Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

К курсовому проектированию

По курсу «Разработка кроссплатформенных приложений»

на тему «Архиватор»

Выполнил:

студент группы 22ВВВ1

Лосяков Г.С.

Приняли:

Юрова О.В.

**ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**Факультет Вычислительной техники

Кафедра “Вычислительная техника”

“УТВЕРЖДАЮ”

Зав. кафедрой ВТ

профессор М. А. Митрохин

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г

**ЗАДАНИЕ**

**на курсовое проектирование по курсу**

  разработка кроссплатформенных приложений

Студенту\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Группа\_\_\_\_\_\_\_\_\_\_\_  
Тема\_проекта Разработка многомодульного приложения на языке Java

Исходные данные (технические требования) на проектирование

Разработать приложение архиватор

Язык программирования Java, среда разработки NetBeans.

Приложение должно обладать графическим интерфейсом и использовать следующие технологии:

1. Java Collections Framework

2. Механизм обработки исключительных ситуаций

3. Java Stream API

4. Java Multithreading

5. Сетевое взаимодействие.

**Объем работы по курсу**

1. Расчетная часть

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Графическая часть

Диаграммы языка UML:

диаграмма вариантов использования;

диаграмма классов;

диаграмма деятельности;

диаграмма развёртывания;

диаграмма последовательности.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Экспериментальная часть

Тестирование многомодульного приложения

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Срок выполнения проекта по разделам

1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_5\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_6\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_7\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_8\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания “\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты проекта “\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Задание получил**      “\_\_\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025г.

**Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Содержание**

Реферат 5

1. [Введение 6](#_Toc91217240)
2. [Постановка задачи 7](#_Toc91217241)
3. [Теоретическая часть задания 8](#_Toc91217242)
4. [Описание алгоритма программы 9](#_Toc91217243)
5. [Описание программы 10](#_Toc91217244)
6. [Тестирование 13](#_Toc91217245)

[Заключение 14](#_Toc91217247)

[Список литературы 15](#_Toc91217248)

[Листинг программы. 16](#_Toc91217249)

**Реферат**

Отчет 37 страниц, 6 рисунков.

АРХИВАТОР, АЛГОРИТМ ХАФФМАНА.

Цель исследования – разработка программы, способной преобразовать файл расширения docx в файл расширения zip, уменьшив его изначальный размер на 20-40 процентов.

В работе рассмотрены правила архивирования, используя алгоритм Хаффмана.

**Введение**

Одна из проблем современной IT-отрасли заключается в стремительно увеличивающемся объеме данных, которые необходимо хранить. Поэтому многие компании ищут способы сэкономить на этом, сократив уменьшив количество данных, которые необходимо хранить. Один из способов заключается в том, чтобы уменьшить размер этих файлов. И главным инструментом для этого является архиватор.

Простейшие архиваторы просто последовательно объединяют (упаковывают) содержимое файлов в архив. Архив должен также содержать информацию об именах и длине оригинальных файлов для их восстановления, поэтому большинство архиваторов также сохраняют метаданные файлов, предоставляемые операционной системой, такие, как время создания и права доступа. Такую функциональность реализует tar — стандартный архиватор систем типа UNIX. При необходимости уменьшения размера к tar-архиву применяют сжатие без потерь программами gzip, bzip2 и т. д.

Целью данной курсовой работы является разработка программы на языке Java, который является широко используемым. Именно с его помощью в данном курсовом проекте реализуется программа архиватор

# Постановка задачи

# 

Требуется разработать программу, которая уменьшает размер изначального файла на 20-40 процентов.

Исходный файл должен быть в формате docx. При старте программы пользователь в меню пользователь должен выбрать тот файл, который он хочет заархивировать. После этого программа должна выполнить необходимую функцию и сохранить сжатый файл в формает zip в ту же папку.

# Теоретическая часть задания

Архиватор — программа, предназначенная для упаковки без потерь одного и более файлов в единый файл-архив или в серию архивов для удобства переноса и/или хранения данных. Распаковка архивов выполняется с помощью того же архиватора либо посредством сторонних совместимых утилит. Большинство современных архиваторов также выполняет сжатие упаковываемых в архив данных. Одним из алгоритмов, на котором основана работа архиватора является алгоритм Хаффмана.

Основная идея заключается в кодировании переменной длины. Мы можем использовать тот факт, что некоторые символы в тексте встречаются чаще, чем другие, чтобы разработать алгоритм, который будет представлять ту же последовательность символов меньшим количеством битов. При кодировании переменной длины мы присваиваем символам переменное количество битов в зависимости от частоты их появления в тексте. В итоге некоторые символы могут занимать всего 1 бит, а другие 2 бита, 3 или больше.

Метод основывается на создании бинарных деревьев. В нем узел может быть либо конечным, либо внутренним. Изначально все узлы считаются листьями (конечными), которые представляют сам символ и его вес. Внутренние узлы содержат вес символа и ссылаются на два узла-наследника. По общему соглашению, бит *«0»* представляет следование по левой ветви, а *«1»* — по правой. В полном дереве *N* листьев и *N-1* внутренних узлов. Рекомендуется, чтобы при построении дерева Хаффмана отбрасывались неиспользуемые символы для получения кодов оптимальной длины.

Мы будем использовать очередь с приоритетами для построения дерева Хаффмана, где узлу с наименьшей частотой будет присвоен высший приоритет. Ниже описаны шаги построения:

1. Создайте узел-лист для каждого символа и добавьте их в очередь с приоритетами.
2. Пока в очереди больше одного листа делаем следующее:  
   * Удалите два узла с наивысшим приоритетом (с самой низкой частотой) из очереди;
   * Создайте новый внутренний узел, где эти два узла будут наследниками, а частота появления будет равна сумме частот этих двух узлов.
   * Добавьте новый узел в очередь приоритетов.
3. Единственный оставшийся узел будет корневым, на этом построение дерева закончится.

**Описание алгоритма**

Для программной реализации алгоритма нам понадобятся 2 кнопки button1 и button2 и элемент FileChooser. В данном элементе необходимо выбрать нужный файл и нажать на одну из кнопок – на кнопку 1, если необходимо сжать файл с расширением docx, и на кнопку 2, если необходимо разархивировать файл с расширением zip.

После нажатия нужной кнопки происходит обращение к классам HuffmanZipCompressor или HuffmanZipDecompressor. Они работают одинаково с той разницей, что первая архивирует файл, а вторая конвертирует архив в исходный файл.

Класс HuffmanZipCompressor сначала считывает zip файл и данные из него. Затем вызывает метод compressWithHuffman, в котором и прописан алгоритм Хаффмана. Реализация заключается в подсчете частоты символов в файле, построения дерева на основе полученных данных, генерации кодов в методе generateCodes и последующей кодировки данных в файле. Затем полученный код из типа string переводится в тип byte. После биты переводятся в байты и полученный результат сохраняется в файл.

# Описание программы

Для написания данной программы использован язык программирования Java. Язык программирования Java - универсальный язык программирования, который завоевал особую популярность у программистов, благодаря возможности создавать кроссплатформенные приложение и другим уникальным особенностям.

Работа программы начинается с меню – пользователю показывается каталог файлов компьютера, 2 кнопки (конвертировать в zip и вернуть в docx), а также полоса, показывающая прогресс выполнения операций.

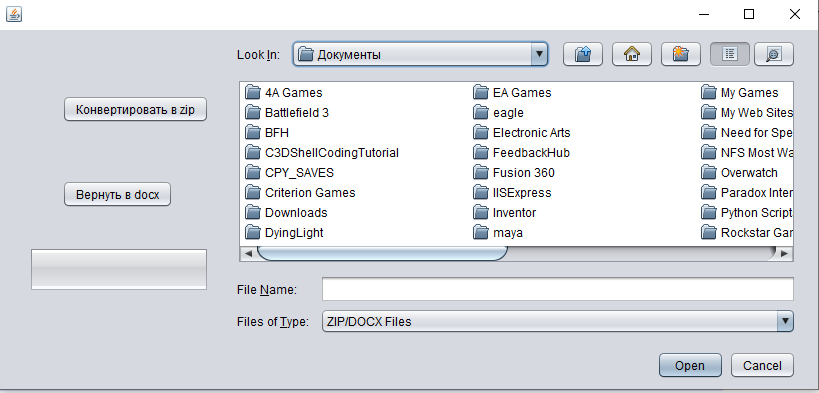


Рисунок 1 – Меню пользователя

В данном меню пользователь может выбрать любой файл, находящийся на его компьютере.

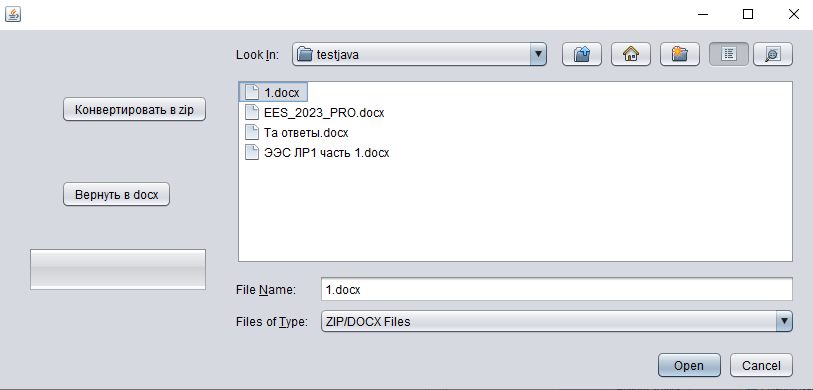


Рисунок 2 – Выбор файла

После выбора файла и нажатия на первую кнопку начнется процесс архивации, по завершении которого пользователь увидит сообщение об успешной архивации его файла.

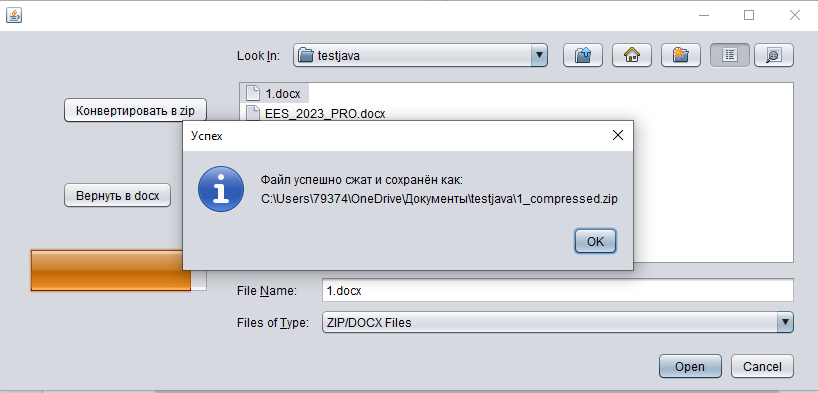


Рисунок 3 – Работа первой кнопки

После выбора файла и нажатия на вторую кнопку начнется процесс реархивации, по завершении которого пользователь увидит сообщение об успешной реархивации файла.

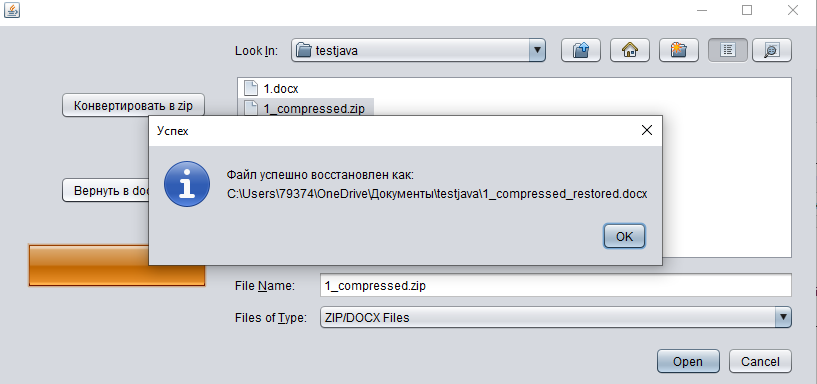


Рисунок 4 – Работа второй кнопки

Данная программа включает в себя 3 класса: NewJFrame, HuffmanZipCompressor, HuffmanZipDecompressor.

# Тестирование

Среда разработки Apace NetBeans 24 предоставляет все средства, необходимые при разработке и отладке многомодульной программы. Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с выбором неправильных файлов и ошибок работы алгоритма.

Ниже продемонстрирован результат тестирования программы при выборе пользователем различных файлов, правильных и неправильных типов.

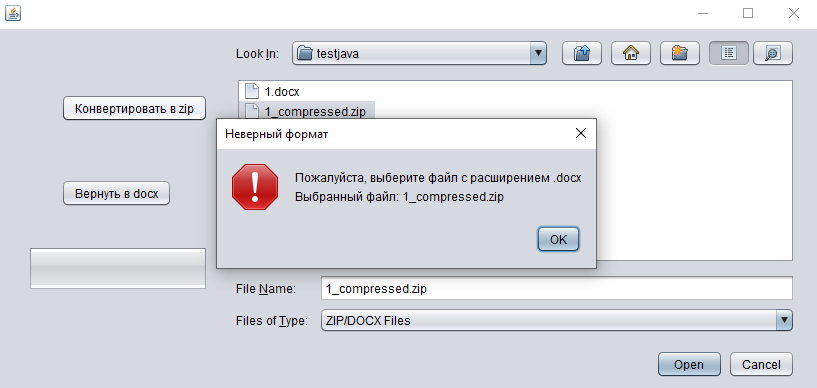


Рисунок 5 – Выбор архива вместо файла с расширением docx

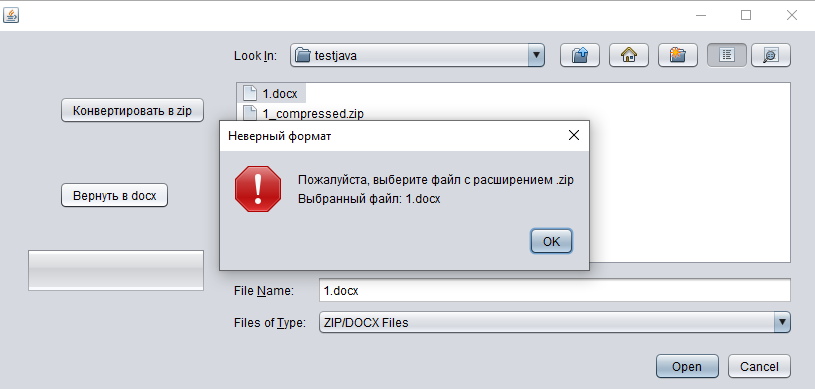


Рисунок 6 – Выбор файла с расширением docx вместо архива

# Заключение

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм Хаффмана для создания архиватора в Apace NetBeans 24.

При выполнении данной курсовой работы были получены навыки разработки программ и освоены приемы по работе с файлами. Углублены знания языка программирования Java.

Программа имеет небольшой, но достаточный для использования функционал возможностей.

# Список литературы

1. Васильев А.Н. «Самоучитель Java с примерами и программами» - Наука и техника, 2016
2. Герберт Шилдт «Java. Полное руководство - Вильямс, 2018
3. Хорстманн К. «Java. Библиотека профессионала», пер. с английского Бернштейн И., Вильямс, 2020
4. Николай Морозов «Справочное пособие по работе с архиваторами», 2024

**Приложение А**

# Листинг программы

package com.mycompany.curs;

import java.io.\*;

import java.nio.file.\*;

import java.util.\*;

import java.util.zip.\*;

import javax.swing.\*;

\*/

import com.mycompany.curs.NewJFrame.HuffmanZipCompressor.ProgressUpdater;

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.FileWriter;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.PrintWriter;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.PriorityQueue;

import java.util.zip.ZipEntry;

import java.util.zip.ZipInputStream;

import java.util.zip.ZipOutputStream;

import javax.swing.JFileChooser;

import javax.swing.JOptionPane;

import javax.swing.filechooser.FileNameExtensionFilter;

import javax.swing.table.DefaultTableModel;

public class NewJFrame extends javax.swing.JFrame {

private File selectedFile;

/\*\*

\* Creates new form NewJFrame

\*/

public NewJFrame() {

initComponents();

jProgressBar1.setMinimum(0);

jProgressBar1.setMaximum(100);

jFileChooser1.setDialogTitle("Выберите файл");

jFileChooser1.setFileFilter(new FileNameExtensionFilter("ZIP/DOCX Files", "zip", "docx"));

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

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

private void initComponents() {

jPanel1 = new javax.swing.JPanel();

jButton1 = new javax.swing.JButton();

jButton2 = new javax.swing.JButton();

jFileChooser1 = new javax.swing.JFileChooser();

jProgressBar1 = new javax.swing.JProgressBar();

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

jButton1.setText("Конвертировать в zip");

jButton1.setToolTipText("");

jButton1.addActionListener(new java.awt.event.ActionListener() {

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

jButton1ActionPerformed(evt);

}

});

jButton2.setText("Вернуть в docx");

jButton2.addActionListener(new java.awt.event.ActionListener() {

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

jButton2ActionPerformed(evt);

}

});

jFileChooser1.addActionListener(new java.awt.event.ActionListener() {

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

jFileChooser1ActionPerformed(evt);

}

});

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(58, 58, 58)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jButton2)

.addComponent(jButton1)))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jProgressBar1, javax.swing.GroupLayout.PREFERRED\_SIZE, 180, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addGap(18, 18, 18)

.addComponent(jFileChooser1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(65, 65, 65)

.addComponent(jButton1)

.addGap(57, 57, 57)

.addComponent(jButton2)

.addGap(39, 39, 39)

.addComponent(jProgressBar1, javax.swing.GroupLayout.PREFERRED\_SIZE, 45, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jFileChooser1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, 359, Short.MAX\_VALUE)

);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

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

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

.addContainerGap()

.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addContainerGap())

);

layout.setVerticalGroup(

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

.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

pack();

}// </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

if (selectedFile == null) {

JOptionPane.showMessageDialog(this,

"Сначала выберите файл в проводнике слева!",

"Ошибка", JOptionPane.ERROR\_MESSAGE);

return;

}

String fileName = selectedFile.getName().toLowerCase();

if (!fileName.endsWith(".docx")) {

JOptionPane.showMessageDialog(this,

"Пожалуйста, выберите файл с расширением .docx\n"

+ "Выбранный файл: " + selectedFile.getName(),

"Неверный формат", JOptionPane.ERROR\_MESSAGE);

return;

}

String inputPath = selectedFile.getAbsolutePath();

String outputPath = inputPath.replace(".docx", "\_compressed.zip");

try {

jProgressBar1.setValue(10);

HuffmanZipCompressor.compressDocxToZip(inputPath, outputPath, this::updateProgress);

JOptionPane.showMessageDialog(this,

"Файл успешно сжат и сохранён как:\n" + outputPath,

"Успех", JOptionPane.INFORMATION\_MESSAGE);

jProgressBar1.setValue(100);

} catch (IOException e) {

JOptionPane.showMessageDialog(this,

"Ошибка при сжатии: " + e.getMessage(),

"Ошибка", JOptionPane.ERROR\_MESSAGE);

jProgressBar1.setValue(0);

}

}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

if (selectedFile == null) {

JOptionPane.showMessageDialog(this,

"Сначала выберите ZIP-файл в проводнике!",

"Ошибка", JOptionPane.ERROR\_MESSAGE);

return;

}

String fileName = selectedFile.getName().toLowerCase();

if (!fileName.endsWith(".zip")) {

JOptionPane.showMessageDialog(this,

"Пожалуйста, выберите файл с расширением .zip\n"

+ "Выбранный файл: " + selectedFile.getName(),

"Неверный формат", JOptionPane.ERROR\_MESSAGE);

return;

}

String inputPath = selectedFile.getAbsolutePath();

String outputPath = inputPath.replace(".zip", "\_restored.docx");

try {

jProgressBar1.setValue(10);

HuffmanZipDecompressor.decompressZipToDocx(inputPath, outputPath, this::updateProgress);

JOptionPane.showMessageDialog(this,

"Файл успешно восстановлен как:\n" + outputPath,

"Успех", JOptionPane.INFORMATION\_MESSAGE);

jProgressBar1.setValue(100);

} catch (IOException | ClassNotFoundException e) {

JOptionPane.showMessageDialog(this,

"Ошибка при распаковке: " + e.getMessage(),

"Ошибка", JOptionPane.ERROR\_MESSAGE);

jProgressBar1.setValue(0);

}

}

private void jFileChooser1ActionPerformed(java.awt.event.ActionEvent evt) {

if (jFileChooser1.getSelectedFile() != null) {

selectedFile = jFileChooser1.getSelectedFile();

}

}

private void updateProgress(int value) {

jProgressBar1.setValue(value);

}

/\*\*

\* @param args the command line arguments

\*/

public static class HuffmanZipCompressor {

public static void compressDocxToZip(String inputPath, String outputPath, ProgressUpdater updater) throws IOException {

byte[] fileData = Files.readAllBytes(Paths.get(inputPath));

updater.update(20);

byte[] compressedData = compressWithHuffman(fileData, updater);

updater.update(70);

try (ZipOutputStream zos = new ZipOutputStream(new FileOutputStream(outputPath))) {

ZipEntry entry = new ZipEntry("compressed.huff");

zos.putNextEntry(entry);

zos.write(compressedData);

zos.closeEntry();

}

updater.update(90);

}

private static byte[] compressWithHuffman(byte[] data, ProgressUpdater updater) throws IOException {

Map<Byte, Integer> frequencyMap = new HashMap<>();

for (byte b : data) {

frequencyMap.put(b, frequencyMap.getOrDefault(b, 0) + 1);

}

updater.update(30);

PriorityQueue<HuffmanNode> pq = new PriorityQueue<>();

for (Map.Entry<Byte, Integer> entry : frequencyMap.entrySet()) {

pq.add(new HuffmanNode(entry.getKey(), entry.getValue()));

}

updater.update(40);

while (pq.size() > 1) {

HuffmanNode left = pq.poll();

HuffmanNode right = pq.poll();

HuffmanNode parent = new HuffmanNode((byte) 0, left.frequency + right.frequency);

parent.left = left;

parent.right = right;

pq.add(parent);

}

updater.update(50);

HuffmanNode root = pq.poll();

Map<Byte, String> huffmanCodes = new HashMap<>();

generateCodes(root, "", huffmanCodes);

updater.update(60);

StringBuilder encodedData = new StringBuilder();

for (byte b : data) {

encodedData.append(huffmanCodes.get(b));

}

updater.update(70);

byte[] compressedBytes = convertBitStringToBytes(encodedData.toString());

updater.update(80);

ByteArrayOutputStream outputStream = new ByteArrayOutputStream();

try (ObjectOutputStream oos = new ObjectOutputStream(outputStream)) {

oos.writeObject(huffmanCodes);

oos.writeInt(data.length);

oos.write(compressedBytes);

}

updater.update(90);

return outputStream.toByteArray();

}

private static void generateCodes(HuffmanNode node, String code, Map<Byte, String> huffmanCodes) {

if (node == null) return;

if (node.left == null && node.right == null) {

huffmanCodes.put(node.data, code);

}

generateCodes(node.left, code + "0", huffmanCodes);

generateCodes(node.right, code + "1", huffmanCodes);

}

private static byte[] convertBitStringToBytes(String bitString) {

int length = (bitString.length() + 7) / 8;

byte[] bytes = new byte[length];

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

if (bitString.charAt(i) == '1') {

bytes[i / 8] |= (1 << (7 - (i % 8)));

}

}

return bytes;

}

static class HuffmanNode implements Comparable<HuffmanNode> {

byte data;

int frequency;

HuffmanNode left, right;

HuffmanNode(byte data, int frequency) {

this.data = data;

this.frequency = frequency;

}

@Override

public int compareTo(HuffmanNode other) {

return this.frequency - other.frequency;

}

}

@FunctionalInterface

interface ProgressUpdater {

void update(int value);

}

}

public static class HuffmanZipDecompressor {

public static void decompressZipToDocx(String zipPath, String docxPath, ProgressUpdater updater)

throws IOException, ClassNotFoundException {

try (ZipInputStream zis = new ZipInputStream(new FileInputStream(zipPath))) {

ZipEntry entry = zis.getNextEntry();

if (!"compressed.huff".equals(entry.getName())) {

throw new IOException("Неверный формат архива");

}

ByteArrayOutputStream baos = new ByteArrayOutputStream();

byte[] buffer = new byte[1024];

int len;

while ((len = zis.read(buffer)) > 0) {

baos.write(buffer, 0, len);

}

updater.update(30);

try (ObjectInputStream ois = new ObjectInputStream(new ByteArrayInputStream(baos.toByteArray()))) {

@SuppressWarnings("unchecked")

Map<Byte, String> huffmanCodes = (Map<Byte, String>) ois.readObject();

int originalSize = ois.readInt();

byte[] compressedData = ois.readAllBytes();

updater.update(60);

byte[] originalData = decompressWithHuffman(compressedData, huffmanCodes, originalSize);

updater.update(80);

Files.write(Paths.get(docxPath), originalData);

updater.update(100);

}

}

}

private static byte[] decompressWithHuffman(byte[] compressedData,

Map<Byte, String> huffmanCodes,

int originalSize) {

Map<String, Byte> reverseCodes = new HashMap<>();

for (Map.Entry<Byte, String> entry : huffmanCodes.entrySet()) {

reverseCodes.put(entry.getValue(), entry.getKey());

}

StringBuilder bitString = new StringBuilder();

for (byte b : compressedData) {

String binaryString = String.format("%8s", Integer.toBinaryString(b & 0xFF)).replace(' ', '0');

bitString.append(binaryString);

}

ByteArrayOutputStream baos = new ByteArrayOutputStream();

StringBuilder currentCode = new StringBuilder();

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

currentCode.append(bitString.charAt(i));

if (reverseCodes.containsKey(currentCode.toString())) {

baos.write(reverseCodes.get(currentCode.toString()));

currentCode.setLength(0);

if (baos.size() == originalSize) {

break;

}

}

}

return baos.toByteArray();

}

}

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

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

public void run() {

new NewJFrame().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JButton jButton2;

private javax.swing.JFileChooser jFileChooser1;

private javax.swing.JPanel jPanel1;

private javax.swing.JProgressBar jProgressBar1;

// End of variables declaration

}