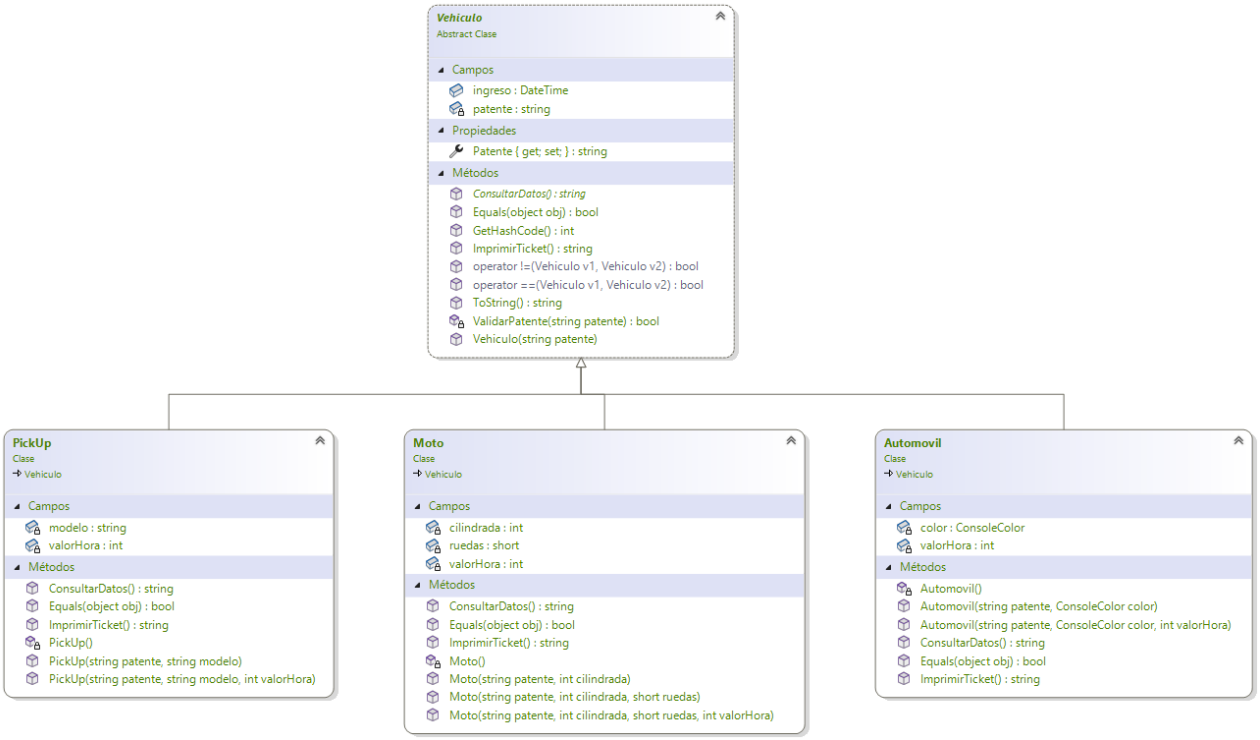
Segundo Parcial POO

1. Crear un proyecto de biblioteca de clases con el nombre **JohnDoePOO.Entidades,** donde John Doe debe ser reemplazado por su ***nombre y apellido***.



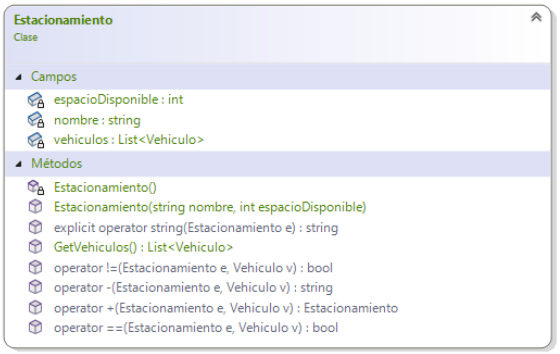
1.1. Clase Vehiculo:
   1.1.1. En el ***constructor***, la fecha de ingreso será cargada 3 horas previas a la creación del objeto mediante el método ***DateTime.Now.AddHours(-3)***.
   1.1.2. La propiedad ***Patente*** validará que la misma tenga el formato de patentes (AAA999 o AA999AA), caso contrario no la asignará. (***Resuelto en primer parcial***)
   1.1.3. ***ConsultarDatos*** no tendrá implementación en Vehiculo.
   1.1.4. ***ToString*** retornará la Patente del Vehiculo con el siguiente formato: "Patente {0}".
   1.1.5. ***ImprimirTicket*** podrá ser anulado en las clases derivadas. Utilizará ***StringBuilder*** para retornar los datos del Vehiculo (reutilizar ToString) y la fecha y hora de ingreso.
   1.1.6. Dos Vehiculos serán iguales si tienen la misma patente y son objetos de la misma clase (utilizar el ***Equals*** sobreescrito en las clases derivadas según corresponda).

1.2. Clase Moto:
   1.2.1. El ***constructor*** de clase asignará el valor hora en 30.
   1.2.2. Por defecto, las ruedas serán 2 y se cargarán en el/los constructores que corresponda.
   1.2.3. ***ConsultarDatos*** retornará todos los datos de la Moto, junto con su tipo.

1.3. Clase PickUp:
   1.3.1. El ***constructor*** de clase asignará el valor hora en 70.
   1.3.2. ***ConsultarDatos*** retornará todos los datos de la PickUp, junto con su tipo.

1.4. Clase Automovil:
   1.4.1. El ***constructor*** de clase asignará el valor hora en 50.
   1.4.2. ***ConsultarDatos*** retornará todos los datos del Automovil, junto con su tipo.

2. Crear un proyecto de biblioteca de clases con el nombre **JohnDoePOO.Datos**, donde John Doe debe ser reemplazado por su ***nombre y apellido***.



2.1. Clase Estacionamiento

2.1.1. El ***constructor*** privado será el único lugar donde se instanciará la lista de Vehiculos.

2.1.2. El ***operador explícito*** retornará la información del Estacionamiento, los lugares disponibles, y los Vehiculos en su lista.

2.1.3. Un Estacionamiento y un Vehiculo serán iguales, si este último se encuentra dentro de la lista del primero.

2.1.4. Se agregará un Vehiculo a la lista (+) siempre y cuando este:

- No figure previamente en la lista de Vehiculos.
- Tenga una patente asignada.
- La cantidad de espacio disponible del estacionamiento sea mayor a la cantidad de Vehiculos en la lista.

2.1.5. Al quitar un Vehiculo de la lista se retornará el ticket de cobro mediante el método ***ImprimirTicket***. Caso contrario el método retornará: "El vehículo no es parte del estacionamiento".

```
Patente:TYR753
Fecha Ingreso: 08/11/2023
Hora Ingreso: 7:30

Hora Engreso: 10:30
Estadía: 3
Costo: 153
```

2.1.5.1. (Ayuda)Para calcular la estadía:

```
DateTime horaSalida = DateTime.Now;
TimeSpan estadia = horaSalida.Subtract(ingreso);
// Para obtener la duración en horas, minutos y segundos
int horas = estadia.Hours;
int minutos = estadia.Minutes;
int segundos = estadia.Seconds;
```

3. Crear un proyecto de consola con el nombre **JohnDoePOO.Consola**, donde John Doe debe ser reemplazado por su ***nombre y apellido***.

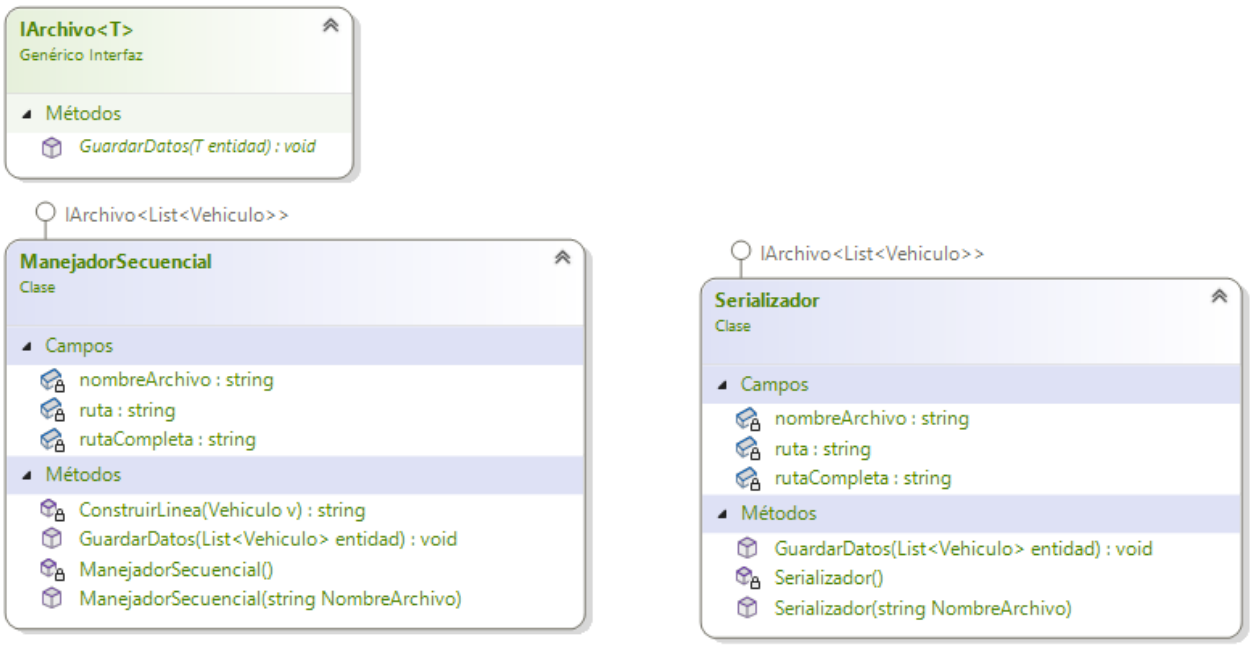
3.1. En la clase Program, copie el siguiente código y pruebe.

```
Estacionamiento e = new Estacionamiento("Instituto43", 6);
// Creación de Vehículos
Console.WriteLine("MOTOS");
Vehiculo m1 = new Moto("ASD123", 75, 4);
Console.WriteLine(m1.ConsultarDatos());
Moto m2 = new Moto("ASDaa123", 175);
Console.WriteLine(m2.ConsultarDatos());
Moto m3 = new Moto("QWE312", 175, 4, 35);
Console.WriteLine(m3.ConsultarDatos());
Console.WriteLine("PICKUPS");
PickUp p1 = new PickUp("TYR753", "78", 51);
Console.WriteLine(p1.ConsultarDatos());
PickUp p2 = new PickUp("TYR753", "Model A");
Console.WriteLine(p2.ConsultarDatos());
Console.WriteLine("AUTOMOVILES");
```

```
Automovil a1 = new Automovil("POI952", ConsoleColor.Red);

Console.WriteLine(a1.ConsultarDatos());
Automovil a2 = new Automovil("MNB651", ConsoleColor.DarkCyan, 23);
Console.WriteLine(a2.ConsultarDatos());
Console.WriteLine("--------------------------------");
Console.WriteLine("--------------------------------");
Console.WriteLine("Estacionamiento Ingreso");
e += m1;
e += p1;
e += a1;
e += m1;
e += p1;
e += a1;
e += m2;
e += p2;
e += a2;
e += m3;
Console.WriteLine((string)e);
Console.WriteLine("--------------------------------");
Console.WriteLine("--------------------------------");
Console.WriteLine("Estacionamiento Egreso");
Console.WriteLine(e - m1);
Console.WriteLine(e - p1);
//Console.WriteLine(e - a1);
//Console.WriteLine(e - m2);
Console.WriteLine(e - p2);
Console.WriteLine(e - a2);
Console.WriteLine(e - m3);
Console.WriteLine("--------------------------------");
Console.WriteLine((string)e);
Console.ReadKey();
```

4. Agregar a la capa de datos lo siguiente:



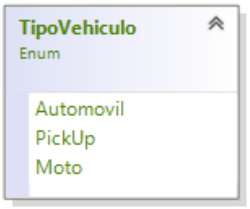
4.1. Interfaz IArchivo
   4.1.1. Debe ser una interfaz genérica.
   4.1.2. Definir el método GuardarDatos.

4.2. Clase ManejadorSecuencial

4.2.1. Debe implementar la interfaz IArchivo.
4.2.2. Tiene un constructor de clase que obtiene la ruta donde se guardará el archivo.
4.2.3. Tiene un constructor de instancia al cual se le pasa como parámetro el nombre del archivo.
4.2.4. El método Guardar, debe persistir los datos recibidos en el archivo secuencial.

4.3. Clase Serializador.
4.3.1. Debe implementar la interfaz IArchivo.
4.3.2. Tiene un constructor de clase que obtiene la ruta donde se guardará el archivo.
4.3.3. Tiene un constructor de instancia al cual se le pasa como parámetro el nombre del archivo.
4.3.4. El método Guardar, debe persistir los datos recibidos en el formato de archivo apropiado según la extensión del archivo recibido.
4.3.5. **Nota aclaratoria:** el punto puede ser desarrollado utilizando dos clases, una para cada tipo de archivo.

5. Crear un proyecto de Windows Form con el nombre **JohnDoePOO.Windows,** donde John Doe debe ser reemplazado por su nombre y apellido.



5.1. Crear una enumeración TipoVehiculo en la capa de Entidades.
5.2. (Ayuda) Los métodos para popular los combos de colores y tipos de vehículos se incluyen en el proyecto.
5.3. Programar el funcionamiento de los formularios y de los botones correspondientes a "**Ingresar**" y "**Retirar**".
5.4. Los movimientos en el estacionamiento se deben reflejar en la grilla de formulario principal.
5.5. De acuerdo con el tipo de vehículo que vaya a ingresar, ocultar o deshabilitar los controles pertinentes en el formulario de ingreso de datos.

Pautas de corrección:

1. El programa debe estar sin errores de compilación y funcionando, el programa en consola o el de Windows, esto es, desarrollar hasta el punto 3 inclusive o el 1,2 y 5 sin los botones de persistencia.
2. Subir el proyecto de forma correcta al GitHub y pasar el link correspondiente al Classroom.
3. ***IMPORTANTE: Favor de no mandar mensajes aclaratorios por WhatsApp, que más que aclarar terminan oscureciendo. <u>You will be blocked!!!</u>***

Pautas de aprobación:

1. Desarrollar hasta el punto 3 inclusive completos o el 1,2 completos y 5 sin los botones de persistencia.
2. Si debe el primer parcial, y logra hacer alguna de las persistencias tanto en consola como en Windows Forms, recupera el primer parcial.