Решение задачи разработки многопользовательского чата.

# Требования и запуск:

Требования для сборки и запуска:

* Java 21, убедиться, что Java 21 корректно установлена в переменной среды PATH
* Maven

Сборка и запуск:

* Перейти в корень клонированного репозитория и запустить "mvn package"
* Клиент будет доступен по пути "multiuser\_chat\client\target\client-1.0-SNAPSHOT-jar-with-dependencies.jar", запускать через "java -jar" под Linux, через "javaw -jar" под Windows.
* Сервер будет доступен по пути "multiuser\_chat\server\target\server-1.0-SNAPSHOT-jar-with-dependencies.jar", запускать через "java -jar".

# Состав и алгоритм работы:

Проект состоит из трёх частей:

1. Клиентская часть
2. Серверная часть
3. Общие примитивы для сетевой передачи данных

Алгоритм работы:

Сервер начинает прослушивать порт 10801 и создаёт экземпляр класса “ServerHandle”, вместе с которым запускается поток сервера.

При подключении клиента, в соответствие клиенту создаётся экземпляр класса “ClientHandle”, содержащего сокет соединения с подключённым клиентом, сразу после создания которого запускается поток клиента, ответственный за приём сообщений от этого подключённого клиента.

Поток приёма сообщений от клиента посылает через очередь сообщение потоку сервера о том, что новосозданный экземпляр ClientHandle нужно добавить в список клиентов.

Затем, когда клиент присылает запрос о регистрации/логина пользователя, поток клиента отправляет через ту же очередь сообщение о регистрации клиента. Поток сервера принимает это сообщение, создаёт запрос в базу данных, и если пользователь с данным именем существует, то отправляет этому клиенту дату последнего захода, обновляет эту дату в базе данных, выставляет в ней пользователя как пользователя в сети, а в ClientHandle заносит id записи в базе данных. Если же такого пользователя нету, то поток сервера регистрирует нового пользователя, сразу выставляя дату захода и статус онлайн, заносит новый id в ClientHandle и отправляет сообщение о регистрации. После всего поток сервера получает из базы данных всех клиентов, что находятся онлайн, и отправляет новый список всем подключённым пользователям.

Если зарегистрированный клиент отправляет сообщение, то это сообщение принимается потоком клиента, через очередь отправляется в поток сервера, а потом заносится в базу данных и отправляется всем клиентам, кроме отправившего.

Если клиент отключается, то поток клиента выходит из цикла постоянного чтения сокета, отправляет через очередь сообщение о том, что этот клиент отключился, и завершает работу. Поток сервера принимает сообщение, выставляет статус оффлайн для клиента в базе данных (если клиент был зарегистрирован), удаляет соответствующий удалённому клиенту ClientHandle из списка соединений, а оставшимся клиентам отправляет обновлённый список активных пользователей.

# Описание классов

Общие классы, интерфейсы:

* Интерфейс IMessage
  + Интерфейс классов, использующихся в межсетевом общении.
  + Метод intoContainer() оборачивает данный класс в специальный класс-контейнер MessageContainer для более лёгкой сериализации.
* Класс MessageContainer
  + Хранит в себе одну из разновидностей сообщений в качестве поля.
  + IntoMessage() возвращает хранящееся внутри сообщение.
* Класс UserlistMessage
  + Хранит имена активных участников, рассылаемый сервером.
* Класс UserMessagesMessage
  + Хранит текстовое сообщение от клиента или от сервера, и имя отправителя.
* Класс WhoImIMessage
  + Хранит имя клиента, под которым его нужно зарегистрировать.

Классы клиента:

* ClientMain
  + Метод main – стартовая точка программы, создаёт экземпляр класса ChatWindow и по очереди обновляет окно интерфейса и внутреннее состояние.
* ClientWindow
  + Класс-наследник BasicWindow – класса библиотеки Lanterna для терминального GUI, ответственный за отображение присланных сообщений и активных пользователей, отображения полей для подключения к серверу, регистрации, отправки сообщений.
  + UpdateTick() – при наличии подключения считывает из очереди в классе MessageSenderReceiver входящие сообщения и обновляет графический интерфейс в зависимости от пришедшего сообщения, используя для этого методы AddMessageToChatbox(msg, sender, time), updateActiveUsers(list)
  + updateActiveUsers(list) – обрабатывает принятый по сети список пользователей и обновляет отображаемый список пользователей.
  + AddLineToChatbox(line) – добавляет строку к окну чата, удаляя старые сообщения, что не помещаются в экран.
  + AddMessageToChatbox(msg, nickname, time) – форматирует сообщение перед добавлением в окно чата, добавляет сообщение через функцию AddLineToChatbox(line).
  + HandleConnect() – обрабатывает нажатие кнопки соединения, создавая экземпляр класса MessageSenderReceiver и запуская соответствующий поток.
  + HandleLogin() – обрабатывает нажатие кнопки входа, отправляя на сервер сообщение о регистрации.
  + HandleMessageSend() – обрабатывает нажатие кнопки отправки сообщения, отправляя на сервер сообщение с текстом от пользователя.
* MessageSenderReceiver
  + Конструктор – создаёт подключение к серверу и соответствующие классы Reader и Writer
  + SendMessage(messageContent) – отправляет содержимое отправляемого пользователем сообщения на сервер.
  + SendLogin(newNickname) – отправляет сообщение о входе/регистрации на сервер.
  + run() – получает сообщения с сервера и помещает их в очередь на обработку.

Классы сервера:

* DB
  + Конструктор – создаёт подключение к базе данных H2 с заданной строкой подключения.
  + makeInMemory() – создаёт базу данных H2 в памяти.
  + EnsureUserlogTableCreation() – создаёт таблицу сообщений со следующим содержимым:
    - id
    - content
    - userid
    - time
  + EnsureRegisteredUsersTableCreation() – создаёт таблицу пользователей со следующим содержимым:
    - id
    - username
    - lastvisit
    - online
  + insertNewMessage – добавляет сообщение в таблицу сообщений.
  + getIdOfUser – получает Id пользователя в таблице по имени, возвращает null, если пользователя с таким именем нету.
  + updateUserStatus – обновляет статус online
  + updateLastJoinAndOnlineStatus – обновляет дату последнего захода и статус онлайн за один запрос.
  + insertNewUser – добавляет нового пользователя, возвращая его id.
  + getUserLastJoinTime – получает время последнего захода пользователя по его id.
  + getUserOnlineStatus – получает время последнего захода пользователя по его id.
  + getOnlineUsers – получает список имён пользователей онлайн.
* IServerUserMessage
  + Интерфейс сообщений между серверным и клиентским потоком.
* NewUserJoinsMessage
  + Сообщает серверному потоку о новом клиенте, содержит UserHandle.
* UserRegisteredMessage
  + Сообщает серверному потоку о том, что клиент прислал запрос на логин. Содержит UserHandle и имя пользователя, под которым пользователь хочет зайти.
* UserSentMessage
  + Сообщает серверному потоку о том, что клиент отправил текстовое сообщение. Содержит UserHandle и текст сообщения.
* UserLeftMessage
  + Сообщает серверному потоку о том, что пользователь отключился от сервера. Содержит UserHandle отключившегося клиента.
* ServerMain
  + Входная точка программы, создаёт базу данных и таблицы, начинает прослушку порта 10801 по любому входящему адресу. Создаёт ServerHandle, серверный поток. В цикле ждёт входящих клиентов, создаёт соответствующий подключённому клиенту UserHandle, запускает соответствующий клиентский поток.
* ServerHandle
  + Представляет собой контекст серверного потока, отвечая за обработку сообщений от клиентских потоков и отправку сообщений клиентам.
  + run() – входная точка серверного потока. В цикле опрашивает очередь сообщений, полученные сообщения обрабатываются методом processMessage(msg)
  + generateListOfUsers() – организует запрос в базу данных, составляя сообщение-список пользователей.
  + sendMessageToEveryone(msg, ignore) – отправляет сообщение msg всем клиентам, кроме клиента, переданного параметром ignore.
  + sendMessageToUser(msg, user) – отправляет сообщение msg конкретному пользователю user.
  + processMessage(msg) – обрабатывает входящее сообщение и изменяет состояние ServerHandle в зависимости от него.
    - Если сообщение – NewUserJoinsMessage, то добавляет пользователя в список подключённых пользователей.
    - Если сообщение – UserRegisteredMessage, то пытается получить id пользователя из базы данных, если успешно, то устанавливает этот id у Userhandle и обновляет статус пользователя в базе данных, если нет, то добавляет нового пользователя в базу данных. Рассылает всем пользователям список пользователей онлайн.
    - Если сообщение – UserSentMessage, то проверяет, регистрировался ли пользователь. Если да, то рассылает сообщение всем, кроме отправителя, и добавляет это сообщение в лог сообщений.
    - Если сообщение – UserLeftMessage, то удаляет пользователя из списка подключённых пользователей. Если пользователь зарегистрирован, то устанавливает его статус в базе данных и рассылает оставшимся пользователем обновлённых список пользователей.
* UserHandle
  + Представляет собой контекст конкретного пользователя, храня в себе сокет соединения с пользователем.
  + run() – входная точка клиентского потока. В цикле читает сокет пользователя, прочитанные сообщения отправляет в очередь сообщений. При разрыве сообщения выходит из цикла и отправляет сообщение серверному потоку об отключении клиента.