MISE EN PRATIQUE DE LA POO EN PHP 2/6

Auteur : Sébastien Inion

Codes sources support : https://replit.com/@Sebastien11/Point-et-geometrie?v=1

Une classe

- Comme c’est un objet que l’on va créer on ne doit pourvoir interagir qu’à travers son interface (ses méthodes).

- On va donc rentre inaccessible ses propriétés dans la grande majorité des cas. Pour cela on mettra le mot clef : private.

- Ce principe est l’**encapsulation**.

- Pour accéder à un attribut de notre objet on utilise $this suivi du nom de l’attribut. ex : $this->_nom.
  Cela évite aussi de confondre avec une variable.

Exemple de classe

```php
<?php
class Personnel {
    // -- Propriétés --
    private $_nom = "Dupont";
    protected $prenom = "Pierre";
    public $age = 55;

    // Méthodes
    public function AffichePersonne() {
        echo "Personne : ";
        echo $this->_nom."\t".$this->prenom."\t".$this->age;
        echo PHP_EOL;
    }
}

// Programme
// Instanciation
$salarie = new Personnel();
// Utilisation de la méthode de l'objet
$salarie->AffichePersonne();
?>
```

Norme : notation PEAR -> $_nom pour les données private

L'objet salarie est créé avec l'opérateur new
Pour appeler une méthode on utilise le signe ->
On peut donner des valeurs par défaut $_nom = « Dupont »

PHP_EOL renvoie une chaine correspondant au saut de ligne sur la plateforme

Utilisation d’une classe

```php
<?php
class Personnel {
    // -- Propriétés --
    private $_nom = "Dupont";
    protected $prenom = "Pierre";
    public $age = 55;

    // Méthodes
    public function AffichePersonne() {
        echo "Personne : ";
        echo $this->_nom."\t".$this->prenom."\t".$this->age;
        echo PHP_EOL;
    }
}

// Programme
// Instanciation
$salarie = new Personnel();
// Utilisation de la méthode de l'objet
$salarie->AffichePersonne();
?>
```

- Si on utilise notre classe comme cela on ne ferait que des Pierre Dupont de 55 ans !!
- De plus l’attribut **$age** est accessibles à l’extérieur
- On va donc rendre les attributs **private**
- Découvrir l’intérêt d’un **constructeur**

Constructeur de classe

```php
<?php
class Personnel {
    // -- Propriétés --
    private $_nom ;
    private $_prenom ;
    private $_age ;

    // Constructeur & destructeur
    function __construct($n, $p, $a) {
        $this->_nom = $n;
        $this->_prenom = $p;
        $this->_age =  $a;
    }

    function __destruct() {
        unset ($this->_nom);
        unset ($this->_prenom);
        unset ($this->_age);
        echo "Destruction de l'objet";

    }

    // Methodes

    public function AffichePersonne() {
        echo "Personne : ";
        echo $this->_nom."\t".$this->_prenom."\t".$this->_age;
        echo PHP_EOL;
    }
}

// Programme
$salarieB = new Personnel("dupond", "jean", 45);
$salarieB->AffichePersonne();
unset($salarieB);
?>
```

Utilisation d'une classe

- Les méthodes qui permettent d'accéder à la valeur d'un attribut depuis l'extérieur de l'objet sont appelées **accesseurs** ou **getters**.

- Les méthodes qui permettent de modifier les valeurs sont appelées **mutateurs** ou **setters**.

- En effet on ne doit pas pouvoir directement modifier un attribut (encapsulation). On passera donc par un mutateur qui pourra appliquer certaines contraintes.

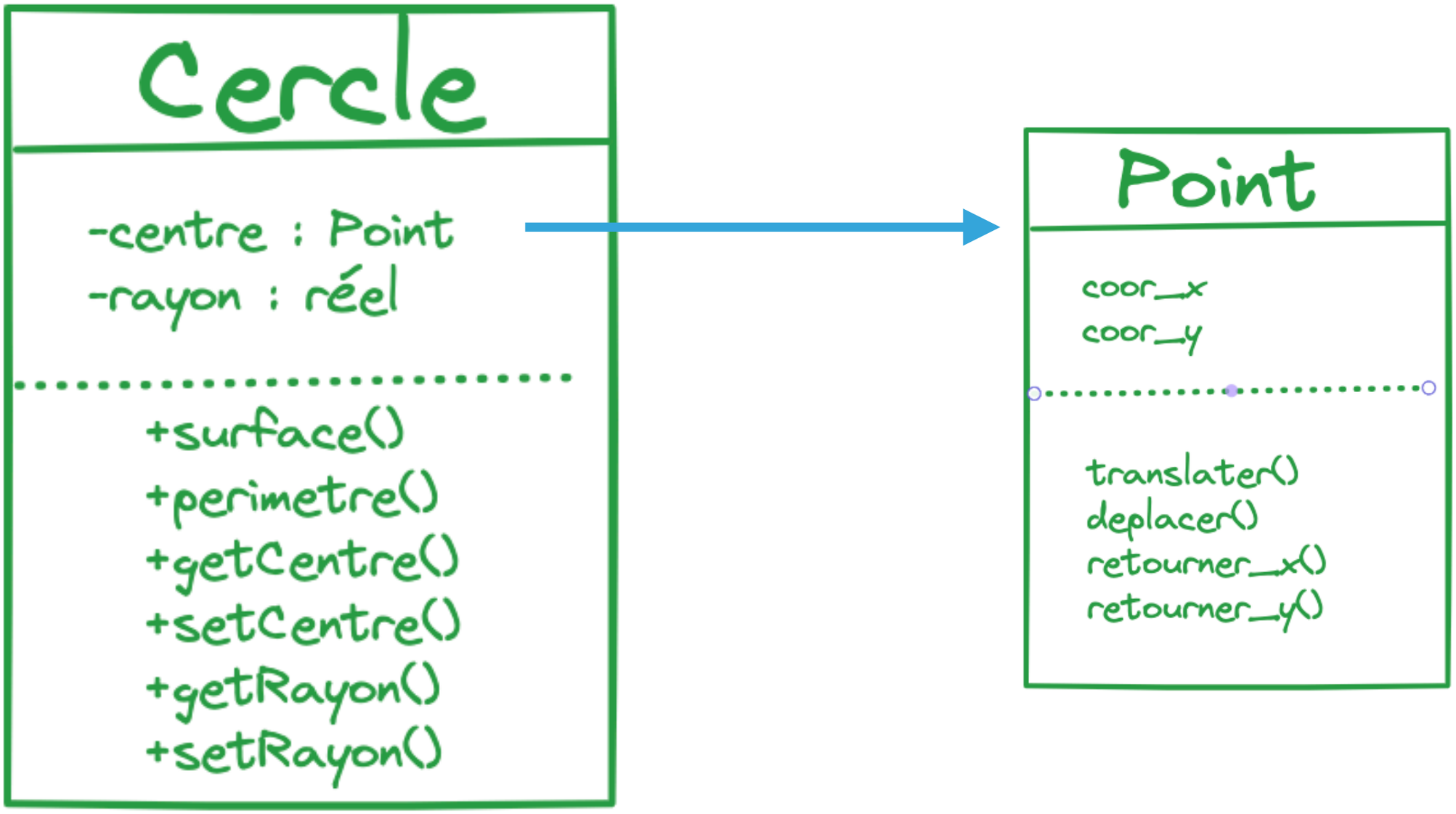
CRÉATION D'UNE CLASSE CERCLE...

Un cercle

- Un cercle peut être définit par son centre et son rayon.
  On a ainsi deux attributs.
  Le centre est un point (donc un objet Point) et le rayon un réel positif.

- On pourra définir en plus du constructeur, des getters et setters une méthode surface() qui nous donnera la surface du cercle.

Création de notre première classe



Le code en PHP

```php
<?php
class Cercle {
  // attributs
  private $centre; // un objet de type Point
  private $rayon; // un réel positif
  // constructeur
  public function __construct($centre,$rayon)
  {
    $this->centre = $centre;
    $this->rayon = $rayon;
  }
  // méthodes getters
  public function getCentre()
  {
    return $this->centre;
  }
  public function getRayon()
  {
    return $this->rayon;
  }
  public function surface(){
    return pi()*$this->rayon*$this->rayon;
  }
  public function perimetre(){
    return 2*pi()*$this->rayon;
  }
  // méthodes setters
  public function setRayon($rayon)
  {
    $this->rayon = $rayon;
  }
  public function setCentre($centre)
  {
    $this->centre = $centre;
  }
  public function __toString(){
    return "Cercle de centre ".$this->centre." et de rayon ".$this->rayon."<br>";
  }
}
```

MERCI POUR VOTRE ATTENTION