Universidad Tecnológica Nacional Facultad Regional Avellaneda										
Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos										
Materia: Laboratorio de Programación II										
Apellido:			Fecha:	27-05-2021						
Nombre:			Docente(2):							
División:	2°C		Nota(2):							
Legajo:			Firma(2):							
Instancia(1):	**PP**		**RPP**	X	**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.

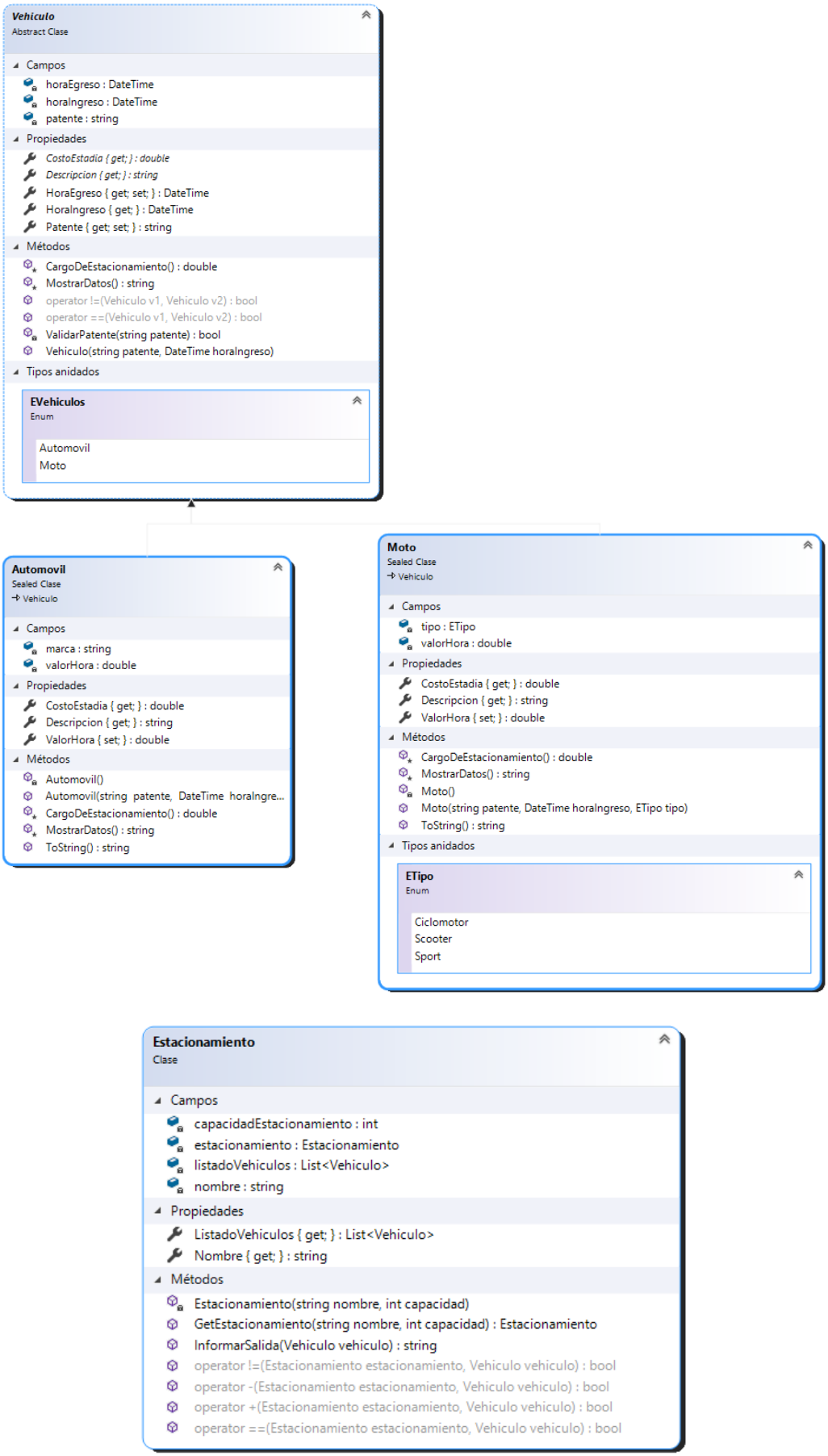
(2) Campos a ser completados por el docente.

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.2D. 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 aplicación que permita controlar el ingreso y egreso de vehículos a un estacionamiento. Para ello se debe:

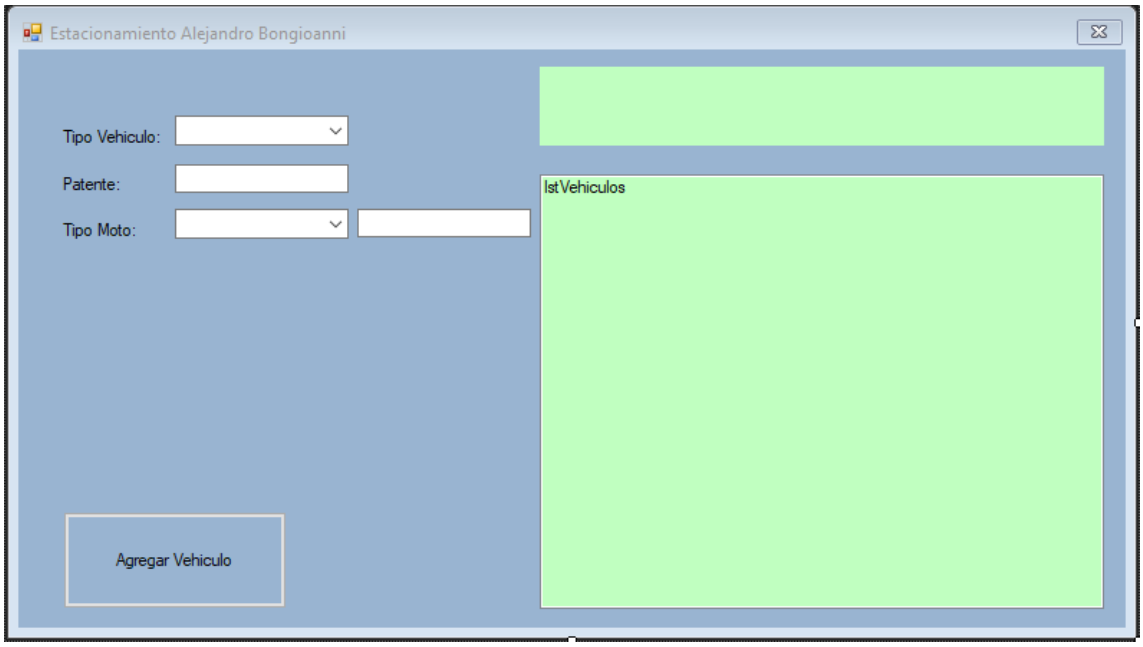
1. Crear un proyecto de tipo biblioteca de clases y colocar el siguiente esquema:


Vehiculo
Abstract Clase
Campos
horaEgreso : DateTime
horaIngreso : DateTime
patente : string
Propiedades
CostoEstadia { get; } : double
Descripcion { get; } : string
HoraEgreso { get; set; } : DateTime
HoraIngreso { get; } : DateTime
Patente { get; set; } : string
Métodos
CargoDeEstacionamiento() : double
MostrarDatos() : string
operator !=(Vehiculo v1, Vehiculo v2) : bool
operator ==(Vehiculo v1, Vehiculo v2) : bool
ValidarPatente(string patente) : bool
Vehiculo(string patente, DateTime horaIngreso)
Tipos anidados
EVehiculos
Enum
Automovil
Moto
Automovil
Sealed Clase
Vehiculo
Campos
marca : string
valorHora : double
Propiedades
CostoEstadia { get; } : double
Descripcion { get; } : string
ValorHora { set; } : double
Métodos
Automovil()
Automovil(string patente, DateTime horaIngre...
CargoDeEstacionamiento() : double
MostrarDatos() : string
ToString() : string
Moto
Sealed Clase
Vehiculo
Campos
tipo : ETipo
valorHora : double
Propiedades
CostoEstadia { get; } : double
Descripcion { get; } : string
ValorHora { set; } : double
Métodos
CargoDeEstacionamiento() : double
MostrarDatos() : string
Moto()
Moto(string patente, DateTime horaIngreso, ETipo tipo)
ToString() : string
Tipos anidados
ETipo
Enum
Ciclomotor
Scooter
Sport
Estacionamiento
Clase
Campos
capacidadEstacionamiento : int
estacionamiento : Estacionamiento
listadoVehiculos : List<Vehiculo>
nombre : string
Propiedades
ListadoVehiculos { get; } : List<Vehiculo>
Nombre { get; } : string
Métodos
Estacionamiento(string nombre, int capacidad)
GetEstacionamiento(string nombre, int capacidad) : Estacionamiento
InformarSalida(Vehiculo vehiculo) : string
operator !=(Estacionamiento estacionamiento, Vehiculo vehiculo) : bool
operator -(Estacionamiento estacionamiento, Vehiculo vehiculo) : bool
operator +(Estacionamiento estacionamiento, Vehiculo vehiculo) : bool
operator ==(Estacionamiento estacionamiento, Vehiculo vehiculo) : bool

2. Clase **Vehículo:**
    a. Sera abstracta.
    b. Poseerá los atributos: patente, horaIngreso, horaEgreso. Todos Privados. El enumerado de la clase será público y tendrá los ítems {Automovil, Moto}.
    c. El Constructor de Vehículo recibirá la patente y la hora de ingreso. La asignación de Patente se realizará desde la propiedad.
    d. Las propiedades Descripcion y CostoEstadia en Vehículo, serán abstractas y de solo lectura.
    e. HoraEgreso, será de lectura y escritura. Se deberá validar que esta no sea inferior a la Hora de Ingreso.
    f. La propiedad Patente, será de lectura y escritura. La asignación se hará previa validación, considerando que:
        i. Posea una longitud entre 6 y 7 caracteres.
    g. ValidarPatente será privada.
    h. CargoEstacionamiento es protegido y en Vehículo retornara la cantidad de horas de la estadía del vehículo, obtenidos entre la hora de ingreso y egreso de este.
    i. MostrarDatos será protegido y expondrá la patente y la hora de ingreso del vehículo. Utilizar StringBuilder.
    j. Dos Vehiculos serán iguales si poseen la misma patente.
3. Clase **Automóvil**:
    a. Hereda de Vehículo.
    b. Sus atributos marca y valorHora, serán privados. valorHora será de clase y su valor se inicializará en el constructor de clase, siendo su valor 120.
    c. El Constructor de Instancia recibirá: La patente, hora de ingreso y la marca del Automovil.
    d. La propiedad ValorHora será de Clase y solo escritura, desde allí se podrá cambiar el valor de la hora para Automóvil, siempre que el valor recibido sea positivo.
    e. La propiedad CostoEstadia, retornara el cargo del estacionamiento para el Automóvil.
    f. El método CargoEstacionemiento, será protegido y retornará el cargo del Automóvil, resultante de multiplicar las horas de la estadía por el valor de la hora.
    g. La propiedad Descripcion retornara la marca del Automóvil.
    h. MostrarDatos será protegido, agregara información indicando que es un "****AUTOMOVIL*****" y la descripción de este.
    i. ToString hará públicos los datos de Automóvil.
4. Clase **Moto**:
    a. Hereda de Vehículo.
    b. Tendrá un enumerado de ETipo con los ítems: {Ciclomotor, Scooter,Sport}
    c. Sus atributos tipo y valorHora, serán privados. valorHora será de clase y su valor se inicializará en el constructor de clase, siendo su valor 100.
    d. El Constructor de Instancia recibirá: La patente, hora de ingreso y el tipo de Moto.
    e. La propiedad ValorHora será de Clase y solo escritura, desde allí se podrá cambiar el valor de la hora para Moto, siempre que el valor recibido sea positivo.

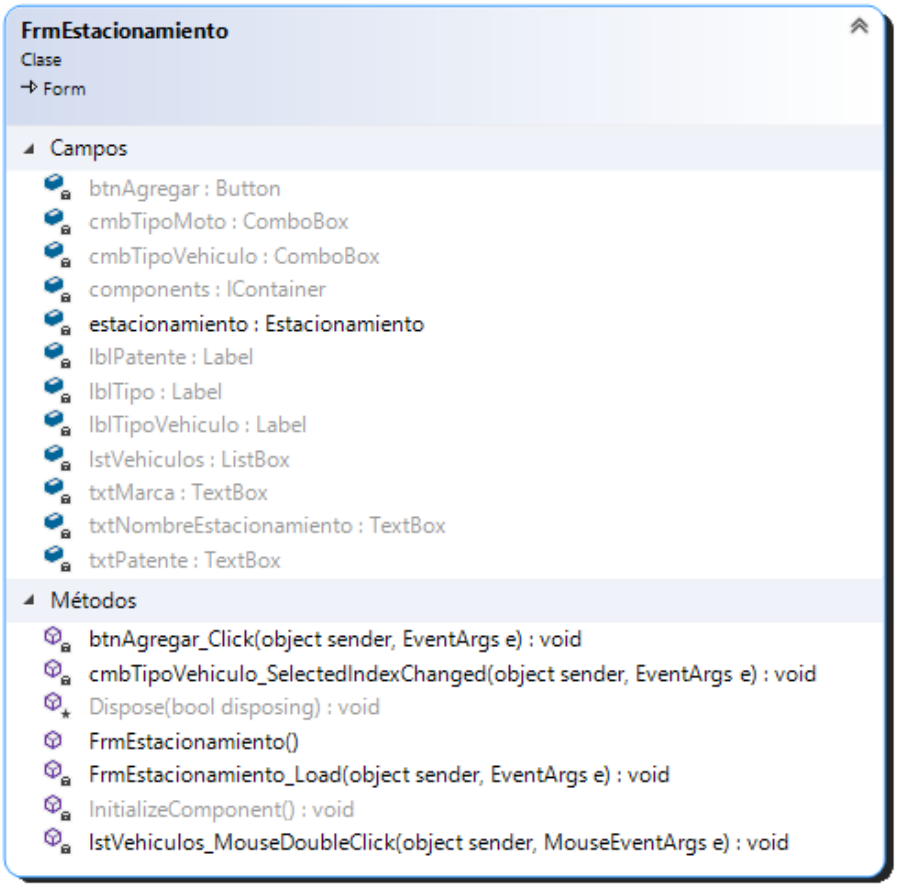
f. La propiedad CostoEstadia, retornara el cargo del estacionamiento para la Moto.
g. El método CargoEstacionamiento, será protegido y retornará el cargo de la Moto, resultante de multiplicar las horas de la estadía por el valor de la hora.
h. La propiedad Descripcion retornara el tipo de Moto.
i. MostrarDatos será protegido, agregara información indicando que es una "****MOTO*****" y la descripción de este.
j. ToString hará públicos los datos de Moto.

5. Clase **Estacionamiento**:
    a. Todos sus atributos serán privados.
    b. El atributo estacionamiento será de clase.
    c. El único constructor será privado y se encargará de inicializar la lista de vehículos, asignar un nombre al estacionamiento y definir su capacidad.
    d. GetEstacionamiento será de clase e implementará un patrón singleton para lo cual deberá:
        i. Si la variable estacionamiento es null, instanciar el objeto.
        ii. Si no es null, modificara la capacidad del estacionamiento.
        iii. En ambos casos, su última acción será retornar el objeto estacionamiento.
    e. La propiedad ListadoVehiculos será de solo lectura y retornará la lista de vehículos del estacionamiento.
    f. InformarSalida será de instancia, recibirá un Vehículo y retornará una cadena que informará:
        i. El nombre del Estacionamiento.
        ii. Los datos del vehículo.
        iii. La hora de salida.
        iv. El cargo del estacionamiento.
    g. Un Estacionamiento y un Vehículo serán iguales, si el Vehículo se encuentra en el estacionamiento.
    h. La sobrecargar del operador + (mas) permitirá agregar un Vehículo al Estacionamiento, siempre y cuando haya espacio disponible y el vehículo no se encuentre en él.
    i. La Sobrecarga del operador – (menos) permitirá retirar un Vehículo del estacionamiento, si es que este se encuentra en él. Antes de remover se deberá asignar una hora de Egreso al vehículo, usar DateTime.Now.
6. Formulario:
    a. Diseñar un formulario como el siguiente que inicie centrado, no permita minimizar, maximizar, ni modificar su tamaño. Se deberá ajustar las propiedades:
        i. StartPosition en Center.
        ii. MinimizeBox y MaximizeBox en False.
        iii. FormBorderStyle en FixedSingle.

b. 

c. Poseerá:
   i. Un botón que permitirá agregar vehículos al estacionamiento.
   ii. Un ComboBox Para elegir entre Vehiculos de Tipo Moto o Automóvil.
   iii. Un ComboBox para el tipo de Moto.
   iv. Un TextBox donde se ingresará la patente.
   v. Un TextBox que Mostrara el nombre del estacionamiento.
   vi. Un Label que será dinámico. Dependiendo el tipo de Vehículo que se seleccione. Su texto Cambiara entre Tipo Moto, si el tipo de Vehículo seleccionado es una Moto o Marca si es un Automóvil.
   vii. Un TextBox donde se ingresará la Marca si el Vehículo es de tipo Automóvil.
   viii. Un ListBox donde se agregarán los vehículos que ingresen al estacionamiento.

ix.



- El Evento Load tendrá el siguiente código:

```
this.cmbTipoVehiculo.DataSource =Enum.GetValues(typeof(Vehiculo.EVehiculos));
this.cmbTipoMoto.DataSource = Enum.GetValues(typeof(Moto.ETipo));
this.estacionamiento = Estacionamiento.GetEstacionamiento("Nombre del Alumno",
20);
this.txtNombreEstacionamiento.Text = this.estacionamiento.Nombre;
```

- El Evento SelectedIndexChanged del ComboBox TipoVehiculo tendrá el siguiente código:

```
if((Vehiculo.EVehiculos)this.cmbTipoVehiculo.SelectedItem ==
Vehiculo.EVehiculos.Automovil)
{
            this.cmbTipoMoto.Visible = false;
            lblTipo.Text = "Marca:";
            this.txtMarca.Location = this.cmbTipoMoto.Location;
            this.txtMarca.Visible = true;
}
else
{
            this.cmbTipoMoto.Visible = true;
            lblTipo.Text = "Tipo Moto:";
            this.txtMarca.Visible = false;
}
```

- El Evento Click del botón Agregar tendrá el siguiente código:

```
Vehiculo vehiculo;
if((Vehiculo.EVehiculos)this.cmbTipoVehiculo.SelectedItem ==
Vehiculo.EVehiculos.Automovil)
{
 vehiculo = new Automovil(this.txtPatente.Text, DateTime.Now,this.txtMarca.Text);
}
else
{
 vehiculo = new
Moto(this.txtPatente.Text,DateTime.Now,(Moto.ETipo)this.cmbTipoMoto.SelectedItem)
;
}

if(this.estacionamiento+vehiculo)
{
 this.lstVehiculos.Items.Add(vehiculo);
 MessageBox.Show(vehiculo.ToString(), "Ingreso al
Estacionamiento",MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

- El Evento MouseDobleClick del ListBox tendrá el siguiente código:

```
if (this.estacionamiento - (Vehiculo)this.lstVehiculos.SelectedItem)
{
MessageBox.Show(this.estacionamiento.InformarSalida((Vehiculo)this.lstVehiculos.S
electedItem), "Ingreso al Estacionamiento", MessageBoxButtons.OK,
MessageBoxIcon.Information);
this.lstVehiculos.Items.Remove((Vehiculo)this.lstVehiculos.SelectedItem);
}
```