Définition :

La programmation orientée objet est la définition de briques logicielle - objets - qui interagissent entre eux. Un object représente un concept, une idée ou toute entité du monde physique, comme une personne, un type de personne, une voiture, un article, etc. Il possède sa propre structure et son propre comportement et sait agir avec ses pairs.

Cette méthode de programmation se différencie de la programmation procédurale. Elle permet une meilleure compréhension du code, un code plus indépendant, une meilleure possibilité d'évolution du code et de maintenance.

Pseudo codes :

Les classes :

Exemple :

En PHP on crée une classe comme ceci:

```php
<?php
    class voiture{
    }
?>
```

et si je veux instancier une voiture:

```php
<?php
$voiture = new voiture();
?>
```

```php
<?php
    class voiture{
        public $nb_roues = 4;
        public $volant = 1;
        public $prix = 5000;
    }
    $voiture = new voiture();
    var_dump( $voiture );
?>
```

Résultat:

```
object(voiture)[1]

public 'nb_roues' => int 4
```

```
public 'volant'   => int 1

public 'prix'     => int 5000
```

Je peux voir la valeur d'un attibut avec la syntaxe suivante:

```php
<?php
    var_dump( $voiture->nb_roues ); // Retournera la valeur 4
?>
<?php

    class voiture{

        public $nb_roues = 4;

        public $volant = 1;

        public $prix = 5000;

    }

    $voiture = new voiture();

    $voiture->prix = 5400;

        print_r( $voiture->prix ); // retourne 5400

?>
```

Il n'est cependant pas conseillé d'utiliser cette syntaxe pour changer la valeur d'un attribut, i
préférable de passer par des methodes qui feront la modification. On appelle ce genre de m
un setter ; et on récupère la valeur avec un getter.

```php
<?php

    class voiture{

        public $nb_roues = 4;

        public $volant = 1;

        public $prix = 5000;

        // Change le prix

        public function setPrix( $prix ){

            $this->$prix = $prix;

        }

        // retourne le prix
```

```php
        public function getPrix( $prix ){
            return $this->$prix;
        }
    }
    $voiture = new voiture();
    // setter
    $voiture->setPrix( 5400 );
    // getter
        print_r( $voiture->getPrix() ); // retourne 5400
?>
```

L'héritage

L'héritage en POO permet d'abstraire certaines fonctionnalités communes à plusieurs class permettant aux classes filles d'avoir leurs propres méthodes.

```php
<?php
    class voiture{
        public $roue = 4;
    }
    class Renault extends voiture{
    }
    class Peugeot extends voiture{
            public $roue = 5;
    }
    $peugeot = new Peugeot();
    $renault = new Renault();
    print_r( $peugeot->roue ); // retourne 4
    print_r( $renault->roue ); // retourne 5
?>
```

Visibilité

La visibilité d'un attribut ou d'une méthode peut être définie en prefixant sa déclaration avec clé: public, protected ou private. Les éléments "public" peuvent être appelés à n'importe quelle p programme. Les "protected" ne peuvent être appelés que par la classe elle même ou les clas parents/enfants. Les "private" sont disponibles que pour la classe en elle même.

Exemple:

```php
<?php

    class voiture{

        public $roue = 4;

        protected $prix = 5000;

        private $nom = "Batmobile";

    }

    $voiture = new voiture();

    print_r( $voiture->roue ); // retourne 4

    print_r( $voiture->prix ); // retourne erreur

    print_r( $voiture->nom ); // retourne erreur

?>
```

Mettre ce genre de protection permet d'indiquer au développeur qu'il doit récupérer les vale attributs en passant par des getter pour des raisons de stratégies.

Exemple:

```php
<?php

    class voiture{

        public $roue = 4;

        protected $prix = 5000;

        private $nom = "Batmobile";

        public function getPrix(){

                return ( $this->prix + 100 );

        }
```

```
        public function getNom(){
                return $this->nom;
        }
    }
    $voiture = new voiture();
    print_r( $voiture->roue ); // retourne 4
    print_r( $voiture->getPrix() ); // retourne 5100
    print_r( $voiture->getNom() ); // retourne Batmobile
?>
```

Pour les méthodes, c'est la même logique:

```
<?php
    class voiture{
        public $roue = 4;
        protected $prix = 5000;
        private $nom = "Batmobile";
        public function getPrix(){
            return ( $this->calcPrix() + 100 );
        }
        // methode de calcule non public
        protected function calcPrix(){
            return ( $this->prix + 10 );
        }
    }
    $voiture = new voiture();
    print_r( $voiture->getPrix() ); // retourne 5110
    print_r( $voiture->calcPrix() ); // retourne erreur
?>
```

Les bases de données :

Connexion à la base :

Exemple

```
<?php
class connect
{

function cn()
{
        $servername = "localhost";
 $username = "root";
 $password = "";
try {
   $conn = new PDO("mysql:host=$servername;dbname=nombd", $username, $password);
   // set the PDO error mode to exception
   $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

```
   //echo "Connecté";
   return $conn;
   }
catch(PDOException $e)
   {
   echo "Connection échoué: " . $e->getMessage();
   }
}
}
?>
```

```
//inclure dans un exemple
<?php
require_once("util/config.php");

?>
```

Les fonctions CRUD :

1. Ajout :

```
function add()
{
$c=new connect();
$conn=$c->cn();
   try {

   $sql = "INSERT INTO `utilisateur`(`ch1`, `ch2`) VALUES (v1, v2)";
   $conn->exec($sql);
   echo "Ajouté";
   }catch(PDOException $e)
   {    echo $sql . "<br>" . $e->getMessage();    }}
```

2. Modification :

```
function update_user()
{
$c=new connect();
$conn=$c->cn();
        try {


   $sql = "UPDATE table SET ch=v1 WHERE id=2";


   // Prepare statement
   $stmt = $conn->prepare($sql);


   // execute the query
   $stmt->execute();


   // echo a message to say the UPDATE succeeded
   echo $stmt->rowCount() . " mis à jours";
   }
catch(PDOException $e)
   {
   echo $sql . "<br>" . $e->getMessage();
   }
}
```

3. Supression :

```
function delete_user()
{
        try {
  $c=new connect();
$conn=$c->cn();
```

```
   // sql to delete a record
   $sql = "DELETE FROM table WHERE id=3";

   // use exec() because no results are returned
   $conn->exec($sql);
   echo "supprimé";
   }
catch(PDOException $e)
   {
   echo $sql . "<br>" . $e->getMessage();
   }
}
```

4. Affichage (mysqli) :

```
function show()
{
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "grh";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT * FROM table";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
```

```
    echo "<table><tr><th>ch1</th><th>ch2</th><th>ch3</th></tr>";
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "<tr><td>".$row["v1"]."</td><td>".$row["v2"]." </td><td>".$row["v3"]."</td></tr>";
    }
    echo "</table>";
} else {
    echo "0 results";
}


}
```