Instruction:

In this course assignment, the following class structure is to be used for the classes to be created. The structure has no methods and is provided as a guideline of what is expected.

Classes section:

```
1  //----------Classes----------
2
3  //Parent
4  class Vehicle {}
5
6  //Child 1
7  class Car ... {}
8
9  //Child 2
10 class Truck ... {}
11
12 //Vehicle Factory will create Vehicles of different types
13 class VehicleFactory {}
14
```

The following implementation structure is also provided. This is where the creation and manipulation of JSON and JavaScript objects and the implementation of vehicle creation are done.

Implementation section:

```
15 //----------Car Implementation----------
16
17 //Create JSON object from schema
18
19 //Convert JSON object to JS Object
20
21 //Add topSpeed to the JS Obj
22
23 //Create CarFactory and a Car
24
25 //----------Truck Implementation----------
26
27 //Create JSON object from schema
28
29 //Convert JSON object to JS Object
30
31 //Add maxLoadWeight to the JS Obj
32
33 //Create TruckFactory and a Truck
34
35
```

All functions and code must be created in a JavaScript file called "*proCA.js"* and must be correctly linked to your index.html file.

Using Inheritance, the student must implement a Factory Design Pattern for creating two different vehicle object types, "Car" and "Truck".

NOTE*: Seven questions need to be answered in this course assignment. For all the answers, screenshots must be taken and pasted into this document, where indicated. The document needs to be saved as a PDF, and this one document needs to be uploaded to Moodle as your final delivery, which will be the last step of this course assignment.*

Question 1: Understanding JSON schema

Using the provided JSON schema code below, **create two JSON objects** (Car and Truck) with all the required properties related to vehicles. This code should be under the "*Create JSON object from schema*" comments in the given implementation sections.

JSON Schema:

```
{
    "$schema": "https://json-schema.org/draft/2019-09/schema",
    "$id": "http://example.com/example.json",
    "type": "array",
    "default": [],
    "title": "Root Schema",
    "items": {
        "type": "object",
        "title": "A Schema",
        "required": [
            "wheels",
            "doors",
            "color"
        ],
        "properties": {
            "wheels": {
                "type": "integer",
                "title": "The wheels Schema",
                "examples": [
                    18,
                    4
                ]
            },
            "doors": {
                "type": "integer",
                "title": "The doors Schema",
                "examples": [
                    2,
                    4
                ]
            },
            "color": {
                "type": "string",
                "title": "The color Schema",
                "examples": [
                    "yellow",
                    "white"
                ]
            }
        },
        "examples": [{
            "wheels": 18,
            "doors": 2,
            "color": "yellow"
        },
        {
            "wheels": 4,
            "doors": 4,
            "color": "white"
        }]
    },
    "examples": [
        [{
            "wheels": 18,
            "doors": 2,
            "color": "yellow"
        },
        {
```

```
            "wheels": 4,
            "doors": 4,
            "color": "white"
        }]
    ]
}
```

For your Answer 1.1, paste the screenshot of your Car JSON object below:

For your Answer 1.2, paste the screenshot of your Truck JSON object below:

Question 2: Convert JSON

You must then **convert the created JSON objects to JavaScript objects** using one of the built-in JS methods taught in this course.

Put this code under the "*Convert JSON object to JS Object*" comments in the given implementation sections.

For your Answer 2.1, paste the screenshot of this Car JSON object conversion code below:

For your Answer 2.2, paste the screenshot of this Truck JSON object conversion code below:

Question 3: Parent

In the classes section, students have been given a partially complete Parent class, *"Vehicle".*

Parent class

```
//Parent
class Vehicle {}
```

Edit this *"Vehicle"* class to include all the required properties specified in the JSON object you created above.

The "Vehicle" class should also have a **method** that returns a description of the vehicle as a **string**.

For your Answer 3, paste the screenshot of your *"Vehicle"* class below:

Question 4: Child classes

Note: the existing Child classes in the classes section are not declared correctly. Replace the "…" with the correct formatting.

Incomplete child classes

```
//Child 1
class Car ... {}

//Child 2
class Truck ... {}
```

Question 4.1: Car Child

Create the Child class for the *"Car" vehicle.*

Incomplete Car class

```
//Child 1
class Car ... {}
```

This vehicle must have the shared properties found in the Parent class.

"Car" needs to have a unique *"topSpeed" property.*

"Car" should also have a unique *"getTopSpeed()"* method, which returns a string, indicating the top speed of this car.

For your Answer 4.1, paste the screenshot of your *"Car"* class below:

Question 4.2: Truck Child

Create the Child class for the *"Truck" vehicle.*

Incomplete Truck class

```
//Child 2
class Truck ... {}
```

This vehicle must have the shared properties found in the Parent class.

"Truck" needs to have a unique *"maxLoadWeight" property.*

"Truck" should also have a unique *"getMaxLoadWeight()"* method, which returns a string indicating the maximum load weight of this truck.

For your Answer 4.2, paste the screenshot of your *"Truck"* class below:

Question 5: Factory

A partially complete Factory class, *"VehicleFactory"* has been given.

Partially complete VehicleFactory class.

```
//Vehicle Factory will create Vehicles of different types
class VehicleFactory {}
```

This class is responsible for the creation of the vehicle objects (both "Car" and "Truck").

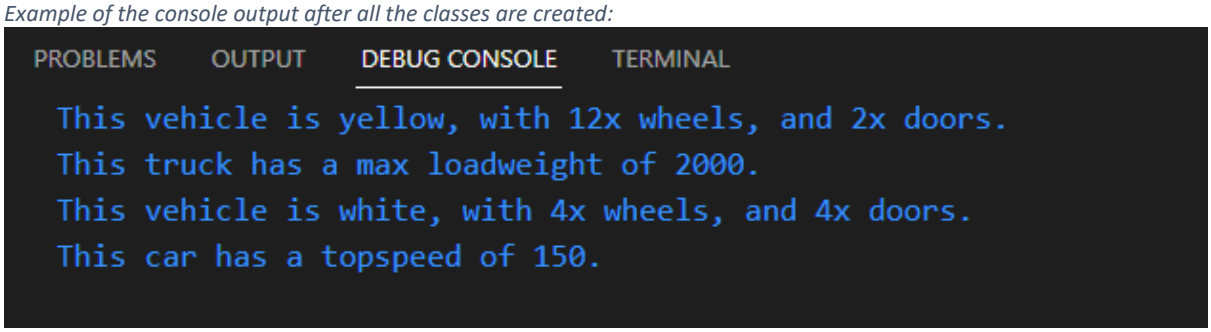
The *"create"* function should **take the desired vehicle specification** (JavaScript object) as an input parameter and **create the relevant vehicle object** based on the vehicle type received in the vehicle specification. Make this function.

For your Answer 5, paste the screenshot of your *"VehicleFactory"* class below:

Question 6: Implementation

Finally, you must implement this Factory Pattern to create both a Car and a Truck.

Example of the console output after all the classes are created:



Question 6.1: Car Factory:

Create a Car Factory, giving it the "car" vehicle type as an input parameter.

Using this new factory instance and the *"create"* function inside that factory (from Question 5), **create a Car object** with the Car properties as an input parameter (in the form of a JavaScript object).

Call the parent's description method (from Question 3) and log the result to the console.

Call the child's unique method (from Question 4.1) and log the result to the console.

Question 6.2: Truck Factory:

Create a Truck Factory, giving it the "truck" vehicle type as an input parameter.

Using this new factory instance and the *"create"* function inside that factory (from Question 5), **create a Truck object** with the Truck properties as an input parameter (in the form of a JavaScript object).

Call the parent's description method (from Question 3) and log the result to the console.

Call the child's unique method (from Question 4.2) and log the result to the console.

For your Answer 6.1, paste the screenshot of your Car Factory implementation below:

For your Answer 6.2, paste the screenshot of your Truck Factory implementation below:

Question 7: UML

Using a UML tool such as https://app.diagrams.net/, **create a Class Diagram** of this Vehicle program.

Make sure to include the following classes:

Vehicle;
Car;
Truck;
VehicleFactory.

Pay special attention to showing all object-oriented programming (OOP) concepts included in this program on the UML.

For your Answer 7, paste the screenshot of your Class Diagram below:

Finally: Submission

No code files are to be submitted or uploaded for this course assignment.

Screenshots of the relevant code must be placed into this Word document, which must then be saved as a PDF.

This PDF should be named the following: “FName_LName_PRO_CA_AUGFT22.pdf”.

(Replace “Fname” with your first name and “Lname” with your last name).

This PDF must be submitted on Moodle in the available Course Assignment Submission before the deadline.

Course Assignment Checklist

PRO Course Assignment checklist		
Question	**Specifics**	**Check**
1.1: Understanding schema (Car)	JSON car object created with correct properties and syntax	
	Code and screenshot are placed in the specified location	
1.2: Understanding schema (Truck)	JSON truck object created with correct properties and syntax	
	Code and screenshot are placed in the specified location	
2.1: Convert JSON (Car)	Car JSON Object converts to JS object using the correct JS method	
	Code and screenshot are placed in the specified location	
2.2: Convert JSON (Truck)	Truck JSON Object converts to JS object using the correct JS method	
	Code and screenshot are placed in the specified location	
3: Parent	Parent class has the correct format and all required properties	
	Parent class has a method returning a string description of the vehicle	
	Code and screenshot are placed in the specified location	
4.1: Car Child	Car child class is formatted correctly	
	Car class has unique "topSpeed" property	
	Car class has a unique "getTopSpeed()" method returning the car's top speed, in	
	Code and screenshot are placed in the specified location	
4.2: Truck Child	Truck child class is formatted correctly	
	Truck class has unique "maxLoadWeight" property	
	Truck class has a unique "getMaxLoadWeight()" method returning the truck's max load weight, in a string.	
	Code and screenshot are placed in the specified location	
5: Factory	VehicleFactory has the correct format and input parameter	
	"Create" function has the correct format (input and syntax)	
	"Create" function created both vehicle objects based on type given	
	Code and screenshot are placed in the specified location	
6.1: Car Factory Implementation	Car Factory instance created, given the correct input parameter	
	"Create" function called with correct input parameters to create a Car object	
	Parent's description method called, logging result to console	
	Child's unique method called, logging result to console	
	Code and screenshot are placed in the specified location	
6.2: Truck Factory Implementation	Truck Factory instance created, given the correct input parameter	
	"Create" function called with correct input parameters to create a Car object	
	Parent's description method called, logging result to console	
	Child's unique method called, logging result to console	
	Code and screenshot are placed in the correct specified location	
7: UML	Class diagram created with required classes	
	Correct properties and methods shown in classes	
	OOP Concepts are shown in UML	
	UML placed in the specified location	
Submission	PDF file submitted on time, into the correct folder	
	PDF file named correctly: "FName_LName_PRO_CA_AUGFT22.pdf".	