Author: Tyler Despatie

Student Number: 101010622

COMP 2404 Assignment 3 Refactoring Part II

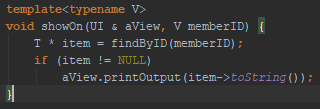
FR 3.2.1) I don’t like how ostream & operator<<(ostream & out, const Song & aSong) appears in Song, Track, Recording, and Playlist.

FR 3.2.2 / FR 3.2.3) I believe it is suitable to remove this code and place it in our template whilst maintaining the same functionality. This removes unnecessary duplication of functions.

FR 3.2.4) We are able to call the .toString() functions that belongs to Song, Track, Recording, and Playlist from our template to obtain the same functionality.

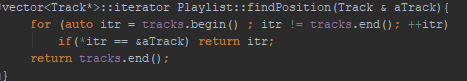
FR 3.2.1) I don’t like how showOn(UI, int) is incorporated in Songs, Tracks, Recordings, and Playlists.

FR 3.2.2 / FR 3.2.3) I believe it is suitable to remove this code and place it in our template whilst maintaining the same functionality. This removes unnecessary duplication of functions.

FR 3.2.4) We are able to call showOn(UI, V) where V is a template for Song, Track, Recording, and Playlist to obtain the same functionality in our template.

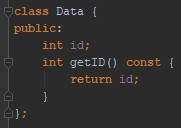
FR 3.2.1) I don’t like how each iterator that appears in Song, Track, Recording, and Playlist has vector<Playlist/Track \*>::iterator a type name like this.

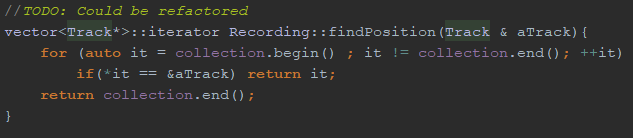
FR 3.2.2 / FR 3.2.3) I believe it is suitable to remove this and replace it with the “auto” keyword as per C++11 standards. This helps the readability and complexity of the code in my opinion.

FR 3.2.4) We can make our variable definitions less cluttered and more readable as well as maintain the same functionality by writing them like this.

FR 3.2.1) I don’t like how id is a variable in each of the classes Song, Track, Recording, Playlist, and User and each of these classes implement a getID() function.

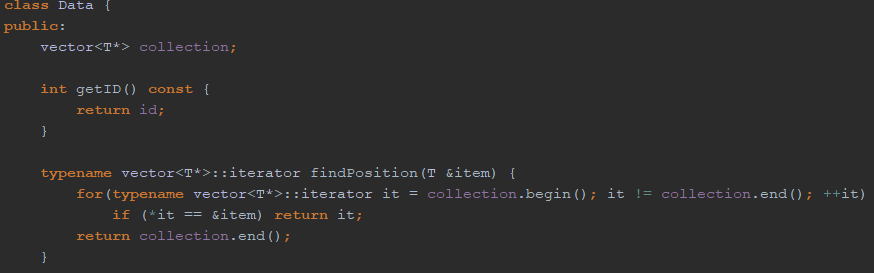
FR 3.2.2 / FR 3.2.3) I believe it is suitable to make a parent class called Data and have each of these classes inherit from Data. This will remove a handful of variable declarations and functions from each class.

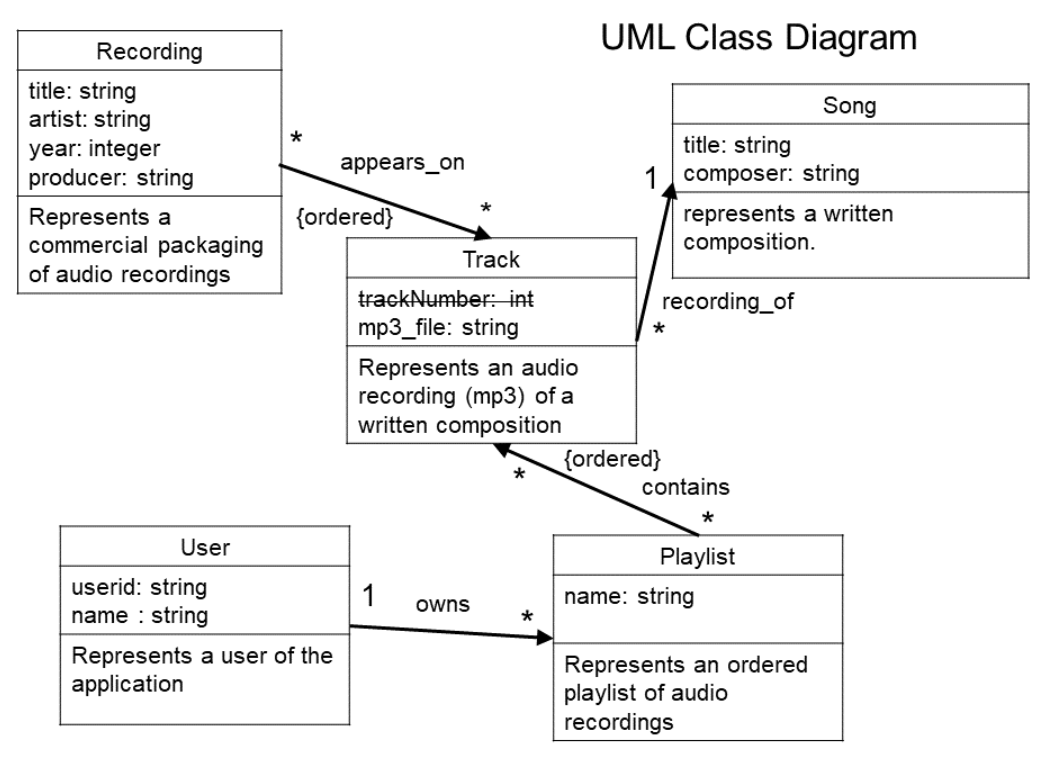
FR 3.2.4) We can remove several functions from the project and have a parent class to keep things tidier. Data will contain the variable id and have the function getID() implemented inside of it.

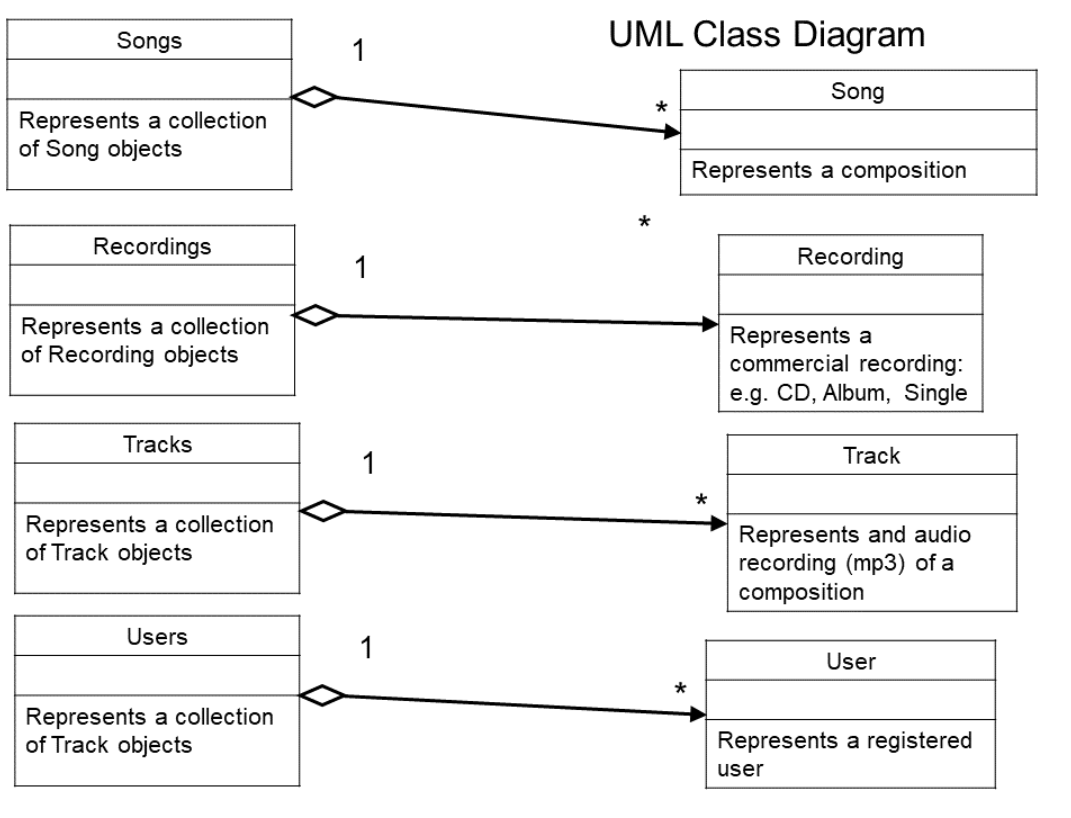
FR 3.2.1) I don’t like how findPosition() is a function in Playlist, User, and Recording.

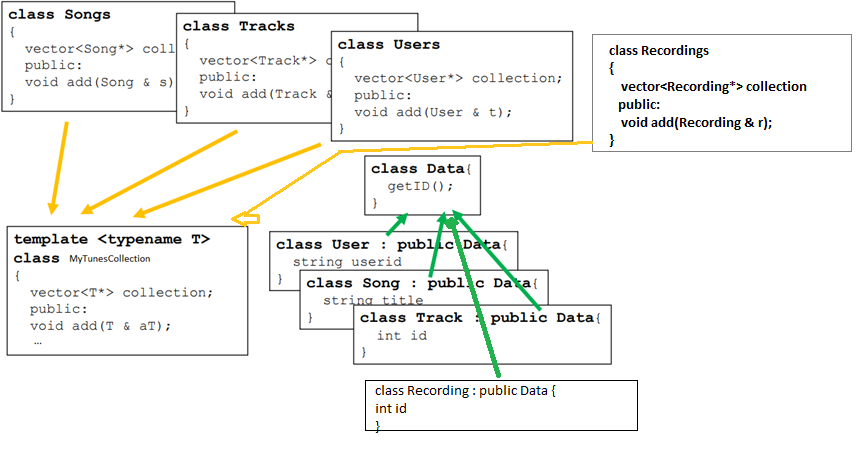
FR 3.2.2 / FR 3.2.3) I believe it is suitable to add this function to the parent class Data because it is duplicated in 3 places and can be made a generic function.

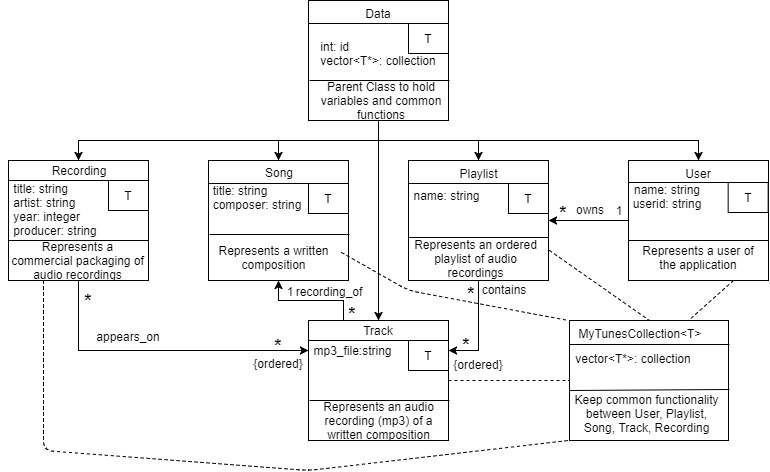
FR 3.2.4) We can remove 3 functions from our project by placing it in the Parent class, along with a generic vector “vector<T\*> collection” because most of the classes inheriting from data require it.



FR 3.2.5) This is a UML Diagram of what our project looks like **before any factoring** has been done.



This is roughly what our project currently looks like **after refactoring** has been completed.

This is what the finished product looks like in UML form.

As you can clearly see, we’ve been able to eliminate 4 different container classes (Users, Recordings, Tracks, Songs) by using a template. We were also able to eliminate a multitude of functions by using inheritance as well (Recording, Song, Playlist, User, and Track inherit from Data).