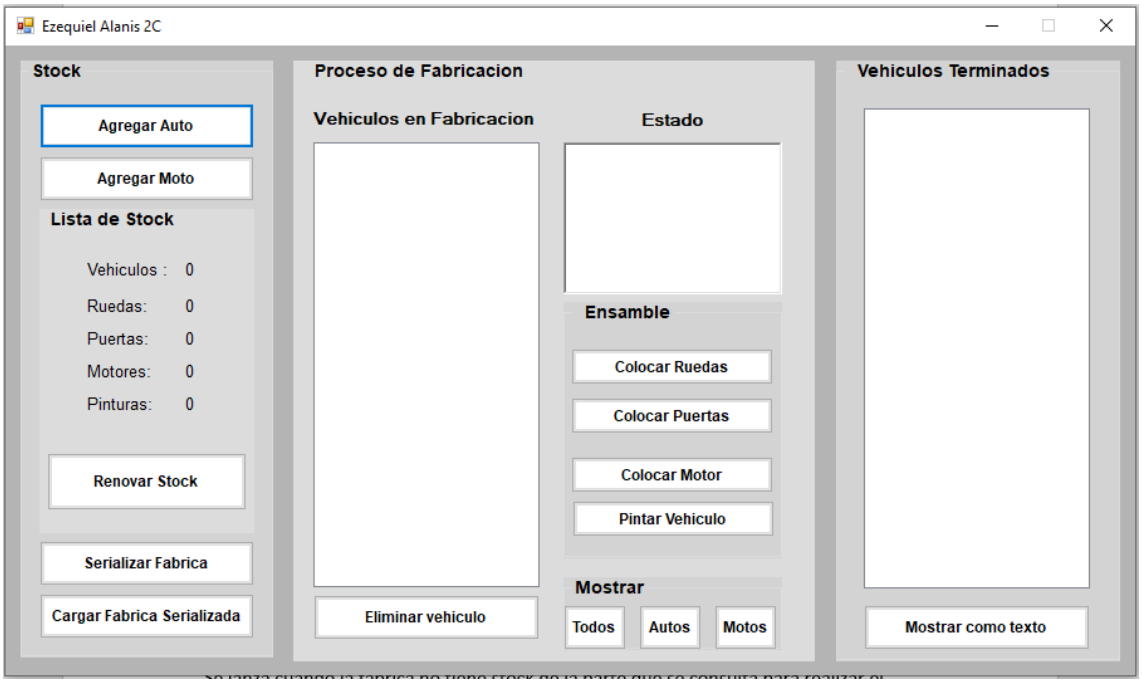
TP 3 Laboratorio.

Alumno: Ezequiel Alanis

División: 2° C

Se realizo un sistema para controlar la fabricación de vehículos, permitirá agregar Autos o Motos a una lista completando distintos datos para la creación de estos, en la cual se le podrán ir agregando las distintas partes. Se podrá ver en el estado del ensamble y en cuanto se completen todas las partes el vehículo pasara a otra lista en donde se almacenan todos los vehículos terminados. Se podrá generar un archivo de texto con la información de todos los autos terminados y se podrá serializar la fabrica completa para poder volver a cagar la información. El proyecto ya viene con una fabrica serializada para poder probar las distintas funcionalidades.



Excepciones:

Fueron creadas 3 excepciones.

StockNoDisponibleException:

Se lanza cuando la fabrica no tiene stock de la parte a ensamblar por la que se consulta. Se utiliza en la clase Fabrica y en los métodos:

`StockRuedasDisponible()`

`StockPuertasDisponible()`

`StockMotorDisponible()`

`StockPinturaDisponible()`

ValidarPatenteException:

Se utiliza en la Clase Vehiculo, en el método ValidarPatente(string patente).

Al crear un vehículo y completar los datos, se verifica que todos los campos hayan sido llenados correctamente, la patente se valida con el método mencionado y si no se genero ninguna, lanza la excepción.

VehiculoRepetidoException:

Cuando se agregan vehículos a la lista, se valida con las sobrecargas que el vehículo no se encuentre en la lista. Esto se hace comparando las patentes, en caso de que ya exista un vehículo con la misma patente, se lanzara la excepción.

Test Unitario:

En el proyecto TestUnitario, en la clase llamada TestUnitario_Fabrica se realizó el test de un constructor, un método y una excepción de la clase. En la clase TestUnitario_Vehiculo se testearon 2 excepciones y un método de esta clase.

Generics E Interfaces:

Se creo la interfaz IArchivos<T> en la cual se recibe el tipo de dato que se va a utilizar.

En el form “FrmPrincipal” se creó un método ActualizarListaVehiculos<T>() (Línea 56).

Este método recibe la clase por la que va a filtrar la lista de vehículos. Se utiliza este método en los botones btnMostrarMotos (364) btnMostrarAutos (374) btnMostrarTodos (384) .

Archivos y Serialización:

La clase ArchivoTexto contiene dos métodos e implementa la interfaz IArchivo<T>, se le asigna el tipo de dato string ya que no se va a utilizar otro.

`GuardarArchivo(string nombreArchivo, string datos)` el cual se encarga de crear el archivo de texto con el nombre recibido por parámetro y con la información recibida como datos.

`LeerArchivo(string nombreArchivo)` el cual se encarga de leer el archivo creado anteriormente y asignarlo a una variable para luego ser retornada a donde se llamó al método.

Ambos métodos se utilizan en el formPrincipal, en el botón “Mostrar Como Texto”. Genera el archivo y lo lee para mostrar en pantalla la información.

La clase ArchivoXml utiliza la interfaz IArchivo<T> en la cual se le asigna el tipo de dato a serializar. Contiene dos métodos:

`GuardarArchivo(string nombreArchivo, string datos)` el cual serializa el tipo de dato recibido y genera un archivo xml con la información.

Se utilizan en el formPrincipal, el botón “Cargar Fabrica Serializada” carga los datos de una fabrica con varios autos y motos para poder testear todas las funciones. Y El botón Serializar Fabrica, serializa todos los datos de la fabrica tal cual están en el momento.