# *Chatting Application Using Java*

Missyolin Samosir
Fakultas Teknik Elektro dan Informatika
S1 Informatika 2021
Laguboti, Sumatera Utara
ifs21008@students.del.ac.id

Agustina Butar-butar
Fakultas Teknik Elektro dan Informatika
S1 Informatika 2021
Laguboti, Sumatera Utara
ifs21019@students.del.ac.id

Lukas Sinaga
Fakultas Teknik Elektro dan Informatika
S1 Informatika 2021
Laguboti, Sumatera Utara
ifs21032@students.del.ac.id

Nada Bakara
Fakultas Teknik Elektro dan Informatika
S1 Informatika 2021
Laguboti, Sumatera Utara
ifs21043@students.del.ac.id

Grase Panjaitan
Fakultas Teknik Elektro dan Informatika
S1 Informatika 2021
Laguboti, Sumatera Utara
ifs21053@students.del.ac.id

***Abstract*—*Chatting Application Using Java* adalah sistem perangkat lunak berbasis Java yang memungkinkan banyak pengguna untuk berkomunikasi secara online dalam waktu nyata. Menggunakan bahasa pemrograman Java, aplikasi dipastikan skalabilitas dan keandalannya. Terdiri dari *server* dan beberapa instance klien, aplikasi memusatkan koneksi pengguna, perutean pesan, dan manajemen data pengguna. Pengguna membuat koneksi dengan server, yang bertindak sebagai perantara dalam komunikasi dengan klien lain. Fitur penting dari obrolan termasuk autentikasi pengguna untuk komunikasi yang aman dan perpesanan *real-time* untuk komunikasi instan. Selain itu, aplikasi ini memungkinkan Pengguna dapat melakukan scroll navigation untuk fungsi *chat* yang panjang. Singkatnya, mengobrol menggunakan aplikasi Java menyediakan platform yang andal dan efisien bagi individu untuk terhubung dan berinteraksi. Menggunakan fungsi Java, ini memberikan solusi yang efisien untuk perpesanan dan kolaborasi waktu nyata.**

***Keywords*— *Chatting application, java, server, client, komunikasi.***

## I. INTRODUCTION

Jaringan komputer merupakan sebuah sistem yang terdiri atas komputer dan perangkat jaringan lainnya yang bekerja sama untuk mencapai suatu tujuan. Dalam mencapai tujuan tersebut, terdapat 2 bagian yang berguna untuk meminta dan memberikan layanan. Adapun yang meminta layanan disebut *client* dan memberikan layanan disebut *server*. *Client server* adalah salah satu konsep arsitektur dari perangkat lunak atau *software* yang menghubungkan dua objek berupa sistem *client* dan sistem *server* yang saling berkomunikasi melalui jaringan komputer ataupun dari komputer yang sama.

Salah satu media komunikasi yang cepat untuk menyebarkan informasi adalah aplikasi *chatting*. Aplikasi ini sangat bermanfaat bagi pengguna untuk bisa saling berinteraksi. Adapun aplikasi chatting ini nantinya dibangun dengan menggunakan salah satu bahasa pemrograman yaitu Bahasa Java. Java bisa mengatasi masalah dalam pembuatan suatu aplikasi s*oftware* yang bisa dijalankan secara langsung di *platform* tanpa perlu penyesuaian ulang pada *platform* nya. Fitur yang ada nantinya pada aplikasi ini seperti fitur *chat*. Ini dibuat berdasarkan referensi yang sudah ada dimana nantinya akan dimuat pada bagian referensi.

## II. LANDASAN TEORI

### *A. Client-Server Communication*

*Server* membuat *socket server* menggunakan kelas *ServerSocket* dan menentukan *port* yang akan mengarahkan datanya. Kemudian server akan menggunakan metode *accept()* pada *socket server* untuk menunggu koneksi dari *client*. Ketika koneksi diterima, *socket client* akan terbentuk. *Server* dan *client* dapat menggunakan *input/output stream* untuk membaca dan menulis data melalui *socket*. Dengan menggunakan *input stream*, pesan yang dikirim *client* akan di baca. kemudian menggunakan *output stream* pesan akan dikirim kembali kepada *server*. *Server* dan *client* dapat melakukan komunikasi secara bergantian, membaca dan menulis data melalui *socke*t, hingga koneksi ditutup menggunakan *method close()* dan seluruh sumber daya yang digunakan akan dilepaskan.

Pada sisi *client*, langkah-langkahnya serupa dengan *server*: Dari sisi *client*, dengan *socket client* kelas *socket* dan akan menghubungkan menuju alamat IP serta nomor *port* yang telah ditentukan oleh *server* sebelumnya. *Client* akan menggunakan *input/output stream* untuk membaca dan menulis data melalui *socket*. *Client* dapat menggunakan *output stream* untuk mengirim pesan ke *server*, sedangkan *input stream* digunakan untuk membaca pesan dari *server*. *Client* dan *server* dapat melakukan komunikasi secara bergantian, membaca dan menulis data melalui *socket*, hingga koneksi ditutup.

### *B. Socket*

*Socket* dalam konteks *client-server* mengacu pada mekanisme komunikasi yang memungkinkan pertukaran data antara komputer (*client*) dan *server* melalui jaringan. *Socket* menyediakan saluran komunikasi dua arah yang dapat digunakan untuk mengirim dan menerima data antara komputer-komputer tersebut.

Socket server adalah program yang berjalan pada *server* dan menerima koneksi dari *client* kemudian akan menangani permintaan dari *client*. *Socket server* akan membuat *socket* tunggal yang mendengarkan pada *port* tertentu dan menerima koneksi dari *client*. *Port* sendiri merupakan angka yang menentukan saluran komunikasi yang spesifik antara dua komputer yang terhubung melalui jaringan. Setiap aplikasi yang berkomunikasi melalui jaringan, *port* akan digunakan untuk mengarahkan data ke tujuan yang tepat.

*C. UI with Java*

Java adalah salah satu teknologi yang dapat dijalankan di berbagai pltaform Sistem Operasi seperti Linux, Windows dan Unix. Java juga dapat mengatasi masalah portabilitas yang sering kali terbatas dan terhambat dalam pembuatan aplikasi *software* karena dapat dijalankan secara langsung tanpa banyak perubahan.

Pengembangan aplikasi Java memungkinkan Anda membuat antarmuka pengguna (UI) menggunakan beberapa kerangka kerja dan pustaka yang tersedia. Salah satu *framework* paling populer untuk membangun antarmuka pengguna di Java adalah JavaFX.

Berikut merupakan beberapa toolkit dan framework yang digunakan untuk membuat UI:

1. JavaFX: salah satu perangkat yang kuat dan fleksibel untuk mengembangkan antarmuka pengguna untuk aplikasi desktop Java. JavaFX ini menawarkan komponen UI yang fleksibel, animasi gaya dengan CSS, dan dukungan media.

2. Swing: salah satu perangkat antarmuka pengguna lama untuk aplikasi desktop Java. Swing menyediakan komponen termasuk tombol, panen, tabel, dll. Terdapat juga pengelola tata letak seperti *FlowLayout, BorderLayout,* dan *GridBagLayout.*

3. AWT(Abstract Window Toolkit): perangkat ini menyediakan komponen antarmuka pengguna dasar seperti tombol, label dan kolom input.

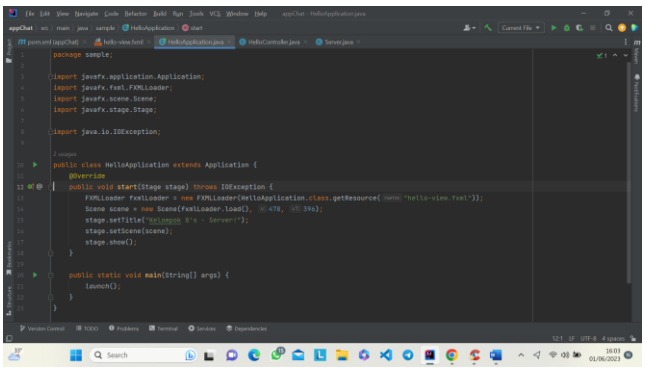
4. Java Native Interface (JNI): perangkat ini mengintegrasi C atau C++ dengan Java. Terdapat juga library dan framework UI yang bisa digunakan seperti Qt atau GTK untuk membuat UI dalam Java.

*D. Chatting Application*

Aplikasi *chatting* adalah salah satu aplikasi berkomunikasi yang memungkinkan berkomunikasi antara 2 atau lebih user dalam suatu jaringan yang diatur dalam sebuah server. Layanan *chatting* ini menjadi salah satu media yang cepat dalam penyebaran informasi. Bila dalam kondisi *online*, pengguna bisa berinteraksi melalui teks atau bisa berbicara langsung. Fungsi dasar dari aplikasi obrolan adalah kemampuan untuk mengirim dan menerima pesan teks. Pengguna dapat menulis pesan teks dan mengirimkannya ke kontak mereka. Penerima akan menerima pesan ini dan akan muncul dalam percakapan.

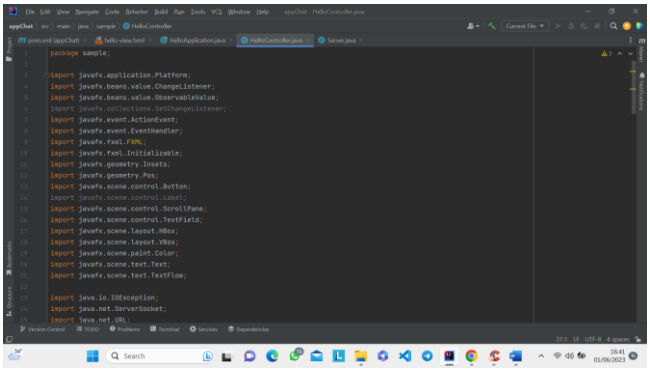
## III. PROSEDUR PERCOBAAN

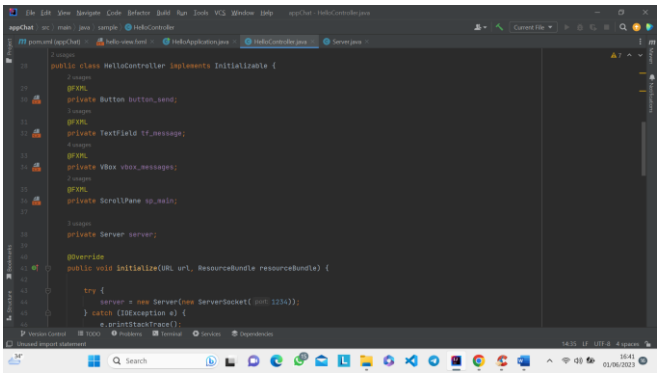
*A. HelloApplication.java*

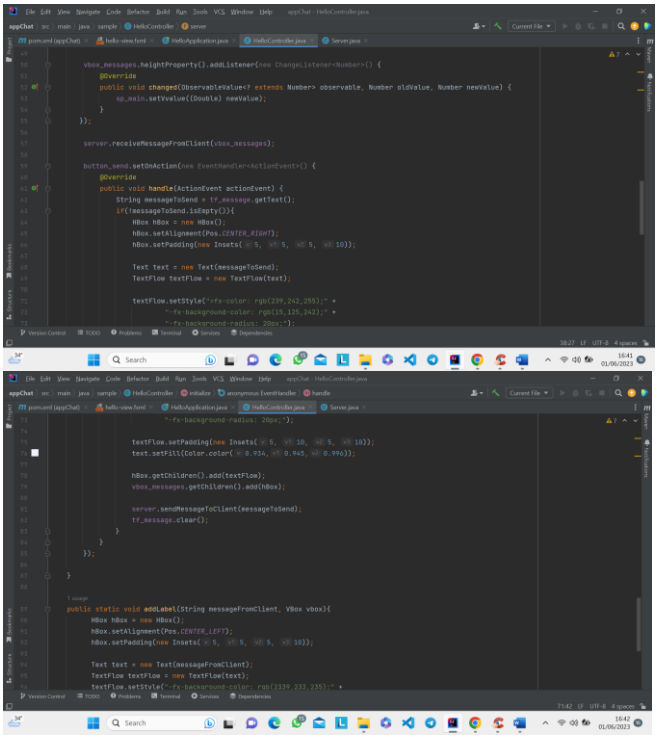


*HelloApplication.java* merupakan program JavaFX yang menampilkan antarmuka pengguna (UI) menggunakan file FXML. Program ini menginisialisasi dan menampilkan jendela aplikasi dengan judul "Kelompok 8's - Server!". Program akan mengimpor paket-paket yang diperlukan untuk menggunakan kelas-kelas JavaFX dan IOException. Pada *line* ke-12, metode start akan di*override* pada *class Application* dan akan dipanggil saat program dimulai. Dalam metode *start*, program menggunakan FXMLLoader untuk memuat file FXML yang disebut "*hello-view.fxml*".
File FXML tersebut berisi struktur tampilan UI yang akan ditampilkan oleh aplikasi. Kemudian, kita membuat objek Scene dengan menggunakan *root node* yang dimuat dari file FXML tersebut. Ukuran *Scene* ditetapkan menjadi 478 piksel lebar dan 396 piksel tinggi.
Selanjutnya, jendela aplikasi akan diatur menggunakan metode *setTitle()*. Kemudian, program mengatur *Scene* yang dibuat sebelumnya sebagai *Scene* utama pada *Stage* dengan menggunakan metode *setScene()*. Akhirnya, program memanggil *stage.show()* untuk menampilkan jendela aplikasi. Pada *line* ke-20, *method main* akan memulai eksekusi pada program Java. *Method main* akan menggunakan *method launch()* dari kelas Application. *Method launch()* akan memulai aplikasi JavaFX dan secara otomatis memanggil *method start()* yang di*override* sebelumnya.

*B. HelloController.java*

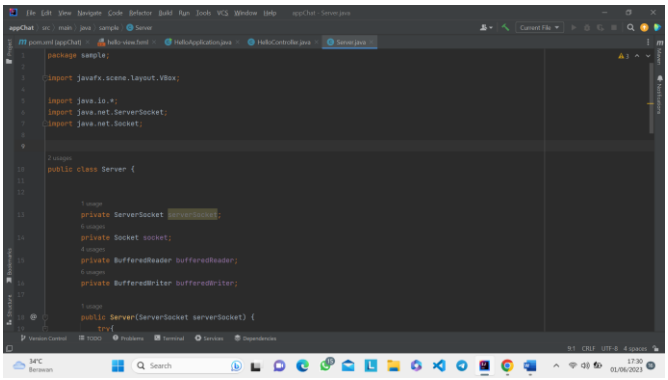


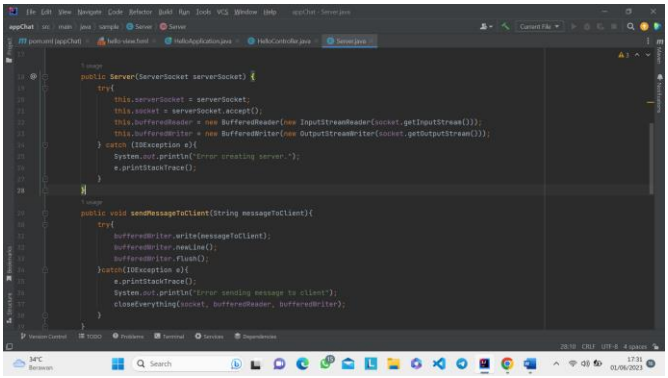


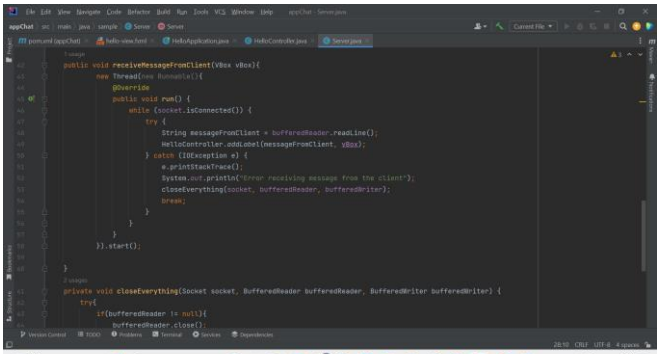


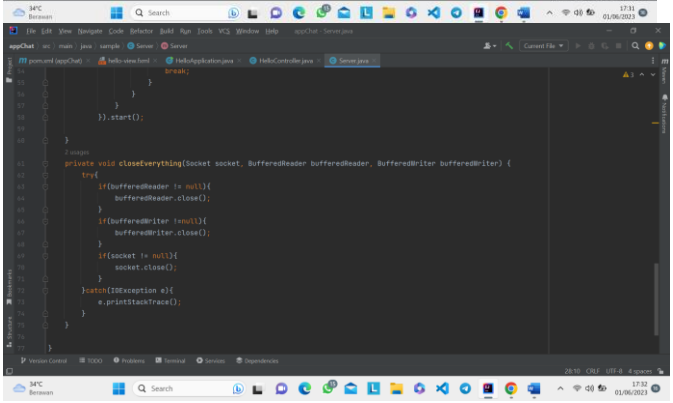
Program di atas adalah bagian dari aplikasi yang mengimplementasikan antarmuka pengguna (UI) dan logika bisnis untuk aplikasi *chat server*. Kode program ini menggunakan beberapa paket dari JavaFX dan Java standar yang diperlukan untuk mengimplementasikan UI dan logika aplikasi. Kelas *HelloController* merupakan *controller* yang mengatur tampilan dan logika aplikasi berdasarkan file FXML yang terhubung dengannya. Metode *initialize* merupakan metode yang diimplementasikan dari antarmuka *Initializable* dan dipanggil setelah file FXML terhubung dengan *controller*. Metode ini digunakan untuk melakukan inisialisasi komponen dan mengatur logika awal aplikasi. *Server socket* dibuat pada port 1234 menggunakan *ServerSocket*. Kemudian, obyek *Server* diinisialisasi dengan *ServerSocket* tersebut. Juga, diberikan *ChangeListener* pada *vbox_messages* untuk memastikan *scroll pane (sp_main)* selalu berada di posisi paling bawah saat pesan baru ditambahkan. *button_send* diberikan *event handler setOnAction* untuk menangani aksi saat tombol diklik. Metode *addLabel* digunakan untuk menambahkan pesan dari klien ke kotak pesan *(vbox_messages)* pada sisi kiri. Metode ini dipanggil dari kelas *Server* saat pesan diterima dari klien. Pesan tersebut ditambahkan dalam *HBox*, *Text*, dan *TextFlow*, kemudian ditambahkan ke *vbox_messages* dengan bantuan *Platform.runLater()* agar pembaruan UI dilakukan dalam *thread JavaFX*.

*C. Server.java*









*Server.java* merupakan program yang bertanggung jawab untuk mengelola komunikasi antara server dan klien dalam aplikasi *chat server*. Program diatas memiliki beberapa variabel dan objek, diantaranya adalah sebagai berikut.

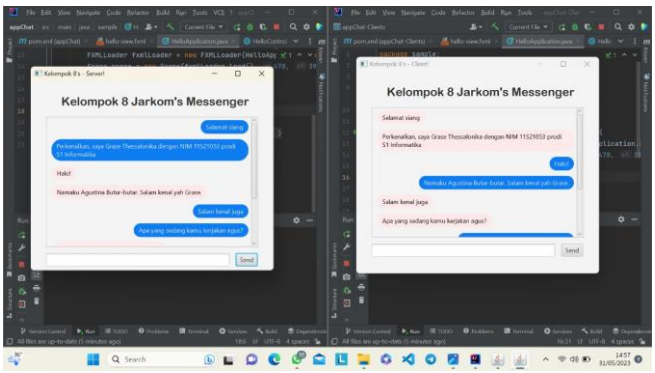
- *serverSocket*: Objek *ServerSocket* yang digunakan untuk menerima koneksi dari klien.
- *socket*: Objek *Socket* yang mewakili koneksi dengan klien.
- *bufferedReader*: Objek *BufferedReader* untuk membaca pesan yang diterima dari klien.

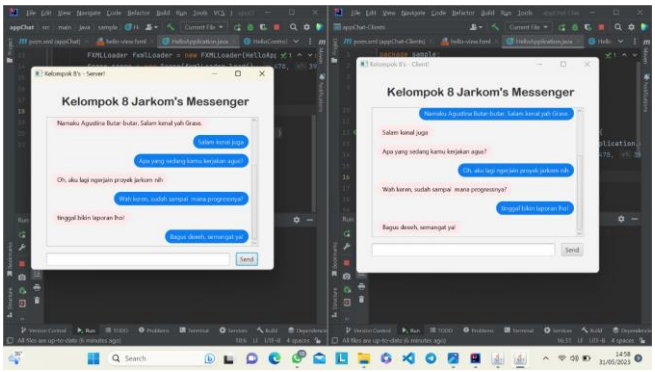
- *bufferedWriter*: Objek *BufferedWriter* untuk mengirim pesan ke klien.

Kemudian program server.java juga memiliki konstruktor yang akan menerima objek *ServerSocket* sebagai parameter dan melakukan inisialisasi variabel-variabel objek *Server* dengan menggunakan *serverSocket* tersebut. Di dalam konstruktor, objek *Socket* dibuat dengan menggunakan *serverSocket.accept()* untuk menerima koneksi dari klien. Objek *BufferedReader* dan *BufferedWriter* dibuat untuk membaca dan menulis pesan melalui *socket.getInputStream()* dan *socket.getOutputStream()*. Kemudian, program akan menggunakan method *sendMessageToClient* untuk menerima pesan dari klien. Di dalam *loop*, metode *bufferedReader.readLine()* digunakan untuk membaca pesan yang dikirim oleh klien. Setiap pesan yang diterima, method *HelloController.addLabel()* akan dipanggil untuk menambahkan pesan tersebut ke VBox dalam antarmuka pengguna. Terakhir, gunakan method *closeEverything* untuk menutup semua sumber daya yang terkait dengan koneksi dan komunikasi dengan klien.

## IV. HASIL DAN ANALISIS

Berikut merupakan hasil ketika server dan client dijalankan.




Pada jendela aplikasi sebelah kiri merupakan socket server yang juga akan membentuk *socket client* pada jendela aplikasi sebelah kanan. Server dapat menjadi yang pertama dalam mengirimkan pesan untuk berkomunikasi, berikutnya pesan yang dikirimkan akan diterima oleh *client*. Dengan begitu *client* dapat memberikan balasan pada komunikasinya. *Socket* akan bertindak sebagai antarmuka yang terhubung satu sama lain melalui jaringan. *Socket* sendiri akan terdiri dari alamat IP dan nomor *port*.

Pada gambar diatas, pesan yang terkirim ditandai dengan *background* pesan berwarna biru, sedangkan pesan yang diterima ditandai dengan *background* pesan berwarna abu-abu. Selain itu, terdapat juga fitur *scroll bar* yang bisa digunakan apabila antar *server* dan *client* memiliki pesan yang cukup panjang. Fitur tersebut nantinya bisa digunakan untuk melihat riwayat chat sebelumnya yang sudah ada.

## V. KESIMPULAN

- *Socket programming* akan memungkinkan komunikasi dua arah antar *server* dan *client* melalui jaringan menggunakan socket.
- *Server* akan menerima koneksi dari *client*. Kemudian server juga akan menyediakan layanan *chat* sehingga dapat berkomunikasi dengan *client*. *Server* akan menjadi pusat pengaturan pesan komunikasi.
- Dari sisi *client*, *client* akan mengirimkan *request* dan membuat koneksi dengan *server*. *Client* akan menggunakan *ip address* dan nomor *port* sesuai dengan yang telah diberikan oleh *server*. *Client* akan menerima pesan dan menampilkannya kepada pengguna.
- Saat *client* mengirim pesan, maka *server* yang akan menampilkannya kepada pengguna.
- Dengan adanya peran *server* dan *client*, aplikasi *chat* dapat menyediakan *platform* untuk pengguna berkomunikasi secara *real-time* melalui jaringan.

## REFERENCES


1. Malhotra, A., Sharma, V., Gandhi, P., & Purohit, N. (2010, April). *UDP based chat application. In 2010 2nd International Conference on Computer Engineering and Technology* (Vol. 6, pp. V6-374). IEEE.
2. Kalita, L. (2014). Socket programming. *International Journal of Computer Science and Information Technologies*, *5*(3), 4802-4807.
3. Sawant, A. A., & Meshram, B. (2013). Network programming in Java using Socket. *Google Scholar*.
4. Code reference : https://youtu.be/_1nqY-DKP9A