
Programación II

Delegados

- Son objetos que encapsulan una referencia a un metodo de una clase u objeto
- **Para su uso hay que:**

1. El primer paso para usar un delegado es definir un tipo
2. Luego se pueden crear instancias del delegado y relacionarlas a métodos específicos
3. Por último, se puede invocar al método relacionado por medio del delegado.

El problema...

```
static int[] MultiplicadoPorDos(int[] vector)
{
    int[] res = new int[vector.Length];
    for (int i = 0; i < vector.Length; i++)
        res[i] = vector[i] * 2;
    return res;
}

static int[] AlCuadrado(int[] vector)
{
    int[] res = new int[vector.Length];
    for (int i = 0; i < vector.Length; i++)
        res[i] = vector[i] * vector[i];
    return res;
}
```

La solución...

```
static int MultiplicadoPorDos(int numero)
{  return numero * 2;  }

static int AlCuadrado(int x)
{ return x * x; }

static int[] Aplica(int[] vector, DelegadoFuncion f)
{
    int[] res = new int[vector.Length];
    for (int i = 0; i < vector.Length; i++)
        res[i] = f(vector[i]);
    return res;
}
```

Definición de un tipo de delegado

- Se utiliza la palabra clave **delegate**
- Hay que especificar la firma y el tipo de retorno del delegado

```
public delegate int DelegadoFuncion(int x);
```

Creación y uso de un delegado

- Se declaran variables o paramétros como un tipo de datos más
- Se los invoca como si fueran una función

```
static void UnaPrueba()
{
    DelegadoFuncion una_funcion = new DelegadoFuncion(MultiplicadoPorDos);
    Console.WriteLine(una_funcion(4));
    una_funcion = new DelegadoFuncion(AlCuadrado);
    Console.WriteLine(una_funcion(4));
}
```

Eventos

- Las clases y objetos pueden disparar eventos para avisar que algo importante a ocurrido
- El evento es el mecanismo por el cual un objeto le indica a otro que invoque determinado método si ocurre determinado suceso.
- Para indicar qué método debe invocar el emisor del evento, se necesita definir un delegado.
- La manera estandar de pasar datos a un evento en .NET es definir una clase derivada de EventArgs

El ejemplo...

Vamos a definir un evento para que una cuenta bancaria avise cuando se produce una extracción. Nos interesa saber la fecha y monto de la misma.

```
class CuentaBancaria
{
    public decimal Saldo { get; set; }
    public String Titular { get; set; }
    public void Extraer(decimal monto)
    {
        Saldo -= monto;
    }

    public override string ToString()
    { return Titular; }
}
```

Definiendo un derivado de EventArgs

```
class ExtraccionEventArgs: EventArgs
{
    public decimal Monto { get; set; }
    public DateTime Fecha { get; set; }

    public ExtraccionEventArgs(decimal monto, DateTime Fecha)
    {
        this.Monto = monto;
        this.Fecha = Fecha;
    }
}
```

Definición del delegado, el evento y disparo del mismo

```
delegate void ExtraccionDelegate(object sender, ExtraccionEventArgs e);
class CuentaBancaria
{
    public event ExtraccionDelegate OnExtraccion;

    public decimal Saldo { get; set; }
    public String Titular { get; set; }
    public void Extraer(decimal monto)
    {
        if (OnExtraccion != null)
            OnExtraccion(this, new ExtraccionEventArgs(monto, DateTime.Today));
        Saldo -= monto;
    }

    public override string ToString()
    { return Titular; }
}
```

Manejo del evento por otra clase

```
        static void Main(string[] args)
        {
            Prueba();
            CuentaBancaria unaCuenta = new CuentaBancaria() { Titular = "Ariel" };
            unaCuenta.OnExtraccion += new ExtraccionDelegate(Cuenta_OnExtraccion);
            CuentaBancaria otraCuenta = new CuentaBancaria() { Titular = "Seba" };
            otraCuenta.OnExtraccion += new ExtraccionDelegate(Cuenta_OnExtraccion);

            unaCuenta.Extraer(1000);
            otraCuenta.Extraer(2000);
            Console.ReadLine();
        }

        static void Cuenta_OnExtraccion(object sender, ExtraccionEventArgs e)
        {
            Console.WriteLine("El dia {0} se produjo una extracción de {1} $ en la cuenta de {2}",
                              e.Fecha.ToShortDateString(), e.Monto, sender);
        }
```

Ejercicios

1. Revisar la definición del control de usuario para el juego
2. Agregar algún mecanismo de eventos para avisar al control de usuario que debe redibujar alguna casilla
3. Agregar un evento para avisar que el juego terminó
4. Agregar un formulario y probar el juego