Práctica 2

MEMORIA

Programació II

Eduardo Wass Rosado
NIUB 16447981
Universitat de Barcelona

Índice

Introducción

Descripción de la práctica

La segunda práctica es una extensión de la primera, ahora utilizaremos la clase LlistaFitxers desde el principio.

Se tiene que hacer un programa que organice nuestra biblioteca musical y nos permita crear gestionar listas de reproducción.

La aplicación debe cumplir los siguientes requisitos:

- La aplicación se compone de **una** Biblioteca, y de un **número indeterminado** de listas de reproducción
- **Biblioteca (LlistaFitxers):**
    - No se permitirán dos canciones iguales (misma ruta).
    - Al añadir una canción a la biblioteca, se debe verificar que el fichero existe.
    - No hay límite en el número de canciones.
- **Listas de reproducción (LlistaReproducció):**
    - Cada lista tiene un título identificador y único
    - Se pueden tener canciones duplicadas en una lista de reproducción
    - Las listas están limitadas a N (N=10) canc
- El programa tendrá dos menús: uno para gestionar la biblioteca, y otro para gestionar las listas de reproducción, cuyas opciones están definidas en la especificación de la práctica.

Objetivos

- Definiremos nuestras propias clases y métodos para el programa: **DadesReproductor**, **CtrlReproductor, ReproductorAudio, LlistaFitxers, LlistaReproducció**
- Aprovecharemos las clases **ReproductorBasic, FitxerAudio**
- La clase **FitxerAudio** heredará de la clase **File** de Java y además añadirá sus propios atributos y métodos
- Se debe seguir el patrón de diseño **MVC**
- El código debe seguir los **patrones de estilo** dados por la guía proporcionada por SUN
- El código debe tener las secciones de comentarios pertinentes para la correcta generación de la documentación utilizando **JavaDoc**

Análisis

Previsión y organización

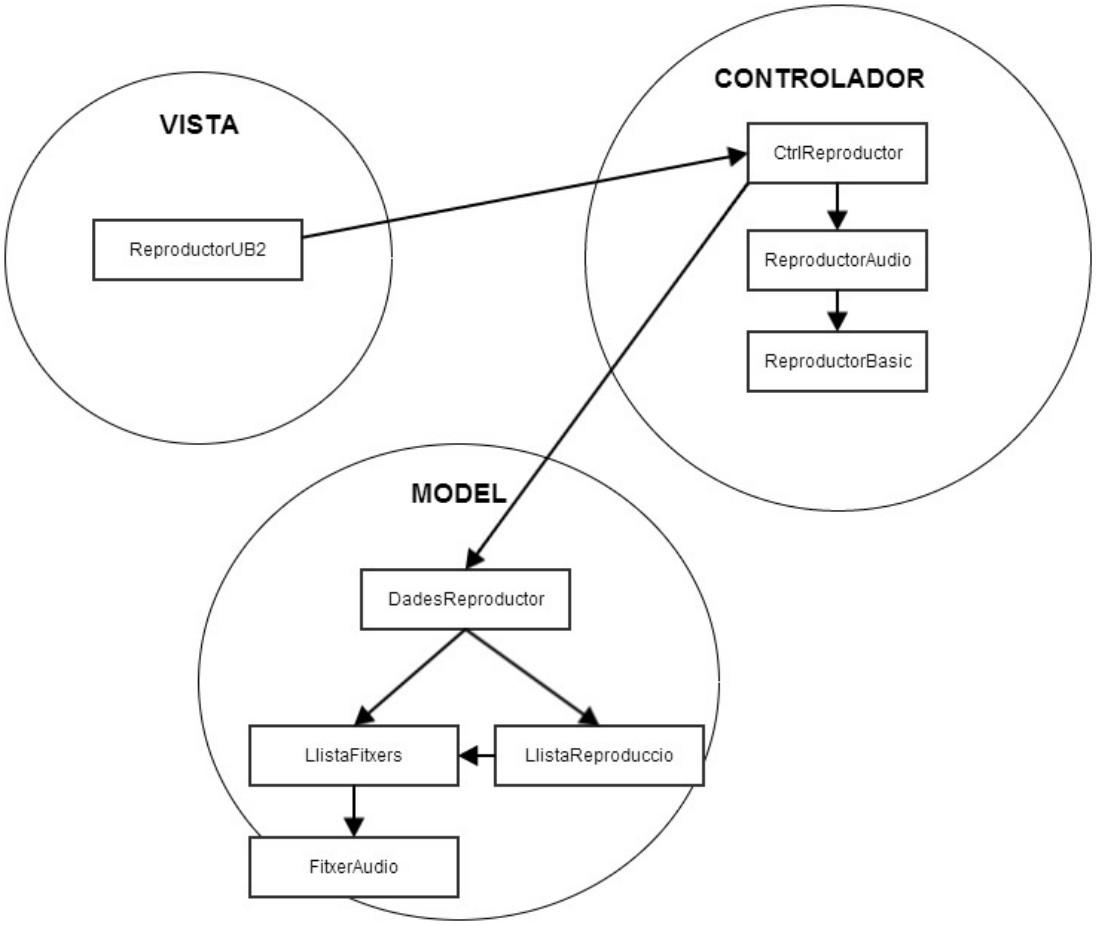
Para la correcta organización del trabajo es importante tener clara la especificación de la práctica para definir correctamente la estructura de datos a utilizar, por lo tanto es importante construir un primer esqueleto de las clases con todos sus atributos/métodos y tener claro que es lo que queremos que haga cada cosa.

Guión de trabajo

- Generar la estructura/esqueleto de las clases.
- Crear JavaDoc para los métodos y tener claro que hará cada función
- Implementar menú de opciones/lógica del programa
- Implementar los métodos y probar las clases descritas en el punto anterior (por orden):
    a. modelo
    b. controlador
    c. vista
- Implementación de las opciones de los menús (por este orden):
    a. Gestión de Biblioteca
    b. Gestión de ficheros, dentro de una lista de reproducción
    c. Gestión de Listas de Reproducción
- Implementación de las opciones guardar y cargar de un fichero (Streams)
- Fase final de pruebas y comprobaciones

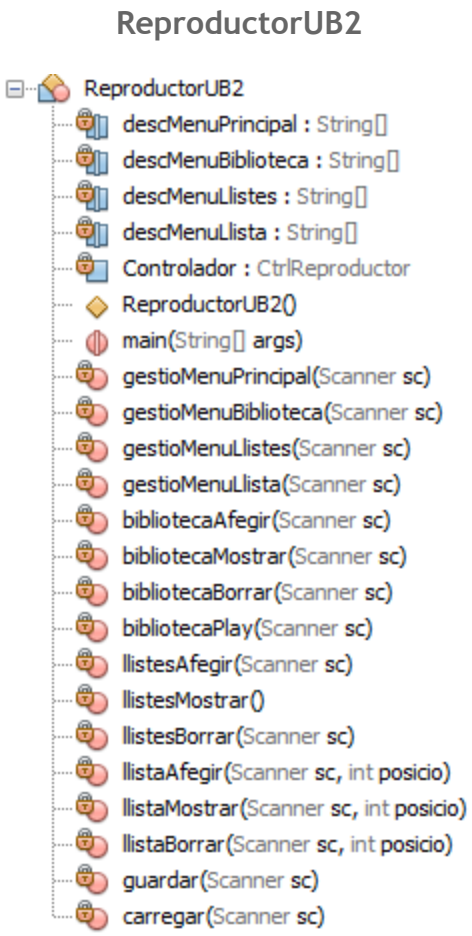
Desarrollo

Esquema MVC y Clases

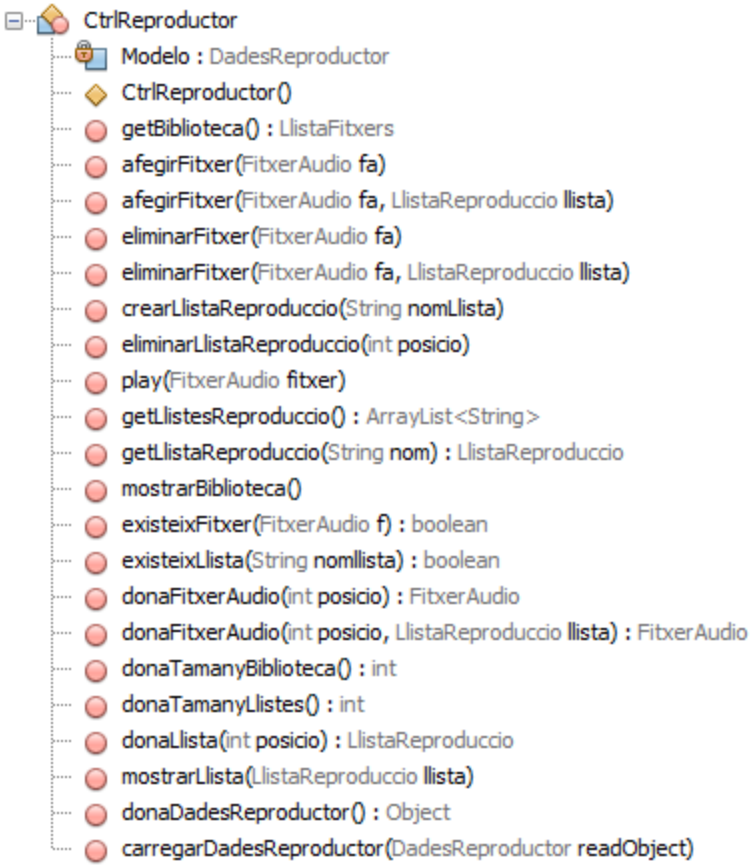


Métodos y funciones

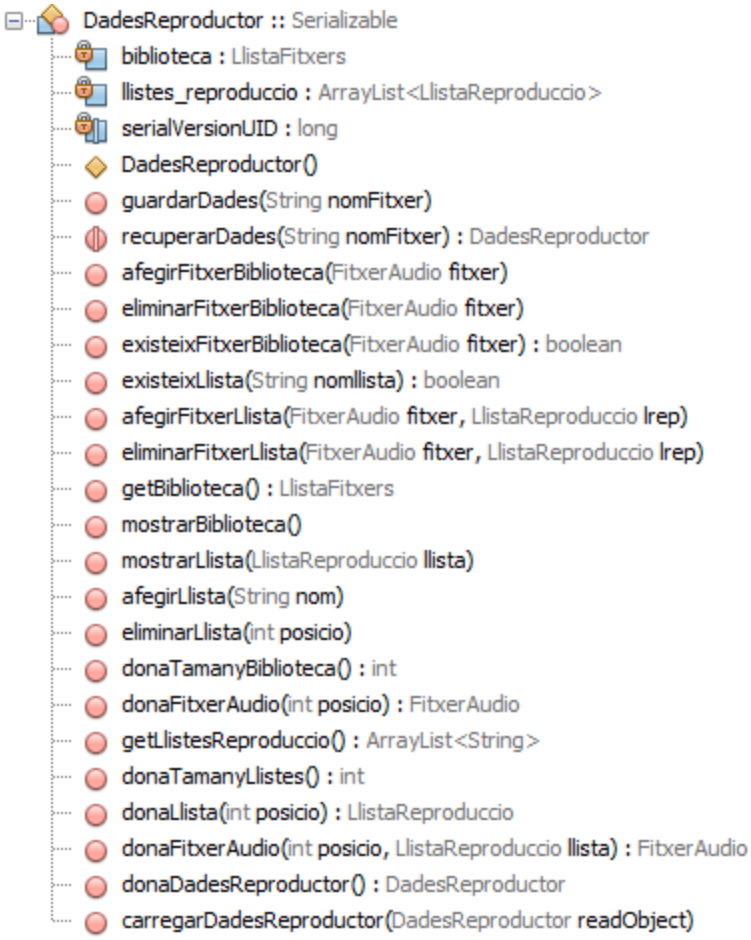
Finalmente, tras la implementación la estructura del programa ha quedado de la siguiente manera:



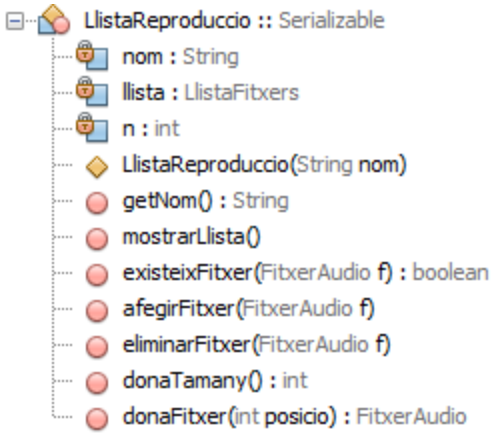
CtrlReproductor

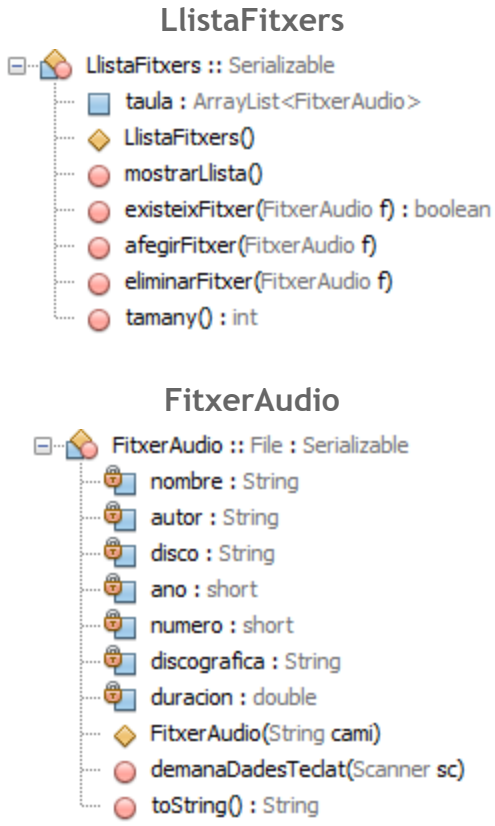


DadesReproductor



LlistaReproduccio





Que has podido reutilizar del código anterior?

- Toda la clase FitxerAudio.
- ArrayList de práctica 1B para implementar LlistaFitxers.
- Partes de la vista ReproductorUB, menú principal, etc.

Problemas y soluciones

A continuación explicaré para mi caso particular, los problemas con los que me encontré y como decidí solucionarlos:

- **Guardar/Cargar la estructura de datos a disco**

  Después de toparme con algunos problemas a la hora de guardar, decidí guardar todo el objeto DadesReproductor, que contienen la estructura global de todo el programa. A la hora de cargar los datos surgió un problema debido a la serialización de los datos, después de investigar por internet vi que se recomienda definir la variable serialVersionUID para todos los objetos serializables, así lo hice y se resolvió mi problema.

Resultados

Pruebas realizadas

Igual que en la última práctica hice pruebas con la ayuda de un fichero de test, además de manualmente, probando una a una las opciones del menú y verificando que se cumplían las limitaciones y reglas impuestas en el enunciado.

Ejemplos de Output

```
---------------------------------
MENU BIBLIOTECA
---------------------------------
        1.- Afegir Fitxer Audio
        2.- Mostrar biblioteca
        3.- Eliminar fitxer
        4.- Reproduir cançó
        5.- Menu Anterior
---------------------------------
Entra una opcio >> 2
Has triat la opció 2

[1] | Titol: Long Live A$AP | Autor: A$AP Rocky | Disc: Long Live A$AP | Pista: 1 | Discogràfica: PirateBay | Duració: 4.49 | Any: 20

[2] | Titol: Long Live A$AP | Autor: A$AP Rocky | Disc: Goldie | Pista: 2 | Discogràfica: PirateBay | Duració: 3.12 | Any: 2013 | Fit

[3] | Titol: Long Live A$AP | Autor: A$AP Rocky | Disc: PMW | Pista: 3 | Discogràfica: PirateBay | Duració: 3.54 | Any: 2013 | Fitxer
```

```
------------------------------
MENU LLISTES DE REPRODUCCIÓ
------------------------------
        1.- Afegir Llista
        2.- Mostrar llistes
        3.- Eliminar llista
        4.- Gestionar llista
        5.- Menu Anterior
------------------------------
Entra una opcio >> 2
Has triat la opció 2

[1] | Llista ASAP
------------------------------
MENU LLISTES DE REPRODUCCIÓ
------------------------------
        1.- Afegir Llista
        2.- Mostrar llistes
        3.- Eliminar llista
        4.- Gestionar llista
        5.- Menu Anterior
------------------------------
Entra una opcio >> 4
Has triat la opció 3

[1] | Llista ASAP
Dona la posicio de la llista que vols gestionar:
```

Responde a pregunta:

> *Explica com has implementat el mètode per eliminar un fitxer d'àudio de la biblioteca. S'eliminarà aquest fitxer d'àudio de totes les llistes de reproducció?*

Sí, el método lo he implementado de manera que cuando un archivo se elimina de la biblioteca se busca también en las listas de reproducción y en caso de existir se elimina también.

Conclusiones

Una vez más vemos las ventajas del uso de objetos y especialmente del uso de la modularización del código y las ventajas que esto conlleva cuando queremos extender funcionalidades de nuestro programa.