PROJECTE DE PRÀCTIQUES

PROGRAMACIÓ II — LLIURAMENT 2

JOSEP NÚÑEZ RIBA
RODRIGO CABEZAS QUIRÓS

10 D'ABRIL DE 2018

Índex

1. Introducció

L'objectiu del projecte és desenvolupar un reproductor multimèdia, tot passant per totes les fases de desenvolupament d'un projecte de software i familiaritzar-se amb l'ús d'eines informàtiques de suport a la programació. Està enfocat cap a la programació orientada a objectes, programació orientada a esdeveniments i programació guiada per una especificació.

Els objectius a assolir en aquest lliurament són els següents:

- Implementar la classe BibliotecaFitxersMultimedia, que contindrà el conjunt total dels fitxers multimèdia de l'aplicació. Haurà de complir el següent:
    1. No permetrà que hi hagin dos fitxers iguals, entenent per igual que tant el path, nom, extensió i descripció dels fitxers són els mateixos.
    2. Al afegir un fitxer a la biblioteca es verificarà que el fitxer existeix al disc.
    3. No hi ha límit en el nombre de fitxers multimèdia.
- Implementar la classe FitxerReproduible, que hereta de FitxerMultimedia tindrà els atributs i comportaments adequats per tal de satisfer l'especialització.
- Implementar les classes Audio i Video, que són especialitzacions de FitxerReproduible amb els seus atributs corresponents.
- Expandir el menú del lliurament anterior per tal de que pugui realitzar les operacions següents:
    1. Gestió biblioteca (afegir, eliminar, display i tornar al menú principal).
    2. Guardar dades de l'aplicació al disc.
    3. Recuperar les dades guardades al disc.
    4. Sortir de l'aplicació.
- Implantar el patró model — vista — controlador. Entenent com a vista el que veu l'usuari, model els recursos i controlador qui s'encarrega de les crides. D'aquesta forma, AplicacioUB2 cridarà a Controlador, qui farà les crides oportunes.
- Implementar la classe Dades, que s'encarregarà de tot el que impliqui dades a la aplicació, incloent-hi la permanència de dades.
- Controlar les excepcions amb la classe proporcionada AplicacioException utilitzant els blocs de try-catch.

2. Anàlisi

Per implementar la classe BibliotecaFixersMultimedia, farem que hereti de la classe CarpetaFitxers. Aquesta classe implementarà la interfície Serializable per tal de poder guardar i carregar les dades al disc. Com que la classe hereta de CarpetaFitxers, la meitat de mètodes necessaris ja estan fets i es pot reutilitzar el codi.

En el que respecta a FitxerReproduible, heretarà de FitxerMultimedia i inclourà els atributs comuns per que siguin després utilitzats per les classes Audio i Video. Aquestes últimes heretaran de FitxerReproduible i tindran com atributs específics: kpbs (velocitat) i fitxerImatge (caràtula) per audio i amplada, alçada i fps per vídeo. Ambdues classes incorporen el mètode reproduir() que serà implementat en el proper lliurament.

Actualitzar el menú simplement requerirà afegir les noves opcions requerides i ampliar el switch, es reutilitza el codi i no cal fer grans modificacions.

Per implantar el patró model — vista – controlador, delegarem totes les accions en la classe Controlador. D'aquesta manera la mateixa classe s'encarrega d'efectuar totes les crides. Les accions sobre Dades també seran executades des de controlador deixant per l'usuari la classe vista.

3. Desenvolupament

- **Classe Controlador:**
  - public Controlador(): Constructor de Controlador.
  - public void afegirVideo(String path, String nomVideo, String codec, float durada, int alcada, int amplada, float fps): Afegeix un fitxer de vídeo a la biblioteca. Delega execució a Dades i a BibliotecaFitxersMultimedia.
  - public void afegirAudio(String cami, String nomAudio, String codec, float durada, String camiImatge, int kbps): Afegeix un fitxer de àudio a la biblioteca. Delega execució a Dades i a BibliotecaFitxersMultimedia.
  - public String mostrarBiblioteca(): Mostra per pantalla un resum del contingut de la biblioteca. Delega execució a Dades i a BibliotecaFitxersMultimedia.
  - public void esborrarFitxer(int id): Esborra el fitxer corresponent a la id que se li passa per paràmetre. Delega execució a Dades i a BibliotecaFitxersMultimedia.
  - public void guardarDadesDisc(String camiDesti): Guarda les dades de la biblioteca al disc. Delega execució a Dades.
  - public void carregarDadesDisc(String camiOrigen): Carrega les dades de la biblioteca del disc. Delega execució a Dades.
  - public String mostrarCamins(): Mostra els paths dels fitxers. Delega execució a CarpetaFitxers.
  - public boolean isEmpty(): Retorna true si la biblioteca és buida. Delega execució a Dades i a CarpetaFitxers.
  - public boolean isRemovable(int i): Retorna true si el fitxer és pot eliminar. I és la id del fitxer. Delega execució a Dades i a BibliotecaFitxersMultimedia.

- **Classe Dades:**
  - public Dades(): Constructor per defecte de la classe Dades.

1. public void esborrarFitxer(int i): Esborra un fitxer donat. I correspon al fitxer.

```
    public void esborrarFitxer(int i){
2.      FitxerMultimedia fitxer=this.biblioteca.getAt(i);
3.      this.biblioteca.removeFitxer(fitxer);
4.  }
```

  - public String toString(): Mètode ToString de la classe Dades.
  - public boolean isEmpty(): Retorna true si el fitxer és pot eliminar, false en cas contrari.
  - public boolean isRemovable(int i): Retorna true en el cas de que el fitxer pugui ser suprimit, en cas contrari retornarà false.
  - public String mostrarCamins(): Mostra els paths dels fitxers que conté la biblioteca.
  - public void afegirAudio(String cami, String nomAudio, String codec, float durada, String camiImatge, int kbps): Afegeix un fitxer d'àudio a la biblioteca.

```
1.  public void afegirAudio(String cami, String nomAudio, String
    codec, float durada, String camiImatge, int kbps) throws Aplica
    cioException {
2.      Reproductor r=new Reproductor();
3.      File image=new File(camiImatge);
4.      Audio
    fitxer=new Audio(cami,nomAudio,codec,durada,image,kbps,r);
5.      this.biblioteca.addFitxer(fitxer);
6.  }
```

  - public void afegirVideo(String path, String nomVideo, String codec, float durada, int alcada, int amplada, float fps): Afegeix un fitxer de vídeo a la biblioteca.
  - public void guardar(String desti): Guarda les dades que conté la biblioteca a la direcció que se li passa per paràmetre.
  - public void carregar(String origen): Carrega les dades de la biblioteca que prèviament s'han guardat. Carregarà les dades de la direcció que se li passa per paràmetre.

```
1.     public void carregar(String origen) throws FileNotFoundExcep
   tion, IOException, ClassNotFoundException {
2.         ObjectInputStream ois;
3.         try (FileInputStream fileStream = new FileInputStream(or
   igen)) {
4.             ois = new ObjectInputStream(fileStream);
5.             this.biblioteca=(BibliotecaFitxersMultimedia)ois.rea
   dObject();
6.         }
7.        ois.close();
8.     }
```

- **Classe FitxerReproduible:**
  - protected FitxerReproduible(String cami, String nom, String codec, float durada, Reproductor r): Constructor per la classe FitxerReproduible. Està declarat com a protected per tal de que només les subclasses d'aquesta hi puguin fer-ne ús.
  - protected abstract void reproduir(): Mètode abstracte, cal ser implementat per les subclasses.
  - public Reproductor getReproductor(): Retorna el reproductor del fitxer.
  - public String getCodec(): Retorna el codec del FitxerReproduible.
  - public float getDurada(): Retorna la durada del FitxerReproduible.
  - public String toString(): Mètode ToString de la classe FitxerReproduible. Sobrecàrrega el mètode de la classe FitxerMultimedia.
- **Classe Video:**
  - public Video(String cami, String nom, String codec, float durada, int alcada, int amplada, float fps, Reproductor r): Constructor per la classe Video. Crida al constructor de la classe superior FitxerReproduible.
  - public int getAlcada(): Retorna la alçada del fitxer de vídeo.
  - public int getAmplada(): Retorna la amplada del fitxer de vídeo.
  - public float getFps(): Retorna els *frames per second* del fitxer de vídeo.
  - public String toString(): Mètode ToString de la classe Video. Sobrecàrrega el mètode de la classe superior FitxerReproduible.
  - public void reproduir(): Reprodueix el fitxer de vídeo. A implementar en el lliurament 3.

- **Classe Audio:**
  - public Audio(String cami, String nom, String codec, float durada, File fitxerImatge, int kbps, Reproductor r): Constructor de la classe Audio. Crida al constructor de la classe superior FitxerReproduible.
  - public int getKbps(): Retorna els kbps del audio.
  - public File getImatge(): Retorna la caràtula del audio.
  - public String toString(): ToString de la classe Audio. Sobrecàrrega el mètode heretat de la classe FitxerReproduible.
  - public void reproduir(): Reprodueix el fitxer. Implementació pel lliurament 3.
- **Classe BibliotecaFitxersMultimedia:**
  - public BibliotecaFitxersMultimedia(): Constructor per defecte de la classe BibliotecaFitxersMultimedia.
  - public void addFitxer(File file): Afegeix un fitxer a la biblioteca. Comprova que tant el directori com el fitxer existeixen. Si el fitxer ja hi és a la biblioteca i/o no és compleixen les condicions anteriors llença una excepció.

```
1. public void addFitxer(File file) throws AplicacioException{
2.          FitxerMultimedia fitxer=(FitxerMultimedia) file;
3.          if((fitxer.exists()) && (!(fitxer.isDirectory()))){
4.              if (this.contains(fitxer)){
5.                  throw new AplicacioException("Aquest fitxer ja
   es troba a la biblioteca.");
6.              }else{
7.                  this.carpeta.add(fitxer);
8.              }
9.          }else{
10.                   throw new AplicacioException("Aquest fitxer no
   existeix.");
11.               }
12.           }
```

  - public void removeFitxer(FitxerMultimedia fitxer): Elimina un fitxer de la biblioteca. Recorre la biblioteca fins que troba el fitxer i l'elimina. En cas d'estar el fitxer repetit, només l'eliminarà un sol cop.

```
1.      public void removeFitxer(FitxerMultimedia fitxer){
2.         boolean removed = false;
3.         int i =0;
4.         while((i<this.getSize())&&(!(removed))){
5.             if (fitxer.equals(this.getAt(i))){
6.                 this.carpeta.remove(i);
7.                 removed = true;
8.             }
9.             i++;
10.              }
11.          }
```

- public boolean isRemovable(int i): Retorna true si el fitxer pot esser eliminat.

- **Classe CarpetaFitxers:**
  - public CarpetaFitxers(): Constructor per defecte. Inicialitza per defecte, la mida màxima a 100 arxius i crea un Array amb 100 slots.
  - public CarpetaFitxers(int mida): Constructor de carpeta amb mida variable. Inicialitza mida màxima en funció de la mida que se li passi i crea l'Array corresponent.
  - public int getSize(): Retorna el nombre de fitxers a la carpeta.
  - public void addFitxer(File fitxer): Si hi ha espai a la carpeta i l'arxiu no hi és en aquesta, afegeix el fitxer. En els altres cassos llença una excepció.
  - public removeFitxer(File fitxer): Suprimeix un fitxer donat de la carpeta. Amb un bucle i un comptador, comprova que l'element iterat sigui equivalent al que es vol eliminar. Si l'arxiu és a la carpeta el borra i incrementa l'espai disponible. Cas contrari llença una excepció.

```
1. public boolean removeFitxer(FitxerMultimedia fitxer){
2.         boolean removed = false;
3.         int i =0;
4.         while((i<this.getSize())&&(!(removed))){
5.             if (fitxer.equals(this.getAt(i))){
6.                 this.carpeta.remove(i);
7.                 removed = true;
8.             }
9.             i++;
10.              }
11.              if(!removed){
12.                  System.out.print("\n\033[31mError!
   L'arxiu no existeix o no es troba en aquesta
   carpeta.\033[0m\n");
13.                  return false;
14.              }
```

```
15.             System.out.print("\n\033[32mFitxer eliminat
   amb èxit.\033[0m\n");
16.             return true;
17.         }
```

- public File getAt(int position): Comprova que la posició de la carpeta no és buida. En cas de no ser-ho retorna l'element corresponent en aquesta posició a la carpeta.
- public void clear(): Buida la carpeta de fitxers.
- public boolean isFull(): Comprova si la mida màxima i la utilitzada són iguals, en cas de ser-ho la carpeta és plena i retorna true. Cas contrari retorna false.
- public boolean isEmpty(): Si la mida de la carpeta és zero, retorna true.
- public String toString(): Retorna un resum amb tota la informació dels fitxers de la carpeta(nom, descripció,...). Utilitza el mètode toString de FitxerMultimedia.

```
1.     public String toString(){
2.         String resum = "Carpeta
   fitxers:\n==============\n\n";
3.         for (int i=0;i<carpeta.size();i++){
4.             resum=resum+"["+(i+1)+"]
   "+this.getAt(i).toString()+"\n";
5.         }
6.         return resum;
7.     }
```

- **Classe FitxerMultimedia:**
  - public FitxerMultimedia(String camí): Constructor per defecte de FitxerMultimedia. Comprova que l'arxiu indicat existeix i que es troba al path indicat (si no és compleix avisa del error). Per defecte inicialitza la descripció com a un String buit**.** El nom del fitxer i la seva extensió la agafa fent un substring del nom del fitxer.

```
1.     public FitxerMultimedia(String cami) {
2.         super(cami);
3.         if((this.exists()) && (!(this.isDirectory()))){
4.             this.setUltimaModificacio(new Date(this.lastMo
   dified()));
5.             this.setCamiAbsolut(this.getAbsolutePath());
6.             String name=this.getName();
7.             int punt=name.lastIndexOf(".");
8.             this.setNomFitxer(name.substring(0,punt));
```

```
9.              this.setExtensio(name.substring(punt));
10.              }else{
11.                  //System.out.println("\nEl fitxer no
   existeix.\n");
12.                  this.setUltimaModificacio(new Date());
13.                  this.setCamiAbsolut(cami);
14.                  this.setNomFitxer("");
15.                  this.setExtensio("");
16.              }
17.              this.setDescripcio("");
18.          }
```

- public FitxerMultimedia(String camí, String desc): Inicialitza tots els atributs del fitxer amb els seus valors corresponents i afegeix la descripció del fitxer.
- public void setUltimaModificacio(Data lastModified): Estableix la data d'última modificació del fitxer**.**
- public void setCamiAbsolut(String camí): Estableix la ruta on es troba el fitxer.
- public void setNomFitxer(String nom): Dóna nom a un fitxer donat.
- public void setExtensio(String extensió): Assigna una extensió a un fitxer.
- public void setDescripció(String desc): Dóna descripció al fitxer.
- public Date getUltimaModificacio(): Retorna la data de la última modificació.
- public String getCamiAbsolut(): Retorna la path on es troba el fitxer.
- public String getNomFitxer(): Retorna el nom del fitxer.
- public String getExtensio(): Retorna l'extensió de l'arxiu.
- public String getDescripcio(): Retorna la descripció del fitxer.
- public Boolean equals(FitxerMultimedia fitxerMultimedia): Compara els atributs d'un fitxer amb els d'un altre, si són equivalents retorna true, cas contrari retorna false.

```
1.      public boolean equals(/*Object*/FitxerMultimedia
   fitxerMultimedia){
2.          boolean retorn=true;
3.          if (!(fitxerMultimedia.getUltimaModificacio().equals(thi
   s.getUltimaModificacio()))){
4.              return false;
5.          }else if (!(fitxerMultimedia.getCamiAbsolut().equals(thi
   s.getCamiAbsolut()))){
```

```
            return false;
        }else if (!(fitxerMultimedia.getNomFitxer().equals(this.
getNomFitxer()))){
            return false;
        }else if (!(fitxerMultimedia.getExtensio().equals(this.g
etExtensio()))){
                return false;
            }else if (!(fitxerMultimedia.getDescripcio().equals
(this.getDescripcio()))){
                return false;
            }else{
                return true;
            }
        }
```

- public String toString(): Retorna un resum amb tots els atributs del fitxer (nom, extensió, path, ...).

- **Classe AplicacioUB2:**
  - public void gestioAplicacioUB(Scanner sc): S'encarrega de gestionar les diferents opcions i accions a realitzar. Amb la classe Scanner, recull la opció seleccionada per l'usuari.
  - public void gestioBiblioteca(Scanner sc): S'encarrega d'afegir, eliminar o de mostrar per pantalla els elements a la biblioteca. Amb la classe Scanner, recull la opció seleccionada per l'usuari.
  - public Object[] demanaVideo(Scanner sc): Demana a l'usuari les dades especifiques per a afegir un fitxer de vídeo a la biblioteca.
  - public Object[] demanaAudio(Scanner sc): Demana a l'usuari les dades especifiques per a afegir un fitxer d'àudio a la biblioteca.
  - public Object[] demana(Scanner sc): Demana a l'usuari les dades bàsiques per a afegir un fitxer a la biblitoeca.

4. Qüestions

- Explicar breument les classes implementades:

  Classe AplicacioUB2: S'encarrega de les accions que l'usuari pot fer sobre la biblioteca (afegir, eliminar, mostrar fitxers,...).
  Classe IniciadorUB: Classe principal que s'encarrega d'executar l'aplicació.
  Classe FitxerMultimedia: Hereta de la classe File. Gestiona els atributs dels fitxers.
  Classe CarpetaFitxers: S'encarrega del emmagatzematge d'arxius.
  Classe FitxerReproduible: Hereta de FitxerMultimedia i inclou els atributs bàsics d'un fitxer que pot esser reproduït.
  Classe Vídeo: Especialització de la classe FitxerReproduible. Incorpora els atributs essencials d'un vídeo (fps, dimensions,...).
  Classe Audio: Especialització de la classe FitxerReproduible. Incorpora els atributs d'un fitxer d'àudio tals com la velocitat o la caràtula.
  Classe BibliotecaFitxersMultimedia: S'encarrega de la gestió de fitxers.
  Classe Dades: Aquesta classe és l'encarregada de tot el que involucri dades al programa, des de afegir un arxiu a la biblioteca a guardar les dades d'aquesta al ordinador.
  Classe controlador: Dirigeix totes les accions. Delega les crides a les classes especifiques. Pròpia del patró model — vista — controlador.

- 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();
}
```

Fins a l'ultima línia és creen 5 objectes: sc (Scanner), aplicació (AplicacioUB2), controlador (Controlador), dades (Dades) i biblioteca (BibliotecaFitxersMultimedia).

- Expliqueu com heu implementat i on heu utilitzat el mètode equals heretat de la classe Object.

  El mètode equals l'hem implementat a la classe FitxerMultimedia. Comprova que tots els atributs dels fitxers siguin iguals. L'hem utilitzat en CarpetaFitxers, en el mètode d'esborrar fitxers i al mètode de contains.

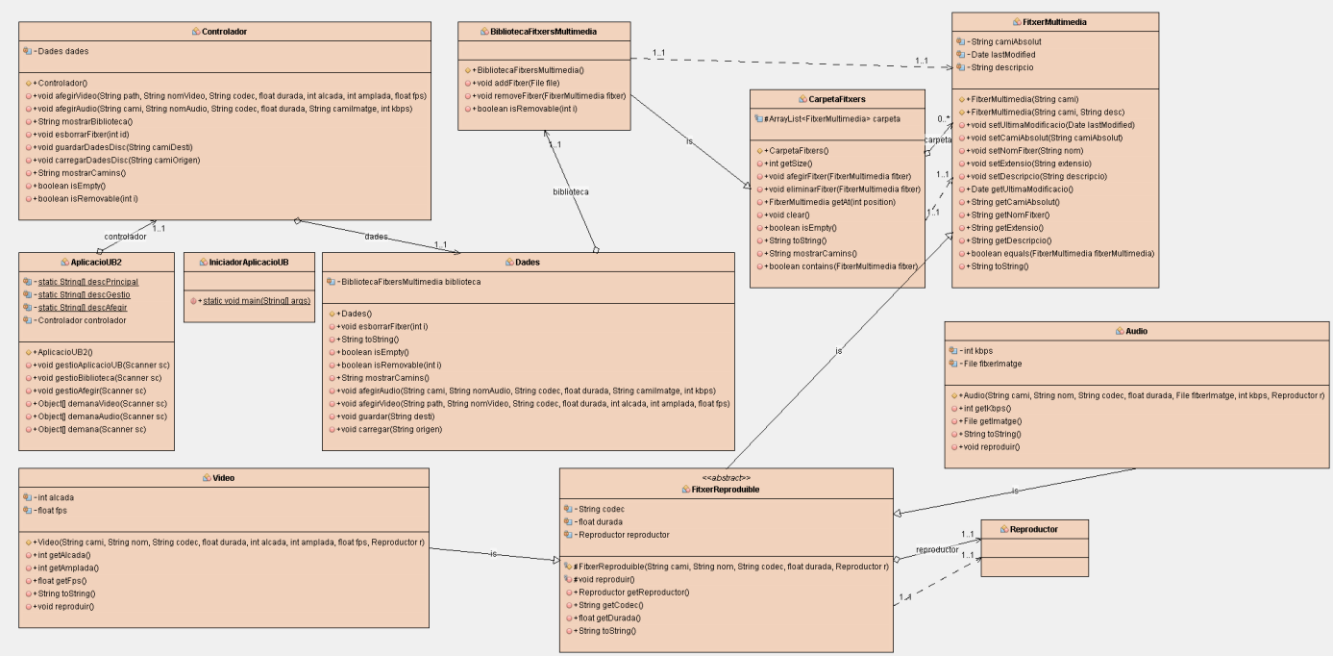
- Expliqueu com heu utilitzat la classe AplicacioException al vostre codi.

  La classe AplicacioException l'hem utilitzada per tractar els cassos crítics en la interacció en l'usuari. Quan l'usuari introdueix unes dades errònies en algun camp de l'aplicació, es llença una excepció que posteriorment es recollida al bloc de catch. Un cop recollida mostra per pantalla, el motiu pel qual l'excepció ha sigut llençada.

- Expliqueu si heu fet servir la sobrecàrrega de mètodes a la classe Controlador i detalleu com i perquè.

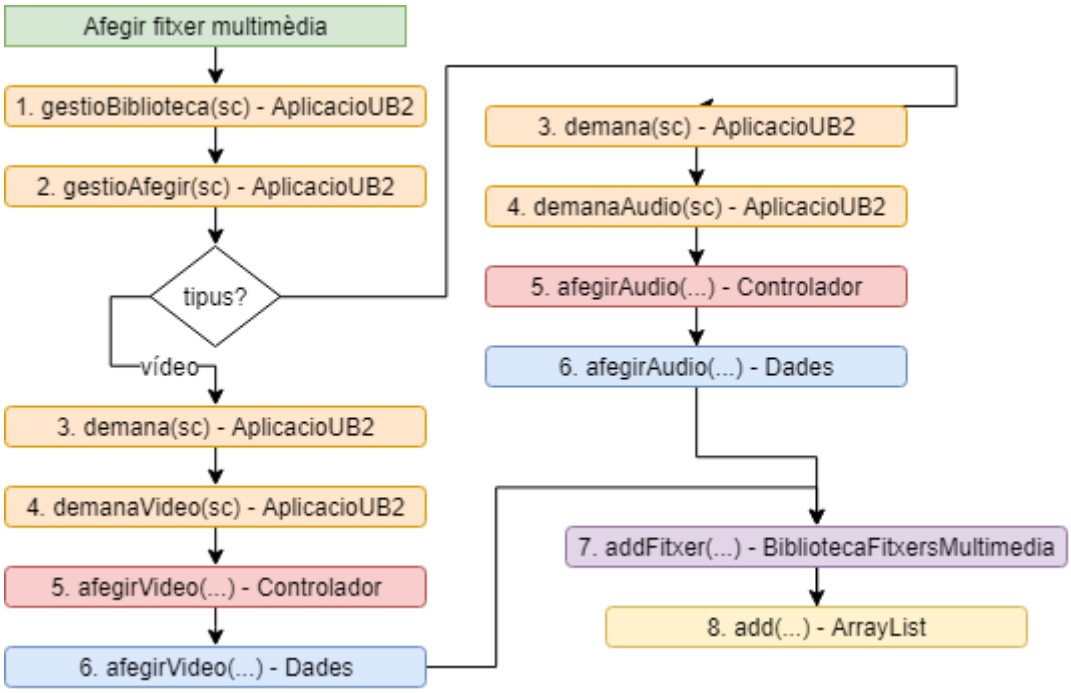
  No hem utilitzat la sobrecàrrega de mètodes a la classe Controlador.

- Prepareu el diagrama de classes del vostre programa.



Clica en la imatge per veure-la més ampliada.

- 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.



- Expliqueu quins canvis hauríeu de fer al codi per tal d'aconseguir que al cridar al mètode reproduir de la següent manera: s'imprimeixi un missatge diferent quan es tracta d'un fitxar d'àudio o un fitxer de vídeo.

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

Per tal de que imprimís un missatge diferent en funció de si es tracta d'un fitxer d'àudio o vídeo, una bona solució seria declarar el mètode reproduir() com a abstracte. D'aquesta manera cada classe que l'hereta l'ha d'implementar, amb la possibilitat de posar missatges diferents en funció del tipus d'arxiu. Aquesta és la manera com nosaltres l'hem implementat.

5. Resultats

- **Proves gestió de biblioteca:**
  - Objectiu: Comprovar el correcte funcionament del mètode gestioAplicacioUB.

Descripció	Resultat esperat	Resultat obtingut / Correcció
Des del menú principal es vol afegir un nou fitxer de vídeo	Apareix opció d'afegir fitxer.	Correcte: Demana tots els detalls del arxiu i l'afegeix.
Des del menú principal es vol eliminar un fitxer.	S'elimina el fitxer desitjat.	Correcte: Elimina el fitxer de la carpeta.
S'introdueix una opció incorrecta al menú principal.	Error, la opció no és correcta.	Correcte: Torna a demanar una opció vàlida.
Des del menú principal es vol afegir un nou fitxer d'àudio.	Demana atributs del àudio.	Correcte: Afegeix l'àudio a la biblioteca1
Es vol veure el contingut de la carpeta.	Mostra per pantalla els elements.	Correcte: Mostra per pantalla els elements.
Es vol eliminar un arxiu que està duplicat.	Només un arxiu és esborrat.	Correcte: El duplicat de l'arxiu segueix a la biblioteca.
Volem tornar al menú anterior.	Surt del menú de gestió.	Correcte: Mostra per pantalla el menú principal.
Afegir un fitxer que no existeix.	Error, el fitxer no existeix.	Correcte: Mostra per pantalla que el fitxer desitjat no existeix al disc.
Operar en un directori que no existeix.	Error, el directori no existeix.	Correcte: Explica per pantalla que el path indicat no existeix al disc.
Mostrar el contingut d'una carpeta buida.	Mostra carpeta buida.	Correcte: Mostra el missatge de que la carpeta és buida.

- **Proves permanència de dades:**
  - Objectiu: Verificar que les dades de la biblioteca són guardades o carregades correctament.

Descripció	Resultat esperat	Resultat obtingut / Correcció
Guardar les dades al disc.	Dades guardades.	Correcte: Un fitxer amb les dades de la biblioteca és crea al directori indicat.
Carregar dades del disc.	Carrega dades noves a la biblioteca.	Correcte: A la biblioteca apareix el contingut prèviament guardat.
Guardar dades a un directori que no existeix.	Error, directori no existeix.	Correcte: Explica per pantalla que el directori no existeix.
Carregar dades a un directori que no existeix.	Error, directori no existeix.	Correcte: Explica per pantalla que el directori no existeix.
Carregar dades d'un directori on no és el fitxer .data.	Les dades no són carregades.	Correcte: No és carrega cap dada a la biblioteca.
Guardar les dades d'una carpeta buida.	El fitxer .data es crea igualment.	Correcte: Crea el fitxer de les dades tot i que la carpeta és buida.
Carregar les dades d'una carpeta buida.	Les dades es carreguen.	Correcte: Les dades és carreguen però a la biblioteca no apareix res.