Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная Техника»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

По курсу «Программирование на Java»

На тему «Реализация многомодульного приложения на языке Java»

Выполнил студент группы 22ВВП2:

Корнилов В.М.

Приняла:

к.т.н., доцент Юрова О.В.

Пенза 2025

# Содержание

[Введение 5](#_Toc198653968)

[1. Постановка задачи 6](#_Toc198653969)

[2. Этапы разработки 7](#_Toc198653970)

[2.1 Среда разработки 7](#_Toc198653971)

[2.2 Техническое задание к программе 7](#_Toc198653972)

[3. Описание клиент-серверной части 9](#_Toc198653973)

[4. Описание программы 10](#_Toc198653974)

[4.1 Файл Sever.java 10](#_Toc198653975)

[4.2 Файл Login.java 11](#_Toc198653976)

[4.3 Файл Chat.java. 13](#_Toc198653977)

[5. Руководство пользователя 15](#_Toc198653978)

[Заключение 18](#_Toc198653979)

[Список используемых источников 19](#_Toc198653980)

[Приложение A. Листинг программы 20](#_Toc198653981)

[Приложение A.1. Файл Server.java 20](#_Toc198653982)

[Приложение A.2. Файл Login.java 31](#_Toc198653983)

[Приложение A.3. Файл Chat.java 39](#_Toc198653984)

[Приложение Б. UML-диаграммы 54](#_Toc198653985)

[Приложение Б.1. UML-диаграмма вариантов использования 54](#_Toc198653986)

[Приложение Б.2. UML- диаграмма последовательности 55](#_Toc198653987)

[Приложение Б.3. UML- диаграмма деятельности 56](#_Toc198653988)

[Приложение Б.4. UML-диаграмма классов 57](#_Toc198653989)

[Приложение Б.5. UML-диаграмма развертывания 58](#_Toc198653990)

# Введение

На сегодняшний день Java остаётся одним из самых популярных языков программирования. Его история началась в 1996 году в компании Sun Microsystems, впоследствии ставшей частью Oracle. Задумываясь как универсальный инструмент «пиши‑один‑раз — запускай‑везде», Java прошла долгий путь развития и превратилась в целую экосистему.

С момента первой версии язык неоднократно обновлялся. На май  2025 года актуальным считался релиз Java SE 24. За прошедшие годы Java из просто гибкого языка выросла в платформу, объединяющую технологии для самых разных задач: от настольных приложений и крупных веб‑порталов до облачных сервисов. Java используют не только на обычных ПК, но и на планшетах, смартфонах и даже в бытовой технике. Достаточно вспомнить популярность операционной системы Android, где подавляющее большинство приложений пишут именно на Java. Главная особенность языка — промежуточная компиляция: исходный код переводится в платформо‑независимый байт‑код, который затем исполняет виртуальная машина JVM. Такая схема обеспечивает кроссплатформенность: один и тот же байт‑код можно запустить на Windows, Linux, macOS и других системах, если для них существует реализация JVM.

Синтаксис Java напоминает C/C++ и C#, поэтому разработчикам с опытом в этих языках освоить её проще. Ещё одно важное преимущество — автоматическая сборка мусора: освобождением памяти занимается JVM, снимая с программиста заботу о ручном управлении ресурсами. Java — строго объектно‑ориентированный язык со статической типизацией, поддержкой наследования и полиморфизма. Такой подход помогает строить крупные, гибкие и масштабируемые приложения, которые можно развивать годами. [2]

1. Постановка задачи

Разработать многомодульное приложение «Текстовый чат». Язык программирования Java, среда разработки NetBeans. Приложение должно обладать графическим интерфейсом и использовать следующие технологии:

1. Java Collection Framework;

2. Механики обработки исключительных ситуаций;

3. Java Stream API;

4. Java Multithreading;

5. Сетевое взаимодействие.

1. Этапы разработки
   1. Среда разработки

NetBeans IDE — это бесплатная, открытая среда разработки, которая даёт программисту всё необходимое для создания профессионального ПО. В ней можно писать как настольные, так и корпоративные, мобильные и веб‑приложения на Java, а также на C/C++, PHP, JavaScript, Groovy и Ruby.

Что предлагает NetBeans:

* Полностью настраиваемое рабочее пространство: панели, действия, «горячие» клавиши легко подгоняются под свои привычки.
* Расширяемый многоязычный редактор с поддержкой Java, C/C++, Ruby, Groovy, PHP, JavaScript, CSS, XML, HTML, JSP, Javadoc и других форматов; при желании можно добавить новый язык.
* Удобства для написания кода  — автоотступы, подсветка синтаксиса, проверка парных скобок и ключевых слов.

## Техническое задание к программе

Текстовый чат — это интерактивное окно, в котором пользователи обмениваются сообщениями в режиме реального времени. Интерфейс обычно делится на две зоны: история переписки и поле ввода, а также список пользователей. Приложение должно включать:

* Авторизация. У пользователей при запуске приложения должно быть окно авторизации, где после ввода и проверки данных с тем, что имеется в базе данных пользователь сможет зарегистрироваться или войти в приложение
* Отправка сообщений. Пользователь вводит текст в поле ввода и нажимает Enter (или при нажатии соответствующей кнопки). Сообщение немедленно появляется в общей истории и становится видимым всем участникам. Так же необходима система личных сообщений для, которыми пользователи смогут делиться друг с другом
* Приём и отображение. Новые сообщения других пользователей автоматически добавляются в ленту чата, упорядоченные по времени. Каждая запись содержит имя отправителя, метку времени и сам текст.
* Отправка файлов. Чат должен иметь поддержку отправки файлов в личном диалоге с пользователем. При получении файла пользователь будет иметь возможность посмотреть путь, в который сохранен файл, а также сразу нажать на него для открытия.
* Уведомления. При получении новых сообщений окно чата может воспроизводить звук, чтобы привлечь внимание, если пользователь находится в другом приложении или чате.

Такой чат служит основой для командной работы, поддержки пользователей и любых ситуаций, где требуется мгновенная текстовая коммуникация. UML-диаграмма вариантов использования приложения представлена в приложении Б.1.

1. Описание клиент-серверной части

Приложение представляет собой классическую клиент‑серверную систему чата. Сервер — отдельный процесс, клиент — графическое Swing‑приложение, запускающееся с формы авторизации, а затем открывающее основное окно чата. После старта клиент показывает форму авторизации или регистрации. Пользователь вводит логин и пароль. Клиент открывает TCP‑соединение на `localhost:1234` и отправляет данные. Сервер принимает соединение, читает данные, проверяет базу данных и отвечает строкой‑кодом результата. При успешном входе («OK») сервер сохраняет сокет, создаёт объект  «ClientHandler», помещает его в потокобезопасный список и запускает его в отдельном потоке.

Сетевой протокол полностью основан на TCP. Вся сессия пользователя держится в одном соединении. Формат обмена — простые текстовые маркеры (LOGIN, FILE, USERLIST и т. д.) плюс бинарный поток, когда передаётся файл. Взаимодействие клиента и сервера представлено в UML-диаграмме последовательности (приложение Б.2), а также в UML-диаграмме развертывания (приложение Б.5).

1. Описание программы
   1. Файл Sever.java

Server.java — это главный исполняемый файл сервера. Он поднимает TCP‑службу и содержит вложенный класс‑обработчик для каждого клиента. На рисунке 1 представлена работа сервера.

Сначала в методе main создаётся ServerSocket на порту 1234 и выводится сообщение «Server started…». В бесконечном цикле принимаются новые соединения, каждое из них передаётся в метод authenticateAndHandle и обрабатывается в отдельном потоке, чтобы подключение следующих клиентов не блокировалось. Метод «authenticateAndHandle» получает символьные потоки ввода‑вывода поверх сокета, читает команду «LOGIN» или «REGISTER».

Для LOGIN он вызывает «checkCredentials», сверяя данные с таблицей «users» в базе SQLite «users.db». Если пара верна, клиенту посылается «OK», создаётся объект ClientHandler, который помещается в потокобезопасный список клиентов и запускается новый поток. В противном случае клиенту отправляется «FAIL».

Для REGISTER сервер проверяет, существует ли уже такой логин (метод «userExists»), если нет, добавляет запись в БД, отвечает «REGISTERED», иначе «EXISTS».

Вложенный класс ClientHandler обслуживание одного пользователя. Главный цикл метода «run» читает от клиента строки. Он обрабатывает команды GET\_HISTORY, FILE и обычный текст:

GET\_HISTORY вызывает «sendHistory» — читает нужный файл‑лог (общий или личный) и отправляет клиенту блок, ограниченный метками.

FILE получает имя адресата, название и размер файла, считывает байты и пересылает их только указанному получателю.

Текстовое сообщение: сначала читается, кому оно адресовано (общий чат или конкретный пользователь), потом сам текст. Сообщение передаётся либо через «broadcast» (для общего чата), либо через «sendPrivateMessage». Обе функции рассылают строку всем или конкретному пользователю, а также пишут ее в общий или личный лог. Запись в лог выполняет «logMessage», функция добавляет тайм‑штамп, проверяет хеш строки, чтобы исключить дубликаты, определяет правильный файл (общий или приватный) и дописывает строку. [3]

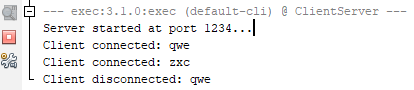


Рисунок 1 – Файл Server.java

* 1. Файл Login.java

Это окно, с которого начинается знакомство пользователя с мессенджером. Как видно из рисунка 2, это компактная форма, в которой находятся: два поля ввода (логин и пароль) и пара кнопок  «Вход» и «Регистрация». За этой формой закреплено несколько важных механик, которые обеспечивают корректную авторизацию и первичную инициализацию клиентской части.

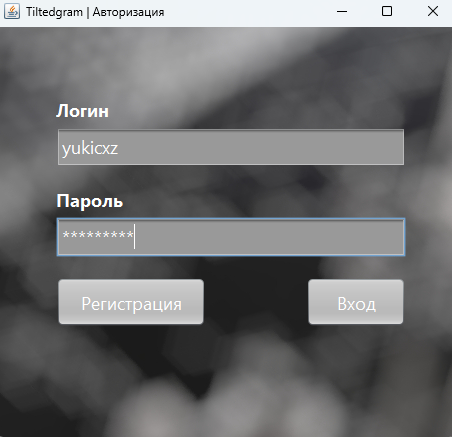


Рисунок 2 – Файл Login.java

После клика на кнопку «Регистрация» обработчик «jRegActionPerformed()» считывает значения полей логина и пароля, затем отправляя их с соответствующей командой на сервер, который проверяет, есть ли такой пользователь (SQL‑запрос к SQLite). Если логин свободен, сервер отвечает REGISTERED — клиент выводит сообщение «Регистрация успешна!», а если логин уже занят, приходит EXISTS — появляется алерт «Пользователь уже существует». На рисунке 3 приведен пример успешной регистрации пользователя.

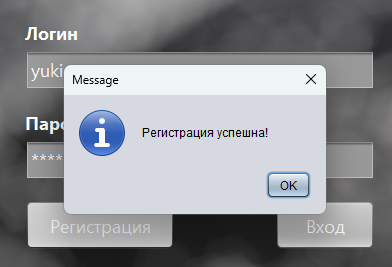


Рисунок 3 – Успешная регистрация

За авторизацию существующего аккаунта Кнопка «Вход», которая запускает метод «jLoginActionPerformed», где логика аналогична, но с другим командным словом (LOGIN). Если ответ от сервера OK, значит пароль подошел. После этого текущий JFrame сворачивается, создаётся экземпляр класса «chat» и ему передаются уже открытый Socket, а также те же DataInputStream/DataOutputStream, чтобы не было лишних соединений, а в конце новое окно чата становится видимым.

В случае, пара логин/пароль неверная сервер отправляет FAIL, а на клиенте всплывает сообщение «Неверный логин или пароль».

* 1. Файл Chat.java.

Данный файл можно назвать «сердцем» клиентской части мессенджера, графическое окно, с которым пользователь взаимодействует львиную долю времени. Сразу после успешной авторизации программа разворачивает на экране форму (рисунок 4): по центру — широкое текстовое поле , в котором будет прокручиваться вся переписка; справа — список активных пользователей, снизу — привычная связка из поля ввода и двух кнопок (одна, чтобы прикрепить файл, а вторая — чтобы отправить сообщение). Так же для удобства пользователя в верхней панели всегда отображается его никнейм и выбранный диалог.

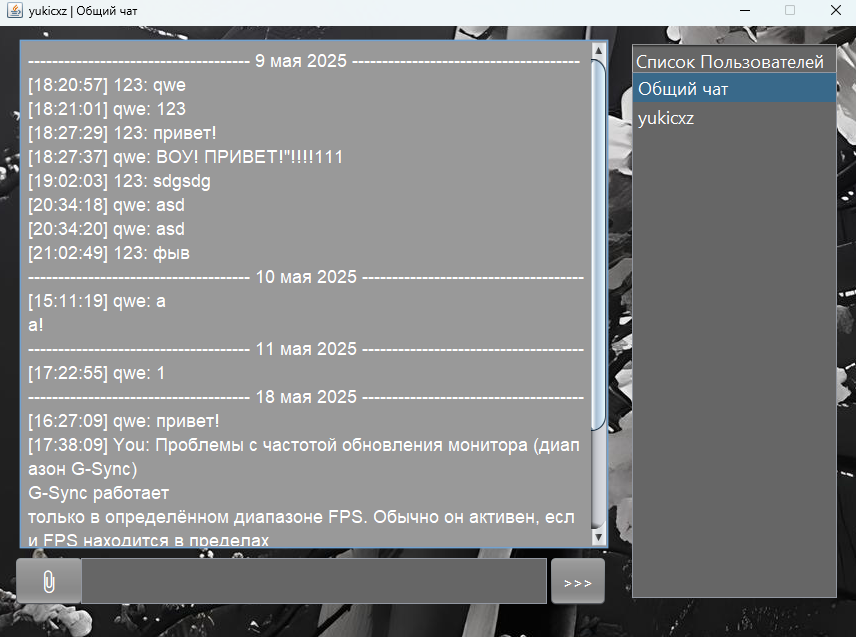


Рисунок 4 – Файл Chat.java

Как только пользователь кликает на конкретного собеседника или на пункт «Общий чат», срабатывает небольшой слушатель выбора: он формирует строку  «GET\_HISTORY» <адресат> и тут же кидает её серверу. Он, в свою очередь, собирает нужный лог из файловой системы, ставит сверху заголовок «HISTORY», построчно передаёт накопленные строки и закрывает блок маркером «---END---». Клиент, поймав начало блока, обнуляет текущий текст в окне чата, а затем выводит его в диалоговое окно.

Кнопка «Отправить» забирает текст из поля, формирует тройку строк и последовательно кидает их на сервер:

* первая строка указывает адресата (ALL либо ник другого пользователя);
* вторая содержит текст сообщения;
* третья сигнализирует о завершении сообщения и содержит метку «---END---», показывая, что блок закончен.

После успешной отправки поле ввода просто очищается — пользователь сразу видит, что текст отправлен.

При нажатии «скрепки» всплывает стандартный JFileChooser. Пользователь выбирает документ, картинку или архив — всё что угодно. Если выбран «Общий чат» или адресат совпадает с вашим собственным ником, клиент выводит небольшое предупреждение и отменяет процесс, чтобы не отправлять файл самому себе или в общий канал. В личных диалогах клиент формирует заголовочный блок, состоящий из команды «FILE», получателя, имени и размера файла.

Взаимодействие всех классов программы можно увидеть на UML-диаграмме классов (приложение Б.4).

1. Руководство пользователя

После запуска сервера и клиентского приложения пользователя встречает окно авторизации (представленное ранее на рисунке 2) в котором новому пользователю предлагается в начале пройти регистрацию, а затем выполнить вход. При регистрации пользователь не может выбрать уже зарегистрированное имя, а также имя и пароль короче 3-х символов. После входа его встречает основное окно чата. В нем отображаются сообщения в выбранном чате, пользователи, которые находятся онлайн, а также две основные кнопки взаимодействия и поле для сообщения. В списке активных пользователь так же находится общий чат, куда автоматически добавляются все новые пользователи.

По желанию пользователь может выбрать нужного получателя и переключится на него. При этом окно сообщений обновится под текущего получателя и загрузит их историю сообщений. Если она отсутствует, то вверху диалога будет выведена текущая дата, на рисунке 5 представлена данная ситуация.

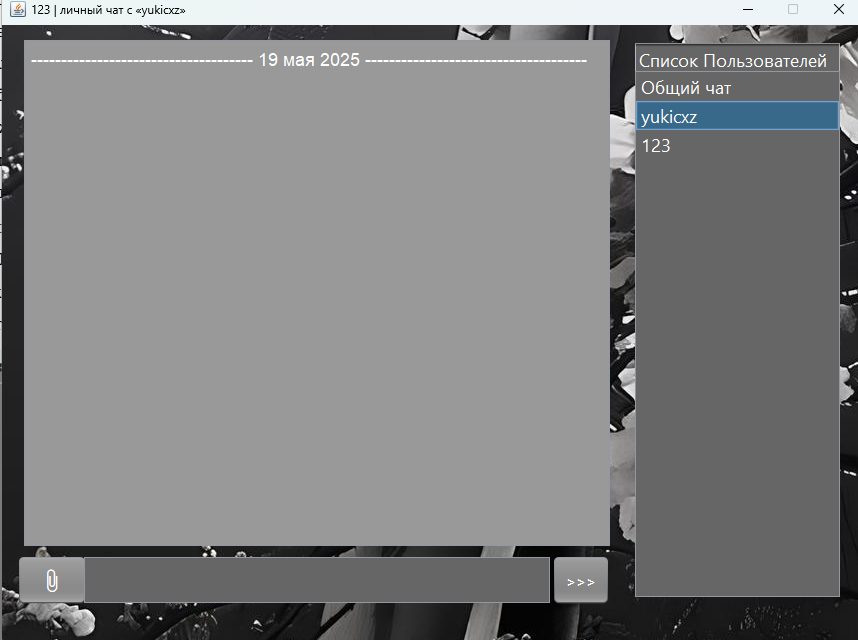


Рисунок 5 – Пустая переписка

На рисунке 6 показано, что при желании пользователь может выбрать себя и использовать диалог с собой как аналог «Избранного» в Telegram.

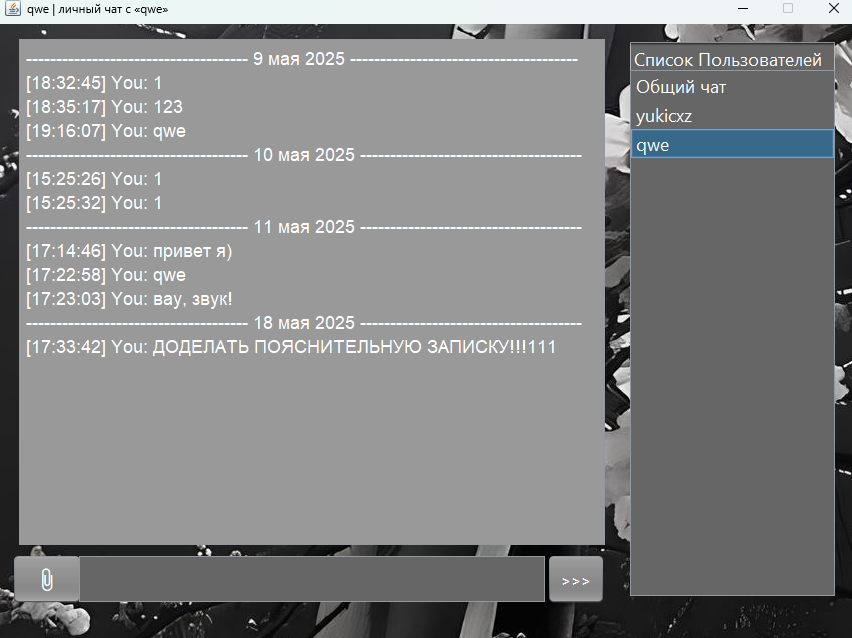


Рисунок 6 – Переписка с самим собой

При вводе сообщения в текстовое поле, пользователь должен нажать на кнопку отправки, чтобы сообщение дошло до получателя. Так же поле ввода не ограничено одной строкой, те при желании пользователь может отправлять многострочные сообщения. Варианты многострочных сообщений приведены на рисунке 7.

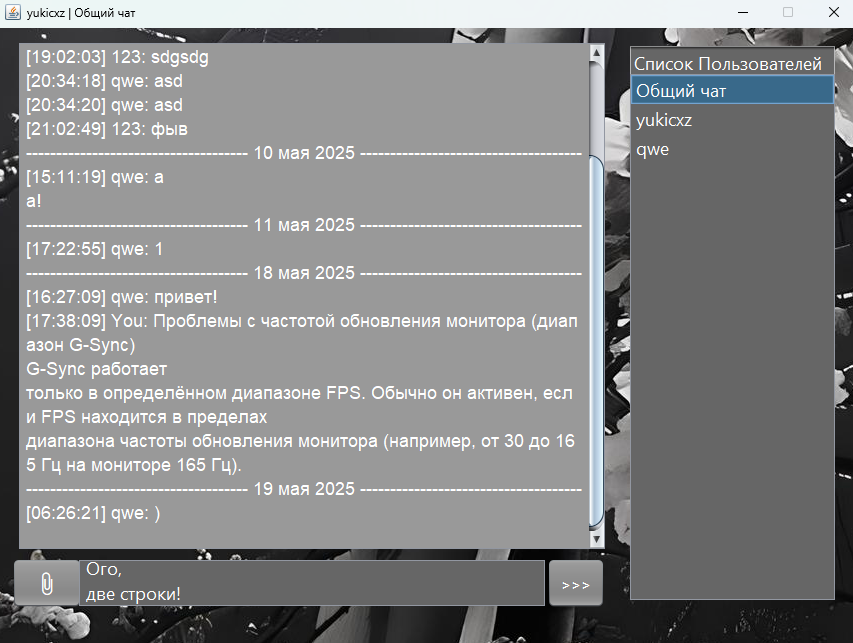


Рисунок 7 – Многострочность

При успешной отправке файла пользователь увидит лишь небольшое сообщение, а при получении увидит, что пришел файл и сохранился по такому-то пути, а также сможет нажать на него, для открытия. Рисунки 8-9 наглядно показывают данную вариацию использования.

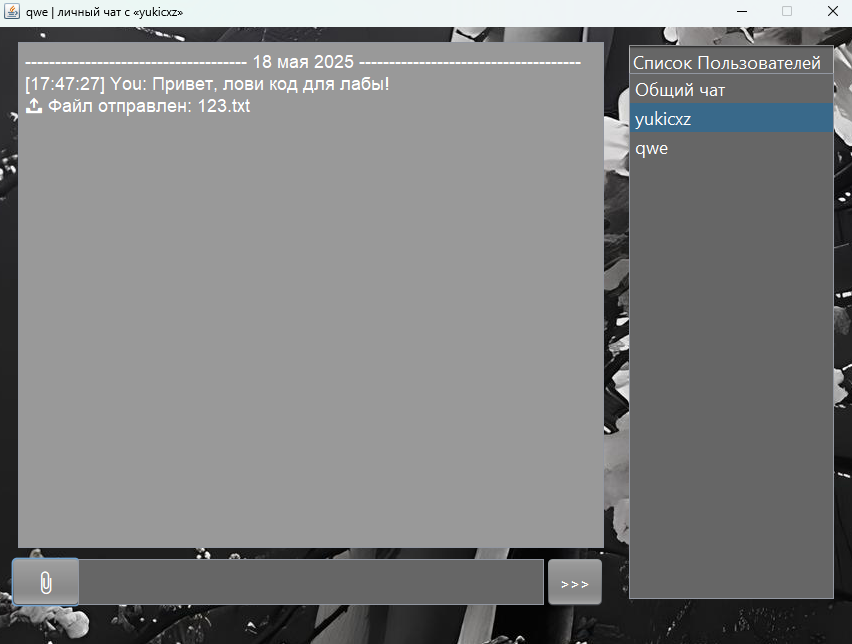


Рисунок 8 – Файл отправлен

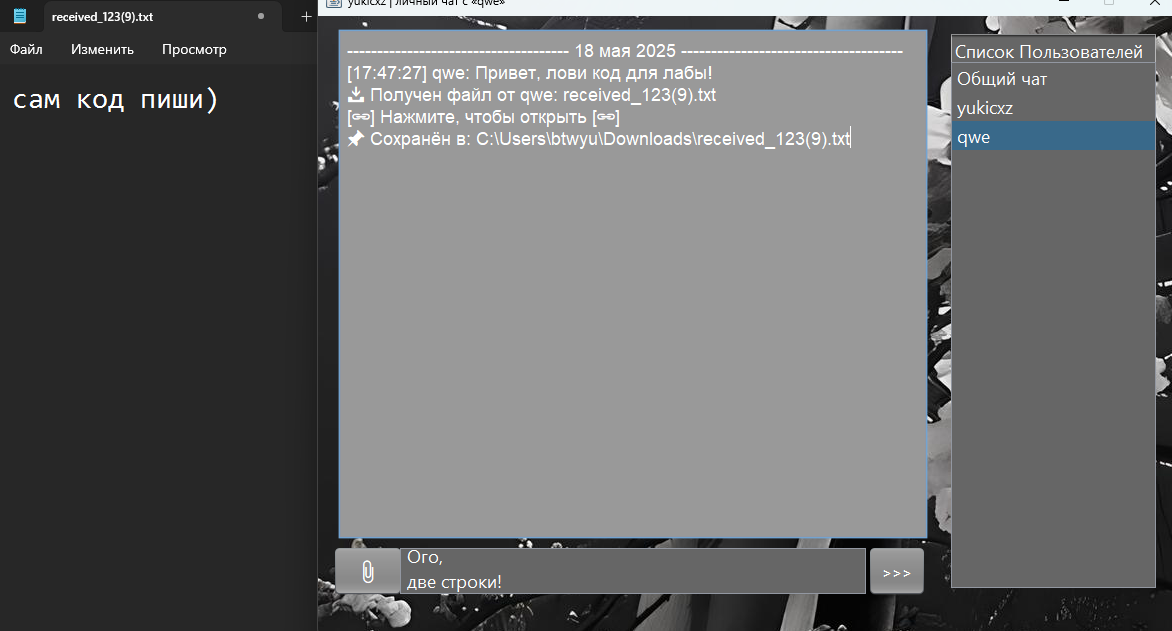


Рисунок 9 – Файл получен

Заключение

При выполнении данного курсового проекта были изучены принципы работы протоколов TCP/IP. Были получены навыки разработки многомодульного приложения с пользовательским интерфейсом и работы в среде NetBeans. Получены навыки разработки клиент-серверных приложений.

В результате выполнения данной курсовой работы было разработано приложение «Текстовый чат», позволяющее пользователям общаться друг с другом лично, отправлять сообщения всем пользователям, а также выполнять отправку файлов.

Список используемых источников

* "Java. Методы программирования" И.Н Блинов, В.С. Романчик – Минск: УниверсалПресс, 2007 – 704 с.
* "Java. Руководство для начинающих", Герберт Шилдт – ООО «Диалектика», 2019 – 816 с.
* Форум программистов и сисадминов Киберфорум [Электронный ресурс]. Режим доступа: <http://www.cyberforum.ru/>, свободный (Дата обращения 29.04.2025).

# **Приложение A.** Листинг программы

## Приложение A.1. Файл Server.java

package com.mycompany.clientserver;

import java.io.\*;

import java.net.\*;

import java.sql.\*;

import java.text.SimpleDateFormat;

import java.time.LocalDate;

import java.time.LocalTime;

import java.time.format.DateTimeFormatter;

import java.util.\*;

import java.util.concurrent.CopyOnWriteArrayList;

import java.util.Date;

public class Server {

// Список всех клиентов (потокобезопасный)

private static List<ClientHandler> clients = new CopyOnWriteArrayList<>();

public static void main(String[] args) throws IOException {

ServerSocket serverSocket = new ServerSocket(1234);

System.out.println("Server started at port 1234...");

while (true) {

Socket socket = serverSocket.accept();

new Thread(() -> authenticateAndHandle(socket)).start();

}

}

private static void authenticateAndHandle(Socket socket) {

try {

DataInputStream dis = new DataInputStream(socket.getInputStream());

DataOutputStream dos = new DataOutputStream(socket.getOutputStream());

String command = dis.readUTF(); // LOGIN или REGISTER

String login = dis.readUTF();

String password = dis.readUTF();

if ("LOGIN".equalsIgnoreCase(command)) {

if (checkCredentials(login, password)) {

dos.writeUTF("OK");

dos.flush();

ClientHandler clientHandler = new ClientHandler(socket, login);

clients.add(clientHandler);

new Thread(clientHandler).start();

System.out.println("Client connected: " + login);

} else {

dos.writeUTF("FAIL");

dos.flush();

}

} else if ("REGISTER".equalsIgnoreCase(command)) {

if (userExists(login)) {

dos.writeUTF("EXISTS");

dos.flush();

} else {

registerUser(login, password);

dos.writeUTF("REGISTERED");

dos.flush();

}

}

} catch (IOException e) {

e.printStackTrace();

}

}

private static boolean userExists(String login) {

try (Connection conn = DriverManager.getConnection("jdbc:sqlite:db/users.db");

PreparedStatement stmt = conn.prepareStatement("SELECT 1 FROM users WHERE username = ?")) {

stmt.setString(1, login);

return stmt.executeQuery().next();

} catch (SQLException e) {

e.printStackTrace();

return false;

}

}

private static void registerUser(String login, String password) {

try (Connection conn = DriverManager.getConnection("jdbc:sqlite:db/users.db");

PreparedStatement stmt = conn.prepareStatement("INSERT INTO users (username, password) VALUES (?, ?)")) {

stmt.setString(1, login);

stmt.setString(2, password);

stmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

private static boolean checkCredentials(String login, String password) {

try (Connection conn = DriverManager.getConnection("jdbc:sqlite:db/users.db");

PreparedStatement stmt = conn.prepareStatement("SELECT \* FROM users WHERE username = ? AND password = ?")) {

stmt.setString(1, login);

stmt.setString(2, password);

return stmt.executeQuery().next();

} catch (SQLException e) {

e.printStackTrace();

return false;

}

}

// Класс, отвечающий за отдельного клиента

private static class ClientHandler implements Runnable {

private Socket socket;

private String username;

private DataInputStream dis;

private DataOutputStream dos;

public ClientHandler(Socket socket, String username) throws IOException {

this.socket = socket;

this.username = username;

this.dis = new DataInputStream(socket.getInputStream());

this.dos = new DataOutputStream(socket.getOutputStream());

}

private static final Set<String> loggedMessages = Collections.synchronizedSet(new HashSet<>());

private String lastSentDate = "";

private String formatDateHeader(String formattedDate) {

int totalLength = 87;

String dateCenter = " " + formattedDate + " ";

int dashes = (totalLength - dateCenter.length()) / 2;

StringBuilder sb = new StringBuilder();

sb.append("-".repeat(Math.max(0, dashes)));

sb.append(dateCenter);

sb.append("-".repeat(Math.max(0, totalLength - sb.length()))); // добиваем справа

return sb.toString();

}

private void logMessage(String sender, String recipient, String message) {

String timestamp = new java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new java.util.Date());

String fullMessage = String.format("[%s] %s", timestamp, message);

String hash = Integer.toString(fullMessage.hashCode());

if (loggedMessages.contains(hash)) return;

loggedMessages.add(hash);

String filename;

if ("ALL".equalsIgnoreCase(recipient)) {

filename = "history/ALL/chat\_log.txt";

} else {

String user1 = sender;

String user2 = recipient;

if (user1.compareTo(user2) > 0) {

String temp = user1;

user1 = user2;

user2 = temp;

}

filename = String.format("history/private/%s\_%s.txt", user1, user2);

}

try (PrintWriter logOut = new PrintWriter(new FileWriter(filename, true))) {

logOut.println(fullMessage);

} catch (IOException e) {

e.printStackTrace();

}

}

private void sendHistory(String recipient) {

String filename;

if (recipient.equalsIgnoreCase("ALL")) {

filename = "history/ALL/chat\_log.txt";

} else {

String user1 = username;

String user2 = recipient;

if (user1.compareTo(user2) > 0) {

String temp = user1;

user1 = user2;

user2 = temp;

}

filename = String.format("history/private/%s\_%s.txt", user1, user2);

}

File file = new File(filename);

boolean headerPrinted = false;

try (BufferedReader reader = file.exists() ? new BufferedReader(new FileReader(file)) : null) {

dos.writeUTF("HISTORY");

String lastDate = "";

String line;

if (reader != null) {

while ((line = reader.readLine()) != null) {

if (!line.startsWith("[")) continue;

String dateTimeStr = line.substring(1, 20);

LocalDate date = LocalDate.parse(dateTimeStr.substring(0, 10));

LocalTime time = LocalTime.parse(dateTimeStr.substring(11));

String rest = line.substring(22);

String formattedDate = date.format(DateTimeFormatter.ofPattern("d MMMM yyyy", new Locale("ru")));

if (!formattedDate.equals(lastDate)) {

lastDate = formattedDate;

dos.writeUTF(formatDateHeader(formattedDate));

headerPrinted = true;

this.lastSentDate = formattedDate;

}

String formatted = String.format("[%s] %s", time.toString(), rest);

dos.writeUTF(formatted);

}

}

// Если ничего не было отправлено — добавим текущую дату

if (!headerPrinted) {

String today = LocalDate.now().format(DateTimeFormatter.ofPattern("d MMMM yyyy", new Locale("ru")));

dos.writeUTF(formatDateHeader(today));

this.lastSentDate = today;

}

dos.writeUTF("---END---");

} catch (IOException e) {

e.printStackTrace();

}

}

public void run() {

sendUserListToAll();

try {

while (true) {

String line = dis.readUTF();

if (line.startsWith("GET\_HISTORY")) {

String target = line.substring("GET\_HISTORY".length()).trim();

sendHistory(target);

continue;

}

if (line.equals("FILE")) {

String recipient = dis.readUTF();

String filename = dis.readUTF();

long fileSize = dis.readLong();

byte[] fileData = new byte[(int) fileSize];

dis.readFully(fileData);

for (ClientHandler client : clients) {

if (client.username.equals(recipient)) {

client.dos.writeUTF("FILE");

client.dos.writeUTF(username);

client.dos.writeUTF(filename);

client.dos.writeLong(fileSize);

client.dos.write(fileData);

client.dos.flush();

break;

}

}

continue;

}

String recipient = line;

StringBuilder messageBuilder = new StringBuilder();

String msgLine;

while (!(msgLine = dis.readUTF()).equals("---END---")) {

messageBuilder.append(msgLine).append("\n");

}

String message = messageBuilder.toString().trim();

if (recipient.equalsIgnoreCase("ALL")) {

broadcast(username + ": " + message);

} else {

sendPrivateMessage(recipient, username + ": " + message);

}

}

} catch (IOException e) {

System.out.println("Client disconnected: " + username);

} finally {

try { socket.close(); } catch (IOException e) {}

clients.remove(this);

sendUserListToAll();

}

}

private void broadcast(String messageText) throws IOException {

LocalDate now = LocalDate.now();

String currentDateFormatted = now.format(DateTimeFormatter.ofPattern("d MMMM yyyy", new Locale("ru")));

String time = new SimpleDateFormat("HH:mm:ss").format(new Date());

String content = String.format("[%s] %s", time, messageText);

for (ClientHandler client : clients) {

if (!currentDateFormatted.equals(client.lastSentDate)) {

client.lastSentDate = currentDateFormatted;

client.dos.writeUTF(formatDateHeader(currentDateFormatted));

}

client.dos.writeUTF(content);

}

logMessage(username, "ALL", messageText);

}

private void sendPrivateMessage(String recipient, String messageText) throws IOException {

LocalDate now = LocalDate.now();

String currentDateFormatted = now.format(DateTimeFormatter.ofPattern("d MMMM yyyy", new Locale("ru")));

String time = new SimpleDateFormat("HH:mm:ss").format(new Date());

String content = String.format("[%s] %s", time, messageText);

for (ClientHandler client : clients) {

if (client.username.equals(recipient) || client.username.equals(this.username)) {

if (!currentDateFormatted.equals(client.lastSentDate)) {

client.lastSentDate = currentDateFormatted;

client.dos.writeUTF("PM:" + username + ":" + formatDateHeader(currentDateFormatted));

}

String direction = client.username.equals(this.username) ? recipient : username;

client.dos.writeUTF("PM:" + direction + ":" + content);

}

}

logMessage(username, recipient, messageText);

}

private void sendUserListToAll() {

for (ClientHandler client : clients) {

try {

client.dos.writeUTF("USERLIST");

clients.stream() //создаёт поток из списка клиентов;

.map(c -> c.username) //преобразует каждый ClientHandler в имя пользователя (String)

.forEach(name -> {

try {

client.dos.writeUTF(name);

} catch (IOException e) {

e.printStackTrace();

}

});

client.dos.writeUTF("---END---");

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

}

## Приложение A.2. Файл Login.java

package com.mycompany.clientserver;

import java.io.IOException;

import java.net.Socket;

import java.io.DataInputStream;

import java.io.DataOutputStream;

/\*\*

\*

\* @author btwyu

\*/

public class login extends javax.swing.JFrame {

/\*\*

\* Creates new form mainform

\*/

public login() {

initComponents();

setTitle("Tiltedgram | Авторизация");

}

/\*\*

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

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

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

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

private void initComponents() {

jReg = new javax.swing.JButton();

tlogin = new javax.swing.JTextField();

jPassword = new javax.swing.JPasswordField();

jLogin = new javax.swing.JButton();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

jLabel3 = new javax.swing.JLabel(); setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

jReg.setBackground(new java.awt.Color(153, 153, 153));

jReg.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N

jReg.setForeground(new java.awt.Color(255, 255, 255));

jReg.setText("Регистрация");

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

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

jRegActionPerformed(evt);

}

});

getContentPane().add(jReg, new org.netbeans.lib.awtextra.AbsoluteConstraints(60, 250, 150, 50));

tlogin.setBackground(new java.awt.Color(153, 153, 153));

tlogin.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N

tlogin.setForeground(new java.awt.Color(255, 255, 255));

getContentPane().add(tlogin, new org.netbeans.lib.awtextra.AbsoluteConstraints(60, 100, 350, 40));

jPassword.setBackground(new java.awt.Color(153, 153, 153));

jPassword.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N

jPassword.setForeground(new java.awt.Color(255, 255, 255));

getContentPane().add(jPassword, new org.netbeans.lib.awtextra.AbsoluteConstraints(60, 190, 350, 40));

jLogin.setBackground(new java.awt.Color(153, 153, 153));

jLogin.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N

jLogin.setForeground(new java.awt.Color(255, 255, 255));

jLogin.setText("Вход");

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

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

jLoginActionPerformed(evt);

}

});

getContentPane().add(jLogin, new org.netbeans.lib.awtextra.AbsoluteConstraints(310, 250, 100, 50));

jLabel1.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N

jLabel1.setForeground(new java.awt.Color(255, 255, 255));

jLabel1.setText("Логин");

getContentPane().add(jLabel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(60, 70, -1, -1));

jLabel2.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N

jLabel2.setForeground(new java.awt.Color(255, 255, 255));

jLabel2.setText("Пароль");

getContentPane().add(jLabel2, new org.netbeans.lib.awtextra.AbsoluteConstraints(60, 160, -1, -1));

jLabel3.setIcon(new javax.swing.ImageIcon(getClass().getResource("/photo\_2024-12-10\_16-28-24.jpg"))); // NOI18N

jLabel3.setText("jLabel3");

getContentPane().add(jLabel3, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 460, 410));

pack();

}// </editor-fold>

private boolean isValidInput(String text) {

return text != null && !text.trim().isEmpty() && text.matches("^[a-zA-Z0-9\_.-]{3,}$");

}

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

// TODO add your handling code here:

String login = tlogin.getText();

String password = new String(jPassword.getPassword());

if (!isValidInput(login) || !isValidInput(password)) {

javax.swing.JOptionPane.showMessageDialog(this, "Недопустимые логин или пароль. Разрешены латинские буквы, цифры, '\_', '-', '.', минимум 3 символа.");

return;

}

try {

Socket socket = new Socket("localhost", 1234);

DataOutputStream dos = new DataOutputStream(socket.getOutputStream());

DataInputStream dis = new DataInputStream(socket.getInputStream());

dos.writeUTF("REGISTER");

dos.writeUTF(login);

dos.writeUTF(password);

dos.flush();

String response = dis.readUTF();

switch (response) {

case "REGISTERED":

javax.swing.JOptionPane.showMessageDialog(this, "Регистрация успешна!");

break;

case "EXISTS":

javax.swing.JOptionPane.showMessageDialog(this, "Пользователь уже существует.");

break;

default:

javax.swing.JOptionPane.showMessageDialog(this, "Ошибка регистрации.");

}

socket.close();

} catch (IOException e) {

javax.swing.JOptionPane.showMessageDialog(this, "Ошибка подключения к серверу.");

e.printStackTrace();

}

}

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

// Получаем логин и пароль из полей ввода

String login = tlogin.getText();

String password = new String(jPassword.getPassword());

if (!isValidInput(login) || !isValidInput(password)) {

javax.swing.JOptionPane.showMessageDialog(this, "Недопустимые логин или пароль. Разрешены латинские буквы, цифры, '\_', '-', '.', минимум 3 символа.");

return;

}

try {

Socket socket = new Socket("localhost", 1234);

DataOutputStream dos = new DataOutputStream(socket.getOutputStream());

DataInputStream dis = new DataInputStream(socket.getInputStream());

dos.writeUTF("LOGIN");

dos.writeUTF(login);

dos.writeUTF(password);

dos.flush();

String response = dis.readUTF();

switch (response) {

case "OK":

javax.swing.JOptionPane.showMessageDialog(this, "Вход успешен!");

String username = login;

new chat(socket, dos, dis, username).setVisible(true);

this.dispose();

return;

case "FAIL":

javax.swing.JOptionPane.showMessageDialog(this, "Неверный логин или пароль.");

break;

default:

javax.swing.JOptionPane.showMessageDialog(this, "Ошибка при входе.");

break;

}

socket.close();

} catch (IOException e) {

javax.swing.JOptionPane.showMessageDialog(this, "Ошибка подключения к серверу.");

e.printStackTrace();

}

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

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

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

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

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

\*/

try {

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

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

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

break;

}

}

} catch (ClassNotFoundException ex) {

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

} catch (InstantiationException ex) {

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

} catch (IllegalAccessException ex) {

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

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

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

}

//</editor-fold>

/\* Create and display the form \*/

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

public void run() {

new login().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JButton jLogin;

private javax.swing.JPasswordField jPassword;

private javax.swing.JButton jReg;

private javax.swing.JTextField tlogin;

// End of variables declaration

}

## Приложение A.3. Файл Chat.java

package com.mycompany.clientserver;

import java.awt.Color;

import java.io.BufferedInputStream;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.net.Socket;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.regex.Pattern;

import javax.sound.sampled.AudioInputStream;

import javax.sound.sampled.AudioSystem;

import javax.sound.sampled.Clip;

import javax.swing.JFileChooser;

import javax.swing.SwingUtilities;

/\*\*

\*

\* @author btwyu

\*/

public class chat extends javax.swing.JFrame {

private Socket socket;

private DataOutputStream dos;

private DataInputStream dis;

private String username;

private String currentRecipient = "ALL"; // Общий чат по умолчанию

private final Map<String, List<String>> chatHistory = new HashMap<>(); //мапа для истории чата

private final Map<Integer, File> clickableFiles = new HashMap<>(); //мапа для отслеживания строки

/\*\*

\* Creates new form chat

\*/

public chat(Socket socket, DataOutputStream dos, DataInputStream dis, String username){

this.socket = socket;

this.dos = dos;

this.dis = dis;

this.username = username;

initComponents();

setTitle("Чат | Вы: «" + username + "»");

setResizable(false);

setupUserListSelection(); // обработка выбора получателя

startListening();

}

private void setupUserListSelection() {

jUsers1.addListSelectionListener(e -> {

if (!e.getValueIsAdjusting()) {

String selected = jUsers1.getSelectedValue();

if (selected != null) {

currentRecipient = selected.equalsIgnoreCase("Общий чат") ? "ALL" : selected;

// ------ динамический заголовок окна ------

if ("ALL".equals(currentRecipient)) {

setTitle(username + " | Общий чат");

} else {

setTitle(username + " | личный чат с «" + currentRecipient + "»");

}

// Очистим окно чата сразу

SwingUtilities.invokeLater(() -> jChat.setText("Загрузка истории...\n"));

// Запрашиваем историю у сервера

try {

dos.writeUTF("GET\_HISTORY " + currentRecipient);

dos.flush();

} catch (IOException ex) {

ex.printStackTrace();

}

}

}

});

}

/\*\*

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

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

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

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

private void initComponents() {

jScrollPane2 = new javax.swing.JScrollPane();

jMessage = new javax.swing.JTextArea();

jScrollPane1 = new javax.swing.JScrollPane();

jChat = new javax.swing.JTextArea();

jUserList = new javax.swing.JScrollPane();

jUsers1 = new javax.swing.JList<>();

jSend = new javax.swing.JButton();

jTextField1 = new javax.swing.JTextField();

jFile = new javax.swing.JButton();

jLabel1 = new javax.swing.JLabel();

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

setMaximumSize(new java.awt.Dimension(856, 503));

setMinimumSize(new java.awt.Dimension(856, 503));

getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

jScrollPane2.setHorizontalScrollBarPolicy(javax.swing.ScrollPaneConstants.HORIZONTAL\_SCROLLBAR\_NEVER); jScrollPane2.setVerticalScrollBarPolicy(javax.swing.ScrollPaneConstants.VERTICAL\_SCROLLBAR\_NEVER);

jMessage.setBackground(new java.awt.Color(102, 102, 102));

jMessage.setColumns(20);

jMessage.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N

jMessage.setForeground(new java.awt.Color(255, 255, 255));

jMessage.setRows(5);

jScrollPane2.setViewportView(jMessage);

getContentPane().add(jScrollPane2, new org.netbeans.lib.awtextra.AbsoluteConstraints(80, 530, 470, 50));

jChat.setEditable(false);

jChat.setBackground(new java.awt.Color(153, 153, 153));

jChat.setColumns(20);

jChat.setFont(jChat.getFont().deriveFont(jChat.getFont().getSize()+6f));

jChat.setForeground(new java.awt.Color(255, 255, 255));

jChat.setLineWrap(true);

jChat.setRows(5);

jChat.addMouseMotionListener(new java.awt.event.MouseMotionAdapter() {

public void mouseMoved(java.awt.event.MouseEvent evt) {

jChatMouseMoved(evt);

}

});

jChat.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jChatMouseClicked(evt);

}

});

jScrollPane1.setViewportView(jChat);

getContentPane().add(jScrollPane1, new org.netbeans.lib.awtextra.AbsoluteConstraints(20, 13, 590, 510));

jUsers1.setBackground(new java.awt.Color(102, 102, 102));

jUsers1.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N

jUsers1.setForeground(new java.awt.Color(255, 255, 255));

jUserList.setViewportView(jUsers1);

getContentPane().add(jUserList, new org.netbeans.lib.awtextra.AbsoluteConstraints(631, 44, 209, 530));

jSend.setBackground(new java.awt.Color(102, 102, 102));

jSend.setFont(new java.awt.Font("Segoe UI", 1, 14)); // NOI18N

jSend.setForeground(new java.awt.Color(255, 255, 255));

jSend.setText(">>>");

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

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

jSendActionPerformed(evt);

}

});

getContentPane().add(jSend, new org.netbeans.lib.awtextra.AbsoluteConstraints(550, 530, -1, 50));

jSend.getAccessibleContext().setAccessibleName("jSend");

jTextField1.setEditable(false);

jTextField1.setBackground(new java.awt.Color(102, 102, 102));

jTextField1.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N

jTextField1.setForeground(new java.awt.Color(255, 255, 255));

jTextField1.setText("Список Пользователей");

getContentPane().add(jTextField1, new org.netbeans.lib.awtextra.AbsoluteConstraints(631, 16, 209, -1));

jFile.setBackground(java.awt.SystemColor.controlDkShadow);

jFile.setFont(jFile.getFont().deriveFont(jFile.getFont().getStyle() & ~java.awt.Font.BOLD, 22));

jFile.setForeground(new java.awt.Color(255, 255, 255));

jFile.setText("📎");

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

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

jFileActionPerformed(evt);

}

});

getContentPane().add(jFile, new org.netbeans.lib.awtextra.AbsoluteConstraints(15, 530, 70, 50));

jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/123.jpg"))); // NOI18N

jLabel1.setLabelFor(this);

jLabel1.setToolTipText("");

getContentPane().add(jLabel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(-490, 0, 1350, 620));

pack();

}// </editor-fold>

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

// TODO add your handling code here:

String message = jMessage.getText();

if (!message.isEmpty()) {

message = message.replace("\n", "\\n");

try {

dos.writeUTF(currentRecipient);

dos.writeUTF(message);

dos.writeUTF("---END---");

dos.flush();

jMessage.setText("");

} catch (IOException ex) {

ex.printStackTrace();

jChat.append("❌ Ошибка при отправке сообщения\n");

}

}

}

private void sendFile(File file) {

if (currentRecipient.equals("ALL")) {

SwingUtilities.invokeLater(() -> jChat.append("⚠️ Нельзя отправлять файлы в общий чат.\n"));

return;

}

if (currentRecipient.equals(username)) {

SwingUtilities.invokeLater(() -> jChat.append("⚠️ Нельзя отправлять файлы самому себе.\n"));

return;

}

try {

// Отправка заголовка

dos.writeUTF("FILE");

dos.writeUTF(currentRecipient);

dos.writeUTF(file.getName());

dos.writeLong(file.length());

dos.flush();

// Отправка содержимого файла

try (FileInputStream fis = new FileInputStream(file)) {

byte[] buffer = new byte[4096];

int bytesRead;

while ((bytesRead = fis.read(buffer)) != -1) {

dos.write(buffer, 0, bytesRead);

}

dos.flush();

}

SwingUtilities.invokeLater(() -> jChat.append("📤 Файл отправлен: " + file.getName() + "\n"));

} catch (IOException ex) {

ex.printStackTrace();

SwingUtilities.invokeLater(() -> jChat.append("❌ Ошибка при отправке файла\n"));

}

}

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

// TODO add your handling code here:

JFileChooser fileChooser = new JFileChooser();

int result = fileChooser.showOpenDialog(this);

if (result == JFileChooser.APPROVE\_OPTION) {

File file = fileChooser.getSelectedFile();

sendFile(file);

}

}

private void jChatMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

try {

int clickedOffset = jChat.viewToModel2D(evt.getPoint());

int clickedLine = jChat.getLineOfOffset(clickedOffset);

File file = clickableFiles.get(clickedLine);

if (file != null && file.exists()) {

java.awt.Desktop.getDesktop().open(file);

}

} catch (Exception ex) {

ex.printStackTrace();

}

}

private void jChatMouseMoved(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

try {

int offset = jChat.viewToModel2D(evt.getPoint());

int line = jChat.getLineOfOffset(offset);

if (clickableFiles.containsKey(line)) {

jChat.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

} else {

jChat.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT\_CURSOR));

}

} catch (Exception e) {

jChat.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT\_CURSOR));

}

}

/\*\*

\* @param args the command line arguments

\*/

private String wrapText(String text, int maxLineLength) {

StringBuilder result = new StringBuilder();

int lineLength = 0;

for (String word : text.split(" ")) {

if (lineLength + word.length() > maxLineLength) {

result.append("\n");

lineLength = 0;

} else if (lineLength > 0) {

result.append(" ");

lineLength++;

}

result.append(word);

lineLength += word.length();

}

return result.toString();

}

private void playNotificationSound() {

try {

InputStream audioSrc = getClass().getResourceAsStream("/ding.wav");

if (audioSrc == null) return;

InputStream bufferedIn = new BufferedInputStream(audioSrc);

AudioInputStream audioStream = AudioSystem.getAudioInputStream(bufferedIn);

Clip clip = AudioSystem.getClip();

clip.open(audioStream);

clip.start();

} catch (Exception e) {

e.printStackTrace();

}

}

private void startListening() {

new Thread(() -> {

try {

List<String> userList = new ArrayList<>();

userList.add("Общий чат");

while (true) {

String line = dis.readUTF();

/\* ──────────────── И С Т О Р И Я ──────────────── \*/

if (line.equals("HISTORY")) {

List<String> history = new ArrayList<>();

while (!(line = dis.readUTF()).equals("---END---")) {

history.add(line);

}

chatHistory.put(currentRecipient, history);

SwingUtilities.invokeLater(() -> {

jChat.setText("");

for (String msg : history) {

// ▸ подмена имени на You:

String out = msg.contains(username + ":")

? msg.replaceFirst(Pattern.quote(username) + ":", "You:")

: msg;

String wrapped = wrapText(out.replace("\\n", "\n"), 90);

jChat.append(wrapped + "\n");

}

});

continue;

}

/\* ──────────────── С П И С О К ─ П О Л Ь З. ──────────────── \*/

if (line.startsWith("USERLIST")) {

userList.clear();

userList.add("Общий чат");

while (!(line = dis.readUTF()).equals("---END---")) {

userList.add(line);

}

SwingUtilities.invokeLater(() -> {

jUsers1.setListData(userList.toArray(new String[0]));

jUsers1.setSelectedIndex(0);

});

continue;

}

/\* ──────────────── Ф А Й Л Ы ──────────────── \*/

if (line.equals("FILE")) {

String sender = dis.readUTF();

String filename = dis.readUTF();

long fileSize = dis.readLong();

File downloadsFolder = new File(System.getProperty("user.home"), "Downloads");

if (!downloadsFolder.exists()) downloadsFolder.mkdirs();

/\* авто‑переименование received\_Имя(1).ext \*/

String base = filename, ext = "";

int dot = filename.lastIndexOf('.');

if (dot != -1) { base = filename.substring(0, dot); ext = filename.substring(dot); }

File received = new File(downloadsFolder, "received\_" + filename);

for (int i = 1; received.exists(); i++)

received = new File(downloadsFolder, String.format("received\_%s(%d)%s", base, i, ext));

try (FileOutputStream fos = new FileOutputStream(received)) {

byte[] buf = new byte[4096];

long left = fileSize;

while (left > 0) {

int read = dis.read(buf, 0, (int)Math.min(buf.length, left));

if (read == -1) throw new IOException("End of stream during file");

fos.write(buf, 0, read);

left -= read;

}

}

int lineNum = jChat.getLineCount();

clickableFiles.put(lineNum, received);

File f = received; // effectively final для лямбды

SwingUtilities.invokeLater(() -> {

jChat.append("📥 Получен файл от " + sender + ": " + f.getName() +

"\n[🔗] Нажмите, чтобы открыть [🔗]\n");

jChat.append("📌 Сохранён в: " + f.getAbsolutePath() + "\n");

playNotificationSound();

});

continue;

}

/\* ──────────────── Т Е К С Т О В Ы Е ──────────────── \*/

String msg = line;

String chatKey;

String displayMsg;

if (msg.startsWith("PM:")) {

String[] parts = msg.split(":", 3);

if (parts.length == 3) {

String otherUser = parts[1];

displayMsg = parts[2];

chatKey = otherUser;

} else {

displayMsg = msg;

chatKey = "ALL";

}

} else {

displayMsg = msg;

chatKey = "ALL";

}

/\* ▸ подмена имени текущего пользователя на You: \*/

if (displayMsg.contains(username + ":")) {

displayMsg = displayMsg.replaceFirst(Pattern.quote(username) + ":", "You:");

}

/\* перенос строк по 90 символов \*/

displayMsg = wrapText(displayMsg.replace("\\n", "\n"), 90);

chatHistory.computeIfAbsent(chatKey, k -> new ArrayList<>()).add(displayMsg);

if (chatKey.equals(currentRecipient)) {

String out = displayMsg;

SwingUtilities.invokeLater(() -> {

jChat.append(out + "\n");

playNotificationSound();

});

}

}

} catch (IOException e) {

SwingUtilities.invokeLater(() -> jChat.append("❌ Соединение потеряно...\n"));

e.printStackTrace();

}

}).start();

}

// Variables declaration - do not modify

private javax.swing.JTextArea jChat;

private javax.swing.JButton jFile;

private javax.swing.JLabel jLabel1;

private javax.swing.JTextArea jMessage;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JScrollPane jScrollPane2;

private javax.swing.JButton jSend;

private javax.swing.JTextField jTextField1;

private javax.swing.JScrollPane jUserList;

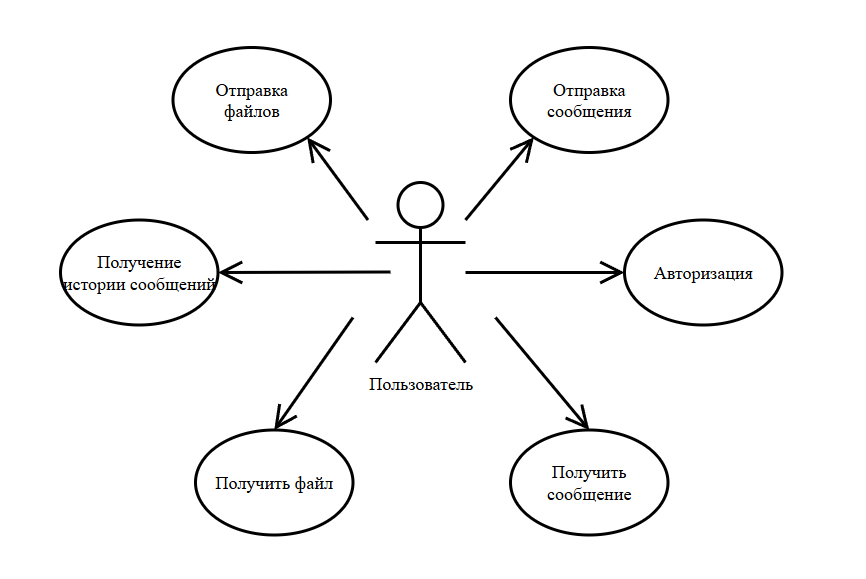
private javax.swing.JList<String> jUsers1;

// End of variables declaration

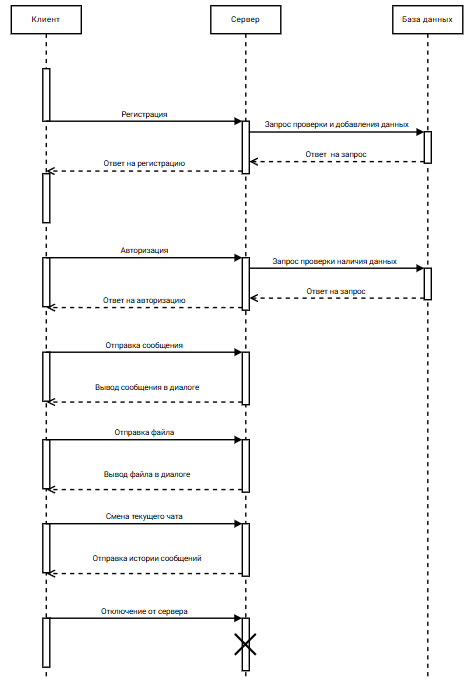
}

# Приложение Б. UML-диаграммы

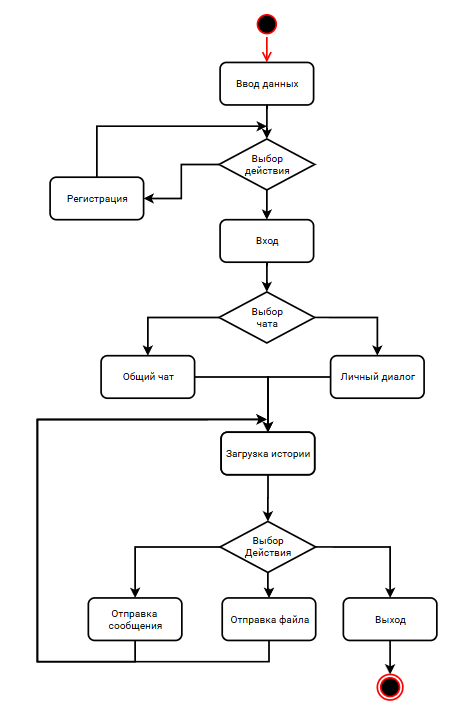
## Приложение Б.1. UML-диаграмма вариантов использования



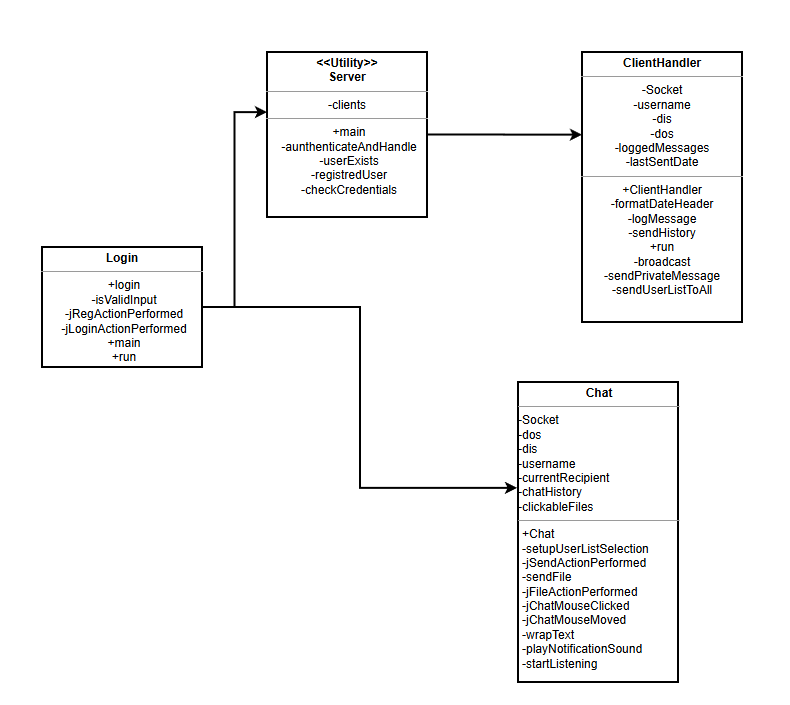
## Приложение Б.2. UML- диаграмма последовательности



## Приложение Б.3. UML- диаграмма деятельности



## Приложение Б.4. UML-диаграмма классов



## Приложение Б.5. UML-диаграмма развертывания

