Universidad Tecnológica Nacional Facultad Regional Avellaneda										
Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos										
Materia: Laboratorio de Programación II										
Apellido:			Fecha:	04-05-2023						
Nombre:			Docente(2):							
División:			Nota(2):							
Legajo:			Firma(2):							
Instancia(1):	**PP**	X	**RPP**		**SP**		**RSP**		**FIN**	

(1) Las instancias validas son: 1er Parcial (**PP**), Recuperatorio 1er Parcial (**RPP**), 2do Parcial (**SP**), Recuperatorio 2do Parcial (**RSP**), Final (**FIN**). Marque con una cruz.

(2) Campos a ser completados por el docente.

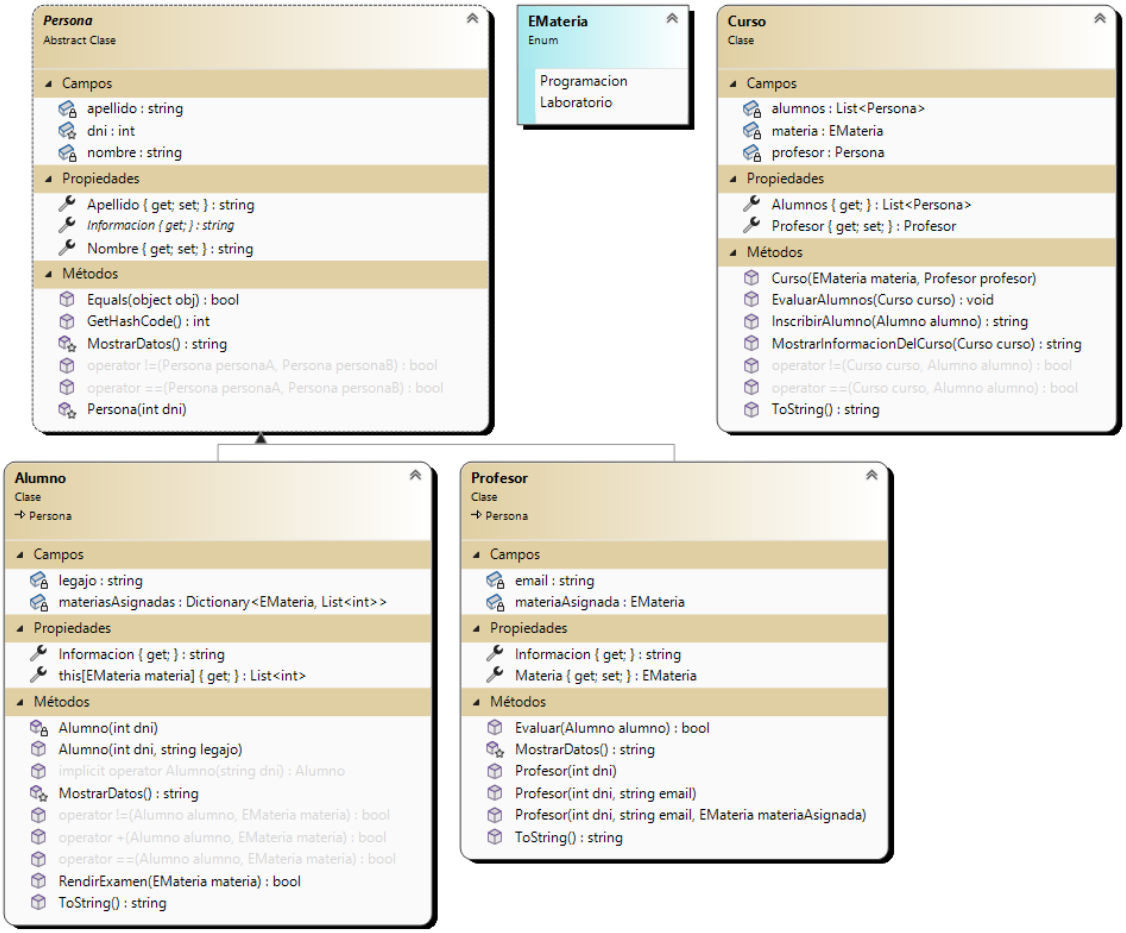
IMPORTANTE:

- **2 (dos) errores en el mismo tema anulan su puntaje.**
- La correcta documentación y reglas de estilo de la cátedra serán evaluadas.
- Colocar sus datos personales en el nombre de la carpeta principal y la solución: Apellido.Nombre.Div. Ej: Pérez.Juan.2C. No sé corregirán proyectos que no sea identificable su autor.
- No se corregirán exámenes que no compilen.
- **Reutilizar** tanto código como crean necesario.
- Colocar nombre de la clase (en estáticos), **this** o **base** en todos los casos que corresponda.
- Aplicar los principios de los 4 pilares de la POO.

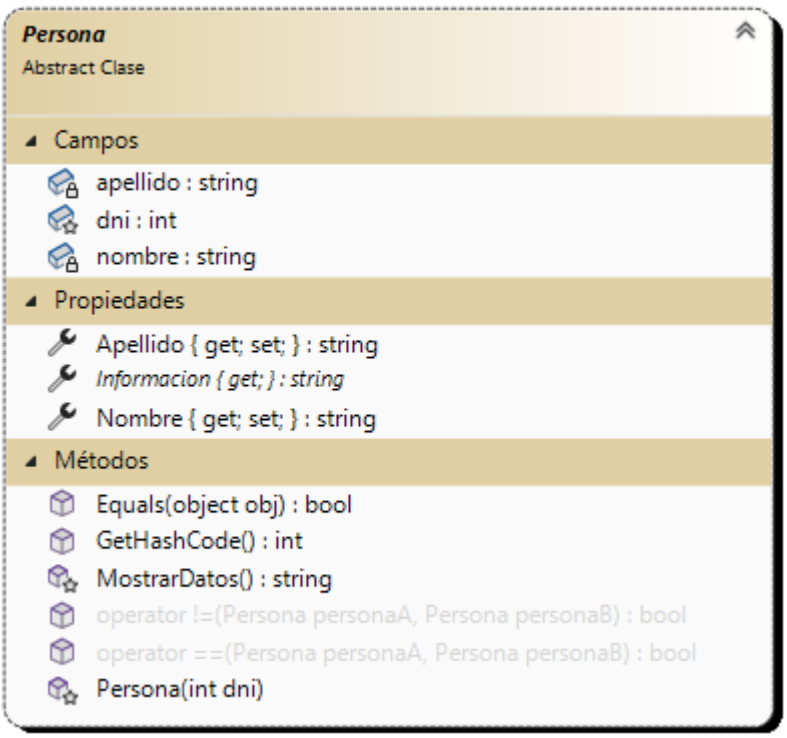
Se desea desarrollar una que gestione la nómina de un centro de atención.

Para ello se debe:

1. Crear un proyecto de tipo biblioteca de clases y con el siguiente esquema:

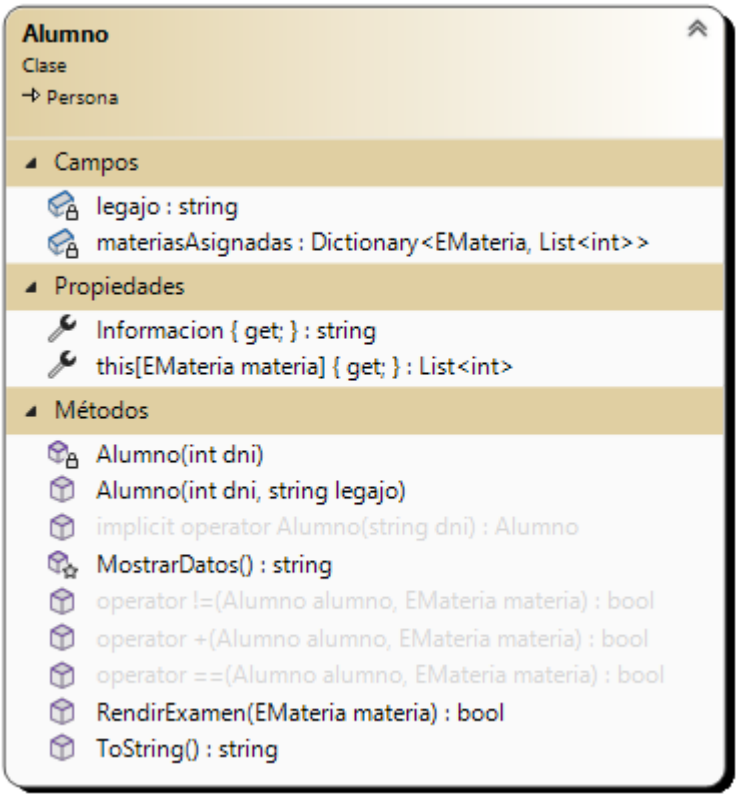


2. Clase Persona:



a. Sera abstracta.
b. Solo el atributo dni será protegido, el resto privados.
c. Su único constructor será protegido.
d. La propiedad Información será abstracta y de solo lectura.
e. Dos personas serán iguales si su dni es el mismo.
f. Sobre escribir Equals a fin de comparar 2 personas por dni.
g. Sobre escribir GetHasCode de la instancia. Retornara el HasCode de dni.
h. Mostrar Datos será protegido y expondrá dni, apellido y nombre de la persona. Utilizar StringBuilder.

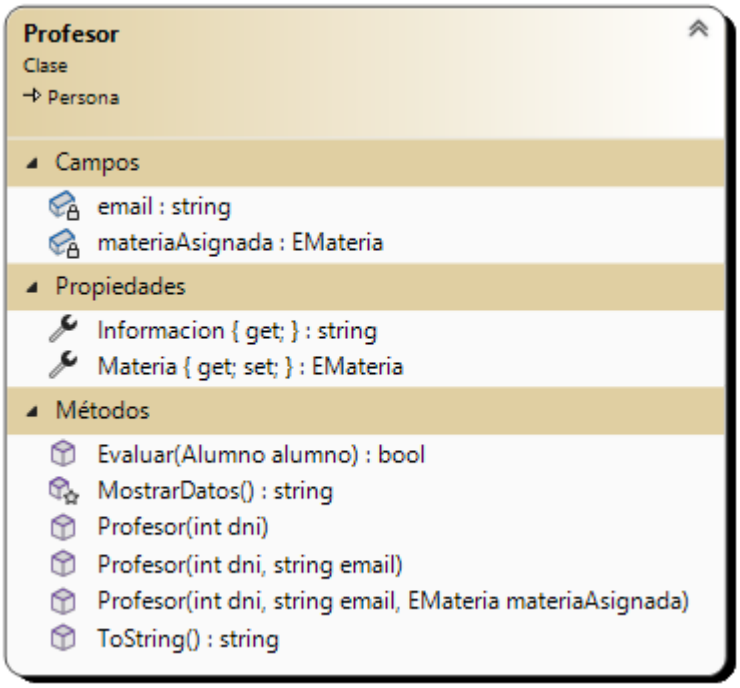
3. **Clase Alumno:**



a. Heredara de Persona. Todos sus atributos son privados.
b. La colección de materias asignadas se instanciará en su constructor privado.
c. Un alumno y una materia serán iguales si el alumno tiene esta materia asignada.
d. Sobrecargar el operador + a fin de que se pueda anotar un alumno a la materia recibida. Se deberá verificar que el alumno no se encuentre previamente inscripto. Si ya se encuentra anotado retornara **false**. De lo contrario se asignará la materia y se retornará **true.**
e. Rendir examen retornara **false** si el alumno no esta anotado a la materia. De lo contrario generara un numero **random** de 1 a 10 y asignara este resultado a las materias asignadas del alumno, retornara **true**.

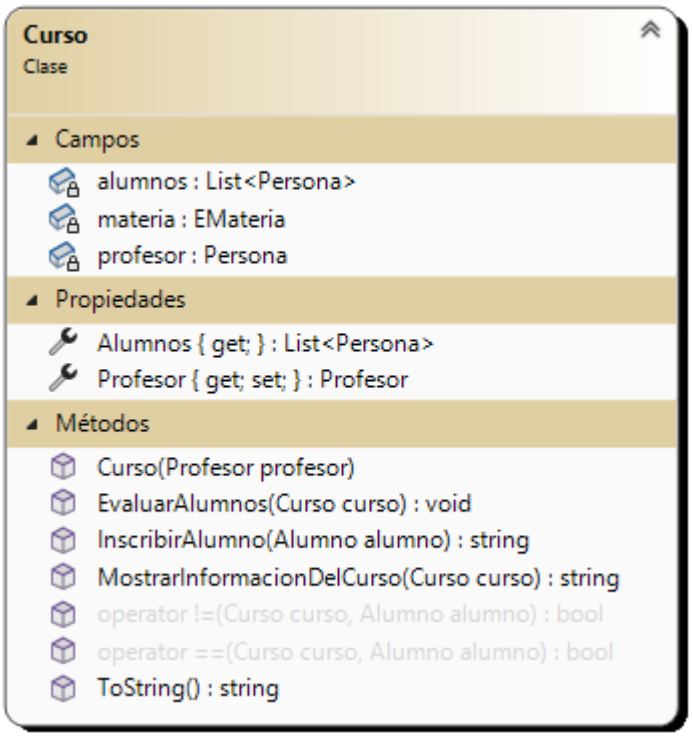
f. Sobrecargar implícitamente un **string** a fin de que se pueda generar una instancia de alumno solo con su **dni**. Asignar como legajo el HasCode de la instancia. Ej: $"L-{alumno.GetHashCode()}"
g. Mostrar datos agregara a los datos de la **persona** el legajo del **alumno** y los nombres de las materias a las cuales se encuentra inscripto.
h. La propiedad información expondrá los datos del alumno, indicando que es un alumno. Ej $"Alumno - {datos}";
i. Indexar la instancia de alumno, la misma retornara una lista con las notas de la materia recibida, si el alumno no esta inscripto a esa materia retornara null.
j. Sobrescribir el toString del alumno y retornar $"Alumno - {this.legajo}"

4. **Clase Profesor:**



a. Heredara de Persona. Todos sus atributos son privados.
b. Mostrar datos agregara a los datos de la **persona** el email del **profesor** y el nombre de la materia a las cual se encuentra asignado.
c. La propiedad información expondrá los datos del profesor, indicando que es un profesor. Ej $"Profesor - {datos}";
d. Sobrescribir el toString del profesor y retornar $"Profesor - {this.materiaAsignada}";
e. El método evaluar recibirá un alumno y lo hará **rendir examen** de la **materia** del profesor.

5. **Clase Curso:**



a. Todos sus atributos serán privados.

b. La materia del curso, seria igual a la materia que tiene asignada el profesor recibido.

c. Su propiedad Alumnos será de solo lectura.

d. Un alumno y un curso serán iguales si este ya se encuentra dentro de la lista de alumnos del curso.

e. El método **inscribir alumno** recibirá un alumno con el cual se verificará que el alumno no se encuentre inscripto al curso y que a su vez se pueda agregarle la materia a este. En caso exitoso lo agregara a la lista de alumnos y retornara $"Se inscribio al alumno a la materia {materia}\n{información del alumno}".

De lo contrario retornara $"Ya inscripto o no se pudo inscribir el alumno a la materia {materia del curso}";

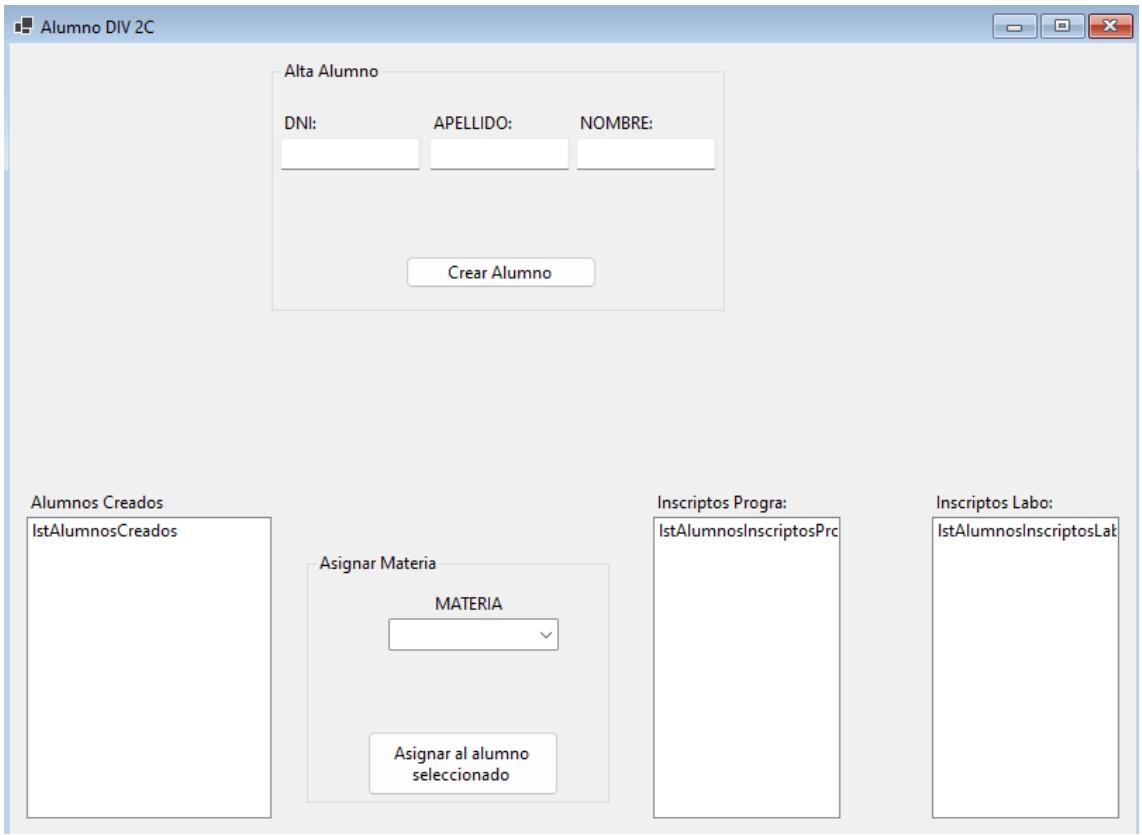
f. Evaluar alumnos será de clase. Recibirá un curso y le tomara examen a todos los alumnos que pertenecen a este.

g. Mostrar información del curso será de clase y mostrar la información del curso junto a la lista de alumno inscriptos.

6. Generar un proyecto de tipo Consola denominado Test y pegar el siguiente código:

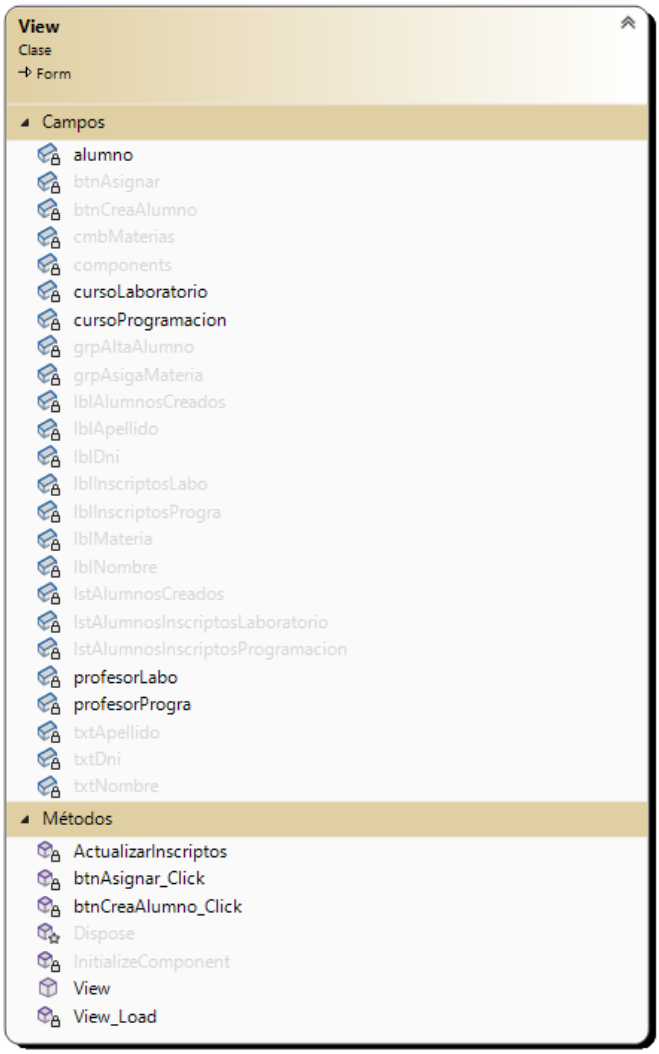
Test.cs

7. Generar un proyecto de tipo Form denominado View con un diseño similar al siguiente:



a. El mismo deberá aparecer centrado en pantalla, no se podrá maximizar ni tampoco cambiar el tamaño de la ventana durante su ejecución.

b. Se deberá respetar las reglas de estilo para los nombre de los controles.



c. Declarar los siguientes atributos como parte de la clase del form:

i. `private Persona alumno;`
ii. `private Persona profesorProgra;`
iii. `private Curso cursoProgramacion;`
iv. `private Persona profesorLabo;`
v. `private Curso cursoLaboratorio`

d. En su constructor colocar:

i. `profesorLabo = new Profesor(99000123, "profesorLabo@email.com", EMateria.Laboratorio);`
ii. `profesorProgra = new Profesor(99000123, "profesorProgra@email.com", EMateria.Programacion);`
iii. `cursoProgramacion = new Curso((Profesor)profesorProgra);`
iv. `cursoLaboratorio = new Curso((Profesor)profesorLabo);`

e. En el evento load del form colocar:

i. `this.cmbMaterias.DataSource = Enum.GetValues(typeof(EMateria));`

f. En el evento click del botón crear alumno colocar:

i. `alumno = (Alumno)this.txtDni.Text;`
ii. `alumno.Nombre = this.txtNombre.Text;`
iii. `alumno.Apellido = this.txtApellido.Text;`
iv. `this.lstAlumnosCreados.Items.Add(alumno);`

g. Generar un método de nominado `ActualizarInscriptos()` y colocar:

i. this.lstAlumnosInscriptosLaboratorio.DataSource = null;
ii. this.lstAlumnosInscriptosPrograma cion.DataSource = null;
iii. this.lstAlumnosInscriptosLaboratorio.DataSource = this.cursoLaboratorio.Alumnos;
iv. this.lstAlumnosInscriptosPrograma cion.DataSource = this.cursoPrograma cion.Alumnos;

h. En el evento click del botón asignar materia colocar el siguiente código:

```
EMateria materia = (EMateria)this.cmbMaterias.SelectedItem;
string mensaje = string.Empty;
if (materia == EMateria.Programacion)
{
   mensaje = cursoProgramacion.InscribirAlumno((Alumno)alumno);
}
else
{
   mensaje = cursoLaboratorio.InscribirAlumno((Alumno)alumno);
}
```