REPRODUCTOR MULTIMÈDIA

UNIVERSITAT DE BARCELONA

Blai Ras Jimenez

Programació II

Lliurament 4

5/06/2016

Índex

1. Introducció

Arriba la quarta i última entrega del nostre reproducció en la que no programar més característiques en sí sinó que programem una cosa igual o més important: la seva interfície gràfica, és a dir, la interfaç del nostre reproductor que conté tot el que hem estat programant les anteriors entregues.

Així doncs, serà el NetBeans un altre cop la nostra eina que mitjançant la llibreria d'interfícies gràfiques Swing ens permetrà introduir-nos en la programació orientada a events i dissenyar cóm serà visualment en nostre reproductor.

Tot i que només li posem ulls i cara al nostre reproductor, haurem de programar codi que permeti dur a terme les accions que fins ara fèiem mitjançant la consola i mitjançant números, i inclús modificar o afegir codi a classes de lliuraments anteriors

2. *Explicació de les classes implementades*

Aquí s'inclou una llista de les classes noves o que no han estat modificades respecte la tercera entrega. Totes aquelles classes que no apareixen per tant no han estat modificades i la seva explicació està en memòries anteriors.

a. *Controlador:* Controlador continua sent la classe que gestiona l'accés a les dades. Com que ara aquests accés es gestiona des del paquet "vista" que ara és una interfície gràfica, canvia. Per tant, Controlador també. Així doncs, les seves modificacions han sigut:

- **i. *getPanel():*** aquest mètode és necessari per obtenir el panell on es troba el nostre reproductor, per lo que només és una simple crida al mètode ja implementat en les llibreries donades getPanel().
- **ii. *getAlbums():*** aquest mètode retorna l'Array d'Albums que es troba en la classe dades.
- **iii. *getAlbumIndex(String titol):*** aquest mètode retorna la posició d'un àlbum obtingut mitjançant el seu títol en l'Array d'Albums. És per tant una simple crida al mètode indexAlbum() de la classe Dades.
- **iv. *esborrarPath(String path):*** aquest mètode esborra un fitxer mitjançant el seu camí. Fins ara, només podíem esborrar mitjançant una referència d'aquell fitxer a esborrar. És per tant una simple crida al nou mètode esborrarPath() de Dades.

```
public void esborrarPath(String path) throws AplicacioException {
    File fitxer = new File(path);
    biblio.removeFitxer(fitxer);
    for (int z = 0; z < albums.size(); z++) {
        if (albums.get(z).contains(fitxer)) {
            for (int i = 0; i < albums.get(z).getMida(); i++){
                albums.get(z).removeFitxer(fitxer);
            }
        }
    }
}
```

v. ***reproduirFitxer() i reproduirFitxerAlbum():*** aquets mètodes s'encarreguen de fer la reproducció d'un únic fitxer. A la pràctica 3 simplement cridava al mètode mostrar o reproduir pròpiament dic, en comptes d'enviar una carpeta amb aquell únic fitxer a reproduir. Així doncs, en aquesta entrega ho he modificat.
También he afegit el mètode reproduirFitxerAlbum(File fitxer) que té la mateixa funcionalitat que reproduirFitxer(int id) però rep el fitxer del àlbum que es vol reproduir, en comptes de la posició. Això es degut a que la posició del fitxer en l'àlbum pot ser diferent a la de la biblioteca. El que faig doncs és agafar el fitxer per paràmetre i afegir-lo a una nova carpeta, per després enviar-la a inicarReproducció():

```
@Override
public void reproduirFitxer(int i) throws AplicacioException {
    CarpetaFitxers carpeta = new CarpetaFitxers();
    i--; //Decremento ja que comenso a mostrar per 1
    carpeta.addFitxer(dades.getCarpeta().getAt(i));
    if (!escoltador.esticPubli()) {
        escoltador.iniciarReproduccio(carpeta, dades.getCiclic());
    } else {
        throw new AplicacioException("Espera a que acabi la publicitat");
    }

}

public void reproduirFitxerAlbum(File fitxer) throws AplicacioException {
    CarpetaFitxers carpeta = new CarpetaFitxers();
    carpeta.addFitxer(fitxer);

    if (!escoltador.esticPubli()) {
        escoltador.iniciarReproduccio(carpeta, dades.getCiclic());
    } else {
        throw new AplicacioException("Espera a que acabi la publicitat");
    }
}
```

b. ***AlbumFitxersMultimedia:*** aquesta classe contínua sent la representació d'un àlbum del nostre reproductor, i per tant conté les seves propietats. En aquesta entrega s'ha modificat en 2 nous mètodes:
   i. ***removeFitxerIndex(inti d):*** aquest mètode esborra un fitxer mitjançant la seva posició en l'Array de FitxersMultimedia. És una simple crida al mètode que ens dona Java "remove(int i)".
   ii. ***isEmpty():*** com el seu nom indica, aquest mètode comprova si un àlbum està buit o no. Fins ara no m'havia sigut necessari. És una simple comprovació de si l'Array de fitxers és igual a zero.

c. ***CarpetaFitxers:*** Aquesta classe continua sent el nostre contendir de fitxers multimèdia mitjançant un ArrayList, per lo que el seu accés ara que tenim una interfície gràfica canvia. Ho fa amb els següents mètodes:

   i. ***getPos(File fitxer):*** aquest mètode retorna un enter corresponent a la posició del fitxer passat per paràmetre en l'ArrayList. Ho fa recorrent tot l'ArrayList i comprovant el fitxer en qüestió amb el passat per paràmetre. Si el troba, retorna la posició, sinó, retorna -1.

```
public int getPos(File fitxer) {
    int pos=-1;
    for (int i = 0; i<taulaFitxers.size(); i++) {
        if (taulaFitxers.get(i).equals(fitxer)) {
            pos = i;
        }
    }
    return pos;
}
```

   ii. ***isEmpty():*** aquest mètode comprova si l'ArrayList de Fitxers està buit. Ho fa amb la comprovació de la seva mida igual a zero.

d. ***Dades:*** aquesta classe continua sent la nostra gestionadora dels continguts pròpiament dits del nostre reproductor, i l'he modificat simplement fent un mètode nou que esborra un fitxer mitjançant el seu camí, ja que fins ara esborràvem mitjançant una referència d'aquest fitxer. Ho faig creant un el fitxer en qüestió mitjançant el seu path per després cridar el mètode remove(File file). A continuació, l'esborro també de qualsevol àlbum en que estigui afegit:

```
public void esborrarPath(String path) throws AplicacioException {
    File fitxer = new File(path);
    biblio.removeFitxer(fitxer);
    for (int z = 0; z < albums.size(); z++) {
        albums.get(z).removeFitxer(fitxer);
    }
}
```

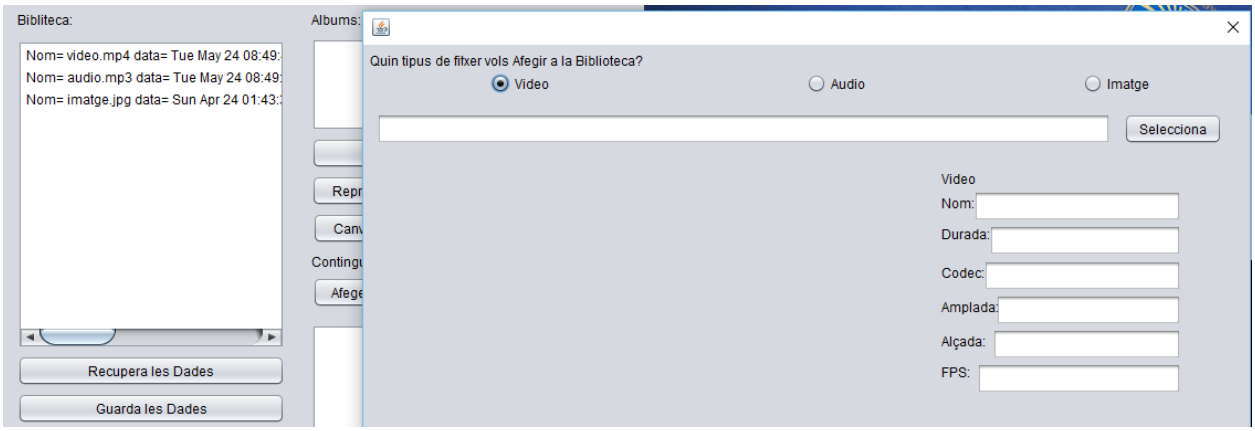
e. ***FrmAfegirFitxerMultimedia:*** arribem ara a la primera "classe" d'interfície gràfica, un JDialog que permet l'adició de fitxers a la biblioteca. Ara, en comptes de demanar nosaltres les dades o que l'usuari entri el camí del fitxer, serà ell el que les introduirà, mitjançant uns JTextBox i un JFileChooser. En primer lloc, el JFileChooser obrirà una finestra del sistema on veurem els nostres fitxers i podrà seleccionar el desitjat, i en segon lloc la classe estarà formada per camps de text on l'usuari podrà entrar les dades desitjades del fitxer.

Concretament, la classe estarà formada per tres jRadioButtons corresponents als tres tipus d'arxius suportats, Audio, Video i Imatges. Al seleccionar algun d'ells (no es permetrà la selecció de més d'un d'ells) s'obrirà el panell corresponent amb les etiquetes adequades pel tipus de fitxer seleccionat. Per tant, cadascun d'ells tindrà un event determinat que al ser seleccionat faran visible el panell adecuat, a part de mostrar el botó selecciona que obra el JFileChooser.

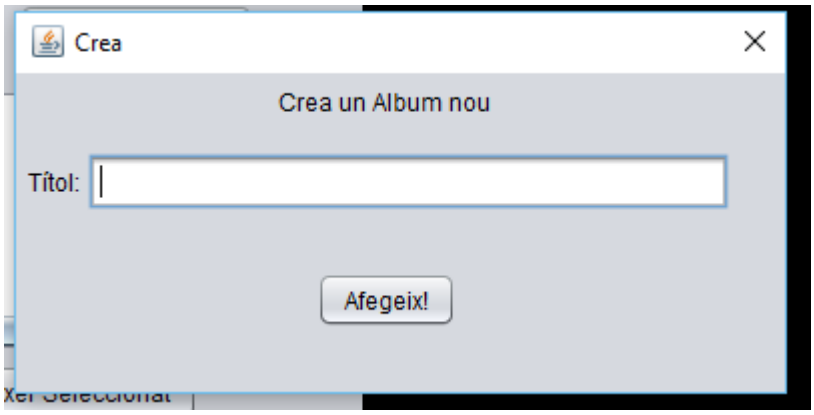
```
private void btnAcceptaActionPerformed(java.awt.event.ActionEvent evt) {

    if (activarAudio.isSelected()) {
        if (txtCami.getText().equals("") || txtCamiCaratula.getText().equals("") || txtNomAudio.getText().equals("") || txtNomCodec.getText().equals("") || txtNomD
            JOptionPane.showMessageDialog(rootPane,"Completa tots els camps abans d'afegir l'Audio!","Error",JOptionPane.ERROR_MESSAGE);
        } else {
            float durada = Float.parseFloat(txtNomDurada.getText());
            int kpbs = Integer.parseInt(txtNomKpbs.getText());
            try {
                controlador.afegirAudio(txtCami.getText(), txtCamiCaratula.getText(), txtNomAudio.getText(), txtNomCodec.getText(), durada, kpbs);
            } catch (AplicacioException ex) {
                JOptionPane.showMessageDialog(rootPane, "Error al intentar afegir un Audio: "+ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            }

        this.dispose();
        }
    }
```

Un cop l'usuari ha introduït les dades, s'apreta el boto "Afegeix", de manera que succeirà l'event d'afegir un fitxer, el quan no es res més que una simple crida al mètode de Controlador (passat per paràmetre) d'afegirAudio, Video o Imatge, el qual s'abrem amb 3 if's de condició.

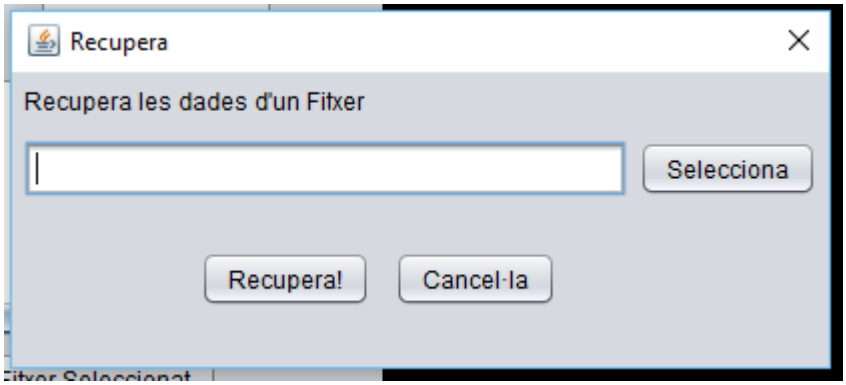
Tot i això, hi ha camps que son d'enters, altres de string i altres de Float, per lo que l'obtenció del seu text es fa amb el mètode parse de la superclasse Float o Integer. A continuació, es tancarà el JDialog amb this.dispose().



f. ***MenuCrearAlbum:*** aquesta classe és un altre JDialog que permet l'adició de nous àlbum al reproductor. És més simple que l'anterior, ja només se'ns obra una finestra amb un camp de text corresponent al títol del àlbum, i es que no necessitem res mes.
També consta d'un botó "Afegeix", que realitza aquesta inserció. Ho fa amb el mètode afegirAlbum de Controlador, sempre i quan abans hagi comprovat que el camp de text no estigui buit. També conté un botó cancel·la, el qual tanca la finestra amb la comanda dispose().



g. ***MenuGuardar i MenuRecuperar:*** aquestes classes son dos JDialogs més que permeten el guardat i la recuperació de dades. Els dos integren un camp de text i un botó de selecció corresponent a on es troba l'arxiu de recuperació o guardat de dades i un altre botó "Guarda" o "Recupera" en cada cas que realitza aquesta acció. També contenen un botó "Cancel·la" que simplement tanca la finestra en ambdós casos.

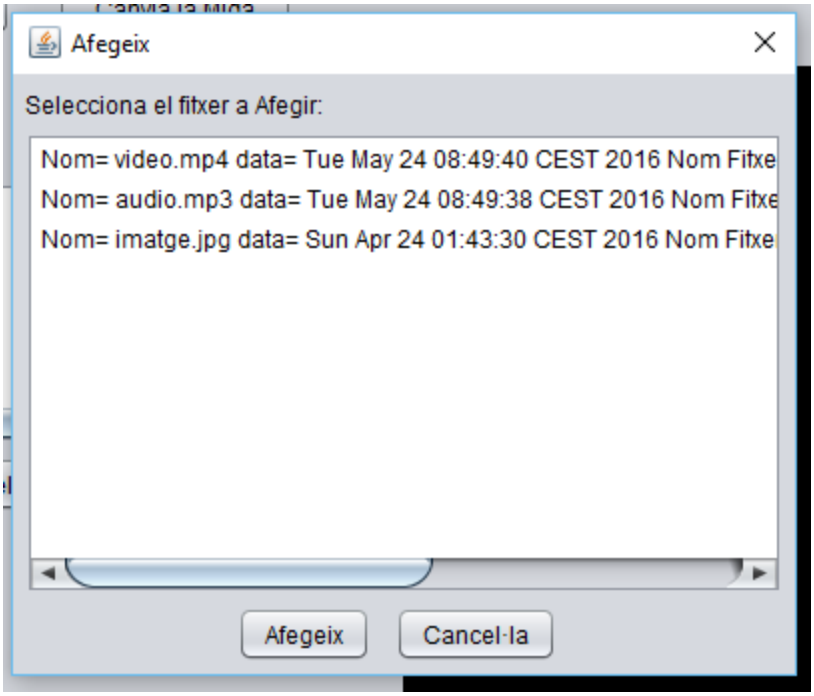


Així doncs, l'acció es realitza amb una crida al mètode de Controlador guardarDadesDisc o recuperarDadesDisc, sempre i quan l'usuari hagi seleccionat abans el camí a guardar/recuperar.

h. ***seleccioFitxersAfegir:*** aquest últim JDialog és una finestra en que es mostra el contingut de la biblioteca per poder seleccionar el fitxer desitjat i afegir-lo a un àlbum concret.

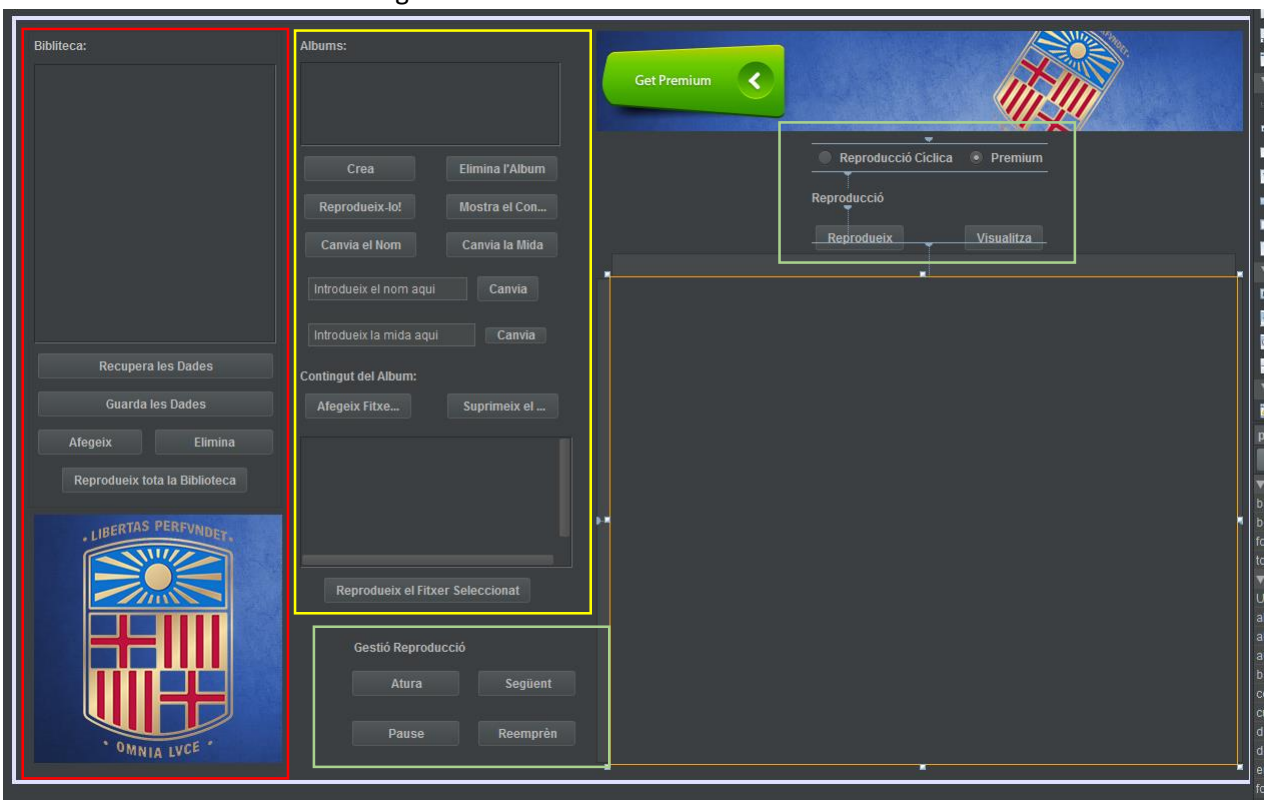
La crida es fa quan es selecciona el botó d'afegir contingut al àlbum, sempre i quan es seleccioni un d'ells en la llista. Llavors, s'obre aquesta classe amb el contingut de la biblioteca de manera que cada arxiu és una posició de la llista.

El JDialog tan sols conté un botó, "Afegeix", que insereix el fitxer seleccionat en l'àlbum, sempre i quan es seleccioni un prèviament. Per mostrar el contingut de la biblioteca realitzo un mètode omplirLlista() que obté la biblioteca mitjançant el controlador obtingut per paràmetre i creant un model on se li van afegint els arxius que rep de la biblioteca.



L'obtenció del àlbum es fa per paràmetre, és a dir, es crida aquesta classe amb el paràmetre Controlador controlador i AlbumFitxersMultimedia album. Per tant, al clicar afegir, es llença un event en el que simplement es crida el mètode de Controlador addFitxer de AlbumFitxerMultimedia.

i. ***AplicacioUB4:*** aquesta és la classe mare del nostre reproductor, la que conté el nostre main i la principal finestra de la nostre interfície gràfica. S'ha suprimit, per tant, InciadorAplicacioUB i ara aquesta classe ja no conté cap dels mètodes de les anteriors entregues.



En vermell: Panell de la Biblioteca
En groc: Panell dels Àlbums
En verd: Panell de la reproducció

Es tracta d'una classe JFrame, que jo he organitzat en 3 panells: biblioteca, àlbums i reproductor.

i. <u>Panell Biblioteca:</u> aquest panell conté la llista d'elements afegits a la biblioteca, un botó de recuperar dades, un altre d'afegir dades i finalment dos botons per inserir o eliminar fitxers.

   Recuperar/Guardar dades simplement obren els seus corresponents JDialogs sempre i quan la biblioteca no estigui buida. El botó afegeix igual, obre el seu JDialog.

El botó elimina ja és més complicat, ja que elimina de la biblioteca i del àlbum. Primer miro si l'usuari ha afegit fitxers prèviament i si hi ha seleccionat el fitxer a eliminar. Si es compleix, passo a obtenir els elements seleccionats amb getSelectedValues() per després cridar l'item seleccionat de llista amb el mètode esborrarPath(), obtenint el seu path amb el mètode getAbsolutePath de File.

```
private void btnEliminaActionPerformed(java.awt.event.ActionEvent evt) {
    if(controlador.getBiblio().isEmpty()) {
        JOptionPane.showMessageDialog(rootPane, "Afegeix algun fitxer abans d'esborrar!","Error",JOptionPane.ERROR_MESSAGE);
    } else {
        if (llistaBiblioteca.getSelectedIndex() == -1) {
            JOptionPane.showMessageDialog(rootPane, "Selecciona un fitxer abans d'eliminar-lo!","Error",JOptionPane.ERROR_MESSAGE);
        } else {
            for (Object item: llistaBiblioteca.getSelectedValues()) {
                try {
                    FitxerMultimedia element = (FitxerMultimedia) item;
                    controlador.esborrarPath(element.getAbsolutePath()); //PHGDOSF
                } catch (AplicacioException ex) {
                    JOptionPane.showMessageDialog(rootPane, "Error al intentar esborrar el fitxer: "+ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
                }
            }
            omplirLlistatBiblioteca();
            omplirContingutAlbum(etiquetaContingutAlbum.getText());
        }
    }
}
```

Després, és important refrescar les llistes tant de biblioteca com de àlbum, amb els mètodes omplirContingutAlbum() i omplirLlistatBiblioteca(). OmplirLlistatBiblioteca obté l'Array i per cada element el va afegint dins d'un nou model, mentre que omplirContingutAlbum obté l'àlbum mitjançant el títol (que es troba en una etiqueta) i itera la llista que rep del mètode mostrarAlbum(), en la que cada element l'anirà a afegint a un nou model.

```
public void omplirLlistatBiblioteca() {
    DefaultListModel model = new DefaultListModel();
    model.clear();
    for (FitxerMultimedia item: controlador.getBiblio().getLlista()){
        model.addElement(item);

    }
    llistaBiblioteca.setModel(model);
}
```

Per últim, el seu botó "Reprodueix tota la carpeta" comprova abans si la biblioteca està buida, en cas de que no ho estigui, crida a reproduirCarpeta() a la vegada que fa visible el panell de la gestió de la reproducció.

ii. Panell Àlbums: aquest panell conté tot allò relacionat amb un àlbum, les seves opcions i el seu contingut. Conté dos llistes, una per mostrar els diferents àlbums i una altre per mostrar el contingut d'un àlbum. A sota dels àlbums, tenim els botons... :

1. *Crea:* crea un nou àlbum. Es llança el JDialog explicat anteriorment.

2. *Elimina la Àlbum:* elimina l'àlbum seleccionat, mitjançant el mètode esborrarAlbum

3. *Reprodueix:* reprodueix tot un àlbum seleccionat. S'obté l'àlbum amb la posició en la seva llista i després el mètode get() de carpetaFitxers. Un cop obtingut, es crida el mètode reproduirCarpeta amb paràmetre (títol).

4. *Mostra el contingut:* mostra el contingut d'un àlbum en la llista inferior. Obté l'àlbum de la mateixa manera i a continuació itera sobre la llista que rep del mètode mostrarAlbum, i així cada element el va inserint en un nou model que afegiré en la llistaContingutAlbum. També estableixo el text d'una etiqueta amb el seu títol per saber de quin àlbum estem parlant.

```
private void mostraContingutActionPerformed(java.awt.event.ActionEvent evt) {
    if (controlador.getAlbums().isEmpty()) {
        JOptionPane.showMessageDialog(rootPane, "Afegeix un album abans de mostrar el seu contingut!","Error",JOptionPane.ERROR_MESSAGE);
    } else {
        int pos = llistaAlbums.getSelectedIndex();
        DefaultListModel model = new DefaultListModel();
        model.clear();
        AlbumFitxersMultimedia album = controlador.getAlbums().get(pos);
        List<String> llista;
        try {

            llista = controlador.mostrarAlbum(album.getTitol());
            Iterator itr = llista.iterator();
            while (itr.hasNext()) {
                model.addElement(itr.next());
            }
        } catch (AplicacioException ex) {
            JOptionPane.showMessageDialog(rootPane, "No s'ha pogut mostrar el contingut del Album "+album.getTitol()+" : "+ex.getMessage(),"Error",JOptionPane
        }
        llistaContingutAlbum.setModel(model);
        etiquetaContingutAlbum.setText(album.getTitol());
    }
}
```

5. *Canvia el nom:* canvia el nom d'un àlbum seleccionat. Obre un nou panell on es mostra un camp de text, en el que l'usuari introduirà el nou títol. Es fa mitjançant el mètode canviarTitolAlbum, obtenint el títol vell del element seleccionat i el nou del camp de text mostrat. Després, el panell torna a ser invisible.

```
private void canviaNomActionPerformed(java.awt.event.ActionEvent evt) {
    if (controlador.getAlbums().isEmpty()) {
        JOptionPane.showMessageDialog(rootPane, "Afegeix un album abans de canviar-li el nom!","Error",JOptionPane.ERROR_MESSAGE);
    } else {
        panellCanviNom.setVisible(true);
    }
}

private void btnCanviaMidaActionPerformed(java.awt.event.ActionEvent evt) {

    for(Object item:  llistaAlbums.getSelectedValues()){
            AlbumFitxersMultimedia elementSeleccionat = (AlbumFitxersMultimedia) item;
            try {
                int mida = Integer.parseInt(txtCanviMida.getText());
                controlador.canviarMidaAlbum(elementSeleccionat.getTitol(), mida);
                panellCanviMida.setVisible(true);
            } catch (AplicacioException ex) {
                JOptionPane.showMessageDialog(rootPane, "Error al canivar la mida del Album: "+ex.getMessage(),"Error",JOptionPane.ERROR_MESSAGE);
            }
    }
    omplirLlistaAlbums();
    panellCanviMida.setVisible(false);
}
```

6. *Canvia la mida:* canvia la mida d'un àlbum seleccionat. S'obre un panell invisible amb un camp de text on l'usuari entrarà l'enter desitjat. Es fa mitjançant el mètode canviarMidaAlbum, enviant el títol de l'element seleccionat i la mida obtinguda pel camp de text per paràmetre.

7. *Afegeix Fitxer:* afegeix fitxer en l'àlbum seleccionat. Simplement obre el JDialog explicat anteriorment.

8. *Suprimeix fitxer:* elimina un fitxer afegit en un àlbum determinat. Obtinc la posició del fitxer amb el mètode getSelectedIndex() i l'àlbum pel text de l'etiqueta. Després, es crida el mètode esborrarFitxer amb aquests dos paràmetres, i es refresca la llista amb el mètode omplirContingutAlbum(string títol):

```
private void btnSuprimeixAlbumActionPerformed(java.awt.event.ActionEvent evt) {
    if (controlador.getAlbums().isEmpty()) {
        JOptionPane.showMessageDialog(rootPane, "Afegeix un album abans de suprimir un fitxer contingut!","Error",JOptionPane.ERROR_MESSAGE);
    } else {
        int pos = llistaContingutAlbum.getSelectedIndex();
        try {
            controlador.esborrarFitxer(etiquetaContingutAlbum.getText(), pos);
        } catch (AplicacioException ex) {
            JOptionPane.showMessageDialog(rootPane, "No s'ha pogut eliminar l'Album "+etiquetaContingutAlbum.getText()+" : "+ex.getMessage(),"
        }
        omplirContingutAlbum(etiquetaContingutAlbum.getText());
    }
}
```

9. *Reproduiex el Fitxer Seleccionat:* reprodueix el fitxer seleccionat en la llista del contingut d'un àlbum. Obtinc la posició del fitxer amb el mètode getSelectedIndex(), i l'àlbum mitjançant el mètode getAlbumIndex() explicat anteriorment.
   Tot això, per obtenir el fitxer fent controlador.getAlbums().get(posició del album en qüestió).getAt(posició del fitxer en la llista).
   A continuació, crido al mètode reproduirFitxerAlbum(File fitxer) que agafarà aquest fitxer, el posarà a una carpeta i cridarà a iniciarReproducció():

```
private void btnReproFitxerAlbumActionPerformed(java.awt.event.ActionEvent evt) {

    if (controlador.getAlbums().isEmpty())  {
        JOptionPane.showMessageDialog(rootPane, "Afegeix un album abans de reproduir un fitxer contingut!","Error",JOptionPane.ERROR_MESSAGE);
    } else {
        if (llistaContingutAlbum.getSelectedIndex() == -1) {
            JOptionPane.showMessageDialog(rootPane, "Selecciona abans un fitxer a reproduir!","Error",JOptionPane.ERROR_MESSAGE);
        } else {

            int id = llistaContingutAlbum.getSelectedIndex();
            int posAlbum = controlador.getAlbumIndex(etiquetaContingutAlbum.getText());

            FitxerMultimedia fitxer = (FitxerMultimedia) controlador.getAlbums().get(posAlbum).getAt(id);

            try {
                controlador.reproduirFitxerAlbum(fitxer);
                panellGestioReproduccio.setVisible(true);
            } catch (AplicacioException ex) {
                JOptionPane.showMessageDialog(rootPane, "Error al intentar reproduir el fitxer: "+ex.getMessage(),"Error",JOptionPane.ERROR_MESSAGE);
            }
        }
    }
}
```

iii. Panell Reproducció: en aquest panell gestiono tot el que està relacionat amb la reproducció. Continc el reproductor en sí i dos panells més, un amb les opcions de reproductor i un altre amb els comandaments.

1. *Panell Gestió Reproducció:* conté els botons d'aturar, següent, pause i reemprèn. Cadascun simplement és una crida al seu mètode corresponent a Controlador, que cridarà al reproductor a dur a terme l'acció. Aquest panell només es mostra quan es vol reproduir.

```
private void btnAturaActionPerformed(java.awt.event.ActionEvent evt) {

    controlador.aturaReproduccio();
}
```

2. *Panell Reproducció Opcions:* conté les opcions de Premium, Reproducció cíclica, reproduir un fitxer i visualitza fitxer. Premium i Ciclica són dos radioButtons que activaran o desactivaran aquestes característiques. Premium, al ser desactivat, ja no es pot tornar a seleccionar, així l'estabiré a enable(false) després de la seva primera selecció.

Cíclica si que pot activar-se i desactivar-se, així que tan l'un com l'altre criden als seus respectius mètodes de controlador, activarModeCiclic() i desactivarModePremium().

```
private void opcioPremiumActionPerformed(java.awt.event.ActionEvent evt) {

    controlador.desactivarModePremium();
    fotoPubli.setVisible(true);
    opcioPremium.setEnabled(false);
}
```

El botó Reprodueix reprodueix un element seleccionat en la llista de la biblioteca. Abans, comprova que el fitxer estigui seleccionat a la llista, i que la biblioteca no estigui buida.

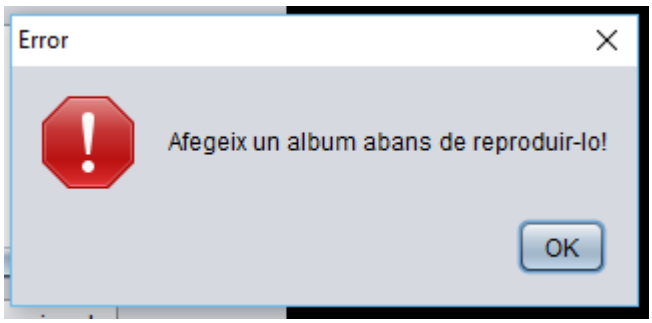
Obtenim la posició del fitxer en la biblioteca mitjançant getSelectedIndex(), per després cridar al mètode reproduirFitxer(enter), a la vegada que fem visible el panell de la gestió de la reproducció.

El botó visualitza té la mateixa funció però per imatges, de manera que el procediment és el mateix però amb el mètode mostrarFitxer() amb un temps il·limitat, a la vegada que fem visible el panell de la gestió de la reproducció.

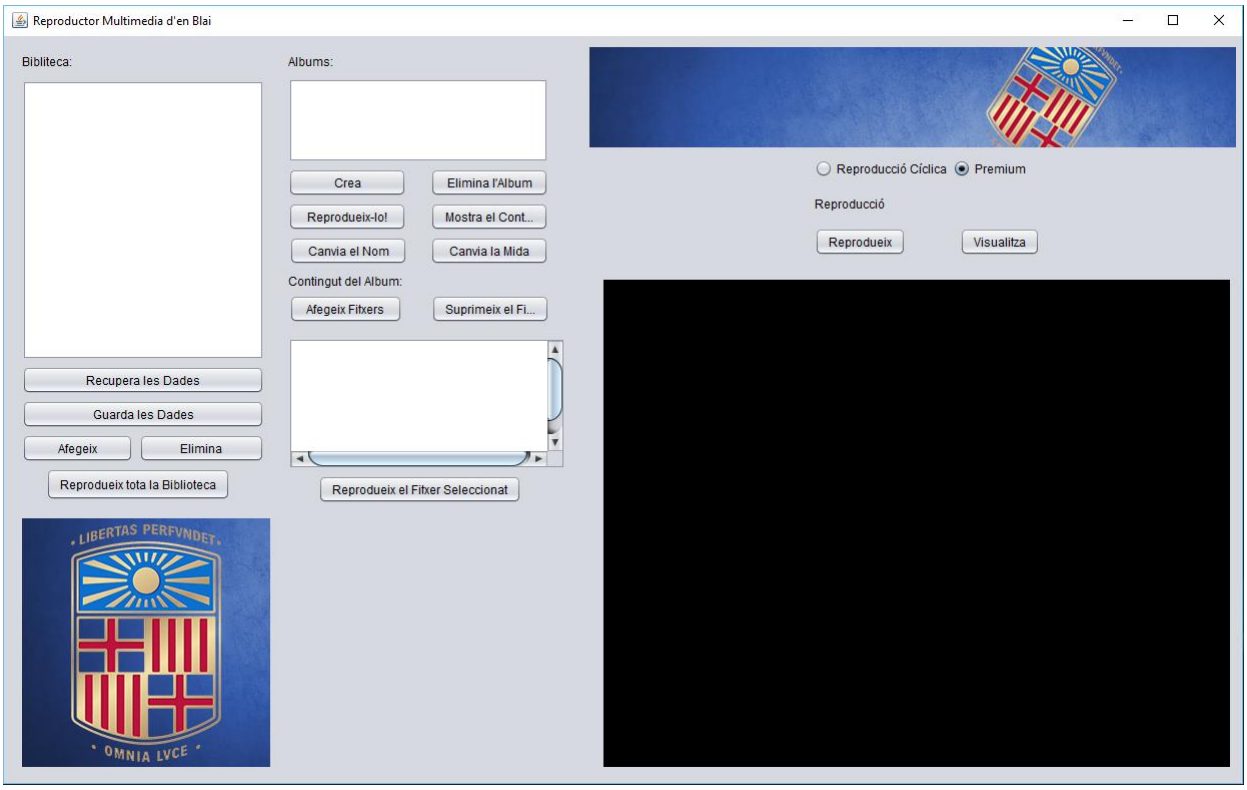
j. Annex: Excepcions

En aquesta entrega totes les excepcions les he gestionat amb un JOptionPane, on mostro la raó de la excepció, el missatge, el nom del JOotionPane i el tipus d'excepció, si es informatiu, error...

"JOptionPane.showMessageDialog(rootPane, "Això es un exemple"+ex.getMessage(), "Error",JOptionPane.ERROR_MESSAGE)



En la seva totalitat, la meva interfície gràfica a quedat així:



3. Preguntes

a. Expliqueu quines classes has pogut reutilitzar del lliurament anterior per a fer aquest. Quins canvis sobre les classes reutilitzades has necessitat fer i perquè.

Explicat a dalt.

b. Explica quin és el model de delegació d'events que es fa servir en el botó de reproducció de la biblioteca.

Al iniciar-se, s'insereix un Action Listener, de manera que al fer clic al botó entrarem en el mètode actionPerfomed, on es crida un mètode de suport (btnReproFitxerActionPerformed(java.awt.event.ActionEvent evt)), que farà les accions necessàries per reproduir tot el contingut de la biblioteca explicades anteriorment.

c. Indiqueu quins tipus d'events heu fet servir al vostre codi

En tota la programació només faig servir l'event actionPerformed de tipus Event Action, que s'invoca quan una acció te lloc. El seu escoltador és ActionListener.

4. Resultats i Comentaris generals i Conclusions

Fins on he pogut "testejar", el meu reproductor duu a terme totes les accions demanades amb la nova interfície gràfica demanada. Per comprovar el seu correcte funcionament he provat de forçar errors, com reproduir un fitxer sense selccionar-lo, què passa si clico reproduir un àlbum i no hi ha cap d'afegit, si afegeixo un fitxer sense posar-li el camí o deixant en blanc alguna etiqueta... De manera que a mesura que provava i trobava errors els anava arreglant.

El que més m'ha sorprès és que les llistes JList i JScrollPane comencin per la posició -1 i no zero... Així que a tot arreu he hagut d'afegir un +1 per agafar la posició.