La Programmation Orientée Objet en PHP

1. Introduction à la Programmation Orientée Objet :

La programmation orientée objet (OOP) organise le code autour des objets et classes.

- Classe : plan ou modèle qui définit les caractéristiques et comportements d'un objet.
- Objet : instance concrète d'une classe.

Exemple :

```
class Voiture {
   public $marque;
   public $couleur;
   public function demarrer() {
      echo "La voiture démarre";
   }
}
$maVoiture = new Voiture();
$maVoiture->marque = "Toyota";
$maVoiture->couleur = "Rouge";
$maVoiture->demarrer();
```

2. Encapsulation et Modificateurs d'Accès :

Encapsulation c-à-d protéger les données et contrôler l'accès.

Modificateurs :

- public : accessible partout
- private : accessible seulement à l'intérieur de la classe
- protected : accessible dans la classe et ses sous-classes

Exemple :

```
class CompteBancaire {
   private $solde = 0;
   public function
deposer($montant) {
      $this->solde += $montant;
   }
   public function afficherSolde() {
      echo $this->solde;
   }
}
```

La Programmation Orientée Objet en PHP

3. Héritage et Polymorphisme :

Héritage : c'est une classe peut utiliser les propriétés et les méthodes d'une autre.

Polymorphisme : même méthode avec comportement différent selon l'objet.

Exemple :

```
class Animal {
    public function parler() {
        echo "L'animal fait un bruit";
    }
}
class Chien extends Animal {
    public function parler() {
        echo "Le chien";
    }
}
```

4. Méthodes Magiques :

Les méthodes magiques commencent par __ et permettent d'automatiser certaines actions.

__construct() : appelé à la création de l'objet

__destruct() : appelé à la destruction de l'objet

__get() et **__set()** : accéder et modifier des propriétés privées

Exemple :

```
class Personne {
    private $nom;
    public function __construct($nom) {
        $this->nom = $nom;
    }
    public function __get($prop) {
        return $this->$prop;
    }
}
$p = new Personne("Ali");

echo $p->nom;
```

La Programmation Orientée Objet en PHP

5. Gestion des Erreurs et Exceptions :

PHP permet de gérer les erreurs avec try / catch / throw.

Exemple :

```
function division($a, $b) {
   if($b == 0) {
      throw new Exception("Division par zéro impossible");
   }
   return $a / $b;
}
try {
   echo division(10, 0);
} catch(Exception $e) {
   echo "Erreur : " . $e->getMessage();
}
```

6. Interfaces et Traits :

Interface : définit des méthodes obligatoires pour les classes qui l'implémentent.

Trait : permet de réutiliser du code dans plusieurs classes.

Exemple Interface :

```
interface Conducteur {
   public function conduire();
}
class Chauffeur implements Conducteur {
   public function conduire() {
      echo "Je conduis un véhicule";
   }
}
```

Exemple Trait :

```
trait Logger {
   public function log($msg) {
      echo "Log : $msg";
   }
}
class Application {
   use Logger;
}
$app = new Application();
$app->log("Application démarrée");
```

La Programmation Orientée Objet en PHP

2. Namespaces et Autoloading :

Namespace : permet d'organiser le code et éviter les conflits de noms.

Autoloading : charge automatiquement les classes quand elles sont utilisées.

Exemple Namespace :

```
namespace MonApp\Models;
class Utilisateur {
    public $nom;
}
```

Exemple Autoloading :

```
spl_autoload_register(function
($class) {
    include $class . '.php';
});
```