|  |
| --- |
| INSTITUTO TECNOLÓGICO DE ESTUDIOS SUPERIORES DE LA REGIÓN CARBONÍFERA |
| Manual de Prácticas de Estructura de Datos |
| Por: Ing. Héctor Javier Padilla Lara |
| Ejercicios propuestos para la materia de Estructura de Datos para Sistemas Computacionales utilizando el lenguaje de programación C#.  **Coordinación de Sistemas Computacionales**  Diciembre de 2010 |
|  |
|  |

|  |
| --- |
|  |

Tabla de contenido

[Unidad 1. 4](#_Toc295767599)

[Unidad 2. 4](#_Toc295767600)

[Ejercicio 1. Lista de Pendientes 4](#_Toc295767601)

[Clase FormPendientes 4](#_Toc295767602)

[Clase ListaPendientes 5](#_Toc295767603)

[Ejercicio 2. Agenda Telefónica 7](#_Toc295767604)

[Formulario Agenda 7](#_Toc295767605)

[Formulario NuevoContacto 8](#_Toc295767606)

[Clase Contacto 9](#_Toc295767607)

[Clase AgendaTelefonica 10](#_Toc295767608)

[Ejercicio 3. Lista de reproducción 12](#_Toc295767609)

[Formulario Lista de Reproducción 12](#_Toc295767610)

[Formulario Nueva Canción 12](#_Toc295767611)

[Formulario Nuevo Video 13](#_Toc295767612)

[Clase Cancion 13](#_Toc295767613)

[Clase Video 14](#_Toc295767614)

[Clase ListaMedios 15](#_Toc295767615)

[Ejercicio 4. Números Cayendo 18](#_Toc295767616)

[Formulario Números Cayendo 18](#_Toc295767617)

[Clase NumerosCayendo 20](#_Toc295767618)

[Unidad 3. 22](#_Toc295767619)

[Ejercicio 1. Alumnos por especialidad 22](#_Toc295767620)

[Formulario Alumnos 22](#_Toc295767621)

[Formulario Nuevo Alumno 24](#_Toc295767622)

[Clase Alumno 25](#_Toc295767623)

[Clase Grupo 25](#_Toc295767624)

[Ejercicio 2. Palabras del diccionario 27](#_Toc295767625)

[Formulario Diccionario 27](#_Toc295767626)

[Formulario Nueva Palabra 29](#_Toc295767627)

[Formulario Información Palabra 30](#_Toc295767628)

[Clase Palabra 30](#_Toc295767629)

[Clase Diccionario 31](#_Toc295767630)

[Ejercicio 3. Retícula 34](#_Toc295767631)

[Formulario Materias 34](#_Toc295767632)

[Formulario Nueva Materia 35](#_Toc295767633)

[Clase Materia 36](#_Toc295767634)

[Clase Reticula 37](#_Toc295767635)

[Unidad 4. 39](#_Toc295767636)

[Ejercicio 1. Calendario de Actividades 39](#_Toc295767637)

[Formulario Calendario de actividades 39](#_Toc295767638)

[Formulario Agregar actividades 41](#_Toc295767639)

[Unidad 5. 42](#_Toc295767640)

[Ejercicio 1. Citas del Dentista 42](#_Toc295767641)

[Formulario Calendario de citas 42](#_Toc295767642)

[Formulario Nueva cita 43](#_Toc295767643)

[Clase Consultas 44](#_Toc295767644)

[Ejercicio 2. Laboratorio de Computo 46](#_Toc295767645)

[Formulario Reservados 46](#_Toc295767646)

[Clase Alumno 46](#_Toc295767647)

[Clase Alumnos 47](#_Toc295767648)

[Clase Reservaciones 48](#_Toc295767649)

[Ejercicio 3. Horario del Docente 49](#_Toc295767650)

# Unidad 1.

En esta unidad el alumno aprende los conceptos básicos de estructuras de datos y de colecciónes en C#.

No tiene prácticas. Se trata de una unidad teórica.

# Unidad 2.

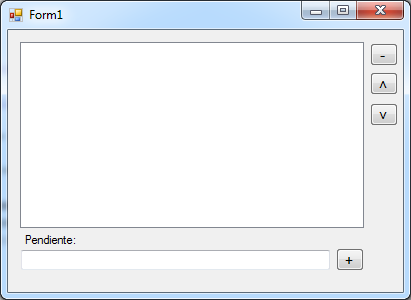
En esta unidad el alumno aprenderá la utilización de estructuras de datos lineales para resolver problemas.

Utilizará las colecciones propias de C#, ArrayList, Lista, Stack y Queue que se corresponde a listas, pilas y colas.

## Ejercicio 1. Lista de Pendientes

Crear un programa en C# que permita mantener al usuario una lista de pendientes. El programa permitirá agregar y eliminar pendientes. Los pendientes no deben repetirse. La lista de pendientes podrá reordenarse.

### Clase FormPendientes



public partial class FormPendientes : Form

{

public FormPendientes ()

{

InitializeComponent();

}

ListaPendientes lista = new ListaPendientes();

private void btnAgregar\_Click(object sender, EventArgs e)

{

lista.Agregar(txtPendiente.Text);

}

private void Form1\_Load(object sender, EventArgs e)

{

lista.Actualizado += new ListaPendientes.Actualizar(lista\_Actualizado);

}

void lista\_Actualizado()

{

lstPendientes.DataSource = null;

lstPendientes.DataSource = lista.Lista;

}

private void btnEliminar\_Click(object sender, EventArgs e)

{

if (lstPendientes.SelectedIndex != -1)

{

lista.Eliminar((string)lstPendientes.SelectedItem);

}

}

private void btnSubir\_Click(object sender, EventArgs e)

{

if (lstPendientes.SelectedIndex != -1)

{

lista.Subir((string)lstPendientes.SelectedItem);

}

}

private void btnBajar\_Click(object sender, EventArgs e)

{

if (lstPendientes.SelectedIndex != -1)

{

lista.Bajar((string)lstPendientes.SelectedItem);

}

}

}

### Clase ListaPendientes

public class ListaPendientes

{

private ArrayList lista;

public ArrayList Lista

{

get { return lista; }

}

public delegate void Actualizar();

public event Actualizar Actualizado;

private void Lanzar()

{

if (Actualizado != null)

Actualizado();

}

public void Agregar(string pendiente)

{

if (lista == null)

lista = new ArrayList();

if (lista.Contains(pendiente))

throw new ArgumentException("El pendiente especificado ya existe.");

else

{

lista.Add(pendiente);

Lanzar();

}

}

public void Eliminar(string pendiente)

{

if (lista != null && lista.Contains(pendiente))

{

lista.Remove(pendiente);

Lanzar();

}

}

public void Subir(string pendiente)

{

if (lista != null)

{

int indice = lista.IndexOf(pendiente);

if (indice > 0 && indice != -1)

{

lista.Remove(pendiente);

lista.Insert(indice - 1, pendiente);

Lanzar();

}

}

}

public void Bajar(string pendiente)

{

if (lista != null)

{

int indice = lista.IndexOf(pendiente);

if (indice != lista.Count - 1 && indice != -1)

{

lista.Remove(pendiente);

lista.Insert(indice + 1, pendiente);

Lanzar();

}

}

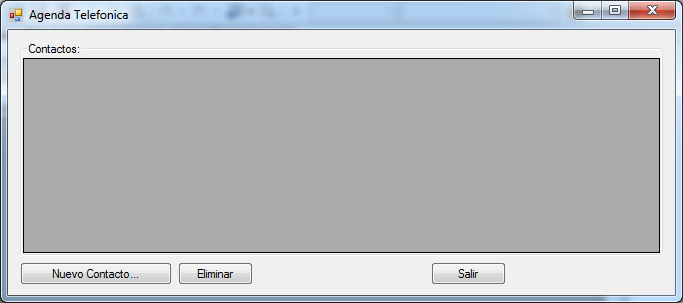
}

}

## Ejercicio 2. Agenda Telefónica

Crear un programa en C# que permita llevar una agenda telefónica, cada contacto de la agenda tendrá como datos su nombre, dirección, teléfono, correo electrónico y fecha de nacimiento.

### Formulario Agenda



public partial class frmAgenda : Form

{

public frmAgenda()

{

InitializeComponent();

}

AgendaTelefonica agenda = new AgendaTelefonica();

private void frmAgenda\_Load(object sender, EventArgs e)

{

agenda.Actualizado += new AgendaTelefonica.Actualizar(agenda\_Actualizado);

}

void agenda\_Actualizado()

{

dgvContactos.DataSource = null;

dgvContactos.DataSource = agenda.ListaContactos;

}

private void btnContacto\_Click(object sender, EventArgs e)

{

frmContacto forma2 = new frmContacto();

forma2.MiAgenda = agenda;

forma2.ShowDialog();

}

private void btnSalir\_Click(object sender, EventArgs e)

{

Close();

}

private void btnEliminar\_Click(object sender, EventArgs e)

{

if (dgvContactos.SelectedRows.Count == 1)

{

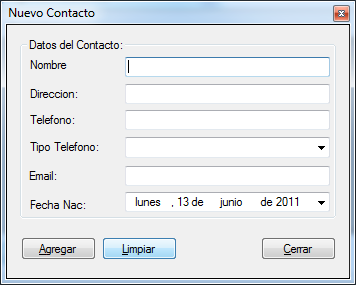
agenda.EliminarContacto((Contacto)dgvContactos.SelectedRows[0].DataBoundItem);

}

}

}

### Formulario NuevoContacto



public partial class frmContacto : Form

{

private AgendaTelefonica agenda;

public AgendaTelefonica MiAgenda

{

get { return agenda; }

set { agenda = value; }

}

public frmContacto()

{

InitializeComponent();

}

private void btnAgregar\_Click(object sender, EventArgs e)

{

agenda.NuevoContacto(txtNombre.Text, txtDireccion.Text, ulong.Parse(txtTelefono.Text),

(string)cmbTipo.SelectedItem,

txtEmail.Text,

dtpFecha.Value);

btnLimpiar.PerformClick();

txtNombre.Focus();

}

private void btnLimpiar\_Click(object sender, EventArgs e)

{

foreach (Control c in grbDatos.Controls)

{

if (c is TextBox)

{

((TextBox)c).Clear();

}

}

cmbTipo.SelectedIndex = -1;

dtpFecha.Value = DateTime.Now;

}

private void btnCerrar\_Click(object sender, EventArgs e)

{

Close();

}

}

### Clase Contacto

public class Contacto

{

private DateTime fecha;

private string email;

private string tipo;

private ulong telefono;

private string direccion;

private string nombre;

public string Nombre

{

get { return nombre; }

set { nombre = value; }

}

public string Direccion

{

get { return direccion; }

set { direccion = value; }

}

public ulong Telefono

{

get { return telefono; }

set { telefono = value; }

}

public string TipoTelefono

{

get { return tipo; }

set { tipo = value; }

}

public string EMail

{

get { return email; }

set { email = value; }

}

public DateTime FechaNacimiento

{

get { return fecha; }

set { fecha = value; }

}

}

### Clase AgendaTelefonica

public class AgendaTelefonica

{

private ArrayList lista;

public ArrayList ListaContactos

{

get { return lista; }

}

public delegate void Actualizar();

public event Actualizar Actualizado;

public void NuevoContacto(string nombre, string direccion, ulong telefono, string tipotelefono, string email, DateTime fechanacimiento)

{

if (lista == null)

lista = new ArrayList();

Contacto c = new Contacto();

c.Nombre = nombre;

c.Direccion = direccion;

c.Telefono = telefono;

c.TipoTelefono = tipotelefono;

c.EMail = email;

c.FechaNacimiento = fechanacimiento;

lista.Add(c);

Lanzar();

}

public void EliminarContacto(Contacto c)

{

if (lista != null && lista.Contains(c))

{

lista.Remove(c);

Lanzar();

}

}

private void Lanzar()

{

if (Actualizado != null)

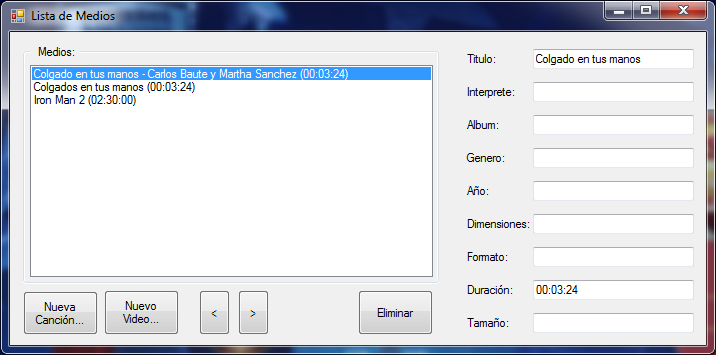
Actualizado();

}

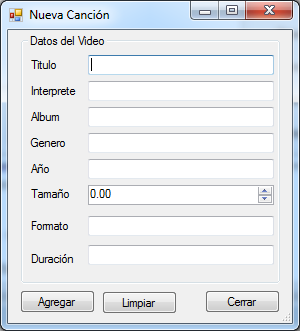
}

## Ejercicio 3. Lista de reproducción

### Formulario Lista de Reproducción



### Formulario Nueva Canción



### Formulario Nuevo Video

### 

### Clase Cancion

[Serializable]

public class Cancion

{

private string titulo;

public string Titulo

{

get { return titulo; }

set { titulo = value; }

}

private string album;

public string Album

{

get { return album; }

set { album = value; }

}

private string interprete;

public string Interprete

{

get { return interprete; }

set { interprete = value; }

}

private string genero;

public string Genero

{

get { return genero; }

set { genero = value; }

}

private string formato;

public string Formato

{

get { return formato; }

set { formato = value; }

}

private ushort año;

public ushort Año

{

get { return año; }

set { año = value; }

}

private string duracion;

public string Duracion

{

get { return duracion; }

set { duracion = value; }

}

private double tamaño;

public double Tamaño

{

get { return tamaño; }

set { tamaño = value; }

}

public override string ToString()

{

return string.Format("{0} - {1} ({2})",titulo,interprete,duracion);

}

}

### Clase Video

[Serializable]

public class Video

{

private string titulo;

public string Titulo

{

get { return titulo; }

set { titulo = value; }

}

private ushort ancho;

public ushort Ancho

{

get { return ancho; }

set { ancho = value; }

}

private ushort alto;

public ushort Alto

{

get { return alto; }

set { alto = value; }

}

private string formato;

public string Formato

{

get { return formato; }

set { formato = value; }

}

private string duracion;

public string Duracion

{

get { return duracion; }

set { duracion = value; }

}

private double tamaño;

public double Tamaño

{

get { return tamaño; }

set { tamaño = value; }

}

public override string ToString()

{

return string.Format("{0} ({1})", titulo, duracion);

}

}

### Clase ListaMedios

public class ListaMedios

{

private ArrayList lista;

public ArrayList Medios

{

get { return lista; }

}

public delegate void Actualizar();

public event Actualizar Actualizado;

private void Lanzar()

{

if (Actualizado != null)

Actualizado();

}

public void NuevaCancion(string Titulo, string Formato, string Interprete, string Album, string Duracion, ushort Año, string Genero, double Tamaño)

{

if (lista == null)

lista = new ArrayList();

Cancion c = new Cancion();

c.Titulo = Titulo;

c.Formato = Formato;

c.Interprete = Interprete;

c.Album = Album;

c.Año = Año;

c.Duracion = Duracion;

c.Genero = Genero;

c.Tamaño = Tamaño;

lista.Add(c);

Lanzar();

}

public void NuevoVideo(string Titulo, ushort Alto, ushort Ancho, string Formato, double Tamaño, string Duracion)

{

if (lista == null)

lista = new ArrayList();

Video v = new Video();

v.Alto = Alto;

v.Ancho = Ancho;

v.Duracion = Duracion;

v.Formato = Formato;

v.Tamaño = Tamaño;

v.Titulo = Titulo;

lista.Add(v);

Lanzar();

}

public void Guardar()

{

FileStream fs = new FileStream("medios.data", FileMode.Create);

BinaryFormatter bf = new BinaryFormatter();

if (lista != null)

bf.Serialize(fs, lista);

fs.Close();

}

public void Cargar()

{

if (File.Exists("medios.data"))

{

FileStream fs = new FileStream("medios.data", FileMode.Open);

BinaryFormatter bf = new BinaryFormatter();

try

{

lista = (ArrayList)bf.Deserialize(fs);

}

catch { }

fs.Close();

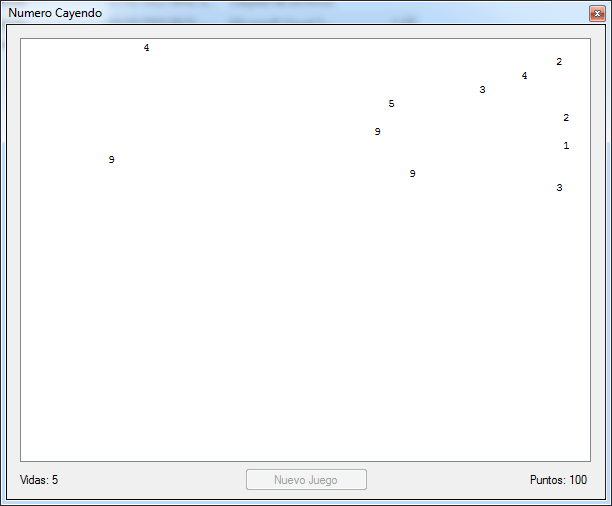
Lanzar();

}

}

## Ejercicio 4. Números Cayendo

### Formulario Números Cayendo



public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

NumerosCayendo juego = new NumerosCayendo();

private void Form1\_Load(object sender, EventArgs e)

{

juego.JuegoIniciado += new NumerosCayendo.DelegadoJuego(juego\_JuegoIniciado);

juego.JuegoFinalizado += new NumerosCayendo.DelegadoJuego(juego\_JuegoFinalizado);

juego.JugadorEquivocado += new NumerosCayendo.DelegadoJuego(juego\_JugadorEquivocado);

juego.NumeroGenerado += new NumerosCayendo.AlGenerarNumero(juego\_NumeroGenerado);

juego.JugadorAcerto += new NumerosCayendo.DelegadoJuego(juego\_JugadorAcerto);

lstNumeros.Items.Clear();

}

SoundPlayer acerto = new SoundPlayer("Coin.wav");

SoundPlayer equivocado = new SoundPlayer("Die.wav");

void juego\_JugadorAcerto()

{

acerto.Play();

timer1.Interval = juego.Velocidad;

//this.Text = timer1.Interval.ToString();

lblPuntos.Text = juego.Puntos.ToString("Puntos: 0");

lstNumeros.Items.RemoveAt(lstNumeros.Items.Count - 1);

}

Random r = new Random();

void juego\_NumeroGenerado(byte num)

{

lstNumeros.Items.Insert(0, num.ToString().PadLeft(r.Next(0, 80), ' '));

}

void juego\_JugadorEquivocado()

{

lblVidas.Text = juego.Vidas.ToString("Vidas: 0");

equivocado.Play();

}

void juego\_JuegoFinalizado()

{

timer1.Stop();

SoundPlayer play = new SoundPlayer("Death.wav");

play.Play();

btnNuevo.Enabled = true;

MessageBox.Show("Juego Terminado", "Información", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

}

void juego\_JuegoIniciado()

{

SoundPlayer play = new SoundPlayer("Start Stage.wav");

play.PlaySync();

btnNuevo.Enabled = false;

timer1.Interval = juego.Velocidad;

lblVidas.Text = juego.Vidas.ToString("Vidas: 0");

timer1.Start();

}

private void btnNuevo\_Click(object sender, EventArgs e)

{

lstNumeros.Items.Clear();

juego.NuevoJuego();

}

private void timer1\_Tick(object sender, EventArgs e)

{

juego.GenerarNumero();

}

private void Form1\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!btnNuevo.Enabled)

if (e.KeyChar >= '0' && e.KeyChar <= '9')

{

juego.Intentar(byte.Parse(e.KeyChar.ToString()));

}

}

}

### Clase NumerosCayendo

public class NumerosCayendo

{

Queue<byte> cola;

private uint punto;

public uint Puntos

{

get { return punto; }

set { punto = value; }

}

private byte vidas;

public byte Vidas

{

get { return vidas; }

set { vidas = value; }

}

private int velocidad;

public int Velocidad

{

get { return velocidad; }

set { velocidad = value; }

}

public delegate void DelegadoJuego();

public event DelegadoJuego JuegoIniciado;

public event DelegadoJuego JuegoFinalizado;

public event DelegadoJuego JugadorEquivocado;

public event DelegadoJuego JugadorAcerto;

public delegate void AlGenerarNumero(byte num);

public event AlGenerarNumero NumeroGenerado;

Random r = new Random();

public void NuevoJuego()

{

vidas = 5;

punto = 0;

velocidad = 1000;

cola = new Queue<byte>();

GenerarNumero();

if (JuegoIniciado != null)

JuegoIniciado();

}

public void GenerarNumero()

{

if (cola.Count < 30)

{

byte num = (byte)r.Next(0, 10);

cola.Enqueue(num);

if (NumeroGenerado != null)

NumeroGenerado(num);

}

else

{

FinJuego();

}

}

public void Intentar(byte num)

{

if (num == cola.Peek())

{

cola.Dequeue();

punto += 100;

if (velocidad - 10 >= 1)

velocidad -= 10;

if (JugadorAcerto != null)

JugadorAcerto();

}

else

{

vidas--;

if (vidas == 0)

{

if (JuegoFinalizado != null)

JuegoFinalizado();

}

else

if (JugadorEquivocado != null)

JugadorEquivocado();

}

}

public void FinJuego()

{

if (JuegoFinalizado != null)

JuegoFinalizado();

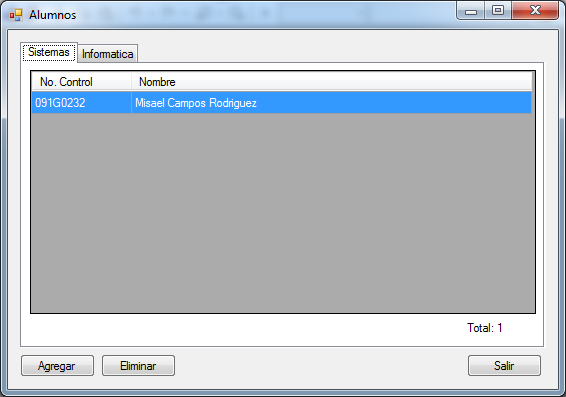
}

}

# Unidad 3.

## Ejercicio 1. Alumnos por especialidad

### Formulario Alumnos



public partial class frmAlumnos : Form

{

public frmAlumnos()

{

InitializeComponent();

}

Grupo grupo = new Grupo();

private void btnAgregar\_Click(object sender, EventArgs e)

{

frmNuevoAlumno nuevo = new frmNuevoAlumno();

nuevo.Grupo = grupo;

nuevo.ShowDialog();

}

private void frmAlumnos\_Load(object sender, EventArgs e)

{

grupo.Actualizado += new Grupo.Actualizar(grupo\_Actualizado);

dgvInformatica.AutoGenerateColumns = false;

dgvSistemas.AutoGenerateColumns = false;

grupo.Cargar();

}

void grupo\_Actualizado(string s)

{

if (s == "Ingenieria en Sistemas")

{

dgvSistemas.DataSource = null;

dgvSistemas.DataSource = grupo.Sistemas;

lblSistemas.Text = grupo.TotalSistemas.ToString("Total: 0");

}

else

{

dgvInformatica.DataSource = null;

dgvInformatica.DataSource = grupo.Informatica;

lblInformatica.Text = grupo.TotalInformatica.ToString("Total: 0");

}

}

private void btnSalir\_Click(object sender, EventArgs e)

{

Close();

}

private void btnEliminar\_Click(object sender, EventArgs e)

{

if (tabControl1.SelectedIndex == 0)

{

if (dgvSistemas.SelectedRows.Count == 1)

grupo.Eliminar((Alumno)dgvSistemas.SelectedRows[0].DataBoundItem);

}

else

{

if (dgvInformatica.SelectedRows.Count == 1)

grupo.Eliminar((Alumno)dgvInformatica.SelectedRows[0].DataBoundItem);

}

}

private void frmAlumnos\_FormClosing(object sender, FormClosingEventArgs e)

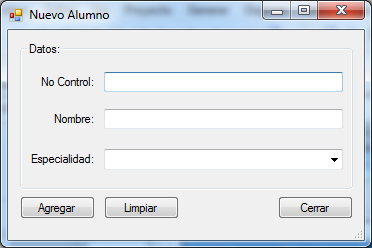
{

grupo.Guardar();

}

}

### Formulario Nuevo Alumno



public partial class frmNuevoAlumno : Form

{

public frmNuevoAlumno()

{

InitializeComponent();

}

private Grupo grupo;

public Grupo Grupo

{

get { return grupo; }

set { grupo = value; }

}

private void btnCerrar\_Click(object sender, EventArgs e)

{

Close();

}

private void btnLimpiar\_Click(object sender, EventArgs e)

{

txtControl.Clear();

txtNombre.Clear();

cmbEspecialidad.SelectedIndex = -1;

txtControl.Focus();

}

private void btnAgregar\_Click(object sender, EventArgs e)

{

try

{

grupo.Agregar(txtControl.Text, txtNombre.Text, (string)cmbEspecialidad.SelectedItem);

btnLimpiar.PerformClick();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

### Clase Alumno

[Serializable]

public class Alumno

{

private string control;

public string NoControl

{

get { return control; }

set { control = value; }

}

private string nombre;

public string Nombre

{

get { return nombre; }

set { nombre = value; }

}

private string especialidad;

public string Especialidad

{

get { return especialidad; }

set

{

if (value != "Licenciatura en Informatica" && value != "Ingenieria en Sistemas")

throw new ApplicationException("Solo manejamos las especialidades de: \n-Licenciatura en Informatica \n-Ingenieria en Sistemas");

especialidad = value;

}

}

}

### Clase Grupo

[Serializable]

public class Grupo

{

private List<Alumno> grupo;

public List<Alumno> Group

{

get { return grupo; }

}

bool Informatico(Alumno a) { return a.Especialidad == "Licenciatura en Informatica"; }

bool Sistematico(Alumno a) { return a.Especialidad == "Ingenieria en Sistemas"; }

public delegate void Actualizar(string s);

public event Actualizar Actualizado;

private void Lanzar(string s)

{

if (Actualizado != null)

Actualizado(s);

}

public List<Alumno> Informatica

{

get { return grupo.FindAll(Informatico); }

}

public int TotalInformatica

{

get { return Informatica.Count; }

}

public List<Alumno> Sistemas

{

get { return grupo.FindAll(Sistematico); }

}

public int TotalSistemas

{

get { return Sistemas.Count; }

}

public void Agregar(string control, string nombre, string especialidad)

{

if (grupo == null)

grupo = new List<Alumno>();

Alumno a = new Alumno();

a.NoControl = control;

a.Nombre = nombre;

a.Especialidad = especialidad;

grupo.Add(a);

Lanzar(especialidad);

}

public void Eliminar(Alumno a)

{

if (grupo != null && grupo.Contains(a))

{

grupo.Remove(a);

Lanzar(a.Especialidad);

}

}

public void Guardar()

{

FileStream fs = new FileStream("alumnos.data", FileMode.Create);

BinaryFormatter bf = new BinaryFormatter();

bf.Serialize(fs, grupo);

fs.Close();

}

public void Cargar()

{

if (File.Exists("alumnos.data"))

{

FileStream fs = new FileStream("alumnos.data", FileMode.Open);

BinaryFormatter bf = new BinaryFormatter();

grupo = (List<Alumno>)bf.Deserialize(fs);

fs.Close();

Lanzar("Licenciatura en Informatica");

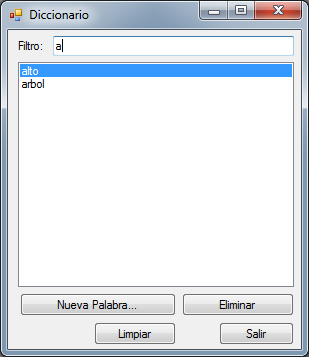
Lanzar("Ingenieria en Sistemas");

}

}

## Ejercicio 2. Palabras del diccionario

### Formulario Diccionario



public partial class frmDiccionario : Form

{

public frmDiccionario()

{

InitializeComponent();

}

Diccionario diccionario = new Diccionario();

private void frmDiccionario\_Load(object sender, EventArgs e)

{

diccionario.Actualizado += new Diccionario.Actualizar(diccionario\_Actualizado);

diccionario.Cargar();

}

void diccionario\_Actualizado()

{

lstPalabras.DataSource = null;

lstPalabras.DataSource = diccionario.Filtrar(txtFiltro.Text);

}

private void btnNuevaPalabra\_Click(object sender, EventArgs e)

{

frmNuevaPalabra nueva = new frmNuevaPalabra();

nueva.Dic = diccionario;

nueva.ShowDialog();

}

private void btnEliminar\_Click(object sender, EventArgs e)

{

if (lstPalabras.SelectedIndex != -1)

{

diccionario.EliminarPalabra((Palabra)lstPalabras.SelectedItem);

}

}

private void btnLimpiar\_Click(object sender, EventArgs e)

{

txtFiltro.Clear();

txtFiltro.Focus();

}

private void btnSalir\_Click(object sender, EventArgs e)

{

Close();

}

private void txtFiltro\_TextChanged(object sender, EventArgs e)

{

diccionario\_Actualizado();

}

private void lstPalabras\_DoubleClick(object sender, EventArgs e)

{

if (lstPalabras.SelectedIndex != -1)

{

frmInfo info = new frmInfo();

info.Word = (Palabra)lstPalabras.SelectedItem;

info.ShowDialog();

}

}

private void frmDiccionario\_FormClosing(object sender, FormClosingEventArgs e)

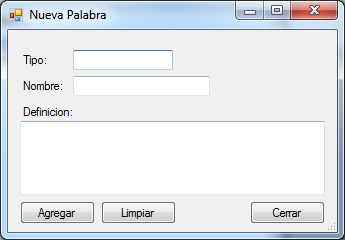
{

diccionario.Guardar();

}

}

### Formulario Nueva Palabra



public partial class frmNuevaPalabra : Form

{

public frmNuevaPalabra()

{

InitializeComponent();

}

private Diccionario dic;

public Diccionario Dic

{

get { return dic; }

set { dic = value; }

}

private void btnLimpiar\_Click(object sender, EventArgs e)

{

txtTipo.Clear();

txtNombre.Clear();

txtDefinicion.Clear();

txtTipo.Focus();

}

private void btnCerrar\_Click(object sender, EventArgs e)

{

Close();

}

private void btnAgregar\_Click(object sender, EventArgs e)

{

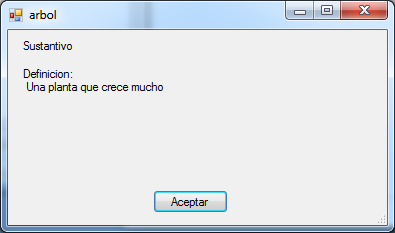
Dic.AgregarPalabra(txtTipo.Text, txtNombre.Text, txtDefinicion.Text);

btnLimpiar.PerformClick();

}

}

### Formulario Información Palabra



public partial class frmInfo : Form

{

public frmInfo()

{

InitializeComponent();

}

private Palabra word;

public Palabra Word

{

get { return word; }

set

{

word = value;

this.Text = word.Nombre;

lblTipo.Text = word.Tipo;

lblDefinicion.Text = string.Format("Definicion:\n {0}", word.Definicion);

}

}

private void btnAceptar\_Click(object sender, EventArgs e)

{

Close();

}

}

### Clase Palabra

[Serializable]

public class Palabra

{

private string tipo;

public string Tipo

{

get { return tipo; }

set { tipo = value; }

}

private string nombre;

public string Nombre

{

get { return nombre; }

set { nombre = value; }

}

private string definicion;

public string Definicion

{

get { return definicion; }

set { definicion = value; }

}

public override string ToString()

{

return nombre;

}

}

### Clase Diccionario

[Serializable]

public class Diccionario

{

private List<Palabra> palabras;

public List<Palabra> Palabras

{

get { return palabras; }

}

public delegate void Actualizar();

public event Actualizar Actualizado;

private void Lanzar()

{

if (Actualizado != null)

Actualizado();

}

int Ordenar(Palabra a, Palabra b)

{

return a.Nombre.CompareTo(b.Nombre);

}

public void AgregarPalabra(string tipo, string nombre, string definicion)

{

if (palabras == null)

palabras = new List<Palabra>();

if (palabras.Exists(delegate(Palabra p) { return p.Nombre.ToLower() == nombre.ToLower(); }))

throw new ArgumentException(string.Format("Ya existe una definicion para la palabra {0}", nombre));

Palabra word = new Palabra();

word.Tipo = tipo;

word.Nombre = nombre;

word.Definicion = definicion;

palabras.Add(word);

palabras.Sort(Ordenar);

Lanzar();

}

public void EliminarPalabra(Palabra p)

{

if (palabras.Contains(p))

{

palabras.Remove(p);

Lanzar();

}

}

public List<Palabra> Filtrar(string filtro)

{

if (filtro == "")

return palabras;

return palabras.FindAll(delegate(Palabra p) { return p.Nombre.StartsWith(filtro.ToLower()); });

}

public void Guardar()

{

FileStream fs = new FileStream("diccionario.data", FileMode.Create);

BinaryFormatter bf = new BinaryFormatter();

bf.Serialize(fs, palabras);

fs.Close();

}

public void Cargar()

{

if (File.Exists("diccionario.data"))

{

FileStream fs = new FileStream("diccionario.data", FileMode.Open);

BinaryFormatter bf = new BinaryFormatter();

palabras = (List<Palabra>)bf.Deserialize(fs);

fs.Close();

Lanzar();

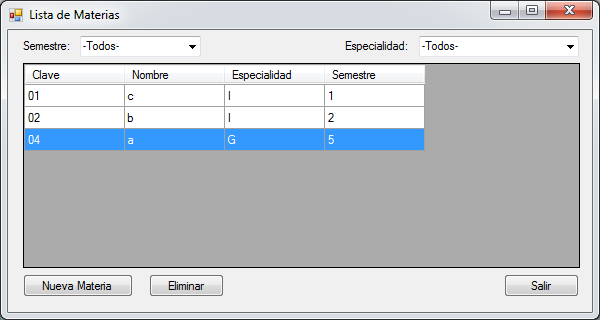
}

}

}

## Ejercicio 3. Retícula

### Formulario Materias



public partial class frmReticula : Form

{

public frmReticula()

{

InitializeComponent();

}

Reticula reticula = new Reticula();

private void btnNuevaMateria\_Click(object sender, EventArgs e)

{

frmNuevaMateria nueva = new frmNuevaMateria();

nueva.Ret = reticula;

nueva.ShowDialog();

}

void reticula\_Actualizado()

{

dgvReticula.DataSource = null;

char e = '\0';

if ((string)cmbEspecialidad.SelectedItem == "Ing. Sistemas (G)")

e = 'G';

if ((string)cmbEspecialidad.SelectedItem == "Lic. Informatica (I)")

e = 'I';

byte s = 0;

if ((string)cmbSemestre.SelectedItem != "-Todos-")

s = byte.Parse((string)cmbSemestre.SelectedItem);

dgvReticula.DataSource = reticula.Filtrar(e, s);

}

private void frmReticula\_Load(object sender, EventArgs e)

{

reticula.Actualizado += new Reticula.Actualizar(reticula\_Actualizado);

reticula.Cargar();

}

private void btnEliminar\_Click(object sender, EventArgs e)

{

if (dgvReticula.SelectedRows.Count == 1)

{

reticula.Eliminar((Materia)dgvReticula.SelectedRows[0].DataBoundItem);

}

}

private void btnCancelar\_Click(object sender, EventArgs e)

{

Close();

}

private void cmbSemestre\_SelectedIndexChanged(object sender, EventArgs e)

{

reticula\_Actualizado();

}

private void cmbEspecialidad\_SelectedIndexChanged(object sender, EventArgs e)

{

reticula\_Actualizado();

}

private void frmReticula\_FormClosing(object sender, FormClosingEventArgs e)

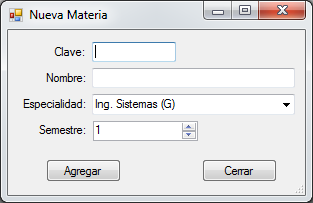
{

reticula.Guardar();

}

}

### Formulario Nueva Materia



public partial class frmNuevaMateria : Form

{

public frmNuevaMateria()

{

InitializeComponent();

}

private Reticula ret;

public Reticula Ret

{

get { return ret; }

set { ret = value; }

}

private void btnAgregar\_Click(object sender, EventArgs e)

{

try

{

char esp = ((string)cmbEspecialidad.SelectedItem == "Ing. Sistemas (G)") ? 'G' : 'I';

Ret.Agregar(txtClave.Text, txtNombre.Text, esp , (byte)nmrSemestre.Value);

txtClave.Clear();

txtNombre.Clear();

cmbEspecialidad.SelectedIndex = -1;

nmrSemestre.Value = 1;

txtClave.Focus();

}

catch (Exception ex) { MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error); }

}

private void btnCerrar\_Click(object sender, EventArgs e)

{

Close();

}

}

### Clase Materia

[Serializable]

public class Materia

{

private string clave;

public string Clave

{

get { return clave; }

set { clave = value; }

}

private string nombre;

public string Nombre

{

get { return nombre; }

set { nombre = value; }

}

private char especialidad;

public char Especialidad

{

get { return especialidad; }

set

{

if (value != 'G' && value != 'I')

throw new ApplicationException("Solo manejamos las carreras de: \n-Ing. Sistemas (G) \n-Lic. Informatica (I)");

especialidad = value;

}

}

private byte semestre;

public byte Semestre

{

get { return semestre; }

set

{

if (value > 9)

throw new ApplicationException("La reticula oficial se distrubuye solo en 9 semestres");

semestre = value;

}

}

}

### Clase Reticula

[Serializable]

public class Reticula

{

List<Materia> reticulas;

public delegate void Actualizar();

public event Actualizar Actualizado;

private void Lanzar()

{

if (Actualizado != null)

Actualizado();

}

public void Agregar(string clave, string nombre, char especialidad, byte semestre)

{

if (reticulas == null)

reticulas = new List<Materia>();

if (reticulas.Exists(delegate(Materia mat) { return (mat.Nombre == nombre && mat.Especialidad == especialidad); }))

throw new ArgumentException("El nombre ya esta utilizada");

if (reticulas.Exists(delegate(Materia mat) { return (mat.Clave == clave && mat.Especialidad == especialidad); }))

throw new ArgumentException("La clave ya esta utilizado");

Materia m = new Materia();

m.Clave = clave;

m.Nombre = nombre;

m.Especialidad = especialidad;

m.Semestre = semestre;

reticulas.Add(m);

Lanzar();

}

public void Eliminar(Materia m)

{

if (reticulas.Contains(m))

{

reticulas.Remove(m);

Lanzar();

}

}

public List<Materia> Filtrar(char especialidad, byte semestre)

{

if (semestre == 0 && especialidad != '\0')

{

return reticulas.FindAll(delegate(Materia m) { return m.Especialidad == especialidad; });

}

else if (semestre != 0 && especialidad == '\0')

{

return reticulas.FindAll(delegate(Materia m) { return m.Semestre == semestre; });

}

else if (semestre != 0 && especialidad != '\0')

{

return reticulas.FindAll(delegate(Materia m) { return m.Semestre == semestre && m.Especialidad == especialidad; });

}

else

{

return reticulas;

}

}

public void Guardar()

{

FileStream fs = new FileStream("reticula.data", FileMode.Create);

BinaryFormatter bf = new BinaryFormatter();

bf.Serialize(fs, reticulas);

fs.Close();

}

public void Cargar()

{

if (File.Exists("reticula.data"))

{

FileStream fs = new FileStream("reticula.data", FileMode.Open);

BinaryFormatter bf = new BinaryFormatter();

reticulas = (List<Materia>)bf.Deserialize(fs);

fs.Close();

Lanzar();

}

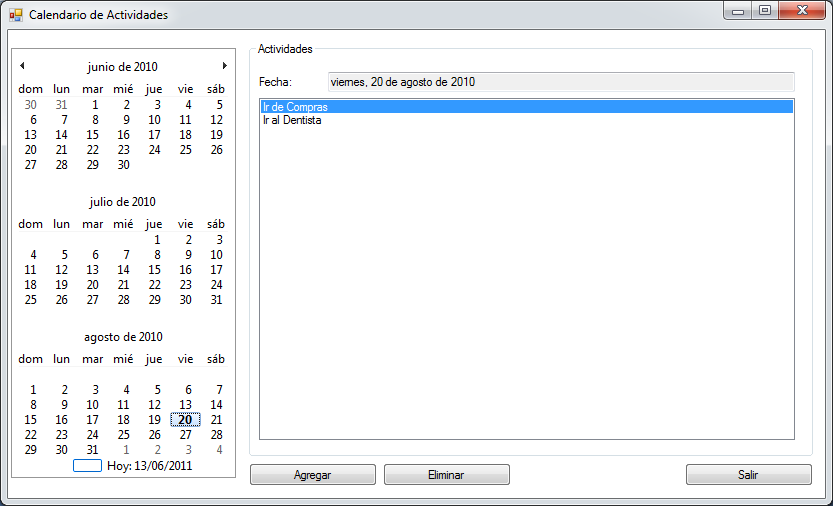
}

}

# Unidad 4.

## Ejercicio 1. Calendario de Actividades

### Formulario Calendario de actividades



public class CalendarioActividades

{

private Dictionary<DateTime, List<string>> actividades;

public Dictionary<DateTime, List<string>> Actividades

{

get { return actividades; }

set { actividades = value; }

}

public delegate void Actualizar();

public event Actualizar Actualizado;

void Lanzar()

{

if (Actualizado != null)

Actualizado();

}

public void Agregar(DateTime fecha, string actividad)

{

if (actividades == null)

{

actividades = new Dictionary<DateTime, List<string>>();

}

if (actividades.ContainsKey(fecha.Date))

{

actividades[fecha.Date].Add(actividad);

}

else

{

List<string> lista = new List<string>();

lista.Add(actividad);

actividades[fecha.Date] = lista;

}

Lanzar();

}

public void Eliminar(DateTime fecha)

{

if (actividades != null && actividades.ContainsKey(fecha.Date))

{

actividades.Remove(fecha.Date);

Lanzar();

}

}

public void Eliminar(DateTime fecha, string actividad)

{

if (actividades != null && actividades.ContainsKey(fecha.Date))

{

if (actividades[fecha.Date].Contains(actividad))

{

actividades[fecha.Date].Remove(actividad);

Lanzar();

}

}

}

public List<string> RecuperarActividades(DateTime fecha)

{

if (actividades != null && actividades.ContainsKey(fecha.Date))

{

return actividades[fecha.Date];

}

else

{

return null;

}

}

public void Guardar()

{

if (actividades != null)

{

FileStream fs = File.Create("actividades.data");

BinaryFormatter bf = new BinaryFormatter();

bf.Serialize(fs, actividades);

fs.Close();

}

}

public void Cargar()

{

if (File.Exists("actividades.data"))

{

FileStream fs = File.OpenRead("actividades.data");

BinaryFormatter bf = new BinaryFormatter();

try

{

actividades = (Dictionary<DateTime, List<string>>)bf.Deserialize(fs);

Lanzar();

}

catch { }

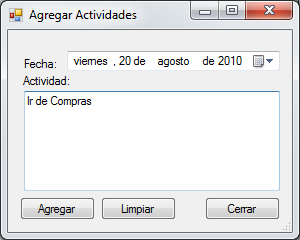
fs.Close();

}

}

}

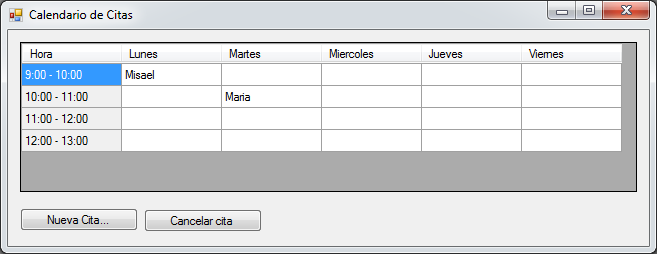
### Formulario Agregar actividades



# Unidad 5.

## Ejercicio 1. Citas del Dentista

### Formulario Calendario de citas



public class Consultas

{

public Consultas()

{

consultas = new DataTable("Consultas");

consultas.Columns.Add(new DataColumn("Hora", typeof(string)));

consultas.Columns.Add(new DataColumn("Lunes", typeof(string)));

consultas.Columns.Add(new DataColumn("Martes", typeof(string)));

consultas.Columns.Add(new DataColumn("Miercoles", typeof(string)));

consultas.Columns.Add(new DataColumn("Jueves", typeof(string)));

consultas.Columns.Add(new DataColumn("Viernes", typeof(string)));

if (!File.Exists("consultas.xml"))

{

for (int i = 9; i < 13; i++)

{

DataRow r = consultas.NewRow();

r["Hora"] = string.Format("{0}:00 - {1}:00", i, i + 1);

consultas.Rows.Add(r);

}

}

else

Cargar();

}

private DataTable consultas;

public DataTable ConsultasDentista

{

get { return consultas; }

}

public void NuevaCita(string paciente, byte hora, string dia)

{

if (hora < 9 || hora > 12)

throw new ArgumentException("No hay consultas en el horario solicitado");

if (consultas.Rows[hora - 9][dia] is string)

throw new ArgumentException("Lo sentimos, ya existe una cita programada para tal fecha");

foreach (DataRow r in consultas.Rows)

{

if (r[dia].ToString() == paciente)

throw new ArgumentException("Usted ya tiene una cita programada para el mismo dia");

}

consultas.Rows[hora - 9][dia] = paciente;

Guardar();

}

public void CancelarCita(byte hora, string dia)

{

if (!string.IsNullOrEmpty((string)consultas.Rows[hora - 9][dia]))

{

consultas.Rows[hora - 9][dia] = null;

Guardar();

}

}

public void Guardar()

{

consultas.WriteXml("consultas.xml");

}

public void Cargar()

{

if (File.Exists("consultas.xml"))

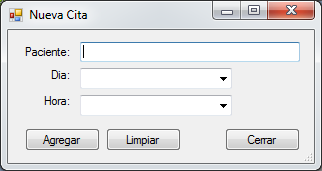
{

consultas.ReadXml("consultas.xml");

}

}

### Formulario Nueva cita



public partial class frmNuevaCita : Form

{

public frmNuevaCita()

{

InitializeComponent();

}

private Consultas citas;

public Consultas Citas

{

get { return citas; }

set { citas = value; }

}

private void btnAgregar\_Click(object sender, EventArgs e)

{

try

{

if (cmbDia.SelectedIndex != -1 && cmbHora.SelectedIndex != -1)

{

citas.NuevaCita(txtPaciente.Text, (byte)(cmbHora.SelectedIndex + 9), (string)cmbDia.SelectedItem);

btnLimpiar.PerformClick();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void btnLimpiar\_Click(object sender, EventArgs e)

{

txtPaciente.Clear();

cmbDia.SelectedIndex = -1;

cmbHora.SelectedIndex = -1;

txtPaciente.Focus();

}

private void btnCerrar\_Click(object sender, EventArgs e)

{

Close();

}

}

### Clase Consultas

public class Consultas

{

public Consultas()

{

consultas = new DataTable("Consultas");

consultas.Columns.Add(new DataColumn("Hora", typeof(string)));

consultas.Columns.Add(new DataColumn("Lunes", typeof(string)));

consultas.Columns.Add(new DataColumn("Martes", typeof(string)));

consultas.Columns.Add(new DataColumn("Miercoles", typeof(string)));

consultas.Columns.Add(new DataColumn("Jueves", typeof(string)));

consultas.Columns.Add(new DataColumn("Viernes", typeof(string)));

if (!File.Exists("consultas.xml"))

{

for (int i = 9; i < 13; i++)

{

DataRow r = consultas.NewRow();

r["Hora"] = string.Format("{0}:00 - {1}:00", i, i + 1);

consultas.Rows.Add(r);

}

}

else

Cargar();

}

private DataTable consultas;

public DataTable ConsultasDentista

{

get { return consultas; }

}

public void NuevaCita(string paciente, byte hora, string dia)

{

if (hora < 9 || hora > 12)

throw new ArgumentException("No hay consultas en el horario solicitado");

if (consultas.Rows[hora - 9][dia] is string)

throw new ArgumentException("Lo sentimos, ya existe una cita programada para tal fecha");

foreach (DataRow r in consultas.Rows)

{

if (r[dia].ToString() == paciente)

throw new ArgumentException("Usted ya tiene una cita programada para el mismo dia");

}

consultas.Rows[hora - 9][dia] = paciente;

Guardar();

}

public void CancelarCita(byte hora, string dia)

{

if (!string.IsNullOrEmpty((string)consultas.Rows[hora - 9][dia]))

{

consultas.Rows[hora - 9][dia] = null;

Guardar();

}

}

public void Guardar()

{

consultas.WriteXml("consultas.xml");

}

public void Cargar()

{

if (File.Exists("consultas.xml"))

{

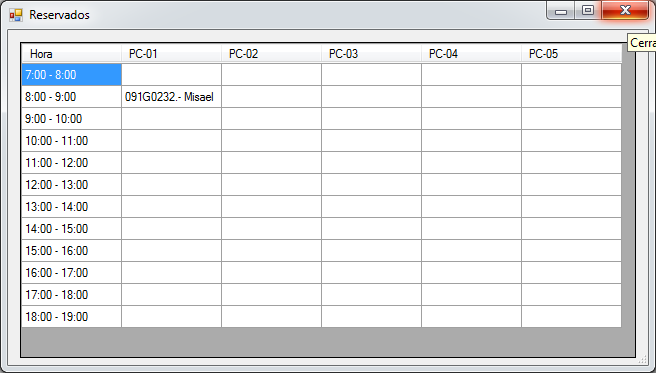
consultas.ReadXml("consultas.xml");

}

}

## Ejercicio 2. Laboratorio de Computo

### Formulario Reservados



### Clase Alumno

public class Alumno

{

public Alumno(string nocontrol, string nombrea)

{

NoControl = nocontrol;

Nombre = nombrea;

}

private string control;

public string NoControl

{

get { return control; }

set { control = value; }

}

private string nombre;

public string Nombre

{

get { return nombre; }

set { nombre = value; }

}

public override string ToString()

{

return string.Format("{0}.- {1}",control,nombre);

}

}

### Clase Alumnos

public class Alumnos

{

public Alumnos()

{

Cargar();

}

private List<Alumno> alumnos;

public List<Alumno> AlumnosUsuarios

{

get { return alumnos; }

set { alumnos = value; }

}

public delegate void Actualizar();

public event Actualizar Actualizado;

private void Lanzar()

{

Guardar();

if (Actualizado != null)

Actualizado();

}

int Ordenar(Alumno a, Alumno b)

{

return a.Nombre.CompareTo(b.Nombre);

}

public void AgregarAlumno(string control, string nombre)

{

if (alumnos == null)

alumnos = new List<Alumno>();

if (alumnos.Exists(delegate(Alumno a) { return a.NoControl == control; }))

throw new ArgumentException("El NoControl ya esta en uso, por favor verifiquelo");

alumnos.Add(new Alumno(control, nombre));

alumnos.Sort(Ordenar);

Lanzar();

}

public void EliminarAlumno(Alumno a)

{

if (alumnos.Contains(a))

{

alumnos.Remove(a);

Lanzar();

}

}

public Alumno Buscar(string filtro)

{

if (alumnos == null)

return null;

Alumno al = alumnos.Find(delegate(Alumno a) { return a.NoControl == filtro; });

if (al == null)

al = alumnos.Find(delegate(Alumno a) { return a.Nombre == filtro; });

if (al == null)

throw new ArgumentException("El alumno especificado no se encuentra enlistado en el laboratorio de computo");

return al;

}

public void Guardar()

{

FileStream fs = File.Create("alumnos.data");

BinaryFormatter bf = new BinaryFormatter();

bf.Serialize(fs, alumnos);

fs.Close();

}

public void Cargar()

{

if (File.Exists("alumnos.data"))

{

FileStream fs = File.OpenRead("alumnos.data");

BinaryFormatter bf = new BinaryFormatter();

alumnos = (List<Alumno>)bf.Deserialize(fs);

fs.Close();

}

}

### Clase Reservaciones

public class Reservaciones

{

public Reservaciones()

{

lab = new DataTable("Laboratorio");

lab.Columns.Add(new DataColumn("Hora", typeof(string)));

lab.Columns.Add(new DataColumn("PC-01", typeof(Alumno)));

lab.Columns.Add(new DataColumn("PC-02", typeof(Alumno)));

lab.Columns.Add(new DataColumn("PC-03", typeof(Alumno)));

lab.Columns.Add(new DataColumn("PC-04", typeof(Alumno)));

lab.Columns.Add(new DataColumn("PC-05", typeof(Alumno)));

if (!File.Exists("reservados.xml"))

{

for (int i = 7; i < 19; i++)

{

DataRow r = lab.NewRow();

r["Hora"] = string.Format("{0}:00 - {1}:00", i, i + 1);

lab.Rows.Add(r);

}

}

else

Cargar();

}

private DataTable lab;

public DataTable Reservados

{

get { return lab; }

}

public void Reservar(string computadora, int hora, int cantidad, Alumno alumno)

{

if (cantidad > 3)

throw new ArgumentException("No puedes usar mas de 3 horas al dia");

for (int i = 0; i < cantidad; i++)

{

if (lab.Rows[hora - 7 + i][computadora] is Alumno)

throw new ArgumentException("La hora solicitada esta ya reservada");

}

for (int i = 1; i < lab.Columns.Count; i++)

{

foreach (DataRow r in lab.Rows)

{

if (r[i] == alumno)

throw new ArgumentException("El alumno ya ha reservado una computadora");

}

}

for (int i = 0; i < cantidad; i++)

{

lab.Rows[hora - 7 + i][computadora] = alumno;

}

Guardar();

}

public void Eliminar(Alumno alumno)

{

for (int i = 1; i < lab.Columns.Count; i++)

{

foreach (DataRow r in lab.Rows)

{

if (r[i] is Alumno && ((Alumno)r[i]).NoControl == alumno.NoControl)

r[i] = null;

}

}

Guardar();

}

public void Guardar()

{

lab.WriteXml("reservados.xml");

}

public void Cargar()

{

if (File.Exists("reservados.xml"))

lab.ReadXml("reservados.xml");

}

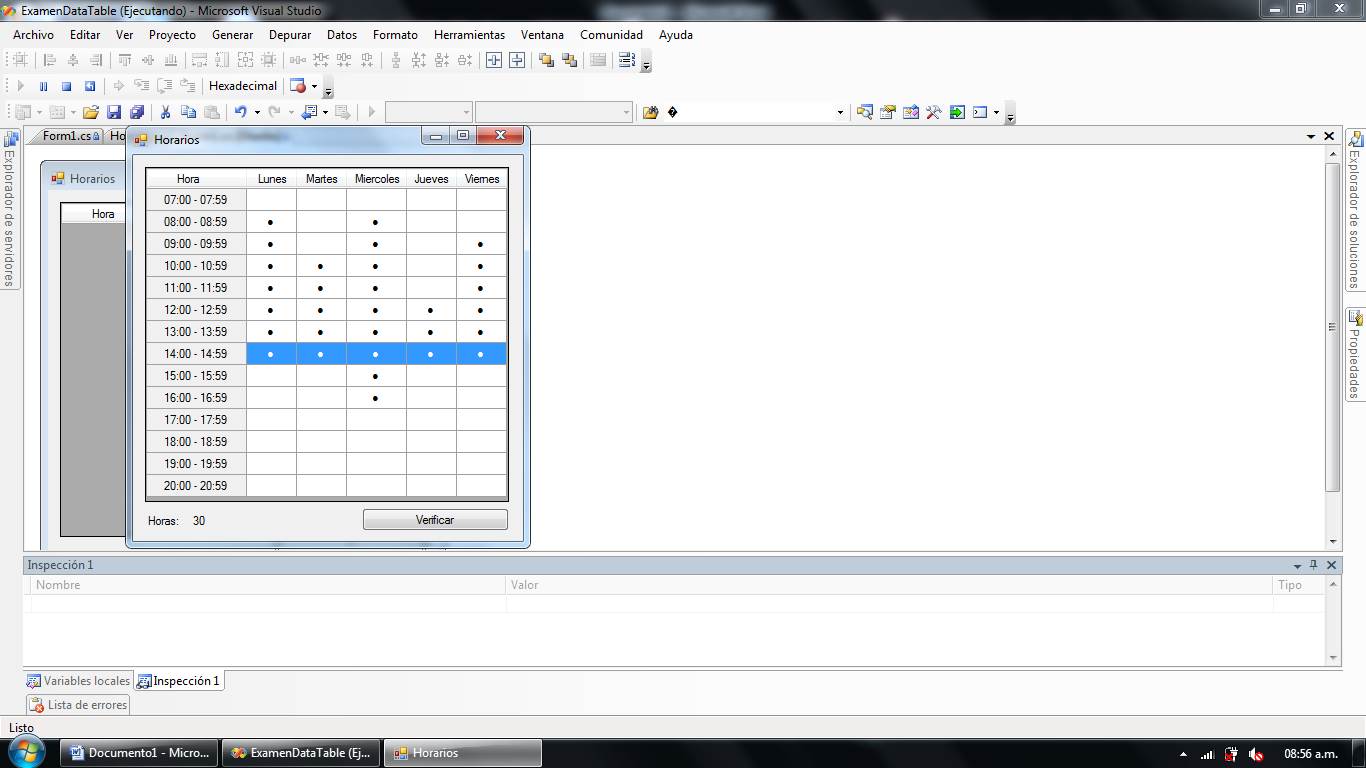
}

## Ejercicio 3. Horario del Docente

Se desea crear el horario de trabajo de un docente. El maestro es libre de elegir las horas a las cuales quiere trabajar tomando en cuenta las siguientes restricciones.

* El docente debe cubrir un total de 30 horas semanales de trabajo.
* Ningún docente puede trabajar más de 10 horas diarias.
* Los docentes deben asistir a trabajar al menos 4 días a la semana (sin importar el número de horas que trabaje al día).
* Todas las horas que trabaje en un mismo día deben ser continuas.

Crear una clase que nos permita dar solución a este problema y un formulario donde el docente seleccionara sus horas como el que se muestra a continuación.



* El docente elegirá su horario de trabajo dando clic en las celdas correspondientes al día y hora a las cuales desea asistir.
* Cada celda cambia de seleccionada a no seleccionada cada vez que el usuario de clic en ella.
* Para facilitarle las cosas, el docente pude seleccionar varias celdas simultáneamente y todas cambiaran de estado.
* No se puede escribir en las celdas del DataGrid.
* Al presionar el botón verificar se le indicara al docente si su horario es autorizado (si cumple todas la reglas) o si debe corregirlo indicando la restricción que ha incumplido.