ACTIVITATS UD4 POO

Actividad 2.13. Primeras clases y objetos

Crearemos la clase Producto

Atributos: Peso, Precio, Stock todas private.

Métodos:

Constructor: Método mágico, asignará valor a todos los atributos

Asignar: Devolverá un array asociativo donde la clave del array es el nombre del atributo y el valor, su valor.

Get: Método mágico.

Crearemos una vista con un formulario para darle valores a los atributos y un controlador que instancia el objeto y muestra sus valores.

Actividad 2.14. Subir archivos con POO.

Modificaremos las actividade 2.10 para incluir objetos:

Crearemos un archivo separado que definirá una clase llamada *Imagen* que se utilizara para ubicar propiedades y métodos del array asociativo $_FILES

Tendrá las propiedades o atributos 'tmp_name', 'name', y 'type' y

los métodos :

"*esta_cargado()*" que devolverá true si existe el archivo temporal.

cambiar_nombre()" que actualizara la propiedad 'name' a la propiedad con el nombre completo (directorio e id --> si hace falta)

"*mover()*" que copiara definitivamente el archivo temporal a su directorio definitivo con su nombre definitivo.

Este fichero constituirá el modelo y se incluirá con un require al principio del controlador y modificaremos toda la programación para utilizar un objeto de esta clase.

Actividad 2.15. Herencia.

MODELO

Será un archivo llamado modelo.php, que contendrá la definición de todas las clases.

CLASES

Estas clases heredarán todas de la clase Producto de la actividad 4.1

Monitor:

Hereda de Productos pero además tiene los siguientes atributos y métodos

Atributos: *Pulgadas* que será privada.

Métodos:

Constructor: Será un constructor mágico que recibe todos los atributos, tanto del producto como del monitor como parámetro. Llamará al constructor padre para asignar los valores de productos y asignará el valor del atributo pulgadas.

Asignar: Igual que el anterior llamará al asignar padre y además añadirá una posición más al array asociativo; la del atributo pulgadas.

DiscoDuro:

Hereda de Producto pero además tiene los siguientes atributos y métodos

Atributos: *capacidad* que será privada.

Métodos:

Constructor: Será un constructor mágico que recibe todos los atributos, tanto del producto como del discoduro como parámetro. Llamará al constructor padre para asignar los valores de productos y asignará el valor del atributo capacidad.

Asignar: Igual que el anterior llamará al asignar padre y además añadirá una posición más al array asociativo; la del atributo capacidad.

VISTAS

Serán los siguientes archivos php todos incluidos en una carpeta llamada vistas.

Formulario.php: Archivo que muestra en pantalla el formulario.

mostrar.php: Archivo que muestra el array asociativo creado con asignar. (muestra todos sus atributos)

CONTROLADOR

Será un único fichero php llamado *index.php*, que mostrará un formulario con todos los atributos de todas las clases y una lista desplegable para elegir si es monitor o disco duro.

En función del tipo de producto elegido, se crea un objeto de ese tipo y utilizando sus métodos, se asignan los valores introducidos a su array asociativo. Finalmente se mostrará por pantalla el contendido de este objeto utilizando la vista diseñada para esto.

Actividad 2.16 POO avanzada.

Vamos a crear una serie de clases e interfaces para generar campos de formulario que sirvan para seleccionar un elemento de entre una lista de elementos dados, es decir, para generar elementos de tipo select y radio.

Para generar uno de estos campos necesitaremos los siguientes datos:

El *título* del campo, que pondremos en un label antes del mismo.

El *nombre* del campo, que pondremos en el atributo name.

Un *array* con los valores de los elementos y los textos que se mostrarán. El array contendrá los valores en las claves y los textos en los elementos del array.

Un *entero* que indique cuál es el elemento seleccionado por defecto.

Crearemos un interface llamado *ISelectorIndividual* que tendrá dos métodos:

Un constructor que reciba todos estos parámetros

Un método *generaSelector* que no recibirá parámetros y que nos servirá para generar el elemento de formulario en cuestión

Crearemos una clase abstracta *SelectorIndividual* que implemente el interface anterior:

El constructor se implementará en esta clase abstracta, por lo tanto, contendrá los datos miembros para almacenar los parámetros del constructor

La función generaSelector será abstracta

Crearemos dos clases *SIRadioOpcion* y *SISelect* que heredarán de la clase *SelectorIndividual* e implementarán la función *generaSelector*, una para generar la lista de radio botones y la otra para generar el select

Finalmente, crearemos un *index.php* que utilice estas dos clases para generar un formulario que contenga los siguientes campos:

- Ciudad: desplegable que contendrá los elementos Alicante, Valencia Castellón
- Nivel idioma: desplegable que contendrá los elementos Alto, Medio y Bajo
- Sexo: radio botones con los elementos Mujer, Hombre
- Estado: Radio botones con los elementos encendido y apagado.

Al pulsar enviar en el formulario se deben mostrar los valores seleccionados en los elementos.