Lliurament 2

Sumari

1. Objectius

En el lliurament 1 heu implementat la classe ***FitxerMultimedia*** i ***CarpetaFitxers***. Aquesta última permet gestionar un conjunt de fitxers. A partir d'aquesta classe, definireu una classe ***BibliotecaFitxersMultimedia*** que contindrà el conjunt total de fitxers multimèdia de l'aplicació i tindrà les següents propietats:

1. No permetrà que hi hagin dos fitxers multimèdia iguals. Entenem que dos fitxers multimèdia són iguals si el camí, nom, extensió i descripció del fitxers multimèdia són iguals[1].

2. Quan s'introdueix la informació d'un fitxer multimèdia a la biblioteca, es verificarà que el fitxer associat existeix a disc.

3. No hi haurà cap límit en el nombre de fitxers multimèdia.

Pel que fa als fitxers multimèdia, heu de tenir en compte que n'hi ha que es poden reproduir (vídeos, àudios) i altres que no (imatges, text). Els fitxers reproduïbles tenen, per tant, propietats diferents i comportaments diferents de la resta. En aquest lliurament haureu de definir una classe, ***FitxerReproduible,*** per tal de satisfer aquesta especialització.

Al mateix temps, heu de considerar que els àudios i vídeos són fitxers reproduïbles, però que, alhora, tenen característiques diferents i que faran que difereixen en el conjunt d'atributs i comportaments que implementin les seves classes corresponents. Mentre que ambdós tenen una durada i un *codec* associat (mp3, wav, flac, etc. en el cas dels àudios i wmv, H.264/MPEG-4, etc. en els vídeos), hi ha d'altres propietats que no són comunes. Un àudio, per exemple, té una qualitat expressada en *kilobits per segon* (kpbs). I un vídeo té una resolució espacial en *píxels d'alçada x píxels d'amplada* i una resolució temporal en *frames per segon* (*fps*). A més, s'haurà de tenir en compte que els mètodes de reproducció en les seves classes corresponents implementen comportaments diferents. Per exemple, en el cas de l'àudio es podrà mostrar una imatge de caràtula o una altra imatge per defecte durant la seva reproducció. Pels motius exposats, heu d'implementar les classes ***Video*** i ***Audio*** com especialitzacions de la classe ***FitxerReproduible***.

D'altra banda, i per tal que l'usuari interactuï amb l'aplicació, caldrà definir un menú amb les opcions següents, aprofitant tot el que us sigui possible del lliurament 1:

1. Gestió Biblioteca: Dona accés a un menú per a la gestió de la biblioteca.

    1.1. Afegir fitxer multimèdia a la biblioteca: Mostra un menú per tal d'afegir un fitxer multimèdia a la biblioteca.

        1.1.1. Afegir vídeo: Demana les dades necessàries per un nou vídeo.

[1] Tot i que no és possible tenir dos fitxers a disc amb el mateix camí, nom i extensió del fitxer, en aquesta pràctica definim la condició d'igualtat d'aquesta manera per practicar conceptes d'orientació a objectes (sobreescriptura del mètode equals).

1.1.2. Afegir àudio: Demana les dades necessàries per un nou àudio.

1.1.3. Menú anterior: Torna al menú anterior.

1.2. Mostrar Biblioteca: Mostra el contingut de la biblioteca, mostrant davant de cada fitxer, el número de la seva posició a la llista començant per 1.

1.3. Eliminar fitxer multimèdia: Elimina de la biblioteca el fitxer multimèdia corresponent a una posició donada.

1.4. Menú Anterior: Torna al menú principal

2. Guardar Dades: Guarda les dades de l'aplicació a un fitxer de disc.

3. Recuperar Dades: Carrega les dades de l'aplicació prèviament guardades d'un fitxer.

4. Sortir: Surt de l'aplicació.

A més, haureu de tenir en compte algunes consideracions més.

Com en el lliurament anterior, seguirem el patró de programació Model-Vista-Controlador. En el lliurament 1 no teníeu controlador i la vista veia i manipulava classes del model. Ara si que tindreu controlador i la vista no accedirà a classes del model, sinó que delegarà aquesta tasca al controlador.

Seguint aquesta organització, haureu de definir una classe ***Controlador*** que implementarà el controlador de la nostra aplicació. **La vista només podrà utilitzar aquesta classe**. Des d'***AplicacioUB2*** es demanaran els valors dels atributs necessaris per passar-los al ***Controlador*** que s'encarregarà de fer les crides oportunes a la nova classe del model, ***Dades*****.** Aquesta classe contindrà totes les dades de l'aplicació i farà totes les accions que afectin a les dades.

A ***Dades*** hi definireu també dues funcions per tal de poder **guardar-se i carregar-se com objecte** al disc de l'ordinador. D'aquesta manera no haurem de tornar a introduir tota la biblioteca per cada nova execució de l'aplicació.

Sempre que demaneu al controlador fer una tasca des de la vista i aquesta pugui anar malament, es a dir pugui donar peu a error, heu de fer servir la classe ***AplicacioException*** (inclosa en la llibreria UtilsProg2) controlant-la amb un try-catch. Per exemple, si es demana afegir un fitxer a la biblioteca i aquest ja existeix a la biblioteca, s'hauria de crear un objecte de tipus ***AplicacioException*** amb un missatge específic i delegar la captura i gestió de la mateixa excepció fins la vista. D'aquesta manera, podreu informar a l'usuari quin ha sigut el problema i alhora evitareu la sortida de l'aplicació.

2. Material pel lliurament

Per aquest lliurament us proporcionem una llibreria **UtilsProg2.jar** que conté les classes:

- ***Menu***
- ***AplicacioException***
- ***ReproductorBasic***

Haureu d'utilitzar aquestes classes en el desenvolupament del lliurament. Podeu trobar **UtilsProg2.jar** al Campus Virtual i afegir-la al vostre projecte. Aquest cop, també haureu d'afegir, igual que ho feu amb **UtilsProg2.jar,** els següents fitxers jar:

- **jna-3.52.jar**
- **platform-3.5.2.jar**
- **vlcj-3.8.0.jar**
- **slf4j-api-1.7.10.jar**
- **slf4j-nop-1.7.10.jar**

els quals fa servir internament ***ReproductorBasic*** per tal de reproduir els fitxers reproduïbles.

3. Descripció del lliurament

A continuació us plantegem una sèrie de passos per guiar-vos en la implementació d'aquest lliurament. Consulteu també el punt 4 del document, on hi trobareu ajuda addicional.

3.1. Creació del projecte

El primer pas serà crear un nou projecte, al qual li podeu posar com a nom ***Cognom1Nom1Cognom2Nom2_L2***.

Igual que en el lliurament 1, la classe principal es dirà ***IniciadorAplicacioUB***. I el paquet per defecte ***edu.ub.prog2.Congom1Nom1Cognom2Nom2.vista***. Recordeu que és possible que se l'hi hagi d'indicar al NetBeans quina és la classe principal si no és la que s'havia establert en el moment de la creació del projecte. En aquest cas, doncs, feu que apunti a la classe ***edu.ub.prog2.Congom1Nom1Cognom2Nom2.vista.IniciadorAplicacioUB***.

3.2. Classe IniciadorAplicacioUB

La classe de la vista ***IniciadorAplicacioUB*** ha de tenir el mètode estàtic ***main*** on s'ha de crear un objecte de tipus ***AplicacioUB2*** anomenat "aplicacio" i fer la crida al mètode gestioAplicacioUB de la classe ***AplicacioUB2*** :

```
aplicacio.gestioAplicacioUB ();
```

3.3. Classe AplicacioUB2

3.3.1. Mètodes principals

El mètode d'objecte ***gestioAplicacioUB*** implementarà el menú de l'aplicació. Com en la classe del lliurament anterior aquest mètode ha de tenir la següent signatura:

```
public void gestioAplicacioUB();
```

3.3.2. Implementació del menú d'opcions i la lògica del programa

Seguint l'exemple del lliurament 1, creeu la lògica del programa, utilitzant la classe ***Menu*** de la llibreria, i tal com s'especifica en els objectius del lliurament. Comproveu especialment que podeu passar d'un menú a un altre correctament i que l'aplicació finalitza correctament.

Per modularitzar el codi de la vista i fer-lo més llegible, cal que creeu un mètode per delegar les funcions de cada submenú del menú principal.

3.4. Creació de les classes principals de l'aplicació

Tal com s'indica a l'apartat d'objectius, en aquest lliurament fareu finalment efectiu el patró MVC i utilitzareu una classe principal per a cada component. Pel que fa a la vista, ja heu definit les classes ***IniciadorAplicacioUB*** i ***AplicacioUB2***. Ara creareu les classes principals del model i del controlador.

Creeu la classe ***Controlador***, que exercirà de controlador de l'aplicació, controlant els diferents processos que es duguin a terme en l'aplicació. Aquesta classe rebrà les indicacions de l'usuari per mitjà de la vista i farà les accions pertinents, comunicant-se amb el model quan sigui necessari. Igual que totes les classes que tinguin a veure amb el control de l'aplicació, aquesta classe anirà dins del [paquet Controlador] (***edu.ub.prog2.Congom1Nom1Cognom2Nom2.controlador***).

Creeu la classe ***Dades***, que contindrà totes les dades de l'aplicació. De moment podeu deixar la classe totalment en blanc. Quan comencem a implementar les opcions del menú ja anireu afegint els mètodes que hi necessiteu. Igual que totes les classes relacionades amb el model, aquesta classe haurà d'estar dins el [paquet Model] (***edu.ub.prog2.Congom1Nom1Cognom2Nom2.model***).

Un cop definides les classes principals, caldrà que definiu els atributs necessaris per establir la comunicació entre aquestes classes principals. ***AplicacioUB2*** (component vista) declararà un atribut de tipus ***Controlador*** que haurà de ser inicialitzat en el constructor de AplicacioUB2. I, anàlogament, ***Dades*** haurà de ser un atribut de la classe ***Controlador*** i inicialitzat durant la construcció d'aquesta classe.

Heu de tenir en compte que la vista ara només pot interactuar amb el model a través del controlador. La vista no manipularà directament (declarant, inicialitzant o invocant mètodes) els objectes del model. Dit d'una altra manera: a la classe

AplicacioUB2 no us hauria de caldre fer-hi import de cap classe del paquet model. A més, **tots els prints s'hauran de fer a la vista, mitjançant el tractament d'excepcions**.

3.5. Gestió de la Biblioteca

Definiu a continuació la classe ***BibliotecaFitxersMultimedia*** que permet gestionar els fitxers de la biblioteca.

Creeu tots els mètodes necessaris per fer la gestió de les opcions 1.1, 1.2 i 1.3, tenint en compte que equivalen a les opcions 1, 2 i 3 del lliurament 1, i per tant, tot i que cal reestructurar el projecte seguint el model MVC, heu de poder reutilitzar gran part del codi. Recordeu que disposeu de la classe ***CarpetaFitxers*** del lliurament 1. Aquesta classe ha de continuar tenint un mida màxima, però es pot utilitzar com a superclasse de ***BibliotecaFitxersMultimedia*** modificant el comportament dels mètodes necessaris. Penseu bé quins són!

Podeu utilitzar la classe ***AplicacioException*** en el mètode addFitxer per tal d'informar de qualsevol problema fent servir aquesta signatura del mètode:

```
public void addFitxer(File fitxer)  throws AplicacioException ;
```

Per comprovar si un fitxer existeix o no a disc, teniu el mètode ***exists*** de la classe ***File*** i per comprovar si un fitxer ja està inclòs a la carpeta podeu utilitzar el mètode ***equals*** de la classe ***FitxerMultimedia*****.** Si un fitxer ja està inclòs a la biblioteca, haureu de llençar una ***AplicacioException*** i capturar-la en el menú de forma que pugueu informar del problema a l'usuari i seguir amb l'execució de l'aplicació.

3.6. Classe Reproductor

Per poder resoldre el següent lliurament, caldrà crear la classe ***Reproductor*** que hereti de ***ReproductorBasic*** de la llibreria UtilsProg2, obtenint així les funcionalitats d'un motor de reproductor d'àudio i vídeo ja encapsulat en ***ReproductorBasic***. Definiu la vostra classe ***Reproductor*** de la següent manera:

```
public class Reproductor extends ReproductorBasic {

}
```

De moment no cal definir cap mètode per fer la reproducció, ja que aquesta part s'implementarà en el lliurament següent.

3.7. Classes FitxerReproduible, Audio i Video

Com s'ha comentat als objectius, els fitxers multimèdia es poden dividir en fitxer reproduïbles (vídeos, àudios) i altres que no ho són (imatges, text). En

conseqüència, la classe ***FitxerReproduible*** deriva d'una especialització dels fitxers multimèdia. En aquesta pràctica no implementarem classes per a imatges i text.

La classe ***FitxerReproduible*** tindrà un mètode abstracte ***reproduir*** amb la següent signatura:

```
protected abstract void reproduir();
```

i que ha de ser implementat en les classes que heretin de ***FitxerReproduible*** , com són ***Video*** i ***Audio***.

Noteu que el fet de declarar un mètode abstracte obliga a declarar també la classe abstracta. A més, té sentit declarar el constructor *protected*, com s'indica a continuació:

```
protected FitxerReproduible(String cami, String nom, String codec, float durada,
Reproductor r);
```

A continuació creeu les classes ***Audio*** i ***Video*** amb els atributs corresponents a les seves característiques descrites en els objectius. Els constructors d'aquestes classes seran:

```
public Video(String cami, String nom, String codec, float durada, int alcada,
int amplada, float fps, Reproductor r);

public Audio(String cami, File fitxerImatge, String nom, String codec, float
durada, int kbps, Reproductor r);
```

Nota: L'atribut nom es correspon a la descripció del fitxer.

Recordeu durant la implementació d'aquestes classes d'anar cridant correctament als constructors de les superclasses de la jerarquia de classes. Per exemple, *camí* és un argument necessari pel constructor de ***FitxerMultimedia***.

De moment no cal que implementeu els mètodes ***reproduir*** d'***Audio*** i ***Video***, ho fareu en el proper lliurament. De totes maneres, és interessant entendre que serà per la reproducció que hem passat un objecte de tipus **Reproductor** pel constructor d'aquests objectes.

3.8. Classe Controlador

La peça que us falta ara és la classe principal del controlador anomenada **Controlador**. Els principals mètodes d'aquesta classe seran els següents:

```
public void afegirVideo(String path, String nomVideo, String codec, float durada, int alcada, int amplada, float fps) throws AplicacioException;

public void afegirAudio(String cami, String camiImatge, String nomAudio, String codec, float durada, int kbps) throws AplicacioException;

public List<String> mostrarBiblioteca();  // llista dels retorns de toString() dels fitxers

public void esborrarFitxer(int id) throws AplicacioException; // id és la posició a llista de getBiblioteca()

public void guardarDadesDisc(String camiDesti) throws AplicacioException;

public void carregarDadesDisc(String camiOrigen) throws AplicacioException;
```

La majoria de funcionalitats d'aquests mètodes es poden delegar en la classe ***Dades***. Per tant, aquests mètodes només faran una crida al mètode corresponent de ***Dades***. Per exemple, la instanciació d'un objecte ***Video*** quan l'usuari hagi demanat afegir un vídeo, es pot fer a la classe ***Dades***. Això ho fem d'aquesta manera, perquè en els propers lliuraments, ***Controlador*** farà la feina de controlar la reproducció dels fitxers de la biblioteca.

Fixeu-vos que les especificacions que rebeu, com a programadors de la vista, per poder crear nous vídeos i àudios es dedueixen dels mètodes ***afegirVideo*** i ***afegirAudio***, que us indiquen quina informació necessita el controlador per crear-los i que, per tant, haureu de demanar a la vista per passar-li.

3.9. Classe Dades (i la persistència de les dades)

Aquesta classe s'encarregarà d'emmagatzemar les dades de l'aplicació que consistiran en una biblioteca. Primer, implementeu els mètodes que proveeixen les funcionalitats requerides pel controlador.

La classe ***Dades*** ha d'implementar a més els mètodes per a guardar-se i recuperar-se ella mateixa, ja que volem poder guardar totes les dades de l'aplicació abans de sortir de l'aplicació per poder recuperar-les després.

Per últim, heu d'implementar un mètode ***toString*** per mostrar el resum de les dades.

3.10. Persistència de dades

El què volem fer ara és poder guardar les dades a un fitxer en disc i poder-les carregar posteriorment. Implementeu les **opcions 4 i 5 del menú**.

Per fer aquesta opció, necessiteu seguir els següents passos, que s'explicaran amb detall a la sessió de Problemes:

1. Obtenir la ruta al fitxer on voleu guardar les dades o des del qual voleu carregar-les. Necessitareu guardar aquesta informació en un objecte de tipus File. Per exemple, si volem utilitzar el fitxer "dades.dat" (Nota: no cal que existeixi a disc, es crea en el moment de fer new), farem:

```
File fitxer = new File("dades.dat");
```

2. L'accés de lectura i escriptura a un fitxer es fa mitjançant *streams*. Per llegir d'un fitxer utilitzarem un objecte de tipus ***FileInputStream***, i per escriure a un fitxer utilitzarem un objecte ***FileOutputStream***:

```
FileInputStream fin=new FileInputStream(fitxer);

FileOutputStream fout= new FileOutputStream(fitxer);
```

3. Finalment, existeixen objectes per gestionar la lectura i escriptura d'un objecte a un *stream*. Per escriure un objecte utilitzarem un objecte de tipus ***ObjectOutputStream***, mentre que per llegir-lo utilitzarem un objecte de tipus ***ObjectInputStream***:

```
ObjectOutputStream oos = new ObjectOutputStream(fout);

ObjectInputStream ois = new ObjectInputStream(fin);
```

Teniu més informació sobre *streams* en el document d'ajuda al lliurament 1. No oblideu tancar correctament els objectes d'accés als fitxers. Tenint en compte que en aquest lliurament només tenim una carpeta de fitxers, podeu guardar directament aquest objecte.

3.11. Excepcions

En aquest lliurament, utilitzarem excepcions per fer la gestió d'alguns errors, com per exemple, l'error d'afegir un fitxer multimèdia a la biblioteca quan aquest ja ha estat inclòs prèviament o l'error corresponent a afegir un fitxer multimèdia que no existeix a disc.

S'utilitzarà la classe ***AplicacioException*** per representar aquests errors particulars. A la part del codi on es produeix un problema a controlar, s'haurà de crear aquesta excepció i llançar-la, fent servir ***throw*** dins del mètode corresponent. La paraula ***throws*** al final de la capçalera del mètode ens permetrà delegar la captura d'aquesta excepció fins al mètode on es vols gestionar. Utilitzeu el mateix constructor de l'excepció per introduir-hi el missatge d'error adequat. Finalment, feu servir *try-catch* per capturar-la i llavors: informar a l'usuari de l'error produït, així com del motiu de l'error i controlar l'execució de l'aplicació segons l'error produït.

4. Ajuda pel lliurament

4.1. UtilsProg2

Utilització de la llibreria **UtilsProg2.jar** i llibreries externes per l'ús del motor del VLC:

- Podeu instanciar objectes de la classe **Menu** a la vostra classe vista per implementar la gestió del menú de la aplicació tal com ho havíeu fet al lliurament 1. Recordeu que teniu un exemple de com utilitzar la classe **Menu** al Campus Virtual.
- La classe ***ReproductorBasic*** és la classe que implementa un motor de reproducció de fitxers reproduïbles. Com que és abstracta, no es pot instanciar directament i, per tant, per utilitzar-ne les funcionalitats haurà de ser estesa per la vostra classe ***Reproductor*** **–** la qual sí que podrà ser instanciada i utilitzada per la classe ***Controlador***.

4.2. Utilització del VLC

Per fer funcionar ***ReproductorBasic*** cal tenir instal·lat el VLC a la màquina i incloure al projecte de NetBeans llibreries addicionals (en format .jar) que trobareu al Campus Virtual tal i com ho feu amb **UtilsProg2.jar**.

A les aules haureu de fer servir Linux perquè a Windows el VLC instal·lat és de 32 bits i no us funcionarà.

Als portàtils amb Windows, haureu de tenir instal·lada la versió experimental del VLC de 64 bits. La podeu baixar aquí:
http://get.videolan.org/vlc/2.2.2/win64/vlc-2.2.2-win64.exe
Si no utilitzéssiu la ruta d'instal·lació del VLC per defecte (C:/Program Files/VideoLAN/VLC/), haureu de passar-li la nova ruta al constructor del ***ReproductorBasic***.

5. Format del lliurament

El lliurament consistirà en tot el codi generat en els diferents punts de l'enunciat, juntament amb la documentació especificada en aquest apartat.

En concret, cal generar un fitxer comprimit amb el nom: **Cognom1Nom1Cognom2Nom2_L2**, que contingui:

1. El projecte sencer de NetBeans
Tot el codi generat ha d'estar correctament comentat per a poder executar el JavaDoc, generant automàticament la documentació en línia del codi.

2. La memòria del lliurament.

La memòria ha de contenir els punts descrits en la normativa de pràctiques i els punts següents:

1. Expliqueu quines classes has pogut reutilitzar del lliurament anterior per a fer aquest. Quins canvis sobre les classes reutilitzades heu necessitat fer i perquè.
2. Expliqueu quants objectes s'han creat en l'execució d'aquest mètode ***main*** si estem a l'última línia:

```
public static void main(String[] args) {
      // Creem un objecte de la vista
      AplicacioUB2 aplicacio=new AplicacioUB2();

      // Inicialitza l'execució de la vista
      aplicacio.gestioAplicacioUB();
}
```

3. Expliqueu com heu implementat i on heu utilitzat el mètode ***equals*** heretat de la classe ***Object***.
4. Expliqueu com heu utilitzat la classe ***AplicacioException*** al vostre codi.
5. Expliqueu si heu fet servir la sobrecàrrega de mètodes a la classe **Controlador** i detalleu com i perquè.
6. Prepareu el diagrama de classes del vostre programa.
7. Feu un diagrama de flux per mostrar el recorregut que fa el vostre programa quan s'executa l'opció d'afegir un fitxer multimèdia a la biblioteca. Especificar els mètodes que es criden en cadascuna de les classes. Feu servir fletxes i números per indicar l'ordre de les crides.
8. Expliqueu quins canvis hauríeu de fer al codi següent per tal d'aconseguir que al cridar al mètode reproduir de la següent manera:

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

   s'imprimeixi un missatge diferent quan es tracta d'un fitxar d'àudio o un fitxer de vídeo.
9. Expliqueu les proves realitzades per comprovar el correcte funcionament de la pràctica, resultats obtinguts i accions derivades.
10. Observacions generals.

6. Data límit del lliurament

Consulteu el calendari de pràctiques al Campus Virtual.