Министерство науки и высшего образования РФ

Пензенский государственный университет

Кафедра «Вычислительная техника»

Отчет

по лабораторной работе №2

по дисциплине «Программирование на языке Java»

на тему: «Работа с коллекциями объектов»

Вариант 2

Выполнили студенты группы 23ВВВ3:

Балаев Г. С.

Саранцев. Е. А.

Приняли:

Юрова О. В.

Карамышева Н. С.

Пенза, 2026

Цель работы

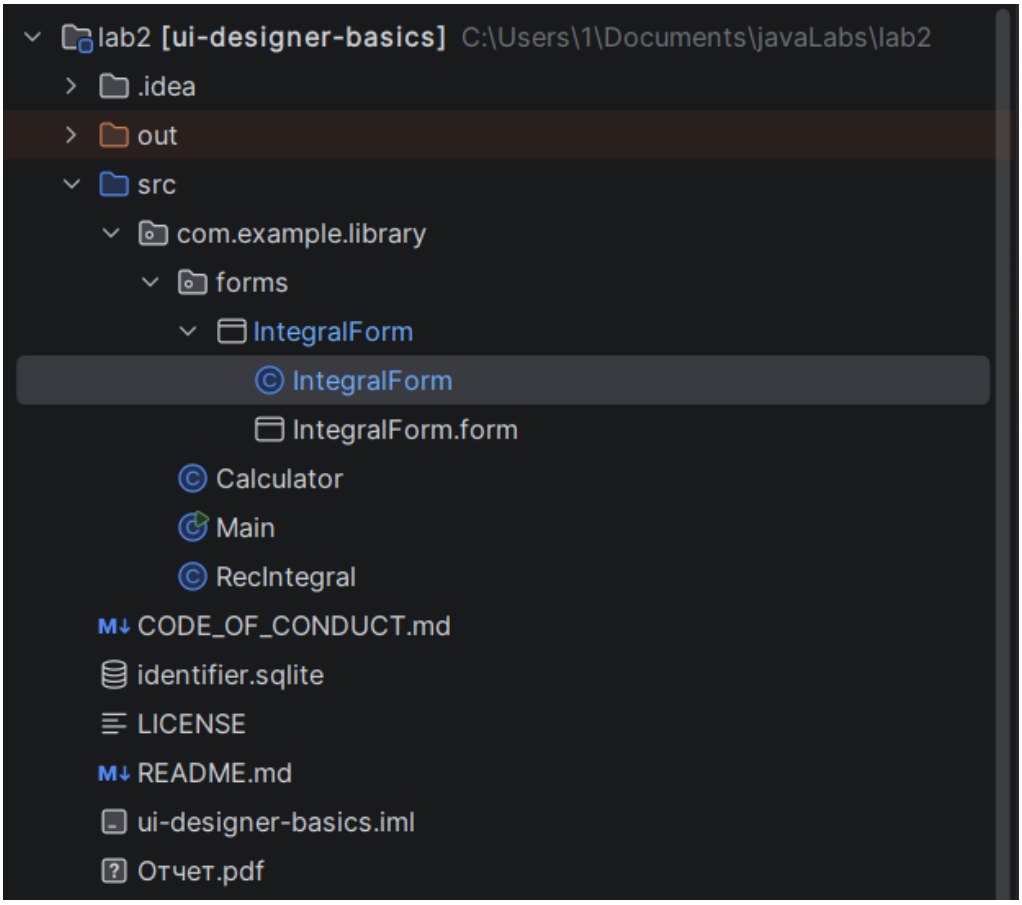
изучить библиотеку стандартных коллекций Java Collections Framework, позволяющую хранить различные структуры данных.

Лабораторное задание

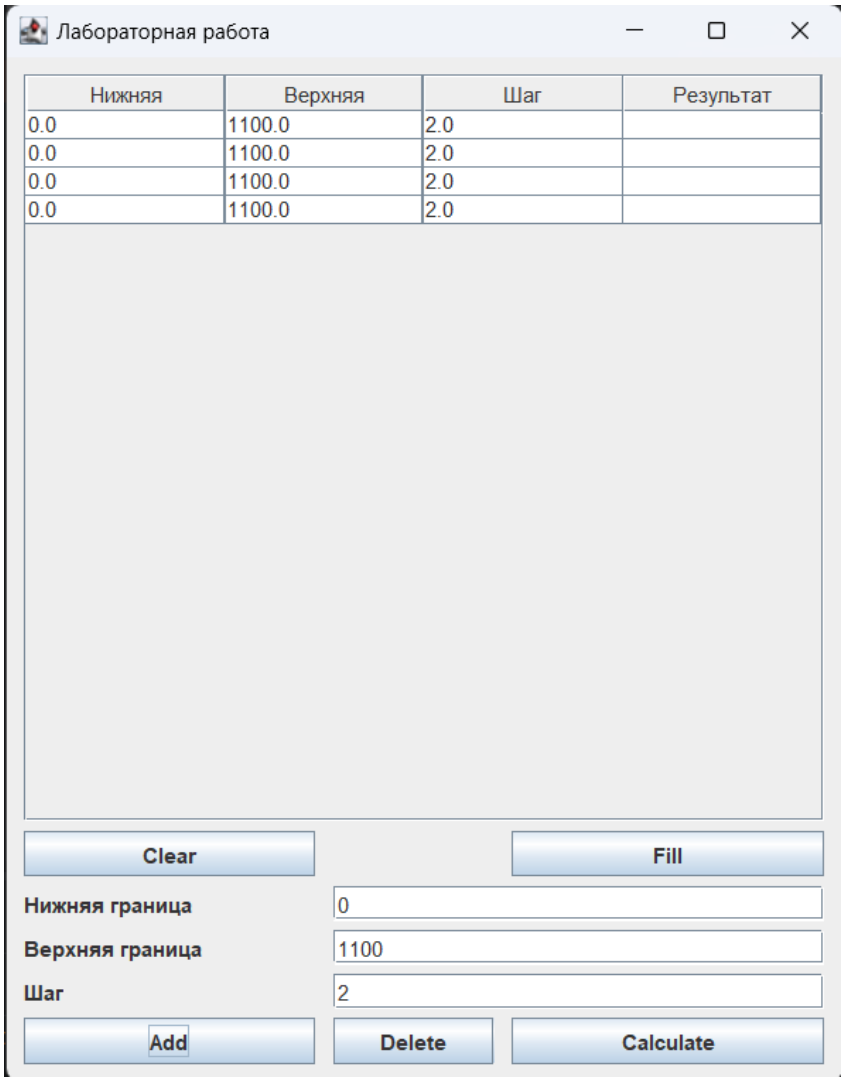
Модифицировать приложение из предыдущей лабораторной работы, реализовав хранение данных таблицы с использованием библиотеки коллекций. Для этого реализовать класс RecIntegral, способный хранить одну запись таблицы. В качестве класса-коллекции выбрать LinkedList. Кроме того, добавить пару кнопок: очистить / заполнить, которые будут очищать таблицу и заполнять ее данными из коллекции соответственно.

Ход работы:

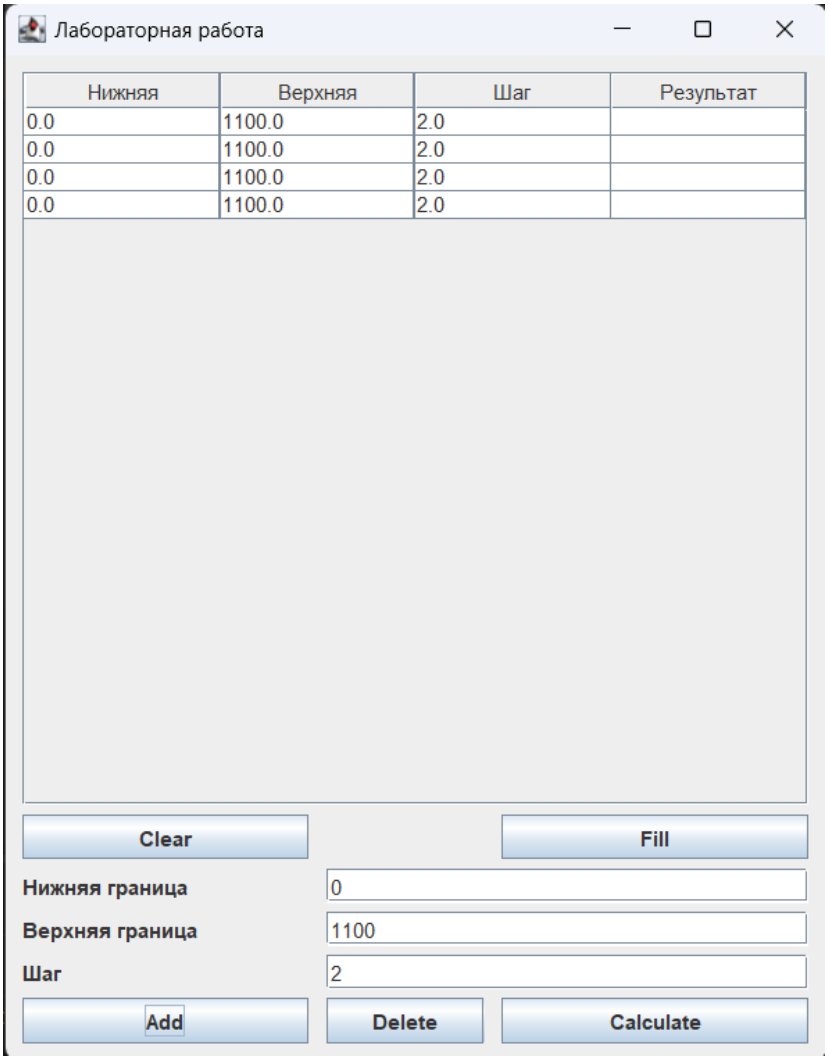
1. Открыли прошлый проект в IDE JetBrains и создали новый класс RecIntegral.



2. Добавили кнопки Clear/Fill, а так же изменили функционал кнопок Add/Delete, чтобы они добавляли/удаляли значение в/из коллекции.



Результат работы программы:



Вывод:

В ходе данной лабораторной работы мы научились работе с коллекциями.

Листинг: Main.java:

```
package com.example.library;
 import javax.swing.*; import
com.example.library.forms.IntegralForm;
 public class Main {     public static void
main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                JFrame frame = new JFrame("Лабораторная работа");

                IntegralForm form = new IntegralForm();
frame.setContentPane(form.getRootPanel());

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.pack();                  frame.setLocationRelativeTo(null);
frame.setVisible(true);
            }
        });
    }
}
```

IntegralForm.java:

```
package com.example.library.forms;


import com.example.library.RecIntegral;


import javax.swing.*;

import javax.swing.table.DefaultTableModel;

import java.util.ArrayList;

import java.util.List;


public class IntegralForm {

    private JPanel rootPanel;

    private JTextField fieldLowerBound;

    private JTextField fieldUpperBound;

    private JTextField fieldStep;

    private JButton btnAdd;

    private JButton btnDelete;

    private JButton btnCalculate;

    private JTable table1;

    private boolean clrClicked;

    private JButton btnClear;
```

```
private JButton btnFill;

private DefaultTableModel model;

private List<RecIntegral> listRecords = new ArrayList<>();

public IntegralForm() {
    String[] columns = {"Нижняя", "Верхняя", "Шаг", "Результат"};
    model = new DefaultTableModel(columns, 0) {
        @Override
        public boolean isCellEditable(int row, int column) {
            return column != 3;
        }
        @Override
        public void setValueAt(Object aValue, int row, int column) {
            super.setValueAt(aValue, row, column);

            if (row >= 0 && row < listRecords.size() && column < 3) {

                double val = Double.parseDouble(aValue.toString());
                RecIntegral rec = listRecords.get(row);

                if (column == 0) rec.setLowerBound(val);
                else if (column == 2) rec.setUpperBound(val);
                else if (column == 2) rec.setStep(val);

                rec.setResult(0);
                super.setValueAt("", row, 3);

            }
        }
    };
    table1.setModel(model);

    btnAdd.addActionListener(e -> {
        if (clrClicked == false) {
```

```
                double low = Double.parseDouble(fieldLowerBound.getText());
                double high = Double.parseDouble(fieldUpperBound.getText());
                double step = Double.parseDouble(fieldStep.getText());

                RecIntegral record = new RecIntegral(low, high, step, 0);
                listRecords.add(record);
                model.addRow(new Object[]{low, high, step, ""});
            }
        });

        btnCalculate.addActionListener(e -> {
            int row = table1.getSelectedRow();
            if (row != -1) {
                double left = Double.parseDouble(model.getValueAt(row,
0).toString());
                double right = Double.parseDouble(model.getValueAt(row,
1).toString());
                double step = Double.parseDouble(model.getValueAt(row,
2).toString());

                double res = RecIntegral.calculate(left, right, step);

                RecIntegral rec = listRecords.get(row);
                rec.setResult(res);
                rec.setLowerBound(left);
                rec.setUpperBound(right);
                rec.setStep(step);

                model.setValueAt(String.format("%.4f", res), row, 3);
            }
        });

        btnDelete.addActionListener(e -> {
            int row = table1.getSelectedRow();
            if (row != -1) {
                listRecords.remove(row);
                model.removeRow(row);
```

```
            }
        });

        btnClear.addActionListener(e -> {
            clrClicked = true;
            model.setRowCount(0);
            //listRecords.clear();
        });

        btnFill.addActionListener(e -> {
            clrClicked = false;
            model.setRowCount(0);
            for (RecIntegral rec : listRecords) {
                String resStr = rec.getResult() == 0 ? "" :
String.format("%.4f", rec.getResult());
                model.addRow(new Object[]{
                        rec.getLowerBound(),
                        rec.getUpperBound(),
                        rec.getStep(),
                        resStr
                });
            }
        });
    }

    public JPanel getRootPanel() {
        return rootPanel;
    }
}
```

RecIntegral.java:

```
package com.example.library;

import static java.lang.Math.sin;

public class RecIntegral {
```

```
    private double lowerBound;
    private double upperBound;
    private double step;
    private double result;

    public RecIntegral(double lowerBound, double upperBound, double
step, double result) {
        this.lowerBound = lowerBound;
        this.upperBound = upperBound;
        this.step = step;
        this.result = result;
    }

    public double getLowerBound() { return lowerBound; }
    public void setLowerBound(double lowerBound) { this.lowerBound =
lowerBound; }

    public double getUpperBound() { return upperBound; }
    public void setUpperBound(double upperBound) { this.upperBound =
upperBound; }

    public double getStep() { return step; }
    public void setStep(double step) { this.step = step; }

    public double getResult() { return result; }
    public void setResult(double result) { this.result = result; }

    private static double f(double x) {
        return sin(x);
    }
    public static double calculate(double left, double right, double
step) {
        int n = (int)((right - left) / step);
        double result = 0;
        double last = left + n * step;
```

```
        for (int i = 0; i < n - 1; i++) {
            double x0 = left + i * step;
            double x1 = last < right ? (f(last) + f(right)) * (right -
last) / 2 : left + (i + 1) * step;
            result += (f(x0) + f(x1)) * step / 2;
        }
        return result;
    }
}
```