INTERFACES

Las interfaces de objetos permiten declarar los métodos que deben ser implementados por una clase, sin tener que definir el contenido de dichos métodos.

Se definen con la palabra clave ***interface***

Todos los métodos declarados en una interfaz deben ser públicos

Para implementar una interfaz, se usa el operador **implements.** Todos los métodos en una interfaz deben ser implementados dentro de la clase; el no cumplir con esta regla resultará en un error fatal.

La clase que implemente una interfaz debe utilizar exactamente las mismas estructuras de métodos que fueron definidos en la interfaz. De no cumplir con esta regla, se generará un error fatal.

Ejemplo:

```
<?php
 interface UserInterface   //  declara interface con 2 métodos
  {   public function setMenu( $menu_array );
      public function getContent($page);
  }
 class WebApp   implements   UserInterface    //  define clase que implementa la interface
 { public function setMenu( $menu_array )
    { $elmenu="<table><tr>";
      foreach($menu_array as  $k=> $v)
       {  $elmenu.="<td><a href='" . $menu_array[$k]['enlace'] . "'>" . $menu_array[$k]['etiqueta'] . "</a></td>"; }
      $elmenu.="</tr></table>";
      return $elmenu;
    }
  public function getContent($page)
       { $content = "<h1>Bienvenido a  " . ucwords($page) . " </h1>";      return $content;    }
}
class WebApp2   implements UserInterface    //  otra clase que implementa la interface
 {  public function getContent($page)            //  pero  con  ERROR ,  ya que falta definir método setMenu
      { $content = date('d-m-Y') . "<h3>$page</h3>";   return $content;   }
 }
//  ----------------------           proceso principal --------------------------
for ($i=1;$i<5;$i++)     //  carga array con opciones de menu
 { $menu[$i]['enlace'] = "enlace$i.php" ;   $menu[$i]['etiqueta'] ="etiqueta$i" ;  }
$miWeb = new WebApp;     //  creación y uso de  objetos
echo  $miWeb->setMenu($menu);
echo  $miWeb->getContent('LA  WEB');
?>
```

Las clases pueden implementar más de una interfaz, separándolas cada una por una coma.

```
interface UserInterface   //  declara interface con 2 métodos
  {   public function setMenu( $menu_array );
      public function getContent($page);
  }
interface Publicidad       //  declara interface con 1 método
  { public function  getAnuncios() ; }

class WebApp implements UserInterface , Publicidad    //  declara clase que implementa 2 interfaces
...
```

Las interfaces se pueden extender al igual que las clases utilizando el operador **extends**.

```php
<?php
interface a  { public function A();}

interface b  { public function B();}

interface c extends a, b { public function C();}

class XX implements c
  {    public function A()    { ..... }
       public function B()    { .....  }
       public function C()    { .....   }
   }
?>
```

Las constantes de interfaces funcionan como las constantes de clases excepto porque no pueden ser sobrescritas por una clase/interfaz que las herede.

```php
<?php
interface a  { const MAX_LENGTH = "100"; }

class XX implements a {...}

 $x=new XX();

echo a::MAX_LENGTH ;    //   Formas válidas de referenciar constantes de interfaces
echo  XX::MAX_LENGTH;
echo $x->MAX_LENGTH;

class ZZ implements a { const MAX_LENGTH = "500";} //  Error:  no se permite sobreescribir constantes

?>
```

Antes de PHP 5.3.9, una clase no puede implementar dos interfaces que especifiquen un método con el mismo nombre, ya que podría causar ambigüedad.
Las versiones más recientes de PHP permiten esto siempre y cuando los métodos duplicados tengan la misma firma.