

1 Project description

Team with Github names:

Badam-Osor Khaltar (badamosor)

Paul McCumber (paulmccumber)

Luke Meszar (lukemeszar)

Title: Music Playlist Manager

Description: Sharing playlists of music between a large group of people can be difficult when everyone doesn't listen to music through the same service. This will allow people to create playlists in a generic format that can then be integrated into the users service of choice. This allows people to share their playlists with their friends who can then export it into the streaming service of their choice.

2 Previous class diagram

Figure 1 shows our class diagram for Part 2.

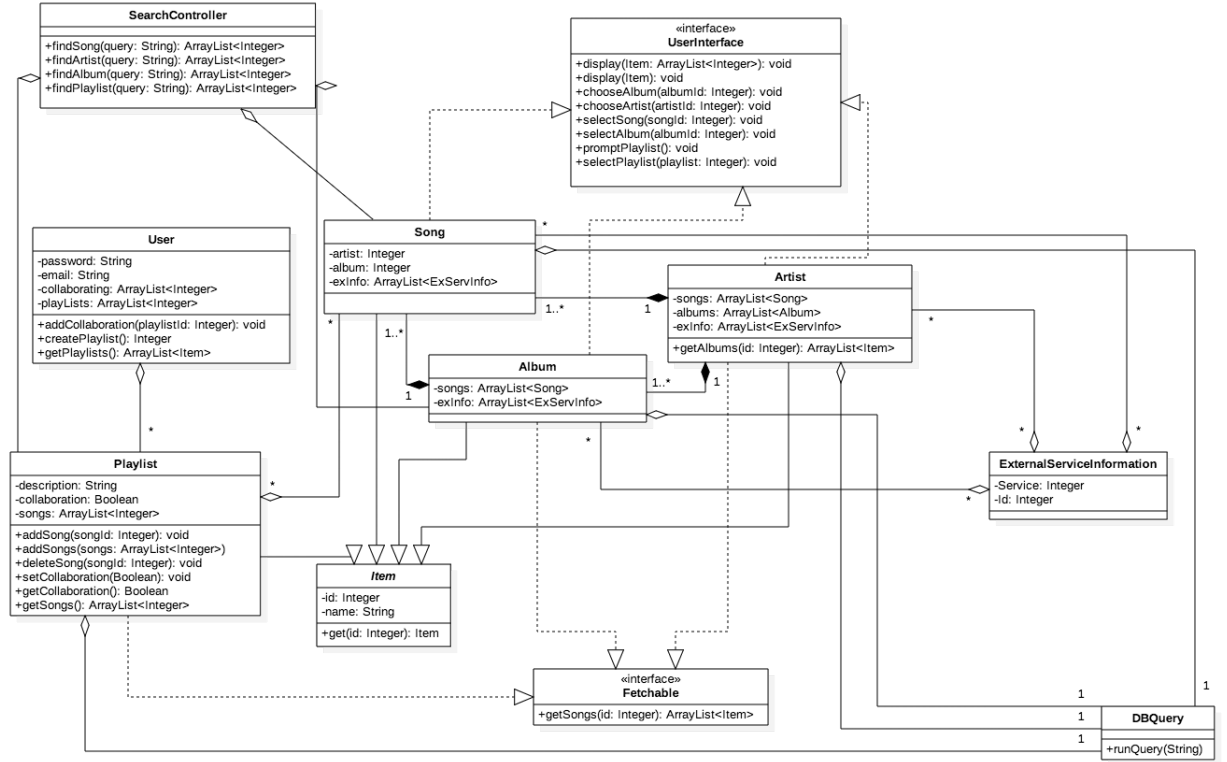


Figure 1: Class Diagram

3 Completed class diagram

We integrated our design for Playlist manager to Django framework. Therefore our class diagram for Part 2 has been changed based on Django framework architecture and model. Figure 2 shows our new class diagram. The classes shown in green color are Django built-in classes. The classes we implemented for our project extends these built-in classes of Django. The classes in blue are implemented using design patterns which we describe in detail in Section 8. Due to the complexity of these classes, we show the diagrams for these classes in Figures 8-10 of the Section 9. All the classes shown in Figure 2 have been implemented.



Figure 3: Create user

– creating playlists

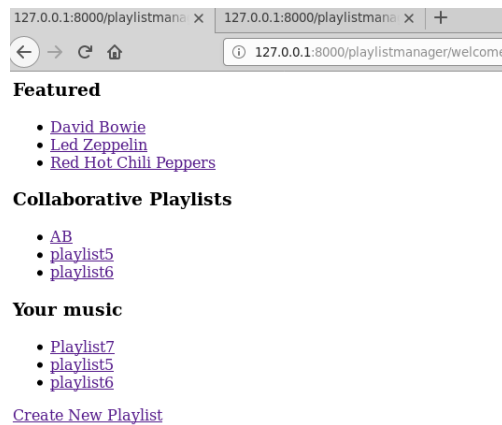


Figure 4: Create playlist

– editing playlists

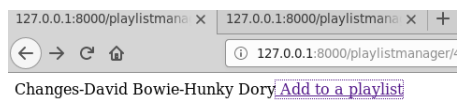


Figure 5: Add to playlist

- exporting to streaming service

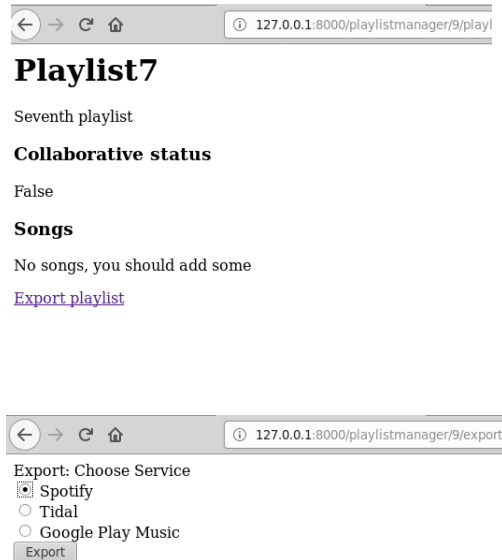


Figure 6: Export to streaming service

- Several design patterns were prototyped including Strategy, Visitor, Observer, and Facade.

The Strategy, Visitor, and Observer patterns were implemented and test drivers written outside of the model. A rough version of the Facade and Strategy were written for the Spotify interface.

Note: Much of the work with the model and the prototyping of design patterns was done in parallel. The nature of the structure of the model was fairly fluid. Therefore we did a separate class diagram for the Search function which will be using Strategy and Visitor. The first task in the next phase will be integrating search into the model.

5 Breakdown

Luke:

- Much of the MVC implementation.

- Got the team rolling in the implementation.
- Redid the project class diagram based on the MVC implementation to date.
- Implemented Facade and Strategy design patterns for Exporter class.

Badam-Osor:

- Much of the MVC implementation.
- Project writeup.
- Implemented Observer pattern.

Paul:

- Implemented the Search prototype with Strategy and Visitor.
- Constructed the class diagram for Search.
- Produced writeups for Design Patterns with markup and examples of the implemented design patterns.

6 GitHub Graph

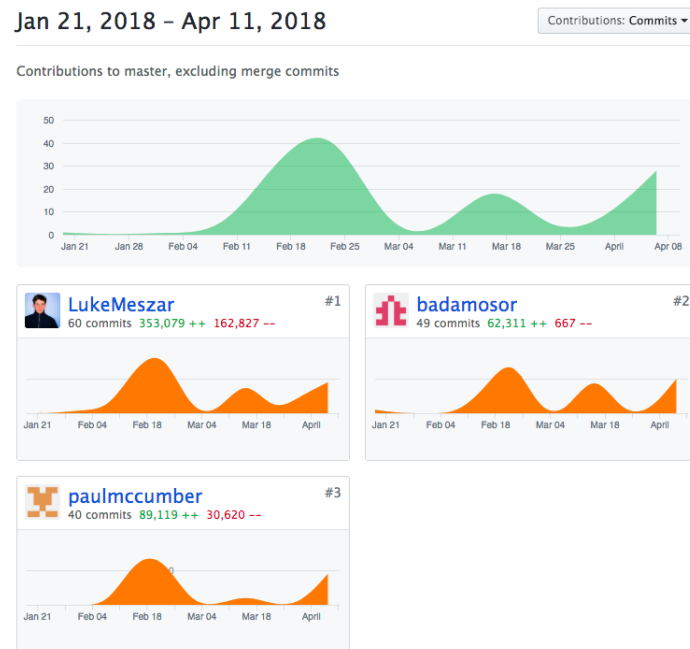


Figure 7: Graph of each member's contribution

7 Estimate of remaining effort

- 6-8 hours integrating the Search into the model.
- 10-20 hours integrating editing of playlists with search results.
- 3-5 hours integrating Observer prototype into the model.
- 30 hours writing rest of functionality.

8 Design Patterns

- Facade - was prototyped in the model to abstract the Spotify interface from the application.

- Strategy - was prototyped in the model as a nice way to export to one of the three streaming services dynamically.
- Visitor - We prototyped one of the search methods using the Visitor design pattern.
- Strategy - We prototyped Search using Strategy and two implementations that can be dynamically switched.
- Observer - We prototyped a playlist that uses the Observer design pattern. Concretely, playlist can be collaboratively edited by its followers. If the collaborative playlist is changed, its followers get notified that a change has occurred.

9 Class diagrams for design patterns

- Figures 8-11 show the class diagram for Strategy, Visitor, Facade and Observer design patterns, respectively.

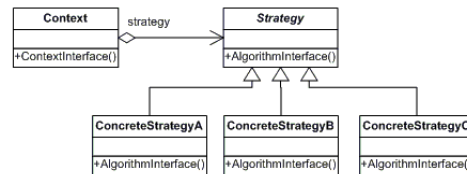


Figure 8: Class diagram of Strategy

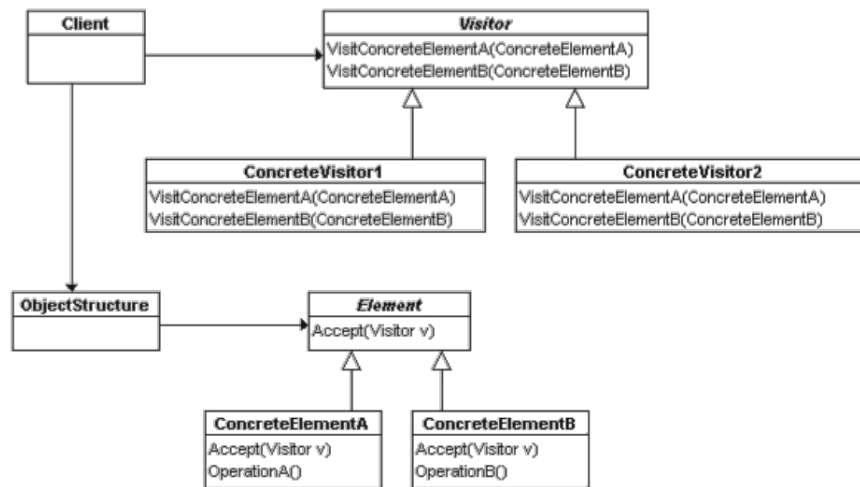


Figure 9: Class diagram of Visitor

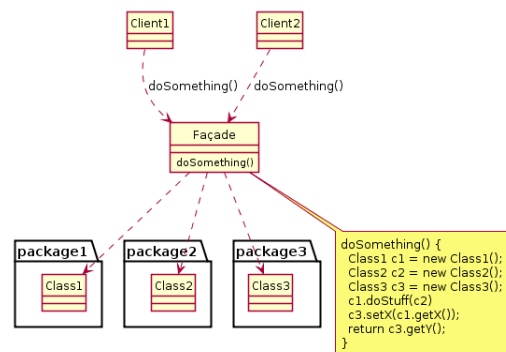


Figure 10: Class diagram of Facade

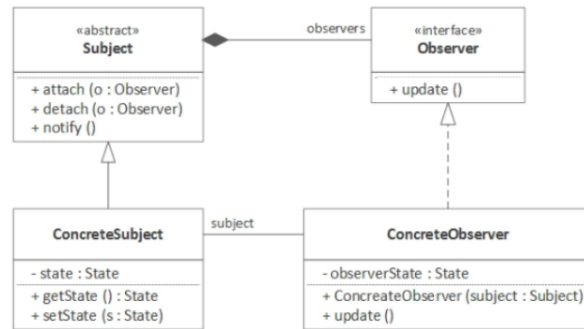


Figure 11: Class diagram of Observer

- Figures 12-14 show the class diagrams that implement above design patterns.

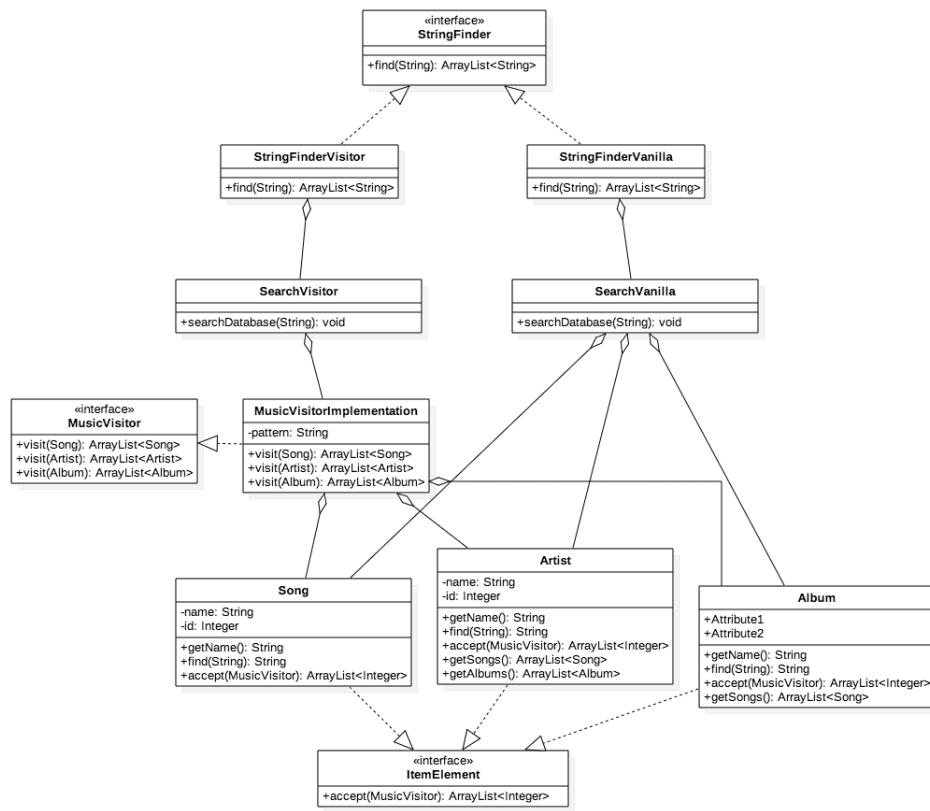


Figure 12: Class diagram of StringFinder

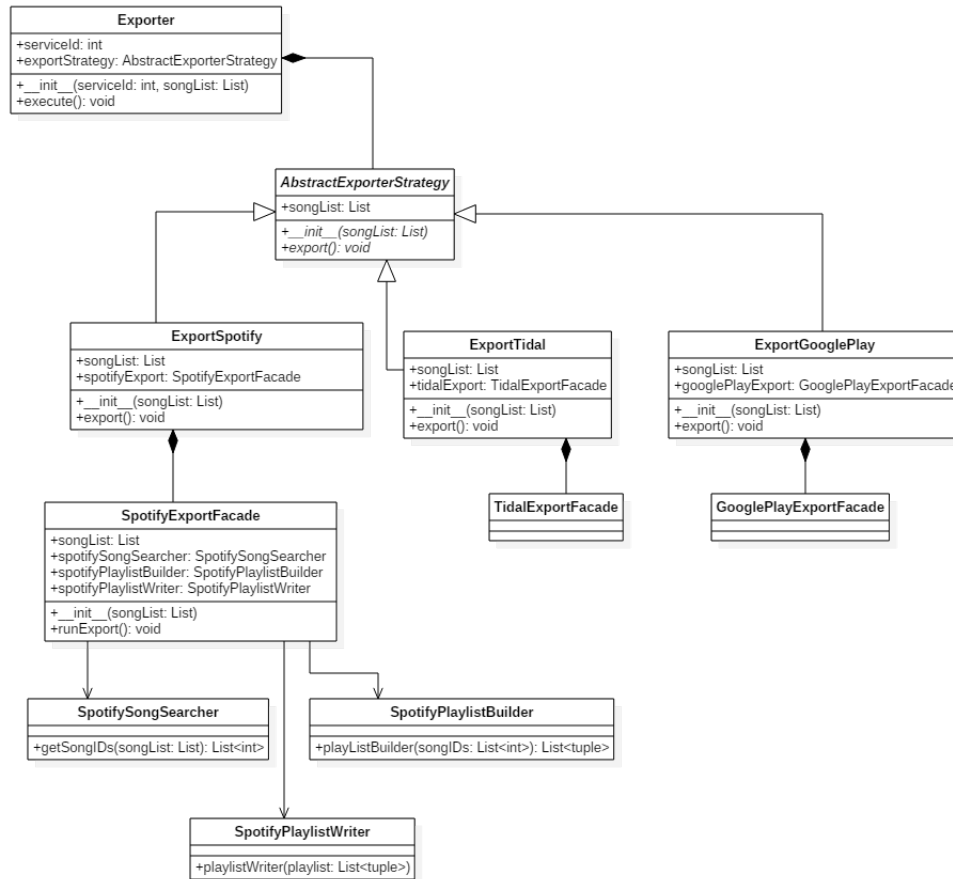


Figure 13: Class diagram of Exporter

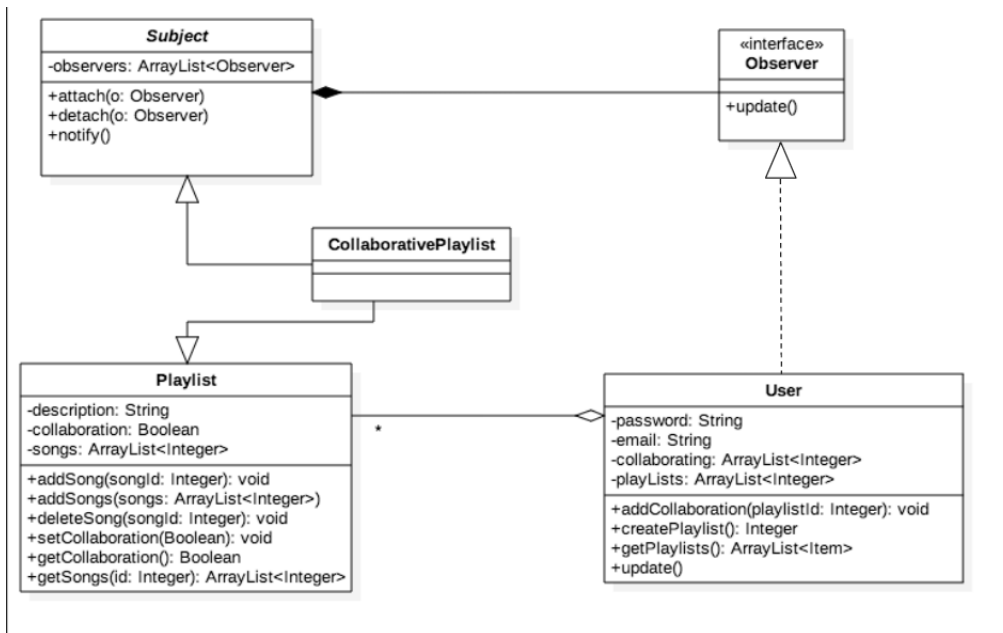


Figure 14: Observer implementation for collaborative playlist

- Figures 15-17 illustrate the participants of the diagrams shown in Figures 12-14, respectively.

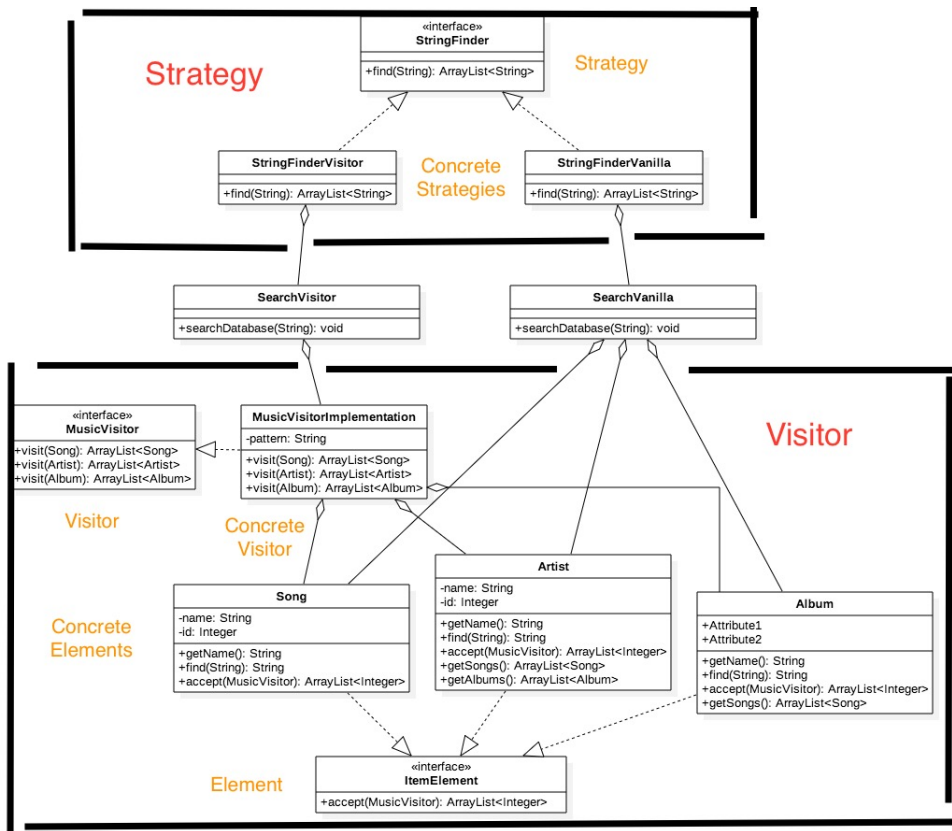


Figure 15: Participants of the class diagram for StringFinder

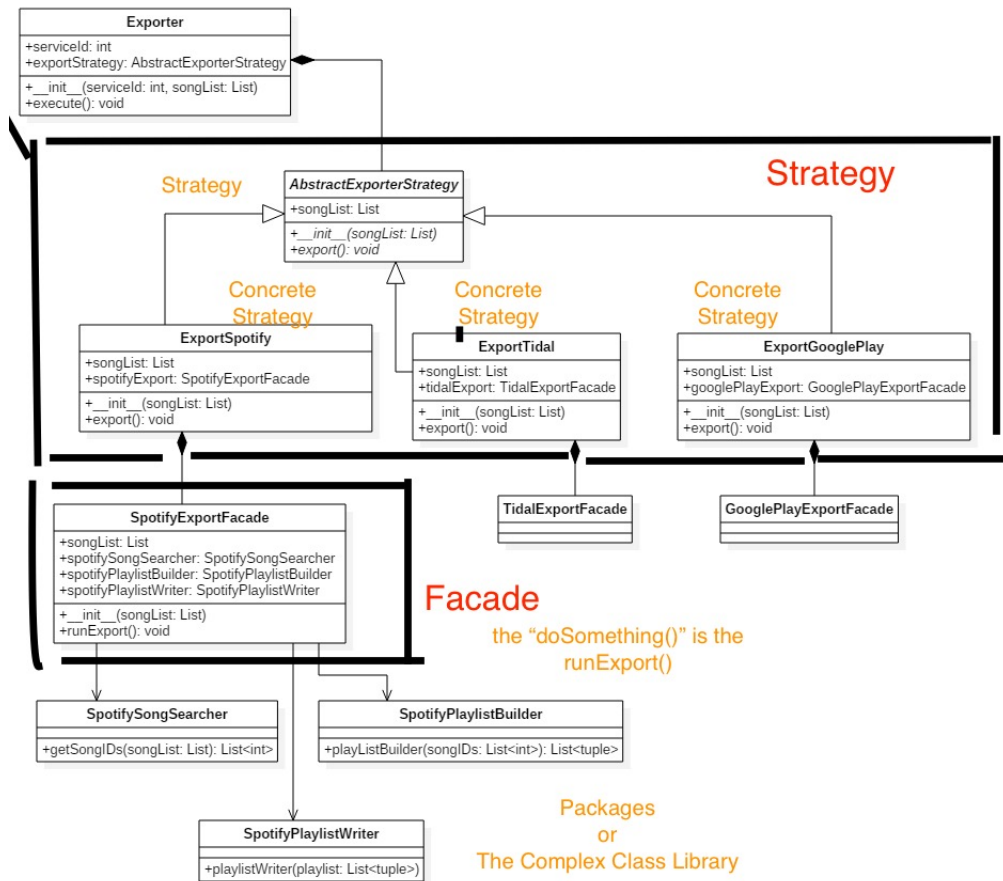


Figure 16: Participants of the class diagram for Exporter

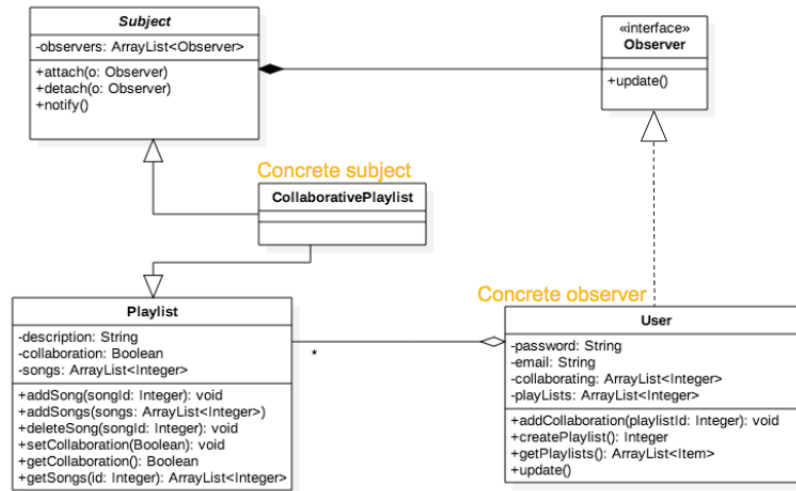


Figure 17: Participants of the class diagram for Observer implementation

10 Final Iteration

We are planning following tasks to be done for the final iteration:

- Integrating the Search into the model.
- Integrating editing of playlists with search results.
- Integrating Observer design pattern into the model.
- Finish the rest of the functionalities.