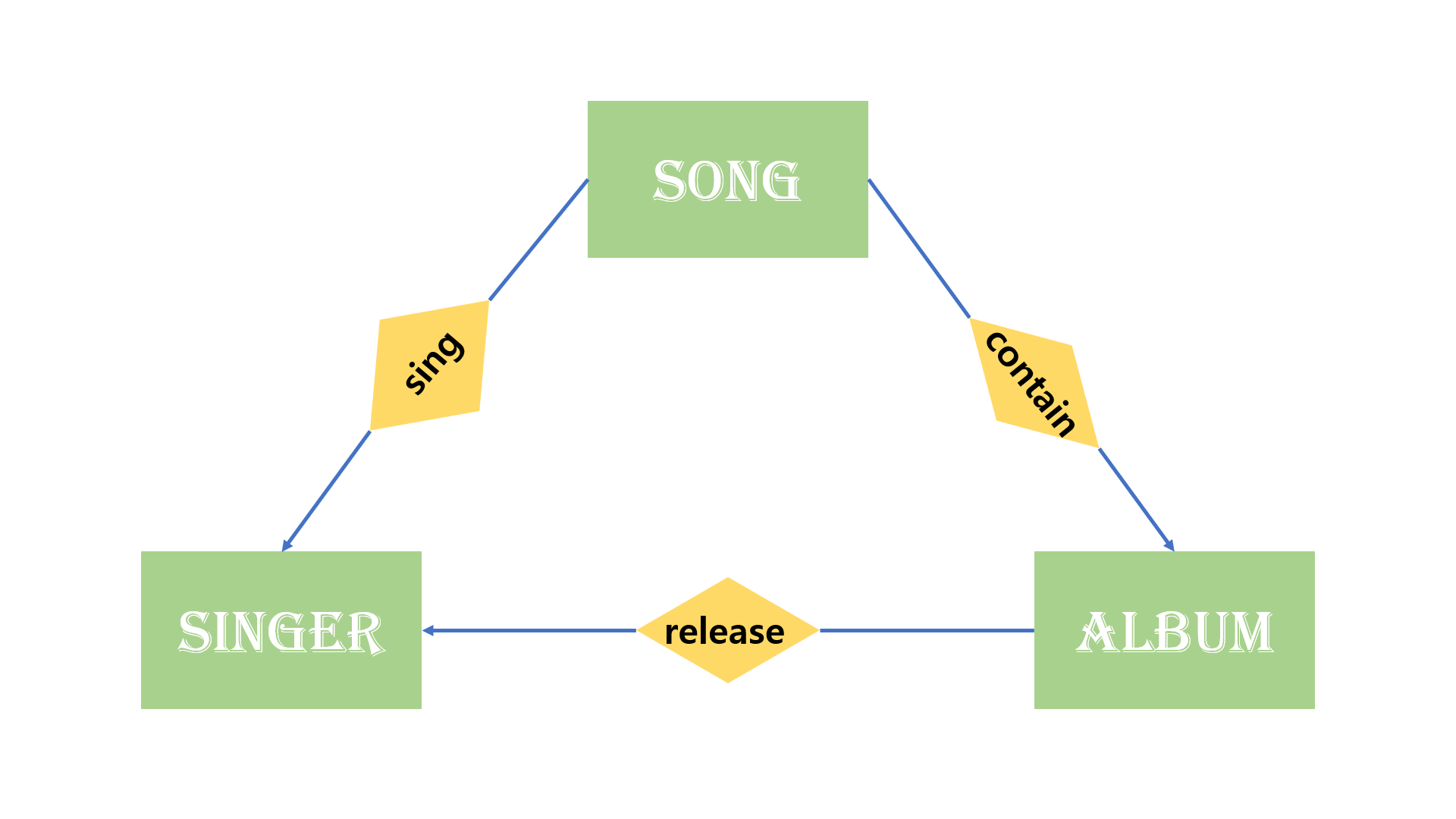
DATABASE

Project report

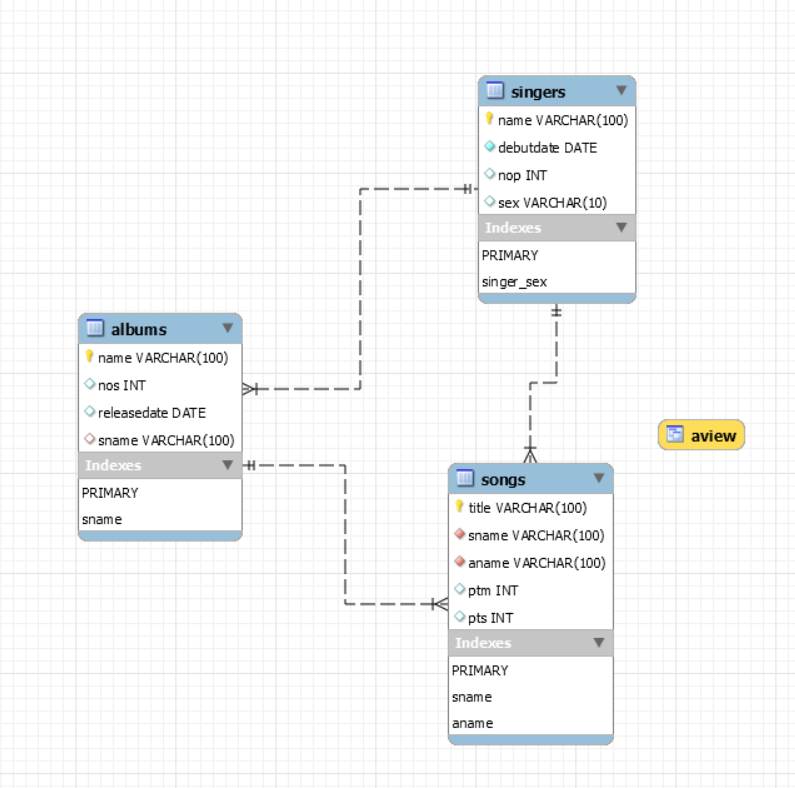
1871009

Kim Ye-eun

1. **ER Diagram**

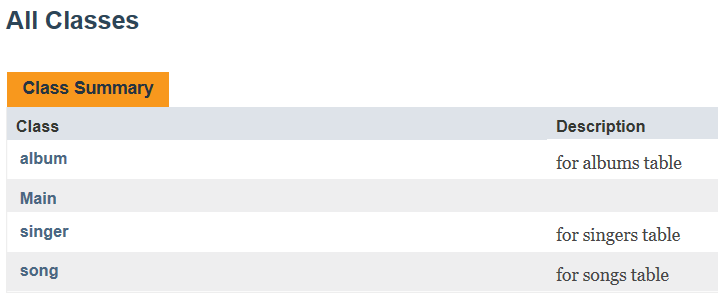


1. **Database schema diagram** (I used MySQL workbench program.)

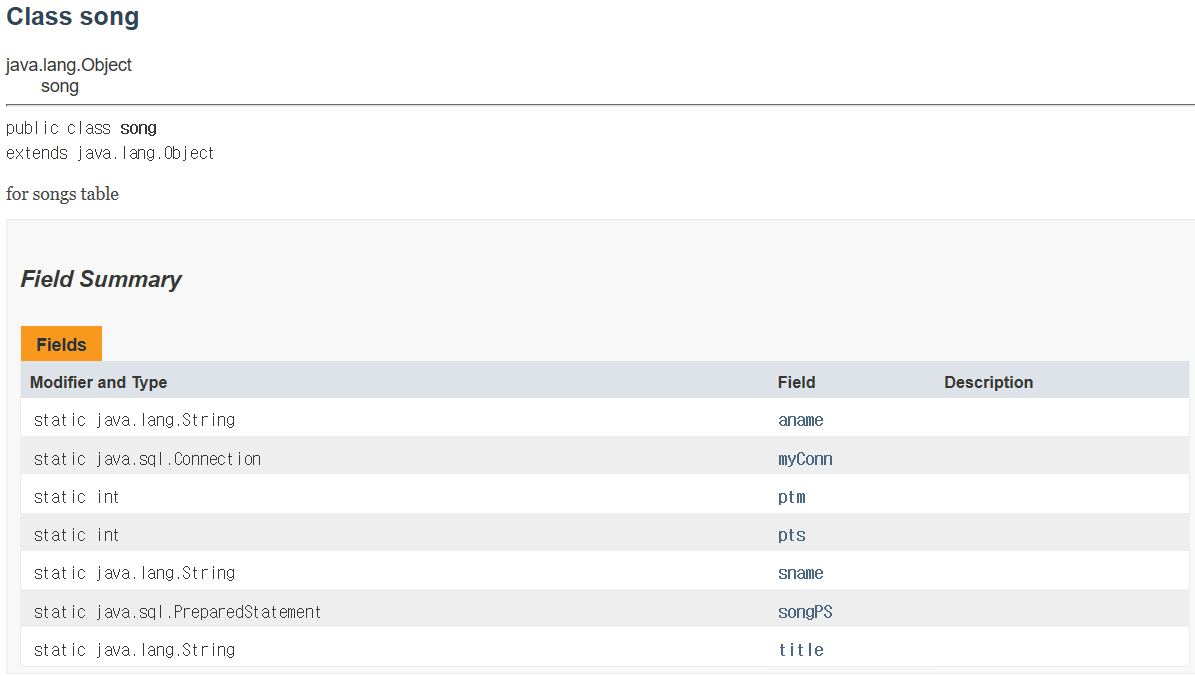


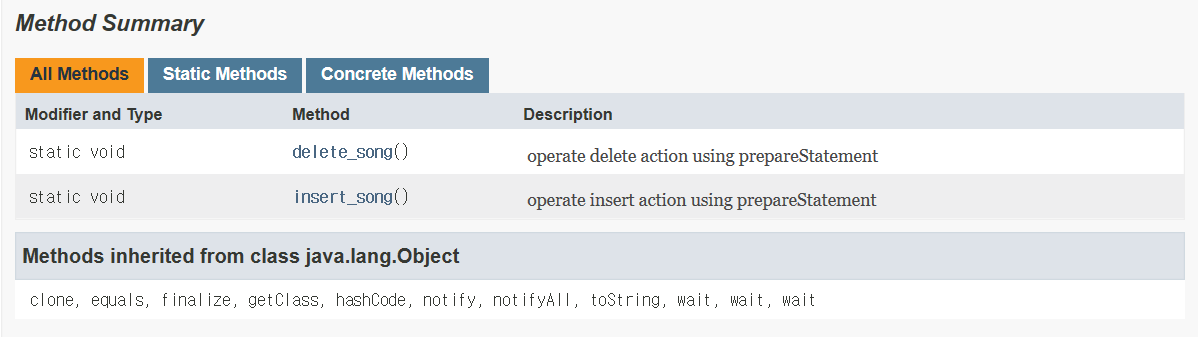
1. **Class and method**

**Classes**

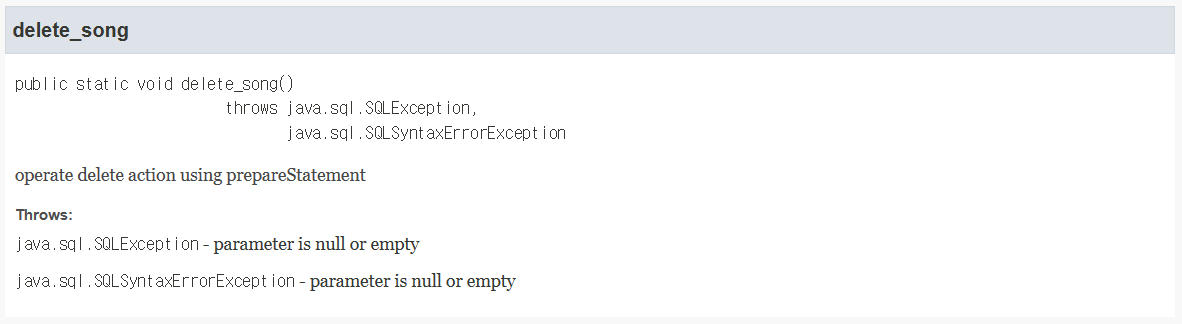


**Class song**

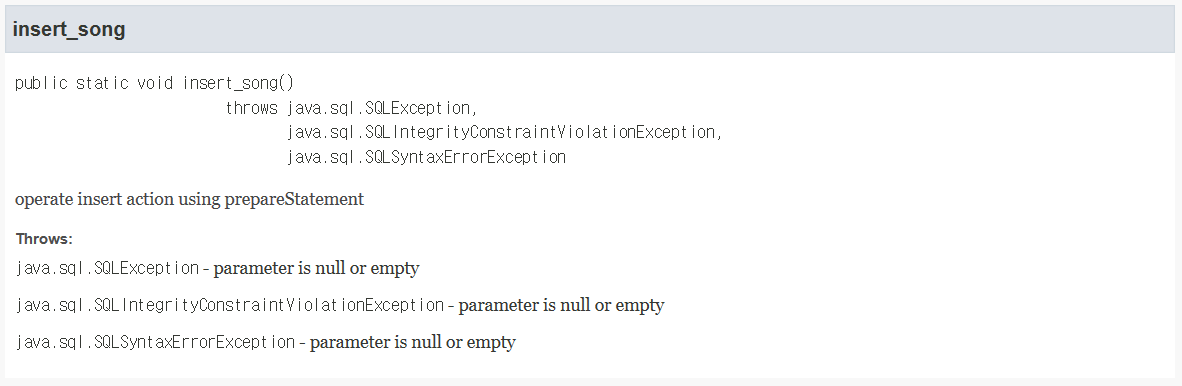




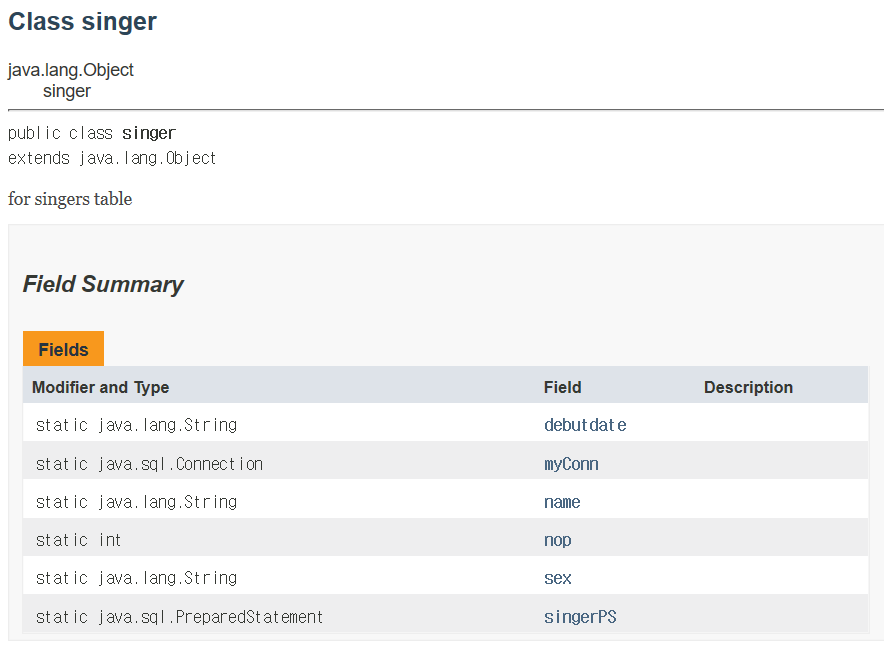
**song.delete\_song() method**

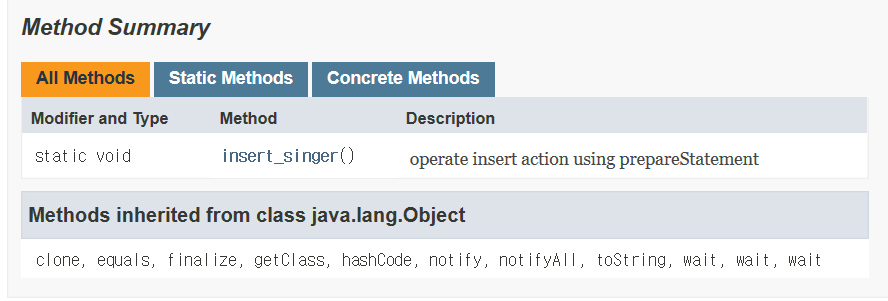


**Song.insert\_song() method**

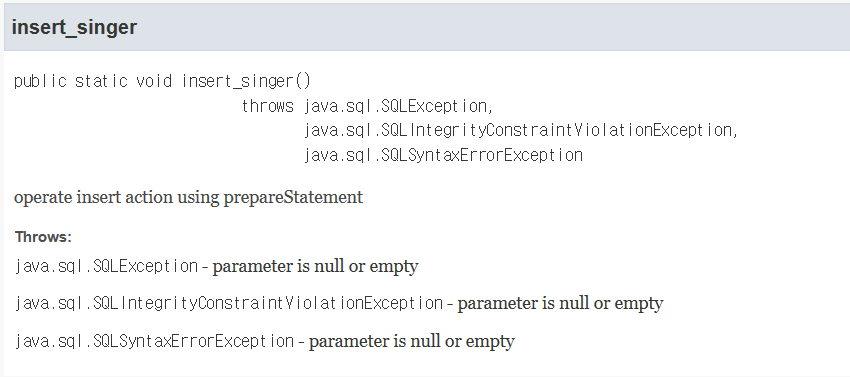


**Class singer**

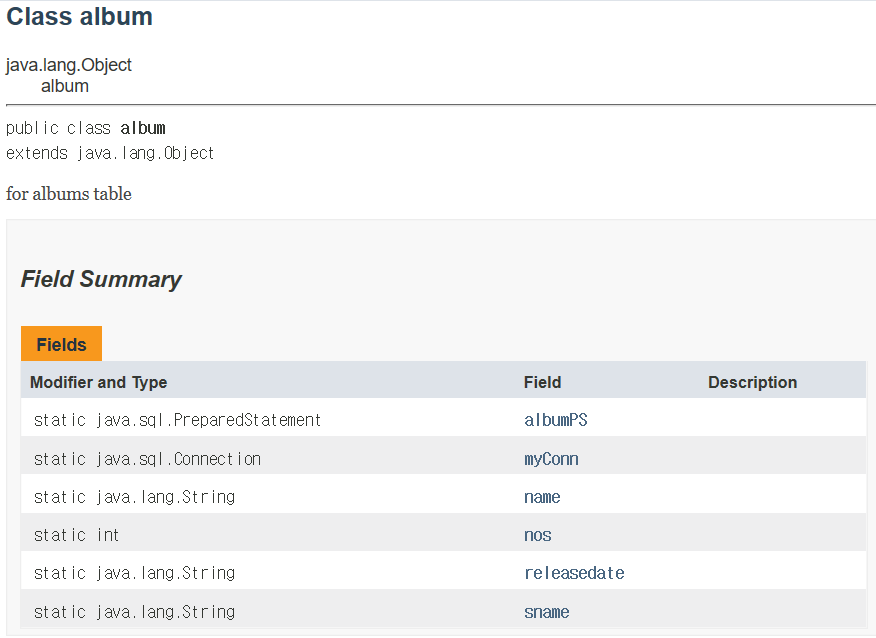
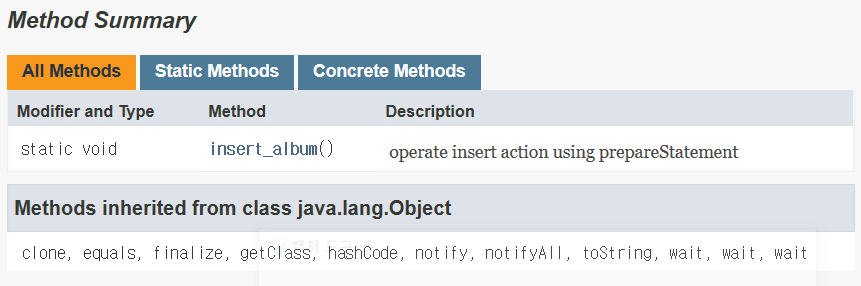




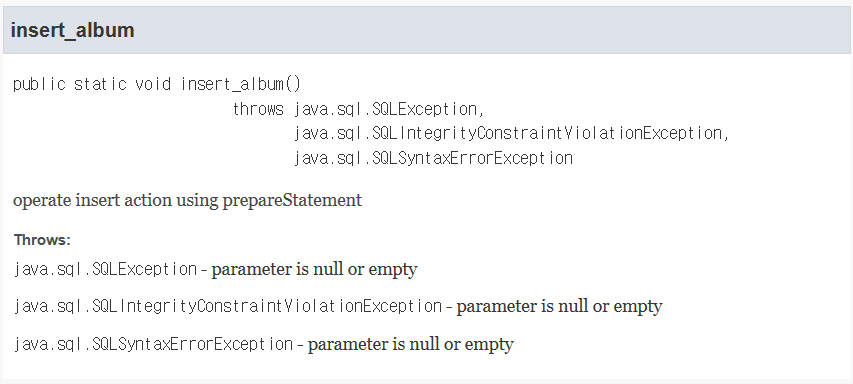
**Singer.Insert\_singer() method**



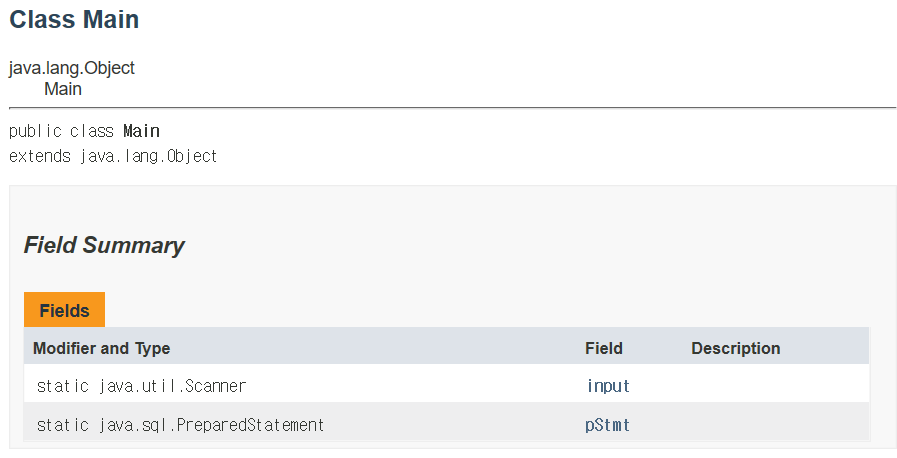
**Class album**

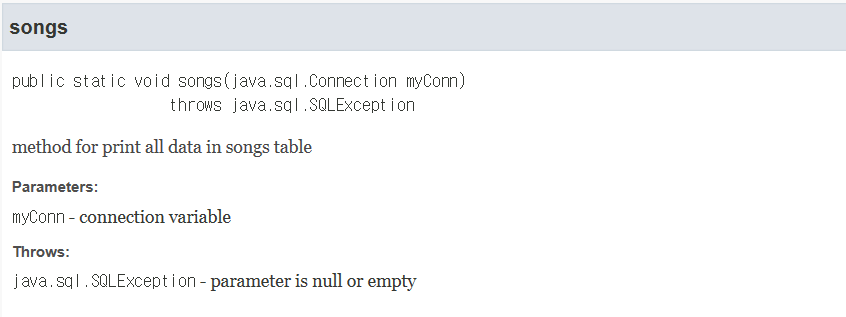
**Album.insert\_album() method**



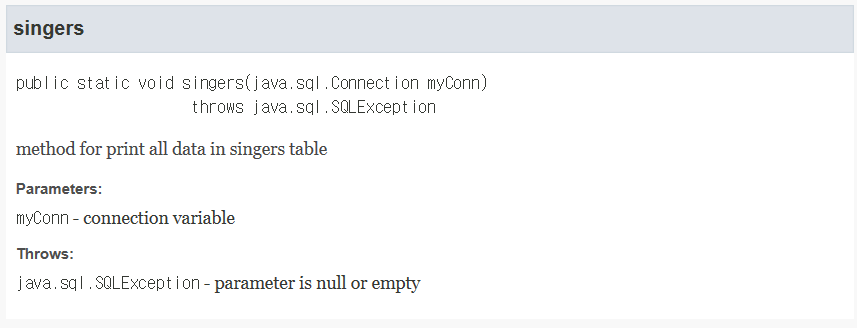
**Main class**

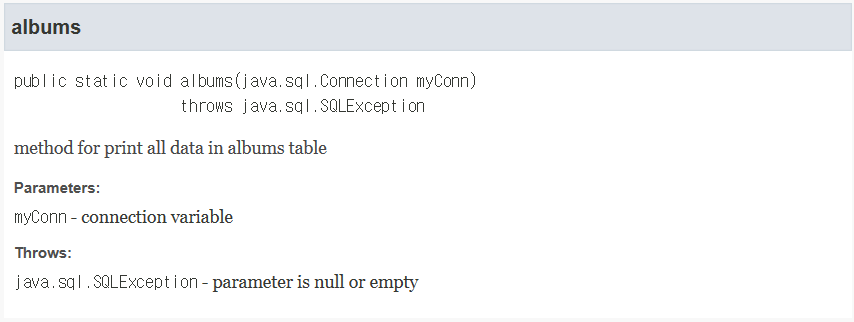
**Songs method**



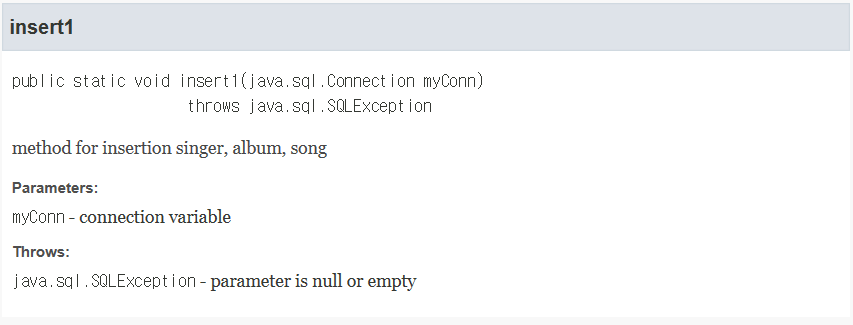
**singers method**



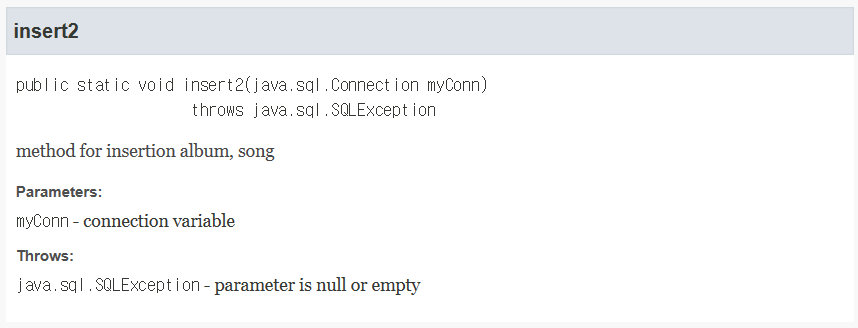
**Albums method**



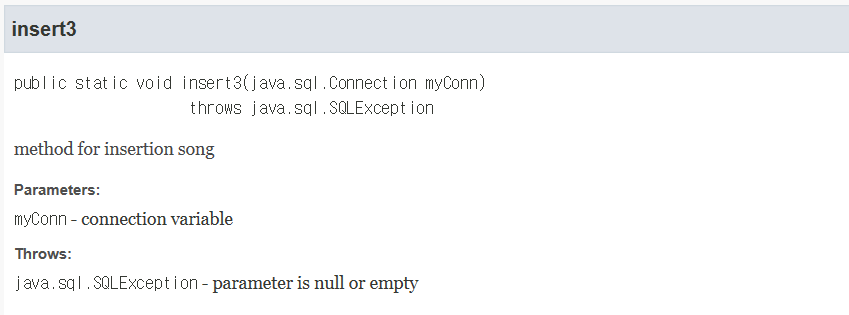
**Insert1 method**



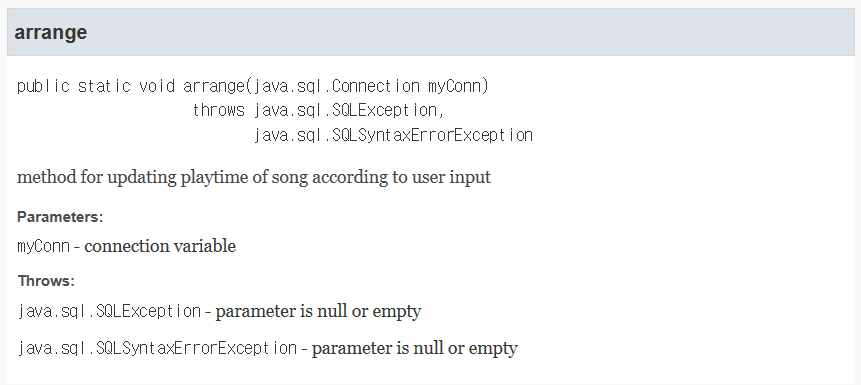
**Insert2 method**



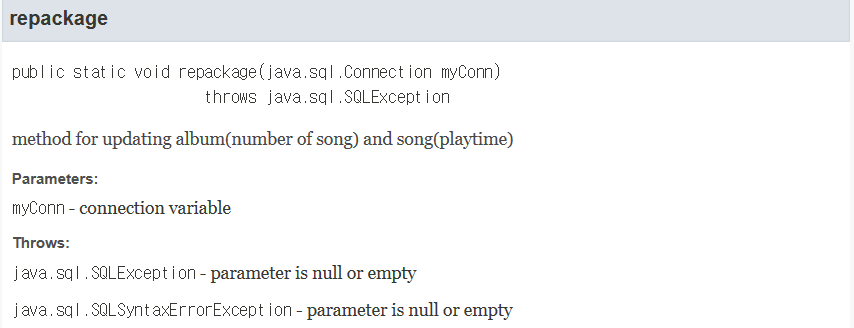
**Insert3 method**



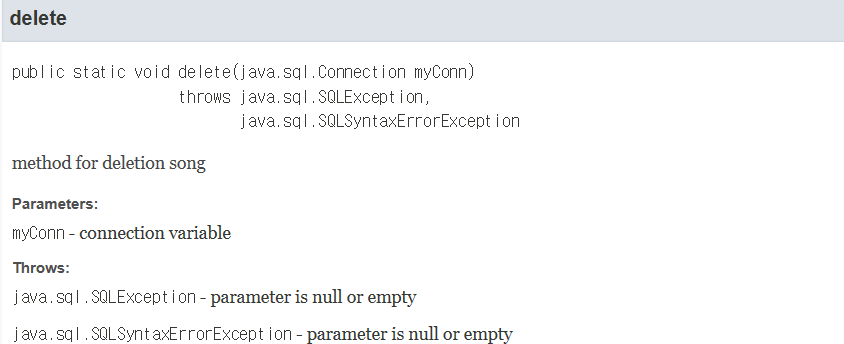
**Arrange method**



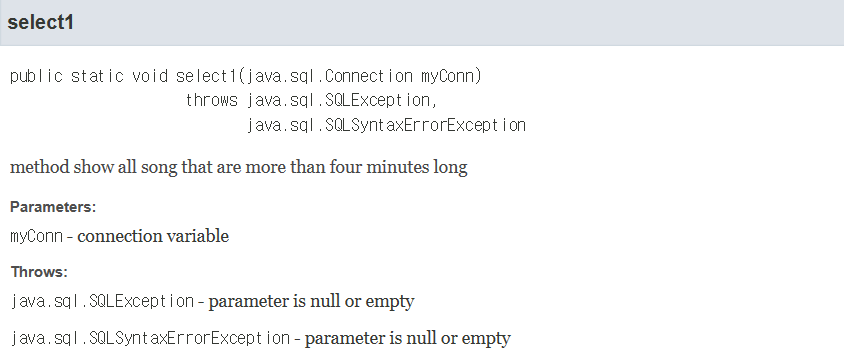
**Repackage method**



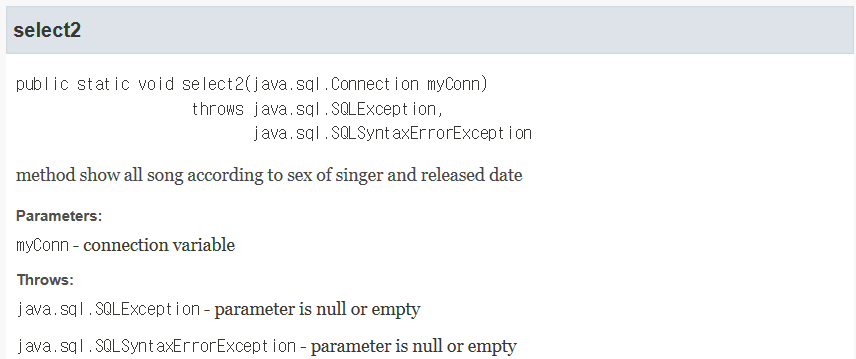
**Delete method**



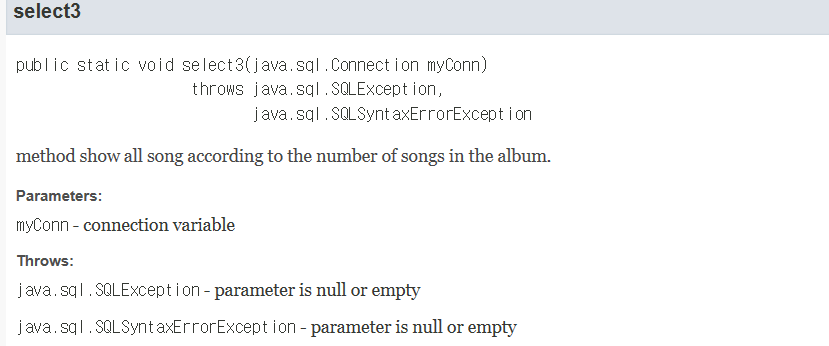
**Select1 method**



**Select2 method**



**Select3 method**



1. **Main class name** – Main class name is Main

**How to run** – save all class in same package, and run Main class.

**Connection configuration** – localhost:3306, database=finalproject, userid=testuser, password=testpw

1. **16 requirements**

**1 requirement –** I create three table(singers, albums and songs). Each table has 4 columns. singers - name, debutdate, nop,sex / albums – name, now, releasedate, sname / songs – title, sname, aname, ptm, pts

#가수 정보 저장할 table 생성/nop는 number of people 구성 인원 수/primary key는 가수의 name

create table singers(name varchar(100) not null, debutdate DATE not null, nop int, sex varchar(10), primary key (name));

#앨범 정보 저장할 table 생성/nos는 number of song 수록된 곡 수/primary key는 앨범의 이름, sname은 singers table 참조하는 foreign key

create table albums(name varchar(100) not null, nos int, releasedate DATE, sname varchar(100), primary key (name), foreign key (sname) references singers(name));

#곡 내용 저장할 table 생성/ptm은 playtime 중 minute, pts는 playtime 중 second

#primary key는 title/sname과 aname은 각각 singers table과 albums table을 참조하는 foreign key

create table songs(title varchar(100) not null, sname varchar(100) not null, aname varchar(100) not null, ptm int, pts int, primary key(title),

foreign key (sname) references singers(name), foreign key (aname) references albums(name));

**2 requirement –** I inserted 37 records for initialization. singers – 5, albums – 12, songs – 20

#singers table부터 insert

insert into singers values ('IU', '2008-09-23', 1, 'female');

insert into singers values ('AKMU', '2014-04-07', 2, 'both');

insert into singers values ('RedVelvet', '2014-08-01', 5, 'female');

insert into singers values ('SHINEE', '2008-05-25', 5, 'male');

insert into singers values ('JungSeunghwan', '2016-11-29', 1, 'male');

#그 다음 albums table insert

insert into albums values ('Love poem', 6, '2019-11-18', 'IU');

insert into albums values ('Modern Times', 13, '2013-10-08', 'IU');

insert into albums values ('heart', 1, '2015-05-18', 'IU');

insert into albums values ('adolescence up', 6, '2016-05-04', 'AKMU');

insert into albums values ('voyage', 10, '2019-09-25', 'AKMU');

insert into albums values ('The red', 10, '2015-09-09', 'RedVelvet');

insert into albums values ('Russian Roulette', 7, '2016-09-07', 'RedVelvet');

insert into albums values ('The Reve Festival Finale', 16, '2019-12-23', 'RedVelvet');

insert into albums values ('Sherlock', 7, '2012-03-19', 'SHINEE');

insert into albums values ('The Misconception of Me', 9, '2013-04-26', 'SHINEE');

insert into albums values ('Odd', 11, '2015-05-18', 'SHINEE');

insert into albums values ('voice', 6, '2016-11-29', 'JungSeunghwan');

#마지막으로 songs table insert

insert into songs values ('Blueming', 'IU', 'Love poem', 3, 37);

insert into songs values ('Modern Times', 'IU', 'Modern Times', 3, 26);

insert into songs values ('Love poem', 'IU', 'Love poem', 4, 18);

insert into songs values ('heart', 'IU', 'heart', 2, 47);

insert into songs values ('red shoes', 'IU', 'Modern Times', 4, 14);

insert into songs values ('that people moving around', 'AKMU', 'adolescence up', 3, 23);

insert into songs values ('RE-BYE', 'AKMU', 'adolescence up', 3, 9);

insert into songs values ('how can i love a breakup, i love you', 'AKMU', 'voyage', 4, 50);

insert into songs values ('Dumb Dumb', 'RedVelvet', 'The Red', 3, 23);

insert into songs values ('Huff n Puff', 'RedVelvet', 'The Red', 3, 1);

insert into songs values ('Psycho', 'RedVelvet', 'The Reve Festival Finale', 3, 31);

insert into songs values ('In&Out', 'RedVelvet', 'The Reve Festival Finale', 3, 13);

insert into songs values ('Remember Forever', 'RedVelvet', 'The Reve Festival Finale', 3, 8);

insert into songs values ('Russian Roulette', 'RedVelvet', 'Russian Roulette', 3, 31);

insert into songs values ('View', 'SHINEE', 'Odd', 3, 11);

insert into songs values ('Sherlock', 'SHINEE', 'Sherlock', 3, 57);

insert into songs values ('Alarm Clock', 'SHINEE', 'Sherlock', 3, 59);

insert into songs values ('Why So Serious?', 'SHINEE', 'The Misconception of Me', 3, 40);

insert into songs values ('you are fool', 'JungSeunghwan', 'voice', 4, 0);

insert into songs values ('the winter', 'JungSeunghwan', 'voice', 3, 57);

**3 requirement –** primary key: name – singers / name – albums / title – songs

foreign key: sname – songs(references singers table) / aname – songs(references albums table) / sname – albums(references singers table)

not null: all primary keys are not null and debutdate column in singers table if not null

**4 requirement –** I create releasedate column just in albums table. Songs table doesn’t has releasedate column.

**5 requirement –** I created index for sex of singers table.

*create index singer\_sex on singers(sex);*

**6 requirement –** I createdview named ‘aview’.

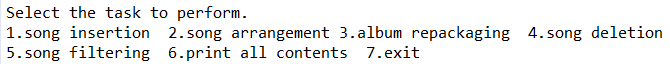
*create view aview as*

*select name, nos, title*

*from albums, songs*

*where albums.name=songs.aname;*

**7 requirement –** I used PreparedStatement to query by parameterized variables.



**8 requirement –** if user choose ‘song insertion’,program get information of song. According to condition, program may get information of singer and album. Then, program insert data I tables.

insert into singers values(?,?,?,?)

insert into albums values(?,?,?,?)

insert into songs values(?,?,?,?,?)

**public** **static** **void** insert1(Connection myConn) **throws** SQLException{//singer,album,song

String sname, aname, title, debutdate, releasedate, sex;

**int** nos, nop, ptm, pts;

//get information of singer, album, song

*input*.nextLine();

System.***out***.print("name of singer:");

sname=*input*.nextLine();

System.***out***.print("name of album:");

aname=*input*.nextLine();

System.***out***.print("title of song:");

title=*input*.nextLine();

System.***out***.print("playtime minute of song:");

ptm=*input*.nextInt();

System.***out***.print("playtime second of song:");

pts=*input*.nextInt();

*input*.nextLine();

System.***out***.print("debut date of singer:");

debutdate=*input*.nextLine();

System.***out***.print("released date of album:");

releasedate=*input*.nextLine();

System.***out***.print("sex of singer:");

sex=*input*.nextLine();

System.***out***.print("number of people belong to singer:");

nop=*input*.nextInt();

System.***out***.print("number of song belong to album:");

nos=*input*.nextInt();

**new** singer(sname,debutdate,nop,sex,myConn);

**new** album(aname,nos,sname,releasedate,myConn);

**new** song(title,sname,aname,ptm,pts,myConn);

singer.*insert\_singer*();//insert singer

album.*insert\_album*();//insert album

song.*insert\_song*();//insert song

System.***out***.println("insertion finished.");

}

/\*\*

\* method for insertion album, song

\* **@param** myConn connection variable

\* **@throws** SQLException parameter is null or empty

\*/

**public** **static** **void** insert2(Connection myConn) **throws** SQLException {//album,song

String sname, aname, title, releasedate;

**int** nos, ptm, pts;

*input*.nextLine();

//get information of album and song

System.***out***.print("name of singer:");

sname=*input*.nextLine();

System.***out***.print("name of album:");

aname=*input*.nextLine();

System.***out***.print("title of song:");

title=*input*.nextLine();

System.***out***.print("playtime minute of song:");

ptm=*input*.nextInt();

System.***out***.print("playtime second of song:");

pts=*input*.nextInt();

*input*.nextLine();

System.***out***.print("released date of album:");

releasedate=*input*.nextLine();

System.***out***.print("number of song belong to album:");

nos=*input*.nextInt();

**new** album(aname,nos,sname,releasedate,myConn);

**new** song(title,sname,aname,ptm,pts,myConn);

album.*insert\_album*();//insert album

song.*insert\_song*();//insert song

System.***out***.println("insertion finished.");

}

/\*\*

\* method for insertion song

\* **@param** myConn connection variable

\* **@throws** SQLException parameter is null or empty

\*/

**public** **static** **void** insert3(Connection myConn) **throws** SQLException{

String sname, aname, title;

**int** ptm, pts;

//get information of song

*input*.nextLine();

System.***out***.print("name of singer:");

sname=*input*.nextLine();

System.***out***.print("name of album:");

aname=*input*.nextLine();

System.***out***.print("title of song:");

title=*input*.nextLine();

System.***out***.print("playtime minute of song:");

ptm=*input*.nextInt();

System.***out***.print("playtime second of song:");

pts=*input*.nextInt();

**new** song(title,sname,aname,ptm,pts,myConn);

song.*insert\_song*();//insert song

System.***out***.println("insertion finished.");

}

**9 requirement –** if user choose ‘song arrangement, program get title of song and sec of song to increase. Then program update playtime of songs table.

update songs set ptm=ptm+? where title=?

update songs set pts=pts+? where title=?

**public** **static** **void** arrange(Connection myConn) **throws** SQLException, SQLSyntaxErrorException {//playtime arrange

//create prepareStatement

*pStmt*=myConn.prepareStatement("update songs set ptm=ptm+? where title=?");

PreparedStatement pStmt2=myConn.prepareStatement("update songs set pts=pts+? where title=?");

String title;

**int** isec, imin, sec;

//get title of song to arrange

*input*.nextLine();

System.***out***.println("Enter the title of song that you want to arrange.");

title=*input*.nextLine();

//get sec of song to increase

System.***out***.println("Enter the second you want to increase.");

sec=*input*.nextInt();

//calculate min, sec to increase

imin=sec/60;

isec=sec%60;

//set parameters

*pStmt*.setInt(1, imin);

pStmt2.setInt(1, isec);

*pStmt*.setString(2, title);

pStmt2.setString(2, title);

//update

*pStmt*.executeUpdate();

pStmt2.executeUpdate();

//close prepareStatement

*pStmt*.close();

pStmt2.close();

System.***out***.println("arrangement finished.");

}

**10 requirement –** if userchoose album repackaging, program get name of album to repackage, new number of song will be contained in repackage album and sec of song to increase. Then program update playtime of songs table and nos of albums table.

update albums set nos=? where name=?’

update songs set ptm=ptm+? where aname=?

update songs set pts=pts+? where aname=?

/\*\*

\* method for updating album(number of song) and song(playtime)

\* **@param** myConn connection variable

\* **@throws** SQLException parameter is null or empty

\* **@throws** SQLSyntaxErrorException parameter is null or empty

\*/

**public** **static** **void** repackage(Connection myConn) **throws** SQLException{

Savepoint savepoint1 = **null**;

**try** {

myConn.setAutoCommit(**false**);//for using transaction

String aname;

**int** sec, imin, isec, nos;

PreparedStatement p2;

PreparedStatement p3;

savepoint1 = myConn.setSavepoint("Savepoint1");

//create prepareStatement

*pStmt*=myConn.prepareStatement("update albums set nos=? where name=?");

p2=myConn.prepareStatement("update songs set ptm=ptm+? where aname=?");

p3=myConn.prepareStatement("update songs set pts=pts+? where aname=?");

//get information of repackaging

*input*.nextLine();

System.***out***.println("Enter the original name of album that you want to repackage.");

aname=*input*.nextLine();

System.***out***.println("Enter number of song to contain this album");

nos=*input*.nextInt();

//get sec of song to increase

System.***out***.println("Enter the second you want to increase.");

sec=*input*.nextInt();

//calculate min, sec to increase

imin=sec/60;

isec=sec%60;

//set parameter

*pStmt*.setInt(1, nos);

*pStmt*.setString(2, aname);

p2.setInt(1, imin);

p2.setString(2, aname);

p3.setInt(1, isec);

p3.setString(2, aname);

//update

*pStmt*.executeUpdate();

p2.executeUpdate();

p3.executeUpdate();

myConn.commit();//commit

myConn.setAutoCommit(**true**);//reset

//close prepareStatement

*pStmt*.close();

p2.close();

p3.close();

System.***out***.println("repackage finished.");

}**catch**(SQLException e) {

myConn.rollback(savepoint1);

}

}

**11 requirement –** if user choose song deletion, program get title of song to delete. Then program delete right song from songs table.

delete from songs where title=?

/\*\*

\* method for deletion song

\* **@param** myConn connection variable

\* **@throws** SQLException parameter is null or empty

\* **@throws** SQLSyntaxErrorException parameter is null or empty

\*/

**public** **static** **void** delete(Connection myConn) **throws** SQLException, SQLSyntaxErrorException{

String title;

//get title of song to delete

*input*.nextLine();

System.***out***.println("Enter the title of song that you want to delete.");

title=*input*.nextLine();

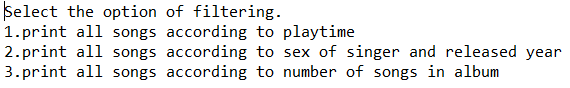
**new** song(title, myConn);

song.*delete\_song*();//delete song

System.***out***.println("deletion finished.");

}

***If user choose song filtering, there are three options.***



**12 requirement –** if user choose 1 option, program get lower bound of minute. Then program show all songs that over n minutes.

**public** **static** **void** select1(Connection myConn) **throws** SQLException, SQLSyntaxErrorException {

*pStmt*=myConn.prepareStatement("select \* from songs where ptm>? or ptm=?");//create prepareStatement

ResultSet rs;

**int** m, ptm, pts;

String title, sname, aname;

//get lower bound

System.***out***.println("enter the lower bound of minute.");

m=*input*.nextInt();

*pStmt*.setInt(1, m);

*pStmt*.setInt(2, m);

rs=*pStmt*.executeQuery();//query

System.***out***.println();

System.***out***.println("list of result data");

//print all songs that equal or over n minute

**while**(rs.next()) {

title=rs.getString("title");

sname=rs.getString("sname");

aname=rs.getString("aname");

ptm=rs.getInt("ptm");

pts=rs.getInt("pts");

System.***out***.println(String.*format*("title:%40s|singer:%15s|album:%30s|playtime:%4dm%4ds",title, sname,aname,ptm,pts));

}

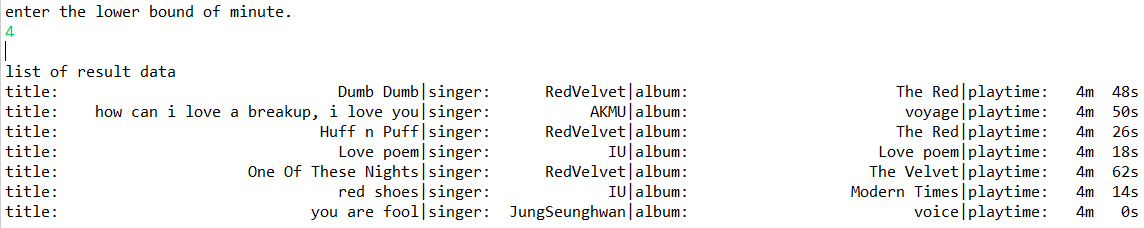
//close resultset and prepareStatement

rs.close();

*pStmt*.close();

}

select \* from songs where ptm>? or ptm=?



**13 requirement** – if user choose 2 option, program get sex of singer and year of released date. Then program show all songs that meet conditions. In that process, I inner join albums table and songs table. Also I used nested query in where clause to filter sex of singer and released year of album.

**public** **static** **void** select2(Connection myConn) **throws** SQLException, SQLSyntaxErrorException {

*pStmt*=myConn.prepareStatement("select title, releasedate "

+ "from songs inner join albums on songs.aname=albums.name "

+ "where (releasedate between ? and ?) and "

+ "(songs.sname=some(select name from singers where sex=?))");//create prepareStatement

ResultSet rs;

**int** a;

String title, releasedate, begindate, enddate, sex = **null**;

//get information of sex

System.***out***.println("Select sex of singer. 1.female 2.male 3.both");

a=*input*.nextInt();

**if**(a==1) sex="female";

**else** **if**(a==2) sex="male";

**else** **if**(a==3) sex="both";

*input*.nextLine();

//get year of released date

System.***out***.println("Enter the year of released date.");

releasedate=*input*.nextLine();

begindate=releasedate+"-01-01";

enddate=releasedate+"-12-31";

//set parameter

*pStmt*.setString(1, begindate);

*pStmt*.setString(2, enddate);

*pStmt*.setString(3, sex);

rs=*pStmt*.executeQuery();//query

System.***out***.println();

System.***out***.println("list of result data");

//print title and released date according to sex of singer and released date

**while**(rs.next()) {

title=rs.getString("title");

releasedate=rs.getString("releasedate");

System.***out***.println(String.*format*("title: %40s|released date: %10s",title, releasedate));

}

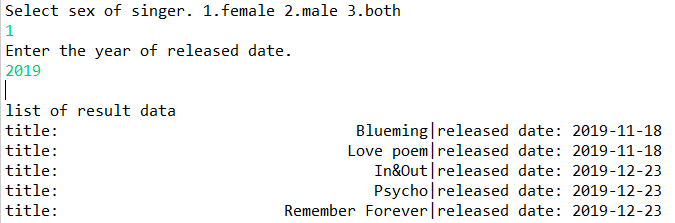
//close resultSet and prepareStatement

rs.close();

*pStmt*.close();

}

select title, releasedate from songs inner join albums on songs.aname =albums.name where (releasedate between ? and ?) and (songs.sname=some(select name from singers where sex=?))



**14 requirement –** if user choose 3 option, program get lower bound number of songs. And show all songs that are belong to album contains songs over n and number song in this album. When I get title of song, number of songs in one album and name of album, I used view that I created in **6 requirement.**

**public** **static** **void** select3(Connection myConn) **throws** SQLException, SQLSyntaxErrorException {

*pStmt*=myConn.prepareStatement("select \* from aview where nos>? or nos=?");//create prepareStatement

ResultSet rs;

**int** nos, nos2;

String title, aname;

//get number of song

System.***out***.println("enter lower bound of number of song belong to one album.");

nos=*input*.nextInt();

//set parameter

*pStmt*.setInt(1, nos);

*pStmt*.setInt(2, nos);

rs=*pStmt*.executeQuery();//query

System.***out***.println();

System.***out***.println("list of result data");

//print songs that are belong to album contains songs over n and number of song

**while**(rs.next()) {

title=rs.getString("title");

nos2=rs.getInt("nos");

aname=rs.getString("name");

System.***out***.println(String.*format*("album: %30s|title: %40s|number of songs: %2d", aname, title, nos2));

}

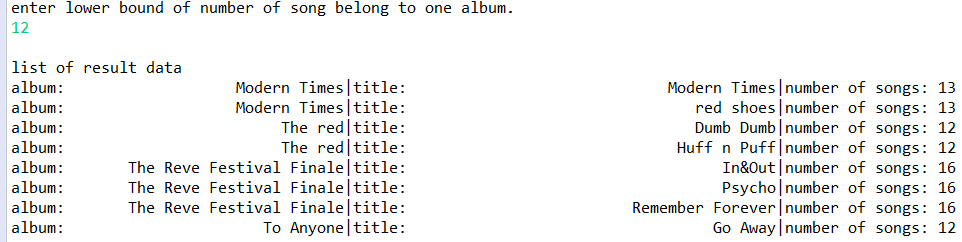
//close resultSet and prepareStatement

rs.close();

*pStmt*.close();

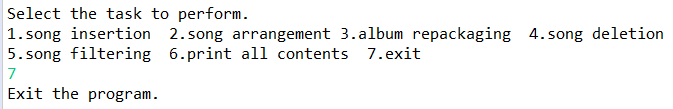
}

select \* from **aview** where nos>? or nos=?



**15 requirement –** if user choose ‘print all contents’, program show all data from albums, songs and singers table. I used three method to get all data from three tables.

**16 requirement –** if user choose ‘exit’, she can exit the program.



1. **SQL scripts**

**create.sql**

#가수 정보 저장할 table 생성/nop는 number of people 구성 인원 수/primary key는 가수의 name

create table singers(name varchar(100) not null, debutdate DATE not null, nop int, sex varchar(10), primary key (name));

#앨범 정보 저장할 table 생성/nos는 number of song 수록된 곡 수/primary key는 앨범의 이름, sname은 singers table 참조하는 foreign key

create table albums(name varchar(100) not null, nos int, releasedate DATE, sname varchar(100), primary key (name), foreign key (sname) references singers(name));

#곡 내용 저장할 table 생성/ptm은 playtime 중 minute, pts는 playtime 중 second

#primary key는 title/sname과 aname은 각각 singers table과 albums table을 참조하는 foreign key

create table songs(title varchar(100) not null, sname varchar(100) not null, aname varchar(100) not null, ptm int, pts int, primary key(title),

foreign key (sname) references singers(name), foreign key (aname) references albums(name));

#singers table의 name column index 생성

create index singer\_sex on singers(sex);

#수록곡이 일정 수 이상을 넘는 앨범의 이름과 수록곡을 편하게 select하기 위해 관련된 값들만 보이도록 하는 view 생성

create view aview as

select name, nos, title

from albums, songs

where albums.name=songs.aname;

#singers table부터 insert

insert into singers values ('IU', '2008-09-23', 1, 'female');

insert into singers values ('AKMU', '2014-04-07', 2, 'both');

insert into singers values ('RedVelvet', '2014-08-01', 5, 'female');

insert into singers values ('SHINEE', '2008-05-25', 5, 'male');

insert into singers values ('JungSeunghwan', '2016-11-29', 1, 'male');

#그 다음 albums table insert

insert into albums values ('Love poem', 6, '2019-11-18', 'IU');

insert into albums values ('Modern Times', 13, '2013-10-08', 'IU');

insert into albums values ('heart', 1, '2015-05-18', 'IU');

insert into albums values ('adolescence up', 6, '2016-05-04', 'AKMU');

insert into albums values ('voyage', 10, '2019-09-25', 'AKMU');

insert into albums values ('The red', 10, '2015-09-09', 'RedVelvet');

insert into albums values ('Russian Roulette', 7, '2016-09-07', 'RedVelvet');

insert into albums values ('The Reve Festival Finale', 16, '2019-12-23', 'RedVelvet');

insert into albums values ('Sherlock', 7, '2012-03-19', 'SHINEE');

insert into albums values ('The Misconception of Me', 9, '2013-04-26', 'SHINEE');

insert into albums values ('Odd', 11, '2015-05-18', 'SHINEE');

insert into albums values ('voice', 6, '2016-11-29', 'JungSeunghwan');

#마지막으로 songs table insert

insert into songs values ('Blueming', 'IU', 'Love poem', 3, 37);

insert into songs values ('Modern Times', 'IU', 'Modern Times', 3, 26);

insert into songs values ('Love poem', 'IU', 'Love poem', 4, 18);

insert into songs values ('heart', 'IU', 'heart', 2, 47);

insert into songs values ('red shoes', 'IU', 'Modern Times', 4, 14);

insert into songs values ('that people moving around', 'AKMU', 'adolescence up', 3, 23);

insert into songs values ('RE-BYE', 'AKMU', 'adolescence up', 3, 9);

insert into songs values ('how can i love a breakup, i love you', 'AKMU', 'voyage', 4, 50);

insert into songs values ('Dumb Dumb', 'RedVelvet', 'The Red', 3, 23);

insert into songs values ('Huff n Puff', 'RedVelvet', 'The Red', 3, 1);

insert into songs values ('Psycho', 'RedVelvet', 'The Reve Festival Finale', 3, 31);

insert into songs values ('In&Out', 'RedVelvet', 'The Reve Festival Finale', 3, 13);

insert into songs values ('Remember Forever', 'RedVelvet', 'The Reve Festival Finale', 3, 8);

insert into songs values ('Russian Roulette', 'RedVelvet', 'Russian Roulette', 3, 31);

insert into songs values ('View', 'SHINEE', 'Odd', 3, 11);

insert into songs values ('Sherlock', 'SHINEE', 'Sherlock', 3, 57);

insert into songs values ('Alarm Clock', 'SHINEE', 'Sherlock', 3, 59);

insert into songs values ('Why So Serious?', 'SHINEE', 'The Misconception of Me', 3, 40);

insert into songs values ('you are fool', 'JungSeunghwan', 'voice', 4, 0);

insert into songs values ('the winter', 'JungSeunghwan', 'voice', 3, 57);

**dropdb.sql**

#모든 table(과 view)에 대해 table과 안에 있는 data들을 모두 지워버리는 명령어 drop 사용

drop view aview;

drop table songs;

drop table albums;

drop table singers;

1. **Java codes**

**Song class**

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.SQLException;

**import** java.sql.SQLIntegrityConstraintViolationException;

**import** java.sql.SQLSyntaxErrorException;

/\*\*

\* for songs table

\*/

**public** **class** song {

**public** **static** String *title*;

**public** **static** String *sname*;

**public** **static** String *aname*;

**public** **static** **int** *ptm*;

**public** **static** **int** *pts*;

**public** **static** Connection *myConn*;

**public** **static** PreparedStatement *songPS*;

/\*\*

\* constructor of song class for delete action

\* **@param** title title of song

\* **@param** myConn connection variable

\*/

song(String title,Connection myConn){

song.*title*=title;

song.*myConn*=myConn;

}

/\*\*

\* constructor of song class for insert action

\* **@param** title title of song

\* **@param** sname singer

\* **@param** aname album

\* **@param** ptm playtime - min

\* **@param** pts playtime - sec

\* **@param** myConn connection variable

\*/

song(String title, String sname, String aname, **int** ptm, **int** pts,Connection myConn){

song.*title*=title;

song.*sname*=sname;

song.*aname*=aname;

song.*ptm*=ptm;

song.*pts*=pts;

song.*myConn*=myConn;

}

/\*\*

\* operate insert action using prepareStatement

\* **@throws** SQLException parameter is null or empty

\* **@throws** SQLIntegrityConstraintViolationException parameter is null or empty

\* **@throws** SQLSyntaxErrorException parameter is null or empty

\*/

**public** **static** **void** insert\_song() **throws** SQLException, SQLIntegrityConstraintViolationException, SQLSyntaxErrorException {

*songPS*=*myConn*.prepareStatement("insert into songs values(?,?,?,?,?)");

//set parameter

*songPS*.setString(1, *title*);

*songPS*.setString(2, *sname*);

*songPS*.setString(3, *aname*);

*songPS*.setInt(4, *ptm*);

*songPS*.setInt(5, *pts*);

*songPS*.executeUpdate();//insert song

*songPS*.close();//close prepareStatement

}

/\*\*

\* operate delete action using prepareStatement

\* **@throws** SQLException parameter is null or empty

\* **@throws** SQLSyntaxErrorException parameter is null or empty

\*/

**public** **static** **void** delete\_song() **throws** SQLException, SQLSyntaxErrorException{

*songPS*=*myConn*.prepareStatement("delete from songs where title=?");

*songPS*.setString(1, *title*);//set parameter

*songPS*.executeUpdate();//delete song

*songPS*.close();//close prepareStatement

}

}

**Singer class**

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.SQLException;

**import** java.sql.SQLIntegrityConstraintViolationException;

**import** java.sql.SQLSyntaxErrorException;

/\*\*

\* for singers table

\*/

**public** **class** singer {

**public** **static** String *name*;

**public** **static** String *debutdate*;

**public** **static** **int** *nop*;

**public** **static** String *sex*;

**public** **static** Connection *myConn*;

**public** **static** PreparedStatement *singerPS*;

/\*\*

\* constructor of singer class

\* **@param** name name of singer

\* **@param** debutdate debut date of singer

\* **@param** nop number of person of singer

\* **@param** sex sex of singer

\* **@param** myConn connection variable

\*/

singer(String name, String debutdate, **int** nop, String sex, Connection myConn){

singer.*name*=name;

singer.*debutdate*=debutdate;

singer.*nop*=nop;

singer.*sex*=sex;

singer.*myConn*=myConn;

}

/\*\*

\* operate insert action using prepareStatement

\* **@throws** SQLException parameter is null or empty

\* **@throws** SQLIntegrityConstraintViolationException parameter is null or empty

\* **@throws** SQLSyntaxErrorException parameter is null or empty

\*/

**public** **static** **void** insert\_singer() **throws** SQLException, SQLIntegrityConstraintViolationException, SQLSyntaxErrorException {

*singerPS*=*myConn*.prepareStatement("insert into singers values(?,?,?,?)");

//set parameter

*singerPS*.setString(1, *name*);

*singerPS*.setString(2, *debutdate*);

*singerPS*.setInt(3, *nop*);

*singerPS*.setString(4, *sex*);

*singerPS*.executeUpdate();//insert singer

*singerPS*.close();//close prepareStatement

}

}

**Album class**

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.SQLException;

**import** java.sql.SQLIntegrityConstraintViolationException;

**import** java.sql.SQLSyntaxErrorException;

/\*\*

\* for albums table

\*/

**public** **class** album {

**public** **static** String *name*;

**public** **static** **int** *nos*;

**public** **static** String *sname*;

**public** **static** String *releasedate*;

**public** **static** Connection *myConn*;

**public** **static** PreparedStatement *albumPS*;

/\*\*

\* constructor of album class

\* **@param** name name of album

\* **@param** nos number of song in album

\* **@param** sname name of singer

\* **@param** releasedate released date

\* **@param** myConn connection variable

\*/

album(String name, **int** nos, String sname, String releasedate,Connection myConn){

album.*name*=name;

album.*nos*=nos;

album.*sname*=sname;

album.*releasedate*=releasedate;

album.*myConn*=myConn;

}

/\*\*

\* operate insert action using prepareStatement

\* **@throws** SQLException parameter is null or empty

\* **@throws** SQLIntegrityConstraintViolationException parameter is null or empty

\* **@throws** SQLSyntaxErrorException parameter is null or empty

\*/

**public** **static** **void** insert\_album() **throws** SQLException, SQLIntegrityConstraintViolationException, SQLSyntaxErrorException {

*albumPS*=*myConn*.prepareStatement("insert into albums values(?,?,?,?)");

//set parameter

*albumPS*.setString(1, *name*);

*albumPS*.setInt(2, *nos*);

*albumPS*.setString(3, *releasedate*);

*albumPS*.setString(4, *sname*);

*albumPS*.executeUpdate();//insert album

*albumPS*.close();//close prepareStatement

}

}

**Main class**

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.sql.SQLIntegrityConstraintViolationException;

**import** java.sql.SQLSyntaxErrorException;

**import** java.sql.Savepoint;

**import** java.sql.Statement;

**import** java.util.Scanner;

**public** **class** Main {

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

**public** **static** PreparedStatement *pStmt*;

**public** **static** **void** main(String[] args) **throws** SQLException, SQLIntegrityConstraintViolationException, SQLSyntaxErrorException {

// **TODO** Auto-generated method stub

//userid=testuser, password=testpw, dbname=finalproject

String userID="testuser";

String userPW="testpw";

String dbName="finalproject";

String url="jdbc:mysql://localhost:3306/"+dbName+"?&serverTimezone=UTC";

Connection myConn=**null**;

**try** {

myConn=DriverManager.*getConnection*(url,userID,userPW);//getConnection

**while**(**true**) {

//select option

System.***out***.println();

System.***out***.println("Select the task to perform.");

System.***out***.println("1.song insertion 2.song arrangement 3.album repackaging 4.song deletion");

System.***out***.println("5.song filtering 6.print all contents 7.exit");

**int** a=*input*.nextInt();

**if**(a==7) {//when user choose exit

System.***out***.println("Exit the program.");

**break**;

}

**else** **if**(a==1) {//when user choose insertion

*singers*(myConn);//show singers list

System.***out***.println("Is there singer of song that you want to insert? 1.yes 2.no");

**int** syn=*input*.nextInt();

**if**(syn==1) {//if singer of song already exists

*albums*(myConn);

System.***out***.println("Is there album of song that you want to insert? 1.yes 2.no");

**int** ayn=*input*.nextInt();

**if**(ayn==1) *insert3*(myConn);//if singer and album of song already exists

**else** **if**(ayn==2) *insert2*(myConn);//if singer of song already exists but album doesn't

**else** System.***out***.println("Invalid input. Please enter again.");//wrong input

}

**else** **if**(syn==2) *insert1*(myConn);//if singer of song doesn't exist

**else** System.***out***.println("Invalid input. Please enter again.");//wrong input

}

**else** **if**(a==2) {//when user choose arrangement

*songs*(myConn);

*arrange*(myConn);

}

**else** **if**(a==3) {//when user choose repackaging

*albums*(myConn);

*repackage*(myConn);

}

**else** **if**(a==4) {//when user choose deletion

*songs*(myConn);

*delete*(myConn);

}

**else** **if**(a==5) {//when user choose filtering

//select option of filtering

System.***out***.println("Select the option of filtering.");

System.***out***.println("1.print all songs according to playtime");

System.***out***.println("2.print all songs according to sex of singer and released year");

System.***out***.println("3.print all songs according to number of songs in album ");

**int** oa=*input*.nextInt();

**if**(oa==1) *select1*(myConn);

**else** **if**(oa==2) *select2*(myConn);

**else** **if**(oa==3) *select3*(myConn);

**else** System.***out***.println("Invalid input. Please enter again.");//wrong input

}

**else** **if**(a==6) {//when user choose print

*songs*(myConn);//print songs

*singers*(myConn);//print singers

*albums*(myConn);//print albums

System.***out***.println("All contents are printed.");

}

**else** System.***out***.println("Invalid input. Please enter again.");//wrong input

}

}

//exception handling

**catch**(SQLSyntaxErrorException e) {

e.getMessage();

System.***out***.println("Invalid input. Exit the program.");

}

**catch**(SQLIntegrityConstraintViolationException e) {

e.getMessage();

System.***out***.println("Invalid input. Exit the program.");

}

**catch**(SQLException e) {

e.printStackTrace();

}

**finally** {

**if**(myConn!=**null**) {//close connection

**try**{

myConn.close();

}**catch** (SQLException e) {

e.printStackTrace();

}

}

}

}

/\*\*

\* method for print all data in songs table

\*

\* **@param** myConn connection variable

\* **@throws** SQLException parameter is null or empty

\*/

**public** **static** **void** songs(Connection myConn) **throws** SQLException{

Statement songState=myConn.createStatement();//create statement

String songSql="select \* from songs";//select all data from albums table

ResultSet songResSet=songState.executeQuery(songSql);//query using statement

String title, sname, aname;

**int** ptm, pts;

System.***out***.println();

System.***out***.println("list of Songs");

//print all data in songResSet

**while**(songResSet.next()) {

title=songResSet.getString("title");

sname=songResSet.getString("sname");

aname=songResSet.getString("aname");

ptm=songResSet.getInt("ptm");

pts=songResSet.getInt("pts");

System.***out***.println(String.*format*("title:%40s|singer:%15s|album:%30s|playtime:%4dm%4ds",title, sname,aname,ptm,pts));

}

//close Statement

songResSet.close();

songState.close();

}

/\*\*

\* method for print all data in singers table

\* **@param** myConn connection variable

\* **@throws** SQLException parameter is null or empty

\*/

**public** **static** **void** singers(Connection myConn) **throws** SQLException {

Statement singerState=myConn.createStatement();//create statement

String singerSql="select \* from singers";//select all data from singers table

ResultSet singerResSet=singerState.executeQuery(singerSql);//query using statement

String name, debutdate, sex;

**int** nop;

System.***out***.println();

System.***out***.println("list of Singers");

//print all data in singerResSet

**while**(singerResSet.next()) {

name=singerResSet.getString("name");

debutdate=singerResSet.getString("debutdate");

nop=singerResSet.getInt("nop");

sex=singerResSet.getString("sex");

System.***out***.println(String.*format*("singer: %15s|debut date: %10s|number of people: %2d|sex of singer: %6s",name, debutdate, nop, sex));

}

//close Statement

singerResSet.close();

singerState.close();

}

/\*\*

\* method for print all data in albums table

\* **@param** myConn connection variable

\* **@throws** SQLException parameter is null or empty

\*/

**public** **static** **void** albums(Connection myConn) **throws** SQLException{

Statement albumState=myConn.createStatement();//create statement

String albumSql="select \* from albums";//select all data from albums table

ResultSet albumResSet=albumState.executeQuery(albumSql);//query using statement

String name, sname, releasedate;

**int** nos;

System.***out***.println();

System.***out***.println("list of Albums");

//print all data in albumResSet

**while**(albumResSet.next()) {

name=albumResSet.getString("name");

nos=albumResSet.getInt("nos");

sname=albumResSet.getString("sname");

releasedate=albumResSet.getString("releasedate");

System.***out***.println(String.*format*("album: %30s|number of songs: %2d|singer: %15s|released date: %10s",name, nos, sname, releasedate));

}

//close Statement

albumResSet.close();

albumState.close();

}

/\*\*

\* method for insertion singer, album, song

\* **@param** myConn connection variable

\* **@throws** SQLException parameter is null or empty

\*/

**public** **static** **void** insert1(Connection myConn) **throws** SQLException{//singer,album,song

String sname, aname, title, debutdate, releasedate, sex;

**int** nos, nop, ptm, pts;

//get information of singer, album, song

*input*.nextLine();

System.***out***.print("name of singer:");

sname=*input*.nextLine();

System.***out***.print("name of album:");

aname=*input*.nextLine();

System.***out***.print("title of song:");

title=*input*.nextLine();

System.***out***.print("playtime minute of song:");

ptm=*input*.nextInt();

System.***out***.print("playtime second of song:");

pts=*input*.nextInt();

*input*.nextLine();

System.***out***.print("debut date of singer:");

debutdate=*input*.nextLine();

System.***out***.print("released date of album:");

releasedate=*input*.nextLine();

System.***out***.print("sex of singer:");

sex=*input*.nextLine();

System.***out***.print("number of people belong to singer:");

nop=*input*.nextInt();

System.***out***.print("number of song belong to album:");

nos=*input*.nextInt();

**new** singer(sname,debutdate,nop,sex,myConn);

**new** album(aname,nos,sname,releasedate,myConn);

**new** song(title,sname,aname,ptm,pts,myConn);

singer.*insert\_singer*();//insert singer

album.*insert\_album*();//insert album

song.*insert\_song*();//insert song

System.***out***.println("insertion finished.");

}

/\*\*

\* method for insertion album, song

\* **@param** myConn connection variable

\* **@throws** SQLException parameter is null or empty

\*/

**public** **static** **void** insert2(Connection myConn) **throws** SQLException {//album,song

String sname, aname, title, releasedate;

**int** nos, ptm, pts;

*input*.nextLine();

//get information of album and song

System.***out***.print("name of singer:");

sname=*input*.nextLine();

System.***out***.print("name of album:");

aname=*input*.nextLine();

System.***out***.print("title of song:");

title=*input*.nextLine();

System.***out***.print("playtime minute of song:");

ptm=*input*.nextInt();

System.***out***.print("playtime second of song:");

pts=*input*.nextInt();

*input*.nextLine();

System.***out***.print("released date of album:");

releasedate=*input*.nextLine();

System.***out***.print("number of song belong to album:");

nos=*input*.nextInt();

**new** album(aname,nos,sname,releasedate,myConn);

**new** song(title,sname,aname,ptm,pts,myConn);

album.*insert\_album*();//insert album

song.*insert\_song*();//insert song

System.***out***.println("insertion finished.");

}

/\*\*

\* method for insertion song

\* **@param** myConn connection variable

\* **@throws** SQLException parameter is null or empty

\*/

**public** **static** **void** insert3(Connection myConn) **throws** SQLException{

String sname, aname, title;

**int** ptm, pts;

//get information of song

*input*.nextLine();

System.***out***.print("name of singer:");

sname=*input*.nextLine();

System.***out***.print("name of album:");

aname=*input*.nextLine();

System.***out***.print("title of song:");

title=*input*.nextLine();

System.***out***.print("playtime minute of song:");

ptm=*input*.nextInt();

System.***out***.print("playtime second of song:");

pts=*input*.nextInt();

**new** song(title,sname,aname,ptm,pts,myConn);

song.*insert\_song*();//insert song

System.***out***.println("insertion finished.");

}

/\*\*

\* method for updating playtime of song according to user input

\* **@param** myConn connection variable

\* **@throws** SQLException parameter is null or empty

\* **@throws** SQLSyntaxErrorException parameter is null or empty

\*/

**public** **static** **void** arrange(Connection myConn) **throws** SQLException, SQLSyntaxErrorException {//playtime arrange

//create prepareStatement

*pStmt*=myConn.prepareStatement("update songs set ptm=ptm+? where title=?");

PreparedStatement pStmt2=myConn.prepareStatement("update songs set pts=pts+? where title=?");

String title;

**int** isec, imin, sec;

//get title of song to arrange

*input*.nextLine();

System.***out***.println("Enter the title of song that you want to arrange.");

title=*input*.nextLine();

//get sec of song to increase

System.***out***.println("Enter the second you want to increase.");

sec=*input*.nextInt();

//calculate min, sec to increase

imin=sec/60;

isec=sec%60;

//set parameters

*pStmt*.setInt(1, imin);

pStmt2.setInt(1, isec);

*pStmt*.setString(2, title);

pStmt2.setString(2, title);

//update

*pStmt*.executeUpdate();

pStmt2.executeUpdate();

//close prepareStatement

*pStmt*.close();

pStmt2.close();

System.***out***.println("arrangement finished.");

}

/\*\*

\* method for updating album(number of song) and song(playtime)

\* **@param** myConn connection variable

\* **@throws** SQLException parameter is null or empty

\* **@throws** SQLSyntaxErrorException parameter is null or empty

\*/

**public** **static** **void** repackage(Connection myConn) **throws** SQLException{

Savepoint savepoint1 = **null**;

**try** {

myConn.setAutoCommit(**false**);//for using transaction

String aname;

**int** sec, imin, isec, nos;

PreparedStatement p2;

PreparedStatement p3;

savepoint1 = myConn.setSavepoint("Savepoint1");

//create prepareStatement

*pStmt*=myConn.prepareStatement("update albums set nos=? where name=?");

p2=myConn.prepareStatement("update songs set ptm=ptm+? where aname=?");

p3=myConn.prepareStatement("update songs set pts=pts+? where aname=?");

//get information of repackaging

*input*.nextLine();

System.***out***.println("Enter the original name of album that you want to repackage.");

aname=*input*.nextLine();

System.***out***.println("Enter number of song to contain this album");

nos=*input*.nextInt();

//get sec of song to increase

System.***out***.println("Enter the second you want to increase.");

sec=*input*.nextInt();

//calculate min, sec to increase

imin=sec/60;

isec=sec%60;

//set parameter

*pStmt*.setInt(1, nos);

*pStmt*.setString(2, aname);

p2.setInt(1, imin);

p2.setString(2, aname);

p3.setInt(1, isec);

p3.setString(2, aname);

//update

*pStmt*.executeUpdate();

p2.executeUpdate();

p3.executeUpdate();

myConn.commit();//commit

myConn.setAutoCommit(**true**);//reset

//close prepareStatement

*pStmt*.close();

p2.close();

p3.close();

System.***out***.println("repackage finished.");

}**catch**(SQLException e) {

myConn.rollback(savepoint1);

}

}

/\*\*

\* method for deletion song

\* **@param** myConn connection variable

\* **@throws** SQLException parameter is null or empty

\* **@throws** SQLSyntaxErrorException parameter is null or empty

\*/

**public** **static** **void** delete(Connection myConn) **throws** SQLException, SQLSyntaxErrorException{

String title;

//get title of song to delete

*input*.nextLine();

System.***out***.println("Enter the title of song that you want to delete.");

title=*input*.nextLine();

**new** song(title, myConn);

song.*delete\_song*();//delete song

System.***out***.println("deletion finished.");

}

/\*\*

\* method show all song that are more than four minutes long

\* **@param** myConn connection variable

\* **@throws** SQLException parameter is null or empty

\* **@throws** SQLSyntaxErrorException parameter is null or empty

\*/

**public** **static** **void** select1(Connection myConn) **throws** SQLException, SQLSyntaxErrorException {

*pStmt*=myConn.prepareStatement("select \* from songs where ptm>? or ptm=?");//create prepareStatement

ResultSet rs;

**int** m, ptm, pts;

String title, sname, aname;

//get lower bound

System.***out***.println("enter the lower bound of minute.");

m=*input*.nextInt();

*pStmt*.setInt(1, m);

*pStmt*.setInt(2, m);

rs=*pStmt*.executeQuery();//query

System.***out***.println();

System.***out***.println("list of result data");

//print all songs that equal or over n minute

**while**(rs.next()) {

title=rs.getString("title");

sname=rs.getString("sname");

aname=rs.getString("aname");

ptm=rs.getInt("ptm");

pts=rs.getInt("pts");

System.***out***.println(String.*format*("title:%40s|singer:%15s|album:%30s|playtime:%4dm%4ds",title, sname,aname,ptm,pts));

}

//close resultset and prepareStatement

rs.close();

*pStmt*.close();

}

/\*\*

\* method show all song according to sex of singer and released date

\* **@param** myConn connection variable

\* **@throws** SQLException parameter is null or empty

\* **@throws** SQLSyntaxErrorException parameter is null or empty

\*/

**public** **static** **void** select2(Connection myConn) **throws** SQLException, SQLSyntaxErrorException {

*pStmt*=myConn.prepareStatement("select title, releasedate "

+ "from songs inner join albums on songs.aname=albums.name "

+ "where (releasedate between ? and ?) and "

+ "(songs.sname=some(select name from singers where sex=?))");//create prepareStatement

ResultSet rs;

**int** a;

String title, releasedate, begindate, enddate, sex = **null**;

//get information of sex

System.***out***.println("Select sex of singer. 1.female 2.male 3.both");

a=*input*.nextInt();

**if**(a==1) sex="female";

**else** **if**(a==2) sex="male";

**else** **if**(a==3) sex="both";

*input*.nextLine();

//get year of released date

System.***out***.println("Enter the year of released date.");

releasedate=*input*.nextLine();

begindate=releasedate+"-01-01";

enddate=releasedate+"-12-31";

//set parameter

*pStmt*.setString(1, begindate);

*pStmt*.setString(2, enddate);

*pStmt*.setString(3, sex);

rs=*pStmt*.executeQuery();//query

System.***out***.println();

System.***out***.println("list of result data");

//print title and released date according to sex of singer and released date

**while**(rs.next()) {

title=rs.getString("title");

releasedate=rs.getString("releasedate");

System.***out***.println(String.*format*("title: %40s|released date: %10s",title, releasedate));

}

//close resultSet and prepareStatement

rs.close();

*pStmt*.close();

}

/\*\*

\* method show all song according to the number of songs in the album.

\* **@param** myConn connection variable

\* **@throws** SQLException parameter is null or empty

\* **@throws** SQLSyntaxErrorException parameter is null or empty

\*/

**public** **static** **void** select3(Connection myConn) **throws** SQLException, SQLSyntaxErrorException {

*pStmt*=myConn.prepareStatement("select \* from aview where nos>? or nos=?");//create prepareStatement

ResultSet rs;

**int** nos, nos2;

String title, aname;

//get number of song

System.***out***.println("enter lower bound of number of song belong to one album.");

nos=*input*.nextInt();

//set parameter

*pStmt*.setInt(1, nos);

*pStmt*.setInt(2, nos);

rs=*pStmt*.executeQuery();//query

System.***out***.println();

System.***out***.println("list of result data");

//print songs that are belong to album contains songs over n and number of song

**while**(rs.next()) {

title=rs.getString("title");

nos2=rs.getInt("nos");

aname=rs.getString("name");

System.***out***.println(String.*format*("album: %30s|title: %40s|number of songs: %2d", aname, title, nos2));

}

//close resultSet and prepareStatement

rs.close();

*pStmt*.close();

}

}