DATABASE

2017-1학기

DATABASE TEAM PROJECT

<TEAM15>

컴퓨터공학과

1515004 권나현

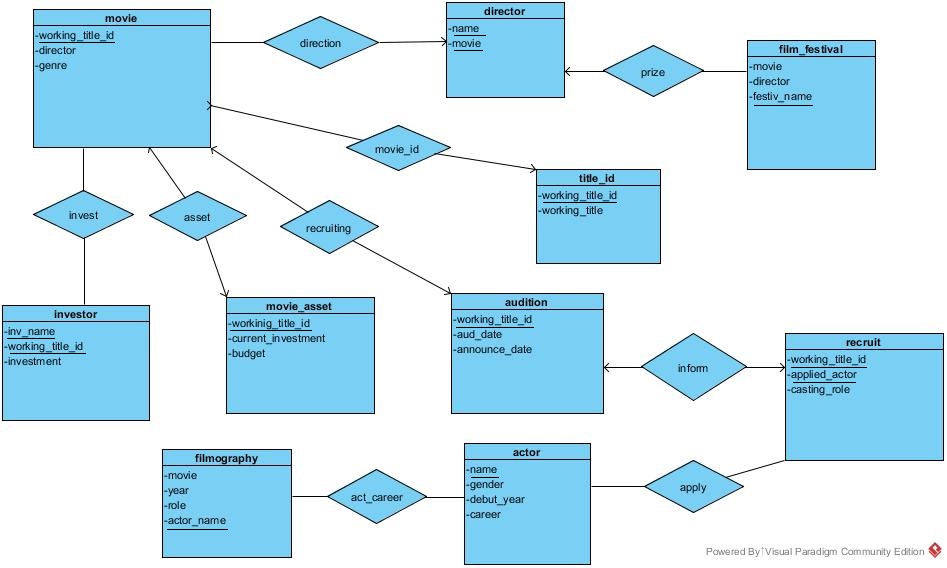
1515005 권성경

1515009 김다영

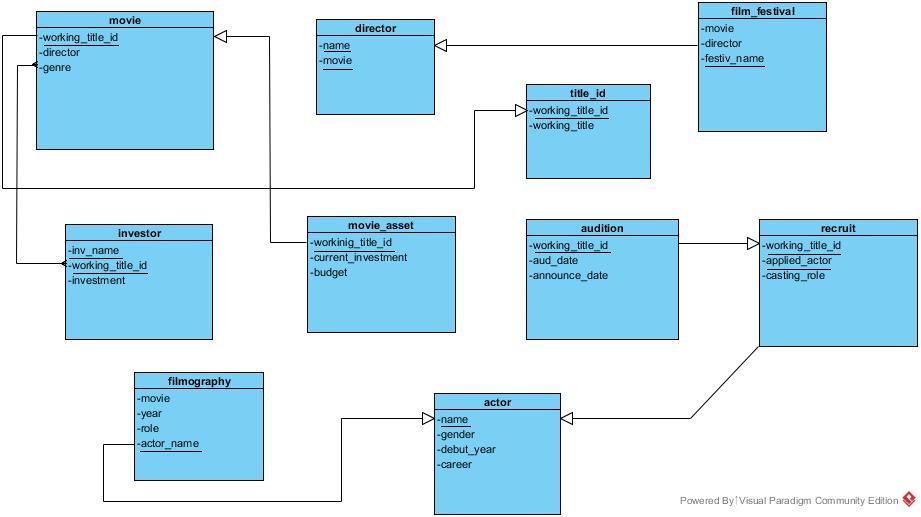
<TOPIC: 영화 제작 프로그램>

영화 제작을 목표로, 영화 제작에 필요한 요소 3가지인 감독, 배우, 투자사를 프로그램 사용자로 설정하여 각각의 역할에 맞게 필요한 기능들을 제공해주는 프로그램이다.

1. ER DIAGRAM



2. SCHEMA DIAGRAM



3. CLASS AND METHOD EXPLANATION

(a)Jdbcexam class:

Director, Actor, Investor 세 가지 버튼을 보여주는 UI를 보여준다. 자신의 직업에 맞게 세 가지 버튼 중 하나를 선택하면, 각각 director 창, actor 창, investor 창을 버튼에 따라 보여준다.

(a)DirectorFrame, DListener

감독이 새로운 영화를 찍는다고 가정하자. 감독은 가제, 자신의 이름, 예산, 장르를 입력하고 insert를 누른다. Insert를 누르면 감독이 새로운 감독일 경우 감독테이블에, 가제 아이디를 제공하여 가제 아이디 테이블에, 영화 테이블에 정보를 삽입한다. 만약 자신의 영화 관련 오디션이 완료되면 done을 누른다.

다시 새로운 영화를 찍는다고 하고, 오디션을 보려고 한다고 가정하자. 가제를 입력하고, 오디션 날짜와 오디션 발표 날짜를 입력하고 insert버튼을 누른다. 그럼, 오디션 테이블에 새로운 오디션 정보가 삽입된다.

감독이 자신의 수상 이력을 등록하고 싶다면, 자신의 수상 영화 제목, 감독 자신의 이름, 해당 수상을 한 영화제 이름을 입력하고 insert버튼을 누른다. 그럼, 필름 페스티벌의 테이블에 정보가 삽입된다.

감독이 현재 투자금을 알고 싶다면 자신의 가제 제목을 입력하고 check버튼을 누른다. 그럼, 영화 정보 테이블에서 현재 투자금 값을 가져와 감독에게 보여준다.

감독이 배우를 뽑고 싶다면, 우선, 자신의 가제를 입력하여 show버튼을 누른다. 그럼, 액터 테이블과 리쿠르트 테이블을 조인하고 해당 가제의 타이틀 아이디와 맞는 relations만 뽑아와 감독에게 액터 정보를 보여준다. 만약 액터의 필모그래피를 보고 싶다면 액터 이름을 입력하고 show버튼을 누른다(옆 패널). 그럼, 필모그래피 테이블에서 해당 배우의 필모그래피 값들을 가져와 보여준다. 감독이 마음에 든다면, 액터 이름을 입력하고(윗 패널), 자신의 가제, 그 배우에게 지정하고 싶은 역할을 입력하고 pick를 누른다. 그럼, 리쿠르트 테이블에 해당 배우의 role파트에 지정 역할이 들어간다.

(b)ActorFrame ,ActorListener

배우가 자신의 정보를 text field에 입력하고 insert버튼을 누르면 DB의 actor테이블에 정보를 삽입하고, delete버튼을 누르면 actor테이블에서 정보를 제거한다. 배우가 자신의 경력을 text field에 입력하고 insert 버튼을 누르면 DB의 filmography table에 정보를 삽입한다. 마찬가지로 delete버튼을 누르면 테이블에서 정보를 제거한다. 또한 사용자가 search버튼을 누르면 DB에 있는 영화 정보를 출력한다. 사용자가 자신의 정보를 입력하고 Apply 버튼을 누르면 DB의 recruit테이블에 사용자의 정보를 삽입하고 신청한 영화의 오디션 정보를 출력한다.

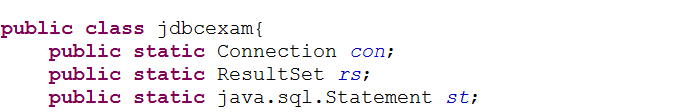
(c)InvestorFrame, IListener

투자자가 어떤 영화에 투자를 할 수 있는지 DB의 영화를 보여준다. 또한 그 영화를 만드는 감독이 어떤 상을 받았는지 정보를 보여준다. 투자자가 정보를 보고 영화를 선택해 text field에 정보를 입력하고, insert를 누르면 DB에 투자자 정보를 삽입한다. 사용자가 delete를 누른 경우, 투자자가 취소하고자 입력한 금액이 투자한 총 금액이라면 DB의 투자자 테이블에서 투자자의 정보를 완전히 삭제하고, 영화의 투자정보 테이블에서 영화에 투자된 금액을 그만큼 감소시킨다. 총 금액이 아닌 일부라면 DB의 투자자 테이블에서 투자자의 정보는 삭제하지 않고 투자 금액만 감소시킨다. 영화의 투자정보 테이블에서도 마찬가지로 투자금을 해당 액수만큼 감소시킨다

4. MAIN CLASS NAME, HOW TO RUN, CONNECTION CONFIGURATION INSTRUCTIONS

-Main class name: class jdbcexam

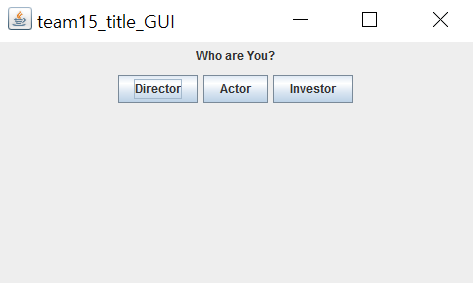
-Connection configuration instructions:





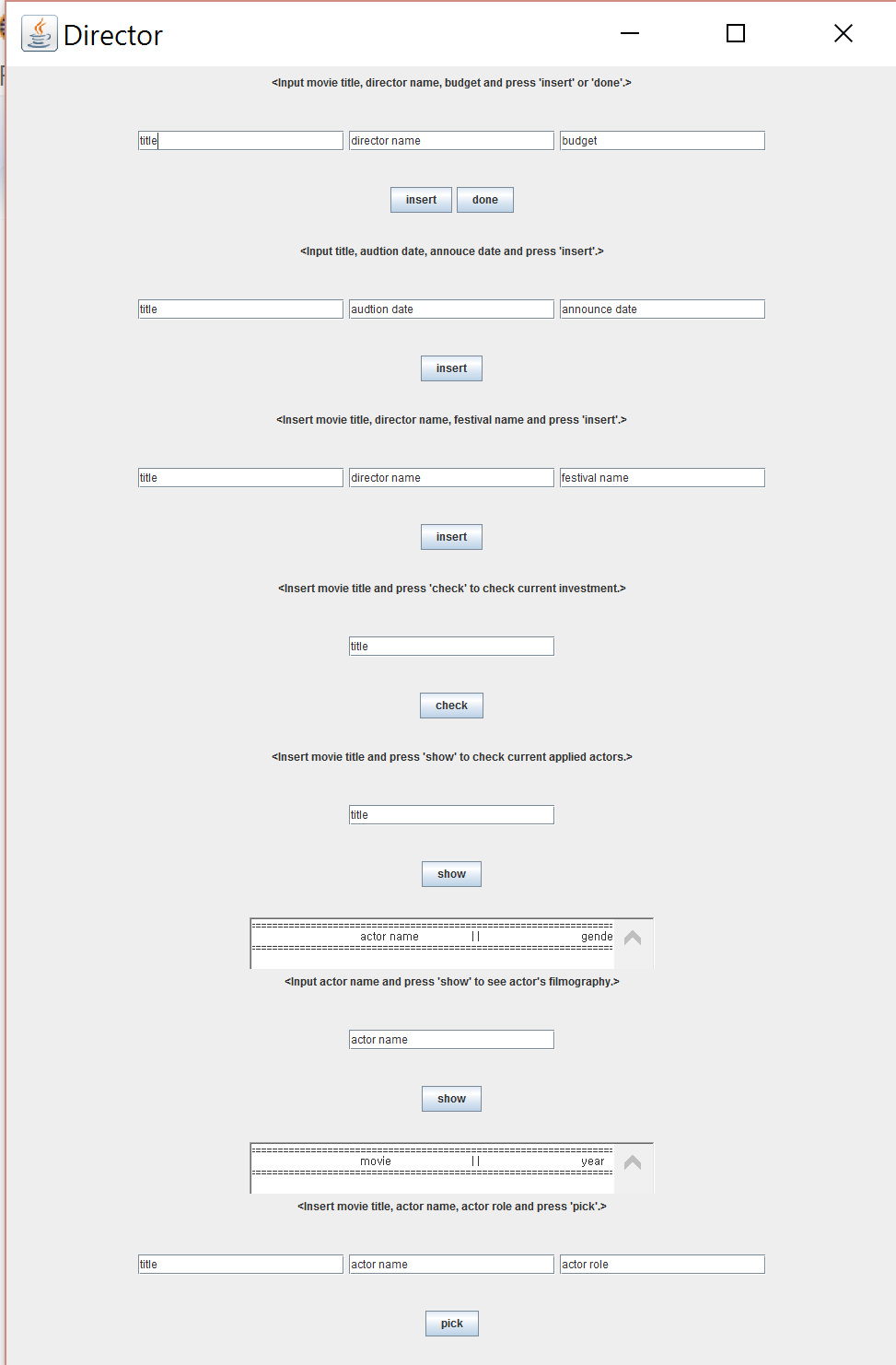
-How to run:

맨 처음 프로그램을 run 하면 3가지 사용자 중 원하는 메뉴를 선택할 수 있는 창이 나온다.



1. Director 버튼 클릭 시

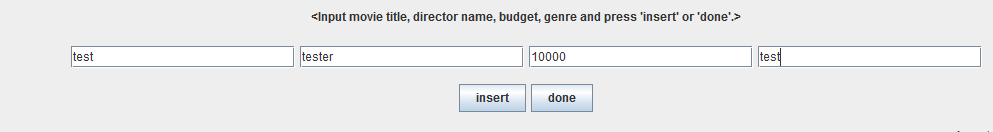
(처음 화면)



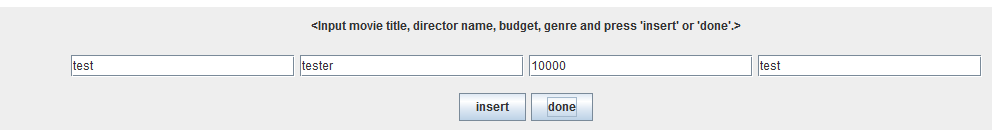
1. 1)영화 제목, 감독 이름, 예산을 작성하고 insert 버튼을 누르면 새로운 영화가 등록된다.

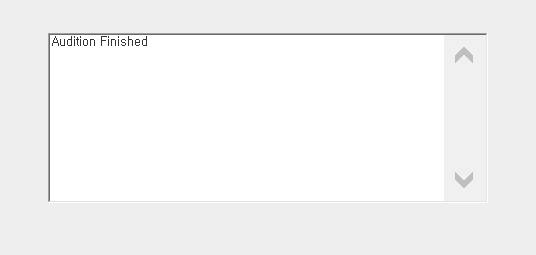
2)같은 정보를 작성하고 done 버튼을 누르면 오디션이 완료됐음이 들어간다.

1)

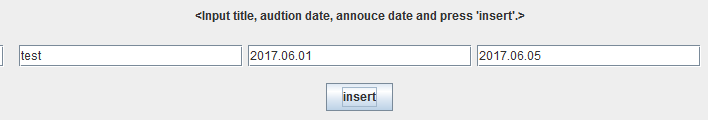


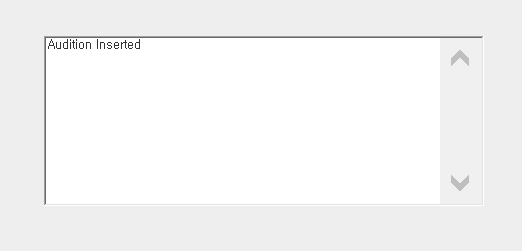


2) 

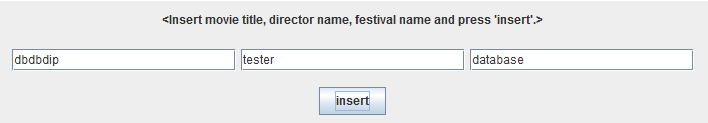


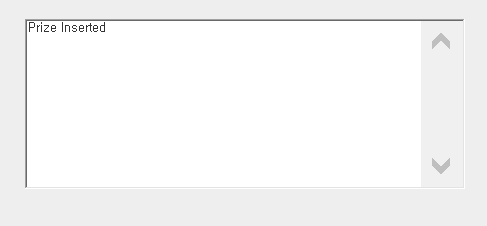
1. 영화 제목, 오디션 날짜, 발표 날짜를 입력하고 insert 버튼을 누르면 오디션이 등록된다.



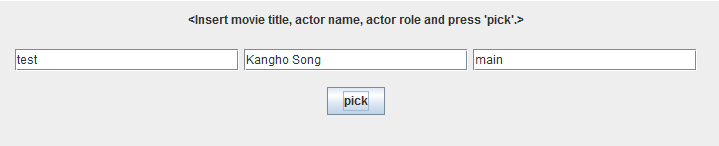


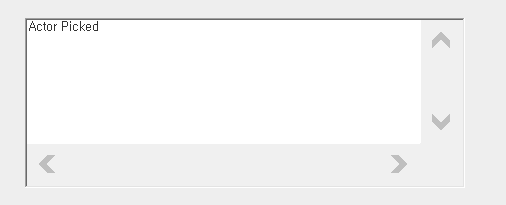
1. 감독이 자신의 수상작 제목, 감독 이름, 영화제 제목을 입력하고 insert 버튼을 누르면 자신의 수상 정보가 등록된다.





1. 영화 제목, 배우 이름, 배우에게 지정할 역할을 입력하고 pick 버튼을 누르면 배우를 얻게 된다.

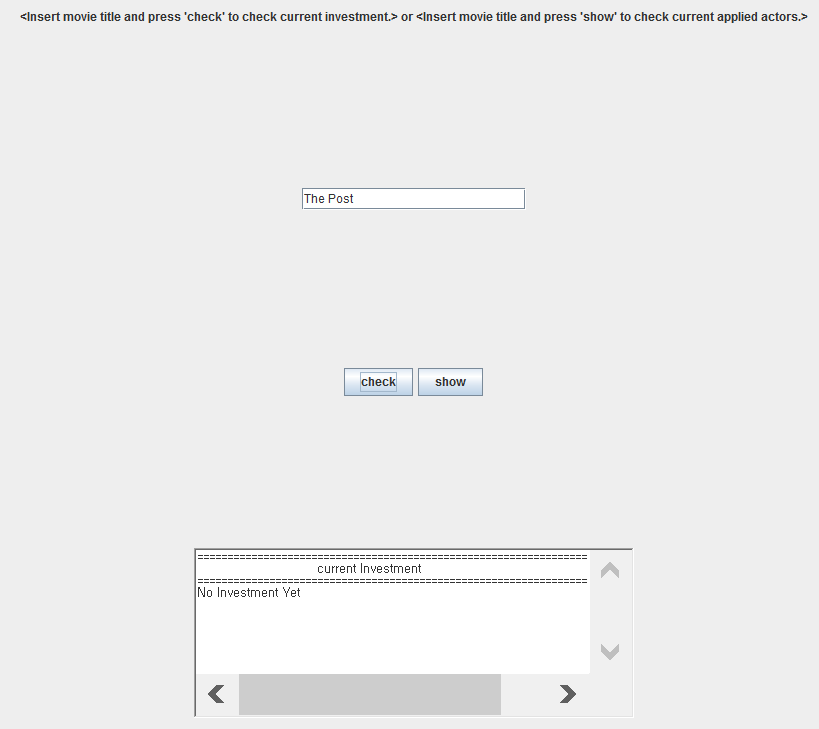




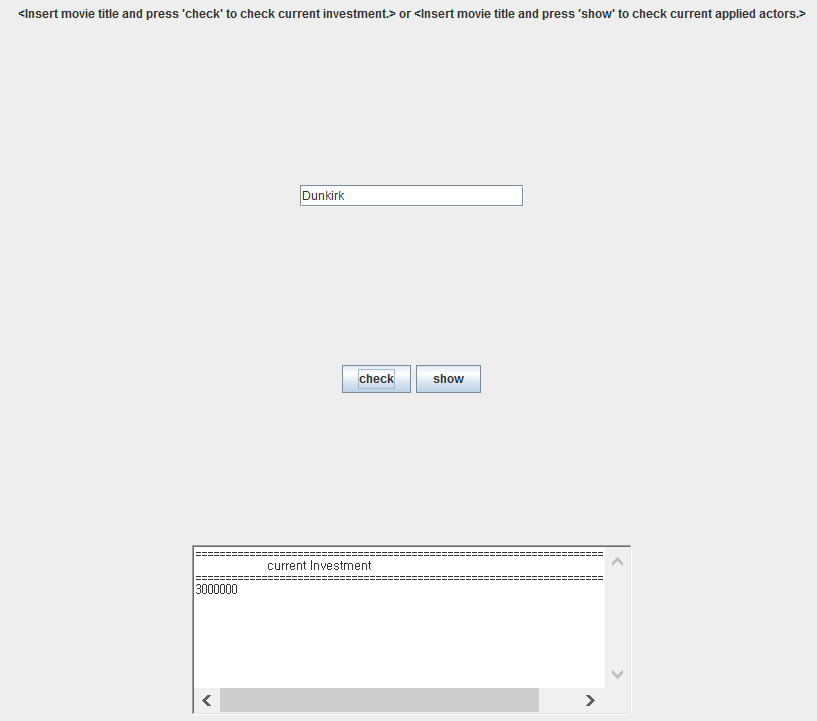
1. 1)영화 제목을 입력하고 check 버튼을 누르면 현재 투자금 현황을 볼 수 있다.

2)영화 제목을 입력하고 show 버튼을 누르면 해당 영화 오디션에 지원한 배우들 정보를 볼 수 있다.

1)

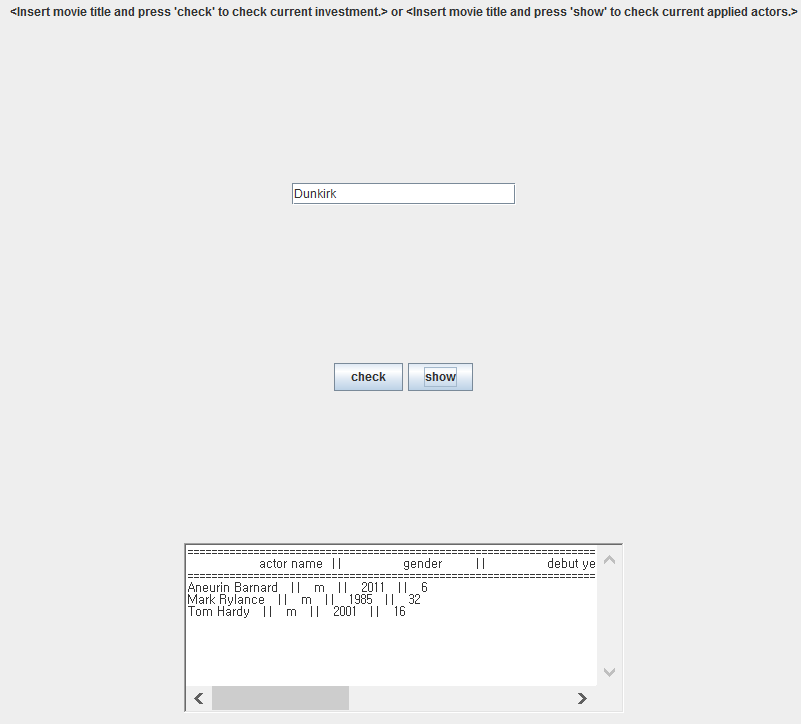


(current investment가 없는 경우)

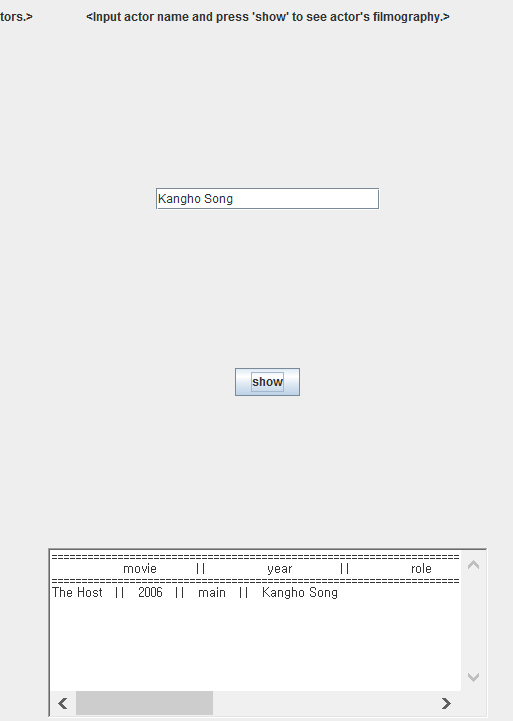


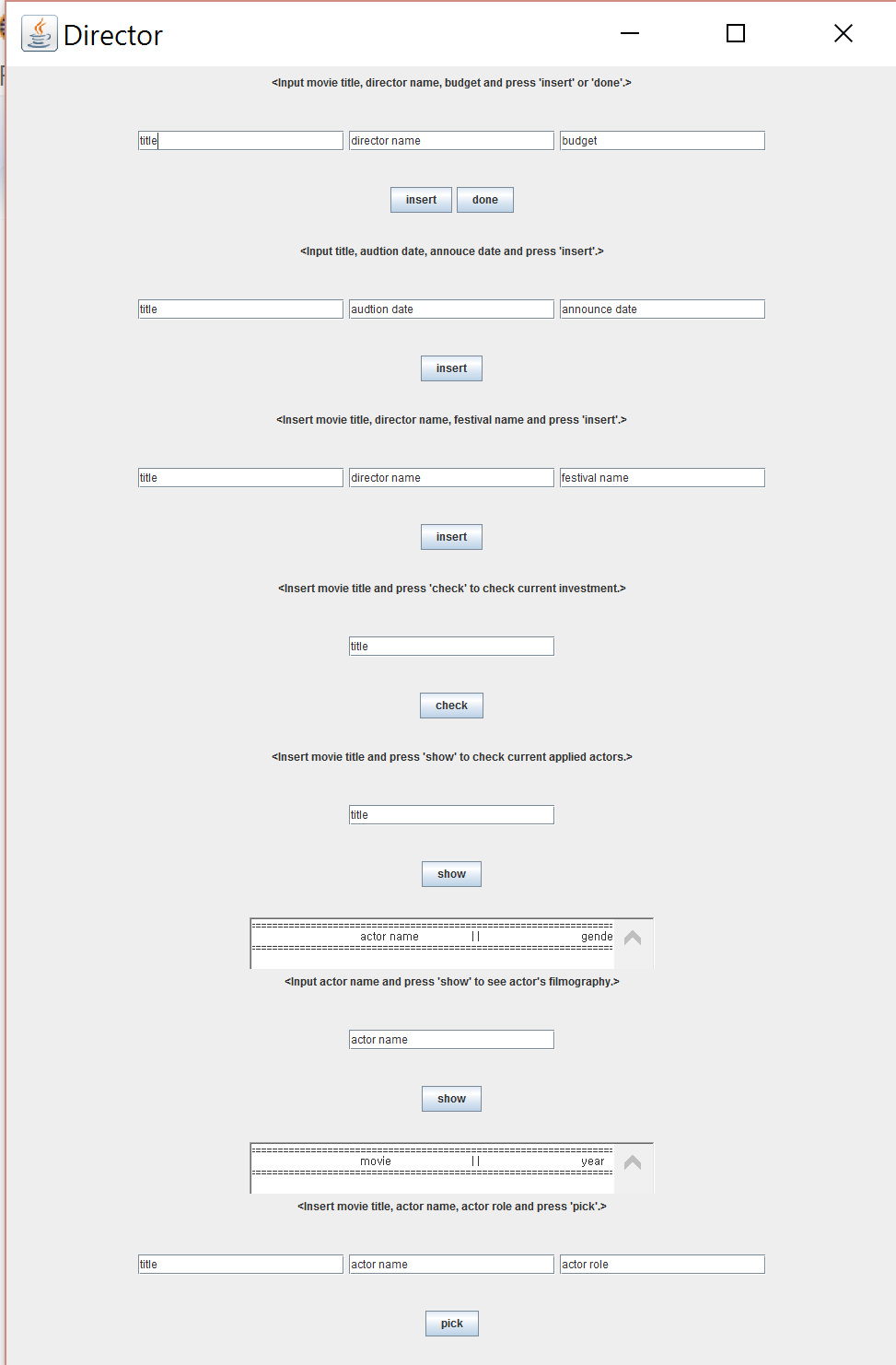
(current investment가 있는 경우)

2)



1. 배우 이름을 입력하고 show 버튼을 누르면 해당 배우의 필모그래피가 확인 가능하다.





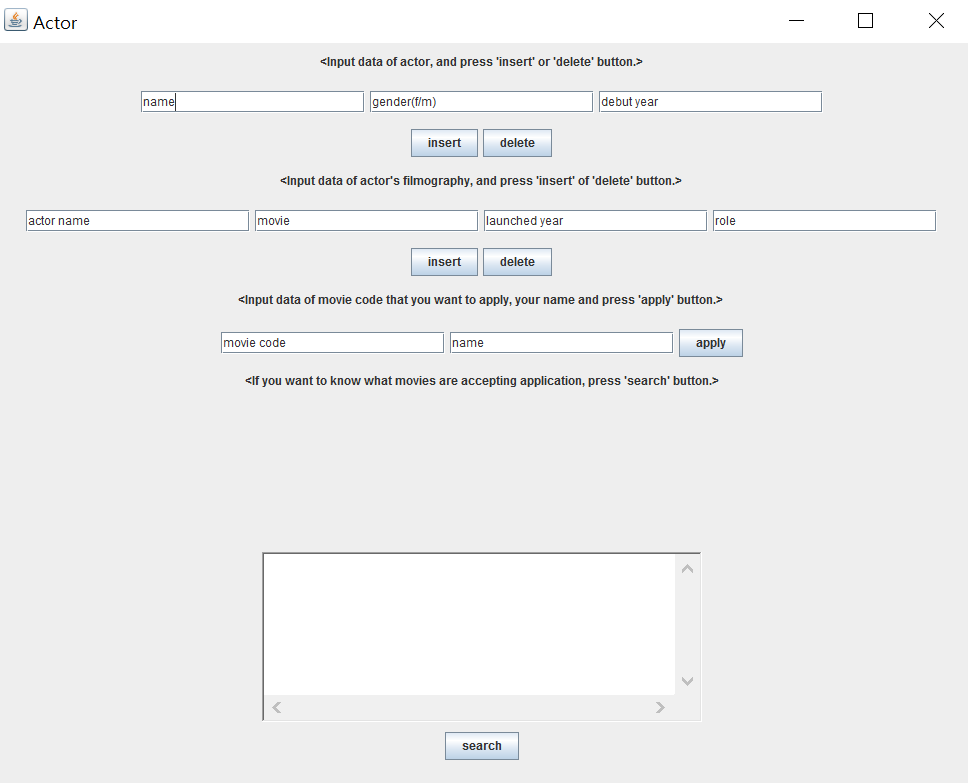
1. 영화 제목, 감독 이름, 예산을 작성하고 insert 버튼을 누르면 새로운 영화가 등록된다.

같은 정보를 작성하고 done 버튼을 누르면 오디션이 완료됐음이 들어간다.

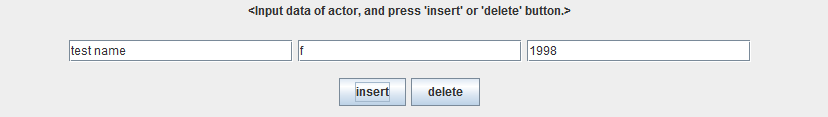
1. 영화 제목, 오디션 날짜, 발표 날짜를 입력하고 insert 버튼을 누르면 오디션이 등록된다.
2. 감독이 자신의 수상작 제목, 감독 이름, 영화제 제목을 입력하고 insert 버튼을 누르면 자신의 수상 정보가 등록된다.
3. 영화 제목을 입력하고 check 버튼을 누르면 현재 투자금 현황을 볼 수 있다.
4. 영화 제목을 입력하고 show 버튼을 누르면 해당 영화 오디션에 지원한 배우들 정보를 볼 수 있다.
5. 배우 이름을 입력하고 show 버튼을 누르면 해당 배우의 필모그래피가 확인 가능하다.
6. 영화 제목, 배우 이름, 배우에게 지정할 역할을 입력하고 pick 버튼을 누르면 배우를 얻게 된다.

(2)Actor 버튼 클릭 시

//처음화면

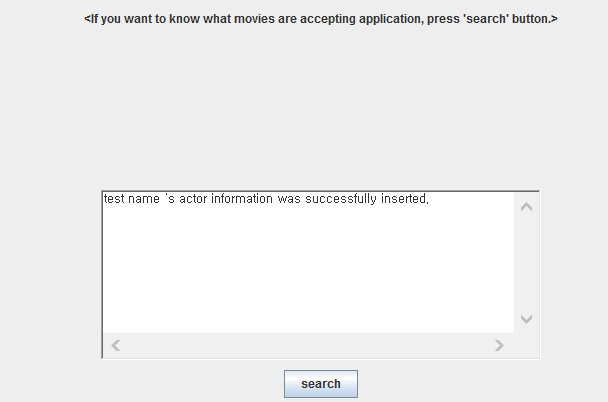


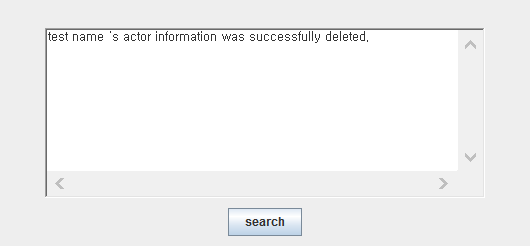
(a)배우 정보 insert/delete



첫 번째 항목에 텍스트필드의 디폴트 값에 맞춰 배우의 정보를 입력하고 insert 버튼을 누르면 배우의 정보가 insert 된다. 또 delete 버튼을 누르면 배우의 정보가 db에서 삭제된다. 배우 정보 삭제는 이름만 입력해도 가능하다.

// Insert/delete에 성공하면 아래의 textarea에 안내문이 뜬다.



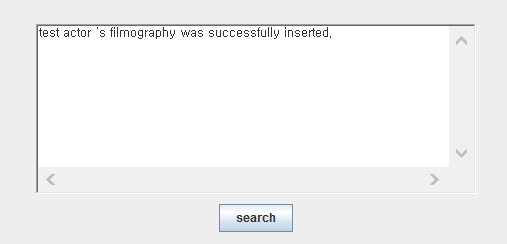


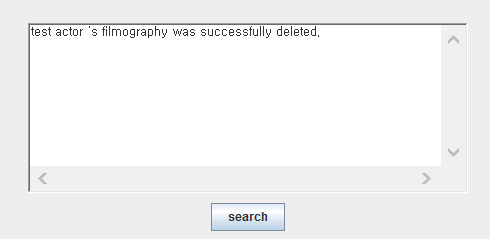
(b)배우 필모그래피 insert/delete



두 번째 항목에 입력하고 싶은 배우의 경력 작품의 정보를 디폴트값에 따라 입력하고, insert/delete 버튼을 누르면 된다. 작업이 성공하면 안내문이 뜬다.

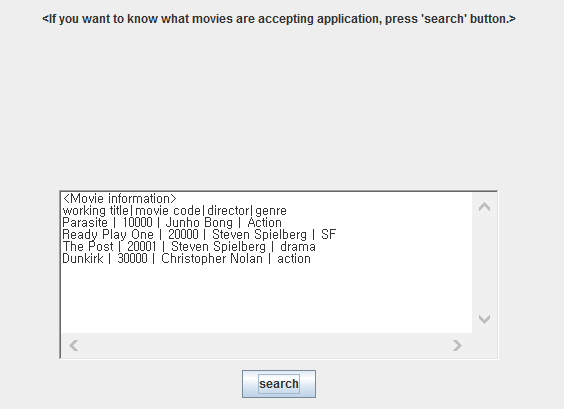
//안내문 화면



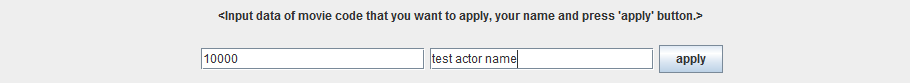


(c)배우가 제작 예정인 영화에 지원

//영화 정보가 출력된 화면

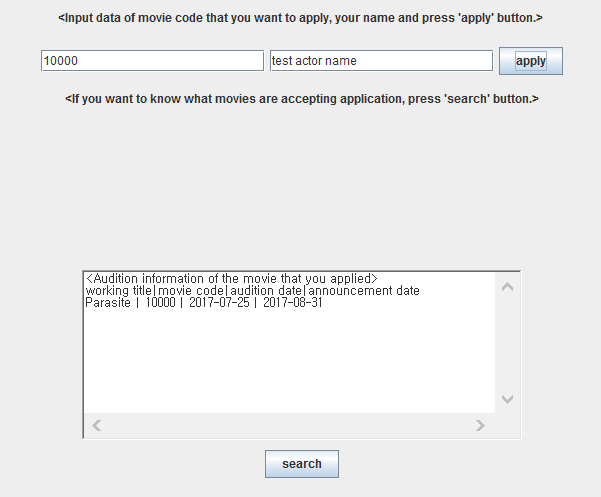


영화에 지원하려면 일단 마지막 항목의 search 버튼을 눌러 제작 예정인 영화를 출력해 주면 지원하고자 하는 영화의 코드를 보고 세 번째 항목에 영화의 코드와 배우의 이름을 입력해 주어야 한다.



입력을 하고 apply 버튼을 누르면 맨 마지막의 textarea에서 지원한 영화의 오디션 날짜와 결과 발표 날짜를 출력해 준다.

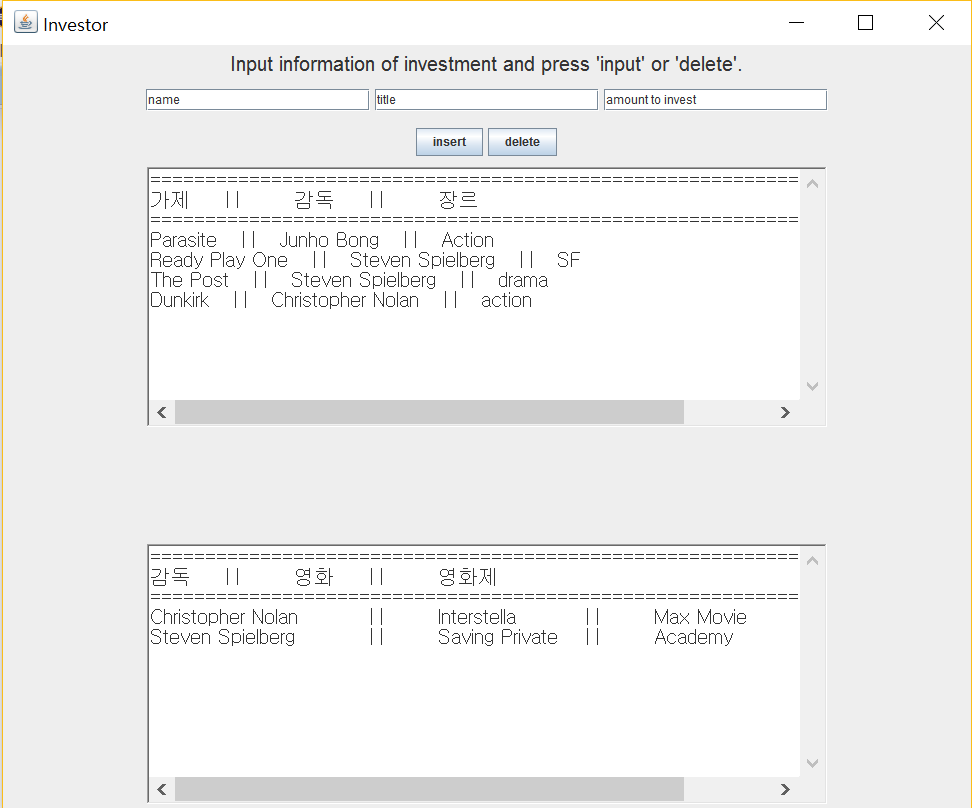
//오디션 정보가 출력된 화면



(3)Investor 버튼 클릭 시

(a)처음 화면

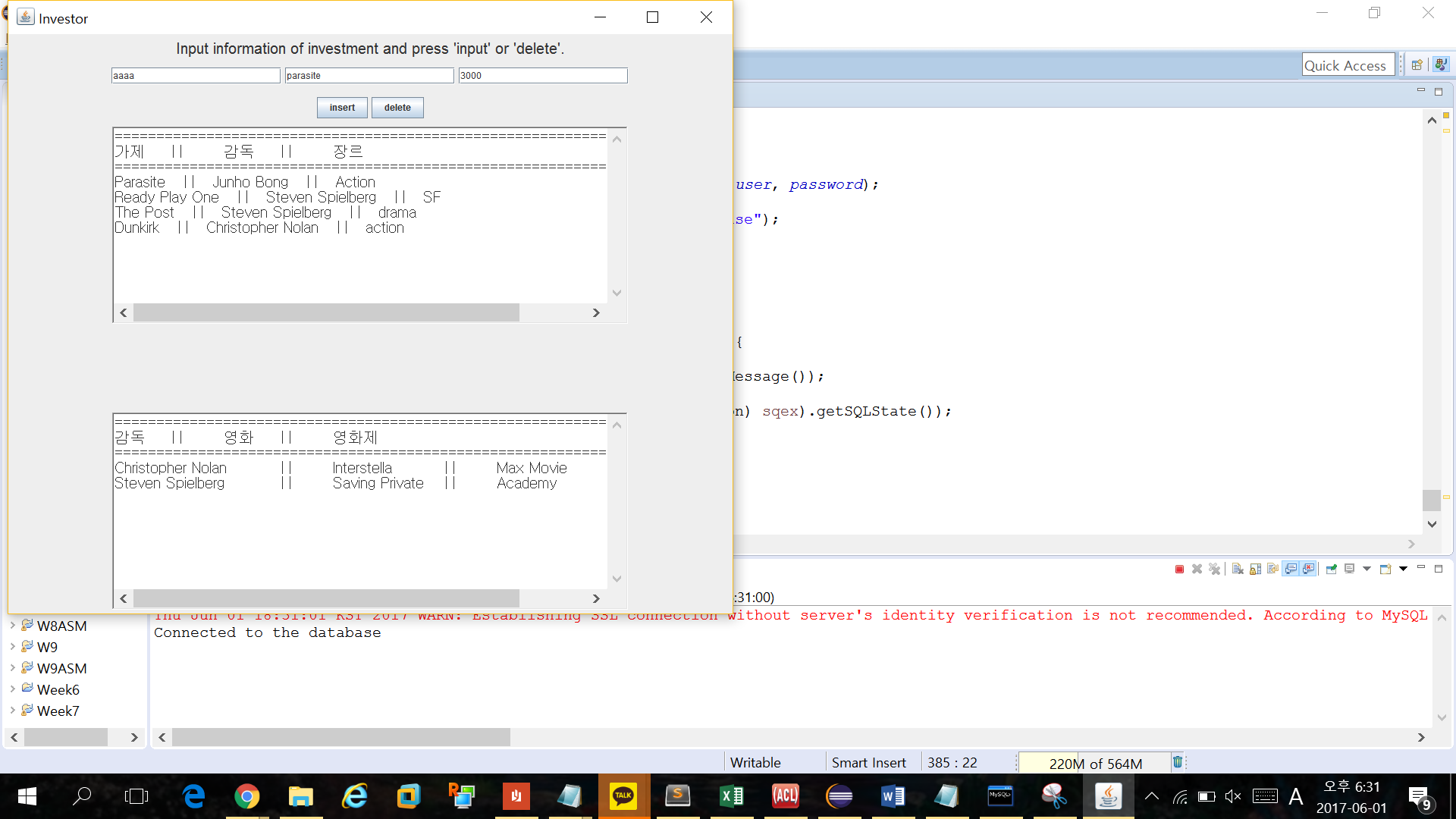
//처음 화면



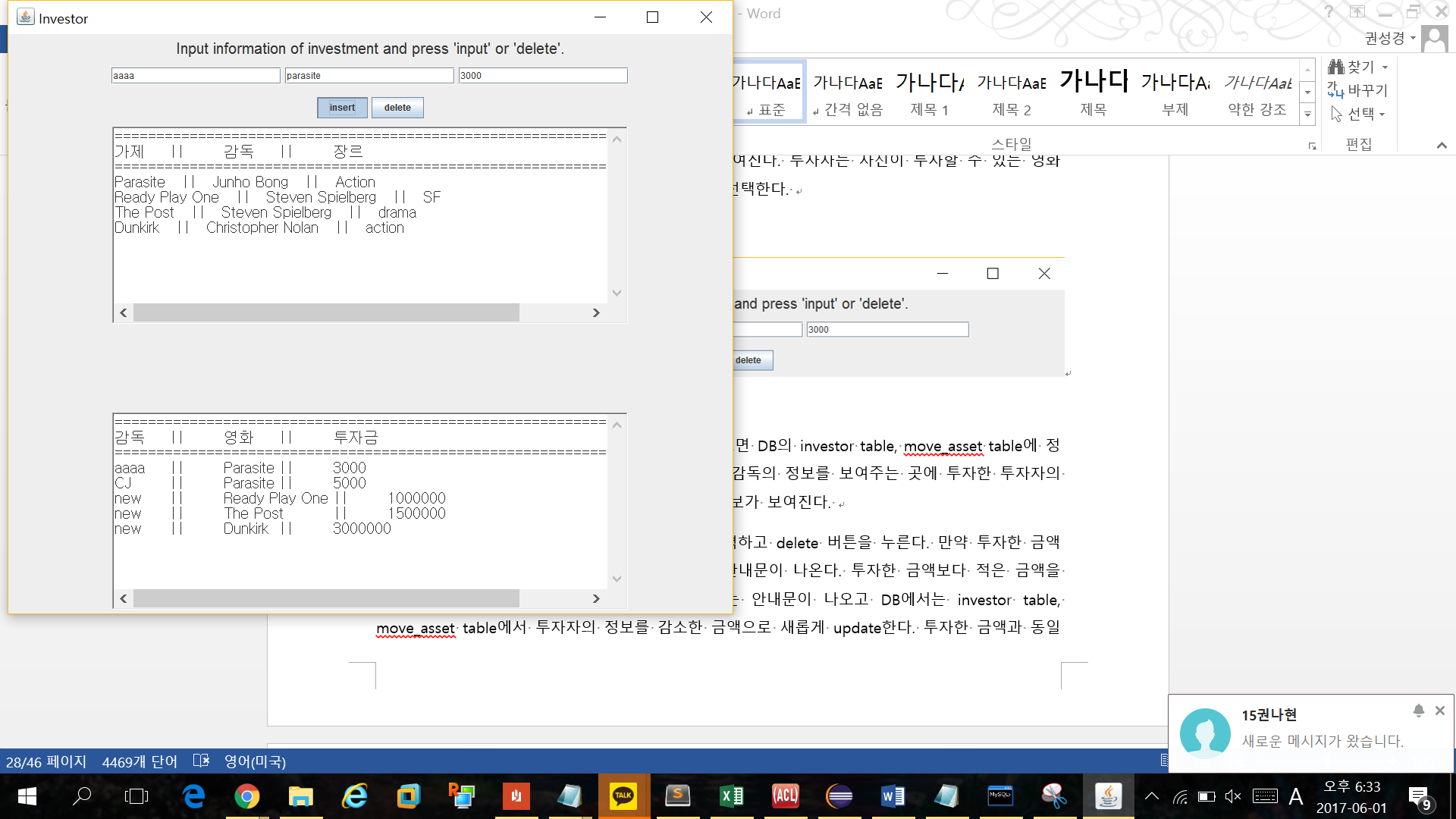
투자자 자신의 이름과 투자하고 싶은 영화 제목, 투자할 돈을 적는 칸이 있고 영화정보, 감독이 어떤 영화제에서 어떠한 영화로 상을 받았는지 보여진다. 투자자는 자신이 투자할 수 있는 영화와 그 영화의 감독의 경력을 보고 투자할 영화를 선택한다.

(b)insert할 때

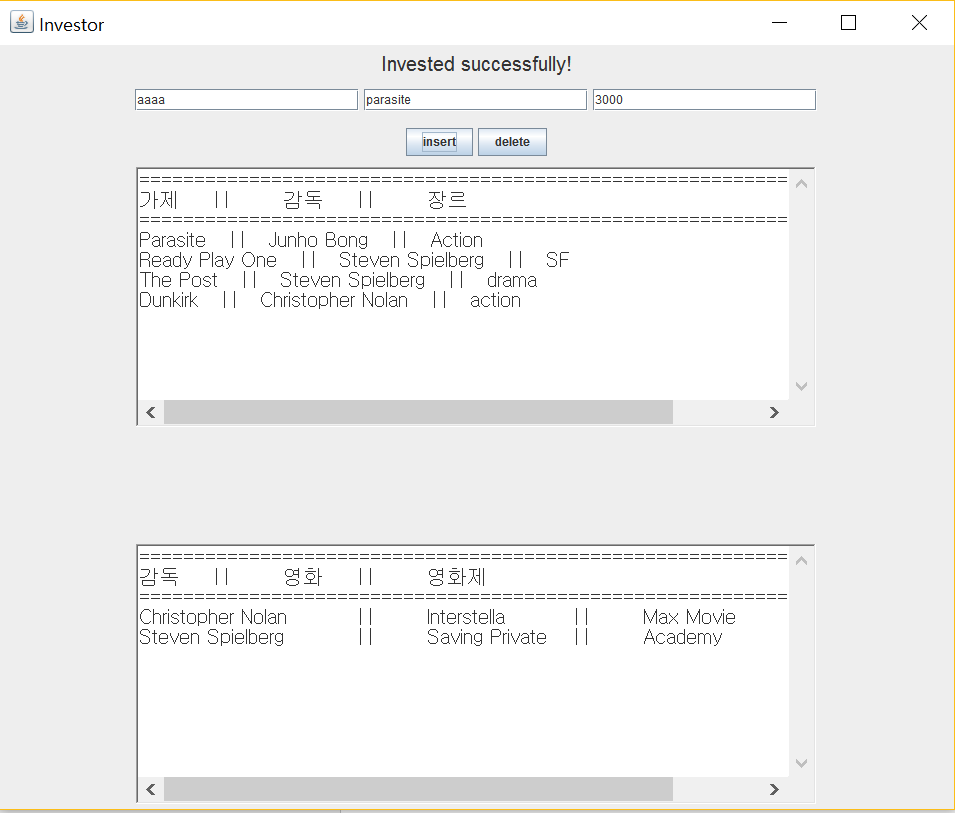
//정보를 입력한 화면



//insert를 누르고 두 번째 text are에 4초 가량 투자자의 정보를 보여주는 화면



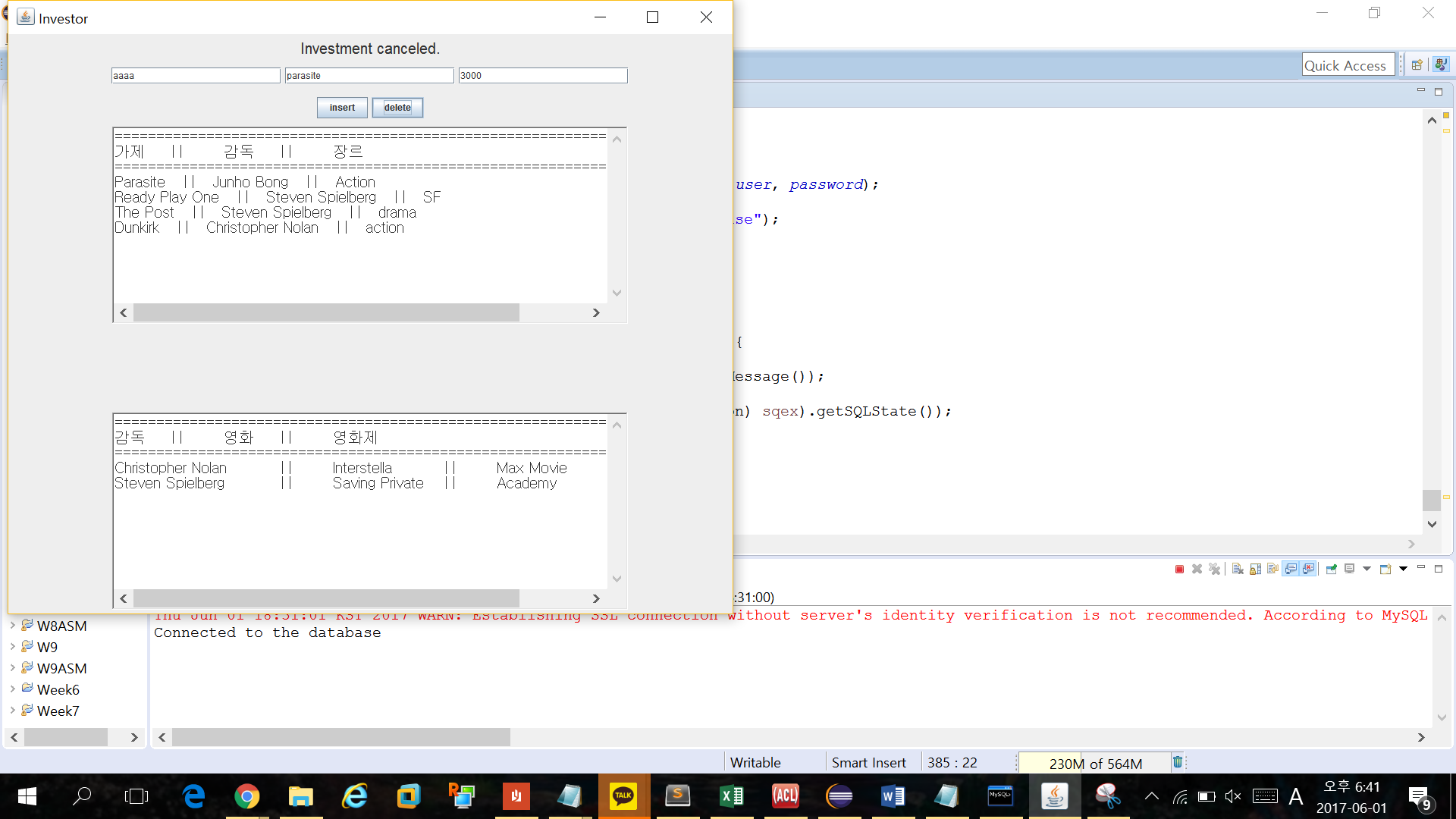
//4초 뒤 다시 아래 화면과 같이 돌아온다.



이름을, 영화, 금액을 textfield에 넣고 insert를 누르면 DB의 investor table, move\_asset table에 정보가 추가된다. 사용자의 interface에서는 상 받은 감독의 정보를 보여주는 곳에 투자한 투자자의 정보가 4초 가량 보여진다. 후에는 다시 감독의 정보가 보여진다.

(c)delete할 때

//투자자 정보를 입력하고 delete를 누른 화면



투자를 취소하고 싶을 땐 자신의 투자 정보를 입력하고 delete 버튼을 누른다. 만약 투자한 금액보다 많은 금액을 입력하면 제거 할 수 없다는 안내문이 나온다. 투자한 금액보다 적은 금액을 입력하면 interface에는 투자 금액이 감소하였다는 안내문이 나오고 DB에서는 investor table, move\_asset table에서 투자자의 정보를 감소한 금액으로 새롭게 update한다. 투자한 금액과 동일한 금액을 입력하면 interface에 투자가 취소되었다는 안내가 나오고, DB에서는 investor table, move\_asset table에서 투자자의 정보를 삭제한다.

5. DETAIL EXPLANATION OF 17 REQUIREMENTS

(1) Should have at least **10 tables** with total of at least **30 columns**(column is total for all tables)

*st.executeUpdate("CREATE TABLE DBCOURSE\_DIRECTOR("*

*+ "NAME VARCHAR(30) PRIMARY KEY,"*

*+ "MOVIE VARCHAR(30));");*

*st.executeUpdate("CREATE TABLE DBCOURSE\_TITLE\_ID("*

*+ "WORKING\_TITLE\_ID INT PRIMARY KEY,"*

*+ "WORKING\_TITLE VARCHAR(30) NOT NULL,"*

*+ "DONE VARCHAR(30));");*

*st.executeUpdate("create table DBCOURSE\_MOVIE("*

*+ "working\_title\_id int not null,"*

*+ "director varchar(30) not null,"*

*+ "genre varchar(30),"*

*+ "primary key (working\_title\_id),"*

*+ "foreign key (working\_title\_id) references dbcourse\_title\_id(working\_title\_id));");*

*st.executeUpdate("CREATE TABLE DBCOURSE\_FILM\_FESTIVAL("*

*+ "MOVIE VARCHAR(30) NOT NULL,"*

*+ "DIRECTOR VARCHAR(30) NOT NULL,"*

*+ "FESTIV\_NAME VARCHAR(30) PRIMARY KEY,"*

*+ "FOREIGN KEY(DIRECTOR) REFERENCES dbcourse\_DIRECTOR(NAME));");*

*st.executeUpdate("create table DBCOURSE\_INVESTOR("*

*+ "inv\_name varchar(30) not null,"*

*+ "working\_title\_id int not null,"*

*+ "investment int,"*

*+ "primary key (inv\_name, working\_title\_id));");*

*st.executeUpdate("create table DBCOURSE\_movie\_asset("*

*+ "working\_title\_id int not null,"*

*+ "current\_investment int,"*

*+ "budget int,"*

*+ "primary key(working\_title\_id),"*

*+ "foreign key(working\_title\_id) references dbcourse\_movie(working\_title\_id));");*

*st.executeUpdate("create table DBCOURSE\_ACTOR("*

*+ "name varchar(30) not null,"*

*+ "gender char(1),"*

*+ "debut\_year varchar(30),"*

*+ "career int,"*

*+ "primary key(name));");*

*st.executeUpdate("create table DBCOURSE\_FILMOGRAPHY("*

*+ "actor\_name varchar(30) not null,"*

*+ "movie varchar(30) not null,"*

*+ "year int,"*

*+ "role varchar(30),"*

*+ "primary key(actor\_name),"*

*+ "foreign key(actor\_name) references DBCOURSE\_ACTOR(name));");*

*st.executeUpdate("create table DBCOURSE\_RECRUIT("*

*+ "working\_title\_id int(5) not null,"*

*+ "applied\_actor varchar(30) not null,"*

*+ "casting\_role varchar(30),"*

*+ "primary key(working\_title\_id,applied\_actor),"*

*+ "foreign key(applied\_actor) references DBCOURSE\_ACTOR(name));");*

*st.executeUpdate("create table DBCOURSE\_AUDITION("*

*+ "working\_title\_id int(5) not null,"*

*+ "aud\_date date,"*

*+ "announce\_date date,"*

*+ "primary key(working\_title\_id));");*

(2)Should have at least **50 records** inserted for initialization (total records for all tables)

st.executeUpdate("insert into dbcourse\_director "

+ "values(\"Junho Bong\",\"The Host\");");

st.executeUpdate("insert into dbcourse\_director "

+"values(\"Steven Spielberg\",\"Saving Private\");");

st.executeUpdate("insert into dbcourse\_title\_id "

+ "values(10000,\"Parasite\",null);");

st.executeUpdate("insert into dbcourse\_title\_id "

+"values(20000,\"Ready Play One\",null);");

st.executeUpdate("insert into dbcourse\_movie "

+"values(10000,\"Junho Bong\",\"Action\");");

st.executeUpdate("insert into dbcourse\_movie "

+"values(20000,\"Steven Spielberg\",\"SF\");");

st.executeUpdate("insert into DBCOURSE\_DIRECTOR "

+ "values(\"Christopher Nolan\",\"Interstella\");");

st.executeUpdate("insert into DBCOURSE\_TITLE\_ID "

+ "values(20001,\"The Post\",NULL);");

st.executeUpdate("insert into DBCOURSE\_TITLE\_ID "

+ "values(30000,\"Dunkirk\",NULL);");

st.executeUpdate("insert into DBCOURSE\_MOVIE "

+ "values(20001,\"Steven Spielberg\",\"drama\");");

st.executeUpdate("insert into DBCOURSE\_MOVIE "

+ "values(30000,\"Christopher Nolan\",\"action\");");

st.executeUpdate("insert into dbcourse\_investor "

+"values(\"CJ\",10000,10000);");

st.executeUpdate("insert into dbcourse\_investor "

+"values(\"new\",20000,1000000);");

st.executeUpdate("insert into dbcourse\_investor "

+"values(\"new\",20001,1500000);");

st.executeUpdate("insert into dbcourse\_investor "

+"values(\"new\",30000,3000000);");

st.executeUpdate("insert into dbcourse\_movie\_asset "

+"values(10000,10000,1000000);");

st.executeUpdate("insert into dbcourse\_movie\_asset "

+"values(20000,1000000,3000000);");

st.executeUpdate("insert into dbcourse\_movie\_asset "

+"values(20001,1500000,2000000);");

st.executeUpdate("insert into dbcourse\_movie\_asset "

+"values(30000,3000000,3000000);");

st.executeUpdate("insert into dbcourse\_actor "

+ "values('Kangho Song','m',1996,21),"

+ "('Simon Pegg','m',1999,18),"

+ "('Olivia Cooke','f',2012,5),"

+ "('Meryl Streep','f',1977,40),"

+ "('Mark Rylance','m',1985,32),"

+ "('Tom Hardy','m',2001,16),"

+ "('Tom Hanks','m',1980,37),"

+ "('T.J.Miller','m',2007,10),"

+ "('Julie Nickson','f',1980,37),"

+ "('Aneurin Barnard','m',2011,6);");

st.executeUpdate("insert into dbcourse\_recruit

"+"values(10000,\"Kangho Song\",\"Main\"),"

+ "(20000,\"Simon Pegg\",\"Sub\"),"

+ "(20000,\"Olivia Cooke\",null),"

+ "(20000,\"Meryl Streep\", \"Main\"),"

+ "(20000,\"Mark Rylance\",null),"

+ "(30000,\"Mark Rylance\",null),"

+ "(30000,\"Tom Hardy\",null),"

+ "(20001,\"Tom Hanks\",null),"

+ "(20000,\"T.J. Miller\",null),"

+ "(20001,\"Julie Nickson\",null),"

+ "(30000,\"Aneurin Barnard\",null);");

st.executeUpdate("insert into DBCOURSE\_FILM\_FESTIVAL "

+ "values(\"Saving Private\",\"Steven Spielberg\",\"Academy\");");

st.executeUpdate("insert into DBCOURSE\_FILM\_FESTIVAL "

+ "values(\"Interstella\",\"Christopher Nolan\",\"Max Movie\");");

st.executeUpdate("insert into dbcourse\_filmography "

+ "values('Kangho Song','The Host',2006,'main'),"

+ "('Simon Pegg','Star Wars',2015,'main'),"

+ "('Olivia Cooke','The Signal',2014,'main'),"

+ "('Meryl Streep','The Devil wears Prada',2006,'main'),"

+ "('Mark Rylance','Bridge of Spies',2015,'main'),"

+ "('Tom Hardy','Mad Max',2015,'main'),"

+ "('Tom Hanks','Bridge of Spies',2015,'main'),"

+ "('T.J.Miller','Deadpool',2016,'sub'),"

+ "('Julie Nickson','Rambo2',1985,'sub'),"

+ "('Aneurin Barnard','Legend',2015,'sub');");

st.executeUpdate("insert into dbcourse\_audition "

+ "values(10000,'2017-07-25','2017-08-31'),"

+ "(20000,'2017-06-21','2017-07-01'),"

+ "(20001,'2017-05-27','2017-06-03'),"

+ "(30000,'2017-05-27','2017-06-05');");

(4) Should have at least **2views** defined

(5) Should have at least **5 indexes** defined

st.executeUpdate("alter table DBCOURSE\_recruit add index(applied\_actor(3))");

st.executeUpdate("alter table DBCOURSE\_investor add index(inv\_name(2))");

st.executeUpdate("alter table DBCOURSE\_actor add index(name(5))");

st.executeUpdate("alter table DBCOURSE\_title\_id add index(working\_title(5))");

st.executeUpdate("alter table DBCOURSE\_filmography add index(actor\_name(3))");

(7) Should include use of **transactions**

**-(a)insert하는 경우의 transaction – investor table에 투자자 정보를 추가하는 것과 movie\_asset table에 영화 투자금 정보를 추가하는 것이 둘 다 실행되거나, 둘 다 실행되지 않는다.**

//transation을 위한 boolean값

**boolean** insertsuccess=**false**;

**try**{

//transation

*conn*.setAutoCommit(**false**);

PreparedStatement pStmt = *conn*.prepareStatement("insert into dbcourse\_investor "+ "values (?,?,?)");

pStmt.setString(1,invst\_name.getText() );

pStmt.setInt(2,id);

pStmt.setInt(3,(Integer.*parseInt*(investment.getText())) );

pStmt.executeUpdate();

*st*.executeUpdate("update dbcourse\_movie\_asset "

+ "set current\_investment = current\_investment+"

+investment.getText()+" where working\_title\_id = "+id+";");

//성공하면 true

insertsuccess=**true**;

}

**catch**(SQLException sqle){

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

}

**finally**{

**if**(insertsuccess){

*conn*.commit();

expl.setText("Invested successfully!");

direc.setText("====================================================================="+"\n"

+"감독"+"\t"+"||"+"\t"+"영화"+"\t"+"||"+"\t"+"투자금"+"\t"+"\n"

+"====================================================================="+"\n");

**try**{

//투자 정보를 받아서 보여줌

*st*= *conn*.createStatement();

*st*.executeQuery("select inv\_name, working\_title, investment "

+ "from dbcourse\_investor, dbcourse\_title\_id "

+ "where dbcourse\_investor.working\_title\_id = dbcourse\_title\_id.working\_title\_id; ");

rs=*st*.getResultSet();

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

String nm = rs.getString("inv\_name");

String mv = rs.getString("working\_title");

String in = rs.getString("investment");

direc.append(nm+"\t||\t"+mv+"\t||\t"+in+"\n");

}

//4초 간 보여줌

**try** {

Thread.*sleep*(4 \* 1000);

} **catch** (InterruptedException ee) { }

//4초 후 다시 감독 정보 보여줌

direc.setText("");

direc.setText("====================================================================="+"\n"

+"감독 "+"\t"+"||"+"\t"+"영화"+"\t"+"||"+"\t"+"영화제"+"\t"+"\n"

+"====================================================================="+"\n");

*st*.executeQuery("select name, movie, festiv\_name "

+ "from (select name, dbcourse\_director.movie, festiv\_name "

+ "from dbcourse\_director, dbcourse\_film\_festival "

+ "where dbcourse\_director.name = dbcourse\_film\_festival.director) as T;");

rs=*st*.getResultSet();

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

String nm = rs.getString("name");

String mv = rs.getString("movie");

String fn = rs.getString("festiv\_name");

direc.append(nm+"\t||\t"+mv+"\t||\t"+fn+"\n");

}

}

**catch** (SQLException sqex) {

System.***out***.println("SQLException: " + sqex.getMessage());

System.***out***.println("SQLState: " + sqex.getSQLState());

}

}

**else**{

//transaction 실패한 경우

*conn*.rollback();

}

}

-(b)투자금을 일부 취소할 경우 truncation- investor테이블에서 투자자의 투자금을 감소시키는 것과, movie\_asset table에서 투자금 정보를 감소시키는 것이 둘 다 성공적으로 실행되거나, 하나가 실패 할 경우 둘 다 일어나지 않는다.

**try**{

//transaction

*conn*.setAutoCommit(**false**);

//movie\_asset테이블에서 삭제

*st*.executeUpdate("update dbcourse\_movie\_asset "

+ "set current\_investment = current\_investment-"

+investment.getText()+" where working\_title\_id = "+id+";");

//investor테이블에서 삭제

*st*.executeUpdate("delete from dbcourse\_investor "

+ "where working\_title\_id = "+id+" and inv\_name = \""+invst\_name.getText()+"\";");

deletesuccess=**true**;

}

**catch**(SQLException sqle){

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

}

**finally**{

**if**(deletesuccess){

*conn*.commit();

expl.setText("Investment canceled.");

}

**else**{

*conn*.rollback();

}

}

-(c)투자금을 전부 취소할 경우 truncation- investor테이블에서 투자자의 정보를 삭제하는 것과, movie\_asset table에서 투자금 정보를 감소시키는 것이 둘 다 성공적으로 실행되거나, 하나가 실패 할 경우 둘 다 일어나지 않는다.

**try**{

//transaction

*conn*.setAutoCommit(**false**);

//movie\_asset테이블에서 투자금 감소시킴

*st*.executeUpdate("update dbcourse\_movie\_asset "

+ "set current\_investment = current\_investment-"

+investment.getText()+" where working\_title\_id = "+id+";");

//investor테이블에서 투자금 감소시킴

*st*.executeUpdate("update dbcourse\_investor "

+ "set investment = investment-"

+investment.getText()+" where working\_title\_id = "+id+";");

deletesuccess=**true**;

}

**catch**(SQLException sqle){

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

}

**finally**{

**if**(deletesuccess){

*conn*.commit();

expl.setText("Investment has decreased.");

}

**else**{

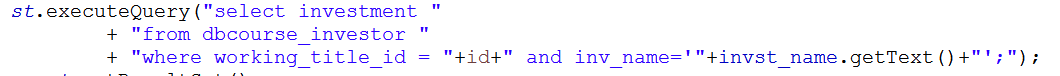
*conn*.rollback();

}

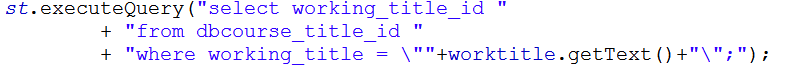
}

(8) Should include queries that use **index (query containing indexed column)**

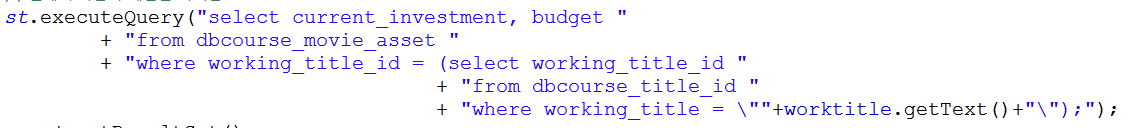
**//inv\_name**



//working+title

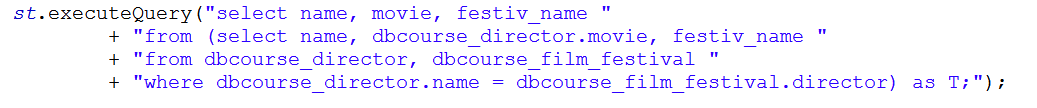


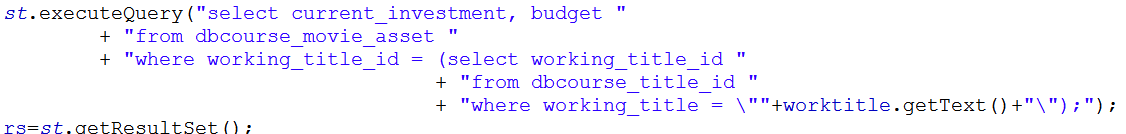
//working\_title



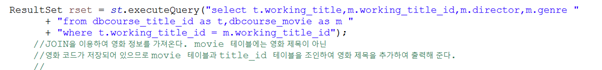
(9) Should include queries that use **view**

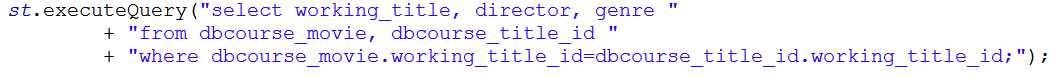
(10) Should include queries with **nested queries.**

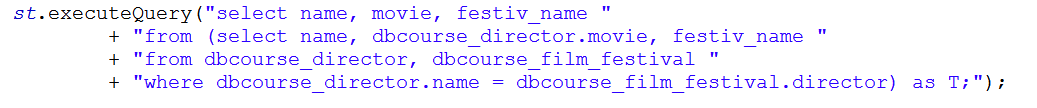


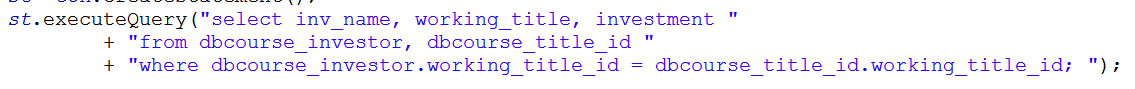
 (11) Should include queries with **join queries.**



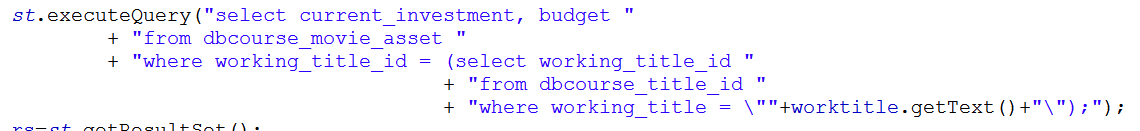


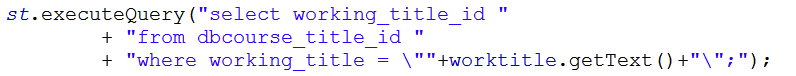


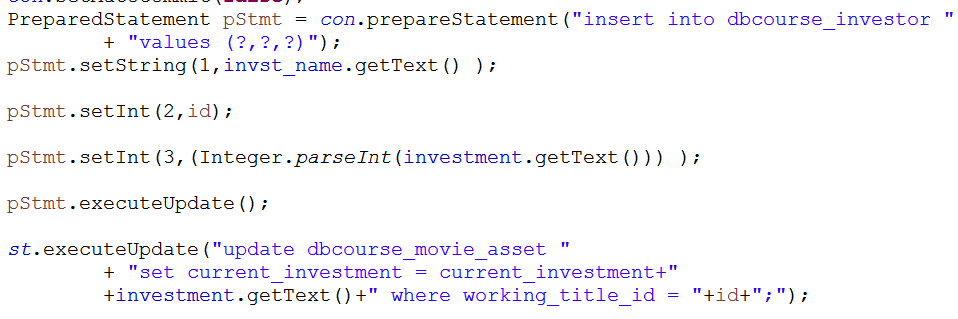


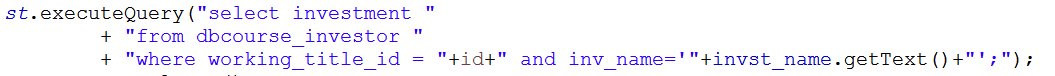


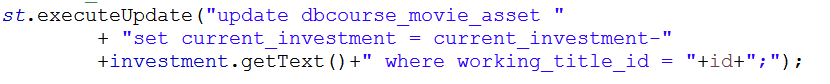
(12) Should include **queries that are parameterized and dynamically created** (in other words, takes input value from user and creates query using the user given value)

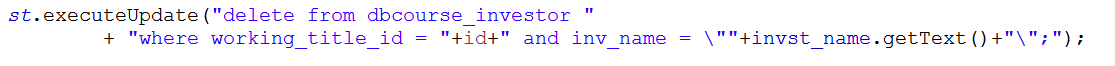


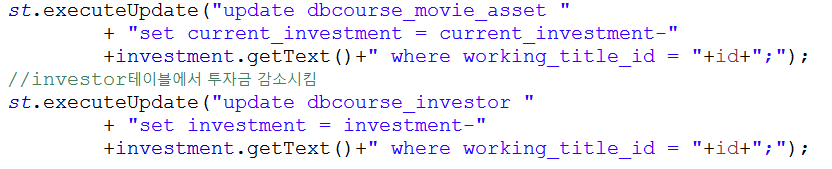












6. TEAM RESPONSIBILITY

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| MEMBER  NAME | SQL | JAVA CODE | REPORT | PRESEN  TATION | DEMO  VIDEO | Etc |
| 권나현 | -ER Diagram  -Schema  Diagram  -배우 메뉴 관련한 SQL쿼리 전체 | -class ActorFrame  -class  ActActionListener | ACTOR  메뉴  관련한  부분 | PPT 양식 제작, 1번, 3번 항목 제작 | actor | INDEX 정리  Table actor, filmography, recruit, audition  Create insert |
| 권성경 | -투자자 관련 sql 쿼리전체  -movie,  investor,  movie\_asset  테이블 의 create 및 insert | -class InvestorFrame  -class  IActionListener | Investor  관련  내용들과 create, insert,index, transaction | Investor 관련  항목  제작 | investor | -전체 table의create, insert 문 순서 및 foreign key 최종 정리 |
| 김다영 | -디렉터 관련 SQL전체  -film festival, director, title\_id 테이블 생성 | -class jdbc exam  (3개의 gui 하나로 통합함)  -class DirectorFrame  -class DActionListener | -director 관련 내용들과 title\_id의 인덱스  -jdbc exam 관련 내용들 | -director관련  피피티 제작 | director | -Normalization을 위한 데이터베이스 수정  -전체 records 조사 및 정리 |

7. SQL SCRIPTS

(1) Table setting query

//1515009 김다영

CREATE TABLE DBCOURSE\_DIRECTOR( NAME VARCHAR(30) PRIMARY KEY,

MOVIE VARCHAR(30));

CREATE TABLE DBCOURSE\_TITLE\_ID(

WORKING\_TITLE\_ID INT PRIMARY KEY,

WORKING\_TITLE VARCHAR(30) NOT NULL,

DONE VARCHAR(30));

//1515005 권성경

create table DBCOURSE\_MOVIE(working\_title\_id int not null,

director varchar(30) not null,

genre varchar(30),

primary key (working\_title\_id),

foreign key (working\_title\_id) references dbcourse\_title\_id(working\_title\_id));

//1515005 권성경

CREATE TABLE DBCOURSE\_FILM\_FESTIVAL(MOVIE VARCHAR(30) NOT NULL,

DIRECTOR VARCHAR(30) NOT NULL,

FESTIV\_NAME VARCHAR(30) PRIMARY KEY,

FOREIGN KEY(DIRECTOR) REFERENCES dbcourse\_DIRECTOR(NAME));

//1515004 권나현

create table DBCOURSE\_INVESTOR( inv\_name varchar(30) not null,

working\_title\_id int not null,

investment int,

primary key (inv\_name, working\_title\_id));

create table DBCOURSE\_movie\_asset( working\_title\_id int not null,

current\_investment int,

budget int,

primary key(working\_title\_id),

foreign key(working\_title\_id) references dbcourse\_movie(working\_title\_id));

create table DBCOURSE\_ACTOR( name varchar(30) not null,

gender char(1),

debut\_year varchar(30),

career int,

primary key(name));

create table DBCOURSE\_FILMOGRAPHY(

actor\_name varchar(30) not null,

movie varchar(30) not null,

year int,

role varchar(30),

primary key(actor\_name),

foreign key(actor\_name) references DBCOURSE\_ACTOR(name));

create table DBCOURSE\_RECRUIT(working\_title\_id int(5) not null,

applied\_actor varchar(30) not null,

casting\_role varchar(30),

primary key(working\_title\_id,applied\_actor),

foreign key(applied\_actor) references DBCOURSE\_ACTOR(name));

create table DBCOURSE\_AUDITION( working\_title\_id int(5) not null,

aud\_date date,

announce\_date date,

primary key(working\_title\_id));

//1515005 권성경

create view invest as

select \*

from DBCOURSE\_MOVIE, DBCOURSE\_INVESTOR

where dbcourse\_movie.working\_title\_id = dbcourse\_investor.working\_title\_id;

create view asset as

select \*

from DBCOURSE\_MOVIE, DBCOURSE\_movie\_asset

where dbcourse\_movie.working\_title\_id = dbcourse\_investor.working\_title\_id;

insert into dbcourse\_director

values('Junho Bong','The Host');

insert into dbcourse\_director

values('Steven Spielberg','Saving Private');

insert into dbcourse\_title\_id

values(10000,'Parasite',null);

insert into dbcourse\_title\_id

values(20000,'Ready Play One',null);

insert into dbcourse\_movie

values(10000,'Junho Bong','Action');

insert into dbcourse\_movie

values(20000,'Steven Spielberg','SF');

//1515009 김다영

insert into DBCOURSE\_DIRECTOR

values('Christopher Nolan','Interstella');

//1515005 권성경

insert into DBCOURSE\_TITLE\_ID

values(20001,'The Post',NULL);

//1515005 김다영

insert into DBCOURSE\_TITLE\_ID

values(30000,'Dunkirk',NULL);

insert into DBCOURSE\_MOVIE

values(20001,'Steven Spielberg','drama');

insert into DBCOURSE\_MOVIE

values(30000,'Christopher Nolan','action');

//1515005 권성경ㅇ

insert into dbcourse\_investor

values('CJ',10000,10000);

insert into dbcourse\_investor

values('new',20000,1000000);

insert into dbcourse\_investor

values('new',20001,1500000);

insert into dbcourse\_investor

values('new',30000,3000000);

insert into dbcourse\_movie\_asset

values(10000,10000,1000000);

insert into dbcourse\_movie\_asset

values(20000,1000000,3000000);

insert into dbcourse\_movie\_asset

values(20001,1500000,2000000);

insert into dbcourse\_movie\_asset

values(30000,3000000,3000000);

//1515004 권나현

insert into dbcourse\_actor

values('Kangho Song','m',1996,21),

('Simon Pegg','m',1999,18),

('Olivia Cooke','f',2012,5),

('Meryl Streep','f',1977,40),

('Mark Rylance','m',1985,32),

('Tom Hardy','m',2001,16),

('Tom Hanks','m',1980,37),

('T.J.Miller','m',2007,10),

('Julie Nickson','f',1980,37),

('Aneurin Barnard','m',2011,6);

insert into dbcourse\_recruit

values(10000,'Kangho Song','Main'),

(20000,'Simon Pegg','Sub'),

(20000,'Olivia Cooke',null),

(20000,'Meryl Streep', 'Main'),

(20000,'Mark Rylance',null),

(30000,'Mark Rylance',null),

(30000,'Tom Hardy',null),

(20001,'Tom Hanks',null),

(20000,'T.J. Miller',null),

(20001,'Julie Nickson',null),

(30000,'Aneurin Barnard',null);

insert into DBCOURSE\_FILM\_FESTIVAL

values('Saving Private','Steven Spielberg','Academy');

insert into DBCOURSE\_FILM\_FESTIVAL

values('Interstella','Christopher Nolan','Max Movie');

insert into dbcourse\_filmography

values('Kangho Song','The Host',2006,'main'),

('Simon Pegg','Star Wars',2015,'main'),

('Olivia Cooke','The Signal',2014,'main'),

('Meryl Streep','The Devil wears Prada',2006,'main'),

('Mark Rylance','Bridge of Spies',2015,'main'),

('Tom Hardy','Mad Max',2015,'main'),

('Tom Hanks','Bridge of Spies',2015,'main'),

('T.J.Miller','Deadpool',2016,'sub'),

('Julie Nickson','Rambo2',1985,'sub'),

('Aneurin Barnard','Legend',2015,'sub');

insert into dbcourse\_audition

values(10000,'2017-07-25','2017-08-31'),

(20000,'2017-06-21','2017-07-01'),

(20001,'2017-05-27','2017-06-03'),

(30000,'2017-05-27','2017-06-05');

alter table DBCOURSE\_movie add index(working\_title\_id(20));

alter table DBCOURSE\_investor add index(inv\_name(30));

(2) Director

select working\_title\_id from DBCOURSE\_TITLE\_ID;

select name

from DBCOURSE\_DIRECTOR

where name='사용자입력';

insert into DBCOURSE\_DIRECTOR values('사용자입력','사용자입력');

INSERT INTO DBCOURSE\_TITLE\_ID VALUES('사용자입력','사용자입력','사용자입력');

INSERT INTO DBCOURSE\_MOVIE VALUES('사용자입력','사용자입력','사용자입력');

INSERT INTO DBCOURSE\_MOVIE\_ASSET VALUES('사용자입력','사용자입력','사용자입력');

UPDATE DBCOURSE\_TITLE\_ID SET DONE='done';

INSERT INTO DBCOURSE\_AUDITION VALUES(?,?,?);

SELECT WORKING\_TITLE\_ID

FROM DBCOURSE\_TITLE\_ID

WHERE WORKING\_TITLE ='사용자입력결과 아이디';

INSERT INTO DBCOURSE\_FILM\_FESTIVAL VALUES('사용자입력','사용자입력','사용자입력');

SELECT CURRENT\_INVESTMENT

FROM DBCOURSE\_MOVIE\_ASSET natural join DBCOURSE\_TITLE\_ID

WHERE WORKING\_TITLE = '사용자입력';

SELECT WORKING\_TITLE\_ID

FROM DBCOURSE\_TITLE\_ID

WHERE WORKING\_TITLE ='사용자입력';

SELECT a.name,a.gender,a.debut\_year,a.career

FROM DBCOURSE\_ACTOR as a,DBCOURSE\_RECRUIT as r

where a.name = r.applied\_actor and r.working\_title\_id = '사용자입력결과 아이디';

SELECT movie, year,role,actor\_name

FROM DBCOURSE\_FILMOGRAPHY

where actor\_name ='사용자입력';

SELECT WORKING\_TITLE\_ID

FROM DBCOURSE\_TITLE\_ID

WHERE WORKING\_TITLE ='사용자입력';

UPDATE DBCOURSE\_RECRUIT

SET CASTING\_ROLE='사용자입력'

WHERE WORKING\_TITLE\_ID = "+ tempId+" and applied\_actor = '사용자입력';

(3) Actor

insert into dbcourse\_actor values ('사용자 입력','사용자 입력','사용자 입력','사용자 입력');

delete from dbcourse\_actor where name = '사용자 입력';

insert into dbcourse\_filmography values ('사용자 입력','사용자 입력','사용자 입력','사용자 입력');

delete from dbcourse\_filmography where actor\_name = '사용자 입력' and movie ='사용자 입력' and year = '사용자 입력' and role = '사용자 입력';

insert into dbcourse\_recruit values ('사용자 입력','사용자 입력',null)");

select distinct t.working\_title,t.working\_title\_id,a.aud\_date,a.announce\_date from dbcourse\_recruit as r, dbcourse\_audition as a, dbcourse\_title\_id as t where r.working\_title\_id = a.working\_title\_id and a.working\_title\_id = t.working\_title\_id and t.working\_title\_id = '사용자입력';

select t.working\_title,m.working\_title\_id,m.director,m.genre from dbcourse\_title\_id as t,dbcourse\_movie as m where t.working\_title\_id = m.working\_title\_id;

(4) Investor

select working\_title, director, genre

from dbcourse\_movie, dbcourse\_title\_id

where dbcourse\_movie.working\_title\_id=dbcourse\_title\_id.working\_title\_id;

select name, movie, festiv\_name

from (select name, dbcourse\_director.movie, festiv\_name

from dbcourse\_director, dbcourse\_film\_festival

where dbcourse\_director.name = dbcourse\_film\_festival.director) as T;

select current\_investment, budget

from dbcourse\_movie\_asset

where working\_title\_id = (select working\_title\_id

from dbcourse\_title\_id

where working\_title = '사용자입력');

select working\_title\_id

from dbcourse\_title\_id

where working\_title = '사용자입력';

update dbcourse\_movie\_asset

set current\_investment = current\_investment-사용자입력숫자

where working\_title\_id = '사용자 입력에 따른 영화 아이디';

select inv\_name, working\_title, investment

from dbcourse\_investor, dbcourse\_title\_id

where dbcourse\_investor.working\_title\_id = dbcourse\_title\_id.working\_title\_id;

select investment

from dbcourse\_investor

where working\_title\_id = 사용자입력에따른숫자 and inv\_name='사용자입력';

delete from dbcourse\_investor

where working\_title\_id = 사용자입력에따른아이디 and inv\_name = '사용자입력';

update dbcourse\_investor

set investment = investment-사용자입력 숫자 where working\_title\_id = 사용자 입력에 따른 영화 아이디;

8. JAVA CODES

**package** dbprojectfinal;

**import** java.awt.BorderLayout;

**import** java.awt.Dimension;

**import** java.awt.FlowLayout;

**import** java.awt.Font;

**import** java.awt.GridLayout;

**import** java.awt.TextArea;

**import** java.awt.Toolkit;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** javax.swing.JButton;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JPanel;

**import** javax.swing.JTextField;

**public** **class** jdbcexam{

**public** **static** Connection *con*;

**public** **static** ResultSet *rs*;

**public** **static** java.sql.Statement *st*;

//1515009 김다영

**class** MenuFrame **extends** JFrame **implements** ActionListener{

JButton i,d,a;

MenuFrame(){

setTitle("team15\_title\_GUI");

setSize(500,300);

setDefaultCloseOperation(***EXIT\_ON\_CLOSE***);

setLayout(**new** BorderLayout());

JPanel p2 = **new** JPanel(**new** FlowLayout());

JLabel l = **new** JLabel("Who are You?");

p2.add(l);

add(p2,"North");

JPanel p = **new** JPanel(**new** FlowLayout());

d = **new** JButton("Director");

d.addActionListener(**this**);

p.add(d);

a = **new** JButton("Actor");

a.addActionListener(**this**);

p.add(a);

i = **new** JButton("Investor");

i.addActionListener(**this**);

p.add(i);

add(p,"Center");

setVisible(**true**);

}

**public** **void** actionPerformed(ActionEvent e){

**if**(e.getSource()==i){

setVisible(**false**);

**new** InvestorFrame();

}

**else** **if**(e.getSource()==d){

setVisible(**false**);

**new** DirectorFrame();

}

**else** **if**(e.getSource()==a)

{

setVisible(**false**);

**new** ActorFrame();

}

}

}

//1515009 김다영

**class** DirectorFrame **extends** JFrame{//director 전용 frame

**public** **int** title\_id;

JButton insertMov;

JButton delete;

JButton insertAud;

JButton insertFes;

JButton checkInv;

JButton pickAct;

JButton showAct;

JButton showAFilm;

JTextField directorName1,directorName2;

JTextField Title1,Title2,Title3,Title4,Title6;

JTextField genre;

JTextField festivName;

JTextField budget;

JTextField audDate;

JTextField annDate;

JTextField actorRole;

JTextField actorName,actorName2;

TextArea result = **new** TextArea();

TextArea result2 = **new** TextArea();

//frame 에 적용될 것들

**public** DirectorFrame(){//디렉터프레임 생성자

Toolkit tk = getToolkit();

Dimension d = tk.getScreenSize();

setTitle("Director");

setSize(d.width,d.height);

setDefaultCloseOperation(***EXIT\_ON\_CLOSE***);

JPanel p = **new** JPanel();

JPanel iml = **new** JPanel();

JPanel imt = **new** JPanel(**new** FlowLayout());

JPanel imb = **new** JPanel(**new** FlowLayout());

JPanel p1 = **new** JPanel(**new** GridLayout(3,1));

JLabel inml= **new** JLabel("<Input movie title, director name, budget, genre and press 'insert' or 'done'.>");

iml.add(inml);

p1.add(iml);

//영화를 등록하기 위해 감독 유저가 해야할 것들 설명

Title1= **new** JTextField(20);

Title1.setText("title");

imt.add(Title1);

directorName1 = **new** JTextField(20);

directorName1.setText("director name");

imt.add(directorName1);

budget = **new** JTextField(20);

budget.setText("budget");

imt.add(budget);

genre = **new** JTextField("genre",20);

imt.add(genre);

p1.add(imt);

//가제, 감독 이름, 예산을 입력받는다.

insertMov = **new** JButton("insert");

insertMov.addActionListener(**new** DActionListener());

imb.add(insertMov);

delete = **new** JButton("done");

delete.addActionListener(**new** DActionListener());

imb.add(delete);

p1.add(imb);

p.add(p1);

//insert 와 done 버튼제공

JPanel ina = **new** JPanel();

JPanel inat = **new** JPanel(**new** FlowLayout());

JPanel inab = **new** JPanel(**new** FlowLayout());

JLabel audl = **new** JLabel();

JPanel p2 = **new** JPanel(**new** GridLayout(3,1));

audl.setText("<Input title, audtion date, annouce date and press 'insert'.>");

ina.add(audl);

p2.add(ina);

//오디션을 등록하기 위해 감독 유저가 해야할 것을 알려준다.

Title2 = **new** JTextField(20);

Title2.setText("title");

inat.add(Title2);

audDate = **new** JTextField("audtion date",20);

inat.add(audDate);

annDate = **new** JTextField("announce date",20);

inat.add(annDate);

p2.add(inat);

//가제, 오디션 날짜, 발표 날짜를 입력받는다.

insertAud = **new** JButton("insert");

insertAud.addActionListener(**new** DActionListener());

inab.add(insertAud);

p2.add(inab);

p.add(p2);

//오디션을 등록할 수 있는 버튼 제공

JPanel p3 = **new** JPanel(**new** GridLayout(3,1));

JPanel inf = **new** JPanel();

JPanel inft = **new** JPanel(**new** FlowLayout());

JPanel infb = **new** JPanel(**new** FlowLayout());

JLabel f = **new** JLabel("<Insert movie title, director name, festival name and press 'insert'.>");

//감독이 자신의 수상이력을 등록할 수 있도록 한다.

inf.add(f);

p3.add(inf);

Title3 = **new** JTextField(20);

Title3.setText("title");

inft.add(Title3);

directorName2 = **new** JTextField(20);

directorName2.setText("director name");

inft.add(directorName2);

festivName = **new** JTextField(20);

festivName.setText("festival name");

inft.add(festivName);

p3.add(inft);

//영화 제목, 감독 이름, 영화제 이름을 입력받는다.

insertFes = **new** JButton("insert");

insertFes.addActionListener(**new** DActionListener());

infb.add(insertFes);

p3.add(infb);

p.add(p3);

//insert 버튼 제공

JPanel p7 = **new** JPanel(**new** GridLayout(3,1));

JPanel pac = **new** JPanel();

JPanel pact = **new** JPanel(**new** FlowLayout());

JPanel pacb = **new** JPanel(**new** FlowLayout());

JLabel pic = **new** JLabel("<Insert movie title, actor name, actor role and press 'pick'.>");

//배우를 뽑을 수 있다.

pac.add(pic);

p7.add(pac);

Title6 = **new** JTextField("title",20);

pact.add(Title6);

actorName = **new** JTextField("actor name", 20);

pact.add(actorName);

actorRole = **new** JTextField("actor role",20);

pact.add(actorRole);

p7.add(pact);

//가제, 배우 이름, 지정할 역할을 입력받는다.

pickAct = **new** JButton("pick");

pickAct.addActionListener(**new** DActionListener());

pacb.add(pickAct);

p7.add(pacb);

//pick 버튼 제공

p.add(p7);

add(p);

JPanel p4 = **new** JPanel(**new** GridLayout(4,1));

JPanel cl = **new** JPanel();

JPanel cf = **new** JPanel(**new** FlowLayout());

JPanel cb = **new** JPanel(**new** FlowLayout());

JLabel cla = **new** JLabel("<Insert movie title and press 'check' to check current investment.>"

+" or <Insert movie title and press 'show' to check current applied actors.>");

cl.add(cla);

p4.add(cl);

//현재 투자상황을 알 수 있도록 해준다.

Title4 = **new** JTextField("title",20);

cf.add(Title4);

p4.add(cf);

//가제를 입력받는다.

checkInv = **new** JButton("check");

checkInv.addActionListener(**new** DActionListener());

cb.add(checkInv);

showAct = **new** JButton("show");

showAct.addActionListener(**new** DActionListener());

cb.add(showAct);

p4.add(cb);

JPanel abc = **new** JPanel();

abc.add(result);

p4.add(abc);

p.add(p4);

//check버튼 제공

JPanel p6 = **new** JPanel(**new** GridLayout(4,1));

JPanel cal = **new** JPanel();

JLabel l5 = **new** JLabel("<Input actor name and press 'show' to see actor's filmography.>");

cal.add(l5);

p6.add(cal);

//감독은 배우의 필모그래피 확인이 가능하다.

JPanel cat = **new** JPanel(**new** FlowLayout());

JPanel cab = **new** JPanel(**new** FlowLayout());

JPanel car = **new** JPanel();

result2.setText("=============================================================================================="

+ "==================================================================================="+"\n"

+"\tmovie "+"\t||\t"+"year"+"\t||\t"+"role"+"\t||\t"+"actor\_name"+"\n"

+"================================================================================================="

+ "================================================================================"+"\n");

actorName2 = **new** JTextField("actor name",20);

cat.add(actorName2);

p6.add(cat);

//배우 이름을 입력받는다.

showAFilm = **new** JButton("show");

showAFilm.addActionListener(**new** DActionListener());;

cab.add(showAFilm);

p6.add(cab);

car.add(result2);

p6.add(car);

p.add(p6);

//버튼 제공

setVisible(**true**);

}

**class** DActionListener **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent e){

**try**{

*con* = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/team15", "team15", "team15");

*st* = *con*.createStatement();

PreparedStatement ps;

**if**(e.getSource()==insertMov){//영화 삽입

**try**{

**int** [] ti = **new** **int**[100];//아이디 배열

*rs* = *st*.executeQuery("select working\_title\_id from DBCOURSE\_TITLE\_ID;");

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

**for**(**int** i=0;i<ti.length;i++)

ti[i]=*rs*.getInt(1);

}//아이디를 다받아서 배열에 저장

title\_id = (**int**)(Math.*random*()\*10000+1);//랜덤 아이디

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

**int** j;

**for**(j=0;j<ti.length;j++){

**if**(ti[j]==title\_id){

title\_id = (**int**)(Math.*random*()\*10000+1);

j=-1;

**break**;

}

}//duplicate 확인

**if**(j!=-1)

**break**;//unique한 아이디이면 ok

}

*rs* = *st*.executeQuery("select name from DBCOURSE\_DIRECTOR where name=\""+directorName1.getText()+"\";");

//감독 이름이 이미 존재하는지 확인

**if**(*rs*.wasNull()){

ps = *con*.prepareStatement("insert into DBCOURSE\_DIRECTOR values(?,?)");

ps.setString(1, directorName1.getText());

ps.setString(2,"No Movie");

ps.executeUpdate();

}//감독 체크하고 없으면 감독테이블에 삽입

ps = *con*.prepareStatement("INSERT INTO DBCOURSE\_TITLE\_ID "

+ "VALUES(?,?,?)");

ps.setInt(1, title\_id);

ps.setString(2, Title1.getText());

ps.setString(3, **null**);

ps.executeUpdate();//새로운 영화에 타이틀 아이디 주기

ps = *con*.prepareStatement("INSERT INTO DBCOURSE\_MOVIE VALUES(?,?,?)");

ps.setInt(1, title\_id);

ps.setString(2, directorName1.getText());

ps.setString(3, genre.getText());

ps.executeUpdate();//무비 테이블에 영화 삽입

ps = *con*.prepareStatement("INSERT INTO DBCOURSE\_MOVIE\_ASSET VALUES(?,?,?)");

ps.setInt(1, title\_id);

ps.setString(2,**null**);

ps.setString(3, budget.getText());

ps.executeUpdate();//무비 에셋 테이블에 추가

result.setText("Movie Inserted");

}

**catch**(SQLException e1){

System.***out***.println("insert movie error");

e1.printStackTrace();

}

}

**else** **if**(e.getSource() == delete){//영화 종료

**try**{

*st*.executeUpdate("UPDATE DBCOURSE\_TITLE\_ID SET DONE=\"done\";");//완료함을 저장

result.setText("Audition Finished");

}

**catch**(SQLException e1){

System.***out***.println("done movie error");

e1.printStackTrace();

}

}

**else** **if**(e.getSource()==insertAud){//오디션 삽입

**try**{

**int** tempId=-1;

ps = *con*.prepareStatement("INSERT INTO DBCOURSE\_AUDITION VALUES(?,?,?)");

*rs* = *st*.executeQuery("SELECT WORKING\_TITLE\_ID FROM DBCOURSE\_TITLE\_ID WHERE WORKING\_TITLE =\""+Title2.getText()+"\";");

//영화 아이디 가져오기

**if**(*rs*.next()){

tempId = *rs*.getInt(1);

ps.setInt(1,tempId);

String d1 = audDate.getText();

String d2 = annDate.getText();

ps.setString(2,d1);

ps.setString(3, d2);

ps.executeUpdate();

}//오디션 삽입

**else**{

result.setText("check your input");

}

result.setText("Audition Inserted");

}

**catch**(SQLException e1){

System.***out***.println("insert audition error");

e1.printStackTrace();

}

}

**else** **if**(e.getSource()==insertFes){//수상작

**try**{

ps= *con*.prepareStatement("INSERT INTO DBCOURSE\_FILM\_FESTIVAL VALUES(?,?,?)");

ps.setString(1, Title3.getText());

ps.setString(2, directorName2.getText());

ps.setString(3, festivName.getText());

ps.executeUpdate();

result.setText("Prize Inserted");

//수상 이력 삽입

}

**catch**(SQLException e1){

System.***out***.println("insert festival error");

e1.printStackTrace();

}

}

**else** **if**(e.getSource()== checkInv){//투자금 확인

**try**{

result.setText("========================================================================"+"\n"

+"\tcurrent Investment\t\n"

+"============================================================================"+"\n");

*rs* = *st*.executeQuery("SELECT CURRENT\_INVESTMENT FROM DBCOURSE\_MOVIE\_ASSET natural join DBCOURSE\_TITLE\_ID WHERE WORKING\_TITLE = \""+Title4.getText()+"\";");

**if**(*rs*.next())

result.append(*rs*.getString(1));//디렉터 유저에게 투자금 알려줌

**else**

result.append("No Investment Yet");

}

**catch**(SQLException e1){

System.***out***.println("check investiment error");

e1.printStackTrace();

}

}

**else** **if**(e.getSource()==showAct){//배우 확인

**try**{

**int** tempId=0;

result.setText("=============================================================================================="

+ "==================================================================================="+"\n"

+"\tactor name "+"\t||\t"+"gender"+"\t||\t"+"debut year"+"\t||\t"+"career"+"\n"

+"================================================================================================="

+ "================================================================================"+"\n");

*rs* = *st*.executeQuery("SELECT WORKING\_TITLE\_ID FROM DBCOURSE\_TITLE\_ID WHERE WORKING\_TITLE =\""+Title4.getText()+"\";");

//영화 아이디 가져오기

**if**(*rs*.next()) {

tempId = *rs*.getInt(1);

*rs* = *st*.executeQuery("SELECT a.name,a.gender,a.debut\_year,a.career FROM DBCOURSE\_ACTOR as a,DBCOURSE\_RECRUIT as r where a.name = r.applied\_actor and r.working\_title\_id = "+tempId+";");

//관련 배우 테이블 불러오기

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

String an = *rs*.getString(1);

String ag = *rs*.getString(2);

String dy= *rs*.getString(3);

**int** cy = *rs*.getInt(4);

result.append(an+" || "+ag+" || "+dy+" || "+cy+"\n");

}

}

}

**catch**(SQLException e1){

System.***out***.println("show actor error");

e1.printStackTrace();

}

}

**else** **if**(e.getSource()==showAFilm){//필모그래피

**try**{

*rs* = *st*.executeQuery("SELECT movie, year,role,actor\_name FROM DBCOURSE\_FILMOGRAPHY where actor\_name =\""+actorName2.getText()+"\";");

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

String mn = *rs*.getString(1);

String y = *rs*.getString(2);

String r= *rs*.getString(3);

String an = *rs*.getString(4);

result2.append(mn+" || "+y+" || "+r+" || "+an+"\n");

}

}

**catch**(SQLException e1){

System.***out***.println("show actor filmography error");

e1.printStackTrace();

}

}

**else** **if**(e.getSource()==pickAct){//배우 뽑기

**try**{

**int** tempId = 0;

*rs* = *st*.executeQuery("SELECT WORKING\_TITLE\_ID FROM DBCOURSE\_TITLE\_ID WHERE WORKING\_TITLE =\""+Title6.getText()+"\";");

//영화 아이디 가져오기

**if**(*rs*.next()){

tempId = *rs*.getInt(1);}

*st*.executeUpdate("UPDATE DBCOURSE\_RECRUIT SET CASTING\_ROLE=\""+actorRole.getText()+"\" WHERE WORKING\_TITLE\_ID = "+ tempId+" and applied\_actor = \""+actorName.getText()+"\";");

result.setText("Actor Picked");

}

**catch**(SQLException e1){

System.***out***.println("pick actor error");

e1.printStackTrace();

}

}

**else**{

}}

**catch**(SQLException e3){

System.***out***.println("connection in event handling error");

}

}

}

}

//1515004 권나현

**class** ActorFrame **extends** JFrame{

**public** String name1,gender,debut\_year;

**public** String name2,movie1,role;

**public** String name3;

**int** movie2;

**public** **int** career,year;

//사용자 입력을 받아 저장할 변수 선언

**public** JButton b1,b2,b3,b4,b5,b6;

**public** JTextField t1,t2,t3,t4,t5,t6,t7,t8,t9;

**public** TextArea result;

//public JScrollPane scrollPane;

//gui component 선언

**public** ActorFrame(){

setSize(1000,800);

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setTitle("Actor");

JPanel panel = **new** JPanel();

//최종 panel

//panel.setLayout(new GridLayout(11,1));

JPanel topic1 = **new** JPanel();

JPanel topic2 = **new** JPanel();

JPanel topic3 = **new** JPanel();

JPanel topic4 = **new** JPanel();

JPanel panel1 = **new** JPanel();

JPanel panel2 = **new** JPanel();

JPanel panel3 = **new** JPanel();

JPanel panel4 = **new** JPanel();

JPanel panel5 = **new** JPanel();

JPanel panel6 = **new** JPanel();

JPanel panel7 = **new** JPanel();

JPanel panel8 = **new** JPanel();

JPanel panel9 = **new** JPanel();

JPanel panel10 = **new** JPanel();

JPanel panel11 = **new** JPanel();

JLabel l1 = **new** JLabel("<Input data of actor, and press 'insert' or 'delete' button.>");

JLabel l2 = **new** JLabel("<Input data of actor's filmography, and press 'insert' of 'delete' button.>");

JLabel l3 = **new** JLabel("<Input data of movie code that you want to apply, your name and press 'apply' button.> ");

JLabel l4 = **new** JLabel("<If you want to know what movies are accepting application, press 'search' button.>");

t1 = **new** JTextField("name",20);

t2 = **new** JTextField("gender(f/m)",20);

t3 = **new** JTextField("debut year",20);

t4 = **new** JTextField("actor name",20);

t5 = **new** JTextField("movie",20);

t6 = **new** JTextField("launched year",20);

t7 = **new** JTextField("role",20);

t8 = **new** JTextField("movie code",20);

t9 = **new** JTextField("name",20);

//사용자의 입력을 받기 위한 텍스트필드 생성

b1 = **new** JButton("insert");

b2 = **new** JButton("delete");

b3 = **new** JButton("insert");

b4 = **new** JButton("delete");

b5 = **new** JButton("apply");

b6 = **new** JButton("search");

//버튼 생성

b1.addActionListener(**new** ActActionListener());

b2.addActionListener(**new** ActActionListener());

b3.addActionListener(**new** ActActionListener());

b4.addActionListener(**new** ActActionListener());

b5.addActionListener(**new** ActActionListener());

b6.addActionListener(**new** ActActionListener());

//button에 action 추가

result = **new** TextArea();

result.setSize(100,20);

// scrollPane = new JScrollPane(result);

// scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_ALWAYS);

//필요한 테이블 정보를 받아 올 text area

panel1.add(t1);

panel1.add(t2);

panel1.add(t3);

panel2.add(b1);

panel2.add(b2);

panel3.add(result);

panel4.add(t4);

panel4.add(t5);

panel4.add(t6);

panel4.add(t7);

panel6.add(b3);

panel6.add(b4);

panel5.add(t8);

panel5.add(t9);

panel5.add(b5);

panel7.add(b6);

panel8.add(l4);

panel9.add(l1);

panel10.add(l2);

panel11.add(l3);

topic1.setLayout(**new** GridLayout(3,1));

topic1.add(panel9);//l1

topic1.add(panel1);

topic1.add(panel2);

topic2.setLayout(**new** GridLayout(3,1));

topic2.add(panel10);//l2

topic2.add(panel4);

topic2.add(panel6);

topic3.setLayout(**new** GridLayout(2,1));

topic3.add(panel11);//l3

topic3.add(panel5);

topic4.setLayout(**new** GridLayout(3,1));

topic4.add(panel8);//l4

topic4.add(panel3);

topic4.add(panel7);

panel.add(topic1);

panel.add(topic2);

panel.add(topic3);

panel.add(topic4);

add(panel);

setVisible(**true**);

}

**class** ActActionListener **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent e){

**try** {

**if**(e.getSource() == b1){

//배우 정보 insert

name1 = t1.getText();

gender = t2.getText();

debut\_year = t3.getText();

career = 2017-(Integer.*parseInt*(debut\_year));

//텍스트필드에서 입력값을 받아 로컬 변수에 저장

PreparedStatement pStmt = *con*.prepareStatement("insert into dbcourse\_actor "

+ "values (?,?,?,?)");

pStmt.setString(1, name1);

pStmt.setString(2, gender);

pStmt.setString(3, debut\_year);

pStmt.setInt(4, career);

//입력값을 이용해 dbcourse\_actor 테이블에 배우 정보 insert

pStmt.executeUpdate();

//prepared statement 실행

result.setText(name1+ " 's actor information was successfully inserted.");

}

**if**(e.getSource() == b2){

//배우 정보 delete

name1 = t1.getText();

//배우의 정보 삭제는 이름만 입력해도 가능한 것으로 설정

PreparedStatement pStmt = *con*.prepareStatement("delete from dbcourse\_actor "

+ "where name = ?");

pStmt.setString(1,name1);

//텍스트필드에서 받아온 입력값을 사용해 dbcourse\_actor 테이블에서 배우 정보 delete

pStmt.executeUpdate();

//prepared statement 실행

result.setText(name1+ " 's actor information was successfully deleted.");

}

**if**(e.getSource() == b3){

//배우 필모그래피 insert

name2 = t4.getText();

movie1 = t5.getText();

year = Integer.*parseInt*(t6.getText());

role = t7.getText();

//배우 경력에 관한 정보를 텍스트필드에서 받아옴

PreparedStatement pStmt = *con*.prepareStatement("insert into dbcourse\_filmography "

+ "values (?,?,?,?)");

pStmt.setString(1,name2);

pStmt.setString(2,movie1);

pStmt.setInt(3,year);

pStmt.setString(4,role);

//받아온 레코드를 dbcourse\_filmography 테이블에 insert

pStmt.executeUpdate();

result.setText(name2+ " 's filmography was successfully inserted.");

}

**if**(e.getSource() == b4){

//배우 필모그래피 삭제

name2 = t4.getText();

movie1 = t5.getText();

year = Integer.*parseInt*(t6.getText());

role = t7.getText();

//필모그래피의 정보는 테이블의 모든 정보를 모두 입력해야 delete 가능

PreparedStatement pStmt = *con*.prepareStatement("delete from dbcourse\_filmography "

+ "where actor\_name = ? and movie =? and year = ? and role = ?");

pStmt.setString(1,name2);

pStmt.setString(2,movie1);

pStmt.setInt(3,year);

pStmt.setString(4,role);

pStmt.executeUpdate();

//입력값을 사용해 dbcourse\_filmography 테이블에서 배우 경력 정보 delete

result.setText(name2+" 's filmography was successfully deleted.");

}

**if**(e.getSource() == b5){

//조회 화면에 띄워진 정보를 보고 영화 지원

/\*배우가 영화에 지원

movie와 title\_id를 working\_title\_id로 조인하여

조회 화면에서 movie의 모든 정보를 제공한다(가제 포함)

배우는 조회 화면에서 지원할 영화를 골라

이름과 지원영화를 쓰고 지원 버튼을 누른다

지원영화를 보고 title\_id 테이블을 이용해 영화의 아이디를 알아낸다

아이디를 이용해 recruit 테이블에 영화의 아이디, 배우 이름을 인서트 하고

audition 테이블의 모든 정보를 조회 화면에 띄워준다

\*/

movie2 = Integer.*parseInt*(t8.getText());

name3 = t9.getText();

PreparedStatement pStmt = *con*.prepareStatement("insert into dbcourse\_recruit "

+ "values (?,?,null)");

pStmt.setInt(1, movie2);

pStmt.setString(2, name3);

pStmt.executeUpdate();

result.setText("<Audition information of the movie that you applied>"+"\n"+"working title|movie code|audition date|announcement date"+"\n");

ResultSet rset = *st*.executeQuery("select distinct t.working\_title,t.working\_title\_id,a.aud\_date,a.announce\_date "

+ "from dbcourse\_recruit as r, dbcourse\_audition as a, dbcourse\_title\_id as t "

+ "where r.working\_title\_id = a.working\_title\_id "

+ "and a.working\_title\_id = t.working\_title\_id "

+ "and t.working\_title\_id = "+movie2);

// JOIN을 이용하여 사용자가 지원한 영화의 오디션 정보를 받아온다

**while**(rset.next()){

result.append(rset.getString("t.working\_title")+" | "

+rset.getString("t.working\_title\_id")+" | "

+rset.getString("a.aud\_date")+" | "

+rset.getString("a.announce\_date")+"\n");

}

//받아온 정보를 TextArea result에 출력한다

}

**if**(e.getSource() == b6){

//지원할 수 있는 영화 정보 조회

ResultSet rset = *st*.executeQuery("select t.working\_title,m.working\_title\_id,m.director,m.genre "

+ "from dbcourse\_title\_id as t,dbcourse\_movie as m "

+ "where t.working\_title\_id = m.working\_title\_id");

//JOIN을 이용하여 영화 정보를 가져온다. movie 테이블에는 영화 제목이 아닌

//영화 코드가 저장되어 있으므로 movie 테이블과 title\_id 테이블을 조인하여 영화 제목을 추가하여 출력해 준다.

//

result.setText("<Movie information>"+"\n"+"working title|movie code|director|genre"+"\n");

**while**(rset.next()){

result.append(rset.getString("t.working\_title")+" | "

+rset.getString("m.working\_title\_id")+" | "

+rset.getString("m.director")+" | "

+rset.getString("m.genre")+"\n");

}

//TextArea에 지원 가능한 영화 정보 출력

}

} **catch** (SQLException sqex) {

System.***out***.println("SQLException: " + sqex.getMessage());

System.***out***.println("SQLState: " + sqex.getSQLState());

}

}

}}

//1515005 권성경

**class** InvestorFrame **extends** JFrame{

JTextField invst\_name;

JTextField worktitle;

JTextField investment;

JButton insert;

JButton delete;

JLabel expl;

JPanel dinfopanel;

TextArea direc;

//============생성자===============

**public** InvestorFrame(){ setSize(1000,820);

setTitle("Investor");

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setVisible(**true**);

JPanel txtpanel = **new** JPanel();

txtpanel.setLayout(**new** GridLayout(3,1));

//=======insert,delete investor field=============

//======explain textfield======

JPanel texplainpanel = **new** JPanel();

expl = **new** JLabel("Input information of investment and press 'input' or 'delete'.");

expl.setFont(**new** Font("b",Font.***PLAIN***,20));

texplainpanel.add(expl);

txtpanel.add(texplainpanel);

//======textfield==========

JPanel tfieldpanel = **new** JPanel();

invst\_name = **new** JTextField("name",20);

worktitle = **new** JTextField("title",20);

investment = **new** JTextField("amount to invest",20);

tfieldpanel.add(invst\_name);

tfieldpanel.add(worktitle);

tfieldpanel.add(investment);

txtpanel.add(tfieldpanel);

//=====insert, delete button========

JPanel tbuttonpanel = **new** JPanel();

insert = **new** JButton("insert");

delete = **new** JButton("delete");

insert.addActionListener(**new** IActionListener());

delete.addActionListener(**new** IActionListener());

tbuttonpanel.add(insert);

tbuttonpanel.add(delete);

txtpanel.add(tbuttonpanel);

add(txtpanel,BorderLayout.***PAGE\_START***);

//===end of insert,delete investor field========

//======show movie info=============

JPanel minfopanel = **new** JPanel();

TextArea movi = **new** TextArea();

minfopanel.add(movi);

add(minfopanel,BorderLayout.***CENTER***);

movi.setFont(**new** Font("myfont",Font.***PLAIN***,20));

movi.setText("====================================================================="+"\n"

+"가제 "+"\t||\t"+"감독"+"\t||\t"+"장르"+"\n"

+"====================================================================="+"\n");

**try**{

//영화의 정보를 rs에 받아서 보여줌

*st*= *con*.createStatement();

*st*.executeQuery("select working\_title, director, genre "

+ "from dbcourse\_movie, dbcourse\_title\_id "

+ "where dbcourse\_movie.working\_title\_id=dbcourse\_title\_id.working\_title\_id;");

*rs*=*st*.getResultSet();

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

String wt = *rs*.getString("working\_title");

String dr = *rs*.getString("director");

String gn = *rs*.getString("genre");

movi.append(wt+" || "+dr+" || "+gn+"\n");

}

}

**catch** (SQLException sqex) {

System.***out***.println("SQLException: " + sqex.getMessage());

System.***out***.println("SQLState: " + sqex.getSQLState());

}

//=======end of show movie===============

//=========show dir info===============

dinfopanel = **new** JPanel();

direc = **new** TextArea(3,0);

dinfopanel.add(direc);

dinfopanel.setSize(900, 300);

direc.setSize(900, 300);

add(dinfopanel,BorderLayout.***PAGE\_END***);

direc.setFont(**new** Font("myfont",Font.***PLAIN***,20));

direc.setText("====================================================================="+"\n"

+"감독 "+"\t"+"||"+"\t"+"영화"+"\t"+"||"+"\t"+"영화제"+"\t"+"\n"

+"====================================================================="+"\n");

**try**{

//감독의 정보를 받아서 보여줌

*st*= *con*.createStatement();

*st*.executeQuery("select name, movie, festiv\_name "

+ "from (select name, dbcourse\_director.movie, festiv\_name "

+ "from dbcourse\_director, dbcourse\_film\_festival "

+ "where dbcourse\_director.name = dbcourse\_film\_festival.director) as T;");

*rs*=*st*.getResultSet();

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

String nm = *rs*.getString("name");

String mv = *rs*.getString("movie");

String fn = *rs*.getString("festiv\_name");

direc.append(nm+"\t||\t"+mv+"\t||\t"+fn+"\n");

}

}

**catch** (SQLException sqex) {

System.***out***.println("SQLException: " + sqex.getMessage());

System.***out***.println("SQLState: " + sqex.getSQLState());

}

//==========================================

}

**class** IActionListener **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent e){

//투자자 삽입

**if**(e.getSource()==insert){

**try**{

*st*= *con*.createStatement();

//현재 투자금과 예산을 가져옴

*st*.executeQuery("select current\_investment, budget "

+ "from dbcourse\_movie\_asset "

+ "where working\_title\_id = (select working\_title\_id "

+ "from dbcourse\_title\_id "

+ "where working\_title = \""+worktitle.getText()+"\");");

*rs*=*st*.getResultSet();

*rs*.next();

String s1=*rs*.getString("current\_investment");

String s2 = *rs*.getString("budget");

**int** cur = Integer.*parseInt*(s1);

**int** bud = Integer.*parseInt*(s2);

//만약 투자하고자 하는 금액을 추가하였을 때 예산을 넘는다면 insert하지 않음

**if**(bud<(cur+(Integer.*parseInt*(investment.getText())))){

expl.setText("Your investment is over budget!");

}

//그렇지 않으면 insert

**else**{

*st*= *con*.createStatement();

//id를 먼저 가져옴

*st*.executeQuery("select working\_title\_id "

+ "from dbcourse\_title\_id "

+ "where working\_title = \""+worktitle.getText()+"\";");

*rs*=*st*.getResultSet();

*rs*.next();

**int** id = *rs*.getInt(1);

//transation을 위한 boolean값

**boolean** insertsuccess=**false**;

**try**{

//transation

*con*.setAutoCommit(**false**);

PreparedStatement pStmt = *con*.prepareStatement("insert into dbcourse\_investor "

+ "values (?,?,?)");

pStmt.setString(1,invst\_name.getText() );

pStmt.setInt(2,id);

pStmt.setInt(3,(Integer.*parseInt*(investment.getText())) );

pStmt.executeUpdate();

*st*.executeUpdate("update dbcourse\_movie\_asset "

+ "set current\_investment = current\_investment+"

+investment.getText()+" where working\_title\_id = "+id+";");

//성공하면 true

insertsuccess=**true**;

}

**catch**(SQLException sqle){

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

}

**finally**{

**if**(insertsuccess){

*con*.commit();

expl.setText("Invested successfully!");

direc.setText("====================================================================="+"\n"

+"감독"+"\t"+"||"+"\t"+"영화"+"\t"+"||"+"\t"+"투자금"+"\t"+"\n"

+"====================================================================="+"\n");

**try**{

//투자 정보를 받아서 보여줌

*st*= *con*.createStatement();

*st*.executeQuery("select inv\_name, working\_title, investment "

+ "from dbcourse\_investor, dbcourse\_title\_id "

+ "where dbcourse\_investor.working\_title\_id = dbcourse\_title\_id.working\_title\_id; ");

*rs*=*st*.getResultSet();

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

String nm = *rs*.getString("inv\_name");

String mv = *rs*.getString("working\_title");

String in = *rs*.getString("investment");

direc.append(nm+"\t||\t"+mv+"\t||\t"+in+"\n");

}

//4초 간 보여줌

**try** {

Thread.*sleep*(4 \* 1000);

} **catch** (InterruptedException ee) { }

//4초 후 다시 감독 정보 보여줌

direc.setText("");

direc.setText("====================================================================="+"\n"

+"감독 "+"\t"+"||"+"\t"+"영화"+"\t"+"||"+"\t"+"영화제"+"\t"+"\n"

+"====================================================================="+"\n");

*st*.executeQuery("select name, movie, festiv\_name "

+ "from (select name, dbcourse\_director.movie, festiv\_name "

+ "from dbcourse\_director, dbcourse\_film\_festival "

+ "where dbcourse\_director.name = dbcourse\_film\_festival.director) as T;");

*rs*=*st*.getResultSet();

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

String nm = *rs*.getString("name");

String mv = *rs*.getString("movie");

String fn = *rs*.getString("festiv\_name");

direc.append(nm+"\t||\t"+mv+"\t||\t"+fn+"\n");

}

}

**catch** (SQLException sqex) {

System.***out***.println("SQLException: " + sqex.getMessage());

System.***out***.println("SQLState: " + sqex.getSQLState());

}

}

**else**{

//transaction 실패한 경우

*con*.rollback();

}

}

}

}

**catch** (SQLException sqex) {

sqex.printStackTrace();

System.***out***.println("SQLException: " + sqex.getMessage());

System.***out***.println("SQLState: " + sqex.getSQLState());

}

}

//투자 취소

**else** **if**(e.getSource()==delete){

**try**{

*st*= *con*.createStatement();

//id먼저 가져옴

*st*.executeQuery("select working\_title\_id "

+ "from dbcourse\_title\_id "

+ "where working\_title = \""+worktitle.getText()+"\";");

*rs*=*st*.getResultSet();

*rs*.next();

**int** id = *rs*.getInt(1);

//투자자가 투자한 금액 가져옴

*st*.executeQuery("select investment "

+ "from dbcourse\_investor "

+ "where working\_title\_id = "+id+" and inv\_name='"+invst\_name.getText()+"';");

*rs*=*st*.getResultSet();

*rs*.next();

**int** invmn = *rs*.getInt(1);

**boolean** deletesuccess=**false**;

//투자금보다 많은 금액은 취소할 수 없음

**if**(invmn<Integer.*parseInt*(investment.getText())){

expl.setText("You can not cancel more than the investment amount.");

}

//투자금==취소하고자 하는 금액일 경우 movie\_asset 뿐 아니라 investor table에서도 제거함

**else** **if**(invmn==Integer.*parseInt*(investment.getText())){

**try**{

//transaction

*con*.setAutoCommit(**false**);

//movie\_asset테이블에서 삭제

*st*.executeUpdate("update dbcourse\_movie\_asset "

+ "set current\_investment = current\_investment-"

+investment.getText()+" where working\_title\_id = "+id+";");

//investor테이블에서 삭제

*st*.executeUpdate("delete from dbcourse\_investor "

+ "where working\_title\_id = "+id+" and inv\_name = \""+invst\_name.getText()+"\";");

deletesuccess=**true**;

}

**catch**(SQLException sqle){

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

}

**finally**{

**if**(deletesuccess){

*con*.commit();

expl.setText("Investment canceled.");

}

**else**{

*con*.rollback();

}

}

}

//투자금> 취소하고자 하는 금액 일 경우 delete하지 않고 투자 금액만 감소 시킴

**else**{

**try**{

//transaction

*con*.setAutoCommit(**false**);

//movie\_asset테이블에서 투자금 감소시킴

*st*.executeUpdate("update dbcourse\_movie\_asset "

+ "set current\_investment = current\_investment-"

+investment.getText()+" where working\_title\_id = "+id+";");

//investor테이블에서 투자금 감소시킴

*st*.executeUpdate("update dbcourse\_investor "

+ "set investment = investment-"

+investment.getText()+" where working\_title\_id = "+id+";");

deletesuccess=**true**;

}

**catch**(SQLException sqle){

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

}

**finally**{

**if**(deletesuccess){

*con*.commit();

expl.setText("amount of Investment is decreased.");

}

**else**{

*con*.rollback();

}

}

}

}

**catch** (SQLException sqex) {

System.***out***.println("SQLException: " + sqex.getMessage());

System.***out***.println("SQLState: " + sqex.getSQLState());

}

}

}//ActionPerformed 끝

}

}

//공통 main method

**public** **static** **void** main(String[] args) {

**try** {

*con* = **null**;

*st* = **null**;

*rs* = **null**;

*con* = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/team15?autoReconnect=true&useSSL=false",

"team15", "team15");

*st* = *con*.createStatement();

//1515009 김다영

*st*.executeUpdate("CREATE TABLE DBCOURSE\_DIRECTOR("

+ "NAME VARCHAR(30) PRIMARY KEY,"

+ "MOVIE VARCHAR(30));");

*st*.executeUpdate("CREATE TABLE DBCOURSE\_TITLE\_ID("

+ "WORKING\_TITLE\_ID INT PRIMARY KEY,"

+ "WORKING\_TITLE VARCHAR(30) NOT NULL,"

+ "DONE VARCHAR(30));");

//1515005 권성경

*st*.executeUpdate("create table DBCOURSE\_MOVIE("

+ "working\_title\_id int not null,"

+ "director varchar(30) not null,"

+ "genre varchar(30),"

+ "primary key (working\_title\_id),"

+ "foreign key (working\_title\_id) references dbcourse\_title\_id(working\_title\_id));");

//1515009 김다영

*st*.executeUpdate("CREATE TABLE DBCOURSE\_FILM\_FESTIVAL("

+ "MOVIE VARCHAR(30) NOT NULL,"

+ "DIRECTOR VARCHAR(30) NOT NULL,"

+ "FESTIV\_NAME VARCHAR(30) PRIMARY KEY,"

+ "FOREIGN KEY(DIRECTOR) REFERENCES dbcourse\_DIRECTOR(NAME));");

//1515005 권성경

*st*.executeUpdate("create table DBCOURSE\_INVESTOR("

+ "inv\_name varchar(30) not null,"

+ "working\_title\_id int not null,"

+ "investment int,"

+ "primary key (inv\_name, working\_title\_id));");

*st*.executeUpdate("create table DBCOURSE\_movie\_asset("

+ "working\_title\_id int not null,"

+ "current\_investment int,"

+ "budget int,"

+ "primary key(working\_title\_id),"

+ "foreign key(working\_title\_id) references dbcourse\_movie(working\_title\_id));");

//1515004 권나현

*st*.executeUpdate("create table DBCOURSE\_ACTOR("

+ "name varchar(30) not null,"

+ "gender char(1),"

+ "debut\_year varchar(30),"

+ "career int,"

+ "primary key(name));");

*st*.executeUpdate("create table DBCOURSE\_FILMOGRAPHY("

+ "actor\_name varchar(30) not null,"

+ "movie varchar(30) not null,"

+ "year int,"

+ "role varchar(30),"

+ "primary key(actor\_name),"

+ "foreign key(actor\_name) references DBCOURSE\_ACTOR(name));");

*st*.executeUpdate("create table DBCOURSE\_RECRUIT("

+ "working\_title\_id int(5) not null,"

+ "applied\_actor varchar(30) not null,"

+ "casting\_role varchar(30),"

+ "primary key(working\_title\_id,applied\_actor),"

+ "foreign key(applied\_actor) references DBCOURSE\_ACTOR(name));");

*st*.executeUpdate("create table DBCOURSE\_AUDITION("

+ "working\_title\_id int(5) not null,"

+ "aud\_date date,"

+ "announce\_date date,"

+ "primary key(working\_title\_id));");

*st*.executeUpdate("create view invest as "

+ "select \* "

+ "from DBCOURSE\_MOVIE, DBCOURSE\_INVESTOR "

+ "where dbcourse\_movie.working\_title\_id = dbcourse\_investor.working\_title\_id;");

*st*.executeUpdate("create view asset as "

+ "select \* "

+ "from DBCOURSE\_MOVIE, DBCOURSE\_movie\_asset, DBCOURSE\_investor "

+ "where dbcourse\_movie.working\_title\_id = dbcourse\_investor.working\_title\_id;");

//1515005 권성경

*st*.executeUpdate("insert into dbcourse\_director "

+ "values(\"Junho Bong\",\"The Host\");");

*st*.executeUpdate("insert into dbcourse\_director "

+ "values(\"Steven Spielberg\",\"Saving Private\");");

//1515009 김다영

*st*.executeUpdate("insert into DBCOURSE\_DIRECTOR "

+ "values(\"Christopher Nolan\",\"Interstella\");");

//1515005 권성경

*st*.executeUpdate("insert into dbcourse\_title\_id "

+ "values(10000,\"Parasite\",null);");

*st*.executeUpdate("insert into dbcourse\_title\_id "

+ "values(20000,\"Ready Play One\",null);");

*st*.executeUpdate("insert into dbcourse\_movie "

+ "values(10000,\"Junho Bong\",\"Action\");");

*st*.executeUpdate("insert into dbcourse\_movie "

+ "values(20000,\"Steven Spielberg\",\"SF\");");

//1515005 김다영

*st*.executeUpdate("insert into DBCOURSE\_TITLE\_ID "

+ "values(20001,\"The Post\",NULL);");

*st*.executeUpdate("insert into DBCOURSE\_TITLE\_ID "

+ "values(30000,\"Dunkirk\",NULL);");

*st*.executeUpdate("insert into DBCOURSE\_MOVIE "

+ "values(20001,\"Steven Spielberg\",\"drama\");");

*st*.executeUpdate("insert into DBCOURSE\_MOVIE "

+ "values(30000,\"Christopher Nolan\",\"action\");");

//1515005 권성경

*st*.executeUpdate("insert into dbcourse\_investor "

+ "values(\"CJ\",10000,10000);");

*st*.executeUpdate("insert into dbcourse\_investor "

+ "values(\"new\",20000,1000000);");

*st*.executeUpdate("insert into dbcourse\_investor "

+ "values(\"new\",20001,1500000);");

*st*.executeUpdate("insert into dbcourse\_investor "

+ "values(\"new\",30000,3000000);");

*st*.executeUpdate("insert into dbcourse\_movie\_asset "

+ "values(10000,10000,1000000);");

*st*.executeUpdate("insert into dbcourse\_movie\_asset "

+ "values(20000,1000000,3000000);");

*st*.executeUpdate("insert into dbcourse\_movie\_asset "

+ "values(20001,1500000,2000000);");

*st*.executeUpdate("insert into dbcourse\_movie\_asset "

+ "values(30000,3000000,3000000);");

//1515004 권나현

*st*.executeUpdate("insert into dbcourse\_actor "

+ "values('Kangho Song','m',1996,21),"

+ "('Simon Pegg','m',1999,18),"

+ "('Olivia Cooke','f',2012,5),"

+ "('Meryl Streep','f',1977,40),"

+ "('Mark Rylance','m',1985,32),"

+ "('Tom Hardy','m',2001,16),"

+ "('Tom Hanks','m',1980,37),"

+ "('T.J.Miller','m',2007,10),"

+ "('Julie Nickson','f',1980,37),"

+ "('Aneurin Barnard','m',2011,6);");

//1515005 권성경

*st*.executeUpdate("insert into dbcourse\_recruit "

+ "values(10000, \"Kangho Song\",\"Main\"),"

+ "(20000,\"Simon Pegg\",\"Sub\"),"

+ "(20000,\"Olivia Cooke\",null),"

+ "(20000,\"Meryl Streep\", \"Main\"),"

+ "(20000,\"Mark Rylance\",null),"

+ "(30000,\"Mark Rylance\",null),"

+ "(30000,\"Tom Hardy\",null),"

+ "(20001,\"Tom Hanks\",null),"

+ "(20000,\"T.J. Miller\",null),"

+ "(20001,\"Julie Nickson\",null),"

+ "(30000,\"Aneurin Barnard\",null);");

//1515009 김다영

*st*.executeUpdate("insert into DBCOURSE\_FILM\_FESTIVAL "

+ "values(\"Saving Private\",\"Steven Spielberg\",\"Academy\");");

*st*.executeUpdate("insert into DBCOURSE\_FILM\_FESTIVAL "

+ "values(\"Interstella\",\"Christopher Nolan\",\"Max Movie\");");

//1515004 권나현

*st*.executeUpdate("insert into dbcourse\_filmography "

+ "values('Kangho Song','The Host',2006,'main'),"

+ "('Simon Pegg','Star Wars',2015,'main'),"

+ "('Olivia Cooke','The Signal',2014,'main'),"

+ "('Meryl Streep','The Devil wears Prada',2006,'main'),"

+ "('Mark Rylance','Bridge of Spies',2015,'main'),"

+ "('Tom Hardy','Mad Max',2015,'main'),"

+ "('Tom Hanks','Bridge of Spies',2015,'main'),"

+ "('T.J.Miller','Deadpool',2016,'sub'),"

+ "('Julie Nickson','Rambo2',1985,'sub'),"

+ "('Aneurin Barnard','Legend',2015,'sub');");

*st*.executeUpdate("insert into dbcourse\_audition "

+ "values(10000,'2017-07-25','2017-08-31'),"

+ "(20000,'2017-06-21','2017-07-01'),"

+ "(20001,'2017-05-27','2017-06-03'),"

+ "(30000,'2017-05-27','2017-06-05');");

*st*.executeUpdate("alter table DBCOURSE\_recruit add index(applied\_actor(3))");

*st*.executeUpdate("alter table DBCOURSE\_investor add index(inv\_name(2))");

*st*.executeUpdate("alter table DBCOURSE\_actor add index(name(5))");

*st*.executeUpdate("alter table DBCOURSE\_title\_id add index(working\_title(5))");

*st*.executeUpdate("alter table DBCOURSE\_filmography add index(actor\_name(3))");

*rs* = *st*.executeQuery("show databases;");

**if** (*st*.execute("show databases;")) {

*rs* = *st*.getResultSet();

}

**new** jdbcexam().**new** MenuFrame();

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

String str = *rs*.getNString(1);

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

}

} **catch** (SQLException sqex) {

System.***out***.println("SQLException: " + sqex.getMessage());

System.***out***.println("SQLState: " + sqex.getSQLState());

}

}

}