**Object Oriented Programming 2 – Java**

Name: Amir Hosein Khanmohammadi

Student ID: 991646689

Instructor: Hosein Marzi

Course: PROG24178

Project Name: Music Catalog System (Beat Browser)

Date: April 05, 2023

Table of Contents

[Introduction 3](#_Toc131717194)

[System Analysis 4](#_Toc131717195)

[Functional Requirements: 4](#_Toc131717196)

[Non-functional Requirements: 4](#_Toc131717197)

[Use Cases: 4](#_Toc131717198)

[User Interfaces: 4](#_Toc131717199)

[System Implementation 5](#_Toc131717200)

[Features and functionalities of the application 6](#_Toc131717201)

[Benefits of using the Music Catalog System 7](#_Toc131717202)

[How to get started with the Music Catalog System 7](#_Toc131717203)

[UML 8](#_Toc131717204)

[Song.java UML 8](#_Toc131717205)

[SongList.java UML 9](#_Toc131717206)

[MainUI.java UML 10](#_Toc131717207)

[Relationship of UMLs 11](#_Toc131717208)

[Screenshots of Application 12](#_Toc131717209)

# Introduction

The purpose of this project is to develop a Music Catalog System that is comprehensive, reliable, and user-friendly. The system stores and tracks detailed information of songs across Ten categories (Pop, Europop, Disco, Rock, Jazz, Hip-Hop, Rap, Electronic, Country, and K-Pop), song information including title, artist, album, release date, genre, length, writer(s), producer(s), and awards and nominations. This comprehensive database ensures that users can keep their music collections organized and up-to-date.

The objective of the Music Catalog System is to provide a tool for music enthusiasts to efficiently manage their music collections. The system aims to simplify the process of adding new songs, editing existing ones, and searching for specific songs. With the Music Catalog System, users can easily browse, search, edit, and delete songs from their collection, as well as add new songs to it, Users can also add their own custom genres to the predefined list of categories

The problem statement that the Music Catalog System addresses is the need for a reliable and efficient way to manage music collections. Many music enthusiasts struggle with keeping their music collections organized and up-to-date, leading to frustration and wasted time searching for specific songs. The Music Catalog System provides a solution to this problem by offering a comprehensive and user-friendly application to manage music collections.

The scope of this project includes developing a Music Catalog System that stores and tracks detailed information of songs across Ten categories, including a user-friendly interface for easy navigation and use. The system allows users to browse, search, edit, and delete songs from their collection, as well as add new songs to it. The project also includes testing the system to ensure its reliability and functionality.

Overall, the Music Catalog System is an essential tool for music enthusiasts who want to manage their music collections efficiently and accurately. It is designed to simplify the process of managing music collections and provide a user-friendly interface for easy access and use.

# System Analysis

The Music Catalog System has both functional and non-functional requirements that need to be considered for its successful implementation.

## Functional Requirements:

* The system should be able to store and track detailed information of songs across ten categories (Pop, Europop, Disco, Rock, Jazz, Hip-Hop, Rap, Electronic, Country, and K-Pop).
* The system should allow users to browse, search, edit, and delete songs from their collection.
* The system should allow users to add new songs to their collection and customize genres.
* The system should provide a search function that enables users to find specific songs by title, artist, album, release date, genre, length, writer(s), producer(s), and awards and nominations.
* The system should provide a user-friendly interface that displays the list of songs by category.

## Non-functional Requirements:

* The system should be reliable, scalable, and maintainable.
* The system should be fast and responsive.
* The system should be secure and protect user data.
* The system should be compatible with different operating systems and devices.
* The system should have a low learning curve and be easy to use.

## Use Cases:

* User adds a new song to the collection
* User edits an existing song in the collection
* User searches for a specific song in the collection
* User deletes a song from the collection
* User creates a custom genre

## User Interfaces:

The Music Catalog System has a graphical user interface (GUI) that is designed to be user-friendly and easy to use. The interface includes a menu bar with options to add, edit, search, and delete songs. There is also a category list that displays the songs by category, and a search bar that enables users to search for specific songs by various parameters.

# System Implementation

The Music Catalog System is implemented using the Java programming language, which is a popular choice for developing desktop applications. Java is known for its platform independence, reliability, and security features, making it an ideal choice for this type of system.

Tools:

* Integrated Development Environment (IDE): Visual Studio Code and IntelliJ IDEA
* User Interface Framework: JavaFX

# Features and functionalities of the application

The Music Catalog System is a comprehensive and user-friendly application designed to help music enthusiasts manage their music collections with ease. Here are some of the key features and functionalities of the application:

1. Browse and search by category: Users can browse their music collections by category, including Pop, Europop, Disco, Rock, Jazz, Hip-Hop, Rap, Electronic, Country, and K-Pop. This feature makes it easy for users to locate specific songs and organize their music collections efficiently.
2. Search by song details: Users can search for songs by title, artist, genre. This feature further enhances the ability to find and manage music efficiently.
3. Edit song details: Users can easily edit song details, such as title, artist, album, release date, genre, length, writer(s), producer(s), and awards and nominations, directly within the Music Catalog System. This makes it easy to keep track of important information about each song and update it as needed.
4. Add new songs: Users can add new songs to their collections with minimal effort. The Music Catalog System stores and tracks detailed information of each song, including title, artist, album, release date, genre, length, writer(s), producer(s), and awards and nominations.
5. Custom genres: Users can add their own custom genres to the predefined list of categories, allowing for even greater flexibility in organizing music collections.
6. Delete songs: The system allows users to delete songs from their collections. A prompt for confirmation before deletion prevents accidental deletion.
7. User-friendly interface: The Music Catalog System provides a user-friendly interface that displays the list of songs by category. The interface is designed to be easy to navigate and use, with buttons and search bars provided for browsing, searching, editing, and deleting songs.
8. Comprehensive database: By default, the system stores and tracks detailed information of 40 songs across ten categories, ensuring that users can keep their music collections organized and up-to-date.

Overall, the Music Catalog System is an essential tool for music enthusiasts who want to manage their music collections efficiently and accurately. The system simplifies the process of managing music collections and provides a user-friendly interface for easy access and use.

# Benefits of using the Music Catalog System

There are several benefits to using the Music Catalog System to manage music collection:

1. Organization: The Music Catalog System makes it easy to organize your music collection, allowing you to sort songs by category and genre. This makes it easier to find specific tracks and create personalized playlists.
2. Customization: With the ability to add custom genres to the predefined list, users can create a highly customized music collection that caters to their unique tastes.
3. Efficiency: The Music Catalog System enables users to manage their music collections with ease, reducing the time and effort required to locate, edit, and add new songs.
4. Accuracy: With the ability to track detailed information about each song, users can ensure that their music collection is accurate and up-to-date.
5. Accessibility: The Music Catalog System allows users to access their music collection from any device, providing a convenient way to enjoy their music no matter where they are.
6. Professionalism: With its thoughtfully designed interface and powerful features, the Music Catalog System provides a professional and polished way to manage music collections.

Overall, the Music Catalog System simplifies the process of managing music collections, allowing users to organize and customize their collections with ease while ensuring accuracy and accessibility. Whether you're a casual music lover or a serious collector, the Music Catalog System is an essential tool for managing your music collection efficiently and effectively.

# How to get started with the Music Catalog System

1. Download the Music Catalog System zip file to your computer.
2. Extract the contents of the zip file to a location of your choice.
3. Navigate to the folder where the Music Catalog System files are located.
4. Open an IDE (like Visual Studio Code).
5. Locate the MainUI.java file in the Music Catalog System folder.
6. Open the MainUI.java file in your IDE.
7. Run the MainUI.java file in your IDE to launch the Music Catalog System application.

# UML

## Song.java UML

|  |
| --- |
| Song |
| -songTitle: String  -songArtist: String  -songAlbum: String  -songReleaseDate: Date  -songGenre: String  -songLength: String  -songWriters: String  -songProducers: String  -songAwards: String  -id: int  -songID: int |
| +Song()  +Song(title:String, artist:String, album:String,  releaseDate:Date, genre:String, length:String,  writers:String, producers:String, awards:String)  +getSongTitle(): String  +setSongTitle(title:String): void  +getSongArtist(): String  +setSongArtist(artist:String): void  +getSongAlbum(): String  +setSongAlbum(album:String): void  +getSongReleaseDate(): Date  +setSongReleaseDate(date:Date): void  +getSongGenre(): String  +setSongGenre(genre:String): void  +getSongLength(): String  +setSongLength(length:String): void  +getSongWriters(): String  +setSongWriters(writers:String): void  +getSongProducers(): String  +setSongProducers(producers:String):void  +getSongAwards(): String  +setSongAwards(awards:String): void  +getSongID(): int  +compareTo(song:Song): int  +toString(): String |

## SongList.java UML

|  |
| --- |
| SongList |
| -songList: ArrayList<Song> |
| +SongList()  +SongList(file:File)  +SongList(file:String)  +get(index:int): Song  +addSong(song:Song): void  +insert(song:Song, index:int): void  +deleteSong(song:Song): void  +set(song:Song, index:int): void  +indexOf(song:Song): int  +size(): int  +clean(): void  +loadFromFile(file:File): void  +loadFromFile(fileName:String): void  +writeToFile(file:File): void  +writeToFile(fileName:String): void  +findSongsByTitle(title:String): SongList  +findSongsByArtist(artist:String): SongList  +findSongsByGenre(genre:String): SongList  +sortSongs(): void |

## MainUI.java UML

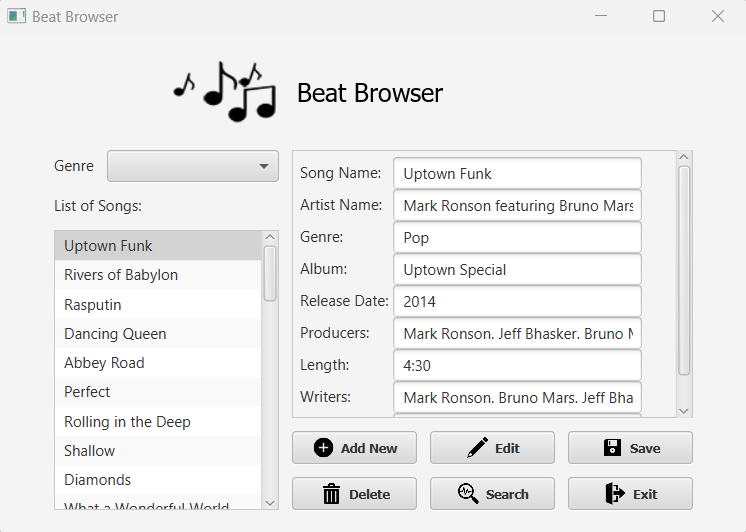
|  |
| --- |
| MainUI |
| -songList : SongList  -songs : ObservableList<String>  -songNames : ListView<String>  -genreList : ComboBox<String>  -songName : TextField  -artistName : TextField  -albumName : TextField  -releaseDate : TextField  -genreName : TextField  -writers : TextField  -producers : TextField  -awards : TextField  -songLength : TextField  -grid : GridPane  -delete : Button  -addNew : Button  -edit : Button  -exit : Button  -save : Button  -search : Button |
| +start(primaryStage: Stage) : void  +main(args: String[]) : void  +displaySongList(initial: boolean) : void  +displaySongDetails() : void  +displayGenres() : void  +addSong() : void  +editSong() : void  +deleteSong() : void  +searchSong() : void |

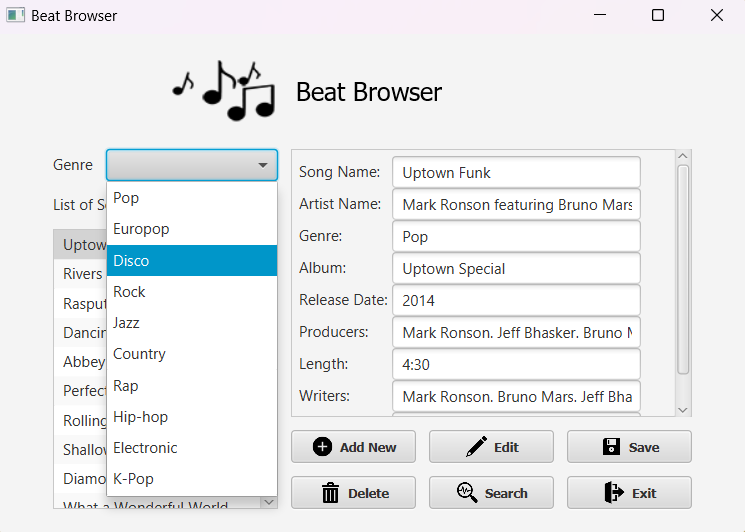
## Relationship of UMLs

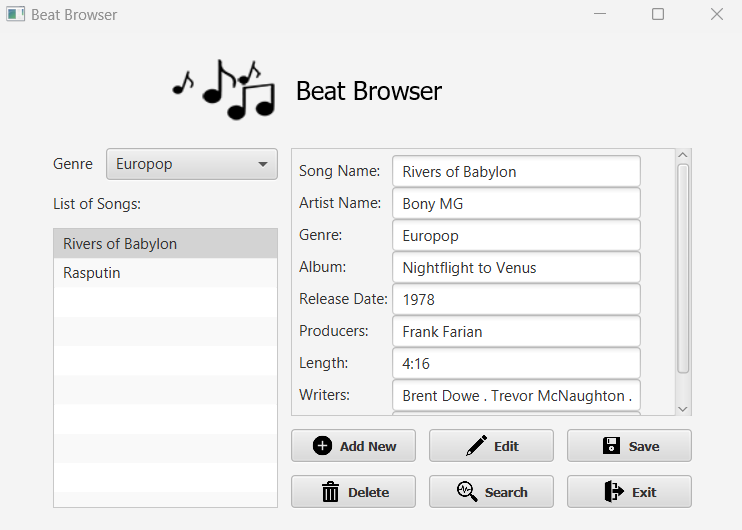
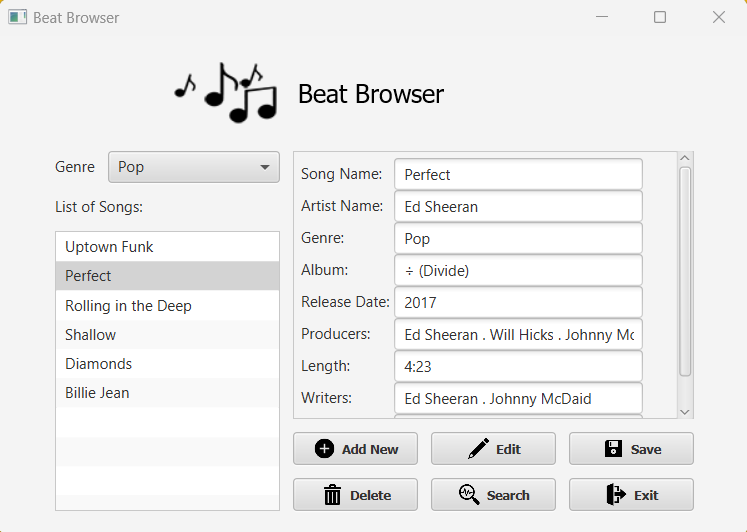
Diagram

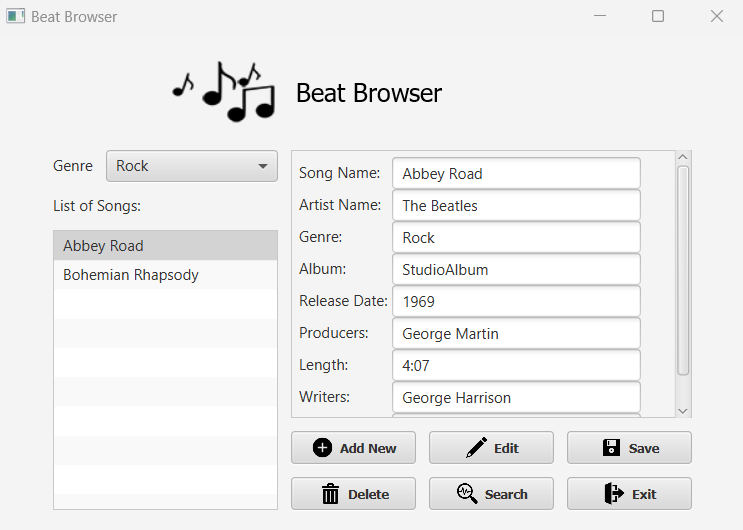
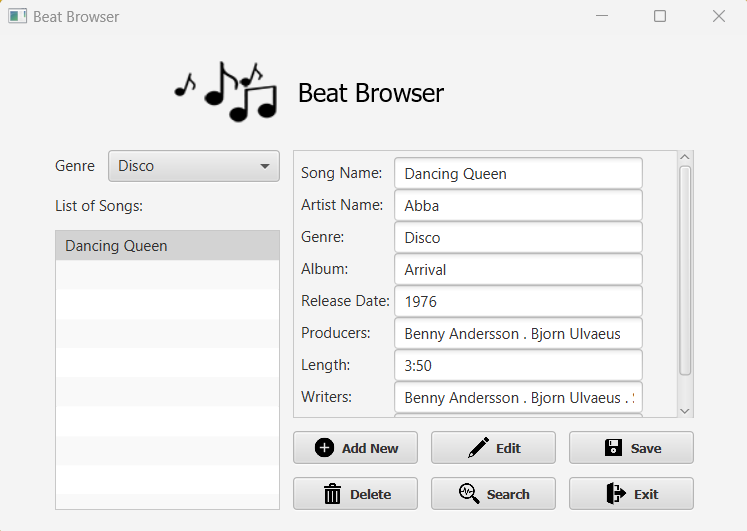
Description automatically generated

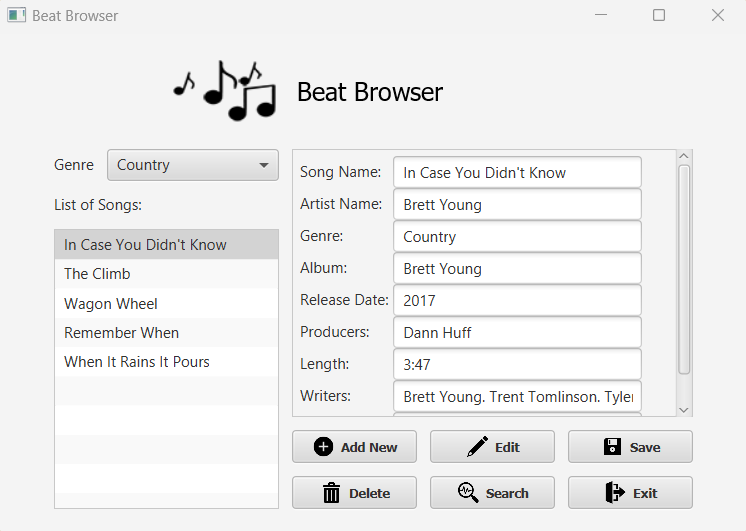
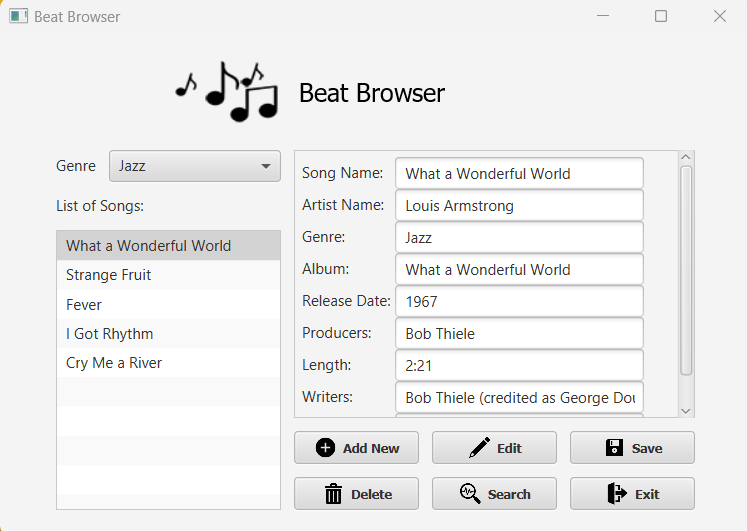
# Screenshots of Application



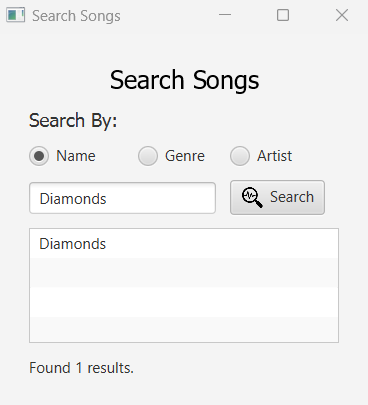








Graphical user interface, text, application, chat or text message

Description automatically generatedGraphical user interface, application

Description automatically generated