• **MusicPlayer::MusicPlayer(SoundController*)** this constructor takes a heap allocated concrete derived class of `SoundController` and sets it as the `MusicPlayer`'s sound controller. The `MusicPlayer` is taking ownership of this object.

• **void MusicPlayer::setSoundController(SoundController*)** - this method takes a pointer to a heap allocated concrete derived class of `SoundController` and sets it as the `MusicPlayer`'s sound controller. Must also unset the current song for the `MusicPlayer`. The `MusicPlayer` is taking ownership of this object. Do not forget to free any old data!

• **void MusicPlayer::addPlaylist(string)** - this method takes in a string, and creates a new playlist with that name. If a playlist with that name already exists then nothing is done.

• **void MusicPlayer::addAlbum(string name, string artist)** - this method takes in two strings, and creates a new album with that name and that artist. If an album with that name and artist already exists then nothing is done.

• **void MusicPlayer::addSong(Song)** - this method takes in a song and adds it to the library. It is guaranteed that name and artist form a unique pair for each song, so if the song passed in has the exact same name and artist as a song already in your library nothing is done. **Note:** You may choose to have this parameter be a reference, just know that the lifetime of the argument is not guaranteed after the life of the function.

• **void MusicPlayer::removePlaylist(string)** - this method takes in a string and removes the playlist with that name. If a playlist with that name does not exist then nothing is done.

• **void MusicPlayer::removeAlbum(string name, string artist)** - this method takes in a name and an artist and removes the corresponding album. Removing an album does not remove the songs that were in it from the library. If the album with that name and artist does not exist then nothing is done.

• **void MusicPlayer::removeSong(string name, string artist)** - this method takes in a name and an artist and removes the corresponding song from the library, including any albums or playlists it is in. If the song removed is the currently active song then `MusicPlayer::stop()` must be called and the current active song must be unset. If the song with that name and artist does not exist then nothing is done.

• **void MusicPlayer::setSong(string name, string artist)** - this method will set the currently active song to the song in the library with the given name and artist. If such a song does exist in the library then any other music that was playing is stopped before changing the song. Must also call `setSong` on the current `SoundController` with that song’s filepath. If the song does not exist in the library then nothing is done.

• **void MusicPlayer::play()** - this method simply calls `play()` on the currently set `SoundController`.

• **void MusicPlayer::pause()** - this method simply calls `pause()` on the currently set `SoundController`.
• void MusicPlayer::stop() - this method simply calls stop() on the currently set SoundController.

• Song* MusicPlayer::findSong(string name, string artist) - if the song with the provided name and artist exists in the library then a pointer to that song is returned. If the song does not exist then a null pointer is returned.

• Album* MusicPlayer::findAlbum(string name, string artist) - if the album with the provided name and artist exists then a pointer to that album is returned. If the album does not exist then a null pointer is returned.

• Playlist* MusicPlayer::findPlaylist(string) - if the playlist with the provided name exists then a pointer to that playlist is returned. If the playlist does not exist then a null pointer is returned.

• void Playlist::addSong(string name, string artist) - if the song with the provided name and artist exists in the library then it is added to the playlist. Otherwise it is not.

• void Playlist::removeSong(string name, string artist) - if the playlist contains a song with the given name and artist then it is removed from the playlist. Otherwise nothing is done.

• void Album::addSong(Song) - if the song does not already exist in the album then it is added. The song must also be added to the library as well if it does not already exist in the library. If the song does already exist in the library a second copy must not be made. Note: You may choose to have this parameter be a reference, just know that the lifetime of the argument is not guaranteed after the life of the function.

• void Album::removeSong(string name, string artist) - if the song with the given name and artist exists in this album then it should be removed from the album. Otherwise nothing is done.

• Song(string name, string artist, string fPath, size_t duration) - Basic parameterized constructor for song. Takes in a string for the name, artist, filepath, and a size_t for duration and creates a Song with all those fields stored.

• ostream& operator<<(ostream&, const Song&) - the output operator for a song. Must print out the name, artist, and duration with hyphens separating them (and spaces on either end of the hyphen). Use the sample executable to determine if you are correctly printing out the songs.

• ostream& operator<<(ostream&, const Album&) - the output operator for an Album. Must print out the name of the album, followed by a hyphen (with spaces on both sides) followed by the artist, followed by a hyphen (with spaces on both sides) followed by each of the titles of the songs in the album separated by a comma with a space after it. The songs must be printed in alphabetical order. Compare with the executable to determine if you are correctly printing out the songs.

• ostream& operator<<(ostream&, const Playlist&) - the output operator for a Playlist. Must print out the name of the playlist, followed by a hyphen (with spaces on both sides) followed by each of the titles of the songs in the playlist separated by a comma with a space after it. The songs must be printed in alphabetical order. Compare with the executable to determine if you are correctly printing out the songs.