|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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 №2** | |
| **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 на систему с графиком из задания 1 нанести две дополнительные функции (линии) Simple Moving Average.

Период первой линии – 10 дней. Период второй – 100 дней.

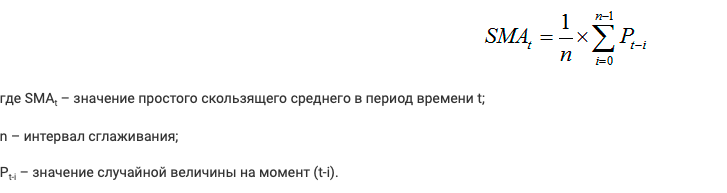
**Описание хода решения заданий**

Данная практическая работа была выполнена на языке Java c использованием графической библиотеки JavaFX, для упрощения выполнения HTTP запросов была использована библиотека okhttp3.

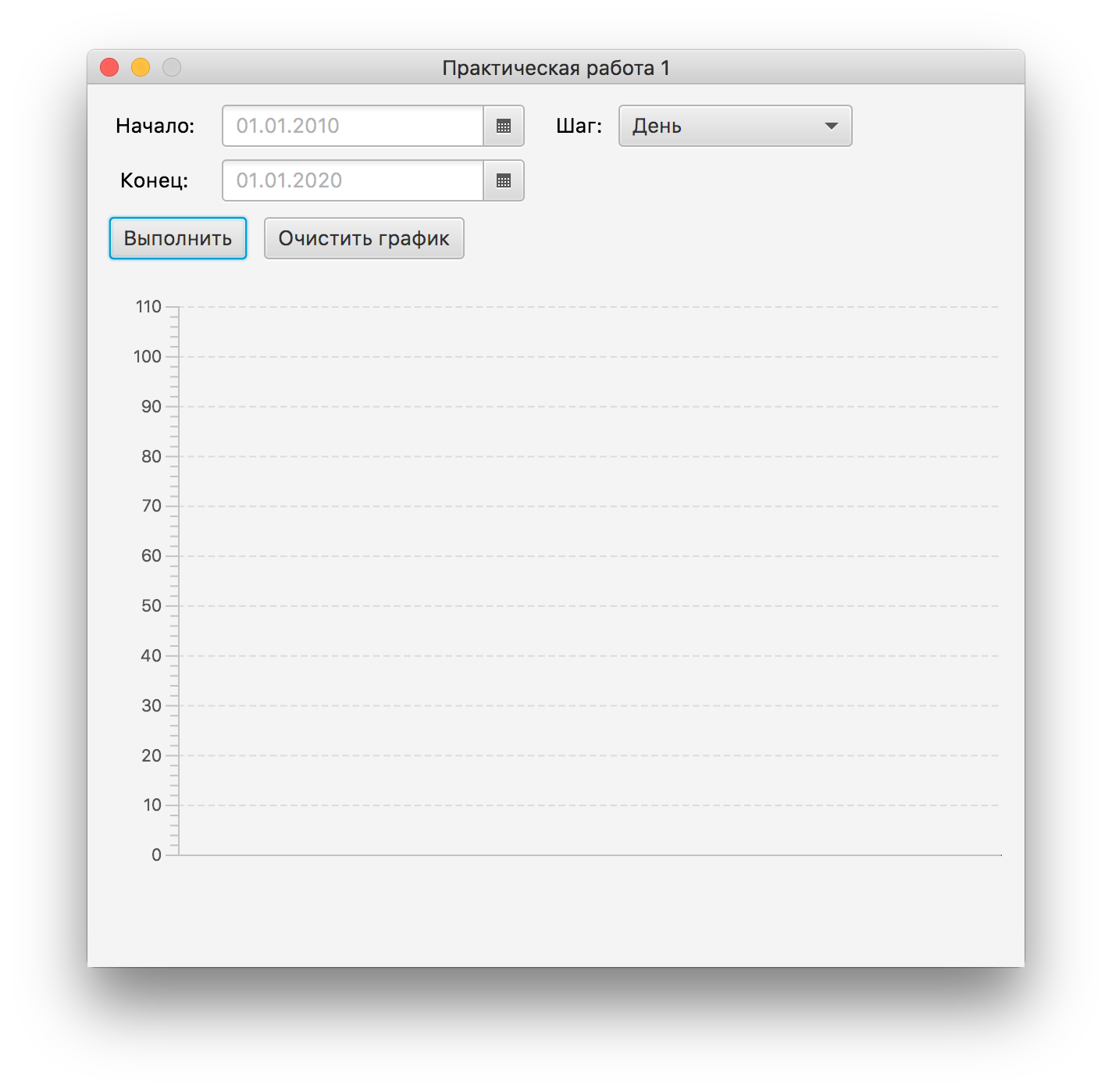
Данная программа имеет 2 поля ввода даты тип DataPicer, 3 текстовых меток - Text, для выбора шага был использован компонент Сombobox, для выполнениня основной функциональности используются 2 компонента типа Button, для отрисовки графиков используется компонент ChartLine.

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

Для расчета SMA используется:



**Тестирование**

**  
Рисунок 1 Тестирование программы**

**  
Рисунок 2 Тестирование программы**

**Выводы**

В данной лабораторной работе были получены практические и теоретические навыки по работе с парсингом сайтов. Так же была разработана программа, которая берет данные с сайта центрального банка России и строит по ним график ретроспективной динамики курса Доллара США к Рублю.

**Список используемой литературы**

**Приложения**

Controller.java

package sample;  
  
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 org.w3c.dom.Document;  
import org.w3c.dom.Element;  
import org.w3c.dom.Node;  
import org.w3c.dom.NodeList;  
import org.xml.sax.InputSource;  
import org.xml.sax.SAXException;  
  
import javax.xml.parsers.DocumentBuilder;  
import javax.xml.parsers.DocumentBuilderFactory;  
import javax.xml.parsers.ParserConfigurationException;  
import java.io.IOException;  
import java.io.StringReader;  
import java.time.format.DateTimeFormatter;  
import java.util.ArrayList;  
  
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*("dd/MM/yyyy");  
 String date\_up = datepickerend.getValue().format(formatters);  
 String date\_down = datepickerstart.getValue().format(formatters);  
  
 String url = "http://www.cbr.ru/scripts/XML\_dynamic.asp?date\_req1="+date\_down.toString()+"&date\_req2="+date\_up.toString()+"&VAL\_NM\_RQ=R01235";  
  
 ArrayList<RecordItem> record = ParseUrl(url);  
  
 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");  
  
 for(RecordItem item : record){  
 series1.getData().add(new XYChart.Data(item.DateRecord, item.Value));  
 }  
  
 XYChart.Series series2 = new XYChart.Series();  
 series2.setName("SMA p=10");  
 int per = 10;  
 SimpleMovingAverage obj = new SimpleMovingAverage(per);  
 for (RecordItem item : record) {  
 obj.addData(item);  
 series2.getData().add(new XYChart.Data(item.DateRecord, obj.getMean()));  
 }  
  
 XYChart.Series series3 = new XYChart.Series();  
 series3.setName("SMA p=100");  
 per = 100;  
 SimpleMovingAverage obj2 = new SimpleMovingAverage(per);  
 for (RecordItem item : record) {  
 obj2.addData(item);  
 series3.getData().add(new XYChart.Data(item.DateRecord, obj2.getMean()));  
 }  
  
 LineChart.getData().addAll(series1,series2,series3);  
 }  
  
 @FXML  
 public void onClearButton(){  
 LineChart.getData().clear();  
 }  
  
 @FXML  
 public void ShownComboBox(){  
 ObservableList<String> steps = FXCollections.*observableArrayList*(  
 "День",  
 "Неделя",  
 "Месяц",  
 "Год"  
 );  
 stepsComboBox.setItems(steps);  
 }  
  
 public ArrayList<RecordItem> ParseUrl(String url){  
 OkHttpClient client = new OkHttpClient();  
 ArrayList<RecordItem> recordItems = new ArrayList<RecordItem>();  
 try{  
 Request request = new Request.Builder().url(url).build();  
 Response response = client.newCall(request).execute();  
  
 DocumentBuilderFactory factory = DocumentBuilderFactory.*newInstance*();  
 DocumentBuilder builder = factory.newDocumentBuilder();  
 Document document = builder.parse(new InputSource(new StringReader(response.body().string())));  
 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;  
 }  
 NodeList RecordNodeList = document.getElementsByTagName("Record");  
  
 for(int i = 0; i < RecordNodeList.getLength(); i+=steps){  
 if(RecordNodeList.item(i).getNodeType() == Node.*ELEMENT\_NODE*){  
 Element item = (Element) RecordNodeList.item(i);  
  
 RecordItem recordItem = new RecordItem(  
 item.getAttribute("Date"),  
 item.getAttribute("Id"),  
 Integer.*valueOf*(item.getChildNodes().item(0).getTextContent()),  
 Double.*valueOf*(item.getChildNodes().item(1).getTextContent().replace(',','.'))  
 );  
 recordItems.add(recordItem);  
 }  
 }  
  
 }catch (IOException ex){  
 System.*out*.println(ex);  
 }catch (ParserConfigurationException e) {  
 e.printStackTrace();  
 } catch (SAXException e) {  
 e.printStackTrace();  
 }  
  
 return recordItems;  
 }  
  
 public ArrayList<Double> SMA100(){  
 return new ArrayList<>();  
 }  
  
 public ArrayList<Double> SMA10(){  
 return new ArrayList<>();  
 }  
}

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);  
 }  
}

package sample;  
  
import java.util.Date;  
  
public class RecordItem {  
 String DateRecord;  
 String Id;  
 int Nominal;  
 Double Value;  
  
 public RecordItem(String dateRecord, String id, int nominal, Double value) {  
 DateRecord = dateRecord;  
 Id = id;  
 Nominal = nominal;  
 Value = value;  
 }  
  
 @Override  
 public String toString() {  
 return "Record{" +  
 "DateRecord=" + DateRecord +  
 ", Id='" + Id + '\'' +  
 ", Nominal=" + Nominal +  
 ", Value=" + Value +  
 '}';  
 }  
}

SimpleMovingAverage.java

package sample;  
  
import java.util.LinkedList;  
import java.util.Queue;  
  
public class SimpleMovingAverage {  
  
 private final Queue<RecordItem> Dataset = new LinkedList<RecordItem>();  
 private final int period;  
 private double sum;  
  
 public SimpleMovingAverage(int period)  
 {  
 this.period = period;  
 }  
  
 public void addData(RecordItem item)  
 {  
 sum += item.Value;  
 Dataset.add(item);  
  
 if (Dataset.size() > period)  
 {  
 sum -= Dataset.remove().Value;  
 }  
 }  
  
 public double getMean()  
 {  
 return sum / period;  
 }  
}