PHP Objet

Programmation Orientée Objet

- PHP reprend les grands principes de la POO
- Définition de classes avec attributs et méthodes
- Héritage
- Classes abstraites

Définition d'une classe

```
class Maclasse {
    // définition des attributs
    public $mapropriete;
    // définition  des méthodes
    public function mamethode() {
        echo $this -> mapropriete;
    }
}
```


Accessibilité des attributs

- Public : accès universel (de partout)
- Private : accès uniquement dans la classe
- Protected : accès dans la classe et dans ses classes dérivées (héritage)

Accessibilité des méthodes

- Public : la méthode est utilisable par tous les objets instances de la classe et de ses classes dérivées
- Private : la méthode est utilisable uniquement dans la classe qui la contient
- Protected : utilisable dans la classe et dans ses classes dérivées (héritage) mais par aucun objet

Constructeurs et destructeurs

- function __construct(arg1,…, argn) { }

La méthode est appelée automatiquement lors de l'instanciation d'un objet avec new(); les parametres sont ceux du constructeur.

- function __destruct(){ }

La méthode est appelée automatiquement après la destruction de l'objet avec unset, soit à la fin du script.

héritage

- Il est possible d'enrichir une classe en créant une classe dérivée à partir d'une classe existante en y ajoutant des propriétés et/ou des méthodes.

```
class Voiture {
    protected $marque;
    protected $type;
    public function set_marque($marque) {
        $this->marque = $marque; }
    public function set_type($type) { $this->type = $type; }
    public function get_marque() { return $this->marque; }
    public function get_type() { return $this->type; }
}
```


héritage

```
class Utilitaire extends Voiture {
    private $volume;
    public function set_volume($volume) { $this->volume =
    $volume; }
    public function get_volume() { return $this->volume; }
}
```

➔ Le mot clé parent permet d'accéder aux méthodes de la classe mère

parent::__construct() permet d'accéder au constructeur

Héritage

```
$voit1 = new Voiture();
$voit1->set_marque("Peugeot");
$voit1->set_type("308");
echo "la voiture est une ".$voit1->get_type()." de la marque :
    ".$voit1->get_marque();
$voit2 = new Utilitaire();
$voit2->set_marque("Peugeot");
$voit2->set_type("patner");
$voit2->set_volume("3 m3");
echo "la voiture est une ".$voit2->get_type()." de la marque :
    ".$voit2->get_marque()." avec un volume de :".$voit2-
    >get_volume();
```


Heritage

```
class Tourisme extends Voiture {
    private $nbplaces;
    public function set_nbplaces($nbplaces) {
        $this->nbplaces = $nbplaces; }
    public function get_nbplaces() { return $this->nbplaces; }
};
$voit1 = new Tourisme();
$voit1->set_marque("Peugeot");
$voit1->set_type("308");  $voit1->set_nbplaces("5");
echo "la voiture est une ".$voit1->get_type()." de la marque
    ".$voit1->get_marque()."avec ".$voit1->get_nbplaces()."
    places"."<br/>";
```


Heritage

- Si l'entreprise gère des véhicules de tourisme et des véhicules utilitaires, la classe voiture ne sert plus à définir des objets ; on peut alors la définir comme une classe abstraite qui empêchera d'instancier des objets de cette classe.
- **abstract** class Voiture { … }
- $voit3 = new Voiture(); → impossible

Exemple avec constructeur

```
class Tourisme extends Voiture {
    private $nbplaces;
    function __construct($marque,$type,$nbplaces) {
        $this->marque = $marque;
        $this->type = $type;
        $this->nbplaces = $nbplaces;
} …}
$voit4 = new tourisme("Peugeot","508","7");
echo "la voiture est une ".$voit4->get_type()." de la marque
    ".$voit4->get_marque()."avec ".$voit4->get_nbplaces()."
    places"."<br/>";
```

Typage des parametres

- PHP5 permet d'imposer un type pour les paramètres des méthodes mais uniquement si ce sont des objets ; le nom du paramètre est alors précédé du nom de la classe.

- function acheter_voiture(tourisme $voit) { … }