Classes

Basic class definitions begin with the keyword class, followed by a class name, followed by a pair of curly braces which enclose the definitions of the properties and methods belonging to the class.

class SimpleClass

{

    // property declaration

    public $var = 'a default value';

    // method declaration

    public function displayVar() {

        echo $this->var;

    }

}

See example below for $this which is defined only in context of non-static methods/properties:

class A

{

    function foo()

    {

        if (isset($this)) {

            echo '$this is defined (';

            echo get\_class($this);

            echo ")\n";

        } else {

            echo "\$this is not defined.\n";

        }

    }

}

class B

{

    function bar()

    {

        A::foo();

    }

}

$a = new A();

$a->foo();  // $this is defined (as foo() is not a static method)

A::foo(); // $this is not defined as foo() is not a static method (even if it were a static method, $this would still be undefined and self would be defined)

$b = new B();

$b->bar();  // this is not defined as foo() is not a static method in A, and even if it were, $this would still be undefined - but self would be defined

B::bar(); //this is not defined as bar() is not a static method in B (and even if it were, $this would still be undefined, but self would be defined)

\*note that if we were to change the non-static methods to static (so that ***self*** would be defined instead of this but we would need to use the syntax :

if(***self*** !== null){} instead of:

if(isset(***self***)){} \*this throws an error indicating to use the syntax above

class A

{

    static function foo()

    {

        if (isset(self)) {

            echo 'self is defined (';

            echo get\_class(self);

            echo ")\n";

        } else {

            echo "\$this is not defined.\n";

        }

    }

}

class B

{

    function bar()

    {

        A::foo();

    }

}

A::foo(); //  throws error --> Cannot use isset() on the result of an expression (you can use "null !== expression" instead)

But if we use:

class A

{

    static function foo()

    {

        if (self !== null) {

            echo 'self is defined (';

            echo get\_class(self);

            echo ")\n";

        } else {

            echo "\$this is not defined.\n";

        }

    }

}

class B

{

    function bar()

    {

        A::foo();

    }

}

A::foo(); //  self is defined

\*self will be defined

Extends:

A class can inherit the methods and properties of another class by using the keyword extends in the class declaration. It is not possible to extend multiple classes; a class can only inherit from one base class.

The inherited methods and properties can be overridden by redeclaring them with the same name defined in the parent class. However, if the parent class has defined a method as final, that method may not be overridden. It is possible to access the overridden methods or static properties by referencing them with parent::.

class SimpleClass

{

    // property 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(); //prints 'Extending class a default value';

//prints 'Extending class' (from the inheriting class method) and 'a default value' (fron the parent class method)

However, if we make the parent method final as in:

class SimpleClass

{

    // property declaration

    public $var = 'a default value';

    // method declaration

    public final  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(); //throws error as it cannot overwrite final method from parent

Autoloading classes:

spl\_autoload\_register(function ($class\_name) {

    include $class\_name . '.php';

});

$obj  = new MyClass1();

$obj2 = new MyClass2();

\*spl\_autoload\_register() loads any class, provided it exists (provided a file defining that class exists, e.g. MyClass1.php / MyClass2.php defining MyClass1 and MyClass2;