VEHICULO – LAVADERO 2016

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
- marca : ***EMarcas*** (con los siguientes enumerados: Honda, Ford, Zanella, Scania, Iveco y Fiat). Crear propiedad de sólo lectura.

Y los siguientes métodos:

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

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: cantidadAsientos (int), para auto, tara (float), para camión y cilindrada (float), para moto. Todos atributos protegidos.
Cada una de estas clases deberá implementar el método MostrarAuto, MostrarCamion y MostrarMoto (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.

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

- (-) vehiculos : List<Vehiculo>
- (-) precioAuto : float
- (-) precioCamion :float
- (-) precioMoto :float

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.

Tendrá una propiedad de sólo lectura (Lavadero : string) que retornará la información completa del lavadero: 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 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 vehiculo del lavadero, siempre y cuando este dicho vehiculo. *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.

Realizar las pruebas de las clases en:
Proyecto de tipo Console Application.
Proyecto de tipo WindowForms Application.

Referencias:
(+) → public
(~) → protected
(-) → private