Les concepts POO en PHP

classe :

```
class nomClass(){
        //methodes ;
        //propriétées ;
}
```
une classe est un ensemble composé de méthodes et de propriétées

objet :

```
$nomObjet = new nomClass() ;
$nomObjet→nomFonction() ;
```

l'objet permet par sont biais d'appelé nos méthodes de la classe sur lequel il a été crée et de ce fait de les faires fonctionner dans les situations que l'on souhaite

instance :

```
$nomObjet = new nomClass() ;
```

l'instanciation permet de crée un object avec une classe que l'on a crée plus tot dans le code

propriété :

```
public $nomVariable ;
private $autreNom ;
protected $encoreNom ;
```
les propriétés sont des variables qui vont pouvoir etre utilisé par nos méthodes qui sont dans la meme classe

méthode :

```
public function nomFonction(){
        //code qui fait fonctionner la méthode avec une valeur retourné ou non
}
```
une méthode permet de décrire en sont sein sont fonctionnement et d'etre appelé ensuite au besoin grace a sa classe

constructeur :

```
function __construct(){
        //choses a initialiser
}
```

le constructeur est toujours appelé à chaque création d'un objet et permets d'initialiser tout se dont on aura besoin pour l'object

statique :

```
public static function nomFonction(){

}
```

Avoir des méthodes et des propriétés statiques permet d'y acceder sans avoir besoin d'instancier la classe

visibilité :

```
public$nom ;
private $encoreUn ;
protected $unAutre ;
```

la visibilité est jusqu'a quel limite les propriétés peuvent etre utilisées (en public c'est de partout, en privé c'est seulement dans la classe et en protéger c'est dans la classe et les autres classe qui hérite)

getter :

```
public function getNom(){
        return $this→variable ;
}
```

les getters permettent de récuperer la valeur d'un attritbut d'une classe

setter :

```
public function getNom($variable){
        $this→variable =$variable ;
}
```

les setters permettent de changer la valeur d'un attritbut d'une classe

héritage :

```
class classeQuiHerite extend classeParent{

}
```

l’héritage permet d’hérité des méthodes de la classe parents sans avoir besoin de les redéfinir

<u>overridding :</u>

```
class parentNom{
public function faitQuelqueChose(){     //class parent
                return $this→maVariable ;
        }
}

public function faitQuelqueChose(){    //class fille
                return strtoupper($this→maVariable );
        }
}
```

grace a la surcharge (overridding) on redéfinit une méthode dans la classe fille qui existait déjà dans la classe parent