**ESMAD | TSIW | POO  
Exercise Sheet nº 7**

**Classes**

Open Visual Studio Code and solve the following exercises:

1. **Class definition (files ex1.html/ex1.js)**

Dice are small polyhedra engraved with specific instructions. The most classic data is the cube (six faces) carved with numbers from one to six.



1. Create a **Dice** class to represent a dice that will allow us to roll it and take values ranging from 1 to 6, a six-sided dice. The class must have a set of private properties such as faceValue (value of each face) and quantity (number of polyhedron faces).
2. Define a no-argument constructor that creates an object of the **Dice** class with six faces (cube).
3. Define the set and get methods for the **faceValue** private property that will store the current face value of the dice.
4. Define the following methods:
   1. A method called **getQuantityFaces** that returns a number representing the number of faces in the dice.
   2. A **roll** method that will "roll" the dice and store the result in the **faceValue** property. The drawn value must be in the range of 1 up to the number of faces of the dice.
5. Find a way to roll the dice 100 times and present a frequency table similar to the one below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Face | 1 | 2 | 3 | 4 | 5 | 6 |
| Frequency | 10 | 18 | 17 | 21 | 15 | 19 |

1. **Class definition (files ex2.html/ex2.js)**

Define a **Car** class to represent a car.



1. A car comprises a brand, license plate, color, current and maximum fuel tank (in liters), consumption (at 100km), and fuel type. Set all properties to **private**.
2. Create a constructor that receives all these parameters and initializes the respective properties.
3. Define access (**get**) and modifier (**set**) methods for all properties with particular attention to the following:
   1. **color**: get method only
   2. **current fuel tank**: value greater than 0 and less than the maximum deposit
   3. **consumption**: value greater than 0
   4. **fuel type**: Gasoline or Diesel
4. Add three cars (brand, plate, color, current fuel, maximum fuel, consumption, fuel type):
   1. “Ford”, ”91-GH-15”, “Green”, 40, 60, 5, “Diesel”
   2. “Opel”, ”23-AB-23”, “White”, 50, 55, 6.5, “Gasoline”
   3. “Nissan”, ”12-CX-45”, “Black”, 22, 70, 4.5, “Diesel”
5. Create the following methods:
   1. Method to fill the tank. The number of liters supplied must be used as a parameter.
   2. A method that receives the number of kilometers traveled and that changes the number of liters considering the base consumption of the vehicle.
6. Add the objects to an array of **cars.**
7. Create the following functions:
   1. A function that returns the number of cars of a given brand passed as a parameter.
   2. A function that, given a type of fuel, returns the sum of liters of cars that have that type of fuel.
8. **App making use of classes (files ex3.html/ex3.js)**

Make an app that manages cinema films and their directors.

* A film is characterized by a title, year of release, director, duration (in minutes), and actors (text box with names separated by commas).
* A director is characterized by a name, date of birth and country. Assume that a film can only have one director.

A black rectangle with a black background

Description automatically generated with low confidence

1. Create a Movie class and a director class with the above-mentioned properties.
2. Create the UI with the following elements:
   1. Form to create a movie.
   2. Form to create a director (will feed a select in the main movie form).
   3. Table for rendering the films (title, director, actors, year and duration).
3. Create functions that allow you to get:
   1. All films by a given director (for simplicity console log all movies).
   2. All the titles of the films where a particular actor/actress enters (for simplicity console log all movies).

You must create elements in the UI that allow you to invoke the above functions.

