Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧЕТ**

по лабораторной работе №6

по дисциплине «Программирование на языке Java»

на тему «Сетевое взаимодействие в Java»

Выполнили:

студенты группы 22ВВП1

Хоссейни Нежад С.А.С.М.

Захаров А. С.

Сергунов М. Р.

Приняли:

Юрова О.В.

Карамышева Н.С.

Пенза 2025

**Название**

Сетевое взаимодействие в Java

**Цель работы**

Научиться создавать клиент-серверные приложения c использованием стандартных классов Java.

**Лабораторное задание**

Модифицировать приложение из предыдущей лабораторной работы, реализовав клиент-серверную архитектуру, обеспечивающую распределенное вычисление определенного интеграла на нескольких вычислительных узлах (клиентах) при этом каждый узел использует несколько нитей, как в предыдущей работе. Сервер не занимается вычислениями, а лишь реализует взаимодействие с пользователем и агрегацию результатов вычислений от клиентов. Нечетные варианты используют протокол UDP, а четные TCP. Оформление лабораторной работы должно быть выполнено в соответствии с требованиями, приведенными в Приложении 2.

Вариант 6:



**Ход работы**

**Добавили новый класс ServerAp.java**

package Server;

import javax.swing.\*;

import javax.swing.UIManager.\*;

import java.io.\*;

import java.net.\*;

import java.nio.charset.StandardCharsets;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

import MyPackage.CommandData;

import MyPackage.RecIntegral;

public class ServerApp {

private static final int PORT = 12345;

private final List<ClientConnection> clientConnections = Collections.synchronizedList(new ArrayList<>());

public static void main(String[] args) {

setLookAndFeel();

setUtf8Output();

ServerApp app = new ServerApp();

Runtime.getRuntime().addShutdownHook(new Thread(() -> {

app.shutdown();

}));

app.startServer();

SwingUtilities.invokeLater(() -> {

Frame frame = new Frame(app);

frame.setTitle("Вычисление интегралов");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

});

}

/\*\*

\* Запуск серверного потока для приёма подключений клиентов.

\*/

private void startServer() {

new Thread(() -> {

try (ServerSocket serverSocket = new ServerSocket(PORT)) {

System.out.println("Сервер запущен на порту " + PORT);

while (true) {

Socket clientSocket = serverSocket.accept();

try {

ClientConnection connection = new ClientConnection(clientSocket);

clientConnections.add(connection);

System.out.println("Клиент подключен: " + clientSocket.getInetAddress());

} catch (IOException ex) {

System.err.println("Ошибка при создании соединения с клиентом.");

ex.printStackTrace();

}

}

} catch (IOException e) {

e.printStackTrace();

}

}).start();

}

/\*\*

\* Распределение вычислительной задачи по всем подключенным клиентам.

\*

\* @param lowLim начальное значение интегрирования

\* @param upLim конечное значение интегрирования

\* @param widthLim шаг вычисления

\* @return суммарный результат интегрирования

\*/

public double distributeTasks(double lowLim, double upLim, double widthLim) {

if (clientConnections.isEmpty()) {

System.out.println("Нет подключенных клиентов.");

return 0.0;

}

int numberOfClients = clientConnections.size();

double intervalWidth = (upLim - lowLim) / numberOfClients;

ExecutorService executor = Executors.newFixedThreadPool(numberOfClients);

List<Future<Double>> futures = new ArrayList<>();

for (int i = 0; i < numberOfClients; i++) {

double low = lowLim + i \* intervalWidth;

double high = low + intervalWidth;

ClientConnection connection = clientConnections.get(i);

futures.add(executor.submit(() -> sendTaskToClient(connection, low, high, widthLim)));

}

double totalResult = futures.stream().mapToDouble(future -> {

try {

return future.get();

} catch (Exception e) {

e.printStackTrace();

return 0.0;

}

}).sum();

executor.shutdown();

return totalResult;

}

/\*\*

\* Отправка задачи на вычисление конкретному клиенту и получение результата.

\*

\* @param connection соединение с клиентом

\* @param low нижняя граница интервала

\* @param high верхняя граница интервала

\* @param width шаг интегрирования

\* @return результат вычисления или 0.0 в случае ошибки

\*/

private double sendTaskToClient(ClientConnection connection, double low, double high, double width) {

try {

ObjectOutputStream oos = connection.getOos();

ObjectInputStream ois = connection.getOis();

CommandData task = new CommandData("calculate", new RecIntegral(low, high, width));

oos.writeObject(task);

oos.flush();

CommandData result = (CommandData) ois.readObject();

if ("result".equals(result.getCommandType())) {

return result.getResIntegral();

}

} catch (IOException | ClassNotFoundException e) {

System.err.println("Ошибка при отправке задачи клиенту.");

e.printStackTrace();

}

return 0.0;

}

/\*\*

\* Установка Nimbus Look and Feel.

\*/

private static void setLookAndFeel() {

try {

for (LookAndFeelInfo info : UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException | InstantiationException |

IllegalAccessException | UnsupportedLookAndFeelException ex) {

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

}

}

/\*\*

\* Настройка вывода в консоль с использованием UTF-8.

\*/

private static void setUtf8Output() {

System.setOut(new PrintStream(System.out, true, StandardCharsets.UTF\_8));

System.setErr(new PrintStream(System.err, true, StandardCharsets.UTF\_8));

}

/\*\*

\* Отключает сервер, уведомляя всех подключенных клиентов о завершении работы

\* и закрывая соответствующие соединения.

\*/

public void shutdown() {

System.out.println("Сервер завершает работу. Уведомление клиентов.");

for (ClientConnection connection : clientConnections) {

try {

ObjectOutputStream oos = connection.getOos();

oos.writeObject(new CommandData("exit", (RecIntegral) null));

oos.flush();

} catch (IOException ex) {

System.err.println("Не удалось отправить команду завершения клиенту.");

ex.printStackTrace();

} finally {

connection.close();

}

}

}

}

**Добавили новый класс ClientConnection.java**

package Server;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.Socket;

/\*\*

\* Класс для управления соединением с клиентом.

\* Обеспечивает обмен данными между сервером и клиентом через сокетное соединение.

\*/

public class ClientConnection {

private Socket socket;

private ObjectOutputStream oos;

private ObjectInputStream ois;

/\*\*

\* Создает новое соединение с клиентом.

\*

\* @param socket сокет для соединения с клиентом

\*/

public ClientConnection(Socket socket) throws IOException {

this.socket = socket;

this.oos = new ObjectOutputStream(socket.getOutputStream());

this.ois = new ObjectInputStream(socket.getInputStream());

}

/\*\*

\* Возвращает выходной поток для отправки данных клиенту.

\*

\* @return объект ObjectOutputStream для сериализованной отправки данных

\*/

public ObjectOutputStream getOos() {

return oos;

}

/\*\*

\* Возвращает входной поток для получения данных от клиента.

\*

\* @return объект ObjectInputStream для десериализации полученных данных

\*/

public ObjectInputStream getOis() {

return ois;

}

/\*\*

\* Возвращает сокет соединения с клиентом.

\*

\* @return объект Socket, представляющий соединение с клиентом

\*/

public Socket getSocket() {

return socket;

}

/\*\*

\* Корректное закрытие соединения и потоков.

\*/

public void close() {

try {

if (oos != null)

oos.close();

} catch (IOException e) {

System.err.println("Ошибка при закрытии ObjectOutputStream.");

e.printStackTrace();

}

try {

if (ois != null)

ois.close();

} catch (IOException e) {

System.err.println("Ошибка при закрытии ObjectInputStream.");

e.printStackTrace();

}

try {

if (socket != null && !socket.isClosed())

socket.close();

} catch (IOException e) {

System.err.println("Ошибка при закрытии Socket.");

e.printStackTrace();

}

}

}

**Разделили логику на сервер и клиента.**

**Создали новый класс CommandData**

package MyPackage;

import java.io.Serializable;

public class CommandData implements Serializable {

private static final long serialVersionUID = 4L;

private String commandType;

private RecIntegral integral;

public CommandData(String commandType, RecIntegral integral) {

this.commandType = commandType;

this.integral = integral;

}

public CommandData(String commandType, double resIntegral) {

this(commandType, new RecIntegral(0, 0, 0, resIntegral));

}

public String getCommandType() { return commandType; }

public double getResIntegral() { return integral.getResIntegral();}

public RecIntegral getRecIntegral() { return integral; }

@Override

public String toString() {

if (integral == null) {

return "CommandData{" +

"commandType='" + commandType + '\'' +

", integral=null" +

'}';

}

return "CommandData{" +

"commandType='" + commandType + '\'' +

", lowLim=" + integral.getLowLim() +

", upLim=" + integral.getUpLim() +

", widthLim=" + integral.getWidthLim() +

", resIntegral=" + integral.getResIntegral() +

'}';

}

}

**Создали свою библиотеку и поместили туда общие классы CommandData и RecIntegral**

**Листинг**

**Сервер**

**ClientConnection.java**

package Server;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.Socket;

/\*\*

\* Класс для управления соединением с клиентом.

\* Обеспечивает обмен данными между сервером и клиентом через сокетное соединение.

\*/

public class ClientConnection {

private Socket socket;

private ObjectOutputStream oos;

private ObjectInputStream ois;

/\*\*

\* Создает новое соединение с клиентом.

\*

\* @param socket сокет для соединения с клиентом

\*/

public ClientConnection(Socket socket) throws IOException {

this.socket = socket;

this.oos = new ObjectOutputStream(socket.getOutputStream());

this.ois = new ObjectInputStream(socket.getInputStream());

}

/\*\*

\* Возвращает выходной поток для отправки данных клиенту.

\*

\* @return объект ObjectOutputStream для сериализованной отправки данных

\*/

public ObjectOutputStream getOos() {

return oos;

}

/\*\*

\* Возвращает входной поток для получения данных от клиента.

\*

\* @return объект ObjectInputStream для десериализации полученных данных

\*/

public ObjectInputStream getOis() {

return ois;

}

/\*\*

\* Возвращает сокет соединения с клиентом.

\*

\* @return объект Socket, представляющий соединение с клиентом

\*/

public Socket getSocket() {

return socket;

}

/\*\*

\* Корректное закрытие соединения и потоков.

\*/

public void close() {

try {

if (oos != null)

oos.close();

} catch (IOException e) {

System.err.println("Ошибка при закрытии ObjectOutputStream.");

e.printStackTrace();

}

try {

if (ois != null)

ois.close();

} catch (IOException e) {

System.err.println("Ошибка при закрытии ObjectInputStream.");

e.printStackTrace();

}

try {

if (socket != null && !socket.isClosed())

socket.close();

} catch (IOException e) {

System.err.println("Ошибка при закрытии Socket.");

e.printStackTrace();

}

}

}

**DataException.java**

package Server;

/\*\*

\* Исключение, которое выбрасывается при некорректных данных.

\* Наследуется от класса Exception.

\*/

public class DataException extends Exception {

/\*\*

\* Создает новое исключение с указанным сообщением.

\*

\* @param message Сообщение об ошибке.

\*/

public DataException (String message){

super(message);

}

}

**FileManager.java**

package Server;

import javax.swing.JFrame;

import java.io.\*;

import java.util.ArrayList;

import java.util.LinkedList;

import javax.swing.\*;

import javax.swing.filechooser.FileNameExtensionFilter;

import MyPackage.RecIntegral;

public class FileManager {

private final JFrame parentFrame;

public FileManager(JFrame parentFrame) {

this.parentFrame = parentFrame;

}

//Text File Operations

public void saveToTextFile(File file, SavedState state) {

try (FileWriter writer = new FileWriter(file)) {

writeRecIntegralList(writer, state.getListRecIntegral());

writer.write("---\n");

writeRecIntegralList(writer, state.getArrListTableData());

} catch (IOException e) {

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

}

}

public SavedState loadFromTextFile(File file) {

LinkedList<RecIntegral> listRecIntegral = new LinkedList<>();

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

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

parseTextFile(reader, listRecIntegral, arrTableData);

return new SavedState(listRecIntegral, arrTableData);

} catch (IOException e) {

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

return null;

}

}

//Serializable File Operations

public void saveToBinaryFile(File file, SavedState data) {

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(file))) {

oos.writeObject(data);

} catch (IOException e) {

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

}

}

public SavedState loadFromBinaryFile(File file) {

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

return (SavedState) ois.readObject();

} catch (IOException | ClassNotFoundException e) {

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

return null;

}

}

//Externalizable File Operations

public void saveToBinaryExternFile(File file, SavedStateExtern data) {

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(file))) {

oos.writeObject(data);

} catch (IOException e) {

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

}

}

public SavedStateExtern loadFromBinaryExternFile(File file) {

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

return (SavedStateExtern) ois.readObject();

} catch (IOException | ClassNotFoundException e) {

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

return null;

}

}

//File Dialogs

private File getFilePath(int mode, String extension, String description) {

JFileChooser fileChooser = new JFileChooser();

FileNameExtensionFilter filter = new FileNameExtensionFilter(description, extension.substring(1));

fileChooser.setFileFilter(filter);

while (true) {

int option = (mode == JFileChooser.SAVE\_DIALOG) ? fileChooser.showSaveDialog(null) : fileChooser.showOpenDialog(null);

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

File file = fileChooser.getSelectedFile();

if (file == null) return null;

if (!file.getName().contains(".")) {

file = new File(file.getAbsolutePath() + extension);

}

else if (!file.getName().toLowerCase().endsWith(extension)) {

JOptionPane.showMessageDialog(parentFrame, "Файл должен иметь расширение " + extension + ". Пожалуйста, выберите другой файл.", "Ошибка", JOptionPane.ERROR\_MESSAGE );

continue;

}

if (mode == JFileChooser.SAVE\_DIALOG && file.exists()) {

int overwriteOption = JOptionPane.showConfirmDialog(parentFrame, "Файл уже существует. Перезаписать?", "Предупреждение", JOptionPane.YES\_NO\_OPTION);

if (overwriteOption != JOptionPane.YES\_OPTION) {

return null;

}

}

if (mode == JFileChooser.OPEN\_DIALOG && !file.exists()) {

JOptionPane.showMessageDialog(parentFrame, "Файл не существует.", "Ошибка", JOptionPane.ERROR\_MESSAGE);

return null;

}

return file;

} else {

return null;

}

}

}

public File getPathSerFileToSaved() {

return getFilePath(JFileChooser.SAVE\_DIALOG, ".ser", "Serialized Files (\*.ser)");

}

public File getPathSerFileToLoad() {

return getFilePath(JFileChooser.OPEN\_DIALOG, ".ser", "Serialized Files (\*.ser)");

}

public File getPathTXTFileToSaved() {

return getFilePath(JFileChooser.SAVE\_DIALOG, ".txt", "Text Files (\*.txt)");

}

public File getPathTXTFileToLoad() {

return getFilePath(JFileChooser.OPEN\_DIALOG, ".txt", "Text Files (\*.txt)");

}

public File getPathExternFileToSaved() {

return getFilePath(JFileChooser.SAVE\_DIALOG, ".dat", "Externalizable Files (\*.dat)");

}

public File getPathExternFileToLoad() {

return getFilePath(JFileChooser.OPEN\_DIALOG, ".dat", "Externalizable Files (\*.dat)");

}

//Private Helpers

private void writeRecIntegralList(FileWriter writer, Iterable<RecIntegral> list) throws IOException {

for (RecIntegral recIntegral : list) {

writer.write(String.format("%f;%f;%f;%f%n",

recIntegral.getLowLim(),

recIntegral.getUpLim(),

recIntegral.getWidthLim(),

recIntegral.getResIntegral()));

}

}

private void parseTextFile(BufferedReader reader, LinkedList<RecIntegral> listRecIntegral, ArrayList<RecIntegral> arrTableData) throws IOException {

String line;

boolean isListRecIntegral = true;

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

if (line.equals("---")) {

isListRecIntegral = false;

continue;

}

String[] parts = line.split(";");

if (parts.length == 4) {

RecIntegral recIntegral = createRecIntegralFromParts(parts);

if (isListRecIntegral) {

listRecIntegral.add(recIntegral);

} else {

arrTableData.add(recIntegral);

}

}

}

}

private RecIntegral createRecIntegralFromParts(String[] parts) {

String lowLimStr = parts[0].replace(',', '.');

String upLimStr = parts[1].replace(',', '.');

String widthLimStr = parts[2].replace(',', '.');

String resIntegralStr = parts[3].replace(',', '.');

double lowLim = Double.parseDouble(lowLimStr);

double upLim = Double.parseDouble(upLimStr);

double widthLim = Double.parseDouble(widthLimStr);

double resIntegral = Double.parseDouble(resIntegralStr);

return new RecIntegral(lowLim, upLim, widthLim, resIntegral);

}

}

**Frame.java**

package Server;

import java.io.File;

import java.util.ArrayList;

import javax.swing.table.DefaultTableModel;

import java.util.LinkedList;

import javax.swing.SwingWorker;

import MyPackage.RecIntegral;

public class Frame extends javax.swing.JFrame {

private final FileManager fileManager = new FileManager(this);

LinkedList<RecIntegral> listRecIntegral = new LinkedList<>();

private ServerApp serverApp;

/\*\*

\* Creates new form Frame

\*/

public Frame() {

initComponents();

}

public Frame(ServerApp serverApp) {

this.serverApp = serverApp;

initComponents();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

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

private void initComponents() {

label1 = new java.awt.Label();

label2 = new java.awt.Label();

label3 = new java.awt.Label();

jTextFieldSH = new javax.swing.JTextField();

jTextFieldNG = new javax.swing.JTextField();

jTextFieldVG = new javax.swing.JTextField();

jScrollPane1 = new javax.swing.JScrollPane();

jTable1 = new javax.swing.JTable();

jButtonDel = new javax.swing.JButton();

jButtonRes = new javax.swing.JButton();

jButtonAdd = new javax.swing.JButton();

jButtonClearTable = new javax.swing.JButton();

jButtonFillTable = new javax.swing.JButton();

bSaveObjectTextFormat = new javax.swing.JButton();

bLoadObjectTextFormat = new javax.swing.JButton();

bSaveObjectSerBinFormat = new javax.swing.JButton();

bLoadObjectSerBinFormat = new javax.swing.JButton();

textTXT = new java.awt.Label();

textSer = new java.awt.Label();

textExtern = new java.awt.Label();

bSaveObjecExternBinFormat = new javax.swing.JButton();

bLoadObjecExterntBinFormat = new javax.swing.JButton();

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

label1.setFont(new java.awt.Font("Dialog", 0, 14)); // NOI18N

label1.setText("Нижняя граница");

label2.setFont(new java.awt.Font("Dialog", 0, 14)); // NOI18N

label2.setText("Ширина шага");

label3.setFont(new java.awt.Font("Dialog", 0, 14)); // NOI18N

label3.setText("Верхняя граница");

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

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

jTextFieldSHActionPerformed(evt);

}

});

jTextFieldNG.setCursor(new java.awt.Cursor(java.awt.Cursor.TEXT\_CURSOR));

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

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

jTextFieldNGActionPerformed(evt);

}

});

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

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

jTextFieldVGActionPerformed(evt);

}

});

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

new Object [][] {},

new String [] {

"Нижняя граница", "Верхняя граница", "Шаг", "Результат"

}

));

jTable1.setRowSelectionAllowed(false);

jTable1.addAncestorListener(new javax.swing.event.AncestorListener() {

public void ancestorAdded(javax.swing.event.AncestorEvent evt) {

jTable1AncestorAdded(evt);

}

public void ancestorMoved(javax.swing.event.AncestorEvent evt) {

}

public void ancestorRemoved(javax.swing.event.AncestorEvent evt) {

}

});

jScrollPane1.setViewportView(jTable1);

if (jTable1.getColumnModel().getColumnCount() > 0) {

jTable1.getColumnModel().getColumn(0).setHeaderValue("Нижняя граница");

jTable1.getColumnModel().getColumn(1).setHeaderValue("Верхняя граница");

jTable1.getColumnModel().getColumn(2).setHeaderValue("Шаг");

jTable1.getColumnModel().getColumn(3).setHeaderValue("Результат");

}

jButtonDel.setText("Удалить");

jButtonDel.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jButtonDelMouseClicked(evt);

}

});

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

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

jButtonDelActionPerformed(evt);

}

});

jButtonRes.setText("Вычислить");

jButtonRes.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jButtonResMouseClicked(evt);

}

});

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

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

jButtonResActionPerformed(evt);

}

});

jButtonAdd.setText("Добавить");

jButtonAdd.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jButtonAddMouseClicked(evt);

}

});

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

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

jButtonAddActionPerformed(evt);

}

});

jButtonClearTable.setText("Очистить");

jButtonClearTable.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jButtonClearTableMouseClicked(evt);

}

});

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

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

jButtonClearTableActionPerformed(evt);

}

});

jButtonFillTable.setText("Заполнить");

jButtonFillTable.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jButtonFillTableMouseClicked(evt);

}

});

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

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

jButtonFillTableActionPerformed(evt);

}

});

bSaveObjectTextFormat.setText("Сохранить");

bSaveObjectTextFormat.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

bSaveObjectTextFormatMouseClicked(evt);

}

});

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

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

bSaveObjectTextFormatActionPerformed(evt);

}

});

bLoadObjectTextFormat.setText("Загрузить");

bLoadObjectTextFormat.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

bLoadObjectTextFormatMouseClicked(evt);

}

});

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

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

bLoadObjectTextFormatActionPerformed(evt);

}

});

bSaveObjectSerBinFormat.setText("Сохранить");

bSaveObjectSerBinFormat.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

bSaveObjectSerBinFormatMouseClicked(evt);

}

});

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

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

bSaveObjectSerBinFormatActionPerformed(evt);

}

});

bLoadObjectSerBinFormat.setText("Загрузить");

bLoadObjectSerBinFormat.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

bLoadObjectSerBinFormatMouseClicked(evt);

}

});

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

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

bLoadObjectSerBinFormatActionPerformed(evt);

}

});

textTXT.setFont(new java.awt.Font("Dialog", 0, 14)); // NOI18N

textTXT.setText("В текстовом виде");

textSer.setFont(new java.awt.Font("Dialog", 0, 14)); // NOI18N

textSer.setText("В двоичном виде с сериализацией");

textExtern.setFont(new java.awt.Font("Dialog", 0, 14)); // NOI18N

textExtern.setText("В двоичном виде с экстернализацией");

bSaveObjecExternBinFormat.setText("Сохранить");

bSaveObjecExternBinFormat.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

bSaveObjecExternBinFormatMouseClicked(evt);

}

});

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

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

bSaveObjecExternBinFormatActionPerformed(evt);

}

});

bLoadObjecExterntBinFormat.setText("Загрузить");

bLoadObjecExterntBinFormat.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

bLoadObjecExterntBinFormatMouseClicked(evt);

}

});

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

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

bLoadObjecExterntBinFormatActionPerformed(evt);

}

});

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

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

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

.addGroup(layout.createSequentialGroup()

.addGap(6, 6, 6)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 835, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addGroup(layout.createSequentialGroup()

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

.addGroup(layout.createSequentialGroup()

.addComponent(bSaveObjectTextFormat, javax.swing.GroupLayout.PREFERRED\_SIZE, 132, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(bLoadObjectTextFormat, javax.swing.GroupLayout.PREFERRED\_SIZE, 132, javax.swing.GroupLayout.PREFERRED\_SIZE))

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

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

.addGroup(layout.createSequentialGroup()

.addGap(11, 11, 11)

.addComponent(bSaveObjectSerBinFormat, javax.swing.GroupLayout.PREFERRED\_SIZE, 132, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(bLoadObjectSerBinFormat, javax.swing.GroupLayout.PREFERRED\_SIZE, 132, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createSequentialGroup()

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

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

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

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

.addGroup(layout.createSequentialGroup()

.addComponent(bSaveObjecExternBinFormat, javax.swing.GroupLayout.PREFERRED\_SIZE, 132, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(bLoadObjecExterntBinFormat, javax.swing.GroupLayout.PREFERRED\_SIZE, 132, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addComponent(textExtern, javax.swing.GroupLayout.PREFERRED\_SIZE, 270, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addGroup(layout.createSequentialGroup()

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

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

.addComponent(label2, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(label1, javax.swing.GroupLayout.DEFAULT\_SIZE, 117, Short.MAX\_VALUE))

.addComponent(label3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

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

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

.addComponent(jTextFieldVG, javax.swing.GroupLayout.PREFERRED\_SIZE, 192, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jTextFieldNG, javax.swing.GroupLayout.DEFAULT\_SIZE, 192, Short.MAX\_VALUE)

.addComponent(jTextFieldSH)))

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

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

.addComponent(jButtonRes, javax.swing.GroupLayout.PREFERRED\_SIZE, 144, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jButtonAdd, javax.swing.GroupLayout.PREFERRED\_SIZE, 144, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButtonClearTable, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, 144, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButtonDel, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, 144, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButtonFillTable, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, 144, javax.swing.GroupLayout.PREFERRED\_SIZE))))))

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

);

layout.setVerticalGroup(

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

.addGroup(layout.createSequentialGroup()

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

.addGroup(layout.createSequentialGroup()

.addGap(60, 60, 60)

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

.addComponent(jTextFieldNG, javax.swing.GroupLayout.PREFERRED\_SIZE, 28, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(label1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

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

.addComponent(jTextFieldVG, javax.swing.GroupLayout.PREFERRED\_SIZE, 28, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(label3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(23, 23, 23)

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

.addComponent(jTextFieldSH, javax.swing.GroupLayout.PREFERRED\_SIZE, 28, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(label2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(64, 64, 64))

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

.addContainerGap()

.addComponent(jButtonAdd, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jButtonDel, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jButtonRes, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(16, 16, 16)

.addComponent(jButtonClearTable, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jButtonFillTable, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)))

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

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

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 130, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

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

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

.addComponent(textExtern, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

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

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

.addComponent(bSaveObjectTextFormat, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(bLoadObjectTextFormat, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(bSaveObjectSerBinFormat, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(bLoadObjectSerBinFormat, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(bSaveObjecExternBinFormat, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(bLoadObjecExterntBinFormat, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addContainerGap(14, Short.MAX\_VALUE))

);

textExtern.getAccessibleContext().setAccessibleDescription("");

pack();

}// </editor-fold>

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

// TODO add your handling code here:

}

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

// TODO add your handling code here:

}

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

// TODO add your handling code here:

}

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

// TODO add your handling code here:

}

private void jTable1AncestorAdded(javax.swing.event.AncestorEvent evt) {

// TODO add your handling code here:

}

private void jButtonDelMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

if(jTable1.getRowCount() != 0)

{

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

int selectRow = jTable1.getSelectedRow();

if(selectRow == -1) return;

model.removeRow(selectRow);

}

}

private void jButtonResMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

if (jTable1.getRowCount() == 0) return;

int selectRow = jTable1.getSelectedRow();

if (selectRow == -1) return;

try {

RecIntegral data = InputValidator.validateAndParse(

jTable1.getValueAt(selectRow, 0).toString(),

jTable1.getValueAt(selectRow, 1).toString(),

jTable1.getValueAt(selectRow, 2).toString()

);

new SwingWorker<Double, Void>() {

@Override

protected Double doInBackground() {

return serverApp.distributeTasks(data.getLowLim(), data.getUpLim(), data.getWidthLim());

}

@Override

protected void done() {

try {

double result = get();

jTable1.setValueAt(result, selectRow, 3);

} catch (Exception e) {

e.printStackTrace();

}

}

}.execute();

} catch (DataException ex) {

javax.swing.JOptionPane.showMessageDialog(

this,

ex.getMessage(),

"Ошибка",

javax.swing.JOptionPane.ERROR\_MESSAGE);

}

}

private void jButtonAddMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

try{

RecIntegral data = InputValidator.validateAndParse(

jTextFieldNG.getText(),

jTextFieldVG.getText(),

jTextFieldSH.getText()

);

((DefaultTableModel) jTable1.getModel()).addRow(

new Object[]{data.getLowLim(), data.getUpLim(), data.getWidthLim()}

);

}

catch(DataException ex){

javax.swing.JOptionPane.showMessageDialog(this,

ex.getMessage(),

"Ошибка",

javax.swing.JOptionPane.ERROR\_MESSAGE);

}

}

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

// TODO add your handling code here:

}

private void jButtonClearTableMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

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

int count = model.getRowCount();

for (int i = 0; i < count; i++) {

double lowLim = (double) model.getValueAt(i, 0);

double upLim = (double) model.getValueAt(i, 1);

double widthLim = (double) model.getValueAt(i, 2);

double resIntegral;

Object value = model.getValueAt(i, 3);

if (value instanceof Number) {

resIntegral = ((Number) value).doubleValue();

} else {

resIntegral = Double.MAX\_VALUE;

}

RecIntegral dataIntegral = new RecIntegral(lowLim, upLim, widthLim, resIntegral);

listRecIntegral.add(dataIntegral);

}

model.setRowCount(0);

}

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

// TODO add your handling code here:

}

private void jButtonFillTableMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

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

for(RecIntegral dataRow : listRecIntegral) {

double lowLim = dataRow.getLowLim();

double upLim = dataRow.getUpLim();

double widthLim = dataRow.getWidthLim();

if (dataRow.getResIntegral() == Double.MAX\_VALUE) {

model.addRow(new Object[]{lowLim, upLim, widthLim});

}

else {

double resIntegral = dataRow.getResIntegral();

model.addRow(new Object[]{lowLim, upLim, widthLim, resIntegral});

}

}

listRecIntegral.clear();

}

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

// TODO add your handling code here:

}

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

// TODO add your handling code here:

}

private void bSaveObjectTextFormatMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

File file = fileManager.getPathTXTFileToSaved();

if (file == null) return;

ArrayList<RecIntegral> arrTableData = getDataArrListFromTable();

SavedState state = new SavedState(listRecIntegral, arrTableData);

fileManager.saveToTextFile(file, state);

}

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

// TODO add your handling code here:

}

private void bLoadObjectTextFormatMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

File file = fileManager.getPathTXTFileToLoad();

if (file == null) return;

SavedState state = fileManager.loadFromTextFile(file);

if (state != null) {

listRecIntegral = state.getListRecIntegral();

setDataToTable(state.getArrListTableData());

}

}

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

// TODO add your handling code here:

}

private ArrayList<RecIntegral> getDataArrListFromTable() {

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

int count = model.getRowCount();

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

for (int i = 0; i < count; i++) {

double lowLim = (double) model.getValueAt(i, 0);

double upLim = (double) model.getValueAt(i, 1);

double widthLim = (double) model.getValueAt(i, 2);

double resIntegral;

Object value = model.getValueAt(i, 3);

if (value instanceof Number) {

resIntegral = ((Number) value).doubleValue();

} else {

resIntegral = Double.MAX\_VALUE;

}

data.add(new RecIntegral(lowLim, upLim, widthLim, resIntegral));

}

return data;

}

private void setDataToTable(ArrayList<RecIntegral> data) {

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

model.setRowCount(0);

for (RecIntegral row : data) {

if (row.getResIntegral() == Double.MAX\_VALUE) {

model.addRow(new Object[]{row.getLowLim(), row.getUpLim(), row.getWidthLim()});

} else {

model.addRow(new Object[]{row.getLowLim(), row.getUpLim(), row.getWidthLim(), row.getResIntegral()});

}

}

}

private void bSaveObjectSerBinFormatMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

File file = fileManager.getPathSerFileToSaved();

if (file == null) return;

ArrayList<RecIntegral> arrTableData = getDataArrListFromTable();

SavedState state = new SavedState(listRecIntegral, arrTableData);

fileManager.saveToBinaryFile(file, state);

}

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

// TODO add your handling code here:

}

private void bLoadObjectSerBinFormatMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

File file = fileManager.getPathSerFileToLoad();

if (file == null) return;

SavedState state = fileManager.loadFromBinaryFile(file);

if (state != null) {

listRecIntegral = state.getListRecIntegral();

setDataToTable(state.getArrListTableData());

}

}

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

// TODO add your handling code here:

}

private void bSaveObjecExternBinFormatMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

File file = fileManager.getPathExternFileToSaved();

if (file == null) return;

ArrayList<RecIntegral> arrTableData = getDataArrListFromTable();

SavedStateExtern state = new SavedStateExtern(listRecIntegral, arrTableData);

fileManager.saveToBinaryExternFile(file, state);

}

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

// TODO add your handling code here:

}

private void bLoadObjecExterntBinFormatMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

File file = fileManager.getPathExternFileToLoad();

if (file == null) return;

SavedStateExtern state = fileManager.loadFromBinaryExternFile(file);

if (state != null) {

listRecIntegral = state.getListRecIntegral();

setDataToTable(state.getArrListTableData());

}

}

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

// TODO add your handling code here:

}

// Variables declaration - do not modify

private javax.swing.JButton bLoadObjecExterntBinFormat;

private javax.swing.JButton bLoadObjectSerBinFormat;

private javax.swing.JButton bLoadObjectTextFormat;

private javax.swing.JButton bSaveObjecExternBinFormat;

private javax.swing.JButton bSaveObjectSerBinFormat;

private javax.swing.JButton bSaveObjectTextFormat;

private javax.swing.JButton jButtonAdd;

private javax.swing.JButton jButtonClearTable;

private javax.swing.JButton jButtonDel;

private javax.swing.JButton jButtonFillTable;

private javax.swing.JButton jButtonRes;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JTable jTable1;

private javax.swing.JTextField jTextFieldNG;

private javax.swing.JTextField jTextFieldSH;

private javax.swing.JTextField jTextFieldVG;

private java.awt.Label label1;

private java.awt.Label label2;

private java.awt.Label label3;

private java.awt.Label textExtern;

private java.awt.Label textSer;

private java.awt.Label textTXT;

// End of variables declaration

}

**InputValidator.java**

package Server;

import MyPackage.RecIntegral;

public class InputValidator {

public static RecIntegral validateAndParse(String lower, String upper, String step) throws DataException {

double lowerVal = parseValue(lower);

double upperVal = parseValue(upper);

double stepVal = parseValue(step);

validateRange(lowerVal, upperVal, stepVal);

return new RecIntegral(lowerVal, upperVal, stepVal);

}

private static double parseValue(String input) throws DataException {

try {

double value = Double.parseDouble(input);

if (value == 0 || (value >= 0.01 && value <= 1)) {

throw new DataException("Значение должно быть от 0.000001 до 1000000");

}

return value;

} catch (NumberFormatException e) {

throw new DataException("Некорректный числовой формат");

}

}

private static void validateRange(double lower, double upper, double step) throws DataException {

if (Math.abs(upper - lower) < step) {

throw new DataException("Интервал должен быть не меньше шага");

}

}

}

**SavedState.java**

package Server;

import java.io.Serializable;

import java.util.ArrayList;

import java.util.LinkedList;

import MyPackage.RecIntegral;

/\*\*

\* Класс для хранения состояния приложения, включающего два списка объектов RecIntegral.

\* Реализует интерфейс Serializable для поддержки сериализации.

\*/

public class SavedState implements Serializable {

private static final long serialVersionUID = 2L;

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

private ArrayList<RecIntegral> arrTableData = new ArrayList<>();

/\*\*

\* Конструктор для создания объекта SavedState.

\*

\* @param listRecIntegral Список RecIntegral для хранения.

\* @param arrTableData Список данных таблицы для хранения.

\*/

public SavedState(LinkedList<RecIntegral> listRecIntegral, ArrayList<RecIntegral> arrTableData) {

this.listRecIntegral = new LinkedList<>(listRecIntegral);

this.arrTableData = new ArrayList<>(arrTableData);

}

/\*\*

\* Возвращает копию списка RecIntegral вне таблицы.

\*

\* @return Копия списка RecIntegral вне таблицы.

\*/

public LinkedList<RecIntegral> getListRecIntegral() {

return new LinkedList<>(listRecIntegral);

}

/\*\*

\* Возвращает копию списка данных таблицы.

\*

\* @return Копия списка данных таблицы.

\*/

public ArrayList<RecIntegral> getArrListTableData() {

return new ArrayList<>(arrTableData);

}

@Override

public String toString() {

return "SavedState{" +

"listRecIntegral=" + listRecIntegral +

", arrTableData=" + arrTableData +

'}';

}

}

**SavedStateExtern.java**

package Server;

import java.io.Externalizable;

import java.io.IOException;

import java.io.ObjectInput;

import java.io.ObjectOutput;

import java.util.ArrayList;

import java.util.LinkedList;

import MyPackage.RecIntegral;

/\*\*

\* Класс для хранения состояния приложения с использованием интерфейса Externalizable.

\* Сохраняет два списка объектов RecIntegral: основной список и данные для таблицы.

\*/

public class SavedStateExtern implements Externalizable {

private static final long serialVersionUID = 3L;

private LinkedList<RecIntegral> listRecIntegral;

private ArrayList<RecIntegral> arrTableData;

/\*\*

\* Конструктор по умолчанию.

\* Инициализирует пустые списки.

\*/

public SavedStateExtern() {

this.listRecIntegral = new LinkedList<>();

this.arrTableData = new ArrayList<>();

}

/\*\*

\* Параметризованный конструктор.

\*

\* @param listRecIntegral список объектов RecIntegral для основного хранения

\* @param arrTableData список объектов RecIntegral для табличного представления

\*/

public SavedStateExtern(LinkedList<RecIntegral> listRecIntegral, ArrayList<RecIntegral> arrTableData) {

this.listRecIntegral = new LinkedList<>(listRecIntegral);

this.arrTableData = new ArrayList<>(arrTableData);

}

/\*\*

\* Сериализует объект. Записывает списки в поток вывода.

\*

\* @param out поток вывода для записи объекта

\* @throws IOException если произошла ошибка ввода-вывода

\*/

@Override

public void writeExternal(ObjectOutput out) throws IOException {

out.writeObject(listRecIntegral);

out.writeObject(arrTableData);

}

/\*\*

\* Десериализует объект. Восстанавливает списки из потока ввода.

\*

\* @param in поток ввода для чтения объекта

\* @throws IOException если произошла ошибка ввода-вывода

\* @throws ClassNotFoundException если класс объекта не найден

\*/

@Override

@SuppressWarnings("unchecked")

public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException {

listRecIntegral = (LinkedList<RecIntegral>) in.readObject();

arrTableData = (ArrayList<RecIntegral>) in.readObject();

}

/\*\*

\* Возвращает копию основного списка объектов RecIntegral.

\*

\* @return новый LinkedList с объектами RecIntegral

\*/

public LinkedList<RecIntegral> getListRecIntegral() {

return new LinkedList<>(listRecIntegral);

}

/\*\*

\* Возвращает копию списка данных для табличного представления.

\*

\* @return новый ArrayList с объектами RecIntegral

\*/

public ArrayList<RecIntegral> getArrListTableData() {

return new ArrayList<>(arrTableData);

}

@Override

public String toString() {

return "SavedStateExtern{" +

"listRecIntegral=" + listRecIntegral +

", arrTableData=" + arrTableData +

'}';

}

}

**ServerApp.java**

package Server;

import javax.swing.\*;

import javax.swing.UIManager.\*;

import java.io.\*;

import java.net.\*;

import java.nio.charset.StandardCharsets;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

import MyPackage.CommandData;

import MyPackage.RecIntegral;

public class ServerApp {

private static final int PORT = 12345;

private final List<ClientConnection> clientConnections = Collections.synchronizedList(new ArrayList<>());

public static void main(String[] args) {

setLookAndFeel();

setUtf8Output();

ServerApp app = new ServerApp();

Runtime.getRuntime().addShutdownHook(new Thread(() -> {

app.shutdown();

}));

app.startServer();

SwingUtilities.invokeLater(() -> {

Frame frame = new Frame(app);

frame.setTitle("Вычисление интегралов");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

});

}

/\*\*

\* Запуск серверного потока для приёма подключений клиентов.

\*/

private void startServer() {

new Thread(() -> {

try (ServerSocket serverSocket = new ServerSocket(PORT)) {

System.out.println("Сервер запущен на порту " + PORT);

while (true) {

Socket clientSocket = serverSocket.accept();

try {

ClientConnection connection = new ClientConnection(clientSocket);

clientConnections.add(connection);

System.out.println("Клиент подключен: " + clientSocket.getInetAddress());

} catch (IOException ex) {

System.err.println("Ошибка при создании соединения с клиентом.");

ex.printStackTrace();

}

}

} catch (IOException e) {

e.printStackTrace();

}

}).start();

}

/\*\*

\* Распределение вычислительной задачи по всем подключенным клиентам.

\*

\* @param lowLim начальное значение интегрирования

\* @param upLim конечное значение интегрирования

\* @param widthLim шаг вычисления

\* @return суммарный результат интегрирования

\*/

public double distributeTasks(double lowLim, double upLim, double widthLim) {

if (clientConnections.isEmpty()) {

System.out.println("Нет подключенных клиентов.");

return 0.0;

}

int numberOfClients = clientConnections.size();

double intervalWidth = (upLim - lowLim) / numberOfClients;

ExecutorService executor = Executors.newFixedThreadPool(numberOfClients);

List<Future<Double>> futures = new ArrayList<>();

for (int i = 0; i < numberOfClients; i++) {

double low = lowLim + i \* intervalWidth;

double high = low + intervalWidth;

ClientConnection connection = clientConnections.get(i);

futures.add(executor.submit(() -> sendTaskToClient(connection, low, high, widthLim)));

}

double totalResult = futures.stream().mapToDouble(future -> {

try {

return future.get();

} catch (Exception e) {

e.printStackTrace();

return 0.0;

}

}).sum();

executor.shutdown();

return totalResult;

}

/\*\*

\* Отправка задачи на вычисление конкретному клиенту и получение результата.

\*

\* @param connection соединение с клиентом

\* @param low нижняя граница интервала

\* @param high верхняя граница интервала

\* @param width шаг интегрирования

\* @return результат вычисления или 0.0 в случае ошибки

\*/

private double sendTaskToClient(ClientConnection connection, double low, double high, double width) {

try {

ObjectOutputStream oos = connection.getOos();

ObjectInputStream ois = connection.getOis();

CommandData task = new CommandData("calculate", new RecIntegral(low, high, width));

oos.writeObject(task);

oos.flush();

CommandData result = (CommandData) ois.readObject();

if ("result".equals(result.getCommandType())) {

return result.getResIntegral();

}

} catch (IOException | ClassNotFoundException e) {

System.err.println("Ошибка при отправке задачи клиенту.");

e.printStackTrace();

}

return 0.0;

}

/\*\*

\* Установка Nimbus Look and Feel.

\*/

private static void setLookAndFeel() {

try {

for (LookAndFeelInfo info : UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException | InstantiationException |

IllegalAccessException | UnsupportedLookAndFeelException ex) {

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

}

}

/\*\*

\* Настройка вывода в консоль с использованием UTF-8.

\*/

private static void setUtf8Output() {

System.setOut(new PrintStream(System.out, true, StandardCharsets.UTF\_8));

System.setErr(new PrintStream(System.err, true, StandardCharsets.UTF\_8));

}

/\*\*

\* Отключает сервер, уведомляя всех подключенных клиентов о завершении работы

\* и закрывая соответствующие соединения.

\*/

public void shutdown() {

System.out.println("Сервер завершает работу. Уведомление клиентов.");

for (ClientConnection connection : clientConnections) {

try {

ObjectOutputStream oos = connection.getOos();

oos.writeObject(new CommandData("exit", (RecIntegral) null));

oos.flush();

} catch (IOException ex) {

System.err.println("Не удалось отправить команду завершения клиенту.");

ex.printStackTrace();

} finally {

connection.close();

}

}

}

}

**Клиент**

**DistributedIntegralCalculator.java**

package Client;

import java.util.concurrent.\*;

import MyPackage.RecIntegral;

public class DistributedIntegralCalculator {

private final RecIntegral integral;

private final int numberOfThreads;

public DistributedIntegralCalculator(RecIntegral integral, int numberOfThreads) {

this.integral = integral;

this.numberOfThreads = numberOfThreads;

}

public double calculate() {

double lowLim = integral.getLowLim();

double upLim = integral.getUpLim();

double intervalLength = upLim - lowLim;

double subIntervalLength = intervalLength / numberOfThreads;

double totalIntegral = 0;

IntegralTask[] tasks = new IntegralTask[numberOfThreads];

ExecutorService executor = Executors.newFixedThreadPool(numberOfThreads);

for (int i = 0; i < numberOfThreads; i++) {

double subLower = lowLim + i \* subIntervalLength;

double subUpper = (i == numberOfThreads - 1) ? upLim : subLower + subIntervalLength;

tasks[i] = new IntegralTask(new RecIntegral(subLower, subUpper, integral.getWidthLim()));

executor.execute(tasks[i]);

}

executor.shutdown();

try {

executor.awaitTermination(Long.MAX\_VALUE, TimeUnit.NANOSECONDS);

} catch (InterruptedException e) {

e.printStackTrace();

}

for (IntegralTask task : tasks) {

totalIntegral += task.getResult();

}

return totalIntegral;

}

}

**IntegralTask.java**

package Client;

import MyPackage.RecIntegral;

public class IntegralTask implements Runnable {

private RecIntegral integral;

private double result;

public IntegralTask(RecIntegral integral) {

this.integral = integral;

}

@Override

public void run() {

result = calculateIntegral(integral);

}

public double getResult() {

return result;

}

private double calculateIntegral(RecIntegral integral) {

double lowLim = integral.getLowLim();

double upLim = integral.getUpLim();

double widthLim = integral.getWidthLim();

boolean isReversed = lowLim > integral.getUpLim();

if (isReversed) {

double tempBorder = lowLim;

lowLim = upLim;

upLim = tempBorder;

}

double currentLowerBorder = lowLim;

long count = (long)((upLim - lowLim) / integral.getWidthLim());

double sum = 0;

for (long j = 0; j < count; j++) {

sum += ((widthLim / 2) \* (Math.sqrt(currentLowerBorder) + Math.sqrt(currentLowerBorder + widthLim)));

currentLowerBorder += widthLim;

}

if((upLim - lowLim) / widthLim > count) {

currentLowerBorder -= widthLim;

double lastStepWeigth = upLim - (currentLowerBorder);

sum += ((lastStepWeigth / 2) \* (Math.sqrt(currentLowerBorder) + Math.sqrt(upLim)));

}

return isReversed ? -sum : sum;

}

}

**Main.java**

package Client;

import java.io.\*;

import java.net.Socket;

import java.nio.charset.StandardCharsets;

import MyPackage.CommandData;

import java.net.SocketException;

public class Main {

public static void main(String[] args) {

System.setOut(new PrintStream(System.out, true, StandardCharsets.UTF\_8));

System.setErr(new PrintStream(System.err, true, StandardCharsets.UTF\_8));

final int PORT = 12345;

final String SERVER\_ADDRES = "localhost";

final int COUNT\_THREAD = 1;

try (Socket socket = new Socket(SERVER\_ADDRES, PORT);

ObjectOutputStream oos = new ObjectOutputStream(socket.getOutputStream());

ObjectInputStream ois = new ObjectInputStream(socket.getInputStream())) {

System.out.println("Подключено к серверу.");

while (true) {

Object obj = ois.readObject();

if (!(obj instanceof CommandData)) {

System.out.println("Ошибка: получен неизвестный объект");

break;

}

CommandData task = (CommandData) obj;

System.out.println("Задача получена: " + task);

if ("exit".equals(task.getCommandType())) {

System.out.println("Получена команда выхода. Завершение работы...");

break;

}

if ("calculate".equals(task.getCommandType())) {

DistributedIntegralCalculator calculator = new DistributedIntegralCalculator(task.getRecIntegral(), COUNT\_THREAD);

double result = calculator.calculate();

System.out.println("Результат вычислен: " + result);

oos.writeObject(new CommandData("result", result));

oos.flush();

}

}

} catch (SocketException e) {

System.out.println("Соединение с сервером разорвано.");

} catch (EOFException e) {

System.out.println("Сервер завершил соединение.");

} catch (IOException | ClassNotFoundException e) {

e.printStackTrace();

System.out.println("Ошибка соединения.");

}

}

}

**Библиотека MyLibrary**

**CommandData.java**

package MyPackage;

import java.io.Serializable;

public class CommandData implements Serializable {

private static final long serialVersionUID = 4L;

private String commandType;

private RecIntegral integral;

public CommandData(String commandType, RecIntegral integral) {

this.commandType = commandType;

this.integral = integral;

}

public CommandData(String commandType, double resIntegral) {

this(commandType, new RecIntegral(0, 0, 0, resIntegral));

}

public String getCommandType() { return commandType; }

public double getResIntegral() { return integral.getResIntegral();}

public RecIntegral getRecIntegral() { return integral; }

@Override

public String toString() {

if (integral == null) {

return "CommandData{" +

"commandType='" + commandType + '\'' +

", integral=null" +

'}';

}

return "CommandData{" +

"commandType='" + commandType + '\'' +

", lowLim=" + integral.getLowLim() +

", upLim=" + integral.getUpLim() +

", widthLim=" + integral.getWidthLim() +

", resIntegral=" + integral.getResIntegral() +

'}';

}

}

**RecIntegral.java**

package MyPackage;

import java.io.Serializable;

/\*\*

\* Класс для хранения данных о вычислении интеграла.

\* Реализует интерфейс Serializable для поддержки сериализации.

\*/

public class RecIntegral implements Serializable {

private static final long serialVersionUID = 1L;

private double widthLim;

private double lowLim;

private double upLim;

private double resIntegral;

/\*\*

\* Конструктор с необязательным resIntegral (по умолчанию Double.MAX\_VALUE)

\*/

public RecIntegral(double lowLim, double upLim, double widthLim) {

this(lowLim, upLim, widthLim, Double.MAX\_VALUE);

}

/\*\*

\* Конструктор для создания объекта RecIntegral.

\*

\* @param lowLim Нижняя граница.

\* @param upLim Верхняя граница.

\* @param widthLim Ширина шага.

\* @param resIntegral Результат вычисления интеграла.

\*/

public RecIntegral(double lowLim, double upLim, double widthLim, double resIntegral) {

this.widthLim = widthLim;

this.lowLim = lowLim;

this.upLim = upLim;

this.resIntegral = resIntegral;

}

/\*\*

\* Возвращает нижнюю границу.

\*

\* @return Нижняя граница.

\*/

public double getLowLim() {

return lowLim;

}

/\*\*

\* Устанавливает нижнюю границу.

\*

\* @param lowLim Нижняя граница.

\*/

public void setLowLim(double lowLim) {

this.lowLim = lowLim;

}

/\*\*

\* Возвращает верхнюю границу.

\*

\* @return Верхняя граница.

\*/

public double getUpLim() {

return upLim;

}

/\*\*

\* Устанавливает верхнюю границу.

\*

\* @param upLim Верхняя граница.

\*/

public void setUpLim(double upLim) {

this.upLim = upLim;

}

/\*\*

\* Возвращает ширину шага.

\*

\* @return Ширина шага.

\*/

public double getWidthLim() {

return widthLim;

}

/\*\*

\* Устанавливает ширину шага.

\*

\* @param widthLim Ширина шага.

\*/

public void setWidthLim(double widthLim) {

this.widthLim = widthLim;

}

/\*\*

\* Возвращает результат вычисления интеграла.

\*

\* @return Результат вычисления интеграла.

\*/

public double getResIntegral() {

return resIntegral;

}

/\*\*

\* Устанавливает результат вычисления интеграла.

\*

\* @param resIntegral Результат вычисления интеграла.

\*/

public void setResIntegral(double resIntegral) {

this.resIntegral = resIntegral;

}

@Override

public String toString() {

return "RecIntegral{" +

"lowLim=" + lowLim +

", upLim=" + upLim +

", widthLim=" + widthLim +

", resIntegral=" + resIntegral +

'}';

}

}

**Вывод:** В ходе выполнения лабораторной работы научились создавать клиент-серверные приложения c использованием стандартных классов Java.