Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Информационных технологий и управления
Кафедра Интеллектуальных информационных технологий

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине «Проектирование программ в интеллектуальных системах»
на тему:

“ПОСТРОЕНИЕ ГРАФИЧЕСКОГО ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА С ИСПОЛЬЗОВАНИЕМ СТАНДАРТНОГО ДРЕВОВИДНОГО КОМПОНЕНТА”

Выполнил: Студент группы 821702
Макаревич Д.А.

Проверил: Садовский М.Е.

Минск, 2020

Задание:

Разработать оконное приложение с одним главным окном.

Вариант 1

Калькулятор должен поддерживать следующий набор операций:

1. Ввод чисел;
2. Ввод знаков приоритета операторов (скобки);
3. Вазовые операция «+», «-», «*», «/», «sqrt», «%», «1/x»;
4. Кнопка получения результата «=»;
5. Из инженерного калькулятора должна быть поддержка операций степенных функции.
6. Данные операции становятся доступными пользователю после выбора соответствующего элемента управления в пункте меню. Если выбор снят, то данных функции становится недоступными.
7. Кнопка сброса всех вычислений. Производиться очистка дерева разбора и поля для ввода выражения.

В калькуляторе должно быть реализовано отображение, предназначенное для просмотра дерева разбора выражения.

Калькулятор должен позволять просматривать вычисление результата по действиям. Просмотр должен осуществляться в двух направлениях (в направлении сверстки выражения, в направлении расхлопывания выражения). Результаты сверстки и расхлопывания выражения должны отображаться в дереве разбора и в поле для ввода выражения.

Графическое отображение

Главное окно приложения:

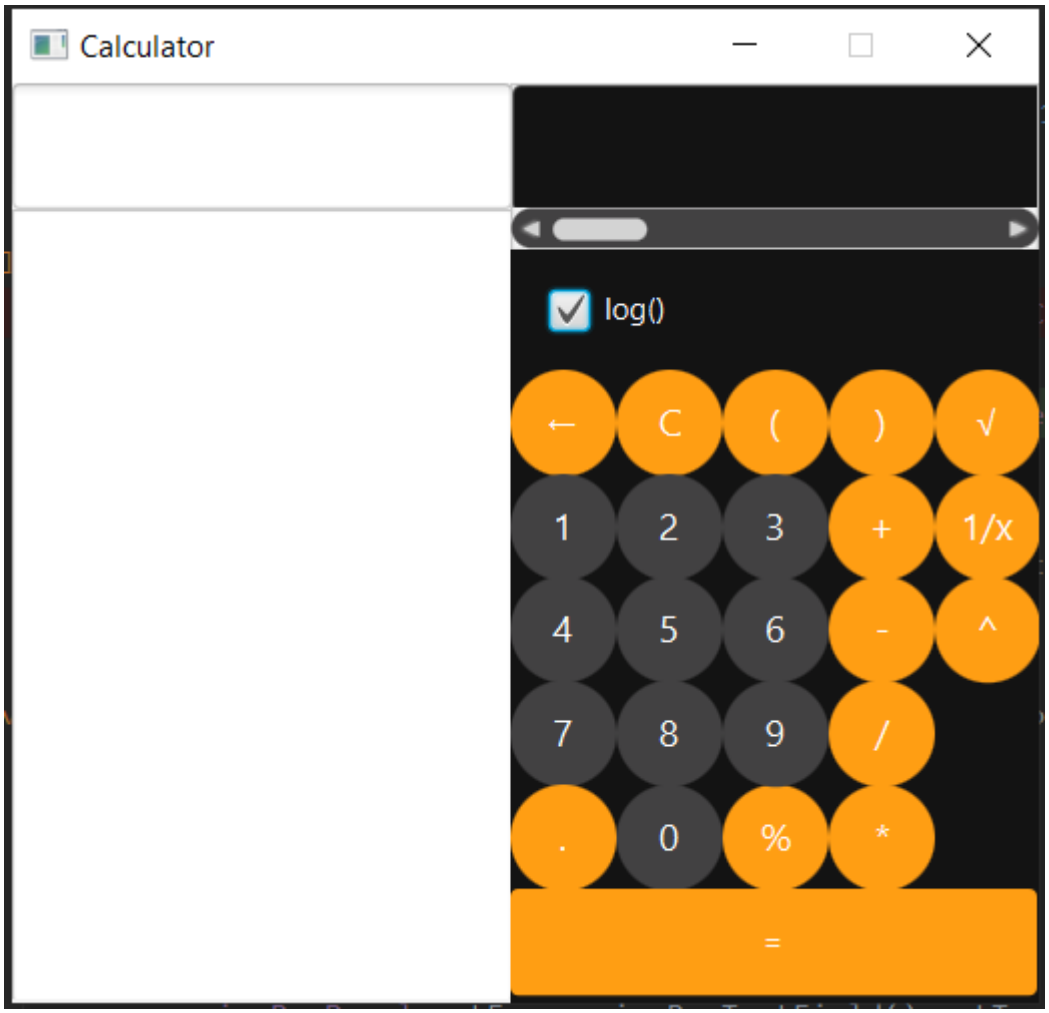

Рисунок 1 Окно приложения

Вычисление выражения (5+5)^(2/2):

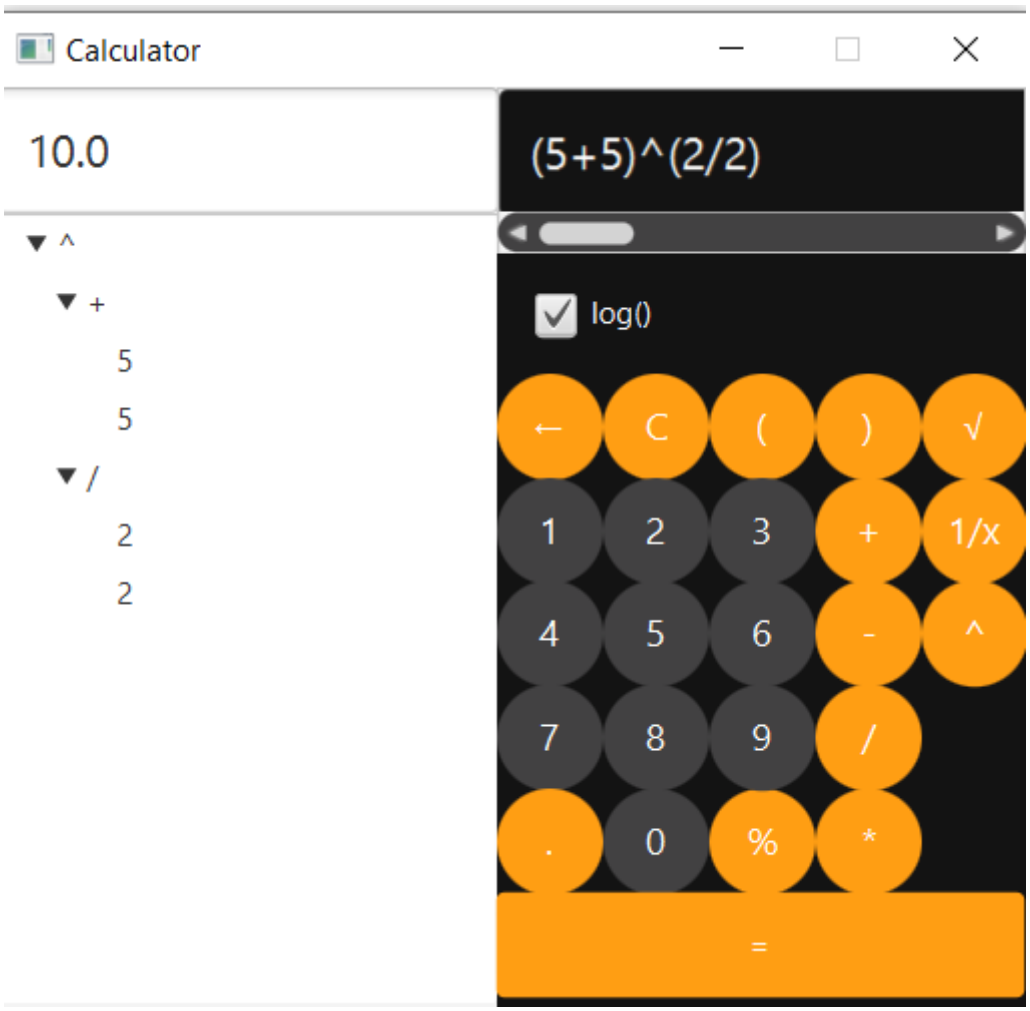

Рисунок 2 Работа калькулятора

Описание классов

Данный класс Создает древовидный компонент и окно калькулятора.

Класс CalculatorForm:

```java
package view;

import ...


public class CalculatorForm {
    private ExpressionTreePanel expressionTreePanel;
    private ExpressionRowPanel expressionRowPanel;
    private OperButtonPanel operButtonPanel;

    private GridPane gridPane;


    public CalculatorForm(ExpressionTreeController expressionTreeController) {
        expressionRowPanel = new ExpressionRowPanel();
        expressionTreePanel = new ExpressionTreePanel(expressionRowPanel, expressionTreeController);
        operButtonPanel = new OperButtonPanel(expressionRowPanel, expressionTreePanel, expressionTreeController);

        gridPane = new GridPane();
        configureGridPane();
    }

    public GridPane getGridPane() { return gridPane; }

    /*
     *      Configs
     */

    private void configureGridPane() {
        gridPane.setAlignment(Pos.CENTER);

        gridPane.add(expressionTreePanel.getGridPane(), columnIndex: 0, rowIndex: 0, colspan: 1, rowspan: 2);
        gridPane.add(expressionRowPanel.getExpRowScrollPane(), columnIndex: 1, rowIndex: 0);
        gridPane.add(operButtonPanel.getGridPane(), columnIndex: 1, rowIndex: 1);
    }
}
```

Рисунок 3 Класс CalculatorForm

Класс строящий “дерево выражения”.

```java
package model;

import ...


public class ExpressionTree {
    private ExpressionTreeNode root;

    public ExpressionTree() { root = null; }

    public ExpressionTree(Expression expression) throws Exception {
        root = construct(expression);
    }

    public ExpressionTreeNode getRoot() { return root; }

    public Expression toInfix() {
        Expression infix = new Expression();
        traverseInfix(root, infix);

        List<Token> toRemove = new ArrayList<>();

        for (int tokenIterator = 2; tokenIterator < infix.tokens().size(); tokenIterator++) {
            if (isOperand(infix.tokens().get(tokenIterator).source())) {
                if (isBracket(infix.tokens().get(tokenIterator + 1).source())
                        && !(isUnaryOperator(infix.tokens().get(tokenIterator - 2).source()))) {
                    toRemove.add(infix.tokens().get(tokenIterator + 1));
                    toRemove.add(infix.tokens().get(tokenIterator - 1));
                }
            }
        }

        toRemove.add(infix.tokens().get(0));
```

Рисунок 4 Класс ExpressionTree

Класс создающий рабочее поле самого калькулятора с кнопками, задает стиль и место положения кнопок.

```
import controller.ExpressionTreeController;
import controller.RPNExpressionConverter;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Insets;
import javafx.scene.control.Button;
import javafx.scene.control.CheckBox;
import javafx.scene.control.TextField;
import javafx.scene.layout.*;
import model.OperatorFactory;


public class OperButtonPanel {
    public static final String CLEAR = "C";
    public static final String BACKSPACE = "←";
    public static final String EQUAL = "=";
    public static final String REVERSE = "1/x";
    public static final String OPEN = "(";
    public static final String CLOSE = ")";

    public static final String ONE = "1";
    public static final String TWO = "2";
    public static final String THREE = "3";
    public static final String FOUR = "4";
    public static final String FIVE = "5";
    public static final String SIX = "6";
    public static final String SEVEN = "7";
    public static final String EIGHT = "8";
    public static final String NINE = "9";
    public static final String ZERO = "0";
```

Рисунок 5 Класс OperButtonPanel

Класс реализующий вершину дерева выражения, вершина может являться либо оператором, либо выражением:

```java
package model;


public class ExpressionTreeNode {
    public enum State { OPERATOR, VALUE };

    private State state;
    private ExpressionTreeNode leftOperand;
    private ExpressionTreeNode rightOperand;
    private Operator operator;
    private Operand value;


    public ExpressionTreeNode(Operator operator) {
        this.operator = operator;
        state = State.OPERATOR;
    }

    public ExpressionTreeNode(Operand value) {
        leftOperand = null;
        rightOperand = null;
        operator = null;
        this.value = value;

        state = State.VALUE;
    }

    public Operator getOperator() { return operator; }

    public Operand getValue() {
        if (value != null) {
            return value;
        } else {
```

Рисунок 6 Класс ExpressionTreeNode

Ход работы

Приложение реализует стандартный калькулятор с построением полного дерева выражения. После того как пользователь ввел выражение и нажал на кнопку “=” приложение вычисляет результат и формирует из самого выражения дерево с возможностью свёртывания и развертывания подвыражений.

Вывод

В данной лабораторной работе было реализовано оконное приложение с помощью библиотеки JavaFx без использование редактора форм. Приложение имеет главное окно в котором содержится древовидный элемент и окно калькулятора. Приложение построено при помощи модели проектирования Model-View-Controller и полностью соответствует всем заявленным требованиям.