados - Ejercicio

```
class Program
{
    static void Main()
    {
        Cuenta5 cont=new Cuenta5();
        Console.WriteLine("Cuenta 5 segundos");
        cont.Run();
        Console.WriteLine("Pasaron 5 segundos");
        Console.ReadKey();
    }
}
```

Ejecute y compruebe
su funcionamiento



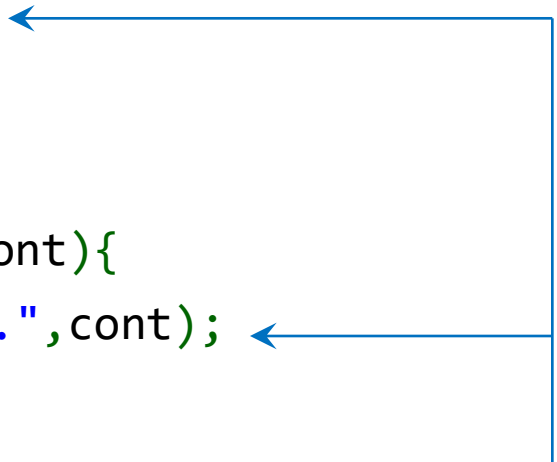
```
C:\Users\I...
Cuenta 5 segundos
Faltan 4 seg.
Faltan 3 seg.
Faltan 2 seg.
Faltan 1 seg.
Faltan 0 seg.
Pasaron 5 segundos
```

Repaso Delegados - Ejercicio

- Vamos a modificar la clase **Cuenta5** para separar del método **Run** el código que se refiere al manejador del evento, es decir aquello que se debe realizar cada vez que se cumple un segundo

Repaso Delegados - Ejercicio

```
class Cuenta5 {  
    public void Run(){  
        int contador=5;  
        while (contador-- > 0){  
            System.Threading.Thread.Sleep(1000);  
            manejadorDelEvento(contador);  
        }  
    }  
    private void manejadorDelEvento(int cont){  
        Console.WriteLine("Faltan {0} seg.",cont);  
    }  
}
```



The diagram consists of two blue arrows. The first arrow originates from the `manejadorDelEvento(contador);` line within the `Run()` method and points to the `manejadorDelEvento(int cont)` method signature. The second arrow originates from the `Console.WriteLine("Faltan {0} seg.",cont);` line within the `manejadorDelEvento` method and points to the same line within the `Run()` method, illustrating the delegation of the event handling logic.

Separamos del método **Run** el código que se refiere al manejo del evento

Repaso Delegados - Ejercicio

- Ahora modificamos la clase **Cuenta5** para llamar al manejador del evento por medio de una variable de tipo delegado. En esta variable se encola el método que usamos de manejador.
- Llamaremos a la variable con un nombre representativo del evento que representa:
`SegCumplido`
- Llamaremos al delegado con un nombre representativo de los métodos que identifica:
`SegCumplidoEventHandler`

Repaso Delegados - Ejercicio

```
delegate void SegCumplidoEventHandler(int cont);
```

```
class Cuenta5 {  
    public SegCumplidoEventHandler SegCumplido;  
    public void Run(){  
        int contador=5;  
        SegCumplido = manejadorDelEvento;  
        while (contador-- > 0){  
            System.Threading.Thread.Sleep(1000);  
            SegCumplido(contador);  
        }  
    }  
    private void manejadorDelEvento(int cont){  
        Console.WriteLine("Faltan {0} seg.",cont);  
    }  
}
```


Repaso Delegados - Ejercicio

- Ya estamos en condiciones de sacar el manejador de la clase **Cuenta5** para que sea otro objeto el que se suscriba al evento con su propio manejador.

```
class Cuenta5 {  
    public SegCumplidoEventHandler SegCumplido;  
    public void Run(){  
        int contador=5;  
        while (contador-- > 0){  
            System.Threading.Thread.Sleep(1000);  
            if (SegCumplido != null) ←  
                SegCumplido(contador);  
        }  
    }  
}
```

Ahora es necesario asegurarse que SegCumplido posee algún método antes de invocarlo

Repaso Delegados - Ejercicio

- Pasamos el manejador a la clase **Program**

```
class Program{  
    static void Main(){  
        Cuenta5 cont=new Cuenta5();  
        cont.SegCumplido=contSegCumplido;  
        Console.WriteLine("Cuenta 5 segundos");  
        cont.Run();  
        Console.WriteLine("Pasaron 5 segundos");  
        Console.ReadKey();  
    }  
    static void contSegCumplido(int cont){  
        Console.WriteLine("Faltan {0} seg.",cont);  
    }  
}
```

Program se suscribe con el manejador **contSegCumplido** al evento que lanza el objeto **cont**

Repaso Delegados - Ejercicio

- Finalmente para cumplir con la convención sobre la nomenclatura y los parámetros involucrados en el mecanismo de eventos se agrega la clase **SegCumplidoEventArgs** y se modifica el programa de la siguiente forma:

Repaso Delegados - Ejercicio

```
delegate void SegCumplidoEventHandler(object sender,  
                                     SegCumplidoEventArgs e);  
  
class SegCumplidoEventArgs:EventArgs{  
    public int Valor{get;set;}  
}  
  
class Cuenta5 {  
    public SegCumplidoEventHandler SegCumplido;  
    public void Run(){  
        int contador=5;  
        while (contador-- > 0){  
            System.Threading.Thread.Sleep(1000);  
            if (SegCumplido != null)  
                SegCumplido(this,new SegCumplidoEventArgs(){Valor=contador});  
        }  
    }  
}
```

Repaso Delegados - Ejercicio

```
using System;
```

```
class Program{  
    static void Main(){  
        Cuenta5 cont=new Cuenta5();  
        cont.SegCumplido=contSegCumplido;  
        Console.WriteLine("Cuenta 5 segundos");  
        cont.Run();  
        Console.WriteLine("Pasaron 5 segundos");  
        Console.ReadKey();  
    }  
    static void contSegCumplido(object sender, SegCumplidoEventArgs e)  
    {  
        Console.WriteLine("Faltan {0} seg.", e.Value);  
    }  
}
```

Eventos

- Un evento es similar a una propiedad donde el campo asociado es un delegado.
- Permiten controlar la forma en que se accede a los campos delegados y dan la posibilidad de asociar código a ejecutar cada vez que se añada o elimine un método de un campo delegado.
- A diferencia de los delegados, a los eventos sólo se le pueden aplicar dos operaciones: **+=** y **-=**.

Eventos

- **Sintaxis:**

```
<modificadores> event <tipoDelegado> <nombreEvento>
{
    add {
        <códigoAdd>
    }
    remove {
        <códigoRevome>
    }
}
```

Eventos - Ejercicio

- Vamos a modificar la clase **Cuenta5** para que en lugar de publicar una variable de tipo Delegado publique un evento. Luego vamos a conseguir que en el momento de suscribirse al evento **SegCumplido** en un objeto **Cuenta5** se dispare automáticamente el método **Run**, que además dejará de ser público.
- Comenzamos definiendo un evento en la clase **Cuenta5** asociado a la variable **SegCumplido**

Eventos - Ejercicio

```
class Cuenta5 {  
    private SegCumplidoEventHandler segCumplido;  
    public event SegCumplidoEventHandler SegCumplido{  
        add{segCumplido += value;}  
        remove{segCumplido -= value; }  
    }  
    public void Run(){  
        int contador=5;  
        while (contador-- > 0){  
            System.Threading.Thread.Sleep(1000);  
            if (segCumplido != null)  
                segCumplido(this,new SegCumplidoEventArgs(){Valor=contador});  
        }  
    }  
}
```

La variable de tipo delegado se hace privada (utilizamos **segCumplido** en minúscula)

Eventos - Ejercicio

```
class Cuenta5 {  
    private SegCumplidoEventHandler segCumplido;  
    public event SegCumplidoEventHandler SegCumplido{  
        add{segCumplido += value;}  
        remove{segCumplido -= value; }  
    }  
}
```

```
public void Run(){  
    int contador=5;  
    while (contador-- > 0){  
        System.Threading.Thread.Sleep(1000);  
        if (segCumplido != null)  
            segCumplido(this,new SegCumplidoEventArgs(){Valor=contador});  
    }  
}
```

Se agrega el evento **SegCumplido** que controlará el acceso al delegado **segCumplido**.

En el método **Main** de la clase **Program** se reemplaza **cont.SegCumplido=contSegCumplido;** por **cont.SegCumplido+=contSegCumplido;**

Eventos - Ejercicio

- Ahora se pretende que en el momento de suscribirse al evento **SegCumplido** de un objeto **Cuenta5** se dispare automáticamente el método **Run**. Además **Run** debe hacerse privado.
- Para resolverlo, simplemente se realiza una invocación al método **Run** desde la sección **add** en la definición del evento **SegCumplido**. **Run** se hace privado y se modifica **Program** que ya no debe invocar a dicho método.

Eventos - Ejercicio

```
class Cuenta5 {  
    private SegCumplidoEventHandler segCumplido;  
    public event SegCumplidoEventHandler SegCumplido{  
        add{segCumplido += value; this.run();}  
        remove{segCumplido -= value; }  
    }  
    private void run(){  
        int contador=5;  
        while (contador-- > 0){  
            System.Threading.Thread.Sleep(1000);  
            if (segCumplido != null)  
                segCumplido(this,new SegCumplidoEventArgs(){Valor=contador});  
        }  
    }  
}
```

Debido a que ahora el método **Run** es privado, por convención lo renombramos con minúscula

Eventos - Ejercicio

La suscripción al evento
provocará el inicio de la cuenta
regresiva

```
class Program{
    static void Main(){
        Cuenta5 cont=new Cuenta5();
        Console.WriteLine("Cuenta 5 segundos");
        cont.SegCumplido+=cont.SegCumplido;
        Console.WriteLine("Pasaron 5 segundos");
        Console.ReadKey();
    }
    static void contSegCumplido(object sender,
                                SegCumplidoEventArgs e){
        Console.WriteLine("Faltan {0} seg.",e.Valor);
    }
}
```