|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт информационных технологий |
| Кафедра вычислительной техники |
|  |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1** | |
| **по дисциплине** | |
| «Структуры и алгоритмы обработки данных» | |
|  | |
| Выполнил студент группы ИКБО-05-18 | Токар И.И. |
| Принял | Расулов М.М. |

|  |  |  |
| --- | --- | --- |
| Лабораторные работы выполнены | «\_\_»\_\_\_\_\_\_\_201\_\_ г. |  |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_ г. |  |

Москва 2020 г.

**Описание задания**

Необходимо построить график погоды на основании метеорологических данных в открытых источниках.

Период, за который нужно собрать данные: 10 лет (с 2010 по 2020 года).

Минимальная единица (шаг) – 1 сутки. Значение температуры для одних суток берется среднедневное (не среднесуточное).

В программе следует предусмотреть возможность выбора шага.

В программе следует предусмотреть возможность выбора периода, за который мы проводим наблюдение за графиком.

Шаг временной шкалы: день, неделя, месяц, год.

Город или локация, по которой ведется сбор данных выбирается студентом по своему усмотрению.

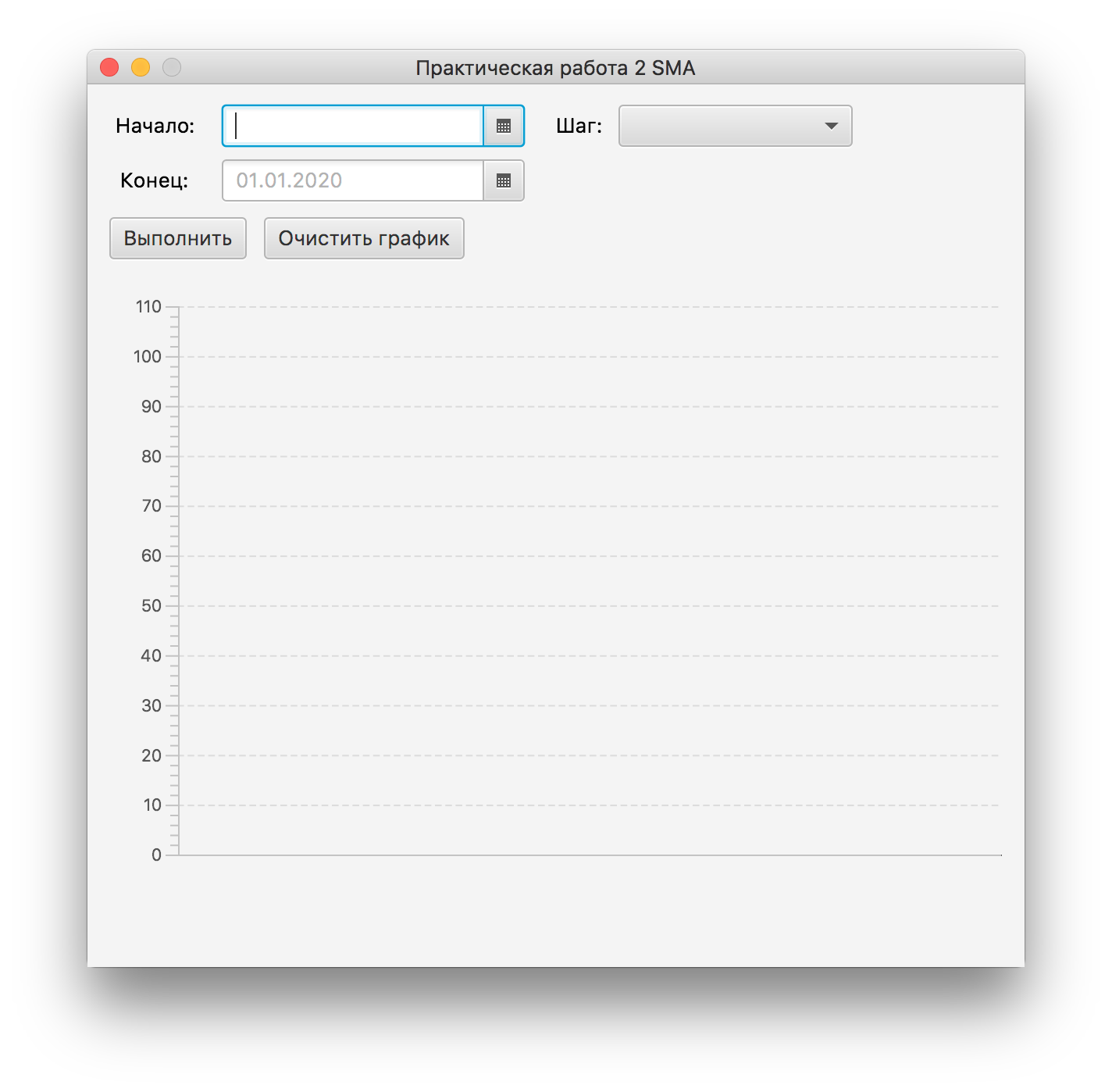
**Описание хода решения заданий**

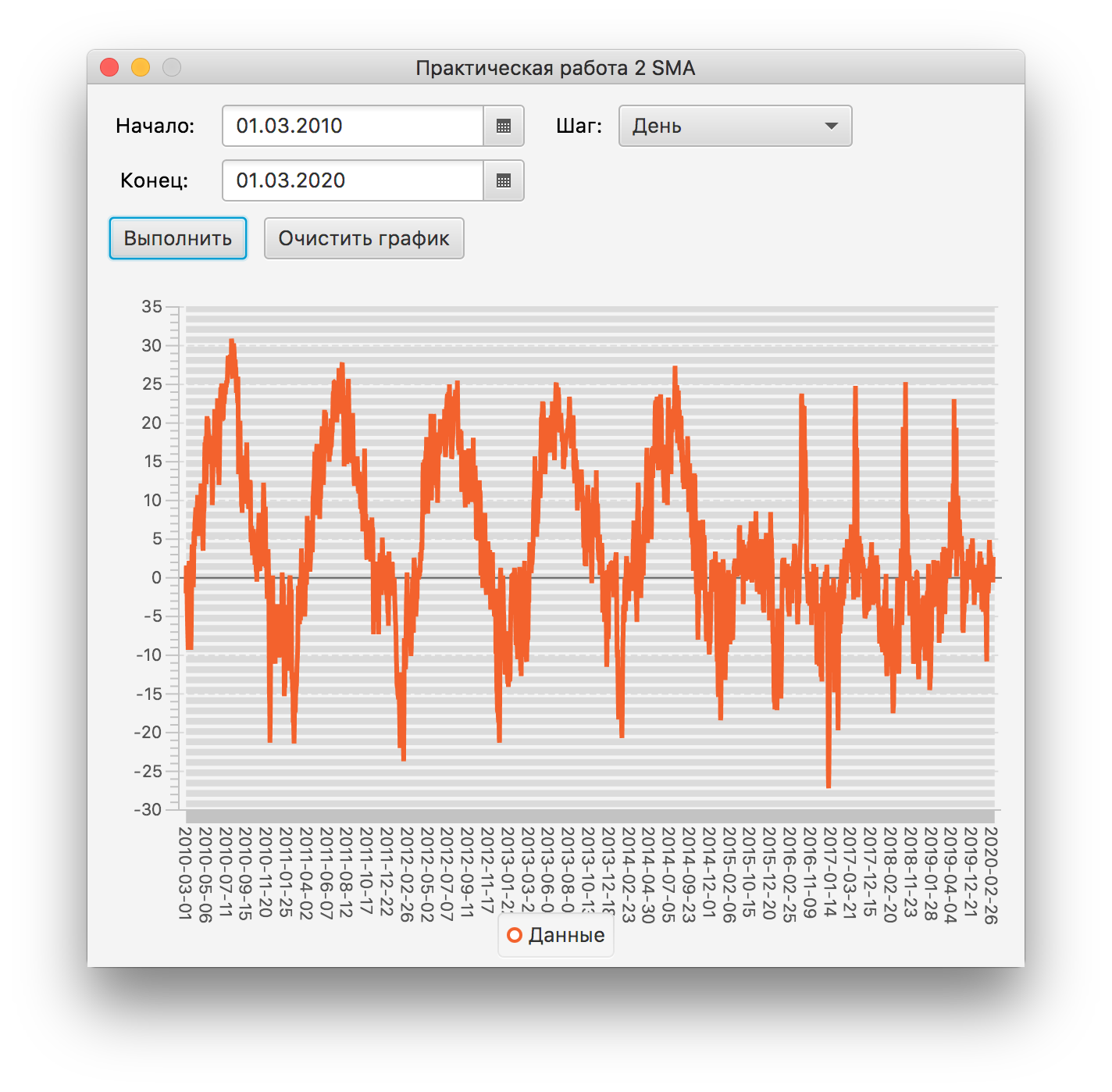
Данная практическая работа была выполнена на языке Java c использованием графической библиотеки JavaFX, для упрощения выполнения HTTP запросов была использована библиотека okhttp3, а для обработки json массивов используется GSON.

Данная программа имеет 2 поля ввода даты тип DataPicer, 3 текстовых меток - Text, для выбора шага был использован компонент Сombobox, для выполнениня основной функциональности используются 2 компонента типа Button, для отрисовки графиков используется компонент ChartLine.

Алгоритм программы заключается в получении данных в формате json и в дальнейшем занесением их в модель таблицы для отображения графика.

**Тестирование**

**  
Рисунок 1 Тестирование программы**

**  
Рисунок 2 Тестирование программы**

**Выводы**

В данной лабораторной работе были получены практические и теоретические навыки по работе с программным интерфейсом API. Так же была разработана программа, которая получает данные в формате json и строит по ним график.

**Список используемой литературы**

1. Роберт Лафоре, Структуры данных и алгоритмы в Java
2. Программирование на языке Java: работа со строками и массивами. Методические указания. [Электронный ресурс] : Учебно-методические пособия — Электрон. дан. — СПб. : ПГУПС, 2015. — 24 с.
3. Кожомбердиева, Г.И. Программирование на языке Java: создание графического интерфейса пользователя: учеб. пособие. [Электронный ресурс] : Учебные пособия / Г.И. Кожомбердиева, М.И. Гарина. — Электрон. дан. — СПб.: ПГУПС, 2012. — 67 с.
4. Вишневская, Т.И. Технология программирования. Часть 1. [Электронный ресурс] / Т.И. Вишневская, Т.Н. Романова. — Электрон. дан. — М. : МГТУ им. Н.Э. Баумана, 2007. — 59 с.

**Приложения**

Main.java

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(getClass().getResource("sample.fxml"));

primaryStage.setTitle("Практическая работа 2 SMA");

primaryStage.setScene(new Scene(root));

primaryStage.setResizable(false);

primaryStage.show();

}

public static void main(String[] args) {

launch(args);

}

}

Controller.java

package sample;

import com.google.gson.Gson;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.fxml.FXML;

import javafx.scene.chart.CategoryAxis;

import javafx.scene.chart.NumberAxis;

import javafx.scene.chart.XYChart;

import javafx.scene.control.Button;

import javafx.scene.control.ComboBox;

import javafx.scene.control.DatePicker;

import javafx.scene.chart.LineChart;

import okhttp3.OkHttpClient;

import okhttp3.Request;

import okhttp3.Response;

import java.io.IOException;

import java.time.format.DateTimeFormatter;

public class Controller {

@FXML

private Button btn;

@FXML

private DatePicker datepickerstart, datepickerend;

@FXML

private LineChart LineChart;

@FXML

private ComboBox stepsComboBox;

@FXML

public void onClickMethod(){

DateTimeFormatter formatters = DateTimeFormatter.ofPattern("yyyy-MM-dd");

String date\_up = datepickerend.getValue().format(formatters);

String date\_down = datepickerstart.getValue().format(formatters);

String url = "https://api.meteostat.net/v1/history/daily?station=27612&start="+date\_down.toString()+"&end="+date\_up.toString()+"&key=L23FNnUI";

Weather data = GetWeather(url);

System.out.println(data.meta.source);

final CategoryAxis xAxis = new CategoryAxis();

final NumberAxis yAxis = new NumberAxis();

xAxis.setLabel("Month");

final LineChart<String,Number> lineChart = new LineChart<String,Number>(xAxis,yAxis);

lineChart.setTitle("Stock Monitoring, 2010");

XYChart.Series series1 = new XYChart.Series();

series1.setName("Данные");

int steps = 1;

switch (stepsComboBox.getSelectionModel().getSelectedItem().toString()){

case "День":

steps = 1;

break;

case "Неделя":

steps = 7;

break;

case "Месяц":

steps = 30;

break;

case "Год":

steps = 365;

break;

default:

steps = 1;

break;

}

for(int i = 0; i < data.data.size(); i+=steps){

String date = data.data.get(i).date;

Float temperature = data.data.get(i).temperature;

if( date != null && temperature != null)

series1.getData().add(new XYChart.Data(date, temperature));

}

LineChart.getData().addAll(series1);

}

@FXML

public void onClearButton(){

LineChart.getData().clear();

}

@FXML

public void ShownComboBox(){

ObservableList<String> steps = FXCollections.observableArrayList(

"День",

"Неделя",

"Месяц",

"Год"

);

stepsComboBox.setItems(steps);

}

public Weather GetWeather (String url){

System.out.println(url);

OkHttpClient client = new OkHttpClient();

Weather data = null;

String userAgent = "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:21.0) Gecko/20100101 Firefox/21.0";

try{

Request request = new Request.Builder()

.url(url)

.header("User-Agent", userAgent)

.build();

Response response = client.newCall(request).execute();

String jsonStr = response.body().string();

System.out.println(jsonStr);

Gson gson = new Gson();

data = gson.fromJson(jsonStr, Weather.class);

}catch (IOException ex){

System.out.println(ex);

}

return data;

}

}