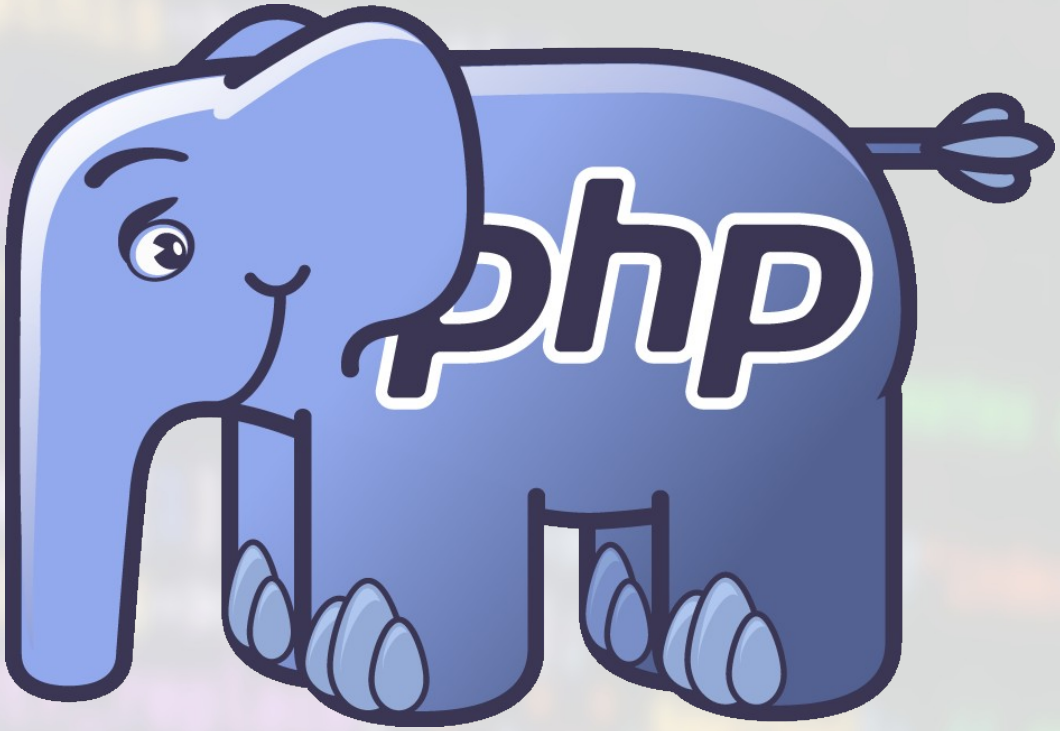
Programmation Orientée Objet en PHP



RAPPELS DE DÉVELOPPEMENT OBJET

Développement objet

- Définition de briques logicielles représentant un concept, une idée ou une entité ainsi que leurs interactions : les objets
- Un objet est une structure de données comprenant également les fonctionnalités de traitement des données
- L'objet est vu au travers de ses spécifications
- Les concepts associés sont :
  - Encapsulation
  - Héritage
  - Polymorphisme

Classe

- Une classe définit un modèle, un moule, à partir duquel tous les objets de la classe seront créés
- La classe décrit les données internes ainsi que les fonctionnalités des objets
- La classe est une vision « inerte »(figé), une recette de cuisine, visant à décrire la structure et le comportement des objets qui seront créés
- La construction d'un objet à partir de la classe génératrice s'appelle instanciation
- Les objets, entités « vivantes » en mémoire, sont des instances de la classe

Instanciation

- La classe est une description « inerte »

- Les objets doivent être instanciés à partir de la classe génératrice pour exister et devenir fonctionnels

- Exemple : la classe Animal
  $bambi = new Animal() ;
  $scrat = new Animal() ;
  $melman = new Animal() ;

Encapsulation

- Procédé consistant à rassembler les données et les traitements au sein des objets
- L'implémentation interne des objets est cachée
- Les objets sont vus à travers leurs spécifications
- Les données internes et les fonctionnalités possèdent un niveau de visibilité et peuvent éventuellement être masquées :
  - Public
  - Privé
  - Protégé

Encapsulation

- Les données internes des objets sont appelées attributs (ou propriétés voire champs)
- Les fonctionnalités des objets sont appelées méthodes
- Méthodes habituelles :
  - Constructeur / destructeur
  - Accesseurs / modificateurs (getters / setters)
- Référence à l'objet courant dans la description de la classe : $this

Visibilité

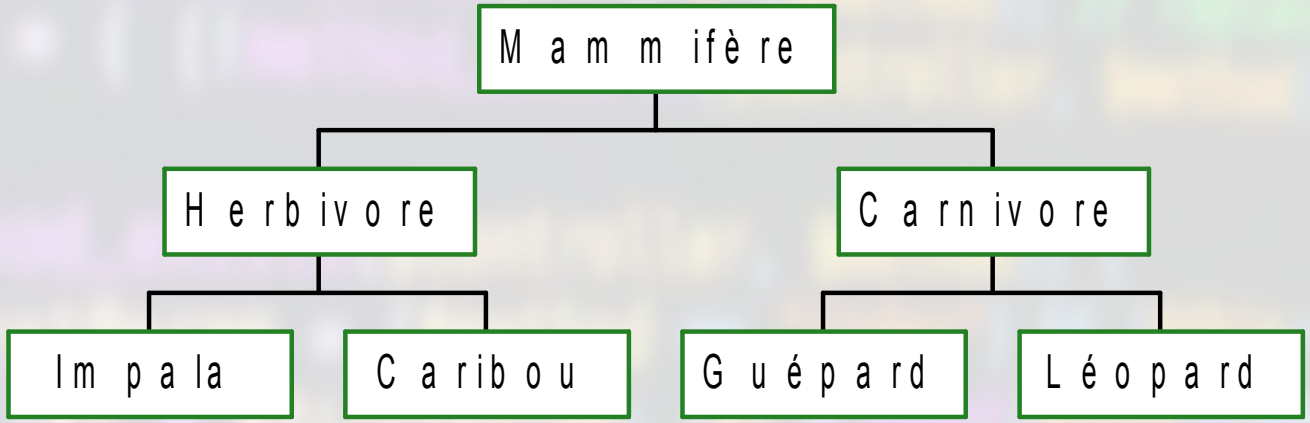
- Publique :
  Les données internes et les méthodes sont accessibles par tous

- Protégé :
  Les données internes et les méthodes sont accessibles aux objets dérivés

- Privé :
  Les données internes et les méthodes ne sont accessibles qu'aux objets de la classe

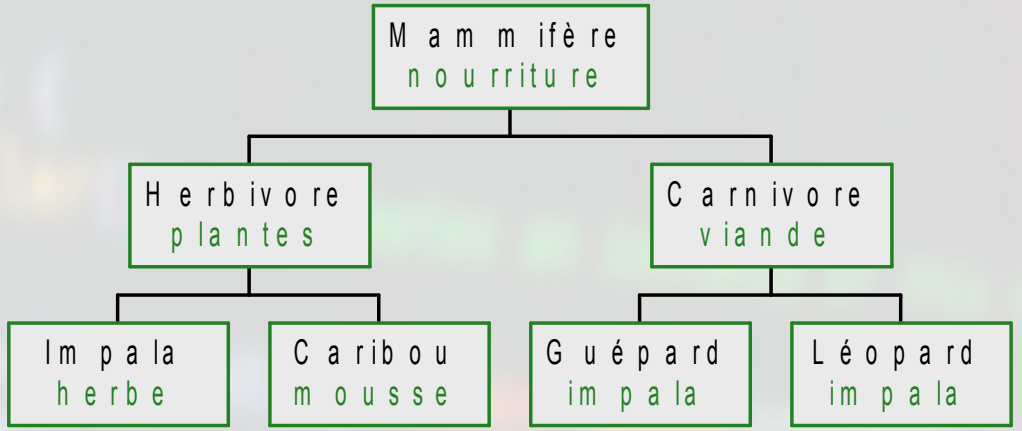
Héritage ou dérivation ou extension

- Création de nouvelles classes à partir du modèle d'une classe existante
- La nouvelle classe possède tous les attributs et méthodes de la classe mère
- La nouvelle classe peut proposer de nouveaux attributs et de nouvelles méthodes ou spécialiser des méthodes de la classe mère



Polymorphisme

- Choix dynamique de la méthode qui correspond au type réel de l'objet
- Méthode **`mange()`**



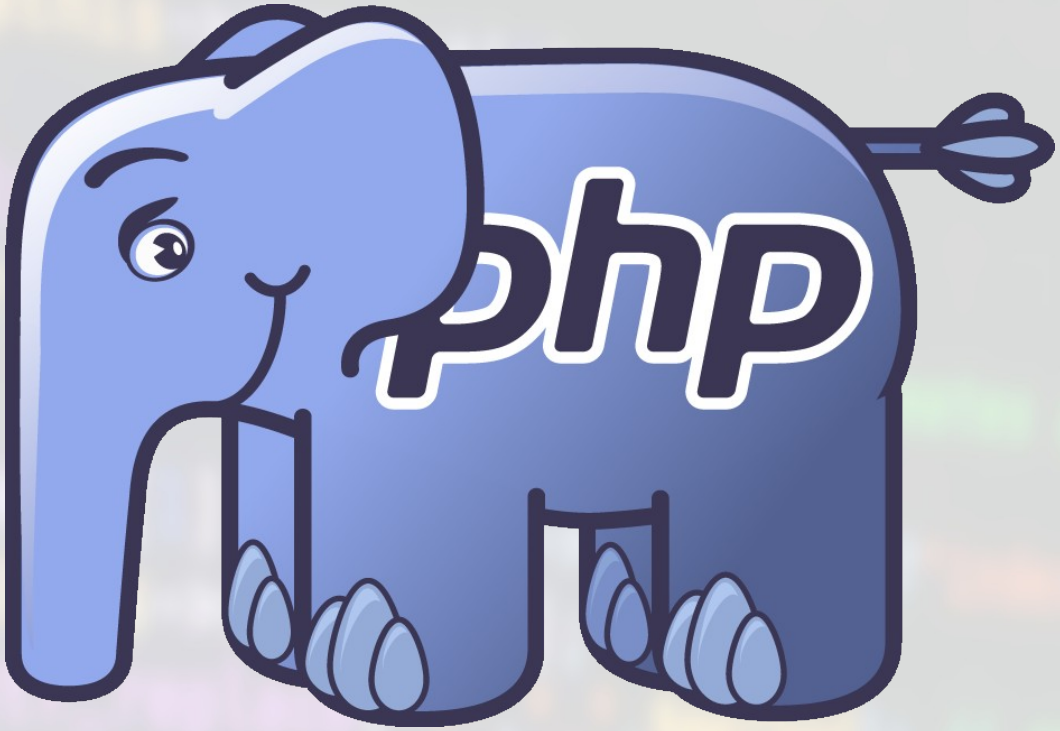

```
function nourriture(Mammifere $m) {
   return $m->mange() ;
}
```

```
$i = new Impala() ; nourriture($i) ;
```

```
$c = new Carnivore() ; nourriture($c) ;
```

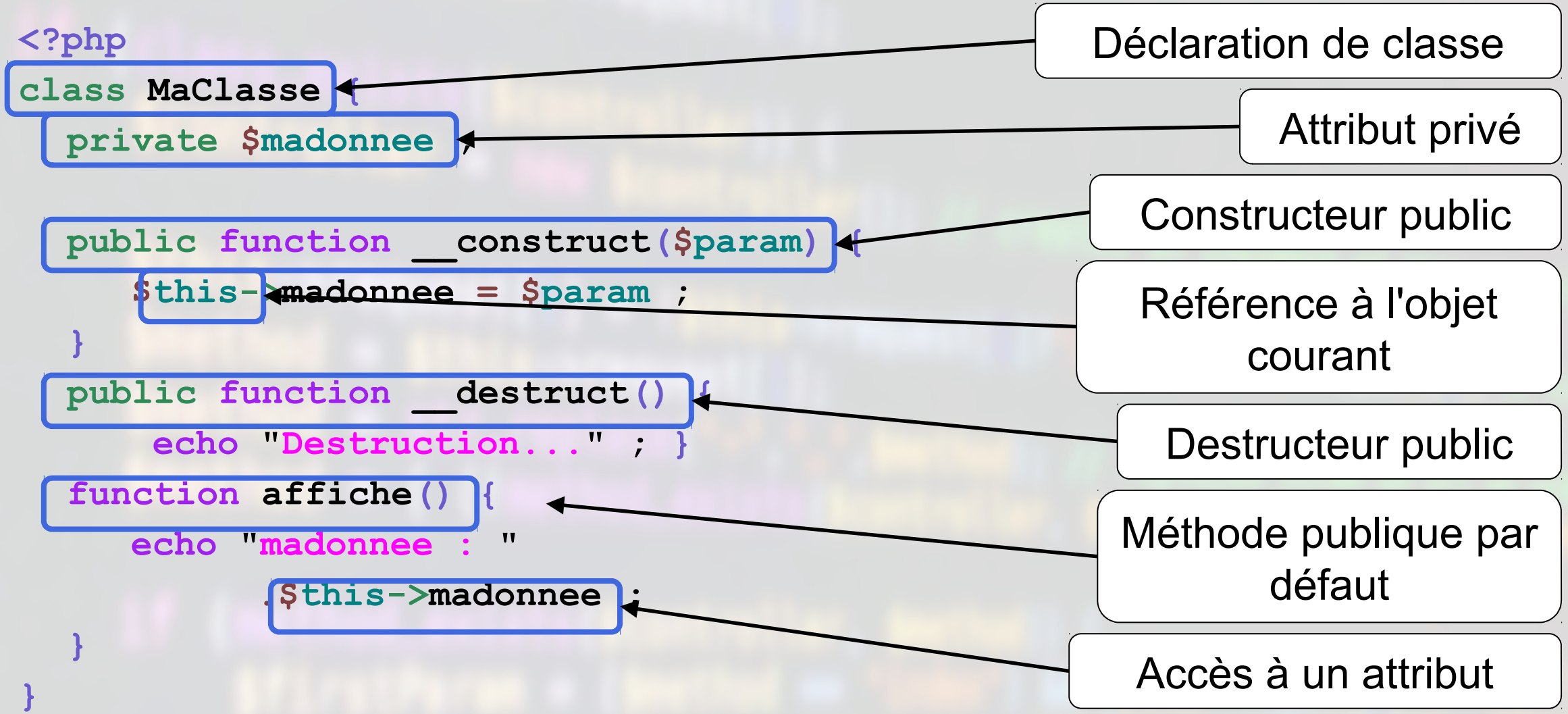
herbe

viande



CLASSE, ATTRIBUTS ET MÉTHODES D'INSTANCE

Définition d'une classe




```php
<?php
class MaClasse {
  private $madonnee ;

  public function __construct($param) {
      $this->madonnee = $param ;
  }
  public function __destruct() {
       echo "Destruction..." ; }
  function affiche() {
      echo "madonnee : "
           .$this->madonnee ;
  }
}
```

Utilisation d'une classe

```php
<?php
require "maclasse.class.php" ;

// Nouvel objet
$o = new MaClasse(12) ;
// Utilisation d'une méthode
$o->affiche() ;
```

```
maclasse contient 12
```

```php
class MaClasse {
   private $madonnee ;

      function __destruct() {
     echo "Destruction..." ; }
```

Inclusion de la définition de la classe

Création d'un objet

Méthode affiche de l'objet $o

L'attribut est privé

Destruction de l'objet $o

Valeur par défaut des attributs

```php
<?php
class MaClasse {
   private $madonnee = "Défaut" ;
   public function affecte($val) {
       $this->madonnee = $val ; }
   public function affiche() {
       echo "madonnee : ".$this->madonnee ; }
}
$o = new MaClasse() ;
$o->affiche() ;
$o->affecte("Nouvelle") ;
$o->affiche() ;
```

madonnee : Défaut

madonnee : Nouvelle

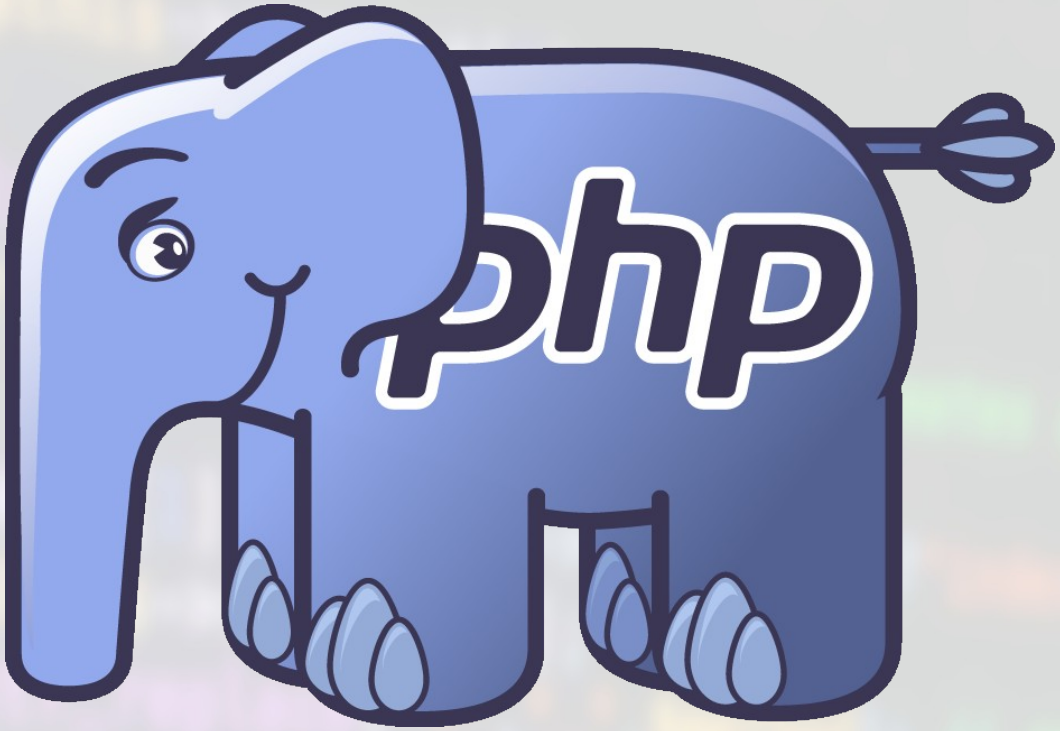
Attribut avec valeur par défaut

Nouvel objet

Affichage

Affectation

Affichage



ATTRIBUTS ET MÉTHODES DE CLASSE

Attributs et méthodes de classe

- Mot clé `static`
- Attributs et méthodes utilisables sans instance de la classe (=attributs et méthode de classe)
- Attributs NE peuvent PAS être accédés depuis une instance (~~`$objet->attribut`~~)
- Attributs partagés par toutes les instances de la classe
- Méthodes peuvent être accédés depuis une instance (`$objet->methode()` mais c'est mal !)
- Dans les méthodes, `$this` n'est pas disponible

Attributs statiques

```php
class MaClasse {
  private static $n = 0 ;
   public function __construct() {
      echo ++MaClasse::$n
           ." instance(s)" ; }
   public function __destruct() {
      echo "destruction" ; self::$n-- ; }
}
```

Attribut privé statique : ne peut être accédé que par des méthodes de la classe

Accès à l'attribut statique

```
1 instance(s)
2 instance(s)
destruction
2 instance(s)
Fatal error: Cannot access private property
MaClasse::$n in dummy.php on line 37
```

Méthodes statiques

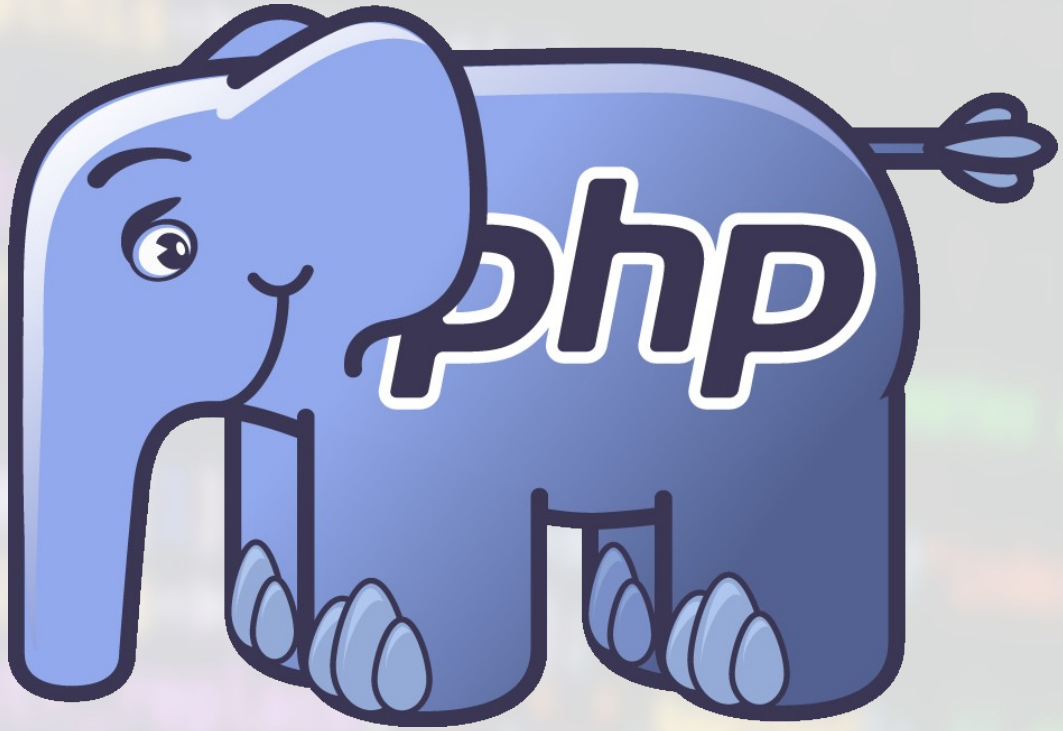
```php
class MaClasse {
   private static $n = 0 ;
   public function __construct() {
      echo ++MaClasse::$n." instance(s)\n" ; }
   public function __destruct() {
      MaClasse::$n-- ; }
   public static function f($i) {
      echo "Dans f() : ".($i*$i) ; }
}
```

Méthode publique statique

```
1 instance(s)
Dans f() : 4
Dans f() : 9
```

Appel à partir d'une instance
Toléré

Appel sans instance



CONSTANTES

Constantes de classe

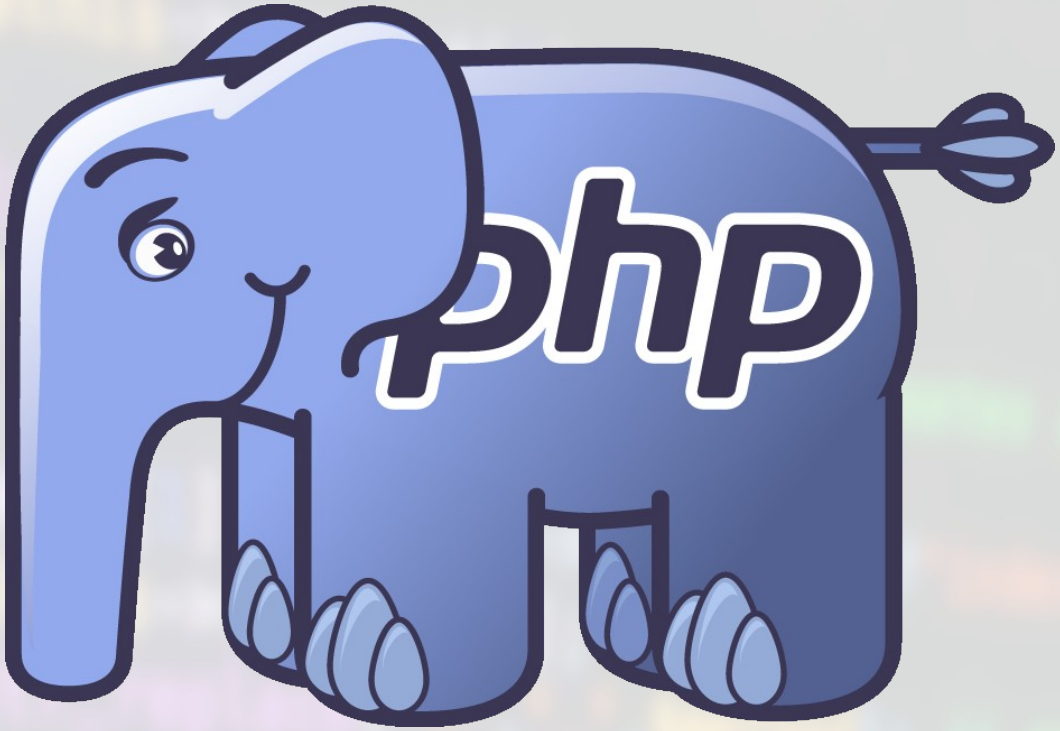
```php
class MaClasse {
   const constante = "Valeur" ;
    public function montre() {
        echo self::constante ;
    }
}

$c = new MaClasse() ;
```

```
Valeur
Valeur
```

Constante publique de classe

Accès à la constante de classe depuis la classe

Accès à la constante de classe à l'extérieur de la classe



HÉRITAGE

Héritage

```php
class Simple {
    function affiche() {
        echo "Je suis Simple" ; }
}
class Etendue extends Simple {
    function affiche() {
        parent::affiche() ;
        echo " mais aussi Etendue" ;
    }
}
$s = new Simple() ;
$e = new Etendue() ;
```

```
Je suis Simple
Je suis Simple mais aussi Etendue
```

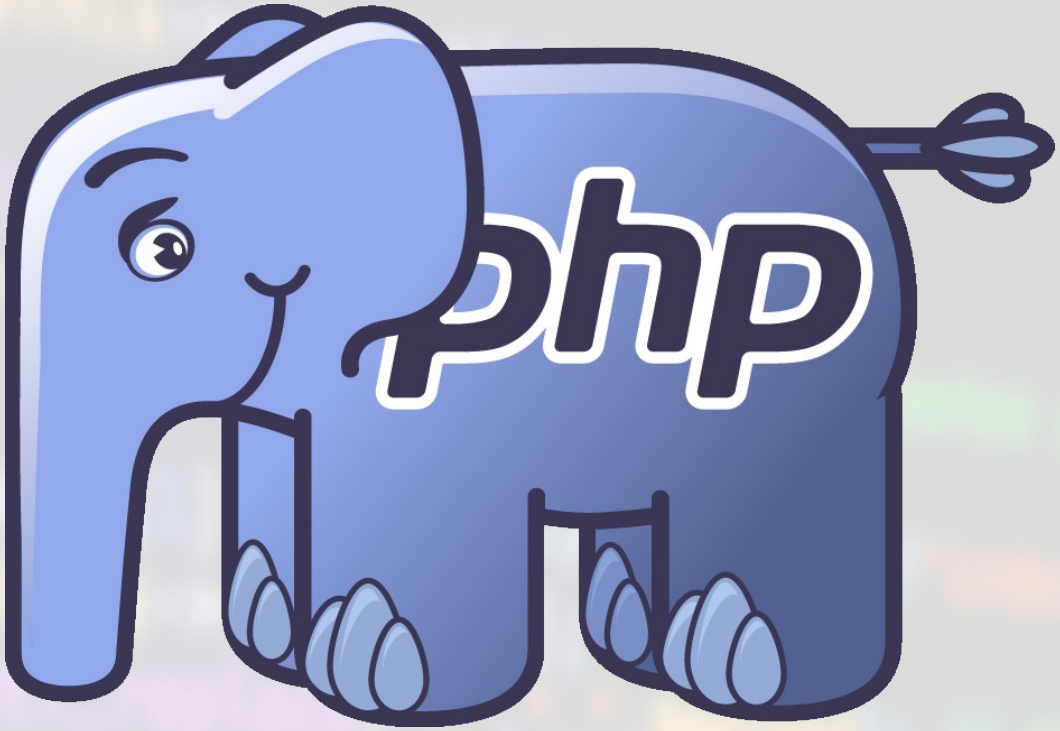
Classe simple

Une méthode publique

Classe étendue héritant de la classe simple

Surcharge de la méthode

Appel de la méthode du parent



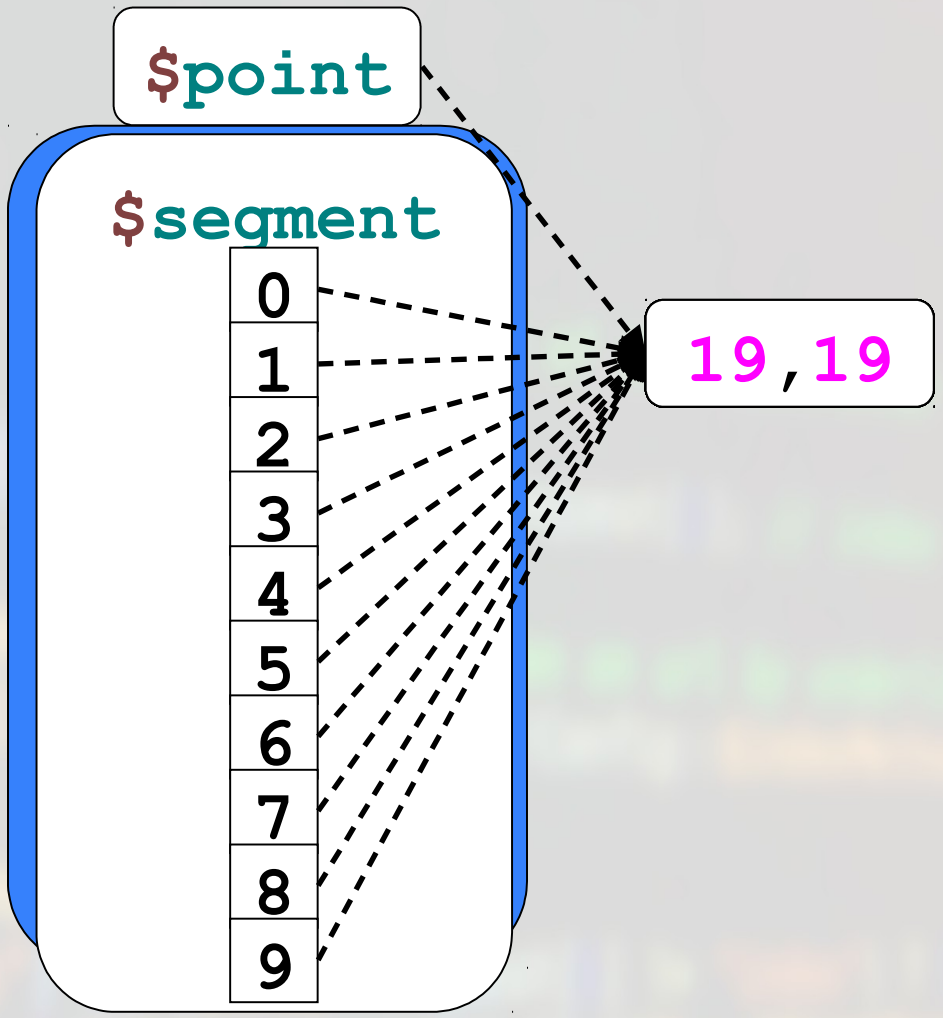
AFFECTATION, CLONAGE, PASSAGE DE PARAMÈTRES

Assignation d'objets

```
class Point {
   private $_x ;
   private $_y ;

   public function __construct($x=0, $y=0) {
      $this->_x = $x ;
      $this->_y = $y ; }

   public function set($x, $y) {
      $this->_x = $x ;
      $this->_y = $y ; }

   public function toString() {
      return "({$this->_x}, {$this->_y})" ; }
}
```

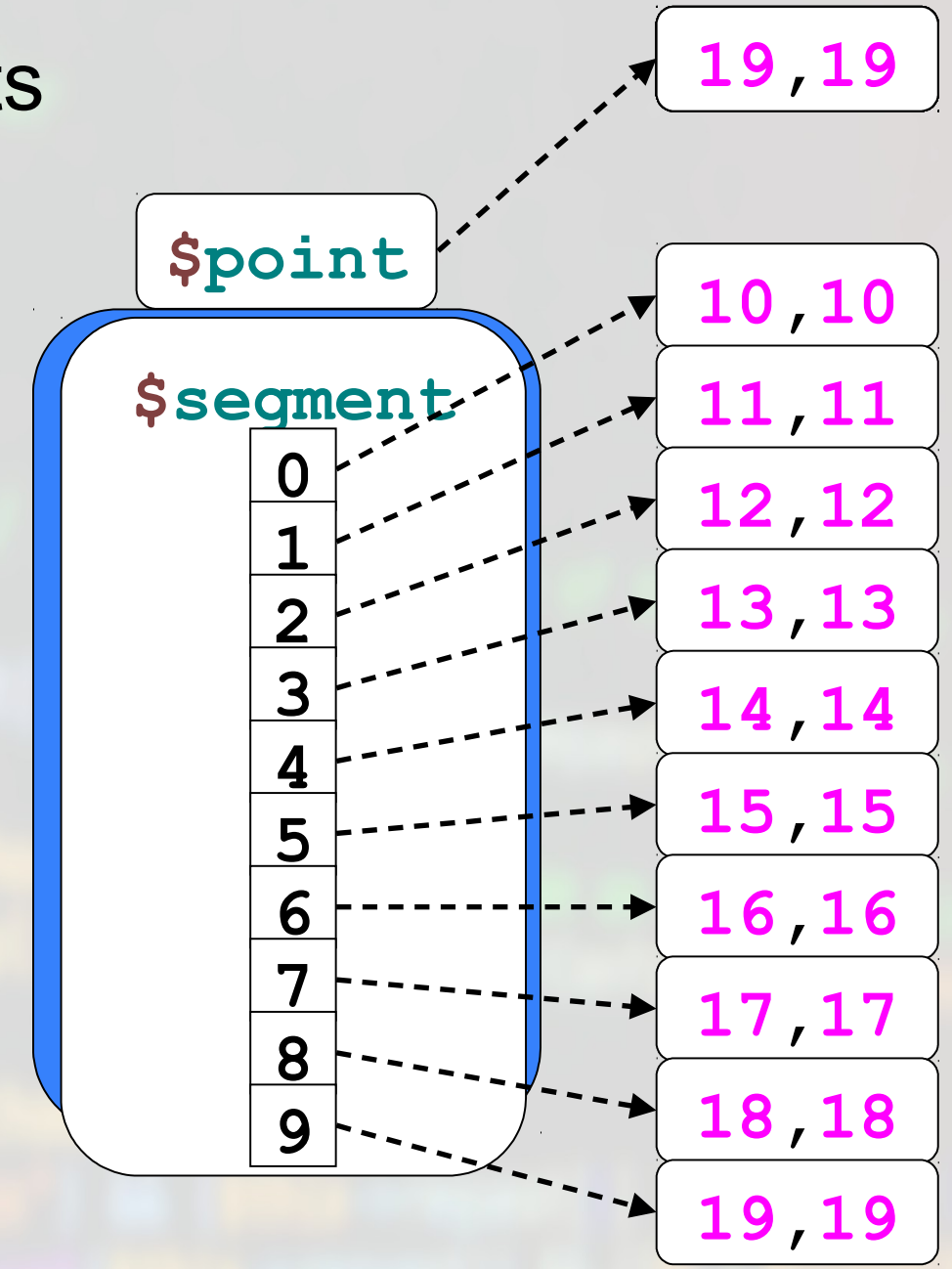
Assignation d'objets

```php
$segment = array() ;
$point = new Point() ;
for ($i=10; $i<20; $i++)
{
   $point->set($i, $i) ;
   $segment[] = $point ;
}

foreach ($segment as $k => $p)
   echo "$k: {$p->toString()}\n" ;
```



Clonage d'objets

```php
$segment = array() ;
$point = new Point() ;
for ($i=10; $i<20; $i++)
{
   $point->set($i, $i) ;
   $segment[] = clone $point ;
}

foreach ($segment as $k => $p)
   echo "$k: {$p->toString()}\n" ;
```



Objets comme arguments de fonctions

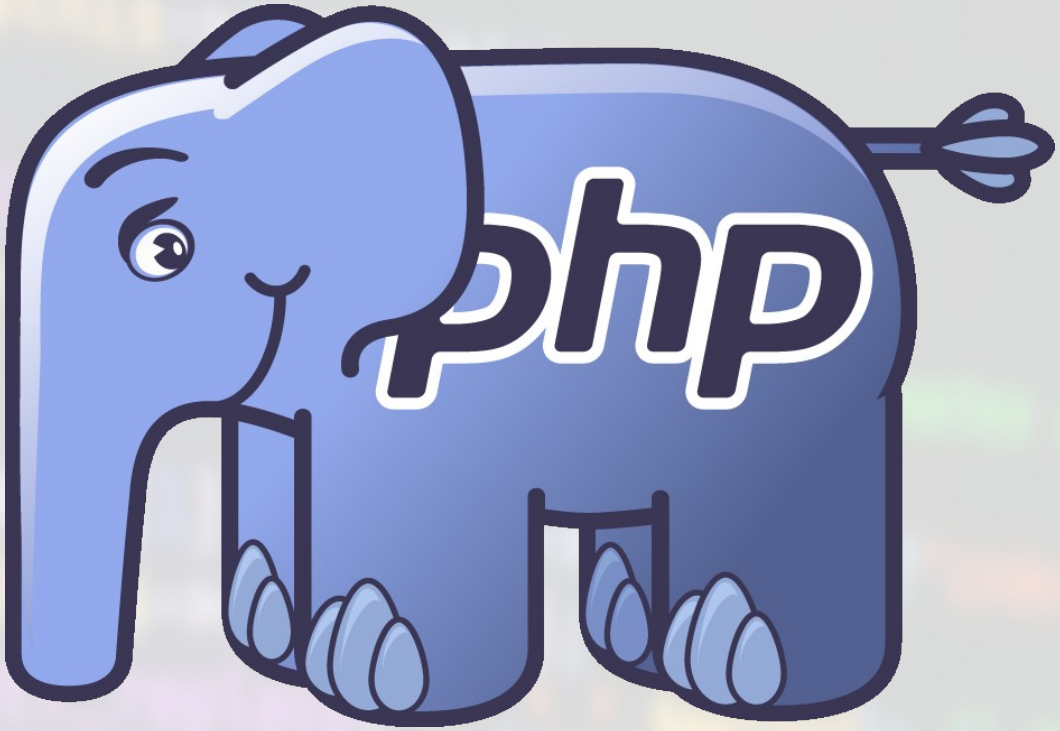
```
function origine($p) {
  $p->set(0, 0) ; }

$point = new Point(10, 10) ;

echo "avant: {$point->toString()}\n" ;
origine($point) ;
echo "apres: {$point->toString()}\n" ;
```

```
avant: (10, 10)
apres: (0, 0)
```

Passage de l'objet **Point**
par référence

Objets dans les chaînes de caractères

- Problème :
  - ambiguïté
  - non évaluable
- Chaîne contenant :
  - un attribut d'un objet dans une chaîne
    ```
    "a : $a->attribut"
    ```
  - le résultat d'une méthode d'un objet dans une chaîne
    ```
    "résultat : $a->calcule()"
    ```
  - une entrée de tableau associatif
  - ```
    "valeur : $tab['cle']"
    ```
  - une variable suivie de texte (sans espace)
    ```
    "\$a contient $aeuros"
    ```
    ```

Objets dans les chaînes de caractères

- Solution :
  - effectuer des concaténations (pénible)
  - délimiter par `{ }`
- Chaîne contenant :
  - un attribut d'un objet dans une chaîne
    ```
    "a : {$a->attribut}"
    ```
  - le résultat d'une méthode d'un objet dans une chaîne
    ```
    "résultat : {$a->calcule()}"
    ```
  - une entrée de tableau associatif
  - ```
    "valeur : {$tab['cle']}"
    ```
  - une variable suivie de texte (sans espace)
    ```
    "\$a contient {$a}euros"
    ```
    ```

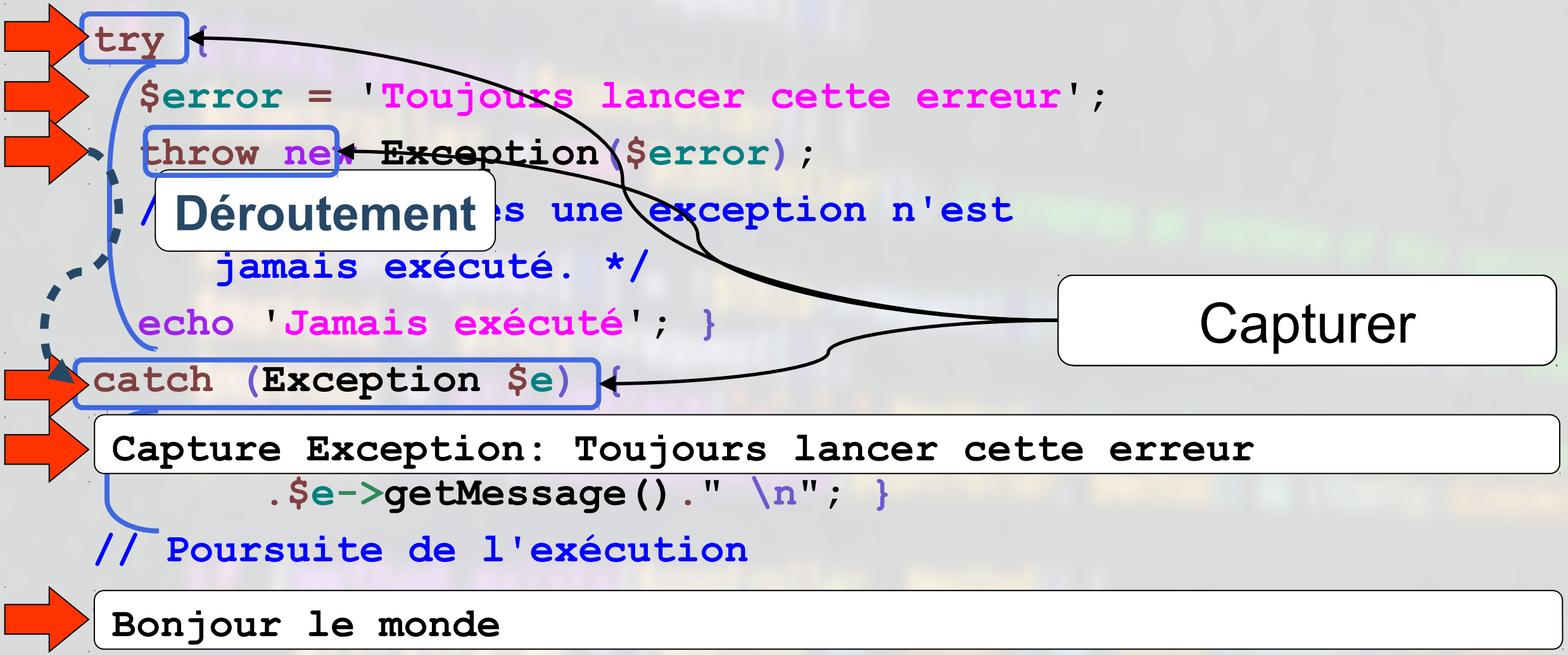


EXCEPTIONS

Gestion des erreurs : exceptions

- Gestion des exception identiques à C++/Java
- Exception peut être :
  - Jetée : `throw`
  - Essayée : `try`
  - Capturée : `catch`
- Exception jetée : code après `throw` non exécuté
- Capture : 1 ou plusieurs blocs (selon type)
- Exception non capturée : erreur fatale

Utilisation des exceptions



Classe **Exception** PHP 5

```php
<?php
class Exception {
  protected $message = ''; // message de l'exception
  protected $code = 0;     // code de l'exception
  protected $file;         // fichier source de l'exception
  protected $line;         // ligne de la source de l'exception

      function __construct(string $message=NULL, int code=0);

final function getMessage();        // message de l'exception
final function getCode();           // code de l'exception
final function getFile();           // nom du fichier source
final function getLine();           // ligne du fichier source
final function getTrace();          // un tableau de backtrace()
final function getTraceAsString(); // chaîne de trace
      function __toString();        // chaîne pour l'affichage
}
```