Synthèse

PROGRAMMATION ORIENTÉE OBJET

Pseudo variable $this :

Représente l'objet qu'on est en train d'utiliser, donc tous ces attributs et ses méthodes.
La portée d'une variable appelée par this est toute la classe.

les types de visibilité :

Ils délimitent la portée des attributs et des méthodes d'un objet, il y en existe 3 :

Private : accessible seulement dans la classe dans la class dans laquelle elle est déclaré

Protected : accessible depuis la class actuelle est dans les classes enfants quand il y a un héritage

Auto chargement des classes :

Permet de ne pas faire plusieurs d'include, ce code charge automatiquement les classes dont il a besoin

```php
<?php
function chargerClass($class)
{
    include $class . ".php";
}
spl_autoload_register("chargerClass");
```

Constantes de classe :

Les constantes de classe sont utilisables sans nécessiter la création d'un d'objet, propre à la classe;

Ex: `new Personnage::FORCE_MOYENNE`

Attribut et méthodes statiques :

Les méthodes statiques sont des méthodes liées à la classe et non à l'objet .

elle peuvent être utilisée sans instancier d’objet

```
public static function Connard(){
print("connard");
}
Personnage::Connard();
```

dans les fonctions static on utilise self

```
public static function Connard($nom){
$nom = self::$_nom;
print("connard ".$nom);

}
Personnage::Connard("Billy");
```

L’hydratation

Hydrater un objet, c’est lui fournir des valeurs venant d’une BDD

L’héritage

Une classe fille hérite de toutes les méthodes et attributs de la classe mère

Si dans la classe fille on veut appeler une méthode parent on fait :

```
parent::laMéthode();
```

L’abstraction

C’est une classe qui ne peut instancier d’objet et sert d’exemple aux filles.

exemple:

```
abstract class Personnage
{
    private $_id;
    private $_nom = 'Inconnu';
    private $_force = 50;
    etc..

 abstract public function attaquer(Personnage $adversaire):Personnage;
    //abstract force les méthodes filles à avoir une classe attaquer()

    public function insulter()
    {
        print("</br>" .$this->getNom()." : Tête de gland !");
    }
```

<u>Note:</u> Si toutes les classes était en abstract, ce ne serait plus une classe abstraite, mais une interface.

Une méthode abstraite ne comporte pas de corps et oblige toutes les classes héritant à avoir la même fonction (et renvoyer le même type au besoin, comme “ :Personnage “).

Classe finales

Une classe finale est une classe qui ne peut avoir classe de fille

methode magique

__construct(), __destruct(), __call(), __callStatic(), __get(), __set(), __isset(), __unset(), __sleep(), __wakeup(), __serialize(), __unserialize(), __toString(), __invoke(), __set_state() __clone(), et __debugInfo().

__construct : sert a construit l'objet avec les paramètres donnée

__tostring : transforme l'objet en string

__destruct : détruit l'objet

Comparaisons

Pour comparer deux objets :

== Vérifie que deux objets sont issus d'une même classe

===

Interface

Une interface est une classe abstraite, mais ne contient pas de code.
souvant une interface est utiliser pour faire une action

Exemple :
Voler - un avion vole, un oiseau vole

Les exceptions

ce sont les erreurs en utilisant le try et catch

```
try {
        print("pas d'erreur");
}
 catch (PDOException $e) {
    print($e->getmessage());
}
```

Classe anonyme

C’est quand on utilise un objet sans le nommer

```
function(){
print("Je suis anonyme !");
}
()=>{
print("Je suis anonyme moi aussi hihi !");
}
```