**Object Oriented Programming**

1. On défini une classe en la déclarant avec le mot clé class :

**<?php**//la définition d'une classe se fait par le mot clé class  
//suivi du nom et de crochet comme suit:  
  
**class** Product  
**{  
  
}***/\*\* Par convention on commence les mots par une majuscule \*/*

1. Instantiation

**<?php***/\*\*  
 \* Les objets ne sont que des instances d'une classe. donc une classe sera une espèce d'empreinte qui peut être  
 \* réitérée  
 \*/***class** Product  
**{  
  
}**//Instantiation d'une classe grâce au mot clé "new"  
$produit1 = **new** Product();  
$produit2 = **new** Product();  
  
print\_r($produit1);  
var\_dump($produit1);  
var\_dump($produit2);

1. Propriétés : en OOP les variables de la classe sont appelées attributs.
2. **<?php***/\*\*  
    \* En POO les variables sont appelées propriétés  
    \* Elles sont définies au sein de la classe  
    \*/***class** Product  
   **{  
    var** $title = 'default product';  
    **var** $productName = 'default name';  
    **var** $productReference = '0000';  
    **var** $price = 1000;  
    **var** $array = **array**();  
   **}**$product = **new** Product();  
   //On peut acceder á la propriété price grace à l'opérateur "->"  
   var\_dump($product->price);

4) Attribuer une valeur à une propriété :

**<?php  
  
class** Product  
**{  
 var** $title = 'default product';  
 **var** $productName = 'default name';  
 **var** $productReference = '0000';  
 **var** $price = '1000';  
**}**$product = **new** Product();  
$product2 = **new** Product();  
// On peut affecter des valeurs différentes aux différents objects à travers du même opérateur flèche  
//Sachez que le fait d'attribuer des valeurs directement comme ceci est une mauvaise pratique. et par conséquent  
//elle n'est jamais utilisée il faut privilégier les setters que nous verrons plus tard  
$product->title = 'incredible product';  
$product2->title = 'This is a shitty product';  
  
//important on ne met pas le dollar en face de la propriété quand on l'appel  
**echo** $product->title;  
**echo** '<br />';  
**echo** $product2->title;

1. Une méthode est une fonction dans un contexte orienté objet
2. **<?php**header('Content-Type: text/html;Charset=utf8;');  
   */\*\*  
    \* Les fonctions en programmation orientée objet sont appelées methodes  
    \*/***class** Product  
   **{  
    var** $title = 'default product';  
    **var** $productName = 'default name';  
    **var** $productReference = '0000';  
    **var** $price = '1000';  
     
    */\*\* Methode qui est déclarée avec le mot clé "function" \*/* **function** displayString()**{  
    echo** "Salut je suis une méthode";  
    **}  
     
    function** displayNumer($number)**{  
    echo** $number;  
    **}  
   }**$object = **new** Product();  
   //Pour acceder à la methode il faut utilser notre opérateur flèche  
   $object->displayString();  
     
   **echo** '<br/>';  
   $object->displayNumer(2);
3. Scope : (public / protected / private )

**<?php**header('Content-Type: text/html;Charset=utf8;');  
*/\*\*  
 \* Le scope est défini par trois mots clés: public, protected, private  
 \* public -> veut dire que la propriété ou méthode est accessible en dehors de la classe  
 \* au sein de la classe elle-même ET dans les classe qui hérite de notre classe  
 \* protected -> veut dire que la propriété ou méthode est accessible au sein de la classe  
 \* même ou d'une classe qui a lien de parenté  
 \* private->la méthode ou la propriété ne peut accédé qu'au sein de la classe  
 \*/***class** Product  
**{  
 public** $title = 'default product';  
 //Generalement quand une méthode est private elle est déclarée avec un underscore  
 **protected** $\_productName = 'default name';  
 **public** $productReference = '0000';  
 **public** $price = '1000';  
  
 **public function** displayString()**{  
 echo** "Salut je suis une méthode";  
 **}  
  
 private function** \_displayPrivateStuff()**{  
 echo** "Je suis un code secret";  
 **}  
}**$object = **new** Product();  
//Pour acceder à la methode il faut utilser notre opérateur flèche  
$object->displayString();  
**echo** '<br/>';  
**echo** $object->\_productName;  
/\*echo '<br/>';  
echo $object->\_productName;  
echo '<br/>';  
echo $object->\_displayPrivateStuff();\*/

1. La variable interne «$this ». Accesseur interne dans un contexte objet
2. **<?php***/\*\*  
    \* La variable interne $this intervient quand on veut accéder à une propriété ou une méthode au sein  
    \* d'une même classe  
    \*/***class** Product  
   **{  
    public** $title = 'default product';  
    //Generalement quand une méthode est private elle est déclarée avec un underscore  
    **private** $\_productName = 'default name';  
    **public** $productReference = '0000';  
    **public** $price = '1000';  
     
    **public function** displayPrice()**{**// echo $this->price;  
    $this->echoSomething($this->price);  
    **}  
     
    private function** echoSomething($var)**{  
    echo** $var;  
    **}** //Une classe peut retourner des informations comme le ferait une fonction  
    **public function** getProductName()**{  
    return** $this->\_productName;  
    **}  
   }**$object = **new** Product();  
   $object->displayPrice();  
     
     
   // Exemple de retour d'information de la part d'une methode d'une classe  
   $price = $object->getProductName();  
   **echo** '<br/>';  
   **echo** $price;
3. Exo écrire une classe Addresse qui peut contenir des addresses :
4. **<?php***/\*\*  
    \* Écrivez une classe Address qui peut contenir des addresses  
    \* propriété rue, code postal, ville et pays. avec des valeurs par défaut  
    \*  
    \* Address peut aussi afficher un code postal  
    \* ou une rue. Attention rue sera une propriété privée  
    \*/***class** Address**{  
    private** $rue='Rue de la soif';  
    **private** $code\_postal='66037';  
    **private** $ville='Canet en Rousillon';  
    **private** $pays='France';  
     
    **public function** displayCodePostal()  
    **{  
    echo** $this->code\_postal;  
    **}  
     
    public function** displayRue()**{  
    echo** $this->rue;  
    **}  
   }**$object = **new** Address();  
   $object->displayCodePostal();  
   $object->displayRue();
5. Exo écrire une classe Selfie qui puisse afficher « MonNom »

**<?php***/\*\*  
 \* Écrivez une classe ou qui puisse afficher Salut je suis Alfonso  
 \* Ou "Alfonso" est un argument d'une methode  
 \*/*$object = **new** Selfie();  
$object->displayString('MonNom');

1. **Le constructeur :**  est une méthode magique qui intervient au moment de l’instantation d’une classe. On dit qu’elle intervient lors d’un évènement (Celui de la construction). Il y a d’autres méthodes qui sont dites magique qui répondent à des « évènements » nous le verront plus tard.
2. **<?php***/\*\*  
    \* Le contructeur est une méthode qui est appelée automatiquement à chaque fois qu'une classe est instanciée  
    \* Ceci est une méthode magique nous verrons plus tard ce que c'est  
    \* la méthode \_\_contruct() est le constructeur  
    \*/***class** User**{  
    private** $\_userId;  
    **public function** \_\_construct($id)**{** $this->\_userId = $id;  
    **}  
     
    public function** getId()  
    **{  
    return** $this->\_userId;  
    **}  
     
   }**$object = **new** User(5);  
   **echo** $object->getId();
3. Les constantes de classe. Elles ont vocation à rester constantes. On peut y acceder de n’importe où dans le code avec NomDeLaClass ::NOM\_CONSTANTE ;

**<?php***/\*\*  
 \* Les constantes  
 \*/***class** User**{  
 protected** $userId;  
 **protected** $privilege;  
  
 **const** *USER\_ADMIN*=1;  
 **const** *USER\_WRITER*=2;  
 **const** *USER\_MODERATOR*=3;  
  
 **public function** \_\_construct($id)  
 **{** $this->userId = $id;  
 **}  
  
 public function** getId()  
 **{  
 return** $this->userId;  
 **}** //J'ai rajouté cette fonction setPrivilege()  
 **public function** setPrivilege($code)**{  
 if**(in\_array($code,[**self**::*USER\_ADMIN*,user::*USER\_WRITER*,user::*USER\_MODERATOR*]))**{** $this->privilege = $code;  
 **}  
 }  
  
 public function** getPrivilege()**{  
 return** $this->privilege;  
 **}  
  
}**$object = **new** User(5);  
$object->setPrivilege(User::*USER\_MODERATOR*);  
**echo** $object->getPrivilege();

1. Les attributs statiques : On peut y avoir accès avec la même logique que précédemment. NomDeLaClass ::$variableStatique ; Il est important de remarquer que ces variables ont un contexte de classe et NON d’0bject.

Exemple typique du compteur :

**<?php***/\*\*  
 \* Les attributs statiques  
 \*/***class** User**{  
 protected** $userId;  
 **protected** $privilege;  
 //ajout compteur  
 **static public** *$counter*;  
  
 **const** *USER\_ADMIN*=1;  
 **const** *USER\_WRITER*=2;  
 **const** *USER\_MODERATOR*=3;  
  
 **public function** \_\_construct($id)  
 **{** $this->userId = $id;  
 **self**::*increaseCounter*();  
 **}  
  
 public function** getId()  
 **{  
 return** $this->userId;  
 **}** //J'ai rajouté cette fonction setPrivilege()  
 **public function** setPrivilege($code)**{  
 if**(in\_array($code,[**self**::*USER\_ADMIN*,user::*USER\_WRITER*,user::*USER\_MODERATOR*]))**{** $this->privilege = $code;  
 **}  
 }  
  
 public function** getPrivilege()**{  
 return** $this->privilege;  
 **}  
  
 static public function** increaseCounter()**{  
 self**::*$counter*++;  
 **}  
  
  
  
}**$object = **new** User(5);  
/\*$object->setPrivilege(user::USER\_MODERATOR);  
echo $object->getPrivilege();\*/  
**echo** '<br/>';  
$object2 = **new** User(8);  
$ss = **new** User(9);  
**echo** User::*$counter*;

1. Exemple d’une calculatrice
2. **<?php  
     
   class** Calculatrice**{  
    private** $\_a;  
    **private** $\_b;  
    **private** $\_result;  
     
    **public function** setVars($a,$b)**{** $this->\_a=$a;  
    $this->\_b=$b;  
    **}  
     
    public function** multiply()**{** $this->\_result=$this->\_a\*$this->\_b;  
    **}  
     
    public function** getResult()**{  
    return** $this->\_result;  
    **}  
   }**$calc = **new** Calculatrice();  
   $calc->setVars(5,6);  
     
   $calc->multiply();  
   **echo** $calc->getResult(); //echo
3. Autoloader. Ce morceau de code nous permet de charger les classe sant devoir les faire les incude/require. Ils se ferront automatiquement à partir du moment que les classes sont appelées ou instantiées.

**<?php***/\*\* version 1 \*/***function** \_\_autoload($class\_name)**{  
 include** $class\_name.'.php';  
**}***/\*\* version 2 \*/***function** loadMyClass($classe)**{  
 include** $classe.'.php';  
**}**spl\_autoload\_register('loadMyClass');  
*/\*\*  
 \* fonction appelée à chaque fois qu'une classe est instantiée  
 \*/*$address1 = **new** Address();  
$person1 = **new** Person();  
  
**echo** "Tout va bien toutes les classes existent";

À annexer à l’exemple precedent

Addresses.php

**<?php  
class** Address  
**{  
 public function** \_\_construct()  
 **{  
 echo** 'La classe Address est appelee <br/>';  
 **}  
}**

Person.php

**<?php  
class** Person  
**{  
 public function** \_\_construct()  
 **{  
 echo** 'La classe Person est appelee <br/>';  
 **}  
}**

1. Exemple de connexion à une base de données. Avec la librairie PDO :
2. **<?php***/\*\*  
    \* On va créer une classe qui intéragie avec la base de données  
    \*/***class** bddManager**{  
    private** $db;  
    **public function** \_\_construct()**{** $this->setDb();  
    **}  
     
    public function** setDb()**{** $connexion = **new** PDO("mysql:host=localhost;dbname=blog;charset=UTF8",'root','');  
    $connexion->setAttribute(PDO::*ATTR\_ERRMODE*, PDO::*ERRMODE\_EXCEPTION*);  
    $connexion->setAttribute(PDO::*ATTR\_EMULATE\_PREPARES*, **false**);  
    $this->db=$connexion;  
    **}  
     
    public function** getAllPost()**{** $query = 'SELECT p.id,p.title,p.description,p.date\_post,u.id AS user\_id,u.username,*COUNT*(c.id) AS nb\_comment  
    FROM USER AS u,post AS p   
    LEFT JOIN COMMENT AS c ON c.post\_id=p.id  
    WHERE u.id=p.poster\_id GROUP BY p.id ORDER BY id DESC';  
    $pdo = $this->db->prepare($query);  
    $pdo->execute(**array**());  
    **return** $pdo->fetchAll(PDO::*FETCH\_ASSOC*);  
    **}  
   }**$object = **new** bddManager();  
   $post = $object->getAllPost();  
   var\_dump($post);

Note évidemment pour que cela marche il faut une base de données qui s’appel « blog » et qui ai une table User et une autre table comment. Enfin il n’y a qu’à regarder ce qu’il y a dans la requête SQL.

1. Exercices :

Démonstration des propriétés OOP avec la création d’un classe voiture qui a des caractéristiques dynamiques. Nous pouvons instantier cette classe et les différentes instances peuvent interagir entre elles :

**<?php  
  
class** Voiture**{  
 private** $\_vitesse = 150;  
 **private** $\_puissance = 115;  
 **private** $\_prix = 20000;  
 **private** $\_gain = 0;  
 **private** $\_nbrCourseGagnees = 0;  
 **private** $\_maintenance = 200;  
  
 **const** *VOITURE\_OK*=1;  
 **const** *VOITURE\_MAINTENANCE*=2;  
  
 */\*\*  
 \** ***@return*** *int  
 \*/* **public function** getVitesse()  
 **{  
 return** $this->\_vitesse;  
 **}** */\*\*  
 \** ***@param*** *int $vitesse  
 \*/* **public function** setVitesse($vitesse)  
 **{** $this->\_vitesse = $vitesse;  
 **}** */\*\*  
 \** ***@return*** *int  
 \*/* **public function** getPuissance()  
 **{  
 return** $this->\_puissance;  
 **}** */\*\*  
 \** ***@param*** *int $puissance  
 \*/* **public function** setPuissance($puissance)  
 **{** $this->\_puissance = $puissance;  
 **}** */\*\*  
 \** ***@return*** *int  
 \*/* **public function** getPrix()  
 **{  
 return** $this->\_prix;  
 **}** */\*\*  
 \** ***@param*** *int $prix  
 \*/* **public function** setPrix($prix)  
 **{** $this->\_prix = $prix;  
 **}** */\*\*  
 \** ***@return*** *int  
 \*/* **public function** getGain()  
 **{  
 return** $this->\_gain;  
 **}** */\*\*  
 \** ***@param*** *int $gain  
 \*/* **public function** setGain($gain)  
 **{** $this->\_gain = $gain;  
 **}** */\*\*  
 \** ***@return*** *int  
 \*/* **public function** getNbrCourseGagnees()  
 **{  
 return** $this->\_nbrCourseGagnees;  
 **}** */\*\*  
 \** ***@param*** *int $nbrCourseGagnees  
 \*/* **public function** setNbrCourseGagnees($nbrCourseGagnees)  
 **{** $this->\_nbrCourseGagnees = $nbrCourseGagnees;  
 **}** */\*\*  
 \** ***@return*** *int  
 \*/* **public function** getMaintenance()  
 **{  
 return** $this->\_maintenance;  
 **}** */\*\*  
 \** ***@param*** *int $maintenance  
 \*/* **public function** setMaintenance($maintenance)  
 **{** $this->\_maintenance = $maintenance;  
 **}  
  
  
 public function** faireLaCourse(Voiture $voiture)**{  
 if**($this->checkMaintenance()==**self**::*VOITURE\_MAINTENANCE*)**{  
 return** 'La voiture ne peut pas courrir';  
 **}  
 if**($voiture->getVitesse() > $this->getVitesse())**{** $voiture->gagnerCourse();  
 $this->perdreCourse();  
 **}elseif**($voiture->getVitesse() < $this->getVitesse())**{** $voiture->perdreCourse();  
 $this->gagnerCourse();  
 **}** $voiture->diminuerMaintenance();  
 $this->diminuerMaintenance();  
 **}  
  
 public function** checkMaintenance()**{  
 if**($this->\_maintenance<100)**{  
 return self**::*VOITURE\_MAINTENANCE*;  
 **}  
 }  
  
 public function** diminuerMaintenance()**{** $this->\_maintenance-=100;  
 **}  
  
 private function** gagnerCourse()**{** $this->\_gain+=1000;  
 $this->\_nbrCourseGagnees++;  
 **}  
  
 private function** perdreCourse()**{** $this->\_gain+=50;  
 **}  
  
}**header('Content-Type: text/html;charset=UTF8');  
  
$voiture1 = **new** Voiture;  
$voiture1->setMaintenance(100);  
$voiture2 = **new** Voiture;  
$voiture2->setVitesse(90);  
$voiture1->faireLaCourse($voiture2);  
**echo** 'Voiture 1 Gain:';  
**echo** $voiture1->getGain();  
**echo** '<br />';  
**echo** 'Voiture 1 Courses Gagnées:';  
**echo** $voiture1->getNbrCourseGagnees();  
**echo** '<br />';  
**echo** 'Voiture 2 Gain:';  
**echo** $voiture2->getGain();  
**echo** '<br />';  
**echo** 'Voiture 2 Courses Gagnées:';  
**echo** $voiture2->getNbrCourseGagnees();  
  
**echo** '<br />';  
$voiture1->faireLaCourse($voiture2);  
**echo** 'Voiture 1 Gain:';  
**echo** $voiture1->getGain();  
**echo** '<br />';  
**echo** 'Voiture 1 Courses Gagnées:';  
**echo** $voiture1->getNbrCourseGagnees();  
**echo** '<br />';  
**echo** 'Voiture 2 Gain:';  
**echo** $voiture2->getGain();  
**echo** '<br />';  
**echo** 'Voiture 2 Courses Gagnées:';  
**echo** $voiture2->getNbrCourseGagnees();

1. Heritage