Universidad Tecnológica Nacional Facultad Regional Avellaneda										
Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos										
Materia: Laboratorio de Programación II										
Apellido:			Fecha:	26-04-2022						
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.

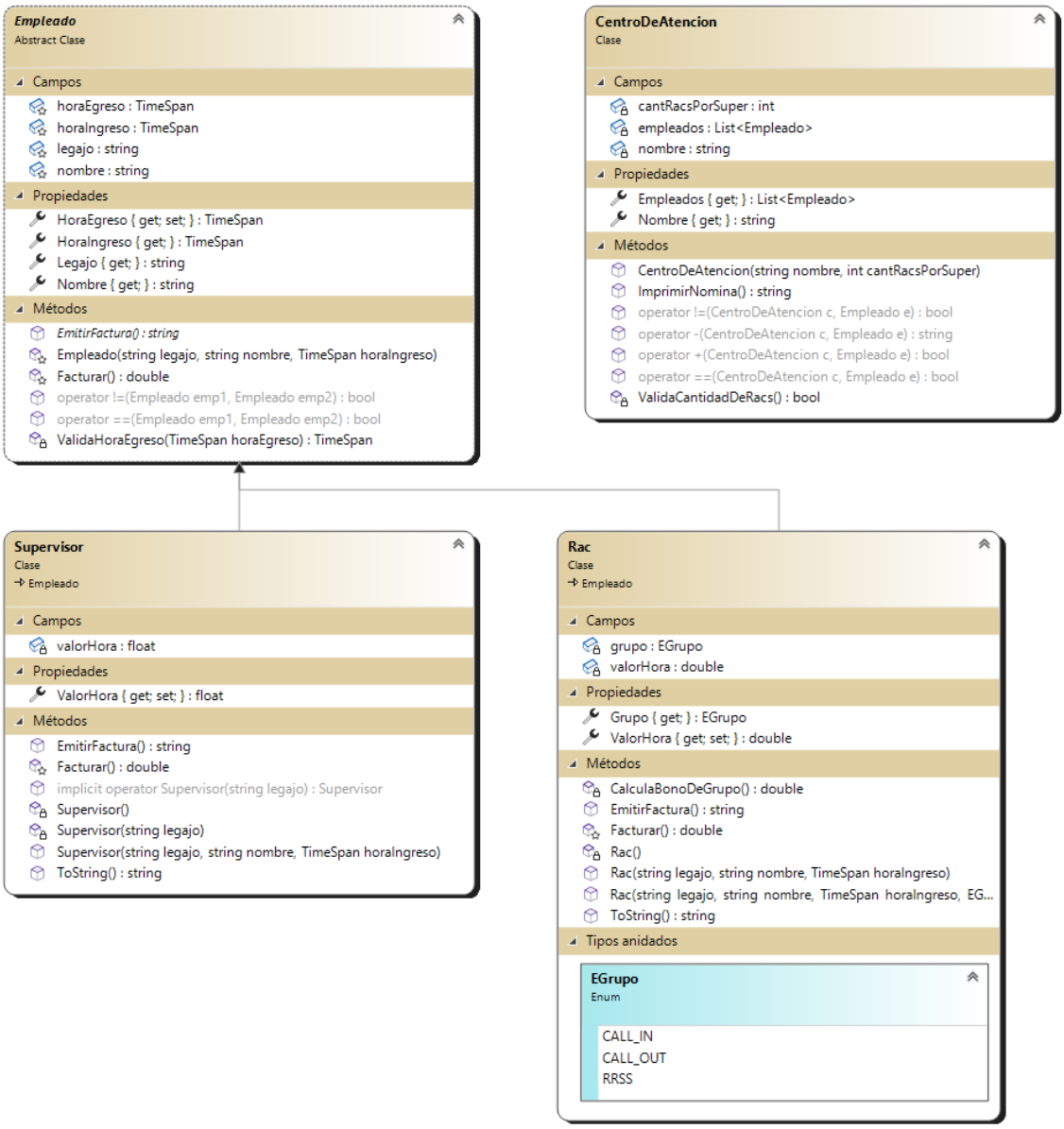
IMPORTANTE:

- **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.2E. 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 Empleado:

a. Sera abstracta.
b. Todos sus atributos, así como su constructor son protegidos.
c. Las propiedades Nombre, Legajo y HoraIngreso serán de solo lectura.
d. La propiedad HoraEgreso, en el **Set** validara el valor recibido (utilizar método ValidaHoraEgreso)
e. El método ValidaHoraEgreso, será privado y se encargara de comparar si el parámetro recibido es mayor que la hora de ingreso de la instancia del empleado. En caso afirmativo retornara el valor recibido, de lo contrario retornara `DateTime.Now.TimeOfDay`.
f. El método Facturar retornara el total de horas trabajas (`TotalHours`) resultante entre la diferencia de la HoraEgreso y HoraIngreso.

g. Las sobrecargas == retornaran True si dos empleados son iguales. Comparar por legajo.

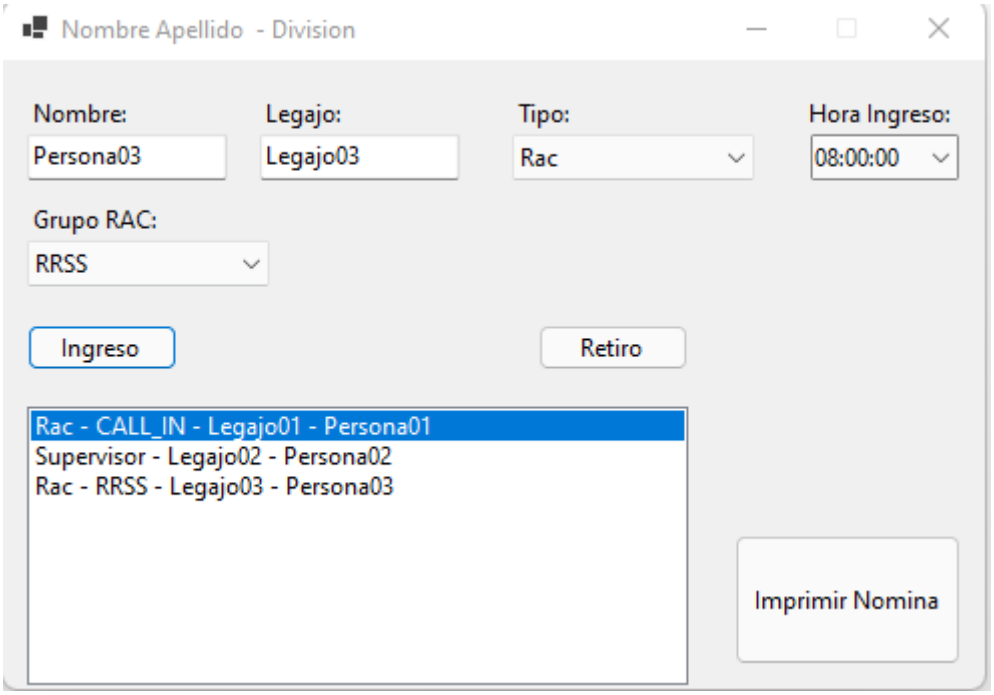
3. **Clase Rac (Representante de atención al cliente):**
   a. Los atributos serán privados y poseerá un enumerado público con los siguientes valores {`CALL_IN, CALL_OUT, RRSS`}.
   b. Por defecto un Rac será del grupo `CALL_IN`.
   c. valorHora será estático y su valor se inicializará en el constructor de clase, siendo su valor inicial `875.90F`.
   d. La propiedad Grupo, será de solo lectura.
   e. La propiedad ValorHora, de clase, en el set solo permitirá asignación de números positivos, de lo contrario mantendrá el valor inicial.
   f. El método EmitirFactura será público y retornara: `$"Factura de: {datos del empleado}\nImporte a facturar: {importe de la facturación}".`
   g. El método CalcularBono retornara:
      i. 0 (cero) para los empleados del grupo CALL_IN.
      ii. 0.1 (cero punto uno) para los empleados del grupo CALL_OUT.
      iii. 0.2 (cero punto dos) para los empleados del grupo RRSS.
   h. El método Facturar será el encargado de multiplicar el total de horas trabajadas por el valor de la hora. El valor de la hora se incrementará porcentualmente según bono del empleado (reutilizar el método calcular bono). Retornar el resultado de la operación.
   i. Sobrescribir el método toString y retornar: `$"{this.GetType().Name} - {grupo} - {legajo} - {nombre}".`
4. **Clase Supervisor:**
   a. Su único atributo será privado y estático. Su valor se inicializara en el constructor de clase, siendo su valor inicial `1025.50F`.
   b. Su constructo privado, permitirá instanciar un supervisor solo con legajo, su nombre será **n/a** y por defecto su hora de ingreso será a las 09 AM (`new TimeSpan(09, 00, 00)`).
   c. La propiedad ValorHora, de clase, en el set solo permitirá asignación de números positivos, de lo contrario mantendrá el valor inicial.
   d. El método EmitirFactura será público y retornara: `$"Factura de: {datos del empleado}\nImporte a facturar: {importe de la facturación}".`
   e. Sobrescribir el método toString y retornar: `$"{this.GetType().Name} - {legajo} - {nombre}".`
   f. El método Facturar será el encargado de multiplicar el total de horas trabajadas por el valor de la hora. Retornar el resultado de la operación.
   g. Sobrecargar de forma implícita los strings, de manera tal que retorne una instancia por defecto de un Supervisor.
5. **Clase CentroDeAtencion:**
   a. Todos sus atributos serán privados. Y sus propiedades de solo lectura.

b. El atributo `cantRacsPorSuper` se inicializará a través del constructor y su función será la de limitar el ingreso de supervisores según la cantidad de Racs.
c. La lista se inicializará en el constructor.
d. La sobrecarga del operador == verificara si un empleado se encuentra en lista de empleados, en caso afirmativo retornara True y de lo contrario False.
e. El método ValidaCantidadDeRacs será privado y retornará un booleano. Retornara True si la cantidad de Racs en la lista es mayor a la cantidad de supervisores por Racs permitido. *Ej: 1 supervisor cada 5 Racs*.
f. La sobrecarga del operador + agregara empleados a la colección del centro de atención. Estos se podrán agregar solo si:
    i. El empleado no existe.
    ii. En caso de ser un supervisor, solo se podrán agregar si la cantidad de Racs lo habilita.
g. La sobrecarga del operador – retornara un string. Si el empleado no existe informara `"Empleado no encontrado".` De lo contrario, asignara al empleado una hora de Egreso (`DateTime.Now.TimeOfDay`) y el mensaje a retornar será la factura emitida por el empleado. Luego remueve el empleado de la lista.
h. El método impimir nonima, retornara la lista completa de empleados. Utilizar StringBuilder.

6. Crear un proyecto de consola denominado Test y probar el siguiente código:

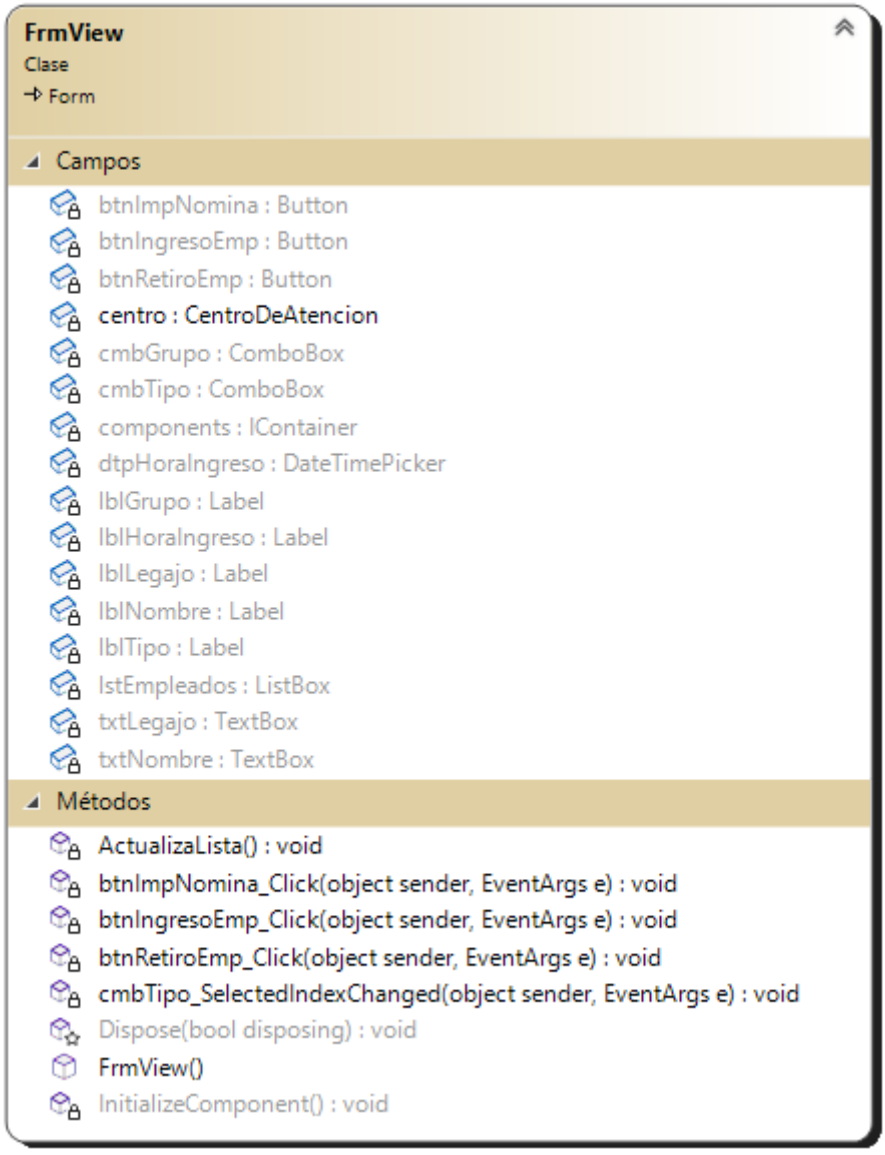
20220426-PP-Codigo Test.docx

7. Crear un proyecto de formulario con un diseño similar:

a. 

b. Este se llamará FrmView y poseerá un atributo privado de tipo CentroAtencion.
c. El diseño del form deberá de respetar las siguientes condiciones:
    i. El formulario deberá de ejecutarse centrado en pantalla, se le debería de quitar la posibilidad de cambiar el tamaño, así como
d. Poseerá los siguientes controles:
    i. 2 TexBox, uno para nombre y otro para legajo.
    ii. 2 ComboBox, uno para tipo de empleado y otro para el grupo de Racs. En ambos modificar la propiedad *DropDownStyle* a **DropDownList**.
    iii. 1 DateTimePicker para la hora de ingreso del empleado. Modificar la propiedad *Format* a **Time**.
    iv. 1 ListBox que contendrá la lista de empleados agregados a la colección.
    v. 3 Buttons, uno para ingresar empleados, para retirar y el ultimo para imprimir la nomina del centro de atención.
e. Respetar las siguientes denominaciones:

i.



f. ***<u>En el constructor del Formulario colocar el siguiente código</u>*:**

```
this.cmbGrupo.DataSource = Enum.GetValues(typeof(Rac.EGrupo));
this.cmbTipo.DataSource = new List<string> {"Supervisor", "Rac"};
this.centro = new CentroDeAtencion("Teleperformance", 3);
```

g. ***<u>En el evento SelectIndexChange del ComboBox Tipo colocar el siguiente código:</u>***

```
            if(this.cmbTipo.SelectedItem.ToString() == "Rac")
            {
                this.lblGrupo.Visible = true;
                this.cmbGrupo.Visible = true;
            }
            else
            {
                this.lblGrupo.Visible = false;
                this.cmbGrupo.Visible = false;
            }
```

h. ***Crear un método privado que retorne void denominado ActualizaLista y colocar el siguiente código*:**

```
            this.lstEmpleados.DataSource = null;
            this.lstEmpleados.DataSource = centro.Empleados;
```

i. ***En el evento Click del botón Ingreso empleados colocar el siguiente código:***

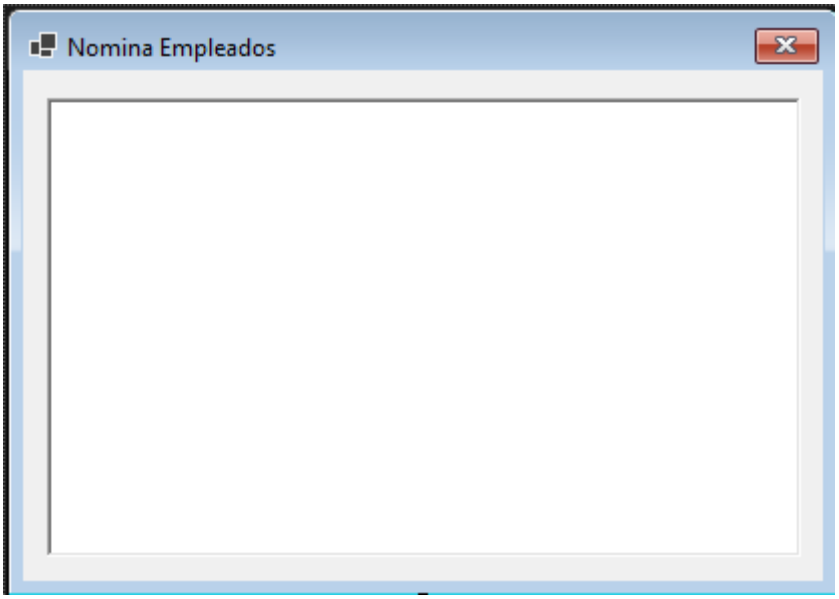
```
            Empleado empleado;
            string mensaje = string.Empty;
            if(this.cmbTipo.SelectedItem.ToString() == "Rac")
            {
                empleado = new Rac(this.txtLegajo.Text,
txtNombre.Text,
this.dtpHoraIngreso.Value.TimeOfDay,(Rac.EGrupo)this.cmbGrupo.Selecte
dItem);
            }
            else
            {
                empleado = new Supervisor(this.txtLegajo.Text,
txtNombre.Text, this.dtpHoraIngreso.Value.TimeOfDay);
            }

            if (centro + empleado)
            {
                mensaje = "Se agrego nuevo empleado";
            }
            else
            {
                mensaje = "No se puedo agregar el empleado";
            }
            MessageBox.Show(mensaje, "Informacion",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            this.ActualizaLista();
```

j. ***En el evento click del botón retiro empleados colocar el siguiente código:***

```
            Empleado? empSeleccionado =
this.lstEmpleados.SelectedItem as Empleado;
            if (empSeleccionado is not null)
            {
                MessageBox.Show(this.centro-
empSeleccionado,"Informacion Salida", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
            }
            this.ActualizaLista();
```

8. Para lograr el 100% de satisfacción de nuestro centro de atención desarrollar un segundo formulario como el siguiente:



   a. El mismo poseerá un rich text box que mostrará la información del centro de atención al presionar el botón imprimir nomina desde el Formulario principal.