Relación 4: Clases, objetos y enumeraciones.

Crear la carpeta /aplicacion/relacion4. Además, crearemos la carpeta /scripts/clases donde haremos un fichero php con la forma "XXX. php" por cada clase. Por ejemplo, la clase InstrumentoBase se guardará en el fichero "InstrumentoBase.php" (mucho ojo con mayúsculas y minúsculas: el nombre de la clase debe coincidir con el nombre de fichero). Se debe garantizar la carga de estos ficheros mediante la autocarga (función spl_autoload_register() – revisar fichero cabecera.php).
Se tendrá el fichero index.php con código para probar el funcionamiento de las clases. En el controlador de index.php, se definirán variables para el acceso a las clases y se mostrará el funcionamiento de las distintas clases en la vista.

Se deben tener en cuenta todos los requisitos indicados en prácticas anteriores como modelo-vista-contralador, comentarios, barra de ubicación, uso de las funciones mb_, definición de tipos en funciones, control de errores, etc

NOTA: Los valores por defecto para atributos, se indican en el constructor en el correspondiente parámetro. Se suelen poner también el valor al definir los atributos.

Se definirán los tipos tanto en los parámetros de los métodos como en las propiedades.
Las propiedades privadas/protegidas se declararán con _ (propiedad descripción se declarará como _descripcion)
Los métodos son públicos salvo que se diga otra cosa.
No se crearán más métodos públicos que los que se indican. Tienes unos requisitos sobre la interfaz de la clase (métodos/propiedades públicas) y no se deben exponer otros elementos.
Se pueden crear tantos métodos privados/protegidos como estimemos.
Por defecto, se debe deshabilitar la sobrecarga dinámica en las clase (redefinir los métodos __get, __set, __isset, __unset)

1.- Crear la clase abstracta IntrumentoBase con el siguiente comportamiento. Definirla inicialmente como no abstracta para probar su funcionamiento. Una vez probado se pasará a abstracta:

- Tiene el atributo descripción de tipo cadena. La descripción se puede fijar mediante el constructor o mediante métodos get/set públicos (ej getCadena/setCadena, no confundir con lo que veremos después como método mágico __get y __set). Directamente no puede ser accesible.
- Tiene el atributo edad que no es accesible desde el exterior aunque si desde las clases descendientes. Su valor por defecto será 10. Se tiene método get para devolver su valor. Se asignará en el constructor.

  Para definir los atributos con valor por defecto se declaran en la clase, con el valor por defecto, se define un parámetro en el constructor con el valor por defecto y se asigna en el constructor. Así garantizas que cuando se llame al constructor, si no se indica ningún valor, se asigne el de por defecto.

- Se definen dos métodos abstractos: sonido (devuelve una frase que representa el sonido) y afinar (devuelve una cadena con los pasos necesarios para afinar el instrumento)
- Se define el método envejecer, que aumenta la edad del instrumento en un año.
- Se tiene una variable de clase protegida que controla cuantos elementos se han creado y una propiedad de instancia para indicar cuál es el orden de creación de la instancia. Cada vez que se crea un instrumento se le asigna a la instancia el número de orden de creación del instrumento actual y se incrementa el valor de la variable de clase.
- Cuando se muestre en un echo aparecerá "instrumento con descripción XXXXX, instancia 9999 de un total de 999. Tiene 999 años. La clase es XXXX". La clase debe ser obtenida mediante get_class. (999 representa un numero, XXX representa una cadena)
- No tiene la sobrecarga de propiedades por defecto.

2.- Se tiene la interfaz IFabricable que permite indicar el proceso de fabricación y como se recicla. Esta interfaz establece los métodos:

- metodoFabricacion: Devuelve los pasos para fabricar el elemento
- metodoReciclado: Devuelve el método de reciclado que se debe aplicar al elemento según su fabricación.

3.- Definir la clase IntrumentoViento que hereda de InstrumentoBase con el siguiente comportamiento:

- La descripción es "instrumento de viento".
- Tiene el atributo material con valor por defecto madera. Este atributo no es accesible directamente desde el exterior aunque si desde las clases derivadas. No se tienen métodos set/get para esta propiedad.
- Al constructor se le pasa como parámetro el material y la edad (valor por defecto 15).
- El método afinar no debe ser redefinible en las clases derivadas de esta.
- Cuando se muestre en un echo aparecerá, junto a todo lo indicado por la clase base, el texto "Instrumento de material XXXX".

4.- Definir la clase Flauta que es un instrumento de viento que es fabricable y de la que no se puede heredar ninguna otra subclase.

- El constructor se hará privado
- Se tiene el método estático CrearDesdeArray (constructor estatico) con parámetro un Array con posiciones asociativas "material" y "edad" (pueden no aparecer por lo que se tienen en cuenta valores por defecto) y que devuelve una instancia de Flauta con los valores indicados.
- Definir el método de clonación para la clase flauta de forma que el nuevo objeto tenga los mismos valores, salvo la edad que será 0 y el número de instancia (propiedad) /número de instancias (propiedad de clase), ya que realmente he creado una nueva instancia.

5.- Definir el tipo enumerado EstadoCivil con las siguientes características:

- Casos de soltero, casado, pareja y viudo con valores 10, 20, 30 y 40 respectivamente.

- Método descripcion que devuelve el nombre del caso concreto.
- Método valor que devuelve el valor del caso concreto.
- Método estático casos que devuelve un array con todos los casos

Definir la clase Persona que tendrá:

- propiedades nombre (string), fecha_nacimiento (string), domicilio (string), localidad (string) y estado (EstadoCivil). Todas las propiedades son privadas.
- Un conjunto de sentencias get/set para devolver/poner su valor.
- Constructor privado al que no se indica ningún parámetro y que rellena las propiedades con valores nombre “prueba”, fecha_nacimiento “01/01/2000”, domicilio “carrera 12”, localidad “antequera” y estado “soltero”.
- Constructor estático registrarPersona, que recibe parámetros para todas las propiedades
- Método toString que devuelve la cadena “*Nombre* es una persona *estado* nacida en *fecha_nacimiento* y que vive en *localidad*”

Probar la clase creando una persona y asignando un estado a partir de un valor aleatorio de entre los valores posibles que obtendremos del EstadoCivil

6.- Crear la clase SerieFibonacci que implementa la serie de Fibonacci. Esta serie se basa en la fórmula F(x)=F(x-1)+F(x-2) siendo F(0)=0 y F(1)=1. Para obtener el valor F(n) se puede usar la fórmula alternativa

$$f_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

Para implementar esta clase podéis usar la interfaz Iterator y tener dos propiedades privadas $_limite y $_claveActual. $limite indica hasta qué valor se generaría la serie. $_claveActual indica cual es el elemento actual para el que se calcula la serie.
El funcionamiento tendría que ser algo así:

```
foreach(new SerieFibonacci(10) as $valor)
{
    echo “$valor  ”;
}
/* Generaria 10 valores
    0 1 1 2 3 5 8 13 21 34 55
*/
```

Crear la función fFibonacci($ultimo), como función estática dentro de la clase SerieFibonacci, que debe implementar la serie de Fibonacci usando para ello un generador.

7.- Crear la clase ClaseMisPropiedades. En esta clase se define una propiedad privada llamada $_propiedades. Se debe permitir añadir en tiempo de ejecución cualquier nueva propiedad (propiedades dinámicas) a una instancia. Además tenemos las propiedades de instancia$propPublica (publica, mixed) y $_propPrivada (privada, mixed, inicializada a 25). Se debe garantizar que no se pueda asignar un valor a una propiedad que llamemos _propPrivada (creada de forma dinámica).

Por ejemplo, este código debería ser válido.

```
$Objeto= new ClaseMisPropiedades ();
$Objeto->propPublica=”publica”;
//$Objeto->_propPrivada=”privada”;  //no es valida al ser privada
$Objeto->propiedad1=25;
$Objeto->propiedad2=”cadena de texto”;
echo “la propiedad 1 vale “.$Objeto->propiedad1.”<br>”.
// echo $Objeto->propiedad3; // esto debería dar un error al no haber asignado previamente la propiedad
```

Además la clase deber ser iterable. Así cuando se haga un foreach($Objeto as $clave=>$valor) se deben devolver como claves las propiedades con la palabra oi_ delante (ejemplo para la propiedad propiedad1, se devolverá como clave oi_propiedad1), incluyendo propPublica y _propPrivada comoúltimos elementos a devolver en el foreach.