UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Programmation Web Accessible
Dynamique PHP
Cours 06

Master 1 MIASHS

2019-2020

celine.jost@univ-paris8.fr

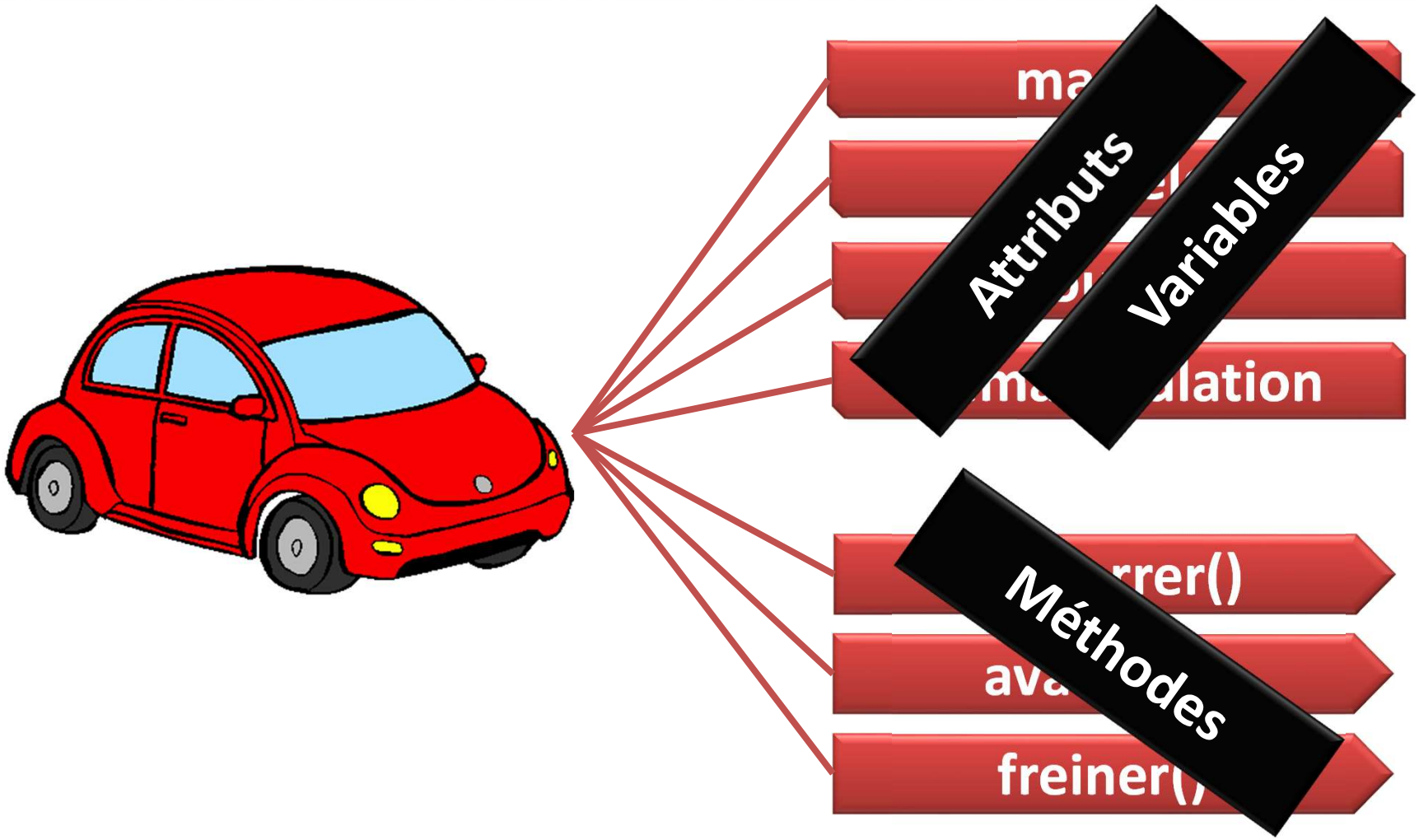
UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

PHP ET LA POO

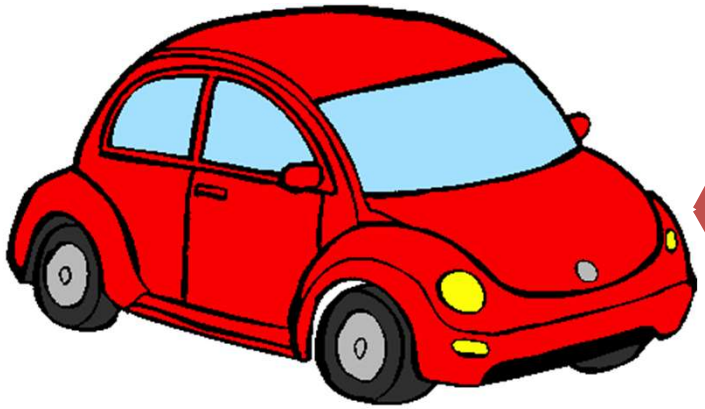
Programmation Orientée Objet

VOCABULAIRE DE BASE

Les objets



Les objets avec PHP



$maVoiture

- `$maVoiture->marque`
- `$maVoiture->modele`
- `$maVoiture->couleur`
- `$maVoiture->immatriculation`
- `$maVoiture->demarrer()`
- `$maVoiture->avancer()`
- `$maVoiture->freiner()`

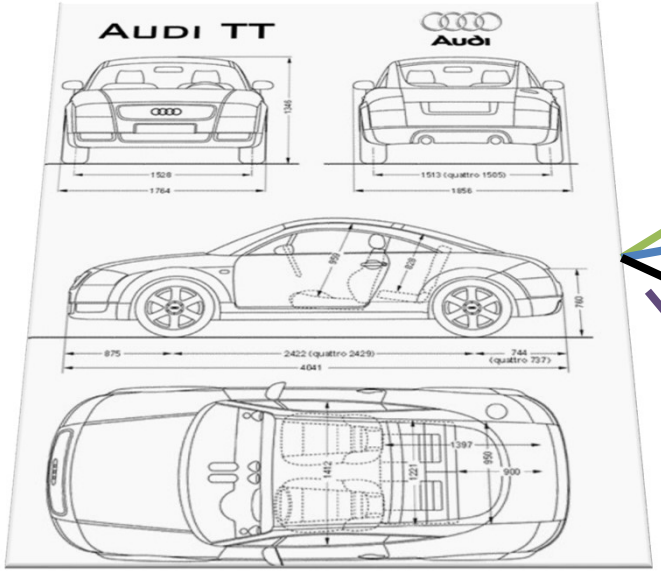
Les objets avec PHP

Petit exemple

$maVoiture

```
$maVoiture->couleur = "verte";

$maVoiture->marque = "Citröen";
$maVoiture->modele = "DS4";

$maVoiture->demarrer();

$maVoiture->avancer();
```

Les classes et les instances



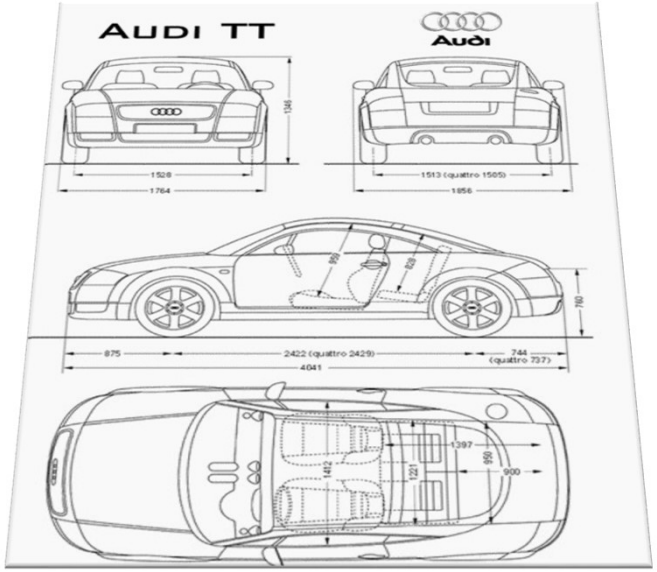
Classe

Instanciation



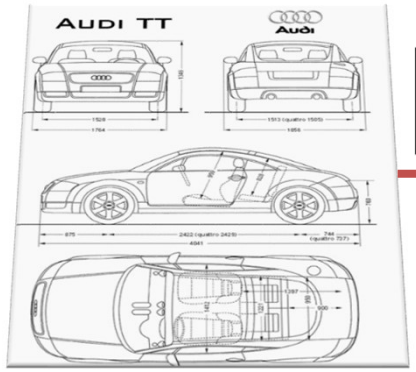


Les classes et les instances



Ca, c'est aussi une instance !

Bilan du vocabulaire



Instanciation

Instance

Attributs

Méthodes

Classe

Objet

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

PHP ET LA POO

Programmation Orientée Objet

ECRITURE EN PHP

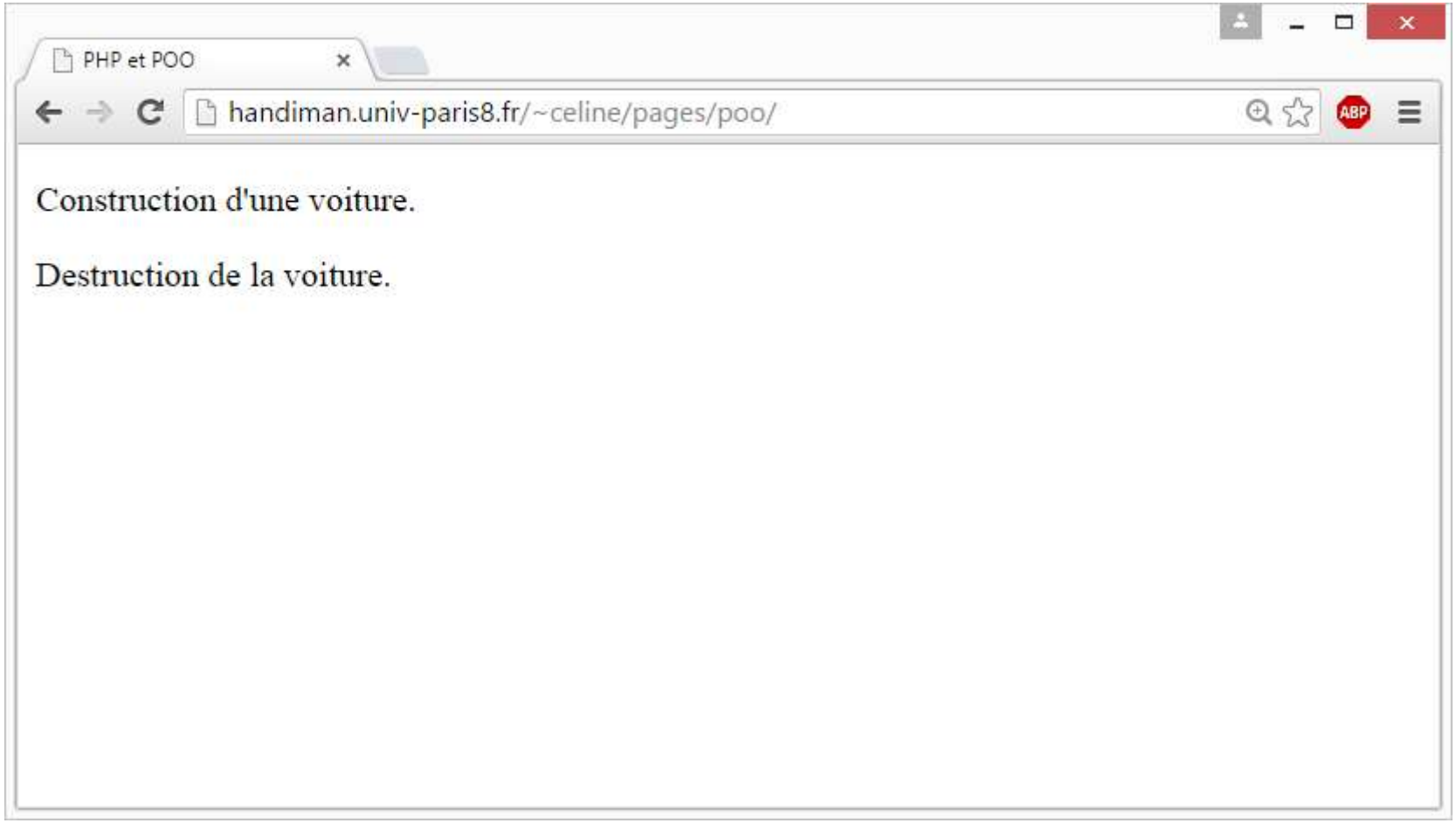
Ecrire une classe en PHP

```php
<?php

  class Voiture{

    public function __construct(){
      echo "<p>Construction d'une voiture </p>";
    }

    public function __destruct(){
      echo "<p>Destruction de la voiture </p>";
    }
  }

?>
```

Créer une instance en PHP

```php
<?php

  $maVoiture = new Voiture(); //appel implicite du
                              //constructeur



  //...



  //appel implicite du destructeur

?>
```

Dans le navigateur…



Les attributs en PHP

```php
<?php
  class Voiture{
    public $marque;
    public $couleur;

    public function __construct(){
      echo "<p>Construction d'une voiture </p>";
    }
    public function __destruct(){
      echo "<p>Destruction de la voiture $this->marque
de couleur $this->couleur </p>";
    }
  }
?>
```

Créer une instance en PHP

```php
<?php

  $maVoiture = new Voiture();

  $maVoiture->marque = "Citroen";
  $maVoiture->couleur = "verte";




?>
```

Dans le navigateur…

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

PHP ET LA POO

Programmation Orientée Objet

LA VISIBILITÉ (ENCAPSULATION)

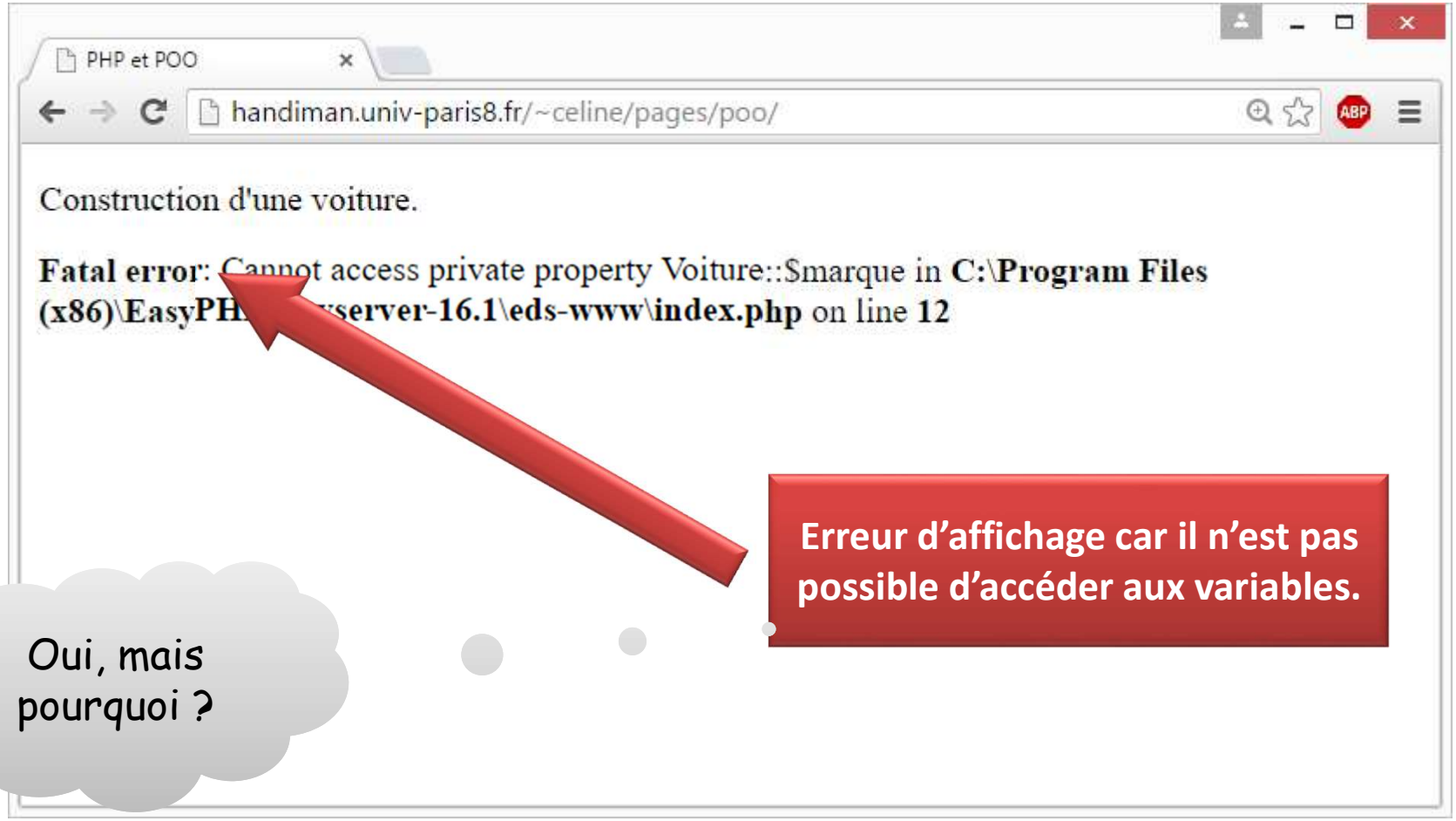
Visibilité : exemple

```
<?php
  class Voiture{
    private $marque;
    private $couleur;

    public function __construct(){
      echo "<p>Construction d'une voiture </p>";
    }
    public function __destruct(){
      echo "<p>Destruction de la voiture $this->marque
de couleur $this->couleur </p>";
    }
  }
?>
```

Visibilité : exemple

```php
<?php
  class Voiture{
    …
  }

  $maVoiture = new Voiture();
  $maVoiture->marque = "Citroen";
?>
```

Dans le navigateur…

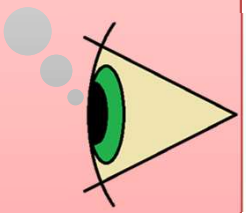


Visibilité : théorie

Dans la classe

```
<?php

  class Voiture{

  }

?>
```

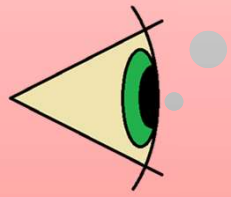
D'ici, je peux tout voir !



En dehors de la classe

```
<?php

  $maVoiture = new Voiture();

  ...

?>
```

D'ici, je ne peux voir que ce qui est accessible de l'extérieur.



UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Visibilité : théorie

Les éléments **private** d'une classe
ne peuvent être manipulés qu'à l'intérieur de la classe,
en lecture et en écriture.

Les éléments **public** d'une classe
peuvent être manipulés à l'intérieur et à l'extérieur de la classe,
en lecture et en écriture.

Visibilité : théorie

Comment accéder
en lecture ou en écriture
à une variable privée ?

Accès en lecture

Accesseur

```php
<?php
  class Voiture{
    private $couleur;
    public function __construct(){
      $this->couleur = "verte";
      echo "<p>Construction voiture $this->couleur.</p>";
    }
    public function getCouleur(){
      return $this->couleur;
    }
    public function __destruct(){
      echo "<p>Destruction voiture $this->couleur.</p>";
    }
  }
?>
```

Accès en lecture

Accesseur

```php
<?php
  class Voiture{
    private $couleur;
    public function __construct($couleur){
      $this->couleur = $couleur;
      echo "<p>Construction voiture $this->couleur.</p>";
    }
    public function getCouleur(){
      return $this->couleur;
    }
    public function __destruct(){
      echo "<p>Destruction voiture $this->couleur.</p>";
    }
  }
?>
```

Accès en lecture

Accesseur

```php
<?php

  $maVoiture = new Voiture("verte");

  echo "Couleur voiture : ".$maVoiture->getCouleur();

?>
```

Dans le navigateur…

Accès en écriture

Modificateur

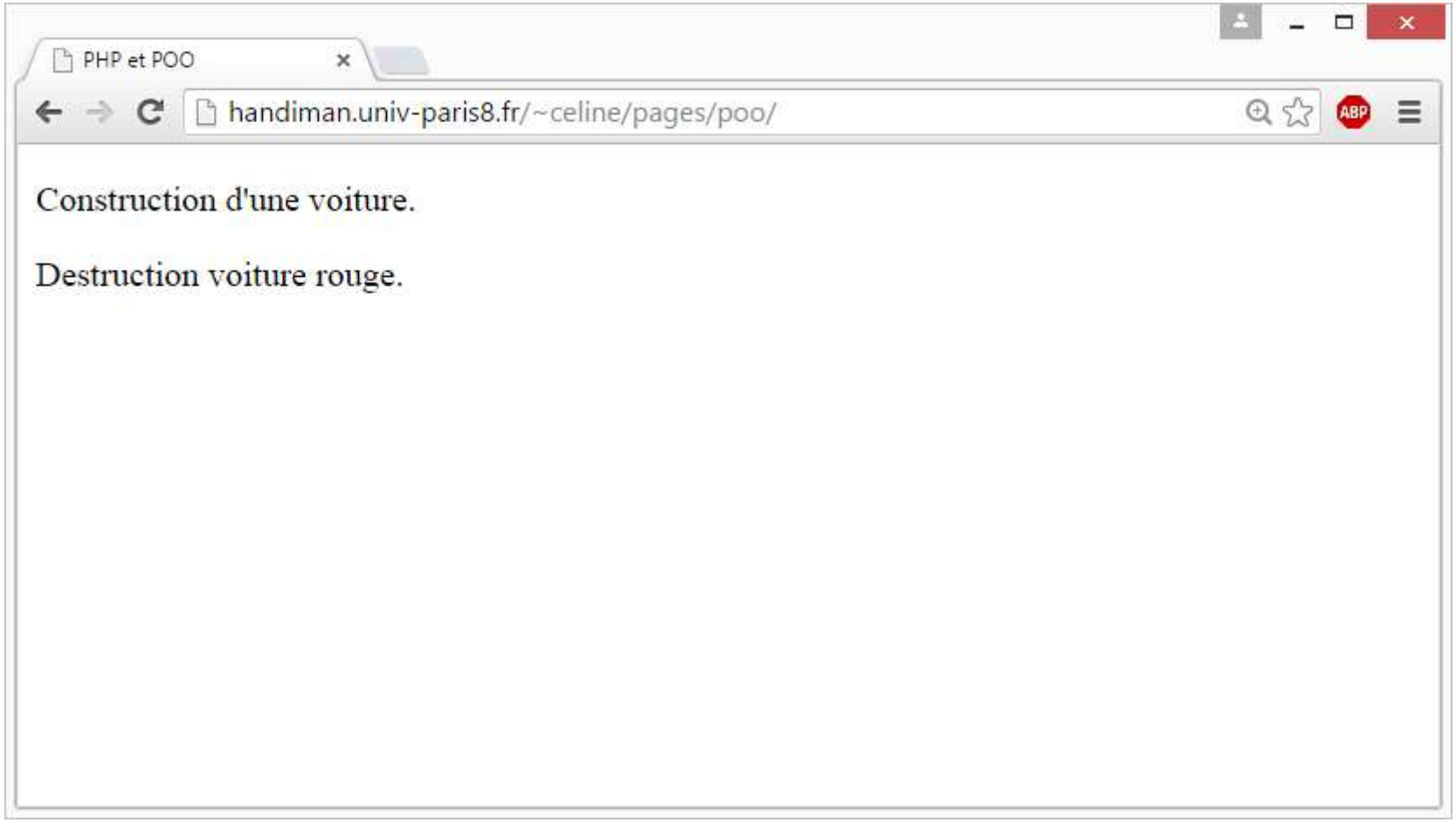
```php
<?php
  class Voiture{
    private $couleur;
    public function __construct(){
      echo "<p>Construction d'une voiture.</p>";
    }
    public function setCouleur($couleur){
      $this->couleur = $couleur;
    }
    public function __destruct(){
      echo "<p>Destruction voiture $this->couleur.</p>";
    }
  }
?>
```

Accès en écriture

Modificateur

```php
<?php

  $maVoiture = new Voiture();

  $maVoiture->setCouleur("rouge");

?>
```

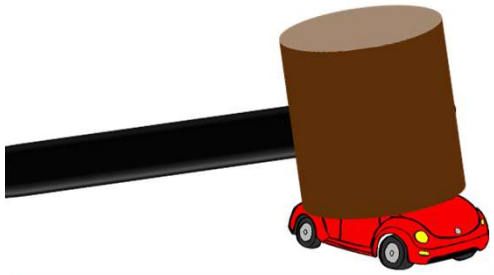
Dans le navigateur…

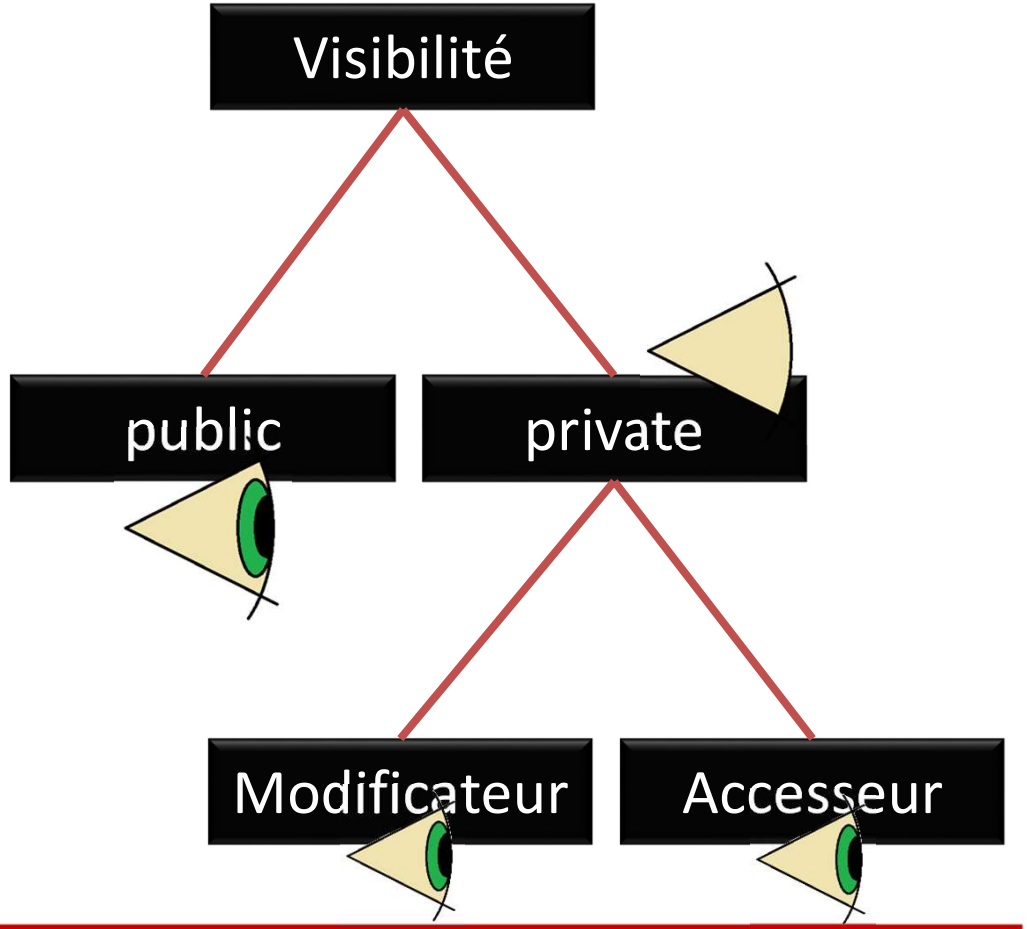


Bilan du vocabulaire

Constructeur

Destructeur





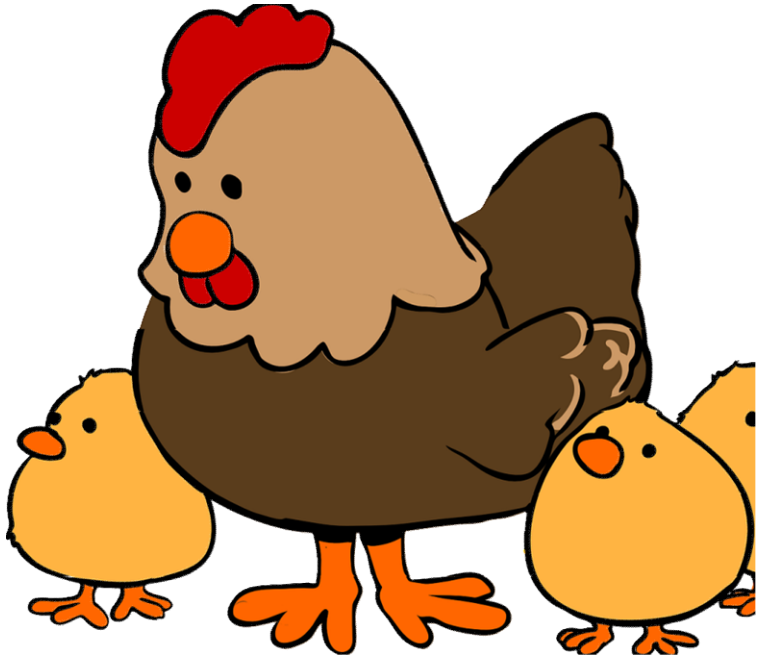
UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

PHP ET LA POO

Programmation Orientée Objet

HÉRITAGE

Un autre concept



Les objets peuvent avoir des parents et des enfants !!!!

Exemple concret

Etudiant
prénom
nom
setUniversité()
setEmploiTemps()
setPromotion()

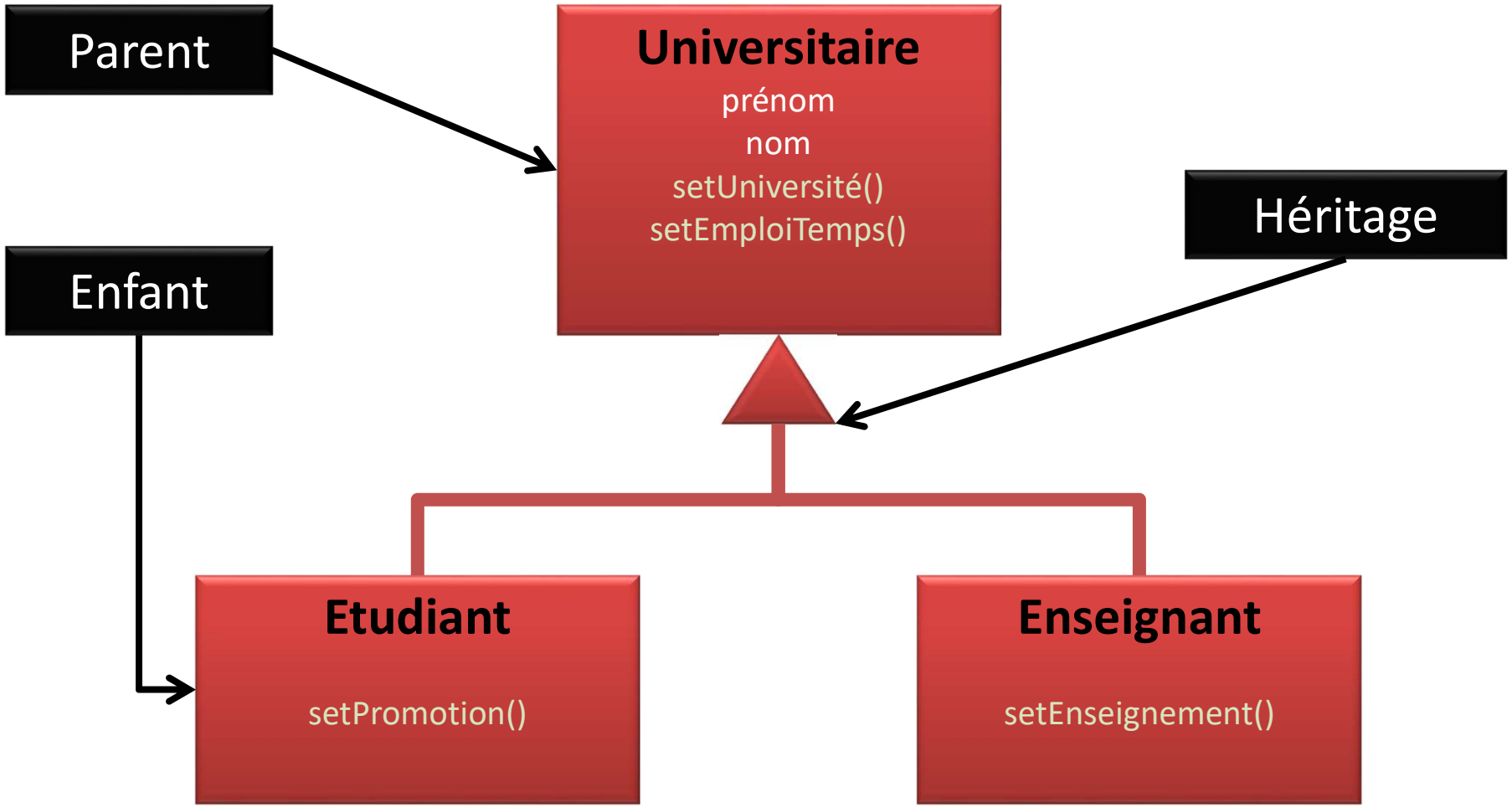
Enseignant
prénom
nom
setUniversité()
setEmploiTemps()
setEnseignement()

Ces classes ont des attributs et des méthodes communes.

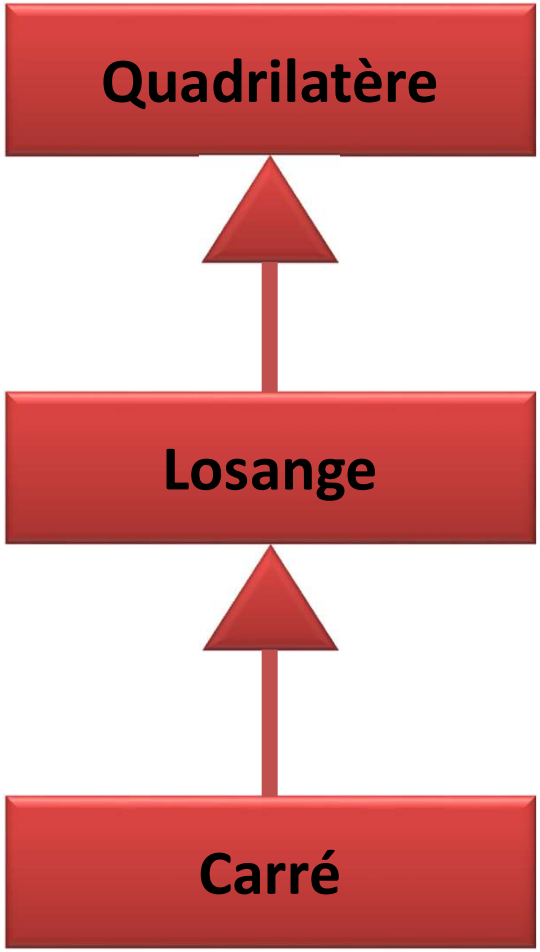
Elles ont un lien.

Exemple concret

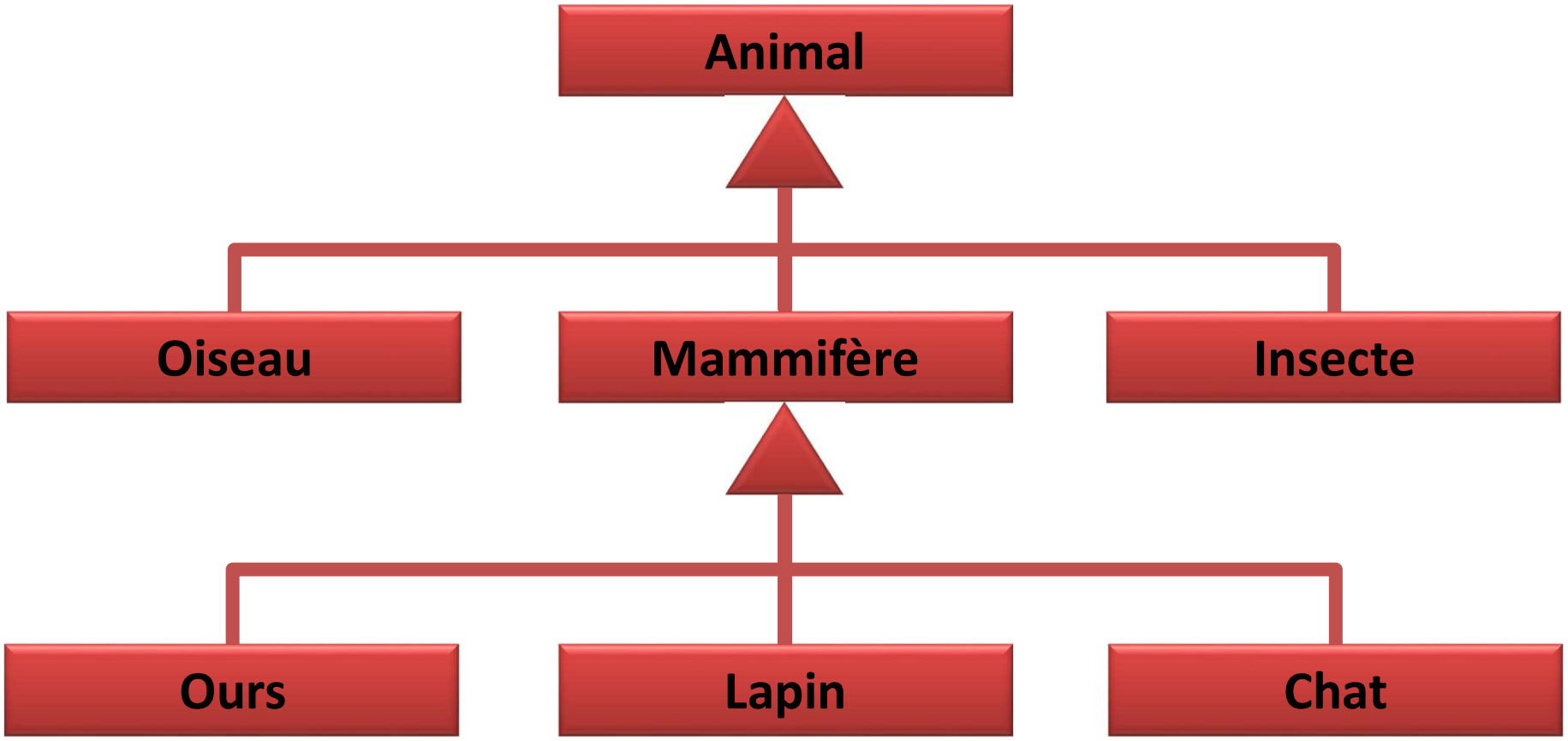
vision de la POO



Autre exemple



Encore un autre exemple



Héritage en PHP

```php
<?php
  class ClasseParent {
    public function __construct(){
      echo "<p>Construction d'un parent.</p>";
    }
    public function fonctionA(){
      echo "<p>Fonction A.</p>";
    }
  }
  class ClasseEnfant extends ClasseParent {
    public function __construct(){
      echo "<p>Construction d'un enfant.</p>";
    }
  }
?>
```

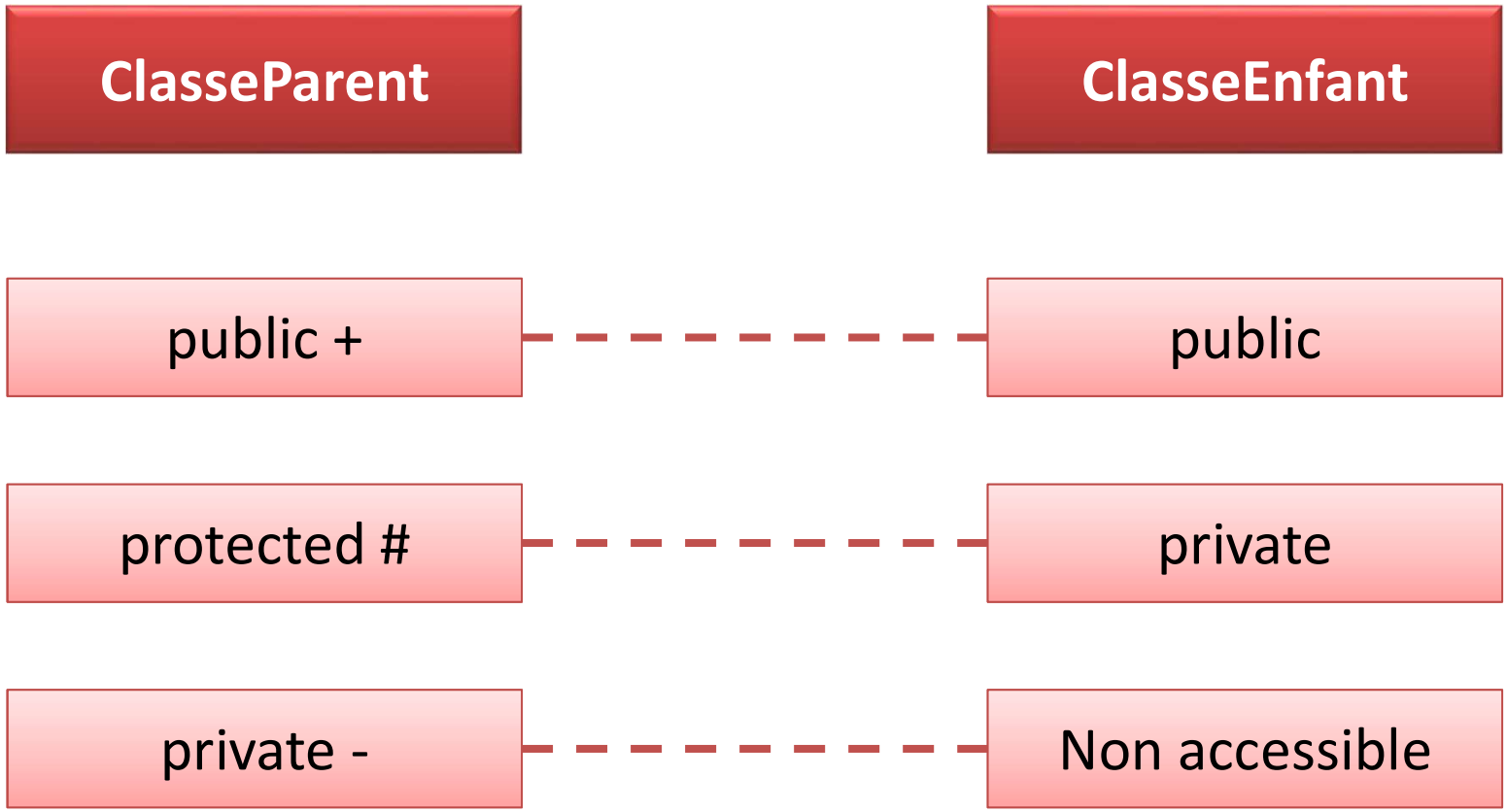
Héritage en PHP

```php
<?php

  $enfant = new ClasseEnfant();

  $enfant->fonctionA();

?>
```

Héritage en PHP

```php
<?php
  class ClasseEnfant extends ClasseParent {
    public function __construct(){
      parent::__construct();
      echo "<p>Construction d'un enfant.</p>";
    }
  }
  $enfant = new ClasseEnfant();
  $enfant->fonctionA();
?>
```

Héritage des visibilités



UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

PHP ET LA POO

Programmation Orientée Objet

ABSTRACTION ET INTERFACE

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Classe abstraite

Une classe abstraite ne peut pas être instanciée :
~~$maClasse = new ClasseAbstraite();~~

Une classe abstraite permet de définir un fonctionnement global, de factoriser du code.

Une classe enfant bénéficie des fonctions implémentées.
Une classe enfant a l'obligation d'implémentation les fonctions abstraites.

Classe abstraite

```php
<?php
  abstract class ClasseAbstraite {
    abstract public function uneFonction();
    abstract protected function uneAutreFonction();

    public function fonctionA() {
      //Ici, il y a du code
    }
  }

  class ClassEnfant implements ClasseAbstraite{
    ...
  }
?>
```

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Interface

Une classe interface ne contient que des signatures de méthodes.

Elle est plus générique qu'une classe abstraite, et permet une compatibilité entre différentes classes.

Une classe enfant a l'obligation d'implémentation les fonctions.

Interface

```php
<?php
  interface iMachine{
    public function allumer();
    public function verifierEtat();
    public function appelerReparateur();
    public function eteindre();
  }

  class Extrudeur implements iMachine{
    public function allumer(){
      ...
    }
    ...
  }
?>
```

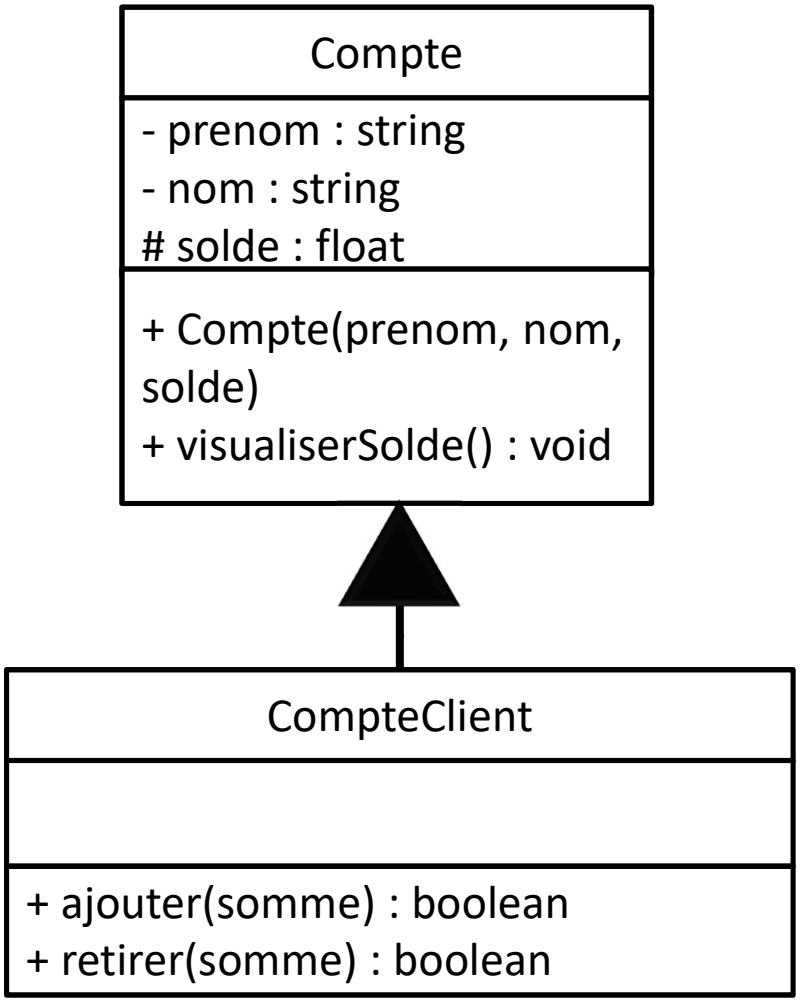
Référence

Ceci n'est qu'une introduction !

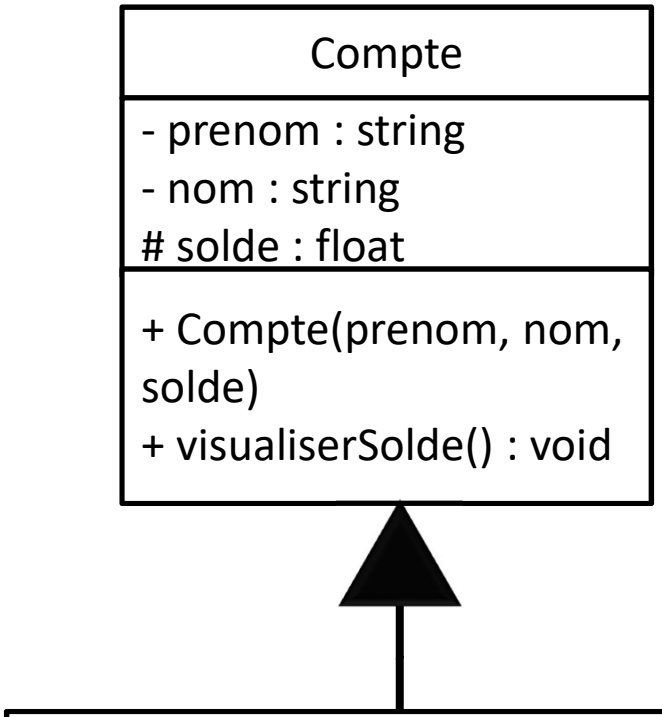
Vous trouverez les informations complètes ici :

http://php.net/manual/fr/language.oop5.php

Travail à faire !

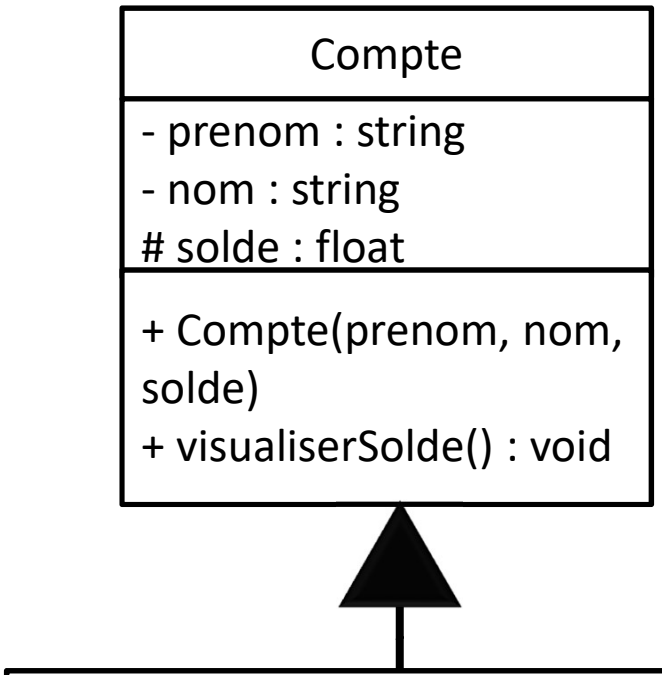


Travail à faire !



Comment on lit ce diagramme :
Il existe une classe Compte qui possède trois attributs. Il y a deux attributs privés de type chaîne de caractère qui sont « prenom » et « nom » et un attribut protégé de type flottant qui est « solde ». Le constructeur prend en paramètre un prenom, un nom, et un solde). La classe Compte possède une méthode publique « visualiserSolde() » qui ne renvoie rien.
Il existe une classe CompteClient qui hérite de Compte. Cette classe possède deux méthodes publiques : la méthode « ajouter(somme) » qui retourne un booléen et la méthode « retirer(somme) » qui retourne un booléen.

Travail à faire !



Comment on lit ce diagramme :
Il existe une classe Compte qui possède trois attributs. Il y a deux attributs privés de type chaîne de caractère qui sont « prenom » et « nom » et un attribut protégé de type flottant. Le constructeur prend en paramètre un prenom, un nom, et un solde). La classe Compte possède une méthode publique « visualiserSolde().
Il existe une classe CompteClient qui hérite de Compte. Cette classe possède deux méthodes publiques : la méthode ajouter(somme) qui retourne un booléen et la méthode retirer(somme) qui retourne un booléen.

Dans Index.php :
1. Créer un compte client $compte1 avec 300 €
2. Ajouter 250 €
3. Visualiser le solde de $compte1
4. Retirer 125 €
5. Visualiser le solde de $compte1