Chapitre 5

PHP Orienté Objet

Année universitaire: 2024/2025

1

Déclarer une classe en PHP

- Une classe est un modèle de données
  - famille d'objets, ou encore moule à objets;
  - tous les objets d'une même classe partagent les mêmes attributs et les mêmes méthodes.
- Le mot clé class permet de déclarer une classe d'objet.

```
class voiture
{
    //code de la classe
}
```

2

2

Déclarer des attributs

- class voiture

```
{
    public $marque = "trabant";
}
```

- public, protected, private sont supportés
  - L'un des trois est obligatoire ou le mot clé var (public)
- Affectation et même déclaration facultatives!!

3

3

Déclarer une méthode

- class voiture

```
{
    function freiner($force_de_freinage)
    {
     //code qui freine
    }
}
```

- public, protected, private sont supportés
- Implicitement « public »

4

4

Déclarer des constantes

```
class voiture
    {
        const ROUES_MOTRICES = 2;
    }
```

- Locale à la classe
- Convention classique: spécifier les constantes en majuscules dans le code pour mieux les identifier.

5

5

Remarques :

- L'opérateur **->**: cet opérateur permet d'accéder à un élément de tel **objet** ;
- L'opérateur **::** : cet opérateur permet d'accéder à un élément de telle **classe**.
- Généralement on utilise l'opérateur :: pour accéder à un attribut (ou une méthode) **statique** ou bien une **constante** de la classe.
- PHP fait une différence entre la comparaison simple « **==** » et complète « **===**».
  - Avec « == », PHP retournera vrai si les deux objets ont les mêmes attributs et valeurs, et s'ils sont des instances de la même classe.
  - Avec « === », PHP retournera vrai si les deux objets font référence au même objet de la même classe.

6

6

Instanciation

```
class voiture
        {
            const ROUES_MOTRICES = 2;
            public $marque;
            function freiner($force_de_freinage)
            {
              //code qui freine
            }
        }
$MaVoiture = new voiture();
```

- Les parenthèses sont optionnelles si le constructeur ne nécessite pas de paramètre.
- En PHP5 toute classe doit être déclarée avant d'être utilisée.

7

7

Accéder à un attribut

```
class voiture
    {
        public $marque = "trabant";
    }
$MaVoiture = new voiture();
echo $MaVoiture->marque;
// affiche trabant
```

8

8

Accéder à une méthode

```
class voiture
    {
        function klaxonner()
        {
        return "tut tut!!";
        }
    }
$MaVoiture = new voiture();
echo $MaVoiture->klaxonner();
// affiche "tut tut!"
```

9

9

Référence à l'objet en cours

```
class voiture
    {
        public $vitesse = 0;
        function avance( $temps)
        {
            echo "avance de ".$temps*$this->vitesse." km en ".$temps." h";
        }
    }
$MaVoiture = new voiture();
$MaVoiture->vitesse = 100; //la vitesse est de 100km/h
$MaVoiture->avance(2); // affiche "avance de 200 km en 2h"
```

10

10

•Les membres statiques

- ■Les propriétés **non** statiques peuvent être appelées en utilisant la syntaxe **->** (opérateur de l'objet): $this -> property
- **$this** est une référence à l'objet appelant (l'objet courant).
- Déclarer des propriétés ou des méthodes **statiques** permet d'y accéder **sans avoir besoin d'instancier la classe**.
- Comme les méthodes statiques peuvent être appelées sans qu'une instance d'objet n'ait été créée, la pseudo variable $this **ne peut être utilisée**.
- On peut accéder à des propriétés statiques à travers l'objet en utilisant **l'opérateur ::** .

11

11

Héritage en PHP5

```
class vehicule
    {
        public $marque = "";
        function avance()
        {
            //code qui fait avancer le véhicule
        }
        function freine()
        {
            //code qui fait freiner le vehicule
        }
    }
class voiture extends vehicule
    {
        function klaxonne()
        {
            //code qui fait klaxonner la voiture
        }
        // la classe voiture possède un attribut marque, une méthode freine et une méthode avance par héritage
    }
```

12

12

Redéfinition de méthode

- Les méthodes héritées peuvent être réécrites dans la classe fille

```
class voiture extends vehicule
{
    public $marque= « peugeot »;
    function klaxonne()
    { //code qui fait klaxonner la voiture }
    function avance()
    { //code qui fait avancer la voiture }
    function freine()
    { //code qui fait freiner la voiture }
}
```

13

13

Accéder à la classe courante

- class vehicule

```
{
    $roues = 4;
    function affiche()
    {
      return "a ".$this->roues." roues";
    }
}
class voiture extends vehicule
{
    function affiche()
    {
      echo "cette voiture ".parent::affiche();
    }
}
$v = new voiture();
$v->affiche(); //affiche cette voiture a 4 roues
```

14

14

Contrôle d'accès

- **public:** une méthode ou attribut publique est accessible depuis toute votre application
- **private:** une méthode ou attribut privée n'est accessible que depuis l'intérieur de la classe
- **Protected:** une méthode ou attribut publique est accessible depuis l'intérieur de la classe, et depuis toutes les classes dérivés

15

15

Classe abstraite

- Début d'implémentation d'une classe
- Non instanciable
- Toute classe contenant au moins une méthode abstraite doit être déclarée abstraite
- Seule la signature d'une méthode abstraite est déclarée (pas d'implémentation)
- Les classes dérivées doivent implémentées toutes les méthodes abstraites
- Les classes dérivées ne sont pas obligées d'implémenter les méthodes déjà implémenter dans la classe parent, et peuvent posséder leurs propres méthodes

16

16

Classe abstraite

```
abstract class vehicule
{
    abstract function avancer();
    function tourner($sens)
    {
        echo "tourne à ".$sens."<br />";
    }
}
```

17

17

Héritage d'une classe abstraite

```
class voiture extends vehicule
{
    function avancer()
    {
        echo "go!<br />";
    }
    function klaxonner()
    {
        echo "tut tut!<br />";
    }
}
```

18

18

Interface

- API (Application Programming Interface) qui spécifie quelles méthodes et variables une classe peut implémenter, sans avoir à définir comment ces méthodes seront gérées
- Non instanciable
- Seule les signatures des méthodes d'une interface sont déclarées (pas d'implémentation)
- Toutes les méthodes de l'interface doivent être implémentées
- Les classes peuvent implémenter plus d'une interface en séparant chaque interface par une virgule

19

19

Interface

```
interface peutAvancer
{
public function avancer();
public function freiner();
}
interface peutTourner
{
public function tourneGauche();
public function tourneDroite();
}
```

20

20

Implémentation d'une interface

```
class voiture implements peutAvancer, peutTourner
    {
    public function avance()
    {
    echo "on avance";
    }
    public function freine()
    {
       echo "on freine";
    }
    public function tourneGauche()
    {
       echo "on tourne à gauche";
    }
    public function tourneDroite()
    {
       echo "on tourne à droite";
    }
    function klaxonner()
    {
      echo "tut tut!<br />";
    }
}
```

21

Abstract VS Interface

- Aucun code n'est présent dans une interface
  - Une interface est donc une classe abstraite qui ne contiendrait que des méthodes abstraites.
  - Une classe ne peut dériver que d'une classe abstraite mais peut implémenter plusieurs interfaces.

22

Classe finale

- Ces classes et méthodes ne pourront jamais être héritées

```
class voiture extends vehicule
    {
        final function avancer()
        {
         echo "on avance";
        }
    }
final class voiture extends vehicule
    {
        public function avancer()
        {
         echo "on avance";
        }
    }
```

23

Les méthodes magiques

- Une méthode magique est une méthode événementielle c'est-à-dire qu'elle sera exécutée si un événement est déclenché, si non rien ne se passe.
- Les méthodes magiques documentées de PHP sont :

__construct	__destruct
__call	__callStatic
__get	__set
__isset	__unset
__sleep	__wakeup
__toString	__invoke
__set_state	

24

Les méthodes magiques

- Les méthodes magiques sont:
- ____**construct** est une méthode magique qui est déclenchée par l'événement instanciation d'une classe (création d'objet).
- ____**set()** est automatiquement sollicitée s'il s'agit d'une affectation de valeur
- ____**get()** s'il s'agit de demander une valeur.
- ____**unset**($name) est sollicitée lorsque unset() est appelée avec des propriétés inaccessibles.

25

25