# Lab #6 - PHP, OOP and Input Handling

## Part 0

Create a new folder in websys for Lab 6 named lab6, and confirm that it is being served by Apache. For this lab, you will be using existing PHP code. Save the document as lab6.php in your lab6 folder.

## Part 1: Object Oriented PHP

For this lab, you will be developing a (very) basic calculator using PHP. The markup has been provided for you, along with some sample code to get you started.

The sample code provided to you defines an *abstract class* called Operation. Abstract classes are not instantiated themselves, but are used to provide a basis for other subclassed implementations while still defining some of their default methods. An example implementation can be found in the Addition subclass, which implements the abstract methods defined in Operation::operate() and Operation::getEquation() to perform basic addition on the two operands passed into the constructor.

To complete Part 1, implement subclasses for Subtraction, Multiplication and Division in the same way.

## Part 2: POST Handling

As you will see, the same form may have multiple submit buttons: only the button pressed will submit its value via the request body in the resulting POST request. Therefore, we can test to see which has been sent, and perform different operations based on the result. To see what has been passed in via $_POST for debugging purposes, we can use var_dump($_POST);.

Hint: You will want to check isset($var_name) to determine if a value exists before reading it. PHP will raise a warning if you attempt to read from a variable that has not yet been defined.

To receive credit for Part 2, for each button, instantiate a new object corresponding to the given operation and the two operands provided. (ie "Multiply" should create a Multiply() instance from the two text fields when POSTed back to itself, and so on). Then, use your new object's getEquation() method to print the full equation inside pre#result.

Another Hint: Review the Addition subclass carefully - Using the principles of Object Oriented Programming, particularly Polymorphism, to your advantage makes life easier.

You don't have to write the code for handling each individual operation in the suggested place in the example: you may move it if it makes more sense to you.

## Receiving Credit

Firstly **- REMEMBER** your Readme files.

Answer any questions in the PHP file provided.

1) Explain what each of your classes and methods does, the order in which methods are invoked, and the flow of execution after one of the operation buttons has been clicked.

2) Also explain how the application would differ if you were to use $_GET, and why this may or may not be preferable.

3) Finally, please explain whether or not there might be another (better +/-) way to determine which button has been pressed and take the appropriate action

**Rubrik**

Part 1 - 10 Points

Part 2 - 20 Points

Questions - 10 Points