Our project included many different ADTs such as Linked Lists, Array List, and Maps. We implemented each which we thought would be best for timing and efficiency. First, we implemented the collection of Playlists as a List of pointers. This list, playlist list, is currently implemented as an array list but has an interface class that will allow it to be replaced with a linked list implementation if desired. The playlist list contains pointers to playlist objects. Playlists are implemented as Linked nodes, that contain pointers to song objects which are stored in our second data system. Our second system is our song library. We implemented the song library as an artist map ADT, implemented with Linked nodes. Each artistMapNode contains an artist key value and a pointer to a list of songs by that artist, stored in alphabetical order. This list, called songList/songArrayList, is implemented as an array list but also has an interface class to allow it to be switched for a linked list implementation. Each Songlist holds that actual song objects corresponding to the artist.

We gave a fair amount of thought to how we implemented each ADT. We decided to make an Array List for the collection of playlists (playlists list) because we frequently need to randomly access a given playlist, by its index, which with an ArrayList implementation we can do in O(n) time. We implemented the playlists themselves Linked Nodes because we commonly need to remove songs from and add songs to the playlists. Adding a song with either implementation is usually O(1) but arrays will also need to be resized from time to time. Removing songs with the linked list implementation does require walking down the list O(n), but with an ArrayList, you have to shift all songs after the removed one down which is also O(n).

For the song library, I decided to use an Artist map because it seemed like the most obvious way to store songs by an artist. I chose a Linked node implementation because the artist are stored in alphabetical order which requires frequent insertions into the middle. For the Songlist because of the frequency of random indexing calls, and because songs are very rarely removed from or added to the song library. I think a linked node implementation, would be very reasonable as well because, like the artist nodes, songs are stored in alphabetical order and so are frequently inserted in central indices.

Lastly, we created functions for the user interface in Command Handler. The Command Handler allows the user to interact with both the List of playlists Artist Map. Command Handler also uses two text files Save which saves all songs in the current library (Song Array List) and Saved Playlist which writes to these files as SavedPlaylist (saves Playlist Array List). The Command Handler also allows us to import these files automatically if they are filled with songs when we open up our DjPlayer again.

**Memory Diagram:**

|  |
| --- |
|  |