VEHICULO – LAVADERO 2017

Crear en un proyecto de tipo ***Class Library*** la siguiente jerarquía de clases:

Clase ***Vehiculo*** que posea como atributos protegidos:

- _patente : string (con una propiedad sólo lectura)
- _cantRuedas : Byte (con una propiedad lectura/escritura)
- _marca : ***EMarcas*** (con los siguientes enumerados: Honda, Ford, Zanella, Scania, Iveco y Fiat). Crear propiedad de sólo lectura.

Y los siguientes métodos:

- (~) Mostrar() : string - virtual
- (+) Vehiculo (string, Byte, EMarcas) (sin sobrecargas)
- (+) ToString(): string - polimorfismo

Sobrecarga de operadores:

- (+) == (Vehiculo, Vehiculo) : bool. Si las patentes y marcas son iguales, retorna TRUE.

Además se pide:
Crear tres clases (***Auto***, ***Camion*** y ***Moto***) que hereden de Vehiculo y que posean un solo atributo propio cada una: _cantidadAsientos (int), _tara (float) y _cilindrada (float) como atributos protegidos respectivamente. Cada una de estas clases deberá sobrescribir el método Mostrar y el ToString (reutilizando código de la clase base) para poder retornar un ***string*** con todos sus atributos.
Generar un constructor en cada clase para inicializar cada uno de los atributos.
Como sobrecarga de constructor, cada una de las clases derivadas tendrá:

- Auto: patente, marca y cantidad de asientos
- Camion: vehiculo y tara
- Moto: marca, cilindrada, patente y cantidad de ruedas

En todos los casos reutilizar código.

Por último se desea construir la clase **Lavadero** que tendrá como atributos:

- (-) _vehiculos : List<Vehiculo>
- (-) _precioAuto : float (de clase)
- (-) _precioCamion :float (estático)
- (-) _precioMoto :float (de clase)
- (-) _razonSocial : string

Todos los atributos se inicializaran desde su constructor con parámetros. El constructor por default, que será privado, será el único encargado de inicializar la lista genérica.
El constructor estático inicializara, mediante un valor aleatorio cada uno de los precios. El rango irá desde los $150 a los $565. (No deben repetirse)

Tendrá una propiedad de sólo lectura (LavaderoToString : string) que retornará la información completa del lavadero: razón social, precios vigentes y el listado completo de los vehículos que contiene. Reutilizar código.
También poseerá una propiedad de sólo lectura Vehiculos, asociada a la lista genérica.

Los métodos que tendrá Lavadero son:

- MostrarTotalFacturado: devolverá la ganancia total del lavadero (Double), dicho método tendrá una sobrecarga que reciba como parámetro la enumeración ***EVehiculos*** (con Auto, Camión y Moto como enumerados) y retornará la ganancia del Lavadero por tipo de vehículo.
- Sobrecarga == entre un lavadero y un vehículo, retornara TRUE, si el vehículo se encuentra en el lavadero.
- Sobrecarga == entre vehículo y lavadero, retorna -1 si no está el vehículo en el lavadero, caso contrario, retorna el índice de donde se encuentra dicho vehículo.
- Sobrecarga del operador +, que agregara un vehículo siempre y cuando el vehículo no se encuentre en el lavadero. *Ej. miLavadero += unAuto;*
- Sobrecarga del operador -, que quitara al vehículo del lavadero, siempre y cuando este dicho vehículo. Reutilizar sobrecargas. *Ej. miLavadero -= unaMoto;*
- Generar un método estático (OrdenarVehiculosPorPatente : int) que reciba dos vehículos y retorne un 0 (cero), si ambas patentes son iguales, si la primera patente es ‘mayor’ que la segunda, retornará un 1 (uno) y si no, retornará un -1 (menos uno).
- Generar un método de instancia (OrdenarVehiculosPorMarca : int) que reciba dos vehículos retorne un 0 (cero), si ambas marcas son iguales, si la primera marca es ‘mayor’ que la segunda, retornará un 1 (uno) y si no, retornará un -1 (menos uno).

La aplicación debe poder ingresar vehículos de distintos tipos y marcas al lavadero, quitarlos, obtener las ganancias totales o por tipo de vehículo y mostrar los vehículos ingresados al lavadero ordenado por los distintos criterios.