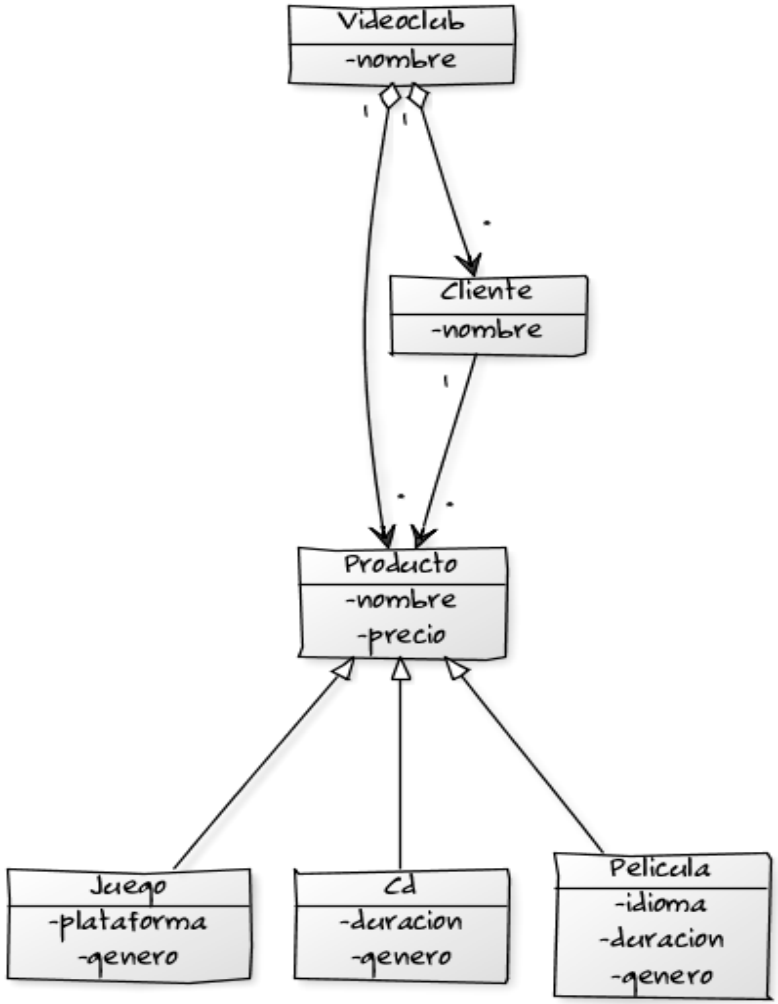
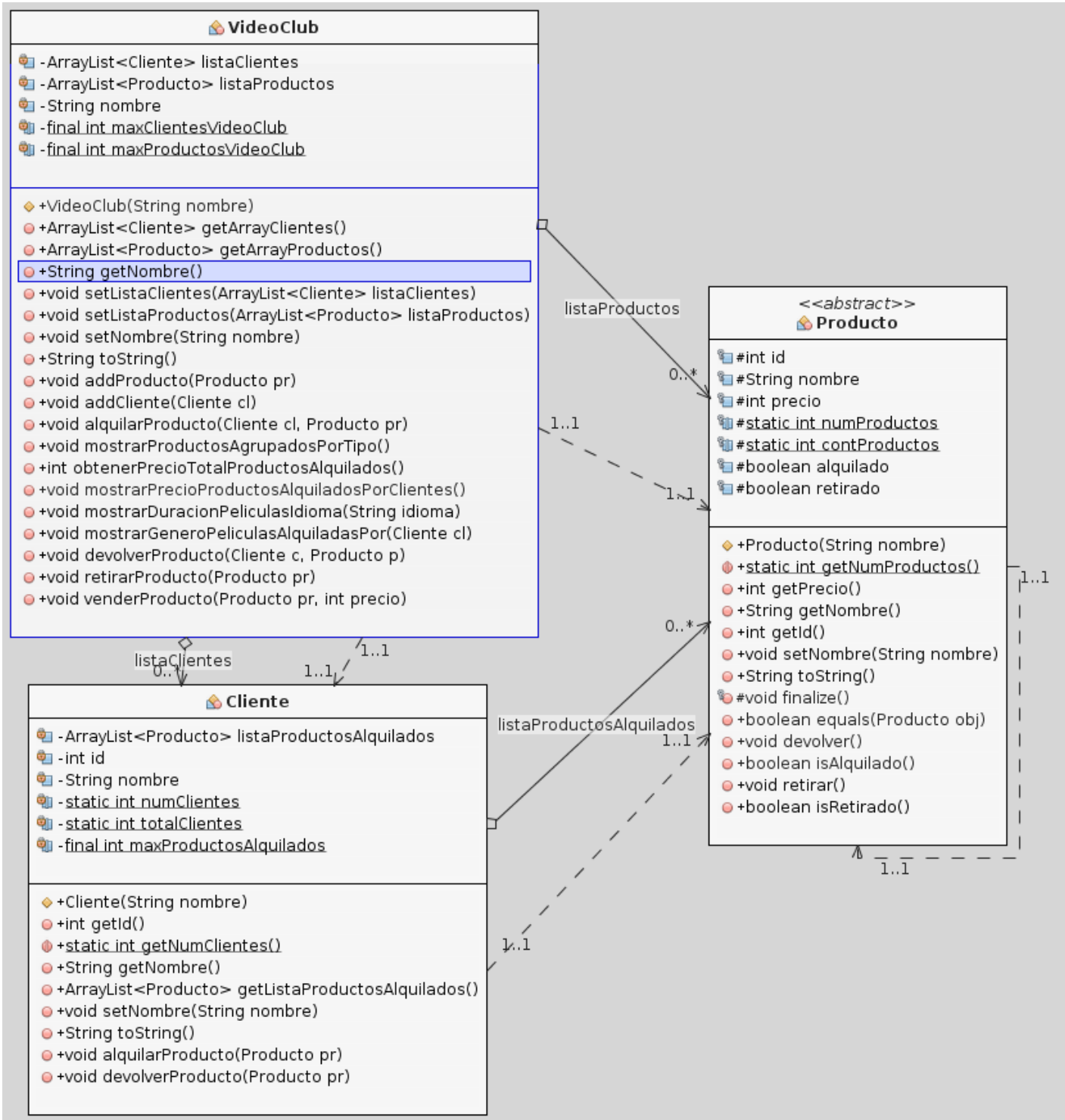
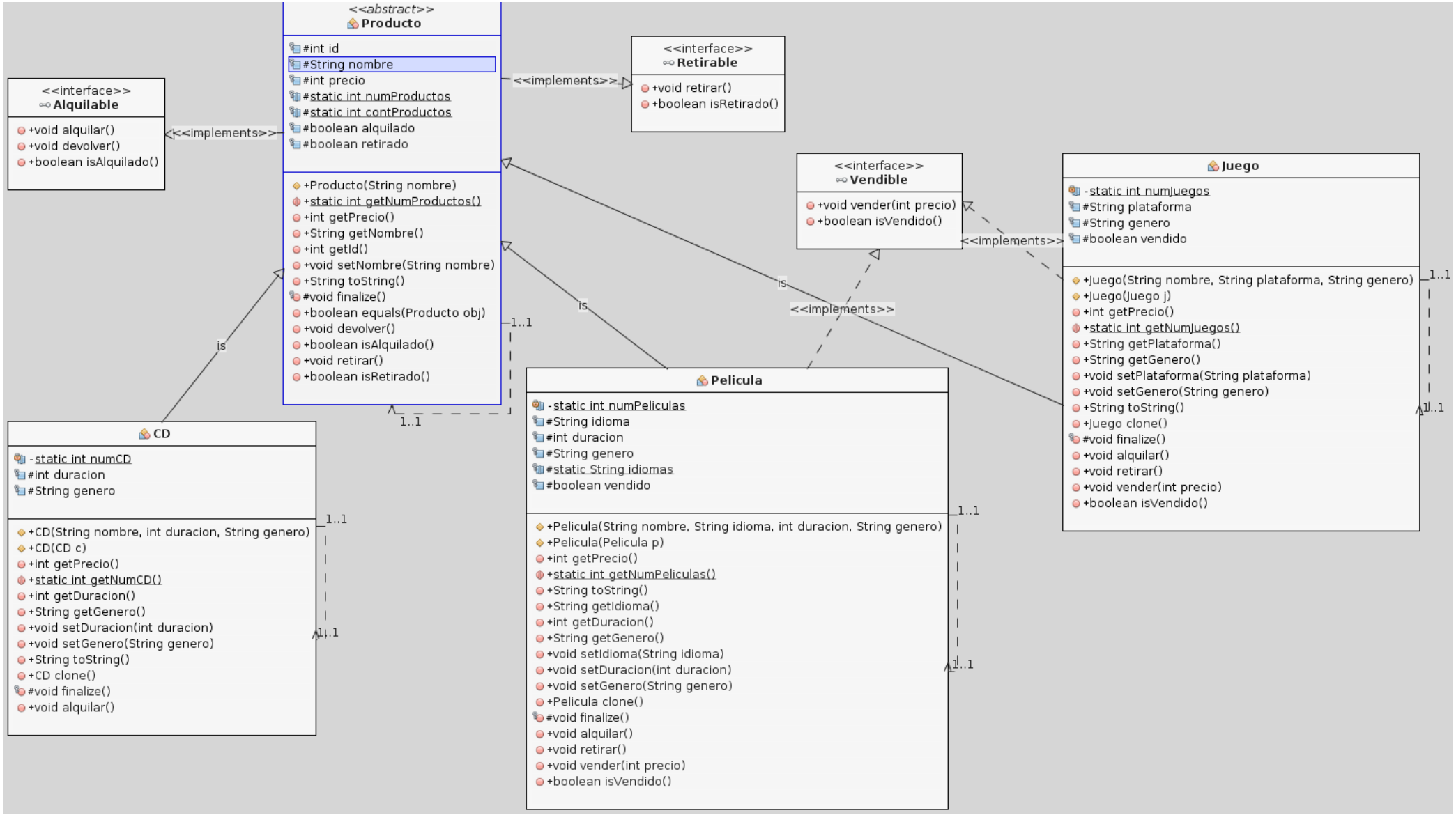
VIDEOCLUB CON INTERFACES

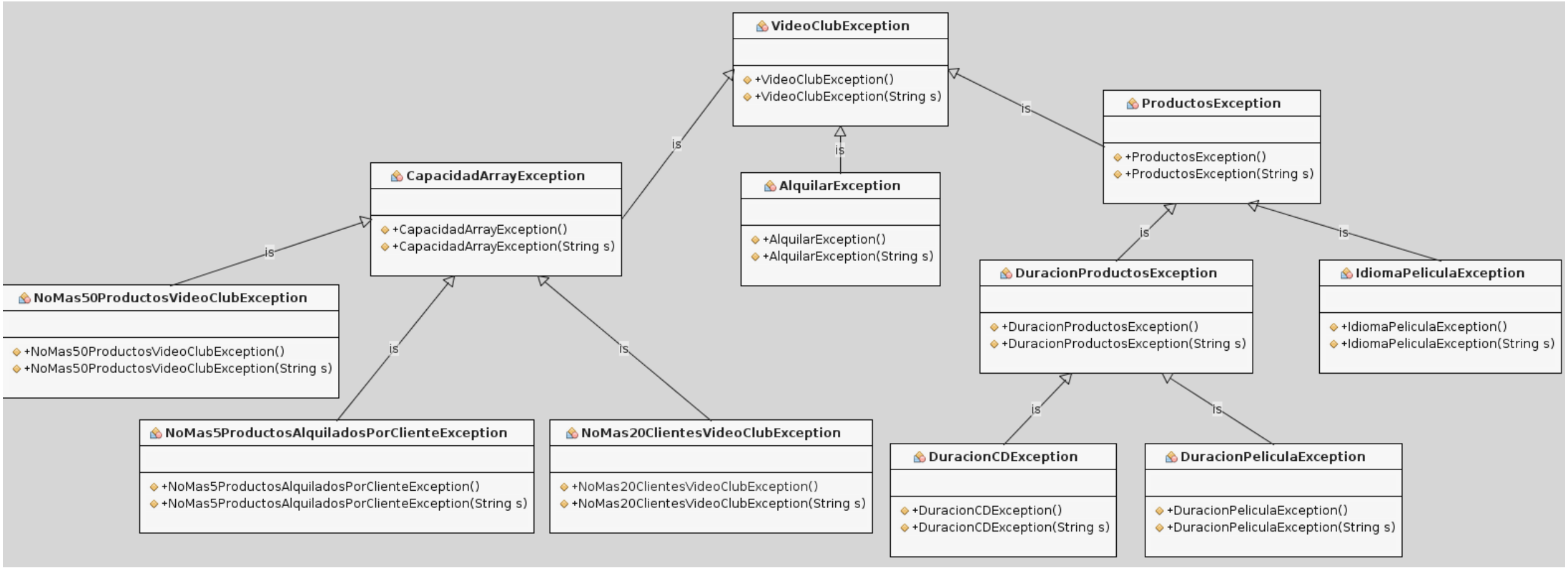
Crear objetos para gestionar un videoclub. El diagrama de clases en UML para representar gráficamente las clases que vamos a utilizar para gestionar los diferentes objetos es el siguiente.




VideoClub
-ArrayList<Cliente> listaClientes
-ArrayList<Producto> listaProductos
-String nombre
-final int maxClientesVideoClub
-final int maxProductosVideoClub
+VideoClub(String nombre)
+ArrayList<Cliente> getArrayClientes()
+ArrayList<Producto> getArrayProductos()
+String getNombre()
+void setListaClientes(ArrayList<Cliente> listaClientes)
+void setListaProductos(ArrayList<Producto> listaProductos)
+void setNombre(String nombre)
+String toString()
+void addProducto(Producto pr)
+void addCliente(Cliente cl)
+void alquilarProducto(Cliente cl, Producto pr)
+void mostrarProductosAgrupadosPorTipo()
+int obtenerPrecioTotalProductosAlquilados()
+void mostrarPrecioProductosAlquiladosPorClientes()
+void mostrarDuracionPeliculasIdioma(String idioma)
+void mostrarGeneroPeliculasAlquiladasPor(Cliente cl)
+void devolverProducto(Cliente c, Producto p)
+void retirarProducto(Producto pr)
+void venderProducto(Producto pr, int precio)
listaProductos
0..*
1..1
1..1
<<abstract>>
Producto
#int id
#String nombre
#int precio
#static int numProductos
#static int contProductos
#boolean alquilado
#boolean retirado
+Producto(String nombre)
+static int getNumProductos()
+int getPrecio()
+String getNombre()
+int getId()
+void setNombre(String nombre)
+String toString()
#void finalize()
+boolean equals(Producto obj)
+void devolver()
+boolean isAlquilado()
+void retirar()
+boolean isRetirado()
1..1
listaClientes
0..
1..1
1..1
0..*
listaProductosAlquilados
1..1
Cliente
-ArrayList<Producto> listaProductosAlquilados
-int id
-String nombre
-static int numClientes
-static int totalClientes
-final int maxProductosAlquilados
+Cliente(String nombre)
+int getId()
+static int getNumClientes()
+String getNombre()
+ArrayList<Producto> getListaProductosAlquilados()
+void setNombre(String nombre)
+String toString()
+void alquilarProducto(Producto pr)
+void devolverProducto(Producto pr)
1..1
1..1


<<abstract>>
Producto
#int id
#String nombre
#int precio
#static int numProductos
#static int contProductos
#boolean alquilado
#boolean retirado
+Producto(String nombre)
+static int getNumProductos()
+int getPrecio()
+String getNombre()
+int getId()
+void setNombre(String nombre)
+String toString()
#void finalize()
+boolean equals(Producto obj)
+void devolver()
+boolean isAlquilado()
+void retirar()
+boolean isRetirado()
<<interface>>
Alquilable
+void alquilar()
+void devolver()
+boolean isAlquilado()
<<implements>>
<<interface>>
Retirable
+void retirar()
+boolean isRetirado()
<<implements>>
<<interface>>
Vendible
+void vender(int precio)
+boolean isVendido()
<<implements>>
Juego
-static int numJuegos
#String plataforma
#String genero
#boolean vendido
+Juego(String nombre, String plataforma, String genero)
+Juego(Juego j)
+int getPrecio()
+static int getNumJuegos()
+String getPlataforma()
+String getGenero()
+void setPlataforma(String plataforma)
+void setGenero(String genero)
+String toString()
+Juego clone()
#void finalize()
+void alquilar()
+void retirar()
+void vender(int precio)
+boolean isVendido()
CD
-static int numCD
#int duracion
#String genero
+CD(String nombre, int duracion, String genero)
+CD(CD c)
+int getPrecio()
+static int getNumCD()
+int getDuracion()
+String getGenero()
+void setDuracion(int duracion)
+void setGenero(String genero)
+String toString()
+CD clone()
#void finalize()
+void alquilar()
Pelicula
-static int numPeliculas
#String idioma
#int duracion
#String genero
#static String idiomas
#boolean vendido
+Pelicula(String nombre, String idioma, int duracion, String genero)
+Pelicula(Pelicula p)
+int getPrecio()
+static int getNumPeliculas()
+String toString()
+String getIdioma()
+int getDuracion()
+String getGenero()
+void setIdioma(String idioma)
+void setDuracion(int duracion)
+void setGenero(String genero)
+Pelicula clone()
#void finalize()
+void alquilar()
+void retirar()
+void vender(int precio)
+boolean isVendido()
is
is
is
<<implements>>
1..1
1..1
1..1
1..1
1..1
1..1
1..1
1..1

Jerarquía de Excepciones:



El proyecto se debe llamar: proyecto_VideoClub_Herencia.
Las clases: Principal y Pedir, están el paquete: paquete_Ejecutable.
Las clases: Cliente, Producto y VideoClub están en el paquete paquete_Clases
Las clases: Pelicula, CD y Juego, están en el paquete: paquete_Subclases.
Las subclases dependientes de IllegalArgumentExcpetion (Excecpciones) están en el paquete: paquete_Excepciones.
Las interfaces: Alquilable, Vendible y Retirable están en el paquete: paquete_Interfaces

-La clase **VideoClubException,** posee dos constructores sobrecargados, uno con un parámetro de entrada para un mensaje y llama al constructor de su superclase con ese mensaje, y otro con ningún argumento, que llama al constructor de su superclase o se deja vacío.

-La clase **ProductosException**, posee dos constructores sobrecargados, uno con un parámetro de entrada para un mensaje y llama al constructor de su superclase con ese mensaje, y otro con ningún argumento, que llama al constructor de su superclase o se deja vacío.

-La clase **DuracionProductosException**, posee dos constructores sobrecargados, uno con un parámetro de entrada para un mensaje y llama al constructor de su superclase con ese mensaje, y otro con ningún argumento, que llama al constructor de su superclase o se deja vacío.

-La clase **DuracionPeliculaException**, posee dos constructores sobrecargados, uno con un parámetro de entrada para un mensaje y llama al constructor de su superclase con ese mensaje, y otro con ningún argumento, que llama al constructor de su superclase con el mensaje cuyo valor es: “Error, la duración de una Pelicula no puede ser negativa o 0”.

-La clase **DuracionCDException**, posee dos constructores sobrecargados, uno con un parámetro de entrada para un mensaje y llama al constructor de su superclase con ese mensaje, y otro con ningún argumento, que llama al constructor de su superclase con el mensaje cuyo valor es: “Error, la duración de un CD no puede ser negativa o 0”.

- La clase **IdiomaPeliculaException**, posee dos constructores sobrecargados, uno con un parámetro de entrada para un mensaje y llama al constructor de su superclase con ese mensaje, y otro con ningún argumento, que llama al constructor de su superclase con el mensaje cuyo valor es: “Error, el idioma de una Película no está definido”.

-La clase **CapacidadArrayException**, posee dos constructores sobrecargados, uno con un parámetro de entrada para un mensaje y llama al constructor de su superclase con ese mensaje, y otro con ningún argumento, que llama al constructor de su superclase o se deja vacío.

- La clase **NoMas5ProductosAlquiladosPorClienteException**, posee dos constructores sobrecargados, uno con un parámetro de entrada para un mensaje y llama al constructor de su superclase con ese mensaje, y otro con ningún argumento, que llama al constructor de su superclase con el mensaje cuyo valor es: “Error, un cliente nunca puede alquilar más de 5 productos”.

- La clase **NoMas50ProductosVideoClubException**, posee dos constructores sobrecargados, uno con un parámetro de entrada para un mensaje y llama al constructor de su superclase con ese mensaje, y otro con ningún argumento, que llama al constructor de su superclase con el mensaje cuyo valor es: “Error, el Video Club no puede tener más de 50 productos”.

- La clase **NoMas20ClientesVideoClubException**, posee dos constructores sobrecargados, uno con un parámetro de entrada para un mensaje y llama al constructor de su superclase con ese mensaje, y otro con ningún argumento, que llama al constructor de su superclase con el mensaje cuyo valor es: “Error, el Video Club no puede tener más de 20 clientes”.

- La clase **AlquilarException**, posee dos constructores, uno con un parámetro de entrada para un mensaje y llama al constructor de su superclase con ese mensaje y otro con ningún argumento, que llama al constructor de su superclase con el mensaje cuyo valor es: “Error, al intentar alquilar un producto”.

-La interfaz **Alquilable**, tiene como métodos:

```
public void alquilar() throws AlquilarException;
public void devolver();
public boolean isAlquilado();
```

-La interfaz **Vendible**, tiene como métodos:

```
public void vender(int precio);
public boolean isVendido();
```

-La interfaz **Retirable**, tiene como métodos:

```
public void retirar();
public boolean isRetirado();
```

-La clase **Producto**: clase abstracta que representa un producto del videoclub, es decir, no podemos instanciar objetos **Producto**. E implementa las interfaces Alquilable y Retirable.
Cuenta con una serie de atributos que heredarán las subclases, **id**, **nombre, precio,** la propiedad estáticas: **contProductos** (id será el mismo que el de contProductos, que se incrementará conforme se instancie un producto nuevo). Se añaden los atributos boolean **alquilado** y **retirado** con valor inicial false
Todos los atributos son privados, pero públicos a las subclases.

El constructor: un objeto de tipo producto se puede crear dándole: nombre. No tiene precio.(Recuerda, se incrementan las propiedades estáticas, y el valor de id coincide con contProductos).

Métodos:
Método estático getContProductos(), nos indica el número de productos que se han creado (en las llamadas de las subclases al constructor)
Método abstracto getPrecio().
Y métodos públicos: getId(), getNombre() , setNombre.
Método: equals(), que admite como entrada un producto y lo compara con el actual, devolviendo true o false,
Método toString(), muestra: id, nombre y precio del producto actual.
Los métodos de la Interface **Alquilable**:

public void alquilar() throws AlquilarException; es abstracto, por lo que no se implementa en la clase Producto.
public void **devolver**(), la propiedad alquilado pasa a ser false.
Pubic boolean **isAlquilado**(), devuelve el valor de alquilado de ese producto.

Los métodos de la Interface **Retirable** :

public void **retirar**(), si el producto actual no está alquilado ni retirado, entonces, la propiedad retirado pasa a ser true, y la propiedad estática numProductos se decrementa. En el caso de que el producto esté alquilado, emite el mensaje de que el producto ya está alquilado y en el caso de que esté ya retirado, emite el mensaje indicándolo.
pubic boolean **isRetirado**(), devuelve el valor de retirado de ese producto.

-Las clases Pelicula, Cd, Juego: clases derivadas de la clase padre **Producto**. Representan los diferentes productos del que dispone el videoclub. Cada producto tiene asignado un precio diferente de alquiler. Cada tipo de **Producto** contará con un método **getPrecio()** que devolverá el precio en función del tipo de producto que sea. Las películas devolverán 2 euros, los cds de música devolverán 1 euro y los juegos 3 euros.
Cada tipo de **Producto** contará con un método **getPrecio()** que devolverá el precio en función del tipo de producto que sea, para Película 2 euros, para CD es 1 euro y para Juego serán 3 euros.

- La clase **Pelicula**, que además implementa la Interfaz Vendible, posee los atributos: numPeliculas, de tipo estático entero, que guarda el número de películas existentes, y se incrementa cada vez que se instancie una nueva película.
También: idioma, duración, género. y propiedad de tipo boolean vendido, con valor inicial a false.
Todos los atributos son privados, pero públicos ante posibles clases que puedan heredar de Pelicula, a excepción del atributo numPeliculas, que será privada.

El constructor está sobrecargado:

- Datos de entrada: nombre, idioma, duración y género.
- Dato de entrada: un objeto de tipo Pelicula.

La duración nunca pueden ser negativa o cero, si esto ocurre entonces se lanzará la excepción DuracionPeliculaException.
El idioma ha de ser uno de un conjunto, que será un array de carácteres estático con valor fijo: “ESPAÑOL”,”INGLÉS”,”FRANCÉS”,”CHINO” y “ALEMÁN”.

Se pide la gestión de la excepción IdiomaPeliculaException.

Métodos:
Método estático: getNumPeliculas().
Método getPrecio() -devuelve 2-, getIdioma(), setIdioma(),setIdioma(), getDuracion(), setDuracion(), getGenero(), setGenero()
Método: toString(), devuelve una cadena con los datos de la película actual.
Método sobrecargado equals(Pelicula), devuelve true o false, si la película actual es igual a una dada..

Los métodos de la Interface **Alquilable**:
alquilar() throws AlquilarException, se sobreescribe (en la clase Producto no se codificó). Si la película NO está retirada NI está alquilada NI vendida, entonces la propiedad alquilada pasa a ser true. Emite un mensaje indicando según el caso, si la película ya está alquilada, o está vendida o está retirada.

retirar() se sobreescribe el método retirar de la clase Producto, si la película NO está vendida, se ejecuta el método retirar de la clase Producto y se decrementa la propiedad numPeliculas. En el caso de que la película esté vendida, emite un mensaje: "No se puede retirar la película: "+nombrePelicula+" porque está vendida".

Los métodos de la Interface **Vendible** :
public void **vender**(int precio), la propiedad vendido pasa a ser true, si la película NO está alquilada ni retirada ni vendida, el precio cambia por el nuevo y las propiedades estáticas: numPeliculas y numProductos se decrementan en una unidad. En el caso de que la película esté alquilada, emite un mensaje :"La película: "+nombrePelicula+" no puede ser vendida porque está alquilada", y lo mismo en el caso de que la película esté vendida o retirada.

pubic boolean **isVendido**(), devuelve el valor de vendido de ese producto.

- La clase **CD**, posee además los atributos: duracion y genero junto con numCD, de tipo estático entero.
Todos los atributos son privados.

El constructor está sobrecargado:
- Datos de entrada: nombre, duración y género.
- Dato de entrada: un objeto de tipo CD.

Si la duración es negativa o cero, se lanzará la excepción DuracionCDException.

Métodos:
Método estático: getNumCD().
Método getPrecio() -devuelve 1-, getDuracion(), setDuracion(), getGenero(), setGenero()
Método: toString(), devuelve una cadena con los datos del CD actual.

Método sobrecargado equals(CD), devuelve true o false, si el CD actual es igual a uno dado..

- La clase **Juego**, que además implementa la Interfaz Vendible. Atributos: plataforma y genero, además numJuegos, de tipo estático entero, que guarda el número de Juegos existentes. .Se le añade la propiedad de tipo boolean vendido, con valores iniciales a false. Todos los atributos son privados

El constructor está sobrecargado:
- Datos de entrada: nombre, plataforma y genero.
- Dato de entrada: un objeto de tipo Juego.

Métodos:
Método estático: getNumJuegos().
Método getPrecio() -devuelve 3-, getPlataforma(), setPlataforma(), getGenero(), setGenero()
Método: toString(), devuelve una cadena con los datos del juego actual.
Método sobrecargado equals(Juego), devuelve true o false, si el Juego actual es igual a uno dado..

Los métodos de la Interface **Alquilable**:
alquilar() throws AlquilarException, se sobreescribe (en la clase Producto no se codificó). Si el juego NO está retirado NI está alquilado NI vendido, entonces la propiedad alquilada pasa a ser true. Emite un mensaje indicando según el caso, si el juego ya está alquilado, o está vendido o está retirado.

retirar() se sobreescribe el método retirar de la clase Producto, si el juego NO está vendido, se ejecuta el método retirar de la clase Producto y se decrementa la propiedad numJuegos. En el caso de que el juego esté vendido, emite un mensaje: "No se puede retirar el juego: "+nombreJuego+" porque está vendido".

Los métodos de la Interface **Vendible** :
public void **vender**(int precio), la propiedad vendido pasa a ser true, si el juego NO está alquilado ni retirado ni vendido, el precio cambia por el nuevo y las propiedades estáticas: numJuegos y numProductos se decrementan en una unidad. En el caso de que el juego esté alquilado, emite un mensaje :"El juego: "+nombreJuego+" no puede ser vendido porque está alquilado", y lo mismo en el caso de que el juego esté vendido o retirado.

pubic boolean **isVendido**(), devuelve el valor de vendido de ese producto.

-La clase **Cliente**: representa a un cliente.

Cuenta con una variable o propiedad que representa la colección (un array de Productos) de productos alquilados (listaProductosAlquilados).

La variable id (coincidirá con totalClientes, que se irá incrementando conforme se vayan creando clientes) y nombre.

Las propiedades son privadas, y cuenta con la propiedad estática totaClientes (aumenta en una unidad cuando se instancia un nuevo cliente) y la propiedad constante (final static maxProductosAlquilados con valor 5).

El constructor admitirá como parámetro: nombre del cliente, asignará como id el mismo valor que numClientes (previamente incrementado) y se creará el array para productosAlquilados.

Métodos:

Método estático getNumClientes()..

Métodos: getId(), getNombre(), setNombre(), getListaProductosAlquilados().

Método: **alquilarProducto(Producto pr)**, que lanza la excepción: NoMas5ProductosAlquiladosPorClienteException si el tamaño del array coincide con el valor maximo de Productos Alquilados por el cliente. Se ejecuta el método alquilar de producto y se añade al array de la lista de productos alquilados, en el caso de que se haya producido la excepción: AlquilarExcepcion, emite un mensaje: “Error “ más el mensaje de la excepción capturada.

Método: **devolverProducto(Producto pr)**, si la lista de productos alquilados contiene el producto pr, entonces se elimina el producto de la lista y se ejecuta el método devolver del producto. En caso contrario, emite el mensaje: “NO se puede devolver ese producto, debido a que "+nombreProducto+" no lo tiene alquilado”

-La clase **Videoclub**: es la clase principal de nuestra aplicación. Representa a un videoclub en el dominio del problema y cuenta con un ArrayList de clientes llamada: listaClientes y productos registrados llamada: listaProductos. Las propiedades son privadas, incluyendo el nombre. Se añaden las propiedades constantes (final static) maxClientesVideoClub cuyo valor es 20 y maxProductosVideoClub cuyo valor es 50.

Constructor: se considera nombre como dato de entrada.

Métodos:

Método: getNombre(), setNombre(), getListaClientes(), getListaProductos(), setListaClientes(), setListaProductos().

Método: toString().

Método: addProducto(Producto pr), añade un producto en el array de productos, controlad la posible excepción: NoMas50ProductosVideoClubException. Si el producto no está en la lista de productos y todavía no se ha alcanzado los 50 productos como máximo, entonces, añadir el producto en a lista, en caso contrario, lanzad la excepción. Y en el caso de que ya existe ese producto en la lista, emitid un mensaje: “"El producto "+nombreProducto+" ya está en la lista de productos del video club”.

Método: addCliente(Cliente c), añade un cliente en el array de clientes, controlad la posible excepción: NoMas20ClientesVideoClubException. Si el cliente no está en la lista de clientes y todavía no se ha alcanzado los 20 clientes como máximo, entonces, añadir el cliente en a lista, en caso contrario, lanzad la excepción. Y en el caso de que ya existe ese cliente en la lista, emitid un mensaje: “"El cliente "+nombreCliente+" ya está en la lista de clientes del video club”.

Método: alquilarProducto(Cliente c, producto pr), llamad al método del cliente para alquilar el producto.

Método: mostrarProductosAgrupadosPorTipo(), Mostrar la lista de productos, en primer lugar todas las películas, luego los CDs y en último lugar los juegos.

Método: obtenerPrecioTotalProductosAlquilados(), devuelve un int. Devuelve el precio total de todos los productos alquilados en el VideoClub.

Método: mostrarPrecioProductosAlquiladosPorClientes(), muestra por cada cliente del videoclub el precio total de todos los productos que ese cliente tiene

alquilados.

Método: mostrarDuracionPeliculasIdioma(String idioma), tal que dado un idioma, devuleve la duración de todas aquellas películas en la lista de productos con ese idioma.

Método: mostrarGeneroPeliculasAlquiladasPor(Cliente c). Muestra el género de todas las películas que el cliente c del videoClub tiene alquiladas.

Método: **public void devolverProducto(Cliente c, Producto p)**, de forma que el cliente c devuelva el producto p.

Método: **public void retirarProducto(Producto pr)**, tal que, se comprueba si ese producto está en la lista de productos del videoclub, si es así, ese producto se retira.

Método: **public void venderProducto(Producto pr, int precio)**, tal que, si ese producto existe en la lista de productos del videoclub, entonces se comprueba si es una película y se vende o es un juego y se vende, en caso de que el producto no exista, entones, se emite un mensaje de error.

Desde la clase Principal, en el método main(), ejecutar:

1. Crear un videoclub que se denomine 'ViedoMax'.
2. Crear 4 clientes: 'Francisco','Javier','Paco','Antonio'.
3. Intenta crear la Película:

   “La armada invencible”, de idioma “EEE”, duración -10 y de “Ficcion”.

   Muestra el número de productos existentes, al igual que número de películas.

   Crear 5 películas:{'El Señor de los Anillos', ’ESPAÑOL’, 200 minutos, “Ficcion”},{'The Hobbit',’ENGLISH’, 45’, “Ficcion”}, {'Star Wars III',’ENGLISH’,89’,”Ficcion”},{'El discurso del Rey', “ESPAÑOL”,78’, “COMEDIA”},{'Shrek',”FRANCAIS”, 123,”INFANTIL”}
4. Crear 2 CD: “U2 Boy in 1980” y “Queen Don’t stop me now”.
5. Crear 3 Juegos: “Simpsons Game” para “Play Station 3”, “Zelda” para “Wii” , “Mario Car” para “Nintendo DS”. Controla las posibles excepciones.
6. Registrar los 4 clientes en el videoclub (addCliente), y los 10 productos (addProductos). Comprueba las posibles excepciones.
7. El cliente: 'Francisco' alquila la película: 'El Señor de los Anillos' , 'The Hobbit' y el juego “Simpsons Game”.//desde el objeto videoclub. Controla las posibles excepciones
8. Muestra por pantalla los productos alquilados por el cliente: ‘Francisco’.
9. El cliente: ‘Francisco’ quiere alquilar la película: ‘The Hobbit’.
10. Mostar el número de películas.
11. Mostrar el número de clientes.
12. El cliente: 'Javier' alquila la película: 'Star Wars III'. //desde el objeto videoclub

13. Obtener la lista de clientes registrados.

14. Obtener la lista de productos registrados**.**

15. Obtener la lista de productos alquilados por el cliente: 'Francisco'.

16. El cliente: ‘Francisco’ quiere devolver la película: ‘El Señor de los Anillos’.

17. Obtener la lista de productos alquilados por el cliente: 'Francisco'.

18. Se pide retirar por defectuosa la película: ‘The Hobbit’

19. Se pide retirar por defectuosa la película: ‘Shrek’

20. La película ‘El Señor de los Anillos’ se vende por 4€.

21. Mostrar los productos agrupados por tipo: Pelicula, CD y Juego

22. Obtener el precio de todos los productos alquilados por cada cliente

23. Obtener el precio total de todos los productos alquilados

24. Se desea saber la duración de las películas del videoClub cuyo idioma sea uno introducido por teclado

25. Indicad qué género poseen las películas que tiene alquiladas el cliente c1.