CS-151 Object Oriented Design Final Project

Chance Kissel, Jake Johnson, Zhirong Chen

CS Department, San Jose State University

CS-151 Section 05: Object Oriented Design

Professor. Zare

May 10th, 2024

1. Introduction

This is the project report for Group 8 for course CS-151: Object Oriented Design. Group 8 consists of members: Chance Kissel, Jake Johnson, and Zhirong Chen. We seek to successfully create and implement an Object Oriented Design based Java project which utilizes various principles and concepts that have been discussed over the course of the semester.

2. Objective

Our objective is to create a console based application that deals with music catalogs and playlists. Core functionalities of our application are prompting users with the ability to sign up or login to the application, providing a catalog of music to view, and allowing users to create, view, and save their own playlist of music. Both account and playlist information will be stored through an implemented file system which saves users data after closing the application.

3. High Level Design

High level design concepts per class:

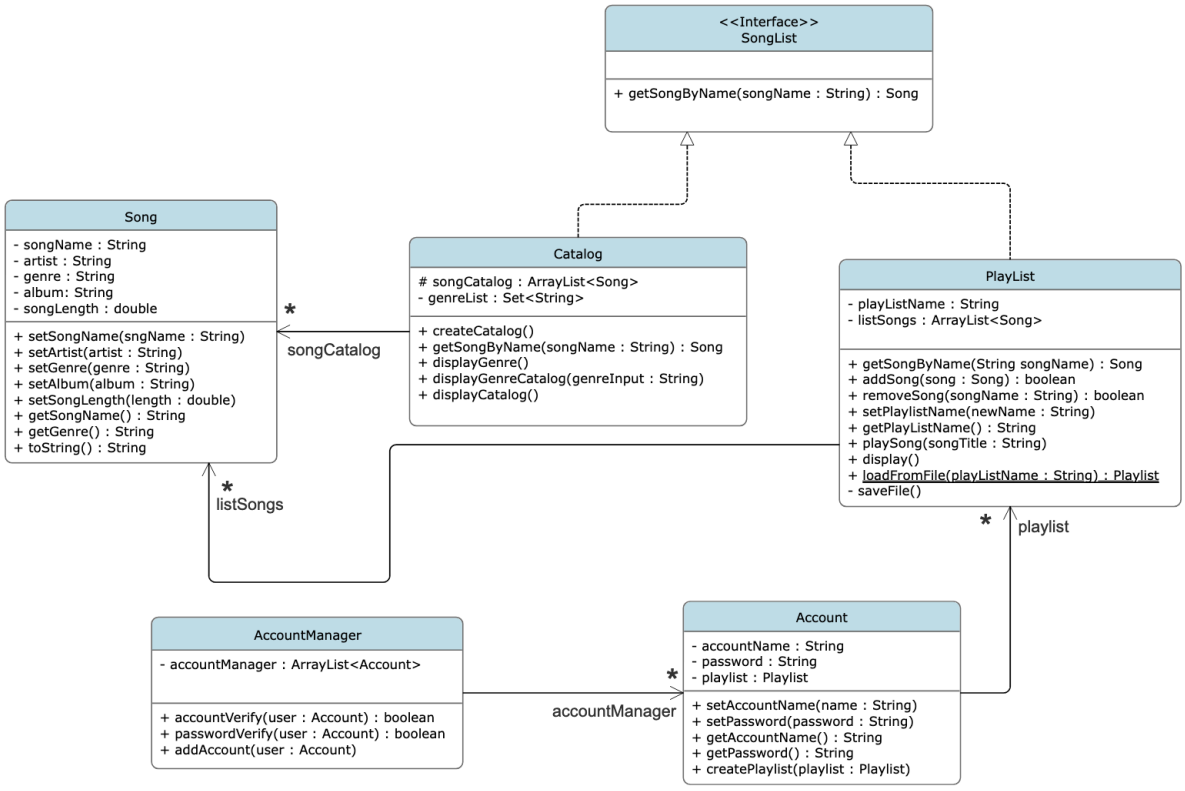
- Account Class:
    - Encapsulation - Data members inside of this class are encapsulated through Access Control to be “private” which means they can only be accessed by the provided public methods within the class.
    - Overloading - This class uses overloading for its constructors to allow for object creation with or without predetermined information.
- Account Manager Class:
    - Encapsulation - Data member, “accountManager” is encapsulated through Access Control to be “private” so it is only accessible within its own class’s provided methods.
    - Java File Handling - This class utilizes Java I/O functionality to read from a file to verify account information or write to a file when a new account is created.
    - Java Exception Handling - This class implements try/catch statements to account for IOExceptions when handling files.
- Song Class:
    - Encapsulation - Data members of this class are encapsulated through Access Control making its members “private” allowing them to only be accessed within the classes getter and setter methods.
    - Overloading - This class implements overloaded constructors to allow for object flexibility when objects are created.
    - Overriding - This class overrides the Java toString method, providing a specialized display of a Song object's information.
- Catalog Class:

    - Encapsulation - Data members within this class are encapsulated through Access Control with the member “songCatalog” being “protected” allowing for access of the catalog within the package and the member “genreList” being “private” so it can only be altered by the members of its own class.
    - Java File Handling - This class uses Java I/O functionality to read from a file containing the list of songs to be placed inside the catalog amid program execution.
    - Java Exception Handling - This class uses try/catch statements to account for IOExceptions when reading through the file.
    - Inheritance - This class implements the interface “SongList” to ensure it contains the “getSongByName” method.
- Playlist Class:
    - Encapsulation - Data members within this class are encapsulated through Access Control with the members “listSongs” and “playListName” being set to private so they are only able to be altered by methods within Playlist.
    - Overloading - This class uses overloaded constructors to allow for the creation of a playlist object with various parameters.
    - Java File Handling - This class implements the Java I/O functionality to read and write to files in order to read playlists and alter playlist files when the user is utilizing the playlist functionalities of the application.
    - Java Exception Handling - This class deals with exceptions by using try catch statements to deal with IOExceptions.
    - Inheritance - This class implements the interface “SongList” to ensure it contains the “getSongByName” method.
- SongList Class:
    - Interface/Inheritance - This class serves as an interface that is implemented by both the Playlist and Catalog classes. It ensures that both of these classes implement a “getSongByName” method and has no implementation within itself.
    - Polymorphism: We have an interface with a function used by multiple classes. Specifically the “Catalog” and “Playlist” classes make use of it.
- Main Class:
    - Static Method - This class utilizes the “static” keyword to implement methods within the Main class that are called multiple times within the code. This helps increase code readability and simplicity while also bypassing the need to create a Main object.
    - Error Handling - Accounts for if the user does not enter correct input when options are provided. Will prompt the user if an error has occurred and ask to re-input a response.
    - Abstraction: The main class serves as an entry point for the program, abstracting object initialization and interactions in the main method.

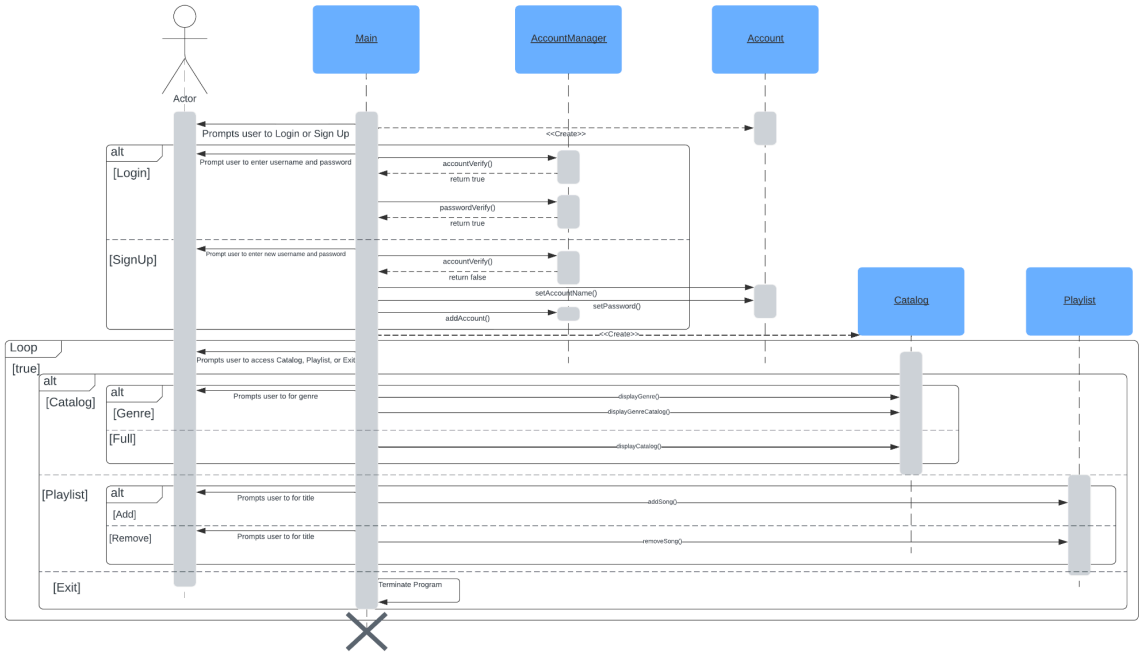
Tech Stack:

This project used VSCode as its main IDE. The code was all programmed in Java and uploaded to GitHub for publication.

4. Class Diagram



5. Sequence Diagram



Original diagram photos have been included in GitHub repository and Canvas submission for easier understanding and viewing.

6. Design Patterns

Initially, we did not plan for the use of any design patterns as we were exposed to them for the first time during the semester and did not think we could successfully implement one. However, now that our code is finished, it is clear that with certain modifications it is possible to implement some of these design patterns.

The Singleton Pattern is the one that stood out to us the most as one that fits our code. Our application only uses one instance of the AccountManager and Catalog fields, an aspect that could satisfy the requirements for a Singleton Design Pattern.

Another pattern that we thought our program could possibly be altered to fit is the Composite Pattern. The code could be restructured to resemble a tree format where AccountManager can manage multiple accounts, each account can have multiple playlists and each playlist can hold multiple songs.

We also felt that the Factory method is a close fit to what we are trying to achieve. This is due to how our program provides an interface that allows creating and storing objects of classes. While these said objects are not interchangeable in their types, they are created and stored in the same rapid fashion that the Factory method promotes, for example Account objects get stored in a file so a user can log in next time they run the program.

7. Results

Our program is a console based music catalog application therefore there are no concrete results to be returned.

8. Demonstrations

Login:

```
Hello! Would You Like To Login or SignUp? : Login/SignUp
Login
Username: kissel13
Password: music1
Welcome Back
```

SignUp:

```
Hello! Would You Like To Login or SignUp? : Login/SignUp
SignUp
Please Enter A Name and Password:
Username: gojiStan3

Password must but at least 6 characters long.
Password: 54321

Error: Please Enter A New Password.
Password must but at least 6 characters long.
Password: 76543a

Thank You For Choosing MusicCatalog
```

Catalog display:

```
Please Enter What Would You Like To Access: Catalog/Playlist/Exit
Catalog

Would You Like To Access A Catalog Genre or Full? : Genre/Full
Genre

What Genre Would You Like To View?
Pop
Rap
Alternative
Funk
R&B

Genre:
Funk

Stayin' Alive, Bee Gees, Disco, Funk, 4.1

Please Enter What Would You Like To Access: Catalog/Playlist/Exit
```

Playlist functionality:

```
Please Enter What Would You Like To Access: Catalog/Playlist/Exit
playlist

Would You Like To Create, View, or Return? : Create/View/Return
create

Plese Enter Name of Playlist:
newPlaylist

Actions : Add/Remove/Return
add

Enter Song Title: Staying Alive
This Song Is Not In The Catalog


Actions : Add/Remove/Return
add

Enter Song Title: Stayin' Alive
Successfully Added To Playlist


Actions : Add/Remove/Return
return
```

```
Please Enter What Would You Like To Access: Catalog/Playlist/Exit
playlist

Would You Like To Create, View, or Return? : Create/View/Return
view

Please Enter Name of Playlist:
newPlaylist

Stayin' Alive, Bee Gees, Disco, Funk, 4.1
```

9. Contributions

- Jake Johnson
    - Classes: Playlist, Catalog (partial), Song (partial)
    - Report: Demonstrations, High Level Design, Instruction Manual for program, Design Patterns
    - Presentation: Primary functions, Example functionality slides
- Zhirong Chen
    - Classes: Playlist (partial), Catalog (partial), Main
    - Report: Sequence Diagram, High Level Design
    - Presentation: OOP principles and implementation, Example functionality slides
- Chance Kissel
    - Classes: Account, AccountManager, Song, SongList, Catalog (partial), Playlist (partial), Main
    - Report: UML Class Diagram, Sequence Diagram, Introduction, Objective, High Level Design
    - Presentation: Intro slides, Example functionality slides, Code Demo

10. References

No references were used for this project.

GitHub Link

- https://github.com/Kissel13/CS151Project