Пензенский государственный университет

Факультет вычислительной техники Кафедра «Вычислительная техника»

Программирование на языке JAVA

Отчет о лабораторной работе № 6 по теме «Сетевое взаимодействие в Java»

Вариант № 10

Выполнили студенты гр. 21ВВП2

Савосин В. А.  
Кирмасов И. Д.

Проверили:

Юрова О. В.  
Карамышева Н. С.

2024

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

**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. В нашем варианте мы используем протокол UDP.

**Листинг программы:**

**Класс сервера**

**package Server;**

**import java.io.\*;**

**import java.net.\*;**

**import java.util.\*;**

**public class Server {**

**private static LinkedList<RecIntegral> tableData = new LinkedList<>();**

**private static List<InetSocketAddress> clientAddresses = new ArrayList<>();**

**private static DatagramSocket socket;**

**public static void main(String[] args) {**

**int port = 4540;**

**try {**

**socket = new DatagramSocket(port);**

**System.out.println("Server started on port: " + port);**

**while (true) {**

**byte[] receiveData = new byte[1024];**

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

**socket.receive(receivePacket);**

**InetAddress clientAddress = receivePacket.getAddress();**

**int clientPort = receivePacket.getPort();**

**InetSocketAddress clientSocketAddress = new InetSocketAddress(clientAddress, clientPort);**

**if (!clientAddresses.contains(clientSocketAddress)) {**

**clientAddresses.add(clientSocketAddress);**

**System.out.println("User " + clientAddress.getHostAddress() + " connected to server.");**

**}**

**byte[] sendData = new byte[1024];**

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

**// Handle received data**

**ObjectInputStream in = new ObjectInputStream(new ByteArrayInputStream(receiveData));**

**Object obj = in.readObject();**

**if (obj instanceof LinkedList<?>) {**

**LinkedList<RecIntegral> newData = (LinkedList<RecIntegral>) obj;**

**synchronized (tableData) {**

**tableData.addAll(newData);**

**System.out.println(tableData.size());**

**}**

**ByteArrayOutputStream outputStream = new ByteArrayOutputStream();**

**ObjectOutputStream out = new ObjectOutputStream(outputStream);**

**out.writeObject(newData);**

**sendData = outputStream.toByteArray();**

**}**

**// Send data to all clients**

**for (InetSocketAddress address : clientAddresses) {**

**sendPacket.setSocketAddress(address);**

**sendPacket.setData(sendData);**

**socket.send(sendPacket);**

**}**

**}**

**} catch (IOException | ClassNotFoundException e) {**

**e.printStackTrace();**

**} finally {**

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

**socket.close();**

**}**

**}**

**}**

**}**

**Класс потока обновления данных**

package lab1;

import java.io.ByteArrayInputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.SocketException;

import java.util.LinkedList;

import javax.swing.table.DefaultTableModel;

public class UpdateTableTask extends Thread {

private DefaultTableModel tableModel;

private DatagramSocket socket;

public UpdateTableTask(DefaultTableModel tableModel, DatagramSocket socket) {

this.tableModel = tableModel;

this.socket = socket;

}

@Override

public void run() {

try {

while (true) {

byte[] receiveBuffer = new byte[1024];

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

socket.receive(receivePacket);

ByteArrayInputStream byteIn = new ByteArrayInputStream(receivePacket.getData());

ObjectInputStream objectIn = new ObjectInputStream(byteIn);

Object obj = objectIn.readObject();

if (obj instanceof LinkedList<?>) {

LinkedList<RecIntegral> newData = (LinkedList<RecIntegral>) obj;

boolean exists = false;

for (int i = 0; i < tableModel.getRowCount(); i++) {

Object lowerBound = tableModel.getValueAt(i, 0);

Object upperBound = tableModel.getValueAt(i, 1);

Object intervals = tableModel.getValueAt(i, 2);

Object result = tableModel.getValueAt(i, 3);

if (lowerBound.equals(newData.getLast().getLowerBound()) &&

upperBound.equals(newData.getLast().getUpperBound()) &&

intervals.equals(newData.getLast().getIntervals()) &&

result.equals(newData.getLast().getResult())) {

exists = true;

break;

}

}

// Добавляем запись только если хотя бы одно из значений не равно нулю

if (!exists && newData.getLast().getLowerBound()!= 0.0 &&

newData.getLast().getUpperBound()!= 0.0 &&

newData.getLast().getIntervals()!= 0.0 &&

newData.getLast().getResult() != 0.0) {

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

}

}

}

} catch (SocketException se) {

System.out.println("Disconnected from the server.");

} catch (IOException | ClassNotFoundException e) {

e.printStackTrace();

} finally {

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

socket.close();

}

}

}

}

**Главный класс клиента**

package lab1;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableCellEditor;

import javax.swing.table.TableCellRenderer;

import javax.swing.table.TableColumn;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.LinkedList;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.PrintWriter;

import java.io.ObjectInputStream;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectOutputStream;

import java.util.logging.Level;

import java.util.logging.Logger;

public class Lab1 extends JFrame {

private DefaultTableModel tableModel;

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

public Lab1() {

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

JMenuBar menuBar = new JMenuBar();

JMenu fileMenu = new JMenu("Файл");

JMenu editMenu = new JMenu("Правка");

JMenuItem saveMenuItem = new JMenuItem("Сохранить");

JMenuItem saveBinaryMenuItem = new JMenuItem("Сохранить в двоичном виде");

JMenuItem loadMenuItem = new JMenuItem("Открыть...");

JMenuItem loadBinaryMenuItem = new JMenuItem("Открыть в двоичном виде...");

JMenuItem clearMenuItem = new JMenuItem("Очистить");

JMenuItem undoMenuItem = new JMenuItem("Восстановить");

tableData = new LinkedList<>();

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

DefaultTableModel tableModel = new DefaultTableModel(columns, 0){

@Override

public boolean isCellEditable(int row, int column){

return column != 3;

}

};

JTable table = new JTable(tableModel){

@Override

public TableCellRenderer getCellRenderer(int row, int column){

if (column == 3){

// запрещаем редактирование 4го столбца

return getDefaultRenderer(Object.class);

}

else{

return super.getCellRenderer(row, column);

}

}

@Override

public TableCellEditor getCellEditor(int row, int column){

if (column == 3){

return getDefaultEditor(Object.class);

}

else{

return super.getCellEditor(row, column);

}

}

};

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

column.setCellEditor(null);

// Создаем текстовые поля для ввода данных

JTextField lowerBoundField = new JTextField(10);

JTextField upperBoundField = new JTextField(10);

JTextField intervalField = new JTextField(10);

clearMenuItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

tableModel.setRowCount(0); // Очищаем таблицу

}

});

undoMenuItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

for (RecIntegral rec : tableData) {

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

}

}

});

saveBinaryMenuItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

JFileChooser fileChooser = new JFileChooser();

int returnValue = fileChooser.showSaveDialog(null);

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

File selectedFile = fileChooser.getSelectedFile();

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

oos.writeObject(tableData);

} catch (FileNotFoundException ex) {

ex.printStackTrace();

} catch (IOException ex) {

ex.printStackTrace();

}

}

}

});

saveMenuItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

JFileChooser fileChooser = new JFileChooser();

int returnValue = fileChooser.showSaveDialog(null);

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

File selectedFile = fileChooser.getSelectedFile();

try (PrintWriter writer = new PrintWriter(new FileOutputStream(selectedFile))) {

writer.println(tableData);

} catch (FileNotFoundException ex) {

ex.printStackTrace();

}

}

}

});

loadMenuItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

JFileChooser fileChooser = new JFileChooser();

int returnValue = fileChooser.showOpenDialog(null);

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

File selectedFile = fileChooser.getSelectedFile();

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

tableData = (LinkedList<RecIntegral>) ois.readObject();

for (RecIntegral rec : tableData) {

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

}

} catch (ClassNotFoundException ex) {

ex.printStackTrace();

} catch (FileNotFoundException ex) {

Logger.getLogger(Lab1.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

Logger.getLogger(Lab1.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

});

loadBinaryMenuItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

JFileChooser fileChooser = new JFileChooser();

int returnValue = fileChooser.showOpenDialog(null);

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

File selectedFile = fileChooser.getSelectedFile();

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

tableData = (LinkedList<RecIntegral>) ois.readObject();

} catch (FileNotFoundException ex) {

ex.printStackTrace();

} catch (IOException ex) {

ex.printStackTrace();

} catch (ClassNotFoundException ex) {

ex.printStackTrace();

}

}

}

});

JButton calculateButton = new JButton("Вычислить");

calculateButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int lastRowIndex;

if(table.getRowCount() > 1){

lastRowIndex = table.getRowCount() - 1;

}else{ lastRowIndex = table.getRowCount();}

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

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

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

double temp = 0.0;

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

try {

RecIntegral newElement = new RecIntegral(lowerBound, upperBound, intervals, 0.0);

tableData.add(newElement);

} catch (InvalidInputException ex) {

JOptionPane.showMessageDialog(null, "Exception occurred while constructing a new Class instance.\n" + ex.toString());

}

// Создание и запуск потоков

IntegrationTask thread1 = new IntegrationTask(lowerBound, upperBound / 3, intervals , lastRowIndex, tableModel, tableData, 0.0);

thread1.start();

try {

thread1.join();

} catch (InterruptedException d) {

d.printStackTrace();

}

IntegrationTask thread2 = new IntegrationTask(upperBound / 3, 2 \* upperBound / 3, intervals, lastRowIndex, tableModel, tableData, thread1.getFinalResult());

thread2.start();

try {

thread2.join();

} catch (InterruptedException d) {

d.printStackTrace();

}

IntegrationTask thread3 = new IntegrationTask(2 \* upperBound / 3, upperBound, intervals, lastRowIndex, tableModel, tableData, thread2.getFinalResult());

thread3.start();

try {

thread3.join();

} catch (InterruptedException d) {

d.printStackTrace();

}

}

});

JButton deleteButton = new JButton("-");

deleteButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int selectedRow = table.getSelectedRow();

if (selectedRow != -1) {

tableModel.removeRow(selectedRow);

}

}

});

menuBar.add(fileMenu);

menuBar.add(editMenu);

setJMenuBar(menuBar);

fileMenu.add(loadMenuItem);

fileMenu.add(loadBinaryMenuItem);

fileMenu.add(saveMenuItem);

fileMenu.add(saveBinaryMenuItem);

editMenu.add(clearMenuItem);

editMenu.add(undoMenuItem);

JPanel panel = new JPanel();

panel.add(lowerBoundField);

panel.add(upperBoundField);

panel.add(intervalField);

panel.add(deleteButton);

panel.add(calculateButton);

add(panel, "South");

JScrollPane scrollPane = new JScrollPane(table);

add(scrollPane);

setSize(700, 400);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

}

public static void main(String[] args) {

new Lab1(); // создаем экземпляр приложения

}

}

**Класс InvalidInputException:**

/\*

\* 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 lab1;

/\*\*

\*

\* @author User

\*/

public class InvalidInputException extends Exception {

public InvalidInputException(String message) {

super(message);

}

}

**Класс IntegrationTask:**package lab1;

import java.util.LinkedList;

import javax.swing.JOptionPane;

import javax.swing.SwingUtilities;

import javax.swing.table.DefaultTableModel;

public class IntegrationTask extends Thread {

private double lowerBound;

private double upperBound;

private double intervals;

private int lastRowIndex;

private double result;

private DefaultTableModel tableModel;

private LinkedList<RecIntegral> tableData;

public IntegrationTask(double lowerBound, double upperBound, double intervals, int lastRowIndex, DefaultTableModel tableModel, LinkedList<RecIntegral> tableData, double result) {

this.lowerBound = lowerBound;

this.upperBound = upperBound;

this.intervals = intervals;

this.lastRowIndex = lastRowIndex;

this.tableModel = tableModel;

this.tableData = tableData;

this.result = result;

}

@Override

public void run() {

double sum = 0.0;

double x = lowerBound;

while (x < upperBound) {

double fx1 = Math.cos(x);

double fx2 = Math.cos(Math.min(x + intervals, upperBound));

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

x += intervals;

}

int decimalPlaces = 4; // number of decimal places to round the result

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

result = result + sum;

tableModel.setValueAt(result, lastRowIndex, 3);

}

public double getFinalResult() {

return result;

}

}  
  
  
**Класс RecIntegral:**/\*

\* 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 lab1;

import java.io.Serializable;

public class RecIntegral implements Serializable {

private double lowerBound;

private double upperBound;

private double intervals;

private double result;

public RecIntegral(double lowerBound, double upperBound, double intervals, double result) throws InvalidInputException{

if (lowerBound < 0.000001 ) {throw new InvalidInputException("Entered values are out of bounds: lowerBound < 0.000001");}

if (lowerBound > 1000000 ) {throw new InvalidInputException("Entered values are out of bounds: lowerBound > 1000000");}

if (upperBound < 0.000001 ) {throw new InvalidInputException("Entered values are out of bounds: upperBound < 0.000001");}

if (upperBound > 1000000 ) {throw new InvalidInputException("Entered values are out of bounds: upperBound > 1000000");}

if (intervals < 0.000001 ) {throw new InvalidInputException("Entered values are out of bounds: intervals < 0.000001");}

if (intervals > 1000000 ) {throw new InvalidInputException("Entered values are out of bounds: interval > 1000000");}

// if (lowerBound >= upperBound){throw new InvalidInputException("LowerBound is greater than upperBound");}

this.lowerBound = lowerBound;

this.upperBound = upperBound;

this.intervals = intervals;

this.result = result;

}

public double getLowerBound() {

return lowerBound;

}

public double getResult() {

return result;

}

public double getUpperBound() {

return upperBound;

}

public double getIntervals() {

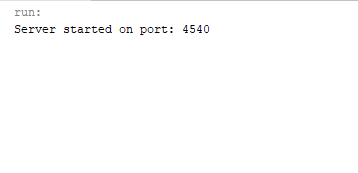
return intervals;

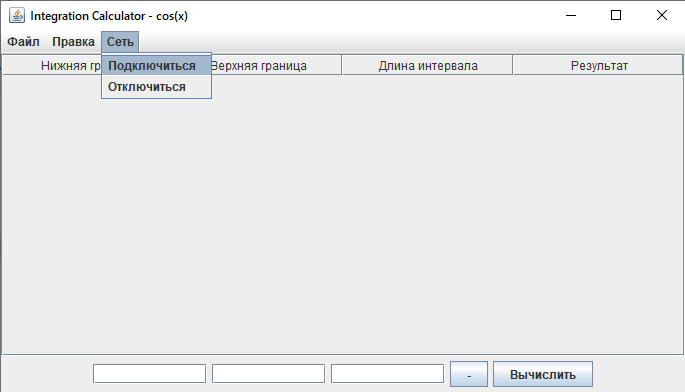
}

}

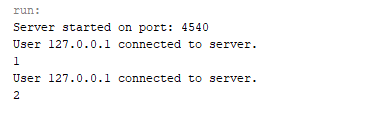
**Результаты работы программы:**

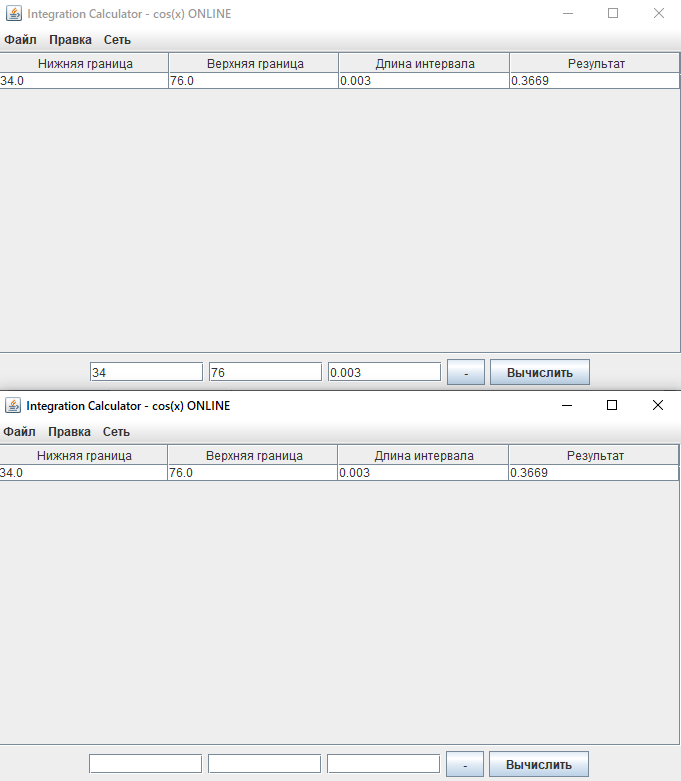
Сначала запускается программа-сервер, задаётся порт.

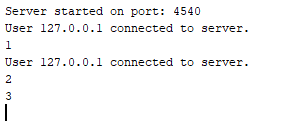


Чтобы подключиться к серверу, нужно выбрать соответствующий пункт меню. 

При подключении клиентов в консоли сервера отображается их IP.



Теперь таблица вычислений синхронизируется с сервером. 

С каждой новой записью от пользователя в структуру сервера записывается объект, что можно видеть по нумерованному вывод в консоли сервера. Цифрами обозначается размер списка. Первые два сообщения пустые, они нужны для сигнала серверу, о том, что пользователь подключился. 

Мы видим, что в качестве агрегатора, сервер работает корректно.

**Вывод:** научились создавать клиент-серверные приложения c использованием стандартных классов Java.