# MMLS v1.0.0

*Design Documentation*
*Prepared by Team 5:*

- Devan Kavalchek <djk6908@rit.edu>
- Alison Hansen <amh1782@rit.edu>
- John Flory <jaf6598@rit.edu>
- Preston Sergent <pws1453@rit.edu>

Summary 2

Domain Model 3

System Architecture 4

Subsystems 5

Name of the subsystem 5

Name of the subsystem 5

Status of the Implementation 6

Appendix 7

## Summary

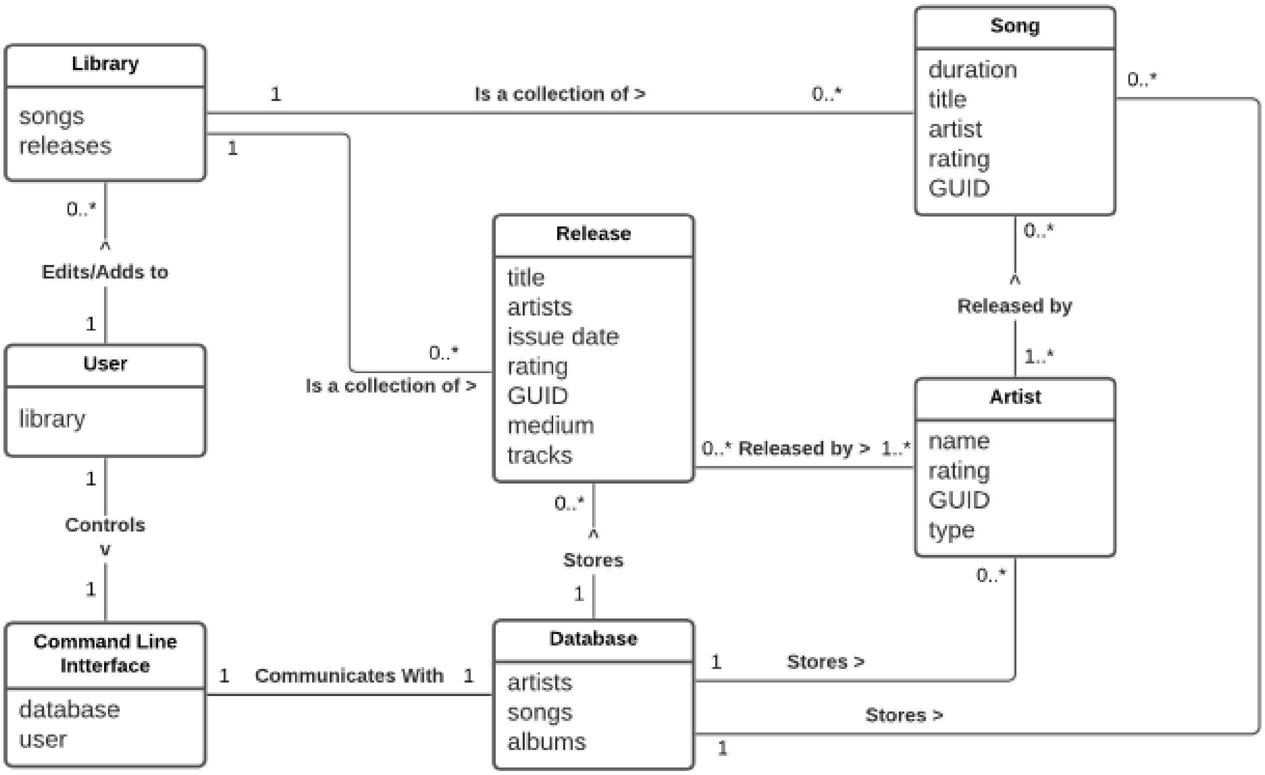
The Muze Music Library System is a music management system that allows users to keep track of their music collection. Users can add songs and releases to their collection, marking when they acquired them.

Users interact with the system via a command line interface. This means that commands are entered via a single line of text. The system responds to these requests with a textual response, reflecting a change that happened within the system and providing feedback to the user.

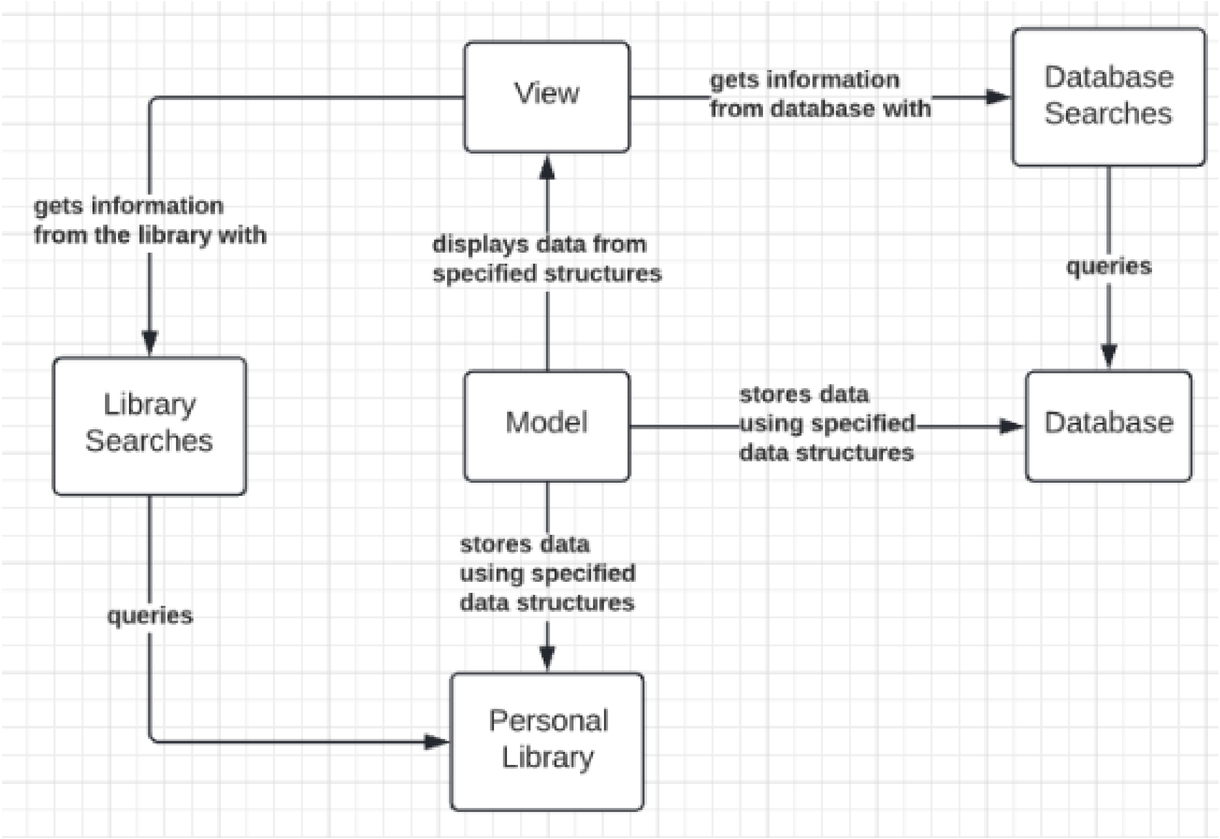
The Muze Music Library System maintains a database of all the available music. It maintains a set of artists, releases and songs. Users can search this database for different attributes such as title, artist, duration, and rating. These songs and releases can be added to the users personal library and rated.

The system maintains a collection of songs and releases that the user has added and rated. This collection is persistent across sessions.

## Domain Model



## System Architecture


View
gets information from database with
Database Searches
gets information from the library with
displays data from specified structures
queries
Library Searches
Model
stores data using specified data structures
Database
stores data using specified data structures
queries
Personal Library

The view handles all interactions with the user. Additionally, this subsystem invokes all actions to other subsystems. The model provides the data structures that are used to store and represent data throughout the entire system. The database and personal library both act as storage. They are queried using objects in database searches and library searches respectively.

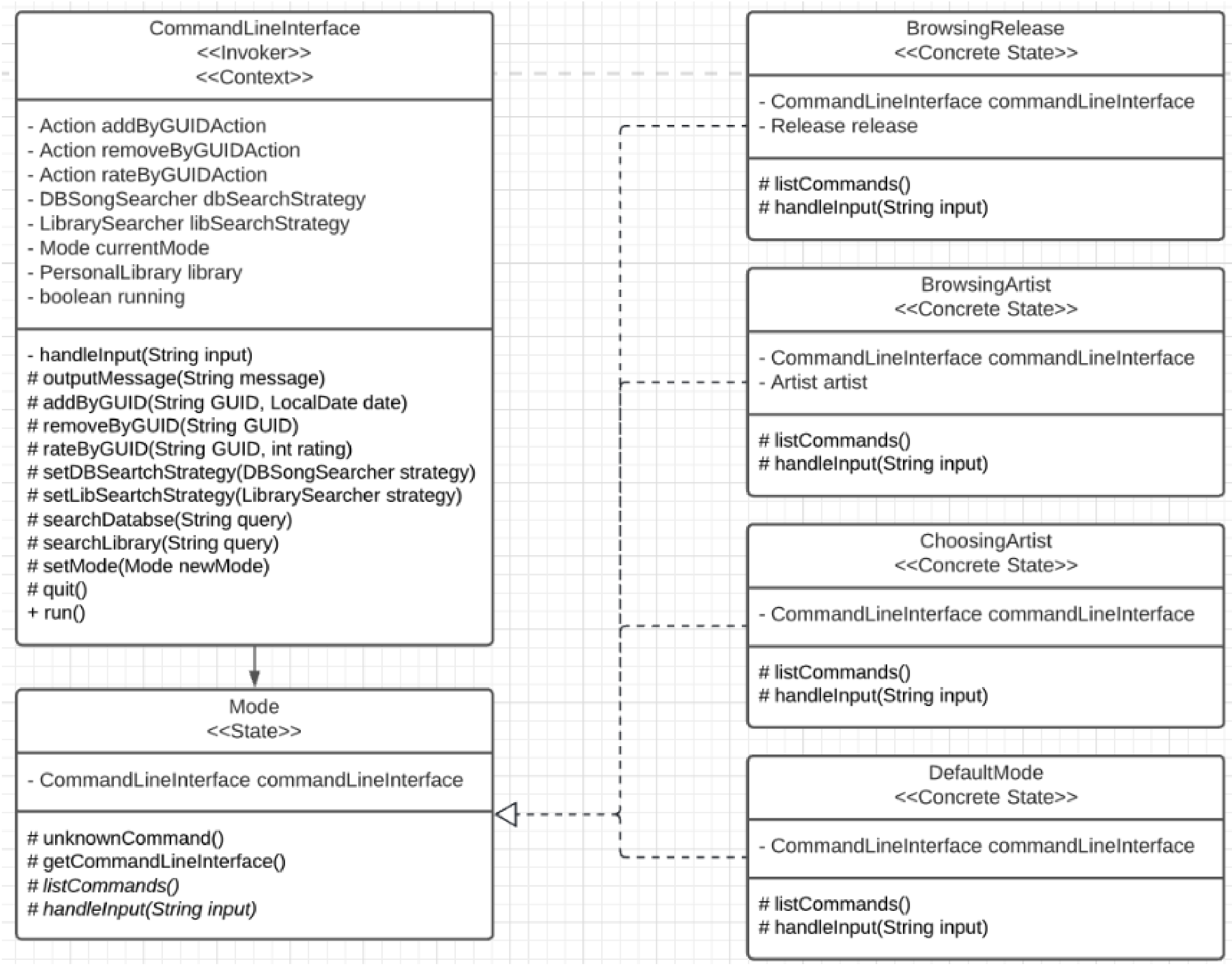
# Subsystems

## *View*

The view controls interactions with the user. It accepts inputs from the user via the command line and responds with output messages accordingly. The view also communicates with the personal library and database to keep track of the users data and to respond to their requests.

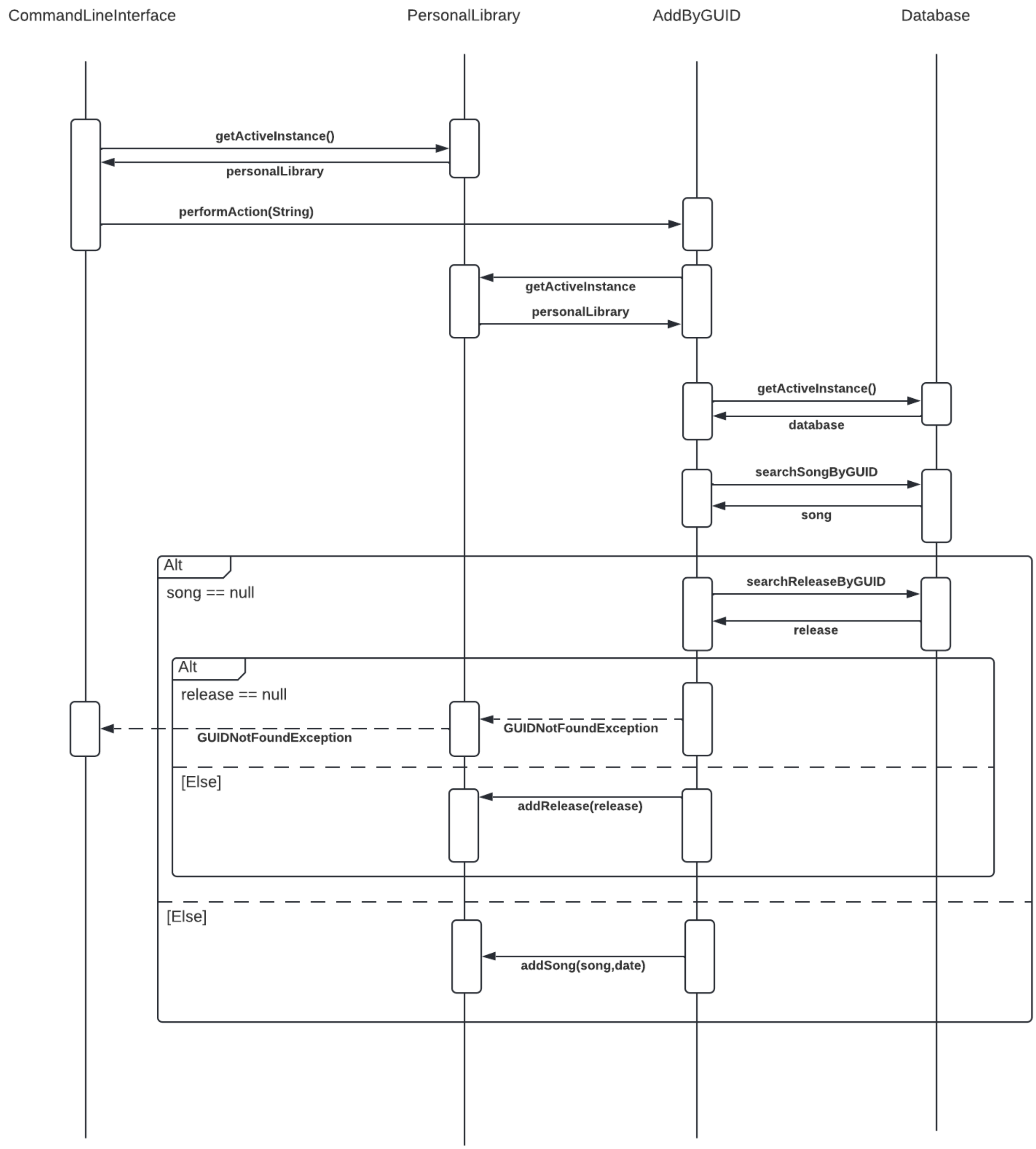
The view has a state, which allows it to handle the user's input differently depending on the state.

### Class Structure Diagram

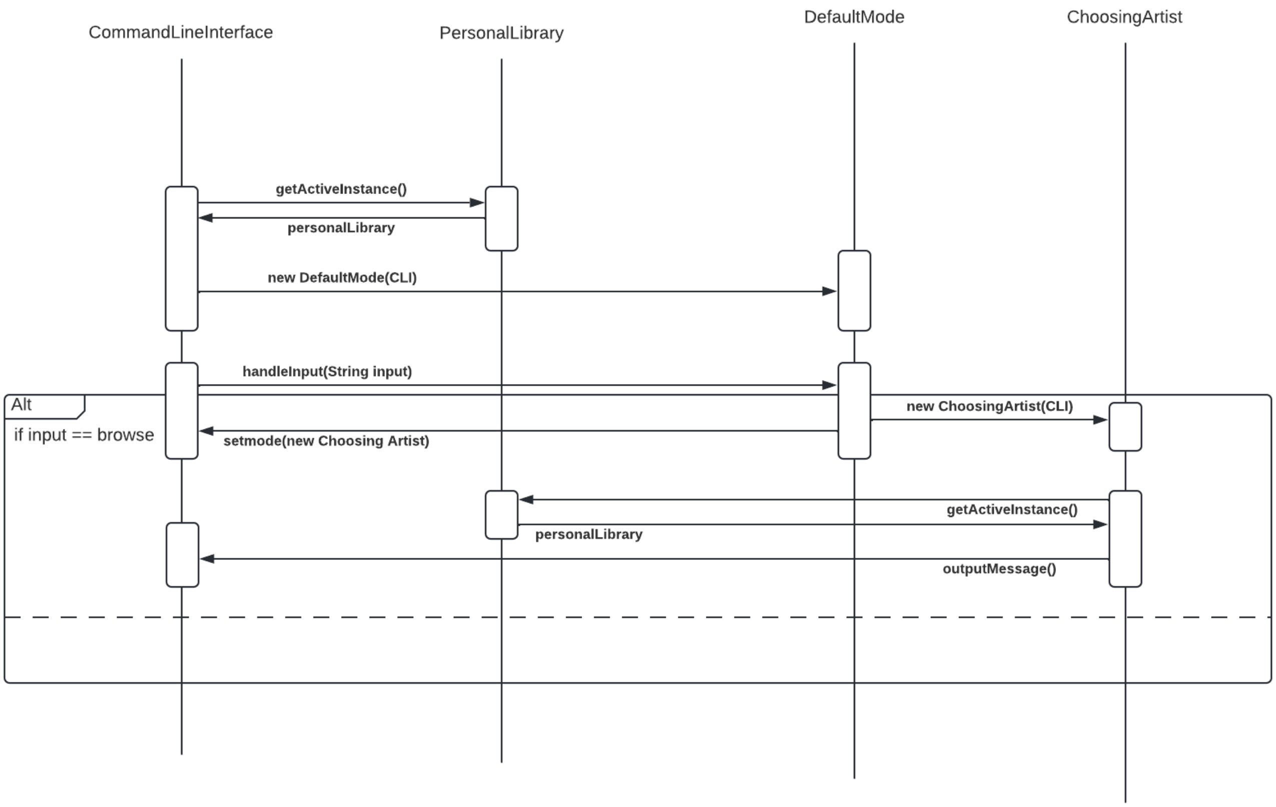

CommandLineInterface
<<Invoker>>
<<Context>>
- Action addByGUIDAction
- Action removeByGUIDAction
- Action rateByGUIDAction
- DBSongSearcher dbSearchStrategy
- LibrarySearcher libSearchStrategy
- Mode currentMode
- PersonalLibrary library
- boolean running
- handleInput(String input)
# outputMessage(String message)
# addByGUID(String GUID, LocalDate date)
# removeByGUID(String GUID)
# rateByGUID(String GUID, int rating)
# setDBSeartchStrategy(DBSongSearcher strategy)
# setLibSeartchStrategy(LibrarySearcher strategy)
# searchDatabse(String query)
# searchLibrary(String query)
# setMode(Mode newMode)
# quit()
+ run()
Mode
<<State>>
- CommandLineInterface commandLineInterface
# unknownCommand()
# getCommandLineInterface()
# *listCommands()*
# *handleInput(String input)*
BrowsingRelease
<<Concrete State>>
- CommandLineInterface commandLineInterface
- Release release
# listCommands()
# handleInput(String input)
BrowsingArtist
<<Concrete State>>
- CommandLineInterface commandLineInterface
- Artist artist
# listCommands()
# handleInput(String input)
ChoosingArtist
<<Concrete State>>
- CommandLineInterface commandLineInterface
# listCommands()
# handleInput(String input)
DefaultMode
<<Concrete State>>
- CommandLineInterface commandLineInterface
# listCommands()
# handleInput(String input)

## Sequence Diagram

## Feature - Adding items by their GUID



## Feature - Browsing the User's Library



## Design Pattern

| Name: View Subsystem | | GoF pattern: State |
|---|---|---|
| Participants | | |
| Class | Role in GoF pattern | Participant's contribution in the context of the application |
| CommandLineInterface | Context | Interacts with the user. Accepts and handles input. Delegates input handling the current state. |
| Mode | State | Defines behaviors that change based on the current state of the command line interface. |
| DefaultMode | Concrete State | A mode of the command line interface that represents the default mode. Allows the user to add, rate, and remove songs or releases. It also allows the user to browse their personal library and search their library and the database. Transitions to the choosing artist state when the user uses the "browse" command. |

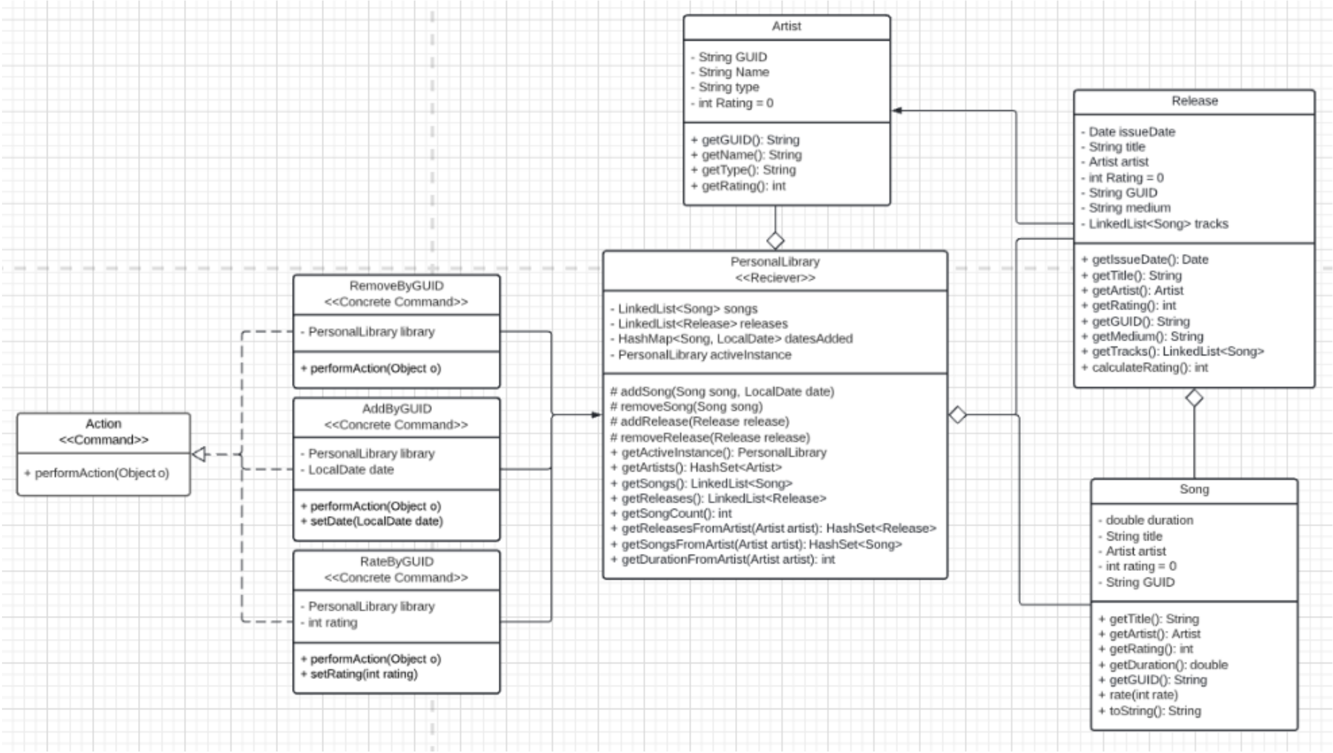
| | | |
|---|---|---|
| **ChoosingArtist** | **Concrete State** | **A mode of the command line interface that displays all of the artists in the user's library. Allows the user to select which artist to browse or to backout to the default mode. Transitions to the default mode if the user uses the "esc" command. Transitions to the browsing artists mode when the user selects a user to browse.** |
| **BrowsingArtist** | **Concrete State** | **A mode of the command line interface that displays all of the singles and releases from a given artist in the user's library. Allows the user to select which release to browse, or to backout to the choosing artist mode. Transitions to the choosing artist mode if the user uses the "esc" command. Transitions to the browsing release mode when the user selects a release to browse.** |
| **BrowsingRelease** | **Concrete State** | **A mode of the command line interface thaT displays all of the singles and releases from a given release. Allows the user to backout to the browsing artist mode. Transitions to the browsing artist mode when the user uses the "esc" command.** |
| **Deviations from the standard pattern: None** | | |
| **Requirements being covered: Browsing library, searching library, searching database, adding songs/releases, removing songs/releases, rating songs** | | |

## *PersonalLibrary*

The PersonalLibrary subsystem manages user storage within the system. When the system is running, the PersonalLibrary stores and manipulates the data. The subsystem contains Actions that allow external classes to query or manipulate the data in the library. Additionally, the PersonalLibrary manages persistence between sessions.

The PersonalLibrary subsystem uses the command design pattern to allow for easily extensible functionality and access from outside classes.

### Class Structure Diagram



## Sequence Diagram

## Design Pattern

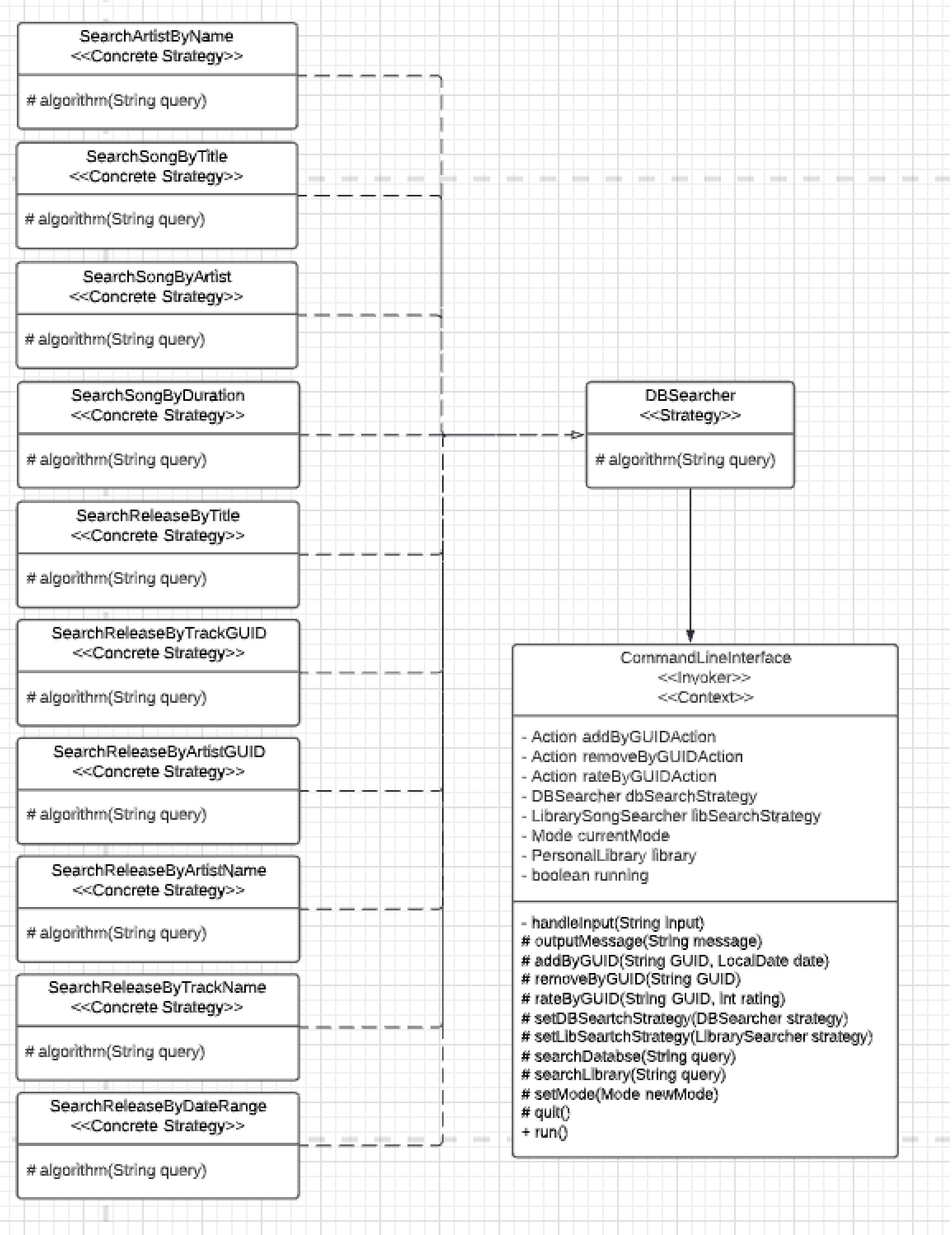
| Name: PersonalLibrary | | GoF Pattern: Command |
|---|---|---|
| **Participants** | | |
| **Class** | **Role in GoF pattern** | **Participant's contribution in the context of the application** |
| **PersonalLibrary** | **Receiver** | **Acts as the information expert. Provides basic functionality and information to classes in the same package. Allows them to implement further functionality that manipulates this class.** |
| **DefaultMode** | **Invoker** | **Invokes the different functions to interact with the Library.** |
| **Action** | **Command** | **Defines a basic behavior that interacts with the Library.** |
| **AddByGUID** | **Concrete Command** | **Adds a song or release to the library with a given GUID.** |
| **RateByGUID** | **Concrete Command** | **Rates a song or release in the library that has a given GUID.** |
| **RemoveByGUID** | **Concrete Command** | **Removes a song or release from the library with the given GUID.** |

| |
|---|
| **Deviations from the standard pattern: Some classes such as AddByGUID and RateByGUID maintain states that can be changed via setters. This allows them to take additional parameters.** |
| **Requirements being covered: Adding, removing and rating songs.** |

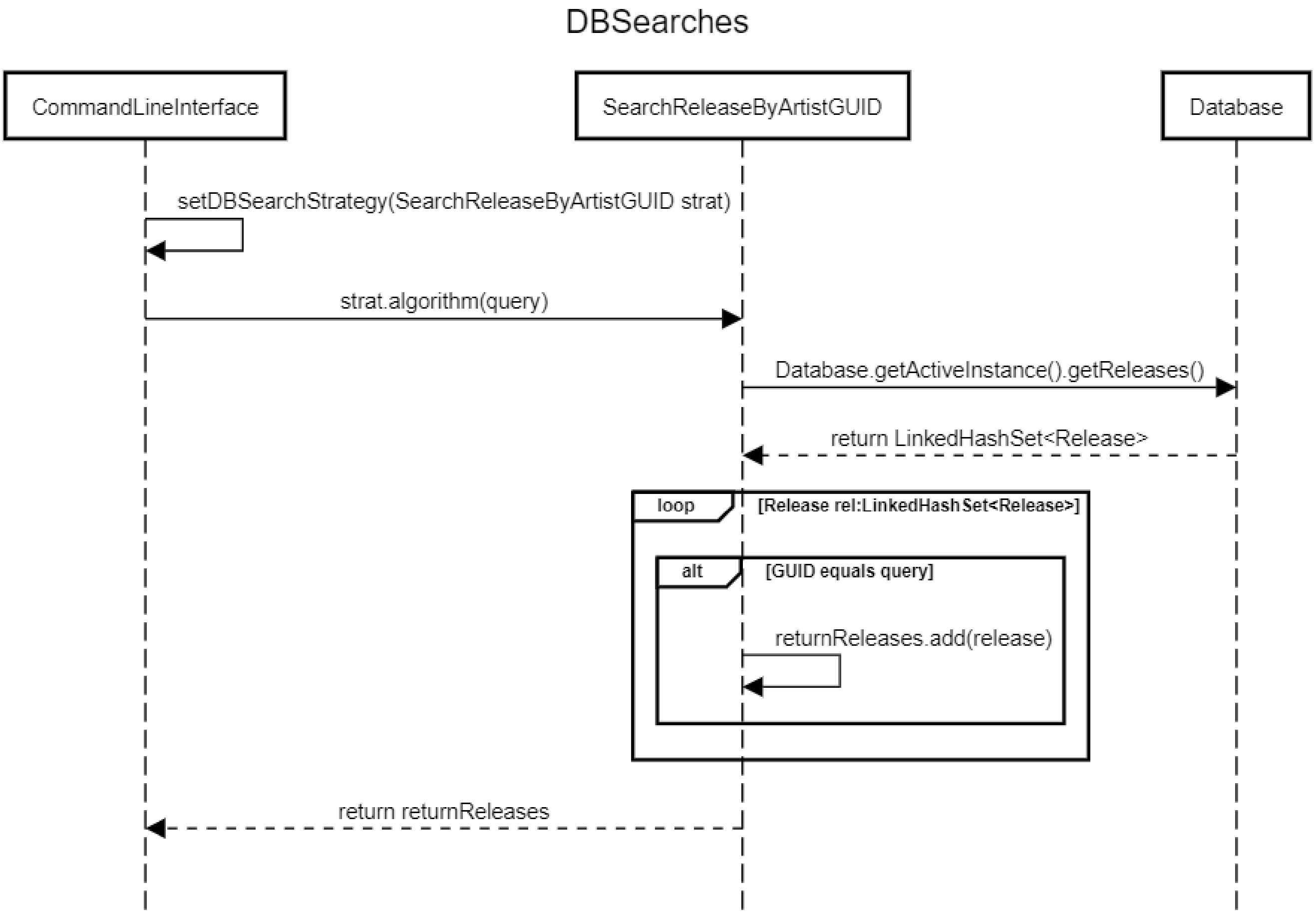
### *DBSearches*

DBSearches is a subsystem used to query the database to return Artists, Releases, or Song based on a given query String. Concrete Strategies implement the DBSearcher interface to allow for different algorithms to be run to query the database based on the different query string passed in from the command line.

**Class Structure Diagram**


SearchArtistByName
<<Concrete Strategy>>
# algorithm(String query)
SearchSongByTitle
<<Concrete Strategy>>
# algorithm(String query)
SearchSongByArtist
<<Concrete Strategy>>
# algorithm(String query)
SearchSongByDuration
<<Concrete Strategy>>
# algorithm(String query)
SearchReleaseByTitle
<<Concrete Strategy>>
# algorithm(String query)
SearchReleaseByTrackGUID
<<Concrete Strategy>>
# algorithm(String query)
SearchReleaseByArtistGUID
<<Concrete Strategy>>
# algorithm(String query)
SearchReleaseByArtistName
<<Concrete Strategy>>
# algorithm(String query)
SearchReleaseByTrackName
<<Concrete Strategy>>
# algorithm(String query)
SearchReleaseByDateRange
<<Concrete Strategy>>
# algorithm(String query)
DBSearcher
<<Strategy>>
# algorithm(String query)
CommandLineInterface
<<Invoker>>
<<Context>>
- Action addByGUIDAction
- Action removeByGUIDAction
- Action rateByGUIDAction
- DBSearcher dbSearchStrategy
- LibrarySongSearcher libSearchStrategy
- Mode currentMode
- PersonalLibrary library
- boolean running
- handleInput(String input)
# outputMessage(String message)
# addByGUID(String GUID, LocalDate date)
# removeByGUID(String GUID)
# rateByGUID(String GUID, int rating)
# setDBSeartchStrategy(DBSearcher strategy)
# setLibSeartchStrategy(LibrarySearcher strategy)
# searchDatabse(String query)
# searchLibrary(String query)
# setMode(Mode newMode)
# quit()
+ run()

## Sequence Diagram


DBSearches
CommandLineInterface
SearchReleaseByArtistGUID
Database
setDBSearchStrategy(SearchReleaseByArtistGUID strat)
strat.algorithm(query)
Database.getActiveInstance().getReleases()
return LinkedHashSet<Release>
loop
[Release rel:LinkedHashSet<Release>]
alt
[GUID equals query]
returnReleases.add(release)
return returnReleases

## Design Pattern

| Name: DBSearches Subsystem | | GoF pattern: Strategy |
|---|---|---|
| **Participants** | | |
| **Class** | **Role in GoF pattern** | **Participant's contribution in the context of the application** |
| **DBSearcher** | **Strategy** | **The interface for an algorithm capable of searching a Database. Concrete implementations should return a list of Artists, Releases, or Songs matching the given query in context.** |
| **CommandLineInterface** | **Client, Context** | **Exposes a method allowing a user to change how they want to query the database. The main user interface of the application.** |
| **SearchArtistByName** | **ConcreteStrategy** | **A DBSearcher that returns Artists that match a name query** |
| **SearchSongByTitle** | **ConcreteStrategy** | **A DBSearcher that returns Songs that match a title query** |
| **SearchSongByArtist** | **ConcreteStrategy** | **A DBSearcher that returns Songs that match an Artist query** |

| | | |
|---|---|---|
| **SearchSongByDuration** | **ConcreteStrategy** | **A DBSearcher that returns Songs that match a query for a duration short or longer than a given duration** |
| **SearchReleaseByTitle** | **ConcreteStrategy** | **A DBSearcher that returns Releases that match a title query** |
| **SearchReleaseByTrackGUID** | **ConcreteStrategy** | **A DBSearcher that returns Releases that contain a Song that matches a Song GUID query** |
| **SearchReleaseByArtistGUID** | **ConcreteStrategy** | **A DBSearcher that returns Releases that match a query for the Artist GUID that produced the Release** |
| **SearchReleaseByArtistName** | **ConcreteStrategy** | **A DBSearcher that returns Releases that match a query for the Artist name that produced the Release** |
| **SearchReleaseByTrackName** | **ConcreteStrategy** | **A DBSearcher that returns Releases that contain a Song with a title that matches the query** |
| **SearchReleasebyDateRange** | **ConcreteStrategy** | **A DBSearcher that returns Releases that were produced between a given date range** |
| **Deviations from the standard pattern: CommandLineInterface is both the client and context** | | |
| **Requirements being covered: The user may search for artists by specifying one or more words in the artist's name. The system shall respond with a list of artists that match the query. The user may search for songs by specifying one or more words in the title, the name of the artist, or duration (songs shorter than or longer than a given duration). The system shall respond with a list of songs that match the query. The user may search for releases by specifying one or more words in the title, the name or GUID of the artist, the name or GUID of one of the tracks, or a release date range (i.e. all releases between two specific dates). The system shall respond with a list of releases that match the query.** | | |

## Status of the Implementation

Currently all operations such as adding, removing and rating must be done by specifying a GUID. We would like to improve this by possibly allowing the user to simply enter a title or artist name. This would make it much simpler for users.

Our implementation is missing the ability to sort search results. Due to time limitations, we were not able to come up with a clean, elegant and user friendly solution for users to sort results. Our ideas were to add another solution, which would essentially be adding a solution to the solution. We felt that this would complicate the system and make it very convoluted.

## Appendix

| | |
|---|---|
| **Class:** `DBSearcher` | |
| **Responsibilities:** Be the interface implemented by concrete strategies used to search the database for Songs, Artists, and Releases | |
| **Collaborators:** ... | |
| **Users:** ... | **Used by:** SearchArtistByName, SearchReleaseByArtistGUID, SearchReleaseByArtistName, SearchReleaseByDateRange, SearchReleaseByTitle, SearchReleaseByTrackGUID, SearchReleaseByTrackName, SearchSongByArtist, SearchSongByDuration, SearchSongByTitle |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** `SearchArtistByName` | |
| **Responsibilities:** Search the DB for Artists containing a substring matching the query string | |
| **Collaborators:** ... | |
| **Users:** Artist, DBSearcher, Database | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** `SearchReleaseByArtistGUID` | |
| **Responsibilities:** Search the DB for Releases matching the query GUID | |
| **Collaborators:** ... | |
| **Users:** Release, DBSearcher, Database | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** `SearchReleaseByArtistName` | |
| **Responsibilities:** Search the DB for | |

| | |
|---|---|
| Releases created by Artists matching the query Artist name | |
| **Collaborators:** ... | |
| **Users:** Release, DBSearcher, Database | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchReleaseByDateRange | |
| **Responsibilities:** Search the DB for Releases that were issued between a given date range given in a query | |
| **Collaborators:** ... | |
| **Users:** Release, DBSearcher, Database | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchReleaseByTitle | |
| **Responsibilities:** Search the Db for Releases that have a matching substring with the given query | |
| **Collaborators:** ... | |
| **Users:** Release, DBSearcher, Database | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchReleaseByTrackGUID | |
| **Responsibilities:** Search the DB for Releases with a matching Song GUID with the given GUID in a query | |
| **Collaborators:** ... | |
| **Users:** Release, DBSearcher, Database | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchReleaseByTrackName | |
| **Responsibilities:** Search the DB for | |

| | |
|---|---|
| Releases with a matching Song name given in a query | |
| **Collaborators:** ... | |
| **Users:** Release, DBSearcher, Database | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** `SearchSongByDuration` | |
| **Responsibilities:** Search the DB for Songs that are longer or shorter than a duration given in the query | |
| **Collaborators:** ... | |
| **Users:** Song, DBSearcher, Database | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** `SearchSongByTitle` | |
| **Responsibilities:** Search the DB for songs containing a title with a substring of a given query | |
| **Collaborators:** ... | |
| **Users:** Song, DBSearcher, Database | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** `SearchSongByArtist` | |
| **Responsibilities:** Search the DB for Songs with a given Artist matching a query | |
| **Collaborators:** ... | |
| **Users:** Song, DBSearcher, Database | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** `LibrarySongSearcher` | |
| **Responsibilities:** The interface implemented | |

| | |
|---|---|
| by any concrete strategy used to search a PersonalLibrary for an Artist, Release, or Song | |
| **Collaborators:** ... | |
| **Users:** ... | **Used by:** SearchArtistByName, SearchArtistByRating, SearchArtistByType, SearchReleaseByArtistGUID, SearchReleasebyArtistName, SearchReleaseByMaxDuration, SearchReleaseByMinDuration, SearchReleaseByRating, SearchReleaseByTitle, SearchReleaseByTrackGUID, SearchReleaseByTrackName, SearchSongByArtistGUID, SearchSongByArtistName, SearchSongByMaxDuration, SearchSongByMinDuration, SearchSongByRating, SearchSongByReleaseGUID, SearchSongByReleaseTitle, SearchSongByTitle |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchArtistByName | |
| **Responsibilities:** Search the PersonalLibrary for Artists with a name matching a given query | |
| **Collaborators:** ... | |
| **Users:** Artist, LibrarySearcher, PersonalLibrary | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchArtistByRating | |

| | |
|---|---|
| **Responsibilities:** Search the PersonalLibrary for Artist’s with a rating equal or greater than the query rating given | |
| **Collaborators:** ... | |
| **Users:** Artist, LibrarySearcher, PersonalLibrary | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchArtistByType | |
| **Responsibilities:** Search the PersonalLibrary for Artists matching the given type in the query | |
| **Collaborators:** ... | |
| **Users:** Artist, LibrarySearcher, PersonalLibrary | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchReleaseByArtistGUID | |
| **Responsibilities:** Search the PersonalLibrary for Releases made by Artists with a matching GUID with the given query | |
| **Collaborators:** ... | |
| **Users:** Release, LibrarySearcher, PersonalLibrary | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchReleaseByArtistName | |
| **Responsibilities:** Search the | |

| | |
|---|---|
| PersonalLibrary for Releases made by Artists with a matching name with the given query | |
| **Collaborators:** ... | |
| **Users:** Release, LibrarySearcher, PersonalLibrary | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchReleaseByMaxDuration | |
| **Responsibilities:** Search the PersonalLibrary for Releases that are less than or equal to a given max duration provided in the query | |
| **Collaborators:** ... | |
| **Users:** Release, LibrarySearcher, PersonalLibrary | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchReleaseByMinDuration | |
| **Responsibilities:** Search the PersonalLibrary for Releases that are greater than or equal to a given minimum duration provided in the query | |
| **Collaborators:** ... | |
| **Users:** Release, LibrarySearcher, PersonalLibrary | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchReleaseByRating | |
| **Responsibilities:** Search the | |

| | |
|---|---|
| PersonalLibrary for Releases that are greater than or equal to a minimum rating provided in a query | |
| **Collaborators:** ... | |
| **Users:** Release, LibrarySearcher, PersonalLibrary | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchReleaseByTitle | |
| **Responsibilities:** Search the PersonalLibrary for Releases with a substring constraining the provided query | |
| **Collaborators:** ... | |
| **Users:** Release, LibrarySearcher, PersonalLibrary | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchReleaseByTrackGUID | |
| **Responsibilities:** Search the PersonalLibrary for Releases that contain a Song with a GUID matching a query | |
| **Collaborators:** ... | |
| **Users:** Release, LibrarySearcher, PersonalLibrary | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchReleaseByTrackName | |
| **Responsibilities:** Search the PersonalLibrary for Releases containing a Song with a name matching the given query. | |
| **Collaborators:** ... | |
| **Users:** Release, LibrarySearcher, DPersonalLibrary | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchSongByArtistGUID | |
| **Responsibilities:** Search the PersonalLibrary for Songs with an Artist GUID matching the given query | |
| **Collaborators:** ... | |
| **Users:** Song, LibrarySearcher, DPersonalLibrary | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchSongByArtistName | |
| **Responsibilities:** Search the PersonalLibrary for Songs with an Artist name matching the given query | |
| **Collaborators:** ... | |
| **Users:** Song, LibrarySearcher, DPersonalLibrary | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchSongByMaxDuration | |
| **Responsibilities:** Search the PersonalLibrary for Songs with a duration less than or equal to a given max duration in the query | |
| **Collaborators:** ... | |
| **Users:** Song, LibrarySearcher, DPersonalLibrary | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchSongByMinDuration | |

| | |
|---|---|
| **Responsibilities:** Search the PersonalLibrary for Songs with a duration greater than or equal to a given minimum duration in the query | |
| **Collaborators:** ... | |
| **Users:** Song, LibrarySearcher, DPersonalLibrary | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchSongByRating | |
| **Responsibilities:** Search the PersonalLibrary for Songs that have a rating greater than or equal to the provided minimum rating in the query | |
| **Collaborators:** ... | |
| **Users:** Song, LibrarySearcher, DPersonalLibrary | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchSongByReleaseGUID | |
| **Responsibilities:** Search the PersonalLibrary for Songs with a Release GUID matching the given query | |
| **Collaborators:** ... | |
| **Users:** Song, LibrarySearcher, DPersonalLibrary | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchSongByArtistTitle | |

| | |
|---|---|
| **Responsibilities:** Search the PersonalLibrary for Songs with a Release title that matches the given query | |
| **Collaborators:** ... | |
| **Users:** Song, LibrarySearcher, DPersonalLibrary | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** SearchSongByTitle | |
| **Responsibilities:** Search the PersonalLibrary for Songs with a title substring that contains the given query | |
| **Collaborators:** ... | |
| **Users:** Song, LibrarySearcher, DPersonalLibrary | **Used by:** DefaultMode |
| **Author:** John Flory | |

| | |
|---|---|
| **Class:** `Action` | |
| **Responsibilities:** This class is the interface that the concrete commands implement. It defines the performAction() method. | |
| **Collaborators:** ... | |
| **Users:** | **Used by:** RateByGUID, AddByGUID, RemoveByGUID, CommandLineInterface |
| **Author:** Alison Hansen | |

| | |
|---|---|
| **Class:** `RateByGUID` | |
| **Responsibilities:** This class uses the active instance of the PersonalLibrary. It is responsible for searching the Database for a Song. It then calls the rate() method on the Song. | |
| **Collaborators:** ... | |
| **Users:** PersonalLibrary, Action, Database, Song | **Used by:** CommandLineInterface |
| **Author:** Alison Hansen | |

| **Class:** `AddByGUID` | |
|---|---|
| **Responsibilities:** This class uses the active instance of the PersonalLibrary. It is responsible for calling the addSong() and addRelease() methods on the PersonalLibrary.  These methods search the database for the song or release and add them if possible. | |
| **Collaborators:** ... | |
| **Users:** PersonalLibrary, Action | **Used by:** CommandLineInterface |
| **Author:** Alison Hansen | |

| **Class:** `RemoveByGUID` | |
|---|---|
| **Responsibilities:** This class uses the active instance of the PersonalLibrary. It is responsible for calling the removeSong() and removeRelease() methods on the PersonalLibrary.  These methods search the library for the song or release and remove them if possible. | |
| **Collaborators:** ... | |
| **Users:** PersonalLibrary, Action | **Used by:** CommandLineInterface |
| **Author:** Alison Hansen | |

| **Class:** `PersonalLibrary` | |
|---|---|
| **Responsibilities:** Represents a user's library.  Stores a list of the user's saved songs and releases.  Defines functionality for adding or removing songs or releases to the library. | |
| **Collaborators:** ... | |
| **Users:** Song, Release, Artist | **Used by:** RemoveByGUID, AddByGUID, RateByGUID, SearchArtistByName, SearchArtistByRating, SearchArtistByType, SearchReleaseByArtistGUID, SearchReleaseByArtistName, SearchArtistByMaxDuration, SearchArtistByMinDuration, |

| | |
|---|---|
| | SearchSongByMinDuration,<br>SearchSongByReleaseGUID,<br>SearchSongByTitle,<br>SearchReleaseByRating,<br>SearchReleaseByTitle,<br>SearchReleaseByTrackGUID,<br>SearchReleaseByTrackName,<br>SearchSongByArtistGUID,<br>SearchSongByArtistName,<br>SearchSongByMaxDuration,<br>SearchSongByRating,<br>SearchSongByReleaseTitle |
| **Author:** Alison Hansen | |

| | |
|---|---|
| **Class:** `Song` | |
| **Responsibilities:** Represent the object of a Song. Stores GUID, title, Artist, rating, and duration to easily search for a Song. Assigned default rating of 0. Allows users to rate Song from 1-5. | |
| **Collaborators:** ... | |
| **Users:** Artist | **Used by:** Database, PersonalLibrary, Release |
| **Author:** Alison Hansen | |

| | |
|---|---|
| **Class:** Artist | |
| **Responsibilities:** Represent the object of an Artist. Stores GUID, name, type, and rating so that an Artist can be easily identified and searched for. | |
| **Collaborators:** ... | |
| **Users:** ... | **Used by:** Database, PersonalLibrary, Release, Song |
| **Author:** Alison Hansen | |

| | |
|---|---|
| **Class:** `Release` | |
| **Responsibilities:** Represent the object of a Release. Calculates the average rating of the individual songs stored within the Release. | |

| | |
|---|---|
| Also assigns the current date as the issue date unless otherwise specified by the user. | |
| **Collaborators:** ... | |
| **Users:** Song, Artist | **Used by:** Database, PersonalLibrary |
| **Author:** Alison Hansen | |

| | |
|---|---|
| **Class:** CommandLineInterface | |
| **Responsibilities:** Accepts and handles user input. Carries out state-based behavior based on the user's input. Responds with the results of any actions of the user. | |
| **Collaborators:** ... | |
| **Users:** AddByGUID, RemoveByGUID, RateByGUID, DBSearcher, LibrarySearcher, Mode, PersonalLibrary | **Used by:** Mode |
| **Author:** Devan Kavalchek | |

| | |
|---|---|
| **Class:** Mode | |
| **Responsibilities:** Defines behaviors that change based on the current state of the command line interface. | |
| **Collaborators:** ... | |
| **Users:** CommandLineInterface | **Used by:** CommandLineInterface |
| **Author:** Devan Kavalchek | |

| | |
|---|---|
| **Class:** DefaultMode | |
| **Responsibilities:** The default mode for the command line interface. Allows the user to add, remove, and rate songs or releases. Allows them to search the database or their personal library. Additionally, it allows users to browse their personal library. | |
| **Collaborators:** ... | |
| **Users:** CommandLineInterface, DBSearcher, LibrarySearcher | **Used by:** CommandLineInterface |
| **Author:** Devan Kavalchek | |

| **Class:** ChoosingArtist | |
|---|---|
| **Responsibilities:** A mode in the command line interface that displays all of the artists in the user's library. Allows them to browse the song and releases of that artist, or backout to the default mode. | |
| **Collaborators:** ... | |
| **Users:** CommandLineInterface, PersonalLibrary | **Used by:** CommandLineInterface |
| **Author:** Devan Kavalchek | |

| **Class:** BrowsingArtist | |
|---|---|
| **Responsibilities:** A mode in the command line interface that displays all of the singles and releases of an artist in the user's library. Allows them to browse the releases or backout the choosing artist mode. | |
| **Collaborators:** ... | |
| **Users:** CommandLineInterface, PersonalLibrary | **Used by:** CommandLineInterface |
| **Author:** Devan Kavalchek | |

| **Class:** BrowsingRelease | |
|---|---|
| **Responsibilities:** A mode in the command line interface that displays all of the tracks of a given release. Allows the users to view the tracks and backout to the browsing artist mode. | |
| **Collaborators:** ... | |
| **Users:** CommandLineInterface, PersonalLibrary | **Used by:** CommandLineInterface |
| **Author:** Devan Kavalchek | |

| **Class:** `Database` | |
|---|---|

| | |
|---|---|
| **Responsibilities:** This is the primary datasource for the program, and the source that the searching algorithms use to find songs, artists, and releases | |
| **Collaborators:** ... | |
| **Users:** Song, Release, Artist | **Used by:** SearchArtistByName, SearchSongByTitle, SearchReleaseByArtistGUID, SearchReleaseByTrackGUID, SearchSongByDuration, SearchReleaseByDateRange, SearchSongByArtist, SearchReleaseByTrackName, SearchReleaseByArtistName, SearchReleaseByTitle, PersonalLibrary, RateByGUID, AddByGUID, DefaultMode, CommandLineInterface, Main |
| **Author:** Preston Sergent | |

| | |
|---|---|
| **Class:** `SaveLibrary` | |
| **Responsibilities:** This class allows the library to be persisted, if chosen by the user, across instantiations of the program. | |
| **Collaborators:** ... | |
| **Users:** Song, Release, Artist | **Used by:** CommandLineInterface |
| **Author:** Preston Sergent | |