Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Кафедра информационных технологий и систем

**Многопоточный сетевой чат**

Курсовая работа

по дисциплине:

«Операционные системы»

Разработал:

Студент группы 0092

Потапенко Е.А\_\_\_\_\_\_

«\_\_».\_\_\_\_\_.2022г.

Проверил:

Фуфин М.Н.\_\_\_\_\_\_

«\_\_».\_\_\_\_\_.2022г.

Великий Новгород

2022

Оглавление

[ОГЛАВЛЕНИЕ 2](file:///C:\Users\nerdy\Downloads\-.docx#__RefHeading___Toc10706_1140798338)

[ВВЕДЕНИЕ 3](file:///C:\Users\nerdy\Downloads\-.docx#__RefHeading___Toc10708_1140798338)

[ГЛАВА 1. РАЗРАБОТКА ПРОЕКТА. 4](file:///C:\Users\nerdy\Downloads\-.docx#__RefHeading___Toc14994_1140798338)

[1.1 Постановка задачи. 4](file:///C:\Users\nerdy\Downloads\-.docx#__RefHeading___Toc15001_1140798338)

[1.2 Выбор инструментальных средств. 4](file:///C:\Users\nerdy\Downloads\-.docx#__RefHeading___Toc15353_1140798338)

[1.3 Выбор способа взаимодействия потоков. 4](file:///C:\Users\nerdy\Downloads\-.docx#__RefHeading___Toc15831_1140798338)

[1.4 Использование сигналов и слотов для реализации графического интерфейса 4](file:///C:\Users\nerdy\Downloads\-.docx#__RefHeading___Toc10821_1069096194)

[1.5 Выбор библиотеки для реализации многопоточной модели 5](file:///C:\Users\nerdy\Downloads\-.docx#__RefHeading___Toc10254_122815407)

[1.6 Алгоритм решения задачи 6](file:///C:\Users\nerdy\Downloads\-.docx#__RefHeading___Toc10520_122815407)

[1.7 Вывод по первой главе. 7](file:///C:\Users\nerdy\Downloads\-.docx#__RefHeading___Toc10590_122815407)

[ГЛАВА 2. РЕАЛИЗАЦИЯ ПРОЕКТА. 8](file:///C:\Users\nerdy\Downloads\-.docx#__RefHeading___Toc10592_122815407)

[2.1 Структура приложения 8](file:///C:\Users\nerdy\Downloads\-.docx#__RefHeading___Toc10594_122815407)

[2.2 Описание программной реализации приложения 8](file:///C:\Users\nerdy\Downloads\-.docx#__RefHeading___Toc2570_1329954984)

[2.3 Пример работы приложения 9](file:///C:\Users\nerdy\Downloads\-.docx#__RefHeading___Toc2572_1329954984)

[2.4 Результат работы. 9](file:///C:\Users\nerdy\Downloads\-.docx#__RefHeading___Toc2576_1329954984)

[ЗАКЛЮЧЕНИЕ 10](file:///C:\Users\nerdy\Downloads\-.docx#__RefHeading___Toc963_812659571)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ. 10](file:///C:\Users\nerdy\Downloads\-.docx#__RefHeading___Toc11022_812659571)

**ВВЕДЕНИЕ**

Целью данной работы является создание многопользовательского чата, с помощью которого несколько пользователей могут обмениваться текстовой информацией.

Данная тема актуальна, т. к. помимо практического применения эта программа служит простой иллюстрацией сетевого взаимодействия.

ГЛАВА 1. РАЗРАБОТКА ПРОЕКТА.

1.1 Постановка задачи.

Задачей этой курсовой работы является разработка сетевого чата. Предполагается наличие простого интерфейса с полями для ввода, вывода текста, кнопкой для его отправки и поля для указания имя пользователя. Программа не предполагает какой либо регистрации в чате, просто подключение к серверу и ввод пользовательского имени.

1.2 Выбор инструментальных средств.

В качестве языка программирования был выбран Java, потому что он поддерживает объектно-ориентированную парадигму программирования, которая более подходит для создания приложений с графическим интерфейсом. В качестве альтернативы может быть рассмотрен язык С++, но Java обеспечивает более простую и удобную работу с потоками и сетевым взаимодействием. Для реализации графической части была использована библиотека Java Swing.

1.3 Выбор способа сетевого взаимодействия.

Есть два основных протокола сетевого взаимодействия это TCP и UDP.

TCP протокол основан на схеме “рукопожатия”, т. е. клиент посылает некоторый сегмент данных, потом ему приходит подтверждение о получении. В случае каких-либо неудач во время установки соединения, необходимые процедуры будут повторяться, т. е. протокол TCP гарантирует надежную доставку сообщения, а также гарантирует правильный порядок следования сообщений.

TCP используется когда необходимо гарантированная доставка информации.

В случае с UDP протоколом клиент просто отправляет некий пакет информации на сервер, данный протокол не гарантирует надежную доставку пакетов, используется для RTP (Real Time Protocol). Например, для прямых трансляций.

Для чата был выбран TCP протокол, т. к. нам важна гарантия доставки сообщения от клиента до сервера и наоборот.

1.4 Реализация графического интерфейса

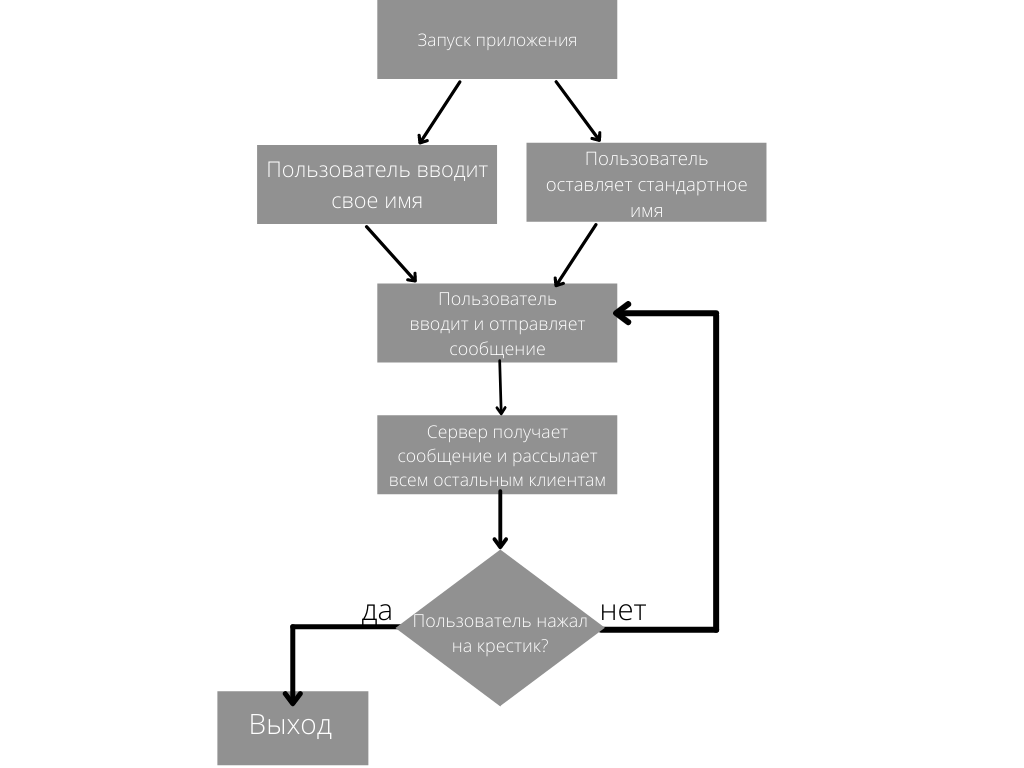
Графический интерфейс был реализован с помощью библиотеки Java Swing.

Данная библиотека уже содержит в себе все необходимые готовые объекты. Т.е. мы можем очень просто создавать различные объекты полей, окон, настраивать их размер, настроить, например, чтобы при нажатии на крестик программа закрывалась или просто сворачивалась. Также в Java присутствует интерфейс ActionListener его можно добавлять на различные графические объекты, чтобы они понимали и реагировали на различные действия пользователя.

**1.5 Выбор библиотек для реализации сетевого протокола TCP**

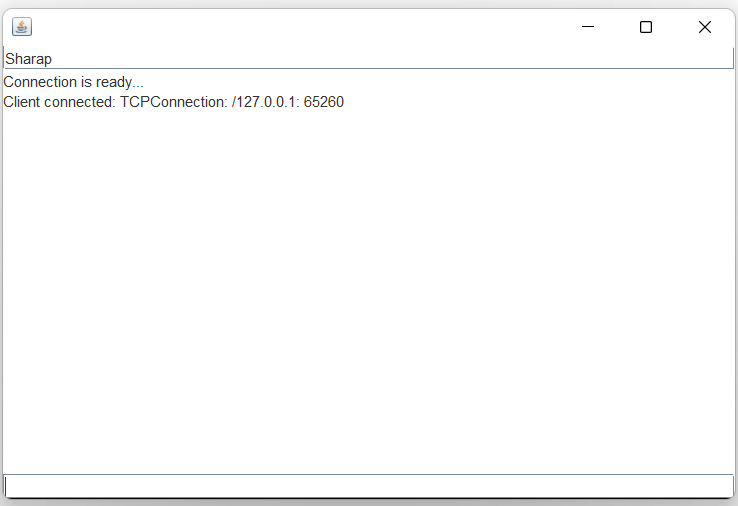
Для реализации протокола сетевого взаимодействия нам понадобятся две стандартные библиотеки: Java.net.Socket (для обмена информацией между потоками используются сокеты) и Java.io (для отлавливания исключений ввода-вывода, потому что пользователь может ввести некорректное сообщение)

1.6 Основная логика работы программы



Запуск приложения осуществляется только после запуска сервера. Клиент без него просто не запустится.

При запуске приложения появляется окно чата, вверху имеется поле для ввода имени, там уже указано стандартное имя, его можно изменить, написать свое, далее есть основное поле для сообщений, там выводятся пользовательские сообщения и сообщения о подключении клиента. В самом низу поле для ввода самого сообщения. Чтобы выйти из приложения необходимо нажать на крестик в правом верхнем углу.



1.7 Вывод по первой главе

Таким образом, для реализации многопользовательского чата основным языком программирования был выбран Java и его стандартные библиотеки (Swing, Java.net, Java.io). Swing – для реализации графического интерфейса, а

Java.net и Java.io – для реализации сетевого взаимодействия.

ГЛАВА 2. РЕАЛИЗАЦИЯ ПРОЕКТА.

2.1 Структура приложения.

Приложение состоит из 3-ех классов и 1 интерфейса.

ClientWindow – реализует клиента и графический интерфейс

ChatServer – реализует сервер, на котором все происходит

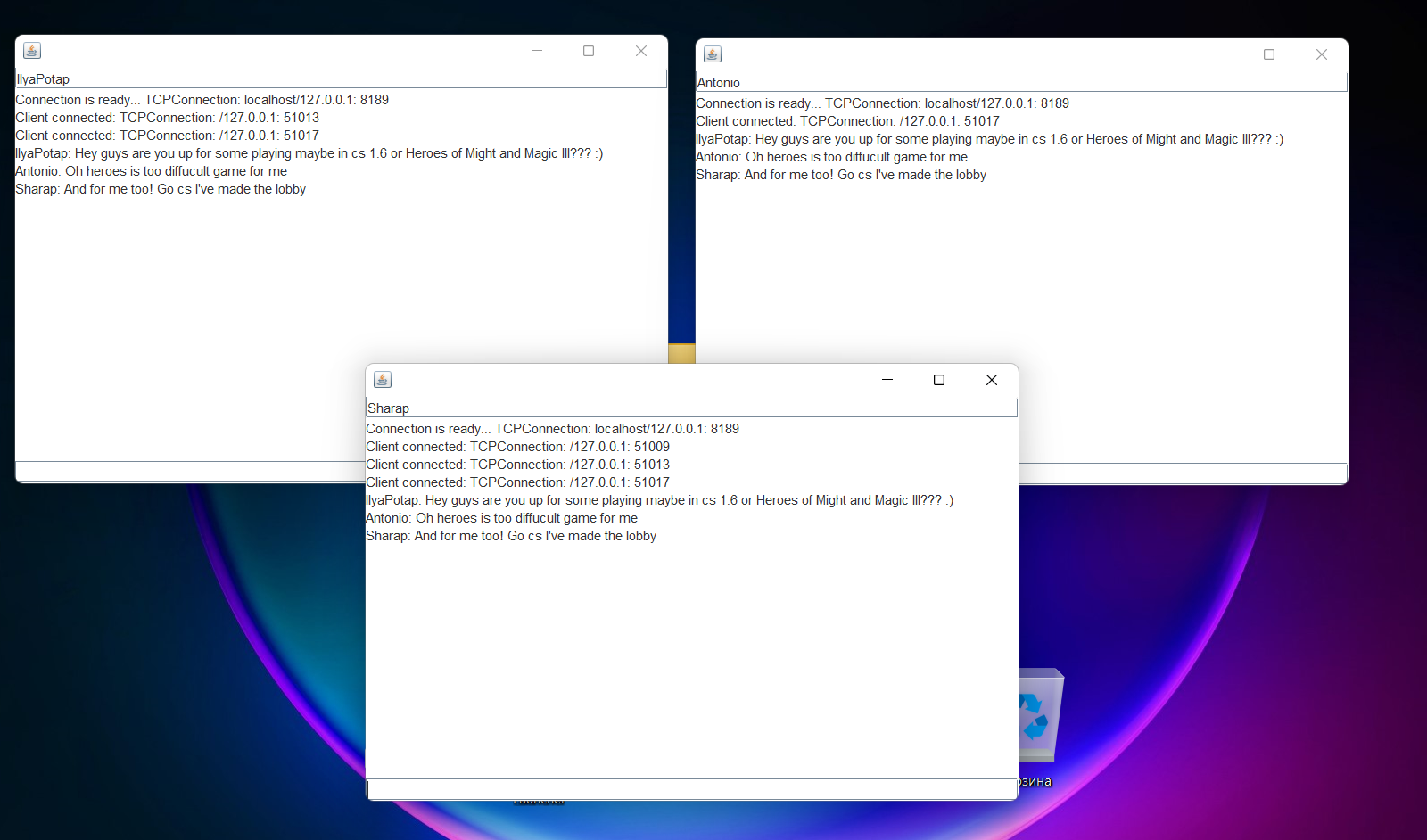
TCPConnection – реализует TCP соединение

TCPConnectionListener – система событий в сети

2.2 Описание программной реализации приложения

Запускается сервер (ChatServer). Создается серверный сокет и далее сервер в бесконечном цикле ожидает подключение клиента. К серверу можно подключиться как с браузера (не будет корректно работать), как с telnet, как с ClientWindow. После запуска ClientWindow создается устанавливается соединение между сервером и клиентом и создается два потока серверный и клиентский.И сначала у клиента вызовется метод OnConnectionReady и выведется сообщение “Connection ready…”, а потом у сервера вызовется такой же метод, в котором соединение с клиентом добавиться в массив connections и выведется сообщение, что клиент подключился. Каждый из потоков крутится в бесконечном цикле, клиентский поток ожидает отправки сообщения, а серверный ожидает его получение. ClientWindow имплементирует два интерфейса один это TCPConnectionListener, где методы для сети и отправки сообщений, а один это ActionListener он висит на текстовом поле, где пользователь вписывает свое сообщение, после нажатия enter, у ClientWindow вызовется метод ActionPerformed, который в свою очередь возьмет текст из текстового поля, очистит поле, и вызовет у connection (объект TCPConnection) метод SendString, который отправит сообщение на сервер. Класс сервера (ChatServer) также имплементирует интерфейс TCPConnectionListener и все это время он ждал получения сообщения. Как только он его получит вызовется метод onReceiveString, который в свою очередь вызовет метод SendToAllConnections, в данном методе сервер пройдется по всему массиву подключенных клиентов и каждому отправит сообщение, чтобы оно у всех появилось. Сделает он это с помощью SendString, у клиентов вызовется из-за этого метод OnReceiveString и у каждого клиента в log(основное поле) напечатается необходимое сообщение.

2.3 Пример работы приложения



После отключение клиента об этом выводится сообщение:

Изображение выглядит как текст

Автоматически созданное описание

2.4 Результаты работы

В ходе выполнения данной курсовой работы был реализован многопользовательский чат на Java с простым интерфейсом, реализованным на Java Swing. Каждый клиент представляет из себя отдельный поток.

ЗАКЛЮЧЕНИЕ

Подводя итоги, можно утверждать, что результат курсовой работы соответствует требованиям к реализации. Для создания приложения был использован язык Java и его стандартные библиотеки.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Карпов В.Е, Коньков К.А «Основы операционных систем. Курс лекций» - Учебное пособие, 2005 - 536с.

2. Н.Соболев, К.Жуковский, Р.Назарчук “Youtube путь к успеху как получать фуры лайков и тонны денег”

3.Герберт Шилдт “Java полное руководство”

4. Карпов В.Е, Коньков К.А «Основы операционных систем. Курс лекций» - Учебное пособие, 2005 - 536с.