- Clases
  - Colección de variables (miembros / propiedades) y funciones (métodos) que trabajan con estas variables
  - Las variables son definidas con `var`
  - Las funciones con `function`
  - Ejemplo:

```php
<?php
class prueba {
  var $variable;

  function prueba() {
    print 'Prueba ' . $variable;
  }
}
?>
```

- Objetos
  - Instancia de una clase

```
<?php
$p1 = new prueba;
$p2 = new prueba;
?>
```

- Ejemplo de clase
  -
```php
<?php
class Carrito {
    var $items;  // Items en nuestro carrito de compras

    // Agregar $num artículos de $artnr al carrito
    function agregar_item($artnr, $num) {
        $this->items[$artnr] += $num;
    }

    // Tomar $num artículos de $artnr del carrito
    function retirar_item($artnr, $num) {
        if ($this->items[$artnr] > $num) {
            $this->items[$artnr] -= $num;
            return true;
        } elseif ($this->items[$artnr] == $num) {
            unset($this->items[$artnr]);
            return true;
        } else {
            return false;
        }
    }
}
?>
```

`$this` → referencia al objeto (instancia) que está usando

- Ejemplo de objeto
  -
```php
<?php
$carrito = new Carrito;
$carrito->agregar_item("0010", 1);

$otro_carrito = new Carrito;
$otro_carrito->agregar_item("0815", 3);
?>
```

- Asignación de objetos
  -

```php
<?php
$instance = new SimpleClass();

$assigned   =  $instance;
$reference  = &$instance;

$instance->var = '$assigned will have this value';

$instance = null; // $instance and $reference become null

var_dump($instance);
var_dump($reference);
var_dump($assigned);
?>
```

```
NULL
NULL
object(SimpleClass)#1 (1) {
  ["var"]=>
    string(30) "$assigned will have this value"
}
```

- Comparación de objetos
  - Dos objetos son iguales (==) si tienen los mismos atributos y valores, y son instancias de la misma clase
  - Dos objetos son idénticos (===) si refieren a la misma instancia de la misma clase

- Herencia (extends)
  - Una clase puede heredar miembros y métodos de otra clase (su clase "padre")

```php
<?php
class SimpleClass {
    // member declaration
    public $var = 'a default value';

    // method declaration
    public function displayVar() {
        echo $this->var;
    }
}
class ExtendClass extends SimpleClass {
    // Redefine the parent method
    function displayVar() {
        echo "Extending class\n";
        parent::displayVar();
    }
}
$extended = new ExtendClass();
$extended->displayVar();
?>
```

- Constructores
  - PHP4
    - ```
      class Carrito {
          var $fecha_hoy;
          var $items = array("VCR", "TV");

          function Carrito() {
              $this->fecha_hoy = date("Y-m-d");
              /* etc. . . */
          }
      }
      ```
  - PHP5
    - ```
      class BaseClass {
        function __construct() {
            print "In BaseClass constructor\n";
        }
      }
      ```
    - Si no encuentra una función __construct() buscará una función con el mismo nombre que la clase

- Constructores. Ejemplo herencia
  -
```php
<?php
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();
?>
```

- Destructores
  - Invocado
    - Automáticamente cuando se pierde toda referencia a un objeto
    - Explícitamente por el programador

```php
<?php
class MyDestructableClass {
   function __construct() {
       print "In constructor\n";
       $this->name = "My Destructable Class";
   }
   function __destruct() {
       print "Destroying " . $this->name . "\n";
   }
}
$obj = new MyDestructableClass();
$obj = null;
?>
```

Si fuera necesaria la ejecución del destructor de la clase padres, deberá ser invocado explícitamente: `parent::__destruct()`

- Visibilidad (de miembros y métodos)
  - Public
    - Pueden ser accedidos desde cualquier lugar
  - Protected
    - Pueden ser accedidos desde la clase que los define o desde cualquiera de sus herederas
  - Private
    - Pueden ser accedidos sólo desde la clase que los define

- Visibilidad de miembros
  -
```php
class MyClass {
    public $publ = 'Public';
    protected $prot = 'Protected';
    private $priv = 'Private';

    function printHello() {
        echo $this->publ;
        echo $this->prot;
        echo $this->priv;
    }
}

class MyClass2 extends MyClass {
    // We can redeclare the public and protected method, but not private
    protected $prot = 'Protected2';

    function printHello() {
        echo $this->publ;
        echo $this->prot;
        echo $this->priv;
    }
}

$obj = new MyClass();
echo $obj->publ; // Works
echo $obj->prot; // Fatal Error
echo $obj->priv; // Fatal Error
$obj->printHello(); // Shows Public, Protected and Private

$obj2 = new MyClass2();
echo $obj2->publ; // Works
echo $obj2->priv; // Undefined
echo $obj2->prot; // Fatal Error
$obj2->printHello(); // Shows Public, Protected2, not Private
```

- Visibilidad de métodos
  -
```
class MyClass {
    public function __construct() { }        // Contructors must be public
    public function MyPublic() { }           // Declare a public method
    protected function MyProtected() { }     // Declare a protected method
    private function MyPrivate() { }         // Declare a private method
    function Foo() {                         // This is public
        $this->MyPublic();
        $this->MyProtected();
        $this->MyPrivate();
    }
}

class MyClass2 extends MyClass {
    function Foo2() {                // This is public
        $this->MyPublic();
        $this->MyProtected();
        $this->MyPrivate();          // Fatal Error
    }
}

$myclass = new MyClass;
$myclass->MyPublic();       // Works
$myclass->MyProtected();    // Fatal Error
$myclass->MyPrivate();      // Fatal Error
$myclass->Foo();            // Public, Protected and Private work

$myclass2 = new MyClass2;
$myclass2->MyPublic();      // Works
$myclass2->Foo2();          // Public and Protected work, not Private
```

- Métodos “Final”
  - No pueden ser sobreescritos por las clases herederas

```php
<?php
class BaseClass {
   public function test() {
       echo "BaseClass::test() called\n";
   }

   final public function moreTesting() {
       echo "BaseClass::moreTesting() called\n";
   }
}

class ChildClass extends BaseClass {
   public function moreTesting() {
       echo "ChildClass::moreTesting() called\n";
   }
}
// Results in Fatal error: Cannot override
// final method BaseClass::moreTesting()
?>
```

- Clases “Final”
  - No pueden ser extendidas
    - ```php
      <?php
      final class BaseClass {
         public function test() {
             echo "BaseClass::test() called\n";
         }

         // Here it doesn't matter if you specify the function as final or not
         final public function moreTesting() {
             echo "BaseClass::moreTesting() called\n";
         }
      }

      class ChildClass extends BaseClass {
      }
      // Results in Fatal error: Class ChildClass may not
      // inherit from final class (BaseClass)
      ?>
      ```
      ```

- Getters y Setters

```
class Setter {
   public $n;
   private $x = array("a" => 1, "b" => 2, "c" => 3);

   private function __get($nm) {
      echo "Getting [$nm]\n";
      if (isset($this->x[$nm])) {
         $r = $this->x[$nm];
         print "Returning: $r\n";
         return $r;
      } else {
         echo "Nothing!\n";
      }
   }

   private function __set($nm, $val) {
      echo "Setting [$nm] to $val\n";
      if (isset($this->x[$nm])) {
         $this->x[$nm] = $val;
         echo "OK!\n";
      } else {
         echo "Not OK!\n";
      }
   }

   private function __isset($nm) {
      echo "Checking if $nm is set\n";
      return isset($this->x[$nm]);
   }

   private function __unset($nm) {
      echo "Unsetting $nm\n";
      unset($this->x[$nm]);
   }
}
```

```
$foo = new Setter();
$foo->n = 1;
$foo->a = 100;
$foo->a++;
$foo->z++;

var_dump(isset($foo->a)); //true
unset($foo->a);
var_dump(isset($foo->a)); //false

// this doesn't pass through the
// __isset() method
// because 'n' is a public property
var_dump(isset($foo->n));

var_dump($foo);
```

```
Setting [a] to 100
OK!
Getting [a]
Returning: 100
Setting [a] to 101
OK!
Getting [z]
Nothing!
Setting [z] to 1
Not OK!
Checking if a is set
bool(true)
Unsetting a
Checking if a is set
bool(false)
bool(true)

object(Setter)#1 (2) {
   ["n"]=>
   int(1)
   ["x:private"]=>
   array(2) {
      ["b"]=>
      int(2)
      ["c"]=>
      int(3)
   }
}
```

- Iteración sobre los miembros de un objeto
  -
```
class MyClass {
    public $var1 = 'value 1';
    public $var2 = 'value 2';
    public $var3 = 'value 3';

    protected $protected = 'protected var';
    private $private = 'private var';

    function iterateVisible() {
       echo "MyClass::iterateVisible:\n";
       foreach ($this as $key => $value) {
           print "$key => $value\n";
       }
    }
}

$class = new MyClass();

foreach ($class as $key => $value) {
    print "$key => $value\n";
}
echo "\n";

$class->iterateVisible();
```

```
var1 => value 1
var2 => value 2
var3 => value 3

MyClass::iterateVisible:
var1 => value 1
var2 => value 2
var3 => value 3
protected => protected var
private => private var
```