Jeffrey Kaufmann
Music Playlists

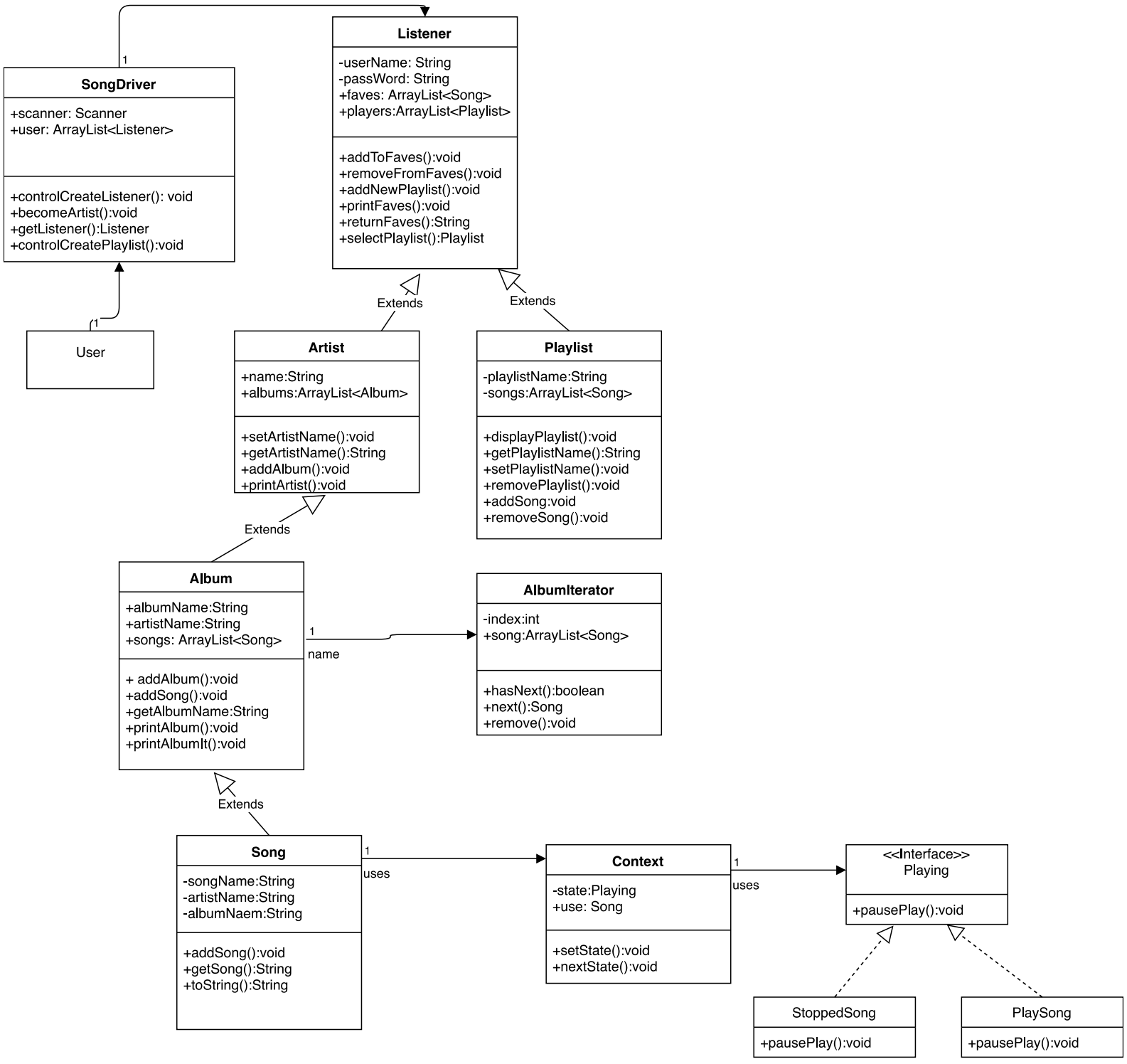
Project Description:
My idea for a project would be to make a music application to create playlist and favorites from your favorite artist and songs. You would be able to add songs to a playlist or your favorites. You could also be an artist and upload a song to an album as well.

Implemented:

Users:	Finished	Notes:
Listener	101: Login with Username and Password 102: Playlist begins empty and users can add a playlist whenever 103: Listeners begin with an empty favorites	Was able to get the login information and basic playlist setup. Nothing is left empty
Artist	203: All Albums and Songs must be named, no blank songs	Artist can upload an album but it must be filled

Unimplemented

Users:	Unfinished:	Notes:
Listener	102: Must create a playlist when you create an account 103: Listeners cannot upload album or song	Was unable to set up a playlist when creating account. Listeners technically can now because Artist is just a subclass of Listener
Artists	201: Artists must log in and create and account 202: Must create an album with at least one song 204: Artists cannot create Playlists	Instead of separating the two I just made a check for if the Listener signs up to be an artist. This removed some of the functionallity



Final Class Diagram:

My final class diagram actually change a lot. The user now accesses a SongDriver which wasn't there before. This is a control for all the classes and setup them up to be used. I also added and iterator and state design pattern. I also changed artist to extend Listener instead of it being it's own class. A user can opt into being an Artist and upload an Album or a Song. This change

saved time because instead of having to create new login information for an Artist a user can now be both a Listener and an Artist. I also completely removed the favorites class. In turn I made a default ArrayList of Songs that a Listener has access to immediately when the account is made. This provides that I didn't have to make an entirely new class that the Listener had to create at sign up.

Design Patterns:
Song: State
Album: Iterator

Song uses a state to "Pause and Play" a song. User selects a song and then uses the .nextState() to change the song from paused to played. It is default set to pause initially.

Album uses an iterator to flip through songs. This is useful when I have to display multiple Albums at the same time. For example, when a user enters an artist's page.

What I've Learned:

I've learned that setting up and working on an overall class diagram is much more important because if done properly it can help ease the workload when actually program. I had to backtrack a lot to fix my class diagram's issues which put me pack in the long run. I also wish that I had saved more time for front end work. The back end ate up a lot of my time so working on a jsp file was almost impossible with the time I had left. Overall this project really helped me understand design patterns and how they work as well as implementing an entire java class system.