Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет» Кафедра ИИТ

Лабораторная работа №10

По дисциплине «СПП» за 6 семестр

Выполнил:

Студент группы ПО-3

Кабачук Д. С.

Проверил:

Крощенко А. А.

Брест 2021

**Вариант 11**

**Цель**: приобрести практические навыки разработки многооконных приложений на JavaFX для работы с базами данных.

**Задание**

На основе БД, разработанной в лабораторной работе No9, реализовать многооконное приложение-клиент, позволяющее выполнять основные операции над таблицей в БД (добавление, удаление, мо- дификацию данных).12)

База данных «Системный администратор».

**Текст программы:**

**Models**

**Equipment**

package sample.model;  
  
public class Equipment {  
  
 int id;  
 String registrationNumber, typeEquipment, manufacturer, employee, warranty;  
  
 public Equipment(int id, String registrationNumber, String typeEquipment, String manufacturer, String employee, String warranty) {  
 this.id = id;  
 this.registrationNumber = registrationNumber;  
 this.typeEquipment = typeEquipment;  
 this.manufacturer = manufacturer;  
 this.employee = employee;  
 this.warranty = warranty;  
 }  
  
 public Equipment(int id, String registrationNumber, String typeEquipment, String manufacturer, String employee) {  
 this.id = id;  
 this.registrationNumber = registrationNumber;  
 this.typeEquipment = typeEquipment;  
 this.manufacturer = manufacturer;  
 this.employee = employee;  
 }  
  
 @Override  
 public String toString() {  
 return "Equipment{" +  
 "id=" + id +  
 ", registrationNumber='" + registrationNumber + '\'' +  
 ", typeEquipment='" + typeEquipment + '\'' +  
 ", manufacturer='" + manufacturer + '\'' +  
 ", employee=" + employee +  
 '}';  
 }  
  
 public String getWarranty() {  
 return warranty;  
 }  
  
 public void setWarranty(String warranty) {  
 this.warranty = warranty;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getRegistrationNumber() {  
 return registrationNumber;  
 }  
  
 public void setRegistrationNumber(String registrationNumber) {  
 this.registrationNumber = registrationNumber;  
 }  
  
 public String getTypeEquipment() {  
 return typeEquipment;  
 }  
  
 public void setTypeEquipment(String typeEquipment) {  
 this.typeEquipment = typeEquipment;  
 }  
  
 public String getManufacturer() {  
 return manufacturer;  
 }  
  
 public void setManufacturer(String manufacturer) {  
 this.manufacturer = manufacturer;  
 }  
  
 public String getEmployee() {  
 return employee;  
 }  
  
 public void setEmployee(String employee) {  
 this.employee = employee;  
 }  
}

**Manufacture**

package sample.model;  
  
public class Manufacturer {  
 int id;  
 String name;  
 String warranty;  
  
 public Manufacturer( String name, String warranty) {  
 this.id = id;  
 this.name = name;  
 this.warranty = warranty;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getWarranty() {  
 return warranty;  
 }  
  
 public void setWarranty(String warranty) {  
 this.warranty = warranty;  
 }  
}

**EquipmentDAO**

package sample.dao;  
  
import sample.model.Equipment;  
  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;  
import java.util.ArrayList;  
import java.util.List;  
  
public class EquipmentDAO {  
  
 public List<Equipment> getAll(){  
 String sql = "SELECT E.ID, E.REGISTRATION\_NUMBER, TE.NAME as TYPE, EM.LNAME as Last, EM.FNAME, M.NAME AS MANUFACTURER, M.WARRANTY, " +  
 "FROM EQUIPMENT E " +  
 "INNER JOIN TYPE\_EQUIPMENT TE ON E.TYPE\_EQUIPMENT = TE.ID " +  
 "INNER JOIN MANUFACTURERS M ON E.MANUFACTURER = M.ID " +  
 "INNER JOIN EMPLOYEES EM ON E.EMPLOYEE = EM.ID";  
  
 List<Equipment> list = new ArrayList<>();  
  
 try {  
 DBConnection.*getConnection*();  
 Statement statement = DBConnection.*connection*.createStatement();  
 ResultSet resultSet = statement.executeQuery(sql);  
  
 while (resultSet.next()){  
 System.*out*.println(resultSet.isLast());  
 list.add(new Equipment(  
 resultSet.getInt("ID"),  
 resultSet.getString("REGISTRATION\_NUMBER"),  
 resultSet.getString("TYPE"),  
 resultSet.getString("MANUFACTURER"),  
 resultSet.getString("LNAME"),  
 resultSet.getString("WARRANTY")  
 ));  
 }  
 } catch (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
  
 return list;  
 }  
  
 public void insertEquipment(Equipment eq){  
 String sql = "insert into Equipment (registration\_number, type\_equipment, " +  
 "manufacturer, employee) values ( ?, (SELECT id FROM TYPE\_EQUIPMENT where NAME = ?)," +  
 "SELECT id FROM MANUFACTURERS where NAME = ?," +  
 "SELECT id FROM EMPLOYEES where lname = ?)";  
 try {  
 DBConnection.*getConnection*();  
 PreparedStatement preparedStatement = DBConnection.*connection*.prepareStatement(sql);  
 preparedStatement.setString(1, eq.getRegistrationNumber());  
 preparedStatement.setString(2, eq.getTypeEquipment());  
 preparedStatement.setString(3, eq.getManufacturer());  
 preparedStatement.setString(4, eq.getEmployee());  
  
 System.*out*.println("1 - "+eq.getRegistrationNumber()+ "2 - "+eq.getTypeEquipment()+  
 "3 - "+ eq.getManufacturer()+ "4 - " + eq.getEmployee());  
 int rows = preparedStatement.executeUpdate();  
 System.*out*.printf("%d rows added", rows);  
  
 List<Equipment> list = getAll();  
 for (Equipment equipment:list) {  
 System.*out*.println(equipment.getEmployee());  
 }  
  
 } catch (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 }  
  
 public void delete(int id){  
 String sql = "Delete from EQUIPMENT where id = ?";  
 try {  
 DBConnection.*getConnection*();  
 PreparedStatement preparedStatement = DBConnection.*connection*.prepareStatement(sql);  
 preparedStatement.setInt(1, id);  
 preparedStatement.executeUpdate();  
  
 } catch (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 }  
  
 public void upDate(Equipment equipment, String str){  
 String sql = "UPDATE EQUIPMENT SET REGISTRATION\_NUMBER = ? WHERE ID = ?";  
 try {  
 DBConnection.*getConnection*();  
 PreparedStatement preparedStatement = DBConnection.*connection*.prepareStatement(sql);  
 preparedStatement.setString(1, str);  
 preparedStatement.setString(2, Integer.*toString*(equipment.getId()));  
  
 int rows = preparedStatement.executeUpdate();  
 System.*out*.printf("%d rows upDate added", rows);  
  
 List<Equipment> list = getAll();  
  
 } catch (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
  
 }  
  
 public List<String> getTypeEquipment(){  
 String sql = "SELECT \* FROM TYPE\_EQUIPMENT";  
  
 List<String> list = new ArrayList<>();  
  
 try {  
 DBConnection.*getConnection*();  
 Statement statement = DBConnection.*connection*.createStatement();  
 ResultSet resultSet = statement.executeQuery(sql);  
  
 while (resultSet.next()){  
  
 list.add(resultSet.getString("name"));  
 System.*out*.println(resultSet.getString("name"));  
 }  
 } catch (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
  
 return list;  
 }  
  
 public List<String> getEmployees(){  
 String sql = "SELECT \* FROM EMPLOYEES";  
  
 List<String> list = new ArrayList<>();  
  
 try {  
 DBConnection.*getConnection*();  
 Statement statement = DBConnection.*connection*.createStatement();  
 ResultSet resultSet = statement.executeQuery(sql);  
  
 while (resultSet.next()){  
  
 list.add(resultSet.getString("lname"));  
 }  
 } catch (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
  
 return list;  
 }  
  
 public List<String> getManufacturers(){  
 String sql = "SELECT \* FROM MANUFACTURERS";  
  
 List<String> list = new ArrayList<>();  
  
 try {  
 DBConnection.*getConnection*();  
 Statement statement = DBConnection.*connection*.createStatement();  
 ResultSet resultSet = statement.executeQuery(sql);  
  
 while (resultSet.next()){  
  
 list.add(resultSet.getString("name"));  
 System.*out*.println(resultSet.getString("name"));  
 }  
 } catch (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
  
 return list;  
 }  
}

**DBConnection**

package sample.dao;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
  
public class DBConnection {  
  
 static final String *URL* = "jdbc:h2:mem:test;INIT=runscript from 'src/main/resources/sql/schema1.sql'\\;" +  
 "runscript from 'src/main/resources/sql/data.sql'";  
  
 static final String *USER* = "sa";  
 static final String *PASS* = "";  
  
 public static Connection *connection* = null;  
  
 public static void getConnection() throws SQLException {  
  
 if(*connection* == null)  
 *connection* = DriverManager.*getConnection*(*URL*, "sa", "");  
  
 }  
}

**Controllers**

**FormController**

package sample.controller;  
  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.scene.control.Button;  
import javafx.scene.control.ComboBox;  
import javafx.scene.control.TextField;  
import javafx.stage.Stage;  
import sample.dao.EquipmentDAO;  
import sample.model.Equipment;  
import sample.model.Manufacturer;  
  
public class FormController{  
  
 @FXML  
 private TextField id;  
  
 @FXML  
 private TextField registration\_number;  
  
 @FXML  
 private ComboBox<String> employee;  
  
  
 @FXML  
 private ComboBox<String> manufacturer;  
  
 @FXML  
 private ComboBox<String> type\_equipment;  
  
 @FXML  
 private Button bottonOk;  
  
 private EquipmentDAO dao = new EquipmentDAO();  
  
 private static HomeController *homeController*;  
  
 Equipment equipment;  
  
  
 @FXML  
 void initialize() {  
 employee.setItems(getEmployee());  
 manufacturer.setItems(getManufacturer());  
 type\_equipment.setItems(getTypes());  
 }  
  
 @FXML  
 void insert(ActionEvent event) {  
  
 equipment = new Equipment(Integer.*parseInt*(id.getText()), registration\_number.getText(),  
 type\_equipment.getValue(), manufacturer.getValue(), employee.getValue());  
 dao.insertEquipment(equipment);  
 *homeController*.refresh();  
 Stage stage = (Stage) bottonOk.getScene().getWindow();  
 stage.close();  
  
 }  
  
 public static void setHomeController(HomeController home){  
 *homeController* = home;  
 }  
  
 ObservableList<String> getEmployee(){  
 ObservableList<String> disks = FXCollections.*observableArrayList*();  
 disks.addAll(dao.getEmployees());  
 return disks;  
 }  
  
 ObservableList<String> getManufacturer(){  
 ObservableList<String> disks = FXCollections.*observableArrayList*();  
 disks.addAll(dao.getManufacturers());  
 return disks;  
 }  
  
 ObservableList<String> getTypes(){  
 ObservableList<String> disks = FXCollections.*observableArrayList*();  
 disks.addAll(dao.getTypeEquipment());  
 return disks;  
 }  
}

**UpDateController**

package sample.controller;  
  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.fxml.FXML;  
import javafx.scene.control.Button;  
import javafx.scene.control.ComboBox;  
import javafx.scene.control.TextField;  
import javafx.scene.control.cell.PropertyValueFactory;  
import javafx.stage.Stage;  
import sample.dao.EquipmentDAO;  
import sample.model.Equipment;  
  
public class UpDateController {  
  
  
 @FXML  
 private TextField id;  
  
 @FXML  
 private TextField registration\_number;  
  
 @FXML  
 private TextField type\_equipment;  
  
 @FXML  
 private TextField manufacturer;  
  
 @FXML  
 private TextField employee;  
  
 @FXML  
 private Button bottonOk;  
  
 private EquipmentDAO dao = new EquipmentDAO();  
  
 private static Equipment *equipment*;  
 private static HomeController *homeController*;  
  
 @FXML  
 void initialize() {  
  
 id.setText(Integer.*toString*(*equipment*.getId()));  
 id.setDisable(true);  
 registration\_number.setText(*equipment*.getRegistrationNumber());  
// registration\_number.setDisable(true);  
 type\_equipment.setText(*equipment*.getTypeEquipment());  
 type\_equipment.setDisable(true);  
 manufacturer.setText(*equipment*.getManufacturer());  
 manufacturer.setDisable(true);  
 employee.setText(*equipment*.getEmployee());  
 employee.setDisable(true);  
 }  
  
 @FXML  
 public void upDate(ActionEvent event){  
 dao.upDate(*equipment*, registration\_number.getText());  
 Stage stage = (Stage) bottonOk.getScene().getWindow();  
 *homeController*.refresh();  
 stage.close();  
 }  
  
 public static void setEquipment(Equipment eq) {  
 *equipment* = eq;  
 }  
  
 public static void setHomeController(HomeController home){  
 *homeController* = home;  
 }  
}

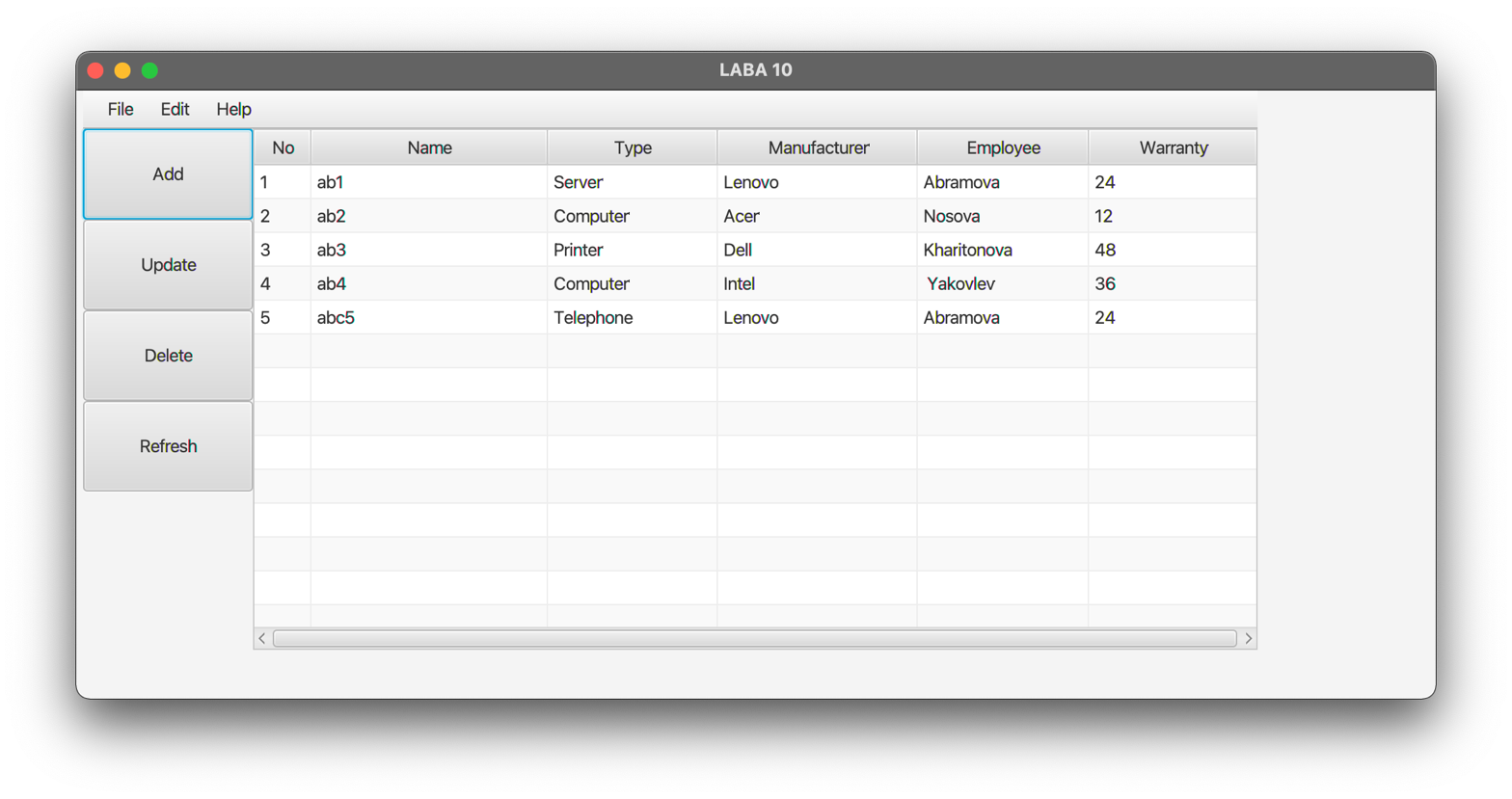
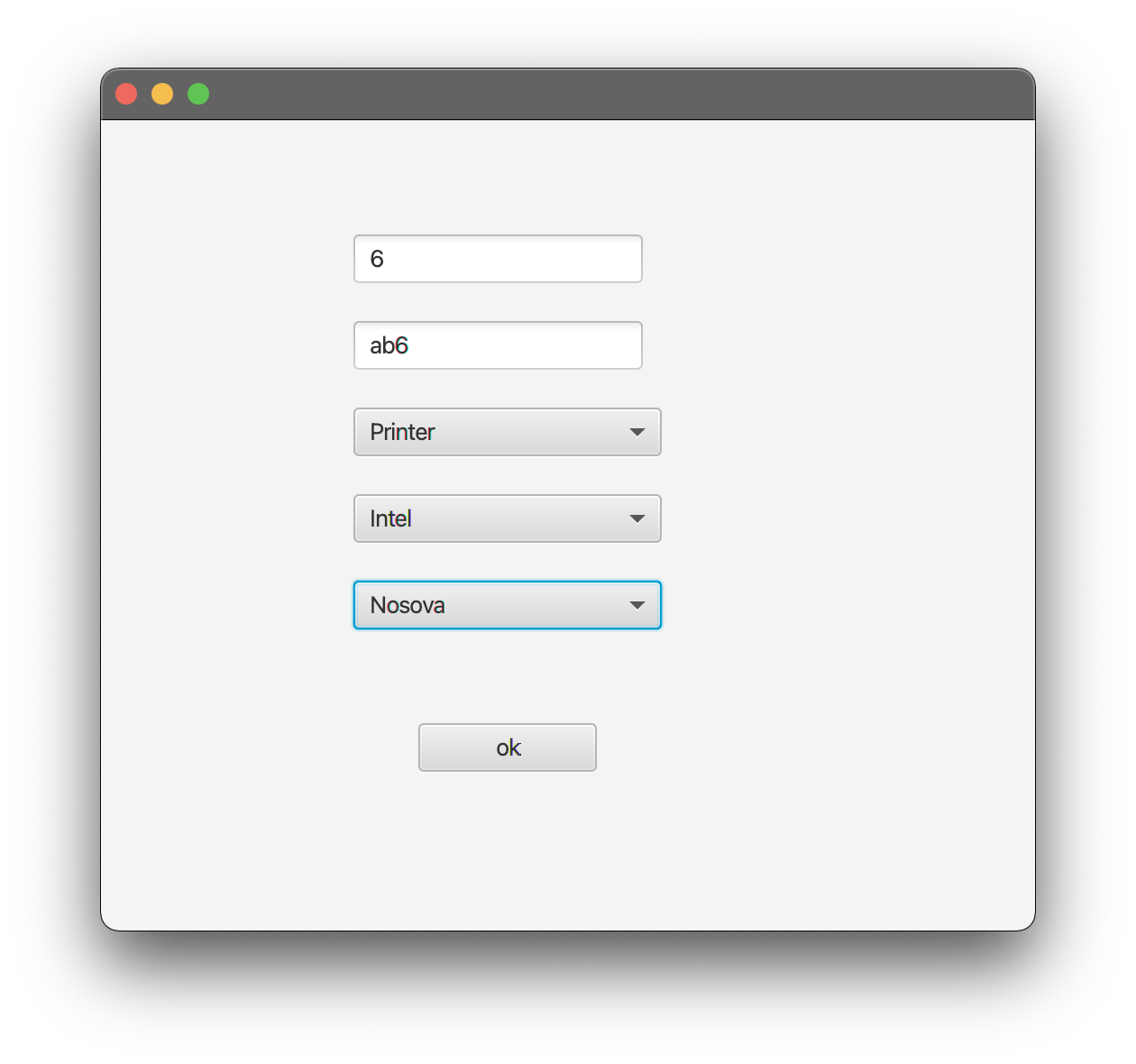
**HomeController**

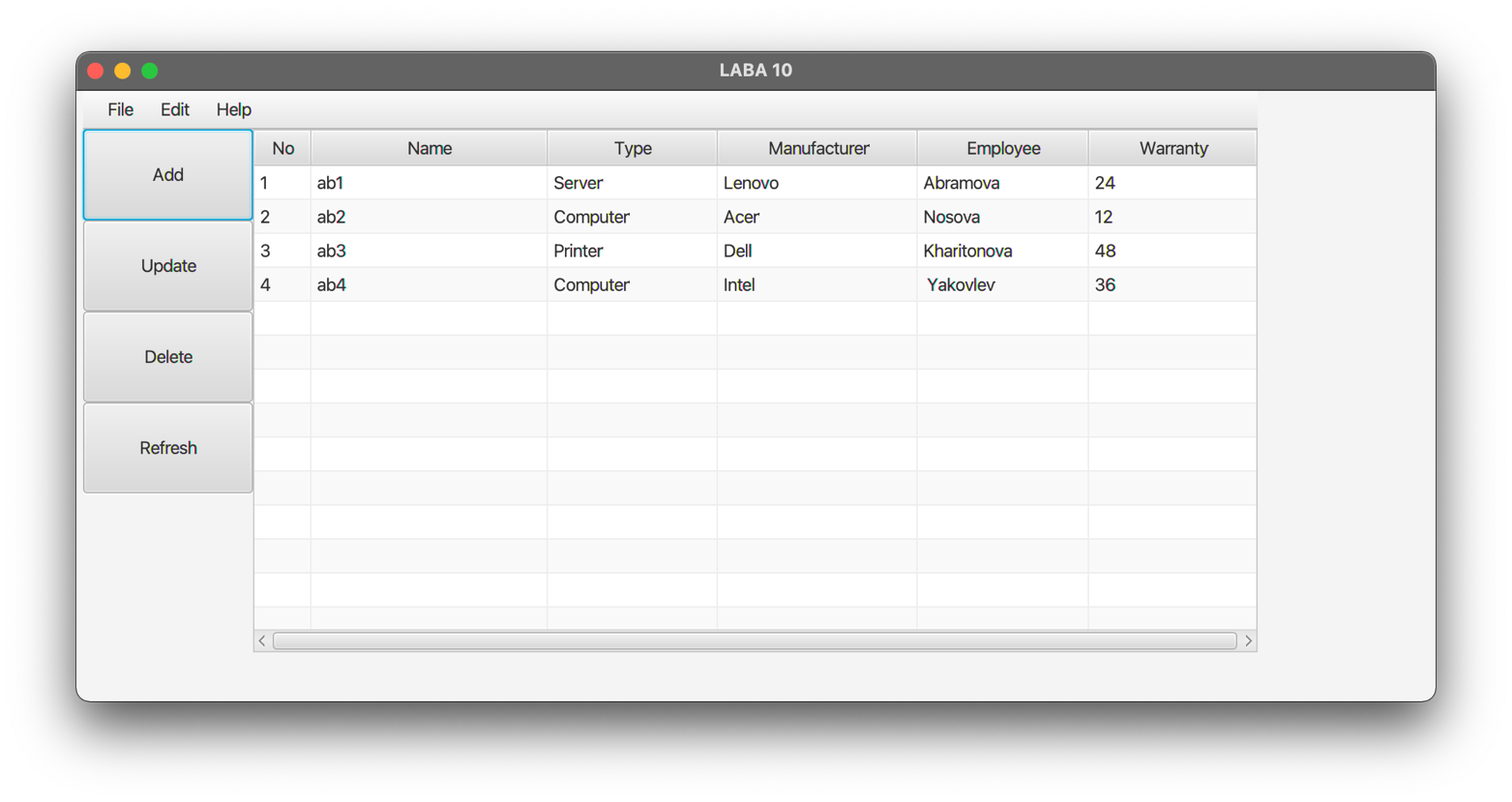
package sample.controller;  
  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.scene.control.TableColumn;  
import javafx.scene.control.TableView;  
import javafx.scene.control.cell.PropertyValueFactory;  
import javafx.stage.Stage;  
import sample.dao.EquipmentDAO;  
import sample.model.Equipment;  
  
import java.io.IOException;  
  
public class HomeController {  
  
 @FXML  
 protected TableView<Equipment> table;  
  
 @FXML  
 private TableColumn<Equipment, String> id;  
  
 @FXML  
 private TableColumn<Equipment, String> registration\_number;  
  
 @FXML  
 private TableColumn<Equipment, String> type\_equipment;  
  
 @FXML  
 private TableColumn<Equipment, String> manufacturer;  
  
 @FXML  
 private TableColumn<Equipment, String> employee;  
  
 @FXML  
 private TableColumn<Equipment, String> warranty;  
  
 ObservableList<Equipment> listview = FXCollections.*observableArrayList*();  
  
 private EquipmentDAO dao = new EquipmentDAO();  
  
 @FXML  
 void initialize() {  
  
 id.setCellValueFactory(new PropertyValueFactory<Equipment, String>("id"));  
 registration\_number.setCellValueFactory(new PropertyValueFactory<>("registrationNumber"));  
 type\_equipment.setCellValueFactory(new PropertyValueFactory<>("typeEquipment"));  
 manufacturer.setCellValueFactory(new PropertyValueFactory<>("manufacturer"));  
 employee.setCellValueFactory(new PropertyValueFactory<>("employee"));  
 warranty.setCellValueFactory(new PropertyValueFactory<>("warranty"));  
  
 listview.addAll(dao.getAll());  
 table.setItems(listview);  
 }  
  
 @FXML  
 void add(ActionEvent event) {  
 try {  
 FormController.*setHomeController*(this);  
 FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("/insert.fxml"));  
 Parent root1 = (Parent) fxmlLoader.load();  
 Stage stage = new Stage();  
 stage.setScene(new Scene(root1));  
 stage.show();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 }  
  
  
 @FXML  
 void upDate(ActionEvent event) {  
 try {  
 UpDateController.*setEquipment*(table.getSelectionModel().getSelectedItem());  
 UpDateController.*setHomeController*(this);  
 FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("/update.fxml"));  
 Parent root1 = (Parent) fxmlLoader.load();  
 Stage stage = new Stage();  
 stage.setScene(new Scene(root1));  
 stage.show();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @FXML  
 public void delete(ActionEvent event) {  
 Equipment equipment = table.getSelectionModel().getSelectedItem();  
 dao.delete(equipment.getId());  
 refresh();  
 }  
  
 public void refresh() {  
 listview.clear();  
 listview.addAll(dao.getAll());  
 table.setItems(listview);  
 }  
  
}

**Main**

package sample;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
public class Main extends Application {  
  
 @Override  
 public void start(Stage primaryStage) throws Exception{  
 Parent root = FXMLLoader.*load*(Main.class.getClassLoader().getResource("home.fxml"));  
 primaryStage.setTitle("LABA 10");  
 primaryStage.setScene(new Scene(root, 1000, 450));  
 primaryStage.show();  
 }  
  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
}

**Результат выполнения:**

****

****

**Вывод:** приобрела практические навыки разработки многооконных приложений на JavaFX для работы с базами данных.