$php->simplonBoulogne();

POO

Sommaire

- Rappel sur les objets
- Rappels sur les constructeurs
- L’encapsulation
- Notions des Getters/Setters (Accesseurs)
- Notions de classes et méthodes abstraites
- Static functions

Un objet ?

- Les objets sont des entités réelles
- Elles peuvent avoir des propriétés et des méthodes
- On instancie une classe pour créer un objet

Rappel constructeurs

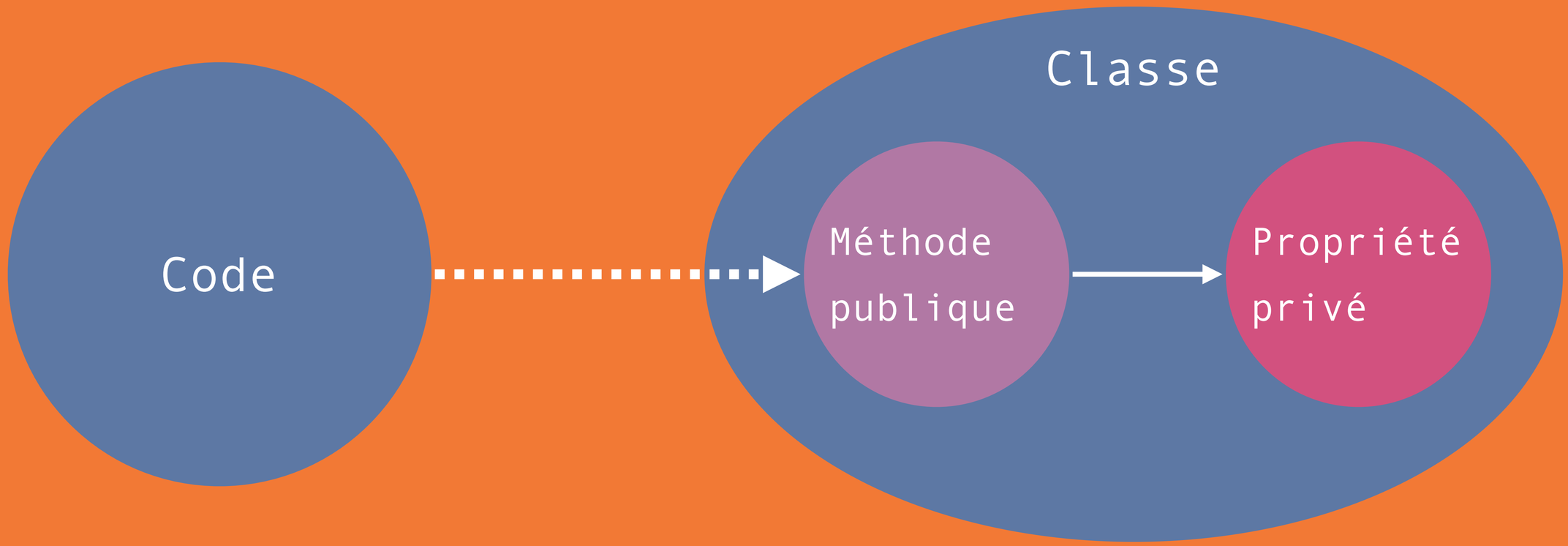
C’est une méthode spéciale :

Elle est appelé **dès** l’instanciation (lors de la création d’un objet, lorsqu’on instancie une classe)

Deux manières de définir le constructeur

___construct() ou « Le nom de la classe » (depuis PHP 5)

Encapsulation



Encapsulation

- Utilisation de méthodes publiques de la classe pour modifier les propriétés
- Protection des données
- Validation des valeurs

Encapsulation

```php
class MyClass {
    private $myVar = 0;
    public function affVar() {
        echo ‘La valeur est: ' . ++$this->myVar;
    }
}

$myInstance = new MyClass();
echo $myInstance->affVar();  // La valeur est: 1
echo $myInstance->myVar;    // Fatal Error
```

Encapsulation

Les accesseurs

- Méthodes spéciales pour accéder / modifier les propriétés d’une classe
- Norme POO

Getter/Setter

```
class Vehicle {
    private $_color, $_brand;

    public function getColor() { return $this->_color; }
    public function setColor($color) {
     $this->_color = $color;
}

    public function getBrand() { return $this->_brand; }
    public function setBrand($brand) {
     $this->_brand = $brand;
}
}
```

Encapsulation

Astuce avec Netbeans

Clic droit - Insert code - Getter/Setter

That’s all !

Héritage

- Les propriétés public et protected sont accessibles à l'intérieur des classes héritées
- Évite la répétition de code
- Permet de partager des méthodes et propriétés en commun, structurer de façon naturelle notre logique objet

Héritage

```
class Electrique extends Voiture {
    public $volt;

    function Electrique($marque, $puissance, $volt){
      parent::__construct($marque, $puissance);
      $this->volt = $volt;
    }

    function getVolt(){ … }
}
```

Classes abstraites

Les classes abstraites sont faites pour être héritées par défaut, elles ne peuvent être instanciées.

Le seul moyen de se servir des méthodes de la classe est de l'héritée.

```
abstract class Vehicle { }
```

Méthodes abstraites

La définition de méthodes abstraites au sein d'une classe abstraite oblige toutes les classes héritées à définir cette fonction.

```php
abstract class Personnage
{
  // On va forcer toute classe héritée à écrire cette
     méthode car chaque personnage frappe
     différemment.
  abstract public function frapper(Personnage $perso);
```

Static

Parfois, il peut être utile d’accéder à une classe sans devoir l’instancier.

Utilisation du mot-clé « static » pour une méthode ou une propriété

Quelques exemples concrets :

- Affichage d’une date
- Connexion à une base de donnée

Static

```php
class Connexion {

    public static function bdd(){
       // Code PDO pour se connecter
    }
}

Connexion::bdd(); //Effectue une connexion à la base
```

Quelques fonctions utiles

class_exists (string class_name)	bool	Si une classe est déclarée
get_class (object object)	object	Retourne la classe spécifiée
get_declared_classes (void)	array	Retourne les classes définies

Challenge 

Créer une base de donnée en utilisant une classe Connexion et une méthode statique pour se connecter (utiliser PDO).

Créer une classe Post avec les propriétés suivantes :

- id_post, title_post, content_post

et les méthodes suivantes :

- addPost, removePost, findAllPost

Créer les pages index.php qui affiche les Posts contenus dans la base de donnée, ajouter.php qui affiche un formulaire d’ajout d’un post

Some ressources

- http://bdelespierre.fr/article/la-poo-en-php-en-10-minutes-ou-moins/
- http://www.lephpfacile.com/cours/
- https://openclassrooms.com/courses/programmez-en-oriente-objet-en-php/l-heritage-3