POO à la sauce PhP

Pour…	On fait…

Déclarer une classe	`class Toto`
	`{`
	`}`
	
	Convention : un nom de classe est en PascalCase

Déclarer des attributs	`class Toto`
	`{`
	`  public $x;`
	`}`
	
	Il est obligatoire de spécifier la visibilité
	Il n'est pas obligatoire de déclarer les attributs ! On peut faire :
	
	`$t = new Toto();`
	`$t->x = 12;`
	`$t->y = 5;`
	
	Convention : un nom d'attribut est en camelCase

Déclarer des méthodes	`class Toto`
	`{`
	`  public $x;`
	
	`  public function hello()`
	`  {}`
	`}`
	
	Il n'est pas obligatoire de spécifier la visibilité.
	Par défaut, les méthodes sont publiques
	Convention: un nom de méthode est en camelCase

Déclarer des constantes	`const CONSTANT = 'valeur constante';` Convention: les constantes sont en majuscules
Instancier un objet	`$pizza = new Pizza() ;`
Définir des valeurs par défaut pour les paramètres	`public function hello ($msg = 'Hello')`
Accéder un attribut ou une méthode d'un objet	`$t->x = 12;` Attention, erreur classique : `$t->$x = 12;`

Définir la visibilité	`public`
	`protected`
	`private`

Faire hériter d'une classe	`class Toto extends Titi`
	`{`
	`}`

Pour…	On fait…

Accéder un attribut ou une méthode de la classe parente	`class Toto`
	`{`
	`  public $x;`
	
	`  public function __construct()`
	`  {`
	`  }`
	`}`
	
	`class Titi extends Toto`
	`{`
	`  public function __construct()`
	`  {`
	`  parent::__construct();`
	`  }`
	`}`

Accéder à un attribut de la classe depuis une méthode de la classe	`$this->x = 12;`

Définir le constructeur d'une classe	`class Toto`
	`{`
	`  private $x;`
	
	`  public function __construct($x)`
	`  {`
	`  $this->x = $x;`
	`  }`
	`}`
	
	Attention: on ne peut pas définir plusieurs constructeurs

Définir la représentation en string d'un objet	`public function __toString()`
	`{`
	`  return "Attribute X = {$this->x}";`
	`}`

Indiquer le type d'un paramètre de fonction	`public function hello(Message $msg)`
	`{}`
	
	Il s'agit de 'Type Hinting' (suggestion), pas d'une contrainte forte. On ne peut l'utiliser qu'avec des classes (pas de int, string, …)

Afficher un objet dans la console	`class Toto`
	`{`
	`  public function __toString()`
	`  {`
	`  return "{$this->x}";`
	`  }`
	`}`
	`$t = new Toto();`
	`error_log($t);`

Déclarer des membres statiques d'une classe	`static $val;`
	
	`static function connect()`
	
	Attention à ne pas confondre les membres 'normaux' et les membres statiques! Erreur classique :
	
	`$t1 = new Toto();`
	`Toto::$val = 10;`
	`error_log("val={$t1->val}"); // logs 'val='`

Accéder à un membre statique	`Toto::$val = 10;` `Toto::connect();`

Pour…	On fait…

Déclarer une classe abstraite	`abstract class Toto {}`
	Une tentative d'instanciation causera une erreur fatale:
	`PHP Fatal error:  Uncaught Error: Cannot instantiate abstract class Toto`

Déclarer une méthode abstraite (aka 'virtuelle pure')	`abstract function doSomething();`
Déclarer une classe ou une méthode comme finale	`final class Toto {}` `public final function getName() {}`
Redéfinir une méthode (override)	`class Toto { public function getName() { return "toto"; } }` `class Titi extends Toto { public function getName() { return "titi"; } }`
Définir une interface	`Interface Iterator { public function rewind(); public function key(); public function current(); public function next(); public function valid(); }`
Utiliser une interface	`class Toto implements Iterator, Serializable { }`