VISIBILITÉ

POO - PHP

INTRODUCTION

La visibilité constitue une des particularités élémentaires de la programmation orientée objet.

Elle permet de définir de quelle manière un élément sera accessible dans le programme.

Comme dans la plupart des langages Orienté Objet il y a trois niveau de visibilité :

- **Publiques**
- **Privées**
- **Protégées**

Peut être définie en préfixant la déclaration d'un élément (propriété, méthode ou constante) avec un des trois mot-clé précédent.

```php
public $propriete;
private function methode(){}
protected const CONSTANTE = "";
```

Par défaut si la visibilité n'est pas renseigner l'élément sera public.

PUBLIC

Le mot-clé ***public*** indique que l'élément est accessible depuis n'importe où dans le programme.

```php
class ClassName{
  public $p1;
}

$a = new ClassName();
$a->p1 = 'oui';
echo $a->p1;//oui
```

Dans cet exemple, nous pouvons lire et modifier directement la valeur de l'attribut publique.

il est peu recommandé de leur donner une visibilité publique car de cette manière il devient possible de les modifier sans qu'aucun contrôle ne soit effectué sur les valeurs.

Note: La seul fonction dans une classe qui doit **toujours** être public est le constructeur qui est appeler lors d'une création d'une instance d'un objet.

PRIVATE

Le mot-clé ***private*** signifie que les éléments ne seront visibles et accessibles que directement que depuis l'intérieur même de la classe.

```php
class ClassName{
  private $p1;
}

$a = new ClassName();
$a->p1 = 'oui';//error
echo $a->p1;
```

Dans cette exemple nous pouvons constater que il est imposible d'accéder ou modifier à un élément privée en dehors de la classe, c'est pourquoi nous utiliserons des accesseurs pour accéder au éléments privées.

LES ACCESSEURS

Les accesseurs sont des méthodes qui permette d'accéder et modifier des propriétés privée en dehors de la classe.

Se sont les méthodes set() et get(), set permet de modifier et get de récupérer la valeur d'une propriété.

```php
class ClassName{
  private $p1;

  function getP1(){
    return $this->p1;
  }

  function setP1($p){
    $this->p1=$p;
  }
}

$a = new ClassName();
$a->p1 = 'oui';//error
$a->setP1("oui");
echo $a->getP1();//oui
```

Note: cette méthode a un inconvenant, il faut faire une fonction get() et set() pour chaque paramètre.

Depuis PHP5 il existe les « fonction magique » qui permette de faire un seul get() et set() pour tous les paramètres:

function __set($property, $value){}

function __get($property)

PROTECTED

Le mot-clé **protected** est un intermédiaire entre le public et le private. Il permet d'accéder au éléments dans la classe et classes parentes/dérivées.

```php
class Vehicule{

  protected $_marque;
  protected $_estRepare;

  public function __construct($marque)
  {
    $this->_marque = $marque;
    $this->_estRepare = false;
  }
}
```

```php
class Voiture extends Vehicule{

  private $_volumeCarburant;

  public function __construct($marque){
    parent::__construct($marque);
    $this->_volumeCarburant = 40;
  }
}
```

Merci de votre attention