Министерство науки и высшего образования Российской Федерации

Новгородский государственный университет имени Ярослава Мудрого

Защита информации

Лабораторная работа 0

Выполнил:

ст. гр. 0092

Петрова К.Р.

Проверил:

Преподаватель Жгун Т.В.

Дата

Великий Новгород

2023 г

Оглавление

[1. Описание задачи 3](#_Toc131428187)

[2. Теоретические положения 3](#_Toc131428188)

[3. Решение задачи 5](#_Toc131428189)

[4. Вывод 16](#_Toc131428190)

[5. Руководство пользователя 17](#_Toc131428191)

[Приложение 19](#_Toc131428192)

# Описание задачи

В данной лабораторной работе необходимо было реализовать инструмент, обеспечивающий частотный анализ файла графического (\*bmp, 8 бит), и текстового формата (\*txt). Кодировка символов – ASCII.

# Теоретические положения

Информационная энтропия — мера неопределённости некоторой системы, в частности непредсказуемость появления какого-либо символа первичного алфавита. В последнем случае при отсутствии информационных потерь энтропия численно равна количеству информации на символ передаваемого сообщения.

Например, в последовательности букв, составляющих какое-либо предложение на русском языке, разные буквы появляются с разной частотностью, поэтому неопределённость появления для некоторых букв меньше, чем для других. Если же учесть, что некоторые сочетания букв (в этом случае говорят об энтропии n-го порядка, см. ниже) встречаются очень редко, то неопределённость уменьшается еще сильнее.

Изображение выглядит как диаграмма

Автоматически созданное описаниеЭнтропия является количеством, определённым в контексте вероятностной модели для источника данных. Например, кидание монеты имеет энтропию:

Изображение выглядит как диаграмма, схематичный

Автоматически созданное описание1 бит на одно кидание (при условии его независимости), а количество возможных состояний равно: возможных состояния (значения).  
У источника, который генерирует строку, состоящую только из букв «А», энтропия равна 0:

Количество возможных состояний равно: возможное состояние (значение) («А») и от основания логарифма не зависит.

Это тоже информация, которую тоже надо учитывать. Примером запоминающих устройств, в которых используются разряды с энтропией, равной нулю, но с количеством информации, равным одному возможному состоянию, то есть не равным нулю, являются разряды данных записанных в ПЗУ, в которых каждый разряд имеет только одно возможное состояние.

Так, например, опытным путём можно установить, что энтропия английского текста равна 1,5 бит на символ, что будет варьироваться для разных текстов. Степень энтропии источника данных означает среднее число битов на элемент данных, требуемых для их (данных) зашифровки без потери информации, при оптимальном кодировании.

Некоторые биты данных могут не нести информации. Например, структуры данных часто хранят избыточную информацию или имеют идентичные секции независимо от информации в структуре данных.  
Количество энтропии не всегда выражается целым числом битов.

Алфавит может иметь вероятностное распределение, далекое от равномерного. Если исходный алфавит содержит n символов, тогда его можно сравнить с «оптимизированным алфавитом», вероятностное распределение которого — равномерное. Соотношение энтропии исходного и оптимизированного алфавита — это эффективность исходного алфавита, которая может быть выражена в процентах. Эффективность исходного алфавита с n символами может быть также определена как его n-арная энтропия.

Энтропия ограничивает максимально возможное сжатие без потерь (или почти без потерь), которое может быть реализовано при использовании теоретически — типичного набора или, на практике, — кодирования Хаффмана, кодирования Лемпеля — Зива — Велча или арифметического кодирования.

# Решение задачи

Программа была реализована на языке Java с использованием среды разработки IntelliJ IDEA.

Данная программа позволяет проводить частотный анализ над файлами изображений (формата .bmp, .png, .jpg) и текстовыми файлами (формат .txt).

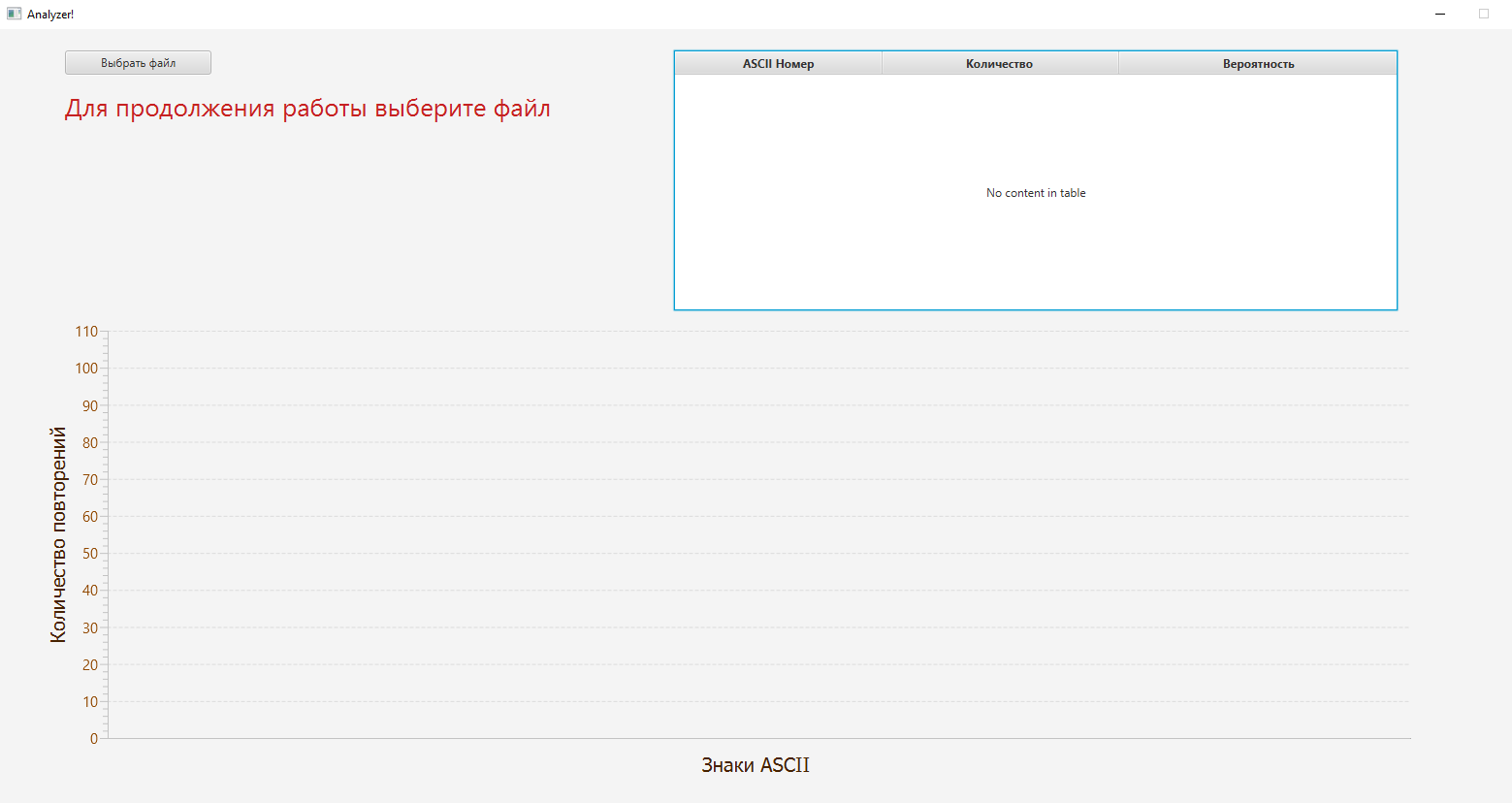


Рисунок 1 Интерфейс программы, обеспечивающий частотный анализ файла

Функционал и способ использования программы описывается в руководстве пользователя (см. пункт 4).

Код программы описан в приложении, либо доступен по ссылке: https://github.com/Varlenn/Probability-analizator

Необходимо было проанализировать распределение символов для файлов различного свойства (картинка с преобладанием одного цвета, картинка в графическом редакторе, фото) и текстов разного свойства (текст учебника, статья в СМИ, литературный текст, переписка в соцсетях).

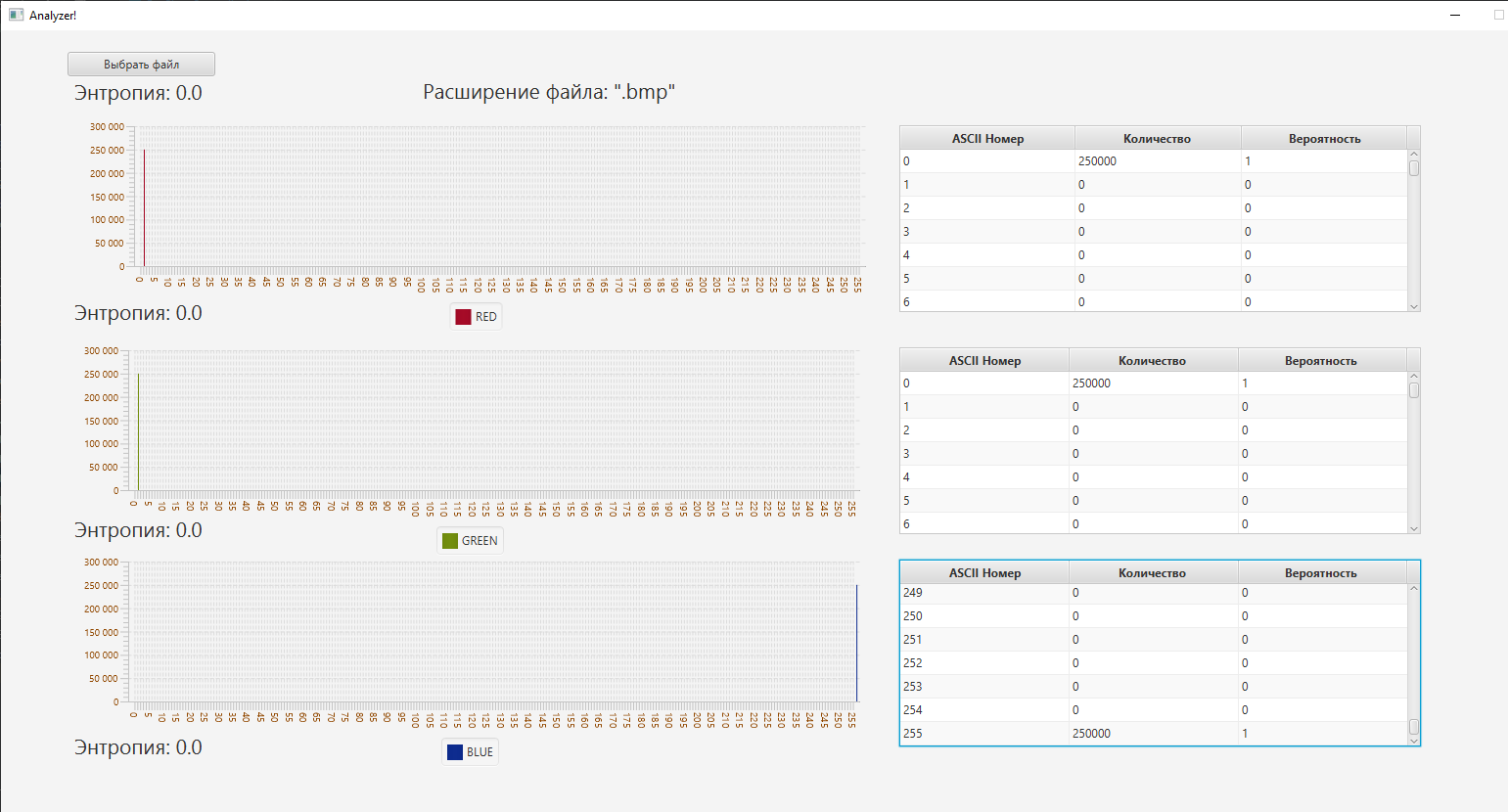
1. Картинка синяя:

Рисунок 2 Картинка, содержащая только синий цвет

Синий цвет задается значениями, изображенными на следующей картинке:

Рисунок 3 Значения каналов для синего цвета в графическом редакторе

Красный и зеленый каналы заданы значениями 0, синий – 255. Это соответствует результатам, выводимым программой, которые можно наблюдать на графике и в таблице.

Рисунок 4 Результат исполнения программы для картинки, содержащей только синий цвет

1. Изображение выглядит как в помещении, фаршированный, постельное белье

   Автоматически созданное описаниеФотография с котятами:

Рисунок 5 Фотография, имеющая преимущественно красные тона

Изображение выглядит как диаграмма

Автоматически созданное описаниеРисунок 6 Результат исполнения программы для фотографии, имеющей преимущественно красные тона

Энтропия общая: 22,63

Энтропия для красного канала: 7,36

Энтропия для зеленого канала: 7,69

Энтропия для синего канала: 7,58

1. Изображение выглядит как прямоугольный

   Автоматически созданное описаниеКартинка зелено-синяя:

Рисунок 7 Картинка, содержащая зеленый и синий цвета

Зеленый цвет задается значениями, изображенными на следующей картинке:

Рисунок 8 Значения каналов для зеленого цвета в графическом редакторе

Изображение выглядит как текст, снимок экрана, в помещении, несколько

Автоматически созданное описаниеДля синего цвета красный и зеленый каналы заданы значениями 0, синий – 255, для зеленого красный и синий каналы заданы значениями 0, зеленый – 255. Таким образом, мы можем наблюдать 5 столбцов, соответствующие значениям этих каналов.

Рисунок 9 Результат исполнения программы для картинки, содержащей зеленый и синий цвета

1. Изображение выглядит как дерево, на открытом воздухе, растение, пруд

   Автоматически созданное описаниеКартинка природы:

Рисунок 10 Фотография, имеющая преимущественно зеленые тона

Изображение выглядит как диаграмма

Автоматически созданное описаниеРисунок 11 Результат исполнения программы для фотографии, имеющей преимущественно зеленые тона

Энтропия общая: 22,25

Энтропия для красного канала: 7,53

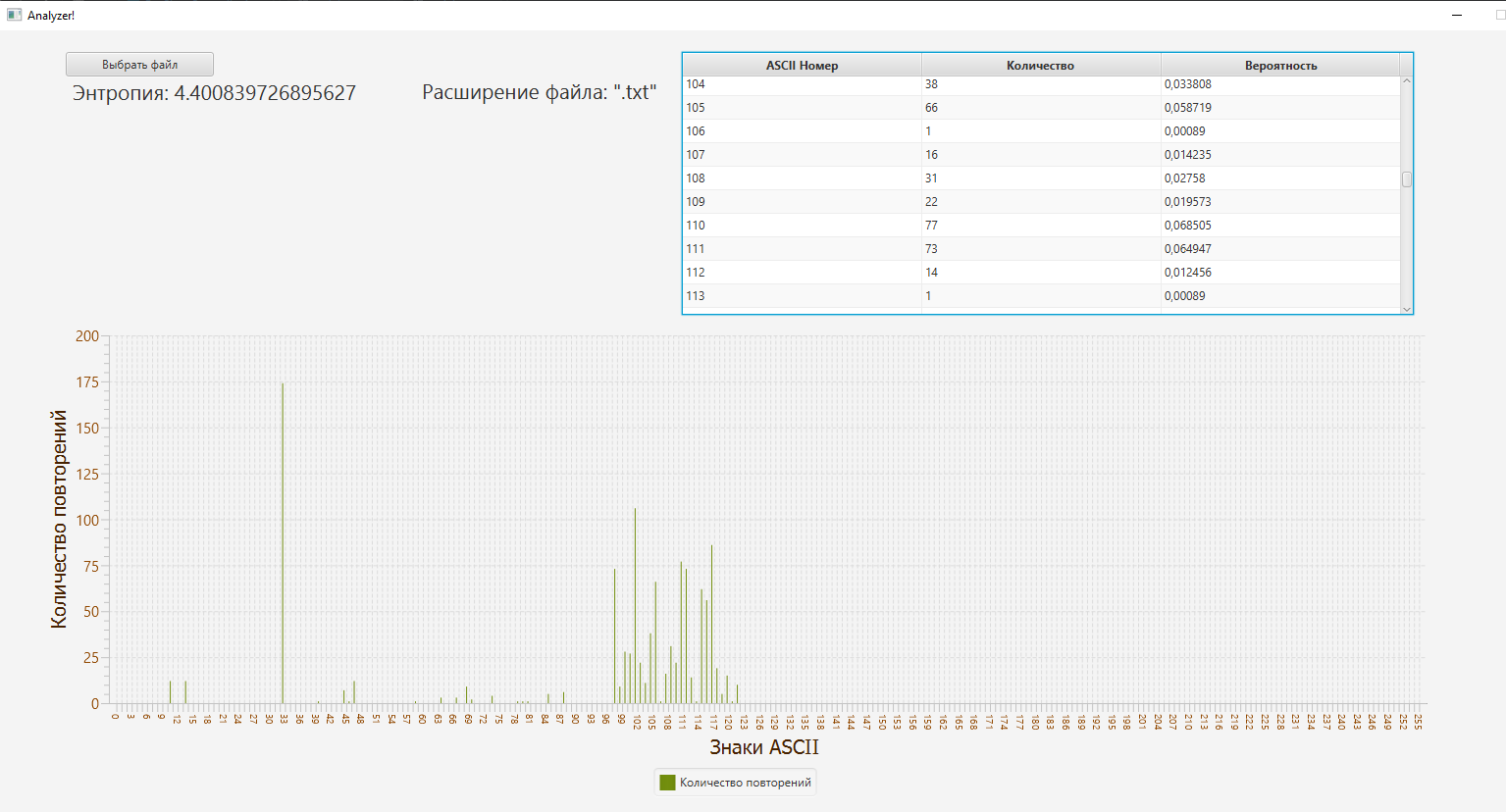
Энтропия для зеленого канала: 7,57

Энтропия для синего канала: 7,15

Далее будут приведены тексты файлов, используемых для частотного анализа.

1. Текст статьи, преимущественно состоящий из латинских символов:

Darknet: selling confidential information?  
Information about leaks of confidential user information to the Darknet is increasingly emerging on the network.  
But what is it, how does it work, and what can you do to save your data?  
What is Darknet?  
The Dark Web is the hidden part of the Internet. Translated from English, DarkNet literally means - shadow  
or the dark web. It is only accessible through certain programs and tools that keep users anonymous and protect their personal information.  
Participants on the Dark Web are called peers. The connection is made between them only in encrypted format  
after mutual approval. The domains and address space on the Dark Web are also their own.  
Even to get into the Darknet, you need special software. One of them is Tor Browser.  
It encrypts the user's traffic while he is in contact with someone on the network. But when he enters or leaves it, it is quite possible to track him.  
While the Dark Web is used for more than just disclosed activities, including  
communication and information exchange, there are often cases of selling confidential information of users.

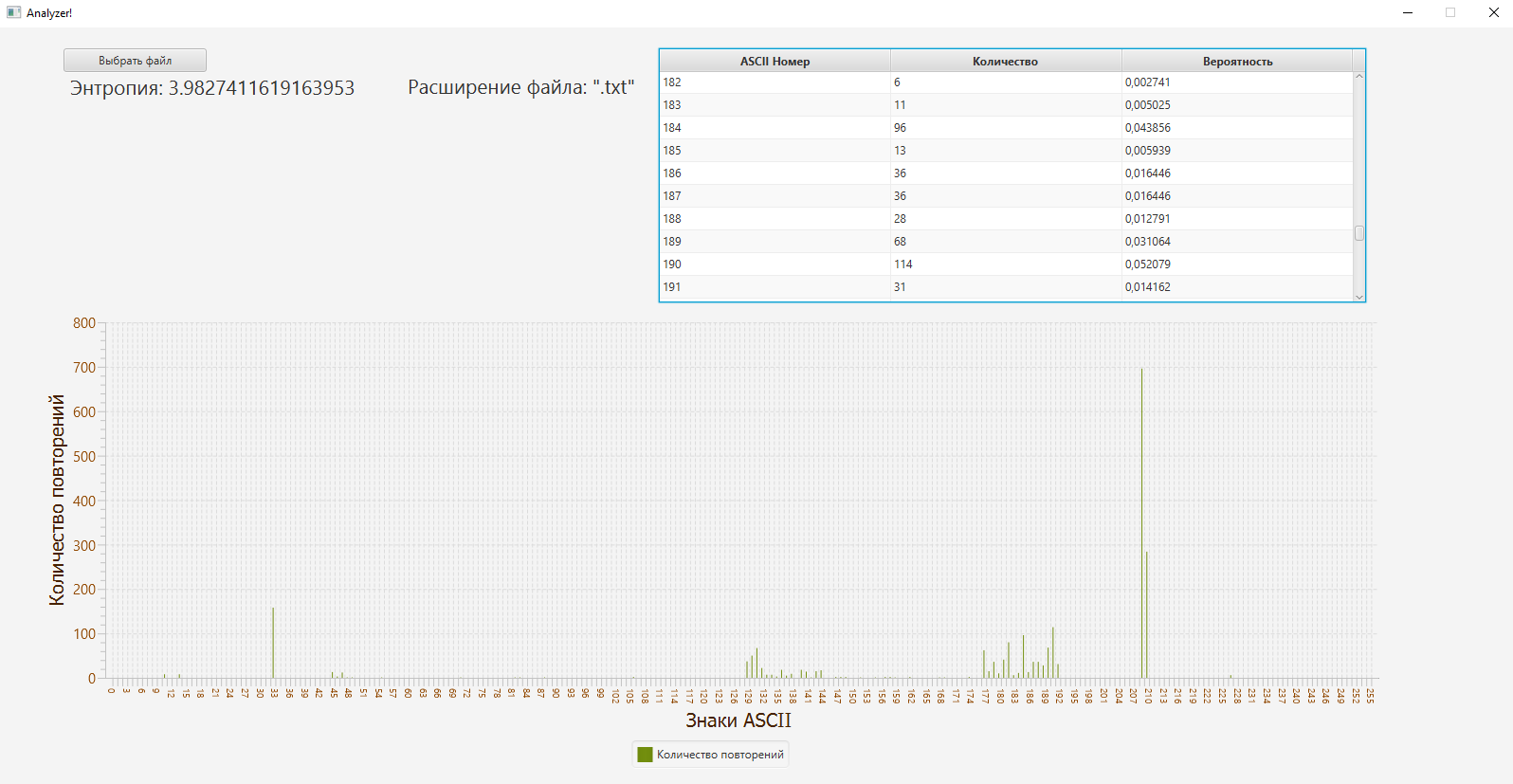
Рисунок 12 Результат исполнения программы для текста, имеющего преимущественно латинские символы 

После исполнения программы можно наблюдать, что большинство символов сосредоточено в левой стороне графика, что соответствует латинским символам.

Энтропия: 4,4

1. Текст статьи, преимущественно состоящий из кириллических символов:

Сквозное шифрование – определение простыми словами  
Входящие и исходящие звонки, аудио и текстовые сообщения, картинки и видео – абсолютно вся информация в чате находится под надежной защитой сквозного шифрования.  
С его помощью данные не смогут попасть в посторонние руки, т.к. доступны исключительно двум или более собеседникам, принимающим участие в чате.  
Это особенно важно в случае передачи данных через открытые или ненадежные сети, такие как интернет или общедоступные сети Wi-Fi.  
Ключи шифрования доступны только участникам беседы, поэтому только они могут зайти в нее и продолжить взаимодействие друг с другом. Да-да, любой может получить  
персональный код шифрования на свою личную переписку с отдельным контактом или группой. Чтобы его найти, достаточно войти в меню “Данные контакта”, где перейти в раздел “Шифрование”.  
Здесь появится ключ в виде 60 цифр или QR-код, необходимый для активации.  
Этот ключ и является кодом безопасности ваших данных, но для просмотра доступна только видимая его часть. Все потому, что абсолютно любое сообщение или аудиозапись  
оснащены своими комплексами замок/ключ. Они не требуют дополнительной настройки, приложение присваивает код автоматически.

Рисунок 13 Результат исполнения программы для текста, имеющего преимущественно кириллические символы

После исполнения программы можно наблюдать, что большинство символов сосредоточено в правой стороне графика, что соответствует кириллическим символам.

Энтропия: 3,98

1. Текст, состоящий из латинских и кириллических символов:

Darknet: selling confidential information?

Information about leaks of confidential user information to the Darknet is increasingly emerging on the network.

But what is it, how does it work, and what can you do to save your data?

What is Darknet?

Сквозное шифрование – определение простыми словами

Входящие и исходящие звонки, аудио и текстовые сообщения, картинки и видео – абсолютно вся информация в чате находится под надежной защитой сквозного шифрования.

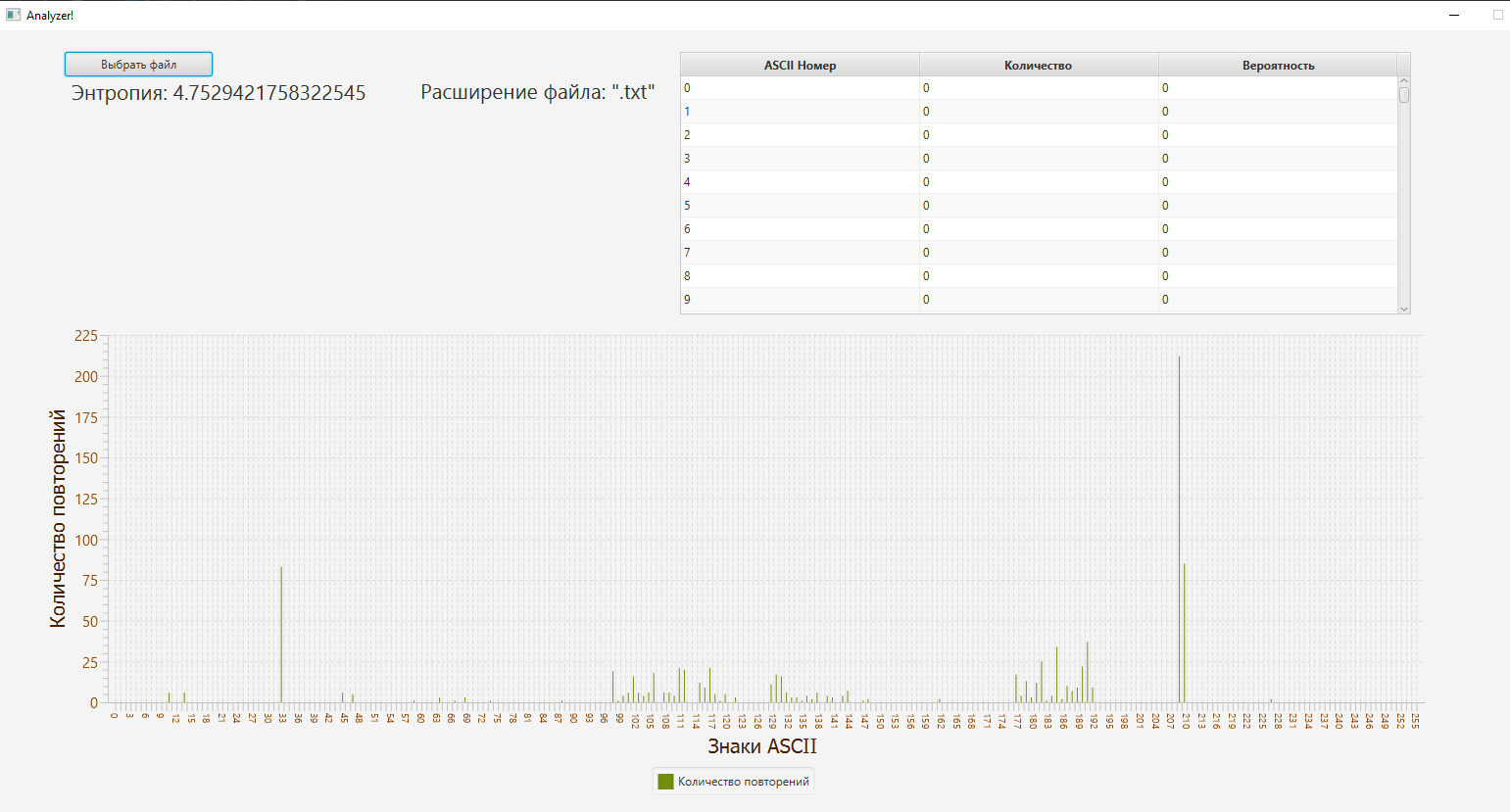
С его помощью данные не смогут попасть в посторонние руки, т.к. доступны исключительно двум или более собеседникам, принимающим участие в чате.

Рисунок 14 Результат исполнения программы для текста, имеющего и латинские и кириллические символы

После исполнения программы видно, что символы распределены по всему графику. Это соответствует номерам латинских и кириллических символов ASCII.

Энтропия: 4,75

1. Текст, содержащий информацию об аппаратных требованиях:

Аппаратные требования

Минимальная конфигурация аппаратных средств для нормального функционирования программы должна содержать:

- Процессор с тактовой частотой не ниже 1.4 ГГц

- Видеоадаптер с 32 МБ видеопамяти, совместимая с OpenGL

- Цветной монитор с разрешением 1920х1080х60 Гц

- Клавиатура

- Манипулятор мышь

- Объём оперативной памяти не менее 1 Гб

- Объём свободного дискового пространства 128 Гб

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 15 Результат исполнения программы для текста, содержащего информацию об аппаратных требованиях

Энтропия: 4,24

# Вывод

В данной лабораторной работе я реализовала инструмент, обеспечивающий частотный анализ файла графического (\*bmp, 8 бит), и текстового формата (\*txt). В результате для каждого файла была вычислена информационная мера. Чем больше беспорядка, тем больше энтропия. Любая система постепенно переходит к своему более вероятному состоянию. В процессе этого в ней увеличивается беспорядок, нарастает хаос, а значит и увеличивается энтропия.

# Руководство пользователя

Изображение выглядит как текст

Автоматически созданное описаниеДля запуска программы необходимо открыть папку «analyzer», в которой должны быть следующие файлы:

Рисунок 1 Необходимые для запуска программы файлы

Для старта программы нужно запустить файл «start».

Изображение выглядит как логотип

Автоматически созданное описаниеДля начала работы с программой необходимо загрузить картинку формата .bmp/.png/.jpg или текстовый файл формата .txt.

Рисунок 2 Выбор файла

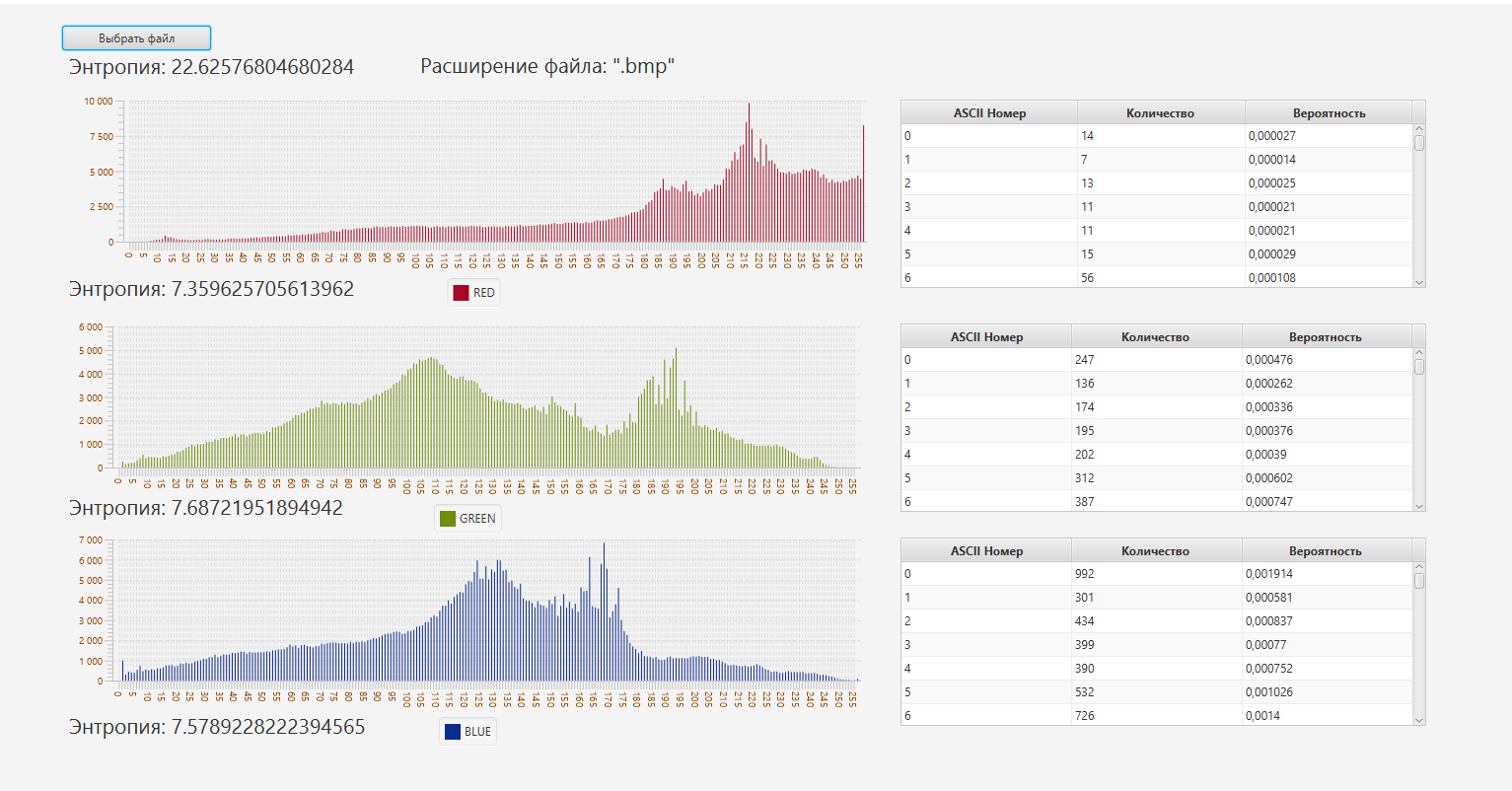
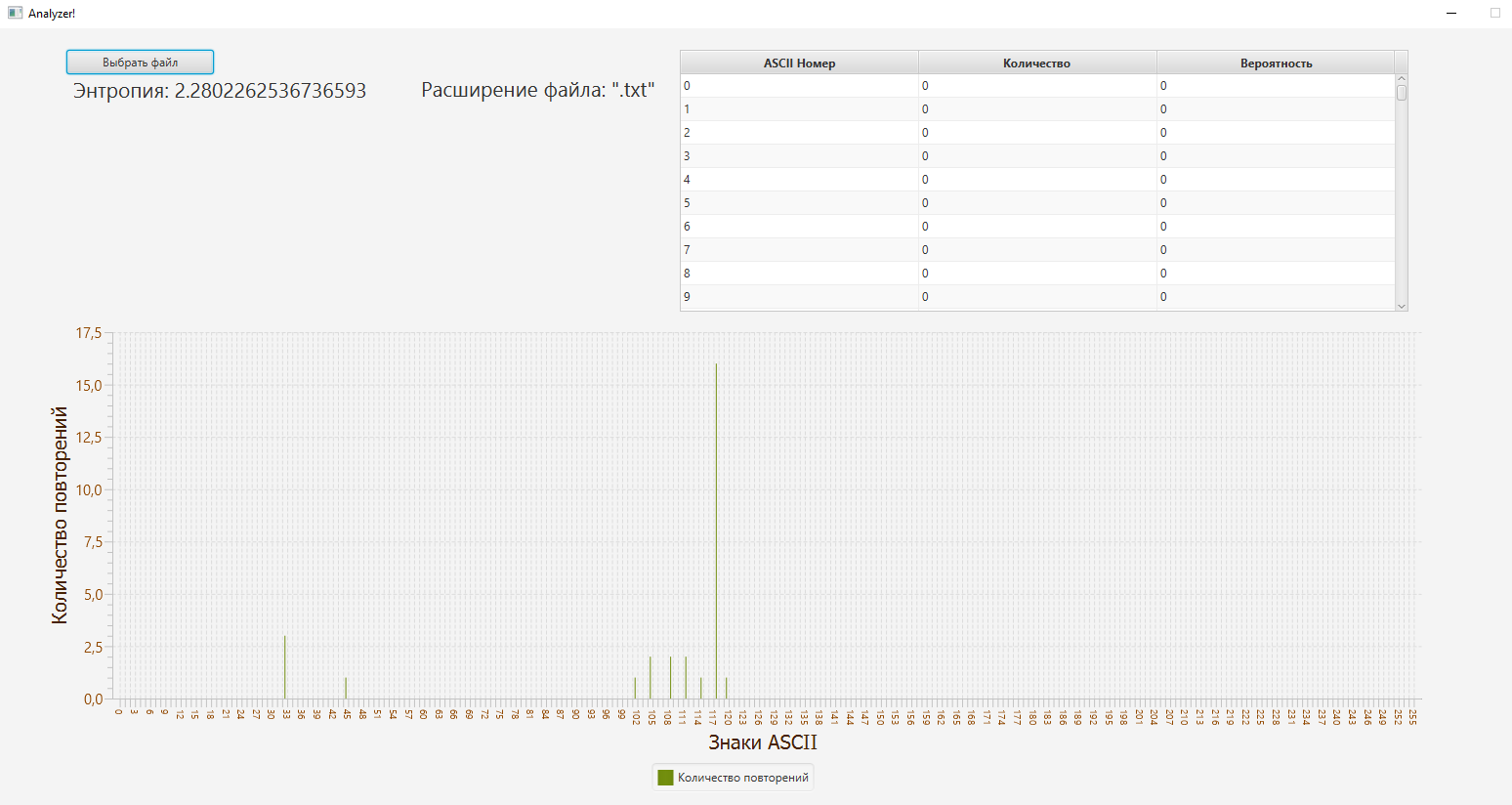
Если была выбрана картинка, то на экране должно появиться следующее:

Рисунок 3 Программа после выбора файла картинки

Выводится информация о расширении файла, общей энтропии, а также энтропии для каждого канала. Каждый канал подписан и имеет соответствующий цвет на графике. Для каждого канала выводится соответствующая таблица, в которой указан ASCII номер элемента, его количество на данной картинке, а также вероятность.

Если был выбран текстовый файл, то на экране должно появиться следующее:

Рисунок 4 Программа после выбора текстового файла

Выводится информация о расширении файла и энтропии. Изображается одна диаграмма, на которой указано количество повторений для каждого знака ASCII. Для загруженного текстового файла выводится таблица, в которой указан ASCII номер элемента, его количество на данной картинке, а также вероятность.

# Приложение

Код программы:

1. Класс BarChartController – реализует функционал программы.

package example.plot;  
  
import javafx.fxml.FXML;  
import javafx.fxml.Initializable;  
import javafx.scene.chart.BarChart;  
import javafx.scene.chart.CategoryAxis;  
import javafx.scene.chart.NumberAxis;  
import javafx.scene.chart.XYChart;  
import javafx.scene.control.\*;  
import javafx.scene.control.cell.PropertyValueFactory;  
import javafx.scene.layout.AnchorPane;  
  
import javax.imageio.\*;  
import javax.swing.\*;  
import java.awt.image.BufferedImage;  
import java.io.File;  
import java.io.IOException;  
import java.io.UnsupportedEncodingException;  
import java.net.URL;  
import java.nio.charset.StandardCharsets;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.sql.SQLException;  
import java.text.DecimalFormat;  
import java.util.HashMap;  
import java.util.Objects;  
import java.util.ResourceBundle;  
import java.util.Scanner;  
  
  
public class BarChartController implements Initializable {  
  
 @FXML  
 private TableColumn<Data, Character> asciiCharColumn;  
  
 @FXML  
 private TableColumn<Data, Character> asciiCharColumn1;  
  
 @FXML  
 private TableColumn<Data, Character> asciiCharColumn2;  
  
 @FXML  
 private TableColumn<Data, Character> asciiCharColumn3;  
  
 @FXML  
 private TableColumn<Data, Integer> asciiNumColumn;  
  
 @FXML  
 private TableColumn<Data, Integer> asciiNumColumn1;  
  
 @FXML  
 private TableColumn<Data, Integer> asciiNumColumn2;  
  
 @FXML  
 private TableColumn<Data, Integer> asciiNumColumn3;  
  
 @FXML  
 private BarChart<?, ?> barChart;  
  
 @FXML  
 private BarChart<?, ?> barChartBlue;  
  
 @FXML  
 private BarChart<?, ?> barChartGreen;  
  
 @FXML  
 private BarChart<?, ?> barChartRed;  
  
 @FXML  
 private Label label;  
  
 @FXML  
 private Label label1;  
  
 @FXML  
 private Label label2;  
  
 @FXML  
 private Label label3;  
  
 @FXML  
 private CategoryAxis labelX;  
  
 @FXML  
 private NumberAxis labelY;  
  
 @FXML  
 private TableColumn<Data, Integer> numColumn;  
  
 @FXML  
 private TableColumn<Data, Integer> numColumn1;  
  
 @FXML  
 private TableColumn<Data, Integer> numColumn2;  
  
 @FXML  
 private TableColumn<Data, Integer> numColumn3;  
  
 @FXML  
 private AnchorPane pane;  
  
 @FXML  
 private Button selectFileButton;  
  
 @FXML  
 private TableColumn<Data, Double> probabilityColumn;  
  
 @FXML  
 private TableColumn<Data, Double> probabilityColumn1;  
  
 @FXML  
 private TableColumn<Data, Double> probabilityColumn2;  
  
 @FXML  
 private TableColumn<Data, Double> probabilityColumn3;  
  
 @FXML  
 private TableView<Data> tableView;  
  
 @FXML  
 private TableView<Data> tableViewBlue;  
  
 @FXML  
 private TableView<Data> tableViewGreen;  
  
 @FXML  
 private TableView<Data> tableViewRed;  
  
 @FXML  
 private Label startLabel;  
  
 @FXML  
 private Label formatLabel;  
  
  
 String fileName;  
 String text;  
 DecimalFormat df;  
 Integer length = 0;  
 Integer lengthRed = 0;  
 Integer lengthGreen = 0;  
 Integer lengthBlue = 0;  
 double entropy = 0;  
 double entropyRed = 0;  
 double entropyGreen = 0;  
 double entropyBlue = 0;  
 HashMap<Integer, Integer> map;  
 private Channel channel;  
 XYChart.Series series;  
 XYChart.Series seriesRed;  
 XYChart.Series seriesGreen;  
 XYChart.Series seriesBlue;  
  
  
 @Override  
 public void initialize(URL url, ResourceBundle resourceBundle) {  
 df = new DecimalFormat("#.######");  
  
 barChartRed.setVisible(false);  
 barChartGreen.setVisible(false);  
 barChartBlue.setVisible(false);  
 tableViewRed.setVisible(false);  
 tableViewGreen.setVisible(false);  
 tableViewBlue.setVisible(false);  
 formatLabel.setVisible(false);  
 label.setVisible(false);  
 label1.setVisible(false);  
 label2.setVisible(false);  
 label3.setVisible(false);  
  
  
 selectFileButton.setOnAction(actionEvent -> {  
 startLabel.setVisible(false);  
 formatLabel.setVisible(true);  
 barChart.setVisible(false);  
 barChartRed.setVisible(false);  
 barChartGreen.setVisible(false);  
 barChartBlue.setVisible(false);  
 tableView.setVisible(false);  
 tableViewRed.setVisible(false);  
 tableViewGreen.setVisible(false);  
 tableViewBlue.setVisible(false);  
 label.setVisible(false);  
 label1.setVisible(false);  
 label2.setVisible(false);  
 label3.setVisible(false);  
  
 map = new HashMap<>();  
 for (int i = 0; i <= 255; i++) {  
 map.put(i, 0);  
 }  
  
 try {  
 fileName = selectFile();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 formatLabel.setText("Расширение файла: \"" + *getFileExtension*(fileName) + "\"");  
 setData();  
 try {  
 showData();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 });  
 }  
  
 private String selectFile() throws IOException {  
 clearData();  
 JFileChooser fileopen = new JFileChooser();  
 File chooserFile = new File((new File("./src/main/resources").getCanonicalPath()));  
 fileopen.setCurrentDirectory(chooserFile);  
 int ret = fileopen.showDialog(null, "Открыть файл");  
 File file = null;  
 if (ret == JFileChooser.*APPROVE\_OPTION*) {  
 file = fileopen.getSelectedFile();  
 }  
 return file.getPath();  
 }  
  
 private static String readUsingScanner(String fileName) throws IOException {  
 Scanner scanner = new Scanner(Paths.*get*(fileName), StandardCharsets.*UTF\_8*.name());  
 String data = scanner.useDelimiter("\\A").next();  
 scanner.close();  
 return data;  
 }  
  
 private static String getFileExtension(String name) {  
 int index = name.indexOf('.');  
 return index == -1 ? null : name.substring(index);  
 }  
  
 private void clearData() {  
 barChart.getData().clear();  
 barChartRed.getData().clear();  
 barChartGreen.getData().clear();  
 barChartBlue.getData().clear();  
 tableView.getItems().clear();  
 tableViewRed.getItems().clear();  
 tableViewGreen.getItems().clear();  
 tableViewBlue.getItems().clear();  
 entropy = 0;  
 entropyRed = 0;  
 entropyGreen = 0;  
 entropyBlue = 0;  
 }  
  
 public void showData() throws SQLException {  
 asciiNumColumn.setCellValueFactory(new PropertyValueFactory<Data, Integer>("asciiNum"));  
 numColumn.setCellValueFactory(new PropertyValueFactory<Data, Integer>("num"));  
 probabilityColumn.setCellValueFactory(new PropertyValueFactory<Data, Double>("probability"));  
  
 asciiNumColumn1.setCellValueFactory(new PropertyValueFactory<Data, Integer>("asciiNum"));  
 numColumn1.setCellValueFactory(new PropertyValueFactory<Data, Integer>("num"));  
 probabilityColumn1.setCellValueFactory(new PropertyValueFactory<Data, Double>("probability"));  
  
 asciiNumColumn2.setCellValueFactory(new PropertyValueFactory<Data, Integer>("asciiNum"));  
 numColumn2.setCellValueFactory(new PropertyValueFactory<Data, Integer>("num"));  
 probabilityColumn2.setCellValueFactory(new PropertyValueFactory<Data, Double>("probability"));  
  
 asciiNumColumn3.setCellValueFactory(new PropertyValueFactory<Data, Integer>("asciiNum"));  
 numColumn3.setCellValueFactory(new PropertyValueFactory<Data, Integer>("num"));  
 probabilityColumn3.setCellValueFactory(new PropertyValueFactory<Data, Double>("probability"));  
  
  
 tableView.setColumnResizePolicy(TableView.*CONSTRAINED\_RESIZE\_POLICY*);  
 tableViewRed.setColumnResizePolicy(TableView.*CONSTRAINED\_RESIZE\_POLICY*);  
 tableViewGreen.setColumnResizePolicy(TableView.*CONSTRAINED\_RESIZE\_POLICY*);  
 tableViewBlue.setColumnResizePolicy(TableView.*CONSTRAINED\_RESIZE\_POLICY*);  
 }  
  
 private void setData() {  
 if (Objects.*equals*(*getFileExtension*(fileName), ".txt")) {  
 barChart.setVisible(true);  
 tableView.setVisible(true);  
 label.setVisible(true);  
  
 try {  
 text = *readUsingScanner*(fileName);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 try {  
 byte[] bytes = Files.*readAllBytes*(Path.*of*(fileName));  
 for (byte b : bytes) {  
 map.merge(Byte.*toUnsignedInt*(b), 1, Integer::*sum*);  
 length++;  
 }  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
  
 series = new XYChart.Series();  
 series.setName("Количество повторений");  
 for (int i = 0; i <= 255; i++) {  
 series.getData().add(new XYChart.Data(String.*valueOf*(i), map.get(i)));  
 }  
  
 barChart.getData().addAll(series);  
 for (int i = 0; i <= 255; i++) {  
 try {  
 tableView.getItems().add(new Data(new String(new byte[]{map.get(i).byteValue()}, "cp866").toCharArray()[0], i, map.get(i), df.format((double) map.get(i) / length)));  
 } catch (UnsupportedEncodingException e) {  
 e.printStackTrace();  
 }  
 if (map.get(i) != 0) {  
 entropy += -((double) map.get(i) / length) \* ((Math.*log*((double) map.get(i) / length)) / Math.*log*(2.0));  
 }  
 }  
 } else {  
 barChartRed.setVisible(true);  
 barChartGreen.setVisible(true);  
 barChartBlue.setVisible(true);  
  
 tableViewRed.setVisible(true);  
 tableViewGreen.setVisible(true);  
 tableViewBlue.setVisible(true);  
  
 label.setVisible(true);  
 label1.setVisible(true);  
 label2.setVisible(true);  
 label3.setVisible(true);  
  
 channel = Channel.*RED*;  
 startAnalysis();  
 seriesRed = new XYChart.Series();  
 seriesRed.setName("RED");  
 for (int i = 0; i <= 255; i++) {  
 seriesRed.getData().add(new XYChart.Data(String.*valueOf*(i), map.get(i)));  
  
 tableViewRed.getItems().add(new Data((char) i, i, map.get(i), df.format((double) map.get(i) / lengthRed)));  
 if (map.get(i) != 0)  
 entropyRed += -((double) map.get(i) / lengthRed) \* ((Math.*log*((double) map.get(i) / lengthRed)) / Math.*log*(2.0));  
 }  
 map.clear();  
  
 channel = Channel.*GREEN*;  
 startAnalysis();  
 seriesGreen = new XYChart.Series();  
 seriesGreen.setName("GREEN");  
 for (int i = 0; i <= 255; i++) {  
 seriesGreen.getData().add(new XYChart.Data(String.*valueOf*(i), map.get(i)));  
  
 tableViewGreen.getItems().add(new Data((char) i, i, map.get(i), df.format((double) map.get(i) / lengthGreen)));  
 if (map.get(i) != 0)  
 entropyGreen += -((double) map.get(i) / lengthGreen) \* ((Math.*log*((double) map.get(i) / lengthGreen)) / Math.*log*(2.0));  
 }  
 map.clear();  
  
 channel = Channel.*BLUE*;  
 startAnalysis();  
 seriesBlue = new XYChart.Series();  
 seriesBlue.setName("BLUE");  
 for (int i = 0; i <= 255; i++) {  
 seriesBlue.getData().add(new XYChart.Data(String.*valueOf*(i), map.get(i)));  
  
 tableViewBlue.getItems().add(new Data((char) i, i, map.get(i), df.format((double) map.get(i) / lengthBlue)));  
 if (map.get(i) != 0)  
 entropyBlue += -((double) map.get(i) / lengthBlue) \* ((Math.*log*((double) map.get(i) / lengthBlue)) / Math.*log*(2.0));  
 }  
 map.clear();  
  
 barChartRed.getData().addAll(seriesRed);  
 barChartGreen.getData().addAll(seriesGreen);  
 barChartBlue.getData().addAll(seriesBlue);  
  
 entropy = entropyRed + entropyGreen + entropyBlue;  
 }  
  
 label.setText("Энтропия: " + entropy);  
 label1.setText("Энтропия: " + entropyRed);  
 label2.setText("Энтропия: " + entropyGreen);  
 label3.setText("Энтропия: " + entropyBlue);  
 }  
  
 private void startAnalysis() {  
 BufferedImage image = null;  
 for (int i = 0; i <= 255; i++) {  
 map.put(i, 0);  
 }  
  
 try {  
 image = ImageIO.*read*(new File(fileName));  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 for (int x = 0; x < image.getWidth(); x++) {  
 for (int y = 0; y < image.getHeight(); y++) {  
 Integer rgb = image.getRGB(x, y);  
  
 if (channel == Channel.*RED*) {  
 map.merge((rgb >> 16) & 0x000000FF, 1, Integer::*sum*);  
 lengthRed += 1;  
 } else if (channel == Channel.*GREEN*) {  
 map.merge((rgb >> 8) & 0x000000FF, 1, Integer::*sum*);  
 lengthGreen += 1;  
 } else if (channel == Channel.*BLUE*) {  
 map.merge((rgb) & 0x000000FF, 1, Integer::*sum*);  
 lengthBlue += 1;  
 }  
 }  
 }  
 }  
  
 private enum Channel {  
 *RED*,  
 *GREEN*,  
 *BLUE* }  
}

1. Класс Data – хранит данные таблицы.

package example.plot;  
  
public class Data {  
 private Character asciiChar;  
 private Integer asciiNum;  
 private Integer num;  
 private String probability;  
  
 public Data(Character asciiChar, Integer asciiNum, Integer num, String probability) {  
 this.asciiChar = asciiChar;  
 this.asciiNum = asciiNum;  
 this.num = num;  
 this.probability = probability;  
 }  
  
 public Character getAsciiChar() {  
 return asciiChar;  
 }  
  
 public void setAsciiChar(Character asciiChar) {  
 this.asciiChar = asciiChar;  
 }  
  
 public Integer getAsciiNum() {  
 return asciiNum;  
 }  
  
 public void setAsciiNum(Integer asciiNum) {  
 this.asciiNum = asciiNum;  
 }  
  
 public Integer getNum() {  
 return num;  
 }  
  
 public void setNum(Integer num) {  
 this.num = num;  
 }  
  
 public String getProbability() {  
 return probability;  
 }  
  
 public void setProbability(String probability) {  
 this.probability = probability;  
 }  
}

1. Класс StartApplication – запускает программу.

package example.plot;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
import java.io.IOException;  
  
public class StartApplication extends Application {  
  
 @Override  
 public void start(Stage stage) throws IOException {  
 FXMLLoader fxmlLoader = new FXMLLoader(StartApplication.class.getResource("/BarChart2.fxml"));  
 Scene scene = new Scene(fxmlLoader.load(), 1600, 800);  
 stage.setTitle("Analyzer!");  
 stage.setResizable(false);  
 stage.setScene(scene);  
 stage.show();  
 }  
  
 public static void main(String[] args) {  
 *launch*();  
 }  
}