Cours Open-Classroom

Campus Numérique 2018 - Véronique ROUAULT

```php
<?php
    // Création de la classe Person ici :
    class Person {
    }
    // Création d'une instance de la classe $me :
    $me = new Person ();
    ?>
```

Les classes et les méthodes des objets

```php
<?php
    class Person {
        public $isAlive = true;

        function __construct($name) {
            $this->name = $name;
        }
        public function dance() {
        return "Je danse !";
        }
    }
    $me = new Person("Marc");
    if (is_a($me, "Person")) {
        echo "Je suis une personne, ";
    }
    if (property_exists($me, "name")) {
        echo "J'ai un nom, ";
    }
    if (method_exists($me, "dance")) {
        echo "et je sais comment danser !";
    }
    ?>
```

Renvoie :

```
Je suis une personne, J'ai un nom, et je sais comment danser !
```

méthodes utilisées intégrées :

is_a() savoir si un objet particulier est une instance d'une classe donnée;

property_exists() : savoir si un objet possède une propriété donnée; et

method_exists() : savoir si un objet possède une méthode donnée.

Héritage

L'héritage est un moyen pour une classe fille de récupérer les propriétés et les méthodes d'une autre classe (parent). Vous pourriez dire que la classe fille étend (ou spécialise) la classe parent.

C'est utilisé pour exprimer la relation "est-un", par exemple, un camion "est-un" véhicule, de sorte que le camion pourrait hériter de véhicule, mais une moto n'est pas un camion, il ne devrait pas hériter de camion (si les deux pouvaient hériter de véhicule).

Nous pouvons faire en sorte qu'une classe PHP puisse hériter d'une autre avec le mot-clé *extends* (étendre).

- vérifie si l'objet $square possède la propriété "hasSides" :

```php
<?php
    class Shape {
        public $hasSides = true;
    }

    class Square extends Shape {

    }

    $square = new Square();
    // Ajoutez votre code ici !
    if (property_exists($square, "hasSides")) {
        echo "J'ai des côtés !";
    }
?>
```

Ecraser une méthode parente

```
Nous avons une classe Vehicle.
La classe fille, Bicycle, écrase la méthode public function honk() de la
classe Vehicle et la remplace par une fonction qui retoune "Beep beep!".
Un'objet $bicycle (instance de la classe Bicycle) est créé.
le résultat de l'appel de sa méthode honk() est affiché.
```

```php
<?php
```

```
    class Vehicle {
        public function honk() {
        return "HONK HONK!";
        }
    }
    class Bicycle extends Vehicle {
        public function honk() {
            return "Beep beep !";
        }
    }
    $bicycle = new Bicycle();
    echo $bicycle->honk();
?>
```

Le mot de la fin : final

Pour qu'une classe parent empeche ses enfants d'écraser certaines de ses méthodes utiliser *final*

```
final
```

```
class Vehicle {
    final public function honk() {
            return "HONK HONK!";
          }
```

Les constantes de classe et l'opérateur de résolution de portée

```
const (pour constant)
```

Pour bien les diférencier des variables, dans la déclaration d'une constante le signe $ disparaît.

La valeur de la constante doit être une chaine de caractère ou un nombre

Le symbole :: est l'opérateur de résolution de portée. Cet opérateur permet d'utiliser la valeur d'une constante de classe. Il permet également de faire appel aux méthodes de la classe parente par exemple (via parent::myFunc()😉.

```
<?php
    class Person {
    }
    class Ninja extends Person {
    const stealth = "MAXIMUM";// définition de la constante
```

```
    }
    echo Ninja::stealth;// Affiche la valeur de la contante avec l'opérateur
de résolution de portée
?>
```

Le mot-clé Static (programmation avancée)

En déclarant une propriété de classe ou une méthode de classe avec ce mot-clé static, on autorise son utilisation sans avoir besoin d'instancer cette classe.

```
<?php
        class King {
          // Pourra être utilisée dans avoir besoin de l'instancer
          public static function proclaim() {
            echo "A kingly proclamation!";
          }
        }
        //Appel de la méthode
        echo King::proclaim();
      ?>
```

RESUME

Créer une classe:

```
class ClassName {
  // Du code
}
```

Définir une méthode de classe :

```
class ClassName {
  public function methodName() {
    // Du code
  }
}
```

Définir une classe fille d'une autre classe parente :

```
class SubClass extends SuperClass {
  // Du code
}
```

Définir une constante de classe :

```
class MyClass {
  const classConstant;
}
```

Utiliser le mot-clé static:

```
class MyClass {
  static public $property = "Property";
  static public function myMethod() {
    // Du code
  }
}
```

Utiliser l'opérateur de résolution de portée :

```
MyClass::$property;
MyClass::myMethod();
```

```

```