Пензенский государственный университет

Факультет вычислительной техники Кафедра «Вычислительная техника»

Программирование на языке JAVA

Отчет о лабораторной работе № 2 по теме «Работа с коллекциями объектов»

Вариант № 10

Выполнили студенты гр. 21ВВП2

Савосин В. А.  
Кирмасов И. Д.

Проверили:

Юрова О. В.  
Карамышева Н. С.

2024

**Цель работы:** Изучить библиотеку стандартных коллекций Java Collections Framework, позволяющую хранить различные структуры данных.

**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.

**Листинг программы:**

package lab1;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableCellEditor;

import javax.swing.table.TableCellRenderer;

import javax.swing.table.TableColumn;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.LinkedList;

public class Lab1 extends JFrame {

private DefaultTableModel tableModel;

private LinkedList<RecIntegral> tableData; // Хранение данных таблицы в коллекции LinkedList

public Lab1() {

setTitle("Integration Calculator - cos(x) ");

tableData = new LinkedList<>();

// Создаем таблицу

String[] columns = {"Нижняя граница", "Верхняя граница", "Длина интервала", "Результат"};

DefaultTableModel tableModel = new DefaultTableModel(columns, 0){

@Override

public boolean isCellEditable(int row, int column){

// разрешаем редактирование всех столбцов кроме 4го

return column != 3;

}

};

JTable table = new JTable(tableModel){

@Override

public TableCellRenderer getCellRenderer(int row, int column){

if (column == 3){

// запрещаем редактирование 4го столбца

return getDefaultRenderer(Object.class);

}

else{

return super.getCellRenderer(row, column);

}

}

@Override

public TableCellEditor getCellEditor(int row, int column){

if (column == 3){

return getDefaultEditor(Object.class);

}

else{

return super.getCellEditor(row, column);

}

}

};

TableColumn column = table.getColumnModel().getColumn(3);

column.setCellEditor(null);

// Создаем текстовые поля для ввода данных

JTextField lowerBoundField = new JTextField(10);

JTextField upperBoundField = new JTextField(10);

JTextField intervalField = new JTextField(10);

JButton clearButton = new JButton("Очистить");

clearButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

tableModel.setRowCount(0); // Очищаем таблицу

}

});

// Добавить кнопку "Заполнить"

JButton fillButton = new JButton("Заполнить");

fillButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

for (RecIntegral rec : tableData) {

tableModel.addRow(new Object[]{rec.getLowerBound(), rec.getUpperBound(), rec.getIntervals()});

}

}

});

// Создаем кнопку "Добавить" и задаем действие при нажатии

JButton addButton = new JButton("+");

addButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

double lowerBound = Double.parseDouble(lowerBoundField.getText());

double upperBound = Double.parseDouble(upperBoundField.getText());

double intervals = Double.parseDouble(intervalField.getText());

tableModel.addRow(new Object[]{lowerBound, upperBound, intervals, ""});

RecIntegral newElement = new RecIntegral(lowerBound, upperBound, intervals);

tableData.add(newElement);

}

});

// Создаем кнопку "Удалить" и задаем действие при нажатии

JButton deleteButton = new JButton("-");

deleteButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int selectedRow = table.getSelectedRow();

if (selectedRow != -1) {

tableModel.removeRow(selectedRow);

}

}

});

JButton calculateButton = new JButton("Вычислить");

calculateButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int selectedRow = table.getSelectedRow();

double lowerBound = Double.parseDouble(tableModel.getValueAt(selectedRow, 0).toString());

double upperBound = Double.parseDouble(tableModel.getValueAt(selectedRow, 1).toString());

double intervals = Double.parseDouble(tableModel.getValueAt(selectedRow, 2).toString());

double result = calculateIntegration(lowerBound, upperBound, intervals);

if (selectedRow != -1) {

tableModel.setValueAt(result, selectedRow, 3);

}

}

});

// Создаем панель для компонентов

JPanel panel = new JPanel();

panel.add(lowerBoundField);

panel.add(upperBoundField);

panel.add(intervalField);

panel.add(addButton);

panel.add(deleteButton);

panel.add(calculateButton);

panel.add(clearButton); // Добавляем кнопку "Очистить" на панель

panel.add(fillButton);

// Добавляем панель и таблицу на окно

add(panel, "South");

JScrollPane scrollPane = new JScrollPane(table);

add(scrollPane);

setSize(700, 400);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

}

// Метод для вычисления интеграла функции cos(x)

private double calculateIntegration(double lowerBound, double upperBound, double step) {

double sum = 0.0;

double x = lowerBound;

while (x < upperBound) {

double fx1 = Math.cos(x); // значение функции в левой точке отрезка

double fx2 = Math.cos(Math.min(x + step, upperBound)); // значение функции в правой точке отрезка

sum += (fx1 + fx2) \* Math.min(step, upperBound - x) / 2;

x += step;

}

//округлим ответ

int decimalPlaces = 4; // количество знаков после запятой, до которого нужно округлить

sum = Math.round(sum \* Math.pow(10, decimalPlaces)) / Math.pow(10, decimalPlaces);

return sum; // возвращаем значение интеграла

}

public static void main(String[] args) {

new Lab1(); // создаем экземпляр приложения

}

}

**Класс RecIntegral:**

package lab1;

/\*\*

\*

\* @author User

\*/

public class RecIntegral {

private double lowerBound;

private double upperBound;

private double intervals;

private double result;

public RecIntegral(double lowerBound, double upperBound, double intervals) {

this.lowerBound = lowerBound;

this.upperBound = upperBound;

this.intervals = intervals;

}

public double getLowerBound() {

return lowerBound;

}

public double getUpperBound() {

return upperBound;

}

public double getIntervals() {

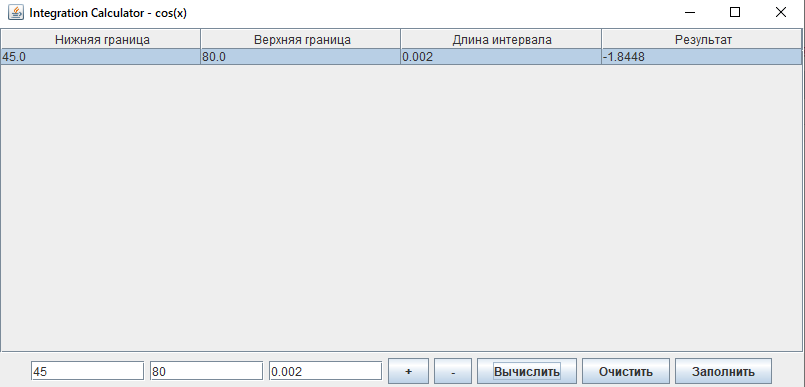
return intervals;

}

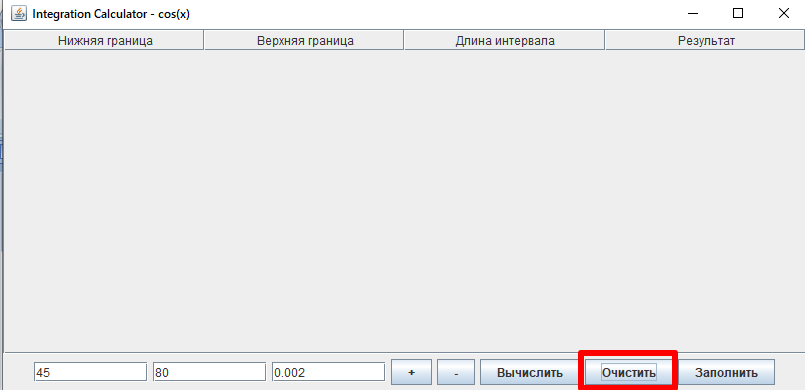
}

**Вывод программы:**

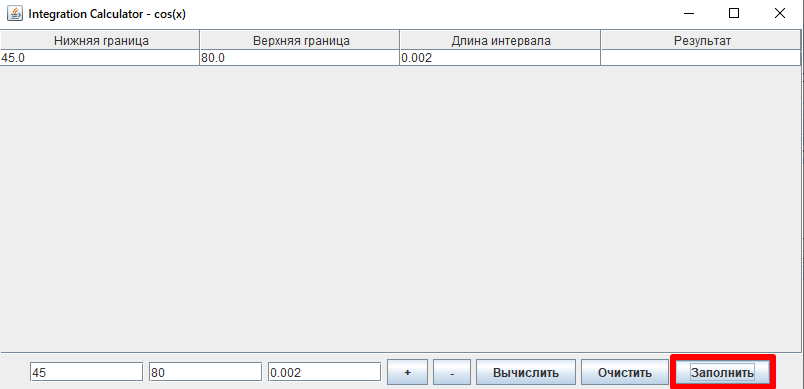
Вводим дааные:



Очищаем данные:



Заполняем данные:



**Вывод:** Изучили библиотеку стандартных коллекций Java Collections Framework, позволяющую хранить различные структуры данных.