

## Question 1:

Quel est le but principal d'un constructeur dans une classe PHP ?

- ▶ A. Définir les propriétés d'une classe
- ▶ B. Initialiser des objets
- ▶ C. Définir des méthodes de classe
- ▶ D. Inclure des fichiers externes

## Réponse: B. Initialiser des objets



Constructor

```
MyClass *MyObjPtr = new MyClass();
```



Destructor

```
delete MyObjPtr;
```

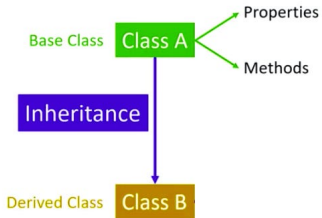
**OOP Destructor**

## Question 2:

Quel mot clé est utilisé pour hériter d'une classe en PHP ?

- ▶ A. implement
- ▶ B. extends
- ▶ C. inherits
- ▶ D. interface

**Réponse:** B. extends



## Question 3:

Quel sera le résultat du code suivant ?

```
class Animal {  
    public function makeSound() {  
        echo "Some generic sound";  
    }  
}
```

```
class Dog extends Animal {  
    public function makeSound() {  
        echo "Bark";  
    }  
}
```

```
$animal = new Dog();  
$animal->makeSound();
```

- ▶ A. Some generic sound
- ▶ B. Bark
- ▶ C. Erreur
- ▶ D. Rien

## Réponse: B. Bark

PowerShell 7 (x64)

```
PS C:\Users\JacobD> php -a  
Interactive shell
```

```
php > function square_val($num, $pow){  
php { return $num**$pow;  
php { }  
php > echo square_val(4, 2);  
16  
php > square_val(4, 2);  
php > echo square_val(4, 3);  
64  
php > echo square_val(4, 0.2);  
1.3195079107729  
php > echo square_val(4, 0.5);  
2  
php > _
```

## Question 4:

Lequel des mots-clés de visibilité suivants limite l'accès des membres de classe à la classe contenant uniquement ?

- ▶ A. public
- ▶ B. private
- ▶ C. protected
- ▶ D. static



Réponse: B. private

```
1 <?php
2 class User {
3     protected $username;
4     protected $password;
5     protected $name;
6     private $role;
7
8     public function __construct($u,$p,$n,$r) {
9         $this->username = $u;
10        $this->password = $p;
11        $this->name = $n;
12        $this->role = $r;
13    }
14    public function getUsername() {
15        return $this->username;
16    }
17    public function getPassword() {
18        return $this->password;
19    }
20    public function getName() {
21        return $this->name;
22    }
23 }
24
```

**PHP**  
**Public**  
**Private**  
**Protecte**

## Question 5:

Que signifie le mot clé « static » lorsqu'il est appliqué à un membre de classe ?

- ▶ A. Le membre n'est accessible qu'au sein de la classe dans laquelle il est défini
- ▶ B. Le membre est partagé entre toutes les instances de la classe
- ▶ C. Le membre ne peut pas être modifié
- ▶ D. Le membre est le même dans toutes les classes

**Réponse:** B. Le membre est partagé entre toutes les instances de la classe

## *PHP The static Keyword*

- ▶ Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.
- ▶ To do this, use the **static** keyword when you first declare the variable.

### **Example**

```
<?php
function myTest() {
    static $x = 0;
    echo $x;
    $x++;
}

myTest();
myTest();
myTest();
?>
```

### **Output**

```
0
1
2
```

## Question 6:

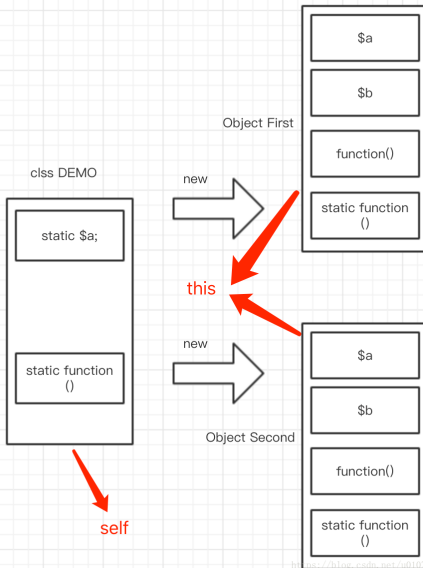
Quel sera le résultat du code suivant ?

```
class Example {  
    public static $count = 0;  
  
    public function __construct() {  
        self::$count++;  
    }  
}
```

```
$obj1 = new Example();  
$obj2 = new Example();  
echo Example::$count;
```

- ▶ A. 0
- ▶ B. 1
- ▶ C. 2
- ▶ D. Erreur

## Réponse: C. 2



## Question 7:

Laquelle des méthodes suivantes est utilisée pour implémenter plusieurs interfaces dans une classe PHP ?

- ▶ A. extends
- ▶ B. implements
- ▶ C. inherit
- ▶ D. use

Réponse: B. implements

## Interface Implementation

Uses the  
**implements**  
keyword

One class may  
implement multiple  
interfaces

```
class StatusUpdate implements Rateable, Searchable {  
    protected $ratings = [];  
  
    public function rate(int $stars, $user) {  
        $this->ratings[$user] = $stars;  
    }  
  
    public function getRating() {  
        $total = array_sum($this->ratings);  
        return $total/count($this->ratings);  
    }  
  
    public function find($query) {  
        /* ..... database query or something ..... */  
    }  
}
```



## Question 8:

Quelle fonction PHP est automatiquement appelée lorsqu'un objet est détruit ?

- ▶ A. `__construct()`
- ▶ B. `__destruct()`
- ▶ C. `destroy()`
- ▶ D. `delete()`



Réponse: B. `__destruct()`

## PHP Destructor

- PHP destructor allows you to clean up resources before PHP releases the object from the memory. For example, you may create a file handle in the constructor and you close it in the destructor.
- To add a destructor to a class, you just simply add a special method called `__destruct()` as follows:

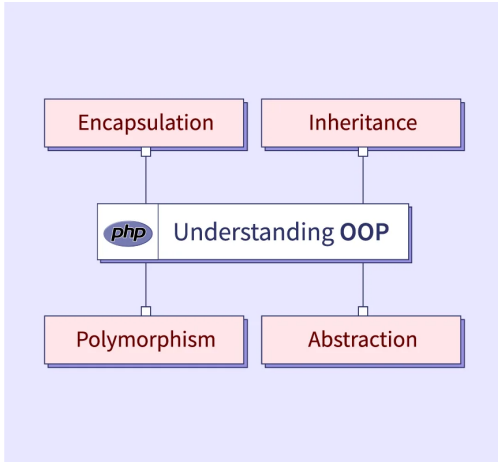
```
1 public function __destruct(){  
2     // clean up resources here  
3 }
```

## Question 9:

Qu'est-ce que le polymorphisme dans le contexte de PHP OOP ?

- ▶ A. La capacité de différentes classes à fournir différentes implémentations de la même méthode
- ▶ B. La capacité d'hériter des propriétés et des méthodes d'une classe parente
- ▶ C. La capacité de définir plusieurs constructeurs dans une classe
- ▶ D. La capacité de cloner un objet

**Réponse:** A. La capacité de différentes classes à fournir différentes implémentations de la même méthode



## Question 10:

Quel est le but du mot-clé « final » dans PHP OOP ?

- ▶ A. Pour empêcher l'instanciation d'une classe
- ▶ B. Pour empêcher l'héritage d'une classe ou la substitution d'une méthode
- ▶ C. Pour rendre une propriété de classe accessible en lecture seule
- ▶ D. Pour rendre une méthode accessible à partir de n'importe quel contexte

**Réponse:** B. Pour empêcher l'héritage d'une classe ou la substitution d'une méthode

Final Methods  Prevent Method Overriding

Final Classes  Prevent Inheritance