PHP – Constructor and Destructor

As in most of the object-oriented languages, you can define a constructor function in a class in PHP also. When you declare an object with the new operator, its member variables are not assigned any value. The constructor function is used to initialize every new object at the time of declaration. PHP also supports having a destructor function that destroys the object from the memory as it no longer has any reference.

The __construct() Function

PHP provides a __construct() function that initializes an object.

```
__construct(mixed ...$values = ""): void
```

The constructor method inside a class is called automatically on each newly created object. Note that defining a constructor is not mandatory. However, if present, it is suitable for any initialization that the object may need before it is used.

You can pass as many as arguments you like into the constructor function. The __construct() function doesn't have any return value.

Let us define a constructor in the Book class used in the previous chapter

Open Compiler

```
<?php
   class Book {

      /* Member variables */
      var $price;
      var $title;

      /*Constructor*/
      function __construct(){
         $this->title = "PHP Fundamentals";
         $this->price = 275;
      }

      /* Member functions */
      function getPrice() {
         echo "Price: $this->price \n";
      }

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

   $b1 = new Book;
   $b1->getTitle();
   $b1->getPrice();
?>
```

It will produce the following **output** −

```
Title: PHP Fundamentals
Price: 275
```

Parameterized Constructor

The member variables of $b1 have been initialized without having to call setTitle() and setPrice() methods, because the constructor was called as soon as the object was declared. However, this constructor will be called for each object, and hence each object has the same values of title and price properties.

To initialize each object with different values, define the **__construct()** function with parameters.

Change the definition of **__construct()** function to the following −

```
function __construct($param1, $param2) {
   $this->title = $param1;
   $this->price = $param2;
}
```

To initialize the object, pass values to the parameters inside a parenthesis in the declaration.

```
$b1 = new Book("PHP Fundamentals", 375);
```

Example

Now, you can have each object with different values to the member variables.

Open Compiler

```php
<?php
   class Book {

      /* Member variables */
      var $price;
      var $title;

      /*Constructor*/
      function __construct($param1, $param2) {
         $this->title = $param1;
         $this->price = $param2;
      }

      /* Member functions */
      function getPrice(){
         echo "Price: $this->price \n";
      }

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

   $b1 = new Book("PHP Fundamentals", 375);
   $b2 = new Book("PHP Programming", 450);

   $b1->getTitle();
   $b1->getPrice();
   $b2->getTitle();
   $b2->getPrice();
?>
```

It will produce the following **output** –

```
Title: PHP Fundamentals
Price: 375
```

```
Title: PHP Programming
Price: 450
```

Constructor Overloading

Method overloading is an important concept in object-oriented programming, where a class may have more than one definitions of constructor, each having different number of arguments. However, PHP doesn't support method overloading. This limitation may be overcome by using arguments with default values in the constructor function.

Change the __construct() function to the following −

```
function __construct($param1="PHP Basics", $param2=380) {
   $this->title = $param1;
   $this->price = $param2;
}
```

Now, declare an object without passing parameters, and the other with parameters. One without parameters will be initialized with default arguments, the other with the values passed.

```
$b1 = new Book();
$b2 = new Book("PHP Programming", 450);
```

It will produce the following **output** −

```
Title: PHP Basics
Price: 380
Title: PHP Programming
Price: 450
```

Type Declaration in Constructor

Since PHP (version 7.0 onwards) allows scalar type declarations for function arguments, the __construct() function may be defined as −

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

In the earlier versions of PHP, using the name of class to define a constructor function was allowed, but this feature has been deprecated since PHP version 8.0.

The __destruct() Function

PHP also has a __destructor() function. It implements a destructor concept similar to that of other object-oriented languages, as in C++. The destructor method will be called as soon as there are no other references to a particular object.

```
__destruct(): void
```

The __destruct() function doesn't have any parameters, neither does it have any return value. The fact that the __destruct() function is automatically called when any object goes out of scope, can be verified by putting var_dump($this) inside the function.

As mentioned above, **$this** carries the reference to the calling object, the dump shows that the member variables are set to NULL

Add destructor function in the Book class as follows −

```
function __destruct() {
   var_dump($this);
   echo "object destroyed";
}
```

As the program exits, the following **output** will be displayed −

```
object(Book)#1 (2) {
  ["price"]=>
  NULL
  ["title"]=>
  NULL
}
object destroyed
```