**Московский государственный технический университет им. Н.Э. Баумана**

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

|  |  |
| --- | --- |
| Утверждаю: |  |
| Галкин В.А | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2014 г. |

**Курсовая работа**

**по курсу «Сетевые технологии»:**

##### «Программа передачи коротких сообщений»

Техническое задание

(вид документа)

писчая бумага

(вид носителя)

5

(количество листов)

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛИ: |  |
| студенты групп  ИУ5-63 и ИУ5-64 |  |
| Леонтьев А.В.  Корнуков Н.О.  Латкин И.И. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2014 г. |
|  |  |

Москва - 2014

Оглавление

[1. Введение 3](#_Toc386905590)

[2. Пакет gui 3](#_Toc386905591)

[2.1. Пакет templates 3](#_Toc386905592)

[2.1.1. Шаблон chat.fxml 3](#_Toc386905593)

[2.1.2. Шаблон connection.fxml 5](#_Toc386905594)

[2.2. Класс ChatController 6](#_Toc386905595)

[2.2.1. Поля 6](#_Toc386905596)

[2.2.2. События 7](#_Toc386905597)

[2.2.3. Методы 7](#_Toc386905598)

[2.2.3. Листинг 8](#_Toc386905599)

[2.3. Класс ChatController 15](#_Toc386905600)

[2.3.1. Поля 15](#_Toc386905601)

[2.3.2. События 16](#_Toc386905602)

[2.3.3. Методы 16](#_Toc386905603)

[2.3.4. Листинг 16](#_Toc386905604)

[2.4. Класс DataController 18](#_Toc386905605)

[2.4.1. Поля 18](#_Toc386905606)

[2.4.2. События 18](#_Toc386905607)

[2.4.3. Методы 18](#_Toc386905608)

[2.4.4. Листинг 18](#_Toc386905609)

[2.5. Класс DataStage 19](#_Toc386905610)

[2.5.1. Поля 19](#_Toc386905611)

[2.5.2. События 19](#_Toc386905612)

[2.5.3. Методы 19](#_Toc386905613)

[2.5.4. Листинг 20](#_Toc386905614)

[2.6. Перечисление DialogResult 20](#_Toc386905615)

[2.6.1. Значения 20](#_Toc386905616)

[2.7. Класс Main 20](#_Toc386905617)

[2.7.1. События 21](#_Toc386905618)

[2.7.2. Методы 21](#_Toc386905619)

[2.8. Перечисление MessageLevel 21](#_Toc386905620)

[2.8.1. Значения 21](#_Toc386905621)

[2.8.2. Поля 21](#_Toc386905622)

[2.8.2. Методы 21](#_Toc386905623)

[2.9. Перечисление Status 21](#_Toc386905624)

[2.9.1. Значения 21](#_Toc386905625)

[2.9.2. Поля 22](#_Toc386905626)

[2.9.2. Методы 22](#_Toc386905627)

[3. Пакет layers 22](#_Toc386905628)

[3.1. Пакет apl 22](#_Toc386905629)

[3.2. Пакет dll 22](#_Toc386905630)

[3.3. Пакет phy 22](#_Toc386905631)

[3.3.1. Пакет settings 22](#_Toc386905632)

[3.4. Пакет exceptions 22](#_Toc386905633)

[4. Пакет util 22](#_Toc386905634)

# 1. Введение

Программный продукт разработан с использованием технологии JavaFX (графический интерфейс пользователя) и RXTX (для работы физического уровня) на языке программирования Java.

Разработка осуществлялась в среде программирования IntelliJ IDEA при помощи распределённой системы контроля версия (git).

# 2. Пакет gui

## 2.1. Пакет templates

### 2.1.1. Шаблон chat.fxml

##### 2.1.1.1. Листинг

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.\*?>

<?import javafx.scene.layout.\*?>

<?import javafx.scene.paint.Color?>

<?import javafx.scene.shape.Circle?>

<?import javafx.scene.web.WebView?>

<VBox fx:id="layout" layoutX="0.0" layoutY="0.0" prefHeight="344.0" prefWidth="483.0" xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/2.2" fx:controller="gui.ChatController">

<children>

<MenuBar>

<menus>

<Menu mnemonicParsing="false" text="Connection">

<items>

<MenuItem mnemonicParsing="false" onAction="#onMenuConnect" text="Connect" />

<MenuItem mnemonicParsing="false" onAction="#onMenuDisconnect" text="Disconnect" />

</items>

</Menu>

<Menu mnemonicParsing="false" text="Help">

<items>

<MenuItem mnemonicParsing="false" onAction="#onMenuAbout" text="About" />

</items>

</Menu>

</menus>

</MenuBar>

<SplitPane dividerPositions="0.5036764705882353" focusTraversable="true" orientation="VERTICAL" prefHeight="200.0" prefWidth="160.0" VBox.vgrow="ALWAYS">

<items>

<AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="100.0" prefWidth="160.0">

<children>

<WebView fx:id="webView" maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" prefHeight="133.0" prefWidth="481.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0" />

</children>

</AnchorPane>

<AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="100.0" prefWidth="160.0">

<children>

<TextArea fx:id="inputField" maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" onKeyReleased="#onKeyReleased" prefHeight="133.0" prefWidth="481.0" wrapText="true" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0" />

</children>

</AnchorPane>

</items>

</SplitPane>

<HBox prefHeight="40.0" prefWidth="318.0">

<children>

<ToolBar maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" prefHeight="40.0" prefWidth="351.0" HBox.hgrow="ALWAYS">

<items>

<Circle fx:id="statusIcon" fill="RED" radius="6.00006103515625" stroke="BLACK" strokeType="INSIDE" />

<Label fx:id="statusText" alignment="CENTER\_LEFT" text="Not connected" textAlignment="LEFT">

<textFill>

<Color blue="0.388" green="0.388" red="0.388" />

</textFill>

</Label>

</items>

</ToolBar>

<ToolBar maxHeight="1.7976931348623157E308">

<Circle fx:id="ctsIcon" fill="RED" radius="6.00006103515625" stroke="BLACK" strokeType="INSIDE" />

<Label fx:id="ctsText" alignment="CENTER\_LEFT" text="CTS" textAlignment="LEFT">

<textFill>

<Color blue="0.388" green="0.388" red="0.388" />

</textFill>

</Label>

</ToolBar>

<ToolBar maxHeight="1.7976931348623157E308">

<items>

<Button fx:id="sendButton" defaultButton="true" disable="true" mnemonicParsing="false" onAction="#sendClick" text="Send" />

</items>

</ToolBar>

</children>

</HBox>

</children>

</VBox>

### 2.1.2. Шаблон connection.fxml

##### 2.1.2.1. Листинг

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.collections.FXCollections?>

<?import javafx.geometry.Insets?>

<?import javafx.scene.control.\*?>

<?import javafx.scene.layout.\*?>

<GridPane fx:id="layout" hgap="10.0" maxHeight="1.7976931348623157E308" prefHeight="205.0" prefWidth="400.0" vgap="10.0" xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/2.2" fx:controller="gui.ConnectionController">

<children>

<Label text="Nickname:" GridPane.columnIndex="0" GridPane.rowIndex="0" GridPane.vgrow="SOMETIMES" />

<TextField fx:id="userName" prefWidth="269.0" text="User" GridPane.columnIndex="1" GridPane.columnSpan="2147483647" GridPane.rowIndex="0" GridPane.vgrow="SOMETIMES" />

<Label text="COM port:" GridPane.columnIndex="0" GridPane.rowIndex="1" />

<ComboBox fx:id="comPort" maxWidth="1.7976931348623157E308" prefWidth="180.0" GridPane.columnIndex="1" GridPane.columnSpan="2" GridPane.rowIndex="1">

<items>

<FXCollections fx:factory="observableArrayList" />

</items>

</ComboBox>

<Label text="Data bits:" GridPane.columnIndex="0" GridPane.rowIndex="2" />

<ComboBox fx:id="dataBits" maxWidth="1.7976931348623157E308" prefWidth="-1.0" GridPane.columnIndex="1" GridPane.rowIndex="2" />

<Label text="Parity check:" GridPane.columnIndex="0" GridPane.rowIndex="3" />

<ComboBox fx:id="parityCheck" maxWidth="1.7976931348623157E308" prefWidth="-1.0" GridPane.columnIndex="1" GridPane.rowIndex="3" GridPane.vgrow="ALWAYS" />

<Button alignment="BASELINE\_CENTER" defaultButton="true" maxWidth="1.7976931348623157E308" mnemonicParsing="false" onAction="#onConnect" text="Connect" GridPane.columnIndex="3" GridPane.halignment="CENTER" GridPane.hgrow="ALWAYS" GridPane.rowIndex="4" />

<Label text="Stop bits:" GridPane.columnIndex="2" GridPane.rowIndex="3" />

<ComboBox fx:id="stopBits" maxWidth="1.7976931348623157E308" prefWidth="-1.0" GridPane.columnIndex="3" GridPane.rowIndex="3" />

<Label text="Baud rate:" GridPane.columnIndex="2" GridPane.rowIndex="2" />

<ComboBox fx:id="baudRate" maxWidth="1.7976931348623157E308" prefWidth="-1.0" GridPane.columnIndex="3" GridPane.rowIndex="2" />

<Button fx:id="refresh" maxWidth="1.7976931348623157E308" mnemonicParsing="false" onAction="#onRefresh" prefWidth="-1.0" text="Refresh" textAlignment="CENTER" GridPane.columnIndex="3" GridPane.hgrow="ALWAYS" GridPane.rowIndex="1" />

</children>

<columnConstraints>

<ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

</columnConstraints>

<padding>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />

</padding>

<rowConstraints>

<RowConstraints minHeight="10.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" vgrow="SOMETIMES" />

</rowConstraints>

</GridPane>

## 2.2. Класс ChatController

Класс-контроллер для главного окна чата.

### 2.2.1. Поля

* @FXML private Button sendButton;  
  Кнопка Send
* @FXML private WebView webView;  
  Окно сообщений, на базе веб-браузера
* @FXML private TextArea inputField;  
  Поле ввода сообщения для передачи
* @FXML private Circle statusIcon;  
  Лампочка состояния связи
* @FXML private Label statusText;  
  Текст состояния связи
* @FXML private Circle ctsIcon;  
  Лампочка, отображающая состояние линии CTS
* public static final String PROGRAM\_NAME = "ComChat";  
  Имя программы
* public static final String PROGRAM\_VERSION = "v0.1 alpha";  
  Версия программы
* private ProtocolStack protocolStack;  
  Ссылка на стек собранный стек протоколов
* private Status status;  
  Состояние связи
* private String localUser = null;  
  Имя локального пользователя
* private String remoteUser = null;  
  Имя удалённого пользователя
* private boolean isAuthorized = false;  
  Факт авторизованности локального пользователя
* private static final String messageSent = "[×] ";  
  Префикс отправленного, но не доставленного сообщения
* private static final String messageReceived = "[«] ";  
  Префикс принятого сообщения
* private static final String messageAck = "[»] ";  
  Префикс доставленного сообщения
* private boolean isConnected = false;  
  Факт наличия соединения
* private Integer messageId = 0;  
  Абсолютный номер сообщения, используется как id для div’ов сообщений в окне чата
* private boolean isClosing = false;  
  Факт того, что приложение находится в процессе завершения своей работы
* private HashMap<Integer, Integer> messageIdToHtmlId = new HashMap<>();  
  Отображение id сообщения -> div id строки в окне чата

### 2.2.2. События

* public void onMenuConnect(ActionEvent event)  
  Нажатие Connect в меню
* public void onMenuDisconnect(ActionEvent event)  
  Нажатие Disconnect в меню
* public void onMenuAbout(ActionEvent event)  
  Нажатие About в меню
* public void onKeyReleased(KeyEvent event)  
  Нажатие Enter в поле ввода сообщения
* public void sendClick(ActionEvent event)  
  Нажатие кнопки Send

### 2.2.3. Методы

* public void initWithData(Stage stage, Object data)  
  Унаследован от DataController
* private void addUserMessage(String author, String message)  
  Добавление пользовательского сообщения в чат
  + author – от чьего имени
  + message – текст сообщения
* private void addSystemMessage(MessageLevel level, String message)  
  Добавление системного сообщения в чат
  + level – уровень сообщения
  + message – текст сообщения
* private void send()  
  Отправка сообщения из поля ввода
* private void receive(Message message)  
  Приём сообщения от прикладного уровня
  + message – принятое сообщение
* private void markMessage(int id)  
  Пометка сообщения как доставленного
  + id – номер сообщения прикладного уровня
* private void onConnect()  
  Обработка подключения
* private String getHtmlPage()  
  Возвращает базовую html-вёрстку окна чата
* private void scrollChatToId(int id)  
  Прокрутка чата к id’ному сообщению
  + id – id сообщения в вёрстке (div#id)
* private void updateStatus(boolean connected)  
  Смена состояния связи
  + connected – факт наличие соединения
* private void updateCompanionStatus(boolean connected)  
  Смена состояния оппонента
  + connected – факт наличия собеседника (факт того, что Auth принят)
* private void updateCTS(boolean CTS)  
  Смена состояния CTS
  + CTS – наличие или отсутствие сигнала на линии CTS
* private void onError(Exception e)  
  Ошибка от любого уровня
  + e - исключение

### 2.2.3. Листинг

package gui;

import javafx.application.Platform;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.TextArea;

import javafx.scene.input.KeyCode;

import javafx.scene.input.KeyEvent;

import javafx.scene.layout.VBox;

import javafx.scene.paint.Color;

import javafx.scene.shape.Circle;

import javafx.scene.web.WebEngine;

import javafx.scene.web.WebView;

import javafx.stage.Modality;

import javafx.stage.Stage;

import javafx.stage.StageStyle;

import layers.ProtocolStack;

import layers.apl.Message;

import org.controlsfx.dialog.Dialogs;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.Text;

import sun.reflect.generics.reflectiveObjects.NotImplementedException;

import java.io.IOException;

import java.util.HashMap;

public class ChatController extends DataController {

@FXML private Button sendButton;

@FXML private WebView webView;

@FXML private TextArea inputField;

@FXML private VBox layout;

@FXML private Circle statusIcon;

@FXML private Label statusText;

@FXML private Circle ctsIcon;

@FXML private Label ctsText;

public static final String PROGRAM\_NAME = "ComChat";

public static final String PROGRAM\_VERSION = "v0.1 alpha";

private ProtocolStack protocolStack;

private Status status;

private String localUser = null;

private String remoteUser = null;

private boolean isAuthorized = false;

private static final String messageSent = "[×] ";

private static final String messageReceived = "[«] ";

private static final String messageAck = "[»] ";

private boolean isConnected = false;

private Integer messageId = 0;

private boolean isClosing = false;

private HashMap<Integer, Integer> messageIdToHtmlId = new HashMap<>();

@Override

public void initWithData(Stage stage, Object data) {

super.initWithData(stage, data);

protocolStack = (ProtocolStack) data;

WebEngine engine = webView.getEngine();

engine.loadContent(getHtmlPage());

statusIcon.setFill(Status.NotConnected.toColor());

statusText.setText(Status.NotConnected.toString());

protocolStack.getPhy().subscribeConnectionStatusChanged(this::updateStatus);

protocolStack.getPhy().subscribeCompanionConnectedChanged(this::updateCompanionStatus);

protocolStack.getPhy().subscribeSendingAvailableChanged(this::updateCTS);

stage.setOnCloseRequest(e -> {

isClosing = true;

if (status != Status.NotConnected)

protocolStack.getApl().disconnect();

});

protocolStack.getApl().subscribeToReceive(this::receive);

Platform.runLater(this::showConnectionDialog);

}

private void updateStatus(boolean connected) {

this.isConnected = connected;

Status newStatus = Status.fromBoolean(connected);

if (status == newStatus)

return;

Platform.runLater(() -> {

status = newStatus;

statusIcon.setFill(status.toColor());

statusText.setText(status.toString());

//sendButton.setDisable(!connected);

if (status == Status.NotConnected && !isClosing) {

isAuthorized = false;

protocolStack.getApl().disconnect();

addSystemMessage(MessageLevel.Info, "Disconnected");

Dialogs.create()

.owner(stage)

.title(PROGRAM\_NAME)

.masthead("Information")

.message("Disconnected")

.showInformation();

}

});

}

private void updateCompanionStatus(boolean connected) {

if (!this.isConnected)

return;

Status newStatus = connected ? Status.Chatting : Status.Connected;

if (status != newStatus) {

//if we are connected and connection form closed (overwise Auth will be sent after form closing)

if (connected && !isAuthorized && localUser != null) {

protocolStack.getApl().send(Message.Type.Auth, localUser);

isAuthorized = true;

}

else if (status == Status.Chatting) {

addSystemMessage(MessageLevel.Info, "Remote user disconnected, waiting for another companion...");

isAuthorized = false;

}

Platform.runLater(() -> {

status = newStatus;

statusIcon.setFill(status.toColor());

statusText.setText(status.toString());

sendButton.setDisable(!connected);

});

}

}

private void updateCTS(boolean CTS) {

Platform.runLater(() -> {

if (CTS)

ctsIcon.setFill(Color.YELLOWGREEN);

else

ctsIcon.setFill(Color.RED);

});

}

private void addUserMessage(String author, String message) {

addUserMessage(author, message, 0);

}

private void addUserMessage(String author, String message, Integer id) {

WebEngine engine = webView.getEngine();

Document document = engine.getDocument();

Node body = document.getElementsByTagName("BODY").item(0);

Element div = document.createElement("div");

messageIdToHtmlId.put(id, messageId);

div.setAttribute("id", messageId.toString());

div.setAttribute("data-message-id", id.toString());

Text text = document.createTextNode(message);

Element mark = document.createElement("b");

if (id != 0)

mark.setTextContent(messageSent);

else

mark.setTextContent(messageReceived);

Element b = document.createElement("b");

b.setTextContent(author + ": ");

div.appendChild(mark);

div.appendChild(b);

div.appendChild(text);

body.appendChild(div);

scrollChatToId(messageId++);

}

private void addSystemMessage(MessageLevel level, String message) {

Platform.runLater(() -> {

WebEngine engine = webView.getEngine();

Document document = engine.getDocument();

Node body = document.getElementsByTagName("BODY").item(0);

Element div = document.createElement("div");

div.setAttribute("id", messageId.toString());

div.setAttribute("style", "color: " + level.toHtmlColor());

Text text = document.createTextNode(message);

Element b = document.createElement("b");

b.setTextContent("[" + level.toString() + "]" + " System message: ");

div.appendChild(b);

div.appendChild(text);

body.appendChild(div);

scrollChatToId(messageId++);

});

}

private void send() {

String message = inputField.getText();

int id = protocolStack.getApl().send(Message.Type.Msg, message);

addUserMessage(localUser, message, id);

inputField.clear();

}

private void receive(Message message) {

//JavaFX UI thread synchronization

Platform.runLater(() -> {

addSystemMessage(MessageLevel.Debug, "Message received: " + message.getType().name());

switch (message.getType()) {

case Msg:

addUserMessage(remoteUser, message.getMsg());

protocolStack.getApl().send(Message.Type.Ack,

Integer.toString(message.getId()));

break;

case Auth:

remoteUser = message.getMsg();

addSystemMessage(MessageLevel.Info, "Remote user connected: " + message.getMsg());

protocolStack.getApl().handshakeFinished();

//TODO: enable "sendability"

break;

case Ack:

int id = Integer.parseInt(message.getMsg());

markMessage(id);

break;

default:

throw new NotImplementedException();

}

});

}

private void markMessage(int id) {

WebEngine engine = webView.getEngine();

Document document = engine.getDocument();

Integer htmlId = messageIdToHtmlId.get(id);

if (htmlId == null)

return;

Element div = document.getElementById(htmlId.toString());

Element mark = (Element) div.getFirstChild();

mark.setTextContent(messageAck);

}

public void sendClick(ActionEvent event) {

send();

}

public void onMenuConnect(ActionEvent event) throws IOException {

showConnectionDialog();

}

private void showConnectionDialog() {

try {

FXMLLoader loader = new FXMLLoader(Main.class.getResource("/gui/templates/connection.fxml"));

Parent root = loader.load();

DataStage connectionStage = new DataStage(loader.getController(), protocolStack);

connectionStage.setTitle("Connection");

final double rem = javafx.scene.text.Font.getDefault().getSize() / 13;

Scene conScene = new Scene(root, 400.0\*rem, 205.0\*rem);

connectionStage.setScene(conScene);

connectionStage.setResizable(false);

connectionStage.initModality(Modality.WINDOW\_MODAL);

connectionStage.initOwner(layout.getScene().getWindow());

connectionStage.initStyle(StageStyle.UTILITY);

connectionStage.showAndWait();

if (connectionStage.getResult() == DialogResult.OK) {

localUser = (String) connectionStage.getResultData();

onConnect();

} else {

Dialogs.create()

.owner(stage)

.title(PROGRAM\_NAME)

.masthead("Information")

.message("Connection cancelled")

.showInformation();

}

}

catch (IOException e) {

Dialogs.create()

.owner(stage)

.title(PROGRAM\_NAME)

.masthead("Error")

.showException(e);

}

}

private void onConnect() {

//not sure why runLater needed here, cause we are already in UI thread and webEngine already loaded

Platform.runLater(() -> addSystemMessage(MessageLevel.Info, "Successfully connected"));

if (!isAuthorized) {

isAuthorized = true;

protocolStack.getApl().send(Message.Type.Auth, localUser);

}

protocolStack.getApl().subscribeOnError(this::onError);

}

private void onError(Exception e) {

protocolStack.getApl().disconnect();

addSystemMessage(MessageLevel.Error, "Connection lost");

Dialogs.create()

.owner(stage)

.title(PROGRAM\_NAME)

.masthead("Error")

.message("Connection lost")

.showException(e);

}

public void onMenuDisconnect(ActionEvent event) {

if (status != Status.NotConnected)

protocolStack.getApl().disconnect();

}

private String getHtmlPage() {

return "<html><head></head><body></body></html>";

}

public void onMenuAbout(ActionEvent event) {

Dialogs.create()

.owner(stage)

.title(PROGRAM\_NAME)

.masthead("About")

.message("BMSTU course project.\nCOM-port based chat for 2 persons.\n\nAuthors:\nLeontiev Aleksey - Application Layer and GUI\nLatkin Igor - Physical Layer\nKornukov Nikita - Data Link Layer\n\nProject's home:\nhttps://github.com/tech-team/comchat")

.showInformation();

}

public void onKeyReleased(KeyEvent event) {

if (event.getCode() == KeyCode.ENTER && !event.isControlDown()) {

if (status != Status.NotConnected)

send();

else

Dialogs.create()

.owner(stage)

.title(PROGRAM\_NAME)

.masthead("Error")

.message("You should connect first.\nUse Connection -> Connect.")

.showError();

}

}

private void scrollChatToId(int id) {

try {

WebEngine engine = webView.getEngine();

engine.executeScript("document.getElementById(" + id + ").scrollIntoView()");

}

catch (NullPointerException ignored) {}

}

}

## 2.3. Класс ChatController

Класс-контроллер для окна настроек подключения

### 2.3.1. Поля

* @FXML private TextField userName;  
  Поле ввода имени пользователя
* @FXML private ComboBox<String> comPort;  
  Список выбора комп-порта
* @FXML private ComboBox<Integer> baudRate;  
  Список выбора скорости передачи
* @FXML private ComboBox<String> dataBits;  
  Список выбора количества информационных битов
* @FXML private ComboBox<String> stopBits;  
  Список выбора количества стоповых битов
* @FXML private ComboBox<String> parityCheck;  
  Список выбора способа проверки чётности
* private ProtocolStack protocolStack;  
  Ссылка на стек собранный стек протоколов

### 2.3.2. События

* public void onConnect(ActionEvent event)  
  Нажатие на кнопку Connect
* public void onRefresh(ActionEvent event)  
  Нажатие на кнопку Refresh (обновления списка ком-портов)

### 2.3.3. Методы

Все методы унаследованы от DataController.

### 2.3.4. Листинг

package gui;

import javafx.application.Platform;

import javafx.collections.FXCollections;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.ComboBox;

import javafx.scene.control.TextField;

import javafx.stage.Stage;

import layers.ProtocolStack;

import layers.exceptions.ConnectionException;

import layers.phy.ComPort;

import layers.phy.settings.comport\_settings.ComPortSettings;

import layers.phy.settings.comport\_settings.DataBitsEnum;

import layers.phy.settings.comport\_settings.ParityEnum;

import layers.phy.settings.comport\_settings.StopBitsEnum;

import org.controlsfx.dialog.Dialogs;

public class ConnectionController extends DataController {

@FXML private TextField userName;

@FXML private ComboBox<String> comPort;

@FXML private ComboBox<Integer> baudRate;

@FXML private ComboBox<String> dataBits;

@FXML private ComboBox<String> stopBits;

@FXML private ComboBox<String> parityCheck;

private ProtocolStack protocolStack;

@Override

public void initWithData(Stage stage, Object data) {

super.initWithData(stage, data);

protocolStack = (ProtocolStack) data;

comPort.setItems(FXCollections.observableArrayList(ComPort.getAvailablePorts()));

baudRate.setItems(FXCollections.observableArrayList(ComPort.getAvailableBaudRates()));

dataBits.setItems(FXCollections.observableArrayList(ComPort.getAvailableDataBits()));

stopBits.setItems(FXCollections.observableArrayList(ComPort.getAvailableStopBits()));

parityCheck.setItems(FXCollections.observableArrayList(ComPort.getAvailableParity()));

comPort.setValue(ComPort.getDefaultPort());

baudRate.setValue(ComPort.getDefaultBaudRate());

dataBits.setValue(ComPort.getDefaultDataBits());

stopBits.setValue(ComPort.getDefaultStopBits());

parityCheck.setValue(ComPort.getDefaultParity());

}

public void onConnect(ActionEvent event) {

if (comPort.getValue().isEmpty())

{

Dialogs.create()

.owner(stage)

.title(ChatController.PROGRAM\_NAME)

.masthead("Error")

.message("Please, select COM port.\nIf you do not see any listed, press Refresh button.")

.showError();

return;

}

if (userName.getText().isEmpty())

{

Dialogs.create()

.owner(stage)

.title(ChatController.PROGRAM\_NAME)

.masthead("Error")

.message("Please, input your username.")

.showError();

return;

}

try {

protocolStack.getApl().connect(

new ComPortSettings(comPort.getValue(),

baudRate.getValue(),

DataBitsEnum.fromString(dataBits.getValue()).toDataBits(),

StopBitsEnum.fromString(stopBits.getValue()).toStopBits(),

ParityEnum.fromString(parityCheck.getValue()).toParity()));

result = DialogResult.OK;

resultData = userName.getText();

this.close();

}

catch(ConnectionException e) {

String message = "Unable to connect with these settings";

Exception withMessage = new Exception(message, e);

Platform.runLater(() -> Dialogs.create()

.owner(stage)

.title("ComChat")

.masthead("Error")

.message(message) //well, that has no effect for exception dialog unfortunately

.showException(withMessage));

}

}

public void onRefresh(ActionEvent event) {

comPort.setItems(FXCollections.observableArrayList(ComPort.getAvailablePorts(true)));

comPort.setValue(ComPort.getDefaultPort());

}

}

## 2.4. Класс DataController

Абстрактный класс-контроллер с возможностью параметризованной инициализации и возврата значения и данных

### 2.4.1. Поля

* protected Object data;  
  Входные данные
* protected Object resultData;  
  Выходные данные
* protected DialogResult result = DialogResult.CLOSED;  
  Результат работы
* protected Stage stage;  
  Stage, на котором происходит отрисовка шаблона

### 2.4.2. События

* protected void close()  
  Закрытие сцены (окна)

### 2.4.3. Методы

* public void initWithData(Stage stage, Object data)  
  Инициализация котроллера
  + stage – сцена, на которой размещены элементы
  + data – произвольный входной объект
* public DialogResult getResult()

Возврат результата работы окна

* public Object getResultData()  
  Возврат результата работы окна (данные)
* public void setStage(Stage stage)  
  Установка сцены для контроллера
  + stage – сцена к установке

### 2.4.4. Листинг

package gui;

import javafx.fxml.FXML;

import javafx.stage.Stage;

public abstract class DataController {

protected Object data;

protected Object resultData;

protected DialogResult result = DialogResult.CLOSED;

protected Stage stage;

public void initWithData(Stage stage, Object data) {

this.stage = stage;

this.data = data;

}

public DialogResult getResult() {

return result;

}

public Object getResultData() {

return resultData;

}

public void setStage(Stage stage) {

this.stage = stage;

}

@FXML

protected void close() {

stage.close();

}

}

## 2.5. Класс DataStage

Абстрактный класс сцены (окна) с возможностью параметризованной инициализации и возврата значения и данных

### 2.5.1. Поля

* private DataController controller;  
  Ссылка на соответствующий сцене котроллер

### 2.5.2. События

* protected void close()  
  Закрытие сцены (окна)

### 2.5.3. Методы

* public DataStage(DataController controller, Object data)  
  Конструктор
  + controller для привязки к сцене
  + data – произвольная входная информация
* public DialogResult getResult()   
  Возврат результата работы окна
* public Object getResultData()  
  Возврат результата работы окна (данные)
* public DataController getController()  
  Возврат контроллера

### 2.5.4. Листинг

package gui;

import javafx.stage.Stage;

public class DataStage extends Stage {

private DataController controller;

public DataStage(DataController controller, Object data) {

this.controller = controller;

controller.initWithData(this, data);

}

public DialogResult getResult() {

return controller.getResult();

}

public Object getResultData() {

return controller.getResultData();

}

public DataController getController() {

return controller;

}

}

## 2.6. Перечисление DialogResult

Перечисление возможных результатов работы окна

### 2.6.1. Значения

* OK  
  Нажата кнопка OK
* CANCEL  
  Нажата кнопка CANCEL
* YES  
  Нажата кнопка YES
* NO  
  Нажата кнопка YES
* CLOSED  
  Диалог закрыт
* ERROR  
  Произошла ошибка

## 2.7. Класс Main

Главный класс

### 2.7.1. События

* public void start(Stage primaryStage)  
  Происходит после инициализации JavaFX
  + primaryStage – объект сцены (главное окно)

### 2.7.2. Методы

* public static void main(String[] args)  
  Точка входа в приложение
  + args – массив аргументов командной строки

## 2.8. Перечисление MessageLevel

Перечисление возможных типов сообщений в чате

### 2.8.1. Значения

* Error("ERROR", "red")  
  Сообщение об ошибке
* Info("INFO", "gray")  
  Информационной сообщение (основной тип)
* Debug("DEBUG", "blue")  
  Отладочное сообщение

### 2.8.2. Поля

* String level;  
  Строковое описание
* String htmlColor;  
  Цвет в html-формате

### 2.8.2. Методы

* MessageLevel(String level, String htmlColor)  
  Конструктор
  + level – уровень сообщения
  + htmlColor – цвет сообщения в html представлении
* public String toString()  
  Получение строкового представления
* public String toHtmlColor()  
  Получение цветового представления

## 2.9. Перечисление Status

Перечисление возможных состояний связи

### 2.9.1. Значения

* NotConnected("Not connected", Color.RED)  
  Не подключено
* Connected("Connected. Waiting for companion", Color.YELLOW)  
  Подключено
* Chatting("Chatting", Color.YELLOWGREEN)  
  Идёт переписка (т.е. подключено и на другом конце есть собеседник)
* Error("Error", Color.RED)  
  Произошла ошибка

### 2.9.2. Поля

* String value;  
  Строковое описание
* Color color;  
  Цветовое представление

### 2.9.2. Методы

* private Status(String value, Color color)  
  Конструктор
  + value – строковое значение
  + color – цвет статуса
* public String toString()  
  Получение строкового представления
* public String toHtmlColor()  
  Получение цветового представления
* public static Status fromBoolean(boolean connected)  
  Создание статуса по булевому значению (подключен/не подключен)
  + connected – факт наличия соединения

# 3. Пакет layers

## 3.1. Пакет apl

### 3.1.1. Интерфейс IApplicationLayer

Абстракция прикладного уровня

##### 3.1.1.1. Методы

* void connect(PhysicalLayerSettings settings);  
  Соединиться
  + settings – настройки для передачи на физический уровень
* void disconnect();  
  Отсоединиться
* int send(Message.Type type, String msg);  
  Послать сообщение
  + type – тип сообщения
  + msg – текст сообщения
* void receive(byte[] data);  
  Принять сообщение
  + data – сериализованное сообщения
* void subscribeToReceive(final Consumer<Message> receiver);  
  Подписаться на получение сообщений
  + receiver - обработчик
* void handshakeFinished();  
  Завершить процедуру авторизации

##### 3.1.1.2. Листинг

package layers.apl;

import layers.ILayer;

import layers.exceptions.ConnectionException;

import layers.phy.settings.PhysicalLayerSettings;

import java.util.function.Consumer;

public interface IApplicationLayer extends ILayer {

void connect(PhysicalLayerSettings settings) throws ConnectionException;

void disconnect();

int send(Message.Type type, String msg);

void receive(byte[] data);

void subscribeToReceive(final Consumer<Message> receiver);

void handshakeFinished();

}

### 3.1.2. Класс ApplicationLayer

Реализация прикладного уровня

##### 3.1.2.1. Поля

* private IDataLinkLayer dll  
  Ссылка на канальный уровень
* private List<Consumer<Message>> receivers  
  Список подписок на приём сообщений
* private int messageId  
  текущий id переданного сообщения
* private List<Consumer<Exception>> onErrorListeners  
  Список подписок на ошибки

##### 3.1.2.1. Методы

Все методы являются реализациями методом интерфейсов IApplicationLayer и ILayer.

##### 3.1.2.2. Листинг

package layers.apl;

import layers.ILayer;

import layers.dll.IDataLinkLayer;

import layers.exceptions.ConnectionException;

import layers.phy.settings.PhysicalLayerSettings;

import java.util.LinkedList;

import java.util.List;

import java.util.function.Consumer;

public class ApplicationLayer implements IApplicationLayer {

private IDataLinkLayer dll;

private List<Consumer<Message>> receivers = new LinkedList<>();

private int messageId = 0;

private List<Consumer<Exception>> onErrorListeners = new LinkedList<>();

@Override

public ILayer getUpperLayer() {

return null;

}

@Override

public IDataLinkLayer getLowerLayer() {

return dll;

}

@Override

public void setUpperLayer(ILayer layer) {

}

@Override

public void setLowerLayer(ILayer layer) {

dll = (IDataLinkLayer) layer;

}

@Override

public void subscribeOnError(Consumer<Exception> listener) {

onErrorListeners.add(listener);

}

private void notifyOnError(Exception e) {

onErrorListeners.forEach(listener -> listener.accept(e));

}

@Override

public void connect(PhysicalLayerSettings settings) throws ConnectionException {

getLowerLayer().connect(settings);

}

@Override

public void disconnect() {

getLowerLayer().disconnect();

messageId = 1;

}

@Override

public int send(Message.Type type, String msg) {

System.out.println("Sent: " + msg);

if (type == Message.Type.Msg)

++messageId;

dll.send(new Message(messageId, type, msg).serialize());

return messageId;

}

@Override

public void receive(byte[] data) {

Message message = Message.deserialize(data);

System.out.println("Received: " + message.getMsg());

receivers.forEach(receiver -> receiver.accept(message));

}

@Override

public void subscribeToReceive(final Consumer<Message> receiver) {

receivers.add(receiver);

}

@Override

public void handshakeFinished() {

getLowerLayer().handshakeFinished();

}

}

### 3.1.3. Класс Message

PDU прикладного уровня, сообщение.

##### 3.1.3.1. Класс Type

**3.1.3.1.1. Значения**

* Auth  
  Сообщение об авторизации
* Msg  
  Пользовательское сообщение
* Ack  
  Подтверждение о доставке пользовательского сообщения

**3.1.3.1.2. Методы**

* public static Type fromInteger(int x)  
  Создание типа по соответствующему ему числу (при десериализации)
  + x – тип в числовом представлении (результат ordinal())

##### 3.1.3.2. Поля

* private int id;  
  Последовательный уникальный идентификатор сообщения
* private Type type;  
  Тип сообщения
* private String msg;  
  Текст сообщения

##### 3.1.3.3. Методы

* public Message(int id, Type type, String msg)  
  Конструктор
  + id – номер сообщения (равен нулю для всех типов сообщений, кроме Msg)
  + type – тип сообщения
* public int getId()  
  Возвращает id сообщения
* public Type getType()  
  Возвращает тип сообщения
* public String getMsg()  
  Возвращает текст сообщения
* public byte[] serialize()  
  Сериализация сообщения в массив байт
* public static Message deserialize(byte[] data)  
  Десериализация сообщения из массива байт
  + data – сериализованное сообщение

##### 3.1.3.4. Листинг

package layers.apl;

import layers.PDU;

import util.ArrayUtils;

import java.io.UnsupportedEncodingException;

import java.nio.ByteBuffer;

import java.util.Arrays;

public class Message implements PDU {

public enum Type {

Auth, Msg, Ack;

public static Type fromInteger(int x) throws Exception {

int max = Type.values().length;

if (x < 0 || x >= max)

throw new Exception("bad range! x was: " + x);

return Type.values()[x];

}

}

private int id;

private Type type;

private String msg;

public Message(int id, Type type, String msg) {

this.id = id;

this.type = type;

this.msg = msg;

}

public int getId() {

return id;

}

public Type getType() {

return type;

}

public String getMsg() {

return msg;

}

@Override

public byte[] serialize() {

byte typeByte = (byte) type.ordinal();

byte[] idBytes = ByteBuffer.allocate(4).putInt(this.id).array();

byte[] infoBytes = ArrayUtils.concatenate(typeByte, idBytes);

byte[] msgBytes = new byte[0];

try {

msgBytes = msg.getBytes("UTF-8");

} catch (UnsupportedEncodingException ignored) {

System.err.println(ignored.getMessage());

}

return ArrayUtils.concatenate(infoBytes, msgBytes);

}

public static Message deserialize(byte[] data) {

byte typeByte = data[0];

byte[] idBytes = Arrays.copyOfRange(data, 1, 5);

byte[] msgBytes = Arrays.copyOfRange(data, 5, data.length);

Type type = null;

try {

type = Type.fromInteger((int) typeByte);

} catch (Exception e) {

e.printStackTrace(); // TODO: review

}

int id = ByteBuffer.wrap(idBytes).getInt();

String msg = null;

try {

msg = new String(msgBytes, "UTF-8");

} catch (UnsupportedEncodingException ignored) { }

return new Message(id, type, msg);

}

}

## 3.2. Пакет dll

## 3.3. Пакет phy

### 3.3.1. Пакет settings

##### 3.3.1.1. Пакет comport\_settings

## 3.4. Пакет exceptions

# 4. Пакет util

## 4.1. Класс ArrayUtils

### 4.1.1. Методы

* public static byte[] concatenate(byte[] A, byte[] B)  
  Объединение двух массивов байт
  + A, B – объединяемые операнды
* public static byte[] concatenate(byte a, byte[] B)  
  Объединение байта и массива байт
  + a, B – объединяемые операнды