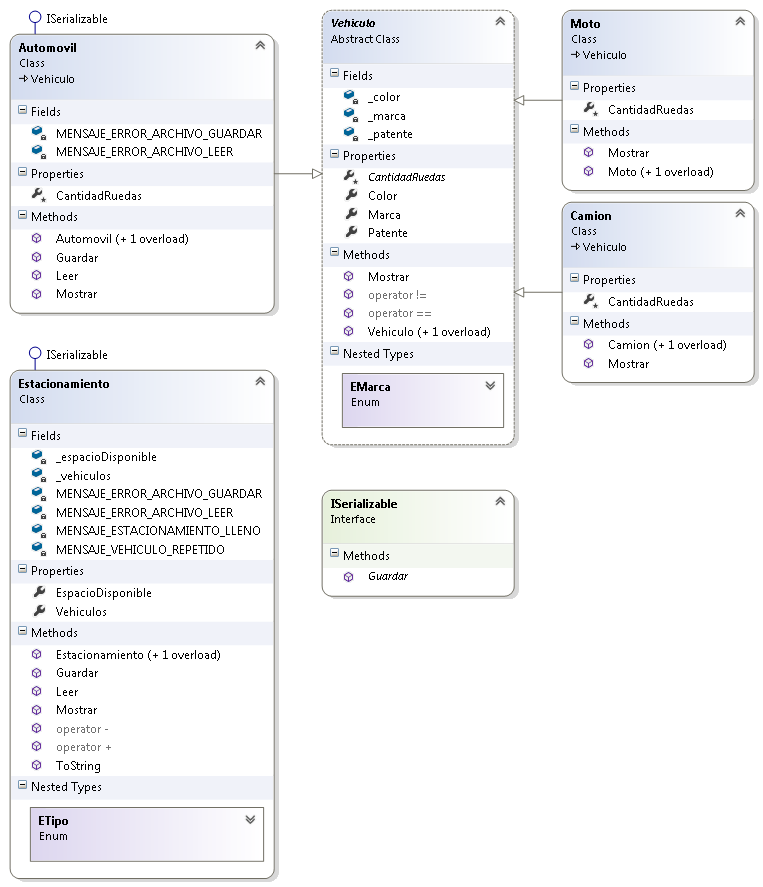
## Biblioteca de clases Entidades



Desarrollar una aplicación para administración de un estacionamiento.

Clase Estacionamiento

Debe implementar la interfaz *ISerializable*.

Atributos:

* \_espacioDisponible: número entero que indicará la cantidad de cocheras con las que cuenta el establecimiento.
* \_vehiculos: lista de objetos del tipo *Vehiculo* donde se guardaran los vehículos que estén ubicados dentro del establecimiento.

Constructores:

* En el constructor por defecto será el único lugar donde se inicialice la lista de vehículos.

Métodos:

* Mostrar es un método de clase que se encargará de retornar una cadena de texto con todos los datos del estacionamiento. Los datos de los vehículos que estén en él dependerán del atributo tipoDeVehiculo donde se pedirán los datos de los Automoviles, Motos, Camiones o Todos.
* Sobrecargar el método de Object ToString para que retorne los datos del estacionamiento, reutilizando el código existente.
* Operador +: agregará un vehículo al establecimiento, siempre y cuando tenga lugar disponible (sino deberá arrojar la excepción EstacionamientoLlenoException) y el vehículo no se encuentre ya en el establecimiento (sino deberá arrojar VehiculoRepetidoException).
* Operador -: quitará el vehículo del estacionamiento, siempre y cuando se encuentre en él.
* Guardar: serializará en un XML los datos del estacionamiento y sus vehículos, debiendo ser su contenido. Retornará True si pudo realizar la operación. Ante cualquier error lanzará ErrorArchivoException.
* Leer: deberá leer el archivo XML pedido y cargar su contenido en Estacionamiento datos. Retornará True si pudo realizar la operación. Ante cualquier error lanzará ErrorArchivoException.

Clase Vehiculo

Atributos:

* \_marca: del tipo enumerado EMarca indicará la marca del vehículo.
* \_patente: cadena de caracteres con la patente del vehículo.
* \_color: ConsoleColor que indica el color del vehículo

Constructor:

* Recibe los 3 atributos y los asigna.

Métodos:

* CantidadRuedas: de solo lectura y abstracto.
* Mostrar: virtual, retornará los datos del vehículo.

Operadores:

* Dos vehículos serán iguales si, y solo si, tienen la misma patente.

Clase Automovil, Moto y Camion

* Heredan de vehículo.
* Cantidad de Ruedas: Automóvil 4, Moto 2 y Camión 8.
* Mostrar: utilizará mostrar de Vehículo y agregará el dato de la cantidad de ruedas, propio de la clase.

Clase Automovil

Implementará ISerializable. Guardará un archivo binario con los datos del Automovil, retornando True si puede guardarlo correctamente. Si ocurre algún error, lanzará la excepción ErrorArchivoException.

Los archivos deberán guardar y recuperar todos los datos de las clases. Hacer lo necesario para que esto ocurra.

## Biblioteca de clases Excepciones:

Contendrá las clases ErrorArchivoException, EstacionamientoLlenoException y VehiculoRepetidoException.

# Main completo:

// Reemplazar por sus datos personales

Console.Title = "Me llamo [APELLIDO], [NOMBRE] y curso en la División 2º[¿C o D?]";

// Instancia de estacionamiento

Estacionamiento estacionamiento = new Estacionamiento(5);

// Instancias de los Vehículos

Camion c1 = new Camion(Vehiculo.EMarca.Scania, "ASD012", ConsoleColor.Black);

Camion c2 = new Camion(Vehiculo.EMarca.Iveco, "ASD913", ConsoleColor.Red);

Moto m1 = new Moto(Vehiculo.EMarca.BMW, "HJK789", ConsoleColor.White);

Moto m2 = new Moto(Vehiculo.EMarca.Yamaha, "IOP852", ConsoleColor.Blue);

Automovil a1 = new Automovil(Vehiculo.EMarca.Scania, "IOP852", ConsoleColor.Green);

Automovil a2 = new Automovil(Vehiculo.EMarca.Chevrolet, "QWE968", ConsoleColor.Gray);

Automovil a3 = new Automovil(Vehiculo.EMarca.Ford, "TYU426", ConsoleColor.DarkBlue);

// Agrego 6 ítems (a1 y a3 no deberían poder agregarse) y muestro

try

{

estacionamiento += c1;

}

catch (EstacionamientoLlenoException e)

{

Console.WriteLine(e.Message);

}

catch (VehiculoRepetidoException e)

{

Console.WriteLine(e.Message);

}

try

{

estacionamiento += c2;

}

catch (EstacionamientoLlenoException e)

{

Console.WriteLine(e.Message);

}

catch (VehiculoRepetidoException e)

{

Console.WriteLine(e.Message);

}

try

{

estacionamiento += m1;

}

catch (EstacionamientoLlenoException e)

{

Console.WriteLine(e.Message);

}

catch (VehiculoRepetidoException e)

{

Console.WriteLine(e.Message);

}

try

{

estacionamiento += m2;

}

catch (EstacionamientoLlenoException e)

{

Console.WriteLine(e.Message);

}

catch (VehiculoRepetidoException e)

{

Console.WriteLine(e.Message);

}

try

{

estacionamiento += a1;

}

catch (EstacionamientoLlenoException e)

{

Console.WriteLine(e.Message);

}

catch (VehiculoRepetidoException e)

{

Console.WriteLine(e.Message);

}

try

{

estacionamiento += a2;

}

catch (EstacionamientoLlenoException e)

{

Console.WriteLine(e.Message);

}

catch (VehiculoRepetidoException e)

{

Console.WriteLine(e.Message);

}

try

{

estacionamiento += a3;

}

catch (EstacionamientoLlenoException e)

{

Console.WriteLine(e.Message);

}

catch (VehiculoRepetidoException e)

{

Console.WriteLine(e.Message);

}

Console.WriteLine(estacionamiento.ToString());

Console.WriteLine("<---------------------------------------------->");

Console.ReadKey();

Console.Clear();

// Quito un item y muestro

estacionamiento -= c1;

Console.WriteLine(estacionamiento.ToString());

Console.WriteLine("<---------------------------------------------->");

Console.ReadKey();

Console.Clear();

// Muestro solo Motos

Console.WriteLine(Estacionamiento.Mostrar(estacionamiento, Estacionamiento.ETipo.Moto));

Console.WriteLine("<---------------------------------------------->");

Console.ReadKey();

Console.Clear();

// Muestro solo Camiones

Console.WriteLine(Estacionamiento.Mostrar(estacionamiento, Estacionamiento.ETipo.Camion));

Console.WriteLine("<---------------------------------------------->");

Console.ReadKey();

Console.Clear();

// Muestro solo Automoviles

Console.WriteLine(Estacionamiento.Mostrar(estacionamiento, Estacionamiento.ETipo.Automovil));

Console.WriteLine("<---------------------------------------------->");

Console.ReadKey();

Console.Clear();

// Guardo el Estacionamiento

Console.WriteLine("Guardando Estacionamiento...");

try

{

estacionamiento.guardar("estacionamiento.xml");

}

catch (ErrorArchivoException e)

{

Console.WriteLine(e.Message + ":" + e.InnerException.Message);

}

Console.WriteLine("<---------------------------------------------->");

// Leo y muestro un Automovil

Console.WriteLine("Leyendo Estacionamiento...");

try

{

Estacionamiento lecturaEstacionamiento;

estacionamiento.leer("estacionamiento.xml", out lecturaEstacionamiento);

estacionamiento = lecturaEstacionamiento;

Console.WriteLine(estacionamiento.ToString());

}

catch (ErrorArchivoException e)

{

Console.WriteLine(e.Message + ":" + e.InnerException.Message);

}

Console.WriteLine("<---------------------------------------------->");

Console.ReadKey();

// Guardo un Automovil

Console.WriteLine("Guardando Auto...");

try

{

a1.guardar("auto.bin");

}

catch (ErrorArchivoException e)

{

Console.WriteLine(e.Message + ":" + e.InnerException.Message);

}

Console.WriteLine("<---------------------------------------------->");

// Leo y muestro un Automovil

Console.WriteLine("Leyendo Auto...");

try

{

Automovil lecturaAuto;

a1.leer("auto.bin", out (lecturaAuto));

a1 = lecturaAuto;

Console.WriteLine(a1.Mostrar());

}

catch (ErrorArchivoException e)

{

Console.WriteLine(e.Message + ":" + e.InnerException.Message);

}

Console.WriteLine("<---------------------------------------------->");

Console.ReadKey();