Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Утверждаю:

Галкин В.А. "__"_____2021 г.

Курсовая работа
«Локальная безадаптерная сеть» по дисциплине «Сетевые технологии в автоматизированных системах обработки информации и управления»

описание программы
(вид документа)

писчая бумага
(вид носителя)

_____25_____
(количество листов)

ИСПОЛНИТЕЛИ:

студенты группы ИУ5-64Б

Алпеев В.С. _____

Калашникова А.В. _____

"__"_____2021 г.

Москва
2021

Оглавление

1. Введение

Программный продукт написан на языке программирования Java.

Для создания графического интерфейса и взаимодействия с COM-портом использовалась библиотека JSerialComm и стандартные элементы управления. Дополнительные функции, не относящиеся к стандартным, приведены ниже.

2. Класс MainWindow

Класс, описывающий главное окно программы.

2.1. Переменные

private JPanel formBlock;

private JTextField textFieldName;

private JLabel labelName;

private JLabel labelCOMPort;

private JComboBox comboBoxPort;

private JComboBox comboBoxSpeed;

private JLabel labelSpeed;

private JComboBox comboBoxBits;

private JLabel labelBits;

private JLabel labelStopBits;

private JComboBox comboBoxStopBits;

private JLabel labelParity;

private JComboBox comboBoxParity;

private JLabel status;

private JLabel labelStatus;

private JButton buttonConnect;

private JButton buttonOpenChat;

private JButton buttonDisconnect;

private JButton buttonParam;

Цвета:

public static final Color VERY_DARK_GREEN = new Color(0, 102, 0);

public static final Color VERY_DARK_RED = new Color(187, 0, 0);

Окна чата:

private SendText sendText;

private ReceivedText receivedText;

Класс-связь модели и окна:

FrameViewModel viewModel;

2.2. Методы

public void configureUI(); - заполнения полей с выбором.

public void changeUIAfterDisconnect(); - изменение внешнего вида окна после разрыва соединения.

3. Класс ReceivedText

Класс, описывающий окно полученных сообщений.

3.1. Переменные

private javax.swing.JPanel JPanelReceived;

private JTextField userTextField;

private JTextArea messageTextArea;

private JButton closeButton;

private JLabel userLabel;

private JScrollPane scrollPane;

Класс-связь модели и окна:

private FrameViewModel viewModel;

3.2. Методы

public void cleanArea(); - очищает тектовое поле от списка сообщений.

4. Класс SendText

Класс, описывающий окно отправленных сообщений.

4.1. Переменные

private JTextField userTextField;

private JTextArea messageTextArea;

private JButton sendButton;

private JButton closeButton;

private JLabel userLabel;

private javax.swing.JPanel JPanelSend;

private JTextField messageTextField;

private JScrollPane scrollPane;

private final String myName;

4.2. Методы

public void cleanArea(); - очищает тектовое поле от списка сообщений.

5. Класс Frame

Класс, описывающий модель кадра.

5.1. Переменные

private FrameTypes type; - тип кадра.

private byte[] data; - байты данных.

private final byte dataLength; - количество байтов данных.

private int frameSize; - длина кадра.

5.2. Методы

public int getFrameSize(); - получить размер кадра.

public FrameTypes getType(); - получить тип кадра.

public byte[] getFrameToWrite(); - подготовка кадра к отправке.

public String getNameString(); - получить строку имени при обработке кадра LINK.

6. Класс FrameViewModel

Класс, описывающий связующее звено между моделью и окном.

6.1. Переменные

private final SerialPort[] ports; - список портов ПК.

ArrayDeque<Frame> infoFramesQueue = new ArrayDeque<>(); - очередь информационных кадров.

private MainWindowUI mainWindowUI; - интерфейс для взаимодействия с UI.

private ReceivedMessageUI receivedMessageUI; - интерфейс для взаимодействия с UI.

private SerialPort port = null; - работающий порт.

private String userName = «»; - имя пользователя.

private String connectedName = имя собеседника.

private boolean sendLogicalConnect = false; - проверка отправки кадра LINKс вашего устройства.

private byte numOfRet = 0; - счетчик количества ошибок передачи кадра.

private String message = «"; - строка-буфер входящего сообщения.

6.2. Методы

public void setPort(int ind); - метод установки COM-порта.

public String getConnectedName(); - получить имя подключенного пользователя.

public String getUserName(); - получить имя пользователя этого ПК.

public void setUserName(String userName); - установить имя пользователя этого ПК;

public void setMainWindowUIInterface(MainWindowUI uiInterface); - установка интерфейса главного окна.

public void setReceivedMessageUIInterface(ReceivedMessageUI uiInterface); - установка интерфейса окна полученных сообщений.

public boolean setPhysicalConnection(); - установка физического соединения.

public void disconnectAll(); - разрывает логическое соединение.

public void disconnectChanges(); - изменения в оформлении окон после разрыва логического соединения.

public void setLogicalConnection(); - установить логическое соединение.

private void processFrame(byte[] data); - обработка полученного кадра.

public void setComPortParams(int speed, int bits, int stopBits, int parity); - установить параметры COM-порта.

public void getComPortParams(Frame frame) - установить параметры COM-порта, пришедшие в кадре.

public void sendMessage(String text); - отправка текстового сообщения.

public void sendErrorFrame(); - отправка фрейма ошибки.

public void sendSuccessFrame(); - отправка фрейма успеха.

public void write(Frame frame); - передача байтов с использованием соответствующей библиотеки.

7. Класс Hamming

Класс, описывающий модель кодирования кодом Хемминга (15,11).

7.1. Переменные

private byte[] data; - данные.

7.2. Методы

public byte[] getData(); - полученые данных.

public byte[] encode(); - кодирование данных.

public boolean decode(); - декодирование данных.

public boolean isEnd(); - проверка наличия символа конца сообщения.

8. Типы перечисления Enum

Класс, описывающий модель кодирования кодом Хемминга (15,11).

8.1. Типы

public enum Baud; - скорость передачи данных.

public enum DataBits; - биты данных.

public enum FrameTypes; - типы фреймов.

public enum Parity; - проверка на четность.

public enum StopBits; - стоп биты.

8.2. Методы

public int getSpeed(); - получить числовое значение скорости.

public int getBitsNum(); - получить числовое значение количества битов данных.

public byte getFrameCode(); - получить код типа фрейма.

public String getStatus(); - получить варианты проверки на четность.

public String getStopBits(); - получить числовое значение количества стоп битов.

9. Интерфейс MainWindowUI

Интерфейс, используемый для взаимодействия с главным окном.

Методы:

void changeLogicalConnectLabel(); - запрос изменения отображения статуса логического соединения.

void changeComPortParams(int speed, int bits, int stopBits, int parity); - запрос изменения выбранных параметров COM-порта в подменю главного окна.

void uiAfterDisconnect(); - запрос изменения внешнего вида окна после разрыва логического соединения.

String setUserName(); - установка имени пользователя этого ПК.

10. Интерфейс ReceivedMessageUI

Интерфейс, используемый для взаимодействия с главным окном.

Методы:

void addReceivedMessage(String text); - запрос добавления полученного сообщения в список.

11. Листинг

11.1. Класс MainWindow

```
package Views;

import ViewModels.FrameViewModel;
import com.fazecast.jSerialComm.SerialPort;
import enums.Baud;
import enums.DataBits;
import enums.Parity;
import enums.StopBits;

import javax.swing.*;
import java.awt.*;

public class MainWindow extends JFrame{
```

```
private JPanel formBlock;
private JTextField textFieldName;
private JLabel labelName;
private JLabel labelCOMPort;
private JComboBox comboBoxPort;
private JComboBox comboBoxSpeed;
private JLabel labelSpeed;
private JComboBox comboBoxBits;
private JLabel labelBits;
private JLabel labelStopBits;
private JComboBox comboBoxStopBits;
private JLabel labelParity;
private JComboBox comboBoxParity;
private JLabel status;
private JLabel labelStatus;
private JButton buttonConnect;
private JButton buttonOpenChat;
private JButton buttonDisconnect;
private JButton buttonParam;

public static final Color VERY_DARK_GREEN = new Color(0, 102, 0);
public static final Color VERY_DARK_RED = new Color(187, 0, 0);

private SendText sendText;
private ReceivedText receivedText;

FrameViewModel viewModel;

public MainWindow(){
   this.getContentPane().add(formBlock);

   this.viewModel = new FrameViewModel();

   this.viewModel.setMainWindowUIInterface(new MainWindowUI() {
      @Override
      public void changeLogicalConnectLabel() {
         status.setText("Подключено");
         status.setForeground(VERY_DARK_GREEN);

         buttonOpenChat.setEnabled(true);
         buttonDisconnect.setEnabled(true);
         buttonParam.setEnabled(true);

         textFieldName.setEditable(false);
         buttonConnect.setEnabled(false);

         sendText = new SendText(viewModel);
         sendText.setTitle("Исходящие");
         sendText.pack();
         sendText.setSize(500,350);
```

```
            receivedText = new ReceivedText(viewModel);
            receivedText.setTitle("Входящие");
            receivedText.pack();
            receivedText.setSize(500,300);
        }

        @Override
        public void changeComPortParams(int speed, int bits, int stopBits, int parity){
            comboBoxSpeed.setSelectedIndex(speed);
            comboBoxBits.setSelectedIndex(bits);
            comboBoxStopBits.setSelectedIndex(stopBits);
            comboBoxParity.setSelectedIndex(parity);

        }

        @Override
        public void uiAfterDisconnect() {
            changeUIAfterDisconnect();
        }

        @Override
        public String setUserName() {
            return textFieldName.getText();
        }
    });

    this.configureUI();

    //этап установки физического соединения
    this.comboBoxPort.addActionListener(e -> {
        this.comboBoxPort.setEnabled(false);
        int ind = comboBoxPort.getSelectedIndex() - 1;

        if(ind < 0) return;

        this.viewModel.setPort(ind);
        //старт потока для установки физического соединения
        PhysicalConnectionThread physicalConnectionThread = new
PhysicalConnectionThread();
        Thread thread = new Thread(physicalConnectionThread);
        thread.start();
    });

    this.buttonConnect.addActionListener(e -> {
        this.viewModel.setUserName(this.textFieldName.getText());
        this.viewModel.setLogicalConnection();
    });

    this.buttonDisconnect.addActionListener(e -> {
        this.viewModel.disconnectAll();
        changeUIAfterDisconnect();
    });
```

```
        this.buttonOpenChat.addActionListener(e -> {
            this.sendText.setVisible(true);
            this.receivedText.setVisible(true);
        });

        this.buttonParam.addActionListener(e                                    ->
viewModel.setComPortParams(comboBoxSpeed.getSelectedIndex(),
comboBoxBits.getSelectedIndex(),
                comboBoxStopBits.getSelectedIndex(), comboBoxParity.getSelectedIndex()));

    }

    public void configureUI(){

        this.comboBoxPort.addItem("Порт не выбран");
        for (SerialPort port : this.viewModel.getPorts()) {
            this.comboBoxPort.addItem(port.getPortDescription());
        }

        for (Baud param : Baud.values())
            this.comboBoxSpeed.addItem(param.getSpeed());

        for (DataBits param : DataBits.values())
            this.comboBoxBits.addItem(param.getBitsNum());

        for(StopBits param : StopBits.values())
            this.comboBoxStopBits.addItem(param.getStopBits());

        for (Parity param : Parity.values())
            this.comboBoxParity.addItem(param.getStatus());
    }

    public void changeUIAfterDisconnect(){
        this.buttonParam.setEnabled(false);

        this.comboBoxPort.setSelectedIndex(0);
        this.comboBoxPort.setEnabled(true);

        this.status.setText("Отключено");
        this.status.setForeground(VERY_DARK_RED);

        this.textFieldName.setEditable(true);

        this.buttonDisconnect.setEnabled(false);
        this.buttonOpenChat.setEnabled(false);

        this.comboBoxSpeed.setSelectedIndex(0);
        this.comboBoxBits.setSelectedIndex(0);
        this.comboBoxStopBits.setSelectedIndex(0);
        this.comboBoxParity.setSelectedIndex(0);
```

```
            if (this.sendText != null){
                this.sendText.cleanArea();
                this.sendText.setVisible(false);
            }
            if (this.receivedText != null){
                this.receivedText.cleanArea();
                this.receivedText.setVisible(false);
            }
        }

        //Поток физического соединения
        class PhysicalConnectionThread implements Runnable {
            @Override
            public void run() {
                boolean result = viewModel.setPhysicalConnection();
                if (result){
                    buttonConnect.setEnabled(true);
                }
                else{
                    comboBoxPort.setSelectedIndex(0);
                    comboBoxPort.setEnabled(true);
                }
            }
        }

    }
```

11.2. Класс ReceivedText

```
package Views;

import ViewModels.FrameViewModel;
import javax.swing.*;
public class ReceivedText extends JFrame{
    private javax.swing.JPanel JPanelReceived;
    private JTextField userTextField;
    private JTextArea messageTextArea;
    private JButton closeButton;
    private JLabel userLabel;
    private JScrollPane scrollPane;

    private FrameViewModel viewModel;

    public ReceivedText(FrameViewModel viewModel) {
        this.getContentPane().add(JPanelReceived);

        this.userTextField.setText(viewModel.getConnectedName());

        viewModel.setReceivedMessageUIInterface(text -> messageTextArea.append(String.format("<%s>: %s\n", viewModel.getConnectedName(), text)));

        this.closeButton.addActionListener(e -> setVisible(false));
```

```
    }

    public void cleanArea(){
        this.messageTextArea.setText("");
    }

}
```

11.3. Класс SendText

```
package Views;

import ViewModels.FrameViewModel;
import javax.swing.*;

public class SendText extends JFrame{
    private JTextField userTextField;
    private JTextArea messageTextArea;
    private JButton sendButton;
    private JButton closeButton;
    private JLabel userLabel;
    private javax.swing.JPanel JPanelSend;
    private JTextField messageTextField;
    private JScrollPane scrollPane;

    private final String myName;

    public SendText(FrameViewModel viewModel) {
        this.getContentPane().add(JPanelSend);
        this.myName = viewModel.getUserName();

        this.userTextField.setText(viewModel.getConnectedName());

        this.sendButton.addActionListener(e -> {
            if (!messageTextField.getText().equals("")){
                messageTextArea.append(String.format("<%s>:  %s\n",  myName,
messageTextField.getText()));
                viewModel.sendMessage(messageTextField.getText());
                messageTextField.setText("");
            }
        });

        this.closeButton.addActionListener(e -> setVisible(false));
    }

    public void cleanArea(){
        this.messageTextArea.setText("");
    }

}
```

11.4. Класс FrameViewModel

```
package ViewModels;
import Coding.Hamming;
import Models.Frame;
import Views.MainWindowUI;
import Views.ReceivedMessageUI;
import com.fazecast.jSerialComm.SerialPort;
import com.fazecast.jSerialComm.SerialPortDataListener;
import com.fazecast.jSerialComm.SerialPortEvent;
import enums.*;

import java.nio.charset.StandardCharsets;
import java.util.ArrayDeque;

public class FrameViewModel {

    private final SerialPort[] ports;

    //очередь фреймов I
    ArrayDeque<Frame> infoFramesQueue = new ArrayDeque<>();

    //интерфейсы
    private MainWindowUI mainWindowUI;
    private ReceivedMessageUI receivedMessageUI;

    //все последующие поля следует обнулять
    private SerialPort port = null;
    private String userName = "";
    private String connectedName = "";

    //флаги
    private boolean sendLogicalConnect = false;
    private byte numOfRet = 0;

    //строка буфер
    private String message = "";

    public FrameViewModel(){
        ports = SerialPort.getCommPorts();
    }

    public void setPort(int ind){
        this.port = ports[ind];

        //открытие COM-порта и установка сигнала DTR
        port.openPort();
        port.setDTR();
    }

    public SerialPort[] getPorts(){
        return this.ports;
```

```
    }

    public String getConnectedName() { return this.connectedName; }

    public String getUserName() { return this.userName; }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public void setMainWindowUIInterface(MainWindowUI uiInterface){
this.mainWindowUI = uiInterface; }
    public void setReceivedMessageUIInterface(ReceivedMessageUI uiInterface){
this.receivedMessageUI = uiInterface; }

    public boolean setPhysicalConnection(){
        long end = System.currentTimeMillis() + 5000;
        while (System.currentTimeMillis() < end){
            if (port.getDSR()){
                port.addDataListener(new SerialPortDataListener() {
                    @Override
                    public int getListeningEvents() {
                        return SerialPort.LISTENING_EVENT_DATA_AVAILABLE;
                    }

                    @Override
                    public void serialEvent(SerialPortEvent serialPortEvent) {
                        if (serialPortEvent.getEventType() !=
SerialPort.LISTENING_EVENT_DATA_AVAILABLE)
                            return;
                        byte[] newData = new byte[port.bytesAvailable()];
                        port.readBytes(newData, newData.length);
                        processFrame(newData);
                    }
                });
                return true;
            }
        }
        port.closePort();
        return false;
    }

    public void disconnectAll(){
        Frame disconnectFrame = new Frame(FrameTypes.UNLINK);
        write(disconnectFrame);
        disconnectChanges();
    }

    public void disconnectChanges(){
        this.sendLogicalConnect = false;
        this.userName = "";
        this.connectedName = "";
```

```
            this.port.removeDataListener();
            this.port.setComPortParameters(Baud.A.getSpeed(),          DataBits.A.getBitsNum(),
SerialPort.ONE_STOP_BIT, SerialPort.NO_PARITY);
            this.port.clearDTR();
            this.port.closePort();
            this.port = null;
        }

        public void setLogicalConnection(){
            Frame connectionFrame = new Frame(FrameTypes.LINK, this.userName);
            write(connectionFrame);
            this.sendLogicalConnect = true;
        }

        private void processFrame(byte[] data){
            Frame frame = new Frame(data);
            switch(frame.getType()){
                case LINK:
                    System.out.println("LINK");
                    this.connectedName = frame.getNameString();
                    this.userName = mainWindowUI.setUserName();
                    mainWindowUI.changeLogicalConnectLabel();
                    if (!this.sendLogicalConnect){
                        Frame connectionFrame = new Frame(FrameTypes.LINK, this.userName);
                        write(connectionFrame);
                    }
                    break;
                case ACK:
                    System.out.println("ACK");
                    this.numOfRet = 0;
                    infoFramesQueue.pollFirst();
                    frame = infoFramesQueue.peekFirst();
                    if (frame != null) write(frame);
                    break;
                case PRM:
                    System.out.println("PRM");
                    getComPortParams(frame);
                    break;
                case UNLINK:
                    System.out.println("UNLINK");
                    disconnectChanges();
                    mainWindowUI.uiAfterDisconnect();
                    break;
                case I:
                    System.out.println("I");
                    Hamming inputFrame = new Hamming(frame.getData());
                    if (!inputFrame.decode()){
                        sendErrorFrame();
                        break;
                    }
                    message += new String(inputFrame.getData(), StandardCharsets.UTF_16);
                    if (inputFrame.isEnd()){
```

```
                    receivedMessageUI.addReceivedMessage(message.substring(0,
message.length() - 1));
                    message = "";
                }
                sendSuccessFrame();
                break;
            case RET:
                System.out.println("RET");
                this.numOfRet += 1;
                if (numOfRet < 15){
                    frame = infoFramesQueue.peekFirst();
                    if (frame != null) write(frame);
                }
                else {
                    System.out.println("UNLINK");
                    this.sendLogicalConnect = false;
                    this.userName = "";
                    this.connectedName = "";
                    this.port.removeDataListener();
                    this.port.clearDTR();
                    this.port.setComPortParameters(Baud.A.getSpeed(),
DataBits.A.getBitsNum(), SerialPort.ONE_STOP_BIT, SerialPort.NO_PARITY);
                    this.port.closePort();
                    this.port = null;
                    mainWindowUI.uiAfterDisconnect();
                }
                break;
            default:
                break;
        }
    }

    public void setComPortParams(int speed, int bits, int stopBits, int parity){
        int params = 0;
        params = (params | speed) << 2;
        params = (params | bits) << 2;
        params = (params | stopBits) << 1;
        params = params | parity;

        port.setComPortParameters(Baud.values()[speed].getSpeed(),
DataBits.values()[bits].getBitsNum(), stopBits, parity);

        Frame frame = new Frame(FrameTypes.PRM, (byte)params);
        write(frame);
    }

    public void getComPortParams(Frame frame){
        int params = frame.getData()[0];

        int parity = params & 1;
        int stopBits = (params & 6) >> 1;
        int bits = (params & 24) >> 3;
```

```
            int speed = (params & 224) >> 5;

            port.setComPortParameters(Baud.values()[speed].getSpeed(),
                    DataBits.values()[bits].getBitsNum(), stopBits, parity);
            mainWindowUI.changeComPortParams(speed, bits, stopBits, parity);

        }

        public void sendMessage(String text) {
            text += (char)3;
            byte[] data = text.getBytes(StandardCharsets.UTF_16);
            byte[] subData;
            Frame frame;
            for (int i = 0; i < data.length;) {
                subData = new byte[Math.min(92, data.length - i)];
                System.arraycopy(data, i, subData, 0, Math.min(92, data.length - i));
                frame = new Frame(FrameTypes.I, subData);
                infoFramesQueue.add(frame);
                i += 92;
            }
            frame = infoFramesQueue.peekFirst();
            if (frame != null) write(frame);
        }

        public void sendErrorFrame(){
            Frame frame = new Frame(FrameTypes.RET);
            write(frame);
        }

        public void sendSuccessFrame(){
            Frame frame = new Frame(FrameTypes.ACK);
            write(frame);
        }

        public void write(Frame frame) {
            if (port.getDSR()){
                port.writeBytes(frame.getFrameToWrite(), frame.getFrameSize());
            }
            else{
                disconnectChanges();
                mainWindowUI.uiAfterDisconnect();
            }
        }

    }
```

11.5. Frame

```
    package Models;
    import Coding.Hamming;
    import enums.FrameTypes;

    import java.nio.charset.StandardCharsets;
```

```
import java.util.Arrays;

public class Frame {

    private FrameTypes type;
    private byte[] data;
    private final byte dataLength;
    private int frameSize;

    public Frame(FrameTypes type){
        this.type = type;
        this.dataLength = 0;
    }

    public Frame(FrameTypes type, byte params){
        this.type = type;
        this.data = new byte[] {params};
        this.dataLength = 1;
        this.frameSize = 7;
    }

    public Frame(FrameTypes type, String data){
        this.type = type;
        if (!data.equals("")) {
            //имя не должно быть более 127 символов
            this.data = data.getBytes(StandardCharsets.UTF_16);
            this.dataLength = (byte)this.data.length;
            this.frameSize = this.dataLength + 6;
        }
        else {
            this.dataLength = 0;
            this.frameSize = 5;
        }
    }

    public Frame(FrameTypes type, byte[] data){
        this.type = type;
        Hamming hamming = new Hamming(data);
        this.data = hamming.encode();
        this.dataLength = (byte)this.data.length;
        this.frameSize = 6 + this.dataLength;
    }

    public Frame(byte[] data){
        for (FrameTypes enumEl : FrameTypes.values()){
            if (enumEl.getFrameCode() == data[3]) {
                this.type = enumEl;
                break;
            }
        }
        this.frameSize = 5;
        if (data[4] != -1){
```

```
            this.dataLength = data[4];
            this.frameSize += this.dataLength + 1;
            this.data = Arrays.copyOfRange(data, 5, 5 + dataLength);
        }
        else{
            this.dataLength = 0;
        }
    }

    public int getFrameSize(){
        return this.frameSize;
    }

    public FrameTypes getType(){
        return this.type;
    }

    public byte[] getData() {
        return data;
    }

    public byte[] getFrameToWrite(){
        byte[] frame;

        if (this.dataLength == 0){
            frame = new byte[5];
            this.frameSize = 5;
        }
        else{
            frame = new byte[6 + dataLength];
            this.frameSize = 6 + dataLength;
            frame[4] = dataLength;
        }

        frame[0] = 0;
        frame[frame.length - 1] = -1;
        frame[1] = -128;
        frame[2] = -128;
        frame[3] = type.getFrameCode();

        for(int i = 0; i < dataLength; i++){
            frame[i+5] = data[i];
        }

        return  frame;
    }

    public String getNameString(){
        if (this.dataLength > 0) return new String(this.data, StandardCharsets.UTF_16);
        else return "";
    }
```

```
}
```

11.6. Класс Hamming

```
package Coding;
import java.util.ArrayDeque;

public class Hamming {

   private byte[] data;

   public Hamming(byte[] data){
      this.data = data;
   }

   public byte[] getData() {
      return data;
   }

   public byte[] encode() {

      //проверочные биты
      byte first = 0;
      byte second = 0;
      byte fourth = 0;
      byte eighth = 0;

      ArrayDeque<Byte> que = new ArrayDeque<>();

      byte index = 0;
      byte bit;

      for (byte datum : data) {
         for (byte j = 0; j < 8; j++) {
            bit = (byte) ((datum & (1 << (7 - j))) >> (7 - j));
            switch (index) {
               case 0 -> {
                  first ^= bit;
                  second ^= bit;
               }
               case 1 -> {
                  first ^= bit;
                  fourth ^= bit;
               }
               case 2 -> {
                  second ^= bit;
                  fourth ^= bit;
               }
               case 3 -> {
                  first ^= bit;
                  second ^= bit;
                  fourth ^= bit;
```

```
            }
            case 4 -> {
                first ^= bit;
                eighth ^= bit;
            }
            case 5 -> {
                second ^= bit;
                eighth ^= bit;
            }
            case 6 -> {
                first ^= bit;
                second ^= bit;
                eighth ^= bit;
            }
            case 7 -> {
                fourth ^= bit;
                eighth ^= bit;
            }
            case 8 -> {
                first ^= bit;
                fourth ^= bit;
                eighth ^= bit;
            }
            case 9 -> {
                second ^= bit;
                fourth ^= bit;
                eighth ^= bit;
            }
            case 10 -> {
                first ^= bit;
                second ^= bit;
                fourth ^= bit;
                eighth ^= bit;
            }
        }
        if (index != 10) index++;
        else {
            index = 0;
            que.add(first);
            first = 0;
            que.add(second);
            second = 0;
            que.add(fourth);
            fourth = 0;
            que.add(eighth);
            eighth = 0;
        }
    }
}
que.add(first);
que.add(second);
que.add(fourth);
```

```
        que.add(eighth);

        byte[] result = new byte[(int)Math.ceil(15.0*(double)data.length/11.0)];
        index = 0;
        int k = 0;
        int n = 0;

        for (int i = 0; i < result.length; i++){
            for (int j = 0; j < 8; j++){
                switch (index) {
                    case 0, 1, 3, 7 -> result[i] |= ((que.peekFirst() != null) ? que.pollFirst() : 0) <<
(7 - j);
                    default -> {
                        if (k < data.length) result[i] |= ((data[k] & (1 << (7 - n))) >> (7 - n)) << (7 -
j);
                        else return result;
                        if (n != 7) n++;
                        else {
                            n = 0;
                            k++;
                        }
                    }
                }
                if (index != 14) index++;
                else index = 0;
            }
        }
        return result;
    }

    public boolean isEnd() {
        return (this.data[this.data.length - 1] == 3) & (this.data[this.data.length - 2] == 0);
    }

    public boolean decode(){
        int index = 1;
        int first = 0;
        int second = 0;
        int third = 0;
        int fourth = 0;

        byte[] result = new byte[11 * data.length / 15];
        byte resultByte = 0;
        int numByteResult = 0;
        int indResult = 0;

        for (byte datum : data) {
            for (int j = 0; j < 8; j++) {
                if ((index & 1) == 1) first ^= 1;
                if ((index & 2) == 2) second ^= 1;
                if ((index & 4) == 4) third ^= 1;
                if ((index & 8) == 8) fourth ^= 1;
```

```
                if ((index != 1) & (index != 2) & (index != 4) & (index != 8)) {
                    resultByte |= ((datum & (1 << (7 - j))) >> (7 - j)) << (7 - indResult);
                    if (indResult != 7) indResult++;
                    else {
                        indResult = 0;
                        result[numByteResult] = resultByte;
                        numByteResult++;
                        resultByte = 0;
                    }
                }

                if (index != 15) index++;
                else {
                    index = 1;
                    if ((first + second + third + fourth) != 0) {
                        return false;
                    }

                    first = 0;
                    second = 0;
                    third = 0;
                    fourth = 0;
                }
            }
        }
        this.data = result;
        return true;
    }

}
```

11.7. Enum Baud

```
package enums;

public enum Baud {

    //доступно 8 значений

    A (9600),
    B (4800);

    private final int speed;

    Baud(int speed) { this.speed = speed; }

    public int getSpeed() {
        return speed;
    }
}
```

11.8. Enum DataBits

```
package enums;

public enum DataBits {

   //доступно 4 зачения

   A (8);
   //B (7),
   //C (6),
   //D (5);

   private final int bitsNum;

   DataBits(int bitsNum){
      this.bitsNum = bitsNum;
   }

   public int getBitsNum() {
      return bitsNum;
   }
}
```

11.9. Enum FrameTypes

```
package enums;

public enum FrameTypes {

   //код должен быть меньше 127
   ACK ((byte)104),
   LINK ((byte)96),
   UNLINK ((byte)97),
   PRM ((byte)120),
   I ((byte)111),
   RET ((byte)115);

   private final byte frameCode;

   FrameTypes(byte frameCode){
      this.frameCode = frameCode;
   }

   public byte getFrameCode() {
      return frameCode;
   }
}
```

11.10. Enum Parity

```
package enums;
```

```
public enum Parity {

   //доступно 2 значения

   A ("Нет"),
   B ("Да");

   private final String status;

   Parity(String status){
      this.status = status;
   }

   public String getStatus() {
      return status;
   }
}
```

11.11. Enum StopBits

```
package enums;

public enum StopBits {

   //доступно 4 значения

   A ("1"),
   B ("1,5"),
   C ("2");

   private final String stopBits;

   StopBits(String stopBits){
      this.stopBits = stopBits;
   }

   public String getStopBits() {
      return stopBits;
   }
}
```