	INSTITUTO TECNOLÓGICO DE NUEVO LAREDO **ING. EN SISTEMAS COMPUTACIONALES**			
	MATERIA: Programación Orientada a Objetos (C#)	**UNIDAD:** 6	**PRÁCTICA:** 5	
NOMBRE DE LA PRÁCTICA: Archivos (flujos)				
MAESTRO: M.C. Bruno López Takeyas		**EMAIL: bruno.lt@nlaredo.tecnm.mx**		

OBJETIVO: El estudiante elaborará aplicaciones visuales y de consola en C# para almacenar datos en archivos (flujos).
MATERIAL Y EQUIPO NECESARIO: • Se recomienda la utilización de software para elaborar diagramas de clases de UML y diagramas de flujo • Elaborar programas de los ejercicios en C#

Desarrolle una aplicación para grabar y leer los datos indicados en un archivo.

1. Diseñe una aplicación de consola que almacene avisos (mensajes de texto) en un archivo secuencial llamado `Avisos.txt` y cuyo programa principal tenga un menú con las siguientes opciones:

```
MENÚ DE OPCIONES
1.- Grabar un aviso
2.- Reporte de avisos almacenados
0.- Salir
```

Cada opción debe invocar un método para realizar la actividad solicitada. El método `GrabarAviso()` debe solicitar al usuario que capture un aviso (a través de una variable de tipo `string`), abrir un archivo en modo `Append` (crearlo en caso de que no exista), agregar el aviso al final del archivo y cerrarlo al terminar la operación.

El método `ReporteAvisos()` debe abrir el archivo en modo lectura, implementar un ciclo que lea todos los avisos almacenados en él para desplegarlos en pantalla y cerrar el archivo al terminar las operaciones de lectura.

NOTA: Todas las operaciones con el archivo deben estar controladas a través de las excepciones correspondientes.

2. Modifique el ejercicio anterior para realizar las mismas operaciones en una aplicación visual. Para ello, diseñe la siguiente forma (Fig. 6.5.1).

Fig. 6.5.1.- Diseño de la forma.

Cuando se inicie la ejecución de la aplicación, el método `Form1_Load()` debe abrir el archivo en modo de lectura para consultar todos los avisos almacenados y desplegarlos en el `listBox` (`lstAvisos`).

Al oprimir el botón para agregar aviso (`btnAgregar`), se debe abrir el archivo en modo `Append` (o crearlo si no existe) para agregar el aviso capturado en el `TextBox` (`txtAviso`) al archivo, desplegarlo en el `listBox` (`lstAvisos`) y cerrarlo al terminar la operación.

NOTA: Todas las operaciones con el archivo deben estar controladas a través de las excepciones correspondientes.

3. Diseñe una aplicación de consola que almacene en un archivo secuencial los datos de los empleados de una empresa. Los datos de cada empleado son:
    - Número (`int`).
    - Nombre (`string`).
    - Sueldo (`double`).

    Para ello, diseñe una clase con estos atributos y sus respectivas propiedades.

    El programa principal debe tener un menú con las siguientes opciones:

```
MENÚ DE OPCIONES
1.- Agregar empleado
2.- Reporte de empleados
0.- Salir
```

Cada opción debe invocar un método para realizar la actividad solicitada. El método `AgregarEmpleado()` debe declarar y crear un objeto local de la clase `Empleado`, solicitar al usuario que capture cada uno de sus datos, abrir un archivo en modo `Append` (crearlo en caso de que no exista), agregar el empleado al final del archivo y cerrarlo al terminar la operación. El método `ReporteEmpleados()` debe abrir el archivo en modo lectura, implementar un ciclo que lea todos los empleados almacenados en él para desplegarlos en pantalla y cerrar el archivo al terminar las operaciones de lectura.

NOTA: Todas las operaciones con el archivo deben estar controladas a través de las excepciones correspondientes.

4. Modifique el ejercicio anterior para realizar las mismas operaciones en una aplicación visual.

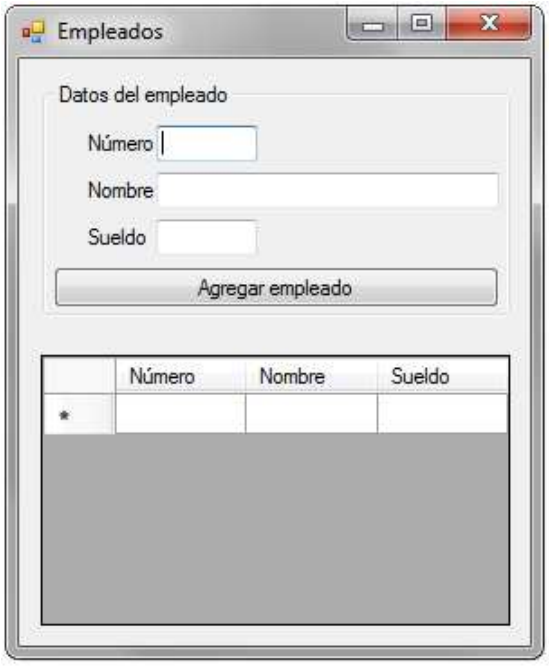


Fig. 6.5.2.- Diseño de la forma.

Cuando se inicie la ejecución de la aplicación, el método `Form1_Load()` debe abrir el archivo en modo de lectura para consultar todos los empleados almacenados y desplegarlos en el `dataGridView` (`dgEmpleados`).

Al oprimir el botón para agregar aviso (`btnAgregarEmpleado`), se debe abrir el archivo en modo `Append` (o crearlo si no existe) para agregar el objeto del empleado con sus datos capturados en los `TextBoxes` al archivo, desplegarlos en el `dataGridView` (`dgEmpleados`) y cerrarlo al terminar la operación.

NOTA: Todas las operaciones con el archivo deben estar controladas a través de las excepciones correspondientes.

5. Utilice la clase `ArchivoSecuencialSerializadoBinario` (Fig. 6.5.4) para almacenar objetos de tipo `Paciente` en un archivo secuencial llamado `Pacientes.dat`.

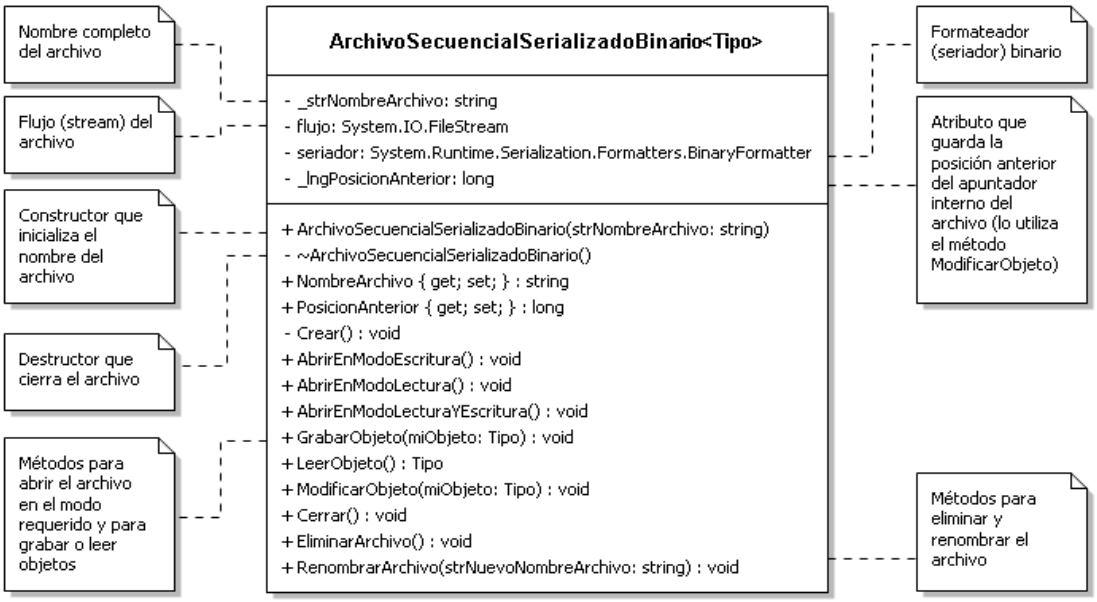


Fig. 6.5.4.- La clase ArchivoSecuencialSerializadoBinario.

Los datos de cada paciente son:

- Nombre (`string`).
- Fecha de nacimiento (`DateTime`).
- Sexo (`char`).
- Tipo de sangre (string).

Las operaciones a realizar con los objetos de los pacientes son:

- Al oprimir el botón Agregar (`btnAgregarPaciente`), se deben capturar los datos de un objeto de tipo `Paciente` en los `textBoxes` y agregarlo al final del archivo.
- Al seleccionar un renglón del `dataGridView` (`dgPacientes`), se deben mostrar sus datos en sus respectivos controles del `groupBox` (`grpDatosPaciente`).
- Al oprimir el botón Eliminar Paciente Seleccionado (`btnEliminarPaciente`), se debe borrar en el archivo el paciente seleccionado del `dataGridView` (`dgPacientes`).
- Cuando se oprima el botón Modificar (`btnModificarPaciente`), significa que el usuario cambió el valor de algún(os) dato(s) del paciente y desea almacenar el objeto en la misma posición que ocupaba en el archivo.

Para ello diseñe la siguiente forma (Fig. 6.5.5):

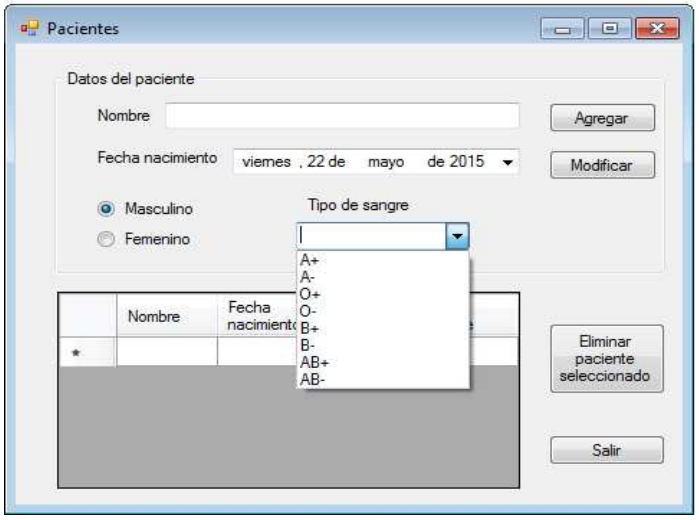


Fig. 6.5.5.- Diseño de la forma

6. Utilice la clase `ArchivoSecuencialSerializadoBinario` (Fig. 6.5.4) para almacenar objetos de tipo `Auto` en un archivo secuencial llamado `Autos.dat`. Los datos de cada automóvil son:
    - Placas (`string`).
    - Marca (`string`).
    - Modelo (año) (`int`).
    - Propietario (`string`).

    Las operaciones a realizar con los objetos de los autos son:
    - Al oprimir el botón Agregar (`btnAgregarAuto`), se deben capturar los datos de un objeto de tipo `Auto` en los `textBoxes` y agregarlo al final del archivo.
    - Al seleccionar un renglón del `dataGridView` (`dgAutos`), se deben mostrar sus datos en los `textBoxes` correspondientes.
    - Al oprimir el botón Eliminar (`btnEliminarAuto`), se debe borrar en el archivo el auto seleccionado del `dataGridView` (`dgAutos`).
    - Cuando se oprima el botón Modificar (`btnModificarAuto`), significa que el usuario cambió el valor de algún(os) dato(s) del auto y desea almacenar el objeto en la misma posición que ocupaba en el archivo.
    - Al oprimir el botón Vaciar Archivo (`btnVaciarArchivo`), se deben borrar todos los autos almacenados y se debe limpiar el `dataGridView` (`dgAutos`).

Para ello diseñe la siguiente forma (Fig. 6.5.6):

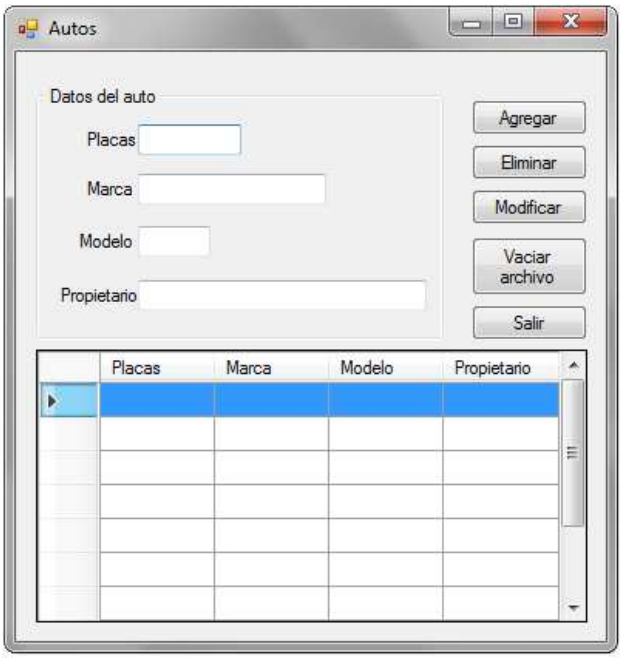


Fig. 6.5.6.- Diseño de la forma.

NOTA: Todas las operaciones con el archivo deben estar controladas a través de las excepciones correspondientes.

7. Utilice la clase `ArchivoSecuencialSerializadoBinario` (Fig. 6.5.4) para almacenar objetos de una clase diseñada a su antojo en un archivo secuencial. Los datos de cada objeto pueden ser elegidos libremente, sin embargo deben cumplir con los siguientes requisitos:
    - La clase debe tener al menos 4 atributos con sus respectivas propiedades.
    - Deben ser de diferentes tipos (incluyendo `int`, `string`, `double`, `bool`, `DateTime`, `char`, etc.)

    Diseñe una forma que realice las siguientes operaciones con los objetos:
    - Al oprimir el botón Agregar (`btnAgregar`), se deben capturar los datos de un objeto y agregarlo al final del archivo.
    - Agregue un `dataGridView` de sólo lectura para mostrar en pantalla los datos de los objetos almacenados en el archivo. Al seleccionar un renglón de dicho `dataGridView`, se deben mostrar sus datos en los controles correspondientes.
    - Al oprimir el botón Eliminar (`btnEliminar`), se debe borrar físicamente del archivo el objeto seleccionado en el `dataGridView`.
    - Cuando se oprima el botón Modificar (`btnModificar`), significa que el usuario cambió el valor de algún(os) dato(s) del objeto y desea almacenarlo en la misma posición que ocupaba en el archivo.
    - Al oprimir el botón Vaciar Archivo (`btnVaciarArchivo`), se debe borrar el archivo con todos los objetos almacenados y se debe limpiar el `dataGridView`.

NOTA: Todas las operaciones de captura de datos y con el archivo deben estar controladas a través de las excepciones correspondientes.

8. Utilice la clase `ArchivoSecuencialSerializadoBinario` (Fig. 6.5.4) para almacenar el número y nombre de empleados de una empresa. Diseñe la siguiente forma, realice las operaciones indicadas por sus botones y valídelas mediante el manejo de excepciones (Fig. 6.5.7):

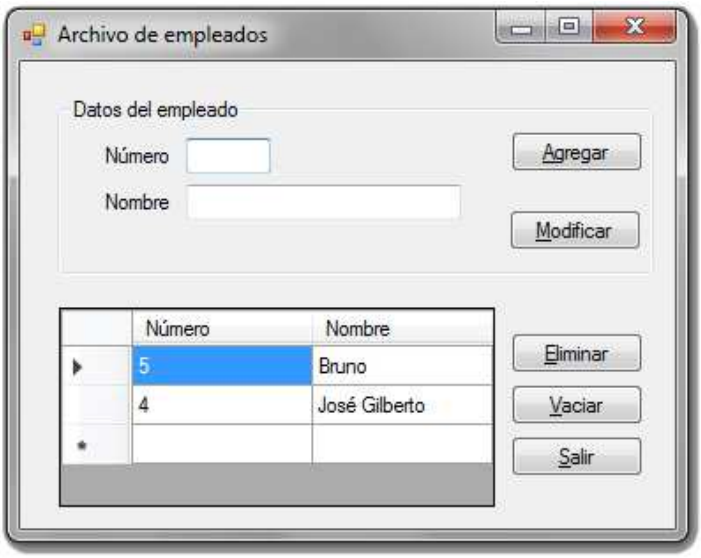


Fig. 6.5.7.- Diseño de la forma.

9. El Depto. de Servicios Escolares del Instituto Tecnológico requiere almacenar en un archivo secuencial los datos de sus estudiantes con sus respectivas materias. Los datos de cada estudiante son:
    - Número de control (`string`)
    - Nombre (`string`)
    - Nombre del padre o tutor (`string`)

- Especialidad (`string`). Las especialidades son Sistemas, Industrial, Arquitectura, Electrónica, Mecatrónica, Eléctrica, Gestión Empresarial, Mecánica, Civil.

Mientras que los datos de cada materia son:

- Clave (`string`)
- Nombre (`string`)
- Cantidad de horas teóricas (`int`). El rango permitido es de 0 a 10.
- Cantidad de horas teóricas (`int`). El rango permitido es de 0 a 10.
- Estatus (`string`). El estatus puede ser “Sin cursar”, “Aprobada en primera”, “Aprobada en segunda”, “Aprobada en especial”, “Reprobada” o “Cursando”.

Reglas de operación del proyecto:

- Valide la captura de todos los datos mediante excepciones (no se permite dejar en blanco ningún campo).
- Valide la captura de las horas teóricas y prácticas de las materias mediante excepciones.
- Al iniciar el proyecto, se deben cargar los datos de los estudiantes del archivo `Estudiantes.dat` y mostrarlos en el `dataGridView` correspondiente (`dgEstudiantes`).
- Al terminar la aplicación, se debe recorrer secuencialmente el `dataGridView` de estudiantes (`dgEstudiantes`) para almacenar de manera permanente los datos en el archivo `Estudiantes.dat`.
- Cada vez que se realice una operación con los datos de los estudiantes y/o materias, se deben actualizar los dataGridViews correspondientes.

Reglas de operación de los estudiantes:

- Al capturar los datos de un estudiante y oprimir el botón “Agregar estudiante”, sus datos deben aparecer en el `dataGridView` correspondiente (`dgEstudiantes`).
- Cuando el usuario seleccione un estudiante del `dataGridView` (`dgEstudiantes`), deben aparecer sus datos en los controles del `groupBox` de estudiantes (`grpDatosEstudiante`) y sus materias en el otro `dataGridView` (`dgMaterias`).
- Para eliminar un estudiante, primero debe seleccionarse del `dataGridView` de estudiantes (`dgEstudiantes`) y después oprimir el botón “Eliminar estudiante” para después actualizar el `dgEstudiantes`. Valide la realización de esta operación.
- El botón “Ordenar estudiantes” sirve para reacomodar de manera ascendente los datos de los estudiantes tomando como referencia el número de control. Para realizar esta operación se debe implementar el método `CompareTo()` de `IComparable` en la clase `Estudiante`.
- Al insertar una materia al kardex del estudiante mediante el método `AgregarMateria()`, si se detecta que el estatus es “Reprobada”, entonces debe dispararse un evento que le notifique al padre o tutor con un mensaje como el siguiente:

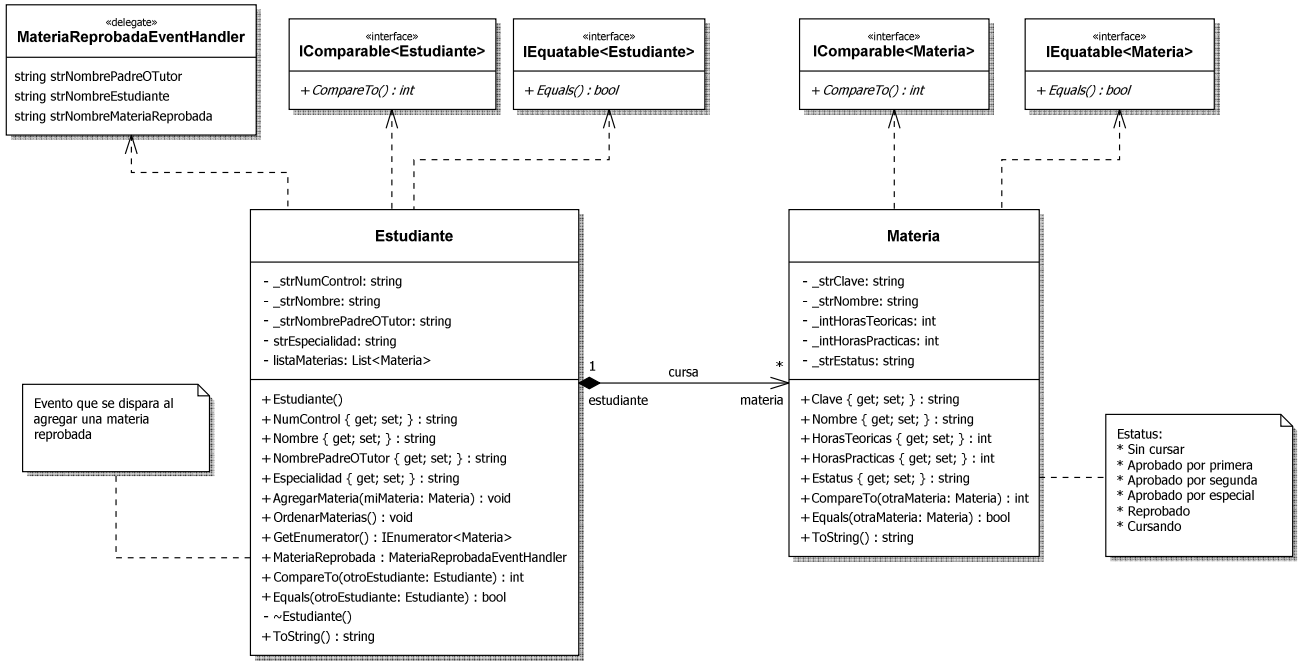
  “Estimado(a) Pedro Ramírez:

  Se le notifica que el estudiante Juan Miguel Ramírez ha reprobado la materia Programación Orientada a Objetos“

Reglas de operación de las materias:

- Para agregar una materia al kardex de un estudiante, primero debe seleccionarse el estudiante de su `dataGridView` (`dgEstudiantes`) y después capturar los datos de la materia para agregarlo a la lista de materias del objeto correspondiente y actualizar los dataGridViews.
- Cuando el usuario seleccione una materia del `dataGridView` (`dgMaterias`), deben aparecer sus datos en los controles del `groupBox` de materias (`grpDatosMaterias`).
- Para eliminar una materia, primero debe seleccionarse el estudiante del `dataGridView` de estudiantes (`dgEstudiantes`), luego escoger la materia a eliminar del `dataGridView` de materias (`dgMaterias`) y después oprimir el botón "Eliminar materia" para después actualizar ambos dataGridViews. Valide la realización de esta operación.
- El botón "Ordenar materias" sirve para reacomodar de manera ascendente los datos de las materias tomando como referencia su clave. Para realizar esta operación se invoca el método `OrdenarMaterias()` de la clase Estudiante y para ello debe implementar el método `CompareTo()` de `IComparable` en la clase `Materia`.

Para ello diseñe el siguiente modelo de clases:



Utilice la clase `ArchivoSecuencialSerializadoBinario` (Fig. 6.5.4) para almacenar estos datos en el archivo `Estudiantes.dat`.

	INSTITUTO TECNOLÓGICO DE NUEVO LAREDO **ING. EN SISTEMAS COMPUTACIONALES**		
	MATERIA: Programación Orientada a Objetos (C#)	**UNIDAD:** 6	**PRÁCTICA:** 5
NOMBRE DE LA PRÁCTICA: Archivos (flujos)			
MAESTRO: M.C. Bruno López Takeyas		**EMAIL: bruno.lt@nlaredo.tecnm.mx**	

Diseñe la siguiente forma: