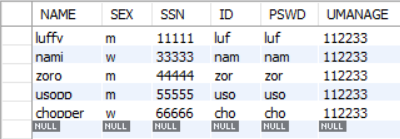
**조민지 프로젝트: DBMS 프로그램 개발**

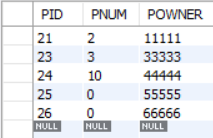
MUSIC WORLD 만들기

* 프로젝트 1,2,3에서 설계한 DB를 기반으로 하는 어플리케이션 프로그램을 개발하였습니다.
* Mysql 을 이용해서 DB의 스키마를 작성했습니다. Intelli J 에서 Java 로 코딩 했습니다.

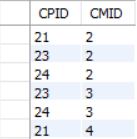
USER 테이블



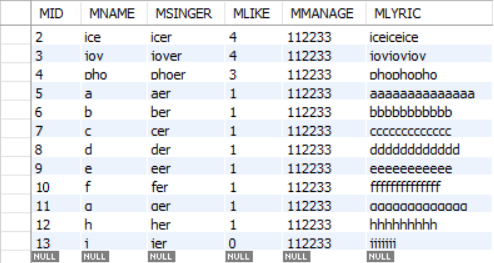
PLAYLIST 테이블



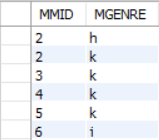
MUSICCONTAIN 테이블



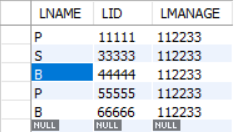
MUSIC 테이블



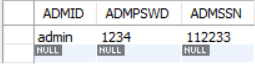
MGENRE 테이블



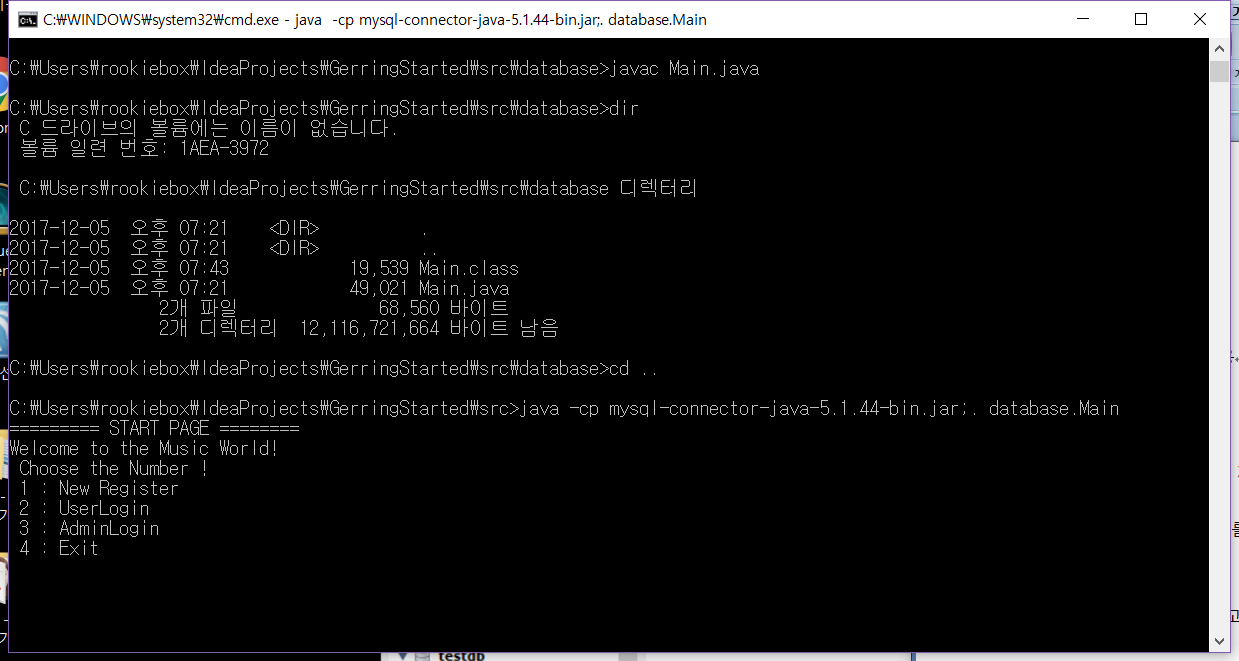
LEVEL 테이블

’

ADMIN 테이블



CLI 에서 Main.java 파일을 컴파일하고 실행했습니다.



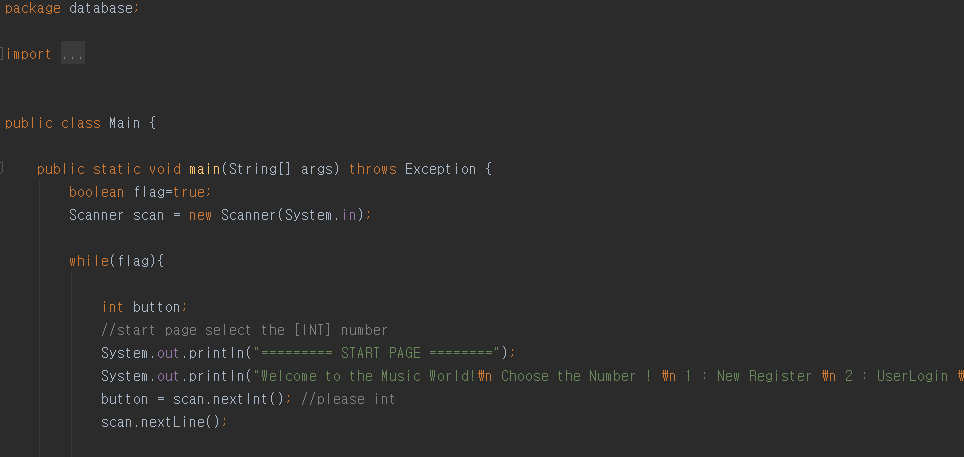
C:\Users\rookiebox\IdeaProjects\GerringStarted\src

java -cp mysql-connector-java-5.1.44-bin.jar;. database.Main

* 1 ===== START PAGE =====

가장 처음에 보여지는 화면입니다.

신규 회원 가입 / 사용자 로그인 / 관리자 로그인 / 프로그램을 [button]으로 선택할 수 있습니다. 올바른 값을 넣어야 합니다.

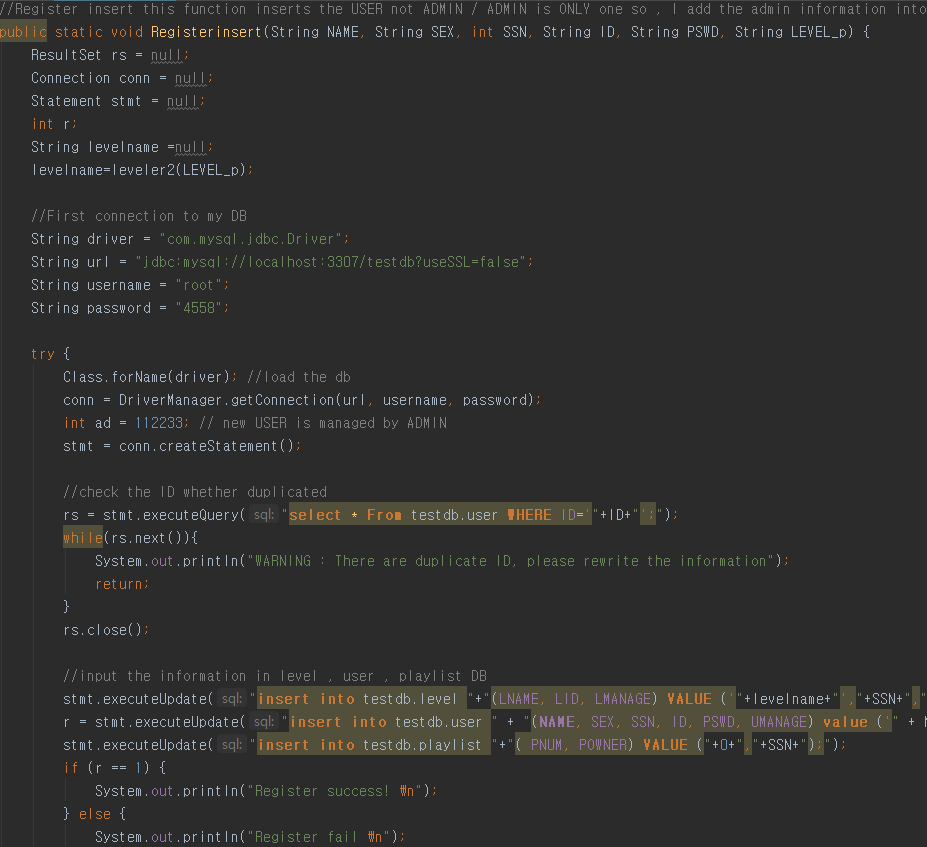


* **1-1 button ==1 신규 회원가입**

사용자가 이름, 성별, 주민번호, ID, password, 설정하고 싶은 level을 입력합니다. 입력 받은 변수를 Registerinsert() 함수로 넘겨줍니다.



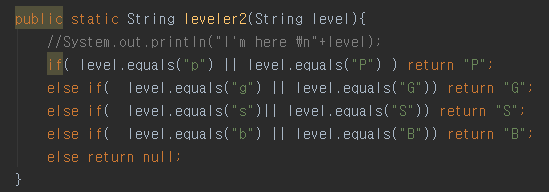
Resigterinsert() 함수



맨 처음 intellij와 mysql 을 연결합니다. String 변수에 미리 만든 스키마 DB의 주소와 로그 인을 위한 정보를 저장합니다.

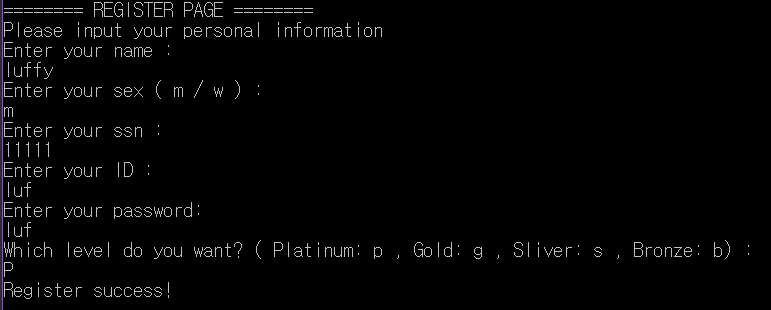
Class.forName() 함수를 사용해서 지정한 DB driver 을 load 합니다.

Lever2() 함수를 이용해서 정형화한 값을 table에 삽입합니다.

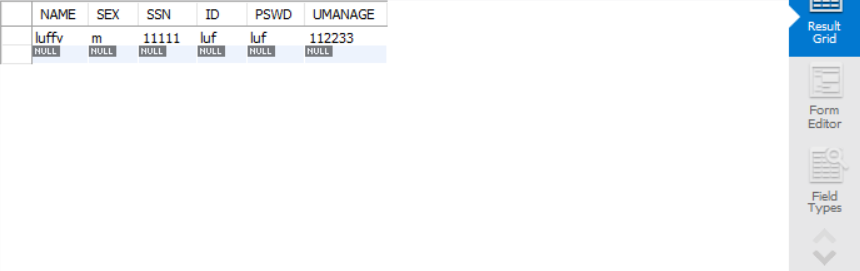


* ID 는 USER table 내에 유일한 값이어야 하므로 중복 체크를 해줍니다. Testdb.user 테이블에 있는 ID 칼럼과 사용자가 입력한 ID 값이 같은 게 있을 경우 SELECT 문을 사용해서 해당 행을 가져옵니다. WHERE 조건을 만족하는 행이 있을 경우 true로 판단하고 경고 창을 띄어줍니다.
* 사용자가 입력한 정보를 토대로 testdb.user, testdb.level, testdb.playlist table에 삽입합니다.

성공하면 Resiger success 를 출력하고 실패하면 Register fail 을 출력합니다.



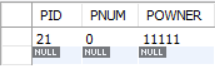
USER table 결과값



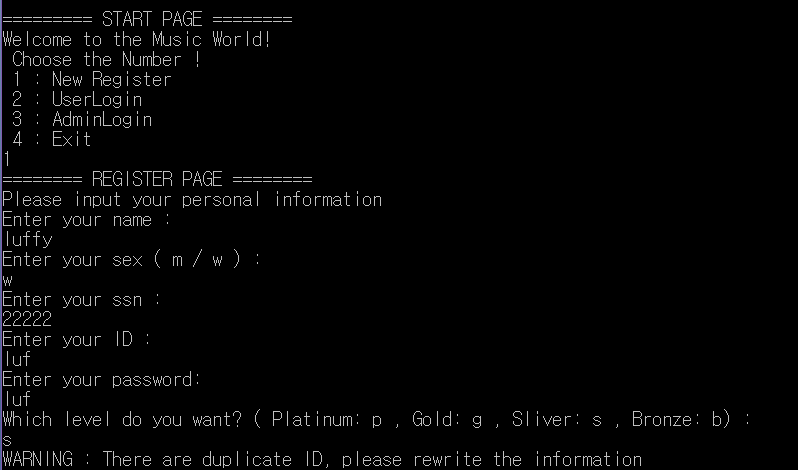
LEVEL table 결과값



PLAYLIST table 결과값

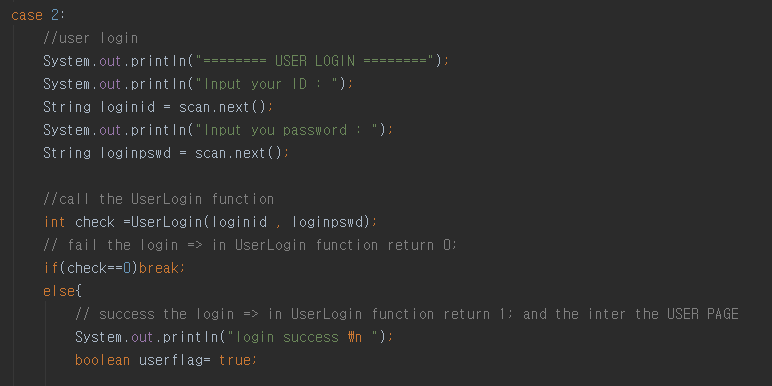


기능) ID 중복 체크



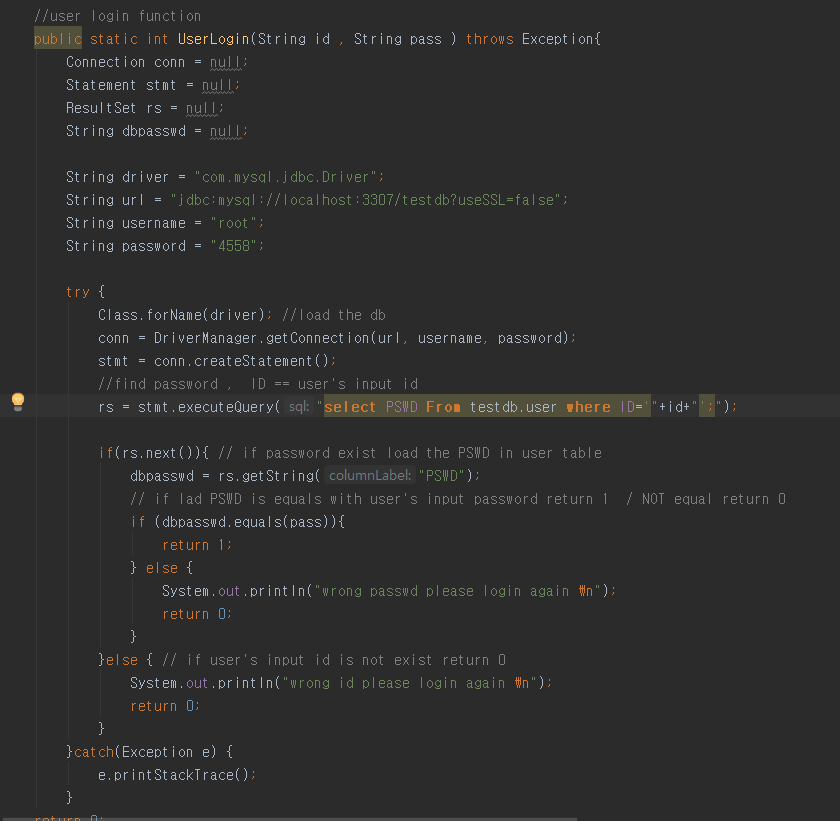
* **1-2 button==2 사용자 로그인**

사용자의 ID 와 password 정보를 입력 받습니다.



UserLogin() 함수를 이용해서 check을 return 받습니다. Check 가 0 이면 로그 인을 실패한 것입니다. Check가 1이면 로그 인을 성공한 것입니다. 성공하면 USER PAGE로 이동합니다.

UserLogin() 함수입니다.



db접근을 위한 connection과 load를 먼저 수행해줍니다.

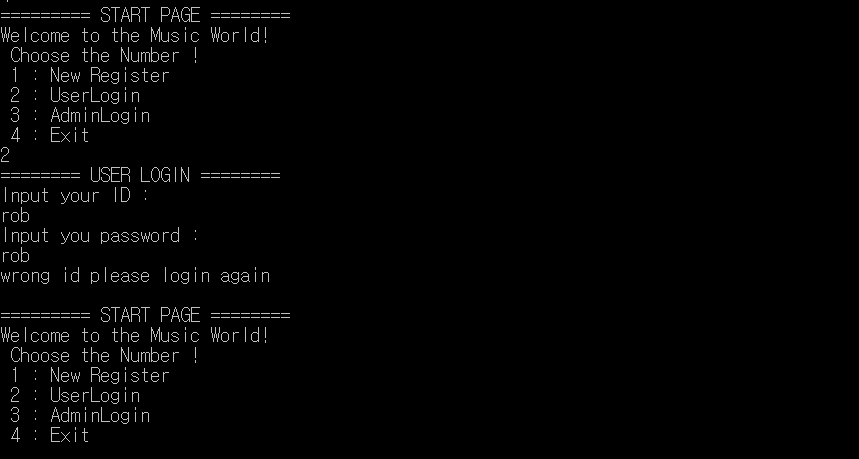
* 우선 입력 받은 ID 정보와 testdb.user에 있는 ID칼럼이 일치하는 PSWD칼럼을 SELECT 이용해서 해당 칼럼의 값을 가져옵니다.

그 값을 [dbpasswd]변수에 저장합니다. WHERE 조건 문을 만족하는 PSWD 값이 없다면 ID 입력이 잘못됐다고 출력해줍니다. 만약 입력 받은 [pass] 정보와 [dbpasswd]가 일치하면 로그인 성공입니다. 일치하지 않으면 passwd 입력이 잘못됐다고 출력해줍니다.

Luffy 는 이미 회원가입을 했습니다. 로그 인을 성공해서 USER PAGE로 이동했습니다.



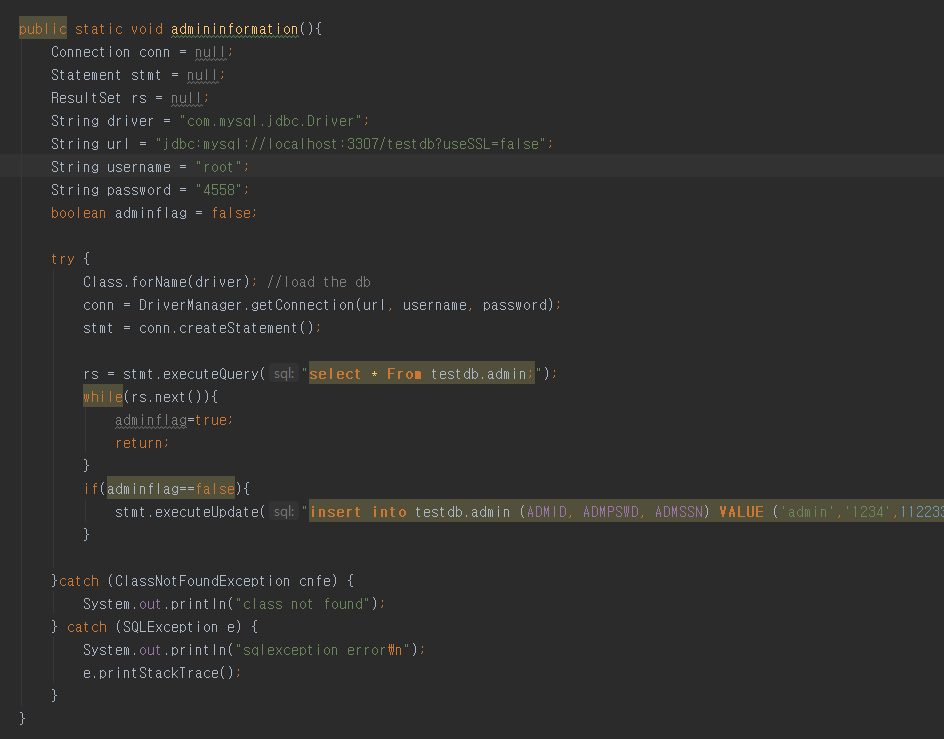
Robin 은 회원가입을 하지 않았습니다. 로그 인을 실패하고 다시 START PAGE로 돌아왔습니다.



* **1-3 button==3 관리자 로그인**

관리자 회원가입을 따로 만들지 않았습니다.

이유: 이 선택란이 일반 사용자에게 노출되면 안되고 이번 과제에 모델로 한 music 앱에도 이러한 서비스가 없었습니다. 제 개인적인 생각으론 관리자 정보같이 중요한 정보는 직접 DB에 적는 게 더 적합할거 같아서 DB에 직접 넣었습니다.

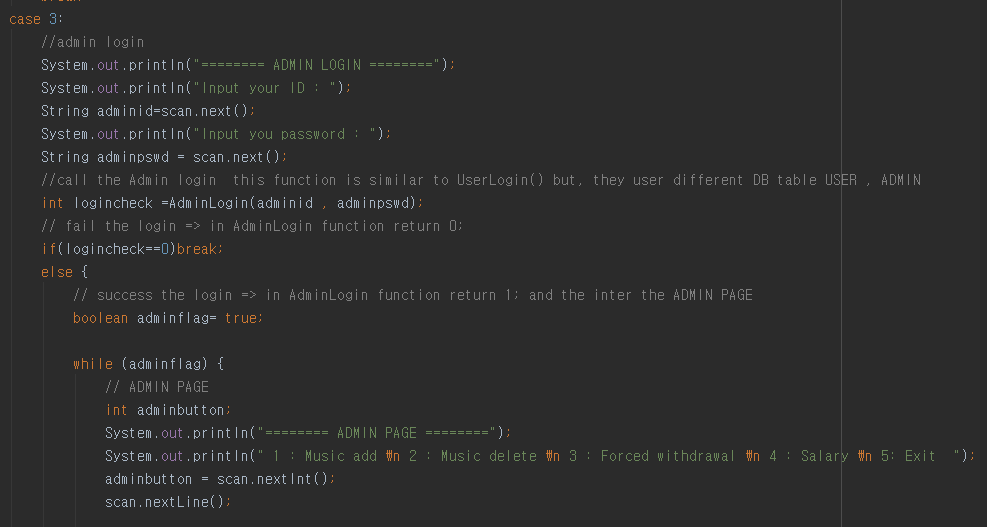


이 함수는 Main.class를 처음 실행할 때 수행되는 함수입니다.

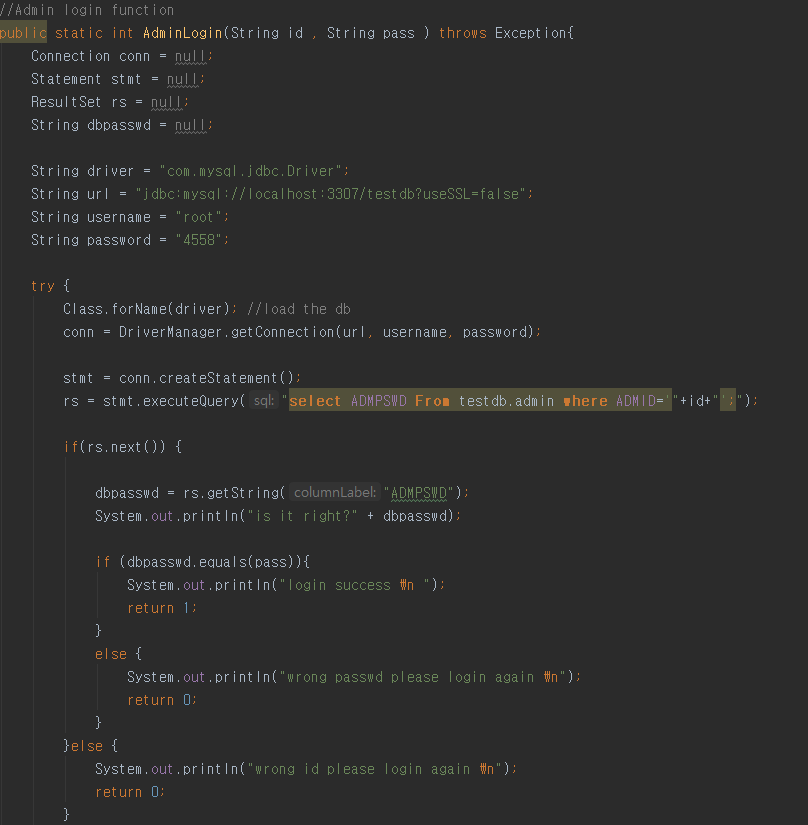
* Testdb.admin테이블에 ADMIN 정보와 ADMPSWD정보와 ADMSSN정보를 직접 넣었습니다.



사용자 로그인과 동일하게 [adminid]와 [adminpswd]를 입력 받고 AdminLogin() 함수를 이용해서 올바른 정보인지 체크합니다. 로그 인에 성공하면 ADMIN PAGE로 이동하고 로그 인에 실패하면 START PAGE로 돌아옵니다.



AdminLogin() 함수

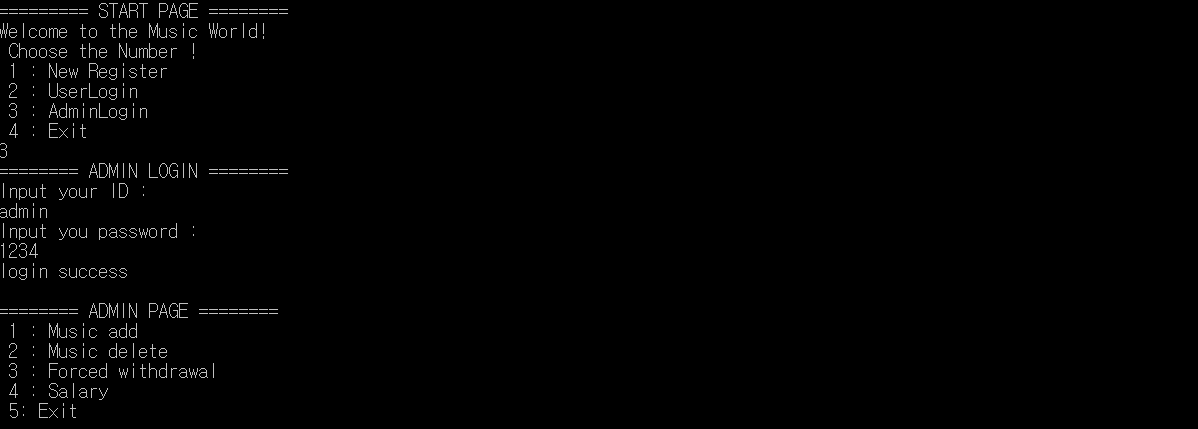


db접근을 위한 connection과 load를 먼저 수행해줍니다.

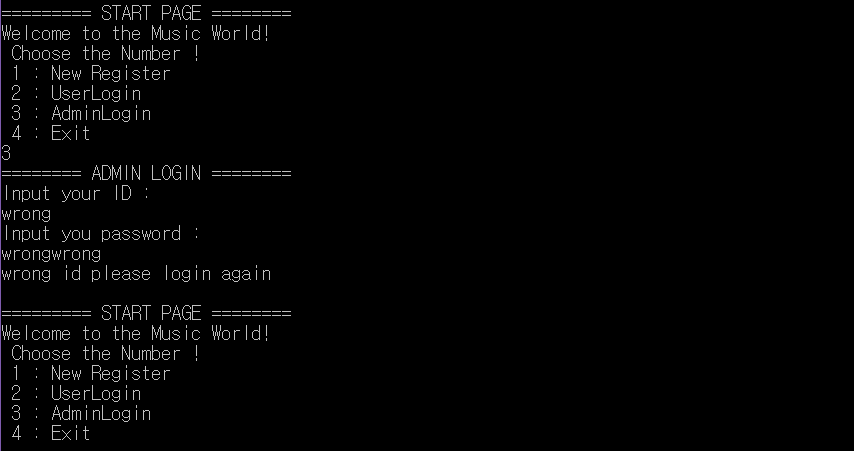
* 사용자가 입력한 id 값과 testdb.admin table에 있는 ID칼럼 값이 같으면 ADMPSWD칼럼의 값을 가져옵니다.

그 값을 [dbpasswd]변수에 저장합니다. 만약 만족하는 값이 없으면 id를 잘못 입력한 것입니다. 사용자가 입력한 pass값과 [dbpasswd]값과 일치하는지 체크합니다. 일치하면 로그인 성공, 일치하지 않으면 잘못된 passwd 값을 입력한 것입니다.

관리자의 아이디는 admin 이고 비밀번호는 1234 입니다. 로그 인을 성공해서 ADMIN PAGE로 이동했습니다.



로그 인을 실패했을 경우 START PAGE 로 이동



* **1-4 button ==4 Music World 프로그램 종류**

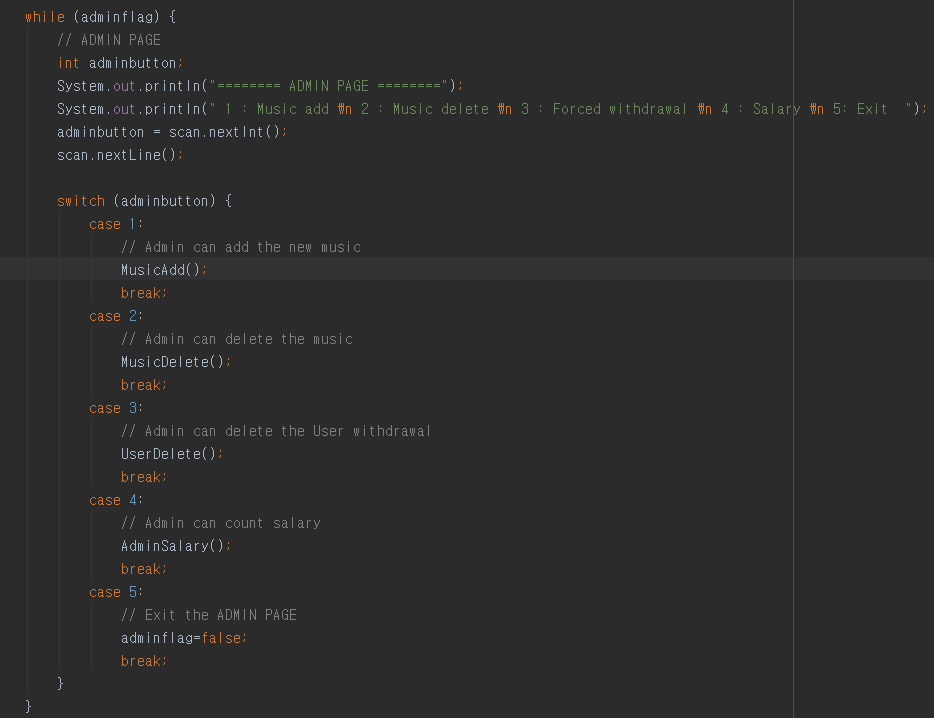
프로그램을 종료합니다.



* 2 =======ADMIN PAGE=======

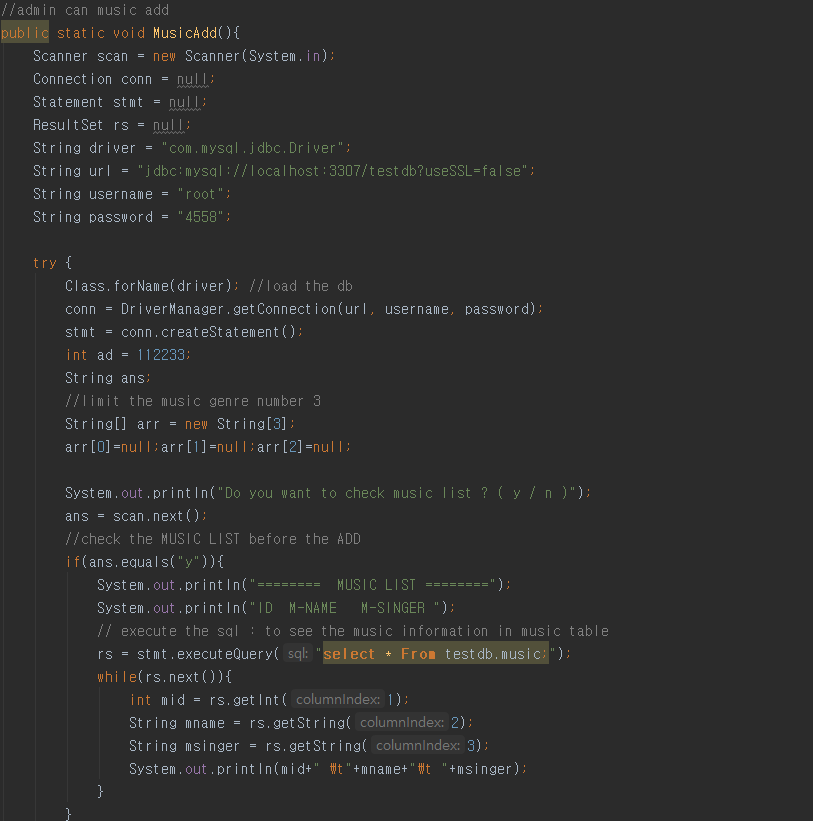
관리자로 로그 인을 성공했을 때 보이는 PAGE입니다.

음악 추가 / 음악 삭제 / 사용자 강제 탈퇴 / 봉급 체크 / 관리자 페이지 나가기 기능이 있습니다. [adminbutton]으로 기능을 선택할 수 있습니다. 올바른 값을 입력해야 합니다.



* **2-1 adminbutton == 1 관리자 음악 추가**

관리자가 MUSIC table에 음악을 추가할 수 있습니다. MusicAdd() 함수를 사용합니다.



db접근을 위한 connection과 load를 먼저 수행해줍니다

음악 한 개에 지정할 수 있는 장르를 3개로 한정했습니다.

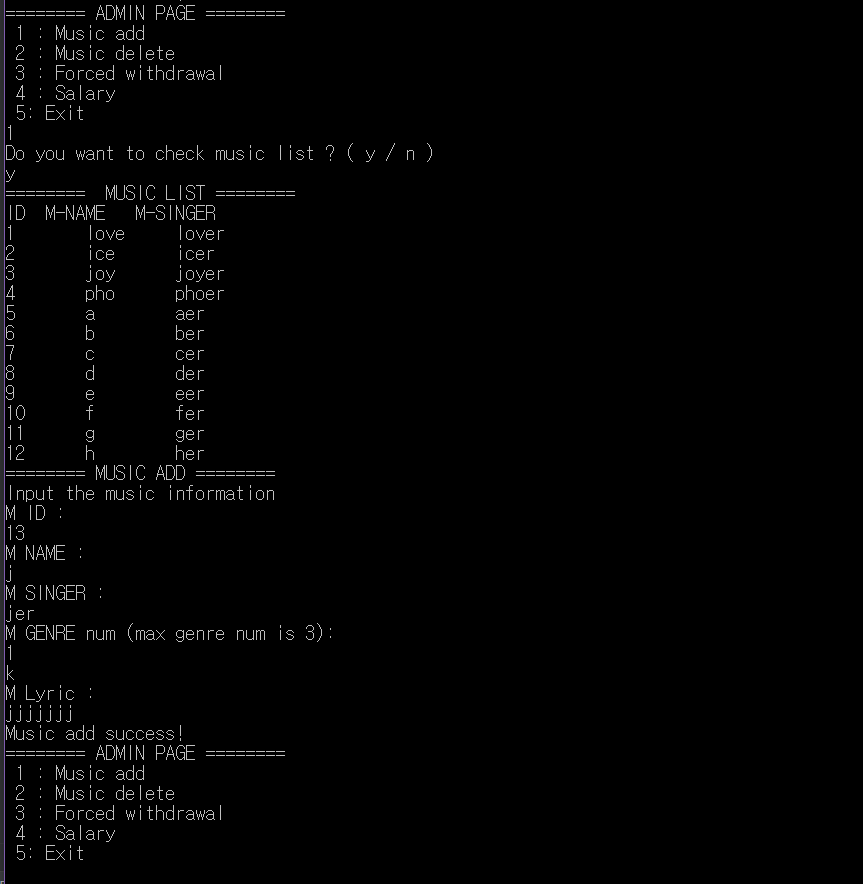
먼저 현재 music store에 있는 노래 리스트를 보고 싶냐고 묻습니다. “y”라고 하면 MUSIC LIST를 보여주고 “n”라고 하면 생략합니다.

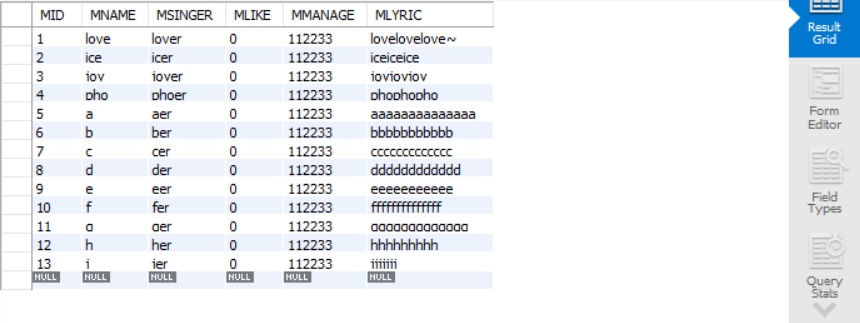
* Testdb.music table에 있는 모든 행의 columnindex가 1,2,3인 MID, MNAME, MSINGER 값들을 변수에 저장해 출력합니다.



새로 추가할 음악에 대한 정보를 입력 받습니다.

* 입력 받은 정보대로 MUSIC table에 넣습니다.
* 음악장르는 다중에트리뷰트입니다. Musiccontain이라는 table을 따로 만들어 각 노래의 장르를 저장했습니다.

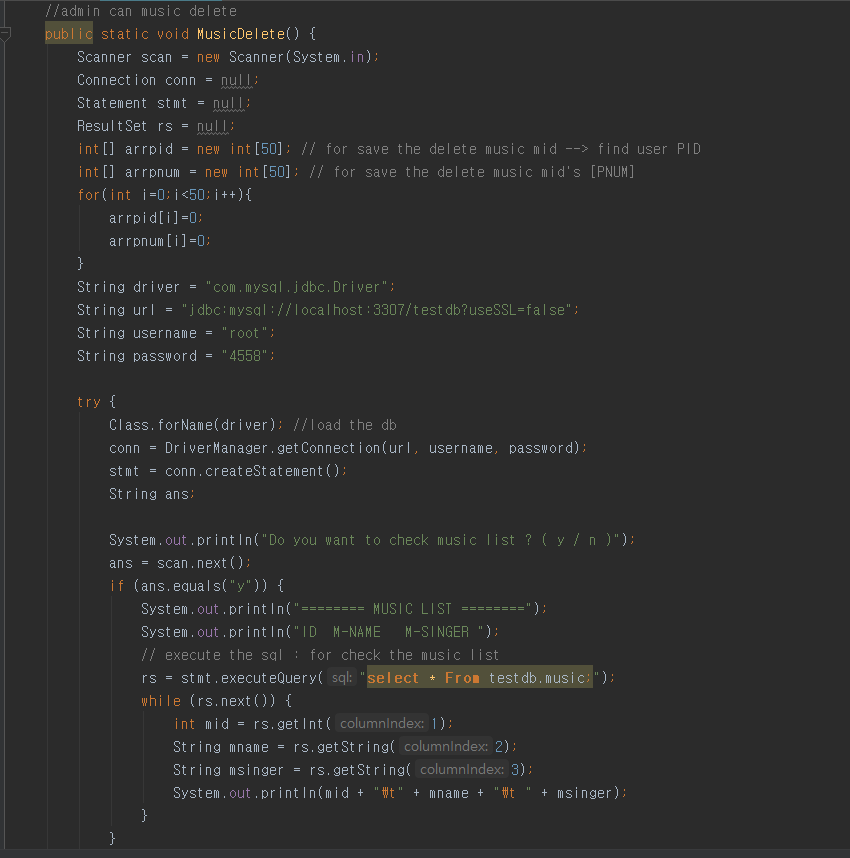




* **2-2 adminbutton == 2 관리자 음악 삭제**

관리자가 MUSIC table에 있는 음악을 삭제할 수 있습니다. MusicDelete() 함수를 사용합니다.

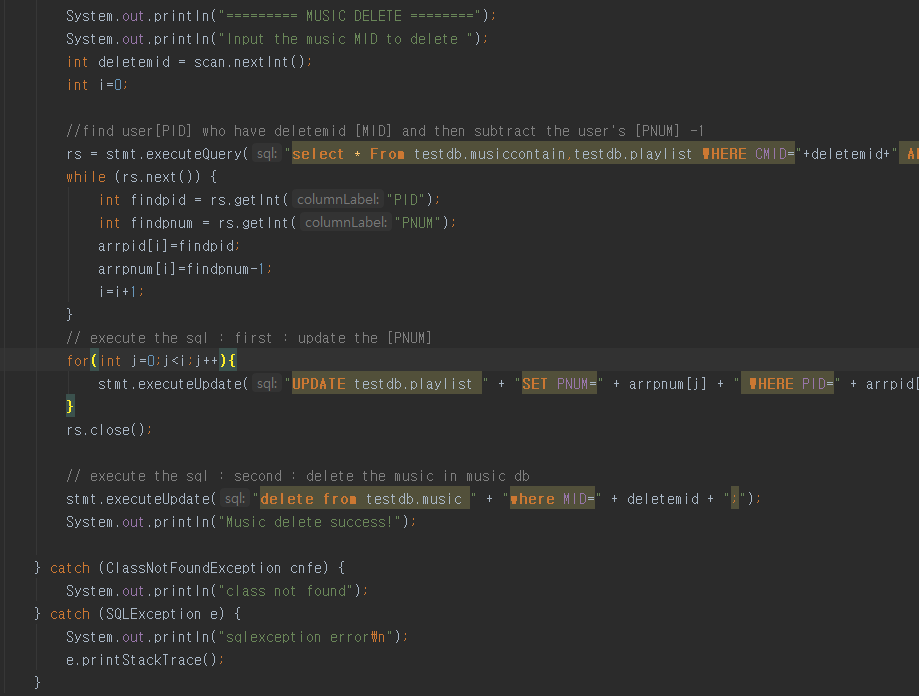
MusicDelete ()



db접근을 위한 connection과 load를 먼저 수행해줍니다

music delete는 처리해줘야 할게 많았습니다.

먼저 현재 music store에 있는 노래 리스트를 보고 싶냐고 묻습니다. “y”라고 하면 MUSIC LIST를 보여주고 “n”라고 하면 생략합니다.



먼저 삭제하고 싶은 music 의 mid 번호를 입력 받습니다.

첫 번째 처리) 관리자는 음악을 music table에서 삭제하면 사용자들이 자신의 play list 에 담아뒀던 음악도 삭제됩니다. 🡪 사용자들이 가지고 있는 음악의 개수가 줄어듭니다 🡪 playlist table의 PNUM 칼럼을 수정해줘야 합니다.

* Testdb.musiccontain과 testdb.playlist을 join하고 SELECT문을 사용해서 CMID=deletemid && CPID = PID를 만족하는 모든 행을 찾습니다. 그 행들의 PID과 PNUM 정보를 각각 findpid[], findpnum[]배열에 저장합니다.
* findpnum[]에 저장한 사용자의 music num의 수에서 -1 한 값으로 Update 해줍니다.

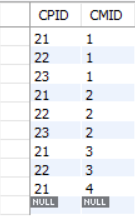
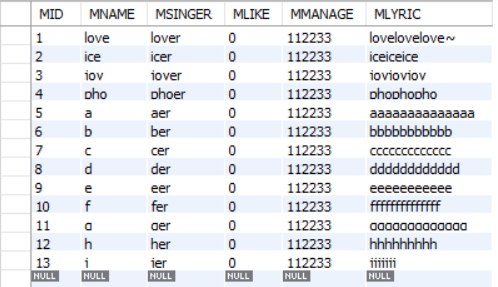
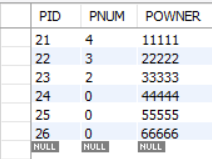
두 번째 처리 ) 관리자는 음악을 삭제합니다.

* Testdb.music table에 행 중 입력 받은 [deletemid]변수와 일치하는 MID 을 찾아서 삭제해줍니다.

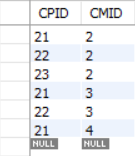
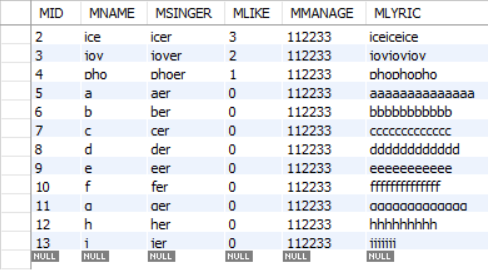
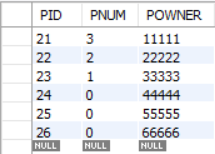


BEFORE 상태

Musiccontain music playlist

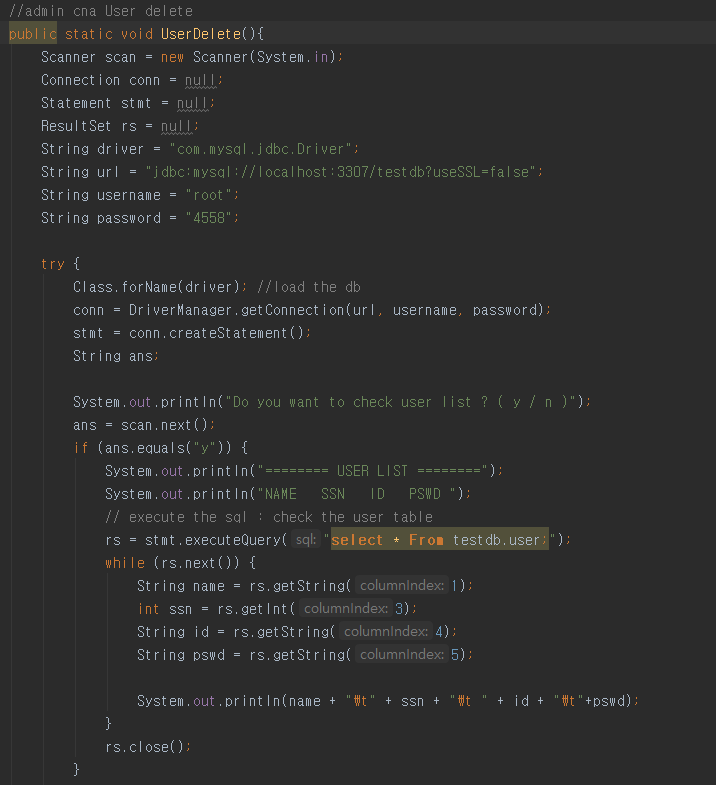
AFTER 상태

* **2-3 adminbutton == 3 관리자가 사용자를 강제로 탈퇴**

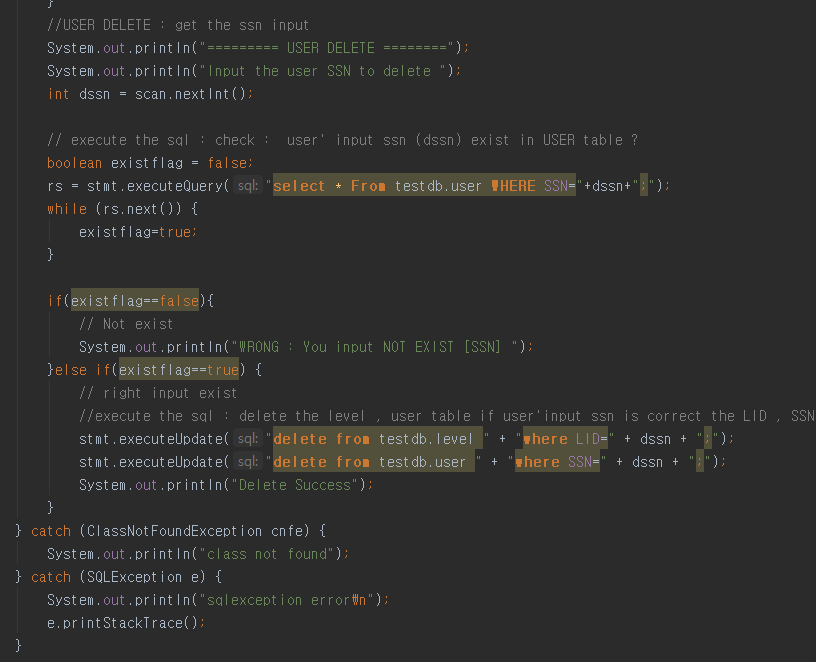
관리자가 사용자를 강제로 탈퇴시킵니다. User,level,playlist,musiccontain table 에 있던 모든 사용자 정보가 지워집니다. UserDelete() 함수를 사용합니다.

UserDelete()



db접근을 위한 connection과 load를 먼저 수행해줍니다

먼저 현재 USER table에 있는 사용자 리스트를 보고 싶냐고 묻습니다. “y”라고 하면 USER LIST를 보여주고 “n”라고 하면 생략합니다



삭제할 사용자의 SSN을 입력 받습니다.

* 입력 받은 ssn과 testdb.user table의 SSN칼럼 중 일치하는 값을 찾습니다. rs.next()의 값이 존재해서 값이 true가 된다면 existflag를 true로 바꿔줍니다.

if조건 문을 사용해서 관리자가 입력한 ssn정보를 가진 사용자가 존재하면 delete sql을 실행하고 존재하지 않으면 WRONG 경고문을 띄어줍니다.

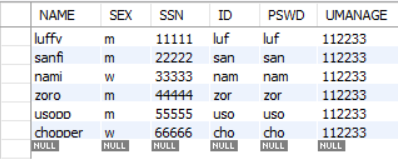
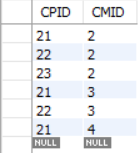
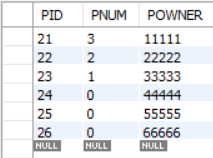
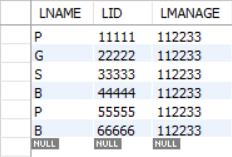
* 입력 받은 ssn값과 일치하는 LID 를 찾아 testdb.level에 있는 행을 삭제하고, ssn값과 일치하는 SSN을 찾아 testdb.user에 있는 행을 삭제합니다.

삭제를 성공하면 Delete Success 문을 출력합니다.

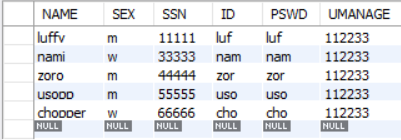
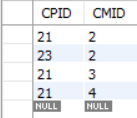
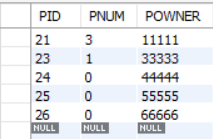
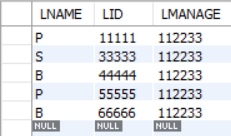


BEFORE 상태

User musiccontain playlist level

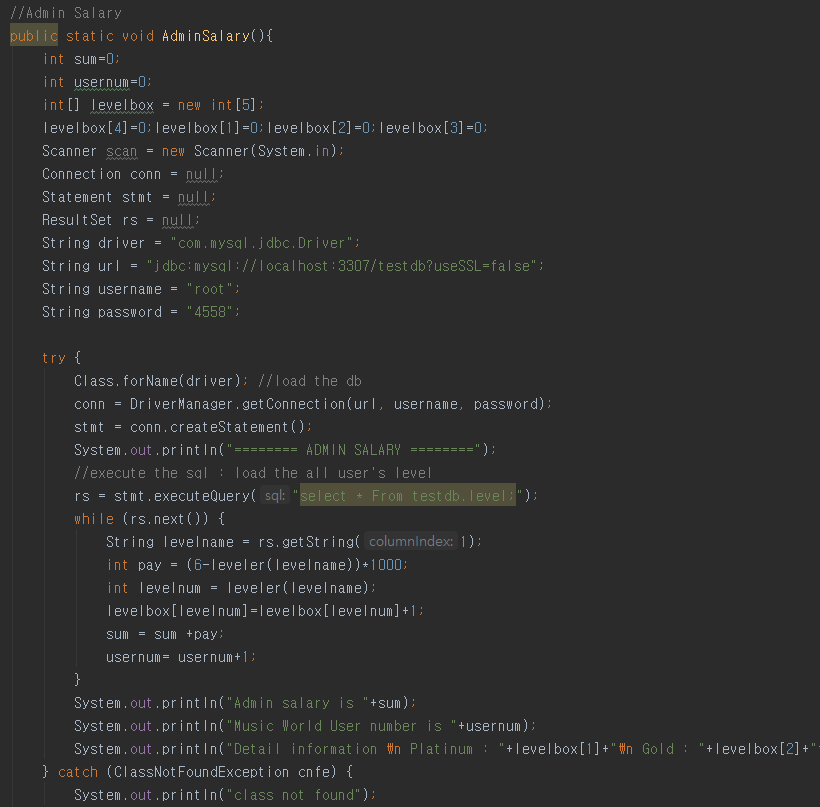
AFTER 상태

* **2-4 adminbutton==4 관리자 월급 체크**

Mucis World 사용자들의 수와 레벨 별 사용자들의 수를 파악할 수 있고 이에 따른 관리자의 월급을 계산할 수 있습니다. AdminSalary() 함수를 사용합니다.

AdminSalary ()

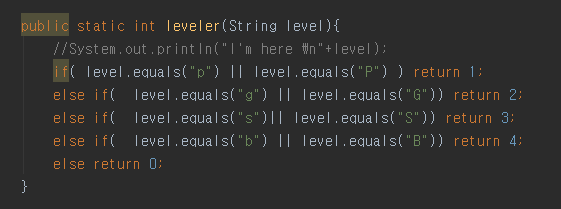


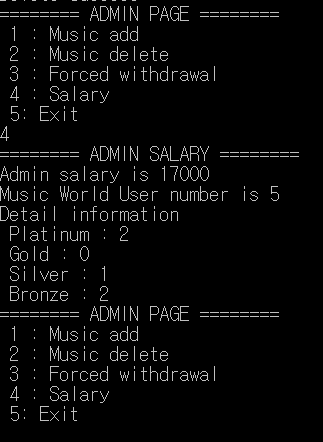
db접근을 위한 connection과 load를 먼저 수행해줍니다

레벨 별 사용자들의 수를 계산하기 위해서

* Testdb.level 테이블 전체를 select 합니다. 첫 번째 칼럼을 불러와서 levelname 변수에 저장하고 lever()함수를 이용해서 값을 리턴 받아 분류해 월급을 계산합니다.

Lever() 함수는 String 을 입력 받아 알맞은 정수 값으로 return 해주는 함수입니다.





* 2-5 adminbutton==5 관리자 페이지 나가기

Adminflag의 값을 false로 변경해서 ADMIN PAGE의 while() 문을 끝내고 START PAGE 로 이동합니다.



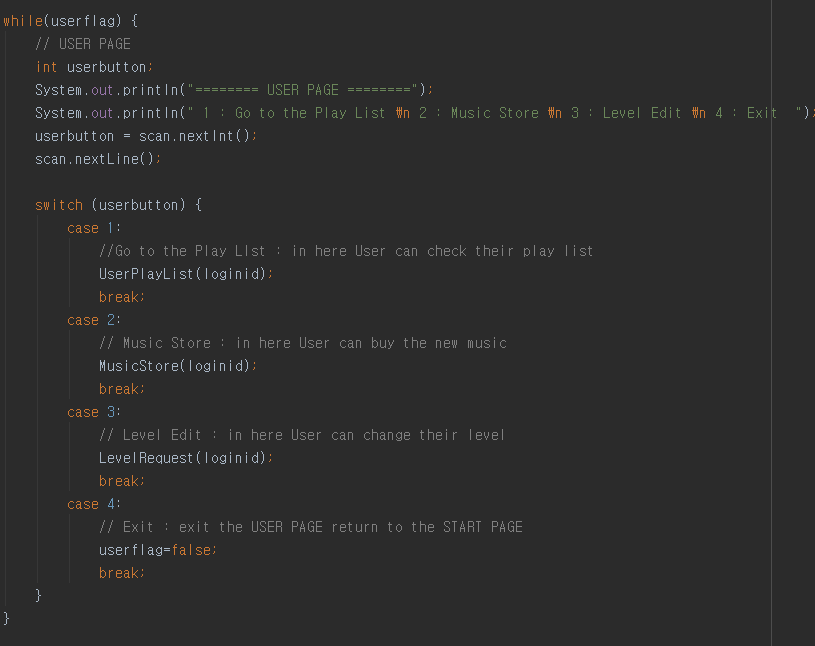


* 3 ======= USER PAGE =======

사용자가 로그 인을 성공했으면 START PAGE 에서 USER PAGE로 넘어옵니다.

USER PAGE에서는 userbutton을 이용해 원하는 기능을 선택합니다.

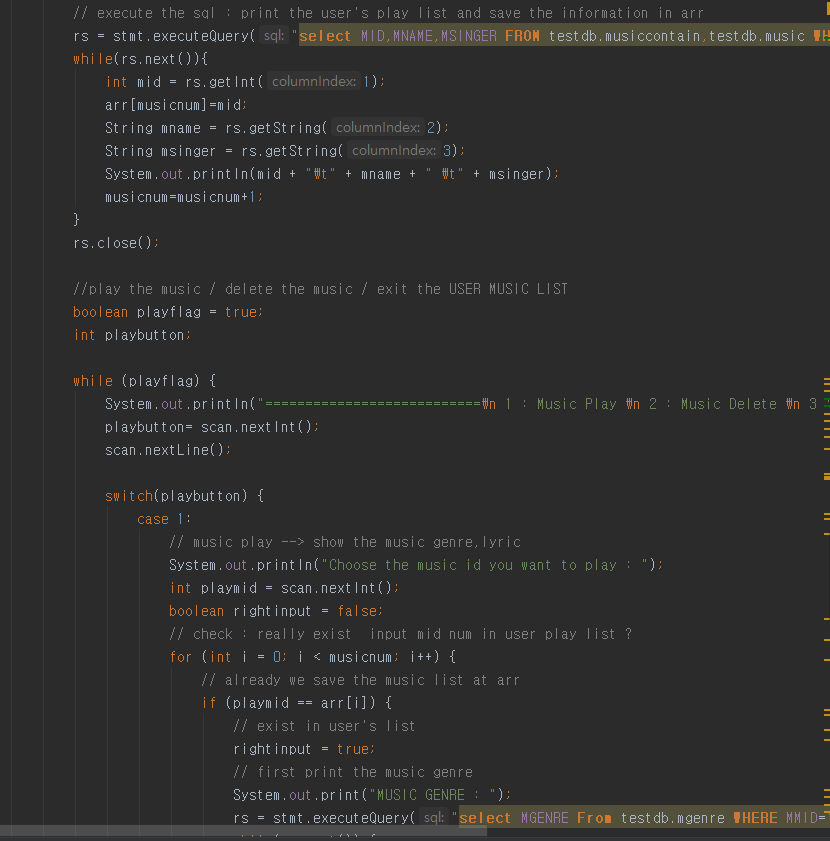
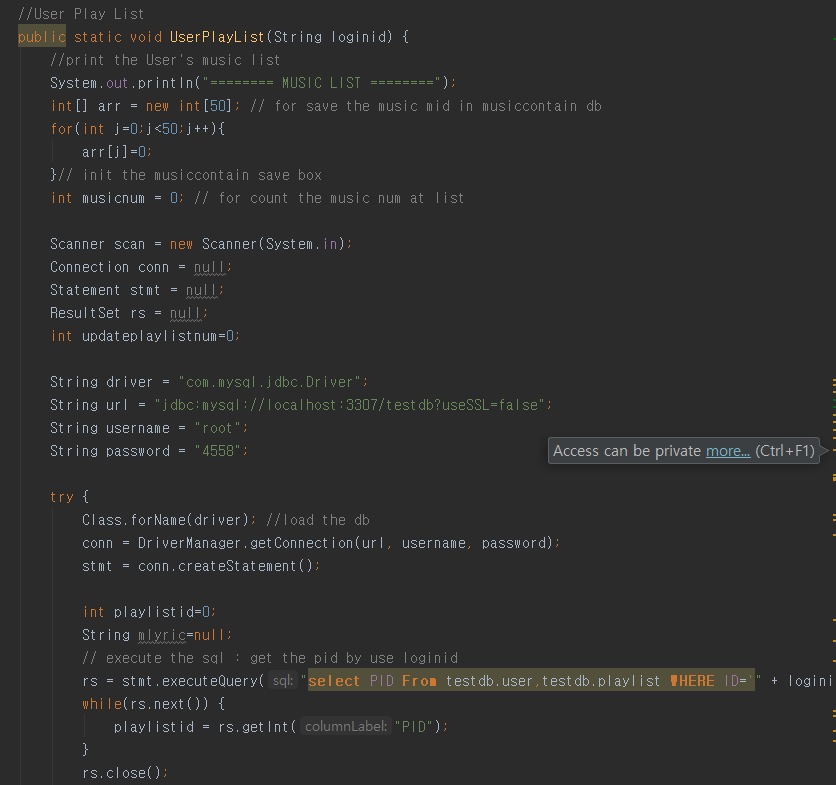
플레이리스트로 이동하거나 음악 상점에 가거나 레벨을 변경할 수 있습니다.



* 3-1 userbutton == 1 사용자 플레이리스트 보기

사용자가 음악 상점에서 구입한 음악의 목록을 보여줍니다.

음악을 재생할 수 있습니다. / 음악을 삭제할 수 있습니다. / playlist 페이지를 나갑니다.



db접근을 위한 connection과 load를 먼저 수행해줍니다

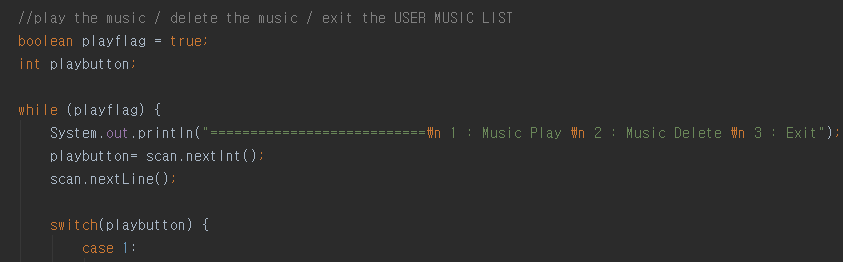
사용자가 로그인할 때 사용한 loginid를 함수의 파라 미터로 받아옵니다.

우선 사용자의 플레이 리스트 안에 들어있는 음악들의 정보를 간단하게 보여줍니다.

* User 테이블과 playlist 테이블을 합친 후 SSN=POWNER 과 ID=loginid를 만족하는 행의 PID 칼럼을 가지고 옵니다. 그 값을 [playlistid]변수에 저장합니다.
* Music 테이블과 musiccontain 테이블을 합친 후 CMIN=MID 와 CPID=playlistid 조건을 만족하는 1,2,3칼럼을 미리 만들어놓은 arr[]에 1번columindex를 저장하고 mname에 2번, msinger에 3번을 저장해 음악정보를 출력합니다.

플레이 리스트에 있는 음악을 play 할 수 있고 음악을 삭제할 수 있고 나갈 수 있습니다.

[play button]을 이용합니다.



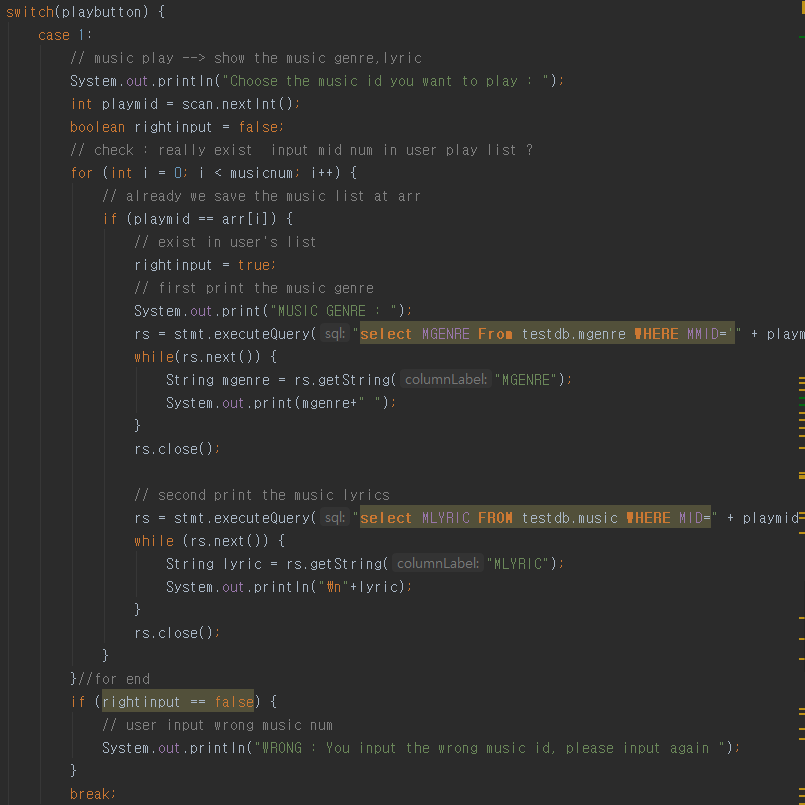
* 3-1-1 playbutton == 1 사용자는 playlist에 있는 음악을 재생할 수 있습니다.

재생을 할 음악의 id를 입력 받아 [playmid]변수에 저장합니다.

추가 기능) 사용자가 입력한 id가 실제로 본인의 playlist에 있는 음악인지 체크합니다.

맨 처음 playlist를 출력했을 때 저장한 arr[]배열을 확인합니다.

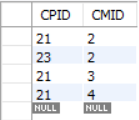
배열 안에 있는 정보와 [playmid]정보가 일치하면 음악에 대한 상세한 정보를 출력해줍니다.



음악에 대한 상세한 정보인 장르와 가사를 출력해줍니다.

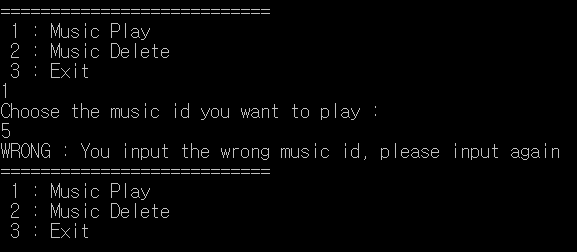
* Testdb.mgenre 테이블에서 MMID칼럼=playmid가 같은 행의 MGENRE를 모두 출력합니다. 장르가 한 개 이면 while문을 1번 돌고 장르가 2개이면 while문을 2번 돌기 때문에 정상적인 출력이 가능합니다.
* Testdb.music 테이블에서 MID칼럼=playmid가 같은 행의 MLYRIC칼럼을 출력합니다.

Luffy의 playlist에는 2,3,4번 노래가 있습니다. Luffy의 CPID는 21입니다.





Luffy 의 playlist 에는 5번 뮤직이 없습니다.

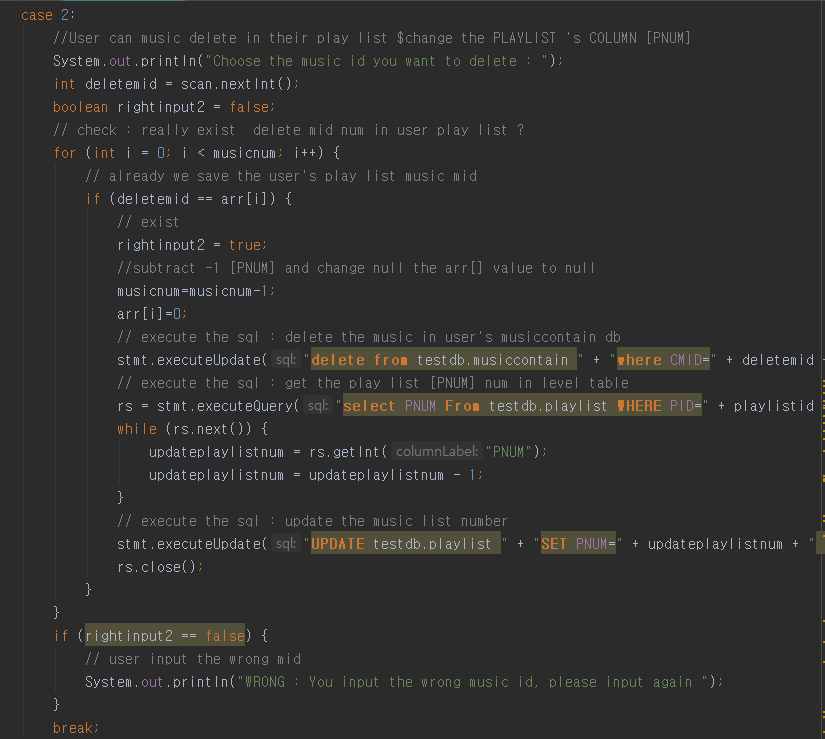


* 3-1-2 playbutton == 2 사용자는 playlist에 있는 음악을 삭제할 수 있습니다.

삭제할 음악의 id를 입력 받아 [deletemid]변수에 저장합니다.

추가 기능) 사용자가 입력한 deletmid가 playlist에 있는 음악인지 체크합니다. Rightinput2 플래그를 사용해 if조건 문을 걸어줍니다.

맨처음 playlist를 출력했을 때 저장한 arr[]배열을 확인합니다.

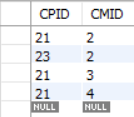
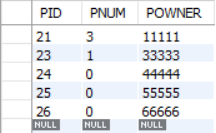


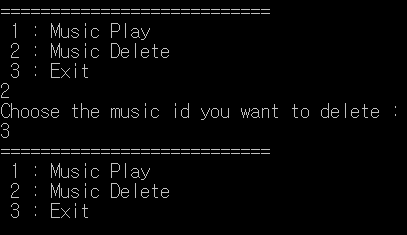
사용자 playlist에서 음악이 삭제되면 playlist 테이블에 있는 [PNUM]의 수도 변경해줘야 합니다. [Updateplaynum]변수에 기존에 있던 숫자를 받아 -1연산을 해서 저장해줍니다.

* Testdb.musiccontain 테이블에 있던 CMID=deletemid 조건과 CPID=playlistid 조건을 둘 다 만족하는 행을 삭제합니다.
* 기존에 있던 [PNUM]의 정보를 가져오기 위해 select문을 사용합니다. PID=playlistid 조건을 만족하는 PNUM칼럼을 testdb.playlist 테이블에서 가져옵니다.
* PID=playlistid 조건을 만족하면 -1연산을 한 [updateplaylistnum]변수를 testdb.playlist 테이블에 있는 PNUM칼럼에 UPDATE를 해줍니다.

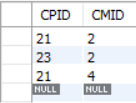
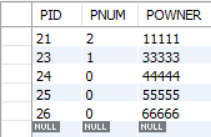
Luffy의 play list에는 1,2,3음악이 있습니다.

Luffy의 CPID는 21번입니다. PNUM이3입니다.

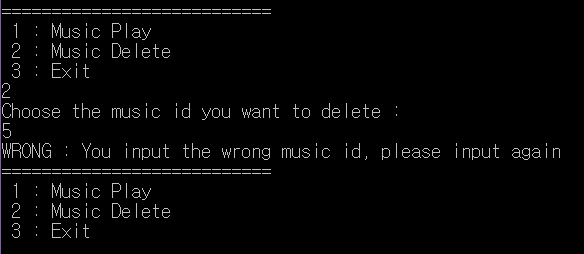
 



삭제 후 musiccontain 테이블이 업데이트 되었습니다.

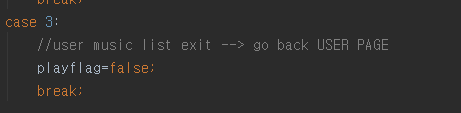
 

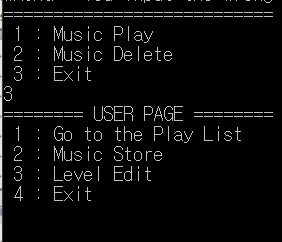
Luffy의 playlist 에는 5번 음악이 없습니다.



* 3-1-3 playbutton == 3 사용자는 playlist페이지에서 나갈 수 있습니다.

USER PAGE로 돌아갑니다.





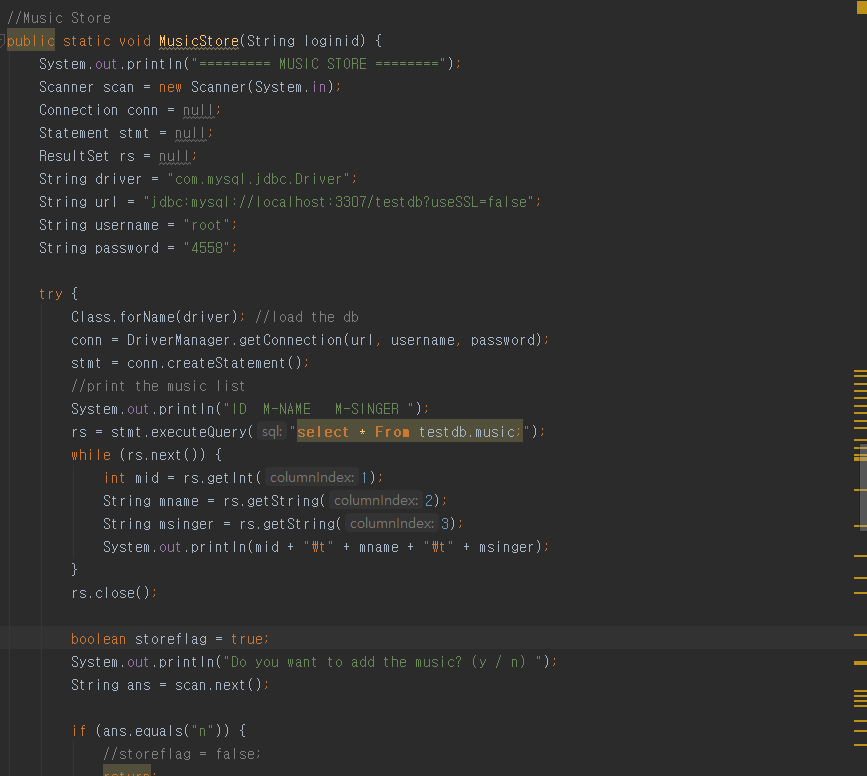
* **3-2 userbutton == 2 음악 상점 가기**

사용자는 음악상점에 가서 음악리스트를 확인하고 음악을 구입할 수 있습니다.

조건) 사용자의 레벨에 따라 playlist의 크기가 다릅니다.

P플레: 40개, G골드: 30개, S실버: 20개, B브론즈: 10개

만약 luffy의 레벨이 B이고 playlist에 음악이 10개가 있다면 꽉 찼다고 경고 창을 띄어줍니다.

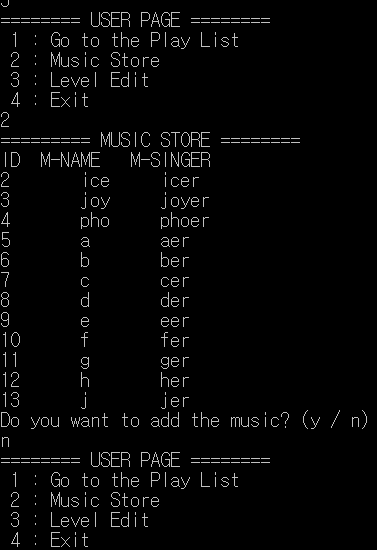


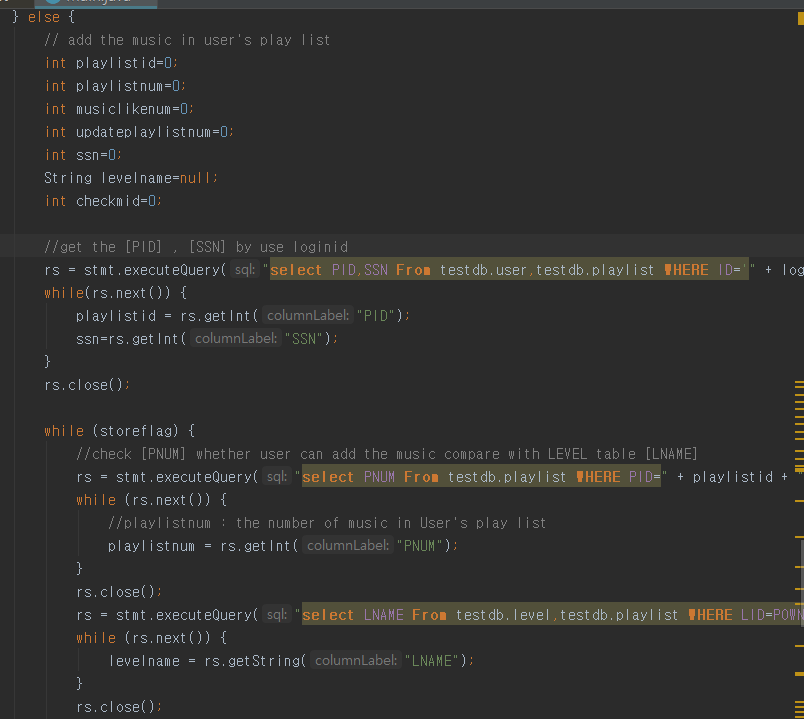
db접근을 위한 connection과 load를 먼저 수행해줍니다

우선 상점에 있는 모든 음악들을 출력해줍니다.

* Testdb.music 테이블에 접근해서 모든 칼럼을 가지고 옵니다. 각 행의 1,2,3 칼럼인 MID, MNAME, MSINGER정보를 출력해줍니다.

사용자는 음악리스트를 체크하고 음악을 추가할건지에 대한 대답을 합니다. “y”라고 하면 음악을 추가하기 위한 정보를 입력 받고 “n”라고 하면 USER PAGE로 이동합니다.





“y”라고 했으면 [loginid]파라 미터를 이용해서 playlistid와 ssn정보를 얻어옵니다.

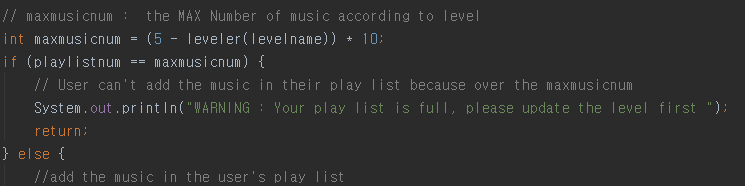
* user테이블과 playlist 테이블을 합쳐 ID=loginid 조건과 SSn=POWNER인 PID와 SSN정보를 가져옵니다.

사용자의 [PNUM]칼럼을 확인해서 음악을 추가할 수 있는지 체크합니다.

* PID=playlistid 조건을 만족하는 PNUM 칼럼을 testdb.playlist 테이블에서 가져와 [playlistnum]변수에 저장합니다.
* Leve, playlist 테이블을 합쳐 LID=POWNER 조건과 POWNER=ssn조건을 만족하는 [LNAME]칼럼 정보를 가져옵니다. [levelname]변수에 저장합니다.

[levelname]정보를 이용해 사용자의 레벨에 따라 담을 수 있는 max음악 개수를 구해 [maxmusicnum]변수에 저장합니다.

Playlistnum 과 maxmusicnum 수를 비교해서 사용자가 음악을 추가할 수 있는지 체크합니다.

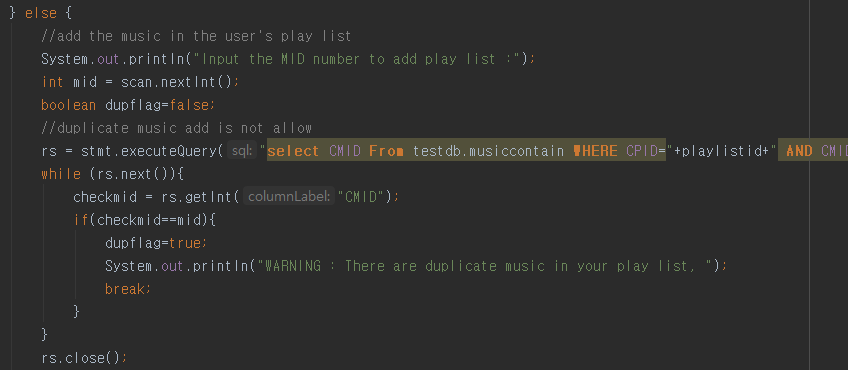


음악을 추가할 수 있으면 추가하고 싶은 음악의 MID를 입력 받아 mid변수에 저장합니다.

추가 기능) 입력 받은 mid값이 사용자가 이미 playlist에 담은 곡인지 중복 체크합니다.

* CPID=playlistid조건과 CMID=mid조건을 만족시키는 CMID값을 testdb.musiccontain테이블에서 찾아 옵니다.

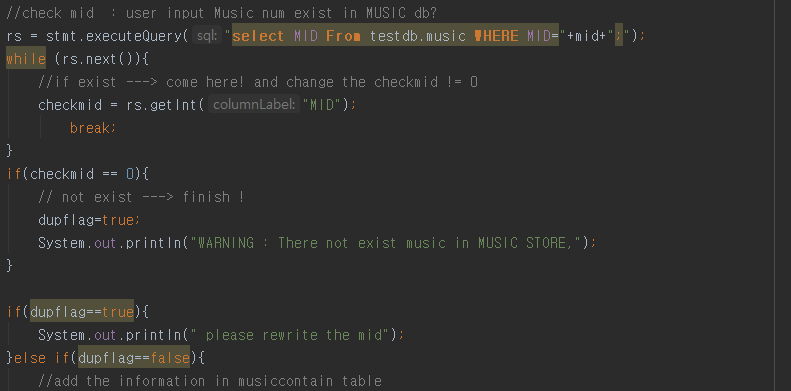
그 값을 [checkmid]변수에 저장합니다. if조건 문을 사용해서 사용자가 입력한 mid넘버와 checkmid넘버를 비교합니다. 동일하면 dapflage를 참으로 변경하고 중복된 노래가 이미 플레이 리스트에 있다는 경고 창을 띄어줍니다.



추가 기능) 중복처리를 해준 입력 값 mid가 music store에 있는 값인지 체크합니다.

* mid 값과 testdb.music에 있는 MID칼럼과 값이 같은 행이 존재하면 checkmid에 값을 저장합니다.

존재하지 않으면 그런 음악이 없다는 경고 창을 띄어줍니다.



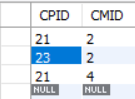
중복처리와 올바른 값을 입력 받았으면 sql 문을 실행해서 입력 받은 mid를 user의musiccontain table에 삽입합니다.

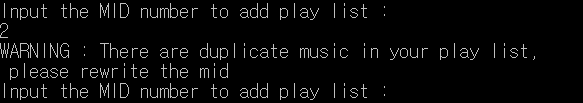


* playlistid와 mid값을 testdb.musiccontain 테이블에 삽입합니다.
* Testdb.music 테이블에 있는 MID=mid인 행의 MLIKE칼럼의 값을 가져와
* +1 연산을 한 뒤 UPDATE를 합니다.
* Testdb.playlist 테이블에 있는 PID=playlistid인 행의 PNUM칼럼의 값을 가져와
* +1 연산을 한 뒤 UPDATE를 합니다.

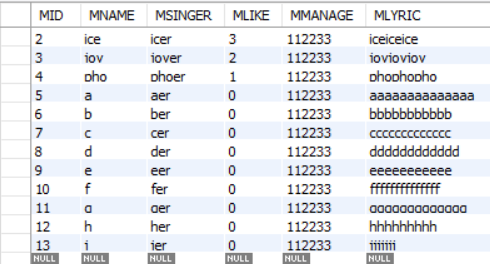
추가할 음악이 더 있냐고 물어보고 “y” / “n” 답변을 moreans에 저장해 if조건 문을 처리해서 반복합니다.

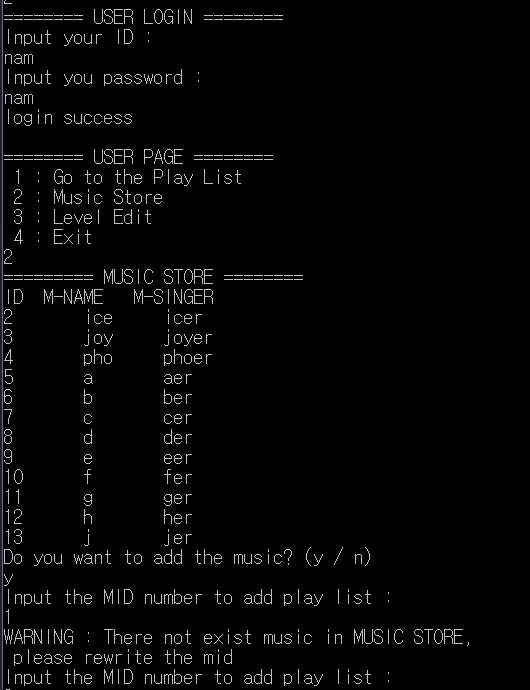
Nami의 PID는 23입니다. Nami는 2번 노래를 가지고 있습니다. 🡪중복처리



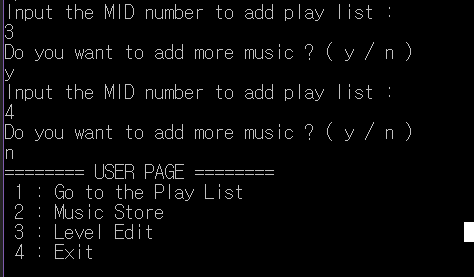


현재 music store 에는 1번 노래가 존재하지 않습니다. 🡪올바른 입력값처리

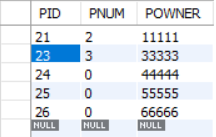
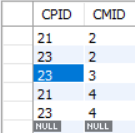




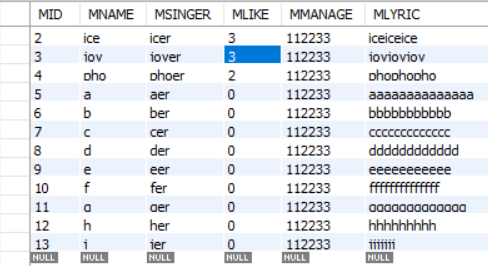
3,4번 노래의 추가는 성공했습니다. PNUM의 개수가 2증가했고, musiccontain 테이블에도 2개의 항목이 추가되었습니다.



Playlist musiccontain

3,4번 노래의 MLIKE값이 1씩 증가했습니다.



* **3-3 userbutton == 3 사용자 레벨 변경하기**

LevelRequest() 함수를 이용합니다.

db접근을 위한 connection과 load를 먼저 수행해줍니다

일단 사용자의 loginid정보를 이용해 현재 레벨과 SSN을 파악합니다.

* ID = loginid조건과 SSN=LID조건을 만족하는 행을 testdb.user, testdb.leve에서 찾습니다. LNAME칼럼을 userlevelname 변수에 저장하고 SSN칼럼을 ssn변수에 저장합니다.

ssn변수를 이용해서 현재 사용자가 가지고 있는 음악의 개수를 찾아 currentmusicnum변수에 저장합니다.

* POWNER=ssn조건을 만족하는 PNUM칼럼을 testdb.playlist테이블에서 찾습니다.

사용자의 현재 레벨 정보를 알려줍니다.

사용자는 레벨 upgrade와 downgrade를 선택할 수 있습니다.

사용자는 레벨 변경 페이지에서 나갈 수 있습니다.

Userlevelbutton을 사용해 사용자가 원하는 기능을 입력 받습니다.

Leveler() 함수를 사용해서 userlevelname을 알맞은 숫자로 반환 받습니다. userlevelnameNum

userlevelnameNum의 값이 1인 경우는 플레유저입니다.

플레유저는 더 이상 upgrade level을 할 수 없습니다.

플레유저는 G, S, B로 선택해서 downgrade level을 할 수 있습니다.

userlevelnameNum의 값이 2인경우는 골드유저입니다.

골드유저는 P로 upgrade level을 할 수 있습니다.

골드유저는 S, B로 선택해서 downgrade level을 할 수 있습니다.

userlevelnameNum의 값이 3인 경우는 실버유저입니다.

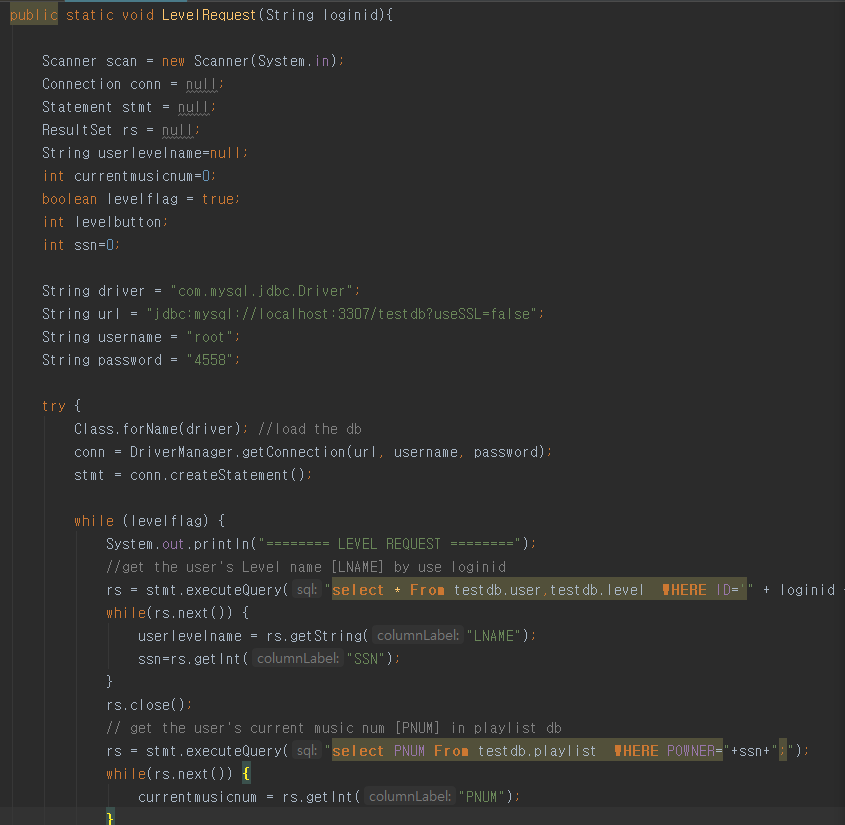
실버유저는 P, S로 선택해서 upgrade level을 할 수 있습니다.

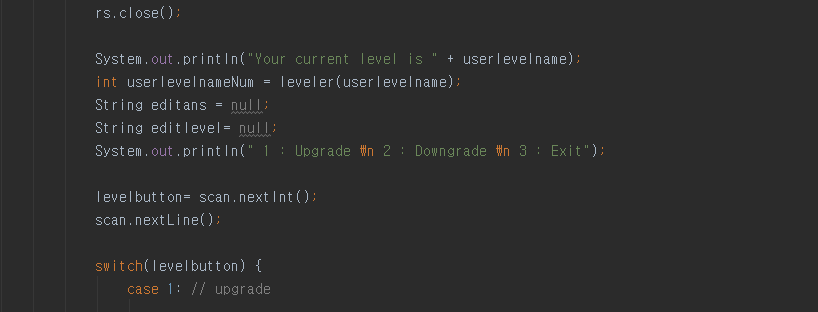
실버유저는 B로 downgrade level을 할 수 있습니다.

userlevelnameNum의 값이 4인 경우는 브론즈유저입니다.

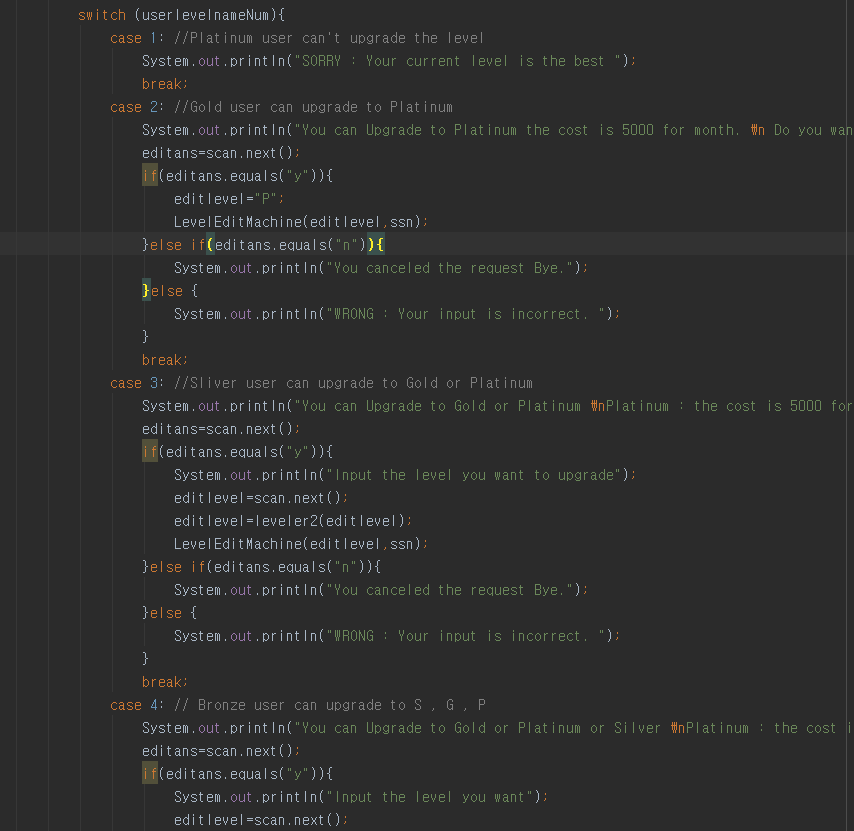
브론즈유저는 S, G, P로 선택해서 upgrade level을 할 수 있습니다.

브론즈유저는 더 이상 downgrade level을 할 수 없습니다.





* 3-3-1 userlevelbutton == 1 사용자 레벨 업그레이드 하기



* 3-3-2 levelbutton == 2 사용자 레벨 다운그레이드 하기

추가 기능)\*\*\*만약 사용자의 현재 레벨이 S인상 태에서 음악이 12개 있다고 가정해보자 이 사용자가 B로 레벨다운을 요청했을 경우 우선 음악을 먼저 삭제하라고 경고 창을 띄어준다.

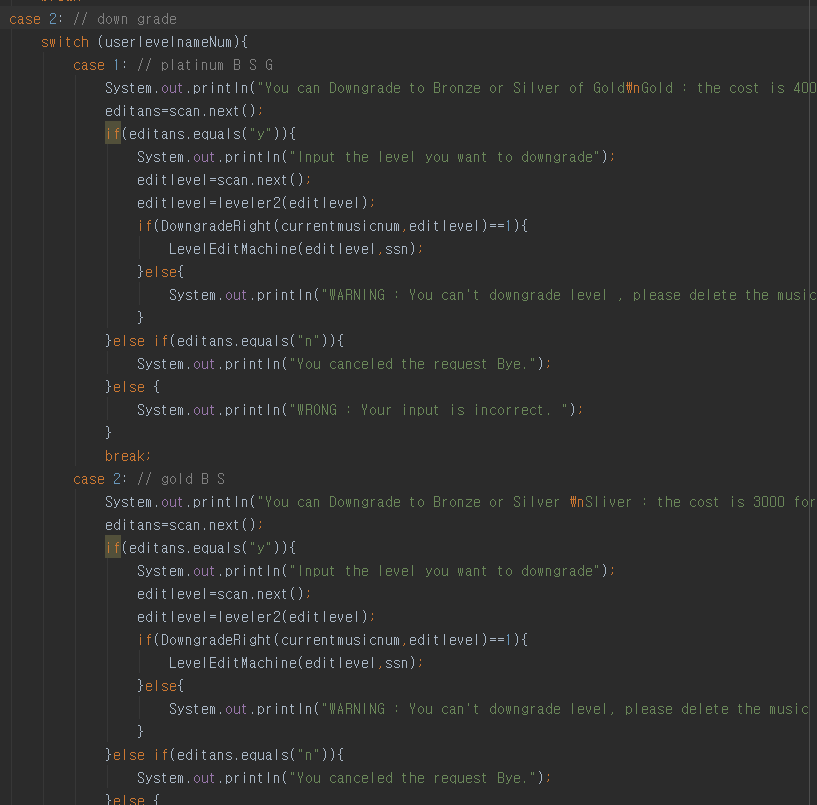
왜냐하면 사용자의 레벨에 따라 playlist 에 담을 수 있는 곡의 개수가 다르기 때문이다.

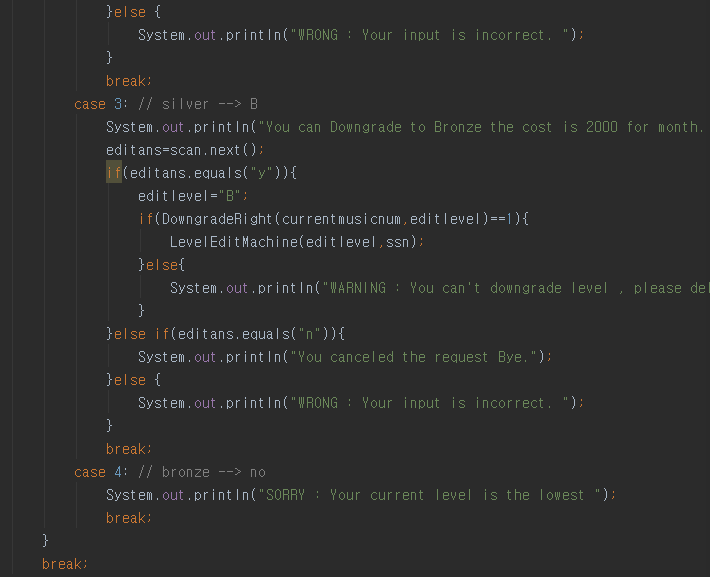
P: 40개

G: 30개

S: 20개

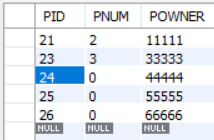
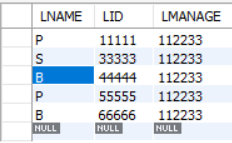
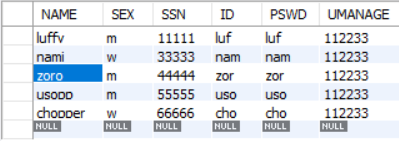
B: 10개



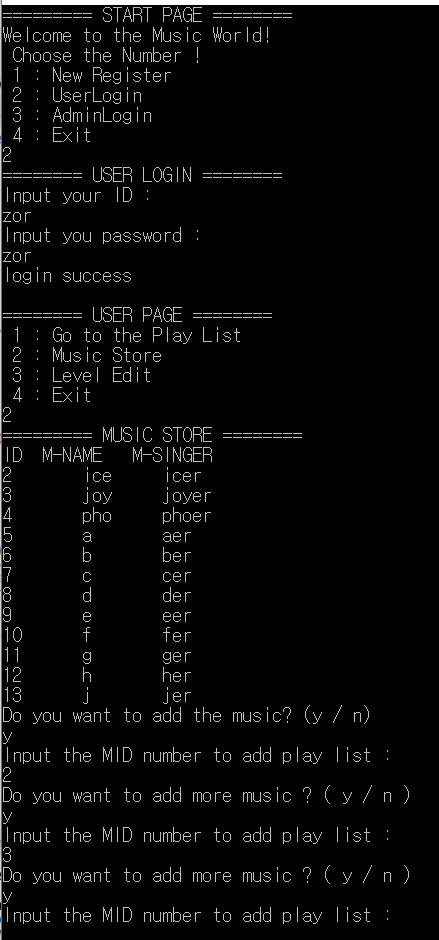


**UPGAGE실험**

Zoro의 PID는 24이다. Zoro의 level은 B이다. 10개의 음악을 담을 수 있다.

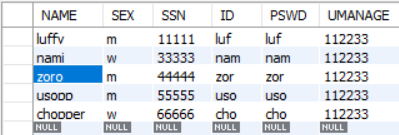
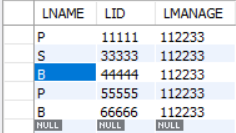
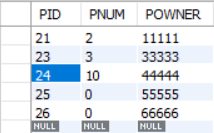


Zoro가 음악상점에 가서 음악을 10개를 구입한다.



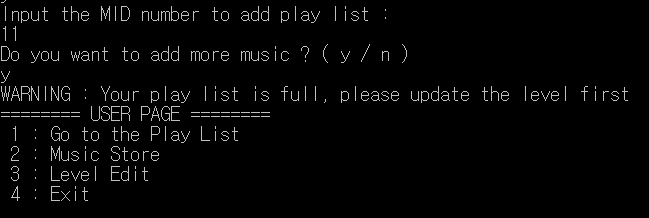
…. 2~11

결과 테이블

Zoro의 PNUM가 10이 되었습니다.

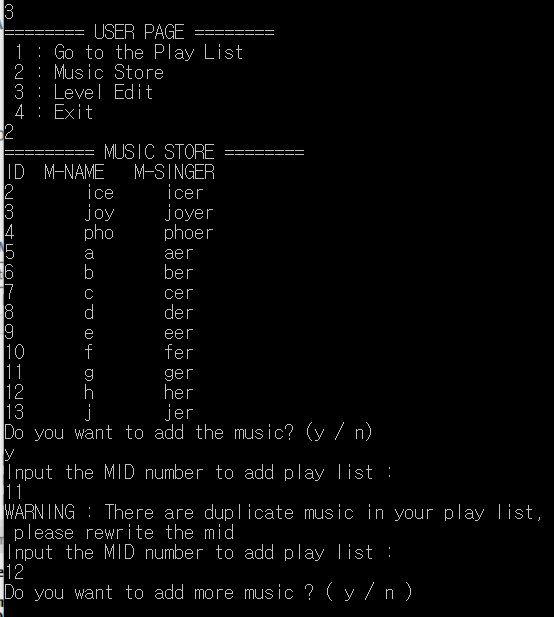
이 상황에서 음악을 더 구입해보겠습니다.



사용자의 플레이 리스트가 다 찼다는 경고 창을 받았습니다.

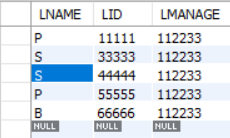
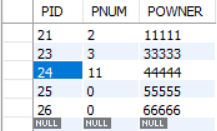
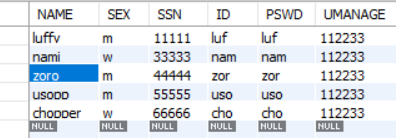
Zoro의 레벨을 S로 바꾸로 음악을 다시 한번 더 구입해보겠습니다.





결과 테이블을 보겠습니다.

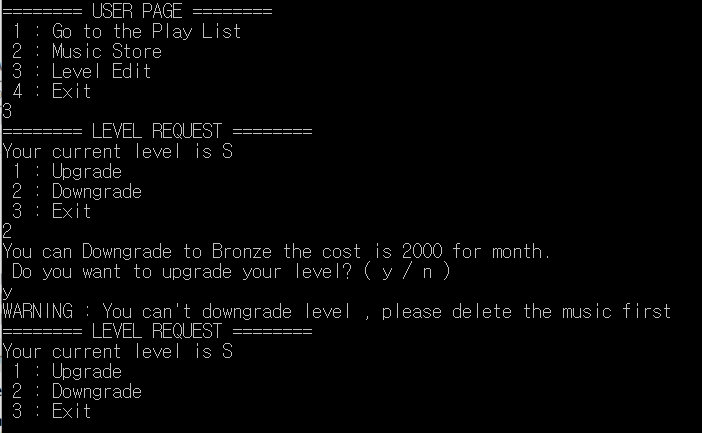
User playlist level



Zoro의 레벨이 B에서 S로 변경되었고 PNUM의 수가 11이 되었습니다.

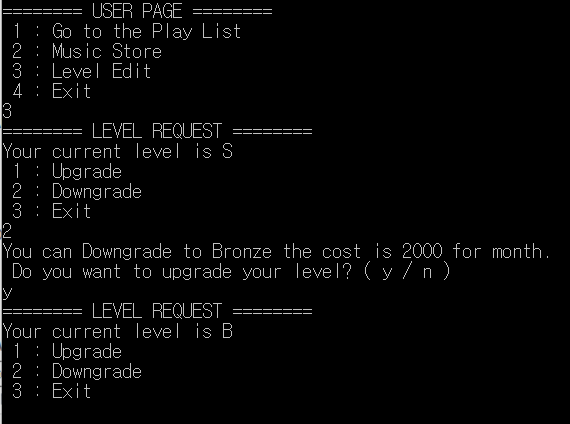
**DOWNGRADE실험**

Zoro가 B로 level down을 요청했으나 거부되었습니다.



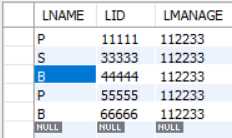
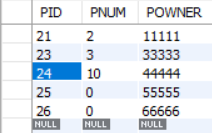
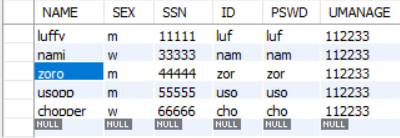
경고창의 말대로 B의 playlist 사이즈는 10이므로 1개의 곡을 삭제하고 다시 요청하겠습니다.





level요청을 성공적으로 수행했습니다.

결과 table을 확인하겠습니다.



Zoro의 PNUM이 10으로 셋팅되었고 레벨이 B로 바뀌었습니다.

LevelEditMachine() 함수를 정의했습니다. upgrade downgrade의 동일한 수행부분을 함수로 묶었습니다.



* 입력 받은 level정보를 파라 미터로 받은ssn을 이용해서 testdb.level 테이블의 LNAME칼럼의 정보를 UPDATE했습니다.

DowngradeRight() 함수를 정의했습니다. Downgrade를 요청한 사용자의 현재 playlist의 음악개수를 파라 미터로 받아서 limitnum을 구해 조건 문으로 비교했습니다. 변경할 수 없으면 return 0 변경할 수 있으면 return 1 값을 반환받습니다.

