UNIVERSITAT DE BARCELONA

Ingeniería Informática

Programación II

Grupo C

Reproductor Multimedia

Práctica 2 - *27/03/18*

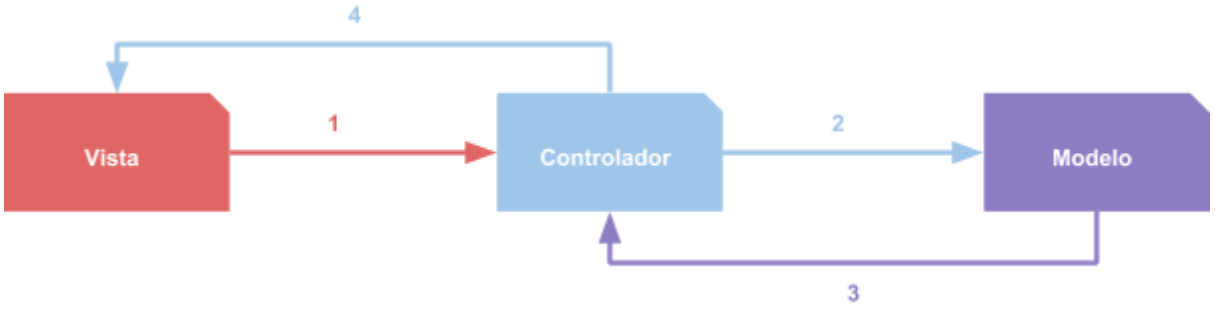
Nombre	NIUB
Manuel Ernesto Martínez Martín	20081703
Aarón Peruga Ortiga	20026941

Índice

Introducción

En esta práctica añadiremos una serie de cosas a lo que ya hicimos en la práctica 1:

1. **Aumentaremos las opciones del menú**.
2. **Aplicaremos** el MvC (**modelo vista-controlador**), donde la vista solo se dedicará a mostrar, recibir información por teclado y ser actualizada debido a las acciones del controlador sobre el modelo.



3. **Usaremos** la librería **JNA** para ejecutar código nativo junto a Java, ya que usamos como base el **VLC** y necesitamos acceder a sus binarios.
4. Practicaremos: **herencia**, **visibilidad**, **paquetes**, **clases abstractas** y **métodos abstractos**, **sobreescritura de métodos**, **serialización de datos**.

Análisis

Puesto que vamos a usar el modelo vista-controlador, nuestra vista deberá tener una instancia al controlador.

Cada acción que queramos hacer de consulta/escritura/borrado… no la hará la vista, sino que hemos de decir que nuestra instancia controlador lo haga.

Hemos de decir que el controlador tiene una instancia de la clase que gestiona todos los datos (Dades), entonces el controlador tampoco hace ningún cambio en los datos sino que ejecuta ese método en su instancia dades y ya ahí hace la operación pertinente que puede retornar algo o ser void.

De esta manera desacoplaríamos la vista del modelo.

Desarrollo

1. **BibliotecaFitxersMultimedia** (hereda de CarpetaFitxers) lo único que hacemos es **@Override** a los métodos: **isFull()** para que retorne false y al **freeSpace()** para que retorne -1 que significa que el espacio libre es ilimitado.

2. En la clase **Dades** hemos añadido estos métodos:

    - **verify(cami)** para comprobar si un archivo existe
    - **buida()** para comprobar si la biblioteca está vacía.
    - **getBibliotecaList()** que usa el toString() de BibliotecaFitxersMultimedia y retorna una lista de Strings.
    - **deleteFitxer(id)** que borrará según el índice comenzando por 1 hasta su tamaño total un fichero.
    - **afegirNouFitxer(parametros)** llamará al método de addFitxer() de BibliotecaFitxersMultimedia heredado de CarpetaFitxers.

3. **Controlador** tendrá además de los métodos del enunciado pdf:

    - **estaBuida()** para comprobar a través de Dades si la biblioteca está vacía.
    - **checkExist()** para comprobar a través de Dades si el fichero existe antes de tener que introducir más datos en la vista.

4. **AplicacioUB2** tiene los menús separados en diferentes métodos, a todos ellos le pasamos como parametro la referencia del Scanner:

    - **gestioAplicacioUB()** menú principal.
    - **gestioBiblioteca(scanner)** submenú de la Biblioteca de ficheros.
    - **gestioAfegir(scanner)** submenú para añadir archivos de audio o video.
    - **writeVideo(scanner)** para introducir por teclado los datos del fichero de video.
    - **writeAudio(scanner)** para introducir por teclado los datos del fichero de audio.
    - **saveFile(scanner)** para escribir por teclado la dirección de destino de los datos.
    - **loadFile(scanner)** para escribir por teclado la dirección de origen de los datos.

Cuestiones Planteadas

1. Nuestro Controlador es Singleton, así que tiene su constructor privado, un método privado para que lo cree si no está creado y un getter público, después de esa aclaración continuamos:

   El Controlador tiene un objeto de la clase Dades y otro de la clase Reproductor.

   De momento los métodos de añadir fichero, borrar, etc se ejecutan a través de la instancia de Dades, o sea : **dadesInstancia.método(parámetros)**, de modo que quien de verdad hace las acciones es el Modelo.

   El Controlador solo hace de enlace entre la Vista y el Modelo.

   Al hacerlo Singleton solucionamos futuros problemas de threads con la interfaz gráfica (evitando que se creen más de uno), tendremos por tanto un solo Reproductor y un solo Dades...

2. Con la implementación de equals() es bueno hacer también la de hashCode() por si alguna vez hay que usar listas hash.

3. A la hora de **Serializar**, comprendemos que el **reproductor** no debe poder guardarse y le añadimos la etiqueta "**transient**".

4. Solo hace falta hacer **implements Serializable** en: **CarpetaFitxers** y **Dades**. Las clases de los ficheros extienden de File, File ya implementa Serializable, entonces no nos hace falta. Si en vez de implementar Serializable en carpetaFitxers lo hicieramos en BibliotecaFitxersMultimedia perderiamos los atributos del padre y no funcionaría como debe hacerlo.

5. Desde Java 7 existe el Try-Catch con recursos, cualquier clase que implemente Closeable no necesita usar un finally con if y otro Try-Catch para hacer close(), puesto que el Try-catch con recursos lo hará él por sí solo si el objeto cuando lo creamos lo hacemos dentro del paréntesis que acompaña el try.

```java
public void save(String camiDesti) throws AplicacioException {
    File savedFile = new File(camiDesti);
    try (FileOutputStream fout = new FileOutputStream(savedFile);
            ObjectOutputStream oos = new ObjectOutputStream(fout)) {
        oos.writeObject(this);
        oos.flush();
    } catch (IOException ex) {
        throw new AplicacioException("No es pot guardar el fitxer");
    }
}
```

Ejemplo del método guardar de Dades con Try-Catch con recursos de java 7.

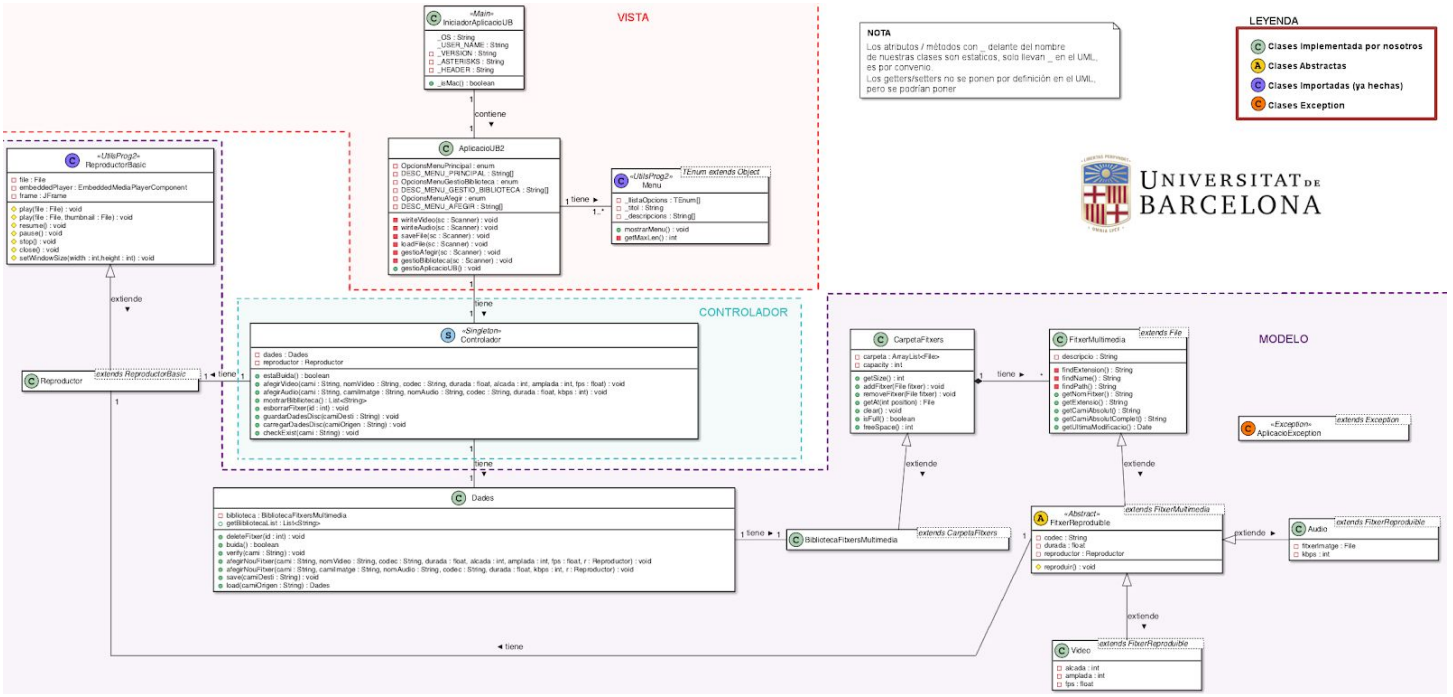
```java
public Dades load(String camiOrigen) throws AplicacioException {
    File loadFile = new File(camiOrigen);
    Dades dades;
    if (!loadFile.exists()) {
        throw new AplicacioException("No existeix el fixter de dades");
    } else {
        try (FileInputStream fis = new FileInputStream(loadFile);
                ObjectInputStream ois = new ObjectInputStream(fis)) {
            dades = (Dades) ois.readObject();
        } catch (IOException ex) {
            throw new AplicacioException("No es pot carregar o no existeix el fitxer");
        } catch (ClassNotFoundException ex) {
            throw new AplicacioException("Error amb les dades");
        }
        return dades;
    }
}
```

Ejemplo del método cargar de Dades con Try-Catch con recursos de java 7.

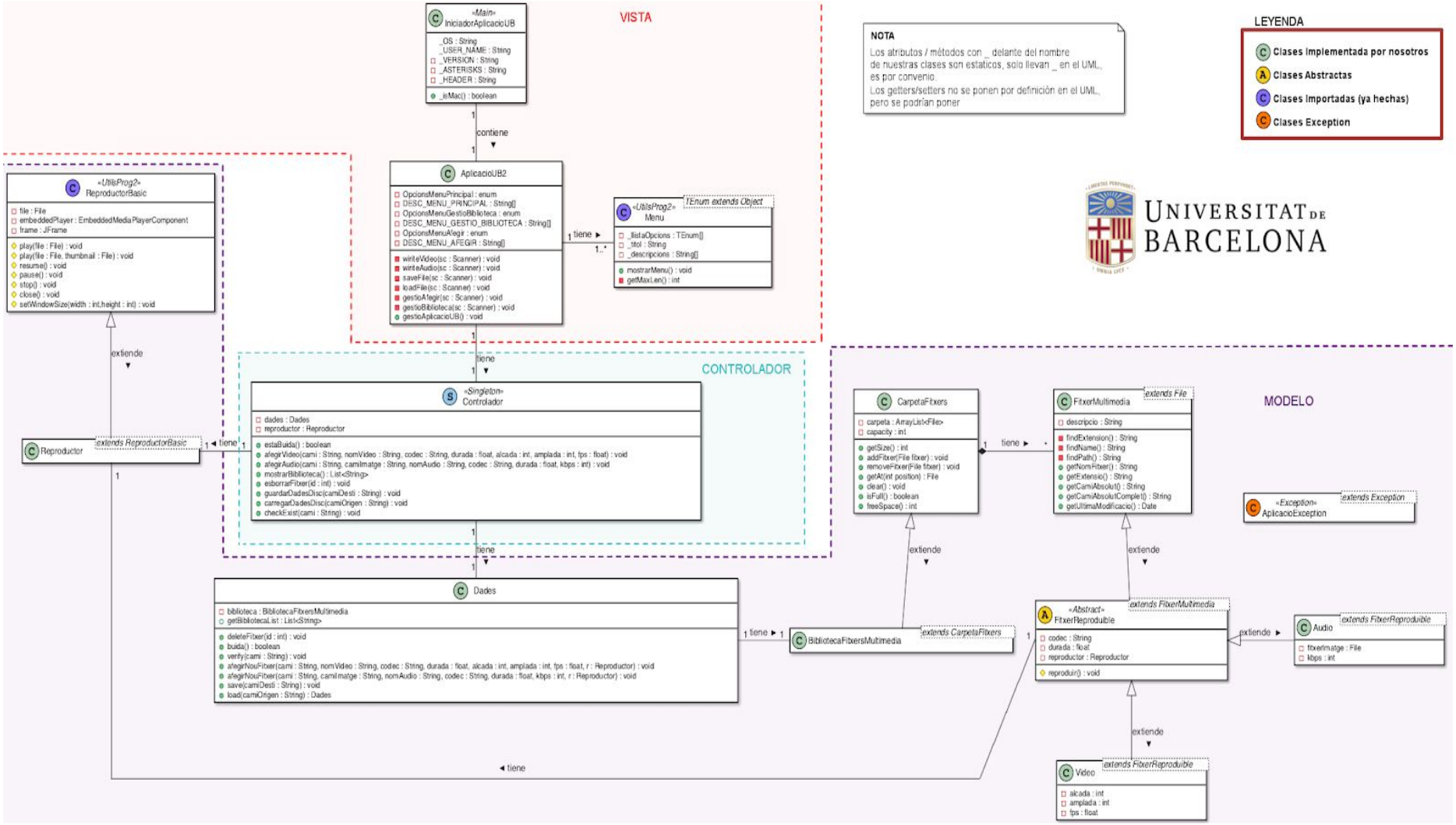
Clases Implementadas

1. **IniciadorAplicacioUB**: Es la clase principal del programa y la que contiene el main, pertenece a la vista.
2. **AplicacioUB2**: Pertenece a la vista y es la clase que contiene el menú (incluido en el **UtilsProg2.jar**) y el controlador con el cual haremos de puente para hacer operaciones en el modelo y mostrarlo en la vista.
3. **CarpetaFitxers**: Es la clase que se encargará de almacenar y destruir los ficheros multimedia en su lista.
4. **BibliotecaFitxersMultimedia**: Hereda de CarpetaFitxers y a diferencia de esta, no hay un máximo de ficheros como límite.
5. **FitxerMultimedia**: Extiende de File, Es la clase que usaremos para asignar un fichero a ésta.
6. **FitxerReproduible**: Extiende de FiletxerMultimedia, es la clase que usaremos para asignar un fichero que puede ser reproducido a ésta, ya que tiene un atributo reproductor y un método reproducir, aunque está no la usaremos explicitamente porque será abstracta.
7. **Audio**: Extiende de FitxerReproduible, Es la clase que usaremos para asignar un fichero de audio a ésta.
8. **Video**: Extiende de FitxerReproduible, Es la clase que usaremos para asignar un fichero de video a ésta.
9. **Reproductor**: Es una clase que hereda de ReproductorBasic, incluido en el **UtilsProg2.jar** la utilizaremos para crear un objeto Reproductor y añadirselo a los ficheros reproducibles.
10. **Dades**: Es la clase donde se gestionará todo lo referente al modelo (añadir/borrar ficheros, guardar/cargar datos, verificar si un archivo existe o si la carpeta de ficheros está vacía), contendrá un objeto de BibliotecaFitxersMultimedia.
11. **Controlador (Singleton)**: Es la clase enlace entre la vista y el modelo, todas las operaciones que hagamos en la vista las ejecutara el controlador y este a su vez como tiene un objeto de clase Dades del modelo llamará al método correspondiente.

Diagrama de Clases UML



(Hecho con PlantUML y Gimp, dentro de docs del proyecto está la imagen por separado)


VISTA
«Main»
IniciadorAplicacioUB
_OS : String
_USER_NAME : String
_VERSION : String
_ASTERISKS : String
_HEADER : String
_isMac() : boolean
contiene
AplicacioUB2
OpcionsMenuPrincipal : enum
DESC_MENU_PRINCIPAL : String[]
OpcionsMenuGestioBiblioteca : enum
DESC_MENU_GESTIO_BIBLIOTECA : String[]
OpcionsMenuAfegir : enum
DESC_MENU_AFEGIR : String[]
writeVideo(sc : Scanner) : void
writeAudio(sc : Scanner) : void
saveFile(sc : Scanner) : void
loadFile(sc : Scanner) : void
gestioAfegir(sc : Scanner) : void
gestioBiblioteca(sc : Scanner) : void
gestioAplicacioUB() : void
tiene
«UtilsProg2»
Menu
TEnum extends Object
_llistaOpcions : TEnum[]
_titol : String
_descripcions : String[]
mostrarMenu() : void
getMaxLen() : int
«UtilsProg2»
ReproductorBasic
file : File
embeddedPlayer : EmbeddedMediaPlayerComponent
frame : JFrame
play(file : File) : void
play(file : File, thumbnail : File) : void
resume() : void
pause() : void
stop() : void
close() : void
setWindowSize(width : int,height : int) : void
extiende
CONTROLADOR
«Singleton»
Controlador
dades : Dades
reproductor : Reproductor
estaBuida() : boolean
afegirVideo(cami : String, nomVideo : String, codec : String, durada : float, alcada : int, amplada : int, fps : float) : void
afegirAudio(cami : String, camiImatge : String, nomAudio : String, codec : String, durada : float, kbps : int) : void
mostrarBiblioteca() : List<String>
esborrarFitxer(id : int) : void
guardarDadesDisc(camiDesti : String) : void
carregarDadesDisc(camiOrigen : String) : void
checkExist(cami : String) : void
Reproductor
extends ReproductorBasic
Dades
biblioteca : BibliotecaFitxersMultimedia
getBibliotecaList : List<String>
deleteFitxer(id : int) : void
buida() : boolean
verify(cami : String) : void
afegirNouFitxer(cami : String, nomVideo : String, codec : String, durada : float, alcada : int, amplada : int, fps : float, r : Reproductor) : void
afegirNouFitxer(cami : String, camiImatge : String, nomAudio : String, codec : String, durada : float, kbps : int, r : Reproductor) : void
save(camiDesti : String) : void
load(camiOrigen : String) : Dades
BibliotecaFitxersMultimedia
extends CarpetaFitxers
MODELO
CarpetaFitxers
carpeta : ArrayList<File>
capacity : int
getSize() : int
addFitxer(File fitxer) : void
removeFitxer(File fitxer) : void
getAt(int position) : File
clear() : void
isFull() : boolean
freeSpace() : int
FitxerMultimedia
extends File
descripcio : String
findExtension() : String
findName() : String
findPath() : String
getNomFitxer() : String
getExtensio() : String
getCamiAbsolut() : String
getCamiAbsolutComplet() : String
getUltimaModificacio() : Date
«Exception»
AplicacioException
extends Exception
«Abstract»
FitxerReproduible
extends FitxerMultimedia
codec : String
durada : float
reproductor : Reproductor
reproduir() : void
Audio
extends FitxerReproduible
fitxerImatge : File
kbps : int
Video
extends FitxerReproduible
alcada : int
amplada : int
fps : float
tiene
extiende
NOTA
Los atributos / métodos con _ delante del nombre de nuestras clases son estaticos, solo llevan _ en el UML, es por convenio.
Los getters/setters no se ponen por definición en el UML, pero se podrían poner
LEYENDA
Clases Implementada por nosotros
Clases Abstractas
Clases Importadas (ya hechas)
Clases Exception
UNIVERSITAT DE BARCELONA

Implementación de equals

La implementación de equals fue hecha en la práctica 1, lo que hicimos fue hacer un **@Override** de los métodos **equals()** y **hashCode()** (evitando así usar los de la SuperClase), este último no venía indicado pero se ha de hacer también, normalmente cuando se hace un equals() de un objeto se ha de redefinir un hashCode() con los mismos atributos que se usaron en equals() y esto es por si los objetos se llegan a utilizar en colecciones hash.

Para definir si dos archivos son iguales lo hemos hecho dentro de equals() y hashCode() con con: **getCamiAbsolut()**, **getNomFitxer()**, **descripcio**, **getExtensio()**, **getUltimaModificacio()**.

En realidad se podría hacer directamente con: getCamiAbsolutComplet(), descripcio y getUltimaModificacio(), (lo hemos hecho de la otra manera porque es como pedía el enunciado).

```
@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final FitxerMultimedia other = (FitxerMultimedia) obj;
    if (!Objects.equals(this.getNomFitxer(), other.getNomFitxer())) {
        return false;
    }
    if (!Objects.equals(this.descripcio, other.descripcio)) {
        return false;
    }
    if (!Objects.equals(this.getExtensio(), other.getExtensio())) {
        return false;
    }
    if (!Objects.equals(this.getCamiAbsolut(), other.getCamiAbsolut())) {
        return false;
    }
    return Objects.equals(this.getUltimaModificacio(), other.getUltimaModificacio());
}
```

Método equals sobreescrito.

```java
@Override
public int hashCode() {
    int hash = 3;
    hash = 83 * hash + Objects.hashCode(this.getCamiAbsolut());
    hash = 83 * hash + Objects.hashCode(this.getNomFitxer());
    hash = 83 * hash + Objects.hashCode(this.descripcio);
    hash = 83 * hash + Objects.hashCode(this.getExtensio());
    hash = 83 * hash + Objects.hashCode(this.getUltimaModificacio());
    return hash;
}
```

Método hashCode() sobreescrito.

Objetos creados en el Main

Al crear un objeto en el main este creará otros que a su vez crearán otros de esta manera:

La clase principal es IniciadorAplicacioUB, dentro del main se crea un objeto **AplicacioUB2**, éste llamará al método getInstance() del **Controlador**, si no está creado el Controlador hará un new Controlador internamente ya que Controlador lo hemos hecho singleton, si ya estaba creado solo llamará a su instancia.

El Controlador en caso de ser creado creará un objeto de la clase **Dades** y un objeto de la clase **Reproductor**.

Dades al crearse creará un objeto de la clase **BibliotecaFitxersMultimedia** y BibliotecaFitxersMultimedia creara un **ArrayList** de ficheros.

A su vez el objeto de tipo AplicacioUB2 llama al método gestioAplicacioUB() y creará otros objetos: uno de la clase **Scanner** para introducir por teclado, tres de la clase **Menu**.

Simplificando:

> **AplicacioUB2, Controlador, Dades, Reproductor, BibliotecaFitxersMultimedia, Scanner, Menu.**

Hay que tener en cuenta que las subclases llaman al ser creadas al constructor de la superclase.

Uso de AplicacioException

En la práctica anterior ya teníamos una clase dedicada para nuestras excepciones, en esta práctica lo que hemos hecho ha sido sustituirla por la que nos han dado.

Usamos la clase AplicacioException desde el modelo para propagar hasta la vista excepciones tales como:

Ya hemos añadido el fichero, El fichero no existe, Nombre de fichero inválido, Índices de la posiciones de ficheros en la biblioteca (para acceder y borrar), Cargar y Guardar datos Serializados.

Las excepciones se lanzan por el Modelo, llegan al Controlador pero este no las trata y las lanza a la vista.

En la vista las tratamos con Try-Catch.

Sobrecarga de métodos en el Controlador

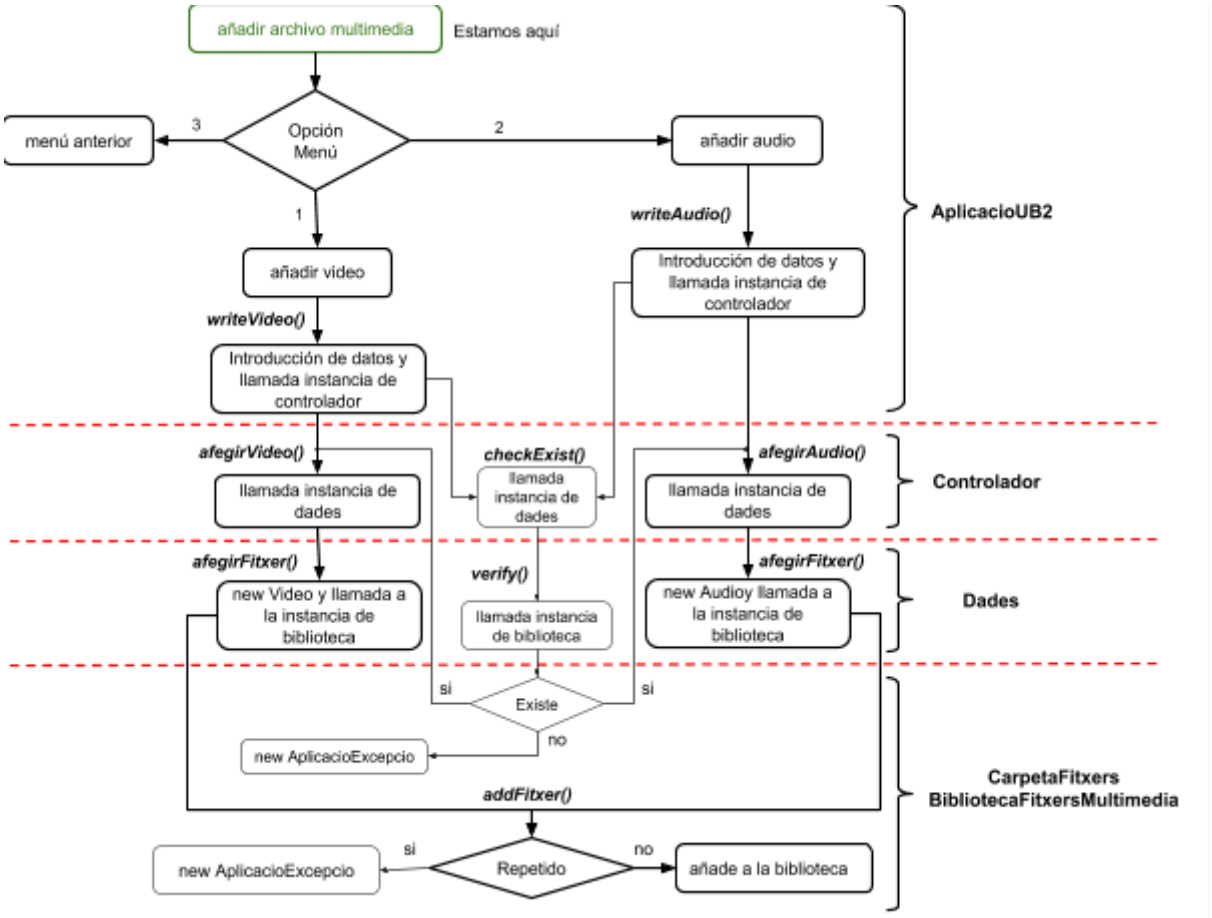
Dado que nuestros métodos tienen el nombre y la firma tal cual está en el enunciado del pdf, no tenemos sobrecarga de métodos.

hasta donde sabemos la sobrecarga de métodos es tener métodos que se llaman igual donde varía solo su firma (parámetros, ...).

Se podría usar para tener dos métodos que se llamarán afegirFitxerMultimedia(), donde dependiendo de los parámetros de creación se llamará al método correspondiente de Dades y añadiera un fichero de audio o video.

Diagrama de Flujo - Añadir Fichero

Cuando damos a “Afegir fitxer multimedia a la biblioteca” empezaremos el siguiente flujo:



Hemos tenido que simplificar en el flujo la comprobación de si el fichero existe y sacar dobles líneas, ya que no nos caben más elementos, pero esta todo correcto.

Cambios para ejecutar fr.reproduir();

FitxerReproduible fr = new FitxerReproduible(repro);
fr.reproduir();

La clase FitxerReproduible es abstracta y su método reproduir() también lo es.

El método lo tenemos así: **protected abstract void reproduir();**

La solución sería que el método no sea abstracto y se realizará su implementación, habría que tener en cuenta la visibilidad de la clase y del método, para que en Dades nos deje crear un FitxerReproduible y a su vez pudiéramos reproducirlo.

Para detectar si es audio o video podemos hacerlo con la extensión y así en un if-else hacer el print correspondiente.

O mejor aún con sobrecarga de constructores, donde dependiendo de los parámetros que le demos al hacer el new FitxerReproduible asigna unos valores que luego en reproduir() se tendrán en cuenta.

Resultados

Hemos hecho pruebas y aquí ponemos algunos ejemplos:

1. Intentar guardar datos con la carpeta vacía (no deja).

```
MENU PRINCIPAL
-------------------------------
        1.- Gestió Biblioteca
        2.- Guardar Dades
        3.- Recuperar Dades
        4.- Sortir
-------------------------------
Entra una opcio >> 2

No hi ha dades que guardar
```

2. Eliminar fichero con carpeta vacía (no deja).

```
-----------------------------------------------------
GESTIÓ BIBLIOTECA
-----------------------------------------------------
        1.- Afegir fitxer multimedia a la biblioteca
        2.- Mostrar Biblioteca
        3.- Elimina fitxer multimédia
        4.- Menu Anterior
-----------------------------------------------------
Entra una opcio >> 3

No hi ha fitxers per esborrar
```

3. Eliminar con ID incorrecto (salta la excepción).

```
-----------------------------------------------------
GESTIÓ BIBLIOTECA
-----------------------------------------------------
        1.- Afegir fitxer multimedia a la biblioteca
        2.- Mostrar Biblioteca
        3.- Elimina fitxer multimédia
        4.- Menu Anterior
-----------------------------------------------------
Entra una opcio >> 3

ID del Fitxer:
5
La posició no pot ser més gran que: 2
mar 27, 2018 1:24:32 AM edu.ub.prog2.utils.AplicacioException <init>

-----------------------------------------------------
GRAVE: La posició no pot ser més gran que: 2
```

4. Añadir archivo que no existe (salta la excepción).

```
---------------------------
AFEGIR FITXER MULTIMEDIA A LA BIBLIOTECA
---------------------------
        1.- Afegir Video
        2.- Afegir Audio
        3.- Menu Anterior
---------------------------
Entra una opcio >> 1

Ruta del Fitxer:
oooo/t.mp3
El fitxer t.mp3 no existeix
mar 27, 2018 1:33:23 AM edu.ub.prog2.utils.AplicacioException <init>

---------------------------
GRAVE: El fitxer t.mp3 no existeix
```

5. Añadir la imagen inexistente al Audio (salta la excepción).

```
AFEGIR FITXER MULTIMEDIA A LA BIBLIOTECA
---------------------------
        1.- Afegir Video
        2.- Afegir Audio
        3.- Menu Anterior
---------------------------
Entra una opcio >> 2

Ruta del Fitxer:
prueba/audioDummy.mp3
Ruta de la imatge del Fitxer:
pp/gg.png
mar 27, 2018 1:34:52 AM edu.ub.prog2.utils.AplicacioException <init>
El fitxer gg.png no existeix

GRAVE: El fitxer gg.png no existeix
```

6. Recuperar datos de un fichero que no existe (salta la excepción).

```
-------------------------------
MENU PRINCIPAL
-------------------------------
        1.- Gestió Biblioteca
        2.- Guardar Dades
        3.- Recuperar Dades
        4.- Sortir
-------------------------------
Entra una opcio >> 3

Ruta completa amb el nom:
ee
No existeix el fixter de dades

mar 27, 2018 1:20:47 AM edu.ub.prog2.utils.AplicacioException <init>
-------------------------------
GRAVE: No existeix el fixter de dades
```

7. Borrado exitoso de un fichero

```
GESTIÓ BIBLIOTECA
-----------------------------------------------------
        1.- Afegir fitxer multimedia a la biblioteca
        2.- Mostrar Biblioteca
        3.- Elimina fitxer multimédia
        4.- Menu Anterior
-----------------------------------------------------
Entra una opcio >> 2

Carpeta de Fitxers:
-----------------------------------
[1] Tipus='Video', Descripció='Video vacio Test', Data='Tue M
[2] Tipus='Audio', Descripció='Audio test', Data='Tue Mar 27

-----------------------------------------------------
GESTIÓ BIBLIOTECA
-----------------------------------------------------
        1.- Afegir fitxer multimedia a la biblioteca
        2.- Mostrar Biblioteca
        3.- Elimina fitxer multimédia
        4.- Menu Anterior
-----------------------------------------------------
Entra una opcio >> 3

ID del Fitxer:
2
Fitxer amd id 2 esborrat
```

En la carpeta prueba tenemos dos dummys y una imagen: audioDummy.mp3, videoDummy.mp4, img.png. Para hacer pruebas.