Universidad Tecnológica Nacional Facultad Regional Avellaneda			
Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos			
Materia: Laboratorio de Programación II			
Apellido:	Langer	Fecha:	14/06/2021
Nombre:	Denise Rocio	Docente(2):	
División:	2°D	Nota(2):	
Legajo:	110053	Firma(2):	

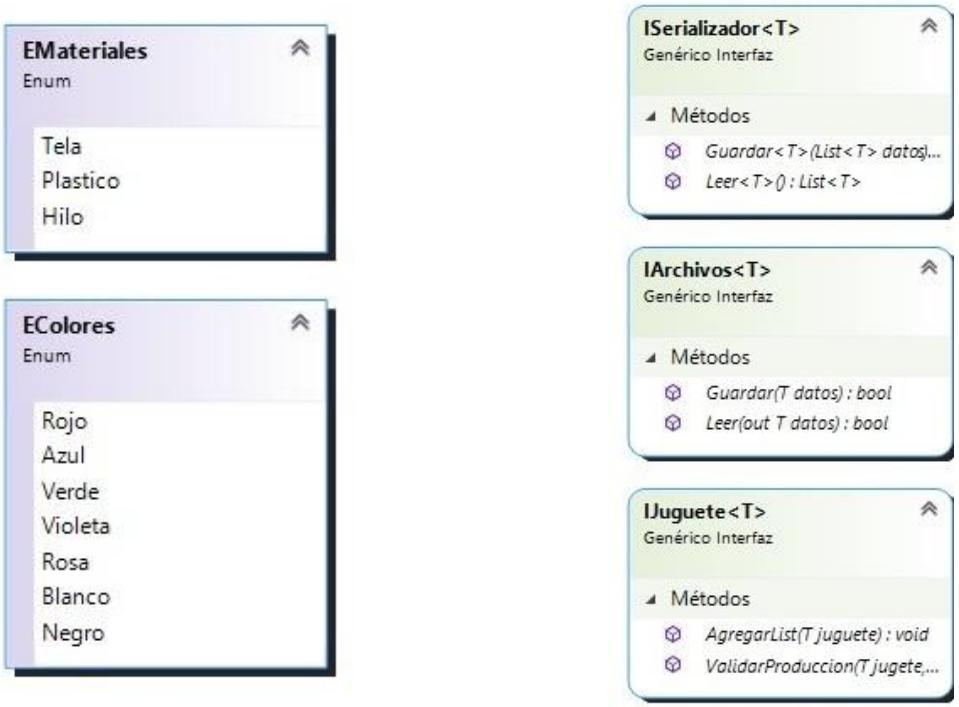
1. Introducción:

El sistema desarrollado para este trabajo práctico representa el proceso de fabricación para distintos tipos de juguetes (específicamente Muñecos, Peluches e Inflables), permitiéndole al usuario:

- Comprar la materia prima necesaria para continuar con el proceso.
- Seleccionar el/los juguetes a ser fabricados, pudiendo indicar distintas características y especificaciones según el producto.
- Editar el diseño completado previamente en caso de necesitar modificar ciertos atributos antes de su fabricación.
- Ver el historial de los últimos juguetes fabricados.

2. UML y Relación con temas explicados:

a. Enumerados e Interfaces

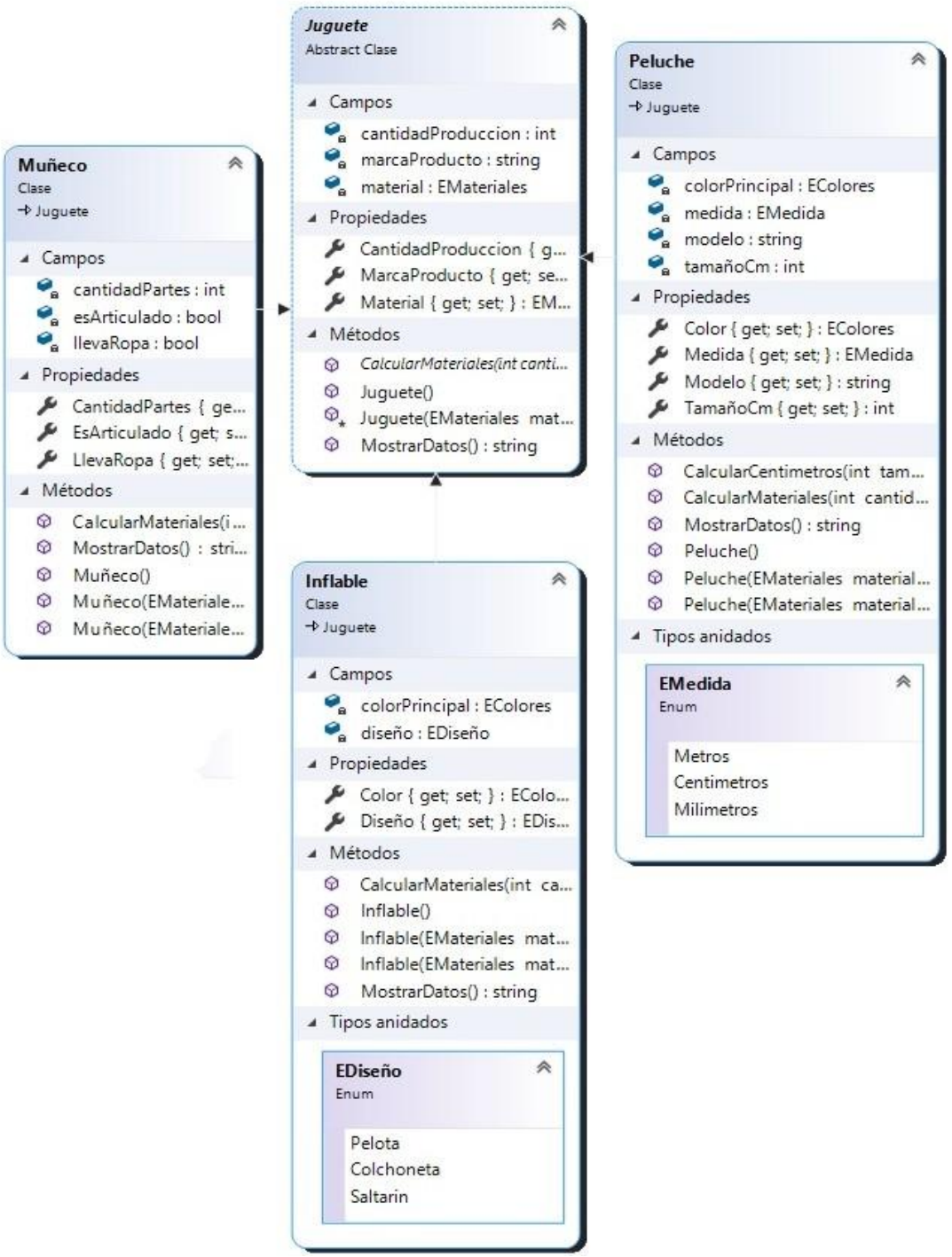


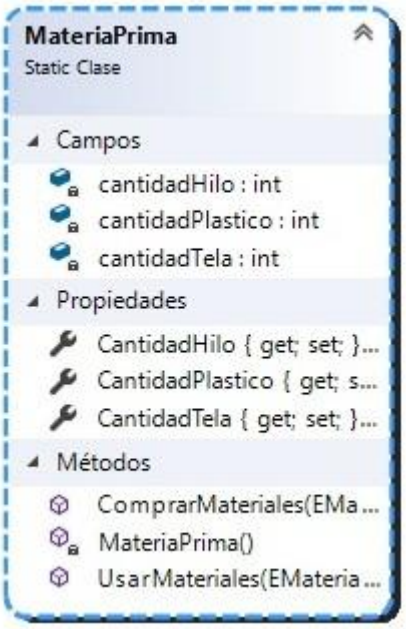
- **ISerializador<T>:** Interfaz genérica que contiene métodos para serializar y deserializar los archivos con formato XML.

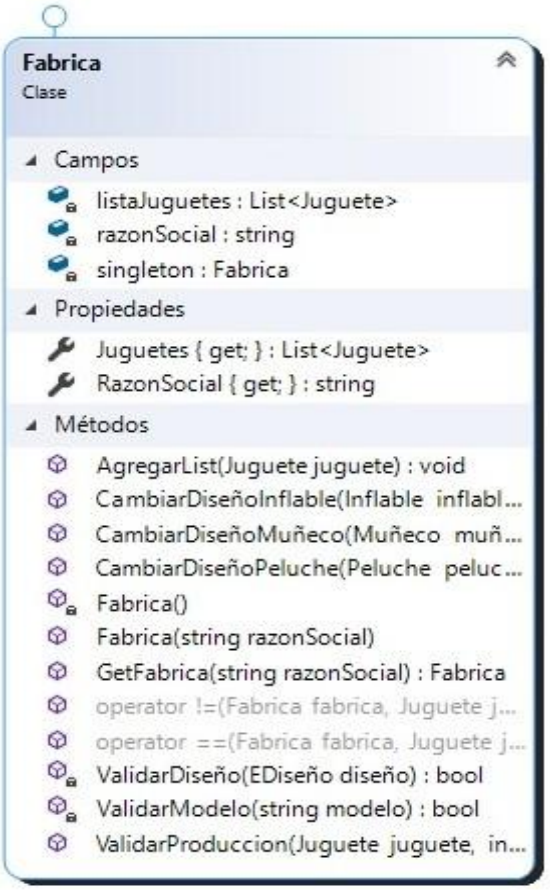
- ***Guardar<T>:*** Serializa los datos que se ingresa como parámetro en un archivo del tipo XML. En caso de no existir el archivo, lo crea.
- ***Leer<T>:*** Deserializa los datos de un archivo.XML a una lista de objetos, validando que exista.

- **IArchivos<T>:** Interfaz genérica que contiene métodos para la creación y lectura de archivos de texto.
  - **Guardar**: Escribe y agrega datos en un archivo de texto. En caso de no existir el archivo, lo crea.
  - **Leer**: Valida que exista un archivo de texto y retorna todo su contenido. En caso de no existir el archivo arroja una excepción.

b. Clases



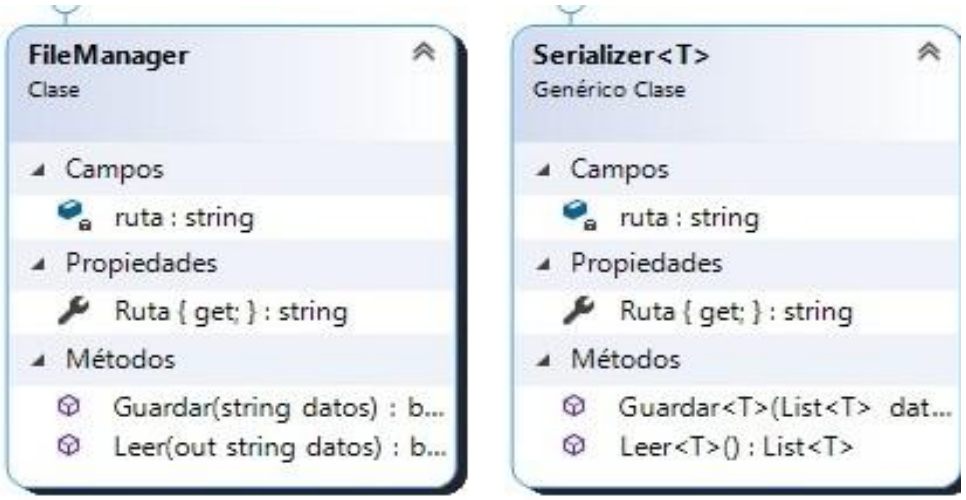




- **Juguete**: Clase abstracta que contiene los atributos, propiedades y métodos a ser heredados por sus clases derivadas (Muñeco, Inflable y Peluche).
- **Fábrica:** Clase que aplica singleton y la interfaz IJuguete<T>.
    - **GetFabrica**: Método estático que utiliza el patrón Singleton. Verifica que ya exista una instancia de Fábrica y la retorna. En caso contrario, instancia la clase y lo retorna.
    - **AgregarList:** Agrega un Juguete a la lista, validando que no se repitan (según criterios específicos de cada tipo de Juguete). En caso de que ya exista, arroja una excepción del tipo JugueteYaExisteException.
    - **ValidarProduccion:** Valida que haya materia prima suficiente para crear un Juguete, según la cantidad que se indique y el tipo de Material elegido. En caso de ser insuficiente, arroja una excepción del tipo NoMaterialesException.
    - **CambiarDiseño:** Modifica ciertos atributos de un Juguete (dependiendo de la clase) y actualiza la lista de Juguetes. En caso de error arroja la excepción JugueteYaExisteException.
    - **Sobrecarga del == :** Valida si un Juguete se repite en listaJuguetes y retorna el resultado con un valor booleano.
    - **ValidarModelo y ValidarDiseño**: Compara si el atributo del Juguete ingresado como parámetro se repite dentro de la lista.
- **Materia Prima:** Clase estática que contiene todos sus elementos estáticos.
    - **ComprarMateriales:** Agrega cantidad de materiales (número entero) a la Materia Prima que se indica como parámetro. En caso de fallas, arroja una excepción del tipo NoMaterialesException.

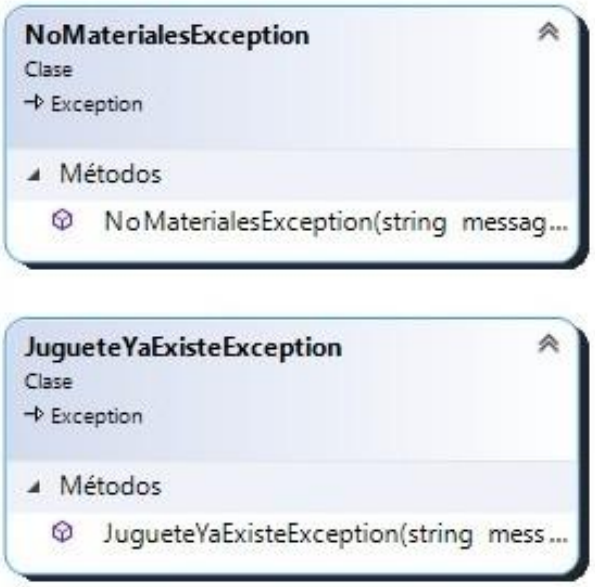
- **UsarMateriales:** Resta una cantidad de materiales (número entero) a la Materia Prima que se indica como parámetro. En caso de fallas, arroja una excepción del tipo NoMaterialesException.

c. Archivos y Serialización



- **FileManager:** Clase que implementa la Interfaz IArchivos<T>. Contiene los métodos y propiedades necesarios para la creación, escritura y lectura de archivos de texto. El archivo será creado en la ruta AppDomain.CurrentDomain.BaseDirectory con el nombre de “Errores.txt”.
- **Serializer**: Clase que implementa la Interfaz ISerializador<T>. Contiene los métodos y propiedades necesarios para la creación, escritura y lectura de archivos con formato XML. Los archivos serán creados en la ruta Environment.CurrentDirectory con los nombres de cada tipo de juguete.

d. Excepciones



- **JugueteYaExisteException**: Excepción que se lanzará cuando ya se registró un Juguete con la misma Marca y/o características.
- **NoMaterialesException**: Excepción que se lanzará cuando no haya suficientes materiales para fabricar un Juguete.

e. *Test Unitarios*

Se encuentran dentro del proyecto “EntidadesTests”. Se realizan los test unitarios para las funcionalidades principales desarrolladas en la biblioteca de clases, tanto para los casos de éxito como para los casos de fallas (validando que se arroje la excepción detallada).

3. Aclaraciones

a. Se crea un proyecto de consola con el nombre “Test” para probar las funcionalidades principales del proyecto (creación de los objetos, agregarlos dentro de la lista, modificación de sus atributos y mostrar los datos actualizados)

b. El tema relacionado a Genericos se implementa en las interfaces.

c. Dentro del formulario principal se instancia un objeto del tipo SoundPlayer, el cual también cuenta con la validación de que exista el archivo especificado.

d. El archivo XML se crea al momento de Fabricar los juguetes de la lista y el archivo de texto se crea al momento de arrojar la primera excepción, guardando todas las excepciones que serán lanzadas durante la ejecución.

Universidad Tecnológica Nacional Facultad Regional Avellaneda			
Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos			
Materia: Laboratorio de Programación II			
Apellido:	Langer	Fecha:	12/07/2021
Nombre:	Denise Rocio	Docente(2):	
División:	2°D	Nota(2):	
Legajo:	110053	Firma(2):	

4. Nuevas Funcionalidades y Aclaraciones:

Ante la necesidad de poder persistir la información de los juguetes a ser fabricados y consultar los que ya fueron previamente fabricados en otras instancias, el usuario solicita la creación de una Base de Datos y las tablas correspondientes para ello.

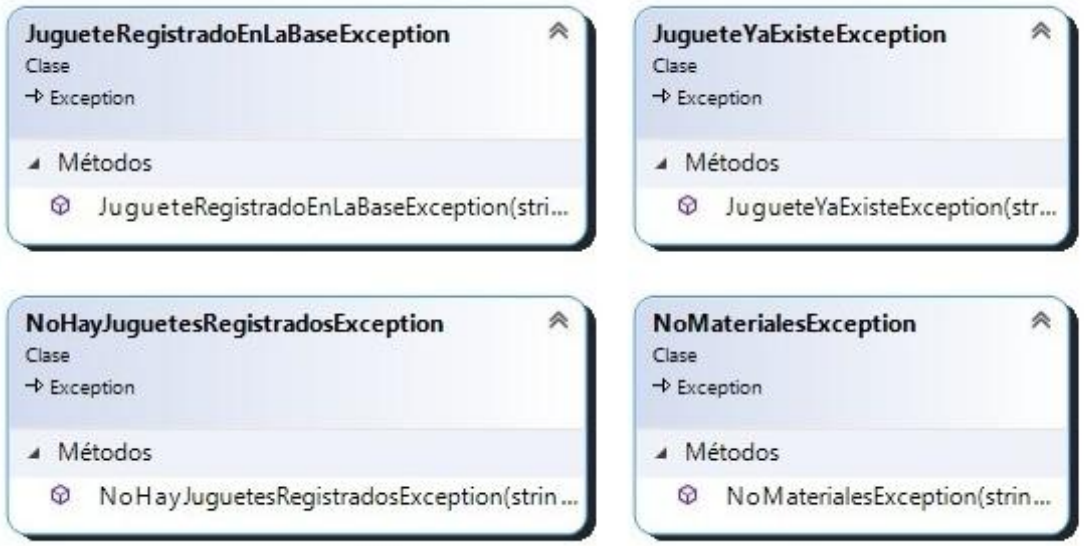
Debido a esto, se realizaron modificaciones lógicas y estructurales del proyecto:

- Se crea la Base de Datos solicitada (adjuntada dentro del proyecto):
  - Dicha Base de Datos cuenta con 6 tablas.
    - Tres de ellas contienen la información de lo que se registra en el momento por el usuario (nombradas como actuales).
    - Las tres restantes cuentan con los datos de los juguetes que ya fueron fabricados (nombradas como historial).
- Al momento de registrar el/los juguetes a ser fabricados, el sistema realiza validaciones previas:
  - Si el mismo juguete ya se ingresó dentro de las tablas actuales, es decir para su próxima fabricación, el sistema se lo indicará al usuario para evitar información duplicada.
  - Si se requiere registrar el mismo juguete que se fabricó en previas situaciones (tablas de historial), el sistema le indicará al usuario que dicho juguete ya fue fabricado previamente, permitiéndole luego agregar dicho registro en las tablas actuales y realizar las modificaciones necesarias.
    - Para dicha funcionalidad, se debe ingresar desde el Menú Principal ⇒ Editar Registros ⇒ Historial Completo y realizar doble click sobre el valor de un campo del registro del DataGrid que se quiere volver a fabricar.
- Además de permitirle comprar la materia prima, actualmente se persiste (en un archivo de extensión .bin) el stock actual de cada material.
- En caso de desearlo, se le permite al usuario modificar ciertos atributos de cada juguete, tanto para los registros manuales, es decir que son cargados en el momento, como también para los registros ingresados desde la tabla de historial.
- Por otro lado, el usuario puede consultar el historial completo de todos los juguetes fabricados e incluso, tal como se describe anteriormente, volver a fabricar el mismo juguete y realizar las modificaciones que sean necesarias en dicho momento.

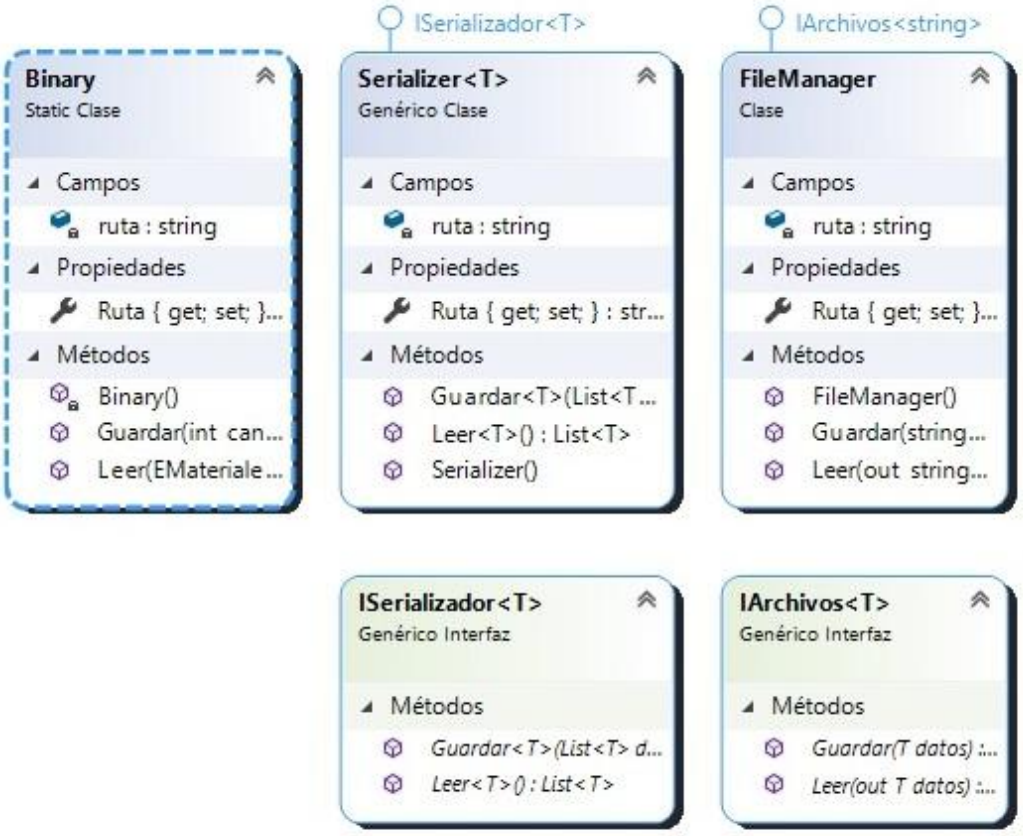
- Luego de fabricar, el sistema limpia las tablas actuales y persiste los datos en las tablas de historial, manteniendo los datos actualizados.
  - En conjunto, se guardarán (en un archivo de extensión xml), los últimos juguetes fabricados.
- Si el usuario realiza un registro incorrecto y no desea fabricar el/los juguetes en cuestión, el sistema le permite eliminarlo de las tablas actuales.
- Por último, el usuario puede consultar la cantidad total de registros en las tablas de historial, discriminado entre peluches, muñecos e inflables.

5. UML actual y aplicación de últimos temas:

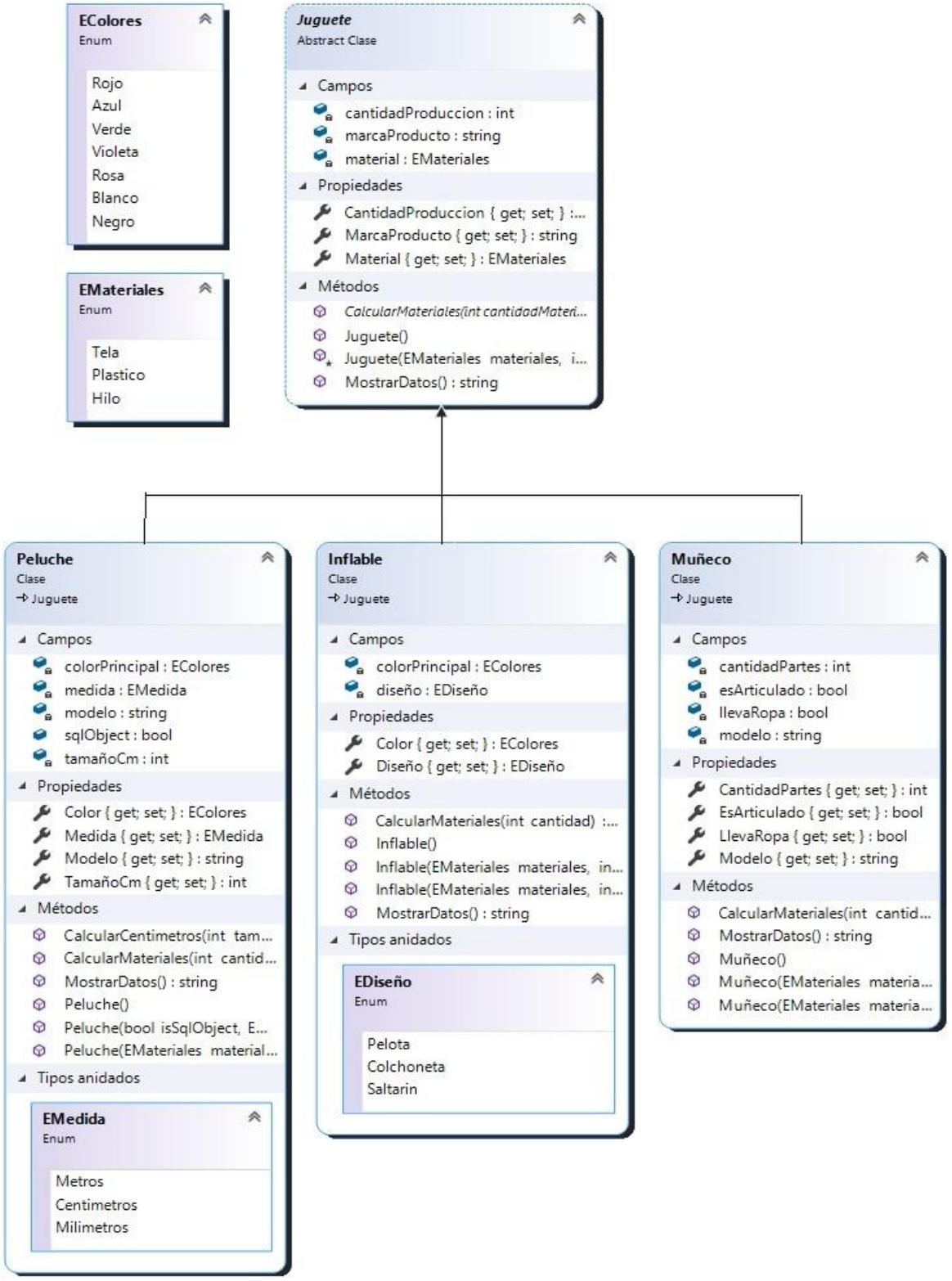
a. Excepciones



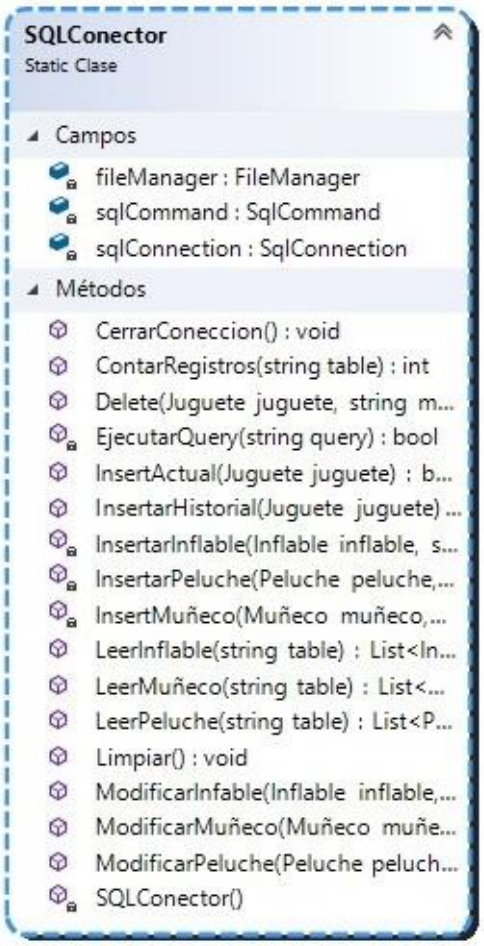
b. Archivos, Serialización e Interfaces



c. Excepciones Pilares POO



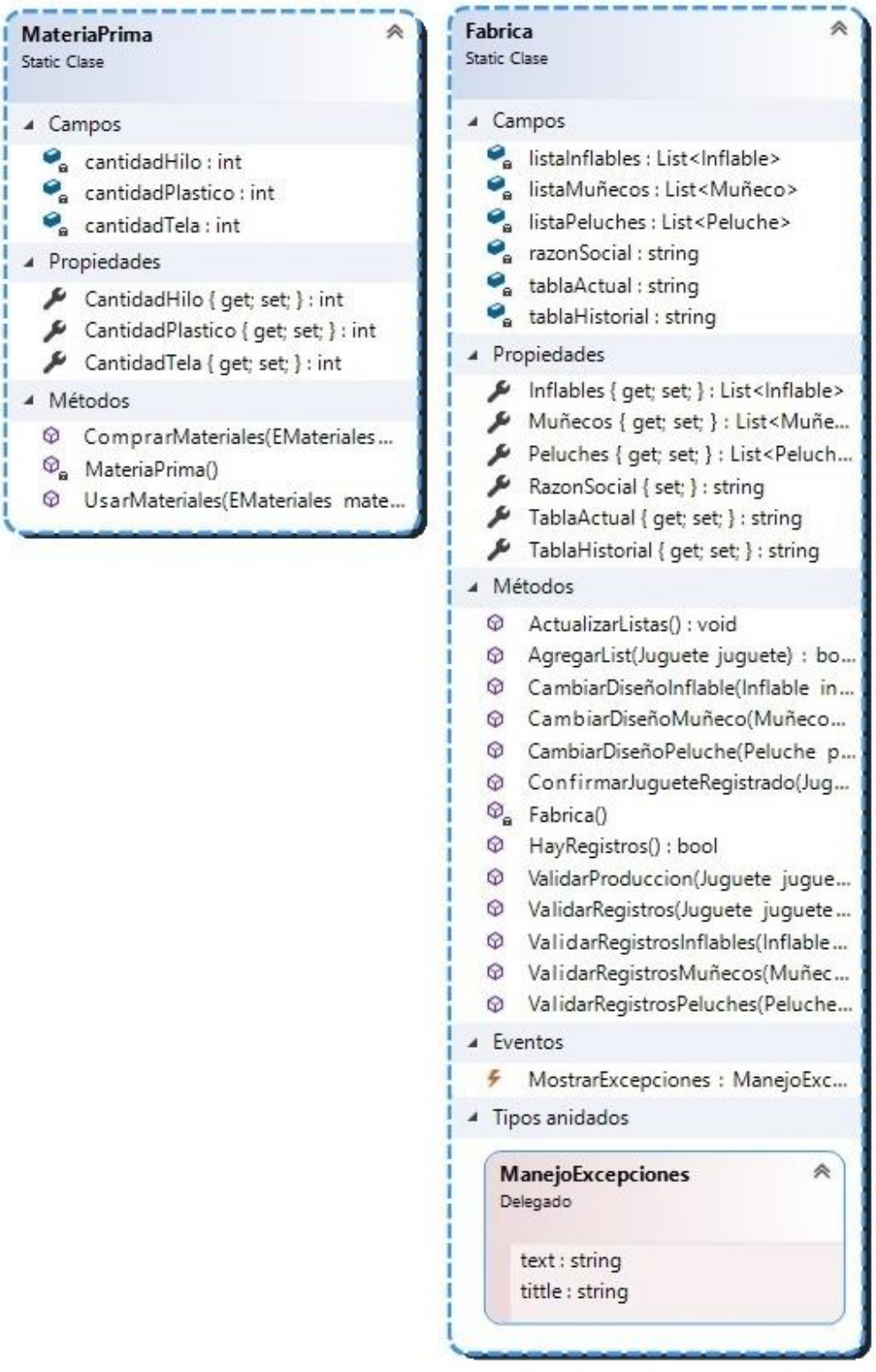
d. SQL / Bases de Datos



- **SQLConector:** clase estática que se conecta con la Base de Datos.
  - **CerrarConexion:** cierra la conexión con la Base de Datos al finalizar la ejecución del programa.
  - **ContarRegistros**: cuenta la cantidad de registros que tiene la tabla ingresada como parámetro.
  - **Delete:** se encarga de eliminar una entidad de la tabla, según la Primary Key compuesta (valores de los parámetros).
  - **EjecutarQuery:** recibe como parámetro la query a ejecutar y retorna si pudo completarse el comando o no con un dato booleano.
  - **InsertActual**: hace un INSERT en las tablas actuales (según el tipo del Juguete que se ingresa como parámetro)
  - **InsertarHistorial:** agrega el juguete ingresado como parámetro en las tablas del historial (en la tabla que corresponda). En caso de que se repita la Primary Key, hace un UPDATE en dicho registro. En caso contrario realiza un INSERT.
  - **InsertarInflable** / **InsertarPeluche** / **InsertMuñeco**: se encargan de hacer el INSERT de la entidad correspondiente en la tabla indicada como parámetro.
  - **LeerPeluche** / **LeerInflable** / **LeerMuñeco**: retorna una lista con todos los registros de las tablas correspondientes para cada entidad.
  - **Limpiar**: hace un TRUNCATE a las tres tablas actuales.

- **ModificarMuñeco** / **ModificarPeluche** / **ModificarInfable**: reciben una instancia del tipo de juguete correspondiente y hace el UPDATE de cada registro en la tabla ingresada como parámetro.

e. <u>Threads</u>



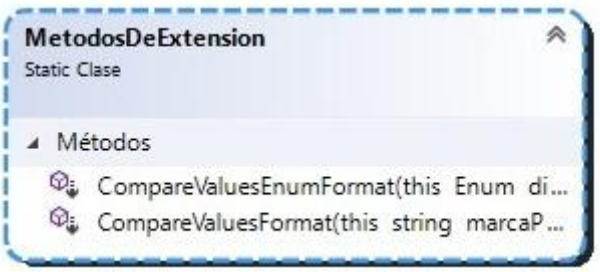
- Durante el uso del programa, se ejecutan (en dos subprocesos / hilos diferentes al principal) los métodos Fabrica.ActualizarListas y Binary.SerializarMateriales. Ambos subprocesos se instancian, comienzan a correr y se agregan a la lista de hilos (listaThreads) dentro del PpalForm.

- **Fabrica.ActualizarListas:** cada 500 milisegundos consulta y obtiene todos los registros de las tablas actuales desde la Base de Datos para mantener constantemente actualizadas las listas de cada juguete en cuestión.
- **Binary.SerializarMateriales:** irá serializando en un archivo binario la cantidad de stock para cada material de la fábrica.

f. Eventos y Delegados

- **ManejoExcepciones**: delegado utilizado para el evento MostrarExcepciones de la clase Fabrica.
- **MostrarExcepciones:** evento que se dispara al momento de producirse una excepción dentro de la clase. Se le asocia el método LanzarErrores (del PpalForm), el cual mostrará un MessageBox de errores con el mensaje y título ingresado como parámetro (setteados al invocar el evento).
- **PasarEntidad:** delegado que en su firma recibe un juguete como parámetro. Se le asigna el método CambiarDiseño en los Formularios de registros de cada juguete, y se invoca al momento de querer modificar los atributos de los mismos. Dicho método insertará cada valor del juguete en los componentes del Formulario.

g. Métodos de extensión



- **CompareValuesFormat:** método de extensión de la clase string. Iguala el formato de los string (a la instancia que llamará al método y al string recibido como parámetro) para comparar sus valores y retornar si son los mismos.
- **CompareValuesEnumFormat:** método de extensión para un Enumerado. Castea su valor a string y compara que tenga el mismo valor que el string recibido como parámetro.

6. Aclaraciones

a. Dentro del proyecto de consola “Test” se prueban las funcionalidades principales del proyecto:

- Creación de los objetos
- Validar que haya materiales para su creación
- Agregarlos dentro de las listas y tablas en la Base de Datos
- Mostrar una excepción al querer agregar nuevamente entidades que tengan las mismas Primary Key
- Modificación de los atributos
- Mostrar los datos actualizados
- Eliminar los objetos.

b. El archivo XML se crea al momento de Fabricar los juguetes. En caso de crearse, se guardará dentro del directorio bin\Debug\Serializado.

c. El archivo de texto se crea al momento de arrojar la primera excepción, guardando todas las excepciones que serán lanzadas durante la ejecución. En caso de crearse, se guardará dentro del directorio bin\Debug\Archivos.

d. El archivo binario se crea y serializa para persiste la cantidad de Materia Prima de la fábrica, siendo leída luego para persistir y mostrar el stock real. Dicho archivo se guardará dentro del directorio bin\Debug\Binario.

e. Se realizan y modifican los test unitarios para distintas funcionalidades desarrolladas, tanto para los casos de éxito como para los casos de fallas (validando que se arroje la excepción deseada).

f. Se adjunta dentro del proyecto el backup de la Base de Datos con el nombre TP.Langer.Denise. En caso de errores o fallas, también se adjunta el script necesario a ejecutar para la creación de la Base de Datos y las tablas.