

## Lab 11

1. Create two PHP pages:  
index.php  
class\_lib.php
2. Create a PHP class in class\_lib.php

```

1 <?php
2 class person {
3
4 }
5 ?>
6

```

3. Add data to your class

```

1 <?php
2 class person {
3 var name;
4 }
5 ?>
6

```

4. Add functions/methods to your class. Save class\_lib.php

```

1 <?php
2 class person {
3 var $name;
4 function set_name($new_name) {
5 $this->name = $new_name;
6 }
7
8 function get_name() {
9 return $this->name;
10 }
11 }
12 ?>
13

```

5. Include your class in your index.php page.

---

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml">
4 <head>
5 <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
6 <title>OOP in PHP</title>
7 <?php include("class_lib.php"); ?>
8 </head>
9 <body>
10 </body>

```

## 6. Instantiate/create your object

```

6 <?php include("class_lib.php"); ?>
7 </head>
8 <body>
9 <?
10 $stefan = new person();
11 ?
12 </body>
13 </html>
14

```

## 7. The 'new' keyword. To create an object out of a class, you need to use the 'new' keyword.

```

6 <?php include("class_lib.php"); ?>
7 </head>
8 <body>
9 <?
10 $stefan = new person();
11 $jimmy = new person;
12 ?
13 </body>
14 </html>
15

```

## 8. Set an objects properties

```

6 <?php include("class_lib.php"); ?>
7 </head>
8 <body>
9 <?php
10 $stefan = new person();
11 $jimmy = new person;
12 //set object properties
13 $stefan->set_name("Stefan Mischook");
14 $jimmy->set_name("Nick Waddles");
15 |
16 ?
17 </body>
18 </html>

```

## 9. Accessing an object's data.

```

6 <?php include("class_lib.php"); ?>
7 </head>
8 <body>
9 <?php
10 $stefan = new person();
11 $jimmy = new person;
12 //set object properties
13 $stefan->set_name("Stefan Mischook");
14 $jimmy->set_name("Nick Waddles");
15 echo "Stefan's full name: " . $stefan->get_name();
16 echo "Nick's full name: " . $jimmy->get_name();
17
18 ?>
```

After finished type above code, save file as index.php. Test index.php on your browser.  
In a short period of time, you've:

- Designed a PHP class.
- Generate/created a couple of objects based on your class.
- Inserted data into your objects.
- Retrieved data from your objects.

## 10. Create constructors in class\_lib.php

```

1 <?php
2 class person {
3 var $name;
4 function __construct($persons_name) {
5 $this->name = $persons_name;
6 }
7
8 function set_name($new_name) {
9 $this->name = $new_name;
10 }
11
12 function get_name() {
13 return $this->name;
14 }
15 }
16 ?>
```

11. Create an object with a constructor in index.php. Overwrite the existing code to practice using the constructor.

```

6 <?php include("class_lib.php"); ?>
7 </head>
8 <body>
9 <?php
10 $stefan = new person("Stefan Mischook");
11 echo "Stefan's full name: " . $stefan->get_name();
12
13 ?>
14 </body>
15 </html>
16

```

12. Restricting access to properties using 'access modifiers'. You restrict access to class properties using something called 'access modifiers'. There are 3 access modifiers:

- a. public
- b. private
- c. protected

a. Public is the default modifier. Overwrite code in your classs\_lib.php to use public modifiers as below.

```

1 <?php
2 class person {
3 var $name;
4 public $height;
5 protected $social_insurance;
6 private $pin_number;
7 function __construct($persons_name) {
8 $this->name = $persons_name;
9 }
10 function set_name($new_name) {
11 $this->name = $new_name;
12 }
13 function get_name() {
14 return $this->name;
15 }
16 }
17 ?>

```

Note: When you declare a property with the 'var' keyword, it is considered 'public'.

13. Restricting access to properties: part 2. When you declare a property as 'private', only the same class can access the property. When a property is declared 'protected', only the same class and classes derived from that class can access the property.

```

2 <html>
3 <?php include("class_lib.php"); ?>
4 </head>
5 <body>
6 <?php
7 $stefan = new person("Stefan Mischook");
8 echo "Stefan's full name: " . $stefan->get_name();
9 /* Since $pinn_number was declared private, this line of code will generate an error. Try it out!
10 */
11 echo "Tell me private stuff: " . $stefan->$pinn_number;
12 ?>
13 </body>
14 </html>

```

14. Restricting access to methods. Modify class\_lib.php as below code.

---

```

1 <?php
2 class person {
3 var $name;
4 public $height;
5 protected $social_insurance;
6 private $pinn_number;
7 function __construct($persons_name) {
8 $this->name = $persons_name;
9 }
10 private function get_pinn_number() {
11 return $this->$pinn_number;
12 }
13 }
14 ?>
15

```

Notes: Since the method get\_pinn\_number() is 'private', the only place you can use this method is in the same class - typically in another method. If you wanted to call/use this method directly in your PHP pages, you would need to declare it 'public'.

15. Reusing code the OOP way: inheritance. Modify your class\_lib.php as below code to create a new employee class.

```

1 <?php
2 class person {
3 var $name;
4 public $height;
5 protected $social_insurance;
6 private $pin_number;
7 function __construct($persons_name) {
8 $this->name = $persons_name;
9 }
10 function set_name($new_name) {
11 $this->name = $new_name;
12 }
13 function get_name() {
14 return $this->name;
15 }
16 }
17
18 class employee extends person {
19 function __construct($employee_name) {
20 $this->set_name($employee_name); |
21 }
22 }
23
24 ?>
25

```

Modify your index.php as below code to use inheritance concept.

```

3 <?php include("class.lib.php"); ?>
4 </head>
5 <body>
6 <title>OOP in PHP</title>
7 </head>
8 <body>
9 <?php
10 // Using our PHP objects in our PHP pages.
11 $stefan = new person("Stefan Mischook");
12 echo "Stefan's full name: " . $stefan->get_name();
13 $james = new employee("Johnny Fingers");
14 echo "---> " . $james->get_name();
15 ?>
16
17 </body>
18 </html>

```