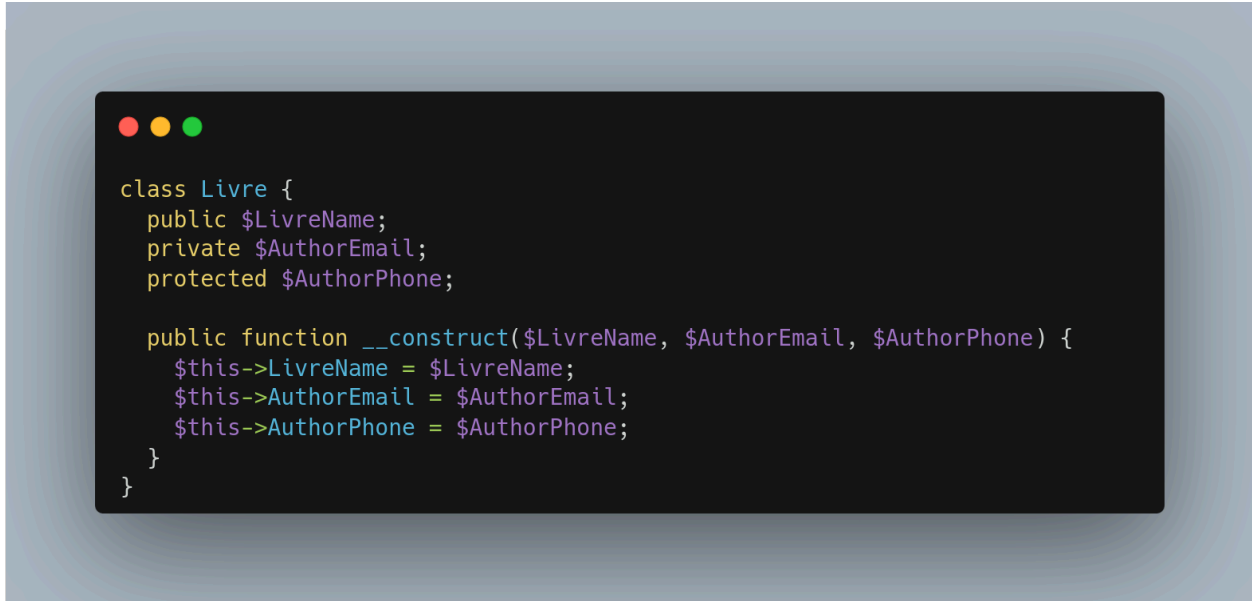


1- Introduction à la Programmation Orientée Objet en PHP:

Le Programmation Orientée Objet(POO), C'est Paradigme de programmation utilisé un style de développement différent que celle de procedural,dans le procédural le développeur Crée des fonctions pour atteint ces objectifs, sur le POO le développeur Crée des Class qui contient des Propriétés (variables) et méthodes (fonctions) Chose qui rend le code mieux organizer est lisible que s'il de Procedural.

En bref, le Class si comme un Moule de Gâteau et l'instance (object) si le Gâteaux il mémé .



```
class Livre {  
    public $LivreName;  
    private $AuthorEmail;  
    protected $AuthorPhone;  
  
    public function __construct($LivreName, $AuthorEmail, $AuthorPhone) {  
        $this->LivreName = $LivreName;  
        $this->AuthorEmail = $AuthorEmail;  
        $this->AuthorPhone = $AuthorPhone;  
    }  
}
```

2- Encapsulation et Modificateur d'accès:

L'encapsulation C'est un principe qui consiste à regrouper les données (propriétés et méthodes) d'un objet dans la même structure.

Les modificateur d'accès on peut dire ce sont les responsable de visibilité des propriétés il ya 3:

- Public : on peut y 'accéder ou modifier directement
- Privé: pour accéder on est besoin l fonction getter et pour modifier une fonction setter est nécessaire. Aussi en cas d'héritage les propriétés sont héritier mais on peut accéder les valeurs

- Protected: pour accéder on est besoin l fonction getter et pour modifier une fonction setter est nécessaire. Aussi en cas d'héritage les propriétés sont héritiers avec les valeurs.

```
class Livre {  
    public $LivreName;  
    private $AuthorEmail;  
    protected $AuthorPhone;  
  
    public function __construct($LivreName, $AuthorEmail, $AuthorPhone) {  
        $this->LivreName = $LivreName;  
        $this->AuthorEmail = $AuthorEmail;  
        $this->AuthorPhone = $AuthorPhone;  
    }  
  
    public getAuthorEmail() {  
        return $this->AuthorEmail;  
    }  
  
    public setAuthorEmail($Email) {  
        $this->AuthorEmail = $Email;  
    }  
}
```

3 - Héritage et Polymorphisme:

- **Héritage** : Un mécanisme en POO où une classe (classe enfant) hérite des propriétés et méthodes d'une autre classe (classe parent). Cela permet de réutiliser le code et d'étendre ses fonctionnalités
- **Polymorphisme** : La capacité des méthodes à se comporter différemment selon l'objet. Il permet d'utiliser une interface commune pour différents types d'objets.



```
class Animal {  
    public $name;  
  
    public function __construct($name) {  
        $this->name = $name;  
    }  
  
    public function speak() {  
        return "$this->name fait un bruit.";  
    }  
}  
  
class Dog extends Animal {  
    public function speak() {  
        return "$this->name aboie.";  
    }  
}  
  
class Cat extends Animal {  
    public function speak() {  
        return "$this->name miaule.";  
    }  
}  
}
```