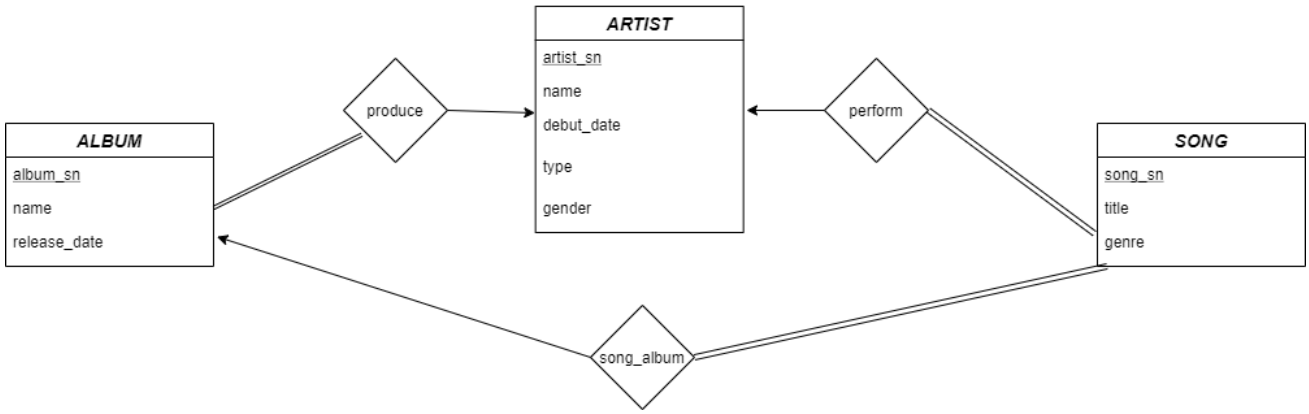
2020-1

Database Project Report

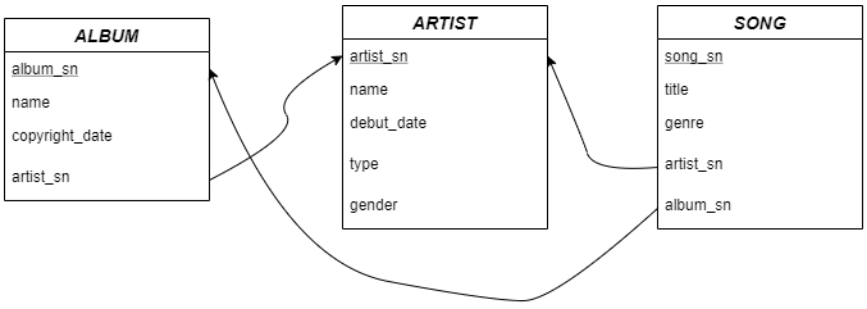
Subject :	Database
Prof :	Min Soo Lee
Major :	Computer Science Engineering
Student No :	1871026
Name :	Inryu Shin
Date :	2020.06.18

Contents

1. ER Diagram



2. Database schema diagram



3. Class and method (Javadoc)

A. List of Packages

Packages

Package	Description
DataAccess	
DataTransfer	
Process	

B. Package DataAccess

- Class and method

Package DataAccess

Class Summary

Class	Description
Album_SongDAO	Data Access Object Classes that connect to a database and perform operations such as input , modification, deletion, or query
AlbumDAO	Data Access Object Classes that connect to a database and perform operations such as input , modification, deletion, or query
ArtistDAO	Data Access Object Classes that connect to a database and perform operations such as input , modification, deletion, or query
SongDAO	Data Access Object Classes that connect to a database and perform operations such as input , modification, deletion, or query

Package DataAccess

Class Album_SongDAO

java.lang.Object
  DataAccess.Album_SongDAO

public class Album_SongDAO
extends java.lang.Object

Data Access Object Classes that connect to a database and perform operations such as input , modification, deletion, or query

Author:
Inryu Shin

Constructor Summary

Constructors

Constructor	Description
Album_SongDAO()	default constructor

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
void	dbClose()	Disconnect with the DB.
java.util.List<Album_SongDTO>	getVIEW()	Execute select query using join that is view query of the DB so that use can see that view table in console.
java.util.List<Album_SongDTO>	selectGenre (java.lang.String genre)	Take genre as a parameter and select tuple from view of that genre.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Package DataAccess

Class AlbumDAO

java.lang.Object
    DataAccess.AlbumDAO

public class AlbumDAO
extends java.lang.Object

Data Access Object Classes that connect to a database and perform operations such as input , modification, deletion, or query

Author:
Inryu Shin

Constructor Summary

Constructors

Constructor	Description
AlbumDAO()	default constructor

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
void	dbClose()	Disconnect with the DB.
boolean	deleteAlbum(int sn)	Take album_sn as parameter and delete Album table tuple of that album_sn.
AlbumDTO	getAlbum(int sn)	Take the album_sn as a parameter and select all tuple from album of that album_sn.
java.util.List<AlbumDTO>	getAlbumList()	Select all from Album
boolean	insertAlbum(AlbumDTO album)	Take AlbumDTO as a parameter and execute insert query.
int	IsExists (java.lang.String album_name)	Take the album_name of the album as a parameter and check if the data with that name exists on the album table.
int	selectSN (java.lang.String album_name)	Take the album_name as a parameter and select album_sn from album of that album name.

Package DataAccess

Class ArtistDAO

java.lang.Object
    DataAccess.ArtistDAO

public class ArtistDAO
extends java.lang.Object

Data Access Object Classes that connect to a database and perform operations such as input , modification, deletion, or query

Author:
Inryu Shin

Constructor Summary

Constructors

Constructor	Description
ArtistDAO()	default constructor

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	dbClose()	Disconnect with the DB.
boolean	deleteArtist(int sn)	Take artist_sn as parameter and delete Artist table tuple of that artitst_sn.
ArtistDTO	getArtist(int sn)	Take the artist_sn as a parameter and select all tuple from artist of that artist's sn.
java.util.List<ArtistDTO>	getArtistList()	Select all from Artist.
java.util.List<ArtistDTO>	gettypeArtist(java.lang.String type)	Take the type as a parameter and select all from artist of that type.
boolean	insertArtist(ArtistDTO artist)	Take ArtistDTO as a parameter and execute insert query.
int	IsExists(java.lang.String artist_name)	Take the name of the artist as a parameter and check if the data with that name exists on the artist table.
int	selectSN(java.lang.String artist_name)	Take the artist_name as a parameter and select artist_sn from artist of that artist's name.
boolean	updateArtist(ArtistDTO dto, int current_artist_sn)	Take ArtistDTO as parameter and update Artist table based on ArtistDTO(parameter)'s member and current_artist_sn
void	updateTransaction(int current_artist_sn, int updated_artist_sn)	Update artist_sn in Artist and Song table by using transaction.

Package DataAccess

Class SongDAO

java.lang.Object
  DataAccess.SongDAO

public class SongDAO
extends java.lang.Object

Data Access Object Classes that connect to a database and perform operations such as input , modification, deletion, or query

Author:
Inryu Shin

Constructor Summary

Constructors

Constructor	Description
SongDAO()	default constructor

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	dbClose()	Disconnect with the DB.
boolean	deleteSong(int sn)	Take song_sn as parameter and delete Song table tuple of that song_sn.
java.util.List<SongDTO>	getjoinSongList(java.lang.String artist_name, java.lang.String album_name)	Take the artist_name and album_name as a parameter and select title,genre from Song with tha artist_name and album_name using join
SongDTO	getSong(int sn)	Take the song_sn as a parameter and select all tuple from Song of that song_sn.
java.util.List<SongDTO>	getSongList()	Select all from Song
boolean	insertSong(SongDTO song)	Take SongDTO as a parameter and execute insert query.
int	IsExists(java.lang.String song_title)	Take the song_title of the Song as a parameter and check if the data with that title exists on the Song table.

C. Package DataTransfer

- Class and method

Package DataTransfer

Class Summary

Class	Description
Album_SongDTO	Object of Album join Song table data transfer (DTO) Columns in the table are treated as member variables.
AlbumDTO	Object of Album table data transfer (DTO) Columns in the table are treated as member variables.
ArtistDTO	Object of Artist table data transfer (DTO) Columns in the table are treated as member variables.
SongDTO	Object of Song table data transfer (DTO) Columns in the table are treated as member variables.

Package DataTransfer

Class Album_SongDTO

java.lang.Object
  DataTransfer.Album_SongDTO

public class Album_SongDTO
extends java.lang.Object

Object of Album join Song table data transfer (DTO) Columns in the table are treated as member variables.

Author:
Inryu Shin

Constructor Summary

Constructors

Constructor	Description
Album_SongDTO()	default_constructor
Album_SongDTO(java.lang.String album_name, java.lang.String title, java.lang.String genre)	constructor

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
java.lang.String	toString()	

Package DataTransfer

Class AlbumDTO

java.lang.Object
    DataTransfer.AlbumDTO

public class AlbumDTO
extends java.lang.Object

Object of Album table data transfer (DTO) Columns in the table are treated as member variables.

Author:
Inryu Shin

Constructor Summary

Constructors

Constructor	Description
AlbumDTO()	default constructor
AlbumDTO(int album_sn, java.lang.String name, java.lang.String release_date, int artist_sn)	constructor
AlbumDTO(java.lang.String name, java.lang.String release_date, int artist_sn)	constructor

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
int	getAlbum_sn()	
int	getArtist_sn()	
java.lang.String	getName()	
java.lang.String	getRelease_date()	
void	printInfo()	print this class based on my consol format style.
void	setAlbum_sn(int album_sn)	set album_sn
void	setArtist_sn(int artist_sn)	set artist_sn
void	setName(java.lang.String name)	set name
void	setRelease_date(java.lang.String release_date)	set release_date
java.lang.String	toString()	

Package DataTransfer

Class ArtistDTO

java.lang.Object
  DataTransfer.ArtistDTO

public class ArtistDTO
extends java.lang.Object

Object of Artist table data transfer (DTO) Columns in the table are treated as member variables.

Author:
Inryu Shin

Constructor Summary

Constructors

Constructor	Description
ArtistDTO()	default constructor
ArtistDTO(int artist_sn, java.lang.String name, java.lang.String debut_date, java.lang.String type, java.lang.String gender)	constructor
ArtistDTO(java.lang.String name, java.lang.String debut_date, java.lang.String type, java.lang.String gender)	constructor

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
int	getArtist_sn()	
java.lang.String	getDebut_date()	
java.lang.String	getGender()	
java.lang.String	getName()	
java.lang.String	getType()	
void	printInfo()	print this class based on my consol format style.
void	setArtist_sn(int artist_sn)	set artist_sn
void	setDebut_date(java.lang.String debut_date)	set debut_date
void	setGender(java.lang.String gender)	set gender
void	setName(java.lang.String name)	set name
void	setType(java.lang.String type)	set type
java.lang.String	toString()	

D. Package Process

- Class and method

Package Process

Class Summary

Class	Description
Album_SongProc	Create this object in the main and use this class's method.
AlbumProc	Create this object in the main and use this class's method.
ArtistProc	Create this object in the main and use this class's method.
SongProc	Create this object in the main and use this class's method.

Package Process

Class Album_SongProc

java.lang.Object
    Process.Album_SongProc

public class Album_SongProc
extends java.lang.Object

Create this object in the main and use this class's method. In this class, After receiving user input within the method, perform the query using the DAO object.

Author:
Inryu Shin

Field Summary

Fields

Modifier and Type	Field	Description
static java.util.Scanner	input	

Constructor Summary

Constructors

Constructor	Description
Album_SongProc()	default constructor

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
void	showbyGenre(java.lang.String genre)	Call the DAO object's selectGenre function into List Album_SongDTO list and then print that list.
void	showView()	Call the DAO object's getView function into List Album_SongDTO list and then print that list.

Package Process

Class AlbumProc

java.lang.Object
    Process.AlbumProc

public class AlbumProc
extends java.lang.Object

Create this object in the main and use this class's method. In this class, After receiving user input within the method, perform the query using the DAO object.

Author:
Inryu Shin

Field Summary

Fields

Modifier and Type	Field	Description
static java.util.Scanner	input	

Constructor Summary

Constructors

Constructor	Description
AlbumProc()	default constructor

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
void	deleteAlbum()	Take input from user of album_sn and call the DAO object's deleteAlbum(sn) function using input value.
void	insertAlbum(java.lang.String album_name, int artist_sn)	Take album_name, artist_sn as parameter and take input from user of release_date.
int	IsExists(java.lang.String album_name)	Take album_name as parameter and and call the DAO object's IsExists function.
int	selectSN(java.lang.String album_name)	Take album_name as parameter and call the DAO object's selectSN function.
void	showAlbumList()	Call the DAO object's getAlbumList function into List list and then print that list.

Package Process

Class ArtistProc

java.lang.Object
  Process.ArtistProc

public class ArtistProc
extends java.lang.Object

Create this object in the main and use this class's method. In this class, After receiving user input within the method, perform the query using the DAO object.

Author:
Inryu Shin

Field Summary

Fields

Modifier and Type	Field	Description
static java.util.Scanner	input	

Constructor Summary

Constructors

Constructor	Description
ArtistProc()	default constructor

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	deleteArtist()	Take input from user of artist_sn and call the DAO object's deleteArtist(sn) function using input value.
void	insertArtist(java.lang.String artist_name)	Take artist_name as parameter and take input from user of debut date, type, gender.
int	IsExists(java.lang.String artist_name)	Take artist_name as parameter and and call the DAO object's IsExists function.
int	selectSN(java.lang.String artist_name)	Take artist_name as parameter and call the DAO object's selectSN function.
void	showAritstList()	Call the DAO object's getArtistsList function into List list and then print that list.
void	typeAritstList(java.lang.String type)	Using type parameter , Call the DAO object's gettypeAtistsList function into List list and then print that list.
void	updateArtist()	Take input from user and create DTO object using those input values and call the DAO object's updateArtist function using DTO.
void	updateTransaction(int current_artist_sn, int updated_artist_sn)	Call the DAO object's updateTransaction function by using parameter value.

Methods inherited from class java.lang.Object

Package Process

Class SongProc

java.lang.Object
  Process.SongProc

public class SongProc
extends java.lang.Object

Create this object in the main and use this class's method. In this class, After receiving user input within the method, perform the query using the DAO object.

Author:
Inryu Shin

Field Summary

Fields

Modifier and Type	Field	Description
static java.util.Scanner	input	

Constructor Summary

Constructors

Constructor	Description
SongProc()	default constructor

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	deleteSong()	Take input from user of song_sn and call the DAO object's deleteSong(sn) function using input value.
void	insertSong(java.lang.String song_title, int album_sn, int artist_sn)	Take song_title, album_sn, artist_sn as parameter and take input from user of genre.
int	IsExists(java.lang.String song_title)	Take song_title as parameter and and call the DAO object's IsExists function.
void	showjoinSongList(java.lang.String artist_name, java.lang.String album_name)	Call the DAO object's getjoinSongList function into Listlist by using parameter and then print that list.
void	showSongList()	Call the DAO object's getSongList function into Listlist and then print that list.

Package DataTransfer

Class SongDTO

java.lang.Object
  DataTransfer.SongDTO

public class SongDTO
extends java.lang.Object

Object of Song table data transfer (DTO) Columns in the table are treated as member variables.

Author:
Inryu Shin

Constructor Summary

Constructors

Constructor	Description
SongDTO()	default constructor
SongDTO(int song_sn, java.lang.String title, java.lang.String genre, int album_sn, int artist_sn)	constuctor
SongDTO(java.lang.String title, java.lang.String genre)	constructor
SongDTO(java.lang.String title, java.lang.String genre, int album_sn, int artist_sn)	constructor

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
int	getAlbum_sn()	
int	getArtist_sn()	
java.lang.String	getGenre()	
int	getSong_sn()	
java.lang.String	getTitle()	
void	printDTO()	print this class based on my consol format style.
void	printInfo()	print this class based on my consol format style.
void	setAlbum_sn(int album_sn)	set album_sn
void	setArtist_sn(int artist_sn)	set artist_sn
void	setGenre(java.lang.String genre)	set genre
void	setSong_sn(int song_sn)	set song_sn
void	setTitle(java.lang.String title)	set title
java.lang.String	toString()	

4. How to run code

- **Main class name**

  (default package) / **Main.java**

- **connection configuration**

host and port name	Localhost:3306
database name	dbprj
user id	dbuser
password	dbpwd

5. Detail about 16 requirements

(1) Should have at least 3 tables with each table having at least 3 columns

(2) Should have at least 30 records inserted for initialization (total records for all tables)

artist

```
mysql> select * from artist;
+-----------+------------+------------+-------+--------+
| artist_sn | name       | debut_date | type  | gender |
+-----------+------------+------------+-------+--------+
|         1 | IU         | 2008-09-18 | solo  | female |
|         2 | OH MY GIRL | 2015-04-20 | group | female |
|         3 | AKMU       | 2014-04-07 | group | mixed  |
|         4 | Paul Kim   | 2014-01-21 | solo  | male   |
|         5 | Colde      | 2016-09-21 | solo  | male   |
|         6 | George     | 2016-03-16 | solo  | male   |
|         7 | JeA        | 2006-03-02 | solo  | female |
+-----------+------------+------------+-------+--------+
7 rows in set (0.00 sec)
```

album

```
mysql> select * from album;
+----------+--------------------+--------------+-----------+
| album_sn | name               | release_date | artist_sn |
+----------+--------------------+--------------+-----------+
|        1 | Love poem          | 2019-11-18   |         1 |
|        2 | Palette            | 2017-04-21   |         1 |
|        3 | CLOSER             | 2015-10-08   |         2 |
|        4 | WINDY DAY          | 2016-05-26   |         2 |
|        5 | PLAY               | 2014-04-07   |         3 |
|        6 | SUMMER EPISODE     | 2017-07-20   |         3 |
|        7 | Rain               | 2016-06-21   |         4 |
|        8 | Your Dog Loves You | 2018-03-28   |         5 |
|        9 | Love part1         | 2019-05-31   |         5 |
|       10 | cassette           | 2018-07-06   |         6 |
|       11 | LEEEE              | 2019-10-03   |         6 |
|       12 | Greedyy            | 2020-06-12   |         7 |
+----------+--------------------+--------------+-----------+
12 rows in set (0.00 sec)
```

song

```
mysql> select * from song;
+---------+--------------------+------------+-----------+----------+
| song_sn | title              | genre      | artist_sn | album_sn |
+---------+--------------------+------------+-----------+----------+
|       1 | Blueming           | rock/metal |         1 |        1 |
|       2 | Lullaby            | balad      |         1 |        1 |
|       3 | Black Out          | dance      |         1 |        2 |
|       4 | Dear Name          | balad      |         1 |        2 |
|       5 | CLOSER             | dance      |         2 |        3 |
|       6 | WINDY DAY          | dance      |         2 |        4 |
|       7 | LIAR LIAR          | dance      |         2 |        4 |
|       8 | Give Love          | fork       |         3 |        5 |
|       9 | Galaxy             | fork       |         3 |        5 |
|      10 | DINOSAUR           | eletronic  |         3 |        6 |
|      11 | Rain               | R&B        |         4 |        7 |
|      12 | Your Dog Loves You | R&B        |         5 |        8 |
|      13 | I fxxking love you | R&B        |         5 |        9 |
|      14 | Love is a Flower   | R&B        |         5 |        9 |
|      15 | let's go picnic    | R&B        |         6 |       10 |
|      16 | idkyet             | R&B        |         6 |       11 |
|      17 | Greedyy            | dance      |         7 |       12 |
+---------+--------------------+------------+-----------+----------+
17 rows in set (0.00 sec)
```

(3) Should include primary key, foreign key, not null constraints in each table

artist

```
mysql> describe artist;
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| artist_sn  | int         | NO   | PRI | NULL    | auto_increment |
| name       | varchar(20) | NO   |     | NULL    |                |
| debut_date | date        | NO   |     | NULL    |                |
| type       | varchar(10) | NO   |     | NULL    |                |
| gender     | varchar(10) | NO   |     | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

album

```
mysql> describe album;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| album_sn     | int         | NO   | PRI | NULL    | auto_increment |
| name         | varchar(20) | NO   |     | NULL    |                |
| release_date | date        | NO   |     | NULL    |                |
| artist_sn    | int         | NO   | MUL | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

song

```
mysql> describe song;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| song_sn   | int         | NO   | PRI | NULL    | auto_increment |
| title     | varchar(20) | NO   |     | NULL    |                |
| genre     | varchar(20) | NO   | MUL | NULL    |                |
| artist_sn | int         | NO   | MUL | NULL    |                |
| album_sn  | int         | NO   | MUL | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

- **MUL** in key column means **foreign key**.

(4) Tables should be in 3rd Normal Form (3NF)

ARTIST

column name	type	length	explanation	Null	key
artist_sn	int	5	아티스트 순번 (AUTO_INC)	X	PK
name	varchar	20	아티스트 이름	X	
debut_date	date		데뷔일	X	
type	varchar	10	타입 (solo / group)	X	
gender	varchar	10	성별 (female / male / mixed)	X	

ALBUM

column name	type	length	explanation	Null	key
album_sn	int	5	앨범 순번 (AUTO_INC)	X	PK
name	varchar	20	앨범 이름	X	
release_date	date		앨범 발매일	X	
artist_sn	int	5	아티스트 순번	X	FK(ARTIST)

SONG

column name	type	length	explanation	Null	key
song_sn	int	5	곡 순번 (AUTO_INC)	X	PK
title	varchar	20	곡 제목	X	
genre	varchar	20	곡 장르	X	
album_sn	int	5	앨범 순번	X	FK (ALBUMS)
artist_sn	int	5	아티스트 순번	X	FK(ARTIST)

- **1NF**

  The value of above table's each attribute contains only a single value from that domain. **(satisfied)**

- **2NF**

  It does not have any non-prime attribute that is functionally dependent on any proper subset of any candidate key of the relation. A non-prime attribute of a relation is an attribute that is not a part of any candidate key of the relation. **(satisfied)**

- **3NF**

  Every non-prime attribute of all above tables (artist, album, song) is non-transitively dependent on every key of tables. **(satisfied)**

(5) At least 1 index should be defined on the tables

createdb.sql

```
…

CREATE INDEX genre_index ON SONG(genre);

…
```

(6) 1 view should be defined, and the view should be defined using at least two other tables

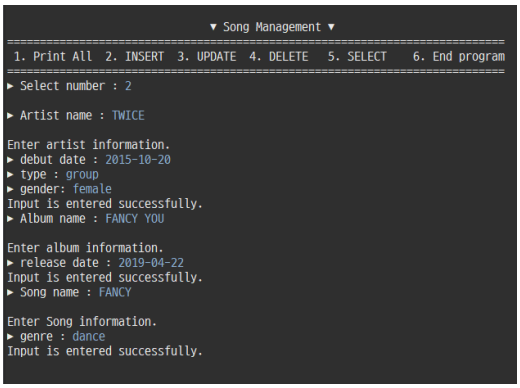
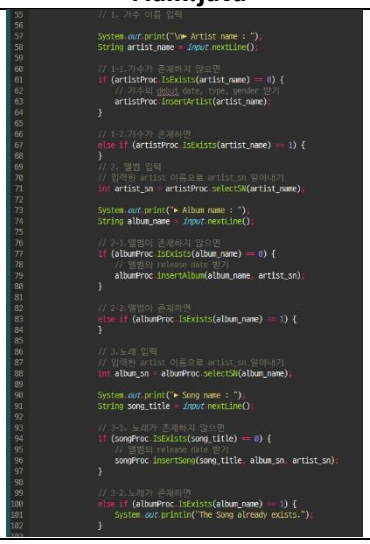
createdb.sql

```
CREATE VIEW album_song(album_name, title, genre) AS
    SELECT name, title, genre
    FROM ALBUM, SONG
    WHERE ALBUM.artist_sn=SONG.artist_sn AND ALBUM.album_sn=SONG.album_sn;
```

(7) All queries (in 8 to 14 below) should have parameterized variables.

> It is described below (8) to (14). **All query operations are hadled in ___DAO.java.**

(8) Should have at least 1 interface (menu and user input) and query to insert into 1 table.

Console	Main.java
	

```
▼ Song Management ▼
========================================================================
 1. Print All  2. INSERT  3. UPDATE  4. DELETE   5. SELECT    6. End program
========================================================================
▶ Select number : 2

▶ Artist name : TWICE

Enter artist information.
▶ debut date : 2015-10-20
▶ type : group
▶ gender: female
Input is entered successfully.
▶ Album name : FANCY YOU

Enter album information.
▶ release date : 2019-04-22
Input is entered successfully.
▶ Song name : FANCY

Enter Song information.
▶ genre : dance
Input is entered successfully.
```

```
            // 1. 가수 이름 입력
            System.out.print("\n▶ Artist name : ");
            String artist_name = input.nextLine();

            // 1-1.가수가 존재하지 않으면
            if (artistProc.IsExists(artist_name) == 0) {
                // 가수의 debut date, type, gender 받기
                artistProc.InsertArtist(artist_name);
            }

            // 1-2.가수가 존재하면
            else if (artistProc.IsExists(artist_name) == 1) {
            }
            // 2. 앨범 입력
            // 입력한 artist 이름으로 artist_sn 알아내기
            int artist_sn = artistProc.selectSN(artist_name);

            System.out.print("▶ Album name : ");
            String album_name = input.nextLine();

            // 2-1.앨범이 존재하지 않으면
            if (albumProc.IsExists(album_name) == 0) {
                // 앨범의 release date 받기
                albumProc.InsertAlbum(album_name, artist_sn);
            }

            // 2-2.앨범이 존재하면
            else if (albumProc.IsExists(album_name) == 1) {
            }

            // 3.노래 입력
            // 입력한 artist 이름으로 artist_sn 알아내기
            int album_sn = albumProc.selectSN(album_name);

            System.out.print("▶ Song name : ");
            String song_title = input.nextLine();

            // 3-1. 노래가 존재하지 않으면
            if (songProc.IsExists(song_title) == 0) {
                // 앨범의 release date 받기
                songProc.insertSong(song_title, album_sn, artist_sn);
            }

            // 3-2.노래가 존재하면
            else if (albumProc.IsExists(album_name) == 1) {
                System.out.println("The song already exists.");
            }
```

ArtistProc.java	ArtistDAO.java
	

```
    /**
     * Take artist_name as parameter and take input from user of debut date, type,
     * gender. Create DTO object using those values and call the DAO object's
     * insertArtist function using DTO.
     *
     * @param artist_name
     */
    public void insertArtist(String artist_name) { // 인자로 이름을 받음

        System.out.println("\nEnter artist information.");

        // System.out.print("name : ");
        // String name = input.nextLine();
        System.out.print("▶ debut date : ");
        String debut_date = input.nextLine();
        System.out.print("▶ type : ");
        String type = input.nextLine();
        System.out.print("▶ gender: ");
        String gender = input.nextLine();

        // ArtistDTO 객체 생성 (입력 받은 값과 생성자 이용)
        ArtistDTO artist = new ArtistDTO(artist_name, debut_date, type, gender);

        boolean r = dao.insertArtist(artist); // 입력받은 데이터 추가

        if (r) {
            System.out.println("Input is entered successfully.");
        } else {
            System.out.println("Failed. Input is not entered");
        }

    }
```

Take artist name as parameter.

Take input from user.

Create ArtistDTO object by using parameter and input values.

Call dao function using ArtistDTO created just before.

```
    /**
     * Take ArtistDTO as a parameter and execute insert query.
     *
     * @param ArtistDTO artist
     * @return boolean
     */
    public boolean insertArtist(ArtistDTO artist) {

        boolean result = false;

        try {
            getConnection();

            String query = "INSERT INTO Artist (name, debut_date, type, gender) VALUES(?,?,?,?)";

            PreparedStatement pstmt = conn.prepareStatement(query);

            pstmt.setString(1, artist.getName());
            pstmt.setString(2, artist.getDebut_date());
            pstmt.setString(3, artist.getType());
            pstmt.setString(4, artist.getGender());

            int r = pstmt.executeUpdate();

            if (r > 0)
                result = true;

        } catch (Exception e) {
            System.out.println("Exception : insertArtist " + e.getMessage());
        } finally {
            dbClose();
        }
        return result;
    }
```

Take ArtistDTO as parameter.

This is insert querry.

Set preparedstatement by using ArtistDTO that is parameter.

Execute update.

(9) Should have at least 1 interface (menu and user input) and query to update on 1 or 2 tables

console

```
▼ Song Management ▼
================================================================================
 1. Print All  2. INSERT  3. UPDATE  4. DELETE  5. SELECT  6. End program
================================================================================
▶ Select number : 3


================================================================================
            1. Artist                                2. Album
================================================================================
▶ Which table do you want to handle? : 1

                                <Artist List>
================================================================================
artist_sn       name          debut_date      type          gender
================================================================================
       1        IU            2008-09-18      solo          female
       2        OH MY GIRL    2015-04-20      group         female
       3        AKMU          2014-04-07      group         mixed
       4        Paul Kim      2014-01-21      solo          male
       5        Colde         2016-09-21      solo          male
       6        George        2016-03-16      solo          male
       7        JeA           2006-03-02      solo          female
       8        TWICE         2015-10-20      group         female
================================================================================
▶ Enter artist_sn you want to update : 7

================================================================================
artist_sn       name          debut_date      type          gender
================================================================================
       7        JeA           2006-03-02      solo          female
================================================================================
▶ artist_sn : 7
▶ name : JeA
▶ debut date : 2020-05-20
▶ type : solo
▶ gender : female
The artist's information has been modified as follows.
================================================================================
artist_sn       name          debut_date      type          gender
================================================================================
       7        JeA           2020-05-20      solo          female
================================================================================
```

ArtistProc.java

```
public void updateArtist() {

    System.out.print("▶ Enter artist_sn you want to update : ");
    int current_artist_sn = input.nextInt();
    input.nextLine();

    ArtistDTO dto = dao.getArtist(current_artist_sn);

    if (dto != null) {

        System.out.println("");
        dto.printInfo();

        System.out.print("▶ artist_sn : ");
        int artist_sn = input.nextInt();
        input.nextLine();
        System.out.print("▶ name : ");
        String name = input.nextLine();
        System.out.print("▶ debut date : ");
        String debut_date = input.nextLine();
        System.out.print("▶ type : ");
        String type = input.nextLine();
        System.out.print("▶ gender : ");
        String gender = input.nextLine();

        dto = new ArtistDTO(artist_sn, name, debut_date, type, gender);

        boolean r = dao.updateArtist(dto, current_artist_sn);
```

Take input from user.

Create ArtistDTO object by using input values.

Call dao function using ArtistDTO as parameter created just before.

ArtistDAO.java

```
public boolean updateArtist(ArtistDTO dto, int current_artist_sn) {

    boolean result = false;
    try {
        getConnection();

        String sql = "UPDATE Artist SET artist_sn=?,name=?,debut_date=?,type=?,gender=? WHERE artist_sn=?";
        PreparedStatement pstmt = conn.prepareStatement(sql);

        pstmt.setInt(1, dto.getArtist_sn());
        pstmt.setString(2, dto.getName());
        pstmt.setString(3, dto.getDebut_date());
        pstmt.setString(4, dto.getType());
        pstmt.setString(5, dto.getGender());
        pstmt.setInt(6, current_artist_sn);

        int r = pstmt.executeUpdate();

        if (r > 0)
            result = true;

    } catch (Exception e) {
        System.out.println("Exception :updateArtist  " + e.getMessage());
    } finally {
        dbClose();
    }
    return result;
}
```

Take ArtistDTO and current_artist_sn as parameter.

This is update query

Update artist whose artist_sn is current_artist_sn to new artist_sn by using ArtistDTO (parameter)

(10) One of the updates should occur on 2 tables by using transactions

console

```
                    ▼ Song Management ▼
=============================================================
 1. Print All  2. INSERT  3. UPDATE  4. DELETE  5. SELECT  6. End program
=============================================================
▶ Select number : 3


=============================================================
          1. Artist                          2. Album
=============================================================
▶ Which table do you want to handle? : 2


                        <Album List>
=============================================================
album_sn        name              release_date    artist_sn
=============================================================
       1        Love poem         2019-11-18            1
       2        Palette           2017-04-21            1
       3        CLOSER            2015-10-08            2
       4        WINDY DAY         2016-05-26            2
       5        PLAY              2014-04-07            3
       6        SUMMER EPISODE    2017-07-20            3
       7        Rain              2016-06-21            4
       8        Your Dog Loves You 2018-03-28           5
       9        Love part1        2019-05-31            5
      10        cassette          2018-07-06            6
      11        LEEEE             2019-10-03            6
      12        Greedyy           2020-06-12            7
      13        FANCY YOU         2019-04-22            8
=============================================================
▶ Enter the artist_sn you want to update : 1
▶ What value would you like to replace artist_sn? : 9
```

```
                        <Album List>
=============================================================
album_sn        name              release_date    artist_sn
=============================================================
       1        Love poem         2019-11-18            9
       2        Palette           2017-04-21            9
       3        CLOSER            2015-10-08            2
       4        WINDY DAY         2016-05-26            2
       5        PLAY              2014-04-07            3
       6        SUMMER EPISODE    2017-07-20            3
       7        Rain              2016-06-21            4
       8        Your Dog Loves You 2018-03-28           5
       9        Love part1        2019-05-31            5
      10        cassette          2018-07-06            6
      11        LEEEE             2019-10-03            6
      12        Greedyy           2020-06-12            7
      13        FANCY YOU         2019-04-22            8
=============================================================
                        <Artist List>
=============================================================
artist_sn       name        debut_date      type        gender
=============================================================
       2        OH MY GIRL  2015-04-20      group       female
       3        AKMU        2014-04-07      group       mixed
       4        Paul Kim    2014-01-21      solo        male
       5        Colde       2016-09-21      solo        male
       6        George      2016-03-16      solo        male
       7        JeA         2020-05-20      solo        female
       8        TWICE       2015-10-20      group       female
       9        IU          2008-09-18      solo        female
=============================================================
```

ArtistDAO.java

```
    public void updateTransaction(int current_artist_sn, int updated_artist_sn) throws SQLException {

        try {
            getConnection();
            conn.setAutoCommit(false);

            String sql = "UPDATE Artist SET artist_sn=? WHERE artist_sn=?";
            PreparedStatement pstmt = conn.prepareStatement(sql);

            pstmt.setInt(1, updated_artist_sn);
            pstmt.setInt(2, current_artist_sn);
            pstmt.executeUpdate();

            sql = "UPDATE Song SET artist_sn=? WHERE artist_sn=?";

            pstmt.setInt(1, updated_artist_sn);
            pstmt.setInt(2, current_artist_sn);
            pstmt.executeUpdate();

            conn.commit();

        } catch (Exception e) {
            System.out.println("Exception : updateTransaction " + e.getMessage());
        } finally {
            conn.setAutoCommit(true);
            dbClose();
        }
    }
```

Take current_artist_sn and updated_artist_sn as values.

This update require two tables(Artist, Song) and they are related by FK. So I should use transaction.

Update artist whose sn is current_artist_sn to updated_artist_sn.

Update song whose sn is current_artist_sn to updated_artist_sn.

(11) Should have at least 1 interface (menu and user input) and queries to delete from 1 table

console

```
                    ▼ Song Management ▼
================================================================
 1. Print All  2. INSERT  3. UPDATE  4. DELETE  5. SELECT  6. End program
================================================================
▶ Select number : 4


================================================================
 1. Artist                  2. Album                   3. Song
================================================================
▶ Which table do you want to handle? : 1

                          <Artist List>
================================================================
artist_sn       name          debut_date     type          gender
================================================================
        2       OH MY GIRL    2015-04-20     group         female
        3       AKMU          2014-04-07     group         mixed
        4       Paul Kim      2014-01-21     solo          male
        5       Colde         2016-09-21     solo          male
        6       George        2016-03-16     solo          male
        7       JeA           2020-05-20     solo          female
        8       TWICE         2015-10-20     group         female
        9       IU            2008-09-18     solo          female
================================================================
▶ Enter artist_sn of Artist you want to delete :  8


================================================================
artist_sn       name          debut_date     type          gender
================================================================
        8       TWICE         2015-10-20     group         female
================================================================
▶ Are you sure you want to delete it?(Y/N) : y
Aritst [ 8 ] 's information has been deleted successfully.
```

ArtistProc.java

```java
/**
 * Take artist_sn as parameter and delete Artist table tuple of that artitst_sn.
 */
public boolean deleteArtist(int sn) {
    boolean result = false;
    try {
        getConnection();

        String sql = "DELETE FROM Artist WHERE artist_sn = ?";
        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, sn);
        int r = pstmt.executeUpdate();

        if (r > 0)
            result = true;

    } catch (Exception e) {
        System.out.println("Exception :deleteArtist " + e.getMessage());
    } finally {
        dbClose();
    }
    return result;
}
```

Take artist_sn as parameter.

delete artist whose artist_sn is same as input.

ArtistDAO.java

```java
public void deleteArtist() {

    System.out.print("▶ Enter artist_sn of Artist you want to delete :  ");
    int sn = input.nextInt();
    System.out.println("\n\n");
    ArtistDTO dto = dao.getArtist(sn);
    if (dto != null) {
        dto.printInfo();

        System.out.print("▶ Are you sure you want to delete it?(Y/N) : ");
        input.nextLine();
        String ans = input.nextLine();
        if (ans.equalsIgnoreCase("y")) {
            boolean r = dao.deleteArtist(sn);

            if (r) {
                System.out.println("Aritst [ " + sn + " ] 's information has been deleted successfully.");
            } else {
                System.out.println("Failed. Aritst [ \"+sn+\" ] 's information has been not deleted.");
            }
        } else {
            System.out.println("Deletion operation canceled.");
        }
    } else {

        System.out.println("Enter correct artist_sn");
```

Take input from user. (artist_sn)

Call dao function using artist_sn value.

(12) Should have at least 1 interface (menu and user input) and queries to select from database.

console

```
                    ▼ Song Management ▼
================================================================================
 1. Print All  2. INSERT  3. UPDATE  4. DELETE  5. SELECT  6. End program
================================================================================
▶ Select number : 5


================================================================================
     1. Artist            2. Join Table          3. Using View
================================================================================
▶ Which table do you want to handle? : 1


================================================================================
                  1. group                  2. solo
================================================================================
▶ Select the type you want to see : 2


                          <Artist List>
================================================================================
artist_sn       name        debut_date     type          gender
================================================================================
       4        Paul Kim    2014-01-21     solo          male
       5        Colde       2016-09-21     solo          male
       6        George      2016-03-16     solo          male
       7        JeA         2020-05-20     solo          female
       9        IU          2008-09-18     solo          female
================================================================================
```

ArtistProc.java

```
public void typeAritstList(String type) {
    List<ArtistDTO> list = dao.gettypeArtist(type);
    System.out.println("                          <Artist List>");
    System.out.println("================================================================================");
    if (list != null && list.size() > 0) {
        System.out.println("artist_sn\t  name \t\tdebut_date\ttype\t\tgender");
        System.out.println("================================================================================");
        for (ArtistDTO dto : list) {
            System.out.println(dto);
        }
    } else {
        System.out.println("저장된 데이터가 없습니다. ");
    }
    System.out.println("================================================================================");
}
```

Take type as parameter.

Call DAO function using type as parameter. gettypeArtist returns List<ArtistDTO> list.

Print the ArtistDTO objects in the list.

ArtistDAO.java

```
public List<ArtistDTO> gettypeArtist(String type) {
    List<ArtistDTO> list = new ArrayList<ArtistDTO>();
    try {
        getConnection();
        String sql = "SELECT * FROM ARTIST WHERE type=?";
        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, type);
        ResultSet r = pstmt.executeQuery();
        while (r.next()) {
            int artist_sn = r.getInt("artist_sn");
            String name = r.getString("name");
            String debut_date = r.getString("debut_date");
            String type2 = r.getString("type");
            String gender = r.getString("gender");
            list.add(new ArtistDTO(artist_sn, name, debut_date, type2, gender));
        }
    } catch (Exception e) {
        System.out.println("Exception : gettypeArtist " + e.getMessage());
    } finally {
        dbClose();
    }
    return list;
}
```

Take type as parameter.

Create List <ArtistDTO> list.

Select artist whose type is parameter value.

Using query result, create ArtistDTO object and add it to the list just before created.

return list.

(13) Should have at least 1 interface (menu and user input) and queries to select using nested queries and join.

console

```
                    ▼ Song Management ▼
=============================================================
 1. Print All  2. INSERT  3. UPDATE  4. DELETE  5. SELECT  6. End program
=============================================================
▶ Select number : 5


=============================================================
     1. Artist          2. Join Table          3. Using View
=============================================================
▶ Which table do you want to handle? : 2


                        <Artist List>
=============================================================
artist_sn       name          debut_date      type          gender
=============================================================
       2        OH MY GIRL    2015-04-20      group         female
       3        AKMU          2014-04-07      group         mixed
       4        Paul Kim      2014-01-21      solo          male
       5        Colde         2016-09-21      solo          male
       6        George        2016-03-16      solo          male
       7        JeA           2020-05-20      solo          female
       9        IU            2008-09-18      solo          female
=============================================================

                        <Album List>
=============================================================
album_sn        name                release_date    artist_sn
=============================================================
       1        Love poem           2019-11-18          9
       2        Palette             2017-04-21          9
       3        CLOSER              2015-10-08          2
       4        WINDY DAY           2016-05-26          2
       5        PLAY                2014-04-07          3
       6        SUMMER EPISODE      2017-07-20          3
       7        Rain                2016-06-21          4
       8        Your Dog Loves You  2018-03-28          5
       9        Love part1          2019-05-31          5
      10        cassette            2018-07-06          6
      11        LEEEE               2019-10-03          6
      12        Greedyy             2020-06-12          7
=============================================================
If you choose Artist's name and Album name
then you can see song list in that Artist's Album
▶ Enter the Artist's name : IU
▶ Enter the Album name : Love poem
```

```
                        <Song List>
=============================================================
                title                           genre
=============================================================
              Blueming                      rock/metal
              Lullaby                         balad
=============================================================
```

ArtistProc.java

```
public void showjoinSongList(String artist_name, String album_name) {

    List<SongDTO> list = dao.getjoinSongList(artist_name, album_name);

    System.out.println("\n                                <Song List>");
    System.out.println("=============================================================");
    if (list != null && list.size() > 0) {
        System.out.println("\t\t\t title\t\t\t\t   genre\t");
        System.out.println("=============================================================");

        for (SongDTO dto : list) {
            dto.printDTO();
        }

    } else {
        System.out.println("no data. ");
    }
    System.out.println("=============================================================");
}
```

Take artist_name and album_name as parameter.

Call DAO function using parameter. getjoinSongList returns List<songDTO> list.

Print the SongDTO objects in the list.

ArtistDAO.java

```
public List<SongDTO> getjoinSongList(String artist_name, String album_name) {
    List<SongDTO> list = new ArrayList<SongDTO>();

    try {
        getConnection();

        String sql = "SELECT title, genre FROM song WHERE album_sn in (SELECT album_sn FROM artist,album WHERE artist.artist_sn=album.artist_sn AND artist.name=? AND album.name=?)";

        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, artist_name);
        pstmt.setString(2, album_name);
        ResultSet r = pstmt.executeQuery();

        while (r.next()) {

            String title = r.getString("title");
            String genre = r.getString("genre");

            list.add(new SongDTO(title, genre));
        }

    } catch (Exception e) {
        System.out.println("Exception : getjoinSongList " + e.getMessage());
    } finally {
        dbClose();
    }

    return list;
}
```

Take artist_name and album_name as parameter.

Create List <SongDTO> list.

Select songs those album_name and artist_name are same as parameter value.

Using query result, create songDTO object and add it to the list just before created.

return list.

(14) Should have at least 1 interface (menu and user input) and queries to select from view

Console

```
                    ▼ Song Management ▼
=============================================================
 1. Print All  2. INSERT  3. UPDATE  4. DELETE   5. SELECT    6. End program
=============================================================
► Select number : 5


=============================================================
      1. Artist              2. Join Table            3. Using View
=============================================================
► Which table do you want to handle? : 3

                              <VIEW>
=============================================================
   name                 title                genre
=============================================================
Love poem               Blueming             rock/metal
Love poem               Lullaby              balad
Palette                 Black Out            dance
Palette                 Dear Name            balad
CLOSER                  CLOSER               dance
WINDY DAY               WINDY DAY            dance
WINDY DAY               LIAR LIAR            dance
PLAY                    Give Love            fork
PLAY                    Galaxy               fork
SUMMER EPISODE          DINOSAUR             eletronic
Rain                    Rain                 R&B
Your Dog Loves You      Your Dog Loves You   R&B
Love part1              I fxxking love you   R&B
Love part1              Love is a Flower     R&B
cassette                let's go picnic      R&B
LEEEE                   idkyet               R&B
Greedyy                 Greedyy              dance
=============================================================
► Which genre do you want to search in ? : dance



=============================================================
   name                 title                genre
=============================================================
Palette                 Black Out            dance
CLOSER                  CLOSER               dance
WINDY DAY               WINDY DAY            dance
WINDY DAY               LIAR LIAR            dance
Greedyy                 Greedyy              dance
=============================================================
```

Album_SongProc.java

```
*/
public void showbyGenre(String genre) {
    List<Album_SongDTO> list = dao.selectGenre(genre);

    System.out.println("=============================================================");
    if (list != null && list.size() > 0) {
        System.out.println("   name\t\t\t  title \t\t\tgenre");
        System.out.println("=============================================================");

        for (Album_SongDTO dto : list) {
            System.out.println(dto);
        }

    } else {
        System.out.println("no data. ");
    }
    System.out.println("=============================================================");
}
```

Take genre and as parameter.

Call DAO function using parameter. selectGenre returns List<Album_SongDTO> list.

Print the Album_SongDTO objects in the list.

Album_SongDAO.java

```
*/
public List<Album_SongDTO> selectGenre(String genre) {
    List<Album_SongDTO> list = new ArrayList<Album_SongDTO>();

    try {
        getConnection();

        String sql = "SELECT album_name, title, genre FROM album_song WHERE genre=?";

        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, genre);
        ResultSet r = pstmt.executeQuery();

        while (r.next()) {
            String name = r.getString("album_name");
            String title = r.getString("title");
            String genre2 = r.getString("genre");
            list.add(new Album_SongDTO(name, title, genre2));
        }

    } catch (Exception e) {
        System.out.println("Exception : selectGenre " + e.getMessage());
    } finally {
        dbClose();
    }

    return list;
}
```

Take genre and as parameter.

Create List <Album_SongDTO> list.

Select songs those genre is same as parameter value.

Using query result, create Album_SongDTO object and add it to the list just before created.

return list.

(15) Should have interface (menu) to print out contents of all tables

console

```
▼ Song Management ▼
=============================================================
 1. Print All  2. INSERT  3. UPDATE  4. DELETE  5. SELECT  6. End program
=============================================================
▶ Select number : 1

                        <Artist List>
=============================================================
artist_sn       name        debut_date      type        gender
=============================================================
        2       OH MY GIRL  2015-04-20      group       female
        3       AKMU        2014-04-07      group       mixed
        4       Paul Kim    2014-01-21      solo        male
        5       Colde       2016-09-21      solo        male
        6       George      2016-03-16      solo        male
        7       JeA         2020-05-20      solo        female
        9       IU          2008-09-18      solo        female
=============================================================

                        <Album List>
=============================================================
album_sn        name                release_date    artist_sn
=============================================================
        1       Love poem           2019-11-18          9
        2       Palette             2017-04-21          9
        3       CLOSER              2015-10-08          2
        4       WINDY DAY           2016-05-26          2
        5       PLAY                2014-04-07          3
        6       SUMMER EPISODE      2017-07-20          3
        7       Rain                2016-06-21          4
        8       Your Dog Loves You  2018-03-28          5
        9       Love part1          2019-05-31          5
       10       cassette            2018-07-06          6
       11       LEEEE               2019-10-03          6
       12       Greedyy             2020-06-12          7
=============================================================
```

```
                        <Song List>
=============================================================
song_sn         title               genre       album_sn    artist_sn
=============================================================
        1       Blueming            rock/metal      1           9
        2       Lullaby             balad           1           9
        3       Black Out           dance           2           9
        4       Dear Name           balad           2           9
        5       CLOSER              dance           3           2
        6       WINDY DAY           dance           4           2
        7       LIAR LIAR           dance           4           2
        8       Give Love           fork            5           3
        9       Galaxy              fork            5           3
       10       DINOSAUR            eletronic       6           3
       11       Rain                R&B             7           4
       12       Your Dog Loves You  R&B             8           5
       13       I fxxking love you  R&B             9           5
       14       Love is a Flower    R&B             9           5
       15       let's go picnic     R&B             10          6
       16       idkyet              R&B             11          6
       17       Greedyy             dance           12          7
=============================================================
```

ArtistProc.java

```
public void showAritstList() {

    List<ArtistDTO> list = dao.getArtistList();

    System.out.println("                        <Artist List>");
    System.out.println("=============================================================");
    if (list != null && list.size() > 0) {
        System.out.println("artist_sn\t  name \t\tdebut_date\ttype\t\tgender");
        System.out.println("=============================================================");

        for (ArtistDTO dto : list) {
            System.out.println(dto);
        }

    } else {
        System.out.println("저장된 데이터가 없습니다. ");
    }
    System.out.println("=============================================================");
}
```

Call DAO function using parameter. getArtistList returns List<ArtistDTO> list.

Print the ArtistDTO objects in the list.

ArtistSongDAO.java

```
 */
public List<ArtistDTO> getArtistList() {
    List<ArtistDTO> list = new ArrayList<ArtistDTO>();

    try {
        getConnection();

        String sql = "SELECT * FROM ARTIST";

        PreparedStatement pstmt = conn.prepareStatement(sql);
        ResultSet r = pstmt.executeQuery();

        while (r.next()) {
            int artist_sn = r.getInt("artist_sn");
            String name = r.getString("name");
            String debut_date = r.getString("debut_date");
            String type = r.getString("type");
            String gender = r.getString("gender");
            list.add(new ArtistDTO(artist_sn, name, debut_date, type, gender));
        }

    } catch (Exception e) {
        System.out.println("Exception : getArtistList " + e.getMessage());
    } finally {
        dbClose();
    }

    return list;
}
```

Create List <ArtistDTO> list.

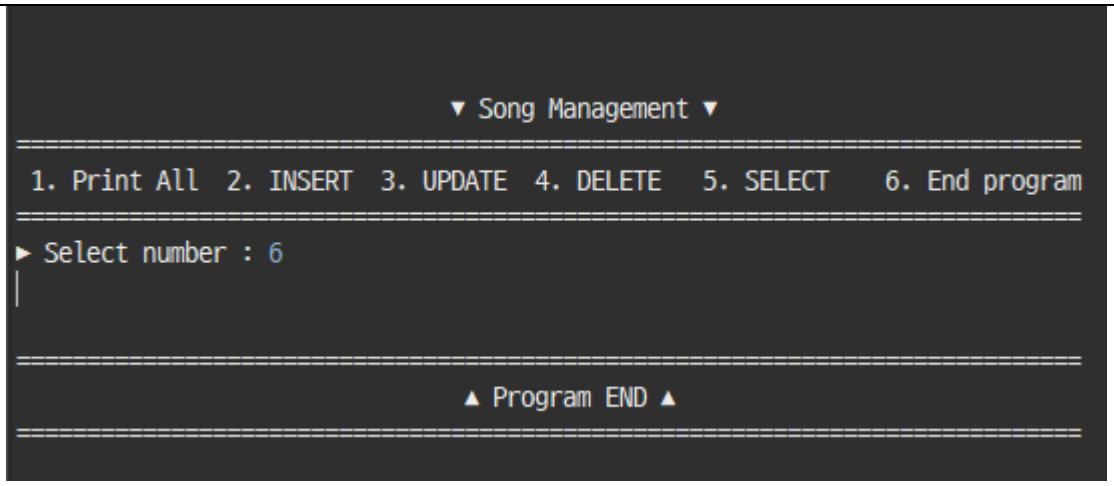
Select all artist.

Using query result, create ArtistDTO object and add it to the list just before created.

return list.

(16) Should have interface (menu) to finish program gracefully. Otherwise menu should repeatedly appear automatically.

Console



Main.java

```
*/
public class Main {

    static public Scanner input = new Scanner(System.in);

    public static void main(String[] args) throws SQLException {

        // Proc 객체 생성

        ArtistProc artistProc = new ArtistProc();
        AlbumProc albumProc = new AlbumProc();
        SongProc songProc = new SongProc();
        Album_SongProc album_songProc= new Album_SongProc();

        int num = 0;
        while (true) {
            System.out.println("\n\n\n                                    ▼ Song Management ▼");
            System.out.println("=============================================================================");
            System.out.println(" 1. Print All  2. INSERT  3. UPDATE  4. DELETE   5. SELECT    6. End program");
            System.out.println("=============================================================================");

            try {
                System.out.print("▶ Select number : ");
                num = input.nextInt();
                input.nextLine();
                if (!(num >= 1 && num <= 6)) {
                    throw new InputMismatchException();
                }

            } catch (InputMismatchException e) {
                System.out.println("Wrong number");
```

This "Song management" program is repeated in while statement.

Take user input.

Omitted

```
            } else if (num == 6) {
                System.out.println("\n\n=============================================================================");
                System.out.println("                                    ▲ Program END ▲");
                System.out.println("=============================================================================");
                break;

            }
```

When user input number '6', break the while.

6. SQL scripts

A. createdb.sql

```
/*create Artist, album, song table*/
CREATE TABLE ARTIST (
    artist_sn INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(20) NOT NULL,
    debut_date DATE NOT NULL,
    type VARCHAR(10) NOT NULL,
    gender VARCHAR(10) NOT NULL
);
CREATE TABLE ALBUM (
    album_sn INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(20) NOT NULL,
    release_date DATE NOT NULL,
    artist_sn INT NOT NULL,
    FOREIGN KEY (artist_sn)
            REFERENCES ARTIST(artist_sn)
            ON DELETE CASCADE
            ON UPDATE CASCADE
);
CREATE TABLE SONG (
    song_sn INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(20) NOT NULL,
    genre VARCHAR(20) NOT NULL,
    artist_sn INT NOT NULL,
    album_sn INT NOT NULL,
    FOREIGN KEY (artist_sn)
            REFERENCES ARTIST(artist_sn)
            ON DELETE CASCADE
            ON UPDATE CASCADE,
    FOREIGN KEY (album_sn)
            REFERENCES ALBUM(album_sn)
            ON DELETE CASCADE
            ON UPDATE CASCADE
);
/*insert valused into artist, album, song table*/
INSERT INTO ARTIST(name, debut_date, type, gender)
VALUES
('IU', '2008-09-18', 'solo', 'female'),
('OH MY GIRL', '2015-04-20', 'group', 'female'),
('AKMU', '2014-04-07', 'group', 'mixed'),
('Paul Kim', '2014-01-21', 'solo', 'male'),
('Colde', '2016-09-21', 'solo', 'male'),
('George', '2016-03-16', 'solo', 'male'),
('JeA', '2006-03-02', 'solo', 'female');
INSERT INTO ALBUM(name, release_date, artist_sn)
VALUES
('Love poem', '2019-11-18', 1),
('Palette', '2017-04-21', 1),
('CLOSER', '2015-10-08', 2),
('WINDY DAY', '2016-05-26', 2),
('PLAY', '2014-04-07', 3),
('SUMMER EPISODE', '2017-07-20', 3),
('Rain', '2016-06-21', 4),
('Your Dog Loves You', '2018-03-28', 5),
('Love part1', '2019-05-31', 5),
('cassette', '2018-07-06', 6),
('LEEEE', '2019-10-03', 6),
('Greedyy', '2020-06-12', 7);
INSERT INTO SONG(title, genre, album_sn, artist_sn)
VALUES
```

```
('Blueming', 'rock/metal', 1,1),
('Lullaby', 'balad', 1,1),
('Black Out', 'dance', 2,1),
('Dear Name', 'balad', 2,1),
('CLOSER', 'dance',3,2),
('WINDY DAY', 'dance',4,2),
('LIAR LIAR', 'dance',4,2),
('Give Love', 'fork', 5,3),
('Galaxy', 'fork', 5,3),
('DINOSAUR', 'eletronic',6 ,3),
('Rain', 'R\&B', 7,4),
('Your Dog Loves You', 'R\&B',8 ,5),
('I fxxking love you', 'R\&B',9 ,5),
('Love is a Flower', 'R\&B',9,5),
('let\'s go picnic', 'R\&B',10,6),
('idkyet', 'R\&B',11,6),
('Greedyy', 'dance',12,7);

/*create index*/
CREATE INDEX artistname_index ON ARTIST(name);
CREATE INDEX albumname_index ON ALBUM(name);
CREATE INDEX genre_index ON SONG(genre);

/*create view (join 2 tables)*/
CREATE VIEW album_song(album_name, title, genre) AS
    SELECT name, title, genre
    FROM ALBUM, SONG
    WHERE ALBUM.artist_sn=SONG.artist_sn AND ALBUM.album_sn=SONG.album_sn;
```

B. dropdb.sql

```
/* drop all tables and data from database (include view) */
DROP VIEW album_song;
DROP TABLE song;
DROP TABLE album;
DROP TABLE artist;
```

7. Java codes

Main.java

```
import java.sql.SQLException;
import java.util.InputMismatchException;
import java.util.Scanner;
import Process.AlbumProc;
import Process.Album_SongProc;
import Process.ArtistProc;
import Process.SongProc;
/**
 * Song Management program continues until user enter number 6.
 *
 * @author Inryu Shin
 *
 */
public class Main {
    static public Scanner input = new Scanner(System.in);
    public static void main(String[] args) throws SQLException {
            // Proc 객체 생성
            ArtistProc artistProc = new ArtistProc();
            AlbumProc albumProc = new AlbumProc();
            SongProc songProc = new SongProc();
            Album_SongProc album_songProc= new Album_SongProc();
            int num = 0;
            while (true) {
                    System.out.println("\n\n\n                                                    ▼ Song Management
▼");

    System.out.println("=========================================================================
======");
                    System.out.println(" 1. Print All   2. INSERT   3. UPDATE   4. DELETE    5.
SELECT    6. End program");

    System.out.println("=========================================================================
======");
                    try {
                            System.out.print("▶ Select number : ");
                            num = input.nextInt();
                            input.nextLine();
                            if (!(num >= 1 && num <= 6)) {
                                    throw new InputMismatchException();
                            }
                    } catch (InputMismatchException e) {
                            System.out.println("Wrong number");
                    }
                    if (num == 1) {
                            System.out.println("");
                            artistProc.showAritstList();
                            albumProc.showAlbumList();
                            songProc.showSongList();
                    } else if (num == 2) {
                            // 1. 가수 이름 입력
                            System.out.print("\n▶ Artist name : ");
                            String artist_name = input.nextLine();
                            // 1-1.가수가 존재하지 않으면
                            if (artistProc.IsExists(artist_name) == 0) {
                                    // 가수의 debut date, type, gender 받기
                                    artistProc.insertArtist(artist_name);
                            }
                            // 1-2.가수가 존재하면
                            else if (artistProc.IsExists(artist_name) == 1) {
                            }
                            // 2. 앨범 입력
```

```
                                // 입력한 artist 이름으로 artist_sn 알아내기
                                int artist_sn = artistProc.selectSN(artist_name);
                                System.out.print("▶ Album name : ");
                                String album_name = input.nextLine();
                                // 2-1.앨범이 존재하지 않으면
                                if (albumProc.IsExists(album_name) == 0) {
                                        // 앨범의 release date 받기
                                        albumProc.insertAlbum(album_name, artist_sn);
                                }
                                // 2-2.앨범이 존재하면
                                else if (albumProc.IsExists(album_name) == 1) {
                                }
                                // 3.노래 입력
                                // 입력한 artist 이름으로 artist_sn 알아내기
                                int album_sn = albumProc.selectSN(album_name);
                                System.out.print("▶ Song name : ");
                                String song_title = input.nextLine();
                                // 3-1. 노래가 존재하지 않으면
                                if (songProc.IsExists(song_title) == 0) {
                                        // 앨범의 release date 받기
                                        songProc.insertSong(song_title, album_sn, artist_sn);
                                }
                                // 3-2.노래가 존재하면
                                else if (albumProc.IsExists(album_name) == 1) {
                                        System.out.println("The Song already exists.");
                                }
                        } else if (num == 3) {

    System.out.println("\n\n=================================================================
==========");
                                System.out.println("                                        1.   Artist
2. Album             ");

    System.out.println("=================================================================
======");
                                try {
                                        System.out.print("▶ Which table do you want to handle? :
");

                                        num = input.nextInt();
                                        input.nextLine();
                                        if (!(num >= 1 && num <= 2)) {
                                                throw new InputMismatchException();
                                        }
                                } catch (InputMismatchException e) {
                                        System.out.println("Wrong number");
                                }
                                if (num == 1) {
                                        System.out.println("");
                                        artistProc.showAritstList();
                                        artistProc.updateArtist();
                                }
                                // transaction 활용
                                if (num == 2) {
                                        System.out.println("");
                                        albumProc.showAlbumList();
                                        System.out.print("▶  Enter  the  artist_sn  you  want  to
update : ");

                                        int current_artist_sn = input.nextInt();
                                        System.out.print("▶ What value would you like to replace
artist_sn? : ");

                                        int updated_artist_sn = input.nextInt();
                                        input.nextLine();
                                        artistProc.updateTransaction(current_artist_sn,
updated_artist_sn);
```

```
                                        System.out.println("");
                                        albumProc.showAlbumList();
                                        artistProc.showAritstList();
                                }
                        } else if (num == 4) {

    System.out.println("\n\n=================================================================
==========");
                                System.out.println(" 1. Artist                                2. Album
3. Song ");

    System.out.println("=================================================================
======");
                                try {
                                        System.out.print("▶ Which table do you want to handle? :
");
                                        num = input.nextInt();
                                        input.nextLine();
                                        if (!(num >= 1 && num <= 3)) {
                                                throw new InputMismatchException();
                                        }
                                } catch (InputMismatchException e) {
                                        System.out.println("Wrong number");
                                }
                                if (num == 1) {
                                        System.out.println("");
                                        artistProc.showAritstList();
                                        artistProc.deleteArtist();
                                } else if (num == 2) {
                                        System.out.println("");
                                        albumProc.showAlbumList();
                                        albumProc.deleteAlbum();
                                } else if (num == 3) {
                                        System.out.println("");
                                        songProc.showSongList();
                                        songProc.deleteSong();
                                }
                        } else if (num == 5) {

    System.out.println("\n\n=================================================================
==========");
                                System.out.println("       1. Artist                  2. Join Table
3. Using View");

    System.out.println("=================================================================
======");
                                try {
                                        System.out.print("▶ Which table do you want to handle? :
");
                                        num = input.nextInt();
                                        input.nextLine();
                                        if (!(num >= 1 && num <= 3)) {
                                                throw new InputMismatchException();
                                        }
                                } catch (InputMismatchException e) {
                                        System.out.println("Wrong number");
                                }
                                if (num == 1) {

    System.out.println("\n\n=================================================================
==========");
                                        System.out.println("                                1. group
2. solo");
```

```
    System.out.println("=============================================================================
======");

                                        try {
                                                System.out.print("▶ Select the type you want to
see : ");
                                                num = input.nextInt();
                                                input.nextLine();
                                                if (!(num >= 1 && num <= 2)) {
                                                        throw new InputMismatchException();
                                                }
                                        } catch (InputMismatchException e) {
                                                System.out.println("Wrong number");
                                        }
                                                if(num==1) {
                                                        System.out.println("\n\n");
                                                        artistProc.typeAritstList("group");


                                                }
                                                else if(num==2) {
                                                        System.out.println("\n\n");
                                                        artistProc.typeAritstList("solo");

                                                }
                                }

                                else if (num == 2) {//Join Table
                                        System.out.println("\n\n");
                                        artistProc.showAritstList();
                                        albumProc.showAlbumList();
                                        System.out.println("If you choose Artist's name and Album
name  ");
                                        System.out.println("then you can see song list in that
Artist's Album ");
                                        System.out.print("▶ Enter the Artist's name : ");
                                        String artist_name=input.nextLine();
                                        System.out.print("▶ Enter the Album name : ");
                                        String album_name=input.nextLine();
                                        System.out.println("\n\n");
                                        songProc.showjoinSongList(artist_name, album_name);




                                } else if (num == 3) {//Using View
                                        album_songProc.showView();
                                        System.out.print("▶ Which genre do you want to search
in ? : ");
                                        String genre=input.nextLine();
                                        System.out.println("\n\n");
                                        album_songProc.showbyGenre(genre);


                                }
                        } else if (num == 6) {

    System.out.println("\n\n=========================================================================
==========");
                                System.out.println("                                        ▲ Program END
▲");

    System.out.println("=============================================================================
======");
```

```
                        break;
                    }
                }
            }
        }
```

Album_SongDAO.java

```
package DataAccess;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import DataTransfer.Album_SongDTO;
/**
 * Data Access Object Classes that connect to a database and perform operations
 * such as input , modification, deletion, or query
 *
 * @author Inryu Shin
 *
 */
public class Album_SongDAO {
    private static Connection conn = null;
    private static Statement Stmt = null;
    private static ResultSet rs;
    private PreparedStatement pstmt;
    /**
     * default constructor
     */
    public Album_SongDAO() {
    }
    /**
     * Construct a connection using initialized userID, userPW, dbName, url.
     *
     * @throws ClassNotFoundException
     * @throws SQLException
     */
    private void getConnection() throws ClassNotFoundException, SQLException {
            if (conn == null) { // Connection객체 얻어오기
                    String userID = "dbuser";
                    String userPW = "dbpwd";
                    String dbName = "dbprj";
                    String url = "jdbc:mysql://localhost:3306/" + dbName +
"?&serverTimezone=UTC";
                    conn = DriverManager.getConnection(url, userID, userPW);
            }
    }
    /**
     * Execute select query using join that is view query of the DB so that use can
     * see that view table in console.
     *
     * @return List Album_SongDTO
     */
    public List<Album_SongDTO> getVIEW() {
            List<Album_SongDTO> list = new ArrayList<Album_SongDTO>();
            try {
                    getConnection();
                    String sql = "SELECT name, title, genre FROM ALBUM, SONG WHERE
ALBUM.artist_sn=SONG.artist_sn AND ALBUM.album_sn=SONG.album_sn; ";
                    Statement stmt = conn.createStatement();
                    ResultSet r = stmt.executeQuery(sql);
                    while (r.next()) {
                            // int album_sn = r.getInt("album_sn");
                            // int artist_sn=r.getInt("artist_sn");
```

```
                        String name = r.getString("name");
                        String title = r.getString("title");
                        String genre = r.getString("genre");
                        list.add(new Album_SongDTO(name, title, genre));
                }
        } catch (Exception e) {
                System.out.println("Exception : getVIEW " + e.getMessage());
        } finally {
                dbClose();
        }
        return list;
}
/**
 * Take genre as a parameter and select tuple from view of that genre.
 *
 * @param genre album_genre
 * @return List Album_SongDTO
 */
public List<Album_SongDTO> selectGenre(String genre) {
        List<Album_SongDTO> list = new ArrayList<Album_SongDTO>();
        try {
                getConnection();
                String sql = "SELECT album_name, title, genre FROM album_song WHERE genre=?";
                PreparedStatement pstmt = conn.prepareStatement(sql);
                pstmt.setString(1, genre);
                ResultSet r = pstmt.executeQuery();
                while (r.next()) {
                        String name = r.getString("album_name");
                        String title = r.getString("title");
                        String genre2 = r.getString("genre");
                        list.add(new Album_SongDTO(name, title, genre2));
                }
        } catch (Exception e) {
                System.out.println("Exception : selectGenre " + e.getMessage());
        } finally {
                dbClose();
        }
        return list;
}
/**
 * Disconnect with the DB.
 */
public void dbClose() {
        if (rs != null) {
                try {
                        rs.close();
                } catch (SQLException e) {
                        System.out.println("Exception : ResultSet close():"
e.getMessage());
                }
        }
        if (pstmt != null) {
                try {
                        pstmt.close();
                } catch (SQLException e) {
                        System.out.println("Exception : PreparedStatement close():"
e.getMessage());
                }
        }
        if (conn != null) {
                try {
                        conn.close();
                } catch (SQLException e) {
                        System.out.println("Exception : Connection close():"
e.getMessage());
                }
```

```
            }
            conn = null;
    }
}
```

AlbumDAO.java

```
package DataAccess;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import DataTransfer.AlbumDTO;
import DataTransfer.ArtistDTO;
import DataTransfer.SongDTO;
/**
 * Data Access Object Classes that connect to a database and perform operations
 * such as input , modification, deletion, or query
 *
 * @author Inryu Shin
 *
 */
public class AlbumDAO {
    private static Connection conn = null;
    private static Statement Stmt = null;
    private static ResultSet rs;
    private PreparedStatement pstmt;
    /**
     * default constructor
     */
    public AlbumDAO() {
    }
    /**
     * Construct a connection using initialized userID, userPW, dbName, url.
     *
     * @throws ClassNotFoundException
     * @throws SQLException
     */
    private void getConnection() throws ClassNotFoundException, SQLException {
            if (conn == null) { // Connection객체 얻어오기
                    String userID = "dbuser";
                    String userPW = "dbpwd";
                    String dbName = "dbprj";
                    String     url     =     "jdbc:mysql://localhost:3306/"     +     dbName     +
"?&serverTimezone=UTC";
                    conn = DriverManager.getConnection(url, userID, userPW);
            }
    }
    /**
     * Take AlbumDTO as a parameter and execute insert query.
     *
     * @param AlbumDTO artist
     * @return boolean
     */
    public boolean insertAlbum(AlbumDTO album) {
            boolean result = false;
            try {
                    getConnection();
                    String  query  =  "INSERT  INTO  Album  (name,  release_date,  artist_sn)
VALUES(?,?,?)";
                    PreparedStatement pstmt = conn.prepareStatement(query);
                    pstmt.setString(1, album.getName());
                    pstmt.setString(2, album.getRelease_date());
```

```
                pstmt.setInt(3, album.getArtist_sn());
                int r = pstmt.executeUpdate();
                if (r > 0)
                        result = true;
        } catch (Exception e) {
                System.out.println("Exception : insertArtist " + e.getMessage());
        } finally {
                dbClose();
        }
        return result;
}
/**
 * Take the album_name of the album as a parameter and check if the data with that
 * name exists on the album table.
 *
 * @param String album_name
 * @return 0(not exists) 1(exists)
 */
public int IsExists(String album_name) {
        int result = 2;
        int n = 2;
        try {
                getConnection();
                String query = "SELECT count(*) FROM Album WHERE name=?";
                PreparedStatement pstmt = conn.prepareStatement(query);
                pstmt.setString(1, album_name);
                ResultSet r = pstmt.executeQuery();
                while (r.next()) {
                        n = r.getInt("count(*)"); // count(*)을 한 attribute의 값 가져오기
                }
                if (n == 0) { // 0이면 해당 이름의 앨범이 존재하지 않는 것
                        result = 0;
                }
                else { // 해당 이름의 앨범이 존재하면
                        result = 1;
                }
        } catch (Exception e) {
                System.out.println("Exception : IsExists " + e.getMessage());
        } finally {
                dbClose();
        }
        return result;
}

/**
 * Take the album_sn as a parameter and select all tuple from album of that
 * album_sn.
 *
 * @param int sn : album_sn
 * @return AlbumDTO
 */
public AlbumDTO getAlbum(int sn) {
        AlbumDTO dto = null;
        try {
                getConnection();
                String sql = "SELECT * FROM Album WHERE album_sn = ?";
                PreparedStatement pstmt = conn.prepareStatement(sql);
                pstmt.setInt(1, sn);
                ResultSet r = pstmt.executeQuery();
                if (r.next()) {
                        int album_sn = r.getInt("album_sn");
                        String name = r.getString("name");
                        String release_date = r.getString("release_date");
                        int artist_sn = r.getInt("artist_sn");
                        dto = new AlbumDTO(album_sn, name, release_date, artist_sn);
                }
        } catch (Exception e) {
```

```
                System.out.println("Exception :getAlbum " + e.getMessage());
        } finally {
                dbClose();
        }
        return dto;
}
/**
 * Take the album_name as a parameter and select album_sn from album of that
 * album name.
 *
 * @param album_name
 * @return album_sn
 */
public int selectSN(String album_name) {
        int album_sn = -1;
        try {
                getConnection();
                String query = "SELECT album_sn FROM Album WHERE name=?";
                PreparedStatement pStmt = conn.prepareStatement(query);
                pStmt.setString(1, album_name);
                ResultSet rs = pStmt.executeQuery();
                while (rs.next()) {
                        album_sn = rs.getInt("album_sn");
                }
        } catch (Exception e) {
                System.out.println("Exception : selectSN " + e.getMessage());
        } finally {
                dbClose();
        }
        return album_sn;
}
/**
 * Select all from Album
 * @return List<AlbumDTO>
 */
public List<AlbumDTO> getAlbumList() {
        List<AlbumDTO> list = new ArrayList<AlbumDTO>();
        try {
                getConnection();
                String sql = "SELECT * FROM Album";
                PreparedStatement pstmt = conn.prepareStatement(sql);
                ResultSet r = pstmt.executeQuery();
                while (r.next()) {
                        int album_sn = r.getInt("album_sn");
                        String name = r.getString("name");
                        String release_date = r.getString("release_date");
                        int artist_sn = r.getInt("artist_sn");
                        list.add(new AlbumDTO(album_sn, name, release_date, artist_sn));
                }
        } catch (Exception e) {
                System.out.println("Exception : getAlbumList " + e.getMessage());
        } finally {
                dbClose();
        }
        return list;
}
/**
 * Take album_sn as parameter and delete Album table tuple of that album_sn.
 */
public boolean deleteAlbum(int sn) {
        boolean result = false;
        try {
                getConnection();
                String sql = "DELETE FROM Album WHERE album_sn = ?";
                PreparedStatement pstmt = conn.prepareStatement(sql);
                pstmt.setInt(1, sn);
                int r = pstmt.executeUpdate();
```

```
                        if (r > 0)
                                result = true;
                } catch (Exception e) {
                        System.out.println("Exception : deleteAlbum " + e.getMessage());
                } finally {
                        dbClose();
                }
                return result;
        }
        /**
         * Disconnect with the DB.
         */
        public void dbClose() {
                if (rs != null) {
                        try {
                                rs.close();
                        } catch (SQLException e) {
                                System.out.println("Exception     :     ResultSet     close():"     +
e.getMessage());
                        }
                }
                if (pstmt != null) {
                        try {
                                pstmt.close();
                        } catch (SQLException e) {
                                System.out.println("Exception   :   PreparedStatement   close():"   +
e.getMessage());
                        }
                }
                if (conn != null) {
                        try {
                                conn.close();
                        } catch (SQLException e) {
                                System.out.println("Exception    :    Connection       close():"    +
e.getMessage());
                        }
                }
                conn = null;
        }
}
```

ArtistDAO.java

```
package DataAccess;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import DataTransfer.ArtistDTO;
/**
 * Data Access Object Classes that connect to a database and perform operations
 * such as input , modification, deletion, or query
 *
 * @author Inryu Shin
 *
 */
public class ArtistDAO {
    private static Connection conn = null;
    private static Statement Stmt = null;
```

```
    private static ResultSet rs;
    private PreparedStatement pstmt;
    /**
     * default constructor
     */
    public ArtistDAO() {
    }
    /**
     * Construct a connection using initialized userID, userPW, dbName, url.
     *
     * @throws ClassNotFoundException
     * @throws SQLException
     */
    private void getConnection() throws ClassNotFoundException, SQLException {
            if (conn == null) { // Connection객체 얻어오기
                    String userID = "dbuser";
                    String userPW = "dbpwd";
                    String dbName = "dbprj";
                    String     url     =     "jdbc:mysql://localhost:3306/"     +     dbName     +
"?&serverTimezone=UTC";
                    conn = DriverManager.getConnection(url, userID, userPW);
            }
    }
    /**
     * Take ArtistDTO as a parameter and execute insert query.
     *
     * @param AritstDTO artist
     * @return boolean
     */
    public boolean insertArtist(ArtistDTO artist) {
            boolean result = false;
            try {
                    getConnection();
                    String  query  =  "INSERT  INTO  Artist  (name,  debut_date,  type,  gender)
VALUES(?,?,?,?)";
                    PreparedStatement pstmt = conn.prepareStatement(query);
                    pstmt.setString(1, artist.getName());
                    pstmt.setString(2, artist.getDebut_date());
                    pstmt.setString(3, artist.getType());
                    pstmt.setString(4, artist.getGender());
                    int r = pstmt.executeUpdate();
                    if (r > 0)
                            result = true;
            } catch (Exception e) {
                    System.out.println("Exception : insertArtist " + e.getMessage());
            } finally {
                    dbClose();
            }
            return result;
    }
    /**
     * Take the name of the artist as a parameter and check if the data with that
     * name exists on the artist table.
     *
     * @param String artist_name
     * @return 0(not exists) 1(exists)
     */
    public int IsExists(String artist_name) {
            int result = 2;
            int n = 2;
            try {
                    getConnection();
                    String query = "SELECT count(*) FROM Artist WHERE name=?";
                    PreparedStatement pstmt = conn.prepareStatement(query);
                    pstmt.setString(1, artist_name);
                    ResultSet r = pstmt.executeQuery();
                    while (r.next()) {
                            n = r.getInt("count(*)"); // count(*)을 한 attribute의 값 가져오기
                    }
```

```
                        if (n == 0) { // 0이면 해당 이름의 아티스트가 존재하지 않는 것
                                result = 0;
                        }
                        else { // 해당 이름의 아티스트가 존재하면
                                result = 1;
                        }
                } catch (Exception e) {
                        System.out.println("Exception : IsExists " + e.getMessage());
                } finally {
                        dbClose();
                }
                return result;
        }
        /**
         * Take the artist_sn as a parameter and select all tuple from artist of that
         * artist's sn.
         *
         * @param int sn : artist_sn
         * @return ArtistDTO
         */
        public ArtistDTO getArtist(int sn) {
                ArtistDTO dto = null;
                try {
                        getConnection();
                        String sql = "SELECT artist_sn, name, debut_date, type, gender FROM Artist
WHERE artist_sn = ?";
                        PreparedStatement pstmt = conn.prepareStatement(sql);
                        pstmt.setInt(1, sn);
                        ResultSet r = pstmt.executeQuery();
                        if (r.next()) {
                                int artist_sn = r.getInt("artist_sn");
                                String name = r.getString("name");
                                String debut_date = r.getString("debut_date");
                                String type = r.getString("type");
                                String gender = r.getString("gender");
                                dto = new ArtistDTO(artist_sn, name, debut_date, type, gender);
                        }
                } catch (Exception e) {
                        System.out.println("Exception :getArtist " + e.getMessage());
                } finally {
                        dbClose();
                }
                return dto;
        }
        /**
         * Take the artist_name as a parameter and select artist_sn from artist of that
         * artist's name.
         *
         * @param artist_name
         * @return artist_sn
         */
        public int selectSN(String artist_name) {
                int artist_sn = -1;
                try {
                        getConnection();
                        String query = "SELECT artist_sn FROM Artist WHERE name=?";
                        PreparedStatement pStmt = conn.prepareStatement(query);
                        pStmt.setString(1, artist_name);
                        ResultSet rs = pStmt.executeQuery();
                        while (rs.next()) {
                                artist_sn = rs.getInt("artist_sn");
                        }
                } catch (Exception e) {
                        System.out.println("Exception : selectSN " + e.getMessage());
                } finally {
                        dbClose();
                }
```

```
		return artist_sn;
	}
	/**
	 * Select all from Artist.
	 *
	 * @return List<ArtistDTO>
	 */
	public List<ArtistDTO> getArtistList() {
		List<ArtistDTO> list = new ArrayList<ArtistDTO>();
		try {
			getConnection();
			String sql = "SELECT * FROM ARTIST";
			PreparedStatement pstmt = conn.prepareStatement(sql);
			ResultSet r = pstmt.executeQuery();
			while (r.next()) {
				int artist_sn = r.getInt("artist_sn");
				String name = r.getString("name");
				String debut_date = r.getString("debut_date");
				String type = r.getString("type");
				String gender = r.getString("gender");
				list.add(new ArtistDTO(artist_sn, name, debut_date, type, gender));
			}
		} catch (Exception e) {
			System.out.println("Exception : getArtistList " + e.getMessage());
		} finally {
			dbClose();
		}
		return list;
	}
	/**
	 * Take the type as a parameter and select all from artist of that type.
	 * artist's name.
	 *
	 * @param type
	 * @return List<ArtistDTO>
	 */
	public List<ArtistDTO> gettypeArtist(String type) {
		List<ArtistDTO> list = new ArrayList<ArtistDTO>();
		try {
			getConnection();
			String sql = "SELECT * FROM ARTIST WHERE type=?";
			PreparedStatement pstmt = conn.prepareStatement(sql);
			pstmt.setString(1, type);
			ResultSet r = pstmt.executeQuery();
			while (r.next()) {
				int artist_sn = r.getInt("artist_sn");
				String name = r.getString("name");
				String debut_date = r.getString("debut_date");
				String type2 = r.getString("type");
				String gender = r.getString("gender");
				list.add(new ArtistDTO(artist_sn, name, debut_date, type2, gender));
			}
		} catch (Exception e) {
			System.out.println("Exception : gettypeArtist " + e.getMessage());
		} finally {
			dbClose();
		}
		return list;
	}
	/**
	 * Take ArtistDTO as parameter and update Artist table based on
	 * ArtistDTO(parameter)'s member and current_artist_sn
	 *
	 * @param ArtistDTO
	 * @param current_artist_sn
	 * @return
	 */
	public boolean updateArtist(ArtistDTO dto, int current_artist_sn) {
```

```
		boolean result = false;
		try {
			getConnection();
			String sql = "UPDATE Artist SET artist_sn=?,name=?,debut_date=?,type=?,gender=? WHERE artist_sn=?";
			PreparedStatement pstmt = conn.prepareStatement(sql);
			pstmt.setInt(1, dto.getArtist_sn());
			pstmt.setString(2, dto.getName());
			pstmt.setString(3, dto.getDebut_date());
			pstmt.setString(4, dto.getType());
			pstmt.setString(5, dto.getGender());
			pstmt.setInt(6, current_artist_sn);
			int r = pstmt.executeUpdate();
			if (r > 0)
				result = true;
		} catch (Exception e) {
			System.out.println("Exception :updateArtist  " + e.getMessage());
		} finally {
			dbClose();
		}
		return result;
	}
	/**
	 * Update artist_sn in Artist and Song table by using transaction.
	 *
	 * @param current_artist_sn
	 * @param updated_artist_sn
	 * @throws SQLException
	 */
	public void updateTransaction(int current_artist_sn, int updated_artist_sn) throws SQLException {
		try {
			getConnection();
			conn.setAutoCommit(false);
			String sql = "UPDATE Artist SET artist_sn=? WHERE artist_sn=?";
			PreparedStatement pstmt = conn.prepareStatement(sql);
			pstmt.setInt(1, updated_artist_sn);
			pstmt.setInt(2, current_artist_sn);
			pstmt.executeUpdate();
			sql = "UPDATE Song SET artist_sn=? WHERE artist_sn=?";
			pstmt.setInt(1, updated_artist_sn);
			pstmt.setInt(2, current_artist_sn);
			pstmt.executeUpdate();
			conn.commit();
		} catch (Exception e) {
			System.out.println("Exception : updateTransaction " + e.getMessage());
		} finally {
			conn.setAutoCommit(true);
			dbClose();
		}
	}
	/**
	 * Take artist_sn as parameter and delete Artist table tuple of that artitst_sn.
	 */
	public boolean deleteArtist(int sn) {
		boolean result = false;
		try {
			getConnection();
			String sql = "DELETE FROM Artist WHERE artist_sn = ?";
			PreparedStatement pstmt = conn.prepareStatement(sql);
			pstmt.setInt(1, sn);
			int r = pstmt.executeUpdate();
			if (r > 0)
				result = true;
		} catch (Exception e) {
			System.out.println("Exception :deleteArtist " + e.getMessage());
		} finally {
```

```
                        dbClose();
                }
                return result;
        }
        /**
         * Disconnect with the DB.
         */
        public void dbClose() {
                if (rs != null) {
                        try {
                                rs.close();
                        } catch (SQLException e) {
                                System.out.println("Exception     :     ResultSet     close():"     +
e.getMessage());
                        }
                }
                if (pstmt != null) {
                        try {
                                pstmt.close();
                        } catch (SQLException e) {
                                System.out.println("Exception   :   PreparedStatement   close():"   +
e.getMessage());
                        }
                }
                if (conn != null) {
                        try {
                                conn.close();
                        } catch (SQLException e) {
                                System.out.println("Exception    :    Connection       close():"    +
e.getMessage());
                        }
                }
                conn = null;
        }
}
```

SongDAO.java

```
package DataAccess;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import DataTransfer.AlbumDTO;
import DataTransfer.SongDTO;
/**
 * Data Access Object Classes that connect to a database and perform operations
 * such as input , modification, deletion, or query
 *
 * @author Inryu Shin
 *
 */
public class SongDAO {
```

```
    private static Connection conn = null;
    private static Statement Stmt = null;
    private static ResultSet rs;
    private PreparedStatement pstmt;
    /**
     * default constructor
     */
    public SongDAO() {
    }
    /**
     * Construct a connection using initialized userID, userPW, dbName, url.
     *
     * @throws ClassNotFoundException
     * @throws SQLException
     */
    private void getConnection() throws ClassNotFoundException, SQLException {
            if (conn == null) { // Connection객체 얻어오기
                    String userID = "dbuser";
                    String userPW = "dbpwd";
                    String dbName = "dbprj";
                    String     url     =     "jdbc:mysql://localhost:3306/"     +     dbName     +
"?&serverTimezone=UTC";
                    conn = DriverManager.getConnection(url, userID, userPW);
            }
    }
    /**
     * Take SongDTO as a parameter and execute insert query.
     *
     * @param SongDTO artist
     * @return boolean
     */
    public boolean insertSong(SongDTO song) {
            boolean result = false;
            try {
                    getConnection();
                    String  query  =  "INSERT  INTO  Song  (title,  genre,  album_sn,  artist_sn)
VALUES(?,?,?,?)";
                    PreparedStatement pstmt = conn.prepareStatement(query);
                    pstmt.setString(1, song.getTitle());
                    pstmt.setString(2, song.getGenre());
                    pstmt.setInt(3, song.getAlbum_sn());
                    pstmt.setInt(4, song.getArtist_sn());
                    int r = pstmt.executeUpdate();
                    if (r > 0)
                            result = true;
            } catch (Exception e) {
                    System.out.println("Exception : insertSong " + e.getMessage());
            } finally {
                    dbClose();
            }
            return result;
    }
    /**
     * Take the song_title of the Song as a parameter and check if the data with
     * that title exists on the Song table.
     *
     * @param String song_title
     * @return 0(not exists) 1(exists)
     */
    public int IsExists(String song_title) {
            int result = 2;
            int n = 2;
            try {
                    getConnection();
                    String query = "SELECT count(*) FROM Song WHERE title=?";
                    PreparedStatement pstmt = conn.prepareStatement(query);
                    pstmt.setString(1, song_title);
                    ResultSet r = pstmt.executeQuery();
                    while (r.next()) {
```

```
                        n = r.getInt("count(*)"); // count(*)을 한 attribute의 값 가져오기
                }
                if (n == 0) { // 0이면 해당 이름의 아티스트가 존재하지 않는 것
                        result = 0;
                }
                else { // 해당 이름의 아티스트가 존재하면
                        result = 1;
                }
        } catch (Exception e) {
                System.out.println("Exception : IsExists " + e.getMessage());
        } finally {
                dbClose();
        }
        return result;
}
/**
 * Take the song_sn as a parameter and select all tuple from Song of that
 * song_sn.
 *
 * @param int sn : song_sn
 * @return SongDTO
 */
public SongDTO getSong(int sn) {
        SongDTO dto = null;
        try {
                getConnection();
                String sql = "SELECT * FROM Song WHERE song_sn = ?";
                PreparedStatement pstmt = conn.prepareStatement(sql);
                pstmt.setInt(1, sn);
                ResultSet r = pstmt.executeQuery();
                if (r.next()) {
                        int song_sn = r.getInt("song_sn");
                        String title = r.getString("title");
                        String genre = r.getString("genre");
                        int album_sn = r.getInt("album_sn");
                        int artist_sn = r.getInt("artist_sn");
                        dto = new SongDTO(song_sn, title, genre, album_sn, artist_sn);
                }
        } catch (Exception e) {
                System.out.println("Exception :getSong" + e.getMessage());
        } finally {
                dbClose();
        }
        return dto;
}
/**
 * Select all from Song
 *
 * @return List<SongDTO>
 */
public List<SongDTO> getSongList() {
        List<SongDTO> list = new ArrayList<SongDTO>();
        try {
                getConnection();
                String sql = "SELECT * FROM Song";
                PreparedStatement pstmt = conn.prepareStatement(sql);
                ResultSet r = pstmt.executeQuery();
                while (r.next()) {
                        int song_sn = r.getInt("song_sn");
                        String title = r.getString("title");
                        String genre = r.getString("genre");
                        int album_sn = r.getInt("album_sn");
                        int artist_sn = r.getInt("artist_sn");
                        list.add(new SongDTO(song_sn, title, genre, album_sn, artist_sn));
                }
        } catch (Exception e) {
```

```
                System.out.println("Exception : getSongList " + e.getMessage());
            } finally {
                dbClose();
            }
            return list;
    }
    // join select : 아티스트, 앨범 이름에 따라 노래 출력하기
    /**
     * Take the artist_name and album_name as a parameter and select title,genre
     * from Song with tha artist_name and album_name using join
     *
     * @param artist_name
     * @param album_name
     * @return List<SongDTO>
     */
    public List<SongDTO> getjoinSongList(String artist_name, String album_name) {
            List<SongDTO> list = new ArrayList<SongDTO>();
            try {
                getConnection();
                String sql = "SELECT title, genre FROM song WHERE album_sn in (SELECT
album_sn FROM artist,album WHERE artist.artist_sn=album.artist_sn AND artist.name=? AND
album.name=?)";
                PreparedStatement pstmt = conn.prepareStatement(sql);
                pstmt.setString(1, artist_name);
                pstmt.setString(2, album_name);
                ResultSet r = pstmt.executeQuery();
                while (r.next()) {
                    String title = r.getString("title");
                    String genre = r.getString("genre");
                    list.add(new SongDTO(title, genre));
                }
            } catch (Exception e) {
                System.out.println("Exception : getjoinSongList " + e.getMessage());
            } finally {
                dbClose();
            }
            return list;
    }
    /**
     * Take song_sn as parameter and delete Song table tuple of that song_sn.
     */
    public boolean deleteSong(int sn) {
            boolean result = false;
            try {
                getConnection();
                String sql = "DELETE FROM Song WHERE song_sn = ?";
                PreparedStatement pstmt = conn.prepareStatement(sql);
                pstmt.setInt(1, sn);
                int r = pstmt.executeUpdate();
                if (r > 0)
                    result = true;
            } catch (Exception e) {
                System.out.println("Exception : deleteSong " + e.getMessage());
            } finally {
                dbClose();
            }
            return result;
    }
    /**
     * Disconnect with the DB.
     */
    public void dbClose() {
            if (rs != null) {
                try {
                    rs.close();
                } catch (SQLException e) {
                    System.out.println("Exception : ResultSet close():" +
```

```
e.getMessage());
                        }
                }
                if (pstmt != null) {
                        try {
                                pstmt.close();
                        } catch (SQLException e) {
                                System.out.println("Exception  :  PreparedStatement  close():"  +
e.getMessage());
                        }
                }
                if (conn != null) {
                        try {
                                conn.close();
                        } catch (SQLException e) {
                                System.out.println("Exception  :  Connection  close():"  +
e.getMessage());
                        }
                }
                conn = null;
        }
```

Album_SongDTO.java

```
package DataTransfer;
import java.util.Formatter;
/**
 * Object of Album join Song table data transfer (DTO) Columns in the table are
 * treated as member variables.
 *
 * @author Inryu Shin
 *
 */
public class Album_SongDTO {
    // member variable : columns of table
    private String name;
    private String title;
    private String genre;
    /**
     * default_constructor
     */
    public Album_SongDTO() {
    }
    /**
     * constructor
     *
     * @param album_name
     * @param title
     * @param genre
     */
    public Album_SongDTO(String album_name, String title, String genre) {
            // this.album_sn=album_sn;
            // this.artist_sn=artist_sn;
            this.name = album_name;
            this.title = title;
            this.genre = genre;
    }
    /**
     * @Override to String print this class easy to see.
     */
    public String toString() {
            Formatter fm = new Formatter();
            String Artistinfo = fm.format("%-20s\t  %-25s\t%-25s", name, title, genre).toString();
            return Artistinfo;
    }
}
```

AlbumDTO.java

```
package DataTransfer;
import java.util.Formatter;
/**
 * Object of Album table data transfer (DTO) Columns in the table are treated as
 * member variables.
 *
 * @author Inryu Shin
 *
 */
public class AlbumDTO {
    // member variable : columns of table
    private int album_sn;
    private String name;
    private String release_date;
    private int artist_sn;
    /**
     * default constructor
     */
    public AlbumDTO() {
    }
    /**
     * constructor
     *
     * @param album_sn
     * @param name
     * @param release_date
     * @param artist_sn
     */
    public AlbumDTO(int album_sn, String name, String release_date, int artist_sn) {
            this.album_sn = album_sn;
            this.name = name;
            this.release_date = release_date;
            this.artist_sn = artist_sn;
    }
    /**
     * constructor
     *
     * @param name
     * @param release_date
     * @param artist_sn
     */
    public AlbumDTO(String name, String release_date, int artist_sn) {
            this.name = name;
            this.release_date = release_date;
            this.artist_sn = artist_sn;
    }
    // getter and setter
    /**
     *
     * @return album_sn
     */
    public int getAlbum_sn() {
            return album_sn;
    }
    /**
     * set album_sn
     *
     * @param album_sn
     */
    public void setAlbum_sn(int album_sn) {
            this.album_sn = album_sn;
    }
    /**
     *
     * @return name
     */
    public String getName() {
            return name;
    }
    /**
     * set name
```

```
     *
     * @param name
     */
    public void setName(String name) {
            this.name = name;
    }
    /**
     *
     * @return release_date
     */
    public String getRelease_date() {
            return release_date;
    }
    /**
     * set release_date
     *
     * @param release_date
     */
    public void setRelease_date(String release_date) {
            this.release_date = release_date;
    }
    /**
     *
     * @return artist_sn
     */
    public int getArtist_sn() {
            return artist_sn;
    }
    /**
     * set artist_sn
     *
     * @param artist_sn
     */
    public void setArtist_sn(int artist_sn) {
            this.artist_sn = artist_sn;
    }
    /**
     * @Override to String print this class easy to see.
     */
    public String toString() {
            Formatter fm = new Formatter();
            String  Artistinfo  =  fm.format("%9s\t     %-19s\t%-20s\t%-14s",  album_sn,  name,
release_date, artist_sn).toString();
            return Artistinfo;
    }
    /**
     * print this class based on my consol format style.
     */
    public void printInfo() {

    System.out.println("=============================================================================
======");
            System.out.println("album_sn\t  name \t\t\trelease_date\tartist_sn\t\t");

    System.out.println("=============================================================================
======");
            System.out.println(this.toString());

    System.out.println("=============================================================================
======");
    }
}
```

ArtistDTO.java

```
package DataTransfer;
import java.util.Formatter;
/**
 * Object of Artist table data transfer (DTO) Columns in the table are treated
```

```
 * as member variables.
 *
 * @author Inryu Shin
 *
 */
public class ArtistDTO {
    // member variable : columns of table
    private int artist_sn;
    private String name;
    private String debut_date;
    private String type;
    private String gender;
    /**
     * default constructor
     */
    public ArtistDTO() {
    }
    /**
     * constructor
     *
     * @param artist_sn
     * @param name
     * @param debut_date
     * @param type
     * @param gender
     */
    public ArtistDTO(int artist_sn, String name, String debut_date, String type, String gender) {
            this.artist_sn = artist_sn;
            this.name = name;
            this.debut_date = debut_date;
            this.type = type;
            this.gender = gender;
    }
    /**
     * constructor
     *
     * @param name
     * @param debut_date
     * @param type
     * @param gender
     */
    public ArtistDTO(String name, String debut_date, String type, String gender) {
            // this.artist_sn=artist_sn;
            this.name = name;
            this.debut_date = debut_date;
            this.type = type;
            this.gender = gender;
    }
    // getter / setter
    /**
     *
     * @return artist_sn
     */
    public int getArtist_sn() {
            return artist_sn;
    }
    /**
     * set artist_sn
     *
     * @param artist_sn
     */
    public void setArtist_sn(int artist_sn) {
            this.artist_sn = artist_sn;
    }
    /**
     *
     * @return name
     */
    public String getName() {
            return name;
    }
    /**
     * set name
     *
```

```
     * @param name
     */
    public void setName(String name) {
            this.name = name;
    }
    /**
     *
     * @return dabut_date
     */
    public String getDebut_date() {
            return debut_date;
    }
    /**
     * set debut_date
     *
     * @param debut_date
     */
    public void setDebut_date(String debut_date) {
            this.debut_date = debut_date;
    }
    /**
     *
     * @return type
     */
    public String getType() {
            return type;
    }
    /**
     * set type
     *
     * @param type
     */
    public void setType(String type) {
            this.type = type;
    }
    /**
     *
     * @return gender
     */
    public String getGender() {
            return gender;
    }
    /**
     * set gender
     *
     * @param gender
     */
    public void setGender(String gender) {
            this.gender = gender;
    }
    /**
     * @Override to String print this class easy to see.
     */
    public String toString() {
            Formatter fm = new Formatter();
            String Artistinfo = fm.format("%9s\t   %-7s\t%-15s\t%-14s\t%-14s", artist_sn, name,
debut_date, type, gender)
                            .toString();
            return Artistinfo;
    }
    /**
     * print this class based on my consol format style.
     */
    public void printInfo() {

    System.out.println("=============================================================================
======");
            System.out.println("artist_sn\t  name \t\tdebut_date\ttype\t\tgender");

    System.out.println("=============================================================================
======");
            System.out.println(this.toString());
```

```
    System.out.println("================================================================================
======");
    }
}
```

SongDTO.java

```
package DataTransfer;
import java.util.Formatter;
/**
 * Object of Song table data transfer (DTO) Columns in the table are treated as
 * member variables.
 *
 * @author Inryu Shin
 *
 */
public class SongDTO {
    // member variable : columns of table
    private int song_sn;
    private String title;
    private String genre;
    private int album_sn;
    private int artist_sn;
    /**
     * default constructor
     */
    public SongDTO() {
    }
    /**
     * constuctor
     *
     * @param song_sn
     * @param title
     * @param genre
     * @param album_sn
     * @param artist_sn
     */
    public SongDTO(int song_sn, String title, String genre, int album_sn, int artist_sn) {
            this.song_sn = song_sn;
            this.title = title;
            this.genre = genre;
            this.album_sn = album_sn;
            this.artist_sn = artist_sn;
    }
    /**
     * constructor
     *
     * @param title
     * @param genre
     * @param album_sn
     * @param artist_sn
     */
    public SongDTO(String title, String genre, int album_sn, int artist_sn) {
            this.title = title;
            this.genre = genre;
            this.album_sn = album_sn;
            this.artist_sn = artist_sn;
    }
    /**
     * constructor
     *
     * @param title
     * @param genre
     */
    public SongDTO(String title, String genre) {
            this.title = title;
            this.genre = genre;
    }
    /**
     *
```

```
     * @return song_sn
     */
    public int getSong_sn() {
            return song_sn;
    }
    /**
     * set song_sn
     *
     * @param song_sn
     */
    public void setSong_sn(int song_sn) {
            this.song_sn = song_sn;
    }
    /**
     *
     * @return title
     */
    public String getTitle() {
            return title;
    }
    /**
     * set title
     *
     * @param title
     */
    public void setTitle(String title) {
            this.title = title;
    }
    /**
     *
     * @return genre
     */
    public String getGenre() {
            return genre;
    }
    /**
     * set genre
     *
     * @param genre
     */
    public void setGenre(String genre) {
            this.genre = genre;
    }
    /**
     *
     * @return album_sn
     */
    public int getAlbum_sn() {
            return album_sn;
    }
    /**
     * set album_sn
     *
     * @param album_sn
     */
    public void setAlbum_sn(int album_sn) {
            this.album_sn = album_sn;
    }
    /**
     *
     * @return artist_sn
     */
    public int getArtist_sn() {
            return artist_sn;
    }
    /**
     * set artist_sn
     *
     * @param artist_sn
     */
    public void setArtist_sn(int artist_sn) {
            this.artist_sn = artist_sn;
    }
    /**
```

```
     * @Override to String print this class easy to see.
     */
    public String toString() {
            Formatter fm = new Formatter();
            String Artistinfo = fm.format("%9s\t  %-18s\t%-11s\t%-9s\t%-9s", song_sn, title, genre,
album_sn, artist_sn)
                            .toString();
            return Artistinfo;
    }
    /**
     * print this class based on my consol format style.
     */
    public void printInfo() {

    System.out.println("=============================================================================
======");
            System.out.println("song_sn\t\t title\t\t\tgenre\t  album_sn\tartist_sn");

    System.out.println("=============================================================================
======");
            System.out.println(this.toString());

    System.out.println("=============================================================================
======");
    }
    /**
     * print this class based on my consol format style.
     */
    public void printDTO() {
            Formatter fm = new Formatter();
            //
System.out.println("=============================================================================
==");
            // System.out.println("title\t\t\t genre\t" );
            //
System.out.println("=============================================================================
==");
            System.out.println(fm.format("%30s\t  %30s\t", title, genre));
            //
System.out.println("=============================================================================
==");
    }
}
```

Album_SongProc.java

```
package Process;
import java.util.List;
import java.util.Scanner;
import DataAccess.Album_SongDAO;
import DataTransfer.AlbumDTO;
import DataTransfer.Album_SongDTO;
/**
 * Create this object in the main and use this class's method. In this class,
 * After receiving user input within the method, perform the query using the DAO
 * object.
 *
 * @author Inryu Shin
 *
 */
public class Album_SongProc {
    Album_SongDAO dao; // ArtistDAO를 멤버변수로 가짐
    static public Scanner input = new Scanner(System.in);
    /**
     * default constructor
```

```
     */
    public Album_SongProc() {
            dao = new Album_SongDAO();
    }
    /**
     * Call the DAO object's getView function into List Album_SongDTO list and then
     * print that list.
     */
    public void showView() {
            List<Album_SongDTO> list = dao.getVIEW();
            System.out.println("\n                                <VIEW>");

    System.out.println("=============================================================================
======");
            if (list != null && list.size() > 0) {
                    System.out.println("   name\t\t\t  title \t\t\tgenre");

    System.out.println("=============================================================================
======");
                    for (Album_SongDTO dto : list) {
                            System.out.println(dto);
                    }
            } else {
                    System.out.println("저장된 데이터가 없습니다. ");
            }

    System.out.println("=============================================================================
======");
    }

    /**
     * Call the DAO object's selectGenre function into List Album_SongDTO list and then
     * print that list.
     * @param genre : Album genre
     */
    public void showbyGenre(String genre) {
            List<Album_SongDTO> list = dao.selectGenre(genre);

    System.out.println("=============================================================================
======");
            if (list != null && list.size() > 0) {
                    System.out.println("   name\t\t\t  title \t\t\tgenre");

    System.out.println("=============================================================================
======");
                    for (Album_SongDTO dto : list) {
                            System.out.println(dto);
                    }
            } else {
                    System.out.println("저장된 데이터가 없습니다. ");
            }

    System.out.println("=============================================================================
======");
    }
}
```

AlbumProc.java

```
package Process;
import java.util.List;
import java.util.Scanner;
import DataAccess.AlbumDAO;
import DataTransfer.AlbumDTO;
import DataTransfer.ArtistDTO;
```

```java
import DataTransfer.SongDTO;
/**
 * Create this object in the main and use this class's method. In this class,
 * After receiving user input within the method, perform the query using the DAO
 * object.
 *
 * @author Inryu Shin
 *
 */
public class AlbumProc {
    // 멤버 변수
    AlbumDAO dao;
    static public Scanner input = new Scanner(System.in);
    /**
     * default constructor
     */
    public AlbumProc() {
            dao = new AlbumDAO();
    }
    /**
     * Take album_name, artist_sn as parameter and take input from user of
     * release_date. Create DTO object using those values and call the DAO object's
     * insertAlbum function using DTO as parameter. *
     *
     * @param album_name
     * @param artist_sn
     */
    public void insertAlbum(String album_name, int artist_sn) { // 인자로 이름을 받음
            System.out.println("\nEnter album information.");
            // System.out.print("name : ");
            // String name = input.nextLine();
            System.out.print("▶ release date : ");
            String release_date = input.nextLine();
            // Artistdto 객체 생성 (입력 받은 값과 생성자 이용)
            AlbumDTO album = new AlbumDTO(album_name, release_date, artist_sn);
            boolean r = dao.insertAlbum(album); // 입력받은 데이터 추가
            if (r) {
                    System.out.println("Input is entered successfully.");
            } else {
                    System.out.println("Failed. Input is not entered");
            }
    }
    /**
     * Take album_name as parameter and and call the DAO object's IsExists function.
     *
     * @param album_name
     * @return
     */
    public int IsExists(String album_name) {
            int n = dao.IsExists(album_name);
            int result;
            if (n == 0)
                    result = 0;
            else
                    result = 1;
            return result;
    }
    /**
     * Take album_name as parameter and call the DAO object's selectSN function.
     *
     * @param album_name
     * @return
     */
    public int selectSN(String album_name) {
            int n = dao.selectSN(album_name);
            return n;
    }
    /**
     * Call the DAO object's getAlbumList function into List<AlbumDTO> list and then
     * print that list.
     */
    public void showAlbumList() {
```

```
            List<AlbumDTO> list = dao.getAlbumList();
            System.out.println("\n                                <Album List>");

    System.out.println("=========================================================================
======");
            if (list != null && list.size() > 0) {
                    System.out.println("album_sn\t  name \t\t\trelease_date\tartist_sn\t\t");

    System.out.println("=========================================================================
======");
                    for (AlbumDTO dto : list) {
                            System.out.println(dto);
                    }
            } else {
                    System.out.println("저장된 데이터가 없습니다. ");
            }

    System.out.println("=========================================================================
======");
    }
    /**
     * Take input from user of album_sn and call the DAO object's deleteAlbum(sn)
     * function using input value.
     */
    public void deleteAlbum() {
            System.out.print("▶ Enter album_sn of Album you want to delete :  ");
            int sn = input.nextInt();
            System.out.println("\n\n");
            AlbumDTO dto = dao.getAlbum(sn);
            if (dto != null) {
                    dto.printInfo();
                    System.out.print("▶ Are you sure you want to delete it?(Y/N) : ");
                    input.nextLine();
                    String ans = input.nextLine();
                    if (ans.equalsIgnoreCase("y")) {
                            boolean r = dao.deleteAlbum(sn);
                            if (r) {
                                    System.out.println("Album [ " + sn + " ] 's information has
been deleted successfully.");
                            } else {
                                    System.out.println("Failed.   Aritst   [   \"+sn+\"   ]   's
information has been not deleted.");
                            }
                    } else {
                            System.out.println("Deletion operation canceled.");
                    }
            } else {
                    System.out.println("Enter Correct artist_sn");
            }
    }
}
```

ArtistProc.java

```
package Process;
import java.io.File;
import java.io.IOException;
import java.sql.SQLException;
import java.util.Collections;
import java.util.Formatter;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
```

```java
import java.util.Scanner;
import java.util.Set;
import java.util.TreeMap;
import java.util.TreeSet;
import DataAccess.ArtistDAO;
import DataTransfer.ArtistDTO;
/**
 * Create this object in the main and use this class's method. In this class,
 * After receiving user input within the method, perform the query using the DAO
 * object.
 *
 * @author Inryu Shin
 *
 */
public class ArtistProc {
    // 멤버변수
    // int isExists=2; //이름 존재여부 확인하는 flag 변수
    ArtistDAO dao; // ArtistDAO를 멤버변수로 가짐
    static public Scanner input = new Scanner(System.in);
    /**
     * default constructor
     */
    public ArtistProc() {
            dao = new ArtistDAO();
    }
    /**
     * Take artist_name as parameter and take input from user of debut date, type,
     * gender. Create DTO object using those values and call the DAO object's
     * insertArtist function using DTO.
     *
     * @param artist_name
     */
    public void insertArtist(String artist_name) { // 인자로 이름을 받음
            System.out.println("\nEnter artist information.");
            // System.out.print("name : ");
            // String name = input.nextLine();
            System.out.print("▶ debut date : ");
            String debut_date = input.nextLine();
            System.out.print("▶ type : ");
            String type = input.nextLine();
            System.out.print("▶ gender: ");
            String gender = input.nextLine();
            // Artistdto 객체 생성 (입력 받은 값과 생성자 이용)
            ArtistDTO artist = new ArtistDTO(artist_name, debut_date, type, gender);
            boolean r = dao.insertArtist(artist); // 입력받은 데이터 추가
            if (r) {
                    System.out.println("Input is entered successfully.");
            } else {
                    System.out.println("Failed. Input is not entered");
            }
    }
    /**
     *
     * Take artist_name as parameter and and call the DAO object's IsExists
     * function.
     *
     * @param artist_name
     */
    public int IsExists(String artist_name) {
            int n = dao.IsExists(artist_name);
            int result;
            if (n == 0)
                    result = 0;
            else
                    result = 1;
            return result;
    }
    /**
     * Take artist_name as parameter and call the DAO object's selectSN function.
     *
     * @param artist_name
```

```
     * @return
     */
    public int selectSN(String artist_name) {
            int n = dao.selectSN(artist_name);
            return n;
    }
    /**
     * Call the DAO object's getArtistsList function into List<ArtistDTO> list and
     * then print that list.
     */
    public void showAritstList() {
            List<ArtistDTO> list = dao.getArtistList();
            System.out.println("                                <Artist List>");

    System.out.println("=========================================================================
======");
            if (list != null && list.size() > 0) {
                    System.out.println("artist_sn\t  name \t\tdebut_date\ttype\t\tgender");

    System.out.println("=========================================================================
======");
                    for (ArtistDTO dto : list) {
                            System.out.println(dto);
                    }
            } else {
                    System.out.println("저장된 데이터가 없습니다. ");
            }

    System.out.println("=========================================================================
======");
    }
    /**
     *
     * Using type parameter , Call the DAO object's gettypeAtistsList function into
     * List<ArtistDTO> list and then print that list.
     *
     * @param type
     */
    public void typeAritstList(String type) {
            List<ArtistDTO> list = dao.gettypeArtist(type);
            System.out.println("                                <Artist List>");

    System.out.println("=========================================================================
======");
            if (list != null && list.size() > 0) {
                    System.out.println("artist_sn\t  name \t\tdebut_date\ttype\t\tgender");

    System.out.println("=========================================================================
======");
                    for (ArtistDTO dto : list) {
                            System.out.println(dto);
                    }
            } else {
                    System.out.println("저장된 데이터가 없습니다. ");
            }

    System.out.println("=========================================================================
======");
    }
    /**
     * Take input from user and create DTO object using those input values and call
     * the DAO object's updateArtist function using DTO.
     */
    public void updateArtist() {
            System.out.print("▶ Enter artist_sn you want to update : ");
            int current_artist_sn = input.nextInt();
            input.nextLine();
            ArtistDTO dto = dao.getArtist(current_artist_sn);
```

```
            if (dto != null) {
                    System.out.println("");
                    dto.printInfo();
                    System.out.print("▶ artist_sn : ");
                    int artist_sn = input.nextInt();
                    input.nextLine();
                    System.out.print("▶ name : ");
                    String name = input.nextLine();
                    System.out.print("▶ debut date : ");
                    String debut_date = input.nextLine();
                    System.out.print("▶ type : ");
                    String type = input.nextLine();
                    System.out.print("▶ gender : ");
                    String gender = input.nextLine();
                    dto = new ArtistDTO(artist_sn, name, debut_date, type, gender);
                    boolean r = dao.updateArtist(dto, current_artist_sn);
                    if (r) {
                            System.out.println("The artist's information has been modified as
follows.");
                            dto.printInfo();
                    } else {
                            System.out.println("The artist's information has not been modified
normally.");
                    }
            } else {
                    System.out.println("Enter correct artist_sn");
            }
    }
    /**
     * Call the DAO object's updateTransaction function by using parameter value.
     *
     * @param current_artist_sn
     * @param updated_artist_sn
     * @throws SQLException
     */
    public void updateTransaction(int current_artist_sn, int updated_artist_sn) throws SQLException {
            dao.updateTransaction(current_artist_sn, updated_artist_sn);
    }
    /**
     * Take input from user of artist_sn and call the DAO object's deleteArtist(sn)
     * function using input value.
     */
    public void deleteArtist() {
            System.out.print("▶ Enter artist_sn of Artist you want to delete :  ");
            int sn = input.nextInt();
            System.out.println("\n\n");
            ArtistDTO dto = dao.getArtist(sn);
            if (dto != null) {
                    dto.printInfo();
                    System.out.print("▶ Are you sure you want to delete it?(Y/N) : ");
                    input.nextLine();
                    String ans = input.nextLine();
                    if (ans.equalsIgnoreCase("y")) {
                            boolean r = dao.deleteArtist(sn);
                            if (r) {
                                    System.out.println("Aritst [ " + sn + " ] 's information has
been deleted successfully.");
                            } else {
                                    System.out.println("Failed.   Aritst   [   \"+sn+\"   ]   's
information has been not deleted.");
                            }
                    } else {
                            System.out.println("Deletion operation canceled.");
                    }
            } else {
                    System.out.println("Enter correct artist_sn");
```

```
            }
    }
}
```

SongProc.java

```
package Process;
import java.util.List;
import java.util.Scanner;
import DataAccess.SongDAO;
import DataTransfer.AlbumDTO;
import DataTransfer.ArtistDTO;
import DataTransfer.SongDTO;
/**
 * Create this object in the main and use this class's method. In this class,
 * After receiving user input within the method, perform the query using the DAO
 * object.
 *
 * @author Inryu Shin
 *
 */
public class SongProc {
    // 멤버 변수
    SongDAO dao;
    static public Scanner input = new Scanner(System.in);
    /**
     * default constructor
     */
    public SongProc() {
            dao = new SongDAO();
    }
    /**
     * Take song_title, album_sn, artist_sn as parameter and take input from user of genre.
     * Create DTO object using those values and call the DAO object's
     * insertSong function using DTO as parameter.
     * @param song_title
     * @param album_sn
     * @param artist_sn
     */
    public void insertSong(String song_title, int album_sn, int artist_sn) { // 인자로 이름을 받음
            System.out.println("\nEnter Song information.");
            System.out.print("▶ genre : ");
            String genre = input.nextLine();
            // SongDTO 객체 생성 (입력 받은 값과 생성자 이용)
            SongDTO song = new SongDTO(song_title, genre, album_sn, artist_sn);
            boolean r = dao.insertSong(song); // 입력받은 데이터 추가
            if (r) {
                    System.out.println("Input is entered successfully.");
            } else {
                    System.out.println("Failed. Input is not entered");
            }
    }
    /**
     *  Take song_title as parameter and and call the DAO object's IsExists function.
     * @param song_title
     * @return
     */
    public int IsExists(String song_title) {
            int n = dao.IsExists(song_title);
            int result;
            if (n == 0)
                    result = 0;
            else
                    result = 1;
            return result;
    }
    /**
     * Call the DAO object's getSongList function into List<SongDTO>list and then print that
list.
```

```
     */
    public void showSongList() {
            List<SongDTO> list = dao.getSongList();
            System.out.println("\n                                <Song List>");

    System.out.println("=================================================================
======");
            if (list != null && list.size() > 0) {
                    System.out.println("song_sn\t\t title\t\t\tgenre\t  album_sn\tartist_sn");

    System.out.println("=================================================================
======");
                    for (SongDTO dto : list) {
                            System.out.println(dto);
                    }
            } else {
                    System.out.println("no data. ");
            }

    System.out.println("=================================================================
======");
    }


    /**
     * Call the DAO object's getjoinSongList function into List<SongDTO>list by using parameter
and then print that list.
     * @param artist_name
     * @param album_name
     */
    public void showjoinSongList(String artist_name, String album_name) {
            List<SongDTO> list = dao.getjoinSongList(artist_name, album_name);
            System.out.println("\n                                <Song List>");

    System.out.println("=================================================================
======");
            if (list != null && list.size() > 0) {
                    System.out.println("\t\t\t title\t\t\t\t   genre\t");

    System.out.println("=================================================================
======");
                    for (SongDTO dto : list) {
                            dto.printDTO();
                    }
            } else {
                    System.out.println("no data. ");
            }

    System.out.println("=================================================================
======");
    }
    /**
     * Take input from user of song_sn and call the DAO object's
     * deleteSong(sn) function using input value.
     */
    public void deleteSong() {
            System.out.print("▶ Enter song_sn of Song you want to delete :  ");
            int sn = input.nextInt();
            System.out.println("\n\n");
            SongDTO dto = dao.getSong(sn);
            if (dto != null) {
                    dto.printInfo();
                    System.out.print("▶ Are you sure you want to delete it?(Y/N) : ");
                    input.nextLine();
                    String ans = input.nextLine();
                    if (ans.equalsIgnoreCase("y")) {
```

```
                                boolean r = dao.deleteSong(sn);
                                if (r) {
                                        System.out.println("Song [ " + sn + " ] 's information has
been deleted successfully.");
                                } else {
                                        System.out.println("Failed.   Aritst   [   \"+sn+\"   ]   's
information has been not deleted.");
                                }
                        } else {
                                System.out.println("Deletion operation canceled.");
                        }
                } else {
                        System.out.println("Enter Correct artist_sn");
                }
        }
}
```