BAB IV

PERANCANGAN

4.1 Metode Perancangan

Aplikasi yang dibuat merupakan aplikasi yang dibangun di atas teknologi Java yang mengimplementasikan paradigma pengembangan sistem berorientasi obyek. Dalam hal ini, penggunaan UML (*Unified Modelling Language*) sangat sesuai untuk melakukan perancangan sistem berorientasi obyek dibanding menggunakan model perancangan lainnya seperti DFD (*Data Flow Diagram*).

4.2 Hasil Perancangan

4.2.1 Rancangan Umum Aplikasi

UML memiliki beberapa konsep dasar yang diabstraksikan dalam bentuk *structural classification*, *dynamic behavior*, dan *model management*. Hal terpenting dalam penggunaan UML adalah pembuatan diagram yang sesuai dengan analisis dan pengembangan sistem. Notasi-notasi UML mampu merepresentasikan rancangan sistem yang berorientasi obyek sehingga menjadi lebih mudah ketika rancangan nantinya diimplementasikan pada bahasa pemrograman berorientasi obyek seperti Java. Pada tahap perancangan ini, dibatasi pada pembuatan 3 (tiga) diagram saja, yaitu *use case diagram*, *class diagram*, dan *sequence diagram*.

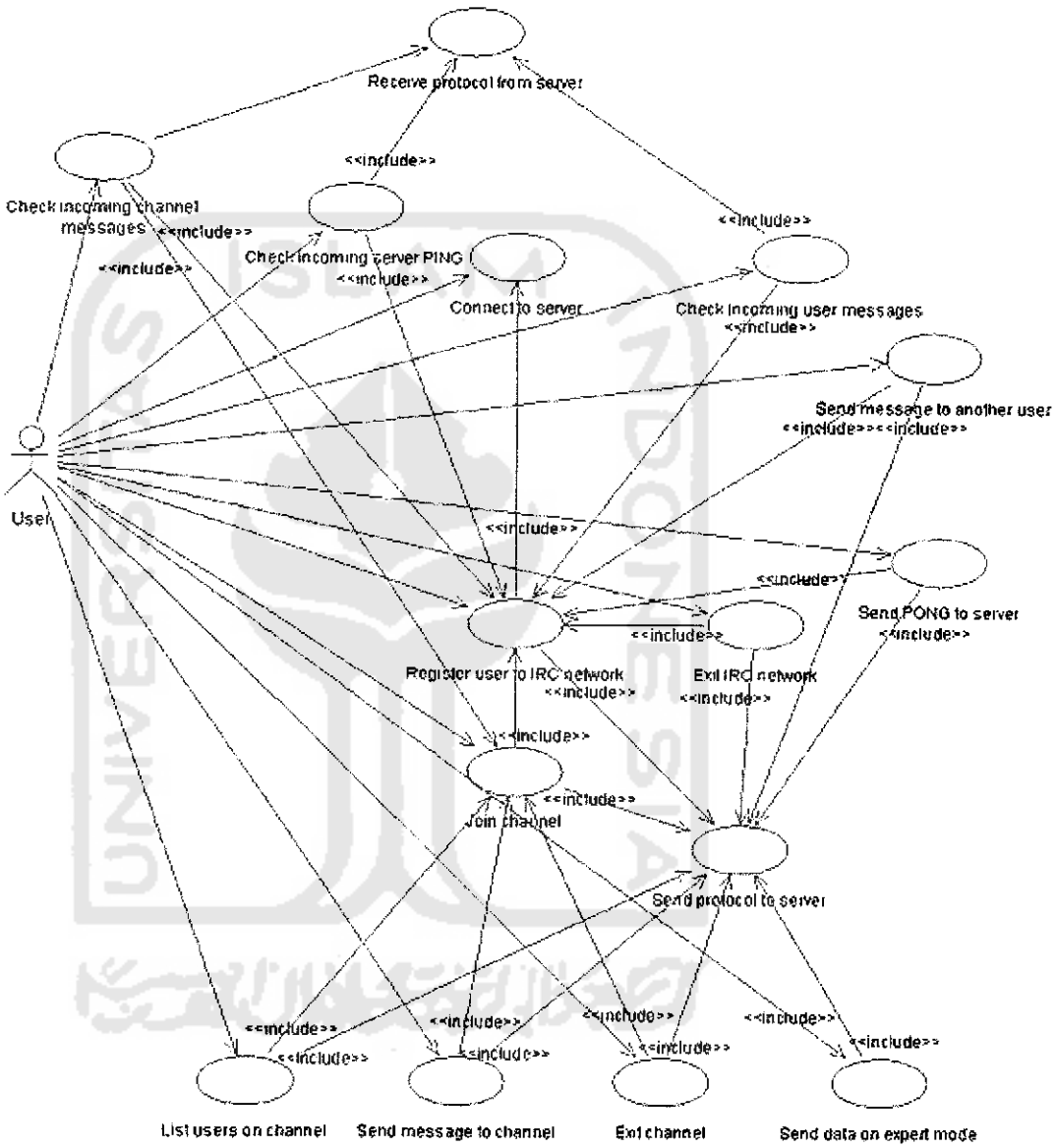
4.2.1.1 Use Case Diagram

Use case diagram berisi gambaran fungsionalitas yang diharapkan dari sebuah sistem dengan fokus penekanan pada apa yang dilakukan oleh sistem, bukan bagaimana sistem melakukan sesuatu. Dalam *use case diagram*, ada dua hal yang saling berhubungan, yaitu *actor* dan *use case*.

Pada rancangan aplikasi yang dibuat, hanya terdapat satu *actor* saja, yaitu *actor* User. Kemudian untuk *actor* User tersebut memiliki beberapa *use case* yang didefinisikan sebagai berikut:

a. Connect to server
b. Register user
c. Join channel
d. List users on channel
e. Send message to channel
f. Send message to another user
g. Check incoming channel messages
h. Check incoming user messages
i. Check incoming server PING
j. Send PONG to server
k. Exit channel
l. Exit IRC network
m. Send protocol to server
n. Receive protocol from server
o. Send data on expert mode

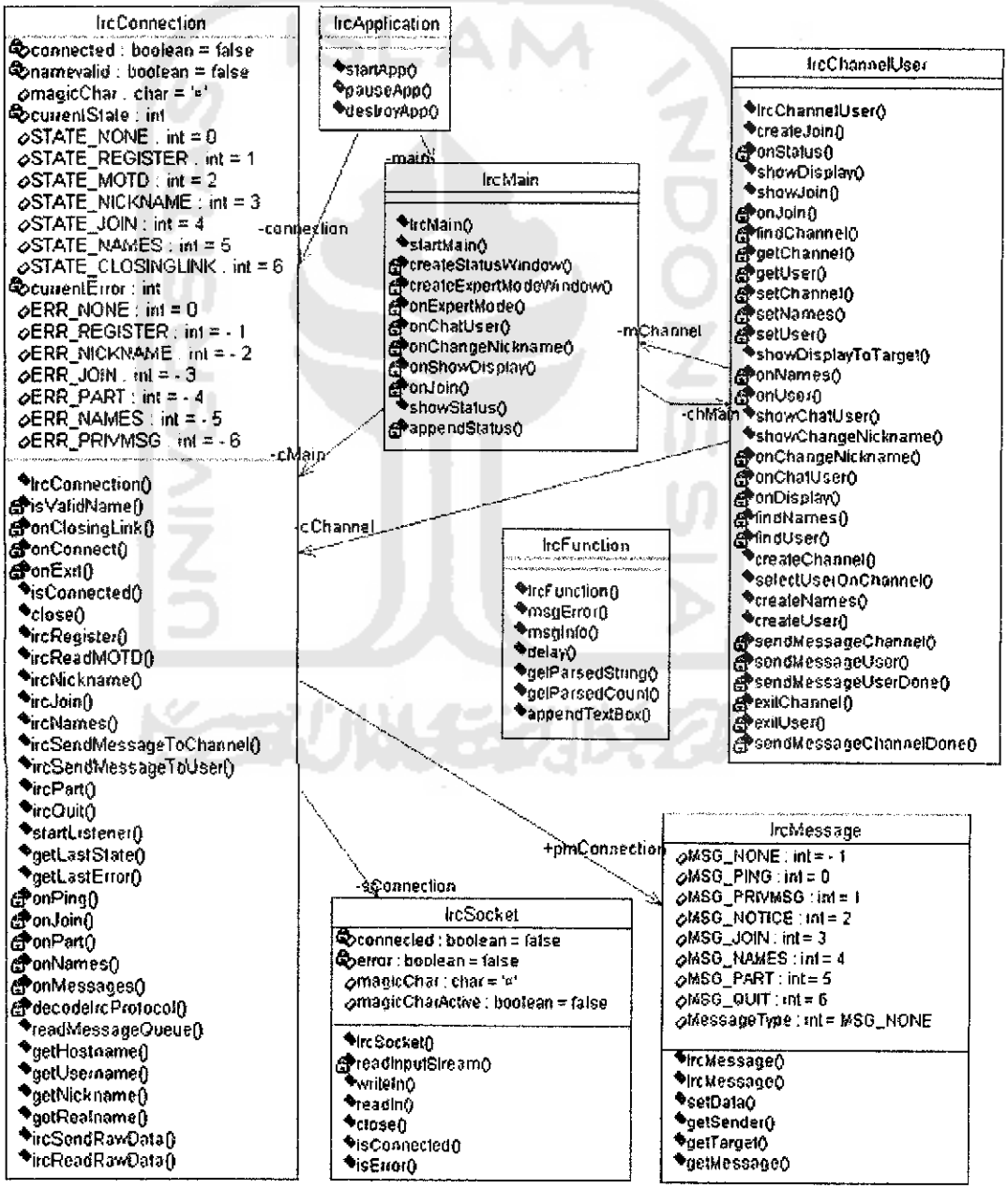
Pada *use case* tersebut terdapat banyak sekali terdapat stereotype include. Lebih jelasnya dapat dilihat pada gambar 4.1.



Gambar 4.1 Use Case Diagram

4.2.1.2 Class Diagram

Aplikasi yang menggunakan perancangan berorientasi obyek dapat diilustrasikan dalam struktur kelas-kelas dan hubungan antar kelas yang ada. Dalam UML digunakan notasi *class diagram* untuk menggambarkan hal tersebut. Lebih jelasnya dapat dilihat pada gambar 4.2.



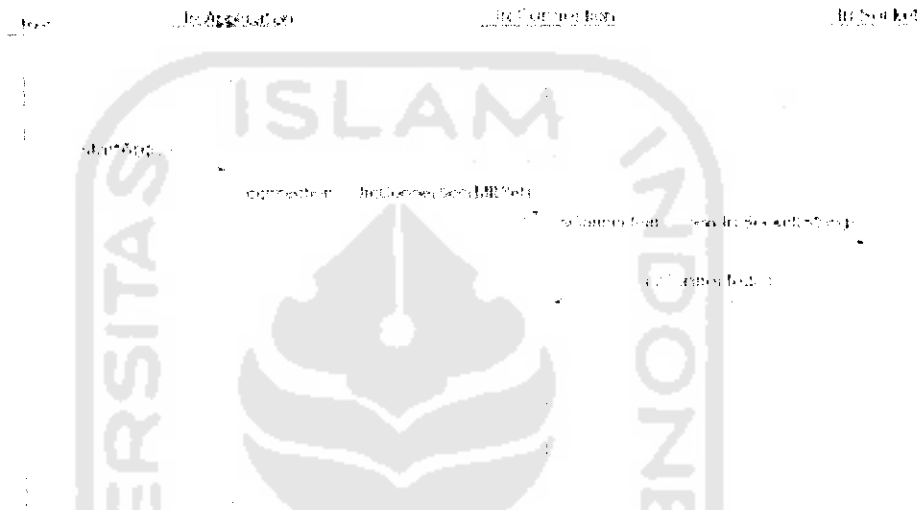
Gambar 4.2 Class Diagram

4.2.1.3 Sequence Diagram

Interaksi antar obyek dapat disusun berdasarkan urutan waktu yang menunjukkan skenario dan urut-urutan pertukaran data. Pada aplikasi J2ME IRC Client terdapat beberapa Sequence Diagram, yaitu:

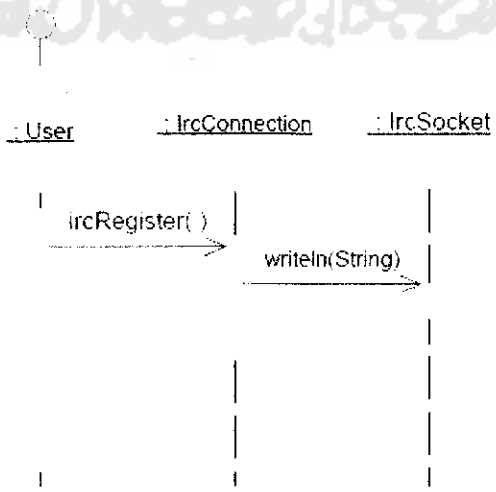
a. Sequence Diagram for Connect to server
b. Sequence Diagram for Register user
c. Sequence Diagram for Join channel (Create new channel)
d. Sequence Diagram for Join channel (Find joined channel)
e. Sequence Diagram for List users on channel
f. Sequence Diagram for Send message to channel
g. Sequence Diagram for Send message to another user
h. Sequence Diagram for Check incoming channel messages
i. Sequence Diagram for Check incoming user messages
j. Sequence Diagram for Check incoming server PING
k. Sequence Diagram for Send PONG to server
l. Sequence Diagram for Exit channel
m. Sequence Diagram for Exit IRC Network
n. Sequence Diagram for Send protocol to server
o. Sequence Diagram for Receive protocol from server
p. Sequence Diagram for Send data on expert mode

Sequence Diagram for Connect to server merupakan skenario pada proses melakukan koneksi ke server. Lebih jelasnya dapat dilihat pada gambar 4.3.



Gambar 4.3 Sequence Diagram for Connect to server

Sequence Diagram for Register user merupakan skenario pada proses registrasi user ke jaringan. Lebih jelasnya dapat dilihat pada gambar 4.4.

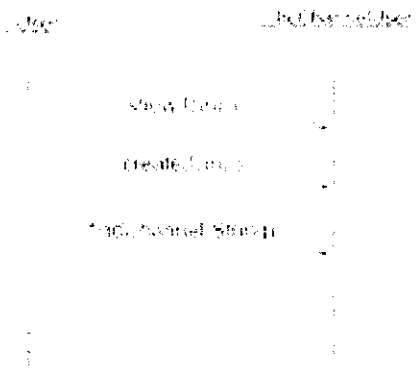


Gambar 4.4 Sequence Diagram for Register user

Sequence Diagram for Join channel (Create new channel) merupakan skenario pada proses join channel. Lebih jelasnya dapat dilihat pada gambar 4.5.

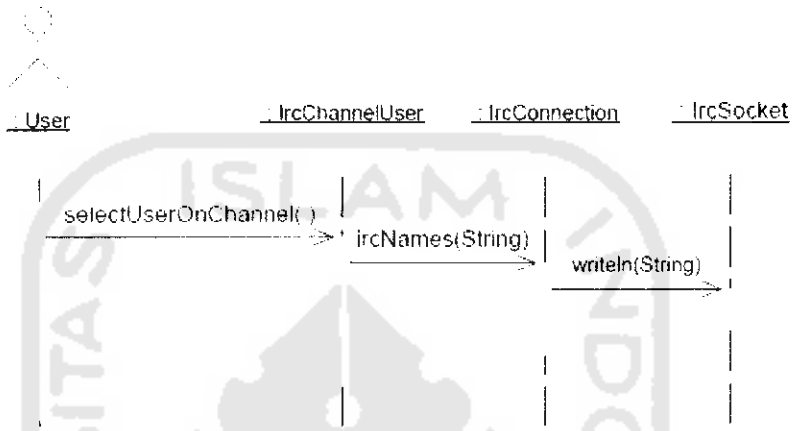
Gambar 4.5 Sequence Diagram for Join channel (Create new channel)

Sequence Diagram for Join channel (Find joined channel) merupakan skenario pada proses menampilkan display channel yang sudah ada sebelumnya. Lebih jelasnya dapat dilihat pada gambar 4.6.



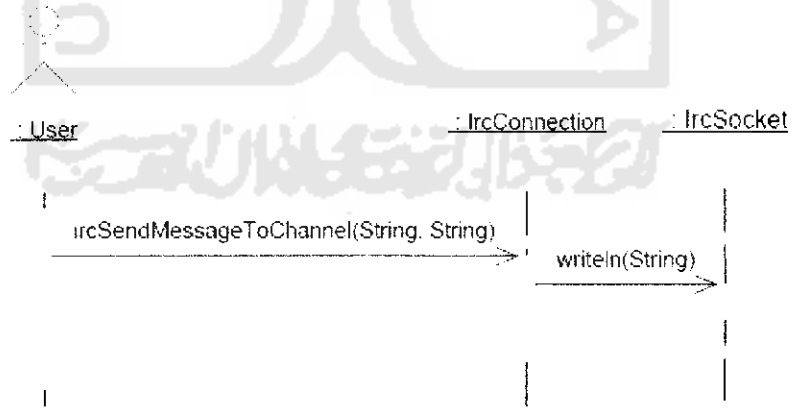
Gambar 4.6 Sequence Diagram for Join channel (Find joined channel)

Sequence Diagram for List users on channel merupakan skenario pada proses menampilkan user-user di channel. Lebih jelasnya dapat dilihat pada gambar 4.7.



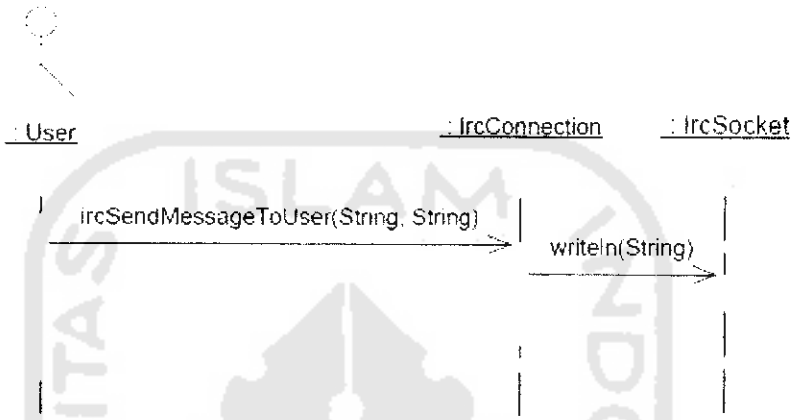
Gambar 4.7 Sequence Diagram for List users on channel

Sequence Diagram for Send message to channel merupakan skenario pada proses pengiriman pesan di channel. Lebih jelasnya dapat dilihat pada gambar 4.8.



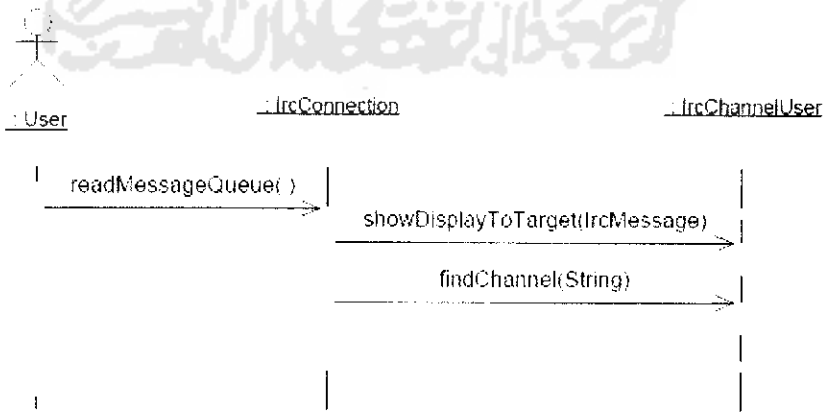
Gambar 4.8 Sequence Diagram for Send message to channel

Sequence Diagram for Send message to another user merupakan skenario pada proses pengiriman pesan antar user. Lebih jelasnya dapat dilihat pada gambar 4.9.



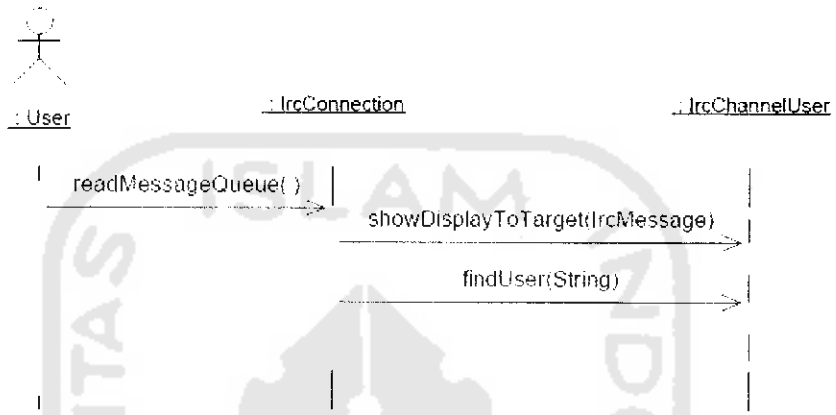
Gambar 4.9 Sequence Diagram for Send message to another user

Sequence Diagram for Check incoming channel messages merupakan skenario pada proses pengecekan pesan di channel. Lebih jelasnya dapat dilihat pada gambar 4.10.



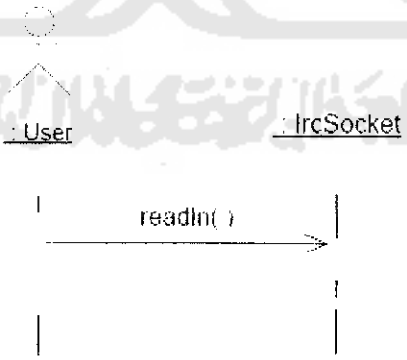
Gambar 4.10 Sequence Diagram for Check incoming channel messages

Sequence Diagram for Check incoming user messages merupakan skenario pada proses pengecekan pesan dari user lain. Lebih jelasnya dapat dilihat pada gambar 4.11.



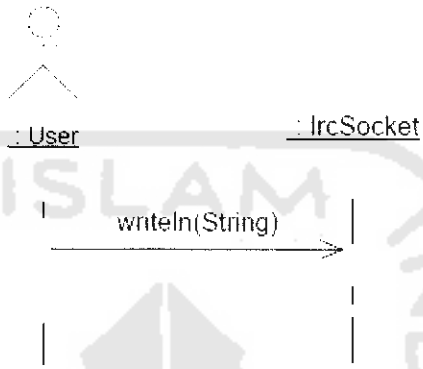
Gambar 4.11 Sequence Diagram for Check incoming user messages

Sequence Diagram for Check incoming server PING merupakan skenario pada proses pengecekan permintaan PING dari server. Lebih jelasnya dapat dilihat pada gambar 4.12.



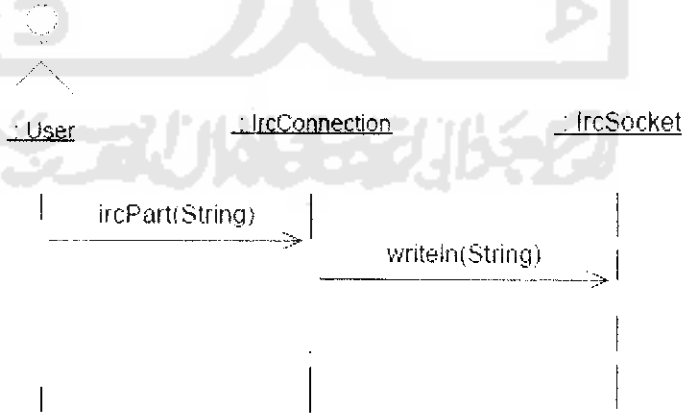
Gambar 4.12 Sequence Diagram for Check incoming server PING

Sequence Diagram for Send PONG to server merupakan skenario pada proses pengiriman respon atas permintaan PING dari server. Lebih jelasnya dapat dilihat pada gambar 4.13.



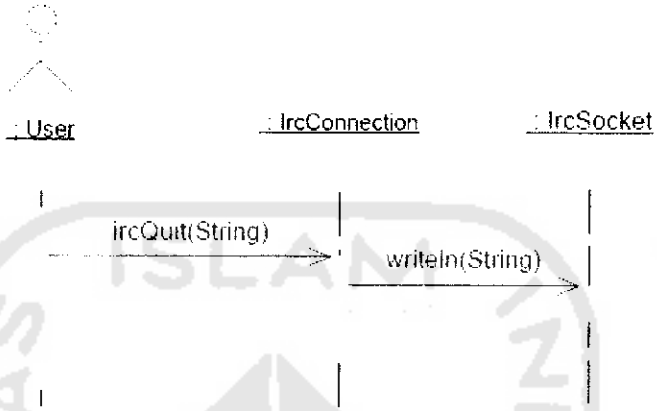
Gambar 4.13 Sequence Diagram for Send PONG to server

Sequence Diagram for Exit channel merupakan skenario proses ketika keluar dari channel. Lebih jelasnya dapat dilihat pada gambar 4.14.



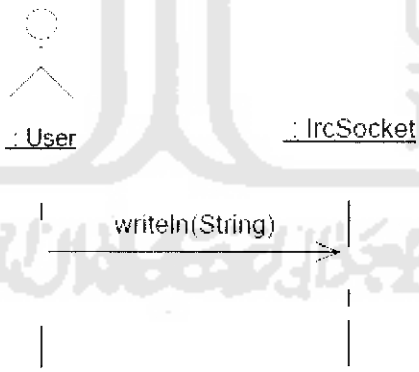
Gambar 4.14 Sequence Diagram for Exit channel

Sequence Diagram for Exit IRC Network merupakan skenario proses ketika keluar dari jaringan IRC. Lebih jelasnya dapat dilihat pada gambar 4.15.



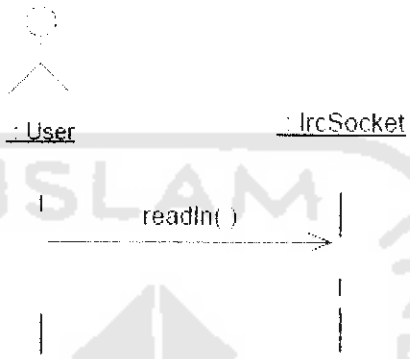
Gambar 4.15 Sequence Diagram for Exit IRC Network

Sequence Diagram for Send protocol to server merupakan skenario pada proses pengiriman data protokol. Lebih jelasnya dapat dilihat pada gambar 4.16.



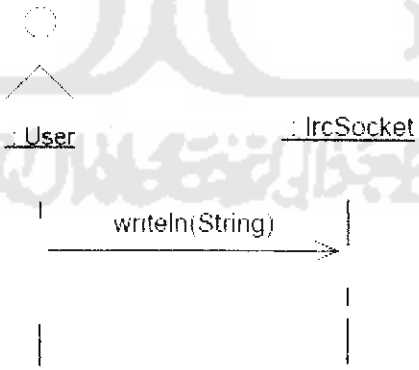
Gambar 4.16 Sequence Diagram for Send protocol to server

Sequence Diagram for Receive protocol from server merupakan skenario pada proses penerimaan data protokol. Lebih jelasnya dapat dilihat pada gambar 4.17.



Gambar 4.17 Sequence Diagram for Receive protocol from server

Sequence Diagram for Send data on expert mode merupakan skenario pada fasilitas Expert Mode. Lebih jelasnya dapat dilihat pada gambar 4.18.



Gambar 4.18 Sequence Diagram for Send data on expert mode

4.2.2 Rancangan Kode Program

Aplikasi J2ME IRC Client merupakan aplikasi yang dibangun menggunakan paradigma pemrograman berorientasi obyek (*Object-Oriented Programming / OOP*). Hal tersebut sangat berpengaruh pada struktur aplikasi yang dibangun, yaitu terutama pada perancangan aplikasinya.

Konsep protokol IRC tersebut diimplementasikan setelah berbagai mekanisme dasar yang ada dalam komunikasi IRC berhasil diindentifikasi secara menyeluruh. Hasil identifikasi tersebut meliputi proses-proses yang terjadi dan obyek-obyek yang terkait dengannya. Dari berbagai proses yang mungkin terjadi terhadap obyek-obyek yang ada, dapat diketahui pula bahwa tiap obyek memiliki pekerjaan-pekerjaan yang berbeda. Dalam konsep OOP, pekerjaan-pekerjaan itu dikenal dengan istilah metode (*method)*.

Sebuah obyek (*object)* merupakan *instance* dari sebuah kelas (*class)*. Sebuah *obyek* adalah sebuah kelas yang sudah diinstansiasi (siap digunakan, sudah memperoleh alokasi memori dan berbagai sumber daya CPU yang diperlukan). Dalam konteks pembahasan rancangan obyek, istilah kelas lebih tepat digunakan. Dalam aplikasi, sebuah kelas dapat diinstansiasi berkali-kali dan dapat menciptakan banyak obyek, tetapi tidak boleh terdapat lebih dari satu kelas dengan nama yang sama.

4.2.2.1 Pemrograman Socket

J2ME telah menyediakan Java API untuk jaringan TCP/IP menggunakan *socket*, yaitu pada paket *javax.microedition.io*. Dalam paket tersebut terdapat interface *StreamConnection* dapat dilakukan untuk melakukan koneksi TCP/IP, mengirim data, dan menerima data. Interface inilah yang dapat digunakan sebagai *socket* TCP/IP untuk komunikasi data di jaringan.

Berikut contoh penggunaan *socket* pada platform J2ME untuk membuka koneksi ke server IRC pada server localhost port 6667:

```
try {
        String url = "socket://localhost:6667";
        socket = (StreamConnection) Connector.open (
                                url, Connector.READ_WRITE );
}
catch (IOException ie) { }
```

Dalam dokumentasi J2ME Wireless Toolkit 2.2 [ANO04a] mengenai paket *javax.microedition.io*, terdapat sebuah interface bernama *SocketConnection* yang dikembangkan dari interface *StreamConnection*. *SocketConnection* sepertinya merupakan implementasi yang lebih baik dan lebih baru dibanding implementasi menggunakan *StreamConnection* (mulai versi MIDP 2.0).

Untuk interface *SocketConnection* terdapat keterangan mengenai proses *penutupan* stream sebagai berikut [ANO04a] :

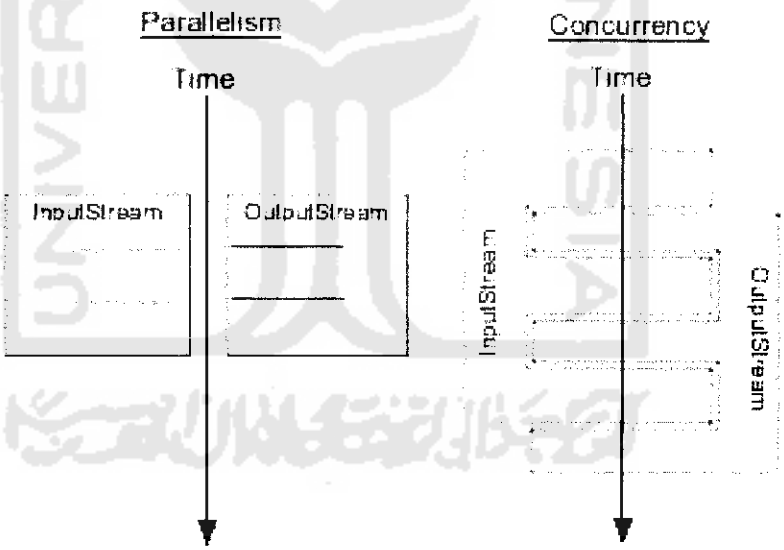
> *"For systems that support duplex communication over the socket connection, closing of the input or output stream SHOULD shutdown just that side of the connection. e.g. closing the InputStream will permit the OutputStream to continue sending data."*

Pernyataan resmi dari Sun Microsystems tersebut secara tidak langsung mengindikasikan adanya sistem yang tidak mendukung komunikasi *duplex* (dua arah secara paralel), padahal dalam protokol IRC mutlak diperlukan adanya jalur terpisah yang bersifat paralel untuk menangani data masuk dan data keluar.

Dugaan tersebut diperkuat lagi dengan adanya keterangan-keterangan lain yang muncul pada beberapa situs, yaitu antara lain situs *blog* milik Russel Beatie [BEA03a] dan sebuah situs forum [ANO04d] di Internet mengenai kegagalan operasi *socket* pada J2ME. Secara umum para programmer Java tersebut mengeluhkan macetnya aplikasi ketika dijalankan pada piranti target (misal: Nokia 7650, Nokia N-Gage), tanpa adanya pesan *error* apapun, padahal tersebut aplikasi berjalan dengan baik ketika dieksekusi pada emulator.

Inti permasalahan yang muncul adalah tidak semua sistem mendukung komunikasi *duplex* (secara *parallel*). Dalam memecahkan masalah tersebut, dapat digunakan solusi pemrosesan yang sering diterapkan pada sistem operasi dengan *single-processor* (satu buah mikroprosesor), yaitu untuk menggantikan pemrosesan secara *parallel* digunakan pemrosesan secara *concurrent*.

Kemudian untuk menangani operasi *socket* nantinya digunakan interface *StreamConnection* dengan pertimbangan kompatibilitas versi yang lebih baik karena interface *SocketConnection* baru tersedia pada versi MIDP 2.0 ke atas. Lebih jelasnya dapat dilihat pada gambar 4.19.



Gambar 4.19 Pemrosesan secara *parallel* dan *concurrent*

4.2.2.1.1 Pemakaian Socket secara *Parallel*

Pada sistem yang mendukung komunikasi *duplex*, *InputStream* dan *OutputStream* dapat digunakan secara bebas tanpa saling mengganggu satu sama lain. Suatu ketika dapat saja *InputStream* ditutup, namun *OutputStream* tetap dapat digunakan karena jalurnya terpisah.

4.2.2.1.2 Pemakaian Socket secara *Concurrent*

Pada sistem yang tidak mendukung komunikasi *duplex*, penggunaan *InputStream* dan *OutputStream* dilakukan secara bergantian. Dampak buruknya adalah total waktu pemrosesan data yang menjadi lebih lambat karena proses input tidak bisa dilakukan bersamaan dengan proses output. Pemakaian dua buah *thread* terpisah untuk input dan output juga tidak akan banyak membantu dalam hal ini, malah akan semakin memperumit struktur aplikasi.

4.2.2.2 Protokol IRC

Secara keseluruhan, perintah-perintah (*message*) yang terdapat dalam RFC 1459 mengenai protokol IRC sangatlah banyak tetapi tidak semuanya harus digunakan.

4.2.2.2.1 IRC Message

Dalam RFC 1459, disebutkan bahwa aturan penulisan *message* adalah menggunakan 'pseudo' BNF. *Message* harus dapat diambil dari paket data yang bersambung di jaringan (*contiguous stream*). Untuk saat ini, metode pemecahannya adalah menggunakan dua karakter, CR dan LF sebagai pemisah antar *message*. Dengan ketentuan tersebut, setiap *message* selalu terdiri dari baris-

baris yang diakhiri dengan sebuah pasangan karakter CRLF (*Carriage Return – Line Feed*), dan *message* tersebut panjangnya tidak boleh melebihi 512 karakter (termasuk karakter CRLF di dalamnya).

4.2.2.2.2 Format Message dalam 'pseudo' BNF

Message yang telah disaring masih dibagi-bagi lagi menjadi beberapa komponen, yaitu <prefix>, <command>, dan beberapa parameter yang diperlukan.

Berikut beberapa contoh format penulisan *message* menggunakan BNF:

```
<message> ::= [':' <prefix> <SPACE>] <command> <params> <crlf>
<prefix> ::= <servername> | <nick> ['!' <user>] ['@' <host>]
<command> ::= <letter> {<letter>} | <number> <number> <number>
<SPACE> ::= ' ' {' '}
<params> ::= <SPACE> [':' <trailing> | <middle> <params>]
<middle> ::= <Any *non-empty* sequence of octets not including
             SPACE or NUL or CR or LF, the first of which
             may not be ':'>
<trailing> ::= <Any, possibly *empty* sequence of octets
              not including NUL or CR or LF
<crlf> ::= CR LF
```

4.2.2.3 Struktur Aplikasi

Dalam pembuatan aplikasi menggunakan bahasa Java, mutlak digunakan pengkodean dengan berbasis kelas karena Java termasuk bahasa pemrograman yang murni OOP (*Object-Oriented Programming*), bahkan prosedur utamanya yang menjadi *entry point* aplikasi pun berada di dalam sebuah kelas. Pada tahap perancangan ini, prototipe *methods* yang ditampilkan hanya yang bersifat *public*

saja karena yang bersifat *private* biasanya hanya digunakan untuk mendukung atau mengurai operasi pada *methods* yang bersifat *public*.

Dengan didasarkan pada analisis kebutuhan yang telah dibuat sebelumnya, maka dapat dibuat perancangan kelas-kelas Java sebagai berikut:

a. Komponen untuk menangani koneksi TCP/IP.

Untuk menangani koneksi TCP/IP dapat dilakukan menggunakan kelas turunan *javax.microedition.io.StreamConnection* tetapi masih cukup rumit dilakukan. Oleh karena itu dibuatlah kelas *IrcSocket* yang menggunakan obyek *StreamConnection* di dalamnya. Lebih jelasnya dapat dilihat pada tabel 4.1.

Tabel 4.1 kelas *IrcSocket*

Nilai Balik	Prototipe / Deskripsi
-	`IrcSocket(java.lang.String host)` Constructor
`void`	`close()` Menutup socket
`boolean`	`isConnected()` Memeriksa status koneksi socket
`boolean`	`isError()` Memeriksa status kesalahan
`java.lang.String`	`readln()` Membaca satu baris data dari socket
`void`	`writeln(java.lang.String data)` Menulis satu baris data ke socket

b. Komponen untuk menangani protokol IRC.

Kelas *IrcConnection* dibuat untuk menyederhanakan proses-proses yang terjadi di level socket dan membungkusnya menjadi sebuah kelas yang lebih mudah dalam pengaksesan. Lebih jelasnya dapat dilihat pada tabel 4.2.

Tabel 4.2 kelas *IrcConnection*

Nilai Balik	Prototipe / Deskripsi
-	`IrcConnection(javax.microedition.midlet.MIDlet m)` Constructor
boolean	`ircNickname(java.lang.String nickname)` Menentukan nickname
void	`ircPart(java.lang.String channel)` Keluar dari channel
void	`ircQuit(java.lang.String quitmsg)` Keluar dari jaringan IRC
void	`ircReadMOTD(javax.microedition.lcdui.TextBox log)` Membaca MOTD (Message of The Day)
java.lang.String	`ircReadRawData()` Membaca data dari socket per baris
void	`ircRegister(java.lang.String username, java.lang.String localname, java.lang.String servername, java.lang.String realname)` Meregister user
void	`ircSendMessageToChannel(java.lang.String channel, java.lang.String message)` Mengirim pesan ke channel
void	`ircSendMessageToUser(java.lang.String user, java.lang.String message)` Mengirim pesan ke user
void	`ircSendRawData(java.lang.String rawdata)` Mengirim data ke socket per baris
boolean	`isConnected()` Memeriksa koneksi

Tabel 4.2 kelas *IrcConnection* (lanjutan)

Nilai Balik	Prototipe / Deskripsi
void	**readMessageQueue**() Membaca antrian pesan masuk untuk diproses ke tampilan
void	**startListener**() Mengaktifkan thread untuk pembacaan data dari socket

Kelas *IrcMessage* digunakan untuk membantu kelas IrcConnection dalam pemrosesan IRC Message ketika akan menampilkan pesan dari channel atau user lain ke display yang sesuai. Lebih jelasnya dapat dilihat pada tabel 4.3.

Tabel 4.3 kelas *IrcMessage*

Nilai Balik	Prototipe / Deskripsi
-	**IrcMessage**() Constructor tanpa parameter
-	**IrcMessage**(java.lang.String from, java.lang.String to, java.lang.String message) Constructor dengan parameter
java.lang.String	**getMessage**() Membaca message
java.lang.String	**getSender**() Mengetahui pengirim message
java.lang.String	**getTarget**() Mengetahui target
void	**setData**(java.lang.String from, java.lang.String to, java.lang.String message) Menulis satu baris data ke socket

c. Komponen untuk menangani tampilan (*display*).

Untuk menangani pengaturan tampilan tidak digunakan kelas yang khusus menanganinya, namun digabung dalam sebuah kelas bernama *IrcChannelUser* yang merupakan kelas yang paling kompleks karena paling banyak mengatur segala aktivitas dan proses di dalam channel serta user-user di jaringan IRC. Lebih jelasnya dapat dilihat pada tabel 4.4.

Tabel 4.4 kelas *IrcChannelUser*

Nilai Balik	**Prototipe / Deskripsi**
-	`IrcChannelUser(javax.microedition.lcdui.Display d, IrcConnection c, IrcMain m)` Constructor
`javax.microedition.lcdui.TextBox`	`createChannel(java.lang.String channel)` Membuat antarmuka untuk channel
`void`	`createJoin()` Membuat antarmuka untuk join ke channel
`javax.microedition.lcdui.List`	`createNames(java.lang.String channel, java.lang.String names)` Membuat daftar user untuk suatu channel
`javax.microedition.lcdui.TextBox`	`createUser(java.lang.String user)` Membuat antarmuka untuk chat user
`int`	`findChannel(java.lang.String channel)` Mencari display untuk suatu channel
`int`	`findNames(java.lang.String channel)` Mencari display yang berisi daftar user untuk suatu channel
`int`	`findUser(java.lang.String user)` Mencari display untuk user chat

Tabel 4.4 kelas *IrcChannelUser* (lanjutan)

Nilai Balik	Prototipe / Deskripsi
void	`selectUserOnChannel()` Memilih user di channel
void	`showChangeNickname()` Menampilkan antarmuka untuk mengganti nickname
void	`showChatUser()` Menampilkan antarmuka untuk melakukan chat ke user lain secara langsung
void	`showDisplay()` Menampilkan daftar display
boolean	`showDisplayToTarget(IrcMessage pm)` Mengaktifkan display untuk target yang dituju
void	`showJoin()` Menampilkan antarmuka join ke channel

d. Komponen untuk menangani lingkungan J2ME.

Kelas *IrcApplication* merupakan kelas turunan dari kelas MIDlet yang mengatur hubungan aplikasi utama ke MIDlet. Dalam kelas ini tidak terdapat *methods* yang mengatur pemrosesan mapun tampilan tetapi kelas ini mengendalikan penciptaan obyek *IrcConnection* dan *IrcMain*. Lebih jelasnya dapat dilihat pada tabel 4.5.

Tabel 4.5 kelas *IrcApplication*

Nilai Balik	Prototipe / Deskripsi
-	`IrcApplication()` Constructor
void	`destroyApp(boolean unconditional)` Aplikasi mengalami *destroy*
void	`pauseApp()` Aplikasi mengalami *pause*
void	`startApp()` Aplikasi mulai dijalankan

e. Komponen utama untuk mengawasi dan mengendalikan seluruh komponen lain.

Kelas *IrcMain* adalah kelas utama pengendali perpindahan antarmuka untuk tiap proses sekaligus menghubungkannya ke obyek-obyek *engine* (pemroses). Lebih jelasnya dapat dilihat pada tabel 4.6.

Tabel 4.6 kelas *IrcMain*

Nilai Balik	Prototipe / Deskripsi
-	`IrcMain(javax.microedition.midlet.MIDlet m, IrcConnection c)` Constructor
void	`showStatus()` Menampilkan antarmuka Status
void	`startMain()` Mengaktifkan thread utama

Selain kelas-kelas tersebut di atas, terdapat sebuah kelas yang berisi kumpulan fungsi-fungsi tertentu yang tidak berkaitan langsung dengan pemrosesan dan tampilan namun cukup membantu menyederhakan pengkodean program. Kelas tersebut dinamakan kelas *IrcFunction*. Lebih jelasnya dapat dilihat pada tabel 4.7.

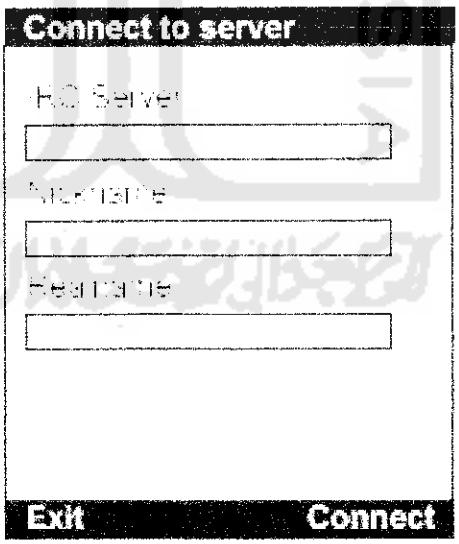
Tabel 4.7 kelas *IrcFunction*

Nilai Balik	Prototipe / Deskripsi
-	`IrcFunction()` Constructor
void	`appendTextBox(javax.microedition.lcdui.TextBox tb, java.lang.String s)` Menambahkan isi data di TextBox
void	`delay(int miliseconds)` Menunggu selama selang waktu tertentu
int	`getParsedCount(java.lang.String data, java.lang.String delimiter)` Menghitung jumlah indeks dari data yang di-parsing
java.lang .String	`getParsedString(java.lang.String data, java.lang.String delimiter, int index)` Melakukan parsing data
void	`msgError(javax.microedition.lcdui.Display d, java.lang.String msg)` Menampilkan pesan error ke display
void	`msgInfo(javax.microedition.lcdui.Display d, java.lang.String msg)` Menampilkan pesan info ke display

4.2.3 Rancangan Antarmuka

Rancangan antarmuka dibuat dengan berusaha memperhatikan faktor usabilitas sehingga interaksi antara pemakai aplikasi tidak menyulitkan dalam hal penggunaannya. Dalam teori-teori usabilitas yang berkaitan dengan sisi interaksi manusia dengan aplikasi, banyak ditekankan beberapa hal penting yang harus dipertimbangkan ketika merancang antarmuka. Salah satunya adalah kesederhanaan dan kemudahan dalam memahami antarmuka aplikasi sehingga terhindar dari kerancuan dan kesalahan mengakses suatu fasilitas.

Untuk melakukan koneksi ke server, diharuskan mengisikan data-data server IRC (misal: irc.allnetwork.org), Nickname (nama samaran/julukan/panggilan), dan Realname (nama asli atau nama lengkap). Lebih jelasnya dapat dilihat pada gambar 4.20.

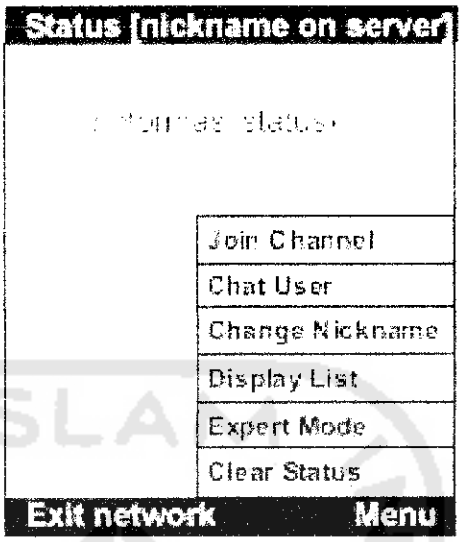


Gambar 4.20 Rancangan antarmuka untuk melakukan koneksi ke server

Apabila data-data yang dimasukkan valid, maka aplikasi melakukan koneksi dan melakukan registrasi user ke jaringan IRC. Antarmuka hanya berfungsi untuk memonitor status koneksi saja. Apabila proses koneksi dan registrasi berhasil, maka di bagian *title* akan ditampilkan nickname dan nama server. Lebih jelasnya dapat dilihat pada gambar 4.21.

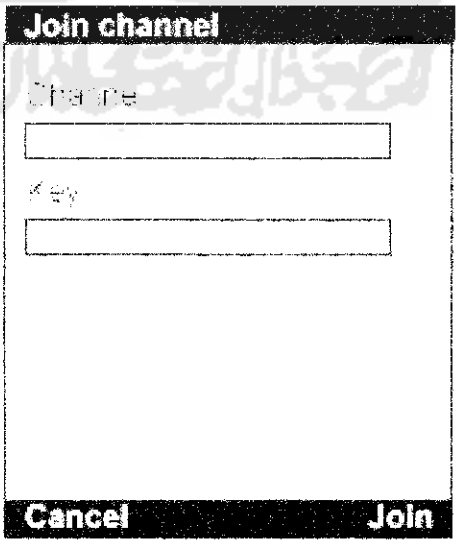
Gambar 4.21 Rancangan antarmuka untuk mengetahui status user

Apabila proses registrasi telah berhasil, maka user telah dapat menggunakan fasilitas-fasilitas di jaringan IRC. Aplikasi J2ME IRC Client memiliki fitur-fitur utama yang dapat diakses melalui menu utama pada display Status. Menu yang ditampilkan yaitu Join Channel , Chat User, Change Nickname, Display List, Expert Mode, dan Status. Lebih jelasnya dapat dilihat pada gambar 4.22.



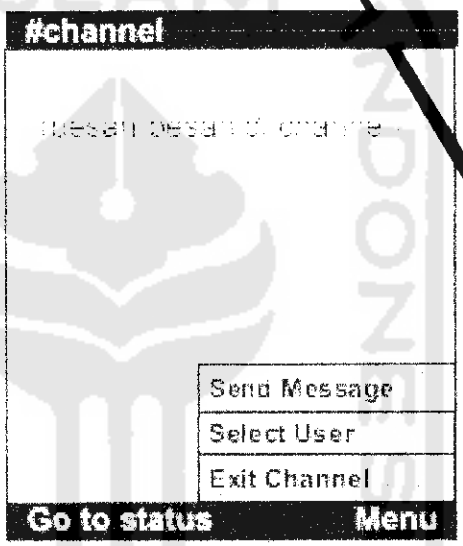
Gambar 4.22 Rancangan antarmuka untuk menu utama

Untuk melakukan join ke channel dilakukan dengan cara memilih menu join channel. Pada display Join channel terdapat kotak masukan untuk nama channel dan *key*. Tiap channel yang tertutup untuk umum biasanya mengharuskan user untuk memasukkan *key* (semacam *password*). Lebih jelasnya dapat dilihat pada gambar 4.23.



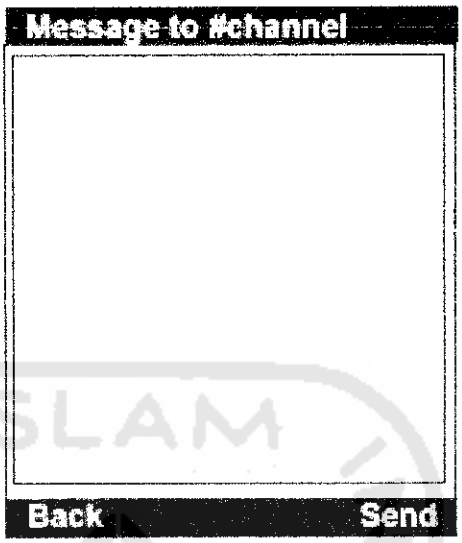
Gambar 4.23 Rancangan antarmuka untuk melakukan join ke channel

Apabila user telah memasuki channel, bagian *title* akan menampilkan nama channel tersebut dan display utama akan menampilkan pesan-pesan publik yang dikirim oleh user-user. Untuk menggunakan fitur-fitur channel, dapat menggunakan menu yang ada dalam display Channel. Menu-menu tersebut digunakan untuk mengirim pesan ke channel, mengirim pesan ke user dalam channel, dan keluar dari channel. Lebih jelasnya dapat dilihat pada gambar 4.24.



Gambar 4.24 Rancangan antarmuka untuk menu channel

Untuk mengirim pesan ke channel dilakukan dengan memilih menu Send Message. Pemasukan data dilakukan menggunakan TextBox, bukan menggunakan TextField agar apabila teks yang dimasukkan panjang dapat dilihat dan dibaca dengan lebih leluasa. Lebih jelasnya dapat dilihat pada gambar 4.25.

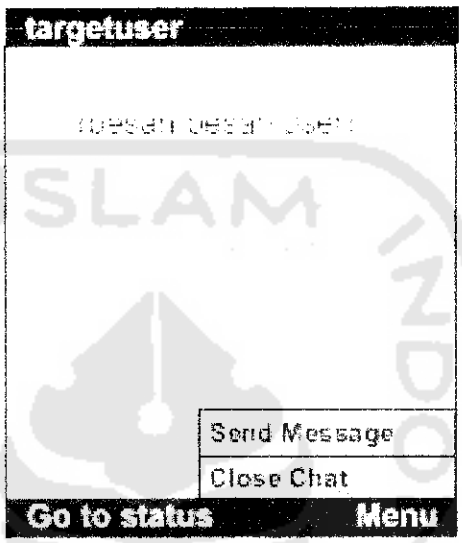


Gambar 4.25 Rancangan antarmuka untuk mengirim pesan ke channel

Untuk mengirim pesan ke user di dalam channel, maka sebelumnya diharuskan memilih dahulu salah satu user tujuan yang ada di channel. Hal tersebut dilakukan dengan memilih menu Select User yang akan menampilkan daftar user yang dapat dipilih. Lebih jelasnya dapat dilihat pada gambar 4.26.

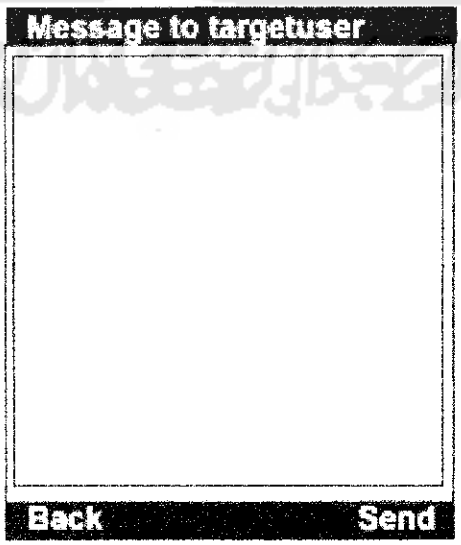
Gambar 4.26 Rancangan antarmuka untuk menampilkan user-user di channel

Antarmuka yang digunakan mirip dengan antarmuka ketika mengirim pesan ke channel, dengan sedikit perbedaan pada bagian menu dan *title*. Lebih jelasnya dapat dilihat pada gambar 4.27.



Gambar 4.27 Rancangan antarmuka untuk menu user

Apabila pesan siap dikirimkan, pilih menu Send untuk mengirim pesan ke user. Lebih jelasnya dapat dilihat pada gambar 4.28.



Gambar 4.28 Rancangan antarmuka untuk mengirim pesan ke user

Ketika melakukan koneksi ke jaringan, kadang-kadang nickname yang dimasukkan ternyata sudah digunakan oleh orang lain. Oleh karena itu herus dilakukan penggantian nickname terlebih dahulu sebelum proses registrasi ke jaringan dapat dilanjutkan. Untuk mengganti nickname dapat dilakukan dengan memilih menu Change Nickname. Lebih jelasnya dapat dilihat pada gambar 4.29.

Gambar 4.29 Rancangan antarmuka untuk mengganti nickname

Ketika user melakukan join ke beberapa channel atau melakukan pertukaran pesan dengan beberapa user, akan banyak terdapat display untuk masing-masing komunikasi tersebut. Pada satu waktu layar hanya dapat menampilkan satu tampilan (display) saja sehingga akan ada banyak display yang berada di *background process*. Untuk menampilkan dan memilih display-display tersebut dapat dilakukan dengan memilih menu Display List. Lebih jelasnya dapat dilihat pada gambar 4.30.

Gambar 4.30 Rancangan antarmuka untuk menampilkan daftar display

Ada satu tambahan fasilitas bagi pengguna IRC berpengalaman yang telah banyak mengenal protokol IRC, yaitu Expert Mode. Secara teknis, pada modus ini pemakai mengirim protokol langsung ke socket TCP/IP. Expert Mode ini merupakan antisipasi yang dilakukan dalam penelitian ini agar nantinya aplikasi tetap dapat mengakses seluruh fasilitas IRC apabila diperlukan, karena fitur-fitur dalam pembuatan aplikasi J2ME IRC Client sangat sederhana dan tidak menyediakan antarmuka khusus untuk fitur-fitur lanjutan (*advanced features*).

Sebagai solusi, Expert Mode akan berguna pada fitur-fitur lanjutan tersebut seperti pengaturan mode channel, melakukan KICK dan BAN pada channel, serta mengakses layanan ChanServ dan NickServ. Untuk rancangan antarmuka Expert Mode, lebih jelasnya dapat dilihat pada gambar 4.31.

Gambar 4.31 Rancangan antarmuka untuk display Expert Mode