RELACIÓN 6: TIENDA (RESUMEN)

Crear un nuevo proyecto llamado tienda con la estructura de carpetas que se ha trabajado en prácticas anteriores. Es un proyecto nuevo por lo que no debe aparecer lo que hemos hecho en prácticas anteriores.
El proyecto pretende gestionar una tienda en la que se venden muebles con diferentes características.
Las clases se colocarán en /aplicacion/clases/curso2025 (garantizar la autocarga) en un fichero con el mismo nombre que la clase.
Se deben de cumplir todas las restricciones indicadas en prácticas anteriores: publicado en página web (borrar lo que hay actualmente), comentarios, barra de ubicación, menu principal, etc

Se pide:
1.- Crear la clase abstracta MuebleBase con el siguiente comportamiento.

- Constante pública **MATERIALES_POSIBLES**. Es un array con los materiales que vamos a tener disponibles (definir al menos 4 materiales). Cada elemento del array será de la forma código => descripcion. Ejemplo 1 => "madera".
- Constante **MAXIMO_MUEBLES**. Representa el número máximo de muebles que se pueden crear.
- Variable de clase private **mueblesCreados**. Número de muebles (instancias) creados hasta el momento.
- Propiedades **Nombre** (cadena de máximo 40, no puede estar vacía, a mayúscula), **Fabricant**e (cadena de máximo 30, por defecto 'FMu:', siempre se añade al principio FMu: si no lo tiene ya), **Pais** (cadena, máximo 20 caracteres, por defecto 'ESPAÑA'), **Anio** (entero, año de inicio de fabricación, valor entre 2020 y el año actual, por defecto 2020), **FechaIniVenta** (cadena-fecha, fecha en la que se empieza a vender el mueble, no puede ser anterior al uno de enero del Anio de inicio de fabricación, por defecto '01/01/2020'), **FechaFinVenta** (cadena-fecha, fecha en la que dejará de venderse, no puede ser anteriora la FechaIniVenta, por defecto 31/12/2040'), **MaterialPrincipal** (entero, número que representa el material de entre los materiales definidos en la constante MATERIALES_POSIBLES) y **Precio** (real, por defecto 30, no puede ser menor de 30).
  No se puede acceder directamente a las propiedades. Se tiene que hacer con métodos set/get. Además, se tendrá el método getMaterialDescripcion que devolverá una cadena con la descripción del material.
  En los métodos set se tienen que verificar las restricciones de cada campo. Si no hay error se devuelve true y se asigna el valor en la propiedad correspondiente. Si hay error se devuelve false y no se asigna ningún valor en la propiedad. Se usará, cuando siempre que sea posible, la API de validaciones vista en la práctica anterior (la api debe estar en /scripts/libreras)
- Esta clase no permite la sobrecarga dinámica de propiedades
- **Constructor**. Recibe valores para las propiedades teniendo en cuenta los posibles valores por defecto, salvo el nombre que es obligatorio que se indique. Se deben verificar que los valores sean correctos llamando a los métodos set correspondientes. Si falla la validación en el nombre se debe lanzar un error. Para cualquier otro campo no se debe producir error. En caso de que el método set correspondiente devuelva false, se asignará el valor por defecto. Además, si se ha superado el máximo de instancias creadas no se debe permitir la creación (lanzar excepción).

- Método **dameListaPropiedades** que devuelve un array con el nombre de las propiedades que tiene. Cread un array con los valores posibles (no uséis funciones php para recoger las propiedades definidas en la clase, las definís vosotros).
  Si estamos en una clase hija se deben devolver las propiedades de la clase más las de la clase base.
- Método **damePropiedad** al que se le pasa como parámetros una cadena que representa la propiedad, el modo y la variable $res (se devolverá aquí el valor de la propiedad). Devuelve true o false si tiene o no esa propiedad.
  Según el valor de modo (1 o 2), se usará una variable-funcion o variable-variable bien para acceder al método get de la propiedad o a la propiedad.
  Tened en cuenta que si el modo es 2 (variable-variable) y estamos en una clase hija, no podremos acceder directamente a las propiedades privadas de la clase base. Llamaremos en este caso al método get correspondiente.
- Método de clase **puedeCrear** que devuelve true si se pueden crear más instancias de la clase y false en caso contrario. Además, devuelve en el parámetro Número el número de muebles que aún se pueden crear de la clase.
- Método que al convertir la instancia en cadena nos devuelve 'MUEBLE de clase XXX con nombre XXX, fabricante XXX, fabricado en XXX a partir del año 999, vendido desde 99/99/9999 hasta 99/99/9999, precio 9999 de material XXXX'.

2.- Crea la clase **MuebleReciclado** que hereda de MuebleBase y tiene la propiedad adicional PorcentajeReciclado (entero, con valor por defecto 10, debe estar entre 0 y 100) y la clase **MuebleTradicional** que hereda de MuebleBase y tiene las propiedades Peso (real, entre 15 y 300, por defecto 15) y Serie (cadena, por defecto S01). De ambas clases no se pueden crear subclases.

Las propiedades de estas clases se pueden establecer en el constructor y mediantes funciones set/get. Tanto en las funciones set como en el constructor se debe tener el mismo funcionamiento que para MuebleBase. Las propiedades deben aparecer cuando se convierta la instancia a cadena.
Tanto el método damePropiedad como dameListaPropiedades debe controlar correctamente todas las propiedades. No se pueden cambiar en MuebleBase la visibilidad de las propiedades de prívate a protected.

3.- Crea la clase Características, en la que se irán indicando otras características. De la siguiente forma:

- Estas características se crearán de forma dinámica.
- Las características pueden ser recorridas en un foreach.
- Siempre se tiene las características (dinámica) **ancho, alto y largo** (enteros) con valor por defecto 100 que representan el ancho, alto y largo del mueble en centímetros.
- Si se indica la característica **ningunamas**, no se podrá indicar ninguna nueva característica (cualquier nueva debe dar un error) pero si se podrá seguir modificando las existentes.

Modificar la clase MuebleBase con:

- Definir una propiedad privada de tipo Caracteristicas.
- Método **anadir** que recibirá pares de argumentos en la forma (string carac, mixed valor) que se corresponden con el nombre de la característica y el valor de la misma y que añadirá la característica a la propiedad privada. (pueden indicarse 2, 4, 6 … parámetros, si se indica un número impar se desecha el ultimo parámetro). Ejemplo, anadir('saliente',

true, ‘precio’, 300)

- Método **exportarCaracteristicas** que devuelve una cadena con todas las características del equipo en la forma car:valor, donde cada característica debe estar en una fila distinta.
- Al convertirse la instancia en cadena deben mostrarse las características.

Crear el fichero /aplicación/principal/pruebas.php que muestre el funcionamiento de las clases y todos los métodos de las mismas.

4.- Ya que no hemos visto todavía ningún método para almacenar datos de forma permanente, vamos a definir un array indexado (posiciones a partir de 1) con 5 muebles (MuebleReciclado/MuebleTradicional). Cada mueble tendrá al menos dos características adicionales a las básicas. Definir este array en un único punto de forma que esté accesible en todas las páginas.

5.- Crear la interfaz de la aplicación con el siguiente funcionamiento:

- Se tendrá el fichero /aplicación/principal/index.php. Este va a ser nuestra ventana principal. Automáticamente /index.php nos debe redireccionar a esta página.
- En el fichero /aplicacion/principal/index.php se mostrará:

  Tabla con todos los muebles que tenemos. Se mostrarán 6 o más columnas por mueble (al menos se mostrará el índice del mueble en el array, el tipo del mueble y el nombre).

  Un solo formulario con dos botones de envío (Mostrar Mueble y Modificar Mueble) y un elemento para indicar el mueble sobre el que trabajar (select, input, etc)
- Al pulsar el botón del formulario M**ostrar Mueble** mostrará las propiedades del mueble seleccionado. Las propiedades a mostrar las obtendrá del método dameListaPropiedades y usará el método damePropiedad para obtener el valor.
- Al pulsar el botón formulario **Modificar Mueble** se abrirá una nueva ventana con un formulario que permitirá modificar los datos del mueble indicado en la caja (el material se debe elegir a partir de los materiales disponibles, no indicando directamente un numero). En el formulario se mostrarán inicialmente los datos que tiene el mueble según el tipo del mismo. Cuando se envíe el formulario, se deben validar los datos enviados según las condiciones del apartado 1 y 2. La mitad mediante una instancia auxiliar del tipo de mueble, llamando a los métodos set, y la mitad otra de las propiedades se debe validar directamente usando condiciones. Si todo es correcto se actualizará el mueble correspondiente y se mostrará un resumen con los datos recogidos. Habrá un enlace a ir a Index.php que nos permitirá volver a la página principal

  Ojo, al no trabajar con sesiones, al recargar la página no aparecerán los cambios en el mueble hemos modificado.