RestCheque

Lecturer

Ufuk Çelikkan

Sercan Kavdır - 20160602136

Emre Serbest - 20160602115

Objectives of the Software

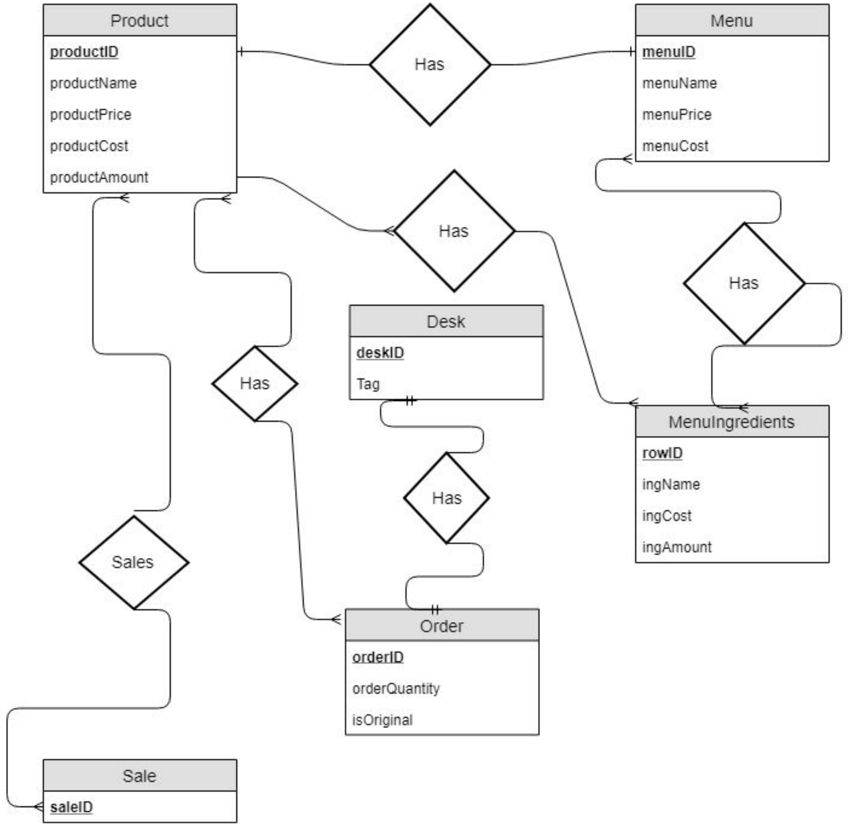
- The owner shall be able to determine the capacity of the restaurant(cafe,bar, etc.), and then, by creating new tables, the user will be able to take the orders from separate tables. Also, the user can delete the table.
- The user will be able to enter the orders to the specific table, and also, delete the orders, will be supported by the program.
- The user can add multiple products as ingredients(potato, chicken, sauces for a dinner or other products that cannot be sold by itself) or by determining the price of a product that can be sold by its own such as coke, water and other drinks.
- When a customer gives an order, the stock of those products will be decreasing according to the order.

Tools

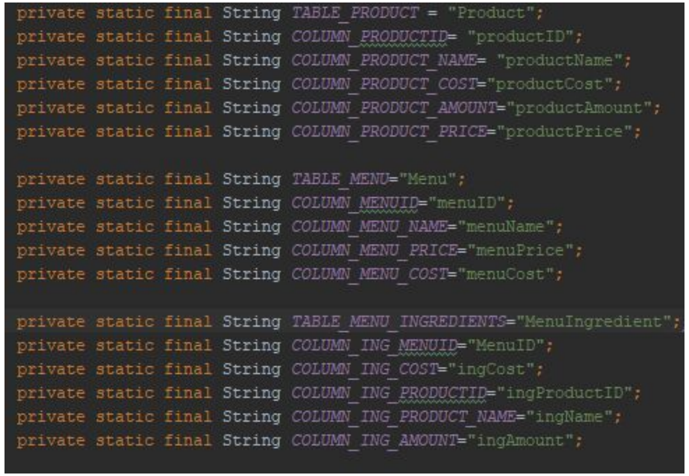
- For the design of GUI, we used JavaFX
- With the help visual layout tool(SceneBuilder), we design our application interface
- In the database part SQLite used. Also, for visual changes DB Browser made us clear
- As an IDE, we used IntelliJ IDEA

Database(ER Diagram)

- In this figure, we can see how the database is organized in our software

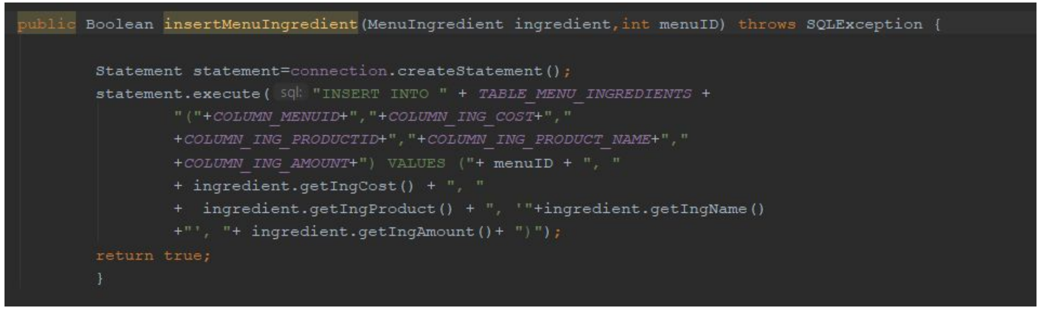


Database(Coding Part)



```java
private static final String TABLE_PRODUCT = "Product";
private static final String COLUMN_PRODUCTID= "productID";
private static final String COLUMN_PRODUCT_NAME= "productName";
private static final String COLUMN_PRODUCT_COST="productCost";
private static final String COLUMN_PRODUCT_AMOUNT="productAmount";
private static final String COLUMN_PRODUCT_PRICE="productPrice";

private static final String TABLE_MENU="Menu";
private static final String COLUMN_MENUID="menuID";
private static final String COLUMN_MENU_NAME="menuName";
private static final String COLUMN_MENU_PRICE="menuPrice";
private static final String COLUMN_MENU_COST="menuCost";

private static final String TABLE_MENU_INGREDIENTS="MenuIngredient";
private static final String COLUMN_ING_MENUID="MenuID";
private static final String COLUMN_ING_COST="ingCost";
private static final String COLUMN_ING_PRODUCTID="ingProductID";
private static final String COLUMN_ING_PRODUCT_NAME="ingName";
private static final String COLUMN_ING_AMOUNT="ingAmount";
```

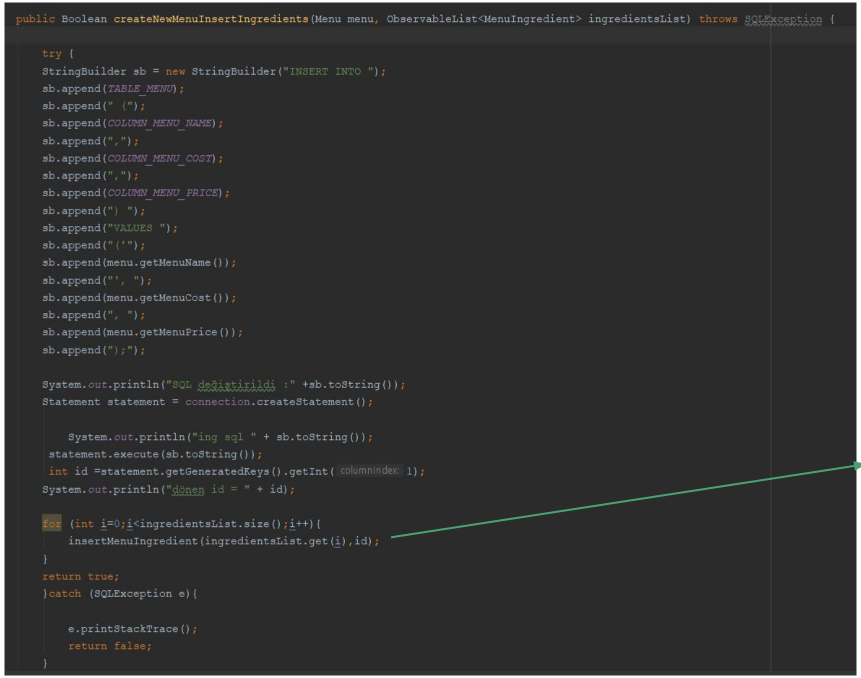
Sample of basic table organization



```java
public Boolean insertMenuIngredient(MenuIngredient ingredient,int menuID) throws SQLException {

        Statement statement=connection.createStatement();
        statement.execute( sql: "INSERT INTO " + TABLE_MENU_INGREDIENTS +
                "("+COLUMN_MENUID+","+COLUMN_ING_COST+","
                +COLUMN_ING_PRODUCTID+","+COLUMN_ING_PRODUCT_NAME+","
                +COLUMN_ING_AMOUNT+") VALUES ("+ menuID + ", "
                + ingredient.getIngCost() + ", "
                +  ingredient.getIngProduct() + ", '"+ingredient.getIngName()
                +"', "+ ingredient.getIngAmount()+ ")");
        return true;
        }
```

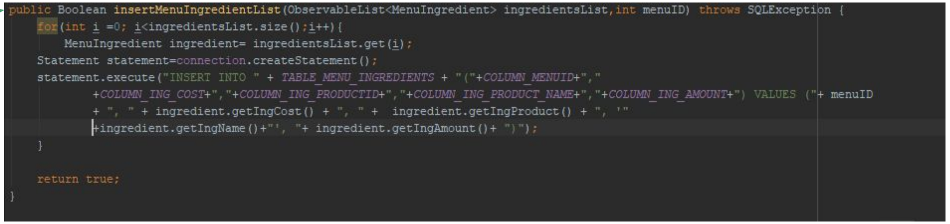
Example of inserting MenuIngredients table

Database(Coding Part)



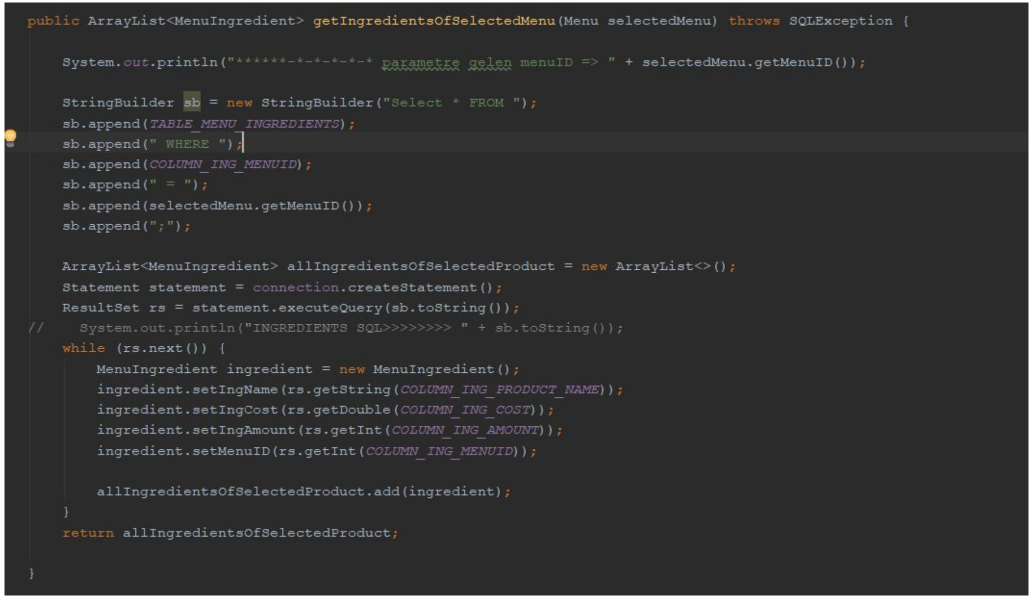
```java
public Boolean createNewMenuInsertIngredients(Menu menu, ObservableList<MenuIngredient> ingredientsList) throws SQLException {

    try {
    StringBuilder sb = new StringBuilder("INSERT INTO ");
    sb.append(TABLE_MENU);
    sb.append(" (");
    sb.append(COLUMN_MENU_NAME);
    sb.append(",");
    sb.append(COLUMN_MENU_COST);
    sb.append(",");
    sb.append(COLUMN_MENU_PRICE);
    sb.append(") ");
    sb.append("VALUES ");
    sb.append("('");
    sb.append(menu.getMenuName());
    sb.append("', ");
    sb.append(menu.getMenuCost());
    sb.append(", ");
    sb.append(menu.getMenuPrice());
    sb.append(");");

    System.out.println("SQL değiştirildi :" +sb.toString());
    Statement statement = connection.createStatement();

        System.out.println("ing sql " + sb.toString());
     statement.execute(sb.toString());
     int id =statement.getGeneratedKeys().getInt( columnIndex: 1);
    System.out.println("dönen id = " + id);

    for (int i=0;i<ingredientsList.size();i++){
        insertMenuIngredient(ingredientsList.get(i),id);
    }
    return true;
    }catch (SQLException e){

        e.printStackTrace();
        return false;
    }
```

Firstly, we insert the menu into menu table and we return ID of the last inserted row from the menuTable then we use this ID while we insert ingredients of the menu into ingredientsTable



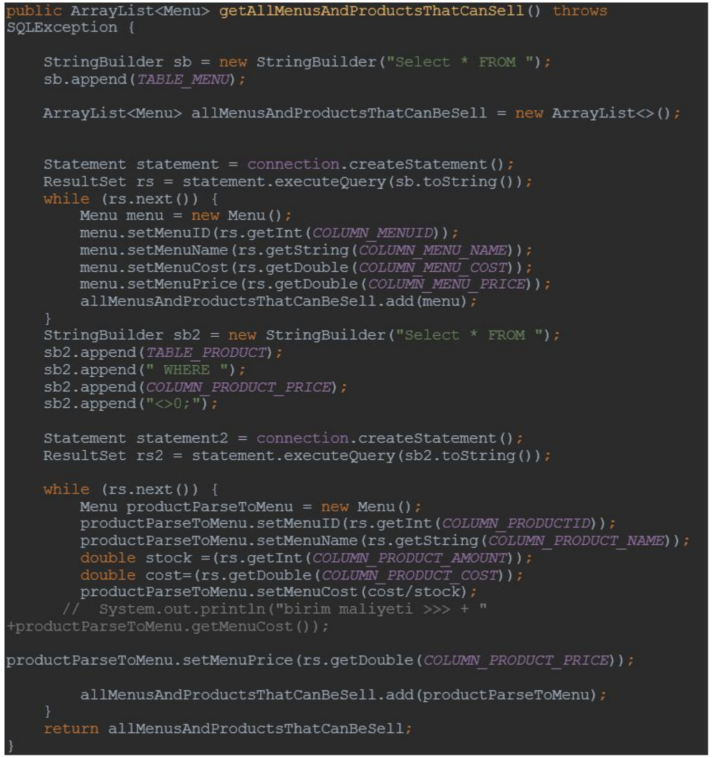
```java
public Boolean insertMenuIngredientList(ObservableList<MenuIngredient> ingredientsList,int menuID) throws SQLException {
    for(int i =0; i<ingredientsList.size();i++){
        MenuIngredient ingredient= ingredientsList.get(i);
    Statement statement=connection.createStatement();
    statement.execute("INSERT INTO " + TABLE_MENU_INGREDIENTS + "("+COLUMN_MENUID+","
            +COLUMN_ING_COST+","+COLUMN_ING_PRODUCTID+","+COLUMN_ING_PRODUCT_NAME+","+COLUMN_ING_AMOUNT+") VALUES ("+ menuID
            + ", " + ingredient.getIngCost() + ", " +  ingredient.getIngProduct() + ", '"
            +ingredient.getIngName()+"', "+ ingredient.getIngAmount()+ ")");
    }

    return true;
}
```

Database(Coding Part)



```java
public ArrayList<MenuIngredient> getIngredientsOfSelectedMenu(Menu selectedMenu) throws SQLException {

    System.out.println("******-*-*-*-*-* parametre gelen menuID => " + selectedMenu.getMenuID());

    StringBuilder sb = new StringBuilder("Select * FROM ");
    sb.append(TABLE_MENU_INGREDIENTS);
    sb.append(" WHERE ");
    sb.append(COLUMN_ING_MENUID);
    sb.append(" = ");
    sb.append(selectedMenu.getMenuID());
    sb.append(";");

    ArrayList<MenuIngredient> allIngredientsOfSelectedProduct = new ArrayList<>();
    Statement statement = connection.createStatement();
    ResultSet rs = statement.executeQuery(sb.toString());
//    System.out.println("INGREDIENTS SQL>>>>>>>> " + sb.toString());
    while (rs.next()) {
        MenuIngredient ingredient = new MenuIngredient();
        ingredient.setIngName(rs.getString(COLUMN_ING_PRODUCT_NAME));
        ingredient.setIngCost(rs.getDouble(COLUMN_ING_COST));
        ingredient.setIngAmount(rs.getInt(COLUMN_ING_AMOUNT));
        ingredient.setMenuID(rs.getInt(COLUMN_ING_MENUID));

        allIngredientsOfSelectedProduct.add(ingredient);
    }
    return allIngredientsOfSelectedProduct;

}
```

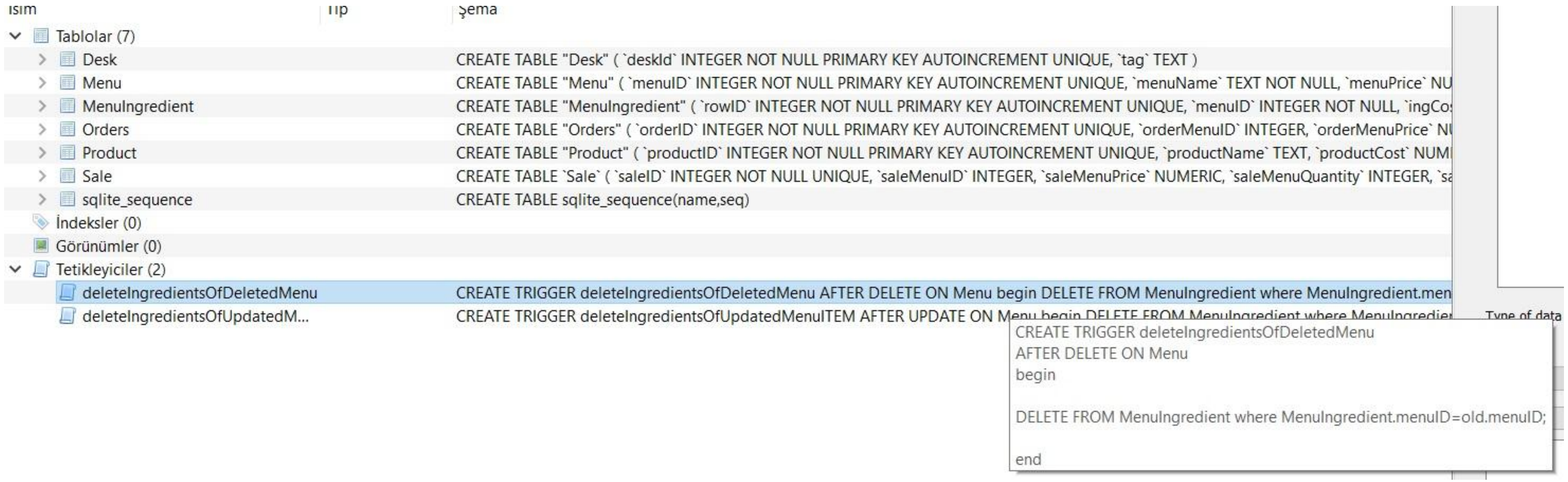
We returned ingredients of the selected menu



```java
public ArrayList<Menu> getAllMenusAndProductsThatCanSell() throws
SQLException {

    StringBuilder sb = new StringBuilder("Select * FROM ");
    sb.append(TABLE_MENU);

    ArrayList<Menu> allMenusAndProductsThatCanBeSell = new ArrayList<>();


    Statement statement = connection.createStatement();
    ResultSet rs = statement.executeQuery(sb.toString());
    while (rs.next()) {
        Menu menu = new Menu();
        menu.setMenuID(rs.getInt(COLUMN_MENUID));
        menu.setMenuName(rs.getString(COLUMN_MENU_NAME));
        menu.setMenuCost(rs.getDouble(COLUMN_MENU_COST));
        menu.setMenuPrice(rs.getDouble(COLUMN_MENU_PRICE));
        allMenusAndProductsThatCanBeSell.add(menu);
    }
    StringBuilder sb2 = new StringBuilder("Select * FROM ");
    sb2.append(TABLE_PRODUCT);
    sb2.append(" WHERE ");
    sb2.append(COLUMN_PRODUCT_PRICE);
    sb2.append("<>0;");

    Statement statement2 = connection.createStatement();
    ResultSet rs2 = statement.executeQuery(sb2.toString());

    while (rs.next()) {
        Menu productParseToMenu = new Menu();
        productParseToMenu.setMenuID(rs.getInt(COLUMN_PRODUCTID));
        productParseToMenu.setMenuName(rs.getString(COLUMN_PRODUCT_NAME));
        double stock =(rs.getInt(COLUMN_PRODUCT_AMOUNT));
        double cost=(rs.getDouble(COLUMN_PRODUCT_COST));
        productParseToMenu.setMenuCost(cost/stock);
      //  System.out.println("birim maliyeti >>> + "
+productParseToMenu.getMenuCost());

productParseToMenu.setMenuPrice(rs.getDouble(COLUMN_PRODUCT_PRICE));

        allMenusAndProductsThatCanBeSell.add(productParseToMenu);
    }
    return allMenusAndProductsThatCanBeSell;
}
```

If the price of the product is greater than 0 it means that the user can sell this product, otherwise the product becomes an ingredient and cannot be sold by itself

Trigger

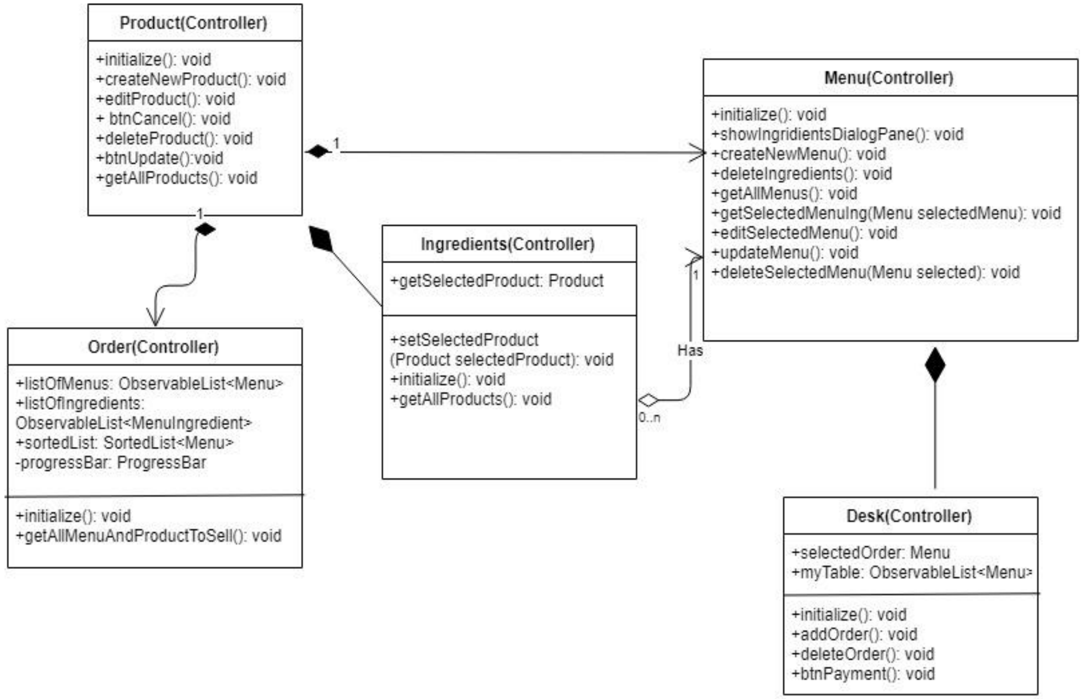


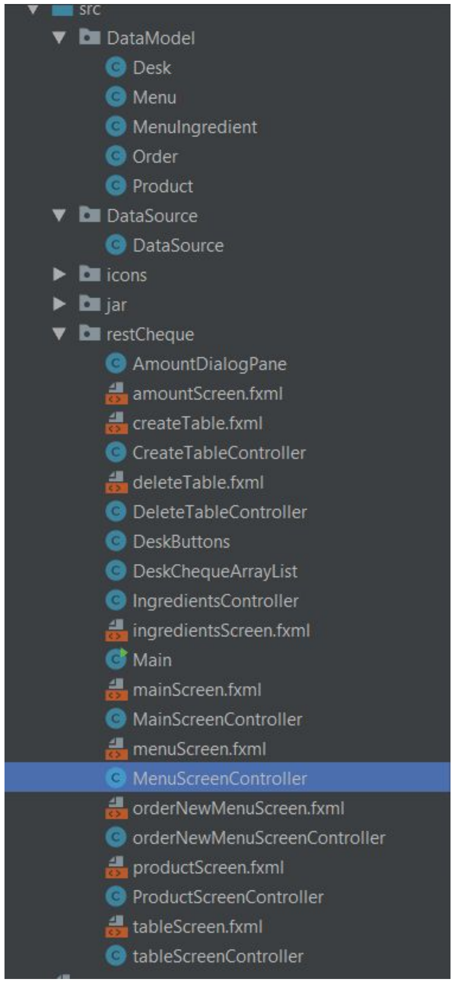
İsim	Tip	Şema
Tablolar (7)		
Desk		CREATE TABLE "Desk" (`deskId` INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE, `tag` TEXT)
Menu		CREATE TABLE "Menu" (`menuID` INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE, `menuName` TEXT NOT NULL, `menuPrice` NU
MenuIngredient		CREATE TABLE "MenuIngredient" (`rowID` INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE, `menuID` INTEGER NOT NULL, `ingCo
Orders		CREATE TABLE "Orders" (`orderID` INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE, `orderMenuID` INTEGER, `orderMenuPrice` N
Product		CREATE TABLE "Product" (`productID` INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE, `productName` TEXT, `productCost` NUM
Sale		CREATE TABLE `Sale` (`saleID` INTEGER NOT NULL UNIQUE, `saleMenuID` INTEGER, `saleMenuPrice` NUMERIC, `saleMenuQuantity` INTEGER, `s
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
İndeksler (0)		
Görünümler (0)		
Tetikleyiciler (2)		
deleteIngredientsOfDeletedMenu		CREATE TRIGGER deleteIngredientsOfDeletedMenu AFTER DELETE ON Menu begin DELETE FROM MenuIngredient where MenuIngredient.men
deleteIngredientsOfUpdatedM...		CREATE TRIGGER deleteIngredientsOfUpdatedMenuITEM AFTER UPDATE ON Menu begin DELETE FROM MenuIngredient where MenuIngredie

```
CREATE TRIGGER deleteIngredientsOfDeletedMenu
AFTER DELETE ON Menu
begin

DELETE FROM MenuIngredient where MenuIngredient.menuID=old.menuID;

end
```

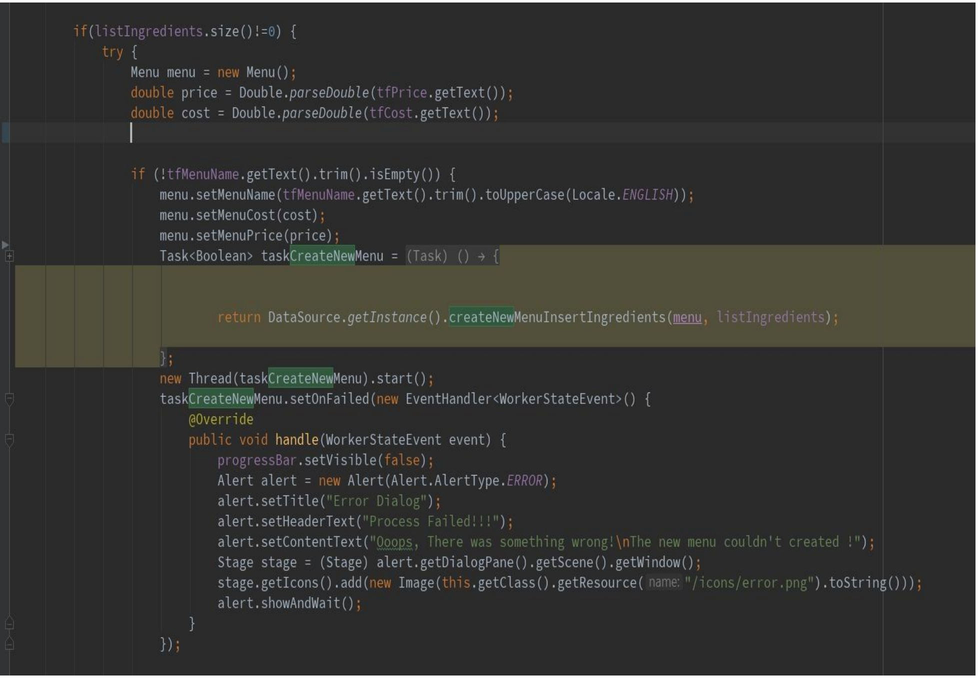
After deleting or updating menu, then with the trigger it deletes or updates ingredients of menu

Class Diagram

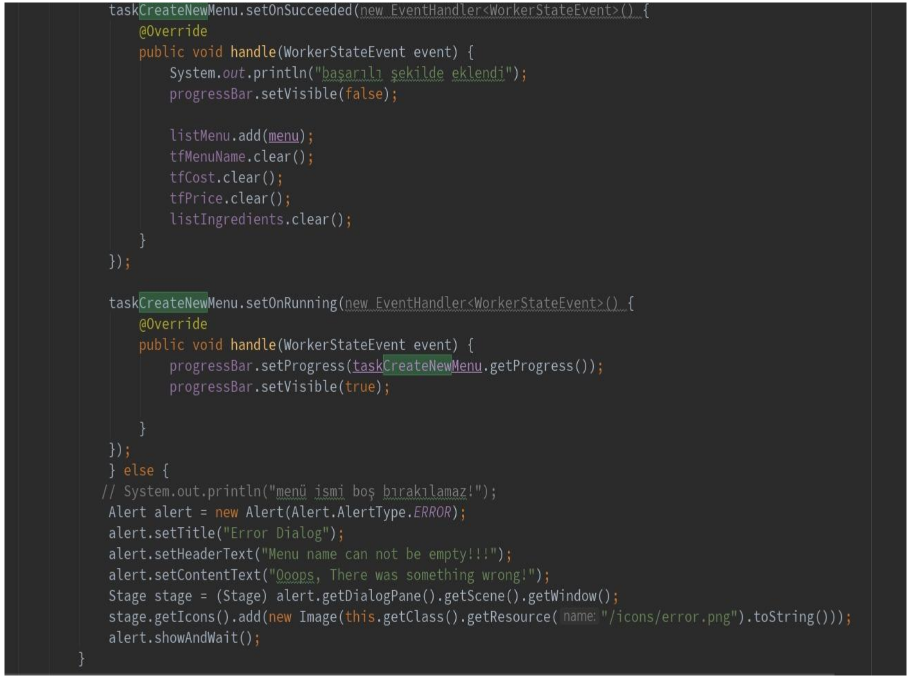



```
src
  DataModel
    Desk
    Menu
    MenuIngredient
    Order
    Product
  DataSource
    DataSource
  icons
  jar
  restCheque
    AmountDialogPane
    amountScreen.fxml
    createTable.fxml
    CreateTableController
    deleteTable.fxml
    DeleteTableController
    DeskButtons
    DeskChequeArrayList
    IngredientsController
    ingredientsScreen.fxml
    Main
    mainScreen.fxml
    MainScreenController
    menuScreen.fxml
    MenuScreenController
    orderNewMenuScreen.fxml
    orderNewMenuScreenController
    productScreen.fxml
    ProductScreenController
    tableScreen.fxml
    tableScreenController
```

Thread



```
if(listIngredients.size()!=0) {
    try {
        Menu menu = new Menu();
        double price = Double.parseDouble(tfPrice.getText());
        double cost = Double.parseDouble(tfCost.getText());


        if (!tfMenuName.getText().trim().isEmpty()) {
            menu.setMenuName(tfMenuName.getText().trim().toUpperCase(Locale.ENGLISH));
            menu.setMenuCost(cost);
            menu.setMenuPrice(price);
            Task<Boolean> taskCreateNewMenu = (Task) () → {


                return DataSource.getInstance().createNewMenuInsertIngredients(menu, listIngredients);

            };
            new Thread(taskCreateNewMenu).start();
            taskCreateNewMenu.setOnFailed(new EventHandler<WorkerStateEvent>() {
                @Override
                public void handle(WorkerStateEvent event) {
                    progressBar.setVisible(false);
                    Alert alert = new Alert(Alert.AlertType.ERROR);
                    alert.setTitle("Error Dialog");
                    alert.setHeaderText("Process Failed!!!");
                    alert.setContentText("Ooops, There was something wrong!\nThe new menu couldn't created !");
                    Stage stage = (Stage) alert.getDialogPane().getScene().getWindow();
                    stage.getIcons().add(new Image(this.getClass().getResource( name: "/icons/error.png").toString()));
                    alert.showAndWait();
                }
            });
```

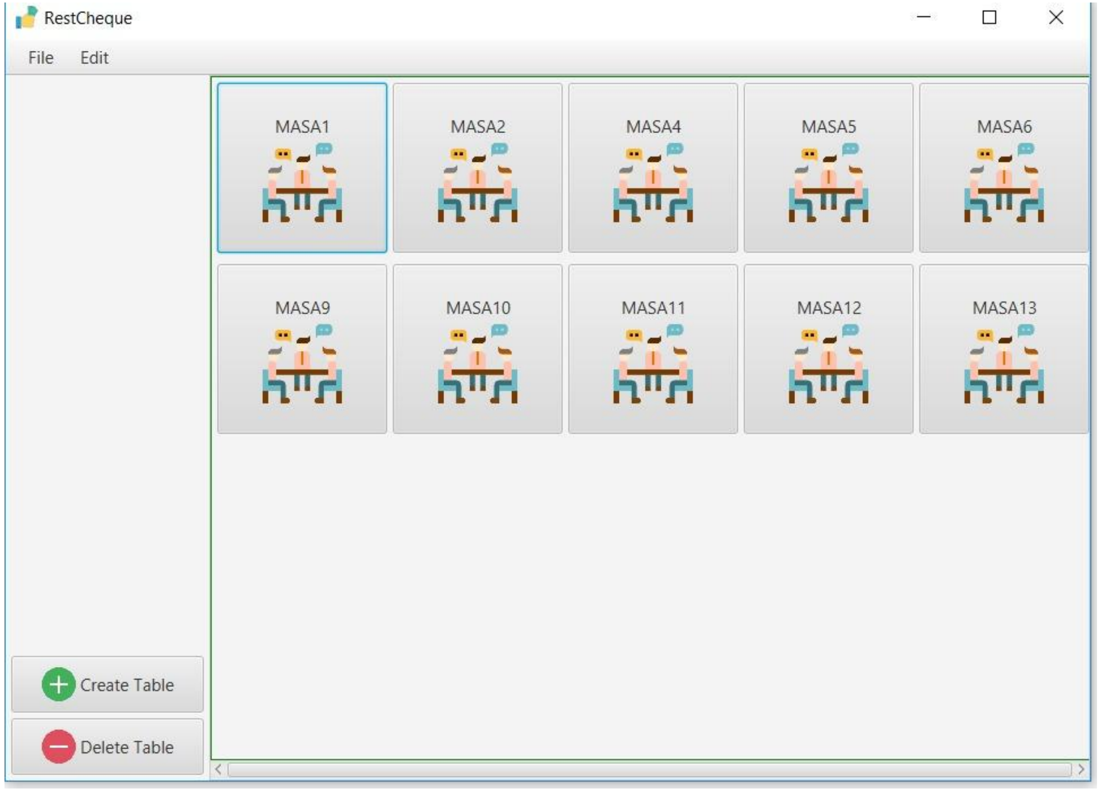


```
        taskCreateNewMenu.setOnSucceeded(new EventHandler<WorkerStateEvent>() {
            @Override
            public void handle(WorkerStateEvent event) {
                System.out.println("başarılı şekilde eklendi");
                progressBar.setVisible(false);

                listMenu.add(menu);
                tfMenuName.clear();
                tfCost.clear();
                tfPrice.clear();
                listIngredients.clear();
            }
        });

        taskCreateNewMenu.setOnRunning(new EventHandler<WorkerStateEvent>() {
            @Override
            public void handle(WorkerStateEvent event) {
                progressBar.setProgress(taskCreateNewMenu.getProgress());
                progressBar.setVisible(true);


            }
        });
        } else {
       // System.out.println("menü ismi boş bırakılamaz!");
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error Dialog");
        alert.setHeaderText("Menu name can not be empty!!!");
        alert.setContentText("Ooops, There was something wrong!");
        Stage stage = (Stage) alert.getDialogPane().getScene().getWindow();
        stage.getIcons().add(new Image(this.getClass().getResource( name: "/icons/error.png").toString()));
        alert.showAndWait();
    }
```

We implemented threads, to create, delete, update menu and products and we showed progress bar while the process is running.

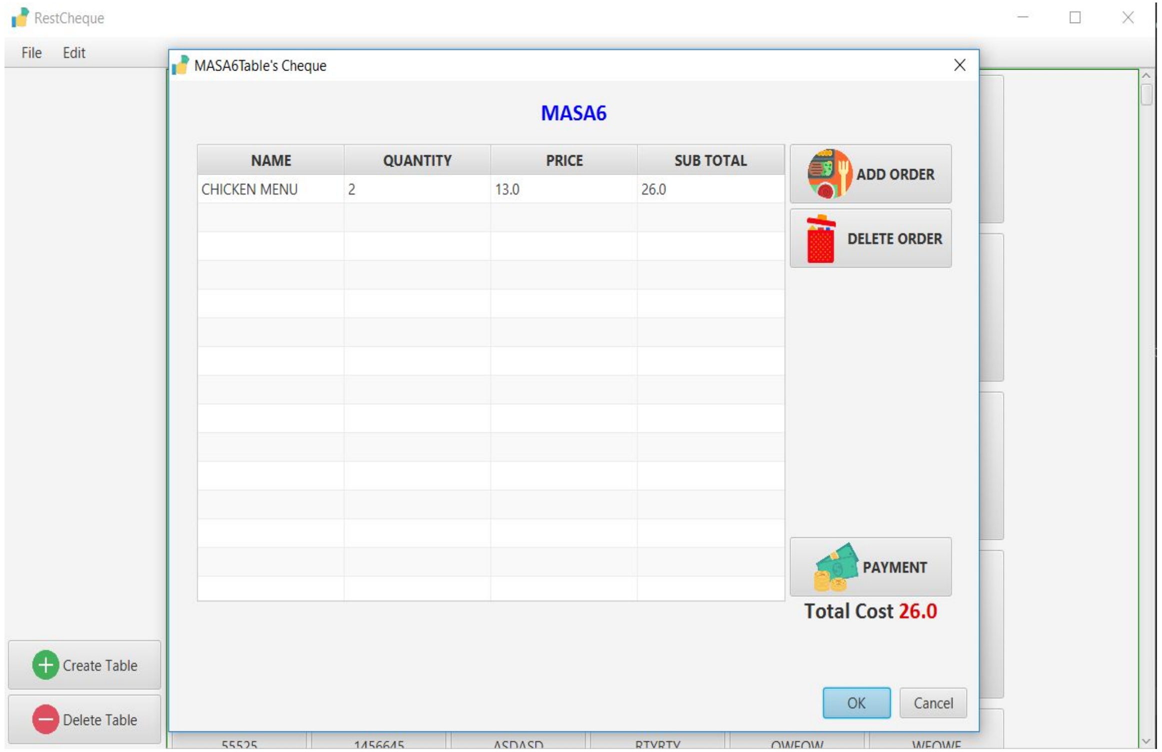
GUI

- The owner creates tables of the restaurant and also the owner can delete the tables
- By clicking to the specific table, the orders can be seen in another dialog pane



GUI

- Here is the orders of a specific table


RestCheque

MASA6Table's Cheque

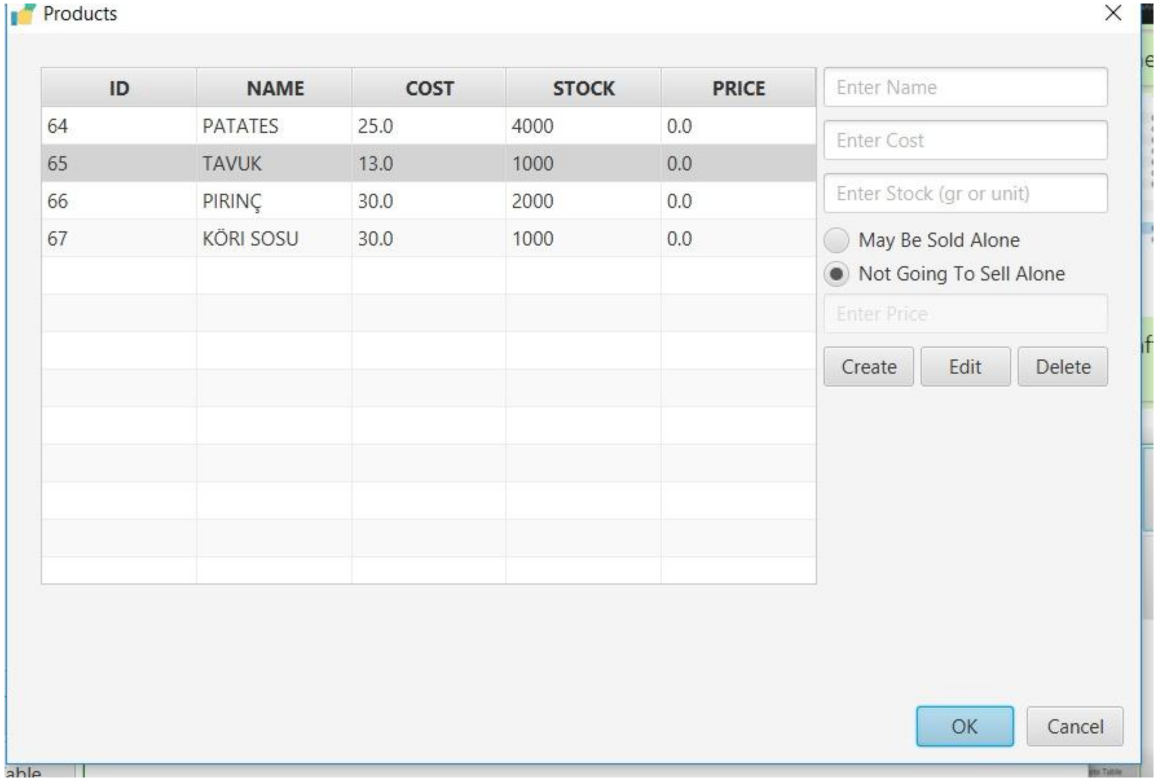
MASA6

NAME	QUANTITY	PRICE	SUB TOTAL
CHICKEN MENU	2	13.0	26.0

Total Cost 26.0

GUI

- All products can be seen and determined in this dialog pane
- Also, the user can delete or edit the product
- If a product can be sold by itself, the user should be select may be sold alone such as coke or other drinks. In addition, if a product is ingredient the user should be select “not going to sell alone” such as curry sauce

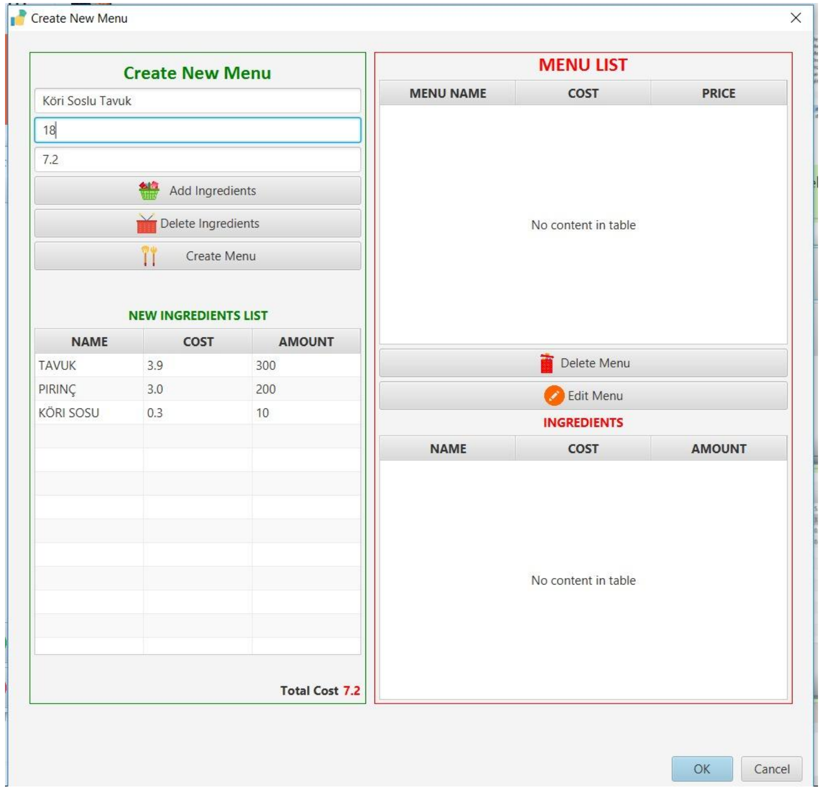

Products

ID	NAME	COST	STOCK	PRICE
64	PATATES	25.0	4000	0.0
65	TAVUK	13.0	1000	0.0
66	PIRINÇ	30.0	2000	0.0
67	KÖRI SOSU	30.0	1000	0.0

Enter Name
Enter Cost
Enter Stock (gr or unit)
May Be Sold Alone [✓ Not Going To Sell Alone]
Enter Price

GUI

- In the left side, the user can create its own menu by adding ingredients in it and also, delete it.
- In the right side, the user can see the created menus and by clicking on edit button, the user can change the menu's ingredients, name, price etc.



Köri Soslu Tavuk

18

7.2

NEW INGREDIENTS LIST

NAME	COST	AMOUNT
TAVUK	3.9	300
PIRINÇ	3.0	200
KÖRI SOSU	0.3	10

Total Cost 7.2

MENU NAME	COST	PRICE

No content in table

INGREDIENTS

NAME	COST	AMOUNT

No content in table