КАЗАНСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. А.Н.Туполева - КАИ

(КНИТУ-КАИ)

Отделение среднего профессионального образования института

Компьютерных технологий и защиты информации

«Колледж информационных технологий»

Лабораторная работа №5

по предмету:

Наладчик технологического оборудования

Выполнил:

студент 4 курса

Группы 4441

Зарипов Ислам Радифович

Казань 2019 г.

**Цель работы:**

Разработать программу на языке Java для работы с .xls файлами.

**Решение:**

Были установлены слушатели на кнопки «Новый», «Открыть», «Сохранить», «Закрыть». При нажатии на копку «Новый» в памяти инициализируется пустой документ, открывается возможность редактировать документ (см. Рис. 2). Если было вызвано событие открытия файла, то отображается окно выбора файла (см. Рис. 3), затем – содержание файла можно редактировать в текстовом поле программы. После изменения, файл можно сохранить, нажав на кнопку «Сохранить». Так же файл можно закрыть нажатие кнопки «Закрыть». Предусмотрена проверка изменения файла при закрытии файла (см. Рис. 4). Для чтения данных из .xls фала была выбрана библиотека Apache POI, подключение через Maven:

<dependencies>

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi</artifactId>

<version>3.12</version>

</dependency>

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>3.12</version>

</dependency>

</dependencies>

Добавлена поддержка функции SIN(X), где X – число, ссылка на число или функция числа ПИ, пример (см. Рис. 6 – SIN(A1), где A1 содержит 3.1415926, результат вычисления = 0).

**Скриншоты работы программы на Java:**

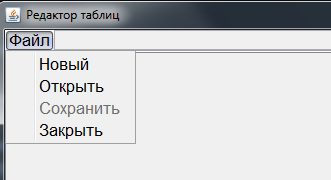
****

Рис. 1. Контекстное меню программы

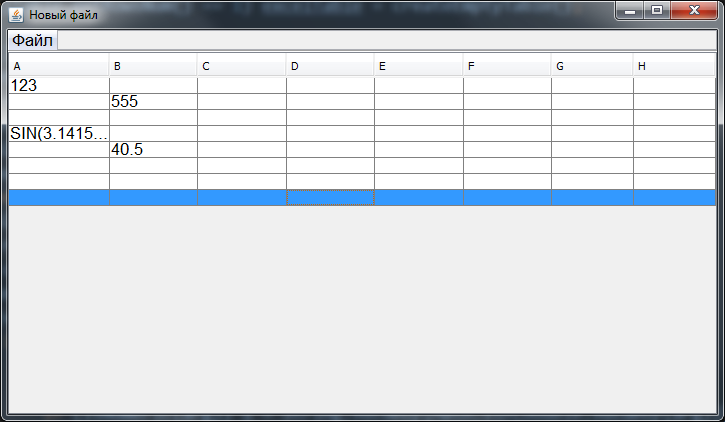


Рис. 2. Окно редактирования нового файла

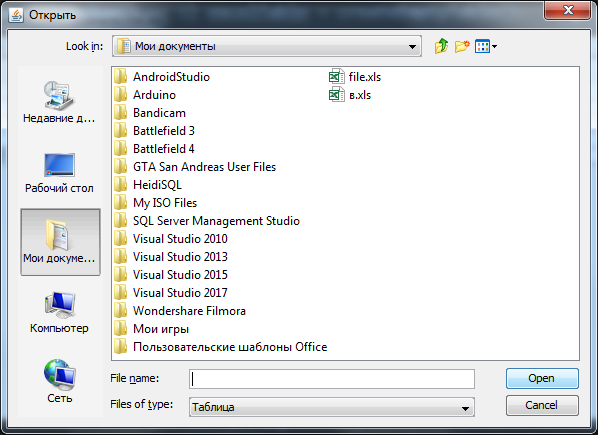


Рис. 3. Окно выбора файла для открытия

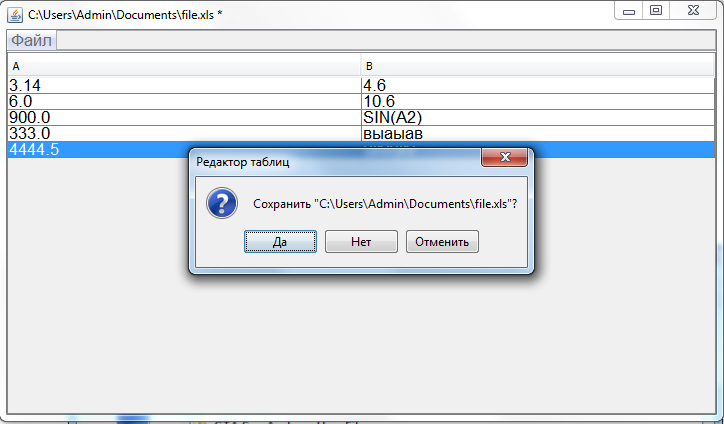


Рис. 4. Окно подтверждения «сохранить или нет»

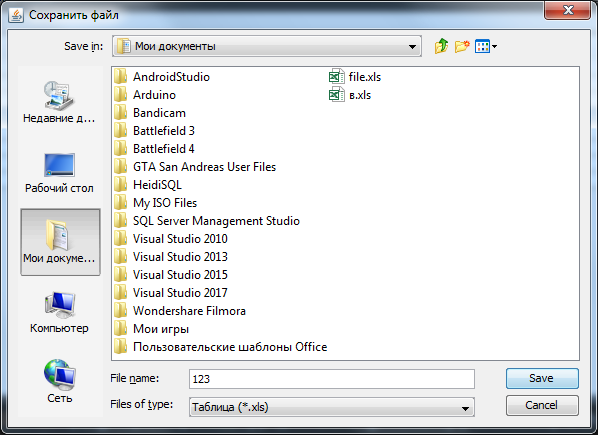


Рис. 5. Окно сохранения нового файла

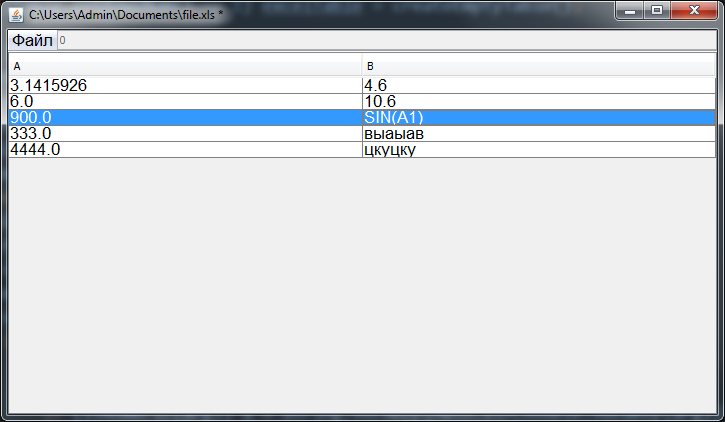


Рис. 6. Пример функции SIN(X)

**Вывод:**

В ходе лабораторной работы была разработана программа на языке Java для работы с .xls файлами.

**Приложение**

**Файл Main.java:**

package ru.inkrot.kit.exceleditor;

import javax.swing.\*;

public class Main {

public Main() {

Screen screen = new Screen();

screen.setVisible(true);

}

public static void main(String[] args) throws Exception {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

new Main();

}

}

**Файл Screen.java:**

package ru.inkrot.kit.exceleditor;

import org.apache.poi.hssf.usermodel.HSSFCell;

import org.apache.poi.hssf.usermodel.HSSFRow;

import org.apache.poi.hssf.usermodel.HSSFSheet;

import org.apache.poi.hssf.usermodel.HSSFWorkbook;

import org.apache.poi.ss.usermodel.Row;

import org.apache.poi.ss.usermodel.Sheet;

import org.apache.poi.ss.usermodel.Workbook;

import javax.swing.\*;

import javax.swing.filechooser.FileNameExtensionFilter;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.util.ArrayList;

public class Screen extends JFrame implements ActionListener {

public final String TITLE = "Редактор таблиц";

public final String ACTION\_NEW = "ACTION\_NEW";

public final String ACTION\_OPEN = "ACTION\_OPEN";

public final String ACTION\_SAVE = "ACTION\_SAVE";

public final String ACTION\_CLOSE = "ACTION\_CLOSE";

private String selectedValue = null;

private Point selectedCell = null;

private boolean edited = false;

private static File currentFile = null;

private Font font;

private JScrollPane tableScroll;

private JTable table;

private JMenuItem saveBtn;

private JTextField formulaField;

public Screen() {

setDefaultCloseOperation(WindowConstants.DO\_NOTHING\_ON\_CLOSE);

setLayout(new BorderLayout());

setSize(725, 422);

setLocationRelativeTo(null);

initGui();

closeEdit();

}

private void initGui() {

UIManager.put("OptionPane.cancelButtonText", "Отменить");

UIManager.put("OptionPane.yesButtonText", "Да");

UIManager.put("OptionPane.noButtonText", "Нет");

addWindowListener(new WindowListener() {

@Override

public void windowOpened(WindowEvent e) {

}

@Override

public void windowClosing(WindowEvent e) {

if (checkIsSaved()) System.exit(0);

}

@Override

public void windowClosed(WindowEvent e) {

}

@Override

public void windowIconified(WindowEvent e) {

}

@Override

public void windowDeiconified(WindowEvent e) {

}

@Override

public void windowActivated(WindowEvent e) {

}

@Override

public void windowDeactivated(WindowEvent e) {

}

});

font = new Font("Arial", 0, 17);

JMenuBar menuBar = new JMenuBar();

JMenu fileMenu = new JMenu("Файл");

fileMenu.setFont(font);

menuBar.add(fileMenu);

JMenuItem newBtn = new JMenuItem("Новый");

newBtn.setFont(font);

newBtn.addActionListener(this);

newBtn.setActionCommand(ACTION\_NEW);

fileMenu.add(newBtn);

JMenuItem openBtn = new JMenuItem("Открыть");

openBtn.setFont(font);

openBtn.addActionListener(this);

openBtn.setActionCommand(ACTION\_OPEN);

fileMenu.add(openBtn);

saveBtn = new JMenuItem("Сохранить");

saveBtn.setFont(font);

saveBtn.addActionListener(this);

saveBtn.setActionCommand(ACTION\_SAVE);

fileMenu.add(saveBtn);

JMenuItem closeBtn = new JMenuItem("Закрыть");

closeBtn.setFont(font);

closeBtn.addActionListener(this);

closeBtn.setActionCommand(ACTION\_CLOSE);

fileMenu.add(closeBtn);

formulaField = new JTextField("");

formulaField.setEnabled(false);

menuBar.add(formulaField);

initTable(null);

setJMenuBar(menuBar);

}

@Override

public void actionPerformed(ActionEvent e) {

String action = e.getActionCommand();

if (action.equals(ACTION\_NEW)) {

if (! checkIsSaved()) return;

currentFile = null;

initTable(null);

setEdited(false);

openEdit();

} else if (action.equals(ACTION\_OPEN)) {

if (! checkIsSaved()) return;

JFileChooser chooser = new JFileChooser();

chooser.setDialogTitle("Открыть");

FileNameExtensionFilter filter = new FileNameExtensionFilter("Таблица", "xls");

chooser.addChoosableFileFilter(filter);

chooser.setAcceptAllFileFilterUsed(false);

int returnVal = chooser.showOpenDialog(this);

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

if (!chooser.getSelectedFile().exists()) {

JOptionPane.showMessageDialog(null, "Файл не найден");

return;

}

currentFile = chooser.getSelectedFile();

initTable(currentFile);

setTitle(currentFile.getAbsolutePath());

openEdit();

}

} else if (action.equals(ACTION\_SAVE)) {

saveCurrentFile();

} else if (action.equals(ACTION\_CLOSE)) {

if (! checkIsSaved()) return;

closeEdit();

}

}

private ExcelTable readFromExcel(File file) {

HSSFWorkbook book = null;

InputStream inputStream = null;

try {

inputStream = new FileInputStream(file);

book = new HSSFWorkbook(inputStream);

} catch (IOException e) {

e.printStackTrace();

}

HSSFSheet sheet = book.getSheetAt(0);

ExcelTable excelTable = new ExcelTable();

for (int r = 0; r <= sheet.getLastRowNum(); r++) {

HSSFRow row = sheet.getRow(r);

excelTable.newRow();

if (row == null) continue;

for (int i = 0; i < row.getLastCellNum(); i++) {

HSSFCell cell = row.getCell(i);

if (cell == null) {

excelTable.addColumn(null);

continue;

}

String value = null;

switch (cell.getCellType()) {

case HSSFCell.CELL\_TYPE\_STRING:

value = row.getCell(i).getStringCellValue();

break;

case HSSFCell.CELL\_TYPE\_NUMERIC:

value = String.valueOf(row.getCell(i).getNumericCellValue());

break;

case HSSFCell.CELL\_TYPE\_FORMULA:

value = String.valueOf(row.getCell(i).getCellFormula());

break;

}

excelTable.addColumn(value);

}

}

try {

book.close();

inputStream.close();

} catch (IOException e) {

e.printStackTrace();

}

return excelTable;

}

private boolean isNumeric(String str) {

if (str == null) return false;

try {

Double.parseDouble(str);

return true;

} catch(NumberFormatException e){

return false;

}

}

private void writeIntoExcel() {

Workbook book = new HSSFWorkbook();

Sheet sheet = book.createSheet();

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

Row row = sheet.createRow(i);

for (int j = 0; j < table.getColumnCount(); j++) {

String val = (String) table.getValueAt(i, j);

if (val == null) {

row.createCell(j).setCellFormula(null);

continue;

}

if (isNumeric(val)) row.createCell(j).setCellValue(Double.valueOf(val));

else if (val.startsWith("SIN(")) row.createCell(j).setCellFormula(val);

else row.createCell(j).setCellValue(val);

}

}

try {

OutputStream outputStream = null;

outputStream = new FileOutputStream(currentFile);

book.write(outputStream);

book.close();

outputStream.close();

} catch (IOException e) {

e.printStackTrace();

}

}

private ExcelTable createEmptyTable() {

ExcelTable excelTable = new ExcelTable();

for (int i = 0; i <= 7; i++) {

excelTable.newRow();

for (int j = 0; j <= 7; j++) {

excelTable.addColumn(null);

}

}

return excelTable;

}

private void initTable(File file) {

ExcelTable excelTable = createEmptyTable();

if (tableScroll != null) remove(tableScroll);

if (file != null) {

excelTable = readFromExcel(file);

if (excelTable.getRowsNum() == 1) excelTable = createEmptyTable();

}

table = new JTable(excelTable.getAsArray(), excelTable.getHeaders());

table.setFont(font);

tableScroll = new JScrollPane(table);

table.setPreferredScrollableViewportSize(new Dimension(250, 100));

table.getTableHeader().setReorderingAllowed(false);

table.addMouseListener(new MouseAdapter() {

public void mousePressed(MouseEvent mouseEvent) {

JTable table = (JTable) mouseEvent.getSource();

int row = table.getSelectedRow();

int column = table.getSelectedColumn();

String value = ((String)table.getValueAt(row, column));

if (value != null) {

if (value.startsWith("SIN(")) {

String arg = value.substring(4, value.length() - 1);

String s;

if (isNumeric(arg)) s = String.valueOf(Math.sin(Double.valueOf(arg)));

else if (arg.equals("PI()")) s = String.valueOf(Math.sin(Math.PI));

else {

int r = Integer.valueOf(String.valueOf(arg.charAt(1))) - 1;

int c = arg.charAt(0) - 65;

String v = ((String) table.getValueAt(r, c));

if (isNumeric(v)) s = String.valueOf(Math.sin(Double.valueOf(v)));

else s = "Ошибка";

}

if (isNumeric(s)) {

if (Math.abs(Double.valueOf(s)) <= Math.pow(10, -7)) s = "0";

}

formulaField.setText(s);

} else formulaField.setText("");

if (selectedCell != null) {

if (row != selectedCell.y && column != selectedCell.x) checkEdited();

}

if (mouseEvent.getClickCount() == 2 && row != -1) {

selectedCell = new Point(column, row);

selectedValue = (String) table.getValueAt(table.getSelectedRow(), table.getSelectedColumn());

}

}

}

});

table.addKeyListener(new KeyListener() {

@Override

public void keyTyped(KeyEvent e) {

}

@Override

public void keyPressed(KeyEvent e) {

}

@Override

public void keyReleased(KeyEvent e) {

int code = e.getKeyCode();

if (code == 10) checkEdited();

}

});

getContentPane().add(tableScroll, BorderLayout.CENTER);

validate();

repaint();

setEdited(false);

}

private void checkEdited() {

int row = table.getSelectedRow();

int column = table.getSelectedColumn();

if (table.getSelectedRow() != -1) {

if (selectedCell == null) return;

String value = (String) table.getValueAt(selectedCell.y, selectedCell.x);

if (selectedValue == null) {

if (value == null) setEdited(true);

}

else if (! selectedValue.equals(value)) setEdited(true);

table.getModel().getValueAt(row, column);

}

}

private void setEdited(boolean edited) {

this.edited = edited;

if (edited) {

if (currentFile != null) Screen.this.setTitle(currentFile.getAbsolutePath() + " \*");

else Screen.this.setTitle("Новый файл \*");

} else {

if (currentFile != null) Screen.this.setTitle(currentFile.getAbsolutePath());

else Screen.this.setTitle("Новый файл");

}

}

private void closeEdit() {

edited = false;

setTitle(TITLE);

saveBtn.setEnabled(false);

table.getTableHeader().setVisible(false);

table.setVisible(false);

formulaField.setText("");

}

private void openEdit() {

saveBtn.setEnabled(true);

table.getTableHeader().setVisible(true);

table.setVisible(true);

}

// return: true - продолжить; false - отмена

private boolean checkIsSaved()

{

if (! edited) return true;

String what = currentFile == null ? "новый файл" : "\"" + currentFile.getAbsolutePath() + "\"";

int result = JOptionPane.showConfirmDialog(null, "Сохранить " + what + "?", TITLE, JOptionPane.YES\_NO\_CANCEL\_OPTION);

if (result == JOptionPane.YES\_OPTION) {

saveCurrentFile();

return true;

} else if (result == JOptionPane.NO\_OPTION) {

return true;

} else return false;

}

private void saveCurrentFile() {

if (currentFile != null) {

writeIntoExcel();

setEdited(false);

} else {

JFileChooser fileChooser = new JFileChooser();

fileChooser.setDialogTitle("Сохранить файл");

FileNameExtensionFilter filter = new FileNameExtensionFilter("Таблица (\*.xls)", "xls");

fileChooser.setAcceptAllFileFilterUsed(false);

fileChooser.addChoosableFileFilter(filter);

int userSelection = fileChooser.showSaveDialog(this);

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

File selectedFile = fileChooser.getSelectedFile();

String path = selectedFile.getPath();

currentFile = new File(path.endsWith(".xls") ? path : path + ".xls");

writeIntoExcel();

setEdited(false);

}

}

}

class ExcelTable {

private ArrayList<ArrayList<String>> rows;

private ArrayList<String> currentRow;

private int columnsNum = 0;

public ExcelTable() {

rows = new ArrayList<ArrayList<String>>();

}

public void newRow() {

currentRow = new ArrayList<String>();

rows.add(currentRow);

}

public void addColumn(String value) {

currentRow.add(value);

if (currentRow.size() > columnsNum) columnsNum = currentRow.size();

}

public String[][] getAsArray() {

String[][] array = new String[getRowsNum()][getColumnsNum()];

int i = 0, j = 0;

for (ArrayList<String> row : rows) {

for (String column : row) {

array[i][j++] = column;

}

i++;

j = 0;

}

return array;

}

public int getColumnsNum() {

return columnsNum;

}

public int getRowsNum() {

return rows.size();

}

private char intToLetter(int i) {

return (char)(i + 64);

}

private String getHeaderCell(int i) {

String cell = "";

if (i == 26) return "Z";

for ( ; i > 0; i /= 26) {

cell += intToLetter(i % 26);

}

return new StringBuilder(cell).reverse().toString();

}

public String[] getHeaders() {

String[] headers = new String[columnsNum];

for (int i = 0; i < columnsNum; i++)

headers[i] = getHeaderCell(i + 1);

return headers;

}

@Override

public String toString() {

return rows.toString();

}

}

}