**Programmation Orientée Objet**

**Classe :**

|  |
| --- |
| class *nomclasse* {  […] *// Définition des attributs et méthodes*  }  $*nomobjet* = new *nomclasse*(…); *// Déclaration d'un objet* |

**Attributs et méthodes :**

Déclaration :

|  |
| --- |
| *// Publique : Accessible en dehors et dans la classe*  public $*nomattribut*  public $*nommethode*([…]) { […] }  $*nommethode*([…]) { […] } *(méthode publique par défaut)*  *// Privé : Accessible uniquement dans la classe*  private $*nomattribut*  private $*nommethode*([…]) { […] }  *// Protégé : Accessible dans la classe et dans les classes enfants*  protected $*nomattribut*  protected $*nommethode*([…]) { […] } |

*Attention : A déclarer dans la classe*

Utilisation :

|  |
| --- |
| $this->*var* *// Utilisation au sein de la classe*  $*nomobjet*->*methode*([…]) *// Utilisation en dehors de la classe (si publique)*  $*nomobjet*->*var* |

**Fonctions utiles :**

|  |  |
| --- | --- |
| **var\_dump(*nomobjet* instanceof *nomclasse*)** | Renvoie true/false en fonction de si l'objet appartient à la classe ou non |

**Valeur statique :**

Valeur commune à tous les objets d'une même classe

|  |
| --- |
| static $*var* = *valeur*; *// Initialiser l'attribut*  static function *nomfct*([…]) { […] } *// Initialiser la méthode*  *nomclasse*::$*var* *// Accéder à la valeur hors de la classe*  *nomclasse*::*nomfct*([…]) *// Accéder à la valeur hors de la classe*  *nomclasse*->*nomfct*([…]) *// Accéder à la valeur hors de la classe* |

**Valeur constante :**

Valeur non modifiable

|  |
| --- |
| const *VAR* = *valeur*; *// Initialiser la valeur*  self::*VAR* *// Accéder à la valeur depuis la classe*  *nomclasse*::*VAR* *// Accéder à la valeur hors de la classe* |

**Getters/Setters :**

Permettent d'accéder à des attributs privés.

|  |
| --- |
| function get() {  return $this->*attribut*;  } |

|  |
| --- |
| function set($*var*) {  $this->*attribut* = $*var*;  } |

**Constructeur/Desctructeur :**

Permettent d'initialiser un objet avec des valeurs particulières ou de le détruire

|  |
| --- |
| function \_\_construct($*var*) {  $this->*attribut* = $*var*;  } |

*Attention : La surcharge n'est pas possible, on ne peut utiliser que des arguments par*

*défaut pour compenser le problème*

|  |
| --- |
| function\_\_destruct() {} |

*Attention : Le destructeur est appelé automatiquement à la fin de la page*

**Héritage :**

Faire hériter la classe Fille de la classe Mere :

|  |
| --- |
| class Fille extends Mere { […] } |

Empêcher l'héritage/redéfinition :

|  |
| --- |
| final class Mere { […] } |

**Polymorphisme :**

On peut surcharger les fonctions.

**Classe abstraite :**

* Classe non complète servant à d'autres classes
* Contient au moins une méthode abstraite
* Une classe abstraite n'est pas instanciable
* Une classe héritant d'une classe abstraite doit implémenter les méthodes abstraites de la classe mère

|  |
| --- |
| abstract class Mere {  abstract function nomfct();  }  class Fille extends Mere {  function nomfct() { … }  } |

**Interfaces :**