FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

Departamento de Computación

Algoritmos y Programación I (75.40)

1° Cuatrimestre de 2017

Profesor Titular: Wachenchauzer, Rosa

Alumno: Martin Feldsztejn
Padron: 101320

Alumno: Carlos Martín Stefanelli D'Elias
Padron: 100488

Grupo: A8
Fecha: 05/06/2017
Practica: ALAN
Corrector: Gaston

El programa está diseñado utilizando objetos y delegación. Cada objeto es el encargado de saber como manejar su información.

La clase Shell comunica al usuario y al command executor. Este a su vez conoce al archivo que esta en cargado y que sabe cómo operar sobre sí mismo.

Por otro lado están los DTOs que guardan informacion, las clases Mark, Note, Sound y SoundWrapper.

Clases de ejecución
La clase Shell es la encargada de comunicarse con el usuario es agnóstica a la lógica y simplemente se encarga de redirigir los parámetros al método correcto del CommandExecutor.
La clase CommandExecutor es la encargada de ejecutar los comandos pedidos. Guarda una representación del archivo que fue leído.

Clases de manejo de archivos
La clase SongFile guarda tanto los sonidos como una lista de Mark para poder trabajar sobre la canción.
La clase Mark es la representación de una marca de tiempo en una canción. Guarda el tempo de la misma y que notas de los distintos canales van a sonar.
La clase FileManager se encarga de abrir el archivo de dos formas: lectura y escritura. Es capaz de convertir un archivo a un SongFile y viceversa.

Clases de reproducción
La clase Song guarda una sucesión de _Note, sabe como reproducirse a si misma.
La clase _Note guarda una lista de los sonidos que deben ser reproducidos en la marca actual (a diferencia de la clase Mark que guarda cuales deben sonar y cuales no) y con que tempo.
La clase SongPlayer se encarga de armar una Song y reproducirla.

TDAs
Usamos los TDAs LinkedList y Stack. Ambos fueron implementados por nosotros. Para eso usamos una clase _Node para la lista que contiene un dato a guardar y el nodo que le sigue y una lista de python para la Pila ya que provee de manera simple la funcionalidad que tiene la misma

Alternativas al momento de armar el diseño
No se nos ocurrieron muchas alternativas al momento de armar el diseño ya que la idea que pedía la consigna del trabajo práctico fue muy precisa y directa.

Requerimientos no funcionales
Los requerimientos no funcionales fueron claves a la hora de modelar el proyecto. El requerimiento de la utilizacion de objetos nos parecio correcto y muy importante para el diseño del mismo.

El requerimiento del TDA de una LinkedList fue correcto ya que al tener que insertar más DTOs con esta implementacion era poco costoso a nivel tiempo dando una buena performance desde este lado.
De haber tenido que ser implementado con funciones La complejidad del mismo hubiese aumentado ya que no se hubiera podido guardar el estado y todos los datos iban a tener que ser enviados entre las mismas constantemente.