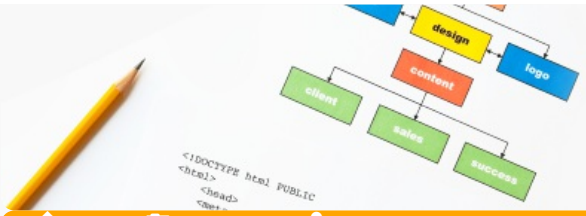

Eventos y Delegados.

11.1. Delegados.

Un delegado en C# es similar a un puntero a una función en C o C++. El uso de un delegado permite al programador encapsular una referencia a un método dentro de un objeto delegado. El objeto delegado se puede pasar entonces al código, el cual puede llamar al método de referencia, sin tener que conocer en tiempo de compilación qué método se debe invocar. A diferencia de los punteros a función de C o C++, los delegados están orientados a objetos, proporcionan seguridad de tipos y son seguros.
Una declaración de delegado define un tipo que encapsula un método con un determinado conjunto de argumentos y un tipo devuelto. Para métodos estáticos, un objeto delegado encapsula el método que se debe llamar. Para métodos de instancia, un objeto delegado encapsula tanto una instancia como un método de la instancia. Si dispone de un objeto delegado y un conjunto apropiado de argumentos, puede invocar el delegado con los argumentos.
Una propiedad interesante y útil de un delegado es que no necesita conocer la clase del objeto al que hace referencia. Cualquier objeto es adecuado; lo único que importa es que los tipos de los argumentos del método y el tipo devuelto coincidan con los del delegado. Esto hace que los delegados sean perfectos para una invocación "anónima".
Ejemplo

```
using System;

namespace Ejemplos
{
   class Program
   {
      public delegate void MyDelegate(string s);

      public static void Hello(string s)
      {
         Console.WriteLine("  Hello, {0}!", s);
      }

      public static void Goodbye(string s)
      {
         Console.WriteLine("  Goodbye, {0}!", s);
      }

      public static void Main(String[] args)
      {
         MyDelegate a, b, c, d;

         // Crea el delegado referenciando al metodo Hello
         a = new MyDelegate(Hello);
         // Crea el delegado referenciando al metodo Goodbye, se puede evitar crear la instancia del delegado y solo asigar el metodo

         b = Goodbye;
         // Se le agrega el delagado a y b a c, ahora tiene referenciado el metodo Hello y Goodbye
         c = a + b;
         // c se le remueve a y se asigne a d, ahora tiene referenciado el metodo Goodbye
         d = c - a;

         Console.WriteLine("Invoking delegate a:");
         a("A");
         Console.WriteLine("Invoking delegate b:");
         b("B");
         Console.WriteLine("Invoking delegate c:");
         c("C");
         Console.WriteLine("Invoking delegate d:");
         d("D");
         Console.ReadLine();

      }
   }
}
```

11.2. Eventos.

Un evento en C# es el modo que tiene una clase de proporcionar notificaciones a los clientes de la clase cuando ocurre algo digno de reseñar en un objeto. El uso más habitual para los eventos se produce en las interfaces gráficas; normalmente, las clases que

representan controles de la interfaz disponen de eventos que se notifican cuando el usuario hace algo con el control (por ejemplo, hacer clic en un botón).
Los eventos, sin embargo, no sólo se utilizan para interfaces gráficas. Los eventos proporcionan un medio apropiado para que los objetos puedan señalizar cambios de estado que pueden resultar útiles para los clientes de ese objeto. Los eventos constituyen unidades importantes para crear clases que se pueden reutilizar en diferentes programas. Los eventos se declaran mediante delegados. Recuerde que un objeto delegado encapsula un método de modo que se pueda llamar de forma anónima. Un evento es el modo que tiene una clase de permitir a los clientes proporcionar delegados a los métodos para llamarlos cuando se produce el evento. Cuando ocurre el evento, se llama a los delegados que proporcionan los clientes para el evento.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Ejemplos
{
   class Program
   {
      static void Main(string[] args)
      {
         Evento e = new Evento();
         //Se crea el evento y asigna el metodo estatico al evento,
         e.evento += OnEvento;
         //hace lo mismo que la primera pero con el EventHandler, ambas funcionan igual
         e.evento += new Evento.EventHandler(OnEvento);

         e.OnEvento("HolaMundo");
      }

      public static void OnEvento(String s)
      {
         Console.WriteLine(s);
      }

      public class Evento
      {
         public delegate void EventHandler(String s);

         public event EventHandler evento;

         public void OnEvento(String s)
         {
            if (evento != null)
            evento(s);
         }
      }
   }
}

/*Resultado*/
HolaMundo
HolaMundo
```