2018-1 Database

Team Project

Team Name: Ya!nado(Team2)

[Student ID]

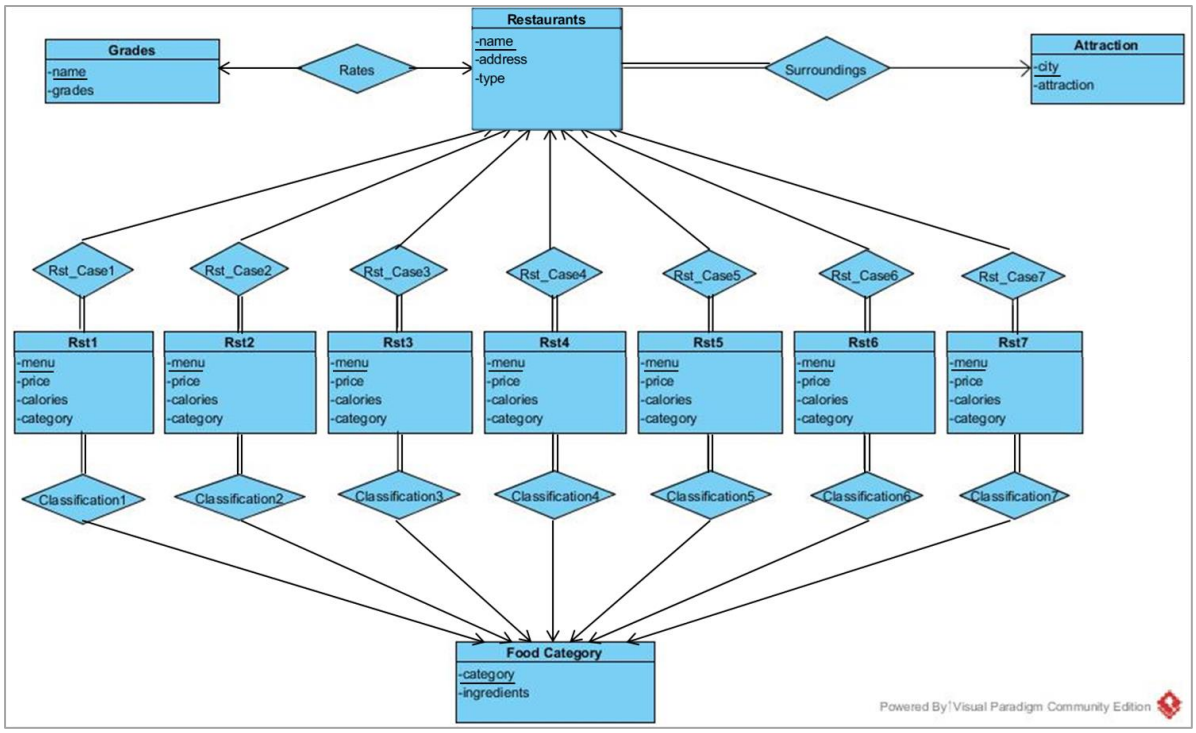
1615039 Nagyeong Yeo

1615047 Dokyung Lee

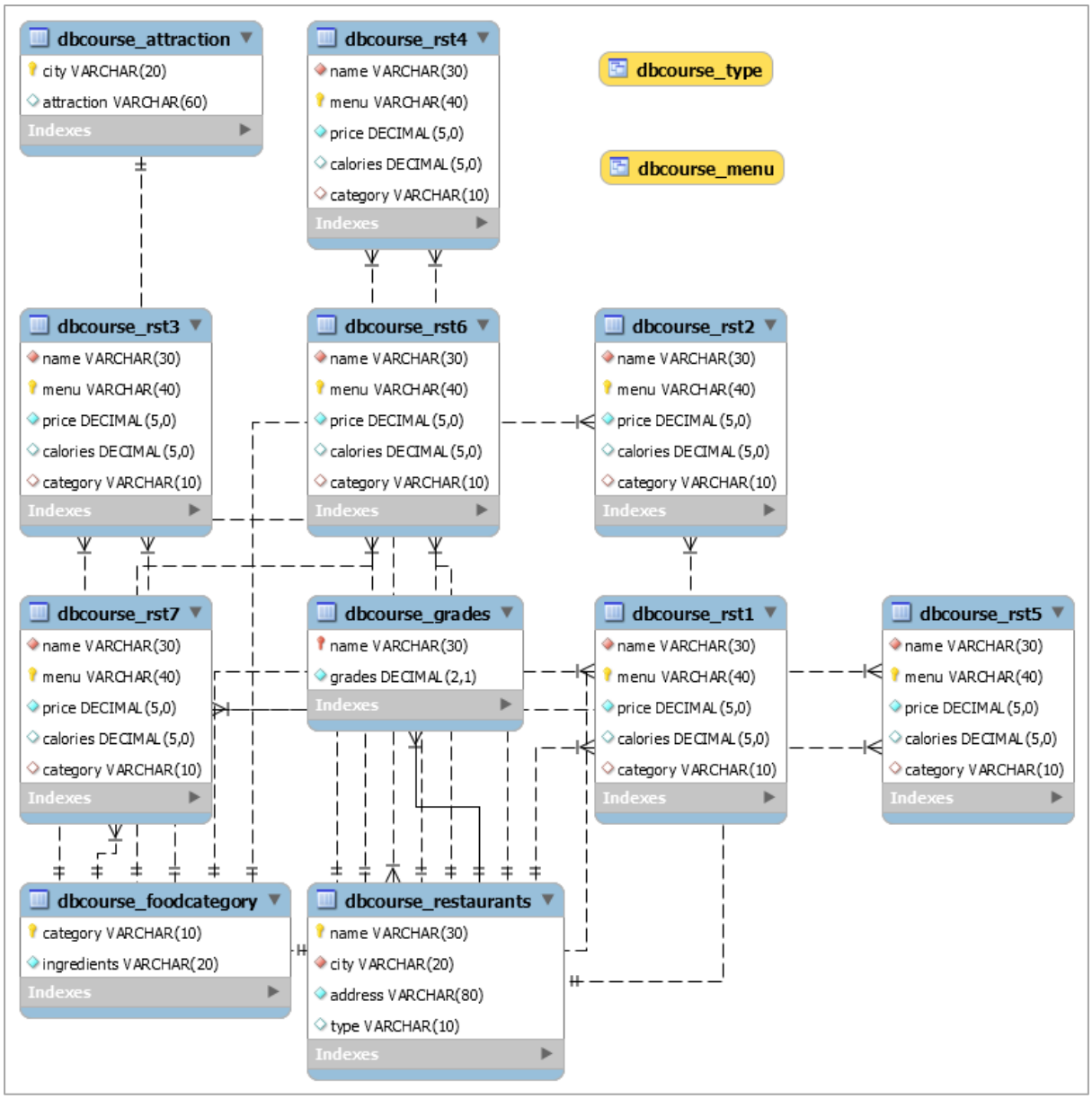
1615051 Youngeun Lee

1615058 Hyojin Lee

1. ER Diagram


Restaurants
-name
-address
-type
Grades
-name
-grades
Rates
Surroundings
Attraction
-city
-attraction
Rst_Case1
Rst_Case2
Rst_Case3
Rst_Case4
Rst_Case5
Rst_Case6
Rst_Case7
Rst1
Rst2
Rst3
Rst4
Rst5
Rst6
Rst7
-menu
-price
-calories
-category
Classification1
Classification2
Classification3
Classification4
Classification5
Classification6
Classification7
Food Category
-category
-ingredients
Powered By Visual Paradigm Community Edition

2. Database schema diagram



3. Class and method explanation

(1) Open

A. Class

team2

Class Open

java.lang.Object
    team2.Open

```
public class Open
extends java.lang.Object
```

class that is a window for selection

Author:

YoungEun Lee, Dokyung Lee, HyoJin Lee

Constructor Summary

Constructors

Constructor and Description
Open()
initialize the application.

B. Method

Method Summary

All Methods | Static Methods | Concrete Methods

Modifier and Type	Method and Description
static void	main(java.lang.String[] args) Launch the application.

Methods inherited from class java.lang.Object

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

Constructor Detail

Open

public Open()

initialize the application.

Method Detail

main

public static void main(java.lang.String[] args)

Launch the application.

(2) Adm

A. Class

team2

Class Adm

java.lang.Object
    team2.Adm

public class **Adm**
extends java.lang.Object

class that is a window for administrator

Author:

YoungEun Lee, DoKyung Lee, HyoJin Lee, NaGyeong Yeo

Constructor Summary

Constructors

Constructor and Description
Adm() initialize the application.

B. Method

Method Summary

All Methods | Static Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
static void	AdmScreen() open the Adm window.
void	whichTable() whichTable is method that is used when we know only restaurant's name, not a table's name restaurant's name is changed into real table's name of database

Methods inherited from class java.lang.Object

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

Constructor Detail

Adm

public Adm()

initialize the application.

Method Detail

AdmScreen

public static void AdmScreen()

open the Adm window.

whichTable

public void whichTable()

whichTable is method that is used when we know only restaurant's name, not a table's name restaurant's name is changed into real table's name of database

(3) Cus

A. Class

team2

Class Cus

java.lang.Object
    team2.Cus

```
public class Cus
extends java.lang.Object
```

class that is a window for customer

Author:

YoungEun Lee, DoKyung Lee, HyoJin Lee, NaGyeong Yeo

Constructor Summary

Constructors

Constructor and Description
Cus() initialize the application.

B. Method

Method Summary

All Methods | Static Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
static void	CusScreen() open the Cus window
void	whichTable() whichTable is method that is used when we know only restaurant's name, not a table's name restaurant's name is changed into real table's name of database

Methods inherited from class java.lang.Object

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

Constructor Detail

Cus

public Cus()

initialize the application.

Method Detail

CusScreen

public static void CusScreen()

open the Cus window

whichTable

public void whichTable()

whichTable is method that is used when we know only restaurant's name, not a table's name restaurant's name is changed into real table's name of database

(4) Insert

A. Class

team2

Class insert

java.lang.Object
    team2.insert

```
public class insert
extends java.lang.Object
```

class that insert some data of request for administrator

Author:

HyoJin Lee, NaGyeong Yeo

Constructor Summary

Constructors

Constructor and Description
`insert(java.lang.String category, java.lang.String ingredients)` constructor to insert category
`insert(java.lang.String tableName, java.lang.String name, java.lang.String menu, int price, java.lang.String calories, java.lang.String category)` constructor to insert menu

B. Method

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
void	**CategoryInsert**() insert new category(+ingredients) into FoodCategory table
void	**RestaurantInsert**() insert new menu(+price, etc) into specific Restaurant table

Methods inherited from class java.lang.Object

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

Constructor Detail

insert

```
public insert(java.lang.String category,
              java.lang.String ingredients)
```

constructor to insert category

Parameters:

category –

ingredients –

insert

```
public insert(java.lang.String tableName,
              java.lang.String name,
              java.lang.String menu,
              int price,
              java.lang.String calories,
              java.lang.String category)
```

constructor to insert menu

Parameters:

tableName –

name –

menu –

price –

calories –

category –

Method Detail

CategoryInsert

```
public void CategoryInsert()
```

insert new category(+ingredients) into FoodCategory table

Parameters:

category -

ingredients -

RestaurantInsert

```
public void RestaurantInsert()
```

insert new menu(+price, etc) into specific Restaurant table

Parameters:

tableName -

name -

menu -

price -

calories -

category -

(5) Delete

A. Class

team2

Class delete

java.lang.Object
    team2.delete

```
public class delete
extends java.lang.Object
```

class that delete some data for request of administrator

Author:

HyoJin Lee

Constructor Summary

Constructors

Constructor and Description
delete(java.lang.String tableName, int min, int max) constructor
delete(java.lang.String tableName, java.lang.String menu) constructor

B. Method

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
void	Calorydelete() delete tuples in the restaurant table that included in the calories range
void	Pricedelete() delete tuples in the restaurant table that included in the price range
void	Restaurantdelete() delete the menu

Methods inherited from class java.lang.Object

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

Constructor Detail

delete

```
public delete(java.lang.String tableName,
              java.lang.String menu)
```

constructor

Parameters:

tableName -

menu -

delete

```
public delete(java.lang.String tableName,
              int min,
              int max)
```

constructor

Parameters:

tableName -

min -

max -

Method Detail

Restaurantdelete

```
public void Restaurantdelete()
```

delete the menu

Pricedelete

```
public void Pricedelete()
```

delete tuples in the restaurant table that included in the price range

Calorydelete

```
public void Calorydelete()
```

delete tuples in the restaurant table that included in the calories range

(6) Update

A. Class

team2

Class update

java.lang.Object
    team2.update

public class **update**
extends java.lang.Object

class that update some data for request of administrator

Author:

HyoJin Lee

Constructor Summary

Constructors

Constructor and Description
update(java.lang.String city, java.lang.String attraction) constructor
update(java.lang.String name, java.lang.String[] colNames, java.lang.String grades) constructor
update(java.lang.String tableName, java.lang.String menu, int price) constructor

B. Method

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
void	**AttractionUpdate()** update the attraction of city
javax.swing.JTable	**GradesUpdate()** update the grades of each restaurant using transaction
void	**RestaurantUpdate()** update the price of menu

Methods inherited from class java.lang.Object

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

Constructor Detail

update

```
public update(java.lang.String city,
              java.lang.String attraction)
```

constructor

Parameters:

city -

attraction -

update

```
public update(java.lang.String name,
              java.lang.String[] colNames,
              java.lang.String grades)
```

constructor

Parameters:

name -

colNames -

grades -

update

```
public update(java.lang.String tableName,
              java.lang.String menu,
              int price)
```

constructor

Parameters:

tableName -

menu -

price -

Method Detail

RestaurantUpdate

```
public void RestaurantUpdate()
```

update the price of menu

AttractionUpdate

```
public void AttractionUpdate()
```

update the attraction of city

GradesUpdate

```
public javax.swing.JTable GradesUpdate()
                                  throws java.sql.SQLException
```

update the grades of each restaurant using transaction

Returns:

table

Throws:

java.sql.SQLException

(7) retrieve_adm

A. Class

team2

Class retrieve_adm

java.lang.Object
    team2.retrieve_adm

```
public class retrieve_adm
extends java.lang.Object
```

class that retrieve the result for request of administrator

Author:

DoKyung Lee

Constructor Summary

Constructors

Constructor and Description
`retrieve_adm(java.lang.String tableName, java.lang.String[] colNames)` constructor that has String and String array parameters
`retrieve_adm(java.lang.String tableName, java.lang.String index, java.lang.String[] colNames)` constructor that has 2 Strings and String array parameters

B. Method

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
javax.swing.JTable	show() method for administrator that returns a table that is a result of query

Methods inherited from class java.lang.Object

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

Constructor Detail

retrieve_adm

```
public retrieve_adm(java.lang.String tableName,
                    java.lang.String[] colNames)
```

constructor that has String and String array parameters

Parameters:

tableName -

colNames -

retrieve_adm

```
public retrieve_adm(java.lang.String tableName,
                    java.lang.String index,
                    java.lang.String[] colNames)
```

constructor that has 2 Strings and String array parameters

Parameters:

tableName -

index -

colNames -

Method Detail

show

```
public javax.swing.JTable show()
                        throws java.sql.SQLException
```

method for administrator that returns a table that is a result of query

Returns:

Throws:

java.sql.SQLException

(8) retrieve_cus

A. Class

team2

Class retrieve_cus

java.lang.Object
    team2.retrieve_cus

```
public class retrieve_cus
extends java.lang.Object
```

class that retrieve the result for request of customer

Author:

NaGyeong Yeo, Dokyung Lee, HyoJin Lee

Constructor Summary

Constructors

Constructor and Description
retrieve_cus(java.lang.String tableName, int priceMin, int priceMax, java.lang.String[] colNames) constructor to retrieve data by using price range
retrieve_cus(java.lang.String name, java.lang.String[] colNames, int a) constructor that has String, String array, and integer parameters
retrieve_cus(java.lang.String tableName, java.lang.String type, java.lang.String[] colNames) constructor to retrieve data by using type

B. Method

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
javax.swing.JTable	show() retrieve restaurant and menu information by using type / retrieve restaurant and menu information by using price range
javax.swing.JTable	show4() method for customer that returns a table that is a result of query

Methods inherited from class java.lang.Object

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

Constructor Detail

retrieve_cus

```
public retrieve_cus(java.lang.String tableName,
                    java.lang.String type,
                    java.lang.String[] colNames)
```

constructor to retrieve data by using type

Parameters:

tableName –

type –

colNames –

retrieve_cus

```
public retrieve_cus(java.lang.String name,
                    java.lang.String[] colNames,
                    int a)
```

constructor that has String, String array, and integer parameters

Parameters:

name –

colNames –

a –

retrieve_cus

```
public retrieve_cus(java.lang.String tableName,
                    int priceMin,
                    int priceMax,
                    java.lang.String[] colNames)
```

constructor to retrieve data by using price range

Parameters:

tableName -

priceMin -

priceMax -

colNames -

Method Detail

show

```
public javax.swing.JTable show()
```

retrieve restaurant and menu information by using type / retrieve restaurant and menu information by using price range

Returns:

show4

```
public javax.swing.JTable show4()
```

method for customer that returns a table that is a result of query

Returns:

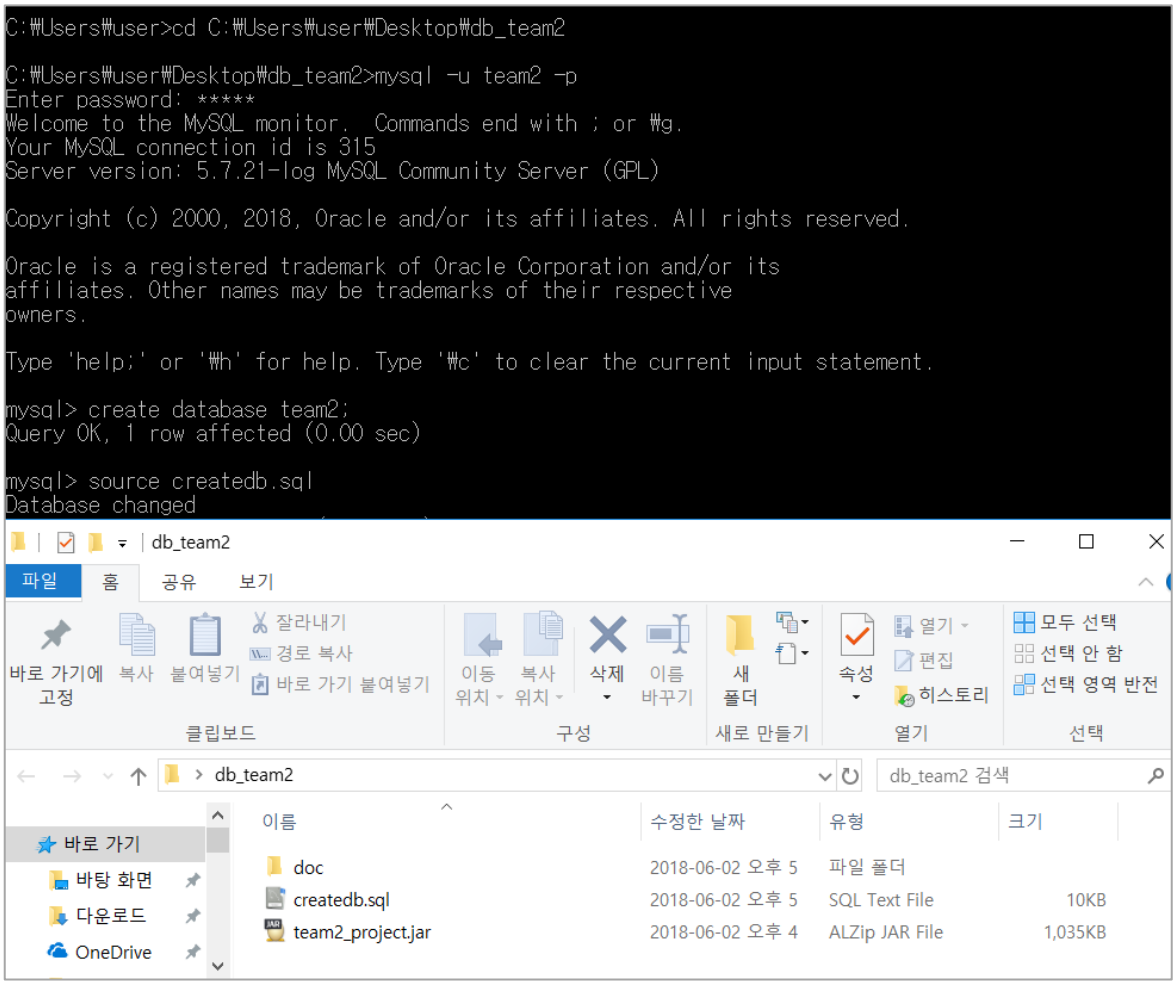
JTable = output of the query

4. Main class name, how to run, connection configuration

(1) Main class name : Open.class

(2) how to run

A. Run MySQL script on command window



B. Exit MySQL and Execute team2_project.jar

(3) connection configuration

A. connection (Class : Open)

```
try {
            Class.forName("com.mysql.jdbc.Driver");
             myConn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/team2?serverTimezone=Asia/Seoul&
useSSL=false", "team2", "team2");
             System.out.println("데이터베이스에 접속했습니다.");

        } catch (Exception e) {
             e.printStackTrace();
             System.out.println(e.getMessage());
        }
```

B. connection close (Class : Open)

```
try {
                        //if myConn is not null and myConn is not closed
                           if( myConn != null && !myConn.isClosed()){
                               myConn.close(); //disconnect the myConn
                              }
                           System.out.println("connection 정상종료");
                 } catch (SQLException e) {
                      // TODO Auto-generated catch block
                      e.printStackTrace();
                 }
```

5. Requirments (1)-(17)

(1)-(6) (createdb.sql)

```
use team2;

/* Hyojin Lee, Dokyung Lee, Youngeun Lee, Nagyeong Yeo*/
create table DBCOURSE_Attraction (
   city varchar(20),
   attraction varchar(60),
   primary key(city)
);
```

```
insert into DBCOURSE_Attraction values("Gangwon-do","Mount Seorak, Jeongdongjin");
insert into DBCOURSE_Attraction values ("Seoul", "Gyeongbokgung Palace, Namsan");
insert into DBCOURSE_Attraction values ("Busan", "Haeundae, Nampo-dong");
insert into DBCOURSE_Attraction values("Jeonju","Jeonju Traditional Korean House, Gyeonggijeon");
insert into DBCOURSE_Attraction values("Jeju","Hallasan, Jeju olle");

/* Hyojin Lee, Dokyung Lee, Youngeun Lee, Nagyeong Yeo*/
create table DBCOURSE_Restaurants (
   name varchar(30),
   city varchar(20) NOT NULL,
   address varchar(80) NOT NULL,
   type varchar(10),
   primary key(name),
   foreign key(city) references DBCOURSE_Attraction(city) on delete cascade
);

insert into DBCOURSE_Restaurants values("Chienrong","Gangwon-do","12, Jungang-ro 68beon-gil, Chuncheon-si, Gangwon-do","Chinese");
insert into DBCOURSE_Restaurants values("Dintaifung","Seoul", "Junggu, Myeongdong 1ga, 59-1, Seoul","Chinese");
insert into DBCOURSE_Restaurants values ("California Pizza Kitchen", "Seoul", "300, Olympic-ro, Songpa-gu, Seoul 05551", "Western");
insert into DBCOURSE_Restaurants values ("Haeundae Smokehouse", "Busan", "24, Haeundaehaebyeon-ro 298beon-gil, Haeundae-gu | 1F Pale de Cz, Busan 48099", "Western");
insert into DBCOURSE_Restaurants values("JamaeGuksu","Jeju","1034-10, Ildoi-dong, Jeju-si, Jeju-do","Korean");
insert into DBCOURSE_Restaurants values("HanKookJib","Jeonju","HanKookJib 2-1, Jeon-dong, Wansan-gu, Jeonju-si, Jeollabuk-do","Korean");
insert into DBCOURSE_Restaurants values ("ChungWoo", "Busan", "8-12, Haeundaehaebyeon-ro 209beon-gil, Haeundae-gu, Busan", "Japanese");

 /* Hyojin Lee, Dokyung Lee */
create table DBCOURSE_Grades (
   name varchar(30),
   grades numeric(2,1) NOT NULL,
   primary key(name),
   foreign key(name) references DBCOURSE_Restaurants(name) on delete cascade
);
```

```
insert into DBCOURSE_Grades values("Chienrong", 4.0);
insert into DBCOURSE_Grades values("Dintaifung", 4.1);
insert into DBCOURSE_Grades values ("California Pizza Kitchen", 3.7);
insert into DBCOURSE_Grades values ("Haeundae Smokehouse", 4.3);
insert into DBCOURSE_Grades values("JamaeGuksu", 4.4);
insert into DBCOURSE_Grades values("HanKookJib", 3.8);
insert into DBCOURSE_Grades values ("ChungWoo", 4.8);

/* Hyojin Lee, Dokyung Lee, Youngeun Lee, Nagyeong Yeo*/
create table DBCOURSE_FoodCategory (
   category varchar(10),
   ingredients varchar(20) NOT NULL,
   primary key(category)
);

insert into DBCOURSE_FoodCategory values("Soup","Meat Broth");
insert into DBCOURSE_FoodCategory values("Porridge","Crabmeat");
insert into DBCOURSE_FoodCategory values("Meat","Beef, Pork, Chicken");
insert into DBCOURSE_FoodCategory values("Rice","Rice");
insert into DBCOURSE_FoodCategory values("Noodles","Wheat flour");
insert into DBCOURSE_FoodCategory values ("Pasta", "Wheat flour");
insert into DBCOURSE_FoodCategory values ("Fries", "Frying Powder");
insert into DBCOURSE_FoodCategory values ("Pizza", "Cheese");
insert into DBCOURSE_FoodCategory values ("Sandwich", "Wheat flour");
insert into DBCOURSE_FoodCategory values ("Seafood", "Fish");
insert into DBCOURSE_FoodCategory values("Pancake","Wheat flour");
insert into DBCOURSE_FoodCategory values("Dumplings","Wheat flour");

 /* Youngeun Lee */
create table DBCOURSE_Rst1 (
   name varchar(30) NOT NULL,
   menu varchar(40),
   price numeric(5,0) NOT NULL,
   calories numeric(5,0),
   category varchar(10),
   primary key(menu),
   foreign key(name) references DBCOURSE_Restaurants(name) on delete cascade,
   foreign key(category) references DBCOURSE_FoodCategory(category) on delete cascade
```

```
);

insert into DBCOURSE_Rst1 values("Chienrong","Soup W/Crabmeat",22000,600,"Porridge");
insert into DBCOURSE_Rst1 values("Chienrong","Sweet&Sour Pork",15000,457,"Meat");
insert into DBCOURSE_Rst1 values("Chienrong","Shrimp with Fride Rice",8000,700,"Rice");
insert into DBCOURSE_Rst1 values("Chienrong","Noodles with black Bean Sauce",6000,864,"Noodles");

/* Youngeun Lee */
create table DBCOURSE_Rst2 (
   name varchar(30) NOT NULL,
   menu varchar(40),
   price numeric(5,0) NOT NULL,
   calories numeric(5,0),
   category varchar(10),
   primary key(menu),
   foreign key(name) references DBCOURSE_Restaurants(name) on delete cascade,
   foreign key(category) references DBCOURSE_FoodCategory(category) on delete cascade
);

insert into DBCOURSE_Rst2 values("Dintaifung","Xiaolongbao",10500,142,"Dumplings");
insert into DBCOURSE_Rst2 values("Dintaifung","Shrimp & Pork Pot Sticker",14000,340,"Dumplings");
insert into DBCOURSE_Rst2 values("Dintaifung","Shirmp & Pork Wonton Soup",10000,459,"Soup");
insert into DBCOURSE_Rst2 values("Dintaifung","Braised Beef Noodles Soup",14000,309,"Noodles");

/* Dokyung Lee */
create table DBCOURSE_Rst3 (
   name varchar(30) NOT NULL,
   menu varchar(40),
   price numeric(5,0) NOT NULL,
   calories numeric(5,0),
   category varchar(10),
   primary key(menu),
   foreign key(name) references DBCOURSE_Restaurants(name) on delete cascade,
   foreign key(category) references DBCOURSE_FoodCategory(category) on delete cascade
);
```

```
insert into DBCOURSE_Rst3 values("California Pizza Kitchen", "Garlic Butter Fries", 9900, 310, "Fries");
insert into DBCOURSE_Rst3 values("California Pizza Kitchen", "Spicy Small Octopus Cream Pasta", 16900, 668, "Pasta");
insert into DBCOURSE_Rst3 values("California Pizza Kitchen", "Fire-Grilled Steak", 36900, 897, "Meat");
insert into DBCOURSE_Rst3 values("California Pizza Kitchen", "The Original BBQ Chicken Pizza", 18900, 455, "Pizza");

/* Dokyung Lee */
create table DBCOURSE_Rst4 (
   name varchar(30) NOT NULL,
   menu varchar(40),
   price numeric(5,0) NOT NULL,
   calories numeric(5,0),
   category varchar(10),
   primary key(menu),
   foreign key(name) references DBCOURSE_Restaurants(name) on delete cascade,
   foreign key(category) references DBCOURSE_FoodCategory(category) on delete cascade
);

insert into DBCOURSE_Rst4 values("Haeundae Smokehouse", "Pulled Pork", 16000, 619, "Meat");
insert into DBCOURSE_Rst4 values("Haeundae Smokehouse", "Sweet Potato Fries", 6000, 452, "Fries");
insert into DBCOURSE_Rst4 values("Haeundae Smokehouse", "Smoke Chicken", 14000, 790, "Meat");
insert into DBCOURSE_Rst4 values("Haeundae Smokehouse", "Panini", 15000, 419, "Sandwich");

/* Hyojin Lee */
create table DBCOURSE_Rst5 (
   name varchar(30) NOT NULL,
   menu varchar(40),
   price numeric(5,0) NOT NULL,
   calories numeric(5,0),
   category varchar(10),
   primary key(menu),
   foreign key(name) references DBCOURSE_Restaurants(name) on delete cascade,
   foreign key(category) references DBCOURSE_FoodCategory(category) on delete cascade
);
```

```
insert into DBCOURSE_Rst5 values("JamaeGuksu","Noodles in anchovy broth",6000,450,"Noodles");
insert into DBCOURSE_Rst5 values("JamaeGuksu","Noodles with pork soup",7000,500,"Noodles");
insert into DBCOURSE_Rst5 values("JamaeGuksu","Noodles mixed with spicy hot sauce",7000,489,"Noodles");
insert into DBCOURSE_Rst5 values("JamaeGuksu","Steamed prok nocks",20000,393,"Meat");

/* Hyojin Lee */
create table DBCOURSE_Rst6 (
   name varchar(30) NOT NULL,
   menu varchar(40),
   price numeric(5,0) NOT NULL,
   calories numeric(5,0),
   category varchar(10),
   primary key(menu),
   foreign key(name) references DBCOURSE_Restaurants(name) on delete cascade,
   foreign key(category) references DBCOURSE_FoodCategory(category) on delete cascade
);

insert into DBCOURSE_Rst6 values("HanKookJib","Korean beef tartare bibimbap",13000,679,"Rice");
insert into DBCOURSE_Rst6 values("HanKookJib","Short rib soup",11000,760,"Soup");
insert into DBCOURSE_Rst6 values("HanKookJib","Stone grill bulgogi",17000,362,"Meat");
insert into DBCOURSE_Rst6 values("HanKookJib","XGreen bean pancake",12000,276,"Pancake");

/* Nagyeong Yeo */
create table DBCOURSE_Rst7 (
   name varchar(30) NOT NULL,
   menu varchar(40),
   price numeric(5,0) NOT NULL,
   calories numeric(5,0),
   category varchar(10),
   primary key(menu),
   foreign key(name) references DBCOURSE_Restaurants(name) on delete cascade,
   foreign key(category) references DBCOURSE_FoodCategory(category) on delete cascade
);
```

```
insert into DBCOURSE_Rst7 values ("ChungWoo", "Stir-fried Beef with been sprouts", 18000, 334.6,
"Meat");
insert into DBCOURSE_Rst7 values ("ChungWoo", "Assorted sushi", 25000, 800, "Seafood");
insert into DBCOURSE_Rst7 values ("ChungWoo", "Grilled Toothfish", 25000, 600, "Seafood");
insert into DBCOURSE_Rst7 values ("ChungWoo", "Fish roe soup", 25000, 180, "Seafood");

/* Hyojin Lee, Dokyung Lee */
create view dbcourse_menu as
 (select * from dbcourse_rst1) union
 (select * from dbcourse_rst2) union
 (select * from dbcourse_rst3) union
 (select * from dbcourse_rst4) union
 (select * from dbcourse_rst5) union
 (select * from dbcourse_rst6) union
 (select * from dbcourse_rst7);

/* Youngeun Lee, Nagyeong Yeo*/
create view dbcourse_type as
  select dbcourse_restaurants.name, dbcourse_menu.menu, dbcourse_menu.price,
dbcourse_restaurants.type
  from dbcourse_restaurants, dbcourse_menu
  where dbcourse_restaurants.name=dbcourse_menu.name;

/* Hyojin Lee, Dokyung Lee, Youngeun Lee, Nagyeong Yeo*/
create index i_name on DBCOURSE_Restaurants(name);
create index i_city on DBCOURSE_Attraction(city);
create index i_category on DBCOURSE_FoodCategory(category);
create index i_price_rst1 on DBCOURSE_rst1(price);
create index i_price_rst2 on DBCOURSE_rst2(price);
create index i_price_rst3 on DBCOURSE_rst3(price);
create index i_price_rst4 on DBCOURSE_rst4(price);
create index i_price_rst5 on DBCOURSE_rst5(price);
create index i_price_rst6 on DBCOURSE_rst6(price);
create index i_price_rst7 on DBCOURSE_rst7(price);
```

Number of tables : 11, Number of columns : 45

Number of records : 59

Constraints : We represent all constraints by using bold character above sql script code.

Number of views : 2 (DBCOURSE_menu, DBCOURSE_type)

Number of indexes : 10 (i_name, i_city, i_category, i_price_rst1~i_price_rst7)

(7) transactions

```
public JTable GradesUpdate() throws SQLException {
        JTable table = new JTable(); // generate new table object
        DefaultTableModel model = new DefaultTableModel(colNames, 0);
        PreparedStatement pstmt1 = null;
        PreparedStatement pstmt2 = null;
        PreparedStatement pstmt3 = null;
        ResultSet myRs = null;
        Statement st = null;
        String sql1 = "select grades from " + tableName + " where name=?";
        String sql2 = "update " + tableName + " set grades=? where name=?";
        String sql3 = "update " + tableName + " set grades=? where name=?";
        String sql4 = "select * from " + tableName;
        try {
            Open.myConn.setAutoCommit(false);
            pstmt1 = Open.myConn.prepareStatement(sql1);
            pstmt2 = Open.myConn.prepareStatement(sql2);
            pstmt3 = Open.myConn.prepareStatement(sql3);
            st = Open.myConn.createStatement();// create a object of Statement
            pstmt1.setString(1, name);
            myRs = pstmt1.executeQuery();// plug in above parameters and execute
            while (myRs.next()) { // approach every record in the table
                grades_pre = myRs.getString(colNames[1]);
            }
            float grades = (Float.parseFloat(grades_pre) + Float.parseFloat(grades_new));
            pstmt2.setFloat(1, grades);
            pstmt2.setString(2, name);
            pstmt2.executeUpdate();
            pstmt3.setFloat(1, grades / 2);
            pstmt3.setString(2, name);
            pstmt3.executeUpdate();

            myRs = st.executeQuery(sql4); // execute query and fetch
            while (myRs.next()) { // approach every record in the table
                model.addRow(new Object[] { myRs.getString(colNames[0]),
myRs.getString(colNames[1]) }); // get all tuples and save to model
            }
            table = new JTable(model);
```

```
            Open.myConn.commit(); // transactions committed
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.out.println(e.getMessage());
            if (Open.myConn != null) {
                try {
                    System.err.print("Transaction is being rolled back");
                    Open.myConn.rollback(); // transactions rolled back
                } catch (SQLException ex) {
                    ex.printStackTrace();
                    System.out.println(ex.getMessage());
                }
            }

        } finally {
            if (pstmt1 != null) {
                pstmt1.close();
            }
            if (pstmt2 != null) {
                pstmt2.close();
            }
            if (pstmt3 != null) {
                pstmt3.close();
            }
            if (st != null) {
                st.close();
            }
            Open.myConn.setAutoCommit(true);// turn on automatic commit
            return table; // return value is table
        }
    }
```

Combine String sql1~sql4(query) by using transaction

(8) queries that use index

A. Class : retrieve_adm, Method : show()

```
sql = "select * from " + tableName + " use index (i_name)";
```

```
sql = "select * from " + tableName + " use index (i_city)";
sql = "select * from " + tableName + " use index (i_category)";
sql = "select * from " + tableName + " use index (" + index + ")";
```

Use i_city(index), i_name(index), i_category(index) and i_price_rst1~i_price_rst7(index) for selecting every column of table

B. Class : retrieve_cus, Method: show4()

```
String sql1 = "select city, attraction from DBCOURSE_Attraction use index (i_city) where city in (select city from DBCOURSE_Restaurants use index (i_name) where name=?)";
```

Use i_city(index) and i_name(index) for selecting city and attraction

(9) queries that use view (View : DBCOURSE_menu, DBCOURSE_type)

Class : retrieve_cus, Method: show()

```
sql = "select name, menu, price, calories, DBCOURSE_menu.category, ingredients from " + tableName + " where DBCOURSE_menu.category=DBCOURSE_foodCategory.category and price >= ? and price <=?";
```

1. Use DBCOURSE_menu(view) for selecting name, menu, price, calories, category and ingredients

```
if (tableName == "DBCOURSE_type") { // retrieve restaurant and menu information by using type
  num = 3; // the number of columns is 3
  sql = "select name, menu, price from " + tableName + " where type=?"; // make query
```

2. Use DBCOURSE_type(view) for selecting name, menu and price

(10) queries with nested queries

Class : retrieve_cus, Method : show4()

```
String sql1 = "select city, attraction from DBCOURSE_Attraction use index (i_city) where city in (select city from DBCOURSE_Restaurants use index (i_name) where name=?)";
```

First, execute selection of city then, execute selection of city and attraction by using previous query

(11) queries with join queries

Class : retrieve_cus, Method : show()

```
sql = "select name, menu, price, calories, DBCOURSE_menu.category, ingredients from " + tableName + " where DBCOURSE_menu.category=DBCOURSE_foodCategory.category and price >= ? and price <=?"
```

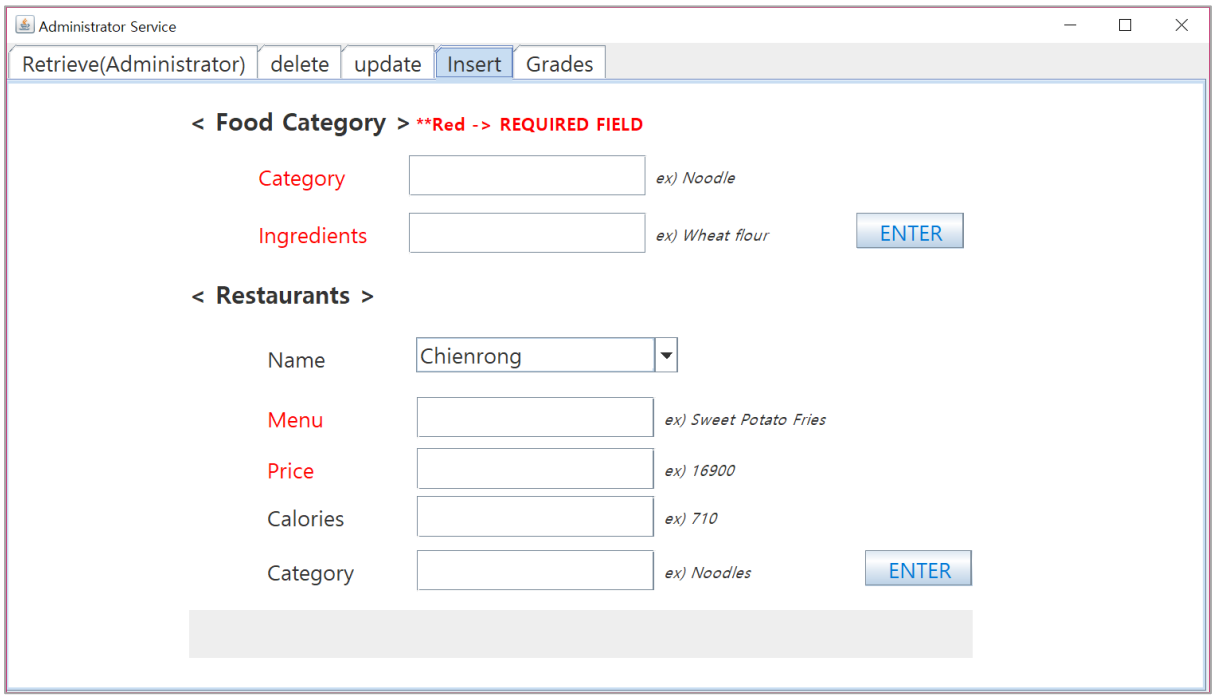
Join DBCOURSE_menu(view) with DBCOURSE_foodCategory by using category column

(12)-(17)

* **queries that are parameterized and dynamically created**

* queries to insert, update, delete and select

A. Insert : service for insert



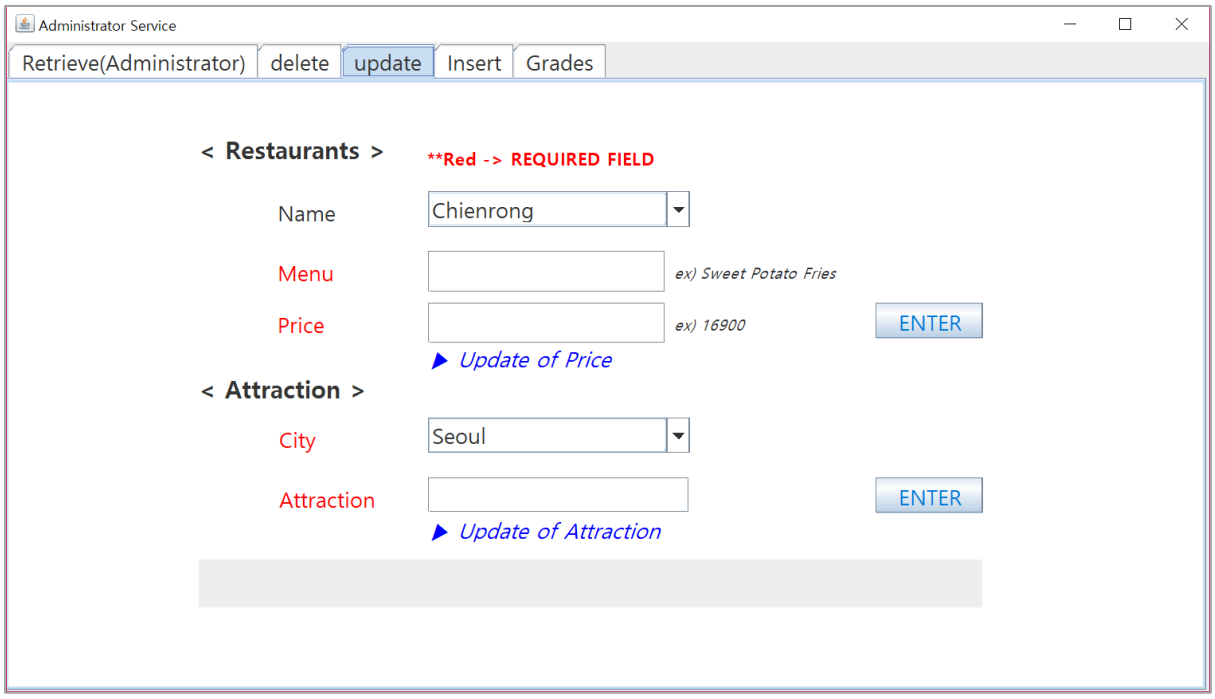

```
package team2;

import java.sql.PreparedStatement;
import java.sql.SQLException;

/**
 * class that insert some data of request for administrator
 *
 * @author HyoJin Lee, NaGyeong Yeo
 *
 */
public class insert {
```

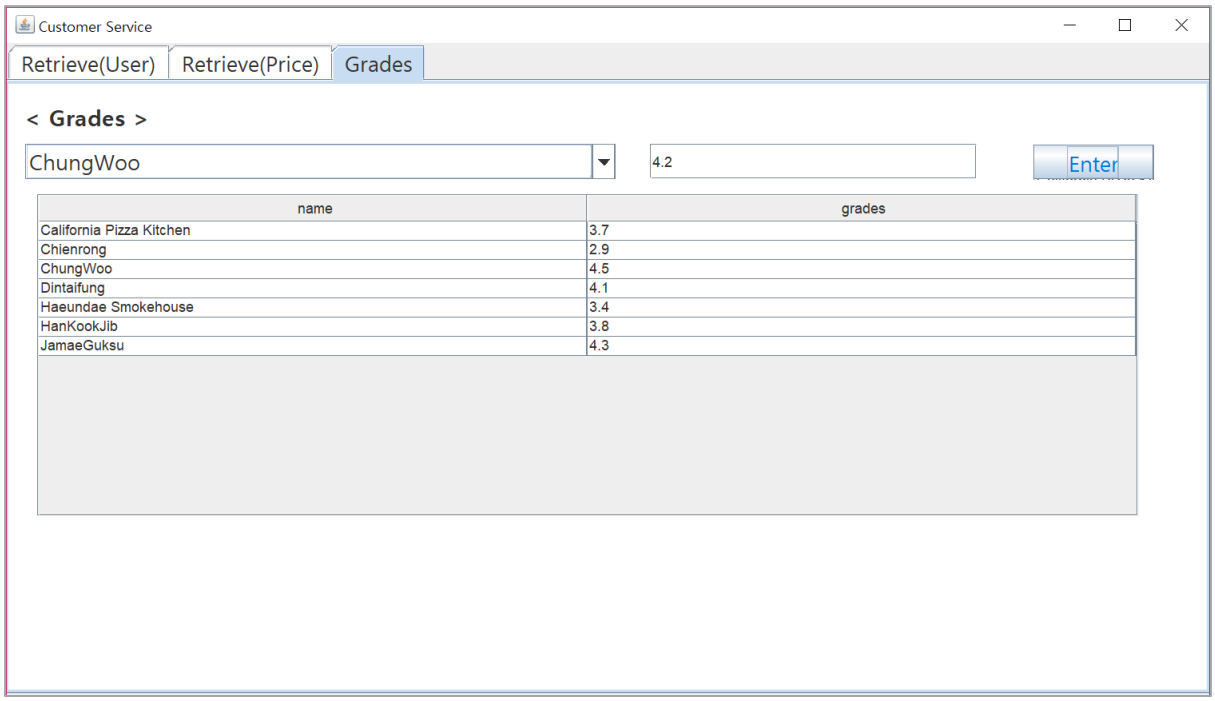
```
    static String tableName = null;
    static String name = null, menu = null, calories = null, category = null, ingredients = null;
    static int price = 0;

    /**
     * constructor to insert category
     *
     * @param category
     * @param ingredients
     */
    public insert(String category, String ingredients) {
        tableName = "DBCOURSE_FoodCategory";
        this.category = category;
        this.ingredients = ingredients;
    }

    /**
     * constructor to insert menu
     *
     * @param tableName
     * @param name
     * @param menu
     * @param price
     * @param calories
     * @param category
     */
    public insert(String tableName, String name, String menu, int price, String calories, String
category) {
        this.tableName = tableName;
        this.name = name;
        this.menu = menu;
        this.price = price;
        this.calories = calories;
        this.category = category;
    }

    /**
     * insert new category(+ingredients) into FoodCategory table
     *
```

```
 * @param category
 * @param ingredients
 */
public void CategoryInsert() {
    PreparedStatement pstmt = null;
    try {
        String sql = "insert into " + tableName + " values (?,?)"; // make query
        // create a object of prepareStatement that have '?' which can
        // be changed with each input of user
        pstmt = Open.myConn.prepareStatement(sql);
        /* input data into parameters */
        pstmt.setString(1, category);
        pstmt.setString(2, ingredients);
        pstmt.executeUpdate(); // execute query
        System.out.println(tableName + " 테이블에 새로운 레코드를 추가했습니다.");
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        if (pstmt != null) {
            try {
                pstmt.close();
            } catch (SQLException e) {
                e.printStackTrace();
                System.out.println(e.getMessage());
            }
        }
    }
}

/**
 * insert new menu(+price, etc) into specific Restaurant table
 *
 * @param tableName
 * @param name
 * @param menu
 * @param price
 * @param calories
 * @param category
 */
```

```
    public void RestaurantInsert() {
        PreparedStatement pstmt = null;
        try {
            String sql = "insert into " + tableName + " values (?,?,?,?,?)"; // make query
            // create a object of prepareStatement that have '?' which can be changed with
            // each input of user
            pstmt = Open.myConn.prepareStatement(sql);
            /* input data into parameters */
            pstmt.setString(1, name);
            pstmt.setString(2, menu);
            pstmt.setInt(3, price);
            pstmt.setString(4, calories);
            pstmt.setString(5, category);
            pstmt.executeUpdate(); // execute query
            System.out.println(tableName + " 테이블에 새로운 레코드를 추가했습니다.");

        } catch (Exception e) {
            e.printStackTrace();
            System.out.println(e.getMessage());
            System.out.println("레코드 추가 실패");
        } finally {
            if (pstmt != null) {
                try {
                    pstmt.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                    System.out.println(e.getMessage());
                }
            }
        }
    }

}
```

B. Update : service for update






```
package team2;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import javax.swing.JTable;
```

```
import javax.swing.table.DefaultTableModel;

/**
 * class that update some data for request of administrator
 * @author HyoJin Lee
 *
 */
public class update {
    static String name = null, menu = null, city = null, attraction = null, colNames[] = null;;
    static int price = 0;
    static String grades_new = null, grades_pre = null;
    static String tableName = null;

    /**
     * constructor
     * @param city
     * @param attraction
     */
    public update(String city, String attraction) {
        tableName = "DBCOURSE_Attraction";
        this.city = city;
        this.attraction = attraction;
    }

    /**
     * constructor
     * @param name
     * @param colNames
     * @param grades
     */
    public update(String name, String[] colNames, String grades) {
        this.name = name;
        this.colNames = colNames;
        this.grades_new = grades;
        tableName = "DBCOURSE_Grades";
    }

    /**
     * constructor
```

```
     * @param tableName
     * @param menu
     * @param price
     */
    public update(String tableName, String menu, int price) {
        this.tableName = tableName;
        this.menu = menu;
        this.price = price;
    }

    /**
     * update the price of menu
     */
    public void RestaurantUpdate() {
        PreparedStatement pstmt=null;
        try {
            String sql = "update " + tableName + " set price=? where menu=?";
            pstmt = Open.myConn.prepareStatement(sql);// create a object of prepareStatement
that have '?' which can
                                                                    // be changed with each input of
user
            pstmt.setInt(1, price); //put the 'price' to first question mark
            pstmt.setString(2, menu); //put the 'menu' to second question mark
            pstmt.executeUpdate(); //plug in above parameters and execute update
        } catch (Exception e) {
            e.printStackTrace();
            System.out.println(e.getMessage());
        }finally {
            if (pstmt != null) {
                try {
                    pstmt.close();
                } catch (SQLException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                    System.out.println(e.getMessage());
                }
            }

        }
```

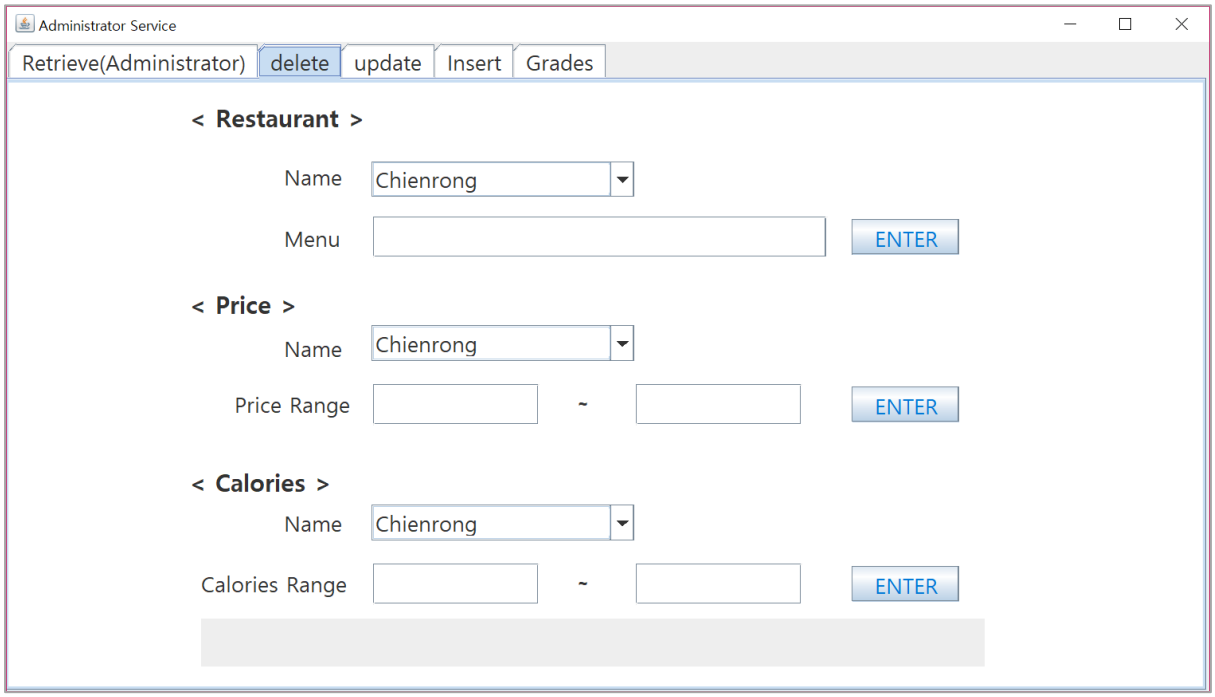
```
    }

    /**
     * update the attraction of city
     */
    public void AttractionUpdate() {
        PreparedStatement pstmt=null;
        try {
            String sql = "update " + tableName + " set attraction=? where city=?";
            pstmt = Open.myConn.prepareStatement(sql);// create a object of prepareStatement
that have '?' which can
                                                                    // be changed with each input of
user
            pstmt.setString(1, attraction); //put the 'attraction' to first question mark
            pstmt.setString(2, city); //put the 'city' to second question mark
            pstmt.executeUpdate();//plug in above parameters and execute update
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.out.println(e.getMessage());
        }finally {
            if (pstmt != null) {
                try {
                    pstmt.close();
                } catch (SQLException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                    System.out.println(e.getMessage());
                }
            }

        }
    }

    /**
     * update the grades of each restaurant using transaction
     * @return table
     * @throws SQLException
     */
```

```
    @SuppressWarnings("finally")
    public JTable GradesUpdate() throws SQLException {
        JTable table = new JTable(); // generate new table object
        DefaultTableModel model = new DefaultTableModel(colNames, 0); //generate new model object
        PreparedStatement pstmt1 = null;
        PreparedStatement pstmt2 = null;
        PreparedStatement pstmt3 = null;
        ResultSet myRs = null;
        Statement st = null;
        String sql1 = "select grades from " + tableName + " where name=?";
        String sql2 = "update " + tableName + " set grades=? where name=?";
        String sql3 = "update " + tableName + " set grades=? where name=?";
        String sql4 = "select * from " + tableName;
        try {
            Open.myConn.setAutoCommit(false); //turn off automatic commit on a connection
            pstmt1 = Open.myConn.prepareStatement(sql1);// create a object of prepareStatement that have '?' which can
                                                        // be changed with each input of user
            pstmt2 = Open.myConn.prepareStatement(sql2);// create a object of prepareStatement that have '?' which can
                                                        // be changed with each input of user
            pstmt3 = Open.myConn.prepareStatement(sql3);// create a object of prepareStatement that have '?' which can
                                                        // be changed with each input of user
            st = Open.myConn.createStatement();// create a object of Statement
            pstmt1.setString(1, name);//put the 'name' to pstmt1's first question mark
            myRs = pstmt1.executeQuery();//plug in above parameters and execute
            while (myRs.next()) { //approach every record in the table
                grades_pre = myRs.getString(colNames[1]); // get the grades value
            }
            float grades = (Float.parseFloat(grades_pre) + Float.parseFloat(grades_new)); //get the sum of previous grades, new grades and save to grades
            pstmt2.setFloat(1, grades);//put the 'grades' to pstmt2's first question mark
            pstmt2.setString(2, name);//put the 'name' to pstmt2's second question mark
            pstmt2.executeUpdate();//plug in above parameters and execute update

            pstmt3.setFloat(1, grades / 2);//put the 'grades/2'(average) to pstmt2's first question mark
```

```
            pstmt3.setString(2, name);//put the 'name' to pstmt2's second question mark
            pstmt3.executeUpdate();//plug in above parameters and execute update

            myRs = st.executeQuery(sql4); //execute query and fetch
            while (myRs.next()) { //approach every record in the table
                model.addRow(new Object[] { myRs.getString(colNames[0]),
myRs.getString(colNames[1]) }); // get all tuples and save to model
            }
            table = new JTable(model);// table becomes a JTable objects with contents of the model
            Open.myConn.commit(); //transactions committed
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.out.println(e.getMessage());
            if (Open.myConn != null) {
                    try {
                        System.err.print("Transaction is being rolled back"); //show the error message
that transactions failed
                        Open.myConn.rollback(); //transactions rolled back
                    } catch(SQLException ex) {
                       ex.printStackTrace();
                     System.out.println(ex.getMessage());
                    }
               }

        } finally {
            if (pstmt1 != null) {
                pstmt1.close();
            }
            if (pstmt2 != null) {
                pstmt2.close();
            }
            if (pstmt3 != null) {
                pstmt3.close();
            }
            if (st != null) {
                st.close();
            }
            Open.myConn.setAutoCommit(true);//turn on automatic commit
```

```
            return table; //return value is table
        }
    }
}
```

C. Delete : service for delete



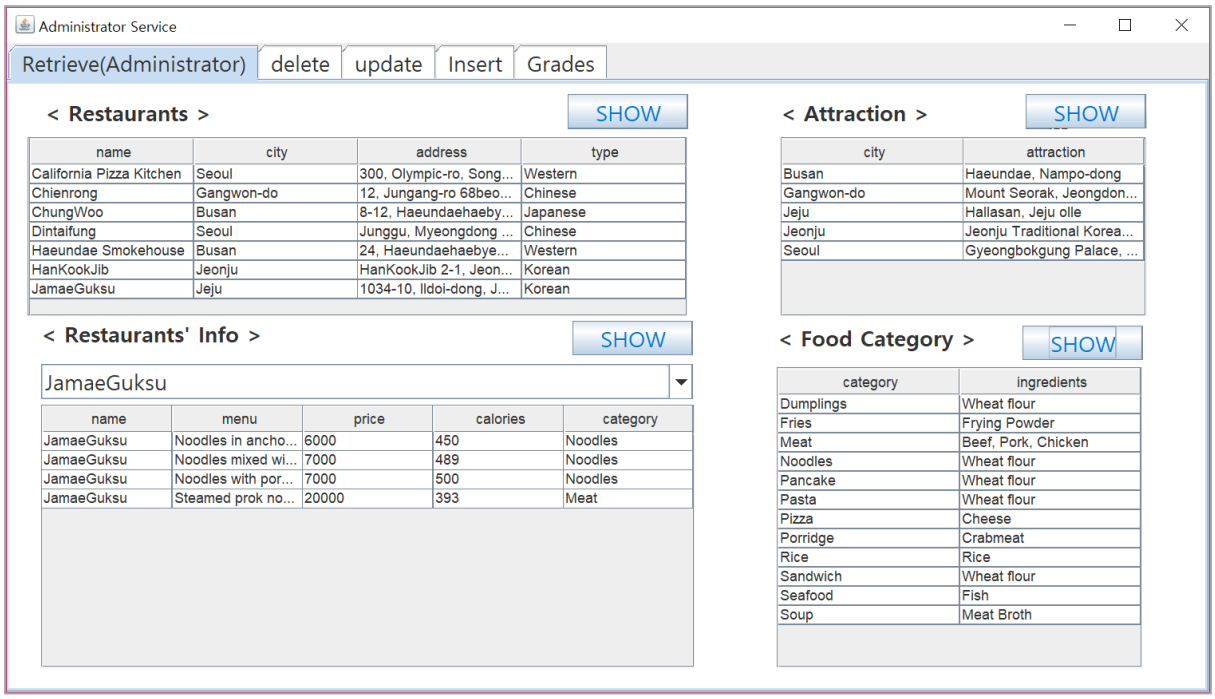

```
package team2;

import java.sql.PreparedStatement;
import java.sql.SQLException;

/**
 * class that delete some data for request of administrator
 * @author HyoJin Lee
 *
 */
public class delete {

    static String menu = null;
    static int price_min = 0, price_max = 0, calory_min = 0, calory_max = 0;
    static String tableName = null;
```

```
/**
 * constructor
 * @param tableName
 * @param menu
 */
public delete(String tableName, String menu) {
    this.tableName = tableName;
    this.menu = menu;
}

/**
 * constructor
 * @param tableName
 * @param min
 * @param max
 */
public delete(String tableName, int min, int max) {
    this.tableName = tableName;

    this.price_min = min;
    this.price_max = max;

    this.calory_min = min;
    this.calory_max = max;
}

/**
 * delete the menu
 */
public void Restaurantdelete() {
    PreparedStatement pstmt=null;
    try {
        String sql = "delete from " + tableName + " where menu=?";
        //create a object of prepareStatement that have '?' which can
        // be changed with each input of user
        pstmt = Open.myConn.prepareStatement(sql);
        pstmt.setString(1, menu);//put the 'menu' to first question mark
        pstmt.executeUpdate();//plug in above parameters and execute update
    } catch (Exception e) {
```

```
            e.printStackTrace();
            System.out.println(e.getMessage());
        }finally {
            if (pstmt != null) {
                try {
                    pstmt.close();
                } catch (SQLException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                    System.out.println(e.getMessage());
                }
            }

        }
    }

    /**
     *   delete tuples in the restaurant table that included in the price range
     */
    public void Pricedelete() {
        PreparedStatement pstmt=null;
        try {
            String sql = "delete from " + tableName + " where price>=? and price<=?";
            //create a object of prepareStatement that have '?' which can
            // be changed with each input of user
            pstmt = Open.myConn.prepareStatement(sql);
            pstmt.setInt(1, price_min); //put the 'price_min' to first question mark
            pstmt.setInt(2, price_max);//put the 'price_max' to second question mark
            pstmt.executeUpdate();//plug in above parameters and execute update
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.out.println(e.getMessage());
        }finally {
            if (pstmt != null) {
                try {
                    pstmt.close();
                } catch (SQLException e) {
                    // TODO Auto-generated catch block
```

```
                e.printStackTrace();
                System.out.println(e.getMessage());
            }
        }

    }
}

/**
 * delete tuples in the restaurant table that included in the calories range
 */
public void Calorydelete() {
    PreparedStatement pstmt=null;
    try {
        String sql = "delete from " + tableName + " where calories>? and calories<?";
        //create a object of prepareStatement that have '?' which can
        // be changed with each input of user
        pstmt = Open.myConn.prepareStatement(sql);
        pstmt.setInt(1, calory_min); //put the 'calory_min' to first question mark
        pstmt.setInt(2, calory_max);//put the 'calory_max' to second question mark
        pstmt.executeUpdate();//plug in above parameters and execute update
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        System.out.println(e.getMessage());
    }finally {
        if (pstmt != null) {
            try {
                pstmt.close();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
                System.out.println(e.getMessage());
            }
        }

    }
}
}
```

D-1. Select(Retrieve) – Administrator : service for retrieve



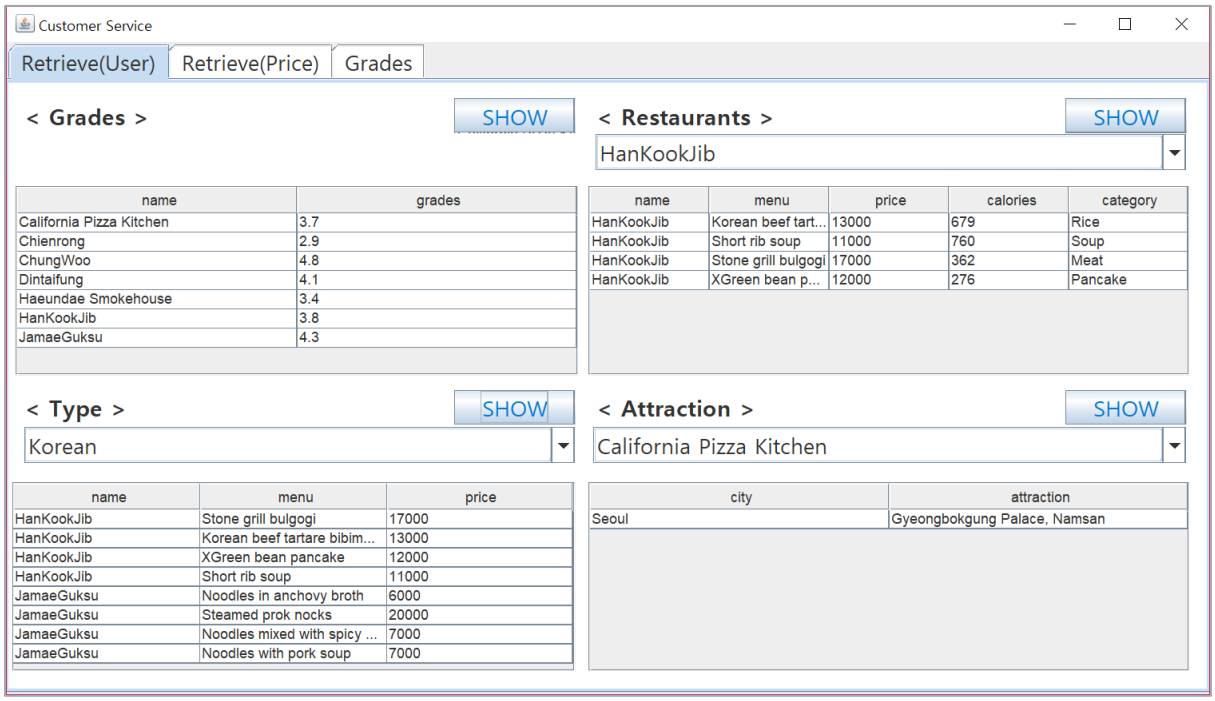

```
package team2;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;

/**
 * class that retrieve the result for request of administrator
 * @author DoKyung Lee
 *
 */
public class retrieve_adm {

    static String colNames[] = null;
    static String tableName = null;
    static String index = null;
```

```
    DefaultTableModel model = null;

    /**
     * constructor that has String and String array parameters
     * @param tableName
     * @param colNames
     */
    public retrieve_adm(String tableName, String[] colNames) {
        this.tableName = tableName;
        this.colNames = colNames;
    }

    /**
     * constructor that has 2 Strings and String array parameters
     * @param tableName
     * @param index
     * @param colNames
     */
    public retrieve_adm(String tableName, String index, String[] colNames) {
        this.tableName = tableName;
        this.index = index;
        this.colNames = colNames;

    }

    /**
     * method for administrator that returns a table that is a result of query
     * @return
     * @throws SQLException
     */
    public JTable show() throws SQLException {
        JTable table = new JTable(); // 'table' is a object of JTable
        model = new DefaultTableModel(colNames, 0); // Constructs a DefaultTableModel with
columns that are elements in columnNames and the number of rows is '0'
        Statement st = null;

        try {
            int num = 0; // the number of columns that have to be shown
            st = Open.myConn.createStatement(); //create a object of Statement
```

```
        String sql = null;
        ResultSet myRs = null; //create a object of ResultSet

        // condition statement about 'tableName' shows the whole table
        if (tableName == "DBCOURSE_Restaurants") { // if 'tableName' is same as "DBCOURSE_Restaurants"
            num = 4; // the number of columns that have to be shown is 4
            sql = "select * from " + tableName + " use index (i_name)"; // use index in query
            myRs = st.executeQuery(sql); // execute query that is same as 'sql' and 'myRs' becomes that result
        } else if (tableName == "DBCOURSE_Attraction") { // second option and comments are same with 'if' part
            num = 2;
            sql = "select * from " + tableName + " use index (i_city)";
            myRs = st.executeQuery(sql);
        } else if (tableName == "DBCOURSE_FoodCategory") { // third option and comments are same with 'if' part
            num = 2;
            sql = "select * from " + tableName + " use index (i_category)";
            myRs = st.executeQuery(sql);
        } else if (tableName == "DBCOURSE_Grades") { // fourth option and comments are same with 'if' part
            num = 2;
            sql = "select * from " + tableName;
            myRs = st.executeQuery(sql);
        } else { // fifth option and comments are same with 'if' part
            num = 5;
            sql = "select * from " + tableName + " use index (" + index + ")";
            myRs = st.executeQuery(sql);
        }

        while (myRs.next()) { // read ResultSet 'myRs' line by line
            switch (num) { // 3 cases are divided depending on value of 'num'
            case 4: // first case - the result data of query occupies domain of column in the model
                model.addRow(new Object[] { myRs.getString(colNames[0]), myRs.getString(colNames[1]),
                        myRs.getString(colNames[2]), myRs.getString(colNames[3]) }); // domain of 'colNames' becomes 'Object' array and also rows of 'tableName'
```

```
                    break;
                case 2: // second case
                    model.addRow(new Object[] { myRs.getString(colNames[0]),
myRs.getString(colNames[1]) });
                    break;
                case 5: // third case
                    model.addRow(new Object[] { myRs.getString(colNames[0]),
myRs.getString(colNames[1]),
                        myRs.getString(colNames[2]), myRs.getString(colNames[3]),
myRs.getString(colNames[4]) });
                    break;
                }
            }
            table = new JTable(model); // table becomes a JTable objects with contents of the model

        } catch (SQLException e) {   //catch errors
            e.printStackTrace();
            System.out.println(e.getMessage());
        } finally {
                if (st != null) {
                    st.close(); // close the statement
                }
                return table;   //return table of the result
            }
    }
}
```

D-2. Select(Retrieve) – Customer : service for retrieve

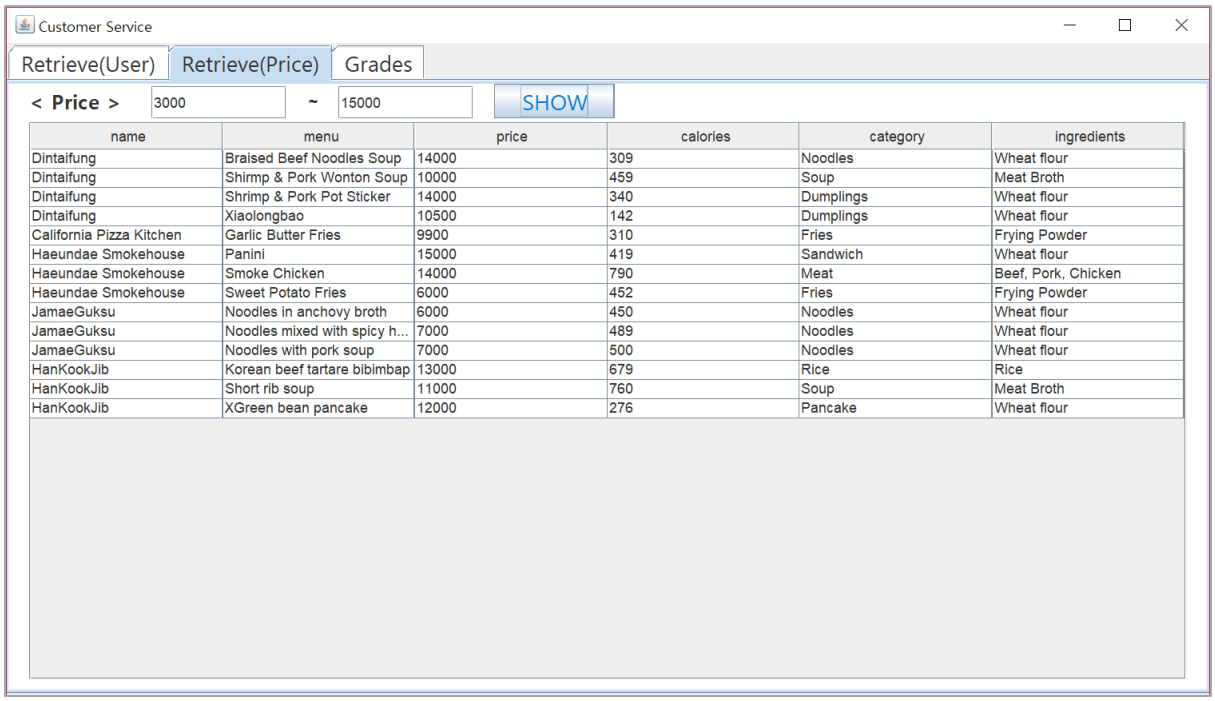


name	grades
California Pizza Kitchen	3.7
Chienrong	2.9
ChungWoo	4.8
Dintaifung	4.1
Haeundae Smokehouse	3.4
HanKookJib	3.8
JamaeGuksu	4.3

name	menu	price	calories	category
HanKookJib	Korean beef tart...	13000	679	Rice
HanKookJib	Short rib soup	11000	760	Soup
HanKookJib	Stone grill bulgogi	17000	362	Meat
HanKookJib	XGreen bean p...	12000	276	Pancake

name	menu	price
HanKookJib	Stone grill bulgogi	17000
HanKookJib	Korean beef tartare bibim...	13000
HanKookJib	XGreen bean pancake	12000
HanKookJib	Short rib soup	11000
JamaeGuksu	Noodles in anchovy broth	6000
JamaeGuksu	Steamed prok nocks	20000
JamaeGuksu	Noodles mixed with spicy ...	7000
JamaeGuksu	Noodles with pork soup	7000

city	attraction
Seoul	Gyeongbokgung Palace, Namsan



name	menu	price	calories	category	ingredients
Dintaifung	Braised Beef Noodles Soup	14000	309	Noodles	Wheat flour
Dintaifung	Shrimp & Pork Wonton Soup	10000	459	Soup	Meat Broth
Dintaifung	Shrimp & Pork Pot Sticker	14000	340	Dumplings	Wheat flour
Dintaifung	Xiaolongbao	10500	142	Dumplings	Wheat flour
California Pizza Kitchen	Garlic Butter Fries	9900	310	Fries	Frying Powder
Haeundae Smokehouse	Panini	15000	419	Sandwich	Wheat flour
Haeundae Smokehouse	Smoke Chicken	14000	790	Meat	Beef, Pork, Chicken
Haeundae Smokehouse	Sweet Potato Fries	6000	452	Fries	Frying Powder
JamaeGuksu	Noodles in anchovy broth	6000	450	Noodles	Wheat flour
JamaeGuksu	Noodles mixed with spicy h...	7000	489	Noodles	Wheat flour
JamaeGuksu	Noodles with pork soup	7000	500	Noodles	Wheat flour
HanKookJib	Korean beef tartare bibimbap	13000	679	Rice	Rice
HanKookJib	Short rib soup	11000	760	Soup	Meat Broth
HanKookJib	XGreen bean pancake	12000	276	Pancake	Wheat flour

```
package team2;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import javax.swing.JTable;
```

```
import javax.swing.table.DefaultTableModel;

/**
 * class that retrieve the result for request of customer
 *
 * @author NaGyeong Yeo, Dokyung Lee, HyoJin Lee
 *
 */
public class retrieve_cus {
    static String colNames[] = null;
    static String tableName = null, type = null, name = null;
    static int priceMin, priceMax;

    /**
     * constructor to retrieve data by using type
     *
     * @param tableName
     * @param type
     * @param colNames
     */
    public retrieve_cus(String tableName, String type, String[] colNames) {
        this.tableName = tableName;
        this.type = type;
        this.colNames = colNames;
    }

    /**
     * constructor that has String, String array, and integer parameters
     *
     * @param name
     * @param colNames
     * @param a
     */
    public retrieve_cus(String name, String[] colNames, int a) {
        this.name = name;
        this.colNames = colNames;
    }

    /**
```

```
     * constructor to retrieve data by using price range
     *
     * @param tableName
     * @param priceMin
     * @param priceMax
     * @param colNames
     */
    public retrieve_cus(String tableName, int priceMin, int priceMax, String[] colNames) {
        this.tableName = tableName;
        this.priceMin = priceMin;
        this.priceMax = priceMax;
        this.colNames = colNames;
    }

    /**
     * retrieve restaurant and menu information by using type / retrieve restaurant
     * and menu information by using price range
     *
     * @return
     */
    public JTable show() {
        JTable table = new JTable();
        DefaultTableModel model = new DefaultTableModel(colNames, 0);
        PreparedStatement pstmt = null;
        try {
            int num = 0;
            String sql;
            if (tableName == "DBCOURSE_type") {
                num = 3; // the number of columns is 3
                sql = "select name, menu, price from " + tableName + " where type=?";
                pstmt = Open.myConn.prepareStatement(sql);
                /* input data into parameter */
                pstmt.setString(1, type);
            } else if (tableName == "DBCOURSE_menu, DBCOURSE_foodCategory")
                num = 6; // the number of columns is 6
                sql = "select name, menu, price, calories, DBCOURSE_menu.category,
ingredients from " + tableName + " where
DBCOURSE_menu.category=DBCOURSE_foodCategory.category and price >= ? and price
<=?";
```

```
            pstmt = Open.myConn.prepareStatement(sql);
            /* input data into parameters */
            pstmt.setInt(1, priceMin);
            pstmt.setInt(2, priceMax);
        }

        ResultSet myRs = pstmt.executeQuery(); // execute query

        /* make table according to the number of columns */
        while (myRs.next()) {
            switch (num) {
            case 2:
                model.addRow(new Object[] { myRs.getString(colNames[0]),
myRs.getString(colNames[1]) });
                break;
            case 3:
                model.addRow(new Object[] { myRs.getString(colNames[0]),
myRs.getString(colNames[1]),
                        myRs.getString(colNames[2]) });
                break;
            case 4:
                model.addRow(new Object[] { myRs.getString(colNames[0]),
myRs.getString(colNames[1]),
                        myRs.getString(colNames[2]), myRs.getString(colNames[3]) });
                break;
            case 5:
                model.addRow(new Object[] { myRs.getString(colNames[0]),
myRs.getString(colNames[1]),
                        myRs.getString(colNames[2]), myRs.getString(colNames[3]),
myRs.getString(colNames[4]) });
                break;
            case 6:
                model.addRow(new Object[] { myRs.getString(colNames[0]),
myRs.getString(colNames[1]),
                        myRs.getString(colNames[2]), myRs.getString(colNames[3]),
myRs.getString(colNames[4]),
                        myRs.getString(colNames[5]) });
                break;
            }
```

```
            }
            table = new JTable(model); // table becomes a JTable objects with contents of the model

        } catch (SQLException e) { // catch errors
            e.printStackTrace();
            System.out.println(e.getMessage());
        } finally {
            if (pstmt != null) {
                try {
                    pstmt.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                    System.out.println(e.getMessage());
                }
            }
        }

        return table; // return table of the result
    }

    /**
     * method for customer that returns a table that is a result of query
     *
     * @return JTable = output of the query
     */
    public JTable show4() {

        JTable table = new JTable(); // 'table' is a object of JTable
        DefaultTableModel model = new DefaultTableModel(colNames, 0); // Constructs a DefaultTableModel with columns
        // that are elements in columnNames and the
        // number of rows is '0'
        PreparedStatement pstmt1 = null;

        try {
            // nested query is used in this query
            String sql1 = "select city, attraction from DBCOURSE_Attraction use index (i_city) where city in (select city from DBCOURSE_Restaurants use index (i_name) where name=?)";
```

```
            // create a object of prepareStatement that have '?' which can
            // be changed with each input of user
            pstmt1 = Open.myConn.prepareStatement(sql1);
            pstmt1.setString(1, name); // first '?' is the same with value of 'name'

            ResultSet myRs = pstmt1.executeQuery(); // execute query in 'pstmt1' and 'myRs' becomes that result

            while (myRs.next()) { // read ResultSet 'myRs' line by line
                // domain of 'colNames' becomes 'Object' array and also rows of 'tableName'
                model.addRow(new Object[] { myRs.getString(colNames[0]), myRs.getString(colNames[1]) });
            }
            table = new JTable(model); // table becomes a JTable objects with contents of the model

        } catch (SQLException e) { // catch errors
            e.printStackTrace();
            System.out.println(e.getMessage());
        } finally {
            if (pstmt1 != null) {
                try {
                    pstmt1.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                    System.out.println(e.getMessage());
                }
            }
        }

        return table; // return table of the result
    }
}
```

6. Team responsibility assignments

Member name	SQL	Java Code	Report (classify by items)	Presentation	Demo video
Dokyung Lee	-Create table scripts -Insert table scripts -ER Diagram	-Class Open, Adm, Cus, retrieve_adm, retrieve_cus -GUI -Testing and debugging -Javadoc	-item: 1, 3, 4, 5, 7, 8	-Presentation -Make final script	-Make demo video
Youngeun Lee	-Create table scripts -Insert table scripts	-Class Open, Adm, Cus -Total GUI -Testing and debugging -Create jar file	-item: 3, 5, 7, 8	-Create ppt -Make basic script	-Make demo video
Nagyeong Yeo	-Create table scripts -Insert table scripts -Make final sql script	-Class Adm, Cus, insert, retrieve_cus -Testing and debugging	-item: 3, 4, 5, 6, 7, 8 -Make final report	-Presentation -Make final script	
Hyojin Lee	-Create table scripts -Insert table scripts -Make final sql script	-Class Open, Adm, Cus, insert, delete, retrieve_cus, update -Testing and debugging	-item: 2, 3, 4, 5, 6, 7, 8 -Make final report	-Create ppt -Make basic script	

7. SQL scripts

A. created.sql

```
use team2;

/* Hyojin Lee, Dokyung Lee, Youngeun Lee, Nagyeong Yeo*/
create table DBCOURSE_Attraction (
   city varchar(20),
   attraction varchar(60),
   primary key(city)
);

insert into DBCOURSE_Attraction values("Gangwon-do","Mount Seorak, Jeongdongjin");
insert into DBCOURSE_Attraction values ("Seoul", "Gyeongbokgung Palace, Namsan");
insert into DBCOURSE_Attraction values ("Busan", "Haeundae, Nampo-dong");
insert into DBCOURSE_Attraction values("Jeonju","Jeonju Traditional Korean House, Gyeonggijeon");
insert into DBCOURSE_Attraction values("Jeju","Hallasan, Jeju olle");

/* Hyojin Lee, Dokyung Lee, Youngeun Lee, Nagyeong Yeo*/
create table DBCOURSE_Restaurants (
   name varchar(30),
   city varchar(20) NOT NULL,
   address varchar(80) NOT NULL,
   type varchar(10),
   primary key(name),
   foreign key(city) references DBCOURSE_Attraction(city) on delete cascade
);

insert into DBCOURSE_Restaurants values("Chienrong","Gangwon-do","12, Jungang-ro 68beon-gil, Chuncheon-si, Gangwon-do","Chinese");
insert into DBCOURSE_Restaurants values("Dintaifung","Seoul", "Junggu, Myeongdong 1ga, 59-1, Seoul","Chinese");
insert into DBCOURSE_Restaurants values ("California Pizza Kitchen", "Seoul", "300, Olympic-ro, Songpa-gu, Seoul 05551", "Western");
insert into DBCOURSE_Restaurants values ("Haeundae Smokehouse", "Busan", "24, Haeundaehaebyeon-ro 298beon-gil, Haeundae-gu | 1F Pale de Cz, Busan 48099", "Western");
insert into DBCOURSE_Restaurants values("JamaeGuksu","Jeju","1034-10, Ildoi-dong, Jeju-si, Jeju-do","Korean");
```

```
insert into DBCOURSE_Restaurants values("HanKookJib","Jeonju","HanKookJib 2-1, Jeon-dong, Wansan-gu, Jeonju-si, Jeollabuk-do","Korean");
insert into DBCOURSE_Restaurants values ("ChungWoo", "Busan", "8-12, Haeundaehaebyeon-ro 209beon-gil, Haeundae-gu, Busan", "Japanese");

 /* Hyojin Lee, Dokyung Lee */
create table DBCOURSE_Grades (
   name varchar(30),
   grades numeric(2,1) NOT NULL,
   primary key(name),
   foreign key(name) references DBCOURSE_Restaurants(name) on delete cascade
);

insert into DBCOURSE_Grades values("Chienrong", 4.0);
insert into DBCOURSE_Grades values("Dintaifung", 4.1);
insert into DBCOURSE_Grades values ("California Pizza Kitchen", 3.7);
insert into DBCOURSE_Grades values ("Haeundae Smokehouse", 4.3);
insert into DBCOURSE_Grades values("JamaeGuksu", 4.4);
insert into DBCOURSE_Grades values("HanKookJib", 3.8);
insert into DBCOURSE_Grades values ("ChungWoo", 4.8);

/* Hyojin Lee, Dokyung Lee, Youngeun Lee, Nagyeong Yeo*/
create table DBCOURSE_FoodCategory (
   category varchar(10),
   ingredients varchar(20) NOT NULL,
   primary key(category)
);

insert into DBCOURSE_FoodCategory values("Soup","Meat Broth");
insert into DBCOURSE_FoodCategory values("Porridge","Crabmeat");
insert into DBCOURSE_FoodCategory values("Meat","Beef, Pork, Chicken");
insert into DBCOURSE_FoodCategory values("Rice","Rice");
insert into DBCOURSE_FoodCategory values("Noodles","Wheat flour");
insert into DBCOURSE_FoodCategory values ("Pasta", "Wheat flour");
insert into DBCOURSE_FoodCategory values ("Fries", "Frying Powder");
insert into DBCOURSE_FoodCategory values ("Pizza", "Cheese");
insert into DBCOURSE_FoodCategory values ("Sandwich", "Wheat flour");
insert into DBCOURSE_FoodCategory values ("Seafood", "Fish");
insert into DBCOURSE_FoodCategory values("Pancake","Wheat flour");
```

```
insert into DBCOURSE_FoodCategory values("Dumplings","Wheat flour");

 /* Youngeun Lee */
create table DBCOURSE_Rst1 (
   name varchar(30) NOT NULL,
   menu varchar(40),
   price numeric(5,0) NOT NULL,
   calories numeric(5,0),
   category varchar(10),
   primary key(menu),
   foreign key(name) references DBCOURSE_Restaurants(name) on delete cascade,
   foreign key(category) references DBCOURSE_FoodCategory(category) on delete cascade
);

insert into DBCOURSE_Rst1 values("Chienrong","Soup W/Crabmeat",22000,600,"Porridge");
insert into DBCOURSE_Rst1 values("Chienrong","Sweet&Sour Pork",15000,457,"Meat");
insert into DBCOURSE_Rst1 values("Chienrong","Shrimp with Fride Rice",8000,700,"Rice");
insert into DBCOURSE_Rst1 values("Chienrong","Noodles with black Bean Sauce",6000,864,"Noodles");

/* Youngeun Lee */
create table DBCOURSE_Rst2 (
   name varchar(30) NOT NULL,
   menu varchar(40),
   price numeric(5,0) NOT NULL ,
   calories numeric(5,0),
   category varchar(10),
   primary key(menu),
   foreign key(name) references DBCOURSE_Restaurants(name) on delete cascade,
   foreign key(category) references DBCOURSE_FoodCategory(category) on delete cascade
);

insert into DBCOURSE_Rst2 values("Dintaifung","Xiaolongbao",10500,142,"Dumplings");
insert into DBCOURSE_Rst2 values("Dintaifung","Shrimp & Pork Pot Sticker",14000,340,"Dumplings");
insert into DBCOURSE_Rst2 values("Dintaifung","Shirmp & Pork Wonton Soup",10000,459,"Soup");
insert into DBCOURSE_Rst2 values("Dintaifung","Braised Beef Noodles Soup",14000,309,"Noodles");
```

```
/* Dokyung Lee */
create table DBCOURSE_Rst3 (
   name varchar(30) NOT NULL,
   menu varchar(40),
   price numeric(5,0) NOT NULL,
   calories numeric(5,0),
   category varchar(10),
   primary key(menu),
   foreign key(name) references DBCOURSE_Restaurants(name) on delete cascade,
   foreign key(category) references DBCOURSE_FoodCategory(category) on delete cascade
);

insert into DBCOURSE_Rst3 values("California Pizza Kitchen", "Garlic Butter Fries", 9900, 310, "Fries");
insert into DBCOURSE_Rst3 values("California Pizza Kitchen", "Spicy Small Octopus Cream Pasta", 16900, 668, "Pasta");
insert into DBCOURSE_Rst3 values("California Pizza Kitchen", "Fire-Grilled Steak", 36900, 897, "Meat");
insert into DBCOURSE_Rst3 values("California Pizza Kitchen", "The Original BBQ Chicken Pizza", 18900, 455, "Pizza");

/* Dokyung Lee */
create table DBCOURSE_Rst4 (
   name varchar(30) NOT NULL,
   menu varchar(40),
   price numeric(5,0) NOT NULL,
   calories numeric(5,0),
   category varchar(10),
   primary key(menu),
   foreign key(name) references DBCOURSE_Restaurants(name) on delete cascade,
   foreign key(category) references DBCOURSE_FoodCategory(category) on delete cascade
);

insert into DBCOURSE_Rst4 values("Haeundae Smokehouse", "Pulled Pork", 16000, 619, "Meat");
insert into DBCOURSE_Rst4 values("Haeundae Smokehouse", "Sweet Potato Fries", 6000, 452, "Fries");
insert into DBCOURSE_Rst4 values("Haeundae Smokehouse", "Smoke Chicken", 14000, 790, "Meat");
insert into DBCOURSE_Rst4 values("Haeundae Smokehouse", "Panini", 15000, 419, "Sandwich");
```

```
/* Hyojin Lee */
create table DBCOURSE_Rst5 (
   name varchar(30) NOT NULL,
   menu varchar(40),
   price numeric(5,0) NOT NULL,
   calories numeric(5,0),
   category varchar(10),
   primary key(menu),
   foreign key(name) references DBCOURSE_Restaurants(name) on delete cascade,
   foreign key(category) references DBCOURSE_FoodCategory(category) on delete cascade
);

insert into DBCOURSE_Rst5 values("JamaeGuksu","Noodles in anchovy broth",6000,450,"Noodles");
insert into DBCOURSE_Rst5 values("JamaeGuksu","Noodles with pork soup",7000,500,"Noodles");
insert into DBCOURSE_Rst5 values("JamaeGuksu","Noodles mixed with spicy hot sauce",7000,489,"Noodles");
insert into DBCOURSE_Rst5 values("JamaeGuksu","Steamed prok nocks",20000,393,"Meat");

/* Hyojin Lee */
create table DBCOURSE_Rst6 (
   name varchar(30) NOT NULL,
   menu varchar(40),
   price numeric(5,0) NOT NULL,
   calories numeric(5,0),
   category varchar(10),
   primary key(menu),
   foreign key(name) references DBCOURSE_Restaurants(name) on delete cascade,
   foreign key(category) references DBCOURSE_FoodCategory(category) on delete cascade
);

insert into DBCOURSE_Rst6 values("HanKookJib","Korean beef tartare bibimbap",13000,679,"Rice");
insert into DBCOURSE_Rst6 values("HanKookJib","Short rib soup",11000,760,"Soup");
insert into DBCOURSE_Rst6 values("HanKookJib","Stone grill bulgogi",17000,362,"Meat");
insert into DBCOURSE_Rst6 values("HanKookJib","XGreen bean pancake",12000,276,"Pancake");

/* Nagyeong Yeo */
```

```
create table DBCOURSE_Rst7 (
   name varchar(30) NOT NULL,
   menu varchar(40),
   price numeric(5,0) NOT NULL,
   calories numeric(5,0),
   category varchar(10),
   primary key(menu),
   foreign key(name) references DBCOURSE_Restaurants(name) on delete cascade,
   foreign key(category) references DBCOURSE_FoodCategory(category) on delete cascade
);

insert into DBCOURSE_Rst7 values ("ChungWoo", "Stir-fried Beef with been sprouts", 18000, 334.6,
"Meat");
insert into DBCOURSE_Rst7 values ("ChungWoo", "Assorted sushi", 25000, 800, "Seafood");
insert into DBCOURSE_Rst7 values ("ChungWoo", "Grilled Toothfish", 25000, 600, "Seafood");
insert into DBCOURSE_Rst7 values ("ChungWoo", "Fish roe soup", 25000, 180, "Seafood");

/* Hyojin Lee, Dokyung Lee */
create view dbcourse_menu as
  (select * from dbcourse_rst1) union
  (select * from dbcourse_rst2) union
  (select * from dbcourse_rst3) union
  (select * from dbcourse_rst4) union
  (select * from dbcourse_rst5) union
  (select * from dbcourse_rst6) union
  (select * from dbcourse_rst7);

/* Youngeun Lee, Nagyeong Yeo*/
create view dbcourse_type as
   select dbcourse_restaurants.name, dbcourse_menu.menu, dbcourse_menu.price,
dbcourse_restaurants.type
   from dbcourse_restaurants, dbcourse_menu
   where dbcourse_restaurants.name=dbcourse_menu.name;

/* Hyojin Lee, Dokyung Lee, Youngeun Lee, Nagyeong Yeo*/
create index i_name on DBCOURSE_Restaurants(name);
create index i_city on DBCOURSE_Attraction(city);
create index i_category on DBCOURSE_FoodCategory(category);
create index i_price_rst1 on DBCOURSE_rst1(price);
```

```
create index i_price_rst2 on DBCOURSE_rst2(price);
create index i_price_rst3 on DBCOURSE_rst3(price);
create index i_price_rst4 on DBCOURSE_rst4(price);
create index i_price_rst5 on DBCOURSE_rst5(price);
create index i_price_rst6 on DBCOURSE_rst6(price);
create index i_price_rst7 on DBCOURSE_rst7(price);
```

B. dropdb.sql

```
/* Hyojin Lee, Nagyeong Yeo*/
use team2;

drop table DBCOURSE_Rst1;
drop table DBCOURSE_Rst2;
drop table DBCOURSE_Rst3;
drop table DBCOURSE_Rst4;
drop table DBCOURSE_Rst5;
drop table DBCOURSE_Rst6;
drop table DBCOURSE_Rst7;
drop table DBCOURSE_FoodCategory;
drop view DBCOURSE_menu;
drop view dbcourse_type;
drop table DBCOURSE_Grades;
drop table DBCOURSE_Restaurants;
drop table DBCOURSE_Attraction;



drop database team2;
```

8. Java codes

A. Open

```
package team2;

import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JButton;
```

```
import java.awt.BorderLayout;
import javax.swing.JLabel;
import java.awt.Font;
import java.awt.event.ActionListener;
import java.sql.*;
import java.awt.event.ActionEvent;
import java.awt.Color;
import javax.swing.ImageIcon;
/**
 * class that is a window for selection
 * @author YoungEun Lee, Dokyung Lee, HyoJin Lee
 *
 */
public class Open {
    static Connection myConn;
    private JFrame frmOpen;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            myConn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/team2?serverTimezone=Asia/Seoul&
useSSL=false", "team2", "team2");
            System.out.println("데이터베이스에 접속했습니다.");

        } catch (Exception e) {
            e.printStackTrace();
            System.out.println(e.getMessage());
        }
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Open window = new Open();
                    window.frmOpen.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
```

```
            }
        }
    });
}

/**
 * initialize the application.
 */
public Open() {
    initialize();
}

/**
 * Initialize the contents of the frame.
 */
private void initialize() {
    frmOpen = new JFrame();//create frame 'frmOpen' of new JFrame.
    frmOpen.getContentPane().setBackground(Color.WHITE);
    frmOpen.setBackground(Color.BLACK);
    frmOpen.setTitle("Open - KRAVEL");
    frmOpen.setBounds(0, 0, 623, 360);
    frmOpen.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    frmOpen.getContentPane().setLayout(null);
    frmOpen.setLocationRelativeTo(null);

// create the button 'btnNewButton' of new JButton.
    JButton btnNewButton = new JButton("Administrator");
 //add the action for button 'btnNewButton'.
    btnNewButton.addActionListener(new ActionListener() {
        /**
           * show the Administrator Panel
           */
        public void actionPerformed(ActionEvent e) {
            Adm nw = new Adm(); //create the 'nw' of new Adm class.
            nw.AdmScreen(); //run the AdmScreen method.

        }
    });
    btnNewButton.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
```

```
        btnNewButton.setBounds(97, 180, 168, 46);
        frmOpen.getContentPane().add(btnNewButton);

        // create the button 'btnCustomer' of new JButton.
        JButton btnCustomer = new JButton("Customer");
        //add the action for button 'btnCustomer'.
        btnCustomer.addActionListener(new ActionListener() {
            /**
               * show the Customer Panel
               */
            public void actionPerformed(ActionEvent e) {
                Cus nw = new Cus(); //create the 'nw' of new Cus class
                nw.CusScreen(); //run the CusScreen method.
            }
        });
        btnCustomer.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
        btnCustomer.setBounds(334, 180, 168, 46);
        frmOpen.getContentPane().add(btnCustomer);

        JLabel lblS = new JLabel("Select the service you want ");
        lblS.setFont(new Font("맑은 고딕", Font.BOLD, 25));
        lblS.setBounds(133, 110, 369, 34);
        frmOpen.getContentPane().add(lblS);

        JLabel lblHiWelcomeTo = new JLabel("Welcome to ₩"KRAVEL - Traveling in Korea₩"");
        lblHiWelcomeTo.setForeground(new Color(0, 100, 0));
        lblHiWelcomeTo.setFont(new Font("Footlight MT Light", Font.PLAIN, 15));
        lblHiWelcomeTo.setBounds(136, 77, 326, 25);
        frmOpen.getContentPane().add(lblHiWelcomeTo);

    // create the button 'btnNewButton_1' of new JButton.
        JButton btnNewButton_1 = new JButton("Disconnect");
     //add the action for button 'btnNewButton_1'.
        btnNewButton_1.addActionListener(new ActionListener() {
            /**
               * close the window and disconnect.
               */
            public void actionPerformed(ActionEvent arg0) {
                  try {
```

```
                    //if myConn is not null and myConn is not closed
                      if( myConn != null && !myConn.isClosed()){
                          myConn.close(); //disconnect the myConn
                        }
                      System.out.println("connection 정상종료");
                } catch (SQLException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        });
        btnNewButton_1.setBounds(217, 242, 168, 34);
        frmOpen.getContentPane().add(btnNewButton_1);
    }
}
```

B. Cus

```
package team2;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.SQLException;

import javax.swing.*;
/**
 * class that is a window for customer
 * @author YoungEun Lee, DoKyung Lee, HyoJin Lee, NaGyeong Yeo
 *
 */
public class Cus {
    static String index = null;
    private JFrame frmCus;
    private String tableName;
    private JTextField textField;
    private JTextField textField_1;
    private JTextField grades_textField_2;
```

```
/**
 * open the Cus window
 */
public static void CusScreen() {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                Cus window = new Cus();
                window.frmCus.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}

/**
 * initialize the application.
 */
public Cus() {
    initialize();
}

/**
 * Initialize the contents of the frame.
 */
private void initialize() {
    frmCus = new JFrame(); // create frame 'frmCus' of new JFrame.
    frmCus.setTitle("Customer Service");
    frmCus.setBounds(0, 0, 1025, 594);
    frmCus.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    frmCus.getContentPane().setLayout(null);
    frmCus.setLocationRelativeTo(null);

    // create TabbedPane 'tabbedPane' of new TabbedPane at gui top.
    JTabbedPane tabbedPane = new JTabbedPane(JTabbedPane.TOP);
    tabbedPane.setBackground(Color.WHITE);
    tabbedPane.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
    tabbedPane.setBounds(0, 0, 1005, 542);
```

```
        frmCus.getContentPane().add(tabbedPane);

        // create Panel 'Retrieve_User1' of new JPanel.
        JPanel Retrieve_User1 = new JPanel();
        Retrieve_User1.setBackground(Color.WHITE);
        // add tab about retrieve(User) customer mode at 'tabbedPane'.
        tabbedPane.addTab("Retrieve(User)", null, Retrieve_User1, null);
        Retrieve_User1.setLayout(null);

        JLabel label = new JLabel("< Type >");
        label.setFont(new Font("맑은 고딕", Font.BOLD, 19));
        label.setBounds(13, 256, 89, 29);
        Retrieve_User1.add(label);

        // create the combobox 'Ret_Cus_type_combobox' of new JComboBox.
        JComboBox Ret_Cus_type_combobox = new JComboBox();
        Ret_Cus_type_combobox.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
        Ret_Cus_type_combobox.setBackground(Color.WHITE);
        Ret_Cus_type_combobox.setBounds(13, 287, 460, 30);
        // set the model of 'Ret_Cus_type_combobox' about combobox components.
        Ret_Cus_type_combobox
                .setModel(new DefaultComboBoxModel(new String[] { "Korean", "Western",
"Chinese", "Japanese" }));
        // add combobox 'Ret_Cus_type_combobox' at 'Retrieve_User1'.
        Retrieve_User1.add(Ret_Cus_type_combobox);

        // create the button 'Ret_Cus_type_button' of new JButton.
        JButton Ret_Cus_type_button = new JButton("SHOW");
        // add the action for button 'Ret_Cus_type_button'.
        Ret_Cus_type_button.addActionListener(new ActionListener() {
             //show the food type at Retrieve Panel when Customer mode.
            public void actionPerformed(ActionEvent e) {
                // declare the tableName of "DBCOURSE_type"
                tableName = "DBCOURSE_type";
                // get 'Ret_Cus_type_combobox' of component in String 'type'
                String type = (String) Ret_Cus_type_combobox.getSelectedItem();
                // set the columns name of food type table.
                String colNames[] = { "name", "menu", "price" };
                // create the 'restype' of new retrieve_cus class.
```

```
                retrieve_cus restype = new retrieve_cus(tableName, type, colNames);
                JTable table = restype.show();
                JScrollPane Ret_type_result = new JScrollPane(table);
                Ret_type_result.setBounds(3, 333, 470, 158);
                Retrieve_User1.add(Ret_type_result);
            }
        });
        Ret_Cus_type_button.setForeground(SystemColor.textHighlight);
        Ret_Cus_type_button.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
        Ret_Cus_type_button.setBounds(372, 256, 101, 29);
        Retrieve_User1.add(Ret_Cus_type_button);

        JScrollPane scrollPane = new JScrollPane();
            scrollPane.setBounds(3, 333, 470, 158);
            Retrieve_User1.add(scrollPane);

        JLabel label_1 = new JLabel("< Restaurants >");
        label_1.setFont(new Font("맑은 고딕", Font.BOLD, 19));
        label_1.setBounds(490, 13, 154, 29);
        Retrieve_User1.add(label_1);

        // create the combobox 'Ret_Cus_rst_combobox' of new JComboBox.
        JComboBox Ret_Cus_rst_combobox = new JComboBox();
        Ret_Cus_rst_combobox.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
        Ret_Cus_rst_combobox.setBackground(Color.WHITE);
        Ret_Cus_rst_combobox.setBounds(490, 43, 493, 30);
        // set the model of 'Ret_Cus_rst_combobox' about combobox components.
        Ret_Cus_rst_combobox.setModel(new DefaultComboBoxModel(new String[] { "Chienrong",
"Dintaifung",
                "California Pizza Kitchen", "Haeundae Smokehouse", "JamaeGuksu", "HanKookJib",
"ChungWoo" }));
        // add combobox 'Ret_Cus_rst_combobox' at 'Retrieve_User1'
        Retrieve_User1.add(Ret_Cus_rst_combobox);

        // create the button 'Ret_Cus_rst_button' of new JButton.
        JButton Ret_Cus_rst_button = new JButton("SHOW");
        // add the action for button 'Ret_Cus_rst_button'.
        Ret_Cus_rst_button.addActionListener(new ActionListener() {
            // show the menu information of restaurants at Retrieve Panel when Customer mode.
```

```
        public void actionPerformed(ActionEvent e) {
            // get 'Ret_Cus_rst_combobox' of component in tableName.
            tableName = (String) Ret_Cus_rst_combobox.getSelectedItem();
            // call the 'whichTable' method.
            whichTable();
            // set the columns name of restaurants table.
            String colNames[] = { "name", "menu", "price", "calories", "category" };
            // create the 'recus' of new retrieve_adm class.
            retrieve_adm recus = new retrieve_adm(tableName, index, colNames);
            JTable table;
            try {
                table = recus.show();
                JScrollPane scrollPane_1 = new JScrollPane(table);
                scrollPane_1.setBounds(484, 86, 502, 158);
                Retrieve_User1.add(scrollPane_1);
            } catch (SQLException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
        }
    });

    Ret_Cus_rst_button.setForeground(SystemColor.textHighlight);
    Ret_Cus_rst_button.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
    Ret_Cus_rst_button.setBounds(882, 13, 101, 29);
    Retrieve_User1.add(Ret_Cus_rst_button);

    JScrollPane scrollPane1 = new JScrollPane();
        scrollPane1.setBounds(484, 86, 502, 158);
        Retrieve_User1.add(scrollPane1);

    JLabel label_2 = new JLabel("< Grades >");
    label_2.setFont(new Font("맑은 고딕", Font.BOLD, 19));
    label_2.setBounds(13, 13, 122, 29);
    Retrieve_User1.add(label_2);

    // create the button 'Ret_Cus_grade_button' of new JButton.
    JButton Ret_Cus_grade_button = new JButton("SHOW");
    // add the action for button 'Ret_Cus_grade_button'
```

```
Ret_Cus_grade_button.addActionListener(new ActionListener() {
     //show the grade of restaurants at Retrieve Panel when Customer mode.
    public void actionPerformed(ActionEvent e) {
        // declare the tableName of "DBCOURSE_Grades"
        tableName = "DBCOURSE_Grades";
        // set the columns name of grade table.
        String colNames[] = { "name", "grades" };
        // create the 'resgrade' of new retrieve_adm class.
        retrieve_adm resgrade = new retrieve_adm(tableName, colNames);
        JTable table;
        try {
            table = resgrade.show();
            JScrollPane Cus_grade_result = new JScrollPane(table);
            Cus_grade_result.setBounds(6, 86, 470, 158);
            Retrieve_User1.add(Cus_grade_result);
        } catch (SQLException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    }
});
Ret_Cus_grade_button.setForeground(SystemColor.textHighlight);
Ret_Cus_grade_button.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
Ret_Cus_grade_button.setBounds(372, 13, 101, 29);
Retrieve_User1.add(Ret_Cus_grade_button);

JScrollPane scrollPane2 = new JScrollPane();
    scrollPane2.setBounds(6, 86, 470, 158);
    Retrieve_User1.add(scrollPane2);

JLabel label_3 = new JLabel("< Attraction >");
label_3.setFont(new Font("맑은 고딕", Font.BOLD, 19));
label_3.setBounds(490, 256, 137, 29);
Retrieve_User1.add(label_3);

// create the combobox 'Ret_Cus_attrac_combobox' of new JComboBox.
JComboBox Ret_Cus_attrac_combobox = new JComboBox();
Ret_Cus_attrac_combobox.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
Ret_Cus_attrac_combobox.setBackground(Color.WHITE);
```

```
        Ret_Cus_attrac_combobox.setBounds(488, 287, 495, 30);
        // set the model of 'Ret_Cus_attrac_combobox' about combobox components.
        Ret_Cus_attrac_combobox.setModel(new DefaultComboBoxModel(new String[]
{ "Chienrong", "Dintaifung",
                "California Pizza Kitchen", "Haeundae Smokehouse", "JamaeGuksu", "HanKookJib",
"ChungWoo" }));
        // add combobox 'Ret_Cus_attrac_combobox' at 'Retrieve_User1'.
        Retrieve_User1.add(Ret_Cus_attrac_combobox);

        // create the button 'Ret_Cus_attrac_button' of new JButton.
        JButton Ret_Cus_attrac_button = new JButton("SHOW");
        // add the action for button 'Ret_Cus_attrac_button'.
        Ret_Cus_attrac_button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                 //show the attraction and city of surrounding of restaurants at Retrieve Pane when
Customer mode.
                // get 'Ret_Cus_attrac_combobox' of component in String 'name'
                String name = (String) Ret_Cus_attrac_combobox.getSelectedItem();
                // set the columns name of attraction table.
                String colNames[] = { "city", "attraction" };
                // create the 'res4' of new retrieve_cus class.
                retrieve_cus res4 = new retrieve_cus(name, colNames, 1);
                JTable table = res4.show4();
                JScrollPane scrollPane_3 = new JScrollPane(table);
                scrollPane_3.setBounds(484, 333, 502, 158);
                Retrieve_User1.add(scrollPane_3);
            }
        });
        Ret_Cus_attrac_button.setForeground(SystemColor.textHighlight);
        Ret_Cus_attrac_button.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
        Ret_Cus_attrac_button.setBounds(882, 256, 101, 29);
        Retrieve_User1.add(Ret_Cus_attrac_button);

        JScrollPane scrollPane3 = new JScrollPane();
            scrollPane3.setBounds(484, 333, 502, 158);
            Retrieve_User1.add(scrollPane3);

        // create Panel 'Retrieve_User2' of new JPanel.
        JPanel Retrieve_User2 = new JPanel();
```

```
Retrieve_User2.setBackground(Color.WHITE);
// add tab about retrieve(Price) customer mode at 'tabbedPane'.
tabbedPane.addTab("Retrieve(Price)", null, Retrieve_User2, null);
Retrieve_User2.setLayout(null);

JLabel label_4 = new JLabel("< Price >");
label_4.setFont(new Font("맑은 고딕", Font.BOLD, 17));
label_4.setBounds(17, 0, 85, 29);
Retrieve_User2.add(label_4);

// create the new JTextField about Textfield 'Ret_Cus_price_min_textField'
JTextField Ret_Cus_price_min_textField = new JTextField();
Ret_Cus_price_min_textField.setBounds(119, 2, 114, 28);
// add textfield about 'Ret_Cus_price_min_textField' at 'Retrieve_User2'
Retrieve_User2.add(Ret_Cus_price_min_textField);
Ret_Cus_price_min_textField.setColumns(10);

JLabel label_5 = new JLabel("~ ");
label_5.setBounds(250, 4, 20, 23);
Retrieve_User2.add(label_5);

// create the new JTextField about Textfield 'Ret_Cus_price_max_textField'
JTextField Ret_Cus_price_max_textField = new JTextField();
Ret_Cus_price_max_textField.setColumns(10);
Ret_Cus_price_max_textField.setBounds(275, 2, 114, 28);
// add textfield about Ret_Cus_price_max_textField at Retrieve_User2
Retrieve_User2.add(Ret_Cus_price_max_textField);

// create the button 'Ret_Cus_price_button' of new JButton.
JButton Ret_Cus_price_button = new JButton("SHOW");
// when user click the button of price
 //show all information of menu of restaurants at Retrieve Panel when Customer mode.
Ret_Cus_price_button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      tableName = "DBCOURSE_menu, DBCOURSE_foodCategory";
      String colNames[] = { "name", "menu", "price", "calories", "category", "ingredients" };
      // get 'Ret_Cus_price_min_textField' of component in String 'str_price1'
      String str_price1 = (String) Ret_Cus_price_min_textField.getText();
      // get 'Ret_Cus_price_max_textField' of component in String 'str_price2'
```

```
                String str_price2 = (String) Ret_Cus_price_max_textField.getText();
                int priceMin = Integer.parseInt(str_price1);
                int priceMax = Integer.parseInt(str_price2);

                // create the 'resSelectCus' of new retrieve_cus class.
                retrieve_cus resSelectCus = new retrieve_cus(tableName, priceMin, priceMax,
colNames);
                JTable table = resSelectCus.show();
                JScrollPane scrollPane_4 = new JScrollPane(table);
                scrollPane_4.setBounds(17, 32, 966, 465);
                Retrieve_User2.add(scrollPane_4);
            }
        });
        Ret_Cus_price_button.setForeground(SystemColor.textHighlight);
        Ret_Cus_price_button.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
        Ret_Cus_price_button.setBounds(405, 0, 101, 29);
        Retrieve_User2.add(Ret_Cus_price_button);

        JScrollPane scrollPane5 = new JScrollPane();
        scrollPane5.setBounds(17, 32, 966, 465);
            Retrieve_User2.add(scrollPane5);

        JPanel Grades = new JPanel();
        Grades.setLayout(null);
        Grades.setBackground(Color.WHITE);
        tabbedPane.addTab("Grades", null, Grades, null);

        // create the combobox 'grades_comboBox_1' of new JComboBox.
        JComboBox grades_comboBox_1 = new JComboBox();
        // set the model of 'grades_comboBox_1' about combobox components.
        grades_comboBox_1.setModel(new DefaultComboBoxModel(new String[] { "Chienrong",
"Dintaifung",
                "California Pizza Kitchen", "Haeundae Smokehouse", "JamaeGuksu", "HanKookJib",
"ChungWoo" }));
        grades_comboBox_1.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
        grades_comboBox_1.setBackground(Color.WHITE);
        grades_comboBox_1.setBounds(14, 50, 493, 30);
        Grades.add(grades_comboBox_1);
```

```
        JLabel label_8 = new JLabel("< Grades >");
        label_8.setFont(new Font("맑은 고딕", Font.BOLD, 19));
        label_8.setBounds(13, 13, 122, 29);
        Grades.add(label_8);

        JButton grades_button = new JButton("Enter");// create the button 'Ret_Cus_price_button' of new JButton.
        grades_button.addActionListener(new ActionListener() { // when user click the button of grades in the 'Grades panel
            public void actionPerformed(ActionEvent e) {
                String colNames[] = { "name", "grades" };
                String name = (String) grades_comboBox_1.getSelectedItem();// get the restaurant name in the comboBox
                // get the user's grade to reflect the user's review
                String grades = (String) grades_textField_2.getText();
                update gradeupdate = new update(name, colNames, grades);// use the update's constructor

                try {
                    JTable table = gradeupdate.GradesUpdate();// get the return table value and save it on the 'table'
                    JScrollPane gra = new JScrollPane(table); // create the Scrollpane include table
                    gra.setBounds(24, 92, 919, 269);
                    Grades.add(gra); // add 'gra' component to Grades panel
                } catch (SQLException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                }

            }
        });
        grades_button.setForeground(SystemColor.textHighlight);
        grades_button.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
        grades_button.setBounds(855, 51, 101, 29);
        Grades.add(grades_button);

        JScrollPane scrollPane6 = new JScrollPane();
            scrollPane6.setBounds(24, 92, 919, 269);
            Grades.add(scrollPane6);
```

```
    // create the new JTextField about Textfield 'grades_textField_2'
    grades_textField_2 = new JTextField();
    grades_textField_2.setBounds(535, 50, 274, 30);
    Grades.add(grades_textField_2);
    grades_textField_2.setColumns(10);

}
/**
     * whichTable is method that is used
     * when we know only restaurant's name, not a table's name
     * restaurant's name is changed into real table's name of database
     */
public void whichTable() {
    String str = null, idx = null;
    if (tableName.equals("Chienrong")) {
        str = "DBCOURSE_Rst1";
        idx = "i_price_rst1";
    } else if (tableName.equals("Dintaifung")) {
        str = "DBCOURSE_Rst2";
        idx = "i_price_rst2";
    } else if (tableName.equals("California Pizza Kitchen")) {
        str = "DBCOURSE_Rst3";
        idx = "i_price_rst3";
    } else if (tableName.equals("Haeundae Smokehouse")) {
        str = "DBCOURSE_Rst4";
        idx = "i_price_rst4";
    } else if (tableName.equals("JamaeGuksu")) {
        str = "DBCOURSE_Rst5";
        idx = "i_price_rst5";
    } else if (tableName.equals("HanKookJib")) {
        str = "DBCOURSE_Rst6";
        idx = "i_price_rst6";
    } else if (tableName.equals("ChungWoo")) {
        str = "DBCOURSE_Rst7";
        idx = "i_price_rst7";
    }
    tableName = str;
    index = idx;
```

```
    }
}
```

C. Adm

```
package team2;

import java.awt.*;
import java.awt.event.*;
import java.sql.SQLException;
import java.util.InputMismatchException;

import javax.swing.*;

/**
 *
 * class that is a window for administrator
 * @author YoungEun Lee, DoKyung Lee, HyoJin Lee, NaGyeong Yeo
 *
 */
public class Adm {
    static String tableName, index;
    private JFrame frmAdm;
    private JTextField Insert_Rest_Menu_textField;
    private JTextField Insert_Rest_Price_textField_1;
    private JTextField Insert_Rest_Calo_textField_1;
    private JTextField Insert_Rest_Cate_textField_1;
    private JTextField delete_Menu_text;
    private JTextField delete_PriceRange1_textField_1;
    private JTextField delete_PriceRange2_textField_2;
    private JTextField delete_CaloriesRange1_textField_3;
    private JTextField delete_CaloriesRange2_textField_4;
    private JTextField update_Attraction_textField_15;
    private JTextField update_Rest_Menu_textField_1;
    private JTextField update_Rest_Price_textField_1;
    private JTextField Insert_FoodCate_Cate_textField_1;
    private JTextField Insert_FoodCate_Ingre_textField_1;

    /**
```

```
 * open the Adm window.
 */
public static void AdmScreen() {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                Adm window = new Adm();
                window.frmAdm.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}

/**
 * initialize the application.
 */
public Adm() {
    initialize();
}

/**
 * Initialize the contents of the frame.
 */
private void initialize() {
    frmAdm = new JFrame(); // create frame 'frmCus' of new JFrame.
    frmAdm.setTitle("Administrator Service");
    frmAdm.setBounds(0, 0, 1027, 594);
    frmAdm.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    frmAdm.getContentPane().setLayout(null);
    frmAdm.setLocationRelativeTo(null);

    // create TabbedPane 'tabbedPane' of new TabbedPane at gui top.
    JTabbedPane tabbedPane = new JTabbedPane(JTabbedPane.TOP);
    tabbedPane.setBackground(Color.WHITE);
    tabbedPane.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
    tabbedPane.setBounds(0, 0, 1005, 542);
    frmAdm.getContentPane().add(tabbedPane);
```

```
        // create Panel 'Retrieve_Adm' of new JPanel.
        JPanel Retrieve_Adm = new JPanel();
        Retrieve_Adm.setBackground(Color.WHITE);
        // add tab about Retrieve(Administrator) Administrator mode at 'tabbedPane'.
        tabbedPane.addTab("Retrieve(Administrator)", null, Retrieve_Adm, null);
        Retrieve_Adm.setLayout(null);

        JLabel label_25 = new JLabel("< Restaurants >");
        label_25.setFont(new Font("맑은 고딕", Font.BOLD, 18));
        label_25.setBounds(30, 13, 157, 21);
        Retrieve_Adm.add(label_25);

        // create the button 'Retrieve_Rest_button' of new JButton.
        JButton Retrieve_Rest_button = new JButton("SHOW");
        // add the action for button 'Retrieve_Rest_button'.
        Retrieve_Rest_button.addActionListener(new ActionListener() {
            // show the restaurants information at Retrieve Panel when Administrator mode.
            public void actionPerformed(ActionEvent e) {
                // declare the tableName of "DBCOURSE_Restaurants"
                tableName = "DBCOURSE_Restaurants";
                // in this button, columns in colNames should be selected
                String colNames[] = { "name", "city", "address", "type" };
                // create a object of class named 'retrieve_adm'
                retrieve_adm readm = new retrieve_adm(tableName, colNames);
                // execute 'show' method in 'retrieve_adm' and it becomes a value of 'table'
                JTable table;
                try {
                    table = readm.show(); // create a scrollpane named 'Ret_rest_result' including
data of 'table'
                    JScrollPane Ret_rest_result = new JScrollPane(table);
                    Ret_rest_result.setBounds(17, 45, 551, 150);
                    // panel named 'Retrieve_Adm' contains 'Ret_rest_result'
                    Retrieve_Adm.add(Ret_rest_result);
                } catch (SQLException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                }
            }
```

```
        });
        Retrieve_Rest_button.setForeground(SystemColor.textHighlight);
        Retrieve_Rest_button.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
        Retrieve_Rest_button.setBounds(467, 9, 101, 29);
        Retrieve_Adm.add(Retrieve_Rest_button);

        JScrollPane scrollPane = new JScrollPane();
        scrollPane.setBounds(16, 45, 552, 150);
        Retrieve_Adm.add(scrollPane);

        JLabel label_26 = new JLabel("< Attraction >");
        label_26.setFont(new Font("맑은 고딕", Font.BOLD, 18));
        label_26.setBounds(645, 13, 166, 21);
        Retrieve_Adm.add(label_26);

        // create the button 'Retrieve_Attrac_button' of new JButton.
        JButton Retrieve_Attrac_button = new JButton("SHOW");
        // add the action for button 'Retrieve_Attrac_button'.
        Retrieve_Attrac_button.addActionListener(new ActionListener() {
            // show the attraction and city at Retrieve Panel when Administrator mode.
            public void actionPerformed(ActionEvent arg0) {
                // declare the tableName of "DBCOURSE_Attraction"
                tableName = "DBCOURSE_Attraction";
                // set the columns name of food type table.
                String colNames[] = { "city", "attraction" };
                // create the 'readm' of new retrieve_adm class.
                retrieve_adm readm = new retrieve_adm(tableName, colNames);
                JTable table;
                try {
                    table = readm.show();
                    JScrollPane Ret_attrac_result = new JScrollPane(table);
                    Ret_attrac_result.setBounds(645, 45, 306, 150);
                    Retrieve_Adm.add(Ret_attrac_result);
                } catch (SQLException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }

            }
```

```
        });
        Retrieve_Attrac_button.setForeground(SystemColor.textHighlight);
        Retrieve_Attrac_button.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
        Retrieve_Attrac_button.setBounds(850, 9, 101, 29);
        Retrieve_Adm.add(Retrieve_Attrac_button);

        JScrollPane scrollPane1 = new JScrollPane();
            scrollPane1.setBounds(645, 45, 306, 150);
            Retrieve_Adm.add(scrollPane1);

        JLabel label_27 = new JLabel("< Food Category >");
        label_27.setFont(new Font("맑은 고딕", Font.BOLD, 18));
        label_27.setBounds(642, 198, 183, 27);
        Retrieve_Adm.add(label_27);

        // create the combobox 'Retrieve_RestInfo_Combobox' of new JComboBox.
        JComboBox Retrieve_RestInfo_Combobox = new JComboBox();
        Retrieve_RestInfo_Combobox.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
        Retrieve_RestInfo_Combobox.setBackground(Color.WHITE);
        Retrieve_RestInfo_Combobox.setBounds(27, 234, 545, 30);
        // set the model of 'Retrieve_RestInfo_Combobox' about combobox components.
        Retrieve_RestInfo_Combobox.setModel(new DefaultComboBoxModel(new String[]
{ "Chienrong", "Dintaifung",
                "California Pizza Kitchen", "Haeundae Smokehouse", "JamaeGuksu", "HanKookJib",
"ChungWoo" }));
        // add combobox 'Retrieve_RestInfo_Combobox' at 'Retrieve_Adm'
        Retrieve_Adm.add(Retrieve_RestInfo_Combobox);

        // create the button 'Retrieve_RestInfo_button' of new JButton.
        JButton Retrieve_RestInfo_button = new JButton("SHOW");
        // add the action for button 'Retrieve_RestInfo_button'.
        Retrieve_RestInfo_button.addActionListener(new ActionListener() {
            // show the menu information of restaurants at Retrieve Panel when Administrator
            // mode.
            public void actionPerformed(ActionEvent e) {
                // get 'Retrieve_RestInfo_Combobox' of component in tableName.
                tableName = (String) Retrieve_RestInfo_Combobox.getSelectedItem();
                // call the 'whichTable' method
                whichTable();
```

```
            // set the columns name of restaurants table.
            String colNames[] = { "name", "menu", "price", "calories", "category" };
            // create the 'readm' of new retrieve_adm class.
            retrieve_adm readm = new retrieve_adm(tableName, index, colNames);
            JTable table;
            try {
                table = readm.show();
                JScrollPane Ret_info_result = new JScrollPane(table);
                Ret_info_result.setBounds(27, 268, 547, 220);
                Retrieve_Adm.add(Ret_info_result);
            } catch (SQLException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
        }

    });
    // create the button 'Retrieve_FoodCate_button' of new JButton.
    JButton Retrieve_FoodCate_button = new JButton("SHOW");
    // add the action for button 'Retrieve_FoodCate_button'
    Retrieve_FoodCate_button.addActionListener(new ActionListener() {
        // show the food category and using ingredients at Retrieve Panel when
        // Administrator mode.
        public void actionPerformed(ActionEvent e) {
            // declare the tableName of "DBCOURSE_FoodCategory"
            tableName = "DBCOURSE_FoodCategory";
            // set the columns name of food category table.
            String colNames[] = { "category", "ingredients" };
            // create the 'readm' of new retrieve_adm class.
            retrieve_adm readm = new retrieve_adm(tableName, colNames);
            JTable table;
            try {
                table = readm.show();
                JScrollPane Ret_cate_result = new JScrollPane(table);
                Ret_cate_result.setBounds(642, 237, 306, 251);
                Retrieve_Adm.add(Ret_cate_result);
            } catch (SQLException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
```

```
            }
        }
    });
    Retrieve_FoodCate_button.setForeground(SystemColor.textHighlight);
    Retrieve_FoodCate_button.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
    Retrieve_FoodCate_button.setBounds(847, 202, 101, 29);
    Retrieve_Adm.add(Retrieve_FoodCate_button);

    JScrollPane scrollPane3 = new JScrollPane();
        scrollPane3.setBounds(27, 268, 547, 220);
        Retrieve_Adm.add(scrollPane3);



    JLabel label_28 = new JLabel("< Restaurants' Info >");
    label_28.setBackground(new Color(240, 240, 240));
    label_28.setFont(new Font("맑은 고딕", Font.BOLD, 18));
    label_28.setBounds(27, 198, 211, 21);
    Retrieve_Adm.add(label_28);

    Retrieve_RestInfo_button.setForeground(SystemColor.textHighlight);
    Retrieve_RestInfo_button.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
    Retrieve_RestInfo_button.setBounds(471, 198, 101, 29);
    Retrieve_Adm.add(Retrieve_RestInfo_button);

    JScrollPane scrollPane4 = new JScrollPane();
        scrollPane4.setBounds(642, 237, 306, 251);
        Retrieve_Adm.add(scrollPane4);

    // create Panel 'delete' of new JPanel.
    JPanel delete = new JPanel();
    delete.setBackground(Color.WHITE);
    // add tab about delete Administrator mode at 'tabbedPane'.
    tabbedPane.addTab("delete", null, delete, null);
    delete.setLayout(null);

    JLabel delete_Rest_label = new JLabel("< Restaurant >");
    delete_Rest_label.setFont(new Font("맑은 고딕", Font.BOLD, 20));
    delete_Rest_label.setBounds(151, 15, 194, 27);
    delete.add(delete_Rest_label);
```

```
        // create the combobox 'delete_RestName_comboBox_1' of new JComboBox.
        JComboBox delete_RestName_comboBox_1 = new JComboBox();
        delete_RestName_comboBox_1.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
        delete_RestName_comboBox_1.setBackground(Color.WHITE);
        delete_RestName_comboBox_1.setBounds(304, 66, 220, 30);
        // set the model of 'delete_RestName_comboBox_1' about combobox components.
        delete_RestName_comboBox_1.setModel(new DefaultComboBoxModel(new String[]
{ "Chienrong", "Dintaifung",
                "California Pizza Kitchen", "Haeundae Smokehouse", "JamaeGuksu", "HanKookJib",
"ChungWoo" }));
        // add combobox 'delete_RestName_comboBox_1' at 'Retrieve_User1'.
        delete.add(delete_RestName_comboBox_1);

        delete_Menu_text = new JTextField();
        delete_Menu_text.setColumns(40);
        delete_Menu_text.setBounds(305, 112, 381, 35);
        delete.add(delete_Menu_text);

        // create the new JTextPane about TextPane 'delete_Result'
        JTextPane delete_Result = new JTextPane();
        delete_Result.setBackground(UIManager.getColor("Button.background"));
        delete_Result.setBounds(161, 448, 656, 40);
        delete.add(delete_Result);

        // create the button 'delete_Rest_button' of new JButton.
        JButton delete_Rest_button = new JButton("ENTER");
        // add the action for button 'delete_Rest_button'.
        delete_Rest_button.addActionListener(new ActionListener() {
            // when user click the button of delete related to restaurant in the 'delete'
            // panel
            public void actionPerformed(ActionEvent e) {
                // save the value of restaurant name
                tableName = (String) delete_RestName_comboBox_1.getSelectedItem();
                String name = tableName;
                // using restaurant name, get the real table name of database
                whichTable();
                // get the menu name to delete
                String menu = (String) delete_Menu_text.getText();
```

```
                // when user click the button without any menu name
                if (menu.equals("")) {
                    menu = null;
                    // show the message to input again
                    delete_Result.setText("menu 를 다시 입력해주세요.");
                    return; // quit the method
                }
                // use the delete's constructor
                delete resdelete = new delete(tableName, menu);
                // implement the Restaurantdelete() method in the delete class
                resdelete.Restaurantdelete();
                // complete the request, then set the text on the textfield to vacuum
                delete_Menu_text.setText("");
                // show the success message
                delete_Result.setText(name + "의 menu 삭제를 성공하였습니다.");
            }
        });
        delete_Rest_button.setForeground(SystemColor.textHighlight);
        delete_Rest_button.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
        delete_Rest_button.setBounds(706, 114, 90, 30);
        delete.add(delete_Rest_button);

        JLabel delete_Price = new JLabel("< Price >");
        delete_Price.setFont(new Font("맑은 고딕", Font.BOLD, 20));
        delete_Price.setBounds(151, 174, 118, 21);
        delete.add(delete_Price);

        // create the combobox 'delete_Price_Name_comboBox_2' of new JComboBox.
        JComboBox delete_Price_Name_comboBox_2 = new JComboBox();
        delete_Price_Name_comboBox_2.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
        delete_Price_Name_comboBox_2.setBackground(Color.WHITE);
        delete_Price_Name_comboBox_2.setBounds(304, 203, 220, 30);
        // set the model of 'delete_Price_Name_comboBox_2' about combobox components.
        delete_Price_Name_comboBox_2.setModel(new DefaultComboBoxModel(new String[]
{ "Chienrong", "Dintaifung",
                "California Pizza Kitchen", "Haeundae Smokehouse", "JamaeGuksu", "HanKookJib",
"ChungWoo" }));
        // add combobox 'delete_Price_Name_comboBox_2' at 'delete'.
        delete.add(delete_Price_Name_comboBox_2);
```

```
        // create the new JTextField about Textfield 'delete_PriceRange1_textField_1'
        delete_PriceRange1_textField_1 = new JTextField();
        delete_PriceRange1_textField_1.setColumns(18);
        delete_PriceRange1_textField_1.setBounds(305, 252, 140, 35);
        // add textfield about 'delete_PriceRange1_textField_1' at Retrieve_User2
        delete.add(delete_PriceRange1_textField_1);

        JLabel label_2 = new JLabel("~ ");
        label_2.setBounds(477, 251, 20, 37);
        delete.add(label_2);

        // create the new JTextField about Textfield 'delete_PriceRange2_textField_2'
        delete_PriceRange2_textField_2 = new JTextField();
        delete_PriceRange2_textField_2.setColumns(18);
        delete_PriceRange2_textField_2.setBounds(525, 252, 140, 35);
        delete.add(delete_PriceRange2_textField_2);

        JButton delete_Price_button = new JButton("ENTER");// create the button
'delete_Price_button' of new JButton.
        delete_Price_button.addActionListener(new ActionListener() {// add the action for button
'delete_Price_button'.
            public void actionPerformed(ActionEvent e) {// when user click the button of delete
related to price in the
                                                        // 'delete' panel
                tableName = (String) delete_Price_Name_comboBox_2.getSelectedItem();// get the
restaurant name in the
                                                                    // comboBox
                String name = tableName;// save the value of restaurant name
                whichTable();// using restaurant name, get the real table name of database
                String min = (String) delete_PriceRange1_textField_1.getText();// get the min value of
price
                String max = (String) delete_PriceRange2_textField_2.getText();// get the max value
of price

                if (min.equals("") || max.equals("")) {// when user click the button without price value
                    delete_Result.setText("삭제할 price 범위를 다시 입력해주세요.");// show the
message to input again
                    // clear the textfield.
```

```
                    delete_PriceRange1_textField_1.setText("");
                    delete_PriceRange2_textField_2.setText("");
                    return; // quit the method
                }
                int price_min = Integer.parseInt(delete_PriceRange1_textField_1.getText());// change
the type of min
                                                                                    // value of price to
int
                int price_max = Integer.parseInt(delete_PriceRange2_textField_2.getText());// change
the type of max
                                                                                    // value of price to
int
                delete pricedelete = new delete(tableName, price_min, price_max); // use the delete's
constructor
                pricedelete.Pricedelete();// implement the Pricedelete() method in the delete class
                delete_PriceRange1_textField_1.setText("");// complete the request, then set the text
on the textfield
                                                        // to vacuum
                delete_PriceRange2_textField_2.setText("");// complete the request, then set the text
on the textfield
                                                        // to vacuum
                delete_Result.setText(name + "의 해당 price 범위 메뉴를 삭제했습니다.");// show
the success message
            }
        });
        delete_Price_button.setForeground(SystemColor.textHighlight);
        delete_Price_button.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
        delete_Price_button.setBounds(706, 254, 90, 30);
        delete.add(delete_Price_button);

        JLabel delete_Calories = new JLabel("< Calories >");
        delete_Calories.setFont(new Font("맑은 고딕", Font.BOLD, 20));
        delete_Calories.setBounds(151, 323, 167, 21);
        delete.add(delete_Calories);

        // create the combobox 'delete_Calories_NamecomboBox_3' of new JComboBox.
        JComboBox delete_Calories_NamecomboBox_3 = new JComboBox();
        delete_Calories_NamecomboBox_3.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
        delete_Calories_NamecomboBox_3.setBackground(Color.WHITE);
```

```
        delete_Calories_NamecomboBox_3.setBounds(304, 353, 220, 30);
        // set the model of 'delete_Calories_NamecomboBox_3' about combobox components.
        delete_Calories_NamecomboBox_3.setModel(new DefaultComboBoxModel(new String[]
{ "Chienrong", "Dintaifung",
                "California Pizza Kitchen", "Haeundae Smokehouse", "JamaeGuksu", "HanKookJib",
"ChungWoo" }));
        // add combobox 'delete_Calories_NamecomboBox_3' at 'Retrieve_User1'.
        delete.add(delete_Calories_NamecomboBox_3);

        // create the new JTextField about Textfield 'delete_CaloriesRange1_textField_3'
        delete_CaloriesRange1_textField_3 = new JTextField();
        delete_CaloriesRange1_textField_3.setColumns(18);
        delete_CaloriesRange1_textField_3.setBounds(305, 402, 140, 35);
        delete.add(delete_CaloriesRange1_textField_3);

        JLabel label_12 = new JLabel("~ ");
        label_12.setBounds(477, 401, 20, 37);
        delete.add(label_12);

        // create the new JTextField about Textfield 'delete_CaloriesRange2_textField_4'
        delete_CaloriesRange2_textField_4 = new JTextField();
        delete_CaloriesRange2_textField_4.setColumns(18);
        delete_CaloriesRange2_textField_4.setBounds(525, 402, 140, 35);
        delete.add(delete_CaloriesRange2_textField_4);

        // create the button 'delete_Calories_button' of new JButton.
        JButton delete_Calories_button = new JButton("ENTER");
        // add the action for button 'delete_Calories_button'.
        delete_Calories_button.addActionListener(new ActionListener() {
             //delete the restaurant menu in calory range at Delete Panel when Administrator
mode.
            public void actionPerformed(ActionEvent e) {
                // declare the tableName about 'delete_Calories_NamecomboBox_3' into String.
                tableName = (String) delete_Calories_NamecomboBox_3.getSelectedItem();
                String name = tableName;
                // call the 'whichTable' method.
                whichTable();
                // declare the min about 'delete_CaloriesRange1_textField_3' into String.
                String min = (String) delete_CaloriesRange1_textField_3.getText();
```

```
            // declare the max about 'delete_CaloriesRange2_textField_4' into String.
            String max = (String) delete_CaloriesRange2_textField_4.getText();
            // if min or max enter the null value
            if (min.equals("") || max.equals("")) {
                delete_Result.setText("삭제할 calory 범위를 다시 입력해주세요.");
                // clear the textfield.
                delete_CaloriesRange1_textField_3.setText("");
                delete_CaloriesRange2_textField_4.setText("");
                return; // try again the enter the value
            }
            // get back the value of 'delete_CaloriesRange1_textField_3' in caloty_min
            int calory_min = Integer.parseInt(delete_CaloriesRange1_textField_3.getText());
            // get back the value of 'delete_CaloriesRange1_textField_4' in caloty_max
            int calory_max = Integer.parseInt(delete_CaloriesRange2_textField_4.getText());
            // create the 'calorydelete' of new delete class
            delete calorydelete = new delete(tableName, calory_min, calory_max);
            // run the 'Calorydelete' method in 'delete' class
            calorydelete.Calorydelete();
            delete_CaloriesRange1_textField_3.setText("");
            delete_CaloriesRange2_textField_4.setText("");
            delete_Result.setText(name + "의 해당 calory 범위 메뉴를 삭제했습니다.");
        }
    });

    delete_Calories_button.setForeground(SystemColor.textHighlight);
    delete_Calories_button.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
    delete_Calories_button.setBounds(706, 404, 90, 30);
    delete.add(delete_Calories_button);

    JLabel delete_RestName1 = new JLabel("Name");
    delete_RestName1.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
    delete_RestName1.setBounds(230, 70, 52, 17);
    delete.add(delete_RestName1);

    JLabel delete_RestMenu = new JLabel("Menu");
    delete_RestMenu.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
    delete_RestMenu.setBounds(230, 116, 52, 26);
    delete.add(delete_RestMenu);
```

```
JLabel delete_RestName2 = new JLabel("Name");
delete_RestName2.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
delete_RestName2.setBounds(230, 210, 52, 23);
delete.add(delete_RestName2);

JLabel delete_PriceRange = new JLabel("Price Range");
delete_PriceRange.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
delete_PriceRange.setBounds(189, 251, 101, 34);
delete.add(delete_PriceRange);

JLabel delete_RestName3 = new JLabel("Name");
delete_RestName3.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
delete_RestName3.setBounds(230, 359, 52, 17);
delete.add(delete_RestName3);

JLabel delete_CaloriesRange = new JLabel("Calories Range");
delete_CaloriesRange.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
delete_CaloriesRange.setBounds(161, 404, 129, 29);
delete.add(delete_CaloriesRange);

// create Panel 'update' of new JPanel.
JPanel update = new JPanel();
update.setBackground(Color.WHITE);
// add tab about delete Administrator mode at 'tabbedPane'.
tabbedPane.addTab("update", null, update, null);
update.setLayout(null);

JLabel update_label_18 = new JLabel("< Attraction >");
update_label_18.setFont(new Font("맑은 고딕", Font.BOLD, 20));
update_label_18.setBounds(159, 245, 200, 21);
update.add(update_label_18);

JLabel update_label_19 = new JLabel("Attraction");
update_label_19.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
update_label_19.setForeground(Color.RED);
update_label_19.setBounds(226, 330, 200, 35);
update.add(update_label_19);

// create the new JTextField about Textfield 'update_Attraction_textField_15'.
```

```
        update_Attraction_textField_15 = new JTextField();
        update_Attraction_textField_15.setColumns(18);
        update_Attraction_textField_15.setBounds(351, 330, 220, 30);
        update.add(update_Attraction_textField_15);

        // create the combobox 'update_Attraction_City_comboBox_4' of new JComboBox.
        JComboBox update_Attraction_City_comboBox_4 = new JComboBox();
        update_Attraction_City_comboBox_4.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
        update_Attraction_City_comboBox_4.setBackground(Color.WHITE);
        update_Attraction_City_comboBox_4.setBounds(351, 280, 220, 30);
        // set the model of 'update_Attraction_City_comboBox_4' about combobox
        // components.
        update_Attraction_City_comboBox_4
                .setModel(new DefaultComboBoxModel(new String[] { "Seoul", "Busan", "Jeonju",
"Gangwon-do", "Jeju" }));
        // add combobox 'update_Attraction_City_comboBox_4' at 'update'.
        update.add(update_Attraction_City_comboBox_4);

        // create the new JTextPane about TextPane 'update_Result'.
        JTextPane update_Result = new JTextPane();
        update_Result.setBackground(UIManager.getColor("Button.background"));
        update_Result.setBounds(159, 399, 656, 40);
        update.add(update_Result);

        // create the button 'delete_Calories_button' of new JButton.
        JButton update_Attraction_button_6 = new JButton("ENTER");
        update_Attraction_button_6.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {// when user click the button of delete
related to attraction in
                                                        // the 'update' panel
                String city = (String) update_Attraction_City_comboBox_4.getSelectedItem();// get
the city name in the
                                                                                // comboBox
                String attraction = (String) update_Attraction_textField_15.getText();// get the
attraction name to
                                                                            // update

                if (attraction.equals("")) {// when user click the button without any attraction name
```

```
                    update_Result.setText("update 할 attraction 을 다시 입력해주세요.");// show the
message to input again
                    return; // quit the method
                }

                update attractupdate = new update(city, attraction); // use the update's constructor
                attractupdate.AttractionUpdate();// implement the AttractionUpdate() method in the
update class
                update_Attraction_textField_15.setText("");// complete the request, then set the text
on the textfield
                                                        // to vacuum
                update_Result.setText(city + "의 attraction update 를 성공했습니다."); // show the
success message
            }
        });
        update_Attraction_button_6.setForeground(SystemColor.textHighlight);
        update_Attraction_button_6.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
        update_Attraction_button_6.setBounds(726, 330, 90, 30);
        update.add(update_Attraction_button_6);

        // create the combobox 'update_Restaurants_City_comboBox_1' of new JComboBox.
        JComboBox update_Restaurants_City_comboBox_1 = new JComboBox();
        update_Restaurants_City_comboBox_1.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
        update_Restaurants_City_comboBox_1.setBackground(Color.WHITE);
        update_Restaurants_City_comboBox_1.setBounds(351, 91, 220, 30);
        // set the model of 'update_Restaurants_City_comboBox_1' about combobox
        // components.
        update_Restaurants_City_comboBox_1.setModel(new DefaultComboBoxModel(new String[]
{ "Chienrong", "Dintaifung",
                "California Pizza Kitchen", "Haeundae Smokehouse", "JamaeGuksu", "HanKookJib",
"ChungWoo" }));
        // add combobox 'update_Restaurants_City_comboBox_1' at 'update'.
        update.add(update_Restaurants_City_comboBox_1);

        // create the button 'update_Rest_button_6' of new JButton.
        JButton update_Rest_button_6 = new JButton("ENTER");
        // add the action for button 'update_Rest_button_6'.
        update_Rest_button_6.addActionListener(new ActionListener() {
            // update the price of menu of restaurants at Delete Panel when Administrator
```

```
        // mode.
        public void actionPerformed(ActionEvent e) {
            // declare the tableName about 'update_Restaurants_City_comboBox_1' into String.
            tableName = (String) update_Restaurants_City_comboBox_1.getSelectedItem();
            String name = tableName;
            // call the 'whichTable' method
            whichTable();
            // declare the menu about 'update_Rest_Menu_textField_1' into String.
            String menu = (String) update_Rest_Menu_textField_1.getText();
            // declare the val about 'update_Rest_Price_textField_1' into String.
            String val = (String) update_Rest_Price_textField_1.getText();
            // if menu or val enter the null value
            if (menu.equals("") || val.equals("")) {
                update_Result.setText("update 할 menu 와 price 값을 둘 다 입력해주세요.");
                return; // try again the enter the value
            }

            // get back the value of 'val' in 'price'
            int price = Integer.parseInt(val);
            // create the 'attractupdate' of new update class
            update attractupdate = new update(tableName, menu, price);
            // run the 'RestaurantUpdate' method in 'update' class
            attractupdate.RestaurantUpdate();
            update_Rest_Menu_textField_1.setText("");
            update_Rest_Price_textField_1.setText("");
            update_Result.setText(name + "의 update 를 성공했습니다.");
        }
    });
    update_Rest_button_6.setForeground(SystemColor.textHighlight);
    update_Rest_button_6.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
    update_Rest_button_6.setBounds(726, 184, 90, 30);
    update.add(update_Rest_button_6);

    JLabel label_1 = new JLabel("Price");
    label_1.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
    label_1.setForeground(Color.RED);
    label_1.setBounds(225, 184, 50, 35);
    update.add(label_1);
```

```
JLabel label_3 = new JLabel("< Restaurants >");
label_3.setFont(new Font("맑은 고딕", Font.BOLD, 20));
label_3.setBounds(159, 45, 170, 21);
update.add(label_3);

JLabel label_13 = new JLabel("Menu");
label_13.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
label_13.setForeground(Color.RED);
label_13.setBounds(225, 141, 50, 35);
update.add(label_13);

// create the new JTextField about Textfield 'update_Rest_Menu_textField_1'.
update_Rest_Menu_textField_1 = new JTextField();
update_Rest_Menu_textField_1.setColumns(18);
update_Rest_Menu_textField_1.setBounds(351, 141, 200, 35);
update.add(update_Rest_Menu_textField_1);

// create the new JTextField about Textfield 'update_Rest_Price_textField_1'.
update_Rest_Price_textField_1 = new JTextField();
update_Rest_Price_textField_1.setColumns(18);
update_Rest_Price_textField_1.setBounds(351, 184, 200, 35);
update.add(update_Rest_Price_textField_1);

JLabel label_16 = new JLabel("Name");
label_16.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
label_16.setBounds(225, 91, 50, 35);
update.add(label_16);

JLabel label_17 = new JLabel("**Red -> REQUIRED FIELD");
label_17.setForeground(Color.RED);
label_17.setFont(new Font("맑은 고딕", Font.BOLD, 15));
label_17.setBounds(350, 45, 259, 35);
update.add(label_17);

JLabel label_18 = new JLabel("ex) Sweet Potato Fries");
label_18.setFont(new Font("맑은 고딕", Font.ITALIC, 13));
label_18.setBounds(557, 141, 188, 35);
update.add(label_18);
```

```
JLabel label_19 = new JLabel("ex) 16900");
label_19.setFont(new Font("맑은 고딕", Font.ITALIC, 13));
label_19.setBounds(557, 184, 188, 35);
update.add(label_19);

JLabel lblExNamsanHaeundae = new JLabel("ex) Namsan, Haeundae");
lblExNamsanHaeundae.setFont(new Font("맑은 고딕", Font.ITALIC, 13));
lblExNamsanHaeundae.setBounds(385, 332, 188, 35);
update.add(lblExNamsanHaeundae);

JLabel lblCity = new JLabel("City");
lblCity.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
lblCity.setForeground(Color.RED);
lblCity.setBounds(226, 280, 50, 35);
update.add(lblCity);

JLabel lblPrice = new JLabel("₩u25B6 Update of Price");
lblPrice.setFont(new Font("맑은 고딕", Font.ITALIC, 18));
lblPrice.setForeground(new Color(0, 0, 255));
lblPrice.setBounds(351, 220, 163, 21);
update.add(lblPrice);

JLabel lblUpdateOf = new JLabel("₩u25B6 Update of Attraction");
lblUpdateOf.setForeground(Color.BLUE);
lblUpdateOf.setFont(new Font("맑은 고딕", Font.ITALIC, 18));
lblUpdateOf.setBounds(351, 363, 200, 21);
update.add(lblUpdateOf);

// create the Panel 'Insert' of new JPanel
JPanel Insert = new JPanel();
Insert.setBackground(Color.WHITE);
// add tab about 'Insert' administrator mode at 'tabbedPane'.
tabbedPane.addTab("Insert", null, Insert, null);
Insert.setLayout(null);

// create the combobox 'Insert_Rest_Name_comboBox_1' of new JComboBox.
JComboBox Insert_Rest_Name_comboBox_1 = new JComboBox();
Insert_Rest_Name_comboBox_1.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
Insert_Rest_Name_comboBox_1.setBackground(Color.WHITE);
```

```
        Insert_Rest_Name_comboBox_1.setBounds(341, 212, 220, 30);
        // set the model of 'Insert_Rest_Name_comboBox_1' about combobox components.
        Insert_Rest_Name_comboBox_1.setModel(new DefaultComboBoxModel(new String[]
{ "Chienrong", "Dintaifung",
                "California Pizza Kitchen", "Haeundae Smokehouse", "JamaeGuksu", "HanKookJib",
"ChungWoo" }));
        // add combobox 'Insert_Rest_Name_comboBox_1' at 'Insert'.
        Insert.add(Insert_Rest_Name_comboBox_1);

        // create the TextPane 'Insert_Result' of new TextPane.
        JTextPane Insert_Result = new JTextPane();
        Insert_Result.setBackground(UIManager.getColor("Button.background"));
        Insert_Result.setBounds(151, 440, 656, 40);
        Insert.add(Insert_Result);

        // create the button 'Insert_Rest_button_6' of new JButton.
        JButton Insert_Rest_button_6 = new JButton("ENTER");
        // add the action for button 'Insert_Rest_button_6'.
        Insert_Rest_button_6.addActionListener(new ActionListener() {
            //insert the menu, price, calory and category in selected restaurants at Insert
            //Panel when Administrator mode.

           public void actionPerformed(ActionEvent e) {
              // declare the tableName about 'Insert_Rest_Name_comboBox_1' into String.
              tableName = (String) Insert_Rest_Name_comboBox_1.getSelectedItem();
              String name = tableName;
              // clarify a table name
              whichTable();

              // declare the menu about 'Insert_Rest_Menu_textField' into String.
              String menu = (String) Insert_Rest_Menu_textField.getText();
              // declare the price_val about 'Insert_Rest_Price_textField_1' into String.
              String price_val = (String) Insert_Rest_Price_textField_1.getText();
              // declare the calories about 'Insert_Rest_Calo_textField_1' into String.
              String calories = (String) Insert_Rest_Calo_textField_1.getText();
              // declare the category about 'Insert_Rest_Cate_textField_1' into String.
              String category = (String) Insert_Rest_Cate_textField_1.getText();

              // if menu or price_val enter the null value
```

```
            if (menu.equals("") || price_val.equals("")) {
                Insert_Result.setText("menu, price 값을 둘 다 입력해주세요.");
                return; // try again the enter the value
            }
            if (calories.equals("")) {
                calories = null;
            }
            if (category.equals("")) {
                category = null;
            }
            try {
                // get back the value of 'delete_PriceRange1_textField_1' in price_min.
                int price = Integer.parseInt(price_val);
                // insert using the method RestaurantInsert()
                insert resinsert = new insert(tableName, name, menu, price, calories, category);
                // run the 'RestaurantInsert' method in 'insert' class.
                resinsert.RestaurantInsert();
            } catch (NumberFormatException e1) {
                // if price and calories is not integer
                // print the panel 'Insert_Result'
                Insert_Result.setText("price, calories 값을 정수로 입력해주세요.");
                return; // try again enter the value.
            }
            /* clear textFields */
            Insert_Rest_Menu_textField.setText("");
            Insert_Rest_Price_textField_1.setText("");
            Insert_Rest_Calo_textField_1.setText("");
            Insert_Rest_Cate_textField_1.setText("");
            Insert_Result.setText(name + "의 menu insert 를 성공하였습니다.");
        }
    });
    Insert_Rest_button_6.setForeground(SystemColor.textHighlight);
    Insert_Rest_button_6.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
    Insert_Rest_button_6.setBounds(717, 390, 90, 30);
    Insert.add(Insert_Rest_button_6);

    // create the new JTextField about Textfield 'Insert_Rest_Menu_textField'.
    Insert_Rest_Menu_textField = new JTextField();
    Insert_Rest_Menu_textField.setColumns(18);
```

```
Insert_Rest_Menu_textField.setBounds(342, 262, 200, 35);
Insert.add(Insert_Rest_Menu_textField);

// create the new JTextField about Textfield 'Insert_Rest_Price_textField_1'.
Insert_Rest_Price_textField_1 = new JTextField();
Insert_Rest_Price_textField_1.setColumns(18);
Insert_Rest_Price_textField_1.setBounds(342, 305, 200, 35);
Insert.add(Insert_Rest_Price_textField_1);

JLabel label_7 = new JLabel("Calories");
label_7.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
label_7.setBounds(216, 345, 96, 35);
Insert.add(label_7);

// create the new JTextField about Textfield 'Insert_Rest_Calo_textField_1'.
Insert_Rest_Calo_textField_1 = new JTextField();
Insert_Rest_Calo_textField_1.setColumns(18);
Insert_Rest_Calo_textField_1.setBounds(342, 345, 200, 35);
Insert.add(Insert_Rest_Calo_textField_1);

JLabel label_8 = new JLabel("Category");
label_8.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
label_8.setBounds(216, 390, 90, 35);
Insert.add(label_8);

// create the new JTextField about Textfield 'Insert_Rest_Cate_textField_1'.
Insert_Rest_Cate_textField_1 = new JTextField();
Insert_Rest_Cate_textField_1.setColumns(18);
Insert_Rest_Cate_textField_1.setBounds(342, 390, 200, 35);
Insert.add(Insert_Rest_Cate_textField_1);

JLabel label_4 = new JLabel("Price");
label_4.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
label_4.setForeground(Color.RED);
label_4.setBounds(216, 305, 50, 35);
Insert.add(label_4);

JLabel label_5 = new JLabel("< Restaurants >");
label_5.setFont(new Font("맑은 고딕", Font.BOLD, 20));
```

```
label_5.setBounds(151, 165, 170, 21);
Insert.add(label_5);

JLabel label_6 = new JLabel("Menu");
label_6.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
label_6.setForeground(Color.RED);
label_6.setBounds(216, 262, 50, 35);
Insert.add(label_6);

JLabel lblName = new JLabel("Name");
lblName.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
lblName.setBounds(216, 212, 50, 35);
Insert.add(lblName);

JLabel lblExSweetPotato = new JLabel("ex) Sweet Potato Fries");
lblExSweetPotato.setFont(new Font("맑은 고딕", Font.ITALIC, 13));
lblExSweetPotato.setBounds(548, 262, 188, 35);
Insert.add(lblExSweetPotato);

JLabel lblEx = new JLabel("ex) 16900");
lblEx.setFont(new Font("맑은 고딕", Font.ITALIC, 13));
lblEx.setBounds(548, 305, 188, 35);
Insert.add(lblEx);

JLabel lblEx_1 = new JLabel("ex) 710");
lblEx_1.setFont(new Font("맑은 고딕", Font.ITALIC, 13));
lblEx_1.setBounds(548, 345, 188, 35);
Insert.add(lblEx_1);

JLabel lblExNoodle = new JLabel("ex) Noodles");
lblExNoodle.setFont(new Font("맑은 고딕", Font.ITALIC, 13));
lblExNoodle.setBounds(548, 390, 188, 35);
Insert.add(lblExNoodle);

JLabel lblRedRequired = new JLabel("**Red -> REQUIRED FIELD");
lblRedRequired.setForeground(Color.RED);
lblRedRequired.setFont(new Font("맑은 고딕", Font.BOLD, 15));
lblRedRequired.setBounds(341, 15, 259, 35);
Insert.add(lblRedRequired);
```

```
JLabel label = new JLabel("< Food Category >");
label.setFont(new Font("맑은 고딕", Font.BOLD, 20));
label.setBounds(151, 15, 200, 30);
Insert.add(label);

JLabel label_9 = new JLabel("Category");
label_9.setForeground(Color.RED);
label_9.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
label_9.setBounds(209, 60, 113, 35);
Insert.add(label_9);

// create the new JTextField about Textfield 'Insert_FoodCate_Cate_textField_1'.
Insert_FoodCate_Cate_textField_1 = new JTextField();
Insert_FoodCate_Cate_textField_1.setColumns(18);
Insert_FoodCate_Cate_textField_1.setBounds(335, 60, 200, 35);
Insert.add(Insert_FoodCate_Cate_textField_1);

JLabel label_10 = new JLabel("Ingredients");
label_10.setForeground(Color.RED);
label_10.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
label_10.setBounds(209, 108, 113, 35);
Insert.add(label_10);

// create the new JTextField about Textfield
// 'Insert_FoodCate_Ingre_textField_1'.
Insert_FoodCate_Ingre_textField_1 = new JTextField();
Insert_FoodCate_Ingre_textField_1.setColumns(18);
Insert_FoodCate_Ingre_textField_1.setBounds(335, 108, 200, 35);
Insert.add(Insert_FoodCate_Ingre_textField_1);

JLabel label_11 = new JLabel("ex) Wheat flour");
label_11.setFont(new Font("맑은 고딕", Font.ITALIC, 13));
label_11.setBounds(541, 108, 188, 35);
Insert.add(label_11);

JLabel label_22 = new JLabel("ex) Noodle");
label_22.setFont(new Font("맑은 고딕", Font.ITALIC, 13));
label_22.setBounds(541, 60, 188, 35);
```

```
Insert.add(label_22);

// create the button 'Insert_FoodCate_button_6' of new JButton.
JButton Insert_FoodCate_button_6 = new JButton("ENTER");
// add the action for button 'Insert_FoodCate_button_6'.
Insert_FoodCate_button_6.addActionListener(new ActionListener() {
    // insert the category and ingredients at Insert Panel when Administrator mode.
    public void actionPerformed(ActionEvent e) {
        // declare the category about 'Insert_FoodCate_Cate_textField_1' into String.
        String category = (String) Insert_FoodCate_Cate_textField_1.getText();
        // declare the ingredients about 'Insert_FoodCate_Cate_textField_1' into String.
        String ingredients = (String) Insert_FoodCate_Ingre_textField_1.getText();
        // if category enter the null value
        if (category.equals("") || ingredients.equals("")) {
            Insert_Result.setText("category, ingredients 를 모두 입력해주세요.");
            return; // try again the enter the value.
        }

        // create the 'resinsert' of new insert class.
        insert resinsert = new insert(category, ingredients);
        // insert using the method CategoryInsert()
        resinsert.CategoryInsert();
        /* clear textFields */
        Insert_FoodCate_Cate_textField_1.setText("");
        Insert_FoodCate_Ingre_textField_1.setText("");
        Insert_Result.setText("category insert 를 성공하였습니다.");
    }
});
Insert_FoodCate_button_6.setForeground(SystemColor.textHighlight);
Insert_FoodCate_button_6.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
Insert_FoodCate_button_6.setBounds(710, 108, 90, 30);
Insert.add(Insert_FoodCate_button_6);

// create the Panel 'Grades' of new JPanel
JPanel Grades = new JPanel();
Grades.setBackground(Color.WHITE);
// add tab about Grades Administrator mode at 'tabbedPane'.
tabbedPane.addTab("Grades", null, Grades, null);
Grades.setLayout(null);
```

```
        // create the button 'btnNewButton' of new JButton.
        JButton btnNewButton = new JButton("SHOW");
        // add the action for button 'btnNewButton'.
        btnNewButton.addActionListener(new ActionListener() {
            // show the grade of restaurants at Grade Panel when Administrator mode.
            public void actionPerformed(ActionEvent e) {
                // declare the tableName of "DBCOURSE_Grades"
                tableName = "DBCOURSE_Grades";
                // in this button, columns in colNames should be selected
                String colNames[] = { "name", "grades" };
                // create a object of class named 'retrieve_adm'
                retrieve_adm resgrade = new retrieve_adm(tableName, colNames);
                // execute 'show' method in 'retrieve_adm' and it becomes a value of 'table'
                JTable table;
                try {
                    table = resgrade.show(); // create a scrollpane named 'grades_scrollPane'
including data of 'table'
                    JScrollPane scrollPane = new JScrollPane(table);
                    scrollPane.setBounds(258, 145, 512, 245);
                    // panel named 'Grades' contains 'grades_scrollPane'
                    Grades.add(scrollPane);
                } catch (SQLException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                }

            }
        });
        btnNewButton.setFont(new Font("맑은 고딕", Font.PLAIN, 18));
        btnNewButton.setForeground(SystemColor.textHighlight);
        btnNewButton.setBounds(668, 85, 102, 29);
        Grades.add(btnNewButton);

        JLabel label_14 = new JLabel("<Grades>");
        label_14.setFont(new Font("맑은 고딕", Font.BOLD, 20));
        label_14.setBounds(258, 80, 112, 37);
        Grades.add(label_14);
```

```
    }

    /**
     * whichTable is method that is used when we know only restaurant's name, not a
     * table's name restaurant's name is changed into real table's name of database
     */
    public void whichTable() {
        String str = null, idx = null;
        // insert the str and idx when restaurant is selected.
        if (tableName.equals("Chienrong")) {
            str = "DBCOURSE_Rst1";
            idx = "i_price_rst1";
        } else if (tableName.equals("Dintaifung")) {
            str = "DBCOURSE_Rst2";
            idx = "i_price_rst2";
        } else if (tableName.equals("California Pizza Kitchen")) {
            str = "DBCOURSE_Rst3";
            idx = "i_price_rst3";
        } else if (tableName.equals("Haeundae Smokehouse")) {
            str = "DBCOURSE_Rst4";
            idx = "i_price_rst4";
        } else if (tableName.equals("JamaeGuksu")) {
            str = "DBCOURSE_Rst5";
            idx = "i_price_rst5";
        } else if (tableName.equals("HanKookJib")) {
            str = "DBCOURSE_Rst6";
            idx = "i_price_rst6";
        } else if (tableName.equals("ChungWoo")) {
            str = "DBCOURSE_Rst7";
            idx = "i_price_rst7";
        }
        tableName = str; // insert the str in tableName
        index = idx; // insert the idx in index
    }
}
```

D. Insert

```
package team2;
```

```
import java.sql.PreparedStatement;
import java.sql.SQLException;

/**
 * class that insert some data of request for administrator
 *
 * @author HyoJin Lee, NaGyeong Yeo
 *
 */
public class insert {
    static String tableName = null;
    static String name = null, menu = null, calories = null, category = null, ingredients = null;
    static int price = 0;

    /**
     * constructor to insert category
     *
     * @param category
     * @param ingredients
     */
    public insert(String category, String ingredients) {
        tableName = "DBCOURSE_FoodCategory";
        this.category = category;
        this.ingredients = ingredients;
    }

    /**
     * constructor to insert menu
     *
     * @param tableName
     * @param name
     * @param menu
     * @param price
     * @param calories
     * @param category
     */
    public insert(String tableName, String name, String menu, int price, String calories, String category) {
```

```
        this.tableName = tableName;
        this.name = name;
        this.menu = menu;
        this.price = price;
        this.calories = calories;
        this.category = category;
    }

    /**
     * insert new category(+ingredients) into FoodCategory table
     *
     * @param category
     * @param ingredients
     */
    public void CategoryInsert() {
        PreparedStatement pstmt = null;
        try {
            String sql = "insert into " + tableName + " values (?,?)"; // make query
            // create a object of prepareStatement that have '?' which can
            // be changed with each input of user
            pstmt = Open.myConn.prepareStatement(sql);
            /* input data into parameters */
            pstmt.setString(1, category);
            pstmt.setString(2, ingredients);
            pstmt.executeUpdate(); // execute query
            System.out.println(tableName + " 테이블에 새로운 레코드를 추가했습니다.");
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            if (pstmt != null) {
                try {
                    pstmt.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                    System.out.println(e.getMessage());
                }
            }
        }
    }
```

```java
/**
 * insert new menu(+price, etc) into specific Restaurant table
 *
 * @param tableName
 * @param name
 * @param menu
 * @param price
 * @param calories
 * @param category
 */
public void RestaurantInsert() {
    PreparedStatement pstmt = null;
    try {
        String sql = "insert into " + tableName + " values (?,?,?,?,?)"; // make query
        // create a object of prepareStatement that have '?' which can be changed with
        // each input of user
        pstmt = Open.myConn.prepareStatement(sql);
        /* input data into parameters */
        pstmt.setString(1, name);
        pstmt.setString(2, menu);
        pstmt.setInt(3, price);
        pstmt.setString(4, calories);
        pstmt.setString(5, category);
        pstmt.executeUpdate(); // execute query
        System.out.println(tableName + " 테이블에 새로운 레코드를 추가했습니다.");

    } catch (Exception e) {
        e.printStackTrace();
        System.out.println(e.getMessage());
        System.out.println("레코드 추가 실패");
    } finally {
        if (pstmt != null) {
            try {
                pstmt.close();
            } catch (SQLException e) {
                e.printStackTrace();
                System.out.println(e.getMessage());
            }
```

```
            }
        }
    }

}
```

E. Delete

```
package team2;

import java.sql.PreparedStatement;
import java.sql.SQLException;

/**
 * class that delete some data for request of administrator
 * @author HyoJin Lee
 *
 */
public class delete {

    static String menu = null;
    static int price_min = 0, price_max = 0, calory_min = 0, calory_max = 0;
    static String tableName = null;

    /**
     * constructor
     * @param tableName
     * @param menu
     */
    public delete(String tableName, String menu) {
        this.tableName = tableName;
        this.menu = menu;
    }

    /**
     * constructor
     * @param tableName
     * @param min
     * @param max
```

```
 */
public delete(String tableName, int min, int max) {
    this.tableName = tableName;

    this.price_min = min;
    this.price_max = max;

    this.calory_min = min;
    this.calory_max = max;
}

/**
 * delete the menu
 */
public void Restaurantdelete() {
    PreparedStatement pstmt=null;
    try {
        String sql = "delete from " + tableName + " where menu=?";
        //create a object of prepareStatement that have '?' which can
        // be changed with each input of user
        pstmt = Open.myConn.prepareStatement(sql);
        pstmt.setString(1, menu);//put the 'menu' to first question mark
        pstmt.executeUpdate();//plug in above parameters and execute update
    } catch (Exception e) {
        e.printStackTrace();
        System.out.println(e.getMessage());
    }finally {
        if (pstmt != null) {
            try {
                pstmt.close();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
                System.out.println(e.getMessage());
            }
        }

    }
}
```

```
/**
 *   delete tuples in the restaurant table that included in the price range
 */
public void Pricedelete() {
    PreparedStatement pstmt=null;
    try {
        String sql = "delete from " + tableName + " where price>=? and price<=?";
        //create a object of prepareStatement that have '?' which can
        // be changed with each input of user
        pstmt = Open.myConn.prepareStatement(sql);
        pstmt.setInt(1, price_min); //put the 'price_min' to first question mark
        pstmt.setInt(2, price_max);//put the 'price_max' to second question mark
        pstmt.executeUpdate();//plug in above parameters and execute update
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        System.out.println(e.getMessage());
    }finally {
        if (pstmt != null) {
            try {
                pstmt.close();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
                System.out.println(e.getMessage());
            }
        }

    }
}

/**
 * delete tuples in the restaurant table that included in the calories range
 */
public void Calorydelete() {
    PreparedStatement pstmt=null;
    try {
        String sql = "delete from " + tableName + " where calories>? and calories<?";
```

```
            //create a object of prepareStatement that have '?' which can
            // be changed with each input of user
            pstmt = Open.myConn.prepareStatement(sql);
            pstmt.setInt(1, calory_min); //put the 'calory_min' to first question mark
            pstmt.setInt(2, calory_max);//put the 'calory_max' to second question mark
            pstmt.executeUpdate();//plug in above parameters and execute update
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.out.println(e.getMessage());
        }finally {
            if (pstmt != null) {
                try {
                    pstmt.close();
                } catch (SQLException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                    System.out.println(e.getMessage());
                }
            }

        }
    }
}
```

F. Update

```
package team2;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;

/**
 * class that update some data for request of administrator
```

```
 * @author HyoJin Lee
 *
 */
public class update {
    static String name = null, menu = null, city = null, attraction = null, colNames[] = null;;
    static int price = 0;
    static String grades_new = null, grades_pre = null;
    static String tableName = null;

    /**
     * constructor
     * @param city
     * @param attraction
     */
    public update(String city, String attraction) {
        tableName = "DBCOURSE_Attraction";
        this.city = city;
        this.attraction = attraction;
    }

    /**
     * constructor
     * @param name
     * @param colNames
     * @param grades
     */
    public update(String name, String[] colNames, String grades) {
        this.name = name;
        this.colNames = colNames;
        this.grades_new = grades;
        tableName = "DBCOURSE_Grades";
    }

    /**
     * constructor
     * @param tableName
     * @param menu
     * @param price
     */
```

```
    public update(String tableName, String menu, int price) {
        this.tableName = tableName;
        this.menu = menu;
        this.price = price;
    }

    /**
     * update the price of menu
     */
    public void RestaurantUpdate() {
        PreparedStatement pstmt=null;
        try {
            String sql = "update " + tableName + " set price=? where menu=?";
            pstmt = Open.myConn.prepareStatement(sql);// create a object of prepareStatement
that have '?' which can
                                                                    // be changed with each input of
user
            pstmt.setInt(1, price); //put the 'price' to first question mark
            pstmt.setString(2, menu); //put the 'menu' to second question mark
            pstmt.executeUpdate(); //plug in above parameters and execute update
        } catch (Exception e) {
            e.printStackTrace();
            System.out.println(e.getMessage());
        }finally {
            if (pstmt != null) {
                try {
                    pstmt.close();
                } catch (SQLException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                    System.out.println(e.getMessage());
                }
            }

        }
    }

    /**
     * update the attraction of city
```

```
     */
    public void AttractionUpdate() {
        PreparedStatement pstmt=null;
        try {
            String sql = "update " + tableName + " set attraction=? where city=?";
            pstmt = Open.myConn.prepareStatement(sql);// create a object of prepareStatement
that have '?' which can
                                                                        // be changed with each input of
user
            pstmt.setString(1, attraction); //put the 'attraction' to first question mark
            pstmt.setString(2, city); //put the 'city' to second question mark
            pstmt.executeUpdate();//plug in above parameters and execute update
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.out.println(e.getMessage());
        }finally {
            if (pstmt != null) {
                try {
                    pstmt.close();
                } catch (SQLException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                    System.out.println(e.getMessage());
                }
            }

        }
    }

    /**
     * update the grades of each restaurant using transaction
     * @return table
     * @throws SQLException
     */
    @SuppressWarnings("finally")
    public JTable GradesUpdate() throws SQLException {
        JTable table = new JTable(); // generate new table object
```

```
        DefaultTableModel model = new DefaultTableModel(colNames, 0); //generate new model
object
        PreparedStatement pstmt1 = null;
        PreparedStatement pstmt2 = null;
        PreparedStatement pstmt3 = null;
        ResultSet myRs = null;
        Statement st = null;
        String sql1 = "select grades from " + tableName + " where name=?";
        String sql2 = "update " + tableName + " set grades=? where name=?";
        String sql3 = "update " + tableName + " set grades=? where name=?";
        String sql4 = "select * from " + tableName;
        try {
            Open.myConn.setAutoCommit(false); //turn off automatic commit on a connection
            pstmt1 = Open.myConn.prepareStatement(sql1);// create a object of prepareStatement
that have '?' which can
                                                // be changed with each input of user
            pstmt2 = Open.myConn.prepareStatement(sql2);// create a object of prepareStatement
that have '?' which can
                                                // be changed with each input of user
            pstmt3 = Open.myConn.prepareStatement(sql3);// create a object of prepareStatement
that have '?' which can
                                                // be changed with each input of user
            st = Open.myConn.createStatement();// create a object of Statement
            pstmt1.setString(1, name);//put the 'name' to pstmt1's first question mark
            myRs = pstmt1.executeQuery();//plug in above parameters and execute
            while (myRs.next()) { //approach every record in the table
                grades_pre = myRs.getString(colNames[1]); // get the grades value
            }
            float grades = (Float.parseFloat(grades_pre) + Float.parseFloat(grades_new)); //get the
sum of previous grades, new grades and save to grades
            pstmt2.setFloat(1, grades);//put the 'grades' to pstmt2's first question mark
            pstmt2.setString(2, name);//put the 'name' to pstmt2's second question mark
            pstmt2.executeUpdate();//plug in above parameters and execute update

            pstmt3.setFloat(1, grades / 2);//put the 'grades/2'(average) to pstmt2's first question
mark
            pstmt3.setString(2, name);//put the 'name' to pstmt2's second question mark
            pstmt3.executeUpdate();//plug in above parameters and execute update
```

```
            myRs = st.executeQuery(sql4); //execute query and fetch
            while (myRs.next()) { //approach every record in the table
                model.addRow(new Object[] { myRs.getString(colNames[0]),
myRs.getString(colNames[1]) }); // get all tuples and save to model
            }
            table = new JTable(model);// table becomes a JTable objects with contents of the model
            Open.myConn.commit(); //transactions committed
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.out.println(e.getMessage());
            if (Open.myConn != null) {
                    try {
                        System.err.print("Transaction is being rolled back"); //show the error message
that transactions failed
                        Open.myConn.rollback(); //transactions rolled back
                    } catch(SQLException ex) {
                       ex.printStackTrace();
                     System.out.println(ex.getMessage());
                    }
               }

        } finally {
            if (pstmt1 != null) {
                pstmt1.close();
            }
            if (pstmt2 != null) {
                pstmt2.close();
            }
            if (pstmt3 != null) {
                pstmt3.close();
            }
            if (st != null) {
                st.close();
            }
            Open.myConn.setAutoCommit(true);//turn on automatic commit
            return table; //return value is table
        }
    }
```

```
}
```

G. retrieve_cus

```
package team2;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;

/**
 * class that retrieve the result for request of customer
 *
 * @author NaGyeong Yeo, Dokyung Lee, HyoJin Lee
 *
 */
public class retrieve_cus {
    static String colNames[] = null;
    static String tableName = null, type = null, name = null;
    static int priceMin, priceMax;

    /**
     * constructor to retrieve data by using type
     *
     * @param tableName
     * @param type
     * @param colNames
     */
    public retrieve_cus(String tableName, String type, String[] colNames) {
        this.tableName = tableName;
        this.type = type;
        this.colNames = colNames;
    }

    /**
```

```
 * constructor that has String, String array, and integer parameters
 *
 * @param name
 * @param colNames
 * @param a
 */
public retrieve_cus(String name, String[] colNames, int a) {
    this.name = name;
    this.colNames = colNames;
}

/**
 * constructor to retrieve data by using price range
 *
 * @param tableName
 * @param priceMin
 * @param priceMax
 * @param colNames
 */
public retrieve_cus(String tableName, int priceMin, int priceMax, String[] colNames) {
    this.tableName = tableName;
    this.priceMin = priceMin;
    this.priceMax = priceMax;
    this.colNames = colNames;
}

/**
 * retrieve restaurant and menu information by using type / retrieve restaurant
 * and menu information by using price range
 *
 * @return
 */
public JTable show() {
    JTable table = new JTable();
    DefaultTableModel model = new DefaultTableModel(colNames, 0);
    PreparedStatement pstmt = null;
    try {
        int num = 0;
        String sql;
```

```
            if (tableName == "DBCOURSE_type") { // retrieve restaurant and menu information by using type
                num = 3; // the number of columns is 3
                sql = "select name, menu, price from " + tableName + " where type=?"; // make query
                // create a object of prepareStatement that have '?'
                // which can be changed with each input of user
                pstmt = Open.myConn.prepareStatement(sql);
                /* input data into parameter */
                pstmt.setString(1, type);
            } else if (tableName == "DBCOURSE_menu, DBCOURSE_foodCategory") { // retrieve restaurant and menu
                // information by using price range
                num = 6; // the number of columns is 6
                // make query
                sql = "select name, menu, price, calories, DBCOURSE_menu.category, ingredients from " + tableName
                        + " where DBCOURSE_menu.category=DBCOURSE_foodCategory.category and price >= ? and price <=?";
                pstmt = Open.myConn.prepareStatement(sql); // create a object of prepareStatement that have '?'
                // which can be changed with each input of user
                /* input data into parameters */
                pstmt.setInt(1, priceMin);
                pstmt.setInt(2, priceMax);
            }

            ResultSet myRs = pstmt.executeQuery(); // execute query

            /* make table according to the number of columns */
            while (myRs.next()) {
                switch (num) {
                case 2:
                    model.addRow(new Object[] { myRs.getString(colNames[0]), myRs.getString(colNames[1]) });
                    break;
                case 3:
                    model.addRow(new Object[] { myRs.getString(colNames[0]), myRs.getString(colNames[1]),
```

```
                    myRs.getString(colNames[2]) });
                break;
            case 4:
                model.addRow(new Object[] { myRs.getString(colNames[0]),
myRs.getString(colNames[1]),
                    myRs.getString(colNames[2]), myRs.getString(colNames[3]) });
                break;
            case 5:
                model.addRow(new Object[] { myRs.getString(colNames[0]),
myRs.getString(colNames[1]),
                    myRs.getString(colNames[2]), myRs.getString(colNames[3]),
myRs.getString(colNames[4]) });
                break;
            case 6:
                model.addRow(new Object[] { myRs.getString(colNames[0]),
myRs.getString(colNames[1]),
                    myRs.getString(colNames[2]), myRs.getString(colNames[3]),
myRs.getString(colNames[4]),
                    myRs.getString(colNames[5]) });
                break;
            }
        }
        table = new JTable(model); // table becomes a JTable objects with contents of the model

    } catch (SQLException e) { // catch errors
        e.printStackTrace();
        System.out.println(e.getMessage());
    } finally {
        if (pstmt != null) {
            try {
                pstmt.close();
            } catch (SQLException e) {
                e.printStackTrace();
                System.out.println(e.getMessage());
            }
        }
    }

    return table; // return table of the result
```

```
    }

    /**
     * method for customer that returns a table that is a result of query
     *
     * @return JTable = output of the query
     */
    public JTable show4() {

        JTable table = new JTable(); // 'table' is a object of JTable
        DefaultTableModel model = new DefaultTableModel(colNames, 0); // Constructs a
DefaultTableModel with columns
        // that are elements in columnNames and the
        // number of rows is '0'
        PreparedStatement pstmt1 = null;

        try {
            // nested query is used in this query
            String sql1 = "select city, attraction from DBCOURSE_Attraction use index (i_city) where
city in (select city from DBCOURSE_Restaurants use index (i_name) where name=?)";

            // create a object of prepareStatement that have '?' which can
            // be changed with each input of user
            pstmt1 = Open.myConn.prepareStatement(sql1);
            pstmt1.setString(1, name); // first '?' is the same with value of 'name'

            ResultSet myRs = pstmt1.executeQuery(); // execute query in 'pstmt1' and 'myRs'
becomes that result

            while (myRs.next()) { // read ResultSet 'myRs' line by line
                // domain of 'colNames' becomes 'Object' array and also rows of 'tableName'
                model.addRow(new Object[] { myRs.getString(colNames[0]),
myRs.getString(colNames[1]) });
            }
            table = new JTable(model); // table becomes a JTable objects with contents of the model

        } catch (SQLException e) { // catch errors
            e.printStackTrace();
            System.out.println(e.getMessage());
```

```
        } finally {
            if (pstmt1 != null) {
                try {
                    pstmt1.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                    System.out.println(e.getMessage());
                }
            }
        }

        return table; // return table of the result
    }
}
```

H. retrieve_adm

```
package team2;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;

/**
 * class that retrieve the result for request of administrator
 * @author DoKyung Lee
 *
 */
public class retrieve_adm {

    static String colNames[] = null;
    static String tableName = null;
    static String index = null;
    DefaultTableModel model = null;
```

```
    /**
     * constructor that has String and String array parameters
     * @param tableName
     * @param colNames
     */
    public retrieve_adm(String tableName, String[] colNames) {
        this.tableName = tableName;
        this.colNames = colNames;
    }

    /**
     * constructor that has 2 Strings and String array parameters
     * @param tableName
     * @param index
     * @param colNames
     */
    public retrieve_adm(String tableName, String index, String[] colNames) {
        this.tableName = tableName;
        this.index = index;
        this.colNames = colNames;

    }

    /**
     * method for administrator that returns a table that is a result of query
     * @return
     * @throws SQLException
     */
    public JTable show() throws SQLException {
        JTable table = new JTable(); // 'table' is a object of JTable
        model = new DefaultTableModel(colNames, 0); // Constructs a DefaultTableModel with
columns that are elements in columnNames and the number of rows is '0'
        Statement st = null;

        try {
            int num = 0; // the number of columns that have to be shown
            st = Open.myConn.createStatement(); //create a object of Statement
            String sql = null;
            ResultSet myRs = null; //create a object of ResultSet
```

```
            // condition statement about 'tableName' shows the whole table
            if (tableName == "DBCOURSE_Restaurants") {
                num = 4; // the number of columns that have to be shown is 4
                sql = "select * from " + tableName + " use index (i_name)"; // use index in query
                myRs = st.executeQuery(sql); // execute query that is same as 'sql' and 'myRs' becomes that result
            } else if (tableName == "DBCOURSE_Attraction") { // second option and comments are same with 'if' part
                num = 2;
                sql = "select * from " + tableName + " use index (i_city)";
                myRs = st.executeQuery(sql);
            } else if (tableName == "DBCOURSE_FoodCategory") { // third option and comments are same with 'if' part
                num = 2;
                sql = "select * from " + tableName + " use index (i_category)";
                myRs = st.executeQuery(sql);
            } else if (tableName == "DBCOURSE_Grades") { // fourth option and comments are same with 'if' part
                num = 2;
                sql = "select * from " + tableName;
                myRs = st.executeQuery(sql);
            } else { // fifth option and comments are same with 'if' part
                num = 5;
                sql = "select * from " + tableName + " use index (" + index + ")";
                myRs = st.executeQuery(sql);
            }

            while (myRs.next()) { // read ResultSet 'myRs' line by line
                switch (num) { // 3 cases are divided depending on value of 'num'
                case 4: // first case - the result data of query occupies domain of column in the model
                    model.addRow(new Object[] { myRs.getString(colNames[0]), myRs.getString(colNames[1]),
                        myRs.getString(colNames[2]), myRs.getString(colNames[3]) }); // domain of 'colNames' becomes 'Object' array and also rows of 'tableName'
                    break;
                case 2: // second case
```

```
                    model.addRow(new Object[] { myRs.getString(colNames[0]),
myRs.getString(colNames[1]) });
                    break;
                case 5: // third case
                    model.addRow(new Object[] { myRs.getString(colNames[0]),
myRs.getString(colNames[1]),
                            myRs.getString(colNames[2]), myRs.getString(colNames[3]),
myRs.getString(colNames[4]) });
                    break;
                }
            }
            table = new JTable(model); // table becomes a JTable objects with contents of the model

        } catch (SQLException e) {   //catch errors
            e.printStackTrace();
            System.out.println(e.getMessage());
        } finally {
                if (st != null) {
                    st.close(); // close the statement
                }
                return table;   //return table of the result
            }
    }
}
```