Secteur Tertiaire Informatique

Filière « Etude et développement »

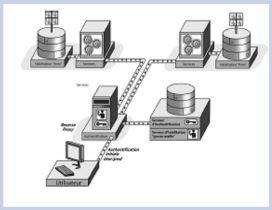
Séquence « Développer des pages Web en lien avec une base de données »

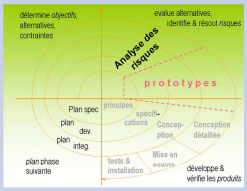
Solution Programmation orientée objet en PHP

Apprentissage

Mise en pratique

Evaluation





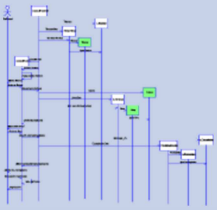


TABLE DES MATIERES

Préambule

Objectifs

Méthodologie

Il s'agit ici de proposer une solution minimale pour le script PHP de la classe Voiture et le script d'utilisation de cette classe.

Rappel : comme on souhaite que cette classe reste indépendante de son usage par un script particulier, elle n'effectue aucun 'affichage' direct (pas de `echo`) ; les messages générés seront stockés dans un attribut accessible par le getter correspondant, et c'est bien le programme principal qui saura les récupérer et les restituer comme bon lui semble à l'utilisateur.

1. LA CLASSE VOITURE

1.1 SCRIPT ET BLOC DE CLASSE

Début du script PHP '`Voiture.php`' :

```php
<?php
/*
classe Voiture
écrit par B. Hézard - 03/16
*/
class Voiture {
```

1.2 ATTRIBUTS PRIVES

```php
class Voiture {
    /* attributs privés */
    private $immatriculation; // string
    private $couleur ; // string
    private $poids ; // int
    private $puissance ; // int
    private $capacite_reservoir ; // float
    private $niveau_essence; // float
    private $nombre_places ; // int
    private $assure ; // bool
    private $message ; // string
```

1.3 CONSTRUCTEUR

```php
/* contructeur */
public function __construct($uneImmat, $uneCouleur, $unPoids,
$unePuissance, $uneCapacite, $unNbre_places)
{
    // initialisation paramètres reçus
    $this->immatriculation = $uneImmat;
    $this->couleur = $uneCouleur;
    $this->poids = $unPoids;
    $this->puissance = $unePuissance;
    $this->capacite_reservoir = (float)$uneCapacite;
    $this->nombre_places = $unNbre_places;
    // initialisation autres attributs
    $this->niveau_essence = 5.0; // la voiture est livrée avec 5l en réservoir
    $this->assure = false;
    $this->message = 'Bonjour !';
}
```

1.4 GETTERS

```php
/* getters */
public function getImmatriculation()
{
    return $this->immatriculation;
}
public function getCouleur()
{
    return $this->couleur;
}
public function getPoids()
{
    return $this->poids;
}
public function getPuissance()
{
    return $this->puissance;
}
```

Et ainsi de suite…

1.5 SETTER

```php
/* setters */
public function setAssure($unBoolAssure)
{
    $this->assure = $unBoolAssure;
    if($unBoolAssure)
    {
        $this->message = 'Merci de m\'avoir assurée';
    }
    else
    {
        $this->message = 'Attention : je ne suis plus assurée !';
    }
}
```

2. TEST INTERMEDIAIRE

```php
<?php
/* programme principal utilisant la classe Voiture */
// fusion classe Voiture
require('Voiture.php');

// début de page HTML
?>
<html>
<meta charset="utf-8" />
<head><title>Ma Porsche</title></head>
<body>
<?php
// j'achete une Porsche (on peut rever...)
$maPorsche  = new Voiture('AA 123 AA', 'Rouge', 850, 25, 70 , 4);
// je regarde le tableau de bord
echo $maPorsche->getMessage() . '<br />' ;

// exploration objet
var_dump($maPorsche);

// fin de page HTML
?>
</body>
</html>
```

3. METHODES DE SERVICE

3.1 REPEINDRE LA VOITURE

```php
// param : string
// return : true = OK ; false = erreur
public function Repeindre($uneCouleur = null)
{
    // contrôle paramètre recu
    if (!(isset($uneCouleur)))
    {
        $this->message = 'Erreur : j\'ai besoin de connaître la nouvelle couleur !';
    return false;
    };
    // message feed-back
    if($uneCouleur != $this->couleur)
    {
        $this->message = ucfirst($uneCouleur);
        $this->message .='!. Tu m\'as changé de couleur ! Je ne me reconnais plus...';}
    else
    { $this->message = 'Merci de m\'avoir rafraîchi le teint !';}
    // modif couleur
    $this->couleur = $uneCouleur;
    return true;
}
```

3.2 FAIRE L'APPOINT D'ESSENCE

```php
// param : float
// return : niveau_essence mis à jour (float)
public function Mettre_essence($quantite){
    // test si quantité peut tenir dans réservoir
    if($quantite > ($this->capacite_reservoir - $this->niveau_essence))
        // erreur : ça va déborder !
        {
            $this->message = $quantite . 'l ! Tu vas mouiller tes chaussures ! J\'ai déjà ' ;
            $this->message .= $this->niveau_essence . ' l.';
        }
    else
        // augmenter le niveau d'essence
        {
            $this->niveau_essence += $quantite;
            $this->message = 'Merci pour le carburant ! J\'ai maintenant ' . $this->niveau_essence . ' l.';
        };
    // dans tous les cas fournir le niveau courant
    return $this->niveau_essence;
}
```

3.3 SE DEPLACER

Méthode privée `consommer()` :

```php
// param : float et float
// return : float
private function consommer($uneDistance, $uneVitesse)
{
    $conso = 0; // initialisation

    // calcul carburant consomme
    if ($uneVitesse < 50)
        {$conso = $uneDistance * 0.1;}
    elseif ($uneVitesse < 90)
        {$conso = $uneDistance * 0.05;}
    elseif ($uneVitesse < 130)
        {$conso = $uneDistance * 0.08;}
    else
        {$conso = $uneDistance * 0.12;};
    return $conso;
}
```

Méthode publique `Se_deplacer()` :

```php
// param : float et float
// return : rien
public function Se_deplacer($uneDistance, $uneVitesse)
{
    $conso = $this->consommer($uneDistance, $uneVitesse);
    //echo $conso ; // pour test
    // test carburant insuffisant
    if($conso > $this->niveau_essence)
    {
        $this->message = $uneDistance . 'km : Erreur : pas assez de carburant pour ce trajet !';
    }
    else // carburant suffisant
    {
        // MAJ niveau carburant
        $this->niveau_essence -= $conso;
        $this->message = $uneDistance . 'km : Tu as consommé ' . $conso . 'l.';
    }
}
```

3.4 METHODE __TOSTRING()

```php
// return : string
public function __toString()
{
    //return "Véhicule $this->immatriculation ; puissance $this->puissance cv ; couleur $this->couleur;";
    // chaîne formatée
    $msg = 'Véhicule %s ; puissance %d cv ; couleur %s.';
    return sprintf($msg , $this->immatriculation , $this->puissance , $this->couleur);
}
```

4. EXEMPLE DE SCRIPT PRINCIPAL

```php
<?php
/* programme principal utilisant la classe Voiture */
// fusion classe Voiture
require('Voiture.php');

// début de page HTML
?>
<html>
<meta charset="utf-8" />
<head><title>Ma Porsche</title></head>
<body>

<?php
// j'achete une Porsche (on peut rever...)
$maPorsche  = new Voiture('AA 123 AA', 'Rouge', 850, 25, 70 , 4);
// je regarde le tableau de bord
echo $maPorsche->getMessage() . '<br />' ;

// exploration objet
var_dump($maPorsche);

// j'assure !
$maPorsche->setAssure(true);
// je regarde le tableau de bord
echo $maPorsche->getMessage() . '<br />' ;
```

```php
// je passe à la pompe
$niveau = $maPorsche->Mettre_essence(50);
// je regarde le tableau de bord
echo $maPorsche->getMessage() . '<br />' ;


// j'ajoute de l'essence
$niveau = $maPorsche->Mettre_essence(20);
// je regarde le tableau de bord
echo $maPorsche->getMessage() . '<br />' ;


// je roule 100km à 90km/h
$maPorsche->Se_Deplacer(100,90);
// je regarde le tableau de bord
echo $maPorsche->getMessage(). '<br />'  ;

// je roule 1000km à 200km/h
$maPorsche->Se_Deplacer(1000,200);
// je regarde le tableau de bord
echo $maPorsche->getMessage(). '<br />'  ;
```

```php
// je repeins ma Porsche
// NB : sans utilisation du retour de la fonction
// $maPorsche->Repeindre(); // pour test
$maPorsche->Repeindre('bleu profond');
// je regarde le tableau de bord
echo $maPorsche->getMessage() . '<br />' ;

$msg = "Niveau d'essence : ";
$msg .= $maPorsche->getNiveau_essence();
$msg .= 'l.<br />';
echo $msg;

// methode magique __toString()
//echo $maPorsche->__toString();
echo $maPorsche;

// exploration objet
var_dump($maPorsche);

// fin de page HTML
?>
</body>
</html>
```

5. EXEMPLE D'AFFICHAGES SUR LE BROWSER

Bonjour !

```
object(Voiture)[1]
  private 'immatriculation' => string 'AA 123 AA' (length=9)
  private 'couleur' => string 'Rouge' (length=5)
  private 'poids' => int 850
  private 'puissance' => int 25
  private 'capacite_reservoir' => float 70
  private 'niveau_essence' => float 5
  private 'nombre_places' => int 4
  private 'assure' => boolean false
  private 'message' => string 'Bonjour !' (length=9)
```

Merci de m'avoir assurée
Merci pour le carburant ! J'ai maintenant 55 l.
201 ! Tu vas mouiller tes chaussures ! J'ai déjà 55 l.
100km : Tu as consommé 8l.
1000km : Erreur : pas assez de carburant pour ce trajet !
Bleu profond!. Tu m'as changé de couleur ! Je ne me reconnais plus...
Niveau d'essence : 47l.
Véhicule AA 123 AA ; puissance 25 cv ; couleur bleu profond.

```
object(Voiture)[1]
  private 'immatriculation' => string 'AA 123 AA' (length=9)
  private 'couleur' => string 'bleu profond' (length=12)
  private 'poids' => int 850
  private 'puissance' => int 25
  private 'capacite_reservoir' => float 70
  private 'niveau_essence' => float 47
  private 'nombre_places' => int 4
  private 'assure' => boolean true
  private 'message' => string 'Bleu profond!. Tu m'as changé de couleur ! Je ne me reconnais plus...' (length=70)
```

6. BILAN DE L'EXERCICE

- Tous les attributs qui stockent les données de l'objet sont déclarés private dans la classe ;
- Créer des méthodes getters pour tous les attributs qui peuvent être communiqués à l'extérieur ;
- Créer des méthodes setters pour tous les attributs modifiables de l'extérieur ; intégrer des tests sur la bonne passation des paramètres nécessaires et sur la validité des valeurs reçues ;
- Créer un constructeur pour la classe ; soit un constructeur standard sans paramètre qui initialise simplement les attributs, soit un constructeur paramétré qui réalise ces initialisations selon les paramètres reçus (à contrôler).
  NB : en PHP, comme on ne peut définir qu'un seul constructeur (pas de surcharge de méthode en PHP), il est fréquent de définir des constructeurs sans paramètres ; l'instanciation d'un objet doit alors être suivie des appels de tous les setters nécessaires ;
- Créer des méthodes publiques pour les services offerts à l'extérieur ; là encore, tester paramètres passés et valeurs reçues ;
- Créer des méthodes privées pour tous les besoins de service interne réutilisables ;
- Enrichir la classe en créant la méthode (magique) __toString() ;
- Toutes ces méthodes ne devraient pas effectuer de sortie directe de manière à pouvoir être réutilisées dans divers contextes ;
- Le code doit être soigneusement documenté (voir les outils comme Doxygen) afin de préciser le mode d'emploi des méthodes exposées ;
- Le programme principal commence par fusionner le code des classes nécessaires ;
- Le programme principal instancie puis manipule des objets et réalise lui-même les restitutions à l'utilisateur ;
- Le programme principal gagne en clarté et en lisibilité car tout le détail est relégué dans les classes manipulées ;

CREDITS

ŒUVRE COLLECTIVE DE l'AFPA
Sous le pilotage de la DIIP et du centre d'ingénierie sectoriel Tertiaire-Services

Equipe de conception (IF, formateur, mediatiseur)
B. Hézard - formateur
Chantal Perrachon – Ingénieure de formation>

Date de mise à jour : 31/03/16