TRabajo práctico especial 2016

Programación Orientada a Objetos

Integrantes: López Bidone, Nicolás

LU: 247409

Email: nicobidone@gmail.com

Ayudante: Juan Bautista, Feldman

Grupo: 06

# Introducción

El presente trabajo práctico especial, tiene como objetivo la representación de un sistema de administración de música, el cual da la opción de organizar una colección musical de acuerdo a ciertos criterios a partir de la información que contiene. La misma está conformada por dos tipos de elementos, las canciones y las listas de reproducción las cuales a su vez tienen la capacidad de anidarse en otras listas.

Durante este informe, se realizará la descripción de las consideraciones que se utilizaron al momento de resolver la consigna dada. Focalizando en el diseño, clases y métodos más importantes, además de las decisiones que llevaron a la resolución del ejercicio solicitado.

# Clases Colection, PlayList y Song

Mediante el análisis inicial, se identificaron los objetos que interactúan principalmente en la resolución de los servicios solicitados en las consignas. Estos son, el elemento canción y la lista de reproducción, cuyos comportamientos fueron representados mediante las clases **Song** y **PlayList** respectivamente.

Con el objetivo de implementar una solución orientada a objetos y aplicando polimorfismo, se abstrajeron determinados comportamientos en una clase que fue denominada **Colection**, en referencia a las colecciones de música de las que se habla en el enunciado. La misma reúne comportamientos que poseen en común **Song** y **PlayList**, con el fin de facilitar la interacción entre las mismas. En este caso, el rol de la clase básica que es representado está dado por **Song**, mientras que la clase compuesta, la cual referencia a todos los elementos del tipo **Colection,** es llamada **PlayList**.

Aquellos comportamientos que fueron abstraídos, hacen referencia al método de búsqueda utilizado basado en filtros (que determinan si el elemento del tipo **Song** cumple o no los requisitos), la cantidad de elementos del tipo **Colection** que existen en un objeto dado (ya sea una canción o una lista de reproducción), la duración total de cada elemento y la forma de representar la información que se contenida en los diferentes objetos.

En lo que refiere a estas operaciones, las diferencias que se dan entre las clases, se basan en que **Song** al ser un elemento simple puede responder por si misma a las consultas. Mientras que **PlayList** debe realizar un recorrido, solicitando la realización de estos procedimientos a los diferentes elementos que contiene, para de esta forma retornar una respuesta propia.

Dado, que no se especificó que el servicio **setPosicion** (clase **PlayList**) el cual cambia de lugar una canción determinada, realizara alguna acción particular para los casos que los valores de posición final y/o inicial estuvieran fuera del rango, de la cantidad de elementos que contiene una lista de reproducción. Se aplicaron las siguientes acciones para modelar excepciones:

* Si solo la posición final se encuentra fuera de rango, se mueve el elemento a la última posición de la lista.
* Si la posición inicial se encuentra fuera de rango no realiza ninguna acción.

Si bien existe una clase **AutoPlayList** que parece incorporar nuevos comportamientos a la estructura de la solución, en realidad la misma depende de **PlayList** dado que utiliza sus procedimientos. Esto se da mediante la creación de una **PlayList** dinámica, generada a partir de una búsqueda aplicando un filtro determinado, la misma se solicita cada vez que se realiza una consulta a un procedimiento de un objeto de esta clase.



*Diagrama de dependencia de las clases principales*

# Filtros

Los Filtros, se utilizan para realizar búsquedas dentro de la colección musical a partir de ciertos criterios, utilizando los atributos de la clase **Song** como parámetros. Allí, pueden observarse dos tipos, los filtros o criterios basados simplemente en atributos, y aquellos que definen operaciones lógicas, las cuales permiten realizar combinaciones con los primeros. A partir de estos hechos, al observar el diagrama presentado, se puede advertir la multiplicidad de las sub-clases del tipo **Filtro**.

Esta representación sin embargo no es aleatoria, si bien en un inicio se intentó abstraer aún más la representación de la información, senotó que para este caso solo se añadía complejidad a la solución, sin ver una mejora significativa en cuanto a la facilidad de uso. Por lo tanto, se limitó la abstracción de información al procesamiento de los datos que son obtenidos a partir de los atributos de la clase **Song**.



*Filtros utilizados en la resolución de la guía*

# Comparador

Dentro de los criterios de comparación basados en atributos, se pueden diferenciar aquellos cuyo contenedor es del tipo String y los que son del tipo Integer. Para facilitar las operaciones con números enteros se implementaron los comparadores, los cuales realizan procedimientos de diferenciación tales como saber si un número es mayor, menor o igual a otro.



*Comparadores implementados*

# Conclusión

A lo largo del proyecto, se han encontrado diversos problemas, que se fueron resolviendo a medida que se fue avanzando en el mismo sin mayores implicaciones. Se logró realizar todo el código en Java referido al trabajo principal, logrando el objetivo de implementar para este problema una solución orientada a objetos.