UNIVERSITAT DE BARCELONA

Ingeniería Informática

Programación II

Reproductor Multimedia

Práctica 3

Grupo C

Nombre	NIUB
Manuel Ernesto Martínez Martín	20081703
Aarón Peruga Ortiga	20026941

Tabla de contenidos

Introducción

En esta tercera entrega del proyecto de programación II se amplía el modelo de datos y añadimos a la aplicación funciones de reproducción.

Hemos realizado las siguientes tareas:

1. Añadir más opciones al menú para poder realizar las nuevas funciones que se nos pide en esta práctica.

2. Hemos implementado una nueva clase,***AlbumFitxersMultimedia***, que heredara de ***CarpetaFitxers*** y nos permitirá crear unas carpetas llamadas “álbum” donde almacenaremos nuestro ficheros multimedia.

   Cada álbum podrá contener el número de archivos que el usuario deseé o por defecto contendrá 10 ficheros multimedia. El álbum podrá contener ficheros repetidos.

3. Hacer que el reproductor funcione sin necesidad de una interfaz gráfica.

4. **Usaremos** la librería **JNA** para ejecutar código nativo junto a Java, ya que usamos como base el **VLC** y necesitamos acceder a sus binarios.

5. Practicaremos: **herencia**, **visibilidad**, **paquetes**, **clases abstractas** y **métodos abstractos**, **sobreescritura de métodos**, **serialización de datos**.

Análisis

Al querer hacer que nuestro reproductor funcione, haremos que este reproduzca los archivos multimedia que almacena en biblioteca.

También podemos seleccionar los archivos y almacenarlos en un álbum, podemos realizar dichas acciones gracias a que la clase **AlbumFitxersMultimedia** heredará de **CarpetaFitxers**, clase que nos permite realizar diversas funciones para almacenar ficheros.

La reproducción de ficheros la realiza la clase **EscoltadorReproducció** la cual heredará **EscoltadorReproduccioBasic**. Como nos dan la opción de que guardemos la configuración del reproductor (reproducción aleatoria activada, reproducción cíclica desactivada, etc.) Guardaremos como atributos de Dades los booleanos de continua y cíclica.

Nuestro programa continúa con el modelo vista-controlador, por lo que cualquier operación que queramos hacer o realizar debemos especificarlo en la vista y nuestra instancia de controlador realizará dicha acción accediendo al modelo.

Desarrollo

Las clases desarrolladas

1. **AplicacioUB3:** en esta clase hemos añadido varios métodos para ampliar la vista con los nuevos menús. Los métodos que hemos implementado son los siguientes:
   - **gestioAplicacioUB()**.
   - **gestioAlbums(Scanner sc)**: vista del submenú para la Gestión Albums.
   - **gestionarAlbum(Scanner sc, String album):** submenú para gestionar un álbum
   - **gestioControl(Scanner sc):** submenú Control/Reproduccio
   - **gestionarReproduccio(Scanner sc):** submenú para gestionar la reproducción de un fichero.
   - **controlReproduirAlbum(Scanner sc)**: pide el nombre del albúm para reproducirlo.
   - **controlReproduirFitxer(Scanner sc):** pide la id del fichero para reproducirlo.
   - **preguntaImatgeFitxer(Scanner sc)**: pregunta la ruta para añadir una carátula a un fichero de audio.
   - **afegirFitxerAlbum(Scanner sc, String album)**: añadimos un fichero al álbum.
   - **eliminarFitxerAlbum(Scanner sc, String album)**: elimina un fichero del álbum.
   - **mostrarUnAlbum(String title)**: muestra los ficheros que contiene un álbum.
   - **crearAlbum(Scanner sc)**: creas un álbum.
   - **preguntaTamanyAlbum(Scanner sc, String album)**: pregunta el tamaño del albúm. Si el usuario responde que no quiere seleccionar el tamaño del álbum este se crea por defecto.
   - **eliminarAlbum(Scanner sc)**: elimina un álbum.
   - **mostrarAlbums()**: muestra todos los álbumes que han sido creados.
   - **seleccionaAlbum(Scanner sc)**: selecciona un álbum creado anteriormente.
2. **Reproductor**: En esta clase hemos añadido dos nuevos métodos que nos ayudarán a reproducir los ficheros multimedia.
   - **reprodueix(FitxerReproduible fr)**: reproduce un archivo de video.
   - **reprodueix(Audio audio, File fitxerImatge)**: reproduce un archivo de audio.

3. **Controlador:** en esta clase implementaremos inControlador, interfaz proporcionada por los profesores, y hemos implementado nuevos métodos a la clase (que llamaran a los de Dades, EscoltadorReproduccio o Reproductor).
    - **afegirAlbum(String titol)**: manda la orden de añadir un nuevo álbum
    - **afegirAlbum(String titol, int size)**: manda añadir un nuevo álbum pero con tamaño personalizado. [**CUESTIONES PLANTEADAS**]
    - **mostrarLlistatAlbums()**: enseña la lista de álbum existentes.
    - **esborrarAlbum(String titol)**: manda borrar un álbum.
    - **existeixAlbum(String titol)**: comprueba si existe el albúm.
    - **mostrarAlbum(String titolAlbum)**: muestra la información respecto a un álbum en concreto.
    - **obrirFinestraReproductor():** abre la ventana de reproducción.
    - **tancarFinestraReproductor()**: cierra la ventana de reproducción.
    - **afegirFitxer(String titolAlnum, int id)**: para añadir un fichero a un álbum.
    - **esborrarFitxer(String titol, int id)**: para borrar un fichero de un álbum.
    - **reproduirFitxer(int id)**: reproduce una carpeta de un fichero.
    - **reproduirCarpeta()**: reproduce la biblioteca
    - **reproduirCarpeta(String titol)**: reproduce un álbum.
    - **reemprenReproduccio()**: vuelve a seguir la reproducción si está activa.
    - **pausaReproduccio()**: pausa la reproducción si está activa.
    - **aturaReproduccio()**: para la reproducción si está activa.
    - **saltaReproduccio()**: salta a la siguiente canción en caso de que se pueda.
    - **activarDesactivarContinua()**: cambia el booleano de continua
    - **activarDesactivarAleatoria()**: cambia el booleano de aleatoria.
    - **getTitolAlbum(int id)**: a partir del id del álbum obtiene su título.
4. **EscoltadorReproduccio**: Hemos implementado una serie de métodos para poder reproducir una serie de ficheros multimedia:
    - **onEndFile()**: es llamado cuando termina la reproducción, llamará al método **next()**.
    - **next()**: es llamado desde **onEndFile()** y en iniciarReproduccio(...) para reproducir la siguiente canción, hará las comprobaciones y comprobará con hasNext().
    - **hasNext()**: comprueba si tiene siguiente.
    - **iniciarReproduccio(CarpetaFitxers llistaReproduint, boolean reproduccioCiclica, reproduccioAleatoria)**: inicia la reproducción.
5. **CarpetaFitxers**: implementa InFileFolder pero no nos altera nada.

6. **AlbumFitxersMultimedia**: hereda de CarpetaFitxers, tiene un atributo título, hace uso del límite de capacidad y también hace override de equals() y hashCode() para poder comprobar duplicados cuando vamos a guardar albums.
7. **Dades**: hemos hecho cambios en esta clase para poder hacer las operaciones nuevas del controlador.
    - **esborrarFitxer(int id)**: borrar un fichero de de la biblioteca lo borrará de todos los álbums también.
    - **Getters y Setters** de reproducción cíclica y aleatoria.
    - **esborrarUnAlbum(String titol)**: borra un álbum.
    - **afegirAlbum(String titol, int size)**: añade un nuevo álbum.
    - **existeixAlbum(String titol)**: comprueba si existe ya el álbum.
    - **albumListToString()**: obtiene una lista de álbums.
    - **albumToString(String titol)**: obtiene la lista de ficheros del álbum.
    - **afegirFitxerAlbum(String titolAlbum, int id)**: añade un archivo al álbum.
    - **esborrarFitxerAlbum(String titol, int id)**: borra un archivo del álbum.
    - **setReproductor(Reproductor reproductor)**: hace un set del reproductor a cada fichero reproducible después de cargar datos.
    - **getCarpetaReproduccio(int id)**: hace una carpeta fantasma para reproducir un solo fichero. [**REPRODUCCIÓN DE UN SOLO FICHERO**]
    - **getCarpetaReproduccio()**: getter de la biblioteca completa.
    - **getCarpetaReproduccio(String titol)**: obtiene la carpeta de un álbum indicado.
    - **getTitolAlbum(int id)**: obtiene el título del álbum a través del id.
    - **getAlbumByTitle(String title)**: obtiene un álbum a través del título.

Hemos determinado que EscoltadorReproducció tendrá estos atributos:

```
private CarpetaFitxers llistaReproduint;
private List<Integer> llistaCtrl;
private boolean reproduccioCiclica, reproduccioAleatoria;
private int posicio;
private boolean reproduint;
```

Cuestiones planteadas

- Hemos hecho otro método en el Controlador para crear un álbum, porque en el original que hacemos override no tiene como parámetro el tamaño del álbum (afegirAlbum() sobrecargado).
- Si un audio no tiene carátula o tiene una que no existe no se mostrará nada, si tiene una válida si se cargará y se verá.

Reutilización de la práctica 2

Clases reaprovechadas

- **Audio**: Clase para representar todos los ficheros reproducibles que sean audio.
- **BibliotecaFitxersMultimedia**: Clase para almacenar todos los ficheros multimedia que el usuario guarde en el reproductor. Hereda de CarpetaFitxers.
- **FitxerMultimedia:** Clase para representar un los archivo multimedia cualquiera. Será la clase padre de FitxerReproduible.
- **FitxerReproduible:** Clase para representar todos los ficheros que se reproducen en el reproductor.
- **Video:** Clase para representar todos los ficheros reproducibles que sean video.

Cambios sobre las clases utilizadas

- **IniciadorAplicacioUB:** Esta clase es donde se inicia la nueva aplicacioUB3.
- **AplicacioUB3:** En esta clase hemos aumentado el nº de menus para cubrir las nuevas necesidades de esta nueva práctica. Hemos creado una menú para “Gestio Albúm” y “Control Reproduccio” juntos a dos submenús de las respectivas clases.
- **Dades:** Hemos tenido que añadir nuevos métodos para cubrir las nuevas opciones del reproductor,es decir, todas aquellas que estén relacionadas con “Gestion Álbum” y “Control Reproduccio” . Estos métodos los ejecutará el controlador cuando el usuario seleccione la opción específica.
- **Reproductor:** En reproductor hemos implementado dos métodos *“reprodueix”*, uno para archivos de audio y otro para archivos de video.
- **Controlador:** En controlador hemos tenido que implementar la interfaz InControlador. Contendrá nuevos métodos para clases de esta práctica.
- **CarpetaFitxers:** Clase para crear todas las carpetas que almacenan ficheros multimedia. Será la clase padre de BibliotecaFitxersMultimedia y AlbumFitxersMultimedia, ahora implementa InFileFolder.

En **DESARROLLO** hay más detalles.

Implementación de modos continuo y aleatorio

Continuo/Cíclico

La implementación del modo reproducción continua la hemos implementado de la siguiente forma:

El usuario pedirá desde la vista si quiere que el modo de reproducción cíclica esté activado o desactivado. En caso de que se active, el controlador llamará a Dades que pondrá el booleano de **reproduccioCiclica** en true.

A continuación, cuando hagamos una reproducción de una carpeta hasNext() analizará si sigue reproduciendose o no, reproducirá el fichero y cuando llegue al final, volverá a dar la vuelta porque utilizamos la operación módulo:

```
posicio = (posicio + 1) % llistaCtrl.size();
```

Aleatorio

En la implementación del modo de reproducción aleatorio se desarrolla de la misma forma que el modo continuo.

El usuario activa desde la vista el modo aleatorio. En caso de que se active, el controlador llamará a **Dades** que pondrá el booleano de **reproduccioAleatoria** en true.

En caso de que el método **reproduccioAleatoria** esté activada el método **inciarReproduccio()** hará un shuffle a **llistaCtrl,** lista que tiene las posiciones los fichero de las reproducciones**,** lo que hará que se desordene la lista y se iniciará la reproducción de la carpeta desde la posición que ha quedado en **llistaCtrl**.

Nuestra llistaCtrl es un ArrayList de Integers guardado en un atributo tipo List<Integer>, y lo llenamos así:

```
llistaCtrl = IntStream.range(0, llistaReproduint.getSize()).boxed().collect(Collectors.toList());
```

Java 8 permite hacer esto en una línea y no tener que hacer un bucle con un contador para meter los números, sería del mismo estilo que una List Comprehension de Python.

Implementación reproducir fichero en Controlador

La implementación del método de reproduirFitxer() de la clase controlador es la siguiente:

Al principio del método abrimos la ventana de reproducción del programa. Luego el usuario indicará el archivo de la BibliotecaFitxers que desee a través del identificador que le pase el usuario.

Se creará una CarpetaFitxers nueva con un solo fichero que será la que usaremos.

Se hace una llamada de Controlador a Dades para crear y retornar la Carpeta

(Es decir una Carpeta fantasma que contiene un solo fichero).

Se ejecuta el método **iniciarReproduccio()** de la clase EscoltadorReproducció y empezará a reproducir el fichero en concreto, contando también si la reproducción es cíclica (nunca será aleatoria).

También cuenta con las excepciones de la clase AplicacióExcepcion para poder gestionar todos los errores de las prácticas.

```
@Override
public void reproduirFitxer(int id) throws AplicacioException {
    try {
        obrirFinestraReproductor();
        escoltador.iniciarReproduccio(dades.getCarpetaReproduccio(id),
                dades.isReproduccioCiclica(), false);
    } catch (AplicacioException ae) {
        tancarFinestraReproductor();
        throw new AplicacioException("Error al reproduir");
    }
}
```

Reproducción de una biblioteca de ficheros

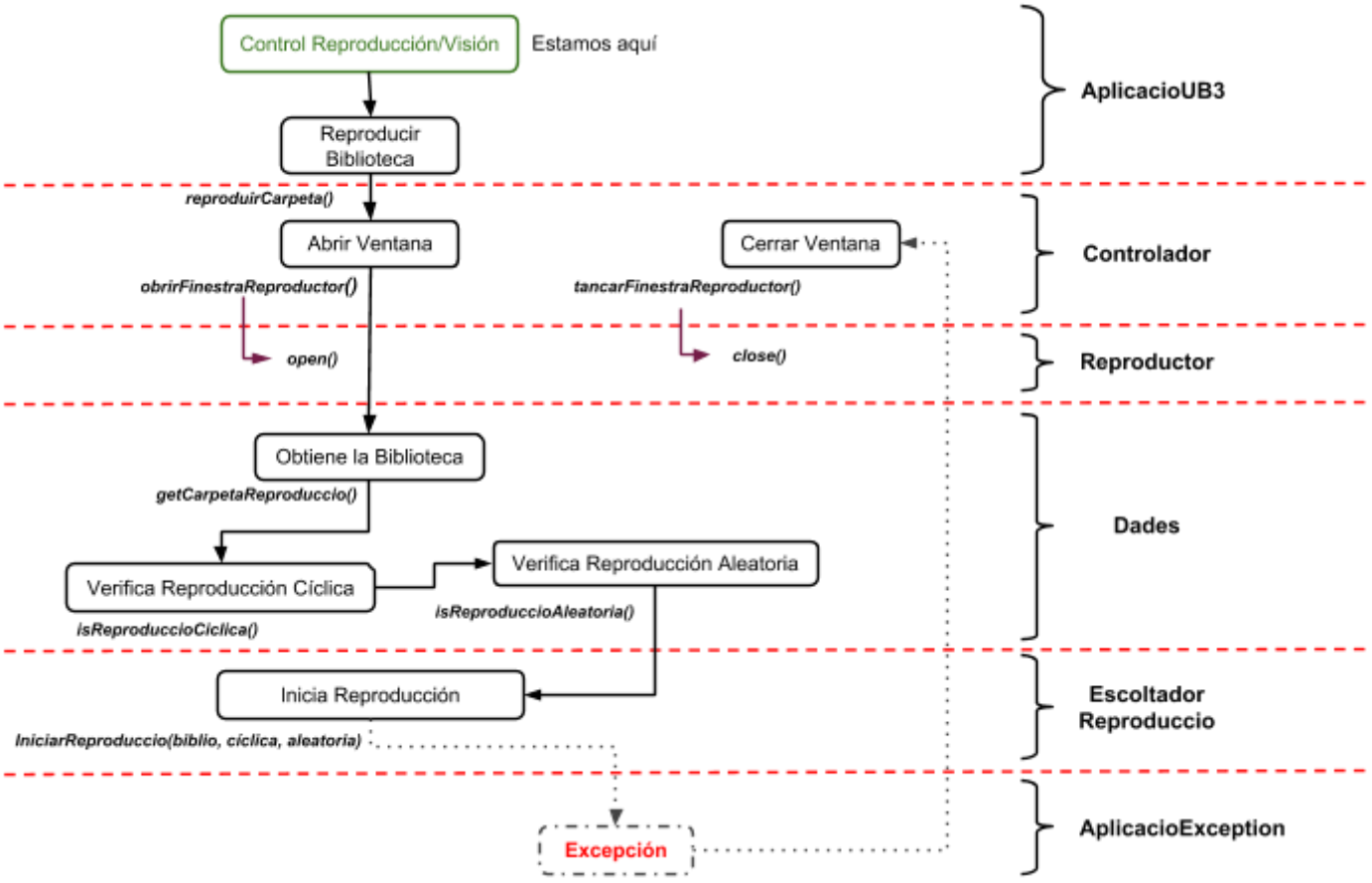
A la hora de reproducir elegimos desde la vista **AplicacioUB3** que carpeta o fichero queremos reproducir.

Controlador llama a la función de abrir la ventana de **Reproductor**, si hay algún fallo la cierra la ventana.

Controlador llama a funciones de **Dades** para obtener la ***CarpetaFitxers*** (álbum, carpeta fantasma de un fichero o biblioteca entera) y comprobar la reproducción: ***aleatoria*** y ***cíclica***.

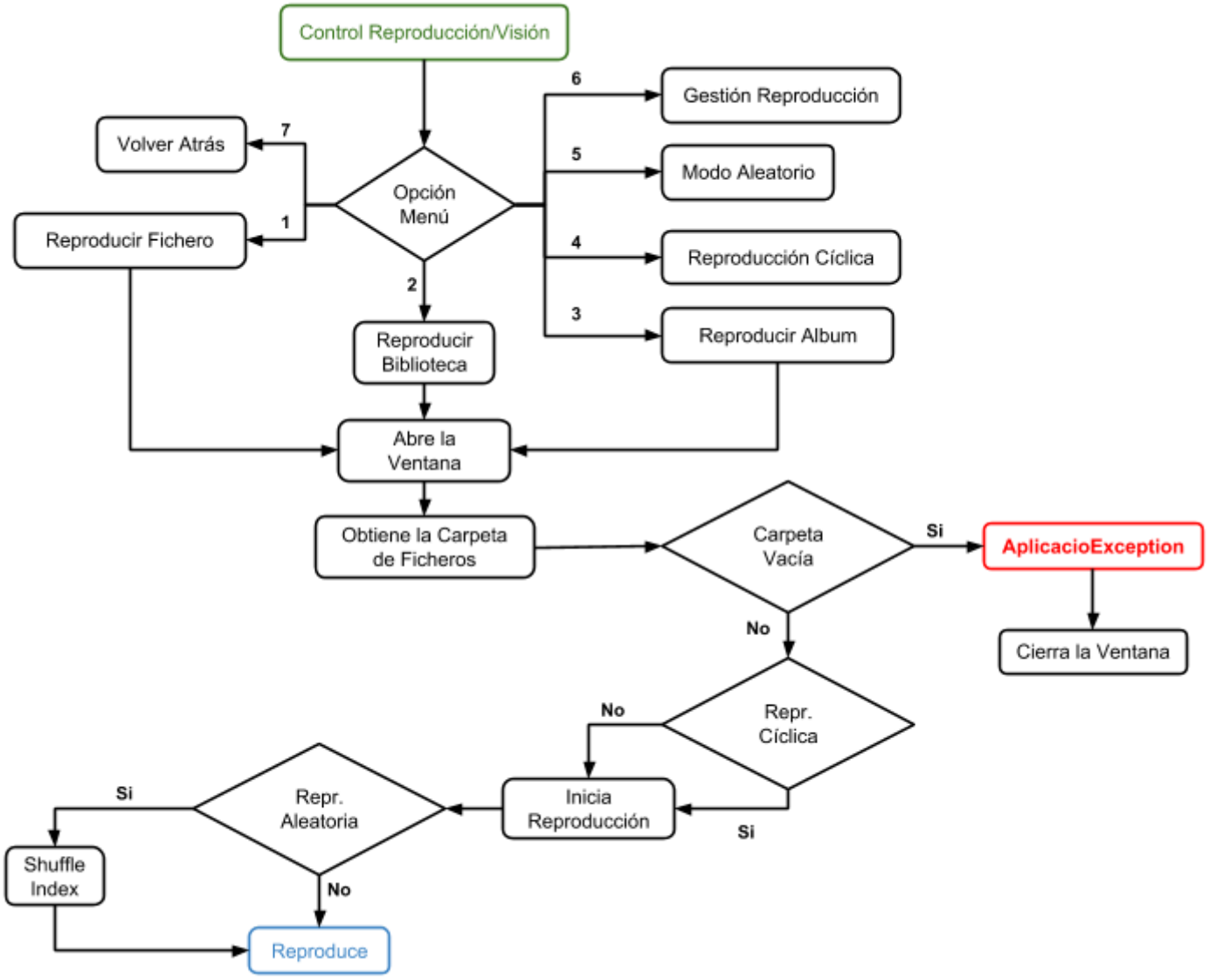
Controlador llamará a *iniciarReproduccio()* de **EscoltadorReproduccio** con los parámetros necesarios.

Esquema conceptual para iniciar una reproducción de la biblioteca (Clases por donde pasamos y sus métodos)

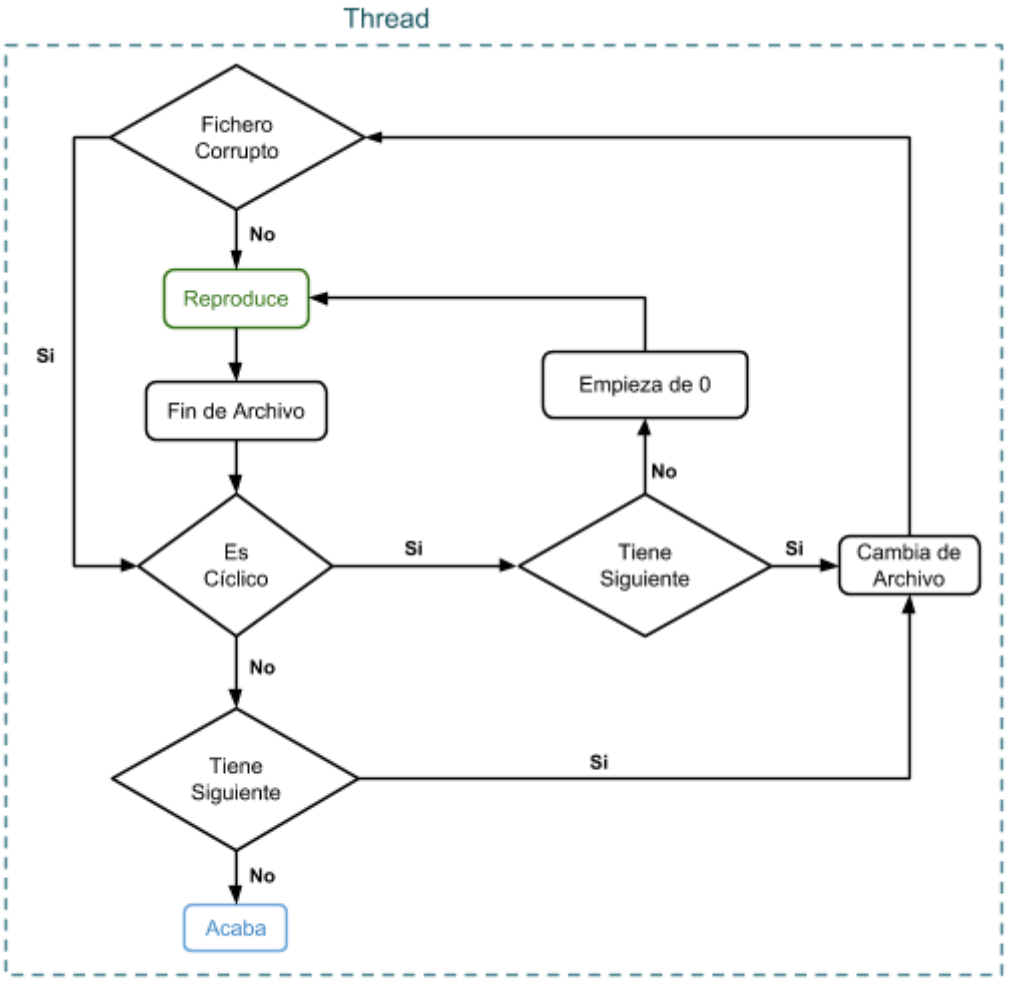


Diagramas de flujo

Iniciar Reproducción



Reproducción en curso



La reproducción se lleva a cabo en otro hilo, mientras podemos interactuar con el programa para hacer otras cosas o parar/saltar/continuar/pausar la reproducción en curso.

Excepciones en gestión de carpeta a reproducir

Cuando hay un fichero corrupto que no se puede reproducir, capturamos la excepción y lo que hacemos básicamente es pasar al siguiente si lo hay sucesivamente hasta encontrar uno que funcione (mientras borramos de llistaCtrl el índice defectuoso).

Si no lo hay y es reproducción cíclica repetimos el mismo fichero.

Si es reproducción cíclica y ningún fichero se puede reproducir no se reproducirá nada (descarta todos).

De esta forma lo hacemos:

```
@Override
protected void onEndFile() {
    next();
}
@Override
protected void next() {
    posicio = (posicio + 1) % llistaCtrl.size();
    if (hasNext()) {
        File file = llistaReproduint.getAt(llistaCtrl.get(posicio));
        if (file instanceof FitxerReproduible) {
            try {
                ((FitxerReproduible) file).reproduir();
            } catch (AplicacioException ex) {
                ////////////////////////////////////////////////////////////
                // Evitamos el loop de ciclica so no funciona ningún fichero
                // Los que no funcionen se elimina el index de la llistaCtrl
                llistaCtrl.remove(posicio);
                posicio--;
                ////////////////////////////////////////////////////////////
                next();
            }
        }
    } else {
        reproduint = false;
    }
}

@Override
protected boolean hasNext() {
    return posicio + 1 <= llistaCtrl.size() || (reproduccioCiclica && !llistaCtrl.isEmpty());
}

public void iniciarReproduccio(CarpetaFitxers llistaReproduint, boolean reproduccioCiclica, boolean reproduccioAleatoria) {
    this.llistaReproduint = llistaReproduint;
    ////////////////////////////////////////////////////////////////////////////////////////////////
    // (JAVA8) Generamos una Lista de interos desde 0 hasta tamaño de carpeta-1
    llistaCtrl = IntStream.range(0, llistaReproduint.getSize()).boxed().collect(Collectors.toList());
    ////////////////////////////////////////////////////////////////////////////////////////////////
    this.reproduccioCiclica = reproduccioCiclica;
    this.reproduccioAleatoria = reproduccioAleatoria;
    posicio = -1;
    reproduint = true;
    if (this.reproduccioAleatoria) {
        Collections.shuffle(llistaCtrl);
    }
    next();
}
```

Cada archivo corrupto o no funcional encontrado, se eliminará el entero índice del arraylist de enteros llistaCtrl, así evitamos un loop infinito si ningún archivo se puede reproducir.

No borramos de la Carpeta, solo del llistaCtrl.

Observaciones generales

- Hemos observado que tener un boolean[] llistaCtrl en EscoltadorReproduccio no era óptimo porque siempre debemos recorrerlo para encontrar si hay un false.

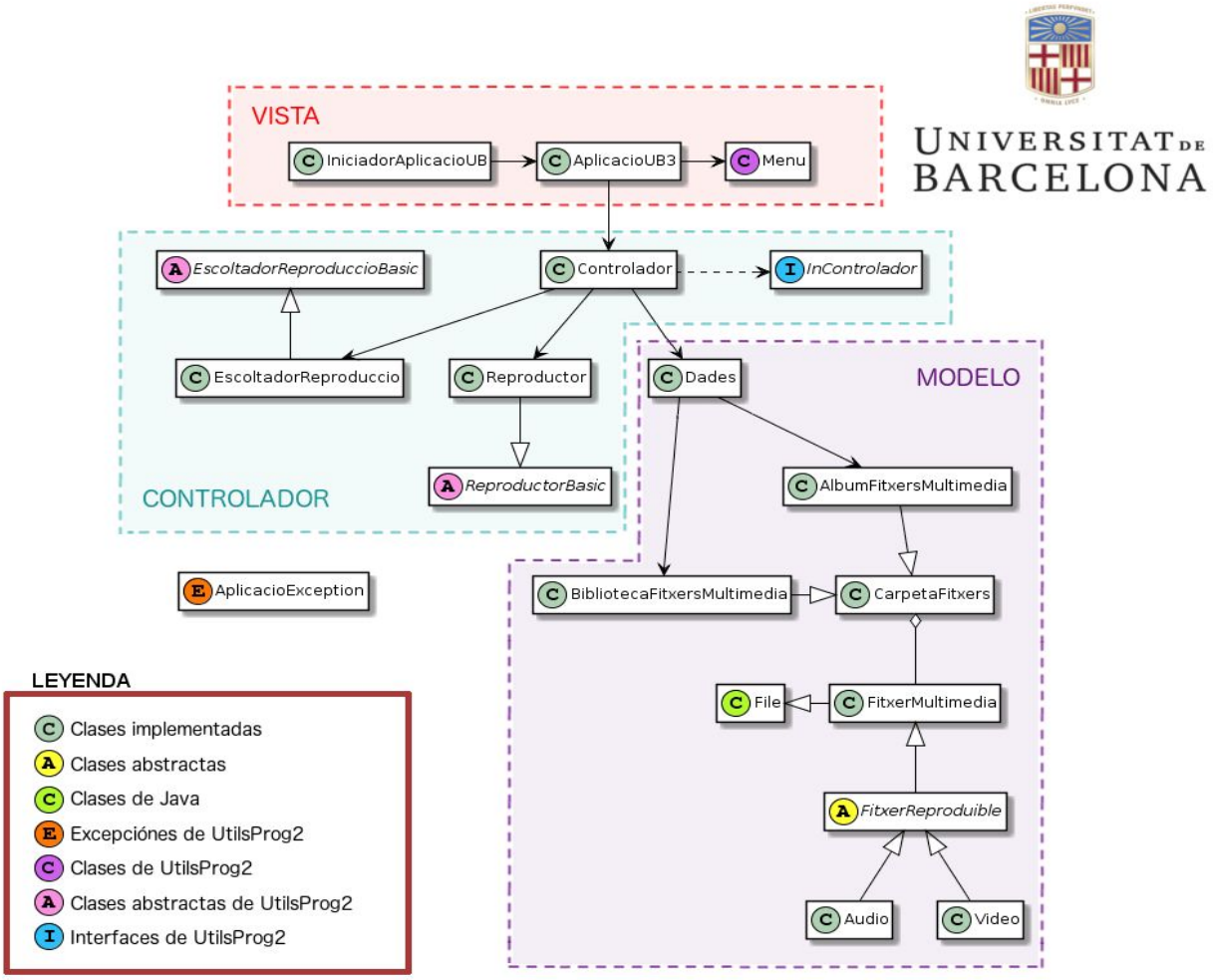
  Entonces hemos hecho un ArrayList de Integers con el tamaño de la carpeta, a ese ArrayList le haremos shuffle si hay que desordenar, etc **[ALEATORIO]**.

  Lo suyo sería hacer una copia de la Carpeta de Fitxers (no usar la original) y hacer shuffle en esa copia sin alterar la original para reproducir en modo aleatorio, ya que si fuera una aplicación como spotify, y tienes un álbum de miles de canciones, si tienes que recorrer una lista de booleanos para ver si hay uno false en el pero de los casos es O(n) y tienes que hacerlo por cada archivo al acabar.

- Creemos que **no es correcto que** un **FitxerReproduible tenga** de **atributo** una referencia de **Reproductor**, ya que es atributo directo de Controlador, lo suyo es usar ese para reproducir los ficheros, aunque lo hemos dejado tal cual se pide, era solo una observación, sabemos que no es otro reproductor diferente, solo hay un objeto Reproductor en toda la práctica, esto equivaldría en C++ a que fuera (**Reproductor *&r**), Puntero de una referencia.

- **La Serialización no es mejor modo de guardar la información**, sobre todo si la aplicación tuviera opciones Online y enviará por socket información a otro programa igual pero con versión diferente, o a otro programa igual escrito en otro lenguaje.

Diagrama UML Simple



Debido al gran volumen de atributos / métodos de estas clases y como no se pedía diagrama UML, hemos prescindido de poner los métodos y atributos para que sea más legible.

Pruebas realizadas

Material de prueba: 7 archivos (EN ESE ORDEN):

1 archivo guardado como audio o video que no sea ninguno (ej: un .txt)
1 canción con carátula.
1 canción con carátula que no exista.
1 archivo guardado como audio o video que no sea ninguno (ej: un .txt)
1 canción sin caratula.
1 video.
1 archivo guardado como audio o video que no sea ninguno (ej: un .txt)

- **Prueba 1**: reproducción de los 7 archivos normal (sin cíclica ni aleatorio).
- **Prueba 2**: reproducción de los 7 archivos NO cíclica pero SI aleatorio.
- **Prueba 3**: reproducción de los 7 archivos SI cíclica y NO aleatorio.
- **Prueba 4**: reproducción de los 7 archivos SI cíclica y SI aleatorio.

Borrado de los 3 archivos (no reproducibles) y quedarán 4 archivos reproducibles

- **Prueba 5**: reproducción de los 4 archivos normal (sin cíclica ni aleatorio).
- **Prueba 6**: reproducción de los 4 archivos NO cíclica pero SI aleatorio.
- **Prueba 7**: reproducción de los 4 archivos SI cíclica y NO aleatorio.
- **Prueba 8**: reproducción de los 4 archivos SI cíclica y SI aleatorio.

Uso solo de archivos no válidos para reproducir

- **Prueba 9**: reproducción de los 3 archivos normal (sin cíclica ni aleatorio).
- **Prueba 10**: reproducción de los 3 archivos NO cíclica pero SI aleatorio.
- **Prueba 11**: reproducción de los 2 archivos SI cíclica y SI aleatorio.

Resultados

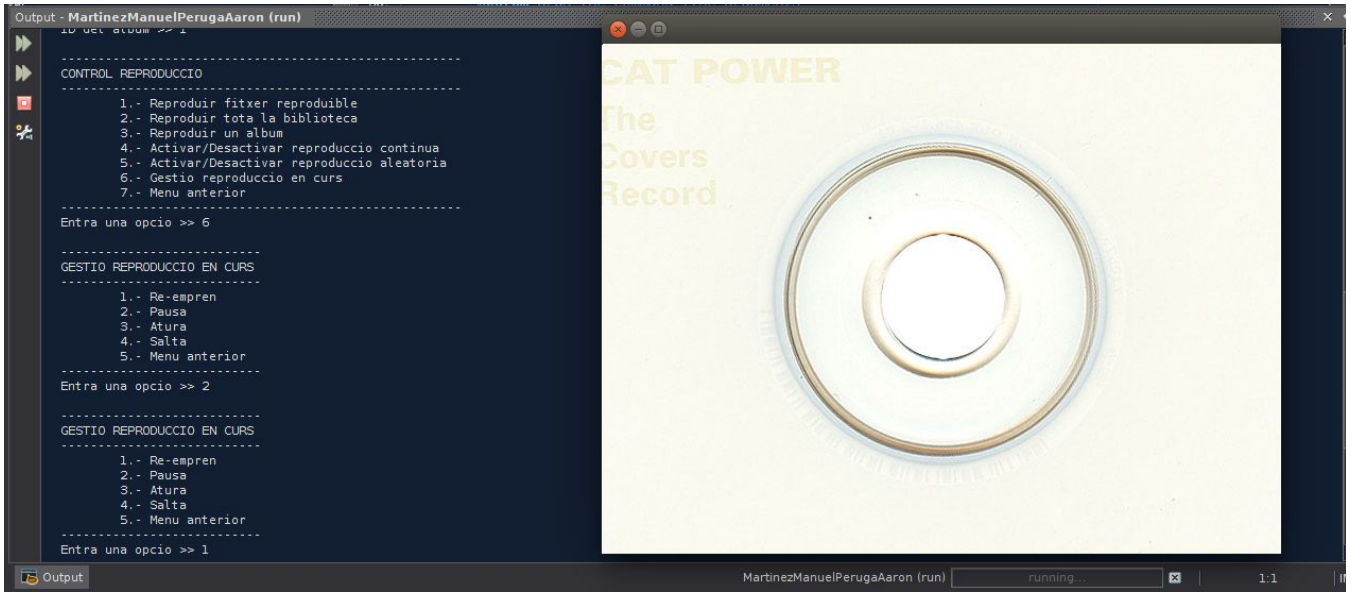
La práctica funciona en su totalidad en windows y linux siempre y cuando esté instalado el vlc y ha pasado las pruebas del apartado anterior.

En mac no funciona debido a la falta de unas librerías que oracle quitó a partir del JDK7.

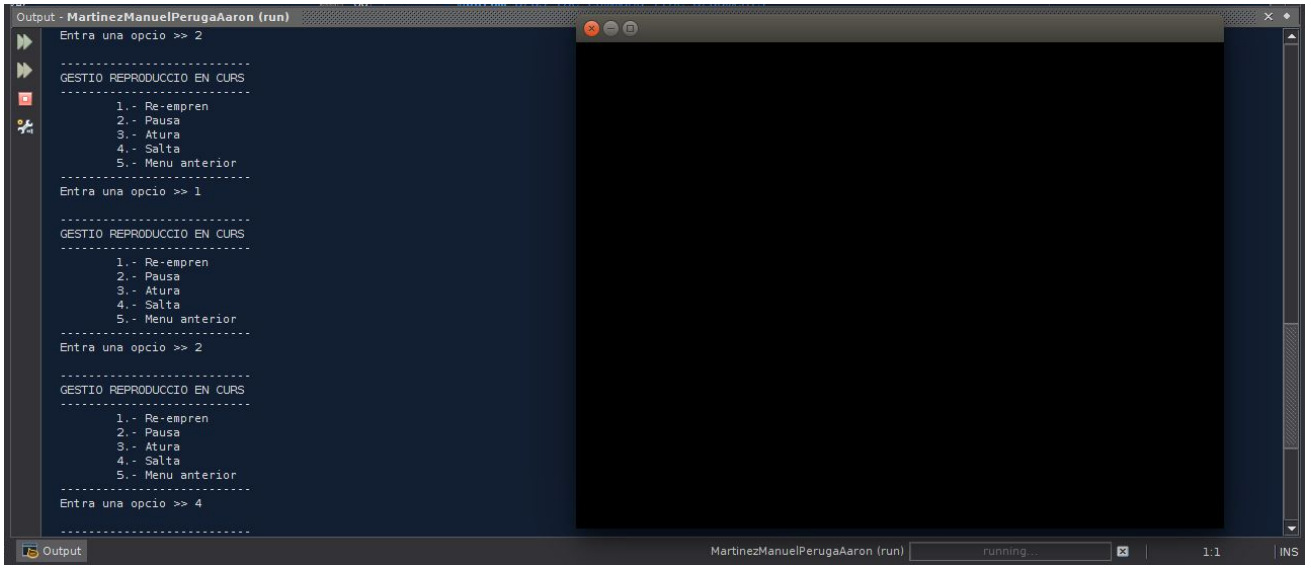
Como resultado final de esta cuarta práctica. Hemos conseguido que nuestro programa pueda reproducir archivos de audio y video desde dentro del proyecto gracias al Reproductor Multimedia VLC. Además el programa nos facilitará el almacenamiento de estos ficheros multimedia en álbums.

En el proyecto también podremos realizar la funciones de la práctica anterior como guardar/recuperar datos, añadir ficheros, etc.

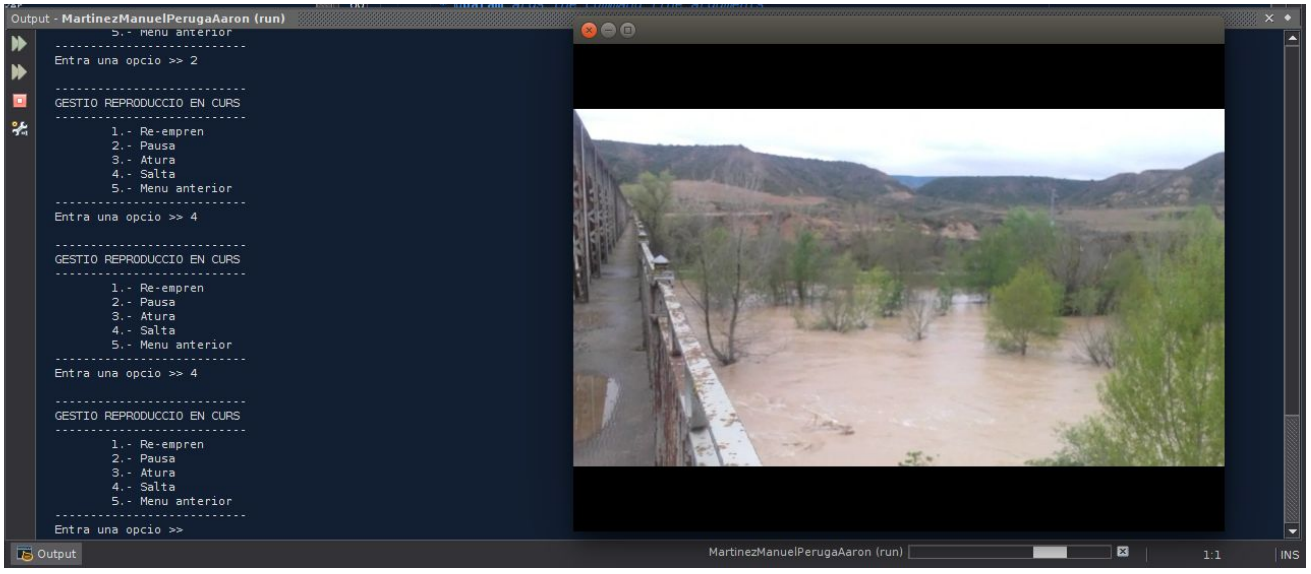
En definitiva, el proyecto puede llevar a cabo todas las funciones que realiza un reproductor multimedia gracias al reproductor multimedia VLC.



Reproducción de un fichero de audio con carátula.



Reproducción de un archivo de audio sin carátula.



Reproducción de un archivo de video