Interfaces - the next level of abstraction

Object interfaces allow you to create code which specifies which methods a class must implement, without having to define how these methods are implemented.

Interfaces are defined in the same way as a class, but with the *interface* keyword replacing the *class* keyword and without any of the methods having their contents defined.

All methods declared in an interface must be public; this is the nature of an interface.

How to declare and implement an interface?

We declare an interface with the interface keyword and, the class that inherits from an interface

with the implements keyword. Let’s see the general case:

interface interfaceName {

// Abstract methods

}

class Child implements interfaceName {

// Defines the interface methods,

// and may have its own methods.

}

In the simple example given below, we will create an interface for the classes that handle cars, which commits all its child classes to setModel() and getModel() methods.

interface Car {

public function setModel($name);

public function getModel();

}

Interfaces, like abstract classes, include abstract methods and constants. However, unlike abstract classes, interfaces can have only public methods, and cannot have variables. The classes that implement the interfaces must define all the methods that they inherit from the interfaces, including all the parameters. So, in our concrete class with the name of miniCar, we add the code to all the abstract methods

class miniCar implements Car {

private $model;

public function setModel($name)

{

$this -> model = $name;

}

public function getModel()

{

return $this -> model;

}

}

Can we implement more than one interface in the same class?

**We can implement a number of interfaces in the same class.**

interface Car {

public function setModel($name);

public function getModel();

}

interface Vehicle {

public function setHasWheels($bool);

public function getHasWheels();

}

// The class implements two interfaces

class miniCar implements Car, Vehicle {

private $model;

private $hasWheels;

public function setModel($name)

{

$this -> model = $name;

}

public function getModel()

{

return $this -> model;

}

public function setHasWheels($bool)

{

$this -> hasWheels = $bool;

}

public function getHasWheels()

{

return ($this -> hasWheels)? "has wheels" : "no wheels";

}

}