```
00001: package com.databases2.rdbms.controller;
00002:
00003: import javafx.fxml.FXML;
00004: import javafx.scene.Parent;
00005:
00006: public abstract class AbstractController<T extends Parent> {
00007:     @FXML
00008:     private T root;
00009:
00010:     public T getRoot() {
00011:         return root;
00012:     }
00013: }
```

```
00001: package com.databases2.rdbms.controller;
00002:
00003: import javafx.fxml.FXML;
00004: import javafx.scene.Parent;
00005:
00006: public abstract class AbstractController<T extends Parent> {
00007:     @FXML
00008:     private T root;
00009:
00010:     public T getRoot() {
00011:         return root;
00012:     }
00013: }
```

```
00001: package com.databases2.rdbms.controller;
00002:
00003: import java.io.IOException;
00004: import java.io.InputStream;
00005: import java.util.ResourceBundle;
00006:
00007: import javafx.fxml.FXMLLoader;
00008: import javafx.scene.Parent;
00009:
00010: public abstract class AbstractFxmlConfiguration {
00011:     protected static final String SUFFIX_FXML = ".fxml";
00012:     protected static final String SUFFIX_PRESENTER = "Presenter";
00013:
00014:     protected <C extends AbstractController<T>, T extends Parent> C loadController(Class<C> clazz) throws IOException {
00015:
00016:         String derivedXmlName = deriveXMLName(clazz);
00017:         System.out.println(derivedXmlName);
00018:
00019:         try (InputStream fxmlStream = Thread.currentThread().getContextClassLoader()
00020:                 .getResourceAsStream(derivedXmlName)) {
00021:             FXMLLoader loader = new FXMLLoader();
00022:
00023:             loader.load(fxmlStream);
00024:             return loader.getController();
00025:
00026:         }
00027:
00028:     }
00029:
00030:     protected String deriveXMLName(Class<?> clazz) {
00031:         return clazz.getSimpleName().replaceAll(SUFFIX_PRESENTER + "$", "") + SUFFIX_FXML;
00032:     }
00033:
```

```
00034:     protected String deriveBundleName(Class<?> clazz) {
00035:         return clazz.getSimpleName().replaceAll(SUFFIX_PRESENTER + "$", "");
00036:     }
00037:
00038: }
```

```
00001: package com.databases2.rdbms.controller;
00002:
00003: import java.io.IOException;
00004:
00005: import org.springframework.beans.factory.config.ConfigurableBeanFactory;
00006: import org.springframework.context.annotation.Bean;
00007: import org.springframework.context.annotation.Configuration;
00008: import org.springframework.context.annotation.Lazy;
00009: import org.springframework.context.annotation.Scope;
00010:
00011: @Scope(ConfigurableBeanFactory.SCOPE_PROTOTYPE)
00012: @Configuration
00013: public class ControllerConfiguration  extends AbstractFxmlConfiguration {
00014:     @Bean
00015:     @Lazy
00016:     public RestaurantApplicationController restaurantApplicationController() throws IOException {
00017:         return loadController(RestaurantApplicationController.class);
00018:     }
00019:
00020:     @Bean
00021:     @Lazy
00022:     public LoginController logincontroller() throws IOException {
00023:         return loadController(LoginController.class);
00024:     }
00025:
00026:     @Bean
00027:     @Lazy
00028:     public CustomerController customerController() throws IOException {
00029:         return loadController(CustomerController.class);
00030:     }
00031:
00032:
00033:
```

```
00034:
00035:
00036: }
```

```
00001: package com.databases2.rdbms.model;
00002:
00003: public class Customer {
00004:     int id;
00005:     String name;
00006:     String phone;
00007:     String feedback;
00008:
00009:     public Customer(int id, String name, String phone, String feedback) {
00010:         this.id = id;
00011:         this.name = name;
00012:         this.phone = phone;
00013:         this.feedback = feedback;
00014:     }
00015:
00016:
00017:     public String getId() {
00018:         return Integer.toString(id);
00019:     }
00020:     public void setId(int id) {
00021:         this.id = id;
00022:     }
00023:     public String getName() {
00024:         return name;
00025:     }
00026:     public void setName(String name) {
00027:         this.name = name;
00028:     }
00029:     public String getPhone() {
00030:         return phone;
00031:     }
00032:     public void setPhone(String phone) {
00033:         this.phone = phone;
```

```
00034:     }
00035:     public String getFeedback() {
00036:         return feedback;
00037:     }
00038:     public void setFeedback(String feedback) {
00039:         this.feedback = feedback;
00040:     }
00041:
00042:
00043:
00044: }
```

```
00001: package com.databases2.rdbms.controller;
00002:
00003: import java.util.List;
00004: import java.util.concurrent.TimeUnit;
00005: import java.util.stream.Collectors;
00006:
00007: import com.databases2.rdbms.db.CustomerDao;
00008: import com.databases2.rdbms.db.FoodItemDao;
00009: import com.databases2.rdbms.db.OrderDao;
00010: import com.databases2.rdbms.db.OrderItemDao;
00011: import com.databases2.rdbms.model.Customer;
00012: import com.databases2.rdbms.model.FoodItem;
00013: import com.databases2.rdbms.model.OrderItem;
00014: import com.databases2.rdbms.model.OrderModel;
00015: import com.jfoenix.controls.JFXTextField;
00016:
00017: import javafx.collections.FXCollections;
00018: import javafx.collections.ObservableList;
00019: import javafx.fxml.FXML;
00020: import javafx.scene.control.TableColumn;
00021: import javafx.scene.control.TableView;
00022: import javafx.scene.control.cell.PropertyValueFactory;
00023: import javafx.scene.layout.AnchorPane;
00024:
00025: public class CustomerController extends AbstractController<AnchorPane> {
00026:
00027:     @FXML
00028:     private TableView<Customer> customerTableView;
00029:     @FXML
00030:     private TableColumn<Customer, String> customerIdCol;
00031:     @FXML
00032:     private TableColumn<Customer, String> customerNameCol;
00033:     @FXML
```

```
00034:     private TableColumn<Customer, String> customerPhoneCol;
00035:     @FXML
00036:     private TableColumn<Customer, String> customerFeedbackCol;
00037:
00038:     @FXML
00039:     private TableView<FoodItem> foodTableView;
00040:     @FXML
00041:     private TableView<FoodItem> selectedFoodTableView;
00042:
00043:     @FXML
00044:     private TableColumn<FoodItem, String> foodIdCol;
00045:     @FXML
00046:     private TableColumn<FoodItem, String> foodNameCol;
00047:     @FXML
00048:     private TableColumn<FoodItem, String> foodPriceCol;
00049:
00050:     @FXML
00051:     private TableColumn<FoodItem, String> selectedFoodIdCol;
00052:     @FXML
00053:     private TableColumn<FoodItem, String> selectedFoodNameCol;
00054:     @FXML
00055:     private TableColumn<FoodItem, String> selectedFoodPriceCol;
00056:     @FXML
00057:     private TableColumn<FoodItem, String> selectedFoodAmountCol;
00058:
00059:     @FXML
00060:     private TableView<OrderModel> orderTableView;
00061:     @FXML
00062:     private TableColumn<OrderModel, String> orderIdCol;
00063:     @FXML
00064:     private TableColumn<OrderModel, String> customerOrderIdCol;
00065:     @FXML
00066:     private TableColumn<OrderModel, String> orderTypeCol;
```

```
00067:
00068:     @FXML
00069:     private JFXTextField amountTextField;
00070:
00071:
00072:     CustomerDao customerDao = new CustomerDao();
00073:     ObservableList<Customer> customerList = FXCollections.observableArrayList();
00074:
00075:     FoodItemDao foodItemDao = new FoodItemDao();
00076:
00077:     ObservableList<FoodItem> foodList = FXCollections.observableArrayList();
00078:     ObservableList<FoodItem> selectedFoodList = FXCollections.observableArrayList();
00079:
00080:     OrderDao orderDao = new OrderDao();
00081:     OrderItemDao orderItemDao = new OrderItemDao();
00082:     ObservableList<OrderModel> orderList = FXCollections.observableArrayList();
00083:
00084:     @FXML
00085:     private void initialize() {
00086:         customerIdCol.setCellValueFactory(new PropertyValueFactory<Customer, String>("id"));
00087:         customerNameCol.setCellValueFactory(new PropertyValueFactory<Customer, String>("name"));
00088:         customerPhoneCol.setCellValueFactory(new PropertyValueFactory<Customer, String>("phone"));
00089:         customerFeedbackCol.setCellValueFactory(new PropertyValueFactory<Customer, String>("feedback"));
00090:
00091:         customerList.addAll(customerDao.getCustomers());
00092:         customerTableView.setItems(customerList);
00093:
00094:         foodIdCol.setCellValueFactory(new PropertyValueFactory<FoodItem, String>("id"));
00095:         foodNameCol.setCellValueFactory(new PropertyValueFactory<FoodItem, String>("name"));
00096:         foodPriceCol.setCellValueFactory(new PropertyValueFactory<FoodItem, String>("price"));
00097:
00098:         selectedFoodIdCol.setCellValueFactory(new PropertyValueFactory<FoodItem, String>("id"));
00099:         selectedFoodNameCol.setCellValueFactory(new PropertyValueFactory<FoodItem, String>("name"));
```

```
00100:         selectedFoodPriceCol.setCellValueFactory(new PropertyValueFactory<FoodItem, String>("price"));
00101:         selectedFoodAmountCol.setCellValueFactory(new PropertyValueFactory<FoodItem, String>("amount"));
00102:
00103:
00104:         orderIdCol.setCellValueFactory(new PropertyValueFactory<OrderModel, String>("id"));
00105:         customerOrderIdCol.setCellValueFactory(new PropertyValueFactory<OrderModel, String>("customerId"));
00106:         orderTypeCol.setCellValueFactory(new PropertyValueFactory<OrderModel, String>("orderType"));
00107:
00108:         foodList.addAll(foodItemDao.getFoods());
00109:         orderList.addAll(orderDao.getOrders());
00110:
00111:         foodTableView.setItems(foodList);
00112:         selectedFoodTableView.setItems(selectedFoodList);
00113:         orderTableView.setItems(orderList);
00114:     }
00115:
00116:     @FXML
00117:     private void backButtonOnAction() {
00118:
00119:     }
00120:
00121:     @FXML
00122:     private void selectButtonOnAction() {
00123:         FoodItem selectedItem = foodTableView.getSelectionModel().getSelectedItem();
00124:         String amount = amountTextField.getText();
00125:
00126:         if (amount == null) {
00127:             amount = "1";
00128:         }
00129:         selectedItem.setAmount(amountTextField.getText());
00130:         selectedFoodList.add(selectedItem);
00131:         foodList.remove(selectedItem);
00132:
```

```
00133:
00134:     }
00135:
00136:     @FXML
00137:     private void deselectButtonOnAction() {
00138:         FoodItem selectedItem = selectedFoodTableView.getSelectionModel().getSelectedItem();
00139:         selectedFoodList.remove(selectedItem);
00140:         foodList.add(selectedItem);
00141:     }
00142:
00143:     @FXML
00144:     private void checkoutButtonOnAction() {
00145:         List<FoodItem> selectedFoods = selectedFoodTableView.getItems();
00146:         Customer selectedCustomer = customerTableView.getSelectionModel().getSelectedItem();
00147:
00148:         orderDao.insertOrder(Integer.parseInt(selectedCustomer.getId()), "Test", selectedFoods);
00149:
00150:
00151:         orderList.clear();
00152:         orderList.addAll(orderDao.getOrders());
00153:
00154:     }
00155:
00156:     @FXML
00157:     private void updateTableButtonOnAction() {
00158:         orderList.clear();
00159:         orderList.addAll(orderDao.getOrders());
00160:
00161:     }
00162:
00163:     @FXML
00164:     private void deleteOrderButtonOnAction() {
00165:         OrderModel selectedOrder = orderTableView.getSelectionModel().getSelectedItem();
```

```
00166:         orderDao.delteOrder(selectedOrder.getId());
00167:
00168:         orderList.remove(selectedOrder);
00169:
00170:     }
00171:
00172: }
```

```
00001: package com.databases2.rdbms.db;
00002:
00003: import java.sql.Connection;
00004: import java.sql.DriverManager;
00005: import java.sql.ResultSet;
00006: import java.sql.SQLException;
00007: import java.sql.Statement;
00008: import java.util.ArrayList;
00009: import java.util.List;
00010:
00011: import com.databases2.rdbms.model.Customer;
00012:
00013: public class CustomerDao implements DAO {
00014:     List<Customer> customers = new ArrayList<>();
00015:
00016:     private Customer createCustomer(ResultSet rs) {
00017:         try {
00018:             Customer customer = new Customer(rs.getInt("CustomerId"), rs.getString("NAME"),
00019:                     rs.getString("PHONE"), rs.getString("FEEDBACK"));
00020:
00021:             return customer;
00022:         } catch (SQLException e) {
00023:             e.printStackTrace();
00024:         }
00025:
00026:         return null;
00027:
00028:     }
00029:
00030:     public List<Customer> getCustomers() {
00031:         String sql = "SELECT * FROM customer";
00032:
00033:         try {
```

```
00034:             Class.forName(DRIVER);
00035:             Connection con = DriverManager.getConnection(DB_URL, USER, PASS);
00036:             Statement stmt = con.createStatement();
00037:             ResultSet rs = stmt.executeQuery(sql);
00038:             while (rs.next()) {
00039:                 Customer p = createCustomer(rs);
00040:                 customers.add(p);
00041:             }
00042:             rs.close();
00043:             con.close();
00044:         } catch (ClassNotFoundException | SQLException ex) {
00045:         }
00046:         return customers;
00047:
00048:     }
00049:
00050: }
```

```
00001: package com.databases2.rdbms.db;
00002:
00003: public interface DAO {
00004:     public static final String DB_URL = "jdbc:mysql://localhost:3306/restaurant?useSSL=false";
00005:
00006:     public static final String DRIVER = "com.mysql.jdbc.Driver";
00007:     public static final String USER = "root";
00008:     public static final String PASS = "Trollinger37";
00009:
00010: }
```

```
00001: package com.databases2.rdbms.db;
00002:
00003: import java.sql.Connection;
00004: import java.sql.DriverManager;
00005:
00006: public class DBDemo {
00007:
00008:     public static void main(String[] args) {
00009:         String JdbcURL = "jdbc:mysql://localhost:3306/restaurant?useSSL=false";
00010:         String Username = "root";
00011:         String password = "Trollinger37";
00012:         Connection con = null;
00013:
00014:         try {
00015:             System.out.println("Connecting to database..............." + JdbcURL);
00016:             con = DriverManager.getConnection(JdbcURL, Username, password);
00017:             System.out.println("Connection is successful!!!!!!");
00018:         } catch (Exception e) {
00019:             e.printStackTrace();
00020:         }
00021:     }
00022: }
```

```
00001: package com.databases2.rdbms.model;
00002:
00003: public class FoodItem {
00004:     String id;
00005:     String name;
00006:     String price;
00007:     int amount;
00008:
00009:
00010:     public String getId() {
00011:         return id;
00012:     }
00013:     public void setId(String id) {
00014:         this.id = id;
00015:     }
00016:     public String getName() {
00017:         return name;
00018:     }
00019:     public void setName(String name) {
00020:         this.name = name;
00021:     }
00022:     public String getPrice() {
00023:         return price;
00024:     }
00025:     public void setPrice(String price) {
00026:         this.price = price;
00027:     }
00028:     public String getAmount() {
00029:         return Integer.toString(amount);
00030:     }
00031:     public void setAmount(String amount) {
00032:         this.amount = Integer.parseInt(amount);
00033:     }
```

```
00034:
00035:
00036: }
```

```
00001: package com.databases2.rdbms.db;
00002:
00003: import java.util.ArrayList;
00004: import java.util.List;
00005: import java.sql.*;
00006: import com.databases2.rdbms.model.FoodItem;
00007:
00008: public class FoodItemDao implements DAO {
00009:     List<FoodItem> foods = new ArrayList<>();
00010:
00011:     private FoodItem createFoodItem(ResultSet rs) {
00012:         FoodItem foodItem = new FoodItem();
00013:         try {
00014:             foodItem.setId(rs.getString("ITEM_ID"));
00015:             foodItem.setName(rs.getString("ITEM_NAME"));
00016:             foodItem.setPrice(rs.getString("PRICE"));
00017:         } catch (SQLException e) {
00018:             e.printStackTrace();
00019:         }
00020:         return foodItem;
00021:     }
00022:
00023:     public List<FoodItem> getFoods() {
00024:         String sql = "SELECT * FROM  FOOD_MENU";
00025:
00026:         try {
00027:             Class.forName(DRIVER);
00028:             Connection con = DriverManager.getConnection(DB_URL, USER, PASS);
00029:             Statement stmt = con.createStatement();
00030:             ResultSet rs = stmt.executeQuery(sql);
00031:             while (rs.next()) {
00032:                 FoodItem p = createFoodItem(rs);
00033:                 foods.add(p);
```

```
00034:             }
00035:             rs.close();
00036:             con.close();
00037:         } catch (ClassNotFoundException | SQLException ex) {
00038:         }
00039:         return foods;
00040:     }
00041:
00042: }
```

```
00001: package com.databases2.rdbms.controller;
00002:
00003: import org.springframework.beans.factory.annotation.Autowired;
00004:
00005: import com.jfoenix.controls.JFXButton;
00006: import com.jfoenix.controls.JFXPasswordField;
00007: import com.jfoenix.controls.JFXTextField;
00008:
00009: import javafx.fxml.FXML;
00010: import javafx.scene.layout.BorderPane;
00011:
00012: public class LoginController extends AbstractController<BorderPane> {
00013:
00014:     @FXML
00015:     private JFXTextField usernameField;
00016:     @FXML
00017:     private JFXPasswordField passwordField;
00018:     @FXML
00019:     private JFXButton loginButton;
00020:
00021:     @Autowired
00022:     private RestaurantApplication application;
00023:
00024:     @FXML
00025:     private void loginButtonOnAction() {
00026:         application.showMainWindow();
00027:     }
00028: }
```

```
00001: package com.databases2.rdbms;
00002:
00003: import org.springframework.boot.SpringApplication;
00004: import org.springframework.boot.autoconfigure.SpringBootApplication;
00005: import org.springframework.boot.builder.SpringApplicationBuilder;
00006:
00007: import com.databases2.rdbms.controller.LoginController;
00008: import com.databases2.rdbms.controller.RestaurantApplication;
00009:
00010: import javafx.application.Application;
00011: import javafx.fxml.FXMLLoader;
00012: import javafx.scene.Parent;
00013: import javafx.scene.Scene;
00014: import javafx.stage.Stage;
00015:
00016: @SpringBootApplication
00017: public class Main extends Application {
00018:
00019:     private static final int WINDOW_WIDTH = 2000;
00020:     private static final int WINDOW_HEIGHT = 1300;
00021:
00022:     public static void main(String[] args) {
00023:         launch(args);
00024:
00025:     }
00026:
00027:     @Override
00028:     public void start(Stage primaryStage) throws Exception {
00029:         RestaurantApplication.setPrimaryStage(primaryStage);
00030:         new SpringApplicationBuilder(RestaurantApplication.class).run();
00031:
00032:     }
00033: }
```

```
00001: package com.databases2.rdbms.db;
00002:
00003: import java.util.List;
00004: import java.util.ArrayList;
00005: import java.awt.print.Printable;
00006: import java.sql.*;
00007:
00008: import com.databases2.rdbms.model.FoodItem;
00009: import com.databases2.rdbms.model.OrderModel;
00010:
00011: public class OrderDao implements DAO {
00012:     OrderItemDao orderItemDao = new OrderItemDao();
00013:
00014:     private OrderModel createOrder(ResultSet rs) {
00015:         try {
00016:             OrderModel order = new OrderModel(rs.getInt("OrderId"), rs.getString("OrderType"), rs.getInt("CustomerId"));
00017:
00018:             return order;
00019:         } catch (SQLException e) {
00020:             e.printStackTrace();
00021:         }
00022:
00023:         return null;
00024:
00025:     }
00026:
00027:     public List<OrderModel> getOrders() {
00028:         String sql = "SELECT * FROM  customer_order";
00029:         List<OrderModel> orders = new ArrayList<>();
00030:
00031:         try {
00032:             Class.forName(DRIVER);
00033:             Connection con = DriverManager.getConnection(DB_URL, USER, PASS);
```

```
00034:             Statement stmt = con.createStatement();
00035:             ResultSet rs = stmt.executeQuery(sql);
00036:             while (rs.next()) {
00037:                 OrderModel order = createOrder(rs);
00038:                 orders.add(order);
00039:             }
00040:             rs.close();
00041:             con.close();
00042:         } catch (ClassNotFoundException | SQLException ex) {
00043:         }
00044:         return orders;
00045:     }
00046:
00047:     public void insertOrder(int customerId, String orderType, List<FoodItem> selectedFoods) {
00048:         System.out.println("insertOrder");
00049:         String sql = "INSERT INTO customer_order (OrderType, CustomerId) VALUES (?, ?)";
00050:
00051:         try {
00052:             Class.forName(DRIVER);
00053:             Connection con = DriverManager.getConnection(DB_URL, USER, PASS);
00054:
00055:             PreparedStatement preparedStatment = con.prepareStatement(sql);
00056:             preparedStatment.setString(1, orderType);
00057:             preparedStatment.setInt(2, customerId);
00058:
00059:             preparedStatment.executeUpdate();
00060:             System.out.println(preparedStatment);
00061:
00062:         } catch (ClassNotFoundException |
00063:
00064:                 SQLException e) {
00065:             e.printStackTrace();
00066:         }
```

```
00067:
00068:         OrderModel latestOrder = getLatestOrder();
00069:         selectedFoods.forEach(food -> orderItemDao.insertOrderItem(Integer.parseInt(latestOrder.getId()),
00070:                 Integer.parseInt(food.getId()), Integer.parseInt(food.getAmount())));
00071:     }
00072:
00073:
00074:     public void delteOrder(String orderId) {
00075:         String sql = "DELETE FROM customer_order WHERE OrderId=" + orderId;
00076:         String deleteFromOrderItem = "DELETE FROM order_item WHERE OrderId=" + orderId;
00077:
00078:         try {
00079:             Class.forName(DRIVER);
00080:             Connection con = DriverManager.getConnection(DB_URL, USER, PASS);
00081:             Statement stmt = con.createStatement();
00082:             System.out.println(sql);
00083:             stmt.executeUpdate(deleteFromOrderItem);
00084:         } catch (ClassNotFoundException | SQLException ex) {
00085:             ex.printStackTrace();
00086:         }
00087:
00088:         try {
00089:             Class.forName(DRIVER);
00090:             Connection con = DriverManager.getConnection(DB_URL, USER, PASS);
00091:             Statement stmt = con.createStatement();
00092:             System.out.println(sql);
00093:             stmt.executeUpdate(sql);
00094:         } catch (ClassNotFoundException | SQLException ex) {
00095:             ex.printStackTrace();
00096:         }
00097:
00098:     }
00099:
```

```
00100:     public OrderModel getLatestOrder() {
00101:         String sql = "SELECT * FROM customer_order ORDER BY OrderId DESC LIMIT 1;";
00102:
00103:         try {
00104:             Class.forName(DRIVER);
00105:             Connection con = DriverManager.getConnection(DB_URL, USER, PASS);
00106:             Statement stmt = con.createStatement();
00107:             ResultSet rs = stmt.executeQuery(sql);
00108:             OrderModel order = null;
00109:             while (rs.next()) {
00110:                 order = createOrder(rs);
00111:             }
00112:             rs.close();
00113:             con.close();
00114:             return order;
00115:
00116:
00117:         } catch (ClassNotFoundException | SQLException ex) {
00118:         }
00119:
00120:         return null;
00121:     }
00122:
00123: }
```

```
00001: package com.databases2.rdbms.model;
00002:
00003: public class OrderItem {
00004:     int id;
00005:     int orderId;
00006:     int fooId;
00007:     int quantity;
00008:
00009:     public OrderItem(int id, int orderId, int fooId, int quantity) {
00010:         super();
00011:         this.orderId = orderId;
00012:         this.fooId = fooId;
00013:         this.quantity = quantity;
00014:     }
00015:
00016:     public String getId() {
00017:         return Integer.toString(id);
00018:     }
00019:
00020:     public void setId(int id) {
00021:         this.id = id;
00022:     }
00023:
00024:     public String getOrderId() {
00025:         return Integer.toString(orderId);
00026:     }
00027:
00028:     public void setOrderId(int orderId) {
00029:         this.orderId = orderId;
00030:     }
00031:
00032:     public String getFooId() {
00033:         return Integer.toString(fooId);
```

```
00034:     }
00035:
00036:     public void setFooId(int fooId) {
00037:         this.fooId = fooId;
00038:     }
00039:
00040:     public String getQuantity() {
00041:         return Integer.toString(quantity);
00042:     }
00043:
00044:     public void setQuantity(int quantity) {
00045:         this.quantity = quantity;
00046:     }
00047:
00048:
00049:
00050:
00051: }
```

```
00001: package com.databases2.rdbms.db;
00002:
00003: import java.util.ArrayList;
00004: import java.util.List;
00005: import java.sql.*;
00006: import com.databases2.rdbms.model.OrderItem;
00007:
00008: public class OrderItemDao implements DAO {
00009:
00010:     List<OrderItem> foods = new ArrayList<>();
00011:
00012:     private OrderItem createOrderItem(ResultSet rs) {
00013:         try {
00014:             OrderItem order = new OrderItem(rs.getInt("OrderItemId"),
00015:                     rs.getInt("OrderId"),  rs.getInt("ProductId"), rs.getInt("quantity"));
00016:
00017:             return order;
00018:         } catch (SQLException e) {
00019:             e.printStackTrace();
00020:         }
00021:
00022:         return null;
00023:
00024:     }
00025:
00026:     public List<OrderItem> getOrderItems() {
00027:         String sql = "SELECT * FROM  order_item";
00028:
00029:         try {
00030:             Class.forName(DRIVER);
00031:             Connection con = DriverManager.getConnection(DB_URL, USER, PASS);
00032:             Statement stmt = con.createStatement();
00033:             ResultSet rs = stmt.executeQuery(sql);
```

```
00034:             while (rs.next()) {
00035:                 OrderItem p = createOrderItem(rs);
00036:                 foods.add(p);
00037:             }
00038:             rs.close();
00039:             con.close();
00040:         } catch (ClassNotFoundException | SQLException ex) {
00041:         }
00042:         return foods;
00043:     }
00044:
00045:     public void insertOrderItem(int orderId, int foodId, int quantity) {
00046:         String sql = "INSERT INTO order_item (OrderId, ProductId, quantity) VALUES (?, ?, ?)";
00047:
00048:         try {
00049:             Class.forName(DRIVER);
00050:             Connection con = DriverManager.getConnection(DB_URL, USER, PASS);
00051:
00052:             PreparedStatement preparedStatment = con.prepareStatement(sql);
00053:             preparedStatment.setInt(1, orderId);
00054:             preparedStatment.setInt(2, foodId);
00055:             preparedStatment.setInt(3, quantity);
00056:
00057:             System.out.println(preparedStatment.toString());
00058:
00059:             preparedStatment.executeUpdate();
00060:             System.out.println(preparedStatment);
00061:
00062:         } catch (ClassNotFoundException |
00063:
00064:                 SQLException e) {
00065:             e.printStackTrace();
00066:         }
```

```
00067:
00068:     }
00069:
00070:
00071: }
```

```
00001: package com.databases2.rdbms.model;
00002:
00003: public class OrderModel {
00004:     int id;
00005:     int customerId;
00006:     String orderType;
00007:
00008:     public OrderModel(int id, String orderType, int customerId) {
00009:         super();
00010:         this.id = id;
00011:         this.customerId = customerId;
00012:         this.orderType = orderType;
00013:     }
00014:
00015:     public String getId() {
00016:         return Integer.toString(id);
00017:     }
00018:
00019:     public void setId(int id) {
00020:         this.id = id;
00021:     }
00022:
00023:     public String getCustomerId() {
00024:         return Integer.toString(customerId);
00025:     }
00026:
00027:     public void setCustomerId(int customerId) {
00028:         this.customerId = customerId;
00029:     }
00030:
00031:     public String getOrderType() {
00032:         return orderType;
00033:     }
```

```
00034: 
00035:     public void setOrderType(String orderType) {
00036:         this.orderType = orderType;
00037:     }
00038: 
00039: 
00040: }
```

```
00001: package com.databases2.rdbms.controller;
00002:
00003: import javax.inject.Inject;
00004:
00005: import org.springframework.beans.factory.annotation.Autowired;
00006: import org.springframework.boot.CommandLineRunner;
00007: import org.springframework.boot.autoconfigure.SpringBootApplication;
00008: import org.springframework.context.ApplicationContext;
00009: import org.springframework.context.annotation.AnnotationConfigApplicationContext;
00010: import org.springframework.context.annotation.Bean;
00011:
00012: import javafx.fxml.FXML;
00013: import javafx.scene.Scene;
00014: import javafx.scene.control.Tab;
00015: import javafx.scene.control.TabPane;
00016: import javafx.scene.layout.AnchorPane;
00017: import javafx.stage.Stage;
00018:
00019: @SpringBootApplication
00020: public class RestaurantApplication implements CommandLineRunner {
00021:     private static Stage primaryStage;
00022:
00023:
00024:
00025:     @Autowired
00026:     private LoginController loginController;
00027:
00028:     @Autowired
00029:     private RestaurantApplicationController applicationController;
00030:
00031:     @Bean
00032:     public Stage getPrimaryStage() {
00033:         return primaryStage;
```

```
00034:     }
00035:
00036:
00037:     @Override
00038:     public void run(String... args) {
00039:          Scene scene = new Scene(this.loginController.getRoot(), 1200, 600);
00040:
00041:         primaryStage.setScene(scene);
00042:         primaryStage.show();
00043:
00044:         AnnotationConfigApplicationContext ctx = new AnnotationConfigApplicationContext();
00045:         ctx.register(ControllerConfiguration.class);
00046:         ctx.refresh();
00047:     }
00048:
00049:     public static void setPrimaryStage(Stage primarystage) {
00050:         RestaurantApplication.primaryStage = primarystage;
00051:     }
00052:
00053:
00054:
00055:     public void showMainWindow() {
00056:         Scene scene = new Scene(applicationController.getRoot(), 1200, 700);
00057:
00058:         primaryStage.setScene(scene);
00059:         primaryStage.show();
00060:     }
00061:
00062:
00063:
00064:
00065:
00066: }
```

```
00001: package com.databases2.rdbms.controller;
00002:
00003: import java.awt.Button;
00004: import java.util.List;
00005:
00006: import com.databases2.rdbms.db.FoodItemDao;
00007: import com.databases2.rdbms.model.FoodItem;
00008: import com.jfoenix.utils.JFXUtilities;
00009:
00010: import javafx.collections.FXCollections;
00011: import javafx.collections.ObservableList;
00012: import javafx.fxml.FXML;
00013: import javafx.scene.control.TableColumn;
00014: import javafx.scene.control.TableView;
00015: import javafx.scene.control.cell.PropertyValueFactory;
00016: import javafx.scene.layout.BorderPane;
00017:
00018: import javax.annotation.PostConstruct;
00019: import javax.inject.Inject;
00020:
00021: import org.springframework.beans.factory.annotation.Autowired;
00022: import org.springframework.boot.CommandLineRunner;
00023: import org.springframework.boot.autoconfigure.SpringBootApplication;
00024: import org.springframework.context.ApplicationContext;
00025: import org.springframework.context.annotation.Bean;
00026:
00027: import javafx.fxml.FXML;
00028: import javafx.scene.Scene;
00029: import javafx.scene.control.Tab;
00030: import javafx.scene.control.TabPane;
00031: import javafx.scene.layout.AnchorPane;
00032: import javafx.stage.Stage;
00033:
```

```
00034: public class RestaurantApplicationController extends AbstractController<BorderPane> {
00035:
00036:     @FXML
00037:     private TabPane rootTabPane;
00038:
00039:
00040:
00041:     @Inject
00042:     private ApplicationContext applicationContext;
00043:
00044:
00045:     @FXML
00046:     private void initialize() {
00047:
00048:     }
00049:
00050:     @PostConstruct
00051:     private void postConstruct() {
00052:     }
00053:
00054:     private <T extends AbstractController<AnchorPane>> void setActiveControllerInTabPane(Class<T> controllerClass,
00055:             String name) {
00056:         Tab newTab = getTab(name);
00057:
00058:         T newController = applicationContext.getBean(controllerClass);
00059:         AnchorPane controllerPane = newController.getRoot();
00060:
00061:         if (newTab == null) {
00062:             newTab = new Tab(name, controllerPane);
00063:
00064:             rootTabPane.getTabs().add(newTab);
00065:             rootTabPane.getSelectionModel().select(newTab);
00066:         }
```

```
00067:
00068:     }
00069:
00070:
00071:
00072:     private Tab getTab(String tabName) {
00073:         return rootTabPane.getTabs().stream().filter(tab -> tab.getText().equals(tabName)).findFirst().orElse(null);
00074:
00075:     }
00076:
00077:     @FXML
00078:     private void customerButtonOnAction() {
00079:         showCustomerControler();
00080:
00081:     }
00082:
00083:     public void showCustomerControler() {
00084:         setActiveControllerInTabPane(CustomerController.class, "Customers");
00085:     }
00086:
00087:
00088: }
```