Пензенский государственный университет

Кафедра «Вычислительная техника»  
  
  
  
  
  
 **ОТЧЁТ**

по лабораторной работе №6

по курсу «Программирование на языке Java»

на тему «Сетевое взаимодействие в Java»

Вариант 1

Выполнили:

cтуденты группы 21ВВП2

Давкин М.А.

Борисова А.В.

Михальцова П.В.

Приняли:

Юрова О.В.

Карамышева Н.С.

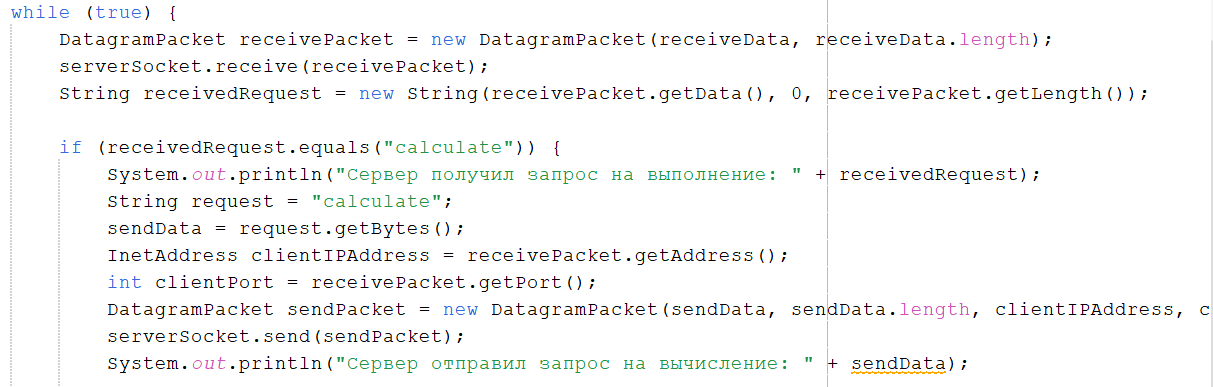
Пенза 2024

**Цель работы:** научиться создавать клиент-серверные приложения c использованием стандартных классов Java.

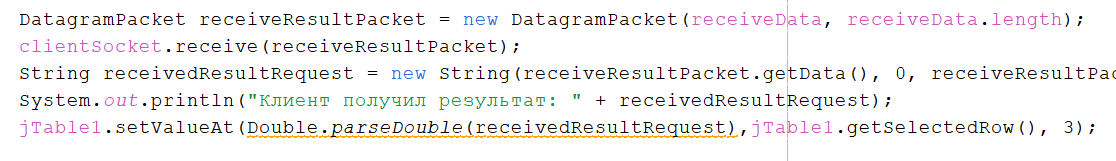
**Лабораторное задание:**

Модифицировать приложение из предыдущей лабораторной работы 5, 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. Использовать протокол UDP.

**Описание метода решения задачи:**

Для работы с датаграммами в программе используются объекты классов DatagramSocket и DatagramPacket. При запуске сервера (Server.java) создается его сокет. Сервер бесконечно ожидает запросы от клиента на выполнение. Если клиент посылает запрос «calculate», сервер возвращает его клиенту, после чего клиент начинает вычисление:

Каждый экземпляр клиента создает его сокет (создается в конструкторе Integral). При нажатии кнопки «вычислить» создается объект класса RecIntegral, и далее начинается взаимодействие с сервером – отправка запроса на выполнение, получение запроса на вычисление, вычисление, отправка результата серверу и получение результата от сервера соответственно. Далее полученный результат записывается в таблицу:

**Листинг программы:**

Integral.java:

package llab1;

import javax.swing.table.\*;

import java.util.ArrayList;

import javax.swing.JOptionPane;

import javax.swing.\*;

import javax.swing.filechooser.FileNameExtensionFilter;

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.BufferedReader;

import java.io.FileWriter;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.net.\*;

public class Integral extends javax.swing.JFrame {

ArrayList<RecIntegral> number = new ArrayList();

DatagramSocket clientSocket = new DatagramSocket();

InetAddress IPAddress = InetAddress.getByName("localhost");

byte[] receiveData = new byte[1024];

byte[] sendData = new byte[1024];

public Integral() throws SocketException, UnknownHostException {

initComponents();

}

@SuppressWarnings("unchecked")

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

private void initComponents() {

TextBottom = new javax.swing.JTextField();

TextTop = new javax.swing.JTextField();

TextStep = new javax.swing.JTextField();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

jLabel3 = new javax.swing.JLabel();

jLabel4 = new javax.swing.JLabel();

jLabel5 = new javax.swing.JLabel();

ButtonAdd = new javax.swing.JButton();

ButtonCalculate = new javax.swing.JButton();

ButtonClear = new javax.swing.JButton();

ButtonDelete = new javax.swing.JButton();

ButtonFill = new javax.swing.JButton();

jScrollPane1 = new javax.swing.JScrollPane();

jTable1 = new javax.swing.JTable();

ButtonLoadText = new javax.swing.JButton();

ButtonSaveText = new javax.swing.JButton();

ButtonLoadBinary = new javax.swing.JButton();

ButtonSaveBinary = new javax.swing.JButton();

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

setBackground(new java.awt.Color(153, 51, 0));

TextBottom.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED, null, null, null, new java.awt.Color(204, 204, 204)));

TextBottom.setMinimumSize(new java.awt.Dimension(1000, 1000));

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

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

TextBottomActionPerformed(evt);

}

});

TextTop.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED, null, null, null, new java.awt.Color(204, 204, 204)));

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

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

TextTopActionPerformed(evt);

}

});

TextStep.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED, null, null, null, new java.awt.Color(204, 204, 204)));

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

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

TextStepActionPerformed(evt);

}

});

jLabel1.setBackground(new java.awt.Color(0, 204, 204));

jLabel1.setFont(new java.awt.Font("Segoe UI Semibold", 0, 14)); // NOI18N

jLabel1.setText("Исходные данные:");

jLabel1.setBorder(javax.swing.BorderFactory.createEmptyBorder(1, 1, 1, 1));

jLabel2.setText("нижняя граница:");

jLabel3.setText("верхняя граница:");

jLabel4.setText("шаг интегрирования:");

jLabel5.setFont(new java.awt.Font("Segoe UI", 2, 12)); // NOI18N

jLabel5.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel5.setText("<html>вычисление определенного интеграла 1/x</html>");

ButtonAdd.setBackground(new java.awt.Color(255, 204, 153));

ButtonAdd.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N

ButtonAdd.setText("добавить");

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

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

ButtonAddActionPerformed(evt);

}

});

ButtonCalculate.setBackground(new java.awt.Color(255, 204, 153));

ButtonCalculate.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N

ButtonCalculate.setText("вычислить");

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

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

ButtonCalculateActionPerformed(evt);

}

});

ButtonClear.setBackground(new java.awt.Color(255, 153, 153));

ButtonClear.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N

ButtonClear.setText("очистить");

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

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

ButtonClearActionPerformed(evt);

}

});

ButtonDelete.setBackground(new java.awt.Color(255, 204, 153));

ButtonDelete.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N

ButtonDelete.setText("удалить");

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

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

ButtonDeleteActionPerformed(evt);

}

});

ButtonFill.setBackground(new java.awt.Color(255, 153, 153));

ButtonFill.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N

ButtonFill.setText("заполнить");

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

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

ButtonFillActionPerformed(evt);

}

});

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

new Object [][] {

},

new String [] {

"нижняя", "верхняя", "шаг", "результат"

}

) {

Class[] types = new Class [] {

java.lang.Object.class, java.lang.Object.class, java.lang.Object.class, java.lang.Double.class

};

boolean[] canEdit = new boolean [] {

false, true, false, false

};

public Class getColumnClass(int columnIndex) {

return types [columnIndex];

}

public boolean isCellEditable(int rowIndex, int columnIndex) {

return canEdit [columnIndex];

}

});

jTable1.addPropertyChangeListener(new java.beans.PropertyChangeListener() {

public void propertyChange(java.beans.PropertyChangeEvent evt) {

jTable1PropertyChange(evt);

}

});

jScrollPane1.setViewportView(jTable1);

ButtonLoadText.setBackground(new java.awt.Color(255, 204, 204));

ButtonLoadText.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N

ButtonLoadText.setText("загрузить (текстовый)");

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

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

ButtonLoadTextActionPerformed(evt);

}

});

ButtonSaveText.setBackground(new java.awt.Color(255, 204, 204));

ButtonSaveText.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N

ButtonSaveText.setText("сохранить (текстовый)");

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

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

ButtonSaveTextActionPerformed(evt);

}

});

ButtonLoadBinary.setBackground(new java.awt.Color(255, 204, 204));

ButtonLoadBinary.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N

ButtonLoadBinary.setText("загрузить (двоичный)");

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

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

ButtonLoadBinaryActionPerformed(evt);

}

});

ButtonSaveBinary.setBackground(new java.awt.Color(255, 204, 204));

ButtonSaveBinary.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N

ButtonSaveBinary.setText("сохранить (двоичный)");

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

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

ButtonSaveBinaryActionPerformed(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()

.addContainerGap()

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

.addGroup(layout.createSequentialGroup()

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 317, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addGroup(layout.createSequentialGroup()

.addGap(18, 18, 18)

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

.addGroup(layout.createSequentialGroup()

.addComponent(ButtonCalculate, javax.swing.GroupLayout.PREFERRED\_SIZE, 98, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(ButtonDelete, javax.swing.GroupLayout.PREFERRED\_SIZE, 98, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createSequentialGroup()

.addComponent(ButtonClear, javax.swing.GroupLayout.PREFERRED\_SIZE, 98, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(ButtonFill, javax.swing.GroupLayout.PREFERRED\_SIZE, 98, javax.swing.GroupLayout.PREFERRED\_SIZE))))

.addGroup(layout.createSequentialGroup()

.addGap(39, 39, 39)

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

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

.addComponent(ButtonLoadText, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

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

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

.addGap(0, 0, Short.MAX\_VALUE))

.addGroup(layout.createSequentialGroup()

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

.addComponent(jLabel1)

.addGroup(layout.createSequentialGroup()

.addGap(6, 6, 6)

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

.addComponent(jLabel3)

.addGroup(layout.createSequentialGroup()

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

.addComponent(jLabel4)

.addComponent(jLabel2))

.addGap(18, 18, 18)

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

.addComponent(TextTop, javax.swing.GroupLayout.PREFERRED\_SIZE, 68, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(TextStep, javax.swing.GroupLayout.PREFERRED\_SIZE, 68, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(TextBottom, javax.swing.GroupLayout.PREFERRED\_SIZE, 68, javax.swing.GroupLayout.PREFERRED\_SIZE))))

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

.addComponent(ButtonAdd, javax.swing.GroupLayout.PREFERRED\_SIZE, 98, javax.swing.GroupLayout.PREFERRED\_SIZE)))

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

.addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED\_SIZE, 88, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addContainerGap())

);

layout.setVerticalGroup(

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

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

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

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel1)

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

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

.addComponent(jLabel2)

.addComponent(TextBottom, 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(jLabel3)

.addComponent(ButtonAdd)

.addComponent(TextTop, 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(jLabel4)

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

.addGroup(layout.createSequentialGroup()

.addGap(19, 19, 19)

.addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED\_SIZE, 48, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addGap(18, 18, 18)

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

.addGroup(layout.createSequentialGroup()

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

.addComponent(ButtonClear)

.addComponent(ButtonFill))

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

.addComponent(ButtonSaveText)

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

.addComponent(ButtonLoadText)

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

.addComponent(ButtonSaveBinary)

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

.addComponent(ButtonLoadBinary)

.addGap(36, 36, 36)

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

.addComponent(ButtonDelete)

.addComponent(ButtonCalculate)))

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 0, Short.MAX\_VALUE))

.addContainerGap())

);

pack();

}// </editor-fold>

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

// TODO add your handling code here:

}

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

try {

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

RecIntegral rec = number.get(jTable1.getSelectedRow());

String request = "calculate";

sendData = request.getBytes();

DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, 9876);

clientSocket.send(sendPacket);

System.out.println("Клиент отправил запрос на выполнение: " + request);

DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

clientSocket.receive(receivePacket);

String receivedRequest = new String(receivePacket.getData(), 0, receivePacket.getLength());

System.out.println("Клиент получил запрос на вычисление: " + receivedRequest);

if (receivedRequest.equals("calculate")) {

Calculate calculation = new Calculate(rec);

calculation.start();

try {

calculation.join();

} catch (InterruptedException e) {

Calculate.currentThread().interrupt();

}

sendData = String.valueOf(rec.N).getBytes();

DatagramPacket sendResultPacket = new DatagramPacket(sendData, sendData.length, IPAddress, 9876);

clientSocket.send(sendResultPacket);

System.out.println("Клиент вычислил и отправил результат: " + sendData);

DatagramPacket receiveResultPacket = new DatagramPacket(receiveData, receiveData.length);

clientSocket.receive(receiveResultPacket);

String receivedResultRequest = new String(receiveResultPacket.getData(), 0, receiveResultPacket.getLength());

System.out.println("Клиент получил результат: " + receivedResultRequest);

jTable1.setValueAt(Double.parseDouble(receivedResultRequest),jTable1.getSelectedRow(), 3);

}

} catch(Throwable t) {}

}

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

// TODO add your handling code here:

}

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

// TODO add your handling code here:

}

private void jTable1PropertyChange(java.beans.PropertyChangeEvent evt) {

// TODO add your handling code here:

}

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

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

try {

number.remove(jTable1.getSelectedRow());

model.removeRow(jTable1.getSelectedRow());

} catch (Throwable t){}

}

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

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

try {

RecIntegral recIntegral = new RecIntegral(TextBottom.getText(), TextTop.getText(), TextStep.getText());

number.add(recIntegral);

TextBottom.setText("");

TextTop.setText("");

TextStep.setText("");

RecIntegral rec = number.get(number.size() - 1);

model.addRow(new Object[] {rec.A, rec.B, rec.H});

} catch(MyException ex) {

JOptionPane.showMessageDialog(null, ex.getMessage());

}

}

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

((DefaultTableModel)jTable1.getModel()).setRowCount(0);

}

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

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

((DefaultTableModel)jTable1.getModel()).setRowCount(0);

if (jTable1.getRowCount() != number.size())

for (RecIntegral rec:number)

model.addRow(new Object[] {rec.A, rec.B, rec.H, rec.N});

}

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

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

number.clear();

try {

JFileChooser dlg = new JFileChooser();

FileNameExtensionFilter filter = new FileNameExtensionFilter("only text(\*.txt)","txt");

dlg.setFileFilter(filter);

dlg.showSaveDialog(this);

FileReader myfile = new FileReader(dlg.getSelectedFile());

try {

BufferedReader reader = new BufferedReader(myfile);

String line;

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

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

if (parts.length == 4){

String a = parts[0];

String b = parts[1];

String h = parts[2];

try {

RecIntegral recIntegral = new RecIntegral(a,b,h);

recIntegral.N = Double.parseDouble(parts[3]);

number.add(recIntegral);

} catch(MyException ex) {

JOptionPane.showMessageDialog(null, ex.getMessage());

}

}

}

} catch (IOException ex) {

JOptionPane.showMessageDialog(null, "Неправильный формат данных");

}

if (jTable1.getRowCount() != number.size())

for (RecIntegral rec:number ){

model.addRow(new Object[] {rec.A, rec.B, rec.H, rec.N});

}

} catch (IOException ex){

ex.printStackTrace();

}

}

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

try {

JFileChooser dlg = new JFileChooser();

FileNameExtensionFilter filter = new FileNameExtensionFilter("only text(\*.txt)","txt");

dlg.setFileFilter(filter);

dlg.showSaveDialog(this);

FileWriter myfile = new FileWriter(dlg.getSelectedFile());

try {

for (RecIntegral rec:number ){

myfile.write((String.valueOf(rec.A) + ';' + String.valueOf(rec.B) + ';'+ String.valueOf(rec.H) + ';' + String.valueOf(rec.N) + '\n'));

}

} catch (IOException ex){

ex.printStackTrace();

}

myfile.flush();

myfile.close();

} catch (IOException ex){

ex.printStackTrace();

}

}

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

ObjectOutputStream out = null;

try {

JFileChooser dlg = new JFileChooser();

FileNameExtensionFilter filter = new FileNameExtensionFilter("only .ser","ser");

dlg.setFileFilter(filter);

dlg.showSaveDialog(this);

out = new ObjectOutputStream(new BufferedOutputStream(new FileOutputStream(dlg.getSelectedFile())));

out.writeObject(number);

out.flush();

out.close();

} catch (IOException ex) {

ex.printStackTrace();

}

}

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

ObjectInputStream in = null;

number.clear();

try {

JFileChooser dlg = new JFileChooser();

FileNameExtensionFilter filter = new FileNameExtensionFilter("only .ser","ser");

dlg.setFileFilter(filter);

dlg.showOpenDialog(this);

in = new ObjectInputStream(new BufferedInputStream(new FileInputStream(dlg.getSelectedFile())));

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

} catch (IOException ex) {

JOptionPane.showMessageDialog(null, "Неправильный формат данных");

} catch (ClassNotFoundException ex) {

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

ex.printStackTrace();

}

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

if (jTable1.getRowCount() != number.size())

for (RecIntegral rec:number ){

model.addRow(new Object[] {rec.A, rec.B, rec.H, rec.N});

}

}

public static void main(String args[]) {

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

public void run() {

try {

new Integral().setVisible(true);

} catch (SocketException ex) {

ex.printStackTrace();

} catch (UnknownHostException ex) {

ex.printStackTrace();

}

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton ButtonAdd;

private javax.swing.JButton ButtonCalculate;

private javax.swing.JButton ButtonClear;

private javax.swing.JButton ButtonDelete;

private javax.swing.JButton ButtonFill;

private javax.swing.JButton ButtonLoadBinary;

private javax.swing.JButton ButtonLoadText;

private javax.swing.JButton ButtonSaveBinary;

private javax.swing.JButton ButtonSaveText;

private javax.swing.JTextField TextBottom;

private javax.swing.JTextField TextStep;

private javax.swing.JTextField TextTop;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabel5;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JTable jTable1;

// End of variables declaration

}

RecIntegral.java:

package llab1;

import java.io.Serializable;

public class RecIntegral implements Serializable

{

double A, B, H, N;

public RecIntegral(String \_A, String \_B, String \_H) throws MyException {

try {

this.A = Double.parseDouble(\_A);

if (this.A < 0.000001 || this.A > 1000000)

throw new MyException("Число A не удовлетворяет диапазону допустимых значений");

}

catch(IllegalArgumentException e) {

throw new MyException("Неверный ввод нижнего порога");

}

try {

this.B = Double.parseDouble(\_B);

if (this.B < 0.000001 || this.B > 1000000)

throw new MyException("Число B не удовлетворяет диапазону допустимых значений");

if (this.A > this.B)

throw new MyException("Нижний порог интегрирования превышает верхний");

}

catch(IllegalArgumentException e) {

throw new MyException("Неверный ввод верхнего порога");

}

try {

this.H = Double.parseDouble(\_H);

if (this.H < 0.000001 || this.H > 1000000)

throw new MyException("Число H не удовлетворяет диапазону допустимых значений");

}

catch(IllegalArgumentException e) {

throw new MyException("Неверный ввод шага интегрирования");

}

}

public RecIntegral() {

A = 0;

B = 0;

H = 0;

}

}

MyException.java:

package llab1;

public class MyException extends Exception {

MyException(String string) {

super(string);

}

}

Calculate.java:

package llab1;

public class Calculate extends Thread {

RecIntegral rec;

public Calculate(RecIntegral rec) {

this.rec = rec;

}

public void run() {

try {

double result = 0.0;

rec.N = (rec.B - rec.A) / rec.H;

int N1 = (int)rec.N;

for (int i = 0; i < N1; i++) {

result += (1 / (rec.A + rec.H \* i) + 1 /(rec.A + rec.H \* (i + 1))) \* (rec.H / 2);

if (rec.N % 1 != 0 ) {

double H1= rec.B - (rec.A + rec.H \* N1);

result += (1 / (rec.A + H1 \* i) + 1 /(rec.A + H1 \* (i + 1))) \* (H1 / 2);

}

}

rec.N = result;

} catch(Throwable t) {}

}

}

Server.java:

package llab1;

import java.io.IOException;

import java.net.\*;

public class Server {

public static void main(String[] args) {

try {

DatagramSocket serverSocket = new DatagramSocket(9876);

System.out.println("Создан сокет сервера");

byte[] receiveData = new byte[1024];

byte[] sendData = new byte[1024];

while (true) {

DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

serverSocket.receive(receivePacket);

String receivedRequest = new String(receivePacket.getData(), 0, receivePacket.getLength());

if (receivedRequest.equals("calculate")) {

System.out.println("Сервер получил запрос на выполнение: " + receivedRequest);

String request = "calculate";

sendData = request.getBytes();

InetAddress clientIPAddress = receivePacket.getAddress();

int clientPort = receivePacket.getPort();

DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, clientIPAddress, clientPort);

serverSocket.send(sendPacket);

System.out.println("Сервер отправил запрос на вычисление: " + sendData);

DatagramPacket receiveResultPacket = new DatagramPacket(receiveData, receiveData.length);

serverSocket.receive(receiveResultPacket);

String result = new String(receiveResultPacket.getData(), 0, receiveResultPacket.getLength());

sendData = result.getBytes();

DatagramPacket sendResultPacket = new DatagramPacket(sendData, sendData.length, clientIPAddress, clientPort);

serverSocket.send(sendResultPacket);

System.out.println("Сервер получил результат и отправил клиенту: " + sendData);

}

}

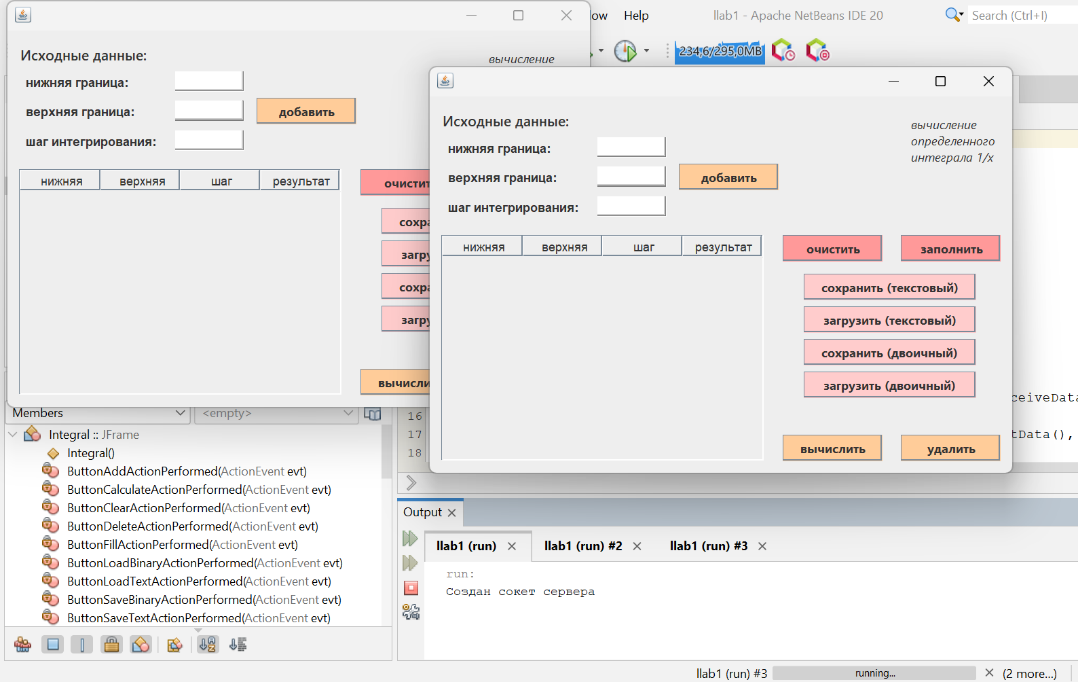
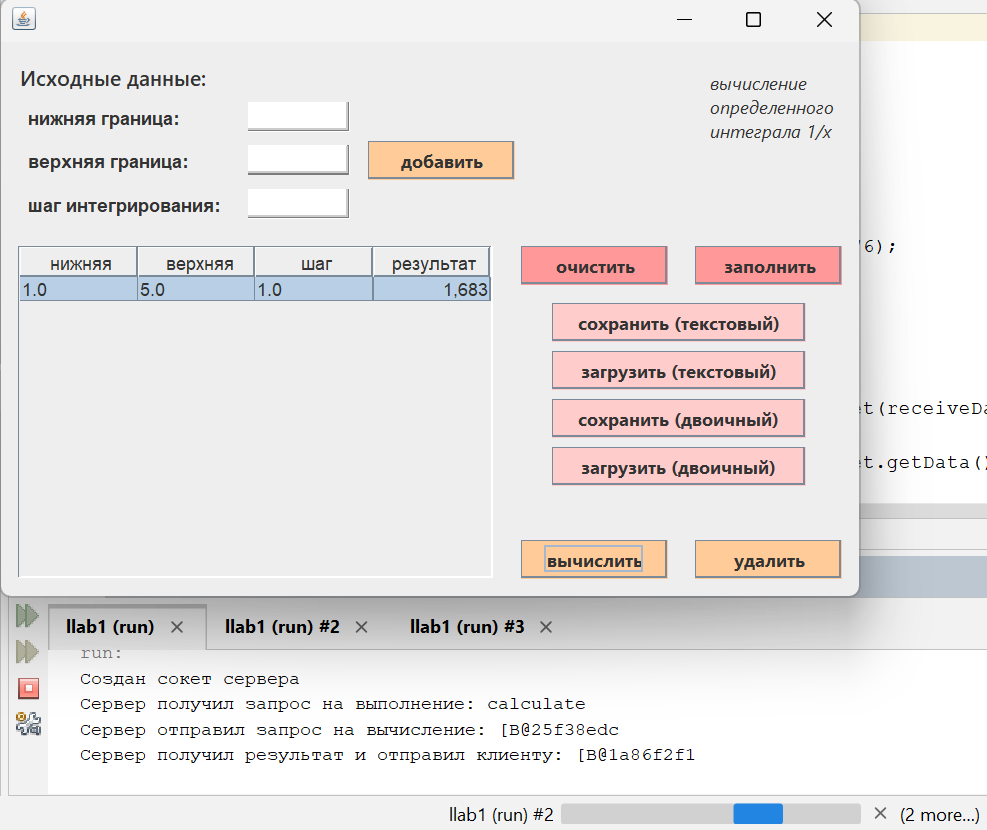
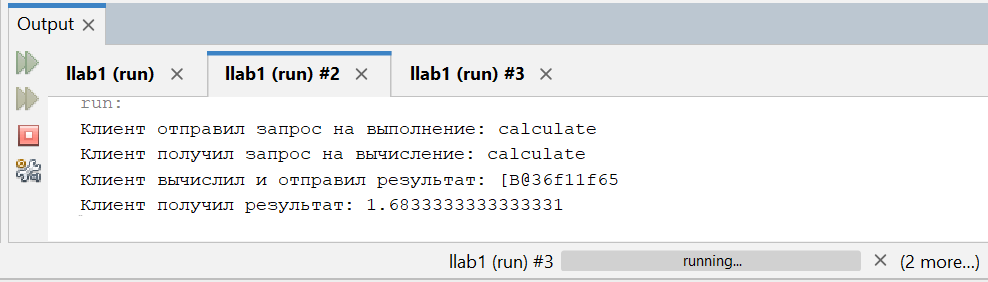
} catch (IOException ex) {

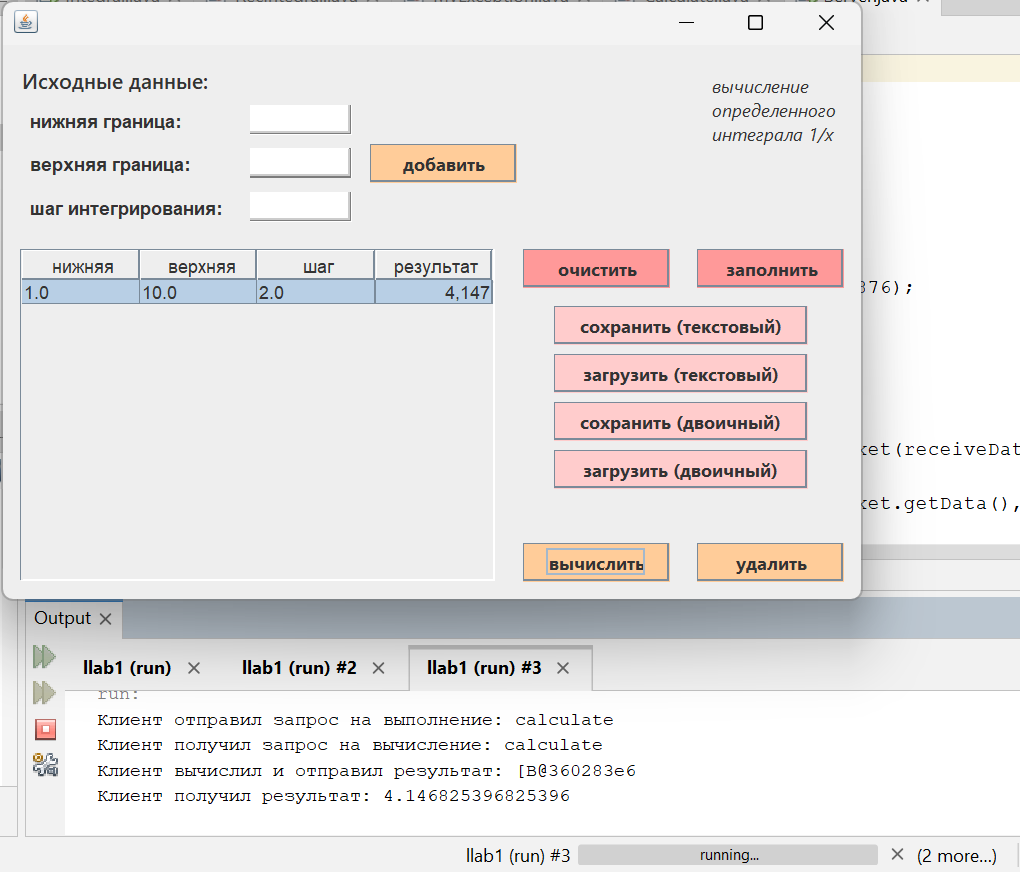
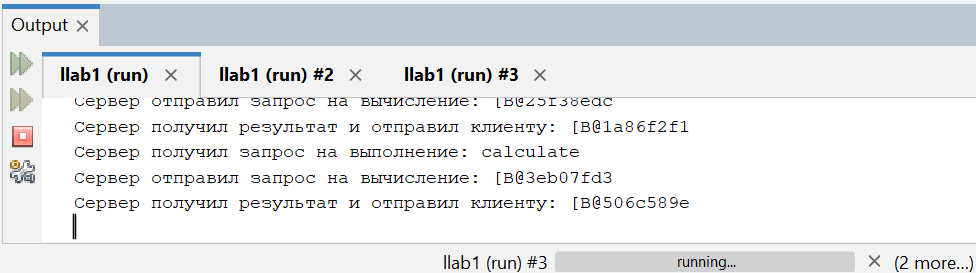
ex.printStackTrace();

}

}

}

**Результаты работы программы:**

**Вывод:** реализована клиент-серверная архитектура приложения на основе протокола UDP – сервер получает от клиента запрос на вычисление определенного интеграла, отправляет запрос и получает результат. Использованы объекты класса DatagramSocket и DatagramPacket, их методы send() и receive().