Министерство образования Российской Федерации

Пензенский государственный университет

**Кафедра «Вычислительная техника»**

**ОТЧЁТ**

По лабораторной работе №4

По дисциплине:

«Программирование на языке JAVA»

На тему:

«Работа с файлами»

Выполнили:

студенты группы 20ВВП1

Верховский М. В.

Лукин В. Д.

Приняли:

к.т.н., доцент, Юрова О.В.

к.т.н., доцент, Карамышева Н.С.

**Пенза 2023**

**Цель работы:** изучить работу с файлами и механизмы сериализации данных.

**Задание на лабораторную работу:**

**Вариант 4**

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

**Ход выполнения лабораторной работы:**

**Листинг**

Table.java

import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.\*;  
import java.util.\*;  
  
import static javax.swing.JOptionPane.ERROR\_MESSAGE;  
  
  
public class Table implements Serializable {  
 private LinkedList<RecIntegral> info = new LinkedList<>();  
 private LinkedList<String> values = new LinkedList<>();  
 private JButton AddButton;  
 private JButton DelButton;  
 private JButton CalcButton;  
 private JTextField Down;  
 private JTextField Up;  
 private JTextField Step;  
 private JTable MyTable;  
 private DefaultTableModel MyModel;  
 private JPanel MyPanel;  
 private JButton ClearButton;  
 private JButton StuffButton;  
 private JButton WriteButton;  
 private JButton DownloadButton;  
 private JButton WriteButtonSer;  
 private JButton DownloadButtonSer;  
  
  
 public Table(){  
  
 //Убираем возможность редактирования столбца с результатом  
 MyTable.setModel(new DefaultTableModel() {  
  
 @Override  
 public boolean isCellEditable(int rowIndex, int columnIndex) { // Возвращает true, если ячейка в rowIndex и columnIndex доступна для редактирования  
 return columnIndex !=3;  
 }  
 });  
  
 // Чтобы добавить и удалить строки из таблицы, нужно использовать DefaultTableModel  
 MyModel = (DefaultTableModel) MyTable.getModel();  
  
 // создание столбцов  
 MyModel.addColumn("Низ");  
 MyModel.addColumn("Верх");  
 MyModel.addColumn("Шаг");  
 MyModel.addColumn("Результат");  
  
 AddButton.addActionListener(new ActionListener() { // ActionListener хранит в себе метод, который активируется при нажатии кнопки  
 @Override  
 public void actionPerformed(ActionEvent e) { // actionPerformed - метод, который активируется при нажатии кнопки  
 //Обработка исключений с помощью блоков try-catch:  
 try {  
 if (Double.parseDouble(Step.getText()) == 0){  
 throw new Exception1("Шаг");  
 }  
  
 } catch (Exception1 ex) {  
 ex.Message1(MyPanel, ERROR\_MESSAGE);  
 return; }  
  
  
 try {  
 if (Double.parseDouble(Down.getText()) > 1000000 || Double.parseDouble(Down.getText()) < 0.000001){  
 throw new Exception1("Нижний предел");  
 }  
 if (Double.parseDouble(Up.getText()) > 1000000 || Double.parseDouble(Up.getText()) < 0.000001){  
 throw new Exception1("Верхний предел");  
 }  
 if (Double.parseDouble(Step.getText()) > 1000000 || Double.parseDouble(Step.getText()) < 0.000001){  
 throw new Exception1("Шаг");  
 }  
  
 } catch (Exception1 ex) {  
 ex.Message2(MyPanel, ERROR\_MESSAGE);  
 return; }  
  
  
 try {  
 if (Double.parseDouble(Down.getText()) > Double.parseDouble(Up.getText())){  
 throw new Exception1("Пределы");  
 }  
  
 } catch (Exception1 ex) {  
 ex.Message3(MyPanel, ERROR\_MESSAGE);  
 return; }  
  
  
 Double Up = Double.parseDouble(Table.this.Up.getText()); // getText возвращает текст, содержащийся в текстовом поле Up  
 Double Down = Double.parseDouble(Table.this.Down.getText());  
 Double Step = Double.parseDouble(Table.this.Step.getText());  
 MyModel.addRow(new Double[]{Down, Up, Step}); // addRow добавляет строку  
 info.add(new RecIntegral(Down, Up, Step, 0));//Работа с коллекцией  
  
 }  
 });  
  
 CalcButton.addActionListener(new ActionListener() { // ActionListener хранит в себе метод, который активируется при нажатии кнопки  
 @Override  
 public void actionPerformed(ActionEvent e) { // actionPerformed - метод, который активируется при нажатии кнопки  
 int indexRow = MyTable.getSelectedRow(); // Метод getSelectedRow() возвращает индекс первой выбранной строки  
  
 if (indexRow == -1) {  
 return;  
 }  
  
 double area = 0; // площадь = интеграл  
 Vector data = MyModel.getDataVector().get(indexRow);// в data записываем значения выбранной строки (значения могут меняться)  
 double a = (Double) data.get(0);  
 double b = (Double) data.get(1);  
 double h = (Double) data.get(2);  
  
 for (int i = 0; i < (b - a) / h; i++) {  
 if (a+(i+1)\*h<=b){  
 area += h \* (0.5 \* (Math.tan(a+i\*h) + Math.tan(a+(i+1)\*h)));  
 }  
 else {  
 area += h \* (0.5 \* (Math.tan(a+i\*h) + Math.tan(b)));  
 }  
  
 }  
  
 MyModel.setValueAt(area, indexRow, 3); // установка значения интеграла в строку с индексом в столбце №3  
 info.get(indexRow).Res = area;//Работа с коллекцией  
  
 }  
 });  
  
 DelButton.addActionListener(new ActionListener() { // ActionListener хранит в себе метод, который активируется при нажатии кнопки  
 @Override  
 public void actionPerformed(ActionEvent e) { // actionPerformed - метод, который активируется при нажатии кнопки  
 int indexRow = MyTable.getSelectedRow(); // Метод getSelectedRow() возвращает индекс первой выбранной строки  
 if (indexRow != -1){  
 MyModel.removeRow(indexRow); // Метод removeRow() имеет параметр ind - индекс строки, которая будет удалена  
 info.remove(indexRow);//Работа с коллекцией  
 }  
 }  
 });  
  
  
 //Очистить таблицу  
 ClearButton.addActionListener(new ActionListener() { // ActionListener хранит в себе метод, который активируется при нажатии кнопки  
 @Override  
 public void actionPerformed(ActionEvent e) { // actionPerformed - метод, который активируется при нажатии кнопки  
 //Каждый раз, когда удаляем строку, количество строк будет меняться. Продолжаем цикл до тех пор, пока не останется строк  
 while (MyModel.getRowCount()>0)  
 {  
 MyModel.removeRow(0);  
 }  
 }  
 });  
  
 //Заполнить таблицу данными из коллекции  
 StuffButton.addActionListener(new ActionListener() { // ActionListener хранит в себе метод, который активируется при нажатии кнопки  
 @Override  
 public void actionPerformed(ActionEvent e) { // actionPerformed - метод, который активируется при нажатии кнопки  
 while (MyModel.getRowCount() > 0) {  
 MyModel.removeRow(0);  
 }  
 //Заполнение таблицы данными из коллекции  
 for (RecIntegral data : info) {  
 MyModel.addRow(new Object[]{data.Down1,data.Up1,data.Step1,data.Res});  
 }  
  
 }  
 });  
  
 //Загрузить в файл  
 WriteButton.addActionListener(new ActionListener(){  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 JFileChooser window = new JFileChooser(); //Создание JFileChooser с указанием директории пользователя по умолчанию  
 window.setCurrentDirectory(new File(".")); //Установка директории по умолчанию  
 window.setDialogTitle("Загрузка в файл"); //Заголовок окна  
 window.setMultiSelectionEnabled(false); //Невозможность выбора сразу нескольких файлов  
 window.setFileSelectionMode(JFileChooser.FILES\_ONLY); //Выбор только файлов  
 window.setSelectedFile(new File("file1.txt")); //Выбор файла по умолчанию  
 window.showDialog(window, "Загрузить в файл"); //Открытие окна выбора файла с настроенным наименованием кнопки  
  
 File MyFile = window.getSelectedFile(); //Чтение выделенного файла  
  
 try {  
 FileWriter myWriter = new FileWriter(MyFile); //Создание объекта класса FileWriter для записи в файл на основе имени файла в файловой системе  
 BufferedWriter buf = new BufferedWriter(myWriter);  
  
 for(int i = 0; i < MyTable.getRowCount(); i++)  
 {  
 for(int j = 0; j < MyTable.getColumnCount(); j++) {  
 buf.write(MyTable.getValueAt(i, j).toString() + " ");  
 }  
 buf.newLine(); //Метод newLine() использует собственное понятие разделителя строк платформы, определенное системным свойством line.separator  
 }  
  
 //Поток закрыт  
 buf.close();  
 myWriter.close();  
  
 } catch (IOException ex) {  
 throw new RuntimeException(ex);  
 }  
  
 }  
 });  
  
 //Загрузить из файла  
 DownloadButton.addActionListener(new ActionListener(){  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 while (MyModel.getRowCount()>0)  
 {  
 MyModel.removeRow(0);  
 }  
  
 JFileChooser window = new JFileChooser(); //Создание JFileChooser с указанием директории пользователя по умолчанию  
 window.setCurrentDirectory(new File(".")); //Установка директории по умолчанию  
 window.setDialogTitle("Загрузка из файла"); //Заголовок окна  
 window.setMultiSelectionEnabled(false); //Невозможность выбора сразу нескольких файлов  
 window.setFileSelectionMode(JFileChooser.FILES\_ONLY); //Выбор только файлов  
 window.setSelectedFile(new File("file1.txt")); //Выбор файла по умолчанию  
 window.showDialog(window, "Загрузить из файла"); //Открытие окна выбора файла с настроенным наименованием кнопки  
  
 File MyFile = window.getSelectedFile(); //Чтение выделенного файла  
  
 try {  
 FileReader myReader = new FileReader(MyFile); //Создание объекта класса FileReader для чтения из файла на основе имени файла в файловой системе  
 BufferedReader buf = new BufferedReader(myReader); //BufferedReader записывает текст в поток вывода символов, буферизуя символы, чтобы обеспечить эффективную запись отдельных символов, массивов и строк  
  
 Object[] lines = buf.lines().toArray(); //Метод lines() — метод, который возвращает поток строк, извлеченных из заданной многострочной строки  
  
 for (int i = 0; i < lines.length; i++) {  
 String[] row = lines[i].toString().split(" "); //Метод split разделяет строку на подстроки, используя разделитель, который определяется с помощью регулярного выражения  
 MyModel.addRow(row);  
 }  
  
 //Поток закрыт  
 buf.close();  
 myReader.close();  
  
 } catch (IOException ex) {  
 throw new RuntimeException(ex);  
 }  
  
 }  
 });  
  
 //Сериализация  
 WriteButtonSer.addActionListener(new ActionListener(){  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 JFileChooser window = new JFileChooser();  
 window.setCurrentDirectory(new File(".")); //Создание JFileChooser с указанием директории пользователя по умолчанию  
 window.setDialogTitle("Выберите файл для сериализации"); //Заголовок окна  
 window.setMultiSelectionEnabled(false); //Невозможность выбора сразу нескольких файлов  
 window.setFileSelectionMode(JFileChooser.FILES\_ONLY); //Выбор только файлов  
 window.setSelectedFile(new File("A.ser")); //Выбор файла по умолчанию  
 window.showDialog(window, "Выбрать"); //Открытие окна выбора файла с настроенным наименованием кнопки  
  
 File MyFile = window.getSelectedFile(); //Чтение выделенного файла  
  
 for (int i = 0; i < MyTable.getRowCount(); i++) {  
 for (int j = 0; j < MyTable.getColumnCount(); j++) {  
 values.add(String.valueOf(MyTable.getValueAt(i, j)));  
 }  
 }  
  
 try {  
 //создаем 2 потока для сериализации объекта и сохранения его в файл  
 FileOutputStream outputStream = new FileOutputStream(MyFile);  
 ObjectOutputStream out = new ObjectOutputStream(outputStream);  
 // сохраняем values в файл  
 out.writeObject(values);  
 //закрываем поток и освобождаем ресурсы  
 outputStream.close();  
 out.close();  
  
 } catch(IOException ex) {  
 ex.printStackTrace();  
 }  
 }  
 });  
  
 //Десериализация  
 DownloadButtonSer.addActionListener(new ActionListener(){  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 while (MyModel.getRowCount()>0)  
 {  
 MyModel.removeRow(0);  
 }  
  
 JFileChooser window = new JFileChooser(); //Создание JFileChooser с указанием директории пользователя по умолчанию  
 window.setCurrentDirectory(new File(".")); //Установка директории по умолчанию  
 window.setDialogTitle("Выберите файл для десериализации"); //Заголовок окна  
 window.setMultiSelectionEnabled(false); //Невозможность выбора сразу нескольких файлов  
 window.setFileSelectionMode(JFileChooser.FILES\_ONLY); //Выбор только файлов  
 window.setSelectedFile(new File("A.ser")); //Выбор файла по умолчанию  
 window.showDialog(window, "Выбрать"); //Открытие окна выбора файла с настроенным наименованием кнопки  
  
 File MyFile = window.getSelectedFile(); //Чтение выделенного файла  
  
 try {  
 //создаем 2 потока для десериализации объекта и выгрузки его из файла  
 FileInputStream inputStream = new FileInputStream(MyFile);  
 ObjectInputStream in = new ObjectInputStream(inputStream);  
  
 values = (LinkedList<String>) in.readObject(); //Метод readObject для чтения объекта из потока  
 //закрываем поток и освобождаем ресурсы  
 in.close();  
 inputStream.close();  
  
 } catch(IOException | ClassNotFoundException ex) {  
 ex.printStackTrace();  
 }  
  
 int i = 0;  
 while(i < values.size())  
 {  
 MyModel.addRow(new Object[] {  
 values.get(i),  
 values.get(i + 1),  
 values.get(i + 2),  
 values.get(i + 3)  
 });  
 i = i + 4;  
 }  
 }  
 });  
  
 }  
  
 public static void main(String[] args) {  
 JFrame frame = new JFrame("Интегральная гипербола кибернетики"); // создаём окно с рамкой  
 frame.setVisible(true); // показывает или скрывает это окно в зависимости от значения параметра  
 frame.setContentPane(new Table().MyPanel);//задаёт содержимое окна с рамкой  
 frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);//Устанавливает операцию при закрытии окна закончить работу приложения,  
 // которая будет выполняться по умолчанию, когда пользователь инициирует «закрытие» этого окна  
 frame.setSize(420,600); // размер окна  
 frame.setLocation(120,100); // расположение окна  
  
 }  
  
}

RecIntegral.java

public class RecIntegral {  
  
 public double Up1;  
 public double Down1;  
 public double Step1;  
 public double Res;  
  
 public RecIntegral(double Down1, double Up1, double Step1, double Res){  
 this.Down1 = Down1;  
 this.Up1 = Up1;  
 this.Step1 = Step1;  
 this.Res = Res;  
  
 }  
  
}

Exception1.java

import java.awt.\*;  
import static javax.swing.JOptionPane.showMessageDialog;  
  
//класс Exception1, унаследованный от класса Exception:  
public class Exception1 extends Exception{  
  
 private String title;  
  
 public Exception1(String title) {  
 this.title = title;  
 }  
  
 //Диалоговое окно с заголовком, cообщением и типом сообщения  
 public void Message1(Component parent, int optionType) {  
 showMessageDialog(parent, "Введён нулевой шаг", title, optionType);  
 }  
 public void Message2(Component parent, int optionType) {  
 showMessageDialog(parent, "Выход за пределы диапазона (0.000001; 1000000)", title, optionType);  
 }  
  
 public void Message3(Component parent, int optionType) {  
 showMessageDialog(parent, "Верхний предел не может быть меньше нижнего предела", title, optionType);  
 }  
}

**Результаты выполнения программы**

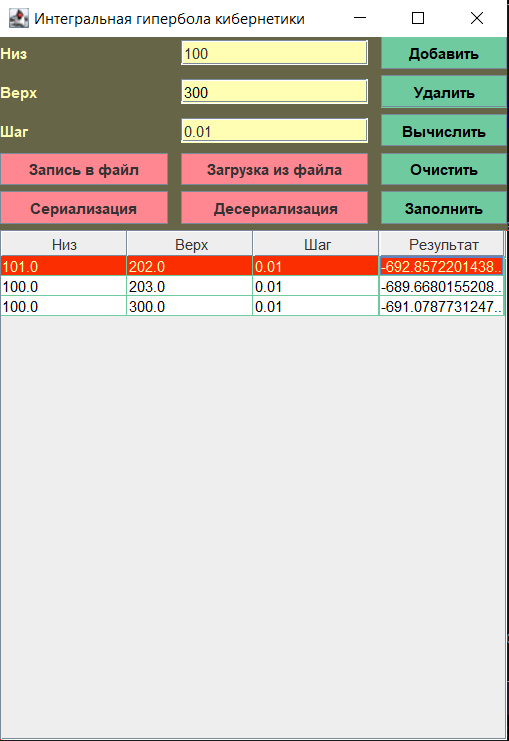


Рисунок №1 – Изначальная таблица

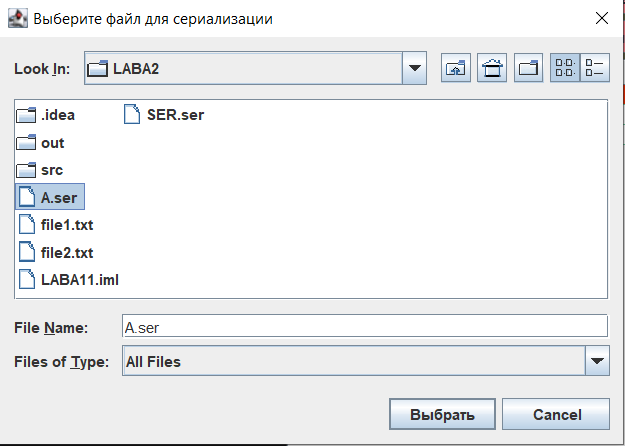


Рисунок №2 – Выбор файла для сериализации

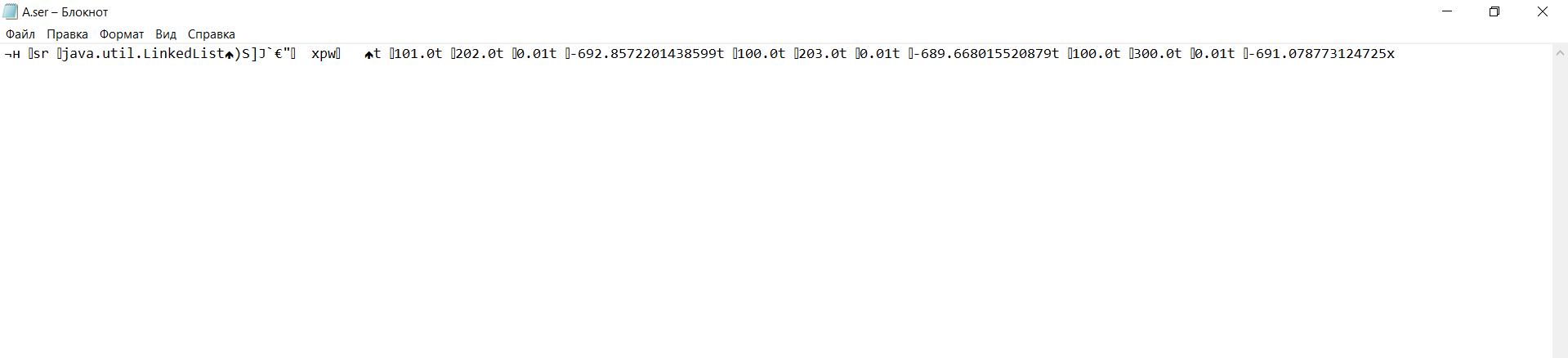


Рисунок №3 – Результат сериализации, записанный в файл по умолчанию A.ser

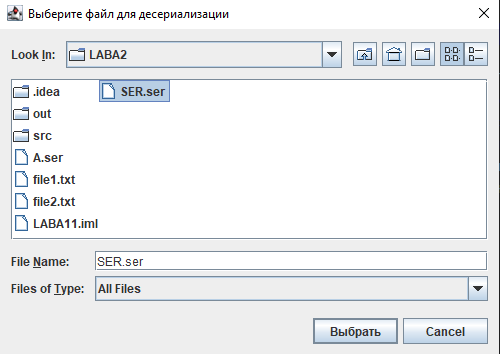


Рисунок №4 – Выбор файла для десериализации

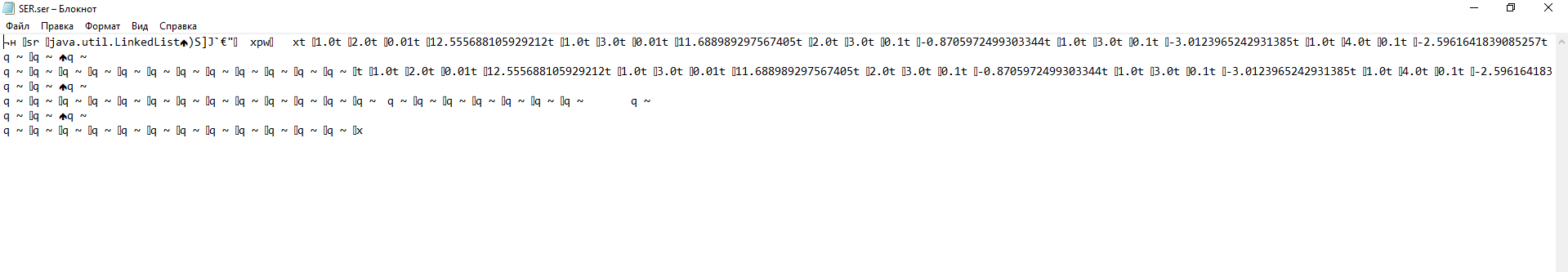


Рисунок №5 – Содержимое файла SER.ser



Рисунок №6 – Результат десериализации

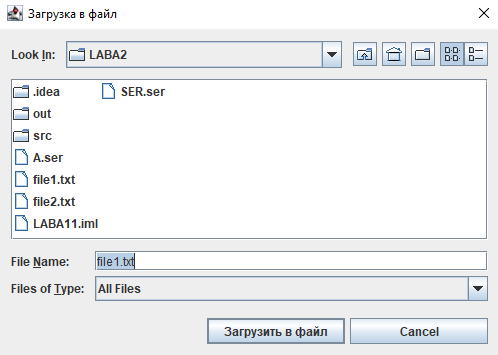


Рисунок №7 – Выбор файла file1.txt по умолчанию для загрузки значений из таблицы в него

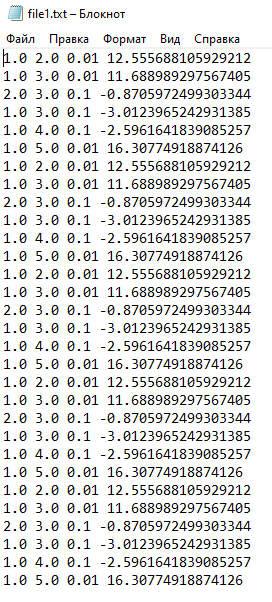


Рисунок №8 – Содержимое файла file1.txt после загрузки в него значений из таблицы

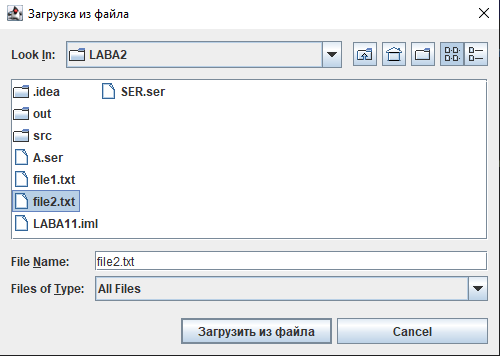


Рисунок №9 – Выбор файла file2.txt для загрузки из него значений в таблицу

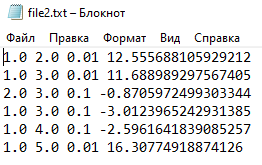


Рисунок №10 – Содержимое файла file2.txt

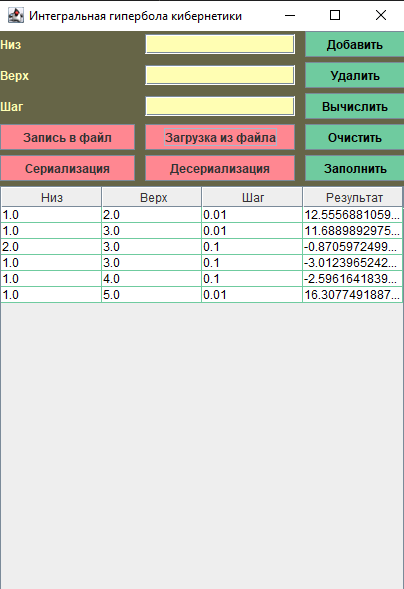


Рисунок №11 – Результат загрузки из файла

**Вывод:** Мы изучили работу с файлами и механизмы сериализации данных.