**CRC**

**Definitions**

**User's Library:** The group of songs that are apart of the Users collection of music.
**Directory:** A folder containing files and other directories. In our case it this will be song files and other folders that will contain songs.

**UI**

| UI/GUI module. | **Controller** |
|---|---|
| Will send requests for UI data and user actions to the controller | |

**Controller**

| Wrap utility classes. | **SongManager** |
|---|---|
| Update the persistent storage | **PersistentStorage** |

**Song Manager**

| Manages 2 list of Libraries. One that is in the user's Library and the other that is the "external" Library | **Library** |
|---|---|
| Will query for songs in file system. Using FileManager. READ ONLY Access | **PersistenceStorage** |
| Will take songs from query and play them by giving it to MusicPlayer | **PlayList** |
| Will take songs from query and add them to playlists | |
| Save the User's Library in the PersistenceStorage | |
| | |

**File Manager**

| Take a path to a dir and create list of Songs | **SongManager** |
|---|---|
| Will use this class to access any file in system. | **Exporter** |
| Will be responsible for Add/Delete/Copy | |

| | |
|---|---|
| operations on files | |
| Should be able to take a path to song file and create a Song object from it | |
| | |

**Library**

| | |
|---|---|
| Holds a list of songs in a folder | **Song** |
| Represents a library that the user wants to load into the program | |
| delete/copy Songs | **FileManager** |

**Song**

| | |
|---|---|
| Represents a song | **Library** |
| There are many songs in a Library | |
| Contains information about the song (metadata) | |
| Contains location of the Song in the file directory | |
| | |

**PersistenceStorage**

| | |
|---|---|
| Save state of the program | **SongManager** |
| Will take information like User Song Library path and save that path in PersistenceStorage | |
| Will initialize SongManager if there is something in the PersistenceStorage. | |
| | |
| | |

**MusicPlayer**

| | |
|---|---|
| Given a Song, play it | **Song** |
| Responsible play, pause, stop, volume | **SongManager** |

| | |
|---|---|
| control of the song playing | |
| Possibly keep track of multiple songs to play in a order. | |
| | |

**Playlist**

| | |
|---|---|
| Hold a list of Songs in the user library that the user wants to group together | **SongManager** |
| A playlist can be exported to the car | **Song** |
| | |

**Exporter**

| | |
|---|---|
| Take a PlayList and Export it to the format accepted by the car. | **PlayList** |
| Will fill in ID3v1 and ID3v2 based on data from song | **FileManager** |
| Copy song in Playlist and then modify copy. Move song to new location specified by user. | |
| | |

**Examples of Execution Flow:**

**User Starts Program:**
Main function is called. Controller is created and checks to see if there is any data in PeristenceStorage. If so then use it to help create the SongManager else this is the first time the user is running the program so have to create a SongManager with no data in the User's Library

**User Provides Path To User's Library:**
SongManager receives request to add a Library and creates a Library object called myLibrary based on the specified folder path. The Library constructor will call the generateSongs(folderPath) in the FileManager class which returns a list of Song objects. The constructor will then create a Library based on that.

**User Provides Path To Other's Songs:**
Similar case as previous one just instead Library is called externLibrary.

**User Wants To Play Song:**
SongManager receives play request, should be given a song identifier. Query to find the song from in the List to get the Song Object. Give the song Object to the MusicPlayer.

**User Wants to Delete Song**
SongManager receives delete request (with song name) and calls Library.deleteSong(). deleteSong() will then use a FileManager object to delete the file.

**User Wants to Copy Some Songs and Import to User's Library**
SongManager gets request to move song(s) from Others to User's Library. It uses Library.copySong() function which uses a FileManager object to copy from destination to source.