데이터베이스시스템

DBMS_Programming(assignment 4)

Music application

제출일 2020.12.05
담당교수 김상욱 교수님
전공 컴퓨터공학과
학번 2015005169
이름 최윤석

README

1.Version

JAVA : jdk-14.0.2
MariaDB : 10.5
MySQL Router, Server, Shell : 8.0
Encoding: UTF-8
Referenced Libraries : mariadb-java-client-2.7.1.jar
Test in Window cmd : chcp 949
Date: 2020.12.05

2. Class Information

2.1 SimpleFunction.class

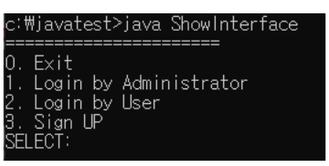
이 클래스는 getUserPass라는 public static 함수를 하나 가지고 있다.
이 함수는 String s를 매개변수로 받아들여 SHA-256을 이용하여 encrypt하여 반환해준다.

2.2 ShowInterface.class

이 클래스는 main method를 포함하고 있으며 사용자로부터 입력을 받고 해당 입력에 따라 적절한 인터페이스를 보여준다. 인터페이스를 보여주는 함수는 다음과 같다.

```
//사용자의 선택에 따라 적절한 메뉴를 보여주는 함수를 불러온다.
1. private void mainMenu(ShowInterface program);

//사용자로 로그인 하였을 때 보여지는 인터페이스
//프로그램 종료, 음악 검색, 플레이리스트 사용, 내 정보확인
//회원 탈퇴 등의 기능을 가지고 있다.
2. private char showUserMenu(void);

// 아이디와 비밀번호를 받아서 User class의 isInDatabase로 넘겨줘
// 사용자가 등록되어 있는지 아닌지 알아낸다.
3. private boolean LoginUser(ShowInterface s);

// 사용자가 특정 플레이리스트를 선택했을 때 보여주는 인터페이스
4. private char add_del_playlist(void);

//사용자가 플레이리스트 관련 작업을 할 때 보여주는 인터페이스
5. private char showUser_Playlist(void);

//사용자가 음악 검색을 선택했을 때 보여주는 인터페이스
6. private char showUser_Music_Search(void);

//관리자가 음악 검색을 선택했을 때 보여주는 인터페이스
7. private char showAdmin_Music_Search(void);

//관리자가 음악에 관리를 원할 때 보여주는 인터페이스
8. private char showAdmin_Music(void);

//관리자로 로그인을 시도하였을 때 아이디와 비밀번호를 입력받고
// setAdmin함수를 사용해 관리자인지 아닌지 판단.
9. private boolean LoginAdmin(ShowInterfacs s);

//프로그램을 종료할지, 관리자로 로그인할지
//사용자로 로그인할지, 회원가입을 할지 선택할 수 있다.
10. private char HomeMenu(void);

//프로그램 종료시 보여주는 인터페이스
11. private void ShowEnd(void);

//회원가입시 필요한 사용자 정보를 입력받아 문자열로
//반환해주는 함수
12. private String[] takeUserInfo(void);

//관리자가 음악을 등록할 때 필요한 정보를 입력받기 위해 사용하는 함수
13. private String[] takeMusicInfo(void);
```

-수행되는 기능



2.3 Admin.class

이 클래스는 music application의 관리자로 로그인 하였을 때 사용할 수 있는 기능들을 함수로 구현한 클래스이다. 이 클래스에는 멤버 변수가 4개 존재하는데 다음과 같다.

```
private String inputname; //사용자(관리자)의 아이디 입력받는 변수
private String inputpass; // 사용자(관리자)의 비밀번호 입력받는 변수

//jdbc를 사용하여 mariadb에 접속하여 musicstreaming database를 사용하기 위한 url주소
private String url = "jdbc:mariadb://localhost:3307/musicstreaming";
Connection admincon; //데이터베이스에 sql query를 보낼 때 사용하는 Connection 객체
```

다음은 mariadb jdbc 커넥터를 사용하기 위하여 static영역을 이용하여 jdbc를 등록한다.

```
static {
    try {Class.forName("org.mariadb.jdbc.Driver");
    }catch (ClassNotFoundException e) {
      e.printStackTrace();
    }
  }
```

멤버 함수로는 다음과 같은 함수들을 가지고 있다.

```
//관리자의 주민등록번호를 받아오는 함수
1. public static String GetAdminSsn(void);

//사용자를 탈퇴시키는 함수
2. public static void Withdrawl(String ssn);

//모든 사용자에 대한 정보를 간단하게 보여주는 함수
3. public static void ShowAllUserSimple(void);

//모든 사용자에 대한 정보를 자세하게 보여주는 함수
4. public static void ShowAllUserDetail(void);

//사용자를 찾는 함수
5. public static void SearchUser(String n);

//admin이 맞는지 확인해주는 함수
6. public boolean isAdmin(void);
```

1) public static String GetAdminSsn(void)

Connection admincon = DriverManager.getConnection
("jdbc:mariadb://localhost:3307/musicstreaming","root","with7742I@");
→ DriverManger.getConnection을 사용하여 url, id, password를 넘겨줘서 sql에 연결한다.
Connection,Statement와 ResultSet을 사용해서 MariaDB로 부터 결과를 받아온다.
프로그램 설계시 관리자는 1명만 사용하기로 정하여서
SELECT 관리자주민번호 FROM 음원관리자
위 Sql문을 사용했다.

2) public static void Withdrawl(String ssn)

Connection admincon = DriverManager.getConnection
("jdbc:mariadb://localhost:3307/musicstreaming","root","with7742I@");

관리자 권한으로 db에 연결하여 사용자를 삭제한다. 이때 사용한 sql문은 다음과 같다.
"DELETE FROM musicstreaming.스트리밍구독자 WHERE 구독자주민번호='"+ssn+"'"

ResultSet rs로 위 sql을 실행한 결과 값을 저장하고 rs.next()의 반환 값에 따라서 삭제할 수 있는 사용자인지 아닌지 구분해 결과를 보여준다.

3) public static void ShowAllUserSimple(void)

위에 1), 2)에서 Connection, Statement, ResultSet을 사용하는 방법을 기술 하였으니 후로는 따로 기술하지 않겠다.
"SELECT ID, 성명, 구독자주민번호 FROM 스트리밍구독자 ORDER BY 성명"
위 sql문을 사용하여 스트리밍 구독자 table에서 id, 이름, 주민번호를 뽑아내어 이름을 기준으로 나열하여 가져왔다. ResultSet rs의 rs.next를 활용하여 rs.next가 null일 때까지 사용자의 정보를 형식에 맞추어 출력해준다.

4) public static void ShowAllUserDetail(void)

"SELECT 성명, 구독자주민번호, 주소, 핸드폰번호, 구독개월 수, 구독시작날짜, ID
FROM 스트리밍구독자 ORDER BY 구독시작날짜"
위 sql문을 사용하여 스트리밍 구독자 table에서 정보를 뽑아내어 사용자의 등록일을 기준으로 나열하여 가져왔다. ResultSet rs의 rs.next를 활용하여 rs.next가 null일 때까지 사용자의 정보를 형식에 맞추어 출력해준다.

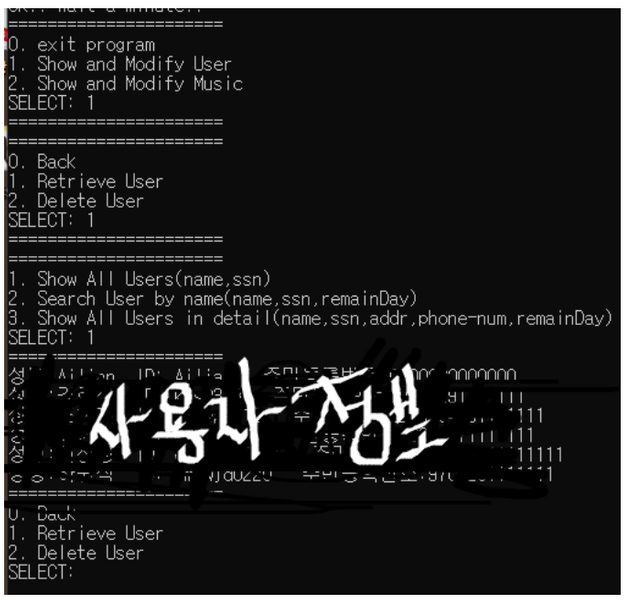
5) public static void SearchUser(String n)

"SELECT 성명, 구독자주민번호, 구독개월수, 구독시작날짜, ID
FROM 스트리밍구독자 WHERE 성명='"+n+"'"
위 sql문을 사용하여 스트리밍 구독자 table에서 tuple 하나를 검색하는데 기준은 넘겨받은 String n과 동일한 이름을 가진 사용자를 검색하는 것이다.. ResultSet rs의 rs.next를 활용하여 rs.next가 null일 때까지 사용자의 정보를 형식에 맞추어 출력해준다. 이 경우 사용자의 구독시작후 남은 날짜를 계산해주기위해 LocalDate와 attribute 구독시작날짜, 구독개월수를 이용하여 계산하였다.

6) public boolean isAdmin(void)

DriverManager.getConnection(url,id,pw)가 성공적으로 값이 반환되면 true를 반환하고 잘못 된 입력이라 db에 접속이 안 될 경우 프로그램을 강제종료시켜버린다.

-수행기능예시



2.4 User class

이 클래스는 music application의 사용자로 로그인 하였을 때 사용할 수 있는 기능들을 함수로 구현한 클래스이다. 이 클래스에는 멤버 변수가 9개 존재하는데 다음과 같다.

```
private String inputname; //사용자(구독자)의 아이디 입력받는 변수
private String inputpass; // 사용자(구독자)의 비밀번호 입력받는 변수

// 관리자 아이디로 확인할 것이 필요한 함수에서 사용
private String adminID="root";
private String adminPW="with7742I@";

// mariadb에 접속하여 musicstreaming database를 사용하기 위한 url주소
private String url = "jdbc:mariadb://localhost:3307/musicstreaming";

// 구독자의 인증이 끝나면 사용할 musicstreaming database에만
// 접근 허용이 된 아이디 부여
private String userID="musicuser";
private String userPW="musicstreaming";

Connection admincon; //db에 sql query를 보낼 때 사용하는 Connection 객체
Statement st;
```

다음은 Admin class와 동일하게 mariadb jdbc 커넥터를 사용하기 위하여 static영역을 이용하여 jdbc를 등록한다.

```
static {
    try {Class.forName("org.mariadb.jdbc.Driver");
    }catch (ClassNotFoundException e) {
      e.printStackTrace();
    }
  }
```

멤버 함수로는 다음과 같은 함수들을 가지고 있다.

```
//사용자(구독자) 회원가입을 위한 함수
1. public static void SignUP (String gender, String address, String Phone,
String ssn, int subscribe, String pw, String name, String id);

//사용자가 데이터베이스에 등록되어있는지 확인하는 함수
2. public boolean isInDatabase(void);

//사용자의 플레이리스트 목록을 보여주는 함수
3. public void showMyPlayList(void);

//사용자가 플레이리스트를 선택하면 플레이리스트의 존재여부를 파악한후
//해당 플레이리스트의 노래목록을 보여주는 함수
4. public void usePlayList(String p_name, ShowInterface s);

//전체 노래 순위 말고 구독자가 많이 플레이한 노래 순으로 출력해주는 함수
5. public void GetUserTitleRanking(void);

//사용자가 선택한 플레이리스트에 노래를 추가하는 함수
6. public void AddTitleToPlayList(String p_name, int addnum);

//사용자가 선택한 플레이리스트에서 노래를 삭제하는 함수
7. public void DelTitleToPlayList(String p_name, int delnum);

//사용자의 플레이리스트를 삭제하는 함수
8. public void DelPlayList(String p_name);

//플레이리스트를 추가하는 함수
9. public void AddPlayList(String p_name);

//내 회원정보를 조회하는 함수
10. public void CheckMyInformation(String myId, String mypass);

//사용자 객체에 Connection 객체와 Statement 객체를 할당해주고 해제하는 함수
11.private void userLogin();
12.public void userLogout();
```

1) public static void SignUp(String gendder,String address, String Phone, String ssn, int subscribe, String pw, String name, String id)

"INSERT INTO 스트리밍구독자 (성별,주소,핸드폰번호,구독자주민번호,구독_관리자주민번호,구독개월수,구독시작날짜,passwd,성명,ID) "
+"VALUES('"+gender+"','"+address+"','"+Phone+"','"+ssn+"','"+adminSsn+"','"+subscribe+"','"+nowDate+"','"+pw+"','"+name

위 sql문을 사용하여 가입을 원하는 구독자로부터 입력받은 정보를 musicstreaming database에 집어넣는다. 예외가 발생하는 경우는 이미 등록되어 있어서 SQL error가 발생하는 것이기 때문에 가입이 불가능 하다는 것을 알려준다.

2) public boolean isInDatabase()

"SELECT * FROM 스트리밍구독자 WHERE ID='"+this.intputname+"'and passwd='"+this.inputpass+"'";
위 sql문을 사용하여 현재 데이터베이스에 해당 아이디와 비밀번호를 가진 사용자가 있는지 검색한다. 만약 rs.next()가 false라면 데이터베이스에 존재하지 않는 사용자이므로 로그인 실패를 알려주고 false를 반환하며, true인 경우 로그인 성공을 알려주고 true를 반환한다.

3) public void showMyPlayList()

"SELECT 플레이리스트명
FROM 플레이리스트
WHERE 플레이리스트_구독자주민번호= ALL
(SELECT 구독자주민번호
FROM 스트리밍구독자
WHERE ID='"+this.inputname+"')";

위 sql문을 사용하여 플레이리스트에서 나의 플레이리스트 이름을 불러온다. 만약 rs.next()가 false라면 플레이리스트가 하나도 없는 것이므로 보유한 리스트가 없음을 알려준다.

4) public void usePlayList(String p_name, ShowInterface s)

"SELECT 구독자주민번호 FROM 스트리밍구독자 WHERE ID='"this.inputname+"'"
위 sql문을 사용하여 구독자 아이디를 이용해 구독자 주민번호를 얻어온다.
그다음 플레이리스트명을 얻어오고 다시

"SELECT 음악이름, 가수, 장르, 음원관리번호 FROM 추가, 음원"
+"WHERE a_구독자주민번호='"+ssn+"' AND a_플레이리스트명='"+p_name+
"' AND 음원관리번호=a_음원관리번호 ORDER BY 음악이름"
위 sql문을 사용하여 해당 플레이리스트에 저장되어있는 노래 목록을 탐색하여 차례대로 정보를 출력하여 준다.

5) public void GetUserTitleRanking()

"SELECT 음악이름, 가수, 장르, 음원관리번호 FROM 추가, 음원"
+"WHERE play_구독자주민번호='"+ssn+"' AND 음원관리번호=
play_음원관리번호 ORDER BY p_count DESC";
위 sql문을 사용하여 구독자가 적어도 한 번이상 재생한 음원들에 한하여 플레이한 횟수에 따라 음원을 순서대로 출력해준다.

6) public void AddTitleToPlayList(String p_name, int addnum)

select 구문을 활용한 sql 예시는 그만 기술하겠다. 이 함수에서는 sql문을 이용하여 원하는 플레이리스트에 음원을 추가하는 함수인데
1. 해당 관리번호를 가진 음원이 있는가
2. 이미 플레이리스트에 있는 음원인가
를 확인한 후 플레이리스트에 존재하지 않고 음원 테이블에 존재하는 음원이라면 해당 플레이리스트에 등록을 해준다. 이 때 사용한 sql은 다음과 같다.

"INSERT INTO 추가 (a_음원관리번호,a_구독자주민번호,a_플레이리스트명)"
+ " VALUES ("+addnum+",'"+ssn+"','"+p_name+"')";

7) public void DelTitleToPlayList(String p_name, int addnum)

이 함수에서는 sql문을 이용하여 원하는 플레이리스트에 음원을 제거하는 함수인데
1. 해당 관리번호를 가진 음원이 있는가
2. 플레이리스트에 있는 음원인가
를 확인한 후 플레이리스트에 존재하고 음원 테이블에 존재하는 음원이라면 해당 플레이리스트에서 제거해준다. 이 때 사용한 sql은 다음과 같다.

"DELETE FROM 추가 WHERE a_음원관리번호='"+delnum+"' AND a_플레이리스트명='"+p_name+"' AND a_구독자주민번호
='"+ssn+"'";

8) public void DelPlayList(String p_name)

이 함수에서는 sql문을 이용하여 원하는 플레이리스트를 제거하는 함수이다.
플레이리스트가 존재하는지 확인 후 존재한다면 삭제해준다.
이 때 사용한 sql은 다음과 같다.

"**DELETE FROM 플레이리스트 "**
+ "WHERE 플레이리스트_구독자주민번호='"+ssn+"' AND 플레이리스트명='"+p_name+"'";

9) public void AddPlayList(String p_name)

이 함수에서는 sql문을 이용하여 원하는 이름의 플레이리스트를 생성하는 함수이다.
플레이리스트가 존재하는지 확인 후 동일한 이름이 없다면 생성한다.
이 때 사용한 sql은 다음과 같다.

"INSERT INTO 플레이리스트 (플레이리스트_구독자주민번호,플레이리스트명) "
+ "VALUES ('"+ssn+"','"+p_name+"')";

10) public void CheckMyInformation(String myId, String mypass)

이 함수에서는 sql문을 이용하여 구독자 본인의 정보를 확인하는 함수이다.

-수행기능예시

```
플레이리스트에 추가하고자하는 곡의 관리번호를 입력해주세
성공적으로 추가했습니다.
======================
0. Back
1. play music
2. add music
3. delete music
SELECT: 0
======================
======================
0. Back
1. Show My playlist List
2. Use playlist
3. Add playlist
4. Delete playlist
SELECT: 2
======================
사용하고싶은 플레이리스트 이름 입력:balad
=====음악1=====
타이틀 :꽃레트
가수 :아이유
장르 :발라드
관리번호 :41
내가 재생한 횟수 :0

======================
0. Back
1. play music
2. add music
3. delete music
SELECT: 1
======================
플레이를 원하는 곡의 관리번호를 입력하여주세요:41
성공적 플레이
======================
0. Back
1. play music
2. add music
3. delete music
SELECT: 0
======================
======================
0. Back
1. Show My playlist List
2. Use playlist
3. Add playlist
4. Delete playlist
SELECT: 2
======================
사용하고싶은 플레이리스트 이름 입력:balad
=====음악1=====
타이틀 :꽃레트
가수 :아이유
장르 :발라드
관리번호 :41
내가 재생한 횟수 :1

======================
0. Back
1. play music
2. add music
3. delete music
SELECT:
```

```
SELECT: 2
======================
Please Put Your ID : captain9981
Please Put Your PW : 01072255395
OK.. Wait a minute..
로그인성공
======================
0. exit program
1. Search Music
2. My playList
3. My MusicRanking
4. My Information
5. WithDrawl
SELECT: 3
======================
=====음악1=====
타이틀 :나랑 같이 걸을래
가수 :적재
장르 :발라드
관리번호 :5
내가 재생한 횟수 :14

=====음악2=====
타이틀 :희재
가수 :성시경
장르 :발라드
관리번호 :4
내가 재생한 횟수 :13

=====음악3=====
```

2.5 Music.class

음악과 관련된 기능이 들어가있는 클래스이며 멤버 변수는 Connection 객체 하나 존재하고 멤버 함수는 다음과 같다.

```
//음악을 재생하는 함수
1. public static void StartMusic(String n);
```

```
//모든 음원에 관한 정보를 간단하게 출력하는 함수
2. public static void ShowAllMusicSimple(void);

//원하는 기능으로 음원을 검색하는 함수
3. public static void SearchMusic(char select, String searchingWord);

//음원 삭제하는 함수
4. public static void WithdrawlMusic(String control);

//음원관리번호를 입력받아 음원을 재생하는 함수
5. public static void PlayMusic(int num, String id);

//해당 플레이리스트에 노래가 있는지 먼저 확인하고
//존재한다면 해당 음원관리번호의 노래를 재생해주는 함수
6. public static void PlayMusicPlaylist(int num, String id, String listName);

//음원을 추가하는 함수
7. public static void UploadMusic(String title, String singer, String genre,
String Album);
```

1) public static void StartMusic(String n)

n은 음원관리 번호이며 File명령어로 .wav파일을 읽어와서 해당경로에 파일이 존재한다면 AudioInputStream, AudioSystem, Clip, JOptionPane을 사용하여 음악을 재생해준다.

2) public static void ShowAllMusicSimple()

음원 테이블에 존재하는 모든 음원을 검색하여 간단한 정보들만 출력해준다. SELECT문을 사용하여 해결하였다. 사용된 SQL명령어는 다음과 같다.

"SELECT 음악이름, 가수 , 음원관리번호 FROM 음원 ORDER BY 음악이름";

3) public static void SearchMusic(char select, String searchingWord)

select에 따라서 음악이름, 가수명, 장르명, 앨범명 검색과 전체 디테일 정보 출력, 음원 플레이횟수 순으로 출력등의 기능을 지원한다. 음악이름으로 검색하는 것의 sql만 살펴보면 다음과 같다.

"SELECT 음악이름, 가수, 장르, count, 앨범이름, 음원관리번호 FROM 음원 WHERE 음악이름 LIKE '%"+searchingWord+"%'";

다음과 같이 LIKE %를 사용하여 검색어의 일부를 이용하여 table에서 골라낼 수 있게 하였으며 골라낸 음원에 대한 자세한 정보를 출력한다.

4) public static void WithdrawlMusic(String control)

control에 해당하는 음원번호를 음원 table에서 삭제하며 delete구문을 사용한다.

"DELETE FROM musicstreaming.음원 WHERE 음원관리번호='"+control+"'";

5) public static void PlayMusic(int num, String id)

음원이 플레이 되면 play테이블의 개인 play횟수를 증가시키고 음원 테이블의 음원 플레이 횟수를 증가시키며 StartMusic으로 음악을 재생한다. 테이블의 count횟수 증가는 update를 사용하였으며 사용된 SQL들은 다음과 같다.

"SELECT count FROM musicstreaming.음원 WHERE 음원관리번호 = '"+num+"'";

"UPDATE 음원 set count = count+1 WHERE 음원관리번호 = '"+num+"'";

"INSERT INTO musicstreaming.play (play_음원관리번호,play_구독자주민번호,p_count)"
+ " VALUES ("+num+", '"+ssn+"', 1)";

"UPDATE musicstreaming.play set p_count = p_count+1"
+ " WHERE play_구독자주민번호='"+ssn+"' AND play_음원관리번호='"+num+"'";

만약 사용자가 처음 플레이하는 음원이라면 Insert구문을 사용하여 play 테이블에 새로운 tuple을 생성한다.

6) public static void PlayMusicPlayList(int num, String id, String listName)

바로 위 5번 함수와 동일한 기능을 하지만 한 가지 더 체크하는 것은 Select구문을 사용하여 listName 이름을 가진 플레이리스트에 해당 음원이 존재하는지 확인하는 것이다. 해당 플레이리스트에 곡이 존재할 경우에만 노래를 재생한다.
사용된 SQL명령어들은

"SELECT a_음원관리번호 FROM 추가 WHERE a_플레이리스트명='"+listName+"' AND a_음원관리번호='"+num+"'";

이 sql문을 사용해서 음악이 존재한다면 후에는 5번과 동일한 sql을 거쳐 음원을 재생했다.

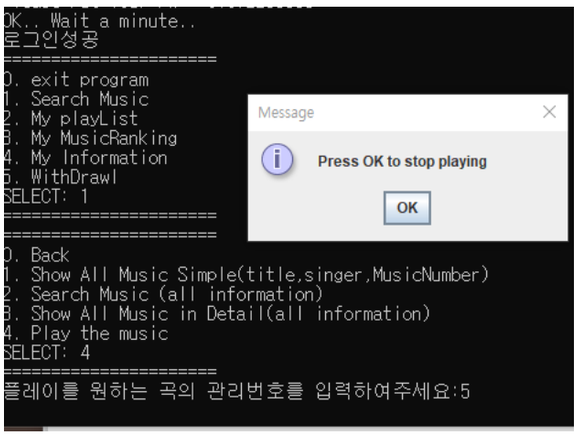
7) public static void UploadMusic(String .....)

음원 정보를 읽어들여 해당 음원을 새롭게 음원 table에 등록한다. 사용된 sql은 다음과 같다.

"SELECT COUNT(*) FROM musicstreaming.음원";

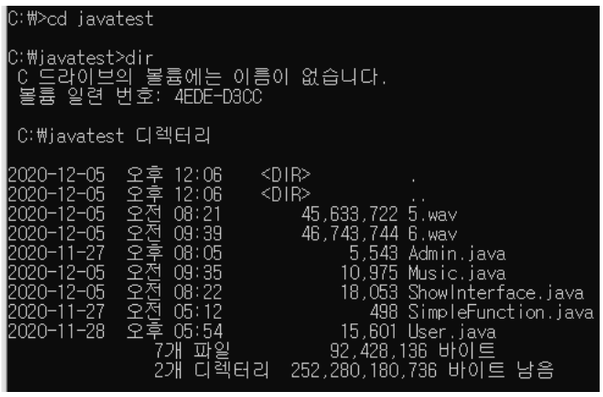
"INSERT INTO musicstreaming.음원(음악이름,가수,장르,count,앨범이름,음원관리번호,음원관리자주민번호)"
+"VALUES('"+title+"','"+singer+"','"+genre+"',0,'"+Alubum+"','"+next+"','"+admin+"')";

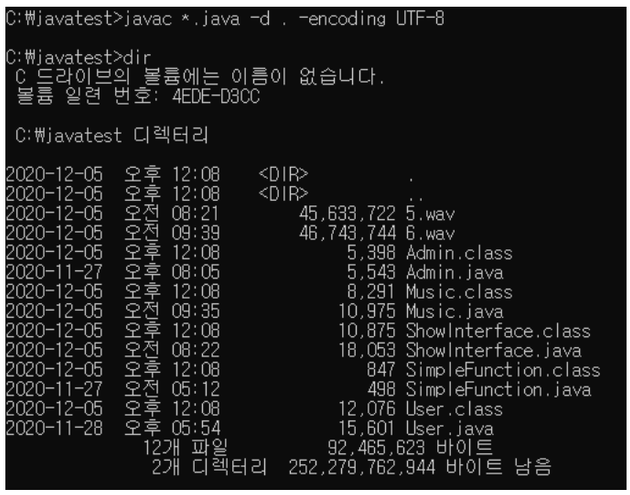
-수행예시


OK.. Wait a minute..
로그인성공
0. exit program
1. Search Music
2. My playList
3. My MusicRanking
4. My Information
5. WithDrawl
SELECT: 1
Message
Press OK to stop playing
OK
0. Back
1. Show All Music Simple(title,singer,MusicNumber)
2. Search Music (all information)
3. Show All Music in Detail(all information)
4. Play the music
SELECT: 4
플레이를 원하는 곡의 관리번호를 입력하여주세요:5


0. Back
1. Show My playlist List
2. Use playlist
3. Add playlist
4. Delete playlist
SELECT: 1
플레이리스트이름: balad
플레이리스트이름: dance
플레이리스트이름: idle
0. Back
1. Show My playlist List
2. Use playlist
3. Add playlist
4. Delete playlist
SELECT: 2
사용하고싶은 플레이리스트 이름 입력:balad
=====음악1=====
타이틀 :내 마음이 움찔했던 순간
가수 :규현
장르 :발라드
관리번호 :6
내가 재생한 횟수 :6
=====음악2=====
타이틀 :팔레트
가수 :아이유
장르 :발라드
관리번호 :41
내가 재생한 횟수 :1
0. Back
1. play music
2. add music
3. delete music
SELECT: 1
플레이를 원하는 곡의 관리번호를 입력하여주세요:5
플레이리스트에 해당곡이 추가되어있지 않습니다.
0. Back
1. play music

3. 컴파일과정


C:₩>cd javatest
C:₩javatest>dir
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 4EDE-D3CC
C:₩javatest 디렉터리
2020-12-05 오후 12:06 <DIR> .
2020-12-05 오후 12:06 <DIR> ..
2020-12-05 오전 08:21 45,633,722 5.wav
2020-12-05 오전 09:39 46,743,744 6.wav
2020-11-27 오후 08:05 5,543 Admin.java
2020-12-05 오전 09:35 10,975 Music.java
2020-12-05 오전 08:22 18,053 ShowInterface.java
2020-11-27 오전 05:12 498 SimpleFunction.java
2020-11-28 오후 05:54 15,601 User.java
7개 파일 92,428,136 바이트
2개 디렉터리 252,280,180,736 바이트 남음



```
C:₩javatest>javac *.java -d . -encoding UTF-8

C:₩javatest>dir
 C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: 4EDE-D3CC

 C:₩javatest 디렉터리

2020-12-05  오후 12:08    <DIR>          .
2020-12-05  오후 12:08    <DIR>          ..
2020-12-05  오전 08:21        45,633,722 5.wav
2020-12-05  오전 09:39        46,743,744 6.wav
2020-12-05  오후 12:08             5,398 Admin.class
2020-11-27  오후 08:05             5,543 Admin.java
2020-12-05  오후 12:08             8,291 Music.class
2020-12-05  오전 09:35            10,975 Music.java
2020-12-05  오후 12:08            10,875 ShowInterface.class
2020-12-05  오전 08:22            18,053 ShowInterface.java
2020-12-05  오후 12:08               847 SimpleFunction.class
2020-11-27  오전 05:12               498 SimpleFunction.java
2020-12-05  오후 12:08            12,076 User.class
2020-11-28  오후 05:54            15,601 User.java
              12개 파일          92,465,623 바이트
               2개 디렉터리  252,279,762,944 바이트 남음
```