Seminario de Lenguajes (.NET)

Práctica 8a

Importante: El ejercicio 3 de la práctica 8a y los ejercicios 1 y 2 de la práctica 8b deben entregarse para su evaluación. La fecha de entrega se publica en la plataforma WebUNLP.

1) Cree una nueva solución en el SharpDevelop y codifique el siguiente programa:

```
using System;

delegate void Notificacion();

class Program
{
	public static void Main()
	{
		Objeto o=new Objeto();
		o.OnFinTrabajo += new Notificacion(F);
		o.OnFinTrabajo += new Notificacion(G);
		o.Trabajar();
		Console.ReadKey(true);
	}
	private static void F(){
		Console.WriteLine("ejecutando F");
	}
	private static void G(){
		Console.WriteLine("ejecutando G");
	}
}

class Objeto{
	private int cont=0;
	private Notificacion onFinTrabajo;
	public event Notificacion OnFinTrabajo{
		add {
			if(cont==0){
				onFinTrabajo += value; cont++;
			}
		}
		remove{
			if(cont>0){
				onFinTrabajo -= value; cont--;
			}
		}
	}
	public void Trabajar(){
		Console.WriteLine("Objeto trabajando");
		onFinTrabajo();
	}
}
```

a) Ejecute paso a paso el programa y observe cuidadosamente su funcionamiento.
b) ¿Qué salida produce por Consola?
c) Modifique el código para cumplir con la convención vista en la teoría respecto de los nombres que deben llevar los eventos, los delegados y los parámetros de los manejadores de eventos (`sender` de tipo `object` y `e` de tipo `EventArgs`)

2) Dado el siguiente programa:

```
using System;

class TicEventArgs:EventArgs
{
    private DateTime hora;
    public DateTime Hora{
        get{return hora;}
        set{hora=value;}
    }
    public TicEventArgs(DateTime hora){
        this.hora=hora;
    }
}

delegate void TicEventHandler(object sender, TicEventArgs e);

class program{
    static int cont=0;
    static Clock reloj=new Clock();
    static void Main(){
        reloj.Tic+=new TicEventHandler(Tic);
    }
    private static void Tic(object sender, TicEventArgs e){
        Console.WriteLine(e.Hora);
        cont++;
        if(cont==10) reloj.Detener();
    }
}

class Clock{
    private bool detener;
    private TicEventHandler tic;
    public event TicEventHandler Tic{
        add{tic += value; this.run();}
        remove{tic -= value;}
    }
    private void run(){
        DateTime hora=DateTime.Parse("1/1/2000");
        DateTime horaAux=DateTime.Now;
        detener=false;
        while (!detener){
            if(hora.Second != horaAux.Second){
                hora=horaAux;
                if(tic!=null) tic(this,new TicEventArgs(hora));
            }
            horaAux=DateTime.Now;
        }
    }
    public void Detener(){
        detener=true;
    }
}
```

Modifíquelo para que no sea posible asignar más de tres método al evento `Tic` de la clase `Clock`. Además el método `Detener()` de la clase `Clock` debe eliminar todos los métodos que se hayan agregado al delegado `tic`.

3) Defina el delegado `PrecioCambiadoEventHandler`, la clase `PrecioCambiadoEventArgs` y la clase `Articulo` de tal forma que el programa siguiente genere la salida por consola que se muestra en la figura 1

```
using System;
class Program
{
   public static void Main(string[] args) {
      Articulo a=new Articulo();
      a.PrecioCambiado += new PrecioCambiadoEventHandler(precioCambiado);
      a.Codigo = 1;
      a.Precio = 10;
      a.Precio = 12;
      a.Precio = 12;
      a.Precio = 14;
      Console.ReadKey(true);
   }
   public static void precioCambiado(object sender, PrecioCambiadoEventArgs e){
      string texto="Artículo {0} valía {1} y ahora vale {2}";
      Console.WriteLine(texto,e.Codigo,e.PrecioAnterior,e.PrecioNuevo);
   }
}
```

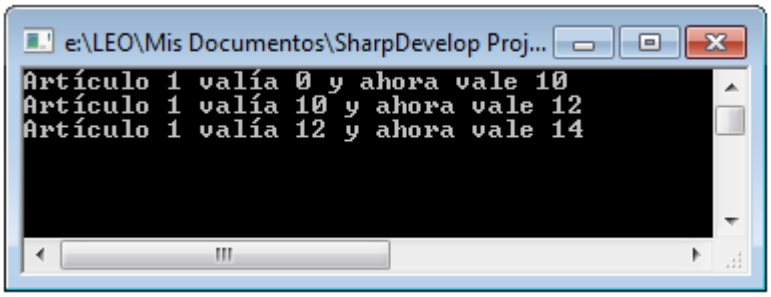

Figura 1

La clase `Articulo` cuenta con las propiedades de lectura/escritura `Codigo` y `Precio`. Además posee el evento `PrecioCambiado` que se produce cuando se cambia el valor de la propiedad `Precio` (observe que si se asigna el mismo valor el evento no se produce).

Seminario de Lenguajes (.NET)

Práctica 8b

Ejercicio 1. Desarrolle una aplicación que permita modificar el tamaño de un label. El mismo se encuentra ubicado dentro de un panel y puede ser "estirado" una cierta cantidad de pasos. La interfaz a utilizar es la indicada en la Figura 1.

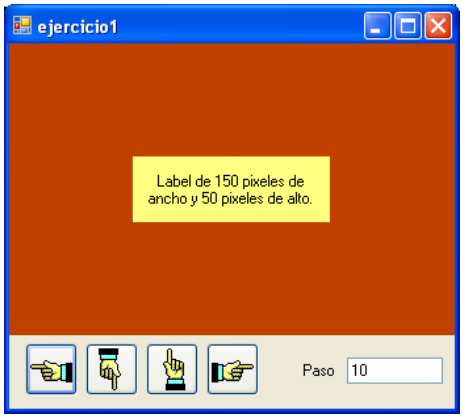

Figura 1

Los botones inferiores permiten realizar el estiramiento antes mencionado.
Si ejecuta ***Ejercicio1.Exe*** verá que el label visualiza permanentemente su tamaño actual. Además los botones se deshabilitan cuando el label ya no puede estirarse en esa dirección. Tenga en cuenta que esta condición puede variar en caso de modificarse el tamaño del formulario.
Dando doble click sobre el label se ubica automáticamente en el centro del formulario volviendo a su tamaño inicial.

Ejercicio 2. Programe una Aplicación Windows con la interface descripta la figura 2. Utilice a modo de referencia el programa ***Ejercicio2.exe*** provisto por la cátedra.

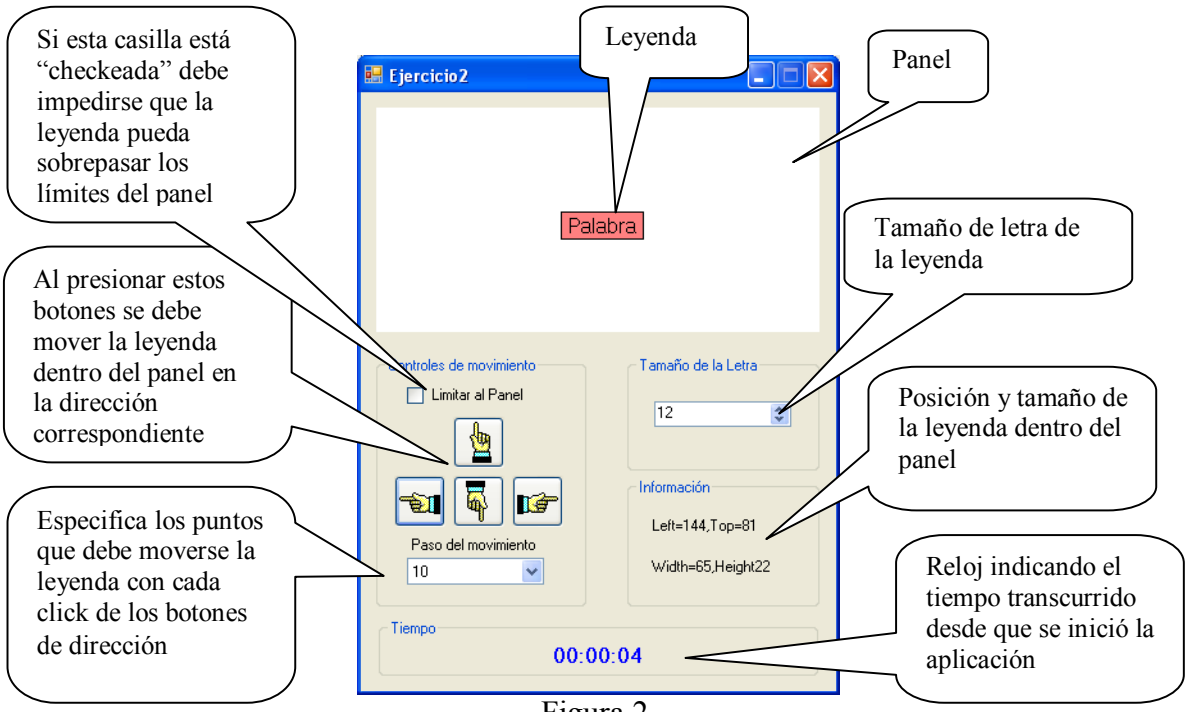

Figura 2

Debe cumplir la siguiente funcionalidad:
Cada vez que se presione alguno de los cuatro botones del grupo "Controles de movimiento" debe actualizarse la visualización del formulario moviendo la leyenda dentro del panel superior. El texto de la misma debe cambiar reflejando cuál ha sido el botón presionado (Arriba, Abajo, Izquierda o

Derecha), debe desplazarse en el sentido correspondiente la cantidad de puntos indicada en el ComboBox “Paso del movimiento”. Además debe actualizarse la información relativa a la posición y tamaño de la leyenda en el grupo “Información”. Si la casilla de verificación “Limitar al panel” se encuentra “tildada” debe mantenerse la leyenda por completo dentro de los confines del panel.
También debe actualizarse la visualización del formulario cada vez que cambie el valor del control que indica el “Tamaño de la letra”. Deben mantenerse las restricciones indicadas en relación a la casilla de verificación, salvo el caso en que el tamaño de la letra sea excesivamente grande impidiendo que entre por completo dentro del panel. Los valores válidos para el tamaño de letra deben cubrir el rango desde 8 hasta 100. Debe impedirse que se puedan especificar valores fuera de este rango. Al iniciarse el formulario debe estar seleccionado el valor 12.
El ComboBox “Paso del movimiento” debe cargarse con valores desde 1 hasta 100. No debe permitirse que el usuario pueda escribir libremente sobre el combo, limitando la elección sólo a los valores listados. Al abrirse el formulario debe estar seleccionado el valor 10.
La ventana del formulario no puede maximizarse ni cambiar de tamaño. El título de la misma debe ser “Ejercicio2”
Haciendo click sobre la leyenda debe abrirse un cuadro de diálogo que permita al usuario cambiar el color del fondo de la leyenda (Figura 3).

Figura 3

Tenga en cuenta que el usuario puede decidir no realizar ningún cambio cancelando en lugar de aceptar. El color de la leyenda debe cambiar sólo si el usuario elige “Aceptar”.
El reloj que indica el tiempo transcurrido desde que se inició la aplicación debe actualizarse cada segundo (utilice un Timer para ello).

Ejercicio 3.
Desarrolle la aplicación que muestre en un formulario un botón con la leyenda “Haceme clic” y un checkBox con la leyenda “Quieto !!” Consulte la documentación de la plataforma .NET e investigue cómo se debe utilizar la clase Random para generar números aleatorios. Si el checkBox no está tildado el botón se moverá aleatoriamente a cualquier parte del formulario ante el intento de ser cliqueado por el usuario. Por el contrario si el checkBox está tildado el botón permanecerá inmóvil pudiendo cliquearse normalmente

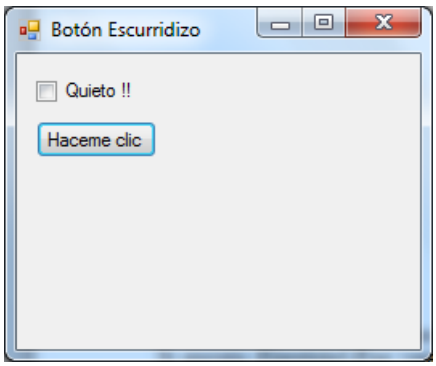

Figura 4

Al hacer clic sobre el botón debe mostrarse el mensaje siguiente:

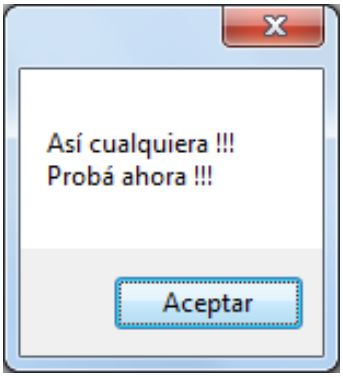

Figura 5

Cuando se cierra este MessageBox el checkBox se destilda automáticamente. Utilice a modo de referencia el programa ***Ejercicio3.exe*** provisto por la cátedra.