Exploring Spotify's Today's Top Hits Playlist

I. Overview

Custom-made database from Spotify's Today's Top Hits playlist contains total four tables artist, album, song and language. Each table has following attributes.

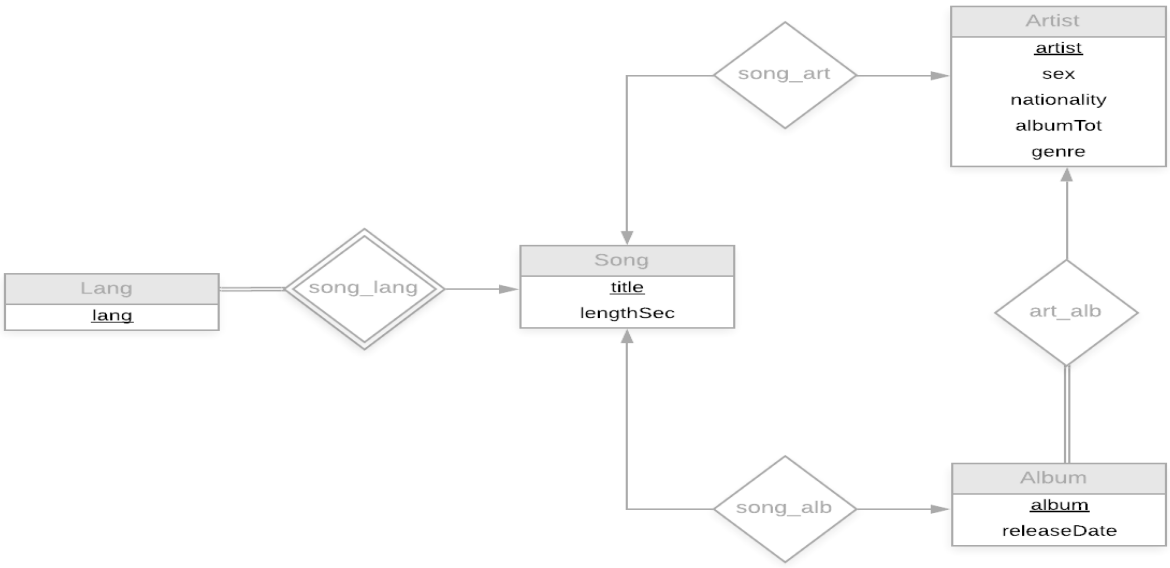
A. Artist table
- name
- sex
- nationality
- total number of albums
- genre

B. Album table
- name
- artist
- release date

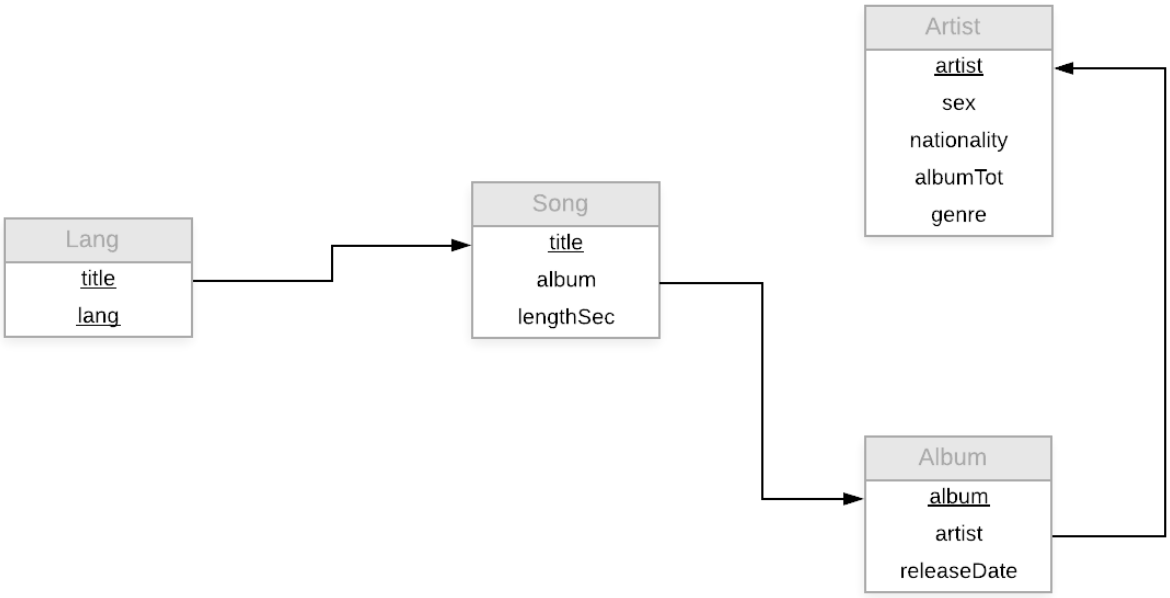
C. Song table
- title
- album
- record length (in second)

D. Language table
- title
- language

II. ER Diagram



III. Database Schema Diagram



IV. Classes and Methods of Java Codes

There are two packages, database and windows. Database package contains classes related to sql queries and main class meanwhile windows package contains java swing components.

A. Package database
- Spotify.java: This is the main class. When run, it creates a GUI window for the entire program. All the preparation works are done here such as declaring systematic variables in public static or creating methods to connect and close.
- Print.java:
  a) *printTable* takes a table and print out in text area.
  b) *printAll* repeats this for all tables.
- Select.java:
  a) *selectGenre* takes user input value and selects artists of the given genre.
  b) *selectOutdated* selects outdated records with given year extent.
  c) *selectMinCount* returns artists who appear more than the given value along with the song titles.
- Insert.java:
  a) *insertArt* receives all attribute values needed and inserts new tuple into artist table.
- Update.java:
  a) *updateArtNat* updates any nationality attributes equal to the first input with second input.

b) *updateLenDate* achieves two operations. Any record length shorter than the input gets updated with the same value while the rest remains. Then, it updates every release date into '2020-07-01'.

- Delete.java:
  a) *deleteLang* takes language as an input value and deletes any tuples according to it.

B. Package windows
   - StartScreen.java: Start screen is created by the call from main class. It has 'connect' button which connects java with MySql and invokes main screen. It automatically disposes itself afterwards.
   - MainScreen.java: Main screen contains all sorts of buttons to run different queries, text area where outputs are printed and 'exit' button. Each button invokes different pop-up windows to receive user input and 'exit' button closes all connections.
   - SelectScreen.java: Select screen is opened via 'select' button in main screen.
   - InsertScreen.java: Insert screen is opened via 'insert' button in main screen.
   - UpdateScreen.java: Update screen is opened via 'update' button in main screen.
   - DeleteScreen.java: Delete screen is opened via 'delete' button in main screen.

V. Main Class and Connection Configuration

As stated above, Spotify.java contains the main method for the whole application. It invokes start screen which then connects through 'connect' button. After start screen is replaced by the main screen, you can execute queries as much as you want and to terminate, simply click 'exit' button.

VI. Assignments

Note that below numbering starts with assignment (1).

a) Artist, album and song tables all have at least 3 columns.
b) Total number of tuples from all tables counts to 38.
c) All tables have stated integrity constraints.
d) All tables are in 3NF form.
e) There is an index created using album attribute in song table and artist attribute in album table.
f) There is one view named playlist defined using song, album and artist tables.
g) All queries take user input via parameterized values.
h) *insertArt* inserts new tuple to artist table.
i) *updateArtNat* updates artist table with new nationality values.

j) *updateLenDate* consists of two steps. One updating record length in song table and the other updating release date in album table. It uses transaction to operate them both.
k) *deleteLang* deletes tuples in language table with given language.
l) *selectGenre* selects all tuples in artist table with given genre.
m) *selectMinCount* selects from view table, artists whose frequency is higher than the input value using nested queries and join.
n) *selectOutdated* selects outdated records from view table.
o) 'print all' button prints out contents of all tables.
p) 'exit' button closes all connections.

VII. SQL Script

The following code can also be found in 'createdb.sql' script.

```
/* Creating database */
create database spotify;
use spotify;

/* Creating tables */
create table artist (
        artist varchar(20),
        sex char(1),
        nationality varchar(20),
        albumTot int,
        genre varchar(20),
        primary key (artist),
        check (sex in ('M', 'F')),
        check (albumTot >= 0));

create table album (
        album varchar(30),
        artist varchar(20) not null,
        releaseDate date,
        primary key (album),
        foreign key (artist) references Artist (artist));

create table song (
        title varchar(40),
        album varchar(30) not null,
        lengthSec int,
        primary key (title),
        foreign key (album) references Album (album),
        check (lengthSec >= 0));

create table lang (
```

```
        title varchar(40),
        lang varchar(20),
        primary key (title, lang),
        foreign key (title) references Song (title));

/* Inserting data */
insert into artist values ('DaBaby', 'M', 'American', 8, 'Hip hop');
insert into artist values ('Harry Styles', 'M', 'English', 2, 'Pop');
insert into artist values ('Dua Lipa', 'F', 'English', 2, 'Pop');
insert into artist values ('Lady Gaga', 'F', 'American', 7,'Pop');
insert into artist values ('The Weeknd', 'M', 'Canadian', 6, 'R&B');
insert into artist values ('Drake', 'M', 'Canadian', 11, 'Hip hop');
insert into artist values ('ROSALIA', 'F', 'Spanish', 2, 'Latin pop');
insert into artist values ('Melanie Martinez', 'F', 'American', 2, 'Pop');

insert into album values ('BLAME IT ON BABY', 'DaBaby', '2020-04-17');
insert into album values ('Fine Line', 'Harry Styles', '2019-12-13');
insert into album values ('Future Nostalgia', 'Dua Lipa', '2020-03-27');
insert into album values ('Chromatica', 'Lady Gaga', '2020-03-29');
insert into album values ('After Hours', 'The Weeknd', '2020-03-20');
insert into album values ('Dark Lane Demo Tapes', 'Drake', '2020-03-01');
insert into album values ('TKN (feat. Travis Scott)', 'ROSALIA', '2020-03-28');
insert into album values ('Cry Baby (Deluxe Edition)', 'Melanie Martinez', '2015-08-14');

insert into song values ('ROCKSTAR (feat. Roddy Ricch)', 'BLAME IT ON BABY', 182);
insert into song values ('Watermelon Sugar', 'Fine Line', 174);
insert into song values ('Break My Heart', 'Future Nostalgia', 222);
insert into song values ('Rain On Me (with Ariana Grande)', 'Chromatica', 182);
insert into song values ('Blinding Lights', 'After Hours', 202);
insert into song values ('Toosie Slide', 'Dark Lane Demo Tapes', 213);
insert into song values ('TKN (feat. Travis Scott)', 'TKN (feat. Travis Scott)', 129);
insert into song values ('Play Date', 'Cry Baby (Deluxe Edition)', 180);
insert into song values ('In Your Eyes', 'After Hours', 238);
insert into song values ('Sour Candy (with BLACKPINK)', 'Chromatica', 158);

insert into lang values ('ROCKSTAR (feat. Roddy Ricch)', 'English');
insert into lang values ('Watermelon Sugar', 'English');
insert into lang values ('Break My Heart', 'English');
insert into lang values ('Rain On Me (with Ariana Grande)', 'English');
insert into lang values ('Blinding Lights', 'English');
insert into lang values ('Toosie Slide', 'English');
insert into lang values ('TKN (feat. Travis Scott)', 'English');
insert into lang values ('TKN (feat. Travis Scott)', 'Spanish');
insert into lang values ('Play Date', 'English');
insert into lang values ('In Your Eyes', 'English');
insert into lang values ('Sour Candy (with BLACKPINK)', 'English');
```

```
insert into lang values ('Sour Candy (with BLACKPINK)', 'Korean');

/* Defining index */
create index album_id on song(album);
create index artist_id on album(artist);

/* Defining view */
create view playlist as
select title, artist.artist, album.album, releaseDate, genre, lengthSec
from song, album, artist
where artist.artist = album.artist and album.album = song.album;
```

The following code can also be found in ‘dropdb.sql’ script.

```
/* Dropping all tables */
drop table lang;
drop table song;
drop table album;
drop table artist;

/* Dropping database */
drop database spotify;
```

VIII. Java Codes

The following code can also be found in ‘sourcefiles.zip’ file.

A. Package database

- Spotify.java

```
package database;
import java.sql.*;

import windows.MainScreen;
import windows.StartScreen;

public class Spotify {

	// Setting systematic variables
	static String user = "root";
	static String pw = "qwerty123";
	public static String db = "spotify";
	static String url = "jdbc:mysql://localhost:3306/" + db +
"?&serverTimezone=UTC";

	// Declaring database variables
	public static Connection conn = null;
```

```
static Statement stmt = null;
static PreparedStatement pStmt = null;
static ResultSet rset = null;
static String sql = "";

// Array of all table names
static String[] tables = {"album", "artist", "lang", "song"};


////////////////////////////////////////////////////////////////////////////////////////////////////////////


// Preparing application
public static void connect() {
	try {
	conn = DriverManager.getConnection(url, user, pw);
	stmt = conn.createStatement();
	} catch (SQLException e) {
		e.printStackTrace();
	}
}

// Terminating application
public static void close() {
	if (rset != null) {
		try {
			rset.close();
			MainScreen.textArea.append("... Closing result set ...\n");
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}

	if (stmt != null) {
		try {
			stmt.close();
			MainScreen.textArea.append("... Closing statement ...\n");
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}

	if (pStmt != null) {
		try {
			pStmt.close();
```

```
                                MainScreen.textArea.append("... Closing prepared
statement ...\n");
                        } catch (SQLException e) {
                                e.printStackTrace();
                        }
                }

                if (conn != null) {
                        try {
                                conn.close();
                                MainScreen.textArea.append("... Closing connection " +
conn.toString() + " ...\n");
                        } catch (SQLException e) {
                                e.printStackTrace();
                        }
                }
                MainScreen.textArea.append("Application orderly terminated. Goodbye!");
        }

        // Main method for application
        public static void main(String[] args) {
                new StartScreen().setVisible(true);
        }

}
```

- Print.java

```
package database;
import java.sql.*;

import windows.MainScreen;

public class Print {

        public static String rightPad(String str, int len) {
           return String.format(" %-" + len + "." + len + "s", str);
        }

        // Printing all tables in given array
        public static void printAll() {
                for (int i = 0; i < Spotify.tables.length; i++) {
                        Print.printTable(Spotify.tables[i]);
                }
        }

        // Printing one table
```

```
	public static void printTable(String table) {
		try {
			Spotify.sql = "select * from " + table;
			Spotify.rset = Spotify.stmt.executeQuery(Spotify.sql);

			// Printing header for given table
			switch (table) {
			case "album":
				MainScreen.textArea.append("[----- ALBUM -----]\n");
				MainScreen.textArea.append(String.format(" Name %20s |
Artist %9s | Release Date\n", "", ""));
				MainScreen.textArea.append(String.format("%63s\n",
"").replace(' ', '-'));
				break;

			case "artist":
				MainScreen.textArea.append("[----- ARTIST -----]\n");
				MainScreen.textArea.append(String.format(" Name %13s |
Sex | Nationality | Total Album Numbers | Genre\n", ""));
				MainScreen.textArea.append(String.format("%76s\n",
"").replace(' ', '-'));
				break;

			case "lang":
				MainScreen.textArea.append("[----- LANGUAGE -----]\n");
				MainScreen.textArea.append(String.format(" Title %27s |
Language\n", ""));
				MainScreen.textArea.append(String.format("%52s\n",
"").replace(' ', '-'));
				break;

			case "song":
				MainScreen.textArea.append("[----- SONG -----]\n");
				MainScreen.textArea.append(String.format(" Title %27s |
Album %19s | Length in Sec %3s\n", "", "", ""));
				MainScreen.textArea.append(String.format("%81s\n",
"").replace(' ', '-'));
				break;

			default:
				MainScreen.textArea.append("Table doesn't exist.\n");
				break;
			}

			// Printing data for given table
			while (Spotify.rset.next()) {
```

```
                        switch (table) {
                        case "album":
                                String alb_name = Spotify.rset.getString("album");
                                String alb_art = Spotify.rset.getString("artist");
                                String alb_date =
Spotify.rset.getString("releaseDate");

                                MainScreen.textArea.append(rightPad(alb_name,
26));
                                MainScreen.textArea.append(rightPad(alb_art, 18));
                                MainScreen.textArea.append(rightPad(alb_date, 12));
                                break;

                        case "artist":
                                String art_name = Spotify.rset.getString("artist");
                                String art_sex = Spotify.rset.getString("sex");
                                String art_nat = Spotify.rset.getString("nationality");
                                int art_tot = Spotify.rset.getInt("albumTot");
                                String art_gen = Spotify.rset.getString("genre");

                                MainScreen.textArea.append(rightPad(art_name ,
21));
                                MainScreen.textArea.append(rightPad(art_sex, 4));
                                MainScreen.textArea.append(rightPad(art_nat , 21));

        MainScreen.textArea.append(rightPad(Integer.toString(art_tot) , 13));
                                MainScreen.textArea.append(rightPad(art_gen , 11));
                                break;

                        case "lang":
                                String lang_tit = Spotify.rset.getString("title");
                                String lang_lang = Spotify.rset.getString("lang");

                                MainScreen.textArea.append(rightPad(lang_tit, 35));
                                MainScreen.textArea.append(rightPad(lang_lang,
17));
                                break;

                        case "song":
                                String song_tit = Spotify.rset.getString("title");
                                String song_alb = Spotify.rset.getString("album");
                                int song_len = Spotify.rset.getInt("lengthSec");

                                MainScreen.textArea.append(rightPad(song_tit, 35));
                                MainScreen.textArea.append(rightPad(song_alb,
32));
```

```
		MainScreen.textArea.append(rightPad(Integer.toString(song_len), 8));
					break;

				default:
					MainScreen.textArea.append("Table doesn't
exist.\n");
					break;

				}
				MainScreen.textArea.append("\n");
			}
			MainScreen.textArea.append("\n");
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}

}
```

- Select.java

```
package database;

import java.sql.SQLException;

import windows.MainScreen;

public class Select {

	// Aligning strings from the left
	public static String rightPad(String str, int len) {
	   return String.format(" %-" + len + "." + len + "s", str);
	}

	// Selecting artists of given genre
	public static void selectGenre(String genre) {
		try {
			switch (genre) {
			case "Hip hop":
				Spotify.sql = "select * from artist where genre = 'Hip hop'";
				break;

			case "Latin Pop":
				Spotify.sql = "select * from artist where genre = 'Latin pop'";
				break;
```

```
                        case "Pop":
                                Spotify.sql = "select * from artist where genre = 'Pop'";
                                break;

                        case "R&B":
                                Spotify.sql = "select * from artist where genre = 'R&B'";
                                break;

                        default:
                                MainScreen.textArea.append("No artist of given genre.\n");
                                break;
                        }

                        Spotify.rset = Spotify.stmt.executeQuery(Spotify.sql);

                        MainScreen.textArea.append(String.format(" Name %13s | Sex |
Nationality | Total Album Numbers | Genre\n", ""));
                        MainScreen.textArea.append(String.format("%76s\n", "").replace(' ',
'-'));

                        while (Spotify.rset.next()) {
                                String art_name = Spotify.rset.getString("artist");
                                String art_sex = Spotify.rset.getString("sex");
                                String art_nat = Spotify.rset.getString("nationality");
                                int art_tot = Spotify.rset.getInt("albumTot");
                                String art_gen = Spotify.rset.getString("genre");

                                MainScreen.textArea.append(rightPad(art_name , 21));
                                MainScreen.textArea.append(rightPad(art_sex, 4));
                                MainScreen.textArea.append(rightPad(art_nat , 21));

        MainScreen.textArea.append(rightPad(Integer.toString(art_tot) , 13));
                                MainScreen.textArea.append(rightPad(art_gen , 11));
                                MainScreen.textArea.append("\n");
                        }
                        MainScreen.textArea.append("\n");
                } catch (SQLException e) {
                        e.printStackTrace();
                }
        }

        // Selecting outdated records from view table with given year extent
        public static void selectOutdated(int period) {
                int curr_year = 2020;
                int min_year = curr_year - period;
```

```
		try {
			Spotify.sql = "select title, artist, releaseDate from playlist where
year(releaseDate) <= " + min_year;

			Spotify.rset = Spotify.stmt.executeQuery(Spotify.sql);

			MainScreen.textArea.append(String.format(" Title %27s | Artist %9s
| Release Date\n", "", ""));
			MainScreen.textArea.append(String.format("%71s\n", "").replace(' ',
'-'));

			while (Spotify.rset.next()) {
				String pl_tit = Spotify.rset.getString("title");
				String pl_art = Spotify.rset.getString("artist");
				String pl_rd = Spotify.rset.getString("releaseDate");

				MainScreen.textArea.append(rightPad(pl_tit, 35));
				MainScreen.textArea.append(rightPad(pl_art, 18));
				MainScreen.textArea.append(rightPad(pl_rd, 12));
				MainScreen.textArea.append("\n");
			}
			MainScreen.textArea.append("\n");
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}

	// Selecting records with at least minCount songs from same artist
	public static void selectMinCount(int min_count) {
		try {
			Spotify.sql = "select title, artist from (song natural join album) join
artist using (artist) where artist in (select artist from playlist group by artist having
count(*) >= " + min_count + ")";

			Spotify.rset = Spotify.stmt.executeQuery(Spotify.sql);

			MainScreen.textArea.append(String.format(" Title %27s |
Artist %9s\n", "", ""));
			MainScreen.textArea.append(String.format("%55s\n", "").replace(' ',
'-'));

			while (Spotify.rset.next()) {
				String pl_tit = Spotify.rset.getString("title");
				String pl_art = Spotify.rset.getString("artist");

				MainScreen.textArea.append(rightPad(pl_tit, 35));
```

```
                        MainScreen.textArea.append(rightPad(pl_art, 18));
                        MainScreen.textArea.append("\n");
                    }
                    MainScreen.textArea.append("\n");
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }

}
```

- Insert.java

```
package database;
import java.sql.*;

public class Insert {

    public static void insertArt(String artist, String sex, String nationality, int
albumTot, String genre) {

        try {
            Spotify.pStmt = Spotify.conn.prepareStatement("insert into Artist
values (?, ?, ?, ?, ?)");

            Spotify.pStmt.setString(1, artist);
            Spotify.pStmt.setString(2, sex);
            Spotify.pStmt.setString(3, nationality);
            Spotify.pStmt.setInt(4, albumTot);
            Spotify.pStmt.setString(5, genre);

            Spotify.pStmt.executeUpdate();

            Print.printTable("artist");

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

}
```

- Update.java

```
package database;
import java.sql.*;

import windows.MainScreen;
```

```
public class Update {

	// Updating all nationalities from before to after
	public static void updateArtNat(String before, String after) {
		try {
			Spotify.pStmt = Spotify.conn.prepareStatement("update artist set
nationality = ? where nationality = ?");

			Spotify.pStmt.setString(1, after);
			Spotify.pStmt.setString(2, before);

			Spotify.pStmt.executeUpdate();
		} catch (SQLException e) {
			e.printStackTrace();
		}

		Print.printTable("artist");
	}

	// Updating record lengths and release dates
	public static void updateLenDate(int minSec) throws SQLException {
		try {
			Spotify.conn.setAutoCommit(false);

			// Transaction step 1: Updating every record length with the given
value if shorter
			Spotify.pStmt = Spotify.conn.prepareStatement("update Song set
lengthSec = ? where lengthSec < ?");

			Spotify.pStmt.setInt(1, minSec);
			Spotify.pStmt.setInt(2, minSec);

			Spotify.pStmt.executeUpdate();

			// Transaction step 2: Updating all release dates of all albums to
2020-07-01
			Spotify.sql = "update album set releaseDate = '2020-07-01'";

			Spotify.stmt.executeUpdate(Spotify.sql);

			Spotify.conn.commit();

			MainScreen.textArea.append("Transaction committed
successfully.\n");
```

```
                Print.printTable("song");
                Print.printTable("album");
            } catch (SQLException e) {
                e.printStackTrace();

                if (Spotify.conn != null) {
                    try {
                        MainScreen.textArea.append("Transaction rolled
back.\n");
                        Spotify.conn.rollback();
                    } catch (SQLException exc) {
                        exc.printStackTrace();
                    }
                }
            } finally {
                Spotify.conn.setAutoCommit(true);
            }
        }

}
```

- Delete.java

```
package database;
import java.sql.*;

public class Delete {

        // Deleting any tuple with given language
        public static void deleteLang(String lang) {
            try {
                Spotify.pStmt = Spotify.conn.prepareStatement("delete from lang
where lang = ?");

                Spotify.pStmt.setString(1, lang);

                Spotify.pStmt.executeUpdate();
            } catch (SQLException e) {
                e.printStackTrace();
            }

            Print.printTable("lang");
        }

}
```

B. Package windows

- StartScreen.java

```
package windows;
import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import database.Spotify;

import javax.swing.JLabel;
import javax.swing.SwingConstants;
import javax.swing.JButton;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.SQLException;
import java.awt.Color;
import javax.swing.ImageIcon;

public class StartScreen extends JFrame {

	private JPanel contentPane;
	JLabel press_connect = new JLabel("Press connect to continue");
	JButton connect = new JButton("CONNECT");
	private final JLabel background = new JLabel("");

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					StartScreen frame = new StartScreen();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
```

```
	 * Create the frame.
	 */
	public StartScreen() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 800, 700);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);
		contentPane.setLayout(null);
		connect.setFont(new Font("Marker Felt", Font.PLAIN, 13));

		connect.setBounds(329, 598, 142, 29);
		contentPane.add(connect);

		press_connect.setForeground(new Color(143, 188, 143));
		press_connect.setFont(new Font("Marker Felt", Font.PLAIN, 28));
		press_connect.setHorizontalAlignment(SwingConstants.CENTER);
		press_connect.setBounds(0, 552, 800, 44);
		contentPane.add(press_connect);
		background.setIcon(new
ImageIcon("/Users/katejeon/Documents/Spring_2020/CPSC_20471/spotify_project/bac
kground.png"));
		background.setBounds(0, 0, 800, 678);

		contentPane.add(background);

		connect.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				Spotify.connect();
				new MainScreen().setVisible(true);
				dispose();
			}
		}
		);

	}

}
```

- MainScreen.java

```
package windows;
import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
```

```
import javax.swing.border.EmptyBorder;

import database.Delete;
import database.Insert;
import database.Print;
import database.Spotify;
import database.Update;

import java.awt.GridLayout;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.Font;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JLabel;
import java.awt.Color;
import javax.swing.SwingConstants;

public class MainScreen extends JFrame {

	private JPanel contentPane;
	JButton insert = new JButton("INSERT");
	JButton update = new JButton("UPDATE");
	JButton select = new JButton("SELECT");
	JButton delete = new JButton("DELETE");
	JButton print_all = new JButton("PRINT ALL");
	JButton exit = new JButton("EXIT");
	JLabel press_exit = new JLabel("Press exit to terminate");
	JScrollPane scrollPane = new JScrollPane();
	public static JTextArea textArea = new JTextArea();

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					MainScreen frame = new MainScreen();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
```

```
}

/**
 * Create the frame.
 */
public MainScreen() {
	setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	setBounds(100, 100, 800, 700);
	contentPane = new JPanel();
	contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
	setContentPane(contentPane);
	contentPane.setLayout(null);

	insert.setFont(new Font("Marker Felt", Font.PLAIN, 17));
	insert.setBounds(25, 34, 134, 40);
	contentPane.add(insert);

	insert.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent e) {
			new InsertScreen().setVisible(true);
		}
	});

	update.setFont(new Font("Marker Felt", Font.PLAIN, 17));
	update.setBounds(334, 34, 134, 40);
	contentPane.add(update);

	update.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent e) {
			new UpdateScreen().setVisible(true);
		}
	});

	select.setFont(new Font("Marker Felt", Font.PLAIN, 17));
	select.setBounds(180, 34, 134, 40);
	contentPane.add(select);

	select.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent e) {
			new SelectScreen().setVisible(true);
		}
	});

	delete.setFont(new Font("Marker Felt", Font.PLAIN, 17));
	delete.setBounds(490, 34, 134, 40);
	contentPane.add(delete);
```

```
		delete.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				new DeleteScreen().setVisible(true);
			}
		});

		print_all.setFont(new Font("Marker Felt", Font.PLAIN, 17));
		print_all.setBounds(645, 34, 134, 40);
		contentPane.add(print_all);

		print_all.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				Print.printAll();
			}
		});

		scrollPane.setBounds(25, 112, 754, 445);
		contentPane.add(scrollPane);

		textArea.setFont(new Font("Courier", Font.PLAIN, 14));
		scrollPane.setViewportView(textArea);

		textArea.setText("... Connected to database " + Spotify.db + " in MySQL
with " + Spotify.conn.toString() + " ...\n");

		exit.setFont(new Font("Marker Felt", Font.PLAIN, 13));
		exit.setBounds(329, 617, 142, 29);
		contentPane.add(exit);

		exit.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				Spotify.close();
			}
		});

		press_exit.setHorizontalAlignment(SwingConstants.CENTER);
		press_exit.setForeground(new Color(0, 0, 0));
		press_exit.setFont(new Font("Marker Felt", Font.PLAIN, 28));
		press_exit.setBounds(0, 571, 800, 44);
		contentPane.add(press_exit);
	}
}
```

- SelectScreen.java

```
package windows;
```

```
import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import database.Delete;
import database.Select;

import javax.swing.JLabel;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.SwingConstants;
import javax.swing.JTextField;
import javax.swing.JButton;

public class SelectScreen extends JFrame {

	private JPanel contentPane;
	private JTextField textField_genre = new JTextField();
	JLabel header1 = new JLabel("Select artist with genre");
	JLabel genre = new JLabel("Genre:");
	private final JButton enter1 = new JButton("ENTER");
	private final JLabel header2 = new JLabel("Select outdated records");
	private final JLabel year_period = new JLabel("Year period:");
	private final JTextField textField_yr_pr = new JTextField();
	private final JButton enter2 = new JButton("ENTER");
	private final JLabel header3 = new JLabel("Select with minimum counts");
	private final JLabel min_count = new JLabel("Minimum count:");
	private final JTextField textField_min_count = new JTextField();
	private final JButton enter3 = new JButton("ENTER");

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					SelectScreen frame = new SelectScreen();
					frame.setVisible(true);
				} catch (Exception e) {
```

```
                    e.printStackTrace();
                }
            }
        });
}

/**
 * Create the frame.
 */
public SelectScreen() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 450, 400);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        // Selecting artist with genre
        header1.setHorizontalAlignment(SwingConstants.CENTER);
        header1.setFont(new Font("Marker Felt", Font.PLAIN, 19));
        header1.setBounds(0, 20, 450, 20);
        contentPane.add(header1);

        genre.setFont(new Font("Arial", Font.PLAIN, 13));
        genre.setBounds(35, 60, 140, 20);
        contentPane.add(genre);

        textField_genre.setFont(new Font("Arial", Font.PLAIN, 13));
        textField_genre.setColumns(10);
        textField_genre.setBounds(185, 60, 225, 20);
        contentPane.add(textField_genre);

        enter1.setFont(new Font("Marker Felt", Font.PLAIN, 11));
        enter1.setBounds(165, 90, 120, 30);
        contentPane.add(enter1);

        enter1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String art_gen = textField_genre.getText();
                Select.selectGenre(art_gen);
                dispose();
            }
        });

        // Selecting outdated records
        header2.setHorizontalAlignment(SwingConstants.CENTER);
```

```
header2.setFont(new Font("Marker Felt", Font.PLAIN, 19));
header2.setBounds(0, 142, 450, 20);

contentPane.add(header2);
year_period.setFont(new Font("Arial", Font.PLAIN, 13));
year_period.setBounds(35, 182, 140, 20);

contentPane.add(year_period);
textField_yr_pr.setFont(new Font("Arial", Font.PLAIN, 13));
textField_yr_pr.setColumns(10);
textField_yr_pr.setBounds(185, 182, 225, 20);

contentPane.add(textField_yr_pr);
enter2.setFont(new Font("Marker Felt", Font.PLAIN, 11));
enter2.setBounds(165, 212, 120, 30);
contentPane.add(enter2);

enter2.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		int period = Integer.parseInt(textField_yr_pr.getText());
		Select.selectOutdated(period);
		dispose();
	}
});

// Selecting with minimum counts
header3.setHorizontalAlignment(SwingConstants.CENTER);
header3.setFont(new Font("Marker Felt", Font.PLAIN, 19));
header3.setBounds(0, 261, 450, 20);
contentPane.add(header3);

min_count.setFont(new Font("Arial", Font.PLAIN, 13));
min_count.setBounds(35, 301, 140, 20);

contentPane.add(min_count);
textField_min_count.setFont(new Font("Arial", Font.PLAIN, 13));
textField_min_count.setColumns(10);
textField_min_count.setBounds(185, 301, 225, 20);

contentPane.add(textField_min_count);
enter3.setFont(new Font("Marker Felt", Font.PLAIN, 11));
enter3.setBounds(165, 331, 120, 30);
contentPane.add(enter3);

enter3.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
```

```
                                int pl_min_count =
Integer.parseInt(textField_min_count.getText());
                                Select.selectMinCount(pl_min_count);
                                dispose();
                        }
                });
        }

}
```

- InsertScreen.java

```
package windows;

import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import database.Insert;

import javax.swing.JLabel;
import javax.swing.SwingConstants;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class InsertScreen extends JFrame {

        private JPanel contentPane;
        JLabel header = new JLabel("Add new artist");
        JLabel name = new JLabel("Name:");
        JLabel sex = new JLabel("Sex:");
        JLabel nationality = new JLabel("Nationality:");
        JLabel albumTot = new JLabel("Total Album Number:");
        JLabel genre = new JLabel("Genre:");
        private JTextField textField_name;
        private JTextField textField_sex;
        private JTextField textField_nationality;
        private JTextField textField_albumTot;
        private JTextField textField_genre;
        private JButton enter;
```

```
/**
 * Launch the application.
 */
public static void main(String[] args) {
	EventQueue.invokeLater(new Runnable() {
		public void run() {
			try {
				InsertScreen frame = new InsertScreen();
				frame.setVisible(true);
			} catch (Exception e) {
				e.printStackTrace();
			}
		}
	});
}

/**
 * Create the frame.
 */
public InsertScreen() {

	setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	setBounds(100, 100, 450, 300);
	contentPane = new JPanel();
	contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
	setContentPane(contentPane);
	contentPane.setLayout(null);

	header.setFont(new Font("Marker Felt", Font.PLAIN, 19));
	header.setHorizontalAlignment(SwingConstants.CENTER);
	header.setBounds(0, 20, 450, 20);
	contentPane.add(header);

	name.setFont(new Font("Arial", Font.PLAIN, 13));
	name.setBounds(35, 60, 140, 20);
	contentPane.add(name);

	sex.setFont(new Font("Arial", Font.PLAIN, 13));
	sex.setBounds(35, 90, 140, 20);
	contentPane.add(sex);

	nationality.setFont(new Font("Arial", Font.PLAIN, 13));
	nationality.setBounds(35, 120, 140, 20);
	contentPane.add(nationality);

	albumTot.setFont(new Font("Arial", Font.PLAIN, 13));
```

```
albumTot.setBounds(35, 150, 140, 20);
contentPane.add(albumTot);

genre.setFont(new Font("Arial", Font.PLAIN, 13));
genre.setBounds(35, 180, 140, 20);
contentPane.add(genre);

textField_name = new JTextField();
textField_name.setFont(new Font("Arial", Font.PLAIN, 13));
textField_name.setBounds(185, 60, 225, 20);
contentPane.add(textField_name);
textField_name.setColumns(10);

textField_sex = new JTextField();
textField_sex.setFont(new Font("Arial", Font.PLAIN, 13));
textField_sex.setColumns(10);
textField_sex.setBounds(185, 90, 225, 20);
contentPane.add(textField_sex);

textField_nationality = new JTextField();
textField_nationality.setFont(new Font("Arial", Font.PLAIN, 13));
textField_nationality.setColumns(10);
textField_nationality.setBounds(185, 120, 225, 20);
contentPane.add(textField_nationality);

textField_albumTot = new JTextField();
textField_albumTot.setFont(new Font("Arial", Font.PLAIN, 13));
textField_albumTot.setColumns(10);
textField_albumTot.setBounds(185, 150, 225, 20);
contentPane.add(textField_albumTot);

textField_genre = new JTextField();
textField_genre.setFont(new Font("Arial", Font.PLAIN, 13));
textField_genre.setColumns(10);
textField_genre.setBounds(185, 180, 225, 20);
contentPane.add(textField_genre);

enter = new JButton("ENTER");
enter.setFont(new Font("Marker Felt", Font.PLAIN, 11));
enter.setBounds(168, 227, 117, 29);
contentPane.add(enter);

enter.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                String art_artist = textField_name.getText();
                String art_sex = textField_sex.getText();
```

```
                                String art_nationality = textField_nationality.getText();
                                int art_albumTot =
Integer.parseInt(textField_albumTot.getText());
                                String art_genre = textField_genre.getText();
                                Insert.insertArt(art_artist, art_sex, art_nationality,
art_albumTot, art_genre);
                                dispose();
                        }
                });
        }

}
```

- UpdateScreen.java

```
package windows;

import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import database.Insert;
import database.Update;

import javax.swing.JLabel;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.SQLException;

import javax.swing.SwingConstants;
import javax.swing.JTextField;

public class UpdateScreen extends JFrame {

        private JPanel contentPane;
        private JTextField textField_nat_from = new JTextField();
        private JTextField textField_nat_to = new JTextField();
        private JTextField textField_min_sec = new JTextField();
        JLabel header1 = new JLabel("Update artist nationality");
        JLabel header2 = new JLabel("Update record length and release date");
        JLabel nat_from = new JLabel("From:");
        JLabel nat_to = new JLabel("To:");
```

```
JLabel min_sec = new JLabel("Minimum second:");
private JButton enter1;
private JButton enter2;

/**
 * Launch the application.
 */
public static void main(String[] args) {
	EventQueue.invokeLater(new Runnable() {
		public void run() {
			try {
				UpdateScreen frame = new UpdateScreen();
				frame.setVisible(true);
			} catch (Exception e) {
				e.printStackTrace();
			}
		}
	});
}

/**
 * Create the frame.
 */
public UpdateScreen() {
	setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	setBounds(100, 100, 450, 300);
	contentPane = new JPanel();
	contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
	setContentPane(contentPane);
	contentPane.setLayout(null);

	// Updating artist nationality
	header1.setHorizontalAlignment(SwingConstants.CENTER);
	header1.setFont(new Font("Marker Felt", Font.PLAIN, 19));
	header1.setBounds(0, 20, 450, 20);
	contentPane.add(header1);
	nat_from.setFont(new Font("Arial", Font.PLAIN, 13));

	nat_from.setBounds(35, 60, 140, 20);
	contentPane.add(nat_from);
	nat_to.setFont(new Font("Arial", Font.PLAIN, 13));

	nat_to.setBounds(35, 90, 140, 20);
	contentPane.add(nat_to);
```

```
textField_nat_from.setFont(new Font("Arial", Font.PLAIN, 13));
textField_nat_from.setBounds(185, 60, 225, 20);
contentPane.add(textField_nat_from);
textField_nat_from.setColumns(10);

textField_nat_to.setFont(new Font("Arial", Font.PLAIN, 13));
textField_nat_to.setBounds(185, 90, 225, 20);
contentPane.add(textField_nat_to);
textField_nat_to.setColumns(10);

enter1 = new JButton("ENTER");
enter1.setFont(new Font("Marker Felt", Font.PLAIN, 11));
enter1.setBounds(162, 122, 117, 29);
contentPane.add(enter1);

enter1.addActionListener(new ActionListener( ) {
        public void actionPerformed(ActionEvent e) {
                String art_nat_from = textField_nat_from.getText();
                String art_nat_to = textField_nat_to.getText();
                Update.updateArtNat(art_nat_from, art_nat_to);
                dispose();
        }
});

// Updating record length
header2.setHorizontalAlignment(SwingConstants.CENTER);
header2.setFont(new Font("Marker Felt", Font.PLAIN, 19));
header2.setBounds(0, 163, 450, 20);
contentPane.add(header2);

min_sec.setFont(new Font("Arial", Font.PLAIN, 13));
min_sec.setBounds(35, 203, 140, 20);
contentPane.add(min_sec);

enter2 = new JButton("ENTER");
enter2.setFont(new Font("Marker Felt", Font.PLAIN, 11));
enter2.setBounds(162, 235, 117, 29);
contentPane.add(enter2);
textField_min_sec.setFont(new Font("Arial", Font.PLAIN, 13));
textField_min_sec.setColumns(10);
textField_min_sec.setBounds(185, 203, 225, 20);

contentPane.add(textField_min_sec);

enter2.addActionListener(new ActionListener( ) {
        public void actionPerformed(ActionEvent e) {
```

```
                                int song_min_sec =
Integer.parseInt(textField_min_sec.getText());
                                try {
                                        Update.updateLenDate(song_min_sec);
                                } catch (SQLException e1) {
                                        // TODO Auto-generated catch block
                                        e1.printStackTrace();
                                }
                                dispose();
                        }
                });
        }

}
```

- DeleteScreen.java

```
package windows;

import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import database.Delete;

import javax.swing.JLabel;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.SwingConstants;
import javax.swing.JTextField;
import javax.swing.JButton;

public class DeleteScreen extends JFrame {

        private JPanel contentPane;
        private JTextField textField_lang;
        JLabel header = new JLabel("Delete language");
        JLabel lang = new JLabel("Language:");
        JButton enter = new JButton("ENTER");

        /**
         * Launch the application.
```

```
 */
public static void main(String[] args) {
	EventQueue.invokeLater(new Runnable() {
		public void run() {
			try {
				DeleteScreen frame = new DeleteScreen();
				frame.setVisible(true);
			} catch (Exception e) {
				e.printStackTrace();
			}
		}
	});
}

/**
 * Create the frame.
 */
public DeleteScreen() {
	setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	setBounds(100, 100, 450, 300);
	contentPane = new JPanel();
	contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
	setContentPane(contentPane);
	contentPane.setLayout(null);

	header.setHorizontalAlignment(SwingConstants.CENTER);
	header.setFont(new Font("Marker Felt", Font.PLAIN, 19));
	header.setBounds(0, 20, 450, 20);
	contentPane.add(header);

	lang.setFont(new Font("Arial", Font.PLAIN, 13));
	lang.setBounds(35, 60, 140, 20);
	contentPane.add(lang);

	textField_lang = new JTextField();
	textField_lang.setFont(new Font("Arial", Font.PLAIN, 13));
	textField_lang.setColumns(10);
	textField_lang.setBounds(185, 60, 225, 20);
	contentPane.add(textField_lang);

	enter.setFont(new Font("Marker Felt", Font.PLAIN, 11));
	enter.setBounds(165, 90, 120, 30);
	contentPane.add(enter);

	enter.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent e) {
```

```
                String lang_lang = textField_lang.getText();
                Delete.deleteLang(lang_lang);
                dispose();
            }
        });
    }

}
```