Programmation orientée Objet

L'héritage

En php l'héritage est quand une classe dérive d'une autre classe.

La classe enfant va hériter de toutes les propriétés et méthodes public et protected de la classe parente.

Exemple :

```php
class Admin extends User{

    function __destruct(){
        echo "L'admin ".$this->login." est bien connecté.";
    }
}
```

Pour reprendre l'exemple du tp précédent, nous pouvons créer une classe Admin qui héritera d'User.

Il est possible de réécrire des méthodes du parent (SURCHARGE) dans l'enfant en leur donnant le même nom.

Il est également possible de bloquer l'héritage avec le mot clé final.

Constante

Une constante ne peut pas être changé une fois déclaré, pour la déclarer il faut utiliser le mot clé « const » suivi de son nom en majuscule. De base une constante est en public, mais vous pouvez les définir sur private ou protected. Exemple :

```php
  private const right = 2;
```

Pour accéder à une constante vous pouvez y accéder de cette façon :

- En dehors de la classe si elle est en public :

```php
-   Admin::RIGHT;
```

- Dans la classe comme ceci :

```php
-   self::RIGHT
```

Cherchez la différence entre this et self et complétez cette partie du cours .

Static

Les propriétés et méthodes statique peuvent être appelé sans créer une instance de la classe.

Il peut être utile de stocker des informations propres à tous vos objets directement dans la classe.

Ou de faire des fonctions qui ne nécessiterons pas d'instance de classe.

Exemple :

```
    protected static $articles;
```

```
public function createArticle($newArticle){
       self::$articles[] = $newArticle;
   }
   public function getArticles(){
      foreach(self::$articles as $article){
          echo $article."</br>";
      }
   }
```

```
$benjamin = new Admin("ben","admin");
$ophelie = new Admin("ophe","admin");

$ophelie->createArticle("Les joies de Wordpress");
$benjamin->createArticle("Php is the Best");

$benjamin->getArticles();
```

Le résultat retourné sera, les articles d'Ophélie et de benjamin, bien que ce soit Benjamin qui appelle la méthode.

Abstract

Une classe abstraite est une classe qui contient au moins une méthode abstraite.

Il faut utiliser le mot clé « abstract ».

Les méthodes abstraites vont être utilisées pour définir des sortes de rails, de chemins afin de forcer la classe enfant à avoir un certain comportement.

En effet si une classe étendu d'une classe abstraite ne contient pas les méthodes abstraites du même nom de la classe parente vous aurez une erreur.

```php
<?php
abstract class Animaux{
    public function hello(){
        echo "coucou";
    }
    abstract public function race();
}
class kangourou extends Animaux{
    public function race(){
        echo "kangourou";
    }
}

$kangoo = new kangourou;
$kangoo->race();
```

Interfaces

Une interface est similaire à une classe, sauf qu'elle ne peut pas contenir de code.

Une interface peut définir des noms de méthodes et des arguments, mais pas le contenu des méthodes.

Toute classe implémentant une interface doit implémenter toutes les méthodes définies par l'interface. Une classe peut implémenter plusieurs interfaces.

Une interface n'est pas une classe, on ne peut ni l'instancier ni l'hériter.

Les méthodes d'une interface sont toujours public.

```php
interface Action{
    public function avancer();
}
```

```php
class kangourou extends Animaux implements Action{
    public function race(){
        echo "kangourou </br>";
    }
    public function avancer(){
        echo "le kangourou saute </br>";
    }
}
```

Polymorphisme

Avec les classes abstraites et les interfaces vous ne le saviez peut-être pas mais vous avez vu ce qu'était le polymorphisme.

Ce mot parait compliqué mais le concept est relativement simple, en POO c'est un modèle de conception qui permet de créer des classes avec des fonctionnalités qui vont être implémenter à l'aide d'une interface commune (classes abstraites ou interface).