PHP – Access Modifiers

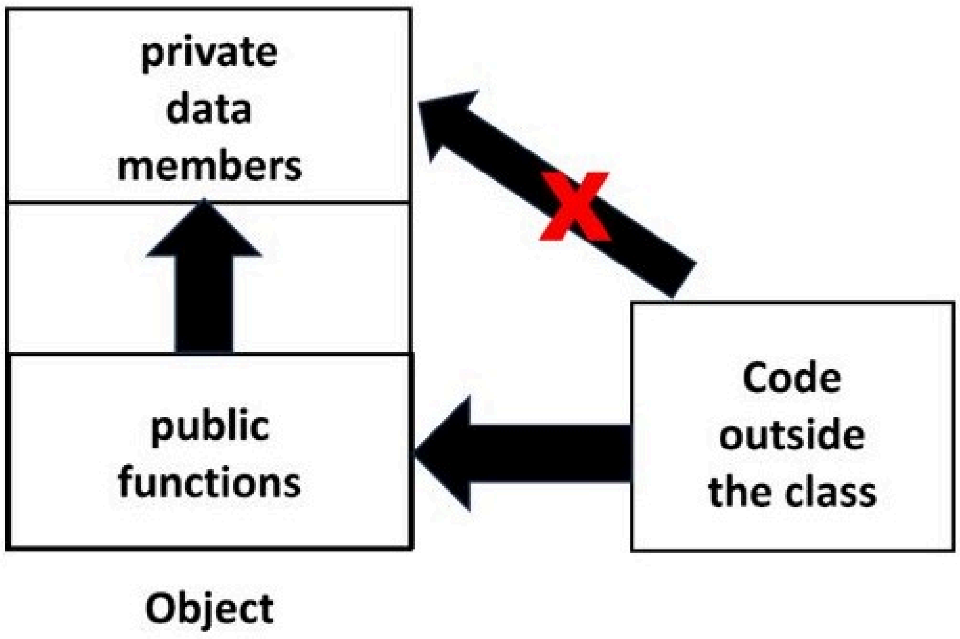
In PHP, the keywords **public, private** and **protected** are known as the **access modifiers**. These keywords control the extent of accessibility or visibility of the class properties and methods. One of these keywords is prefixed while declaring the member variables and defining member functions.

Whether the PHP code has free access to a class member, or it is restricted from getting access, or it has a conditional access, is determined by these keywords −

- **Public** − class members are accessible from anywhere, even from outside the scope of the class, but only with the object reference.
- **Private** − class members can be accessed within the class itself. It prevents members from outside class access even with the reference of the class instance.
- **Protected** − members can be accessed within the class and its child class only, nowhere else.

The principle of data encapsulation is the cornerstone of the object-oriented programming methodology. It refers to the mechanism of keeping the data members or properties of an object away from the reach of the environment outside the class, allowing controlled access only through the methods or functions available in the class.

To implement encapsulation, data members of a class are made **private** and the methods are made **public**.



Public Members

In PHP, the class members (both member variables as well as member functions) are public by default.

Example

In the following program, the member variables title and price of the object are freely accessible outside the class because they are public by default, if not otherwise specified.

Open Compiler

```
<?php
   class Book {
      /* Member variables */
      var $price;
      var $title;

      /*Constructor*/
      function __construct(string $param1="PHP Basics", int $param2=380) {
         $this->title = $param1;
         $this->price = $param2;
      }

      function getPrice() {
         echo "Title: $this->price \n";
      }

      function getTitle() {
         echo "Price: $this->title \n";
      }
   }
   $b1 = new Book();
   echo "Title : $b1->title Price: $b1->price";
?>
```

It will produce the following **output** –

```
Title : PHP Basics Price: 380
```

Private Members

As mentioned above, the principle of encapsulation requires that the member variables should not be accessible directly. Only the methods should have the access to the data members. Hence, we need to make the member variables private and methods public.

```php
<?php
   class Book {
      /* Member variables */
      private $price;
      private $title;

      /*Constructor*/
      function __construct(string $param1="PHP Basics", int $param2=380) {
         $this->title = $param1;
         $this->price = $param2;
      }

      public function getPrice() {
         echo "Price: $this->price \n";
      }

      public function getTitle() {
         echo "Title: $this->title \n;";
      }
   }
   $b1 = new Book();
   $b1->getTitle();
   $b1->getPrice();
   echo "Title : $b1->title Price: $b1->price";
?>
```

Output

Now, the getTitle() and getPrice() functions are public, able to access the private member variables title and price. But, while trying to display the title and price directly, an error is encountered as they are not public.

```
Title: PHP Basics
Price: 380
```

```
Fatal error: Uncaught Error: Cannot access private property
Book::$title in hello.php:31
```

Protected Members

The effect of specifying protected access to a class member is effective in case of class inheritance. We know that public members are accessible from anywhere outside the class, and private members are denied access from anywhere outside the class.

The **protected** keyword grants access to an object of the same class and an object of its inherited class, denying it to any other environment.

Let us set the title member in Book class example to protected, leaving price to private.

```
class Book {
   /* Member variables */
   private $price;
   protected $title;
   # rest of the code kept as it is
}
$b1 = new Book();
$b1->getTitle();
$b1->getPrice();
```

PHP allows the both the member variables to be accessed, as the object belongs to the same class.

Let us add a **mybook** class that inherits the **Book** class −

```
class mybook extends Book {
   # no additional members defined
}
```

whose object is still able to access the member variables, as the child class inherits public and protected members of the parent class.

However, make **mybook** class as an independent class (not extending Book class) and define a getmytitle() function that tries to access protected title member variable of Book class.

```
class mybook {
   public function getmytitle($b) {
      echo "Title: $b->title <br/>";
   }
}
$b1 = new mybook();
$b = new Book();
$b1->getmytitle($b);
```

As the getmytitle() function tries to print title of Book object, an error message showing **Cannot access protected property Book::$title** is raised.

Example

Try to run the following code −

Open Compiler

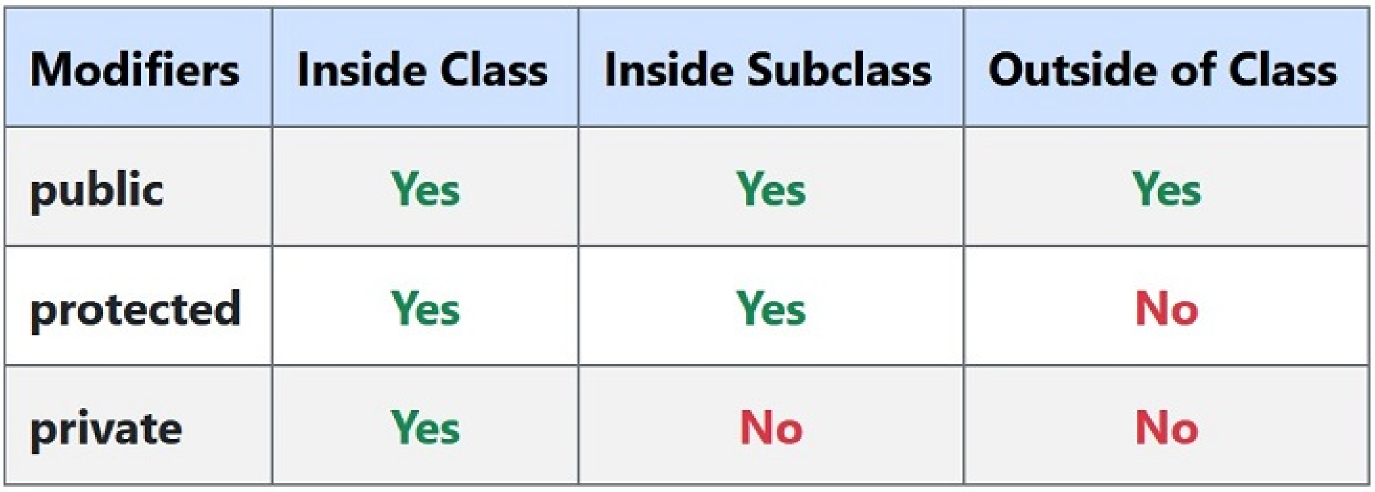
```
<?php
   class Book {
      private $price;
      protected $title;
      function __construct(string $param1="PHP Basics", int $param2=380) {
         $this->title = $param1;
         $this->price = $param2;
      }
      public function getPrice(){
         echo "Price: $this->price <br/>";
      }
      public function getTitle(){
         echo "Title: $this->title <br/>";
      }
   }
   class mybook {
      public function getmytitle($b) {
         echo "Title: $b->title <br/>";
      }
   }
   $b1 = new mybook();
   $b = new Book();
   $b1->getmytitle($b);
?>
```

It will produce the following **output** –

```
PHP Fatal error:  Uncaught Error: Cannot access protected property
   Book::$title in /home/cg/root/97848/main.php:18
```

Hence, it can be seen that the protected member is accessible by object of same class and inherited class only. For all other environment, protected members are not accessible.

The accessibility rules can be summarized by the following table –



Modifiers	Inside Class	Inside Subclass	Outside of Class
public	Yes	Yes	Yes
protected	Yes	Yes	No
private	Yes	No	No