CSIS 2664
Project 2

Due: Friday 18 September, 07:00

- This project is compulsory and will contribute to your final mark.
- You may not consult with any other person (excluding me) – not personally or by email or any other medium of communication.
- You may use any text book or Stack Overflow or other Internet resources. You should please look at and study the examples provided in the tutorials and on BlackBoard.

Scenario

You are running a vehicle selling centre that sells bakkies and cars of two manufacturers, namely Ford and Toyota. Bakkies have a tonnage and an indicator whether it has a double cab or not. Cars have an indicator whether it is an SUV or not. All vehicles have, model name, year of model and price.

Assignment

Use the abstract factory design pattern and

1. Draw a class diagram for the scenario.
   - Note that datatypes and return types must be indicated.
   - Submit in pdf format.
   - You can use any software package as long as you adhere to the conventions as discussed in Chapter 30. Note that you should not use an online tool as you will not have internet access during the summative assessments. If you have Visio, it is OK, but I struggle to position and align objects and it takes time – which you don't have in a summative assessment. Since some people might not have Visio available at home, I created a template in PowerPoint that should work equally well and may be faster: See the file *Class diagram template.pptx* on BlackBoard.
   - It is important that you get to know your tool as you will have limited time during the summative assessments.

2. Develop a console application to enter and list the details of all cars and bakkies in the showroom. Note that there may be many vehicles on the floor and there may be more than one vehicle of a specific model and year. It is also possible that there is not stock of a specific category of vehicle of a specific manufacturer, e.g. there may be no Toyota bakkies on the floor.

Important

1. All files must have your name and student number in a comment block at the top of the file. No name, no marks!
2. All files must be named properly according to their role in the pattern. Classes with the same role should be in the same file. In other words, all product classes should be in a file `1_Products.cs` and all client classes should be in a file `5_Client.cs`. Besides your name and student number, the comment block should also indicate the role of the classes in the pattern.

3. Make sure that you comply with Object Oriented Programming principles.
4. Ensure that you abide with proper naming conventions
5. Apply the SOLID and other common design principles. Indicate with comments where you applied a specific principle.

   SRP : This should not be a problem since we work with minimal classes.

   OCP : It should be possible to add another type of vehicle, e.g. motorbikes and it should be possible to add vehicles from another manufacturer without breaking the client program.

   LSP : We work with hierarchies, e.g. `ICar` : `IVehicle`. So, ensure that the properties of the subclasses are correctly assigned when you instantiate an object.

   ISP : In the *Products* file, enter the common properties (manufacturer, model, year, price) in a class, `IVehicle` and then put the specific bakkie and car properties in separate interfaces (`ICar` and `IBakkie`) that inherit from `IVehicle`.

   DIP : Develop a class `Stock` with methods `AddVehicle` and `ListStock`. It should not matter for the client class (`Main` method) whether the vehicles were stored in an array or `List<>`.

   DRY : Make sure that you do not repeat fragments of code unnecessary. For example, in the *ConcreteProducts* file, do not duplicate a `GetDetails` method with the exact same content in different classes.

   Hint : Do the development initially without SOLID and focus on the pattern only. Then refine your solution to comply with the design principles.