Universidad Tecnológica Nacional

Facultad Regional Avellaneda

SEGUNDO PARCIAL LAB II – A321 – 2024

Criterios de evaluación

- Sus datos personales deben estar en el nombre de la carpeta principal y la solución: *Apellido.Nombre.SP. Ejemplo: Pérez.Juan.SP*. No se corregirán proyectos sin autor identificable.
- No se corregirán exámenes que no compilen.
- No se corregirán exámenes que posean commits luego de la finalización del parcial.
- Respetar todas las consignas dadas.
- Todas las clases, métodos, atributos, propiedades, etc. Deben ser nombradas exactamente como fue pedido en el enunciado.
- El proyecto no contiene errores de ningún tipo.
- El código compile y se ejecute de manera correcta.
- El código no se encuentre repetido con otro compañero (queda anulado ambos parciales).
- Se deberá reutilizar código cada vez que se pueda, aunque no esté explícitamente en el contenido del texto.
- Se deberá documentar el código según las reglas de estilo de la cátedra.

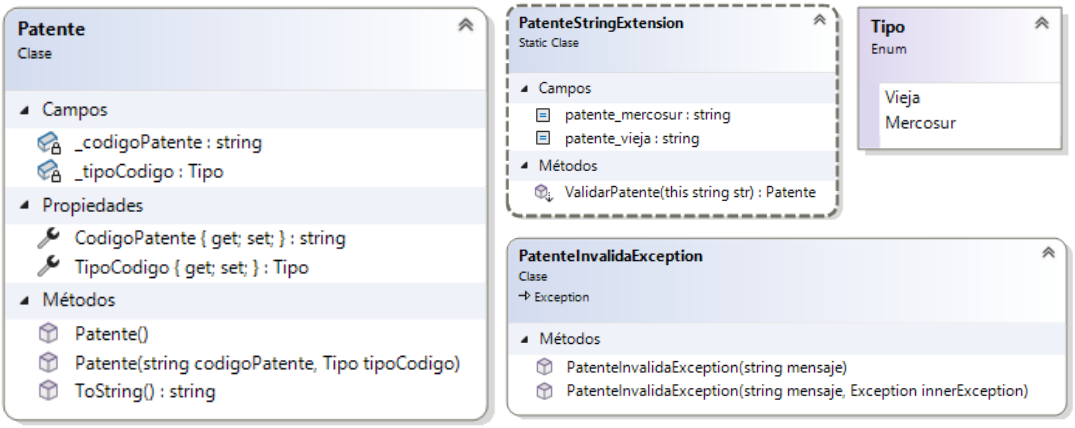
Forma de entrega

- Se deberá subir al repositorio de GitHub declarado.
- Deberá estar correctamente publicado y accesible para su descarga por cualquier método.
- No se corregirán exámenes que no estén correctamente publicados en el GitHub.

Consigna

Realizar las consignas brindadas a partir de la solución entregada. **Modificar el nombre de la solución con el nombre en el siguiente formato: [APELLIDO].[NOMBRE].SP**

Dentro del proyecto **Entidades** se deberá de respetar el siguiente esquema:



Patente (1 punto)

Posee todos sus atributos privados, dos constructores, propiedades públicas y un método.

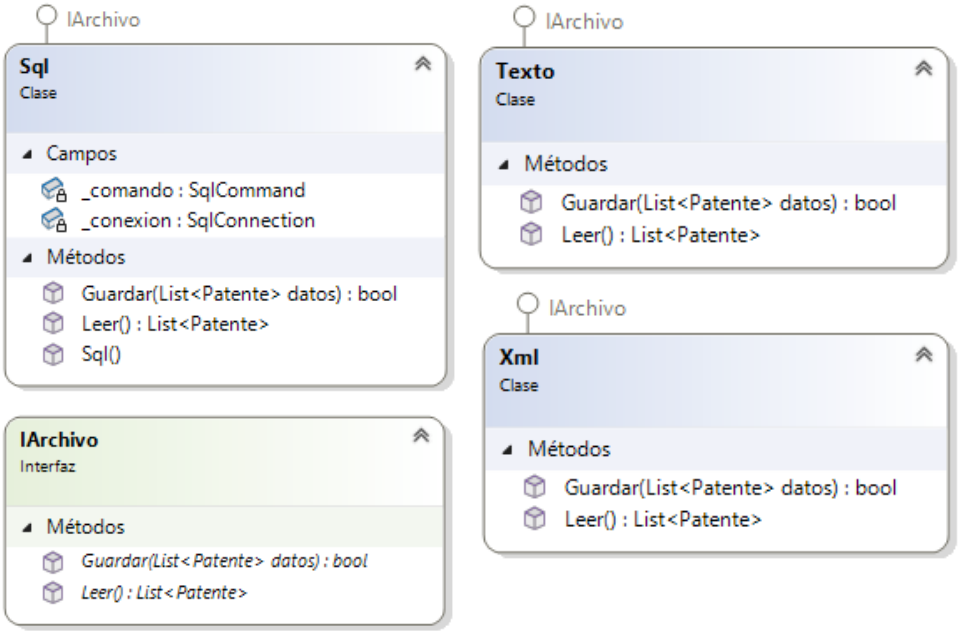
- Ambos constructores son de ***instancia***, uno vacío y un constructor que será el encargado de inicializar sus atributos.
- Las **propiedades** deben ser de lectura y escritura:
  - `CodigoPatente`: Obtiene o establece el código de la patente (`_codigoPatente`).
  - `TipoCodigo`: Obtiene o establece el tipo de código de la patente (`_tipoPatente`).
- **Sobreescritura**:
  - `ToString`. Retornará el código de la patente.

PatenteStringExtension (1 punto)

Dentro de la **clase estática** se deberá realizar un método de ***clase*** llamado `ValidarPatente`.

- `ValidarPatente`
  - `ValidarPatente` recibirá un *string*, y validará que coincida con los formatos de `patente_mercosur` o `patente_vieja`.
  - Si es `patente_mercosur` se deberá generar una nueva patente con tipo `Mercosur`.
  - Si es `patente_vieja` se deberá generar una nueva patente con tipo `Vieja`.
  - Caso contrario se lanzará una excepción del tipo `PatenteInvalidaException` con el mensaje *"{0} no cumple el formato"*.

Dentro del proyecto **Archivos** se deberá de respetar el siguiente esquema:



IArchivo (0.5 puntos)

Crear una interface que posea dos firmas con las siguientes características:

- `Guardar`. Recibirá un parámetro que será una lista de tipo **Patente** y retornará un *bool*.
- `Leer`. Retornará una lista de tipo **Patente**.

Sql (1 punto)

Posee todos sus atributos privados, un constructor y métodos, además implementará la interface **IArchivo**.

- En su único constructor inicializara la conexión a la base de datos que se llamara **lab_sp**, en el atributo `_conexion` establecer la cadena de conexión e inicializar el atributo `_comando`. Colocar que el tipo de comando sea `Text` y establecer la conexión con `_comando`.
- El método `Guardar()` deberá guardar en la base de datos en la tabla *patentes* los datos proporcionados por parámetro. Retornará *true* si se insertó correctamente en la tabla, *false*, caso contrario. Se debe implementar un controlador de **excepciones**.
- El método `Leer()` deberá leer los datos de la tabla *patentes* y retornará la información obtenida. Se debe implementar un controlador de **excepciones**.

Xml (0.5 puntos)

Posee métodos, además implementara la interface **IArchivo**.

- El método `Guardar()` deberá guardar en un archivo XML llamado *patentes.xml* que se almacenará en el Escritorio las patentes que reciba por parámetro. Retornará *true* si se guardó correctamente, *false*, caso contrario. Se debe implementar un controlador de

excepciones.
Environment.GetFolderPath(Environment.SpecialFolder.Desktop)

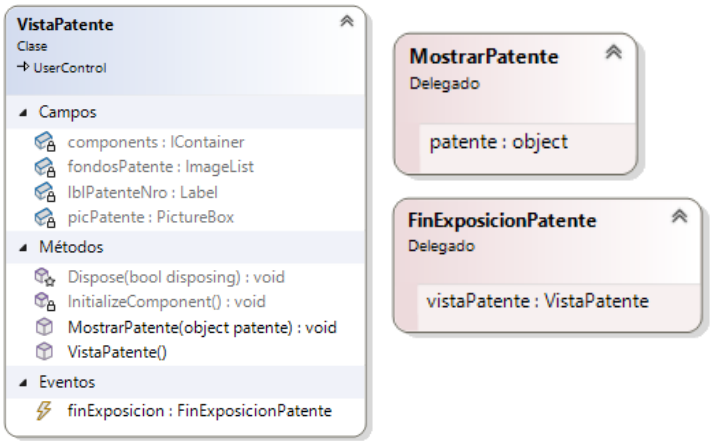
- El método `Leer()` deberá leer el archivo XML *patentes.xml* y retornará el listado de las patentes. Se debe implementar un controlador de **excepciones**.

Texto (0.5 puntos)

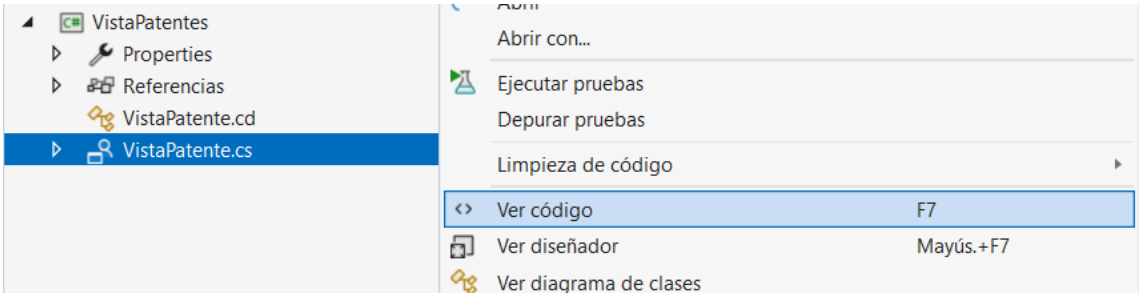
Posee métodos, además implementará la interface **IArchivo**.

- El método `Guardar()` deberá guardar en un archivo llamado *patentes.txt* que se almacenará en el Escritorio las patentes que recibe por parámetro. Si el archivo no existe se debe de crear. Retornará *true* si se guardó correctamente, *false*, caso contrario. Reutilizar código para guardar la patente en el archivo txt.
  Ayuda para llamar a la validación: ReadLine().ValidarPatente()
  Environment.GetFolderPath(Environment.SpecialFolder.Desktop.
- El método `Leer()` deberá leer el archivo *patentes.txt* y retornará el listado de las patentes, utilizar el método `ValidarPatente` para pasar el texto obtenido a un objeto de tipo Patente. Se debe implementar un controlador de **excepciones**.

Dentro del proyecto **VistaPatentes (1 punto)**



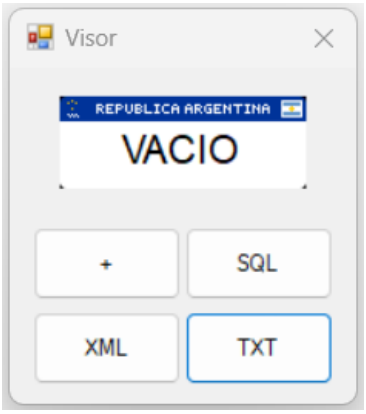
Para realizar las funcionalidades deben de hacer clic derecho en la vista llamada VistaPatente.cs y seleccionar la opción de *Ver código*.



- Agregar dos delegados al **namespace** (por fuera de la clase) con el siguiente formato `void FinExposicionPatente(VistaPatente vistaPatente)` y `void MostrarPatente(object patente)`.
- Dentro del método `MostrarPatente` se deberá lograr que se muestre la patente durante un tiempo X y luego notificar por medio de un evento que finalizó dicha exposición. Completar debajo de los comentarios a fin de lograr la funcionalidad deseada.

Dentro del proyecto **Formulario (3 putos).**

El formulario contará con 4 botones. Al presionar los botones, se leerá de distintos orígenes una colección de patentes y se mostrarán en el control de usuario:



- Declarar un atributo del tipo lista de `Threads` e inicializarlo en el constructor.
- En el evento `Closing` del formulario, asegurarse de que todos los hilos estén terminados. Implementar el método `FinalizarSimulacion` que cumpla con dicha función.
- En el evento `Load` del formulario, asociar al evento `finExposicion` para el objeto `vistaPatente` con `ProximaPatente`.
- En el **botón +** trabajar con los tres objetos tipo Patente que se brindaron y realizar lo siguiente:
  - Guardar los tres objetos en la tabla patentes. Mostrará los siguientes mensajes en un `MessageBox` según corresponda: *"¡Patentes guardadas en la base de datos!"* o *"¡Error al guardar en la base de datos!"*. Usar método `Guardar()`.
  - Guardar los tres objetos en un archivo xml. Mostrará los siguientes mensajes en un `MessageBox` según corresponda: *"¡Patentes guardadas en el archivo xml!"* o *"¡Error al guardar en el archivo xml!"*. Usar método `Guardar()`.
  - Guardar los tres objetos en un archivo. Mostrará los siguientes mensajes en un `MessageBox` según corresponda: *"¡Patentes guardadas en el archivo!"* o *"¡Error al guardar en el archivo!"*. Usar método `Guardar()`.
- `ProximaPatente`, si hay elementos en la lista de patentes:
  - Instanciará un hilo parametrizado para el método `MostrarPatente` del objeto `VistaPatente` recibido.
  - Inicializará el hilo recién creado con el próximo elemento de la lista *(tomar el primero y eliminarlo una vez agregado al hilo).*
  - Agregará el hilo a la lista del atributo `hilos`.
- `IniciarSimulacion`:
  - Finalizará los hilos activos.

    - Llamará al método `ProximaPatente` para cada uno de los objetos del tipo `VistaPatente` del formulario.
- En el **botón SQL** leerá los datos que se encuentren en la tabla *patentes* e inicializará la simulación (`IniciarSimulacion`). Se debe implementar un controlador de **excepciones**.
- En el **botón XML** leerá los datos que se encuentren en el archivo *patentes.xml* e inicializará la simulación (`IniciarSimulacion`). Se debe implementar un controlador de **excepciones**.
- En el **botón TXT** leerá los datos que se encuentren en el archivo *patentes.txt* e inicializará la simulación (`IniciarSimulacion`). Se debe implementar un controlador de **excepciones**.