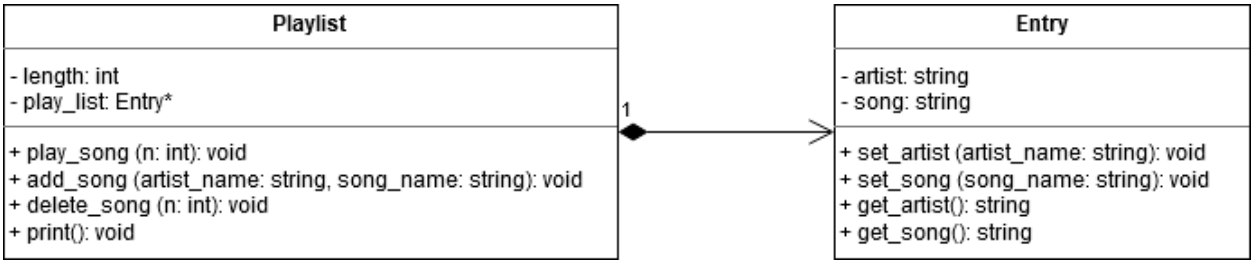
Design Document Project 0

Overview of Classes

Two classes were designed, a Playlist class, and an Entry class. The Playlist class was designed to store a playlist of songs along with the related artist. The Entry class was designed to contain information about the entries in a playlist (song/artist name).

The relationship between the two classes is that the Playlist class uses the Entry objects to contain the information for each song in the playlist. The Playlist class then manipulates the data contained in the Entry objects using the public member functions of the Entry class.

UML Class Diagram



Design Decisions

Playlist

The constructor uses dynamic memory allocation to create an array (Entry) of a specified size and the member variable pointer (*play_list)* can then be used to access all the Entry objects in the array. The constructor then assigns a variable to store the length of the playlist so it can then be used in other functions to check that only elements within bounds of the array are accessed.

The destructor deallocates the memory for the playlist which was allocated in the constructor and reassigns the pointer (*play_list*) to nullptr. These two actions are designed to prevent memory leaks and dangling pointers from occurring.

No operators were overwritten. The keyword *const* was used in all member functions as well as the constructor for their parameters. This was appropriate since in each case the parameter being passed into the function should not be altered or reassigned within the function.

Entry

The constructor for the Entry class assigns an empty string to both *artist* and *song* to ensure that the attributes of each object are empty. This identifier is used in the Playlist class to identify Entry objects which have not been updated with an inserted song & artist name.

No destructor was required since there was no dynamic memory allocation in the constructor. No operators were overridden.

The const keyword was used for the parameters of the *set_artist* and *set_song* functions since the parameter passed into the function should not be altered, the parameter is simply being assigned to the attribute of the object.

Test Cases

To test if my program met the specified requirements, I created specific inputs to try and cover the many possible inputs which could cause problems or require extra checks to unwanted errors.

Some specific cases which I tested for:

- Removing a song from the playlist using an index outside of the bounds of the playlist array
- Trying to remove a song from an index in the playlist which is empty.
- Adding a duplicate song
- Adding a duplicate song with different casing than the version on the playlist
- Trying to add a song to the playlist when the playlist is full.
- Trying to add a song to the playlist using an index outside the bounds of the playlist array.
- Including special characters in a song or artist name
- Trying to play a song using an index outside the bounds of the playlist array.
- Trying to play a song using an index in the playlist which does not contain a song.
- Trying to delete a song using an index outside the bounds of the playlist array.