unidad didáctica 03

Actividades

Índice

1. **investigación**
2. **objetos**
3. **proyecto Videoclub**
4. **proyecto Videoclub 2.0**

1. investigación

Actividad 301

Investiga la diferencia entre un paradigma orientado a objetos basado en clases (*PHP*) respecto a uno basado en prototipos (*JavaScript*).

2. objetos

Actividad 302

`302Empleado.php`: Crea una clase `Empleado` con su *nombre*, *apellidos* y *sueldo*. Encapsula las propiedades mediante *getters/setters* y añade métodos para:

- Obtener su nombre completo → `getNombreCompleto(): string`
- Que devuelva un booleano indicando si debe o no pagar impuestos (se pagan cuando el sueldo es superior a 3333€) → `debePagarImpuestos(): bool`

Actividad 303

`303EmpleadoTelefonos.php`: Copia la clase del ejercicio anterior y modifícala. Añade una propiedad privada que almacene un array de números de teléfonos. Añade los siguientes métodos:

- `public function anyadirTelefono(int $telefono) : void` → Añade un teléfono al array.
- `public function listarTelefonos(): string` → Muestra los teléfonos separados por comas.
- `public function vaciarTelefonos(): void` → Elimina todos los teléfonos.

Actividad 304

`304EmpleadoConstructor.php`: Copia la clase del ejercicio anterior y modifícala. Elimina los setters de `nombre` y `apellidos`, de manera que dichos datos se asignan mediante el constructor (utiliza la sintaxis de PHP7). Si el constructor recibe un tercer parámetro, será el sueldo del `Empleado`. Si no, se le asignará 1000€ como sueldo inicial.

`304EmpleadoConstructor8.php`: Modifica la clase y utiliza la sintaxis de PHP 8 de promoción de las propiedades del constructor.

Actividad 305

`305EmpleadoConstante.php`: Copia la clase del ejercicio anterior y modifícala. Añade una constante `SUELDO_TOPE` con el valor del sueldo que debe pagar impuestos, y modifica el código para utilizar la constante.

Actividad 306

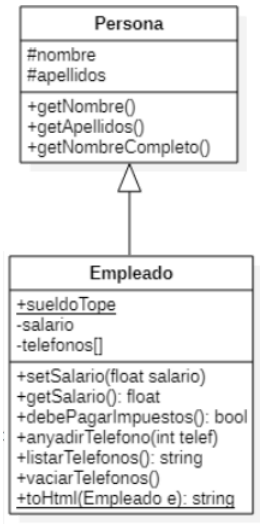
`306EmpleadoSueldo.php`: Copia la clase del ejercicio anterior y modifícala. Cambia la constante por una variable estática `sueldoTope`, de manera que mediante getter/setter puedas modificar su valor.

Actividad 307

`307EmpleadoStatic.php`: Copia la clase del ejercicio anterior y modifícala. Completa el siguiente método con una cadena HTML que muestre los datos de un empleado dentro de un párrafo y todos los teléfonos mediante una lista ordenada (para ello, deberás crear un getter para los teléfonos):

- `public static function toHtml(Empleado $emp): string`

Actividad 308



`308Persona.php`: Copia la clase del ejercicio anterior en `308Empleado.php` y modifícala.

Crea una clase `Persona` que sea padre de `Empleado`, de manera que `Persona` contenga el nombre y los apellidos, y en `Empleado` quede el salario y los teléfonos.

Actividad 309

`309PersonaH.php`: Copia las clases del ejercicio anterior y modifícalas. Crea en `Persona` el método estático `toHtml(Persona $p)`, y modifica en `Empleado` el mismo método `toHtml(Persona $p)`, pero cambia la firma para que reciba una `Persona` como parámetro.

Para acceder a las propiedades del empleado con la persona que recibimos como parámetro, comprobaremos su tipo:

```php
<?php
class Empleado extends Persona {
    /// resto del código


    public static function toHtml(Persona $p): string {
        if ($p instanceof Empleado) {
            // Aqui ya podemos acceder a las propiedades y métodos de Empleado
        }
    }
}
```

Actividad 310

`310PersonaE.php`: Copia las clases del ejercicio anterior y modifícalas.

Añade en `Persona` un atributo `edad`.

A la hora de saber si un empleado debe pagar impuestos, lo hará siempre y cuando tenga más de 21 años y dependa del valor de su sueldo. Modifica todo el código necesario para mostrar y/o editar la edad cuando sea necesario.

Actividad 311

`311PersonaS.php`: Copia las clases del ejercicio anterior y modifícalas.
Añade nuevos métodos que hagan una representación de todas las propiedades de las clases `Persona` y `Empleado`, de forma similar a los realizados en HTML, pero sin que sean estáticos, de manera que obtenga los datos mediante `$this`.

- `function public __toString(): string`

> **magic methods**
>
> El método `__toString()` es un método mágico que se invoca automáticamente cuando queremos obtener la representación en cadena de un objeto.

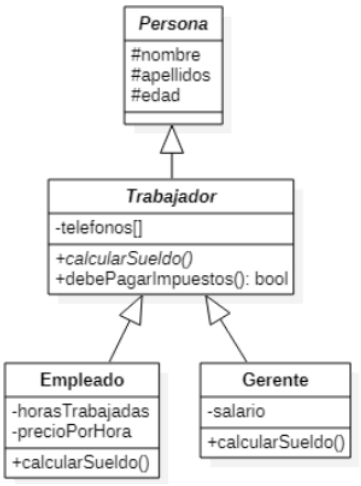
Actividad 312

`312PersonaA.php`: Copia las clases del ejercicio anterior y modifícalas.
Transforma `Persona` a una clase abstracta donde su método estático `toHtml(Persona $p)` tenga que ser redefinido en todos sus hijos.

Actividad 313

`313Trabajador.php`: Copia las clases del ejercicio anterior y modifícalas.

- Cambia la estructura de clases conforme al gráfico respetando todos los métodos que ya están hechos.
- `Trabajador` es una clase abstracta que ahora almacena los `telefonos` y donde `calcularSueldo` es un método abstracto de manera que:
    - El sueldo de un `Empleado` se calcula a partir de las horas trabajadas y lo que cobra por hora.
    - Para los `Gerente`s, su sueldo se incrementa porcentualmente en base a su edad: `salario + salario*edad/100`



Actividad 314

`314Empresa.php`: Utilizando las clases de los ejercicios anteriores:

- Crea una clase `Empresa` que además del nombre y la dirección, contenga una propiedad con un array de `Trabajador`es, ya sean `Empleado`s o `Gerente`s.
- Añade *getters/setters* para el nombre y dirección.
- Añade métodos para añadir y listar los trabajadores.
    - `public function anyadirTrabajador(Trabajador $t)`
    - `public function listarTrabajadoresHtml() : string` -> utiliza `Trabajador::toHtml(Persona $p)`
- Añade un método para obtener el coste total en nóminas.
    - `public function getCosteNominas(): float` -> recorre los trabajadores e invoca al método `calcularSueldo()`.

Actividad 315

`315EmpresaI.php`: Copia las clases del ejercicio anterior y modifícalas.

a) Crea un interfaz JSerializable, de manera que ofrezca los métodos:

- `toJSON(): string` → utiliza la función json_encode(mixed). Ten en cuenta que como tenemos las propiedades de los objetos privados, debes recorrer las propiedades y colocarlas en un mapa. Por ejemplo:

```php
<?php
public function toJSON(): string {
    foreach ($this as $clave => $valor) {
        $mapa->$clave = $valor;
    }
    return json_encode($mapa);
}
?>
```

- `toSerialize(): string` → utiliza la función serialize(mixed)

b) Modifica todas las clases que no son abstractas para que implementen el interfaz creado.

3. proyecto Videoclub

En los siguientes ejercicios vamos a simular un pequeño proyecto de un Videoclub (basado en la propuesta que hace el tutorial de desarrolloweb.com), el cual vamos a realizar mediante un desarrollo incremental y siguiendo la práctica de programación en parejas (*pair programming*).

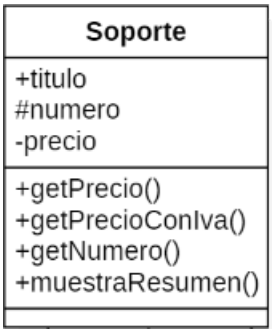
Antes de nada, crea un repositorio privado en GitHub y sube el proyecto actual de Videoclub. Una vez creado, invita a tu compañero al repositorio como colaborador.

Inicializa en local tu repostorio de git, mediante `git init`.
Añade y sube los cambios a tu repositorio, mediante `git add .` y luego `git commit -m 'Inicializando proyecto'`.
Conecta tu repositorio con GitHub y sube los cambios (mira la instrucciones de GitHub: comandos `git remote` y `git push`).
Tu compañero deberá descargar el proyecto con sus credenciales.

> **proyecto no real**
>
> El siguiente proyecto está pensado desde un punto de vista formativo. Algunas de las decisiones que se toman no se deben usar (como hacer `echo` dentro de las clases) o probar el código comparando el resultado en el navegador.

Cada clase debe ir en un archivo php separado. Para facilitar su implementación, se muestra la estructura UML del modelo y un fragmento de código para probar las clases:

Actividad 321



Crea una clase para almacenar soportes (`Soporte.php`). Esta clase será la clase padre de los diferentes soportes con los que trabaje nuestro videoclub (cintas de vídeo, videojuegos, etc...):

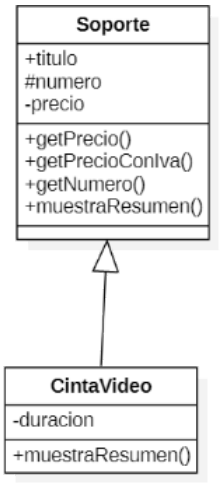
- Crea el constructor que inicialice sus propiedades. Fíjate que la clase no tiene métodos *setters*.
- Definir una constante mediante un propiedad privada y estática denominada `IVA` con un valor del 21%
- Crear un archivo (`inicio.php`) para usar las clases y copia el siguiente fragmento:

```php
<?php
    include ("Soporte.php");

    $soporte1 = new Soporte("Tenet", 22, 3);
    echo "<strong>" . $soporte1->titulo . "</strong>";
    echo "<br>Precio: " . $soporte1->getPrecio() . " €";
    echo "<br>Precio con IVA: " . $soporte1->getPrecioConIVA() . " €";

    echo $soporte1->muestraResumen();
```

En el navegador:

```
Tenet
Precio: 3 €
Precio con IVA: 3.48 €

Tenet
3 € (IVA no incluido)
```

Actividad 322



Crea la clase `CintaVideo` la cual hereda de `Soporte`. Añade el atributo `duracion` y sobreescribe tanto el contructor como el método `muestraResumen` (desde `CintaVideo` deberás llamar al método `muestraResumen` del padre).

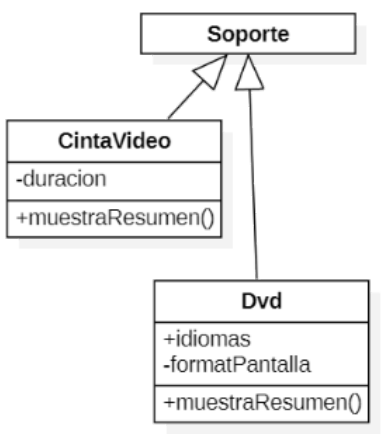
Añade a `inicio.php` el código para probar la clase:

```php
<?php
    include "CintaVideo.php";

    $miCinta = new CintaVideo("Los cazafantasmas", 23, 3.5, 107);
    echo "<strong>" . $miCinta->titulo . "</strong>";
    echo "<br>Precio: " . $miCinta->getPrecio() . " €";
    echo "<br>Precio con IVA: " . $miCinta->getPrecioConIva() . " €";

    echo $miCinta->muestraResumen();
```

En el navegador:

Los cazafantasmas
Precio: 3.5 euros
Precio IVA incluido: 4.06 euros
Película en VHS:
Los cazafantasmas
3.5 € (IVA no incluido)
Duración: 107 minutos

Actividad 323



Crea la clase `Dvd` la cual hereda de `Soporte`. Añade los atributos `idiomas` y `formatoPantalla`. A continuación sobreescribe tanto el contructor como el método `muestraResumen`.

Añade a `inicio.php` el código para probar la clase:

```php
<?php
include "Dvd.php";

$miDvd = new Dvd("Origen", 24, 15, "es,en,fr", "16:9");
echo "<strong>" . $miDvd->titulo . "</strong>";
echo "<br>Precio: " . $miDvd->getPrecio() . " €";
echo "<br>Precio con IVA: " . $miDvd->getPrecioConIva() . " €";

echo $miDvd->muestraResumen();
```

En el navegador:

Origen
Precio: 15 euros
Precio IVA incluido: 17.4 euros
Película en DVD:
Origen
15 € (IVA no incluido)
Idiomas:es,en,fr
Formato Pantalla:16:9

Actividad 324

Crea la clase `Juego` la cual hereda de `Soporte`. Añade los atributos `consola`, `minNumJugadores` y `maxNumJugadores`. A continuación añade el método `muestraJugadoresPosibles`, el cual debe mostrar *Para un jugador*, Para X jugadores o De X a Y jugadores dependiendo de los valores de las atributos creados. Finalmente, sobreescribe tanto el

contructor como el método `muestraResumen`.

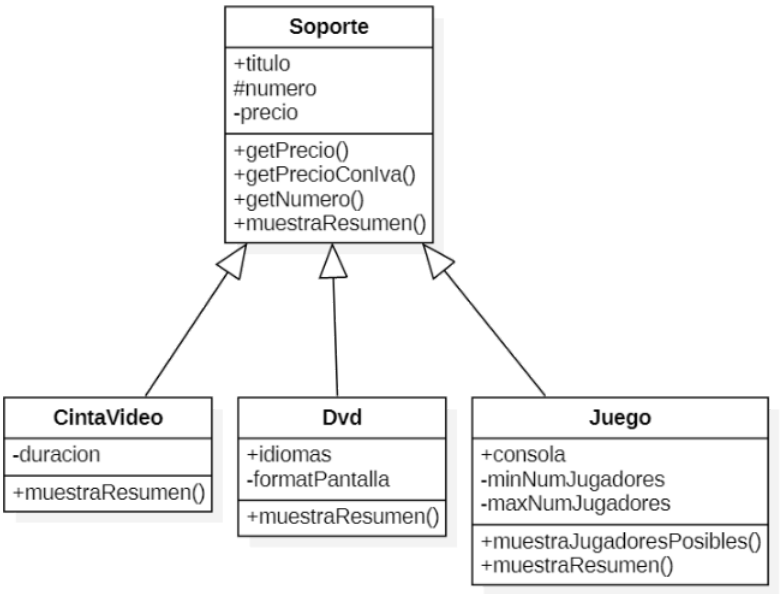
Añade a `inicio.php` el código para probar la clase:

```php
<?php
    include "Juego.php";

    $miJuego = new Juego("The Last of Us Part II", 26, 49.99, "PS4", 1, 1);
    echo "<strong>" . $miJuego->titulo . "</strong>";
    echo "<br>Precio: " . $miJuego->getPrecio() . " €";
    echo "<br>Precio con IVA: " . $miJuego->getPrecioConIva() . " €";

    echo $miJuego->muestraResumen();
```

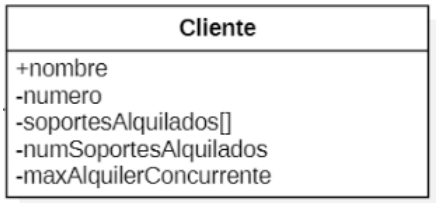
En el navegador:

The Last of Us Part II
Precio: 49.99 euros
Precio IVA incluido: 57.9884 euros
Juego para: PS4
The Last of Us Part II
49.99 € (IVA no incluido)
Para un jugador

Llegados a este punto, nuestro modelo es similar al siguiente diagrama:



Actividad 325



Crear la clase `Cliente`. El constructor recibirá el `nombre`, `numero` y `maxAlquilerConcurrente`, este último pudiendo ser opcional y tomando como valor por defecto 3. Tras ello, añade *getter/setter* únicamente a `numero`, y un *getter* a `numSoportesAlquilados` (este campo va a almacenar un contador del total de alquileres que ha realizado). El array de soportes

alquilados contedrá clases que hereden de `Soporte`. Finalmente, añade el método `muestraResumen` que muestre el nombre y la cantidad de alquileres (tamaño del array `soportesAlquilados`).

Actividad 326

Dentro de `Cliente`, añade las siguiente operaciones:

- `tieneAlquilado(Soporte $s): bool` → Recorre el array de soportes y comprueba si está el soporte
- `alquilar(Soporte $s): bool` -→ Debe comprobar si el soporte está alquilado y si no ha superado el cupo de alquileres. Al alquilar, incrementará el `numSoportesAlquilados` y almacenará el soporte en el array. Para cada caso debe mostrar un mensaje informando de lo ocurrido.

Actividad 327

Seguimos con `Cliente` para añadir las operaciones:

- `devolver(int $numSoporte): bool` → Debe comprobar que el soporte estaba alquilado y actualizar la cantidad de soportes alquilados. Para cada caso debe mostrar un mensaje informando de lo ocurrido
- `listarAlquileres(): void` → Informa de cuantos alquileres tiene el cliente y los muestra.

Crea el archivo `inicio2.php` con el siguiente código fuente para probar la clase:

```php
<?php
include_once "CintaVideo.php";
include_once "Dvd.php";
include_once "Juego.php";
include_once "Cliente.php";

//instanciamos un par de objetos cliente
$cliente1 = new Cliente("Bruce Wayne", 23);
$cliente2 = new Cliente("Clark Kent", 33);

//mostramos el número de cada cliente creado
echo "<br>El identificador del cliente 1 es: " . $cliente1->getNumero();
echo "<br>El identificador del cliente 2 es: " . $cliente2->getNumero();

//instancio algunos soportes
$soporte1 = new CintaVideo("Los cazafantasmas", 23, 3.5, 107);
$soporte2 = new Juego("The Last of Us Part II", 26, 49.99, "PS4", 1, 1);
$soporte3 = new Dvd("Origen", 24, 15, "es,en,fr", "16:9");
$soporte4 = new Dvd("El Imperio Contraataca", 4, 3, "es,en","16:9");

//alquilo algunos soportes
$cliente1->alquilar($soporte1);
$cliente1->alquilar($soporte2);
$cliente1->alquilar($soporte3);

```

```php
//voy a intentar alquilar de nuevo un soporte que ya tiene alquilado
$cliente1->alquilar($soporte1);
//el cliente tiene 3 soportes en alquiler como máximo
//este soporte no lo va a poder alquilar
$cliente1->alquilar($soporte4);
//este soporte no lo tiene alquilado
$cliente1->devolver(4);
//devuelvo un soporte que sí que tiene alquilado
$cliente1->devolver(2);
//alquilo otro soporte
$cliente1->alquilar($soporte4);
//listo los elementos alquilados
$cliente1->listaAlquileres();
//este cliente no tiene alquileres
$cliente2->devolver(2);
```

En el navegador:

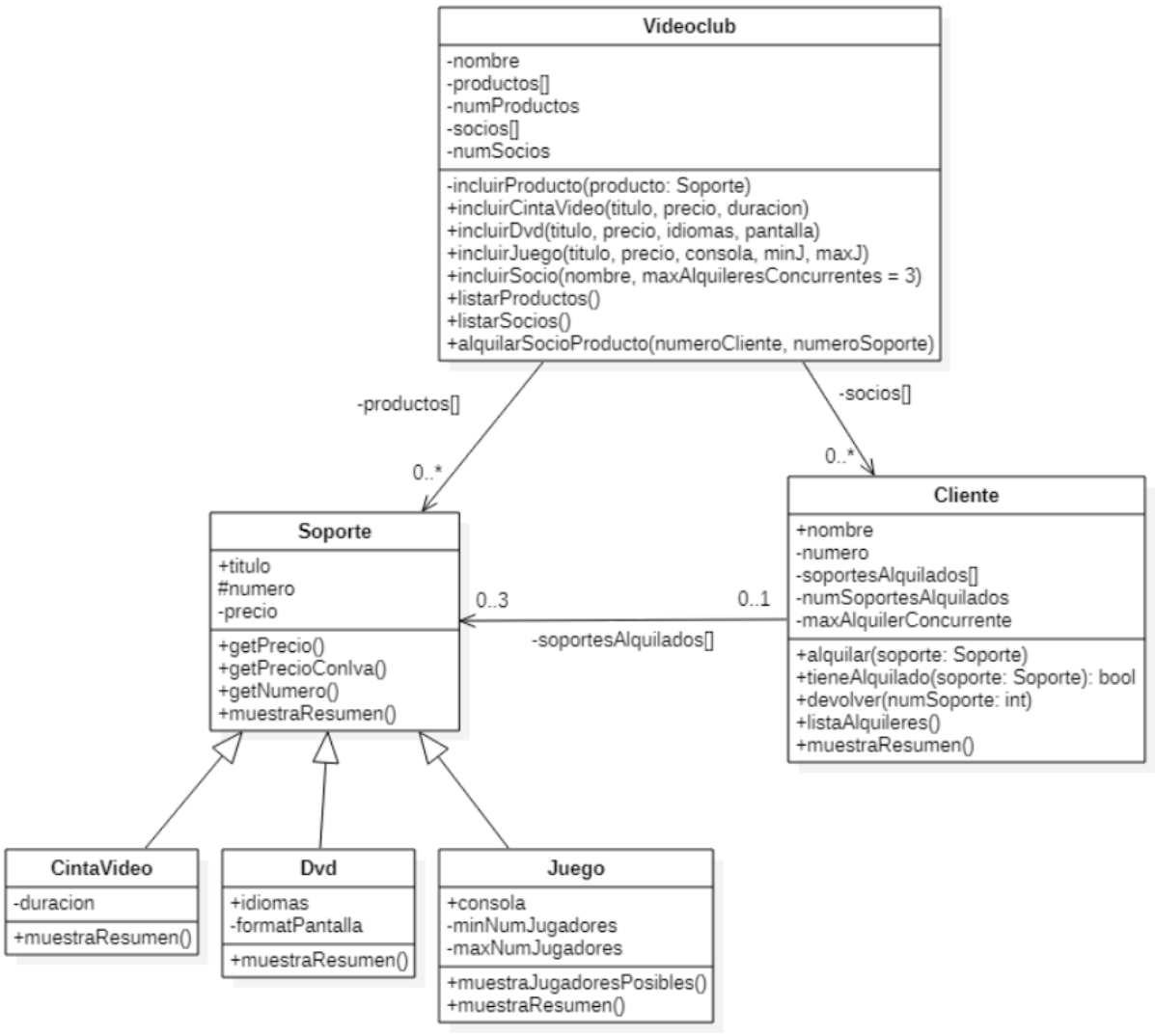
```
El identificador del cliente 1 es: 23
El identificador del cliente 2 es: 33
Alquilado soporte a: Bruce Wayne


Película en VHS:
Los cazafantasmas
3.5 € (IVA no incluido)
Duración: 107 minutos


Alquilado soporte a: Bruce Wayne


Juego para: PS4
The Last of Us Part II
49.99 € (IVA no incluido)
Para un jugador


Alquilado soporte a: Bruce Wayne


Película en DVD:
Origen
15 € (IVA no incluido)
Idiomas:es,en,fr
Formato Pantalla:16:9


El cliente ya tiene alquilado el soporte Los cazafantasmas


Este cliente tiene 3 elementos alquilados. No puede alquilar más en este videoclub hasta que no devuelva algo


No se ha podido encontrar el soporte en los alquileres de este cliente


No se ha podido encontrar el soporte en los alquileres de este cliente


Este cliente tiene 3 elementos alquilados. No puede alquilar más en este videoclub hasta que no devuelva algo


El cliente tiene 3 soportes alquilados


Película en VHS:
Los cazafantasmas
3.5 € (IVA no incluido)
Duración: 107 minutos


Juego para: PS4
The Last of Us Part II
49.99 € (IVA no incluido)
Para un jugador


Película en DVD:
Origen
15 € (IVA no incluido)
Idiomas:es,en,fr
Formato Pantalla:16:9


Este cliente no tiene alquilado ningún elemento
```

Actividad 328

Llegado a este punto, vamos a relacionar los clientes y los soportes mediante la clase `Videoclub`. Así pues crea la clase que representa el gráfico, teniendo en cuenta que:

- `productos` es un array de `Soporte`.
- `socios` es una array de `Cliente`.
- Los métodos públicos de incluir algún soporte, crearán la clase y llamarán al método privado de `incluirProducto`, el cual es el encargado de introducirlo dentro del array.

El modelo completo quedará de la siguiente manera:



Y para probar el proyecto, dentro inicio3.php colocaremos:

```php
<?php
include_once "Videoclub.php"; // No incluimos nada más

$vc = new Videoclub("Severo 8A");

//voy a incluir unos cuantos soportes de prueba
$vc->incluirJuego("God of War", 19.99, "PS4", 1, 1);
$vc->incluirJuego("The Last of Us Part II", 49.99, "PS4", 1, 1);
$vc->incluirDvd("Torrente", 4.5, "es","16:9");
$vc->incluirDvd("Origen", 4.5, "es,en,fr", "16:9");
$vc->incluirDvd("El Imperio Contraataca", 3, "es,en","16:9");
```

```php
$vc->incluirCintaVideo("Los cazafantasmas", 3.5, 107);
$vc->incluirCintaVideo("El nombre de la Rosa", 1.5, 140);

//listo los productos
$vc->listarProductos();

//voy a crear algunos socios
$vc->incluirSocio("Amancio Ortega");
$vc->incluirSocio("Pablo Picasso", 2);

$vc->alquilaSocioProducto(1,2);
$vc->alquilaSocioProducto(1,3);
//alquilo otra vez el soporte 2 al socio 1.
// no debe dejarme porque ya lo tiene alquilado
$vc->alquilaSocioProducto(1,2);
//alquilo el soporte 6 al socio 1.
//no se puede porque el socio 1 tiene 2 alquileres como máximo
$vc->alquilaSocioProducto(1,6);

//listo los socios
$vc->listarSocios();
```

En el navegador:

```
Incluido soporte 0
Incluido soporte 1
Incluido soporte 2
Incluido soporte 3
Incluido soporte 4
Incluido soporte 5
Incluido soporte 6


Listado de los 7 productos disponibles:
1.- Juego para: PS4
God of War
19.99 € (IVA no incluido)
Para un jugador
2.- Juego para: PS4
The Last of Us Part II
49.99 € (IVA no incluido)
Para un jugador
3.- Película en DVD:
Torrente
4.5 € (IVA no incluido)
Idiomas:es
Formato Pantalla:16:9
4.- Película en DVD:
Origen
4.5 € (IVA no incluido)
Idiomas:es,en,fr
Formato Pantalla:16:9
5.- Película en DVD:
El Imperio Contraataca
3 € (IVA no incluido)
Idiomas:es,en
Formato Pantalla:16:9
6.- Película en VHS:
Los cazafantasmas
3.5 € (IVA no incluido)
Duración: 107 minutos
7.- Película en VHS:
El nombre de la Rosa
1.5 € (IVA no incluido)
Duración: 140 minutos


Incluido socio 0
Incluido socio 1


Alquilado soporte a: Pablo Picasso

Película en DVD:
Torrente
4.5 € (IVA no incluido)
Idiomas:es
Formato Pantalla:16:9

 Alquilado soporte a: Pablo Picasso


Película en DVD:
Origen
4.5 € (IVA no incluido)
Idiomas:es,en,fr
Formato Pantalla:16:9


El cliente ya tiene alquilado el soporte Torrente


Este cliente tiene 2 elementos alquilados. No puede alquilar más en este videoclub hasta que no devuelva algo

Listado de 2 socios del videoclub:
1.- Cliente 0: Amancio Ortega
Alquileres actuales: 0
2.- Cliente 1: Pablo Picasso
Alquileres actuales: 2
```

Actividad 329

Transforma `Soporte` a una clase abstracta y comprueba que todo sigue funcionando. ¿Qué conseguimos al hacerla abstracta?

Actividad 330

Crea un interfaz `Resumible`, de manera que las clases que lo implementen deben ofrecer el método `muestraResumen()`. Modifica la clase `Soporte` y haz que implemente el interfaz. ¿Hace falta que también lo implementen los hijos?

4. proyecto Videoclub 2.0

Antes de comenzar con la segunda parte del videoclub, crea una etiqueta mediante `git tag` con el nombre `v0.329` y sube los cambios a GitHub.

Actividad 331

Modifica las operaciones de alquilar, tanto en `Cliente` como en `Videoclub`, para dar soporte al encadenamiento de métodos. Posteriormente, modifica el código de prueba para utilizar esta técnica.

Actividad 332

Haciendo uso de *namespaces*:

- Coloca todas las clases/interfaces en `Dwes\ProyectoVideoclub`.
- Cada clase debe hacer `include_once` de los recursos que emplea.
- Coloca el/los archivos de prueba en el raíz (sin espacio de nombres).
- Desde el archivo de pruebas, utiliza `use` para poder realizar accesos sin cualificar.
- Etiqueta los cambios como `v0.331`.

Actividad 333

Reorganiza las carpeta tal como hemos visto en los apuntes: `app`, `test` y `vendor`.

- Crea un fichero `autoload.php` para registrar la ruta donde encontrar las clases
- Modifica todo el código necesario, incluyendo `autoload.php` donde sea necesario y borrando los *includes* previos.

Actividad 334

A continuación vamos a crear un conjunto de excepciones de aplicación. Estas excepciones son simples, no necesitan sobreescribir ningún método. Así pues, crea la excepción de aplicación `VideoclubException` en el *namespace* `Dwes\ProyectoVideoclub\Util`. Posteriormente crea los siguientes hijos (deben heredar de `VideoclubException`), cada uno en su propio archivo:

- `SoporteYaAlquiladoException`.
- `CupoSuperadoException`.
- `SoporteNoEncontradoException`.
- `ClienteNoEncontradoException`.

Actividad 335

En `Cliente`, modifica los métodos `alquilar` y `devolver`, para que hagan uso de las nuevas excepciones (lanzándolas cuando sea necesario) y funcionen como métodos encadenados. Destacar que estos métodos, no se capturar estás excepciones, sólo se lanzan. En `Videoclub`, modifica `alquilarSocioPelicula` para capturar todas las excepciones que ahora lanza `Cliente` e informar al usuario en consecuencia.

Actividad 336

Vamos a modificar el proyecto para que el videoclub sepa qué productos están o no alquilados:

- En `Soporte`, crea una propiedad pública cuyo nombre sea `alquilado` que inicialmente estará a `false`. Cuando se alquile, se pondrá a `true`. Al devolver, la volveremos a poner a `false`.
- En `Videoclub`, crea dos nuevas propiedades y sus getters:
    - `numProductosAlquilados`
    - `numTotalAlquileres`

Actividad 337

Crea un nuevo método en `Videoclub` llamado `alquilarSocioProductos(int numSocio, array numerosProductos)`, el cual debe recibir un array con los productos a alquilar.

Antes de alquilarlos, debe comprobar que todos los soportes estén disponibles, de manera que si uno no lo está, no se le alquile ninguno.

Actividad 338

Crea dos nuevos métodos en `Videoclub`, y mediante la definición, deduce qué deben realizar:

- `devolverSocioProducto(int numSocio, int numeroProducto)`
- `devolverSocioProductos(int numSocio, array numerosProductos)`

Deben soportar el encadenamiento de métodos. Recuerda actualizar la propiedad `alquilado` de los diferentes soportes.

Cuando hayas realizado todos los ejercicios, crea una etiqueta mediante `git tag` con el nombre `v0.337` y sube los cambios a GitHub.