

EXERCICES PHP

CORRECTION

Les variables

Exercice 1 :

Donner les valeurs de \$x, \$y, \$z après l'exécution du script suivant :

```
$x="PostgreSQL";  
$y="MySQL";  
$z=$x;  
$x="PHP 5";  
$y.=$x;
```

Correction :

```
$x = PHP 5  
$y= MySQLPHP 5  
$z= PostgreSQL
```

Les conditions

Exercice 2

Rédiger un script conditionnel pour tester si un nombre est à la fois un multiple de 3 et de 7.

Correction

```
<?php  
$x=1325775;  
if($x%3==0 AND $x%7==0)  
{  
echo $x ."est multiple de 3 et de 7 <br />";  
}  
else  
{  
echo $x . "n'est pas multiple de 3 et de 7 <br>";}  
?>
```

Exercice 3

Écrire une code PHP qui permet de sélectionner une personne de sexe masculin dont l'âge est compris entre 40 et 60 ans compris et afficher un message de bienvenue approprié.

Correction

```
<?php
$sexe="M";
$age=52;

if($sexe=="M" && $age>=40 && $age <= 60)
{
echo "Bonjour Monsieur, vous avez entre 40 et 60 ans <br >"; }
else
{
echo "Soit que vous n'êtes pas un homme, soit que votre âge n'est pas entre 40 et 60 ans<br>"; }
?>
```

Les boucles

Exercice 4

Écrire la table de multiplication de 9 (de 1 à 100), en utilisant la boucle for. Cette table doit être présentée sous forme d'une liste non ordonnée puis sous la forme d'une table HTML.

```
<?php
echo '<ul>';
for($i = 1 ; $i <= 10 ; $i++)
echo '<li>' . $i . ' * 7 = ' . ($i *7) . '</li>';
echo '</ul>';
?>
```

Exercice 5 :

Écrire la table de multiplication de 9 (de 1 à 100) en utilisant la boucle while.

Correction

```
<?php
$i = 1;
while ($i <= 100) {
    echo $i . ' x 9 = ' . $i * 9 . '<br>';
    $i++;
}
?>
```

Exercice 6 :

Écrire un programme qui permet de calculer la somme :
 $1+2+2^2 + 2^3 + 2^4 + 4^5 + \dots + 2^{100}$

Correction

```
<?php
$x = 2;
$n = 100;
$somme = 0;

for ($i=0; $i<= $n ; $i++) {
    $terme = $x**$i;
    $somme = $somme + $terme;
}

echo $somme;

?>
```

Exercice 7

Écrire un programme qui affiche la somme suivante :

$$1^3 + 2^3 + 3^3 + 4^3 + \dots + n^3$$

Les fonctions**Exercice 8**

Écrire une fonction dont son exécution permet de tester la parité d'un nombre et affiche un message selon le résultat de ce test.

Correction

```
< ?php

function estpair($x){

    if($x % 2 == 0){
        echo "Le nombre ".$x. " est pair";
    }else
    {
        echo "Le nombre ".$x. " est impair";
    }
}

estpair(23);
estpair(10);

?>
```

Exercice 9

Écrire une fonction `jeux_de()` qui génère trois nombres aléatoires à l'aide de la fonction `random_int`, représentant trois dés cubiques à six faces numérotées de 1 à 6, et qui renvoie la valeur booléenne `True` si les dés forment 432, et la valeur booléenne `False` sinon. Puis tester le fonctionnement de cette fonction.

Correction

```
<?php

function jeu_de(){

    $de1 = random_int(1, 6);
    $de2 = random_int(1, 6);
    $de3 = random_int(1, 6);

    if (($de1 == 4 && $de2 == 2 && $de3 == 1)
    || ($de1 == 4 && $de2 == 1 && $de3 == 2 )
    || ($de1 == 1 && $de2 == 2 && $de3 == 4)
    || ($de1 == 1 && $de2 == 4 && $de3 == 2 )
    || ($de1 == 2 && $de2 == 4 && $de3 == 1)
    || ($de1 == 2 && $de2 == 1 && $de3 == 4 ))
    {
        return True;
    }
    else
    {
        return False;
    }
}

if(jeu_de()){
    echo "Bien joué";
}
else
{
    echo "Perdu";
}
}
```

Exercice 10

Écrire une fonction qui permet d'afficher le nombre tirage aléatoire nécessaire pour trouver un nombre de quatre chiffres passé en paramètre.

Correction :

Utilisation de la boucle while

```
<?php

function find_nombre($nombre)
{

    $compteur = 0;
    $nombre_alea = -1;

    while($nombre_alea != $nombre)
    {
        $nombre_alea = random_int(0, 10000);
        $compteur++;
    }
    echo "Le nombre ".$nombre. " est trouvé après " . $compteur. " essais ";
}

Appel de la fonction
find_nombre(243);
```

Méthode 2 utilisant la boucle for

```
<?php

function find_nombre($nombre)
{

    $compteur = 0;
    $nombre_alea = -1;
    for($compteur = 0; $nombre_alea != $nombre; $compteur++)
    {
        $nombre_alea = random_int(0, 1000);
    }
    echo "Le nombre ".$nombre. " est trouvé après " . $compteur. " essais ";
}

Appel de la fonction :
find_nombre(324);
?>
```

Les tableaux

Exercice 11

Soit le tableau \$tab = ['a'=>'z', 0=>20, 'nom'=>'Mastafi']. Quelles sont les clés du tableau ? Quelles sont les valeurs ?

Correction

Les clés sont : 'a' , 0 et 'nom'
Les valeurs sont : 'z', 20, et 'Mastafi'

Exercice 12

Considérons le tableau \$tab = [1, 45, 100]
Écrire le code qui permet d'ajouter le nombre 123 à ce tableau ainsi que le mot 'Bonjour' ayant la clé 'salut'.

Correction

```
<?php

$tab = [1, 45, 100];

$tab[] = 123;
$tab[salut] = Bonjour;

Affichage de verification

echo $tab[0]. "<br>";
echo $tab[1]. "<br>";

echo $tab[2]. "<br>";
echo $tab[3]. "<br>";

echo $tab[salut];

?>
```

Exercice 13

Le tableau suivant liste l'âge des membres du bureau d'une association.
Afficher sous forme d'une liste ordonnée les noms de tous les membres.

```
$tabAge = [  
    'Lionel' => 24  
    'Claire' => 65  
    'Teresa' => 43  
    'Christopher' => 23  
    'Ahmed' => 19  
    'Amine' => 48  
    'Gérard' => 39  
];
```

Correction

```
echo "<ol>";  
foreach ($tabAge as $cle => $valeur)  
    echo "<li>" . $cle . "</li>";  
echo "</ol>";
```

Exercice 14

Qu'affiche le code suivant :

```
$tab = [ 10 => "Etudiant", 20 => "Bonjour" ];  
  
if (in_array(10, $tab))  
    echo ' 10 est dans le tableau <br>';  
  
if(in_array("Bonjour", $tab))  
    echo ' in_array : nombre est dans le tableau <br>';
```

La fonction `in_array('test', $tab)` : permet de tester si 'test' est dans le tableau \$tab.
Tester le code pour vérifier votre réponse.

Exercice 15

Écrire un tableau multidimensionnel associatif dont les clés sont des noms de personnes et les valeurs sont des tableaux numérotés contenant le prénom, la ville et l'âge de la personne.

Correction

```
<?php
$tableau=array(
"Winder"=>array("Claire","Marseille",53),
"Mastafi"=>array("Amine","Bordeaux",23),
"Assude"=>array("Christpher","Madrid",19)
);
?>
```

Vous pouvez appeler la fonction **print_r(\$tableau)** pour voir le résultat.

Exercice 16

Écrire un tableau multidimensionnel associatif dont les clés sont des noms de personnes et les valeurs sont des tableaux associatifs dont les clés sont le prénom, la ville et l'âge de la personne avec une série de valeurs associées.

Correction

```
<?php
$tableau=array("Winder"=>array("prenom"=>"Claire","ville"=>"Marseille","age"=>27),
"Mastafi"=>array("prenom"=>"Amine","ville"=>"Bordeau","age"=>35),
"Assude"=>array("prenom"=>"Christopher","ville"=>"Madrid","age"=>45));
?>
```

Vous pouvez appeler la fonction **print_r(\$tableau)** pour voir le résultat.

Exercice 17

En utilisant la boucle foreach, lire les données le tableau de l'exercice 15.

Correction

```
<?php
$tableau=array(
"Winder"=>array("Claire","Marseille",53),
"Mastafi"=>array("Amine","Bordeaux",23),
"Assude"=>array("Christpher","Madrid",19)
);
foreach($tableau as $cle=>$valeur) {
echo "<b>Élément $cle :</b><br />";
foreach($valeur as $indice=>$val)
{
echo "élément $indice :", $val, "<br />"; }
}
?>
```

Exercice 18

En utilisant la boucle foreach, lire les données le tableau de l'exercice 16.

Correction

```
<?php
$tableau = array(
"Winder"=>array("prenom"=>"Claire","ville"=>"Marseille","age"=>27),
"Mastafi"=>array("prenom"=>"Amine","ville"=>"Bordeau","age"=>35),
"Assude"=>array("prenom"=>"Christopher","ville"=>"Madrid","age"=>45));
foreach($tableau as $cle=>$valeur)
{
echo "<b>Element $cle :</b><br />";
foreach($valeur as $cle2=>$valeur2)
{
    echo " $cle2 :", $valeur2, "<br />";
}
}
?>
```

Les formulaires

Exercice 19

Créer un formulaire qui permet de saisir le nom, prénom, l'adresse, la ville et le code postal des étudiants. Les données saisies sont ensuite traitées et affichées dans un tableau HTML par un fichier PHP séparé.

Correction

Fichier : *form_etudiants.php*

```
<!DOCTYPE html>
<html>
<head>
  <title>Coordonnées des étudiants</title>
</head>
<body>
<form action="coordonnees_etudiant.php" method="post">
<fieldset>
<legend><b>Saisissez vos coordonnées </b></legend>
<table border="0" >
  <tr>
    <td>Nom : </td>
    <td><input type="text" name="nom"></td>
  </tr>
  <tr>
    <td>Prénom : </td> <td><input type="text" name="prenom"></td>
  </tr>
  <tr>
    <td>Adresse : </td>
    <td><input type="text" name="adresse"></td>
  </tr>
  <tr>
    <td>Ville :</td>
    <td><input type="text" name="ville"></td>
  </tr>
  <tr>
```

```

        <td>Code postal :</td>
        <td><input type="text" name="code_postal"></td>
    </tr> <tr>
        <td></td>

    <td><input type="submit" value="enregistrer" /></td> </tr>
</table>
</fieldset>
</form>

</body>
</html>

```

Fichier coordonnees_etudiant.php

```

<?php
$nom = $_POST['nom'];
$prenom = $_POST['prenom'];
$adresse = $_POST['adresse'];
$ville = $_POST['ville'];
$code_postal = $_POST['code_postal'];

$tableau = array(
    'Nom' => $nom,
    'Prénom' => $prenom,
    'Adresse' => $adresse,
    'Ville' => $ville,
    'Code Postal' => $code_postal
);

echo "<table border='1' >";
echo "<caption><b>Confirmation de vos coordonnées</b></caption>";

foreach($tableau as $cle=>$valeur)
{
    echo "<tr> <td> $cle : &nbsp;  </td> <td>".stripslashes($valeur) . "</td></tr>";
    }
echo "</table>"; ?>

```

Exercice 20

Créer un programme qui permet à l'utilisateur de saisir le prix hors taxes et la TVA correspondante et ensuite il affiche le prix TTC et la TVA saisie. (Un seul fichier PHP).

```
<!DOCTYPE html>
<html>
<head>
  <title>Produit</title>
</head>
<body>

<form method="post">
<fieldset>
<legend><b>Saisissez le prix HT et le taux de TVA </b></legend> <table border="0"
>
<tr>
  <td>Prix Hors Taxes : </td>
  <td><input type="text" name="prix_ht" > </td>
</tr> <tr>
  <td>Taux de TVA (en %) : </td>
  <td><input type="text" name="tva" > </td>
</tr>

<?php
if(isset($_POST['prix_ht'])){
if(!empty($_POST['prix_ht']) AND !empty($_POST['tva']) ) {

  $prix_ht = $_POST['prix_ht'];
  $tva = $_POST['tva'];
  ?>
<tr><td>Montant de la TVA : </td>
  <td><?= round($prix_ht*$tva/100,2); ?></td>
</tr>
<tr><td>Prix TTC : </td>
  <td><?=round($prix_ht*(1+$tva/100),2); ?></td>
```

```
</tr>
  <?
} else

{
echo "<b>Le formulaire est incomplet!</b>"; }

}
?>

<tr><td>&nbsp;</td>
<td><input type="submit" value="Calculer" name="prixttc" ></td> </tr>
  </table>
  </fieldset>
  </form>

</body>
</html>
```

NB. La fonction **isset()** permet de tester si une variable existe. Dans le cas de cet exercice elle permet de tester si le bouton « enregistrer » dont le nom est « prix_ht » est sélectionné.

Programmation POO en PHP

Exercice 21

Écrire une classe représentant un client. Elle a deux attributs *nom* et *ville* et *adresse email*. Puis écrire une méthode qui affiche « Le client X habite à Y, son adresse email est : Z ».

Créer des objets client, affectez leurs propriétés, et utilisez la méthode d'affichage.

Correction

Fichier : Client.php

```
<?php
class Client
{
    public $nom;
    public $ville;
    public $email;
    public function getinformation()
    {
        $texte = "Le client " . $this->nom. " habite à " . $this->ville.
        ", son adresse email est : " . $this->email. "<br />";
        return $texte;
    }
}
//Création d'objets
$client1 = new Client();
$client1->nom = "Richard";
$client1->ville = "Paris";
$client1->email = "richard@yahoo.fr";

echo $client1->getinformation();

?>
```

Exercice 22 : Classes sans constructeur

Écrivez une classe représentant une ville. Elle doit avoir les propriétés nom, département, région et nombre d'habitants et une méthode affichant « la ville X est dans le département Y de la région Z et a N habitants ».

Créez des objets ville, affectez leurs propriétés, et utilisez la méthode d'affichage.

Proposition de solution

Fichier : Ville.php

```
<?php
class Ville {
    public $nom;
    public $departement;
    public $region;
    public $habitants;

    public function getInfos() {

        echo "La ville ".$this->nom. " est dans le département: ".$this->departement. " de la région ".$this->region. " et a ".$this->habitants. " habitants <br>";
    }
}

//Création d'objets ville
$ville_1 = new Ville();
$ville_1->nom = "Strasbourg";
$ville_1->departement = "Bas Rhin";
$ville_1->region = "Grang Est";
$ville_1->habitants = 279284;

$ville_1->getInfos();
```

Exercice 23

Modifier la classe de l'exercice 21 en ajoutant un constructeur. Créer des objets clients et exécuter la méthode d'affichage du texte.

Correction

```
<?php
class Client
{
    public $nom;
    public $ville;
    public $email;

    public function __construct($nom, $ville, $email) {
        $this->nom=$nom;
        $this->ville=$ville;
        $this->email=$email;
    }

    public function getinformation()
    {
        $texte = "Le client " . $this->nom. " habite à " . $this->ville.
        ", son adresse email est : " . $this->email. "<br />";
        return $texte;
    }
}

//Création d'objets
$client1 = new Client("Richard", "Paris", "richard@yahoo.fr" );

echo $client1->getinformation();

?>
```

Exercice 24

Modifier la classe de l'exercice 23 en créant une méthode qui permet de modifier l'adresse email du client. Créer des objets client, et utilisez l'ensemble des méthodes.

Correction

```
<?php
class Client
{
    public $nom;
    public $ville;
    public $email;

    public function __construct($nom, $ville, $email) {
        $this->nom=$nom;
        $this->ville=$ville;
        $this->email=$email;
    }

    public function getinformation()
    {
        $texte = "Le client " . $this->nom. " habite à " . $this->ville.
        ", son adresse email est : " . $this->email. "<br />";
        return $texte;
    }

    public function setemail($email)
    {
        $this->email = $email;
    }
}

//Création d'objets
$client1 = new Client("Richard", "Paris", "richard@yahoo.fr" );

echo $client1->getinformation();
echo "<br>";
$client1-> setemail("aminet14@gmail.com");
echo $client1->getinformation(); ?>
```

Exercice 25 : Classe avec constructeur

Modifiez la classe de l'exercice précédent en lui ajoutant un constructeur.
Réalisez les mêmes opérations de création d'objets et d'affichage.

Solution

```
<?php
class Ville2 {
    private $nom;
    private $departement;
    private $region;
    private $habitants;

    function __construct($nom, $departement, $region, $habitants) {
        $this->nom=$nom;
        $this->departement=$departement;
        $this->region=$region;
        $this->habitants=$habitants;
    }

    public function getInfos() {

        echo "La ville ".$this->nom. " est dans le département: ".$this->departement. " de la région ".$this->region. " et a ".$this->habitants. " habitants <br>";
    }
}

//L'affectation des valeurs se fait lors de la création de l'objet par
//le biais du constructeur

$ville_1 = new Ville2("Strasbourg", "Bas Rhin", "Grang Est",279284);

$ville_1->getInfos();
```

Exercice 26 : constructeur et destructeur

Créer une classe qui représente un agent bancaire qui permet de récupérer et afficher son nom, prénom, son grade et lieu de travail.

Solution :

Fichier Banquier.php

```
<?php
class Banquier {
    private $nom;
    private $prenom;
    private $grade;
    private $ville_agence;

    function __construct($nom,$prenom,$grade, $ville_agence="Paris") {
        $this->nom=$nom;
        $this->prenom=$prenom;
        $this->grade=$grade;
        $this->ville_agence=$ville_agence;
    }

    public function getNom() {
        return $this->nom;
    }
    public function getPrenom() {
        return $this->prenom;
    }
    public function getGrade() {
        return $this->grade;
    }
    public function getVille_agence(){
        return $this->ville_agence;
    }
}
```

```
/* pour le constructeur La méthode __destruct(): sera appelée  
automatiquement soit après la destruction explicite de l'objet,  
soit après la fin du script et la disparition de toutes les références à  
l'objet. La méthode destructeur va permettre de nettoyer les ressources  
avant que PHP ne libère l'objet de la mémoire.  
*/  
  
function __destruct() {  
    echo "L'action $this->nom n'existe plus!<br />";  
}  
}  
  
//Création d'objets  
$agent_admin = new Banquier("Didier","Clair", "agent administratif",  
    "Marseille");  
  
$chef_agence = new Banquier("Strock","Karine", "chef d'agence", "Paris");  
  
//recupération des informations  
  
echo "Le banquier " . $agent_admin->getNom(). " " . $agent_admin->  
    >getPrenom(). " est " . $agent_admin->getGrade(). " et travaille à  
    " . $agent_admin->getVille_agence(). "<br>";  
  
echo "Le banquier " . $chef_agence->getNom(). " " . $chef_agence->  
    >getPrenom(). " est " . $chef_agence->getGrade(). " et travaille à  
    " . $chef_agence->getVille_agence(). "<br>";
```

Exercice 27: Getters et setters et constructeur

Créer une classe Véhicule qui a pour propriétés le type et nombre de places, puis créer les méthodes qui permettent d'une part de mettre à jour ces propriétés et d'autre part les récupérer.

Créer des objets de type Véhicule, afficher leurs propriétés

Modifier les propriétés de ces objets et réafficher de nouveau leurs propriétés.

Solution**Fichier Vehicule.php**

```
<?php
class Vehicule
{
    // Attributs
    private $type;
    private $place;
    // Méthodes
    public function __construct($type, $place)
    {
        $this->setType($type);
        $this->setPlace($place);
    }

    Public function setType($type)
    {
        $this->type = $type ;
    }

    Public function setPlace($place)
    {
        $this->place = $place ;
    }

    Public function getType()
```

```
{  
return $this->type ;  
}  
  
Public function getPlace()  
{  
return $this->place ;  
}  
  
}  
  
// Création d'un objet Vehicule  
$vehicule_1 = new Vehicule("tourisme", 5);  
  
//Récupération des valeurs des propriétés de l'objet vehicule_1  
echo "Type: ".$vehicule_1->getType(). "<br>";  
echo "Nombre de place: ".$vehicule_1->getPlace(). "<br>";  
echo "<br><br>";  
  
//Mise à jour des données de L'objet vehicule_1  
  
$vehicule_1->setType("tourisme et utilitaire");  
$vehicule_1->setPlace(7);  
  
//Récupération des nouvelles valeurs des propriétés de l'objet vehicule_1  
echo "Type: ".$vehicule_1->getType(). "<br>";  
echo "Nombre de place: ".$vehicule_1->getPlace(). "<br>";
```

Exercice 28: Les classes avec contrôles de validité dans le constructeur

Écrire une classe qui représente les employés d'une entreprise ayant les propriétés suivantes : nom, prénom et âge. Cette classe doit être dotée d'un constructeur permettant de valider les conditions suivantes : toutes ces attributs sont requis et que l'âge doit être entre 18 et 65 ans compris. Ensuite, créer des employés et afficher leur nom, prénom et âge.

Solution

La classe : Employes.php

```
<?php

// classe Employes
class Employes {
// attributs de la classe
    private $prenom;
    private $nom;
    private $age;

// getters et setters
    public function getPrenom(){
        return $this->prenom;
    }
    public function getNom(){
        return $this->nom;
    }
    public function getAge(){
        return $this->age;
    }
    public function setPrenom($prenom){

        //le prénom est obligatoire, il ne doit pas être null
        // trim() retourne la chaîne str, après avoir supprimé les caractères
        invisibles en début et fin de chaîne.

        $prenom = trim($prenom);
        if ($prenom === "") {
            echo "Le prénom ne doit pas être vide <br>";
        }
    }
}
```

```
    } else {
        $this->prenom = $prenom;
    }
}

public function setNom($nom){
    // le nom ne doit pas être vide
    $nom = trim($nom);

    if ($nom === "") {
        echo "Le nom ne doit pas être vide <br>";
    }else{
        $this->nom = $nom;
    }
}

public function setAge($age){
    // L'âge ne doit pas être un nombre < 18 ou supérieur à 65
    //et ne doit pas être vide
    if ($age < 18 || $age > 65 || $age==="") {
        echo "L'âge doit être un entier positif entre 18 et 65 ans <br>";
    } else {
        $this->age = $age;
    }
}

// constructeur
public function __construct($prenom, $nom, $age) {
    // Initialisation des attributs par les setters
    $this->setPrenom($prenom);
    $this->setNom($nom);
    $this->setAge($age);
}
}

$employe1 = new Employes("Lea", "Bart", 23);
echo "Prénom: " . $employe1->getPrenom(). "<br>";
echo "Nom: " . $employe1->getNom(). "<br>";
echo "Age: " . $employe1->getAge(). "<br>";
```

Exercice 29: héritage

On reprend l'exercice sur la création de la classe Véhicule, quelles modifications peut-on faire sur la visibilité des propriétés. Créer une classe dérivée (classe fille) appelée Voiture qui hérite de la classe Véhicule tout lui ajoutant deux attributs marque et volume de carburant ainsi qu'une méthode permettant de savoir l'état de démarrage selon le volume du carburant.

Solution

On peut séparer les deux classes dans deux fichiers différents : Vehicule.php et Voiture.php

Fichier Vehicule.php

```
<?php
class Vehicule
{
    // Attributs
    private $type;
    private $place;
    // Méthodes
    public function __construct($type, $place)
    {
        $this->setType($type);
        $this->setPlace($place);
    }

    Public function setType($type)
    {
        $this->type = $type ;
    }

    Public function setPlace($place)
    {
        $this->place = $place ;
    }
}
```

```
Public function getType()  
{  
    return $this->type ;  
}  
  
Public function getPlace()  
{  
    return $this->place ;  
}  
  
}
```

Fichier Voiture.php

```
<?php  
  
// Inclure obligatoirement la classe parent avant le code de la classe  
fille. (once : veut dire une seule fois)  
require_once('Vehicule.php');  
  
class Voiture extends Vehicule  
{  
    // Attributs  
    private $marque;  
    private $volumeCarburant;  
    // Constructeur  
    public function __construct($type,$place,$marque,$volumeCarburant)  
    {  
        // Appel du constructeur de la classe parente  
        parent::__construct($type,$place);  
  
        //initialisation de nouveaux attributs  
        $this->setMarque($marque);  
        $this->setVolumeCarburant($volumeCarburant);  
    }  
  
    Public function setMarque($marque)
```

```
{
    $this->marque = $marque ;
}

Public function setVolumeCarbuarant($volumeCarburant)
{
    $this->volumeCarburant = $volumeCarburant ;
}

Public function getMarque()
{
    return $this->marque ;
}

Public function getVolumeCarbuarnt()
{
    return $this->volumeCarburant ;
}

// Démarre la voiture si Le réservoir
// n'est pas vide
public function demarrer()
{
    if ($this->getVolumeCarbuarnt() > 0)
    {
        echo 'Le véhicule démarre sans aucun problème';
        return true;
    }
    echo 'Le réservoir est vide...';
    return false;
}

}

$ma_voiture = new Voiture("tourisme", 5, "Mercedes", 12);
```

```
echo "Type: ".$ma_voiture->getType(). "<br>";
echo "Nombre de place: ".$ma_voiture->getPlace(). "<br>";
echo "Marque: ".$ma_voiture->getMarque(). "<br>";
echo "Volume de carburant: ".$ma_voiture->getVolumeCarburant().
"<br><br>";
$ma_voiture->demarrer();
?>
```

Exercice 30: héritage

Créez une classe représentant une personne. Elle déclare les propriétés nom et prénom et un constructeur.

Créez une classe client dérivée de la classe personne en y ajoutant la propriété adresse et une méthode getCoordonnee() qui affiche les coordonnées complètes de la personne.

Créez une classe électeur dérivée de la même classe, et ajoutez-y deux propriétés bureau de vote et vote, ainsi que deux méthodes ; une méthode boolean qui permet retourner la valeur True à l'attribut vote et une autre méthode isVoter qui permet de tester si un électeur a voté.

Solution

La classe Personne.php

```
<?php
//Classe personne
class Personne
{
    private $nom;
    private $prenom;

    function __construct($nom, $prenom) {

        $this->nom=$nom;
        $this->prenom=$prenom;
    }
}
```

```
public function setNom($nom){
    $this->nom = $nom;
}

public function setPrenom($prenom){
    $this->prenom = $prenom;
}

public function getNom() {
    return $this->nom;
}

public function getPrenom() {
    return $this->prenom;
}
}
```

La classe Client.php

```
<?php

require_once('Personne.php');

class Client extends Personne
{
    private $adresse;
    public function __construct($nom,$prenom,$adresse) {

        // Appel du constructeur de la classe parente
        parent::__construct($nom,$prenom);
        $this->setAdresse($adresse);
    }

    //setters
    public function setAdresse($adresse){
        $this->adresse = $adresse;
    }
}
```

```
}

// getters
public function getAdresse() {
    return $this->adresse;
}

public function getCoordonnee() {

    $cord = "Le client " . $this->getPrenom() . " " . $this->getNom(). "
habite à " . $this->adresse. "<br>";
    return $cord;
}

}
```

La classe Electeur.php

```
<?php
//Classe electeur
class Electeur extends Personne {

    public $bureau_de_vote;
    public $vote;

    function __construct($nom,$prenom,$bureau_de_vote) {

        parent::__construct($nom, $prenom);
        $this->setBureauDeVote($bureau_de_vote);

    }

    //setter
    public function setBureauDeVote($bureau_de_vote){
        $this->bureau_de_vote = $bureau_de_vote;
    }
}
```

```
//getter

public function getBureaudeVote() {
    return $this->bureau_de_vote;
}

//methode boolean definissant si un electeur a voté
public function aVoter() {
    $this->vote=TRUE;
}

public function isVoter(){

    if($this->vote){
        echo "L'électeur ".$this->getNom(). " ". $this->getPrenom(). " a voté
dans le bureau ".$this->getBureauDeVote() ."<br>";
    }
    else {
        echo "L'électeur ".$this->getNom(). " ". $this->getPrenom(). " est
bien inscrit dans le bureau ".$this->getBureauDeVote() .", mais il n'a pas
voté <br>";
    }
}
}
```

Fichier de test de création d'objet : traitement.php

```
<?php
require_once('Personne.php');
require_once('Client.php');
require_once('Electeur.php');

$client1 = new Client("Mastafi","Amine","Boulevard Jacques Monod");

echo $client1->getCoordonnee();
```

```
$electeur1 = new Electeur("Winder", "Christpher","Bureau 1");  
  
$electeur1->aVoter();  
$electeur1->isVoter();  
  
?>
```