Министерство образования Республики Беларусь

Учреждение образования

“БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ”

Факультет компьютерного проектирования

Дисциплина: СиТАиРИС

**ЛАБОРАТОРНАЯ РАБОТА № 1**

**Объектно-ориентированный анализ и моделирование,**

**разработка статических моделей**

Студент гр. 814301

Чарыев С.

Проверил:

Артем .Л

Минск 2020

1. **Задание:** а). Проектирование моделей в стандарте UML.

б). Программная реализация на JAVA в отдельном java-пакете.

в). Использование и полученного пакета для разработки программной системы.

г). Придумать для данных схем (моделей) описание (легенду) задачи реального мира.

1. **Решение:**

Программа реализации приложения расчета прогноза погоды с использованием баз данных.

**MainServer (в этом файле мы производим подключение клиента к серверу):**

package server.sample;

import javafx.application.Application;

import javafx.application.Platform;

import javafx.event.EventHandler;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.stage.Stage;

import javafx.stage.WindowEvent;

public class MainServer extends Application {

@Override

public void start(Stage primaryStage) throws Exception {

Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));

primaryStage.setTitle("Сервер");

primaryStage.setScene(new Scene(root, 500, 500));

primaryStage.setOnCloseRequest(new EventHandler<WindowEvent>() {

@Override

public void handle(WindowEvent t) {

Platform.exit();

System.exit(0);

}

});

primaryStage.show();

}

public static void main(String[] args) {

launch(args);

}

}

**ServerController(в этом файле мы запускаем работу сервера):**

package server.sample;

import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.TextField;

import server.Database.Database;

import server.server.ServerInstance;

public class ServerController {

@FXML

private TextField database;

@FXML

private TextField login;

@FXML

private TextField psswrd;

@FXML

private TextField prt;

@FXML

private Button srvOn;

@FXML

private Button srvOff;

public void connectDB() {

Database.connect(database.getText(), login.getText(), psswrd.getText(), prt.getText());

}

public void startServer() {

new Thread(ServerInstance.INSTANCE.getInstance()).start();

srvOn.setVisible(false);

srvOff.setVisible(true);

}

public void stopServer() {

ServerInstance.INSTANCE.getInstance().stop();

srvOn.setVisible(true);

srvOff.setVisible(false);

}

}

**MainClient(в этом файле мы считываем и запоминаем данные для подключения к серверу):**

package server.sample;

import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.TextField;

import server.Database.Database;

import server.server.ServerInstance;

public class ServerController {

@FXML

private TextField database;

@FXML

private TextField login;

@FXML

private TextField psswrd;

@FXML

private TextField prt;

@FXML

private Button srvOn;

@FXML

private Button srvOff;

public void connectDB() {

Database.connect(database.getText(), login.getText(), psswrd.getText(), prt.getText());

}

public void startServer() {

new Thread(ServerInstance.INSTANCE.getInstance()).start();

srvOn.setVisible(false);

srvOff.setVisible(true);

}

public void stopServer() {

ServerInstance.INSTANCE.getInstance().stop();

srvOn.setVisible(true);

srvOff.setVisible(false);

}

}

**Client(в этом файле мы подключаемся к серверу):**

package client.sample;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.\*;

import java.util.ArrayList;

public class Client {

private Socket clientSocket;

private ObjectInputStream clientInpStream;

private ObjectOutputStream clientOutpStream;

public void connect() {

try {

clientSocket = new Socket("localhost", 2525);

clientOutpStream = new ObjectOutputStream(clientSocket.getOutputStream());

clientInpStream = new ObjectInputStream(clientSocket.getInputStream());

} catch (Exception e) {

e.printStackTrace();

}

}

public void send(Object obj) {

try {

clientOutpStream.writeObject(obj);

} catch (Exception e) {

e.printStackTrace();

}

}

public boolean receiveResultBool() {

boolean result = true;

try {

result = (boolean) clientInpStream.readObject();

} catch (Exception e) {

e.printStackTrace();

}

return result;

}

public ArrayList<String> receiveResultList() {

ArrayList<String> list = null;

try {

list = (ArrayList<String>)clientInpStream.readObject();

} catch (Exception e) {

}

return list;

}

public String receiveResultString() {

String result = null;

try {

result = (String) clientInpStream.readObject();

} catch (Exception e) {

}

return result;

}

}

**Класс ViewInfoController**

package controllers;  
  
import java.io.BufferedWriter;  
import java.io.File;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.ArrayList;  
import entityClass.Info;  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
import javafx.fxml.FXML;  
import javafx.scene.control.Button;  
import javafx.scene.control.TableColumn;  
import javafx.scene.control.TableView;  
import javafx.scene.control.TextField;  
import javafx.scene.control.cell.PropertyValueFactory;  
import javafx.stage.FileChooser;  
import javafx.stage.Stage;  
import sample.Alert;  
import sample.Client;  
import sample.ClientInstance;  
  
import sceneLoaders.SceneLoaderInstance;  
  
public class ViewInfoController {  
  
 @FXML  
 private Button backButt;  
 @FXML  
 private TextField search;  
 @FXML  
 private TableView<Info> infoTable;  
 @FXML  
 private TableColumn<Info, String> cityColumn;  
 @FXML  
 private TableColumn<Info, String> dateColumn;  
 @FXML  
 private TableColumn<Info, String> typeColumn;  
 @FXML  
 private TableColumn<Info, Integer> pressureColumn;  
 @FXML  
 private TableColumn<Info, Integer> temperatureColumn;  
 @FXML  
 private TableColumn<Info, Integer> windspeedColumn;  
 @FXML  
 private TableColumn<Info, Integer> overcastColumn;  
 @FXML  
 private TableColumn<Info, Integer> humidityColumn;  
 @FXML  
 private TableColumn<Info, Integer> idColumn;  
  
 @FXML  
 void back(){  
 backButt.getScene().getWindow().hide();  
 SceneLoaderInstance.*INSTANCE*.getInstance().loadScene("adminMainWindow", "");  
 }  
  
 public void initialize() {  
 Client client = ClientInstance.*INSTANCE*.getInstance();  
 client.connect();  
 fillTableView();  
 search.textProperty().addListener(( observable, oldValue, newValue)->  
 filterList(oldValue, newValue));  
 }  
  
 public void fillTableView() {  
 ClientInstance.*INSTANCE*.getInstance().send("getInformation");  
 ArrayList<String> list = ClientInstance.*INSTANCE*.getInstance().receiveResultList();  
 ObservableList<Info> information = FXCollections.*observableArrayList*();  
 String[] infoString;  
 for (int i = 0; i < list.size(); i++) {  
 infoString = list.get(i).split(" ", 9);  
 Info info = new Info();  
 info.setId(Integer.*valueOf*(infoString[0]));  
 info.setHumidity(Integer.*valueOf*(infoString[1]));  
 info.setOvercast(Integer.*valueOf*(infoString[2]));  
 info.setTemperature(Integer.*valueOf*(infoString[3]));  
 info.setPressure(Integer.*valueOf*(infoString[4]));  
 info.setWindspeed(Integer.*valueOf*(infoString[5]));  
 info.setCity(infoString[6]);  
 info.setType(infoString[7]);  
 info.setDate(infoString[8]);  
 information.add(info);  
 }  
 idColumn.setCellValueFactory(new PropertyValueFactory<>("id"));  
 cityColumn.setCellValueFactory(new PropertyValueFactory<>("city"));  
 dateColumn.setCellValueFactory(new PropertyValueFactory<>("date"));  
 typeColumn.setCellValueFactory(new PropertyValueFactory<>("type"));  
 pressureColumn.setCellValueFactory(new PropertyValueFactory<>("pressure"));  
 temperatureColumn.setCellValueFactory(new PropertyValueFactory<>("temperature"));  
 windspeedColumn.setCellValueFactory(new PropertyValueFactory<>("windspeed"));  
 overcastColumn.setCellValueFactory(new PropertyValueFactory<>("overcast"));  
 humidityColumn.setCellValueFactory(new PropertyValueFactory<>("humidity"));  
 infoTable.setItems(information);  
 }  
 public void saveToFile() {  
 Stage stage = new Stage();  
 FileChooser fileChooser = new FileChooser();  
 fileChooser.setTitle("Сохранить в файл");  
 File file = fileChooser.showSaveDialog(stage);  
 if (file != null) {  
 try {  
 BufferedWriter outWriter = new BufferedWriter(new FileWriter(file));  
 for (Info info : infoTable.getItems()) {  
 outWriter.write(info.toString());  
 outWriter.newLine();  
 }  
 outWriter.close();  
 } catch (IOException e) {  
 Alert.*display*("Ошибка записи в файл!");  
 }  
 }  
 }  
 public void filterList(String oldValue, String newValue) {  
 ObservableList<Info> filteredList = FXCollections.*observableArrayList*();  
 if (search == null || (newValue.length() < oldValue.length() || newValue == null)) {  
 fillTableView();  
 } else {  
 newValue = newValue.toUpperCase();  
 for (Info info : infoTable.getItems()) {  
 String filter = info.getCity();  
 if (filter.toUpperCase().contains(newValue) || filter.toUpperCase().contains(newValue)) {  
 filteredList.add(info);  
 }  
 }  
 infoTable.setItems(filteredList);  
 }  
 }  
}

1. **Результат программы:**



