Getting Started with OOP
The Object of my Affections

**Terms**

Class

Instantiation

Encapsulation

Functions – methods

Getter

Setter

Variables – properties

A class is similar to a blueprint. You can make new objects based on the class. This process is called instantiation.

The class has methods. These are functions which the object can perform.

The class will have properties. These are written as variables.

Having all the processes for an object to be used (its own data and methods) is called encapsulation.

Create two files: cat\_lib.php and cat.php .

cat\_lib.php

<?php

 /\*\*\*\*\*

 file: cat\_lib.php

 used with cat.php

 \*\*\*\*\*\*\*\*\*\*\*\*/

 class Cat

 {

 var $name;

 var $breed;

 function setName($nameofcat)

 {

 $this -> name = $nameofcat;

 }

 function getName()

 {

 return $this -> name;

 }

 function setBreed($breedname)

 {

 $this -> breed = $breedname;

 }

 function getBreed()

 {

 return $this -> breed;

 }

 function antics()

 {

 echo "This kitty chases imaginary mice.";

 }

 }

?>

cat.php

<?php

 /\*\*\*\*\*\*\*\*\*\*\*

 cat.php

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*/

 include('cat\_lib.php');

 $kitty = new Cat();

 $kitty -> setName('Mrkosa');

 echo "My kitty is called ".$kitty -> getName()." .";

 $kitty -> setBreed('Persian');

 echo "My kitty is a ".$kitty -> getBreed(). " .";

 $kitty -> antics();

?>

Run the files and examine the code. You should see this on your screen:

**My kitty is called Mrkosa .My kitty is a Persian .This kitty chases imaginary mice.**

**Assignment**: Create two new files. When they are run, you should see this on your screen:

**My dog has the name of Poochie . The breed is Brittany Spaniel .This dog dreams a lot.**

**Part 2.**

In this section you will gather data from variables posted in an HTML form.

<html>

<!--

calc.html

calls calc.php

-->

<head>

<title>Calculations</title>

<link rel="stylesheet" type="text/css" href = "calc.css" />

</head>

<body>

<form action="calc.php" method=post>

<fieldset>

<legend>Calculations </legend>

<p ><label >Integer 1</label></p>

<p > <input type="text" name="numOne" /></p>

<p ><label >Integer 2</label></p>

<p > <input type="text" name="numTwo" /></p>

<p class ="submit" > <input type = "submit" value = "Calculate" /> </p>

</fieldset>

</form>

</body>

</html>

Here is an incomplete calc.php file.

<?php

 /\*\*\*\*\*\*\*\*\*\*\*\*

 file: calc.php

 uses calc.inc

 called from calc.html

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

 include('calc.inc');

 $numOne = $\_POST['numOne'];

 $numTwo = $\_POST['numTwo'];

 $calc = new Calc();

 $calc -> setInput1($numOne);

 $calc -> setInput2($numTwo);

 //add them and display the sum

 echo "The sum of the two integers is ".$calc -> getSum()." .";

 //display the difference

 *missing code*

 //display the product

 *missing code*

 //display the quotient

 *missing code*

?>

Here is an incomplete calc.inc file. Work out the missing lines of code.

<?php

 /\*\*\*\*\*\*\*\*\*\*\*\*\*

 file: calc.inc

 provide class for calc.php

 \*\*\*\*\*\*\*\*\*\*\*/

 class Calc

 {

 var $input;

 var $input2;

 var $output;

 function setInput1($int)

 {

 $this -> input = $int;

 }

 function setInput2($int)

 {

 $this -> input2= $int;

 }

function getSum()

 {

 $this->output = $this->input + $this ->input2;

 return $this->output;

 }

function getProduct()

 {

 *missing lines*

 }

 function getDifference()

 {

 *missing lines*

 }

 function getQuotient()

 {

 *missing lines*

 }

 }

?>