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 №4

по предмету:

Наладчик технологического оборудования

Выполнил:

студент 4 курса

Группы 4441

Зарипов Ислам Радифович

Казань 2019 г.

**Цель работы:**

Разработка программы позволяющий пользователю создавать, редактировать, сохранять документы формата «DOC».

**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 .doc фала была выбрана библиотека Apache POI, подключение через Maven:

<dependency>

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

<artifactId>poi-ooxml</artifactId>

<version>3.11</version>

</dependency>

Добавлена возможность смены шрифта, размера и цвета текста (см. Рис. 2 – верхняя панель) и пример (см. Рис. 6).

**Скриншоты работы программы на Java:**

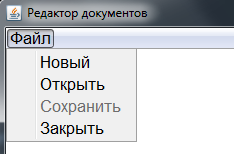
****

Рис. 1. Контекстное меню программы

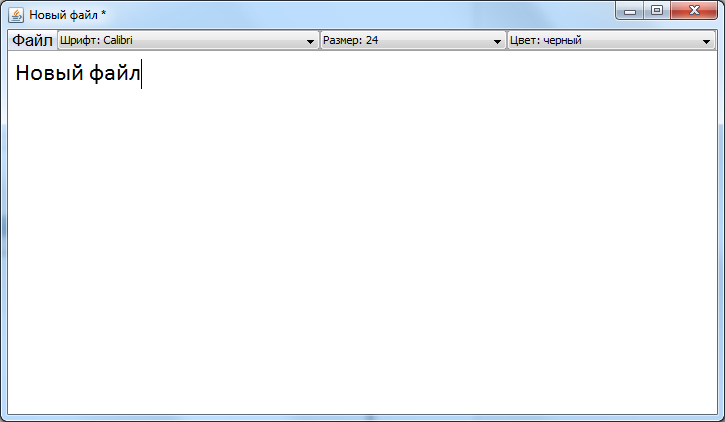


Рис. 2. Окно редактирования нового файла

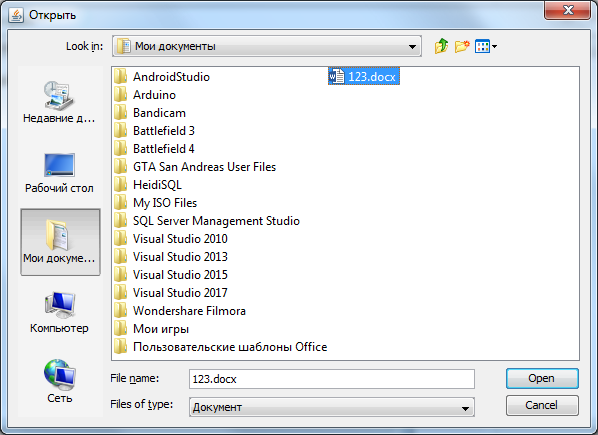


Рис. 3. Окно выбора файла для открытия

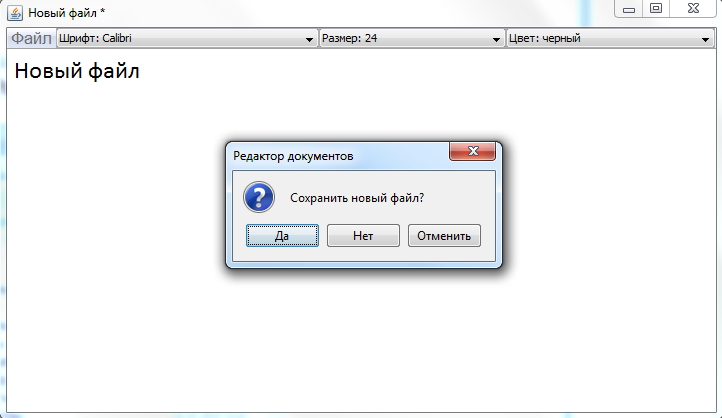


Рис. 4. Окно подтверждения «сохранить или нет»

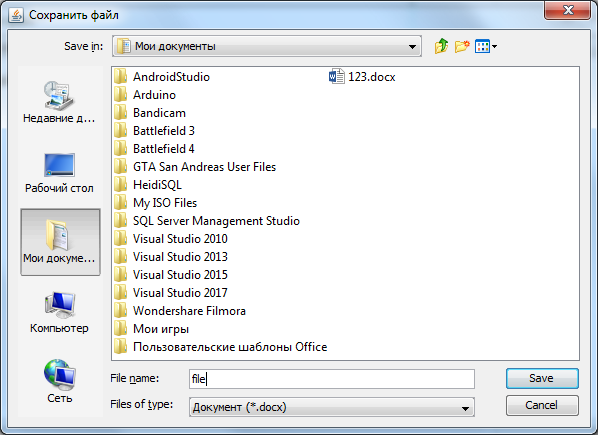


Рис. 5. Окно сохранения нового файла

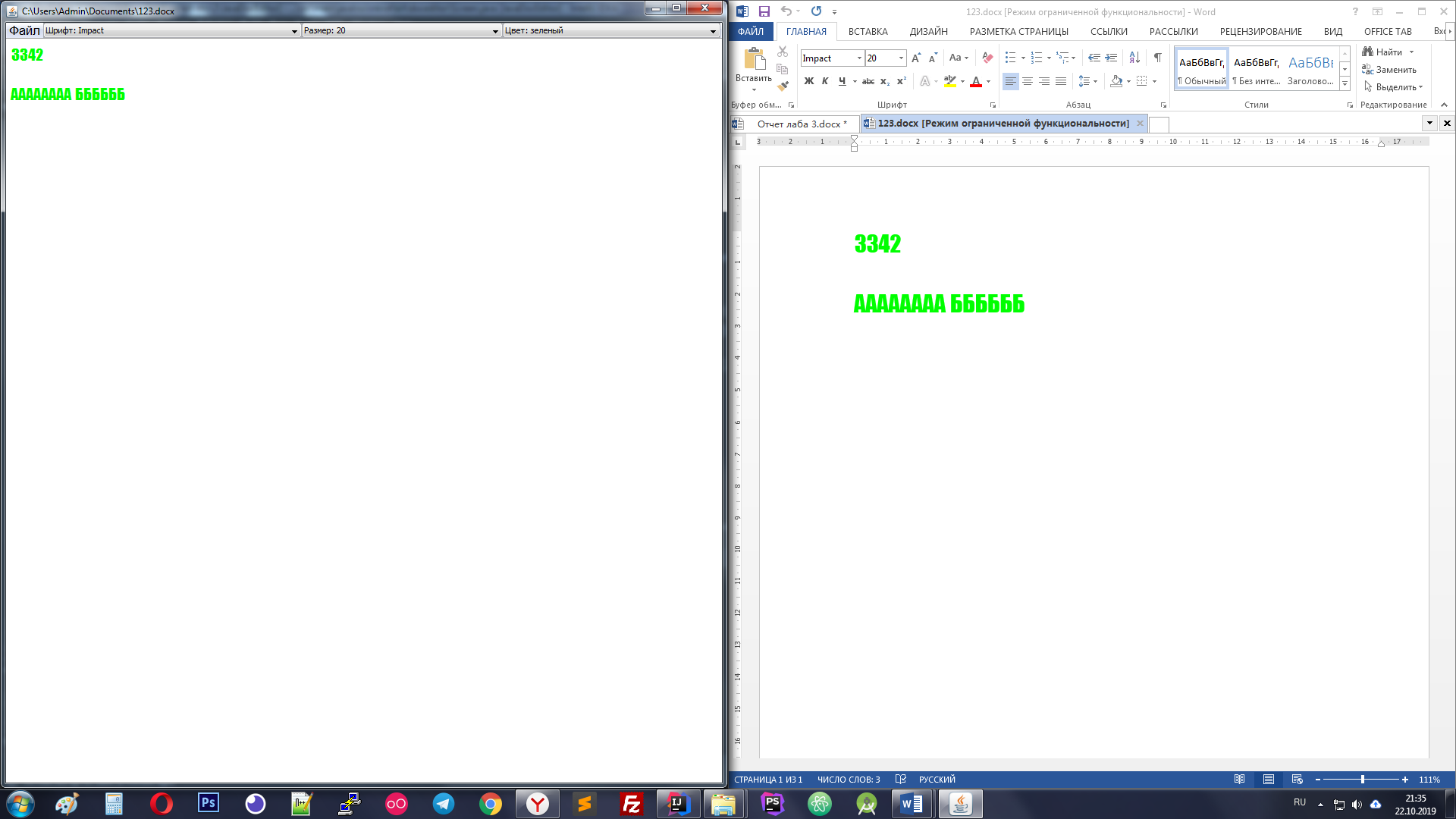


Рис. 6. Пример изменения параметров шрифта в документе

**Вывод:**

В ходе лабораторной работы была разработана программа, позволяющая пользователю создавать, редактировать, сохранять документы формата «DOC».

**Приложение**

**Файл Main.java:**

package ru.inkrot.kit.doceditor;

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.doceditor;

import org.apache.poi.xwpf.usermodel.XWPFDocument;

import org.apache.poi.xwpf.usermodel.XWPFParagraph;

import org.apache.poi.xwpf.usermodel.XWPFRun;

import javax.swing.\*;

import javax.swing.filechooser.FileNameExtensionFilter;

import java.awt.\*;

import java.awt.event.\*;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.List;

public class Screen extends JFrame implements ActionListener, ItemListener {

public final String TITLE = "Редактор документов";

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";

public String loadedFileText = null;

public String loadedFileTextFont = null;

public int loadedFileTextSize = 0;

public Color loadedFileTextColor = null;

private boolean edited = false;

private static File currentFile = null;

private JTextArea textArea;

private JMenuItem saveBtn;

private JComboBox fontCombo, sizeCombo, colorCombo;

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 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);

menuBar.add(fontCombo = new JComboBox(new String[]{"Шрифт: Arial", "Шрифт: Calibri", "Шрифт: Comic Sans MS", "Шрифт: Impact", "Шрифт: Times New Roman"}));

menuBar.add(sizeCombo = new JComboBox(new String[]{"Размер: 8", "Размер: 12", "Размер: 16", "Размер: 20", "Размер: 24", "Размер: 26"}));

menuBar.add(colorCombo = new JComboBox(new String[]{"Цвет: черный", "Цвет: синий", "Цвет: желтый", "Цвет: зеленый", "Цвет: красный"}));

fontCombo.addItemListener(this);

sizeCombo.addItemListener(this);

colorCombo.addItemListener(this);

textArea = new JTextArea();

textArea.setFont(font);

textArea.setBorder(BorderFactory.createCompoundBorder(

textArea.getBorder(),

BorderFactory.createEmptyBorder(5, 5, 5, 5)));

textArea.setBounds(0, 0, 700, 400);

textArea.addKeyListener(new KeyListener() {

@Override

public void keyTyped(KeyEvent e) {}

@Override

public void keyPressed(KeyEvent e) {

}

@Override

public void keyReleased(KeyEvent e) {

if (loadedFileText != null) {

String text = textArea.getText();

if (text == null) text = "";

if (! loadedFileText.equals(text)) setEdited(true);

else setEdited(false);

}

}

});

add(textArea, BorderLayout.CENTER);

setJMenuBar(menuBar);

}

@Override

public void itemStateChanged(ItemEvent e) {

if (e.getStateChange() != ItemEvent.SELECTED) return;

JComboBox c = (JComboBox) e.getSource();

String option = (String) c.getSelectedItem();

if (c.equals(fontCombo)) {

String font = option.substring(7);

textArea.setFont(new Font(font, 0, textArea.getFont().getSize()));

if (! loadedFileTextFont.equals(font)) setEdited(true);

else setEdited(false);

} else if (c.equals(sizeCombo)) {

int size = Integer.parseInt(option.substring(8));

textArea.setFont(new Font(textArea.getFont().getFamily(), 0, size));

if (loadedFileTextSize != size) setEdited(true);

else setEdited(false);

} else if (c.equals(colorCombo)) {

Color color = strToColor(option.substring(6));

textArea.setForeground(color);

if (! loadedFileTextColor.equals(color)) setEdited(true);

else setEdited(false);

}

}

@Override

public void actionPerformed(ActionEvent e) {

String action = e.getActionCommand();

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

if (! checkIsSaved()) return;

currentFile = null;

loadedFileTextFont = "Calibri";

loadedFileTextColor = Color.BLACK;

loadedFileTextSize = 24;

loadedFileText = "";

textArea.setFont(new Font(loadedFileTextFont, 0, loadedFileTextSize));

textArea.setForeground(loadedFileTextColor);

textArea.setText(loadedFileText);

fontCombo.setSelectedItem("Шрифт: " + loadedFileTextFont);

sizeCombo.setSelectedItem("Размер: " + loadedFileTextSize);

colorCombo.setSelectedItem("Цвет: " + colorToStr(loadedFileTextColor));

textArea.setText(loadedFileText);

setEdited(false);

openEdit();

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

if (! checkIsSaved()) return;

JFileChooser chooser = new JFileChooser();

chooser.setDialogTitle("Открыть");

FileNameExtensionFilter filter = new FileNameExtensionFilter("Документ", "doc", "docx");

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;

}

loadWord(chooser.getSelectedFile());

openEdit();

}

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

saveCurrentFile();

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

if (! checkIsSaved()) return;

textArea.setText("");

closeEdit();

}

}

private void setEdited(boolean edited) {

edited = edited || !loadedFileText.equals(textArea.getText());

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);

fontCombo.setVisible(false);

sizeCombo.setVisible(false);

colorCombo.setVisible(false);

saveBtn.setEnabled(false);

textArea.setEnabled(false);

}

private void openEdit() {

fontCombo.setVisible(true);

sizeCombo.setVisible(true);

colorCombo.setVisible(true);

saveBtn.setEnabled(true);

textArea.setEnabled(true);

}

private void writeWord(File file) {

try {

XWPFDocument document = new XWPFDocument();

FileOutputStream out = new FileOutputStream(file);

XWPFParagraph paragraph = document.createParagraph();

XWPFRun run = paragraph.createRun();

loadedFileTextFont = textArea.getFont().getFontName();

loadedFileTextSize = textArea.getFont().getSize();

loadedFileTextColor = textArea.getForeground();

loadedFileText = textArea.getText();

setEdited(false);

run.setFontFamily(loadedFileTextFont);

run.setFontSize(loadedFileTextSize);

Color c = loadedFileTextColor;

run.setColor(String.format("%02X%02X%02X", c.getRed(), c.getGreen(), c.getBlue()));

String[] lines = textArea.getText().split("\n");

for (int i = 0; i < lines.length; i++) {

run.setText(lines[i]);

if (i != lines.length - 1) run.addBreak();

}

document.write(out);

out.close();

} catch (IOException e) {

e.printStackTrace();

}

}

private void loadWord(File file) {

try {

currentFile = file;

FileInputStream fis = new FileInputStream(file);

XWPFDocument doc = null;

try {

doc = new XWPFDocument(fis);

List<XWPFParagraph> paragraphs = doc.getParagraphs();

XWPFParagraph paragraph = paragraphs.get(0);

XWPFRun run = paragraph.getRuns().get(0);

loadedFileTextFont = run.getFontFamily();

if (run.getColor() != null) loadedFileTextColor = Color.decode("#" + run.getColor());

else loadedFileTextColor = Color.BLACK;

loadedFileTextSize = run.getFontSize();

loadedFileText = paragraph.getText();

textArea.setFont(new Font(loadedFileTextFont, 0, loadedFileTextSize));

textArea.setForeground(loadedFileTextColor);

textArea.setText(loadedFileText);

} catch (Exception e) {

loadedFileTextFont = "Calibri";

loadedFileTextColor = Color.BLACK;

loadedFileTextSize = 24;

loadedFileText = "";

setEdited(false);

return;

}

textArea.setFont(new Font(loadedFileTextFont, 0, loadedFileTextSize));

textArea.setForeground(loadedFileTextColor);

textArea.setText(loadedFileText);

fontCombo.setSelectedItem("Шрифт: " + loadedFileTextFont);

sizeCombo.setSelectedItem("Размер: " + loadedFileTextSize);

colorCombo.setSelectedItem("Цвет: " + colorToStr(loadedFileTextColor));

setEdited(false);

fis.close();

} catch (Exception e1) {

e1.printStackTrace();

}

}

private Color strToColor(String str) {

if (str.equals("синий")) return Color.BLUE;

if (str.equals("желтый")) return Color.YELLOW;

if (str.equals("зеленый")) return Color.GREEN;

if (str.equals("красный")) return Color.RED;

return Color.BLACK;

}

private String colorToStr(Color color) {

if (color.equals(Color.BLUE)) return "синий";

if (color.equals(Color.YELLOW)) return "желтый";

if (color.equals(Color.GREEN)) return "зеленый";

if (color.equals(Color.RED)) return "красный";

return "черный";

}

// 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) {

writeWord(currentFile);

setEdited(false);

} else {

JFileChooser fileChooser = new JFileChooser();

fileChooser.setDialogTitle("Сохранить файл");

FileNameExtensionFilter filter = new FileNameExtensionFilter("Документ (\*.docx)", "docx");

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(".docx") ? path : path + ".docx");

writeWord(currentFile);

setEdited(false);

}

}

}

}