Factory Design Pattern

Task 1

Set up the project with all of the files.

Run the Main program. It should allow you to:

- Enter a type of Car you want to share
- Construct a ShareCar object
- Call the object's share method to actually construct the Car (or not if the Car is not available).

Test the code by entering "Ferrari", "Bentley" and "Lamborghini". Does it work as expected?

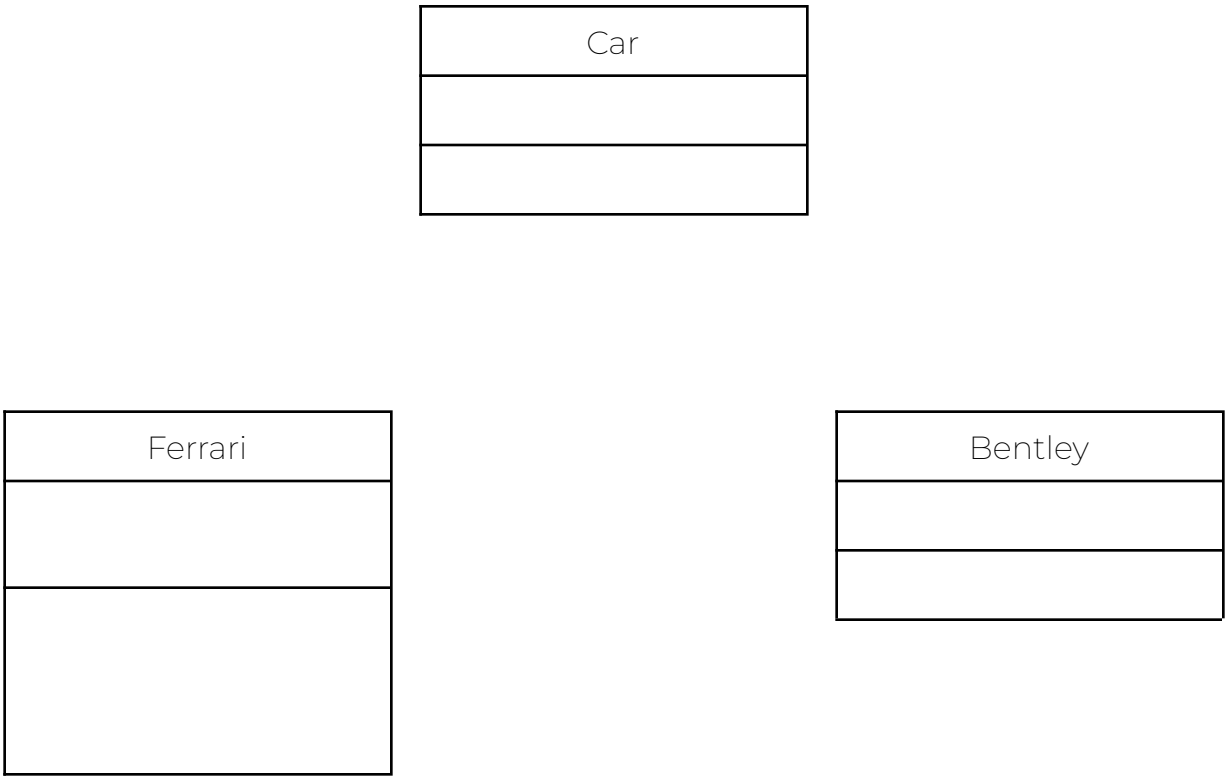
Task 2 - Code Analysis

Answer the following questions:

The Car class is an abstract class. Why is this so?	
The Car class has an abstract method, drive(). What is the consequence of making the method abstract?	
To rent a car we construct a RentCar object and then call which method?	
What is the relationship between Ferrari and Car?	

Task 3 - UML Diagram

Complete the UML diagram to show the relationship between the classes. Complete the information inside the Ferrari class only.



Task 4 - Code Development

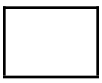
The program designers want another Car to be available for renting/buying/sharing - a Lamborghini.

Adapt the code to include a Lamborghini. You will need to make a new class and edit certain project files to accommodate this.

Test your program to make sure it works as expected.

Task 5

How many class files did you have to edit after creating the Lamborghini class?



Task 6 - Video Task

Your answer to Task 5 was probably more than 1 and in very complex computer programs, your answer would have been much more than 1.

This indicates that our program design is not very efficient and it is difficult to expand.

A solution to this problem is to follow the **Factory Design Pattern**.

Watch the video on the Unit Page and redesign your code to follow this pattern.

Task 7

Add a new class to the program, Lamborghini, and include it in the CarFactory.

Add another new class to the program, Bugatti, and include it in the CarFactory.

The **Factory Pattern** is a design pattern where a factory method creates and returns objects, allowing the client, such as the ShareCar class, to request objects *without knowing the exact class being instantiated*.

If you are unsure of this last paragraph, or any other aspect of the Factory Design Pattern, do some further research and try to establish in your own mind the true nature of the Factory and how it serves clients with objects that they request.