Universidad de Barcelona

Arthur Font

Cristian Rodríguez

Proyecto de Prácticas

Programación II

Práctica 3 – 05/05/2019

Barcelona

2019

Index

1. Introducción

El objetivo de la práctica es implementar un album para poder gestionar sus archivos y reproducirlos. Utilizamos tanto reproducción de un solo arcivo como reproducción continua de varios archivos, ya sea con la biblioteca o con un album. Encima impletaremos los controles básicos de la reproducción. A su vez, trabajamos el modus operandi de modelo, vista y controlador. Además profundizamos el estudio y manejo de las heréncias y las interfícies.

2. Análisis

Al principio discutimos las funciones de las clases a implementar y definimos las relaciones entre ellas.
Hemos querido realizar esta práctica yendo desde el terreno mas ambigüo hasta lo más específico. Comenzando así por el menú, que al principio no realizaba nada ya que era solo era estructural. Continuamos con la implementación de la estructura para guardar los diferentes tipos de ficheros. Ahora ya podemos empezar a implementar las funciones de la aplicación como por ejemplo reproducir un archivo, poner modo aleatorio, etc.

3. Desarrollo

Hemos creado la clase AlbumFitxerMutlimedia que funciona a partir de ArrayList de tipo FitxerMultimedia y también hemos implementado las funciones de reproducción de archivos multimedia. También hemos creado la clase Reproductor y EscoltadorReproduccio que sirven para gestionar la reproducción de los archivos.

Implementación de la clase Reproductor:

```
public class Reproductor extends ReproductorBasic implements Serializable{
    public Reproductor(EscoltadorReproduccio controlador){
        super(controlador);
    }

    public void reprodueix(FitxerReproduible fr) throws AplicacioException {
        super.play(fr);
    }
    public void reprodueix(Audio audio, File fitxerImatge) throws
```

```
AplicacioException {
        super.play(audio, fitxerImatge);

    }
}
```

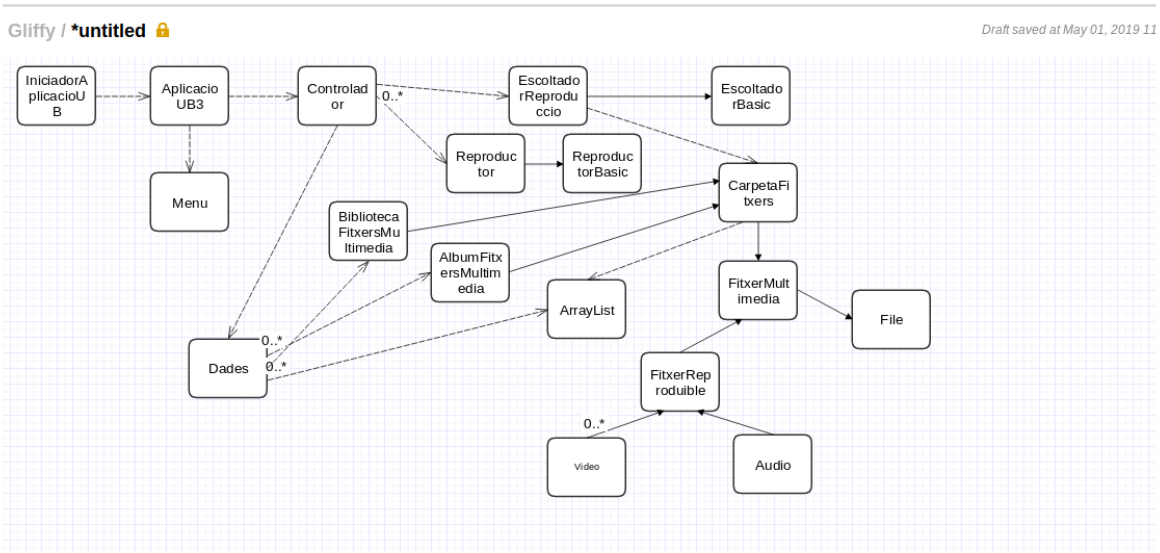
Implementación de los metodos principales de clase EscoltadorReproduccio:

```
public void iniciarReproduccio(CarpetaFitxers llistaReproduint, boolean
reproduccioCiclica) throws AplicacioException {
        this.llistaReproduint = llistaReproduint;
        this.llistaCtrl = new boolean [llistaReproduint.getSize()];
        Arrays.fill(llistaCtrl, Boolean.FALSE);
        this.reproduccioCiclica = reproduccioCiclica;
        this.id = 0;
        FitxerMultimedia f;
        f = llistaReproduint.getAt(id);
        ((FitxerReproduible) f).reproduir();
        llistaCtrl[id] = true;
    }

    @Override
    public void onEndFile() {
        FitxerMultimedia f;
        if (this.hasNext()) {
            this.next();
            f = llistaReproduint.getAt(id);
            try {
                ((FitxerReproduible) f).reproduir();
                llistaCtrl[id] = true;
            } catch (AplicacioException ex){}
        }
    }

    @Override
    protected void next() {
        if (this.isReproduccioAleatoria()) {  //aleatorio
            id = (int) Math.round(Math.random()*(llistaReproduint.getSize()-1));
            boolean trobatFalse = false; int i = 0;
            while (i < llistaCtrl.length && !trobatFalse) {
                trobatFalse = !llistaCtrl[i];
                i++;
            }
            if (!trobatFalse) {
                Arrays.fill(llistaCtrl, Boolean.FALSE);
            }
            while (llistaCtrl[id]) { //se ha reproducido?
```

```
            id++;
            if (id == llistaReproduint.getSize()) {
                id = 0;
            }  //aqui es aleatorio y no se ha reproducido
        }
    }
    else {
        if (id+1 == llistaReproduint.getSize()) {
            id = 0;
        }
        else id++;
    }
}


@Override
protected boolean hasNext() {
    if (this.isReproduccioCiclica()){
        return true;
    }else {
        int i = 0;
        boolean reproduit = true;
        while (i < llistaCtrl.length && reproduit) {
            reproduit = llistaCtrl[i];
            i++;
        }
        return !reproduit;
    }
}
```

4. Respuestas

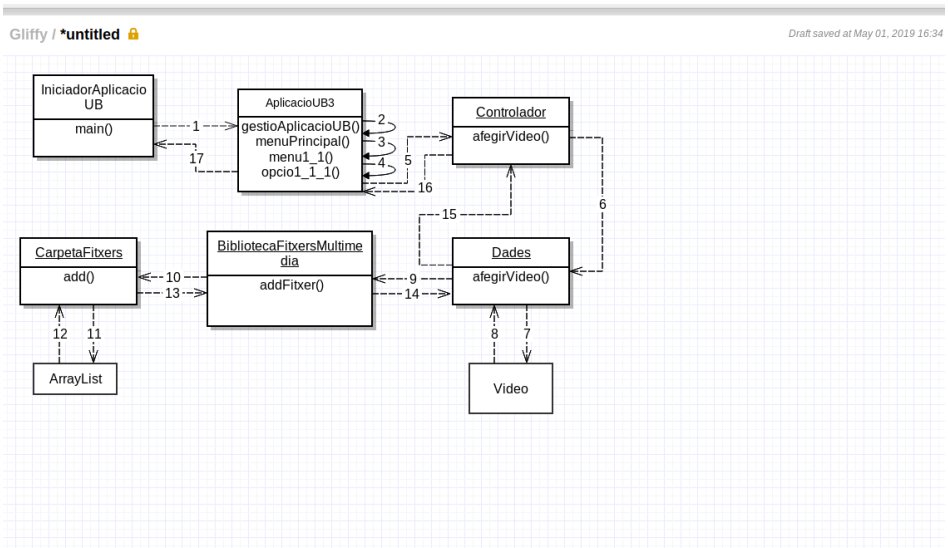
1)



2) Creamos dos atributos privados de tipo boolean y los modificamos a partir de setter y getters. En función de si los boolean son true o false hacemos diferentes “cases”. La implementación del next() y del hasNext() està en el apartado desarrollo mostrado anteriormente.

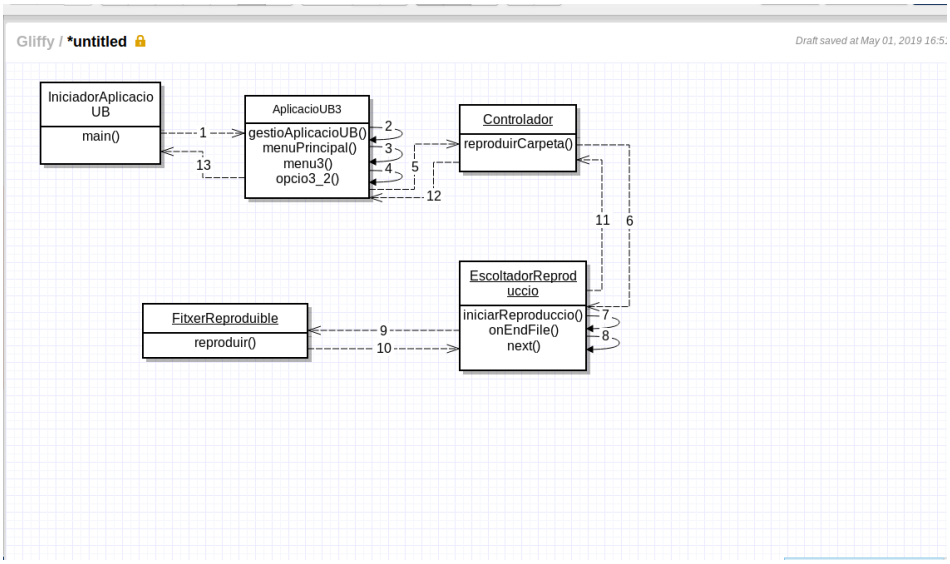
3) Creamos una carpeta de fitxers con un solo FitxerMultimedia para asi no tener que diferenciar entre la reproducción de la biblioteca entera, de un album o, como en este caso, de un fitxer.

4)



5) Llamamos desde el main al metodo reproduirCarpeta sin parámetros para despues en el controlador pasar por parametro la biblioteca directamente con el reproductor, la biblioteca se trata como si fuera una carpetaFItxers ya que hereda de esta misma.

6)



Comprobamos si existe la carpeta a reproducir y despues comprobamos que no esta vacio. Si no hay problemas pasamos la carpeta al reproductor directamente y la reproducimos, la reproducción es basada en orientación a eventos. Si hay problemas lanzamos la excepcion adecuada.

7) Crear un reproductor multimedia ha sido muy interesante ya que es el primer proyecto que es tangible pues estamos reproduciendo ficheros reales.