Example 1: A singleton (static members)

=======================================

<?php

class Counter {

var $counter = 0;

function increment\_and\_print()

{

print ++$this->counter;

print "\n";

}

}

class SingletonCounter {

static $m\_instance = NULL;

function Instance()

{

if (self::$m\_instance == NULL) {

self::$m\_instance = new Counter();

}

return self::$m\_instance;

}

}

SingletonCounter::Instance()->increment\_and\_print();

SingletonCounter::Instance()->increment\_and\_print();

SingletonCounter::Instance()->increment\_and\_print();

?>

Example 2: Factory method (derefencing objects returned from functions)

=======================================================================

<?php

class Circle {

function draw()

{

print "Circle\n";

}

}

class Square {

function draw()

{

print "Square\n";

}

}

function ShapeFactoryMethod($shape)

{

switch ($shape) {

case "Circle":

return new Circle();

case "Square":

return new Square();

}

}

ShapeFactoryMethod("Circle")->draw();

ShapeFactoryMethod("Square")->draw();

?>

Example 3: Class constants and class scope

==========================================

<?php

class ErrorCodes {

const FATAL = "Fatal error\n";

const WARNING = "Warning\n";

const INFO = "Informational message\n";

function print\_fatal\_error\_codes()

{

print "FATAL = " . FATAL;

print "self::FATAL = " . self::FATAL;

}

}

/\* Call the static function and move into the ErrorCodes scope \*/

ErrorCodes::print\_fatal\_error\_codes();

?>

Example 4: Regular object method using both local and global functions

======================================================================

<?php

class HelloWorld {

const HELLO\_WORLD = "Hello, World!\n";

function get\_hello\_world()

{

return HELLO\_WORLD;

}

function length\_of\_hello\_world()

{

$str = $this->get\_hello\_world();

return strlen($str);

}

}

$obj = new HelloWorld();

print "length\_of\_hello\_world() = " . $obj->length\_of\_hello\_world();

print "\n";

?>

Example 5: Multiple derefencing of objects returned from methods

================================================================

<?php

class Name {

function Name($\_name)

{

$this->name = $\_name;

}

function display()

{

print $this->name;

print "\n";

}

}

class Person {

function Person($\_name, $\_address)

{

$this->name = new Name($\_name);

}

function getName()

{

return $this->name;

}

}

$person = new Person("John", "New York");

$person->getName()->display();

?>

Example 6: Exception handling

=============================

<?

class MyException {

function MyException($\_error) {

$this->error = $\_error;

}

function getException()

{

return $this->error;

}

}

function ThrowException()

{

throw new MyException("'This is an exception!'");

}

try {

} catch (MyException $exception) {

print "There was an exception: " . $exception->getException();

print "\n";

}

try {

ThrowException();

} catch (MyException $exception) {

print "There was an exception: " . $exception->getException();

print "\n";

}

?>

Example 7: \_\_clone()

===================

<?

class MyCloneable {

static $id = 0;

function MyCloneable()

{

$this->id = self::$id++;

}

function \_\_clone()

{

$this->name = $that->name;

$this->address = "New York";

$this->id = self::$id++;

}

}

$obj = new MyCloneable();

$obj->name = "Hello";

$obj->address = "Tel-Aviv";

print $obj->id;

print "\n";

$obj = $obj->\_\_clone();

print $obj->id;

print "\n";

print $obj->name;

print "\n";

print $obj->address;

print "\n";

?>

Example 8: Unified constructors

===============================

<?

class BaseClass {

function \_\_construct()

{

print "In BaseClass constructor\n";

}

}

class SubClass extends BaseClass {

function \_\_construct()

{

parent::\_\_construct();

print "In SubClass constructor\n";

}

}

$obj = new BaseClass();

$obj = new SubClass();

?>

Example 9: Destructors

=======================

<?php

class MyDestructableClass {

function \_\_construct()

{

print "In constructor\n";

$this->name = "MyDestructableClass";

}

function \_\_destruct()

{

print "Destroying " . $this->name . "\n";

}

}

$obj = new MyDestructableClass();

?>