Пензенский государственный университет

Факультет вычислительной техники

Кафедра «Вычислительная техника»

Отчет о лабораторной работе №4

по дисциплине «Разработка кроссплатформенных приложений»

на тему «Работа с файлами»

Выполнили: ст-ты гр. 22ВОЭ1

Старёхин Б.А.

Николаев Д.А.

Проверили: к.т.н., доцент каф. ВТ

Юрова О.В.

Карамышева Н.С.

2025

Цель работы

Изучить работу с файлами и механизмы сериализации данных.

Задание

Модифицировать приложение из предыдущей лабораторной работы, реализовав сохранение в файл и загрузку данных из файла. Предусмотреть сохранение данных, как в текстовом виде, так и в двоичном (с использованием механизма сериализации). Для этого нужно добавить 4 кнопки для сохранения и загрузки в текстовом и двоичном виде соответственно. Кроме того, в программе нужно предусмотреть использование стандартного диалога открытия файла (JFileChooser).

Ход работы

Листинг программы

package my.contacteditor;

import javax.swing.table.DefaultTableModel;

import javax.swing.JOptionPane;

import javax.swing.\*;

import java.util.ArrayList;

import java.util.LinkedList;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.util.Scanner;

import java.io.\*;

public class ContactEditorUI extends javax.swing.JFrame {

private LinkedList <RecIntegral> records = new LinkedList();

private ArrayList<RecTableBin> rectable = new ArrayList<>();

ArrayList<RecIntegral> arrInteg = new ArrayList<> ();

private FileInputOutput fio = new FileInputOutput();

private DefaultTableModel tModel;

private File file;

private Boolean flag = false;

public ContactEditorUI() {

initComponents();

tModel = (DefaultTableModel) jTable1.getModel();

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jPanel1 = new javax.swing.JPanel();

jLabel1 = new javax.swing.JLabel();

jTextField1 = new javax.swing.JTextField();

jLabel2 = new javax.swing.JLabel();

jTextField2 = new javax.swing.JTextField();

jLabel3 = new javax.swing.JLabel();

jTextField3 = new javax.swing.JTextField();

jButton1 = new javax.swing.JButton();

jButton2 = new javax.swing.JButton();

jButton3 = new javax.swing.JButton();

jScrollPane1 = new javax.swing.JScrollPane();

jTable1 = new javax.swing.JTable();

jButton4 = new javax.swing.JButton();

jButton5 = new javax.swing.JButton();

jButton6 = new javax.swing.JButton();

jButton7 = new javax.swing.JButton();

jButton8 = new javax.swing.JButton();

jButton9 = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "", javax.swing.border.TitledBorder.DEFAULT\_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT\_POSITION, new java.awt.Font("Tahoma", 1, 12))); // NOI18N

jLabel1.setText("верхний предел");

jTextField1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jTextField1ActionPerformed(evt);

}

});

jLabel2.setText("нижний предел");

jLabel3.setText("шаг");

jButton1.setText("добавить");

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

jButton2.setText("вычислить");

jButton2.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton2ActionPerformed(evt);

}

});

jButton3.setText("удалить");

jButton3.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton3ActionPerformed(evt);

}

});

jTable1.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N

jTable1.setModel(new javax.swing.table.DefaultTableModel(

new Object [][] {

},

new String [] {

"верх.пред", "ниж.пред", "шаг", "результат"

}

) {

boolean[] canEdit = new boolean [] {

false, false, false, false

};

public boolean isCellEditable(int rowIndex, int columnIndex) {

return canEdit [columnIndex];

}

});

jScrollPane1.setViewportView(jTable1);

jButton4.setText("заполнить");

jButton4.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton4ActionPerformed(evt);

}

});

jButton5.setText("очистить");

jButton5.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton5ActionPerformed(evt);

}

});

jButton6.setText("сохранить в ТВ");

jButton6.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton6ActionPerformed(evt);

}

});

jButton7.setText("загрузить из ТВ");

jButton7.setActionCommand("загрузить ТВ");

jButton7.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton7ActionPerformed(evt);

}

});

jButton8.setText("сохранить в ДВ");

jButton8.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton8ActionPerformed(evt);

}

});

jButton9.setText("загрузить из ДВ");

jButton9.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton9ActionPerformed(evt);

}

});

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jScrollPane1)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jTextField1)

.addComponent(jLabel2)

.addComponent(jTextField2)

.addComponent(jLabel3)

.addComponent(jTextField3))

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(107, 107, 107)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jButton5)

.addGap(35, 35, 35)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jButton3, javax.swing.GroupLayout.PREFERRED\_SIZE, 87, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButton8))

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jButton2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButton7))))

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jButton4)

.addGap(29, 29, 29)

.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED\_SIZE, 87, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButton6))))

.addGroup(jPanel1Layout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButton9)))

.addGap(26, 26, 26)))

.addContainerGap())

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(23, 23, 23)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jButton4)

.addComponent(jButton1)

.addComponent(jButton6)))

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jButton5)

.addComponent(jButton2)

.addComponent(jButton7)))

.addGap(18, 18, 18)

.addComponent(jLabel3))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jButton3)

.addComponent(jButton8)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(61, 61, 61))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()

.addComponent(jButton9)

.addGap(47, 47, 47)))

.addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT\_SIZE, 171, Short.MAX\_VALUE)

.addContainerGap())

);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

pack();

}// </editor-fold>

private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

double vpred;

double npred;

double step;

try {

vpred = Double.parseDouble(jTextField1.getText());

npred = Double.parseDouble(jTextField2.getText());

step = Double.parseDouble(jTextField3.getText());

if (vpred < 0.000001 || vpred > 1000000 ||

npred < 0.000001 || npred > 1000000 ||

step < 0.000001 || step > 1000000) {

throw new CustomException("Значения должны находиться в диапазоне от 0.000001 до 1000000.");

}

if (vpred < npred){

throw new CustomException("Нижний предел должен быть меньше верхнего предела.");

}

if ((vpred - npred) < step){

throw new CustomException("Шаг интегрирования должен быть меньше интервала.");

}

DefaultTableModel tModel = (DefaultTableModel) jTable1.getModel();

tModel.addRow(new Object[] {vpred, npred, step});

arrInteg.add(new RecIntegral(vpred, npred, step));

jTextField1.setText("");

jTextField2.setText("");

jTextField3.setText("");

} catch (CustomException ex) {

JOptionPane.showMessageDialog(this, ex.getMessage(),

"Ошибка диапазона", JOptionPane.WARNING\_MESSAGE);

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(this, "Некорректное значение введено. Попробуйте снова.",

"Ошибка ввода", JOptionPane.ERROR\_MESSAGE);

}

}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

double vp = 0;

double np = 0;

double st = 0;

double result = 0;

DefaultTableModel tModel = (DefaultTableModel) jTable1.getModel();

int rowNom = jTable1.getSelectedRow();

if (rowNom == -1) {

JOptionPane.showMessageDialog(null, "выберите строку!");

}

else {

vp = Double.parseDouble(tModel.getValueAt(rowNom, 0).toString());

np = Double.parseDouble(tModel.getValueAt(rowNom, 1).toString());

st = Double.parseDouble(tModel.getValueAt(rowNom, 2).toString());

result = CalcInt (vp,np,st);

tModel.setValueAt(result, rowNom, 3);

arrInteg.set (rowNom, new RecIntegral (np, vp, st, result));

}

try {

rectable.add(new RecTableBin(np, vp, st, result));

records.add(new RecIntegral (np,vp,st,result));

}

catch (ErrRecIntegralVal ex) {

Logger.getLogger(ContactEditorUI.class.getName()).log(Level.SEVERE, null, ex);

}

}

public double CalcInt (double vp, double npr, double step){

double start, h, sumS=0;

start = vp;

while (start > npr) {

h = Math.min(step, start - npr);

sumS += h \* (Math.exp(-start) + Math.exp(-(start - h))) / 2;

start -= h;

}

return sumS;

}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

DefaultTableModel tModel = (DefaultTableModel) jTable1.getModel();

int rowNom = jTable1.getSelectedRow();

if (rowNom == -1) {

JOptionPane.showMessageDialog(null, "выберите строку!");

}

else {

tModel.removeRow(rowNom);

}

}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {

DefaultTableModel tModel = (DefaultTableModel) jTable1.getModel();

tModel.setNumRows(0);

}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

DefaultTableModel tModel = (DefaultTableModel) jTable1.getModel ();

tModel.setNumRows(0);

for (RecIntegral item: arrInteg) {

tModel.addRow(new Object[] {item.getVP(),item.getNP(),item.getST(),item.getRES()});

}

}

private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {

JFileChooser fileChooser = new JFileChooser();

int option = fileChooser.showSaveDialog(null);

if (option == JFileChooser.APPROVE\_OPTION) {

file = fileChooser.getSelectedFile();

flag = true;

}

else{

flag = false;

}

if (flag == true){

fio.OutputTxt(records, file);

}

else if (flag == false){

fio.OutputTxt(records);

}

}

private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

JFileChooser fileChooser = new JFileChooser();

int option = fileChooser.showOpenDialog(null);

if (option == JFileChooser.APPROVE\_OPTION) {

file = fileChooser.getSelectedFile();

flag = true;

}

else{

flag = false;

}

tModel.setRowCount(0);

if (flag == true){

fio.InputTxt(tModel, file);

}

else if (flag == false){

fio.InputTxt(tModel);

}

}

private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

JFileChooser fileChooser = new JFileChooser();

int option = fileChooser.showSaveDialog(null);

if (option == JFileChooser.APPROVE\_OPTION) {

file = fileChooser.getSelectedFile();

flag = true;

}

else{

flag = false;

}

try {

if (flag == true){

fio.OutputBin(rectable, file);

jButton6.setEnabled(false);

}

else if (flag == false){

fio.OutputBin(rectable);

}

jButton8.setEnabled(true);

}

catch (IOException ex) {

Logger.getLogger(ContactEditorUI.class.getName()).log(Level.SEVERE, null, ex);

}

}

private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

JFileChooser fileChooser = new JFileChooser();

int option = fileChooser.showOpenDialog(null);

if (option == JFileChooser.APPROVE\_OPTION) {

file = fileChooser.getSelectedFile();

flag = true;

}

else{

flag = false;

}

tModel.setRowCount(0);

rectable.clear();

try {

if (flag == true){

fio.InputBin(tModel,rectable, file);

}

else if (flag == false){

fio.InputBin(tModel,rectable);

}

}

catch (IOException | ClassNotFoundException ex) {

Logger.getLogger(ContactEditorUI.class.getName()).log(Level.SEVERE, null, ex);

}

}

public static void main(String args[]) {

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new ContactEditorUI().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JButton jButton2;

private javax.swing.JButton jButton3;

private javax.swing.JButton jButton4;

private javax.swing.JButton jButton5;

private javax.swing.JButton jButton6;

private javax.swing.JButton jButton7;

private javax.swing.JButton jButton8;

private javax.swing.JButton jButton9;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JPanel jPanel1;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JTable jTable1;

private javax.swing.JTextField jTextField1;

private javax.swing.JTextField jTextField2;

private javax.swing.JTextField jTextField3;

// End of variables declaration

}

package my.contacteditor;

public class RecIntegral {

private double vp;

private double np;

private double st;

private double result;

private boolean ValidValue(double value) {

return value >= 0.000001 && value <= 1000000;

}

public RecIntegral (double np, double vp, double st){

this.vp = vp;

this.np = np;

this.st = st;

this.result = 0.0;

}

public RecIntegral (double np, double vp, double st, double result){

this.vp = vp;

this.np = np;

this.st = st;

this.result = result;

}

public RecIntegral (double np, double vp, double st, String someAdditionalParam) throws CustomException {

if (np < 0.000001 || np > 1000000) {

throw new CustomException ("Значение должно быть в диапазоне от 0.000001 до 1000000.");

}

if (vp < 0.000001 || vp > 1000000) {

throw new CustomException ("Значение должно быть в диапазоне от 0.000001 до 1000000.");

}

if (st < 0.000001 || st > 1000000) {

throw new CustomException ("Значение должно быть в диапазоне от 0.000001 до 1000000.");

}

if (np > vp) {

throw new CustomException ("Нижний предел должен быть меньше верхнего предела.");

}

if ((vp - np) < st) {

throw new CustomException ("Шаг интегрирования должен быть меньше интервала.");

}

this.vp = vp;

this.np = np;

this.st = st;

}

public double getNP(){

return this.np;

}

public double getVP(){

return this.vp;

}

public double getST(){

return this.st;

}

public double getRES(){

return result;

}

public void setRES(double result){

this.result = result;

}

}

package my.contacteditor;

public class CustomException extends Exception {

public CustomException(String message) {

super (message);

}

}

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

package my.contacteditor;

public class ErrRecIntegralVal extends Exception {

public ErrRecIntegralVal(String message) {

super (message);

}

}

package my.contacteditor;

public class RecIntegral {

private double vp;

private double np;

private double st;

private double result;

private boolean ValidValue(double value) {

return value >= 0.000001 && value <= 1000000;

}

public RecIntegral (double np, double vp, double st){

this.vp = vp;

this.np = np;

this.st = st;

this.result = 0.0;

}

public RecIntegral (double np, double vp, double st, double result){

this.vp = vp;

this.np = np;

this.st = st;

this.result = result;

}

public RecIntegral (double np, double vp, double st, String someAdditionalParam) throws CustomException {

if (np < 0.000001 || np > 1000000) {

throw new CustomException ("Значение должно быть в диапазоне от 0.000001 до 1000000.");

}

if (vp < 0.000001 || vp > 1000000) {

throw new CustomException ("Значение должно быть в диапазоне от 0.000001 до 1000000.");

}

if (st < 0.000001 || st > 1000000) {

throw new CustomException ("Значение должно быть в диапазоне от 0.000001 до 1000000.");

}

if (np > vp) {

throw new CustomException ("Нижний предел должен быть меньше верхнего предела.");

}

if ((vp - np) < st) {

throw new CustomException ("Шаг интегрирования должен быть меньше интервала.");

}

this.vp = vp;

this.np = np;

this.st = st;

}

public double getNP(){

return this.np;

}

public double getVP(){

return this.vp;

}

public double getST(){

return this.st;

}

public double getRES(){

return result;

}

public void setRES(double result){

this.result = result;

}

}

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

package my.contacteditor;

import java.awt.HeadlessException;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.util.ArrayList;

import java.util.LinkedList;

import javax.swing.JOptionPane;

import javax.swing.table.DefaultTableModel;

public class FileInputOutput{

public void OutputTxt (LinkedList <RecIntegral> records){

try (BufferedWriter writer = new BufferedWriter(new FileWriter(new File ("data.txt"), false))) {

for (RecIntegral rec : records) {

String line = String.valueOf(rec.getNP()) + "/" + String.valueOf(rec.getNP()) + "/" + String.valueOf(rec.getST()) + "/" + String.valueOf(rec.getRES());

writer.write(line);

writer.newLine();

}

//writer.close();

JOptionPane.showMessageDialog(null, "Данные успешно сохранены в текстовый файл.", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (IOException e) {

JOptionPane.showMessageDialog(null, "Ошибка при сохранении файла: " + e.getMessage(), "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

public void OutputTxt (LinkedList <RecIntegral> records, File file){

try (BufferedWriter writer = new BufferedWriter(new FileWriter(file, false))) {

for (RecIntegral rec : records) {

String line = String.valueOf(rec.getNP()) + "/" + String.valueOf(rec.getVP()) + "/" + String.valueOf(rec.getST()) + "/" + String.valueOf(rec.getRES());

writer.write(line);

writer.newLine();

}

//writer.close();

JOptionPane.showMessageDialog(null, "Данные успешно сохранены в текстовый файл.", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (IOException e) {

JOptionPane.showMessageDialog(null, "Ошибка при сохранении файла: " + e.getMessage(), "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

public void InputTxt (DefaultTableModel tModel){

try (BufferedReader reader = new BufferedReader(new FileReader("data.txt"))) {

String line;

while ((line = reader.readLine()) != null) {

String[] data = line.split("/");

Double[] rowData = new Double[data.length];

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

rowData[i] = Double.valueOf(data[i].trim()); // Преобразование в Double

}

tModel.addRow(rowData);

}

}

catch (IOException e) {

JOptionPane.showMessageDialog(null, "Ошибка при загрузке данных: " + e.getMessage());

}

}

public void InputTxt (DefaultTableModel tModel, File file){

try (BufferedReader reader = new BufferedReader(new FileReader(file))) {

String line;

while ((line = reader.readLine()) != null) {

String[] data = line.split("/");

Double[] rowData = new Double[data.length];

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

rowData[i] = Double.valueOf(data[i].trim());

}

tModel.addRow(rowData);

}

}

catch (IOException e) {

JOptionPane.showMessageDialog(null, "Ошибка при загрузке данных: " + e.getMessage());

}

}

public void OutputBin (ArrayList<RecTableBin> rectable) throws FileNotFoundException, IOException{

if (rectable == null || rectable.isEmpty()) {

JOptionPane.showMessageDialog(null, "Нет данных для сохранения.", "Ошибка", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

FileOutputStream fs = new FileOutputStream("databin.ser");

try (ObjectOutputStream oos = new ObjectOutputStream(fs)) {

oos.writeObject(rectable);

//oos.close();

}

JOptionPane.showMessageDialog(null, "Данные успешно сохранены в двоичный файл.", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (IOException e) {

JOptionPane.showMessageDialog(null, "Ошибка при сохранении файла: " + e.getMessage(), "Ошибка", JOptionPane.ERROR\_MESSAGE);

} catch (HeadlessException ex) {

throw new RuntimeException(ex);

}

}

public void OutputBin (ArrayList<RecTableBin> rectable, File file) throws FileNotFoundException, IOException{

if (rectable == null || rectable.isEmpty()) {

JOptionPane.showMessageDialog(null, "Нет данных для сохранения.", "Ошибка", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

FileOutputStream fs = new FileOutputStream(file);

try (ObjectOutputStream oos = new ObjectOutputStream(fs)) {

oos.writeObject(rectable);

//oos.close();

}

JOptionPane.showMessageDialog(null, "Данные успешно сохранены в двоичный файл.", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (IOException e) {

JOptionPane.showMessageDialog(null, "Ошибка при сохранении файла: " + e.getMessage(), "Ошибка", JOptionPane.ERROR\_MESSAGE);

} catch (HeadlessException ex) {

throw new RuntimeException(ex);

}

}

public void InputBin (DefaultTableModel tModel, ArrayList<RecTableBin> rectable) throws FileNotFoundException, IOException, ClassNotFoundException{

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream("databin.ser"))) {

rectable = (ArrayList<RecTableBin>) ois.readObject();

for (RecTableBin rec : rectable) {

tModel.addRow(new Double[]{rec.getVP(), rec.getNP(), rec.getST(), rec.getRES()});

}

//ois.close();

JOptionPane.showMessageDialog(null, "Данные успешно загружены из двоичного файла.", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (IOException | ClassNotFoundException e) {

JOptionPane.showMessageDialog(null, "Ошибка при загрузке файла: " + e.getMessage(), "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

public void InputBin (DefaultTableModel tModel, ArrayList<RecTableBin> rectable, File file) throws FileNotFoundException, IOException, ClassNotFoundException{

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(file))) {

rectable = (ArrayList<RecTableBin>) ois.readObject();

for (RecTableBin rec : rectable) {

tModel.addRow(new Double[]{rec.getVP(), rec.getNP(), rec.getST(), rec.getRES()});

}

// ois.close();

JOptionPane.showMessageDialog(null, "Данные успешно загружены из двоичного файла.", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (IOException | ClassNotFoundException e) {

JOptionPane.showMessageDialog(null, "Ошибка при загрузке файла: " + e.getMessage(), "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

}

Словесное описание алгоритма

1. Объявление метода, который обрабатывает событие нажатия кнопки jButton1.
2. Объявление трех переменных типа double: vpred, npred и step, которые будут использоваться для хранения значений.
3. Получение текста из текстового поля jTextField1, преобразование его в число типа double и присвоение переменной vpred.
4. Получение текста из jTextField2, преобразование в double и присвоение переменной npred.
5. Получение текста из jTextField3, преобразование в double и присвоение переменной step.
6. Проверяем, находятся ли значения vpred, npred и step в заданном диапазоне.
7. Проверяем, npred должен быть меньше чем vpred.
8. Проверяем, шаг интегрирования должен быть меньше интервла.
9. Если хотя бы одно из значений не корректно, то выбрасываем исключение.
10. Получение модели таблицы из jTable1 и приведение ее к типу DefaultTableModel, чтобы можно было добавлять строки.
11. Добавление новой строки в модель таблицы с тремя значениями: vpred, npred и step.
12. Добавление нового объекта RecIntegral в коллекцию arrInteg.
13. Очистка текстовых полей после добавления данных в таблицу.
14. Блок catch обрабатывает исключение типа CustomException, которое может быть выброшено в предыдущем блоке try, переменная ex будет содержать информацию об этом исключении.
15. Здесь вызывается метод showMessageDialog класса JOptionPane, который отображает диалоговое окно с сообщением, this указывает на текущий объект, ex.getMessage() возвращает сообщение об ошибке, которое было передано при создании исключения CustomException, JOptionPane.WARNING\_MESSAGE указывает, что это предупреждающее сообщение, что визуально выделяет его в диалоговом окне.
16. Блок catch обрабатывает исключение типа NumberFormatException, которое возникает, если введенные данные не могут быть преобразованы в число.
17. Вызывается метод showMessageDialog, чтобы отобразить сообщение об ошибке, сообщение информирует пользователя о том, что введенное значение некорректно и предлагает попробовать снова, JOptionPane.ERROR\_MESSAGE указывает, что это сообщение об ошибке, что также визуально выделяет его в диалоговом окне.
18. Объявление метода, который обрабатывает событие нажатия кнопки jButton2.
19. Объявление четырех переменных типа double: vp, np, st и result, которые будут использоваться для хранения значений из выбранной строки таблицы и результата вычислений.
20. Получение модели таблицы из jTable1 и приведение ее к типу DefaultTableModel.
21. Получение номера выбранной строки в таблице. Если строка не выбрана, rowNom будет равен -1.
22. Проверка, выбрана ли строка. Если нет, отображается сообщение с просьбой выбрать строку.
23. Если строка выбрана, извлечение значений из выбранной строки таблицы и преобразование их в double. Значения присваиваются переменным vp, np и st.
24. Вызов метода CalcInt, который выполняет вычисления с использованием значений vp, np и st, и присвоение результата переменной result.
25. Установка значения result в четвертую колонку (индекс 3) выбранной строки таблицы.
26. Происходит вызов метода set у объекта arrInteg. Новое значение, которое устанавливается по индексу, является объект класса RecIntegral, которому передаются четыре параметра: np, vp, st и result.
27. Объявление метода CalcInt, который принимает три параметра: vp, npr и step. Метод возвращает значение типа double.
28. Объявление переменных:

start — начальное значение, инициализируется значением vp.

h — переменная для хранения текущего шага.

sumS — переменная для накопления суммы, инициализируется нулем.

1. Инициализация переменной start значением vp.
2. Начало цикла do-while, который будет выполняться хотя бы один раз.
3. Вычисление значения h, которое будет равно меньшему из step и разности npr и start. Это гарантирует, что шаг не превысит оставшееся расстояние до npr.
4. Обновление sumS с использованием метода трапеций для численного интегрирования. Здесь вычисляется площадь трапеции, основанной на текущем start и start + h.
5. Увеличение start на значение h, чтобы перейти к следующему интервалу.
6. Условие продолжения цикла: пока start меньше npr, цикл будет выполняться.
7. Возврат накопленной суммы sumS, которая представляет собой результат интегрирования.
8. Объявление метода, который обрабатывает событие нажатия кнопки jButton3.
9. Получение модели таблицы jTable1 и приведение её к типу DefaultTableModel, чтобы можно было изменять данные в таблице.
10. Получение номера выбранной строки в таблице. Если строка не выбрана, rowNom будет равен -1.
11. Проверка, выбрана ли строка.
12. Показ сообщения пользователю с просьбой выбрать строку.
13. Удаление выбранной строки из модели таблицы.
14. Объявление метода, который обрабатывает событие нажатия кнопки jButton5.
15. Получение модели таблицы jTable1 и приведение её к типу DefaultTableModel, чтобы можно было изменять данные в таблице.
16. Устанавливает количество строк в таблице в 0, что фактически очищает все текущие данные, отображаемые в jTable1.
17. Объявление метода, который обрабатывает событие нажатия кнопки jButton4.
18. Получение модели таблицы jTable1 и приведение её к типу DefaultTableModel, чтобы можно было изменять данные в таблице.
19. Устанавливает количество строк в таблице в 0, что фактически очищает все текущие данные, отображаемые в jTable1.
20. Цикл for, который проходит по всем элементам arrInteg, каждый элемент представляет собой объект RecIntegral.
21. Внутри цикла добавляем новую строку в таблицу для каждого элемента, для каждого item извлекаем значения, вызывая методы getNP(), getVP(), getST(), и getRES(). Это передается методу addRow, который добавляет данные в таблицу jTable1.
22. Объявление метода, который обрабатывает событие нажатия кнопки jButton6.
23. Создается экземпляр JFileChooser, который предоставляет стандартный диалог для выбора файлов. Используется для сохранения файла.
24. Вызывается метод showSaveDialog, который открывает диалог для сохранения файла. Метод возвращает целочисленное значение, указывающее, какое действие выбрал пользователь.
25. Проверяется, выбрал ли пользователь опцию "Сохранить".
26. Если пользователь подтвердил выбор, получаем выбранный файл с помощью метода getSelectedFile() и сохраняем его в переменной file.
27. Устанавливаем флаг flag в true, что указывает на то, что файл был успешно выбран.
28. Если пользователь не выбрал файл, устанавливаем флаг flag в false.
29. Если флаг flag равен true, вызываем метод OutputTxt класса fio, передавая ему записи и выбранный файл для сохранения.
30. Если флаг flag равен false, вызываем метод OutputTxt без указания файла.
31. Объявление метода, который обрабатывает событие нажатия кнопки jButton7
32. Создается новый экземпляр JFileChooser для открытия файла.
33. Вызывается метод showOpenDialog, который открывает диалог для выбора файла. Возвращает результат выбора пользователя.
34. Проверяем, выбрал ли пользователь файл.
35. Если файл выбран, сохраняем его в переменной file.
36. Устанавливаем флаг flag в true, чтобы указать, что файл был успешно выбран.
37. Если файл не был выбран, устанавливаем flag в false.
38. Очищаем модель таблицы, устанавливая количество строк в 0.
39. Если файл был выбран, вызываем метод InputTxt класса fio, передавая ему модель таблицы и выбранный файл для загрузки данных.
40. Если файл не был выбран, вызываем метод InputTxt без указания файла.
41. Объявляет пакет, к которому принадлежит класс.
42. Объявляет новый класс RecIntegral.
43. Объявляются четыре переменные, они будут использоваться для хранения значений в каждом объекте класса RecIntegral.
44. Объявляет конструктор класса. Он принимает три параметра: np, vp и st.
45. this используется для разграничения между полями класса и параметрами конструктора, значения параметров присваиваются соответствующим переменным экземпляра. Переменной result присваивается значение 0.0.
46. Второй конструктор класса, который принимает четыре параметра, позволяет создавать объект с уже инициализированным значением result.
47. result получает значение, переданное в конструктор.
48. Метод RecIntegral является конструктором класса RecIntegral. Он принимает четыре параметра: np, vp, st и someAdditionalParam. Конструктор может выбрасывать исключение CustomException.
49. Если значение np меньше 0.000001 или больше 1000000, выбрасывается исключение CustomException с сообщением о том, что значение должно находиться в указанном диапазоне.
50. Если значение vp меньше 0.000001 или больше 1000000, выбрасывается исключение CustomException с сообщением о том, что значение должно находиться в указанном диапазоне.
51. Если значение st меньше 0.000001 или больше 1000000, выбрасывается исключение CustomException с сообщением о том, что значение должно находиться в указанном диапазоне.
52. Если значение np меньше чем vp, то выбрасывается исключение CustomException с сообщением о том, что нижний предел должен быть меньше верхнего предела.
53. Если значение vp - np меньше чем st, то выбрасывается исключение CustomException с сообщением о том, шаг интегрирования должен быть меньше чем интервал.
54. Если все проверки пройдены, значения vp, np и st присваиваются соответствующим полям класса, что позволяет сохранить их для дальнейшего использования в экземпляре класса.
55. геттер, который возвращает значение переменной vp.
56. геттер, который возвращает значение переменной np.
57. геттер, который возвращает значение переменной st.
58. геттер, который возвращает значение переменной res.
59. Объявляем пакет, к которому принадлежит класс.
60. Объявляем новый класс CustomException, который является подклассом стандартного класса Exception. Это означает, что CustomException будет представлять собой пользовательское исключение, которое можно выбрасывать и обрабатывать в программе.
61. Конструктор класса CustomException. Он принимает строку message, которая будет содержать сообщение об ошибке, связанной с исключением.
62. Вызываем конструктор суперкласса Exception с помощью ключевого слова super. Это передает сообщение об ошибке в класс Exception, чтобы оно могло быть использовано при обработке исключения.
63. Это метод-обработчик события для кнопки jButton8. Он вызывается, когда пользователь нажимает на эту кнопку.
64. Создается новый экземпляр JFileChooser, который позволяет пользователю выбирать файлы для сохранения.
65. Открывается диалоговое окно для сохранения файла. Метод showSaveDialog возвращает результат выбора, который сохраняется в переменной option.
66. Если пользователь подтвердил выбор файла, то выбранный файл сохраняется в переменной file, и флаг flag устанавливается в true. В противном случае флаг устанавливается в false.
67. Начинается блок try, который позволяет обрабатывать исключения, которые могут возникнуть в процессе выполнения кода внутри него.
68. Если флаг flag равен true, вызывается метод OutputBin класса fio, который, вероятно, отвечает за вывод данных в бинарный файл, передавая ему rectable и file.
69. Если флаг flag равен false, вызывается тот же метод, но без указания файла.
70. Если в блоке try произошла ошибка ввода-вывода (IOException), она будет поймана, и стек исключений будет записан в лог.
71. Это метод-обработчик события для кнопки jButton9.
72. Создается новый экземпляр JFileChooser для выбора файла.
73. Открывается диалоговое окно для открытия файла. Результат выбора сохраняется в переменной option.
74. Если пользователь выбрал файл, он сохраняется в переменной file, и флаг flag устанавливается в true. В противном случае флаг устанавливается в false.
75. Сначала обнуляется количество строк в tModel, а затем очищается коллекция rectable, чтобы подготовить их к вводу новых данных.
76. Начинается блок try для обработки возможных исключений.
77. Если флаг flag равен true, вызывается метод InputBin класса fio, который, вероятно, отвечает за ввод данных из бинарного файла в tModel и rectable.
78. Если флаг flag равен false, вызывается тот же метод, но без указания файла.
79. Если в блоке try произошла ошибка ввода-вывода или не найден класс (IOException или ClassNotFoundException), она будет поймана, и стек исключений будет записан в лог.
80. Эта строка определяет пакет, к которому принадлежит данный класс. В данном случае my.contacteditor указывает, что класс ErrRecIntegralVal находится в пакете contacteditor, который, в свою очередь, является частью пакета my.
81. Здесь мы объявляем класс ErrRecIntegralVal, который является публичным (public) и наследует от класса Exception. Это означает, что ErrRecIntegralVal является пользовательским исключением, которое может быть выброшено и обработано в других частях программы. Наследование от Exception позволяет использовать стандартные механизмы обработки исключений в Java, такие как try-catch.
82. Это конструктор класса ErrRecIntegralVal. Конструктор имеет один параметр — строку message, которая будет использоваться для передачи сообщения об ошибке. Публичный модификатор доступа позволяет создавать экземпляры этого класса из других классов.
83. В этой строке вызывается конструктор родительского класса Exception с помощью ключевого слова super. Это передает сообщение об ошибке в конструктор Exception, что позволяет исключению хранить и выводить это сообщение, когда оно будет обработано.
84. Эта строка указывает на то, что класс RecTableBin находится в пакете my.contacteditor.
85. Импортируется интерфейс Serializable из пакета java.io. Этот интерфейс позволяет объектам класса RecTableBin быть сериализованными, что означает, что их состояние может быть сохранено и восстановлено.
86. Объявление класса RecTableBin, который реализует интерфейс Serializable. Это означает, что экземпляры этого класса могут быть преобразованы в поток байтов и обратно.
87. Эта строка определяет статическую константу serialVersionUID, которая используется для контроля версии сериализованных объектов. Это важно для обеспечения совместимости при десериализации объектов, если класс изменился.
88. Здесь объявляются четыре приватных переменных экземпляра типа double. Эти переменные будут хранить значения, связанные с объектом RecTableBin: vp, np, st, result.
89. Это конструктор класса, который принимает четыре параметра: vp, np, st и result. Он также может выбрасывать исключение ErrRecIntegralVal, если переданные значения не соответствуют заданным условиям.
90. Это условие проверяет, находятся ли значения vp, np и st в заданных пределах (от 0.000001 до 1000000). Если хотя бы одно из значений выходит за эти пределы, то в теле условия должна быть обработка, но в данном коде она отсутствует. Это может быть ошибкой, так как исключение ErrRecIntegralVal не выбрасывается.
91. Если значения прошли проверку, они присваиваются соответствующим переменным экземпляра класса.
92. Это перегруженный конструктор, который принимает три параметра: vp, np и st, и также может выбрасывать исключение ErrRecIntegralVal.
93. Здесь result не инициализируется, что может привести к тому, что переменная result останется неинициализированной или будет иметь значение по умолчанию.
94. Метод getVP() возвращает значение переменной vp. Это геттер, который позволяет другим классам получать значение vp.
95. Метод getNP() возвращает значение переменной np. Аналогично, это геттер для np.
96. Метод getST() возвращает значение переменной st. Это геттер для st.
97. Метод getRES() возвращает значение переменной result. Это геттер для result.
98. Импортируются классы для работы с файлами, потоками ввода-вывода, а также для создания графических диалоговых окон и работы с таблицами.
99. Объявление класса
100. Определяем класс FileInputOutput, который будет содержать методы для ввода и вывода данных.
101. Метод для вывода данных в текстовый файл
102. Метод OutputTxt принимает LinkedList объектов RecIntegral и сохраняет их в текстовый файл data.txt.
103. Используется BufferedWriter для записи данных в файл. Файл открывается в режиме записи (перезапись).
104. Для каждого объекта RecIntegral формируется строка, которая содержит значения, разделенные символом /. Затем строка записывается в файл.
105. В случае ошибок при записи данных выводится сообщение с информацией об ошибке.
106. Перегруженный метод для вывода данных в указанный файл
107. Этот метод аналогичен предыдущему, но принимает File как параметр для указания, куда сохранять данные.
108. Методы для ввода данных из текстового файла
109. Метод для загрузки данных из файла data.txt в таблицу.
110. Используется BufferedReader для чтения данных из файла.
111. Каждая строка файла разбивается по символу /, а полученные данные преобразуются в массив Double, который затем добавляется в DefaultTableModel.
112. Перегруженный метод для ввода данных из указанного файла
113. Этот метод аналогичен предыдущему, но принимает File как параметр.
114. Метод для бинарного вывода
115. Метод для сохранения данных в бинарный файл. Сначала проверяется, есть ли данные для сохранения.
116. Если данных нет, выводится сообщение об ошибке.
117. Метод OutputBin принимает два параметра: rectable и file исключения FileNotFoundException и IOException.
118. Сначала проверяем, не пустой ли список rectable. Если он пуст или равен null, выводим сообщение об ошибке и выходим из метода.
119. Создаем FileOutputStream, который будет записывать данные в указанный файл.
120. Вложенный блок try-with resources создает ObjectOutputStream, который позволяет сериализовать объекты.
121. После успешной записи данных выводится сообщение о том, что данные сохранены.
122. Если произошла ошибка ввода-вывода, выводится сообщение об ошибке. Если возникла ошибка HeadlessException, выбрасывается RuntimeException.
123. Метод InputBin принимает tModel и rectable .
124. Создаем ObjectInputStream, который будет читать данные из файла databin.ser.
125. Метод readObject считывает объекты из потока и приводит их к типу ArrayList<RecTableBin>.
126. Для каждого объекта rec из rectable добавляем новую строку в tModel, используя значения, полученные через геттеры.
127. Если данные успешно загружены, выводится сообщение об успехе.
128. Если произошла ошибка при чтении файла или десериализации, выводится сообщение об ошибке.
129. Этот метод аналогичен предыдущему, но принимает дополнительный параметр file, который указывает файл для загрузки данных.
130. Создаем ObjectInputStream, который будет читать данные из указанного файла.
131. Заполняем tModel данными из rectable.
132. Обрабатываем возможные ошибки.

Результат выполнения программы

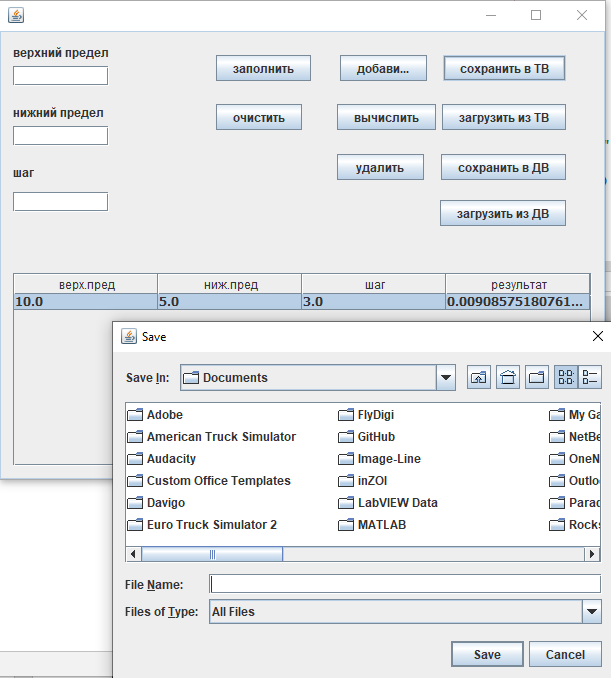


Рисунок 1 — Выбор файла для сохранения

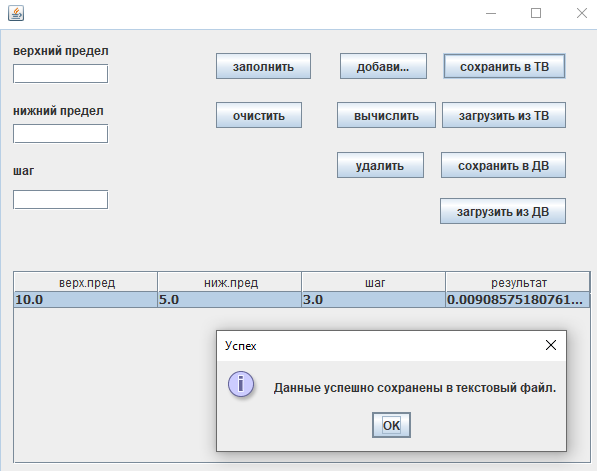


Рисунок 2 — Данные успешно сохранены

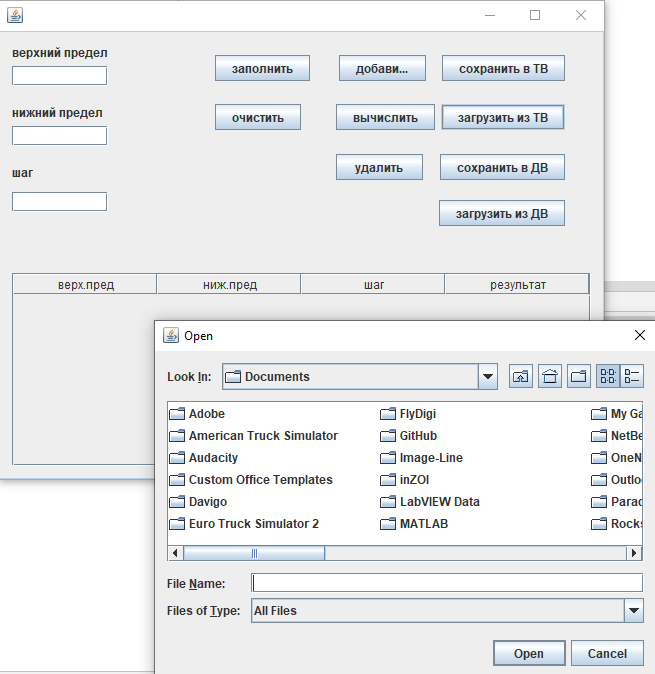


Рисунок 3 — Выбор файла из которого нужно загрузить данные

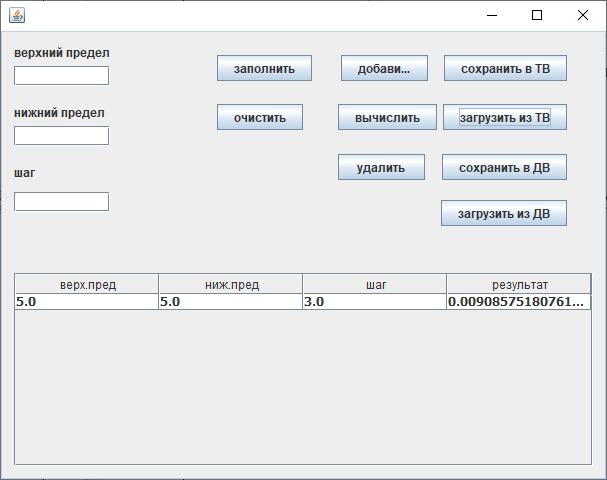


Рисунок 4 — Данные успешно загружены

Вывод

В данной лабораторной работе был изучена работа с файлами и механизмы сериализации данных.