Classes and Functions in PHP

CS/IT 490 WD, Fall 2013

Last update 2013-09-24

Written by Rachel J. Morris

Functions

- We can create our own functions in PHP.
- These can be inside of a class, or outside of one.
- No return-type specified, just start with ***function***.

Functions

Function Definition:

```
11  <?
12  function SaveFormInfo( $formArray )
13  {
14      $filepath = "output.txt";
15      $fileHandler = fopen( $filepath, "w" );
16      if ( $fileHandler == false )
17      {
18          print_r( error_get_last() );
19          exit();
20      }
21
22      foreach ( $_POST as $key=>$value )
23      {
24          fwrite( $fileHandler, $key . ": " . $value . "\n\n" );
25      }
26
27      fclose( $fileHandler );
28  }
29  ?>
```

Function Call:

```
7   <?
8   SaveFormInfo( $_POST );
9   ?>
```


Written by Rachel J. Morris

Functions

- Functions can have parameters – remember to begin variable names with $
- Arguments are passed **by value** automatically, but you can pass them **by reference**.
- Functions can be overloaded

Written by Rachel J. Morris

Functions

- Passing an argument by-reference:

```php
<?
function DoubleValues( &$var1, &$var2 )
{
    $var1 *= 2;
    $var2 *= 2;
}
?>
```

These parameters have an ampersand before the $variablename

```php
<?
$number1 = 3;
$number2 = 5;

DoubleValues( $number1, $number2 );
?>
```

No special characters on the arguments themselves.

Written by Rachel J. Morris

Classes

- Classes can contain functions and variables
- Classes can have constructors & destructors
- We can create static members / functions
- Classes can inherit from other classes
- We can have abstract classes (think of virtual and pure virtual functions in C++)

Written by Rachel J. Morris

Classes - Members

- You can create member functions and member variables, along with accessibility in PHP.

```
class SimpleClass
{
    private $m_number;

    public SetNumber( $n )
    {
        $this->m_number = $n;
    }
}
```

Notice that accessing member variables requires

$this->*varname*

Classes – Constructors & Destructors

- The old way of creating constructors was to make a function with the class name.
- However, there is a new way to handle constructors.

Classes – Constructors & Destructors

```php
class SimpleClass
{
    private $m_number;

    function __construct()
    {
        $this->m_number = -1;
    }

    function __destruct()
    {
        echo( "<p>Clean Up SimpleClass</p>" );
    }
}
```


Written by Rachel J. Morris

Classes – Constructors & Destructors

- For a child class, you can call its parent's constructor with:

```php
class Student extends Person
{
    private $m_gpa;

    function __construct()
    {
        // Call parent's constructor
        parent::__construct();
        $this->m_gpa = 0;
    }
}
```

parent::__construct();

Classes – Constructors & Destructors

- You cannot overload constructors
- Constructors CAN have arguments
- How can you get around inability to overload constructors?

Written by Rachel J. Morris

Classes – Variable Parameters

- You can create a function with a variable number of parameters with an associative array!

```php
class Person
{
    private $m_firstName, $m_lastName;

    // Have any # of parameters passed by passing one array!
    function __construct( $paramArray )
    {
        $this->m_firstName = $this->m_lastName = "unset";

        if ( isset( $paramArray["first"] ) )
        {
            $this->m_firstName = $paramArray["first"];
        }
        if ( isset( $paramArray["last"] ) )
        {
            $this->m_lastName = $paramArray["last"];
        }
    }
```

```php
<?
// Constructors now require an array argument, but it can be empty.
$person1 = new Person( Array() );
$person2 = new Person( Array( "first"=>"Bob", "last"=>"Robertson" ) );
?>
```


Written by Rachel J. Morris

Classes - Instantiation

- Create a new instance of a class like this:

```php
<?
$person1 = new Person();
$person2 = new Student();
?>
```

- To access members of a class, you must use the -> operator

```php
<p>Person 1: <?=$person1->GetName()?></p>
<p>Person 2: <?=$person2->GetName()?>, GPA: <?=$person2->GetGPA()?></p>
```


Written by Rachel J. Morris

Classes – Inheritance & Composition

- A class can inherit from another using the **extends** keyword:

```
class Student extends Person
{
    private $m_gpa;

    function __construct()
    {
        // Call parent's constructor
        parent::__construct();
        $this->m_gpa = 0;
    }

    public function SetGPA( $g )
    {
        $this->m_gpa = $g;
    }

    public function GetGPA()
    {
        return $this->m_gpa;
    }
}
```


Written by Rachel J. Morris

Classes – Inheritance & Composition

- Composition is simply having one class as a member variable of another class

```php
class Person
{
    private $m_name;
    private $m_info;

    function __construct( $paramArray )
    {
        $this->m_name = $paramArray["name"];
        $this->m_info = new StudentInfo();
    }
```

This will be a class instantiation

Class is instantiated here

Classes – Inheritance & Composition

- Composition is simply having one class as a member variable of another class

```php
class Person
{
    private $m_name;
    private $m_info;

    function __construct( $paramArray )
    {
        $this->m_name = $paramArray["name"];
        $this->m_info = new StudentInfo();
    }

    function OutputInfo()
    {
        echo( $this->m_name );
        echo( ", " );
        echo( $this->m_info->GetGPA() );
    }
}
```

We can access members of the composed class with

$this->*varname->membername*

Classes – Static Members

Last update 2013-09-24

Written by Rachel J. Morris

Classes - Abstraction

Last update 2013-09-24

Written by Rachel J. Morris

References

-

Written by Rachel J. Morris