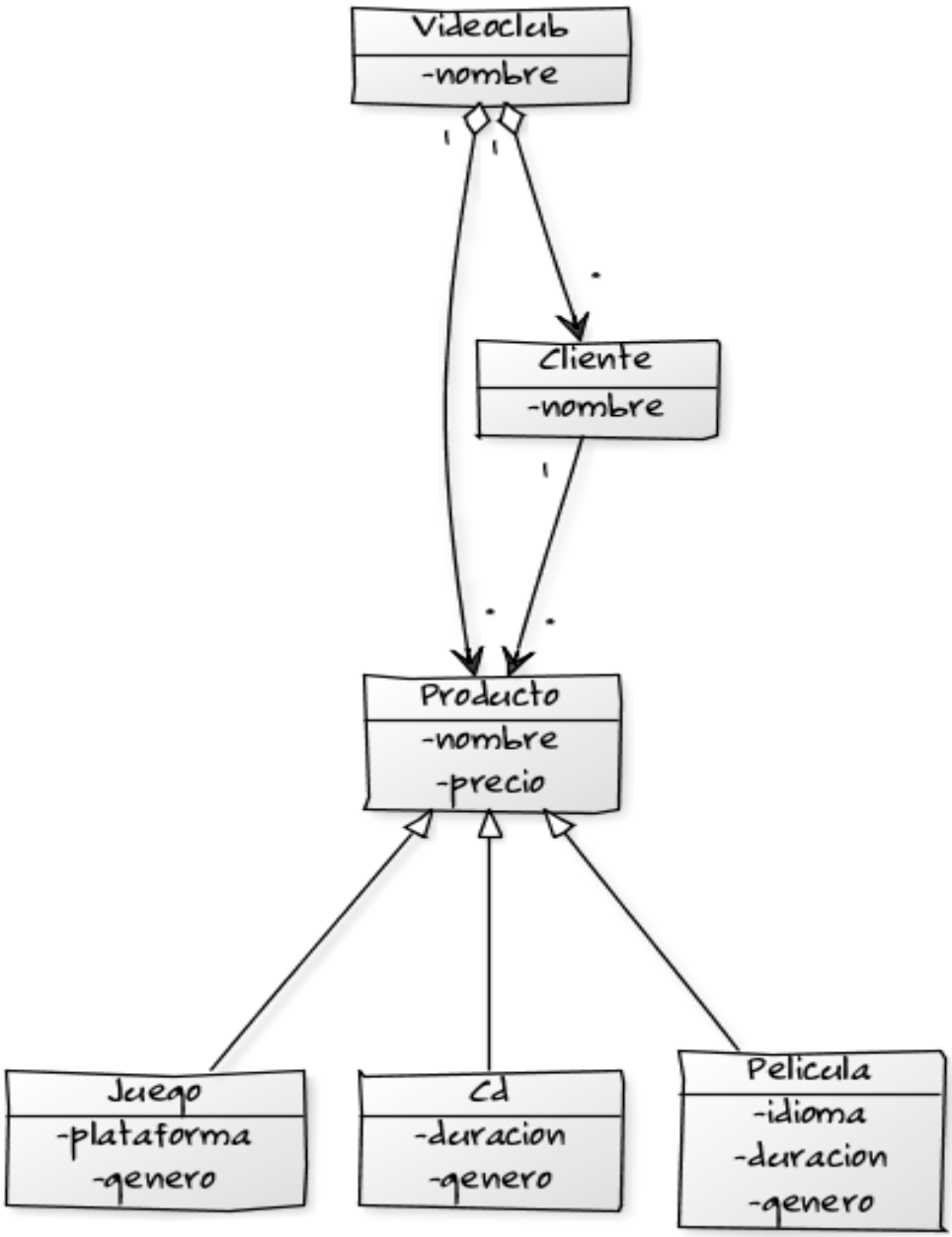
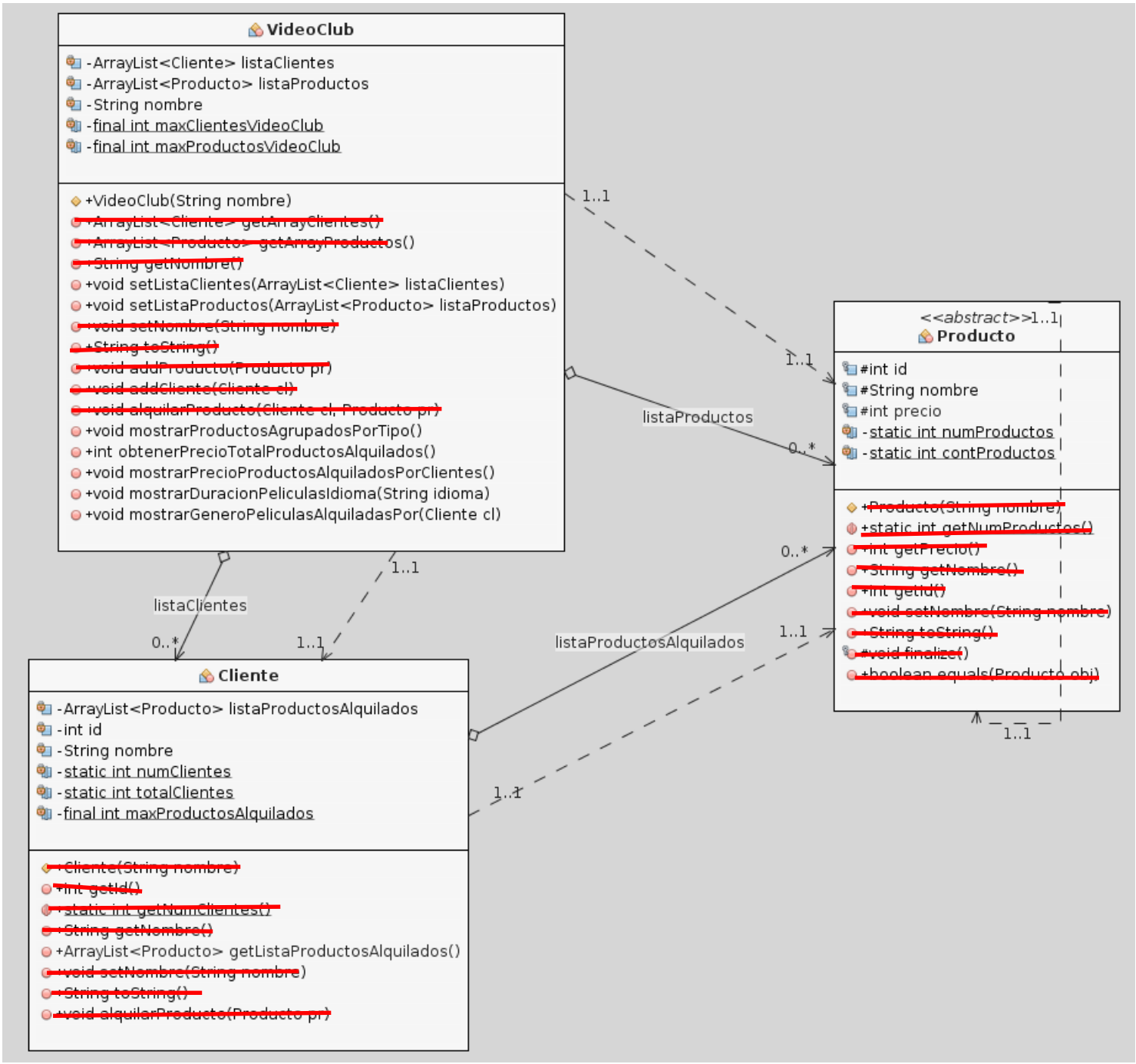
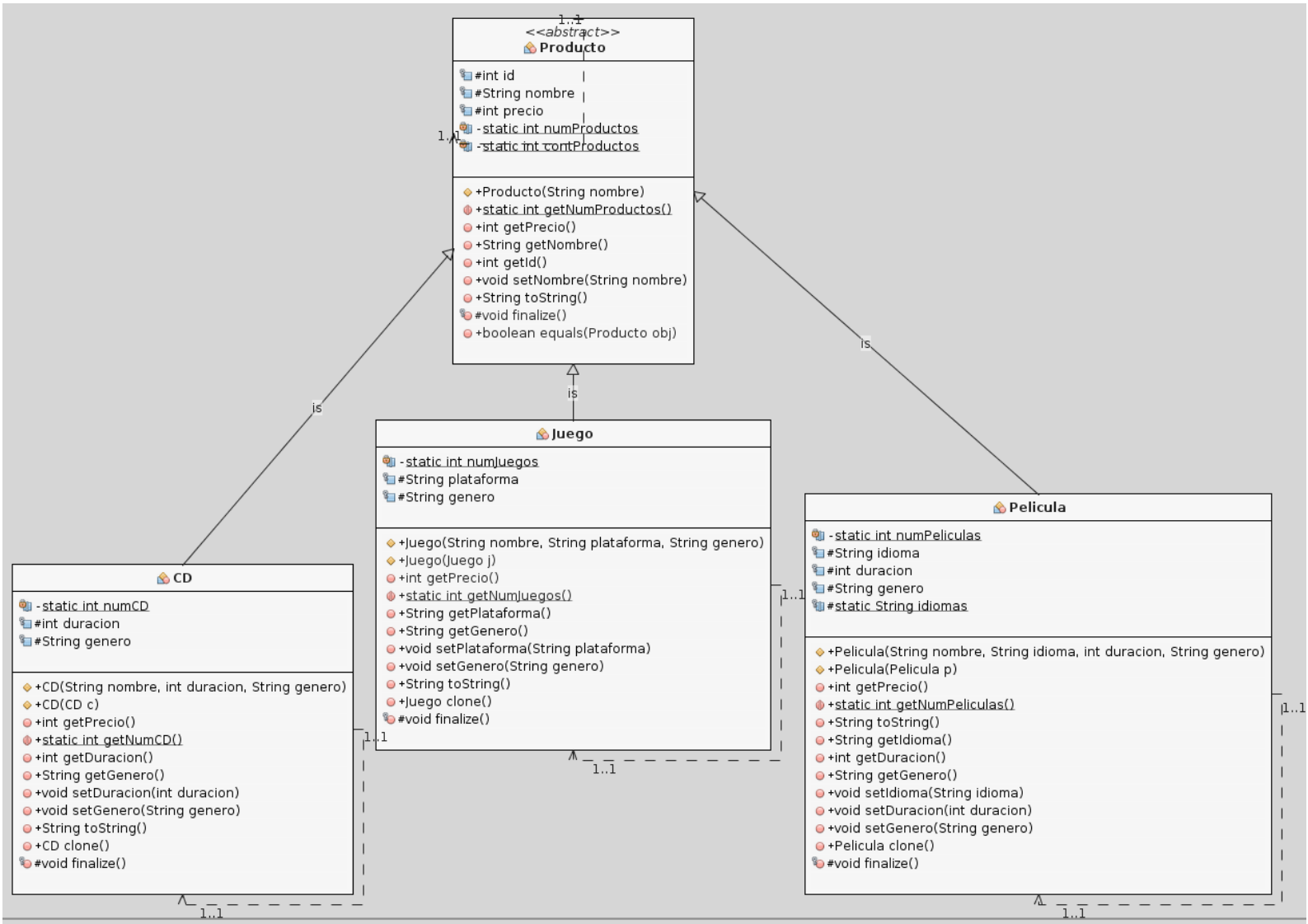
EJERCICIOS DE CLASES CON ARRAYS

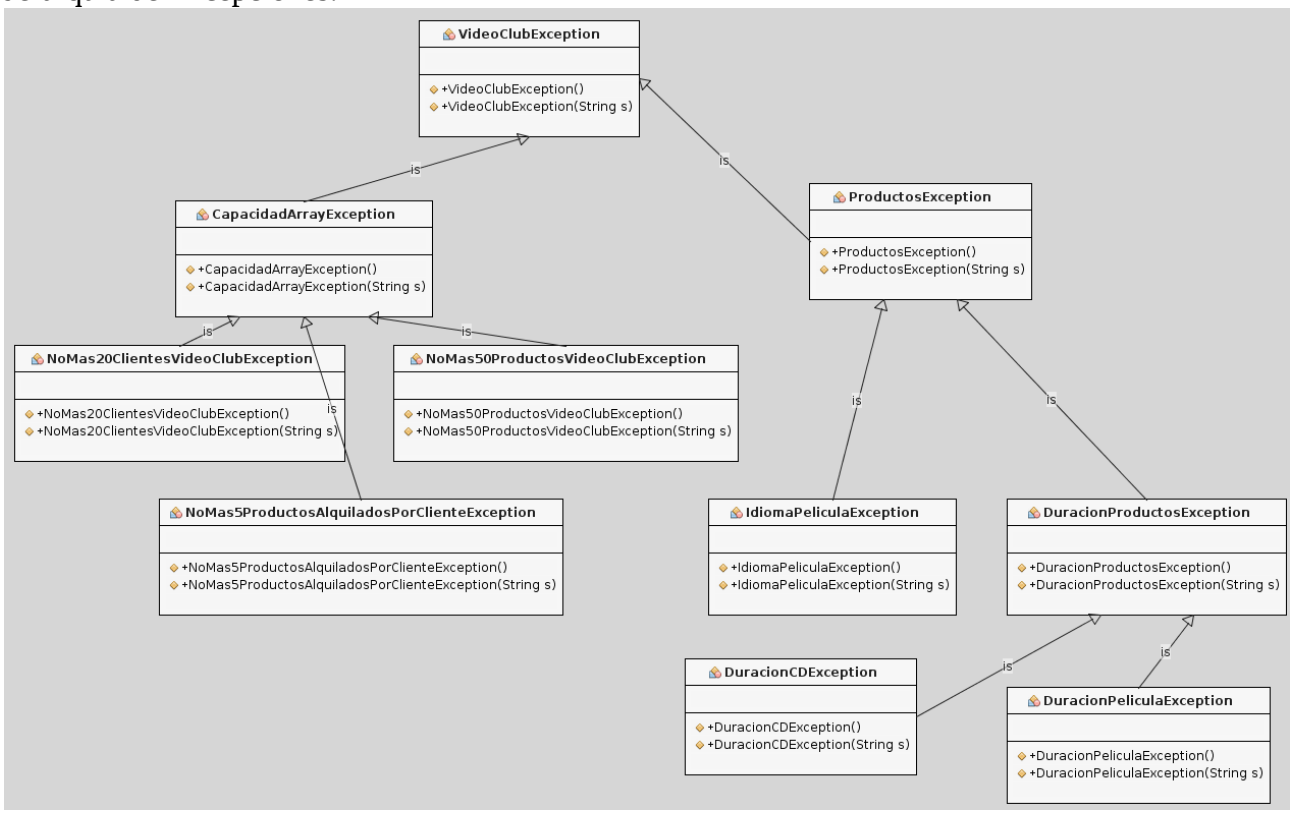
Crear objetos para gestionar un videoclub. El diagrama de clases en UML para representar gráficamente las clases que vamos a utilizar para gestionar los diferentes objetos es el siguiente.




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)
1..1
1..1
listaProductos
0..*
<<abstract>>1..1
Producto
#int id
#String nombre
#int precio
-static int numProductos
-static int contProductos
+Producto(String nombre)
+static int getNumProductos()
+int getPrecio()
+String getNombre()
+int getId()
+void setNombre(String nombre)
+String toString()
#void finalize()
+boolean equals(Producto obj)
1..1
listaClientes
0..*
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)
1..1


<<abstract>>
Producto
#int id
#String nombre
#int precio
-static int numProductos
-static int contProductos
+Producto(String nombre)
+static int getNumProductos()
+int getPrecio()
+String getNombre()
+int getId()
+void setNombre(String nombre)
+String toString()
#void finalize()
+boolean equals(Producto obj)
Juego
-static int numJuegos
#String plataforma
#String genero
+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()
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()
Pelicula
-static int numPeliculas
#String idioma
#int duracion
#String genero
#static String idiomas
+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()
is
is
is
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.

-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 **Producto**: clase abstracta que representa un producto del videoclub, es decir, no podemos instanciar objetos **Producto**.
Cuenta con una serie de atributos que heredarán las subclases, **id**, **nombre y precio,** y las propiedades estáticas contProductos (id será el mismo que el de contProductos, que se incrementará conforme se instancie un producto nuevo) y numProductos (que llevará la contabilidad de los productos existentes, incluyendo los que se van dando de alta y los que se dan da baja).
Todos los atributos son privados, pero públicos a las subclases.
El constructor: un objeto de tipo producto se puede crear dándole: nombre.
Métodos:
Método estático getNumProductos(), que devuelve el valor del total de productos.
Método abstractos getPrecio().
Y métodos públicos: getId(), getNombre() , setNombre, equals(), finalize() y toString().

-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.

- La clase **Pelicula**, posee además los atributos: numPeliculas, de tipo estático entero, que guarda el número de películas existentes.
Además, añadir el idioma, duración y género.
Todos los atributos son privados, pero públicos ante posibles clases que puedan heredar de Pelicula, a excepción del atributo numPeliculas, que será privada.
Métodos:
toString(), getNumPeliculas(), getPrecio(), getIdioma(), setIdioma(), getDuracion(), setDuracion(), getGenero() y setGenero(). Y el método sobrecargado equals(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 carácter estático con valor fijo: “ESPAÑOL”,”INGLÉS”,”FRANCÉS”,”CHINO” y “ALEMÁN”.
Se pide la gestión de la excepción IdiomaPeliculaException.

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

- La clase **CD**, posee además los atributos: numCD, de tipo estático entero, que guarda el número de CD existentes.
Además, añadir: la duración y género. Todos los atributos son privados.
Métodos: toString(), getNumCD(), getPrecio(), getDuracion(), setDuracion(), getGenero() y setGenero().Y el método sobrecargado equals(CD)
Si la duración es negativa o cero, se lanzará la excepción DuracionCDException.

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

- La clase **Juego**, posee además los atributos: numJuegos, de tipo estático entero, que guarda el número de Juegos existentes.
Además, añadir la plataforma y género. Todos los atributos son privados
Métodos: toString(), getNumJuegos(), getPrecio(), getPlataforma(), setPlataforma(), getGenero() y setGenero().
Y el método sobrecargado equals(Juego).
El constructor está sobrecargado:
- Datos de entrada: nombre, plataforma y genero.
- Dato de entrada: un objeto de tipo Juego.

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

Cuenta con una variable o propiedad que representa la colección (un array de Productos) de productos alquilados (productosAlquilados), de como máximo 5.

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

Las propiedades son privadas, y cuenta con las propiedades estáticas numClientes (que lleva la contabilidad de los clientes existentes, tanto si de dan de alta como los de baja), y totaClientes

Métodos:

estático getNumClientes..

getId, getNombre, setNombre, getProductos, alquilarProducto(objetoProducto) (que añada al array productosAlquilados ese objetoProducto)

– controlad la excepción: NoMas5ProductosClienteException en el método alquilarProducto- y toString.

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:

* mostrarPeliculasAlquiladas, tal que muestre la lista de solo las películas alquiladas por el cliente actual.

* disenoTicket, tal que se diseñe un ticket, donde aparezcan en primer lugar las películas (nombre y precio) y a continuación, el precio de todas las películas. En segundo lugar los juegos (nombre y precio) y a continuación, la suma de todos los juegos. En tercer lugar, los CD's (nombre y precio) y a continuación el precio de todos los CD's. Finalmente, el precio total

* calculoPago, tal que devuelva el total a pagar por el alquiler de las películas del cliente actual.

* numPelilculasAlquiladas, tal que devuelva el número de películas alquiladas que posee el cliente actual.

* mostrarPeliculasAlquiladasComienzoM, tal que muestre el nombre completo de las películas que tiene alquiladas el cliente actual cuyo comienzo del nombre sea "M".

* peliculasAlquiladasIdioma, tal que dado un determinado idioma, muestre el id y nombre de las películas de ese determinado idioma alquiladas por el cliente actual.

-La clase **Videoclub**: es la clase principal de nuestra aplicación. Representa a un videoclub en el dominio del problema y cuenta con una colección (array) de clientes (máximo 20) y productos registrados (máximo 50). Las propiedades son privadas, incluyendo el nombre.

Métodos: toString, getNombre, setNombre, addProducto (añade un producto en el array de productos, controlad la posible excepción: NoMas50PrdouctosVideoClubException), addCliente (añade un cliente al array de clientes, controlad la posible excepción: NoMas20ClientesVideoClubException), getProductos, getClientes, getNumProductos, getNumClientes y el método alquilar, t.q. dándole un producto y un cliente, llame al método: alquilarProducto del cliente.

Añadir los métodos:

mostrarProductosAgrupadosPorTipo() // Aptdo 16

mostrarPrecioProductosAlquiladosPorClientes() //Aptdo 17

obtenerPrecioTotalProductosAlquilados() //Aptdo 18

mostrarDuracionPeliculasIdioma(String idioma) // Aptdo 19

mostrarGeneroPeliculasAlquiladasPor(Cliente c1) // Aptdo 20

IMPORTANTE: Para probar las excepciones, supongamos que se admiten máximo 3 clientes y 9 productos en el VideoClub y que además solo se permite alquilar 2 productos por cliente.

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. Mostar el número de películas.
9. Mostrar el número de clientes.
10. El cliente: 'Javier' alquila la película: 'Star Wars III'. //desde el objeto videoclub
11. Obtener la lista de clientes registrados.
12. Obtener la lista de productos registrados**.**
13. Obtener la lista de productos alquilados por el cliente: 'Francisco'.
14. Mostrar los productos agrupados por tipo: Pelicula, CD y Juego
15. Obtener el precio de todos los productos alquilados por cada cliente
16. Obtener el precio total de todos los productos alquilados
17. Se desea saber la duración de las películas del videoClub cuyo idioma sea uno introducido por teclado
18. Indicad qué género poseen las películas que tiene alquiladas el cliente c1.