Présentation de la POO en PHP

La programmation orientée objet est une méthode particulière de programmation dont le but est de faciliter la maintenance et la réutilisation de scripts PHP. Elle consiste à formaliser des objets (du monde réel ou non) sous forme d'objets formels informatiques. Cela se fait essentiellement parce qu'on appel une classe, qui regroupe des fonctions et des propriétés pouvant agir sur l'objet ou le caractérisant.

Voici comment une classe peut être codée en PHP 8 :

```
<?php
    class Voiture
    {
        /**
        * Déclaration des attributs
        */
        private $vitesse;
        private $nombre_portes;
        private $nombre_roues;
        /**
        * Cette méthode un peu spéciale est le constructeur, elle est exécutée lorsque vous "créez"
votre objet. Elle doit initialiser les attributs de la classe.
        */
        public function __construct()
        {
            $this->vitesse = 0;
            $this->nombre_portes = 3;
            $this->nombre_roues = 4;
        }
        /**
        * Première méthode accessible par tous et modifiant la vitesse de la voiture
        */
        public function accelerer($acceleration_moyenne_temps*$temps)
        {
            $this->vitesse = $acceleration_moyenne_temps*$temps;
        }
        /**
        * Seconde méthode accessible à tous et modifiant le nombre de portes
        */

        public function modifier_nb_portes( $nb_portes)
        {
            $this->nombre_portes = $nb_portes;
        }
    }
?>
```

Si on veut créer deux voitures dans ce code. Au lieu toujour modifier les données d'un tableau avec les variables "nombre_portes" ou "vitesse"; avec la programmation orientée objet, on accède à deux objets différents en deux lignes de code; ceci en "instanciant" une classe.

C'est-à-dire qu'à travers ce processus informatique, on crée une version de l'objet ayant des caractéristiques propres. C'est pourquoi la création d'un deuxième objet, le rendra indépendant du premier, bien que tout deux soient instanciés à travers la même classe (même fonctions et mêmes attributs). Ces deux objets aurant des valeurs d'attributs et des valeurs d'entrées différents.

```
<?php
require_once "Voiture.php";
        $objet_voiture = new Voiture();
?>
```

Questions :

1. **Ajouter l'attribut $Couleur pour la classe voiture**
2. **Changer tout le code pour l'attribut ajouté.**
3. **Faites la même chose pour l'attribut $Marque .**
4. **Changez tout le code après l'ajout de ce nouvel attribut.**
5. **Utiliser le code suivant pour acceder aux attributs :**

```
public function get(){
        return $this->$att;
                }
public function set($val){
        $this->$att=$val;
                }
```

6. **Ajouter tous les getter et les setter de la classe voiture.**
7. **Tester l'existence des getter et setter par le code :**

```
if(isset($objet->attribut))
    echo "L'attribut existe";
else
     echo "L'attribut n'existe pas";
```

8. **afficher le contenu de l'attribut d'une voiture que vous instanciez.**

```
$objet=new Maclasse();
echo $objet->attribut;
```