PHP et la Programmation Orientée Objet

Avec Php, les fonctions issues des bibliothèques sont accessibles « globalement » (sans préfixe pointé du nom du module). Il est impossible alors d'identifier le module auquel appartient une fonction. Pour s'y retrouver, les développeurs de PHP respectent certaines conventions d'écriture. Par exemple les fonctions MySql commencent toutes par le mot mysql_
Exemple : mysql_query()

Mais ce n'est pas suffisant, car les fonctions sont toutes liées à des structures de données bien précises.

Le modèle Objet est ce que l'on a trouvé de mieux pour réunir entre eux Structures de Données et Traitements. Une classe représente le comportement type d'objets de même nature. Par exemple la classe Etudiant décrit la structure type des objets de type Etudiant et leurs comportements (les fonction associées à ce type).

Php 5, bien que n'étant pas à proprement parlé un langage purement objet, intègre le concept de Classe, dispose d'un opérateur (**new**) de construction d'objets (instances) et bien autres choses.

Remarque : Dans les langages objet, une classe peut aussi représenter un module fonctionnel, c'est à dire une structure regroupant des fonctions utilitaires. C'est le cas par exemple de la classe Collections en java. Exemple : Collections.sort(...) permet de trier une collection donnée en argument (en php, on écrirait Classe::méthode()).

Pour les convention d'écriture, voir celles de Symfony :
https://symfony.com/doc/current/contributing/code/standards.html

Extrait :
nommage des méthodes :

```
Method names follow the 'studlyCaps' (also referred to as 'bumpy case'
    or 'camel caps') naming convention, with care taken to minimize the
    letter count. The initial letter of the name is lowercase, and each
    letter that starts a new 'word' is capitalized.
```

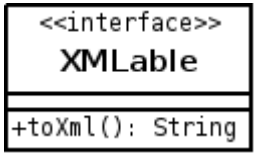
=> appelé aussi *lowerCamelCaps*

Quelques références : http://fr2.php.net/manual/fr/language.oop5.php
Exception : http://fr2.php.net/manual/fr/language.exceptions.php
Static : http://fr2.php.net/manual/fr/language.oop5.static.php
Variable non objet et référence : http://fr2.php.net/manual/fr/language.references.php

Plus loin avec PHP les bonnes manières (en français) : http://eilgin.github.io/php-the-right-way/

Ci-dessous un tableau comparatif de certains concepts clés de l'objet présents dans **PHP** et leur équivalent en **Java**. Un tableau à avoir sous la main quand on commence à coder en PHP Objet.

Concept Objet	Exemple	Php5	Java

Interface		`interface XMLable {`
		`  public abstract function toXML( );`
		
		`}`
			`public interface XMLable {`
			`  public abstract String toXml();`
			`}`

Classe		`class Personne {`
		`  private $nom;`
		`  private $prenom;`
		`  private $login;`
		`}`
			`public class Personne {`
			`  private String nom;`
			`  private String prenom;`
			`  private String login;`
			`}`

Constructeur		`function __construct($nom,`
		`  $prenom, $login) {`
		`  $this->nom = $nom;`
		`  $this->prenom = $prenom;`
		`  $this->login = $login;`
		`}`
			`public Personne(String nom,`
			`  String prenom, String login) {`
			`  this.nom = nom;`
			`  this.prenom = prenom;`
			`  this.login = login;`
			`  }`

Classe d'implémentation	Personne réalise le contrat défini par XMLable	`class Personne implements XMLable {`
		`  ...`
		`  public function toXML() {`
		`  ...`
		`  }`
		`}`
			`public class Personne`
			`  implements XMLable{`
			`  ...`
			`  public String toXml(){`
			`  ...`
			`  }`
			`}`

Héritage	PersonneSecure est une Personne disposant d'un mot de passe	`class PersonneSecure extends`
		`Personne {`
		`  private $md5pw;`
		`  ...`
		`}`
			`public class PersonneSecure`
			`  extends Personne {`
			`  private String md5pw;`
			`  ...`
			`}`

Concept Objet	Exemple	Php5	Java

Redéfinition du constructeur dans une classe enfant	Le constructeur de PersonneSecure personnalise la construction d'une Personne	`class PersonneSecure extends Personne {`
		`  private $md5pw;`
		`  function __construct($nom,`
		`  $prenom, $login, $md5pw) {`
		
		`  parent::__construct($nom,`
		`  $prenom,`
		`  $login);`
		`  $this->md5pw=$md5pw;`
		`  }`
		`}`
			`public class PersonneSecure`
			`  extends Personne {`
			`  private String md5pw;`
			
			`  public PersonneSecure(String nom,`
			`  String prenom, String login,`
			`  String md5pw) {`
			`  super(nom, prenom, login);`
			`  this.md5pw=md5pw;`
			`  }`
			`}`

Redéfinition d'une méthode d'instance d'une classe enfant	La méthode toString est redéfinie dans PersonneSecure (suppose que la classe Personne ait correctement redéfini la méthode toString)	`public function __toString() {`
		`  return parent::__toString() .`
		`  " $this->md5pw";`
		`}`
			`public String toString(){`
			`  return super.toString() +`
			`  " " + this.md5pw;`
			`}`

Création d'objets (instance)	Création d'une instance de Personne et affichage de son état.	`$p = new Personne('Arsac', 'Jacques',`
		`'jarsac');`
		
		`echo $p;`
		`//ou`
		`echo $p->__toString();`
			Personne p =
			`  new Personne("Arsac", "Jacques", "jarsac");`
			System.*out*.println(p.toString());
			// ou
			System.*out*.println(p);

Accès aux attributs public sans les connaître		`// $p est une var. objet`
		`foreach ($p as $att => $value) {`
		`  print "$att => $value\n";`
		`}`
		`// voir aussi : get_class_methods`
		`//  get_class_vars`
			utiliser l'introspection en utilisant l'API de reflection.

Concept Objet	Exemple	Php5	Java

Exception	Gérer l'exception potentielle	`try {`
		`  ...`
		`}catch (Exception $e) {`
		`  ...`
		`}`
			`try {`
			`  ...`
			`} catch (Exception e) {`
			`  ...`
			`}`

	Déclencher une exception	`throw new` MyException('PB');	**throw new** MyException("PB");

Structures «static»	Exemple d'école : un compteur d'instances	`class Personne {`
		` public static $nbInstances = 0;`
		`...`
		`}`
			`class Personne {`
			` public static int nbIntances=0;`
			`...`
			`}`

Définition d'une méthode de classe	Obtenir la valeur du compteur	`class Personne {`
		` private static $nbInstances = 0;`
		` public static function getNbInstances()`
		` { return self::$nbInstances; } ...}`
			`public static int getNbInstances()`
			`{ return Personne.nbInstances; }`

	Accès à un membre static	`Personne::$nbInstances++; si public`
		`ou self::$nbInstances++; en interne`
		`Personne::getNbInstances()`
			`Personne.nbIntances++;`

La logique objet est implicitement par **référence**. La logique implicite des variables non objet est par **valeur**.	Un même objet peut être référencé par plusieurs variables. Exemple : $p1 et $p2 référencent le même objet.	`$p1 = new Personne(` `  'Arsac', 'Jacques','jarsac');` `$p2 = $p1;` `echo $p1 . " ";` `echo $p2  . " ";` `$p1->setNom('ARSAC');` `echo $p2 . " ";` `ce qui donne :` Nom : Arsac, Prénom : Jacques Nom : Arsac, Prénom : Jacques Nom : **ARSAC,** Prénom : Jacques	idem
Constante	par convention les identificateurs sont en MAJUSCULES	`const MATCH = 1;`	`final int MATCH = 1;`