IMPLEMENTASI JAVA SOCKET PADA GAME JARINGAN GO-FISH

TUGAS AKHIR

Diajukan sebagai Salah Satu Syarat

Untuk Memperoleh Gelar Sarjana

Jurusan Teknik Informatika



Disusun oleh :

Nama : Oki Prasetia Sakti

NIM : 06523135

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

YOGYAKARTA

2012

LEMBAR PENGESAHAN PEMBIMBING

IMPLEMENTASI JAVA SOCKET PADA GAME JARINGAN GO-FISH

TUGAS AKHIR

Disusun Oleh :

Nama : Oki Prasetia Sakti

NIM : 06 523 135

Yogyakarta, 27 Februari 2012

Dosen Pembimbing Tunggal

R.Teduh Dirgahayu , ST., M.Sc., Ph.D

LEMBAR PERNYATAAN KEASLIAN HASIL TUGAS AKHIR

Saya yang bertandatangan di bawah ini,
Nama : Oki Prasetia Sakti
No. Mahasiswa : 06 523 135

Menyatakan bahwa seluruh komponen dan isi dalam Laporan Tugas Akhir ini adalah hasil karya saya sendiri. Apabila dikemudian hari terbukti bahwa ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, maka saya siap menanggung resiko dan konsekuensi apapun.
Demikian pernyataan ini saya buat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 27 Feburari 2012

Oki Prasetia Sakti

LEMBAR PENGESAHAN PENGUJI

IMPLEMENTASI JAVA SOCKET PADA GAME JARINGAN GO-FISH

TUGAS AKHIR

oleh :

Nama : Oki Prasetia Sakti

No. Mahasiswa : 06 523 135

Telah Dipertahankan di Depan Sidang Penguji Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Informatika Fakultas
Teknologi Industri Universitas Islam Indonesia

Yogyakarta,

Tim Penguji

R.Teduh Dirgahayu , ST., M.Sc., Ph.D
Ketua

Syarif Hidayat, S.Kom., MIT.
Anggota I

Ahmad M. Rafie Pratama, ST., MIT.
Anggota II

Mengetahui,
Ketua Jurusan Teknik Informatika
Universitas Islam Indonesia

Yudi Prayudi, S.SI, M.Kom.

HALAMAN PERSEMBAHAN

Allah SWT

Yang selalu melimpahkan Rahmat dan Hidayahny, memberikanku akal pikiran untuk berkarya,serta anugrah yang tak terhitung banyaknya

Kedua Orang Tuaku

Terimakasih atas kasih sayang, doa, dan jerih payahnya mendidik hingga saat ini,serta dukungan dalam setiap langkahku.

MOTTO

"Jika kamu tidak memiliki apa yang kamu sukai, maka sukailah apa yang kamu miliki saat ini"

“Belajarlah dari orang lain dan catat kunci sukses mereka kemudian belajarlah untuk menjadi lebih baik dari mereka untuk orang-orang disekitar kita”

“Lakukan segala sesuatu dengan ikhlas untuk mendapat RidhoNya”

KATA PENGANTAR

Assalamualaikum Wr. Wb.

Puji dan syukur penulis panjatkan kehadirat Allah SWT yang telah memberikan petunjuk, kekuatan, kemampuan, serta pengetahuan dan kehidupan yang sangat berharga, shalawat serta salam kepada utusan Allah yang paling Mulia Nabi besar Muhammad SAW yang telah mencintai ummatnya dan menjadi teladan yang menghantarkan pada kemuliaan dan kebahagiaan hidup didunia maupun diakhirat kelak.

Setelah melalui proses yang panjang dengan upaya keras dan dukungan dari berbagai pihak, maka penulis akhirnya dapat menyelesaikan tugas akhir yang berjudul Rekayasa Protokol Game Jaringan Go-Fish menggunakan Java Socket. Laporan tugas akhir ini adalah sebagai salah satu syarat untuk mendapatkan gelar sarjana pada jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.

Dalam pelaksanaan tugas akhir dan pembuatan laporan tugas akhir ini penulis mendapat dukungan dan bantuan dari berbagai pihak, penulis hendak mengucapkan terimakasih sebesar-besarnya kepada:

1. Allah SWT atas segala berkah dan rahmat-Nya sehingga Tugas Akhir ini dapat diselesaikan.
2. Orang tua, kakak, dan adik atas kasih sayang, segala limpahan doa, dan dukungannya.
3. Bapak Gumbolo Hadi Susanto, Ir., M.Sc. selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
4. Ketua Jurusan Teknik Informatika FTI UII Bapak Yudi Prayudi S.Si.,M.Kom.
5. Bapak Teduh Dirgahayu selaku dosen pembimbing tunggal dalam penyusunan Tugas Akhir ini.

6. Dosen pengajar di Teknik Informatika UII yang memberikan banyak ilmu. Semoga menjadi ilmu yang bermanfaat.
7. Teman-teman angkatan 2006 (FIRE) di Jurusan Teknik Informatika Universitas Islam Indonesia.
8. Terima kasih juga kepada semua teman-teman yang tidak dapat disebutkan satu persatu, yang telah memberikan dukungan, semangat dan ilmu sehingga terselesaikannya laporan ini.

Penulis menyadari bahwa penyusunan laporan ini tidak luput dari kekurangan dan kesalahan, untuk itu penulis sangat mengharapkan kritik dan saran demi kesempurnaan laporan ini.

Akhir kata semoga laporan ini dapat bermanfaat bagi para pembaca khususnya rekan-rekan mahasiswa dalam mengembangkan ilmu pengetahuan.

Yogyakarta, 27 Feburari 2012

Oki Prasetia Sakti

SARI

Permainan kartu remi merupakan salah satu permainan yang cukup banyak dimainkan oleh masyarakat di Indonesia, salah satu jenis permainan kartu remi yang sering dimainkan adalah cangkulan (*go-fish*). Dengan pesatnya perkembangan teknologi sekarang ini permainan kartu juga bisa dijalankan di komputer dan juga bisa dimainkan di sebuah jaringan (LAN) secara multiplayer.

Socket adalah *link* komunikasi dua arah antara dua buah program yang berjalan di jaringan. Agar dua komputer bisa saling berkomunikasi harus terpasang socket pada tiap mesin komputer. Bahasa pemograman yang sudah mendukung *socket programming* salah satunya adalah bahasa pemograman Java.

Dalam tugas akhir ini diharapkan menjadi gambaran bagaimana merancang sebuah game jaringan berbasis client-server menggunakan socket dengan bahasa pemograman Java.

Kata Kunci : client-server, game, socket, multiplayer

TAKARIR

Client	pengguna layanan
Go-fish	permainan kartu cangkulan
IPC	komunikasi antar proses
Interface	antarmuka
Layer	lapisan
Level	tingkat
Low-level	lapisan bawah
Multiplayer	banyak pemain
Network	jaringan
Server	penyedia layanan
Socket	titik akhir dari *link* komunikasi dua arah antara dua program yang berjalan pada jaringan
Stand-alone	berdiri sendiri

DAFTAR ISI

HALAMAN JUDUL i

LEMBAR PENGESAHAN PEMBIMBING ..... **Error! Bookmark not defined.**

LEMBAR PERNYATAAN KEASLIAN HASIL TUGAS AKHIR iii

LEMBAR PENGESAHAN PENGUJI ..... **Error! Bookmark not defined.**

HALAMAN PERSEMBAHAN v

MOTTO vi

KATA PENGANTAR vii

SARI ix

TAKARIR x

DAFTAR ISI xi

BAB I PENDAHULUAN 1

1.1 Latar Belakang 1

1.2 Rumusan Masalah 1

1.3 Batasan Masalah 2

1.4 Tujuan Penelitian 2

1.5 Manfaat Penelitian 2

1.6 Metodologi Penelitian 2

1.7 Sistematika Penulisan 3

BAB II LANDASAN TEORI 5

2.1 Game Go-Fish 5

2.1.1 Peraturan dan cara main dalam permainan *go-fish* 5

2.1.2 Peraturan dan cara main dalam permainan *go-fish* dalam sistem 6

2.2 Konsep Dasar Protokol 7

2.2.1 TCP/IP 8

2.3 Socket 9

2.4 Model Aplikasi Client-Server 11

2.5 Pemograman Socket di Java 12

2.6 Port 13

BAB III METODOLOGI 14

3.1 Gambaran Arsitektur Sistem 14

3.2 Perangkat Keras dan Perangkat Lunak yang Dibutuhkan 16

3.2.1 Perangkat Keras yang Dibutuhkan 16

3.2.2 Kebutuhan Perangkat Lunak yang Dibutuhkan 16

3.3 Metode Perancangan Perangkat Lunak 17

3.3.1 Class Diagram 17

3.3.2 Activity Diagram 19

3.3.3 Sequence Diagram 22

BAB IV HASIL DAN PEMBAHASAN 25

4.1 Hasil.... 25

4.2 Implementasi 25

4.3 Batasan Implementasi 25

4.4 Pengujian hasil implementasi 25

4.4.1 Tampilan server ketika sedang dijalankan 26

4.4.2 Tampilan server ketika client bergabung dengan server 26

4.4.3 Tampilan client ketika melakukan koneksi ke server 27

4.4.4 Tampilan client ketika terdapat pemain lain yang bergabung ke permainan. 27

4.4.5 Tampilan ketika empat pemain sudah memasuki server 28

4.4.6 Tampilan client saat permainan dimulai 29

4.4.7 Tampilan client saat memilih menu pilih kartu 29

4.4.8 Tampilan client saat belum boleh melempar kartu 30

4.4.9 Tampilan client saat menerima informasi kartu yang dilempar oleh pemain lain. 31

4.4.10 Tampilan client ketika melempar jenis kartu yang berbeda 31

4.4.11 Tampilan client mendapatkan informasi pemenang putaran 32

4.4.12 Tampilan client saat kartunya sudah habis 33

4.4.13 Tampilan client saat salah satu pemain menghabis kartu 33

4.4.14 Tampilan client saat pemain terakhir menghabiskan kartunya 34

BAB V KESIMPULAN DAN SARAN 35

5.1 Kesimpulan 35

5.2 Saran 35

DAFTAR PUSTAKA 36

DAFTAR GAMBAR

Gambar 2.1 Layer TCP/IP 8

Gambar 2.2 Model *IPC* dengan socket..... 9

Gambar 2.3 Model Aplikasi *Client-Server* pada protokol TCP 11

Gambar 3.1 *socket* pada *client-server*. 14

Gambar 3.2 Komunikasi client-server..... 15

Gambar 3.3 *Class* pada *game go-fish* dan relasi antar *class*nya..... 18

Gambar 3.4 *Activity* diagram keseluruhan sistem. 20

Gambar 3.5 Sequence diagram koneksi client ke server..... 22

Gambar 3.6 Pertukaran data client dengan thread. 23

Gambar 3.7 Pertukaran data client,thread dan server..... 24

Gambar 4.1 Tampilan server ketika baru dijalankan..... 26

Gambar 4.2 Tampilan server saat menerima koneksi dari client 26

Gambar 4.3 Tampilan client ketika pertama kali memasuki server 27

Gambar 4.4 Tampilan client ketika pemain yang lain bergabung ke server 28

Gambar 4.5 Tampilan ketika ke empat pemain memasuki server..... 28

Gambar 4.6 Tampilan client memulai permainan 29

Gambar 4.7 Tampilan client saat memilih menu pilih kartu 30

Gambar 4.8 Tampilan client belum diperbolehkan melempar kartu..... 30

Gambar 4.9 Informasi dari server tentang kartu yang dilempar setiap pemain..... 31

Gambar 4.10 Tampilan client ketika jenis kartu yang dilempar berbeda..... 32

Gambar 4.11 Tampilan client ketika pengumuman pemenang putaran 32

Gambar 4.12 Tampilan ketika kartu pemain sudah habis 33

Gambar 4.13 Tampilan client mendapatkan informasi pemenang permainan 34

Gambar 4.14 Tampilan client saat permainan selesai. 34

BAB I

PENDAHULUAN

1.1 Latar Belakang

Permainan kartu remi merupakan permainan yang sering dimainkan oleh orang banyak sejak dahulu untuk mengisi waktu luang dan juga ada yang dijadikan sebagai sebuah hobi. Jenis permainan kartu disetiap negara memiliki jenis permainannya sendiri. Di Indonesia dikenal dengan istilah permainan "41","remi","cangkulan" dan sebagainya. Dengan semakin berkembangnya teknologi sekarang ini, permainan kartu ini juga bisa dimainkan di sebuah komputer.

Sejak awal perkembangan TCP/IP dan internet hingga sekarang, semakin banyak layanan yang diberikan oleh jaringan TCP/IP dan di antaranya memberi dampak positif yang besar terhadap industri-industri pembuat *game*. Bagi para pembuat dan pengembang *game* ataupun yang tertarik untuk membuat *game*, tentunya sangat menginginkan *game* yang dihasilkan tidak hanya berjalan secara *stand-alone* tetapi juga di dalam jaringan (*multi player*) [JN07].

Socket adalah mekanisme komunikasi yang memungkinkan terjadinya pertukaran data antar program atau proses baik dalam satu mesin maupun antar mesin. Komunikasi *socket* terutama diciptakan untuk tujuan menjembatani komunikasi antara dua buah program yang dijalankan pada mesin yang berbeda (jaringan). Kelebihan lain dari komunikasi *socket* adalah mampu menangani banyak klien sekaligus (*multiple clients*).

1.2 Rumusan Masalah

Bagaimana merancang dan membuat game multiplayer menggunakan Java socket.

1.3 Batasan Masalah

1. Aplikasi *game* berbasis *console* (command prompt).
2. Jumlah pemain dibatasi empat pemain.
3. Aplikasi pada client hanya menampilkan respon dari server.
4. Tidak menggunakan sinkronisasi antar thread.

1.4 Tujuan Penelitian

Tujuan melakukan penelitian ini adalah untuk merancang aplikasi *game* berbasis *client-server* supaya *client* dan *server* bisa saling berkomunikasi menggunakan *socket*.

1.5 Manfaat Penelitian

Penelitian ini memiliki manfaat antara lain:

Memberikan gambaran bagaimana perancangan komunikasi antara client dan server menggunakan Java *socket*.

1.6 Metodologi Penelitian

Tahapan yang diambil dalam penelitian ini adalah:

1. Studi literatur

   Memperoleh informasi dengan mengumpulkan, membaca dan mempelajari berbagai referensi dari buku, jurnal, makalah dan tulisan ilmiah lainnya yang dibutuhkan dalam penulisan tugas akhir ini.

2. Analisis Kebutuhan

   Menganalisa proses permainan pada *game go-fish* yang hendak diimplementasikan ke dalam aplikasi komputer serta menganalisis peran *client server* dalam permainan tersebut.

3. Perancangan Sistem

   Merancang game dengan Java yang diterapkan ke dalam sebuah server game sehingga antara *server* dan *client* bisa saling berkomunikasi dengan *socket*.

4. Pengujian

   Melakukan pengujian dengan melakukan komunikasi antara *client* dan *server* dengan mengimplementasikannya pada game go-fish.

1.7 Sistematika Penulisan

BAB I Pendahuluan

Membahas tentang latar belakang masalah, batasan masalah, rumusan masalah, tujuan penelitian serta manfaat dari penelitian dan metodologi penelitian yang diangkat menjadi materi penulisan laporan tugas akhir ini.

BAB II Landasan Teori

Membahas dasar-dasar teori yang digunakan dalam mengimplementasikan game jaringan dengan Java *socket*. Dasar teori tersebut berisi peraturan permainan *game go-fish* , TCP/IP, protokol TCP, perbedaan protokol TCP dan UDP, arsitektur *client-server*, dan pemrograman socket di Java.

BAB III Metodologi

Memuat uraian tentang gambaran umum sistem, analisis kebutuhan sistem yang mencakup kebutuhan perangkat keras dan perangkat lunak, yang digunakan untuk penyelesaian tugas akhir.

BAB IV Hasil dan Pembahasan

Memuat dokumentasi dari hasil implementasi Java *socket* mulai dari tahap analisis kebutuhan, perancangan dan pengujian, yaitu dengan mencoba melakukan komunikasi antara *client* dan s*erver* dengan mengimplementasikannya ke dalam *game go-fish*.

BAB V Kesimpulan dan Saran

Memuat kesimpulan-kesimpulan dari seluruh rangkaian proses perancangan game, baik pada tahap analisis, perancangan, implementasi. Bab ini juga membahas saran yang dapat digunakan oleh pihak yang berkepentingan maupun untuk peneliti terhadap kekurangan serta keterbatasan dalam penelitian ini.

BAB II

LANDASAN TEORI

2.1 Game Go-Fish

Permainan *Go-Fish* atau lebih dikenal dengan nama cangkulan adalah sebuah permainan kartu remi yang sering dimainkan oleh orang-orang untuk mengisi waktu luangnya, biasanya dimainkan oleh 3 atau 4 orang pemain atau lebih. Kartu remi adalah sekumpulan kartu seukuran tangan yang digunakan untuk permainan kartu. Kartu ini sering juga digunakan untuk hal-hal lain, seperti sulap, enkripsi dan pembuatan rumah kartu. Satu set kartu remi berisi 52 lembar, dibagi menjadi 4 jenis kartu (sekop, hati, wajik, keriting), masing-masing terdiri atas 13 kartu (dari 2 sampai 10 lalu *jack, queen, king* dan *ace*).

2.1.1 Peraturan dan cara main dalam permainan *go-fish*

Berikut adalah aturan-aturan yang berlaku dan cara main permainan *go-fish* (cangkulan) yang sebenarnya yang biasa dimainkan, aturannya sebagai berikut :

1. Tiap-tiap pemain mendapatkan kartu sebanyak tujuh kartu.
2. Pelempar kartu pada putaran pertama harus mengikuti jenis kartu dari kartu yang diambil dari tumpukan kartu.
3. Para pemain melempar kartu yang nantinya akan dibandingkan, siapa yang paling tinggi nilai kartunya keluar sebagai pemenang dan berhak melempar kartu pertama berikutnya.
4. Setiap kartu yang dilempar, jenis kartunya harus sama dengan jenis kartu dari pemain yang pertama kali melempar kartu disetiap putaran.
5. Jika pemain tidak mempunyai jenis kartu yang sama untuk dilempar maka pemain tersebut harus mengambil kartu dari tumpukan kartu.
6. Tiap pemain saling berurutan ketika melempar kartu.
7. Jika tumpukan kartu habis maka pemain yang tidak mempunyai jenis kartu yang sama untuk dilempar maka pemain tersebut akan dilewat tidak melempar kartu satu putaran.

8. Jika pemain yang mendapat giliran melempar kartu belum melempar kartunya, maka pemain yang lain belum boleh melempar kartunya.
9. Pemain akan keluar sebagai pemenang jika pemain tersebut sudah tidak lagi memegang kartu atau kartunya sudah habis.
10. Permainan selesai jika semua pemain sudah menghabiskan kartu yang dipegang atau terdapat seorang pemain yang kartunya belum habis.

2.1.2 Peraturan dan cara main dalam permainan *go-fish* dalam sistem

Pada sistem yang akan dibuat adalah permainan kartu yang menyerupai permainan *go-fish* (cangkulan) yang aturan permainannya umumnya mengikuti aturan permainan *go-fish* (cangkulan) yang sebenarnya, namun pada sistem yang akan dibuat terdapat beberapa peraturan yang tidak mengikuti aturan sebenarnya, berikut aturan dan tata cara main *game go-fish* (cangkulan) di sistem :

1. Pada penelitian ini, jumlah pemain dibatasi tepat 4 pemain, jika jumlah pemain belum 4 pemain maka pemain yang lain menunggu sampai jumlah pemain terpenuhi untuk memulai permainan.
2. Tiap-tiap pemain mendapatkan kartu sebanyak 13 kartu.
3. Pemain yang pertama kali melempar saat putaran pertama permainan adalah pemain yang pertama masuk ke *server* permainan.
4. Tidak ada urutan pemain melempar kartu kedua, ketiga dan keempat, setelah pemain pertama melempar kartu maka pemain yang lain bisa langsung melempar kartu.
5. Jika pemain tidak memiliki jenis kartu yang sama dengan jenis kartu yang ditandingkan maka pemain tersebut akan dilewat satu putaran.
6. Tidak ada proses pemain mengambil kartu dari tumpukan kartu jika tidak mempunyai jenis kartu yang sama dengan yang ditandingkan karena semua kartu langsung dibagikan ke semua pemain pada saat awal permainan.
7. Jika seorang pemain yang melempar kartu terakhir dan pemain lainnya tidak melempar kartu maka giliran pertama melempar kartu pada putaran berikutnya diatur oleh *server*.

8. Permainan selesai jika semua pemain sudah melempar semua kartunya.

2.2 Konsep Dasar Protokol

Protokol dapat dimisalkan sebagai penerjemah dua orang yang berbeda bahasa ingin berkomunikasi. Protokol mengatur bagaimana sebuah komputer dapat berkomunikasi dengan baik dengan komputer lain. Oleh karena itu, komunikasi tidak tergantung dari sistem operasi, melainkan tergantung pada protokol.

Standar protokol Internet ada dua yakni OSI (*Open System Interconnection*), dan TCP/IP, namun pada akhirnya TCP/IP menjadi standar *de facto* yaitu standar yang diterima karena pemakaiannya secara sendirinya semakin berkembang. Pada dasarnya, komunikasi data merupakan proses mengirimkan data dari satu komputer ke komputer yang lain. Untuk dapat mengirimkan data, pada komputer harus ditambahkan alat khusus, yang dikenal sebagai *network interface* (*interface* jaringan) .

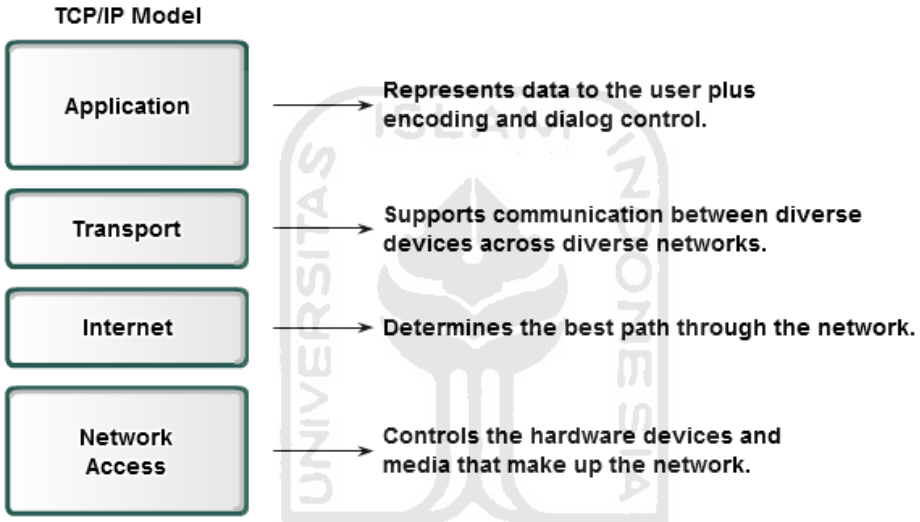
Dalam proses pengiriman data ini terdapat beberapa masalah yang harus dipecahkan. Pertama, data harus dapat dikirimkan ke komputer yang tepat, sesuai tujuannya. Hal lain yang perlu diperhatikan ialah, pada komputer tujuan pengirimian data mungkin terdapat lebih dari satu aplikasi menunggu datangnya data. Data yang dikirim harus sampai ke aplikasi yang tepat pada komputer yang tepat tanpa kesalahan [PB00].

Cara alamiah untuk menghadapi setiap masalah yang rumit ialah memecahkan masalah tersebut menjadi bagian yang lebih kecil. Dalam memecahkan masalah pengiriman data di atas, para ahli jaringan komputer pun melakukan hal yang sama. Untuk setiap masalah komunikasi data, diciptakan solusi khusus berupa aturan-aturan untuk menangani masalah tersebut. Untuk menangani semua masalah komunikasi data, keseluruhan aturan ini harus bekerja sama satu dengan lainnya. Sekumpulan aturan untuk mengatur proses pengiriman data ini disebut sebagai protokol komunikasi data.

2.2.1 TCP/IP

TCP/IP adalah sekumpulan protokol komunikasi data pada jaringan. Masing-masing protokol bertanggung jawab atas bagian-bagian tertentu dari komunikasi data. Berkat prinsip ini, tugas masing-masing protokol menjadi jelas dan sederhana. Protokol yang satu tidak perlu mengetahui cara kerja protokol yang lain, sepanjang ia masih bisa saling mengirim dan menerima data.

TCP/IP ini dimodelkan dengan 4 layer, sebagaimana terlihat pada gambar 2.1 di bawah ini.



Gambar 2.1 Layer TCP/IP [RR10].

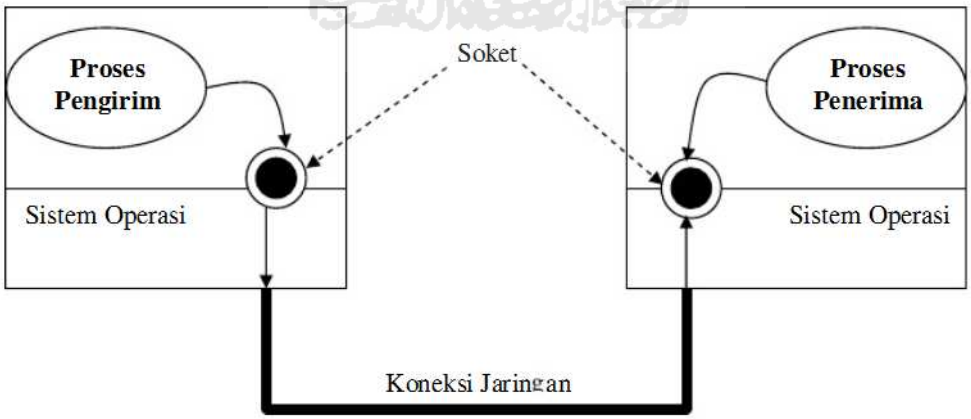
Masing-masing layer tersebut memiliki tanggungjawab yang berbeda, yaitu :

1. *Application Layer*, layer ini menangani permintaan pemakai untuk mengirim dan menerima data. Contoh layanan ini adalah FTP, SNMP, dan HTTP.
2. *Transport Layer*, layer ini bertanggungjawab menyediakan aliran data antara *host* asal dan *host* tujuan untuk layer aplikasi diatasnya agar dapat saling berkomunikasi. Terdapat dua protokol penting dalam layer ini, yaitu :

a. *Transmission Control Protocol* (TCP), berfungsi untuk melakukan transmisi data per-segmen (paket data dipecah-pecah dalam jumlah yang sesuai dengan besaran paket kemudian dikirim satu persatu hingga selesai), ketika ada kerusakan dalam paket data, maka TCP akan mengirimkan pesan *error* kepada komputer tujuan dan paket dikirim kembali, karena hal inilah TCP biasa disebut dengan *connection oriented protocol*.
b. *User Datagram Protocol* (UDP), berbeda dengan TCP, UDP merupakan protokol yang tidak reliable dan *connectionless*, ketika terjadi kegagalan dalam pengiriman paket data, UDP tidak akan mengirimkan pemberitahuan pesan *error* dan tidak mengirim ulang paket data.

3. *Internet Layer,* layer ini bertanggung jawab dalam proses pengiriman paket ke alamat yang tepat [SE04].
4. *Network Access Layer*, layer ini bertanggung jawab membuat koneksi fisik ke media jaringan, termasuk hubungan *device driver* pada sistem operasi dengan kartu jaringan pada komputer, dan proses enkapsulasi paket data.

2.3 Socket



Gambar 2.2 Model *IPC* dengan socket [WS11].

pada gambar 2.2 merupakan komunikasi dengan *socket*. *Socket* adalah salah satu titik akhir dari *link* komunikasi dua arah antara dua program yang

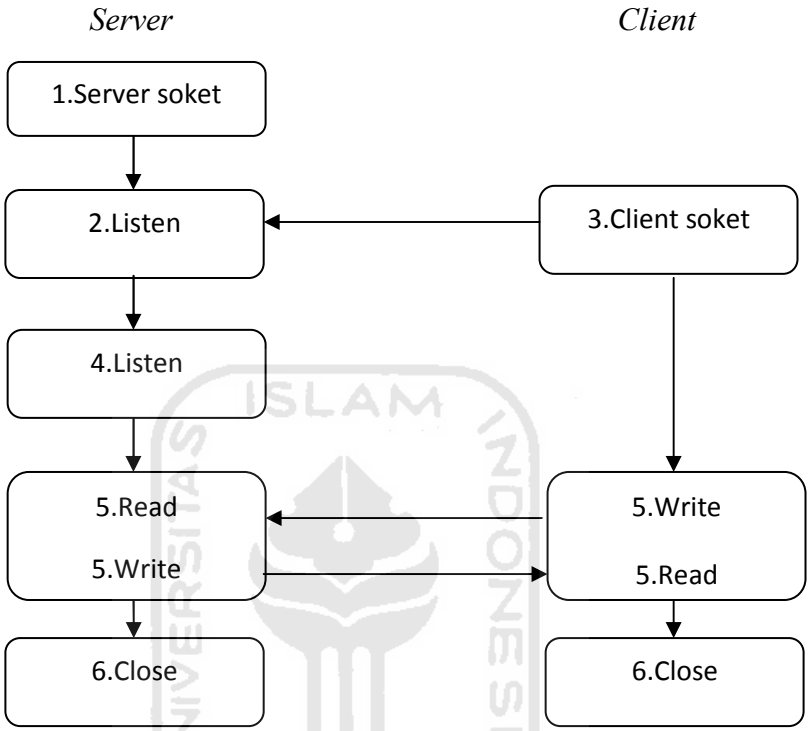
berjalan pada jaringan. *Socket* merupakan fasilitas *IPC* (*inter proses communication*) untuk aplikasi jaringan. Agar suatu *socket* dapat berkomunikasi dengan *socket* lainnya maka *socket* butuh diberi suatu alamat unik sebagai identifikasi. Alamat *socket* terdiri atas alamat IP dan nomor portnya. Contoh alamat *socket* adalah 192.168.29.30:3000, dimana nomor 3000 adalah nomor portnya. Alamat IP dapat menggunakan alamat jaringan lokal (LAN) maupun alamat Internet. Jadi *socket* dapat digunakan untuk *IPC* pada LAN maupun Internet.

Secara umum ada dua macam komunikasi dengan menggunakan *socket*, yaitu komunikasi stream dan komunikasi datagram. Komunikasi stream sering juga disebut dengan komunikasi yang berorientasi koneksi (*Connection oriented communication*), sedangkan komunikasi datagram disebut juga dengan komunikasi tak berkoneksi (*connectionless communication*). Protokol standar untuk komunikasi stream dikenal dengan istilah TCP (*Transmission Control Protocol*), sedangkan standar protokol komunikasi datagram dikenal dengan UDP (*User Datagram Protocol*).

Pada UDP, setiap kali suatu paket data dikirimkan, informasi soket pengirim dan alamat soket tujuan dikirmkan. Hal demikian tidak dibutuhkan oleh TCP, karena TCP akan membuat setup koneksi dengan soket tujuan terlebih dahulu. Setelah koneksi terbentuk, tidak dibutuhkan mengirimkan informasi soket pengirim tiap kali data dikirimkan. Ini karena proses tujuan akan mengidentifikasi setia data yang tiba pada soket tujuan sebagai data dari proses pengirim. Koneksi yang terbentuk pada TCP bersifar dua arah [KS11].

2.4 Model Aplikasi Client-Server

Model aplikasi yang menggunakan komunikasi *socket* dengan protokol TCP dapat dilihat pada gambar 2.4.



Gambar 2.3 Model Aplikasi *Client-Server* pada protokol TCP [JN07].

Objek *socket* pada sisi client dan server berbeda sedikit. Pada sisi aplikasi server, suatu *socket* server dibentuk (1) dan melakukan operasi *listen* (2). Operasi ini pada intinya menunggu permintaan koneksi dari sisi *client*. Sedangkan pada sisi *client*, dibentuk *socket* biasa.

Pada saat *socket client* (3), informasi alamat *socket server* dilewatkan sebagai argumen dan *socket* client mencoba meminta koneksi ke *socket server*. Pada saat permintaan koneksi *client* sampai pada *server*, maka *server* akan membuat suatu *socket* biasa. *Socket* ini yang nantinya akan berkomunikasi dengan *socket* pada sisi client. Setelah itu *socket* server dapat kembali melakukan *listen* (4) untuk menunggu permintaan koneksi dari client lainnya. Langkah 4 ini

umumnya hanya dilakukan jika aplikasi server mengimplementasikan multithreading.

Setelah terciptanya koneksi antara *client* dan *server*, maka keduanya dapat saling bertukar pesan (5). Salah satu atau keduanya kemudian dapat mengakhiri komunikasi dengan menutup *socket* (6).

2.5 Pemograman Socket di Java

Salah satu bahasa pemograman yang sudah mendukung pemograman *socket* adalah Java. Hal tersebut disediakan pada paket java.net. Paket java.net berisi kelas-kelas untuk pemograman jaringan menggunakan protokol UDP ataupun TCP. Java mendukung UDP dengan menyediakan dua kelas, yaitu kelas java.net.DatagramPacket dan java.net.DatagramSocket. Sedangkan dukungan untuk TCP ada pada kelas java.net.Socket dan java.net.ServerSocket.

Link komunikasi yang dibuat melalui *socket* TCP bersifar *connection-oriented*, menjamin data yang dikirimkan sampai ke penerima data. Koneksi antara *client* dan *server* tetap terbuka selama terjadi dialog dan dihancurkan hanya saat salah satu pihak meminta mengakhiri komunikasi. Sedangkan link komunikasi yang dibuat melalui *socket* UDP bersifat *connectionless*, tidak menjamin data yang dikirimkan sampai ke penerima, tidak ada koneksi antara client dan server yang dipelihara selama dialog berlangsung.

Biasanya terdapat 4 operasi dasar pada *socket* yang terpasang di *client* yaitu mengirim data dengan *remote machine*, mengirim data, menerima data dan memutuskan koneksi. Sedangkan pada *socket server* biasanya terdapat 3 operasi dasar yaitu mengikat port, menerima koneksi dan mengelola data yang datang maupun yang akan dikirimkan.

Data yang dikirim melalui Internet atau jaringan di dalam sebuah paket disebut datagram. Setiap datagram berisi *header* dan *playload*. *Header* berisi alamat dan port untuk setiap paket, *address* dan port untuk tiap paket yang masuk, sedangkan *playload* berisi data. Sebuah paket datagram bisa saja hilang atau *corrupted* ketika dikirimkan dan membutuhkan pengiriman lagi. *Socket*

melindungi programmer dari detail *low-level* dari sebuah jaringan, seperti deteksi kesalahan, ukuran paket, pengiriman kembali sebuah paket dan masih banyak lagi [JN07].

2.6 Port

Dalam protokol TCP/IP, sebuah port mengijinkan sebuah komputer membentuk beberapa sesi koneksi dengan komputer lainnya. Port dapat digunakan sebagai identifikasi aplikasi dan layanan yang menggunakan koneksi TCP/IP. Dilihat dari penomorannya, port dibagi menjadi 3 jenis,yaitu sebagai berikut [PB00] :

- *Well-known port*. Nomor port ini pada awalnya berkisar antara 0 hinga 255 tapi kemudian diperlebar antara 0 hingga 1023. Nomor port yang termasuk ke dalam *well-known port*, selalu merepresentasikan layanan jaringan yang sama, dan ditetapkan oleh *Internet Assigned Number Authority* (IANA). Beberapa di antara port-port number yang berada di dalam *range well-known port* masih belum ditetapkan dan direservasikan untuk digunakan oleh layanan yang bakal ada di masa depan. *Well-known port* didefinisikan dalam RFC 1060.

- *Registered port*. Port-port ini digunakan oleh *vendor-vendor* komputer atau jaringan yang berbeda untuk mendukung aplikasi dan sistem operasi yang mereka buat. Nomor *registered port* berkisar dari 1024 hingga 49151 dan beberapa port diantaranya *dynamically assigned port*.

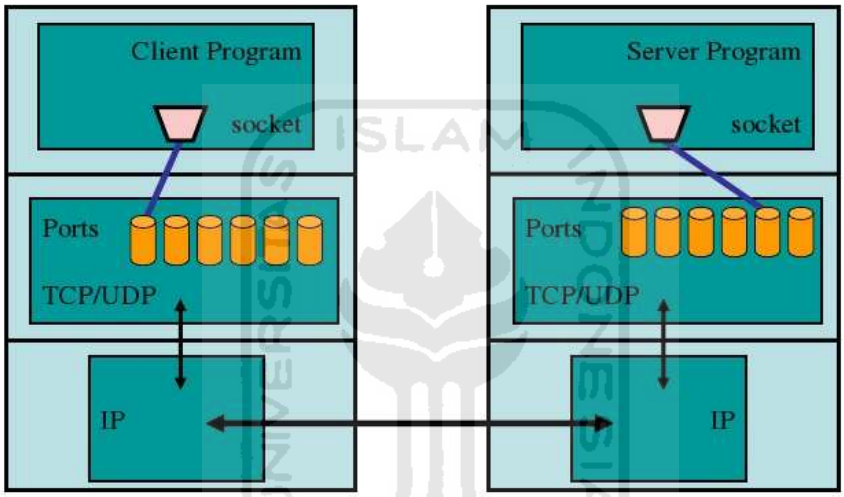
- *Dynamically assigned port*. Port-port ini ditetapkan oleh sistem operasi atau aplikasi yang digunakan untuk melayani *request* dari pengguna sesuai dengan kebutuhan. *Dynamically assigned port* berkisar dari 1024 hingga 65536 dan dapat digunakan atau dilepaskan sesuai kebutuhan.

BAB III

METODOLOGI

3.1 Gambaran Arsitektur Sistem

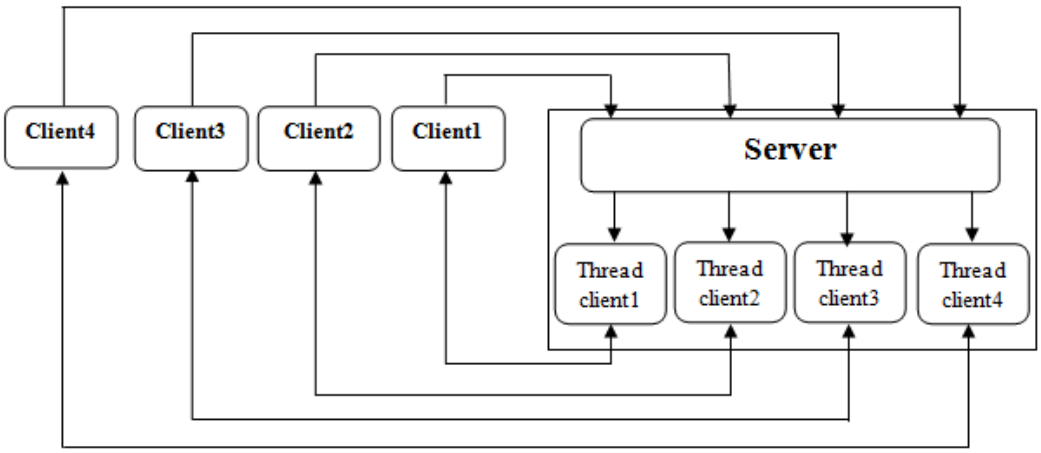
Sistem yang akan dibuat merupakan implementasi Java *socket* yang berbasiskan *client-server*, yaitu terdapat sebuah server yang menunggu datangnya koneksi dari client melalui socket. Gambar 3.1 merupakan bagaimana client dan server saling berhubungan melalui socket.



Gambar 3.1 *socket* pada *client-server* [NP10].

setiap komputer berkomunikasi melalui *socket*. Program pada *client* dan *server* sama-sama membuka *socket* untuk bisa saling berkomunikasi. Program yang berada di *client* harus mengetahui alamat *socket* servernya. Alamat *socket* adalah gabungan dari ip address *server* dan nomor port aplikasi tersebut dijalankan, contoh alamat *socket* adalah 192.168.0.1:1234.

Secara umum ada dua macam komunikasi dengan menggunakan *socket*, yaitu komunikasi stream atau lebih sering dikenal dengan TCP dan komunikasi datagram, yang lebih sering dikenal dengan UDP. Pada *game go-fish* ini menggunakan komunikasi stream (TCP), karena karakteristik dari komunikasi stream (TCP) ini sangat sesuai dengan kebutuhan dari *game go-fish* yang akan dirancang, seperti *connection-oriented* dan menjamin data sampai ke tujuannya.



Gambar 3.2 komunikasi *client-server*

Pada gambar 3.2 dijelaskan bagaimana hubungan antara *client* dan *server*. Pada saat pertama kali *client* memasuki *server*, *client* menghubungi *server* dengan parameter IP address *server* dan port yang membuka layanan *game go-fish*, ketika client menghubungi *server* maka *server* akan membuat *thread* yang nantinya bertugas melakukan pertukaran data dengan client. Thread merupakan cara di Java untuk menjalankan banyak bagian secara konkuren, thread merupakan *lightweight process* yang tidak menghabiskan banyak *overhead* seperti proses normal [HB10].

Pada lingkungan *multithreading* seperti yang diterapkan dalam sistem, *thread* merupakan unit penjadwalan terkecil. Hal ini berarti satu program dapat mempunyai lebih dari satu *thread* yang berjalan secara konkuren. Besarnya data yang dikirimkan baik oleh *client* maupun *server* diatur oleh bahasa pemograman karena *socket* pada bahasa pemograman Java melindungi programmer dari *low-level* yang detail dari sebuah jaringan, seperti deteksi kesalahan, ukuran paket dan pengiriman kembali sebuah paket.

3.2 Perangkat Keras dan Perangkat Lunak yang Dibutuhkan

3.2.1 Perangkat Keras yang Dibutuhkan

Perangkat keras penunjang yang diperlukan pada proses perancangan *game go-fish* menggunakan Java adalah sebagai berikut.

1. Komputer *Server*

   Kebutuhan komputer yang diperuntukkan sebagai *server*, dengan spesifikasi minimal sebagai berikut :

   a. Processor dengan kecepatan 1,6 GHz
   b. Memori minimum 1 GB direkomendasikan 2 GB
   c. Hardisk dengan kapasitas minimum 5GB
   d. Mouse
   e. Keyboard
   f. Satu buah *Network Interface Card* (NIC)

2. Komputer *Client*

   Kebutuhan komputer yang diperuntukkan sebagai *client*, dengan spesifikasi minimal sebagai berikut :

   a. Processor dengan kecepatan 1,6 GHz
   b. Memori minimum 512 MB direkomendasikan 1 GB
   c. Hardisk dengan kapasitas minimum 5 GB
   d. Mouse
   e. Keyboard
   f. Satu buah *Network Interface Card* (NIC)

3.2.2 Kebutuhan Perangkat Lunak yang Dibutuhkan

Selain perangkat keras, perangkat lunak juga diperlukan dalam pembangunan aplikasi dan dalam menjalankan aplikasi. Pada implementasi Java *socket* pada game jaringan go-fish, perangkat lunak yang diperlukan adalah :

a. Sistem operasi, sistem operasi yang dibutuhkan untuk pembangunan aplikasi *game go-fish* adalah Windows XP atau Windows 7 (*seven*).

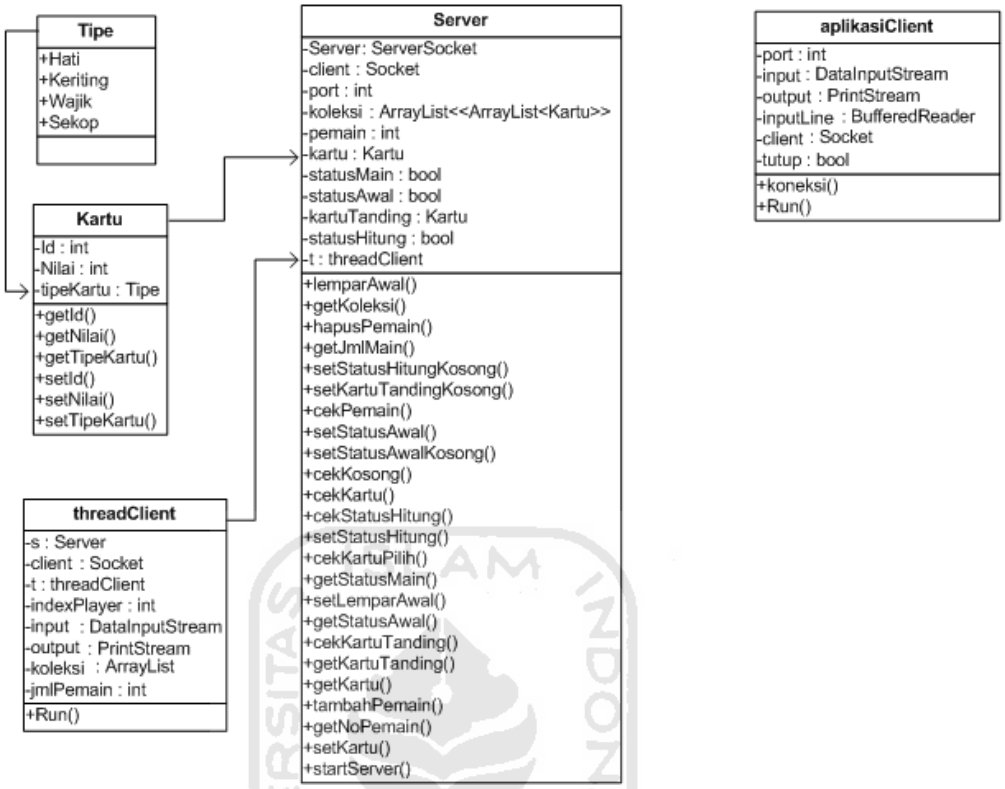
b. Java jdk 1.6, merupakan bahasa pemograman yang dipakai untuk membuat *game go-fish*.
c. Netbeans, merupakan salah satu *aplication building tools* yang berupa IDE (*Integrated Development Environment*) yang digunakan membangun keseluruhan aplikasi.

3.3 Metode Perancangan Perangkat Lunak

Metode perancangan dalam membangun aplikasi *game* berbasis *client-server* ini menggunakan metode perancangan terstruktur dengan menggunakan diagram *Unified Modeling Language* (UML). UML merupakan sebuah bahasa standar untuk visualisasi, merancang dan mendokumentasikan sistem perangkat lunak. Dalam perancangan sistem ini, sistem yang dibangun digambarkan dalam bentuk diagram UML yaitu menggunakan *class* diagram, *activity* diagram dan *sequence* diagram.

3.3.1 Class Diagram

Berikut class-class dan relasinya yang berada pada sistem *game* berbasis client-server :



Gambar 3.3 *Class* pada *game go-fish* dan relasi antar *class*nya

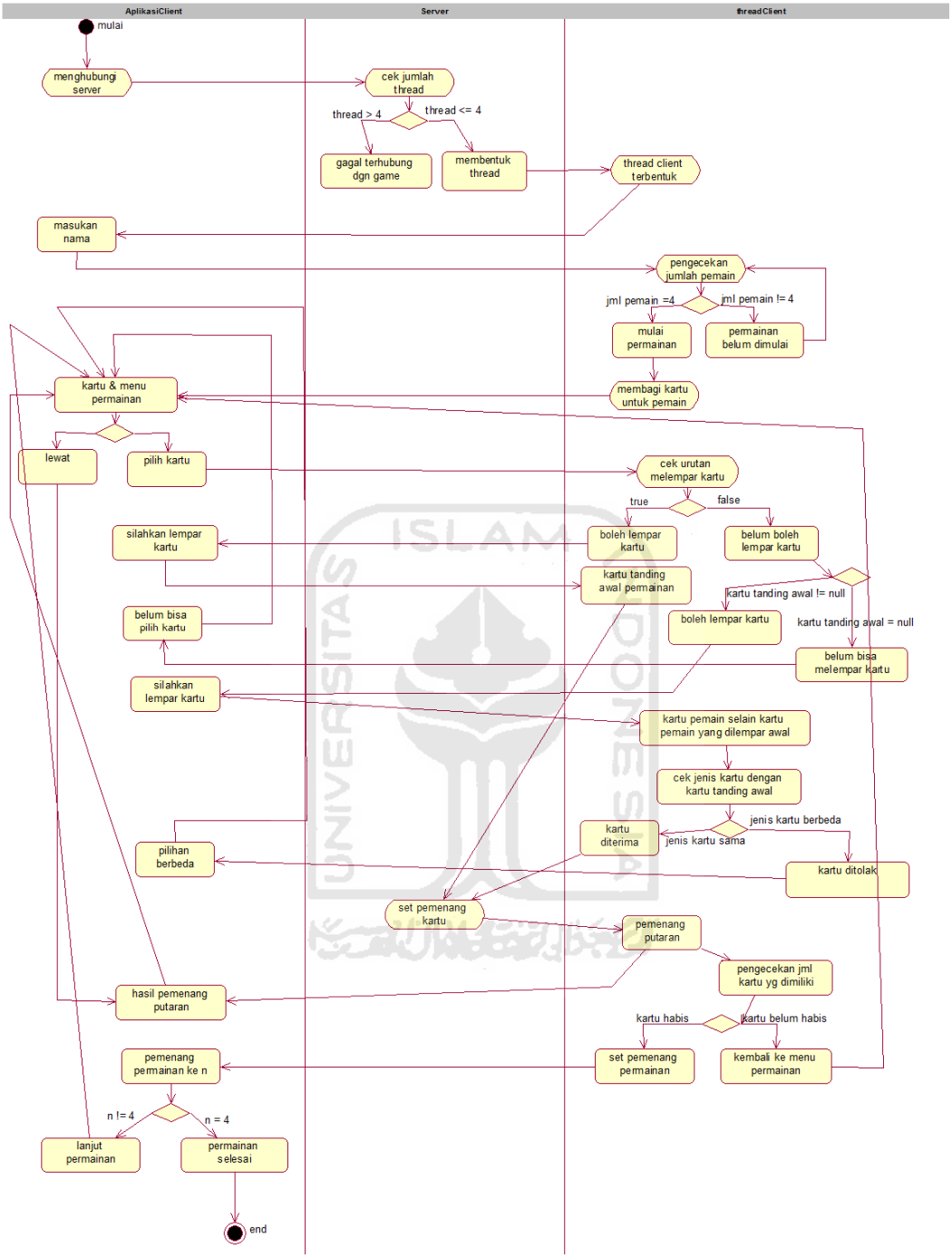
Pada gambar 3.3 menunjukan *class-class* dan relasi antar *class* yang berada pada sistem *game go-fish*. Terdapat lima *class* untuk membangun sistem *game go-fish*, empat *class* untuk membangun server, yaitu *class* Tipe, *class* Kartu, *class* threadClient dan *class* Server. Satu *class* untuk membangun aplikasi untuk *client*/pemain yaitu *class* aplikasiClient.

Relasi antar *class* seperti yang terlihat pada gambar 3.3 berupa relasi asosiasi. Asosiasi adalah hubungan statis antar class. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Penunjuk panah menunjukan arah *query* antar *class* [DS03]. Pada gambar 3.3 *class* Tipe berasosiasi dengan *class* Kartu kemudian *class* Kartu berasosiasi dengan *class* Server dan *class* threadClient berasosiasi dengan *class* Server.

3.3.2 Activity Diagram

Untuk menggambarkan aktivitas yang terjadi di dalam sistem digunakan *activity diagram.* Didalam *activity diagram* akan digambarkan berbagai aliran aktivitas dalam sistem yang akan dibangun, bagaimana aliran aktifitas dalam sistem, bagaimana aliran aktifitas berawal, keputusan yang mungkin terjadi dan bagaimana aktifitas itu berakhir.

Activity diagram umumnya tidak menggambarkan secara detail urutan proses, namun hanya memberikan gambaran global bagaimana urutan proses yang terjadi. Gambaran *activity diagram* dalam keseluruhan sistem ini dapat dilihat pada gambar 3.4



Gambar 3.4 *Activity* diagram keseluruhan sistem.

Gambar 3.4 merupakan gambar *activity diagram* untuk keseluruhan sistem. Pada sistem *game go-fish* terdapat 3 *swimlane*, yaitu aplikasiClient, server dan threadClient. Pada awal permainan aplikasiClient akan menghubungi server, server kemudian melakukan pengecekan terhadap jumlah thread yang sudah ada, jika jumlah thread lebih dari 4 maka pemain/aplikasiClient tidak bisa ikut dalam permainan sedangkan jika thread kurang dari 4 maka server menerima koneksi yang dilakukan aplikasiClient dengan membuat thread, thread nantinya akan saling berkomunikasi dengan aplikasiClient ketika permainan dimulai.

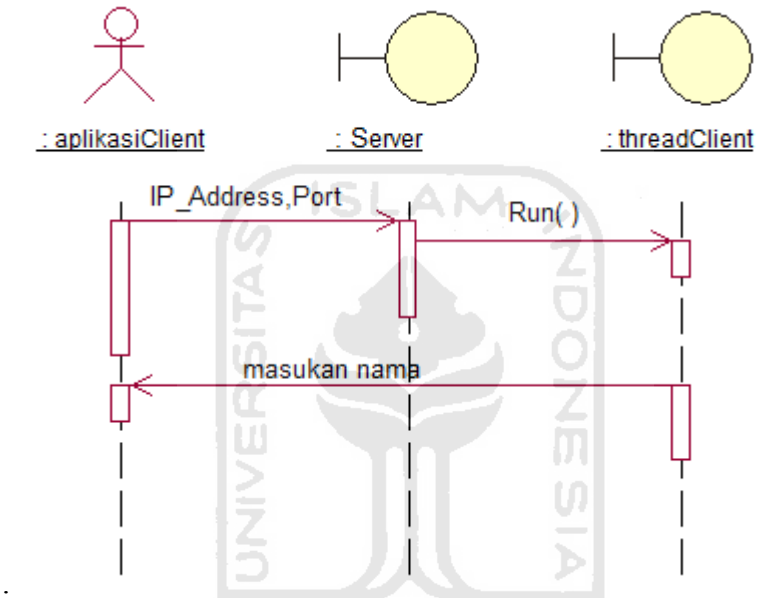
Setelah thread terbentuk kemudian akan mengirimkan pesan kepada aplikasiClient untuk mengisikan nama untuk keperluan permainan. Pada thread terdapat pengecekan jumlah pemain jika jumlah pemain belum memenuhi persyaratan permainan thread akan menunggu sampai persyaratan permainan terpenuhi (permainan dimulai ketika jumlah pemain sudah mencapai 4 pemain). Ketika permainan dimulai, thread akan mengirimkan kartu ke setiap pemain dan juga menu permainan.

Pada *game go-fish* ini terdapat menu permainan dengan 2 pilihan. Pilihan pertama yaitu pilih kartu, digunakan ketika pemain akan melempar salah kartu yang dimilikinya. Pilihan kedua yaitu lewat, digunakan ketika pemain tidak ingin atau tidak bisa melempar salah satu kartu yang dimilikinya (jenis kartu harus sama dengan jenis kartu dari pemain yang melempar pertama kali). Setelah semua pemain yang memilih menu pilih kartu melemparkan kartunya, server akan membandingkan kartu dari setiap pemain. Pemain yang melempar kartu dengan nilai tertinggi keluar sebagai pemenang putaran.

Setelah didapatkan pemenang pada suatu putaran, si pemenang putaran berhak melempar kartu pertama pada putaran berikutnya. Alur permainan putaran yang baru sama seperti alur putaran yang sebelumnya sampai kartu ditangan tiap-tiap pemain habis.

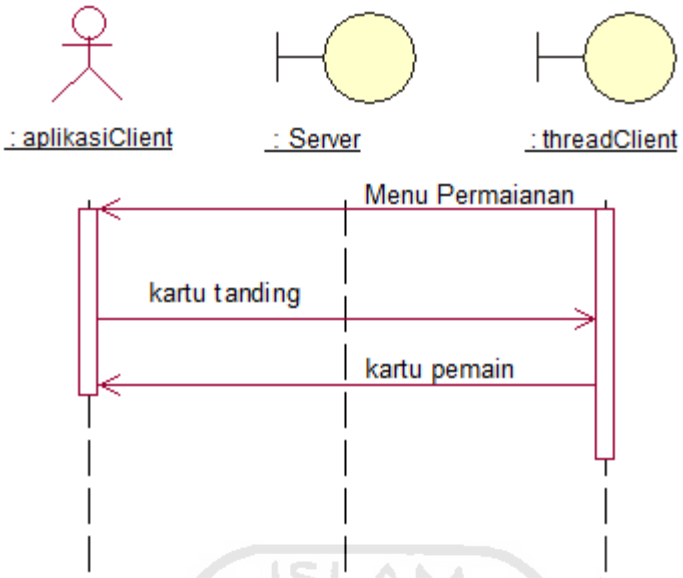
3.3.3 Sequence Diagram

Fungsi utama sequence diagram yaitu untuk menggambarkan skenario atau urutan langkah-langkah yang dilakukan sebagai respon yang memicu aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan.



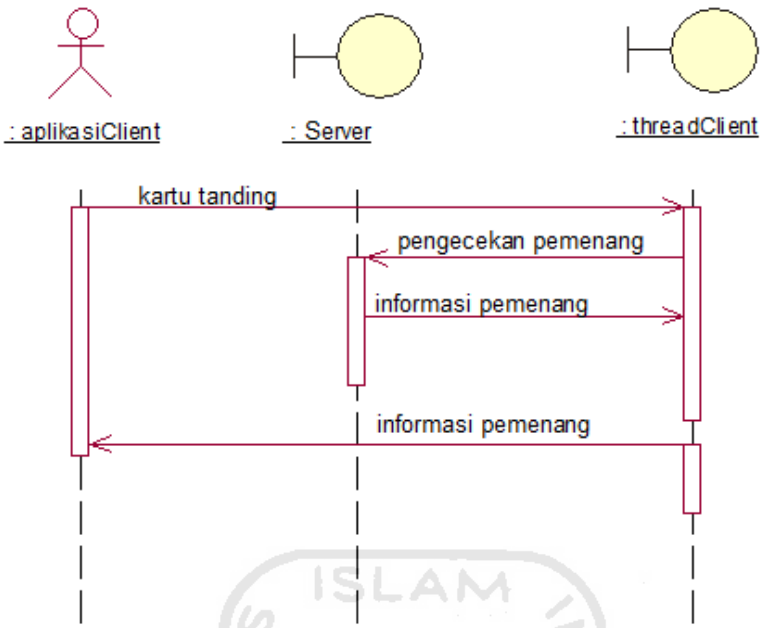
Gambar 3.5 Sequence diagram koneksi client ke server

Pada gambar 3.5 menjelaskan bagaimana client/pemain melakukan koneksi ke server dengan paramater IP_address server permainan dan nomor port server yang telah diset untuk membuka layanan untuk permainan game kartu. Setelah client berhasil memasuki server maka server akan membuat thread untuk client / pemain dengan menjalankan fungsi Run ( ). Setelah thread terbentuk thread mengirimkan sebuah permintaan untuk mengisikan nama pemain keperluan permainan.



Gambar 3.6 Pertukaran data client dengan thread.

Pada gambar 3.6 menjelaskan pertukaran data antara client dengan thread setelah pembentukan koneksi di mana thread mengirimkan menu permainan ke pemain / client. Menu permaianan memiliki 2 pilihan yaitu lempar kartu dan lewat. Pada gambar 3.6 diambil contoh jika pemain memilih menu lempar kartu untuk melempar satu kartu untuk ditandingkan. Thread kemudian akan memberikan informasi kartu tanding tiap-tiap pemain ke semua pemain.



Gambar 3.7 Pertukaran data client,thread dan server**.**

Pada gambar 3.7 pertukaran data antara client, thread dan server untuk pengecekan pemenang. Setelah semua pemain melemparkan kartu (kartu ini dikirim dari pemain ke threadnya), thread akan mengirim kartu tanding tiap pemain ke server. Server melakukan pengecekan kartu tanding. Pemain yang memiliki kartu tanding dengan angka tertinggi akan dinyatakan sebagai pemenang putaran. Informasi pemenang ini dikirimkan oleh server ke thread, kemudian thread mengirimkan informasi pemenang ke semua pemain.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil

Pada tahap ini akan dijelaskan mengenai pengujian *game go-fish* yang telah dibuat yang merupakan hasil implementasi dari Java *socket* . Pengujian ini bertujuan untuk mengetahui proses kerja aplikasi secara menyeluruh, serta untuk mengetahui adanya kelemahan dan kekurangan di dalam aplikasi. Pengujian dilakukan dengan memainkan *game* tersebut sehingga terjadi proses komunikasi antara *client* dan *server*.

4.2 Implementasi

Implementasi aplikasi *game go-fish* berbasis *client-server* ini dibangun dengan bahasa pemograman Java dengan tampilan aplikasi berbasis command prompt. Implementasi aplikasi ini dibagi menjadi dua bagian, yaitu batasan implementasi dan pengujian hasil implementasi.

4.3 Batasan Implementasi

Batasan implementasi dalam penelitian ini adalah :

1. GUI dari *game go-fish* hanya berupa command prompt.
2. Permainan game hanya sekali putaran.
3. Thread tidak tersinkronisasi sehingga tidak adanya urutan pelempar kedua, ketiga dan keempat.

4.4 Pengujian hasil implementasi

Bagian ini akan membahas dan menganalisis dari game client-server yang telah dirancang dengan cara melakukan proses komunikasi antara *client* dan *server*. Jika proses komunikasi antara *client* dan *server* berjalan lancar maka aplikasi yang telah dibuat sudah dikatakan berhasil untuk bisa bertukar data.

4.4.1 Tampilan server ketika sedang dijalankan

Ketika *server* pertama kali dijalankan akan mendapatkan tampilan seperti gambar 4.1, yang menunjukan bahwa *server* telah berjalan dan menunggu *client* untuk bergabung dengan *server*.



Gambar 4.1 Tampilan server ketika baru dijalankan

4.4.2 Tampilan server ketika client bergabung dengan server

Setelah *server* menyala maka posisi *server* sedang "mendengarkan" permintaan koneksi dari *client* yang akan bergabung. Saat *client* masuk ke *server* maka *server* akan menampilkan informasi *client* yang masuk ke server berupa IP address.



Gambar 4.2 Tampilan server saat menerima koneksi dari client

4.4.3 Tampilan client ketika melakukan koneksi ke server

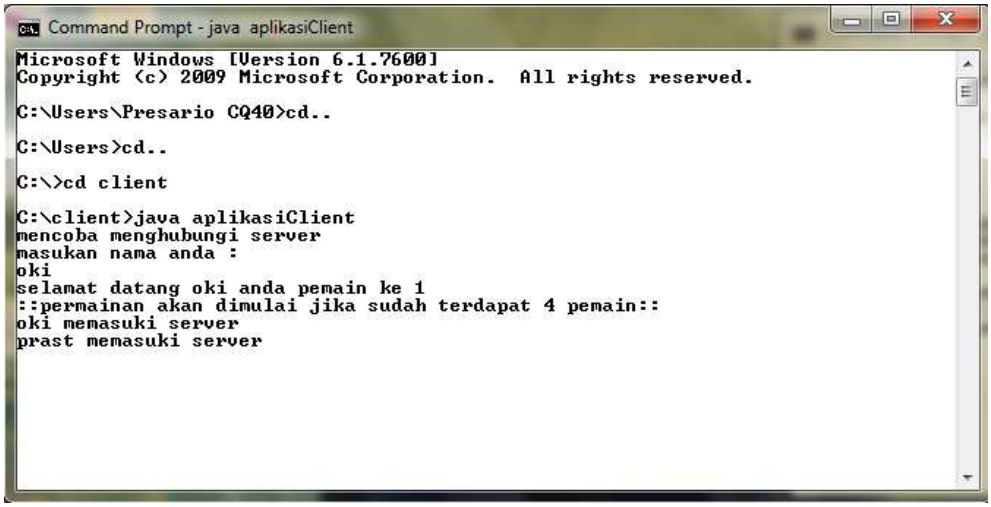
Ketika aplikasi *client* pertama kali dijalankan, *client* langsung menghubungi *server*. Setelah berhasil terhubung ke *server*, *server* akan mengirimkan permintaan untuk agar *client* memasukan nama yang berfungsi untuk keperluan identifikasi pemain. Setelah nama dimasukan, maka akan muncul informasi dari server tentang nama pemain dan nomor pemain dan juga pemberitahuan untuk menunggu sampai pemain yang bergabung dengan server berjumlah empat pemain.



Gambar 4.3 Tampilan client ketika pertama kali memasuki server

4.4.4 Tampilan client ketika terdapat pemain lain yang bergabung ke permainan

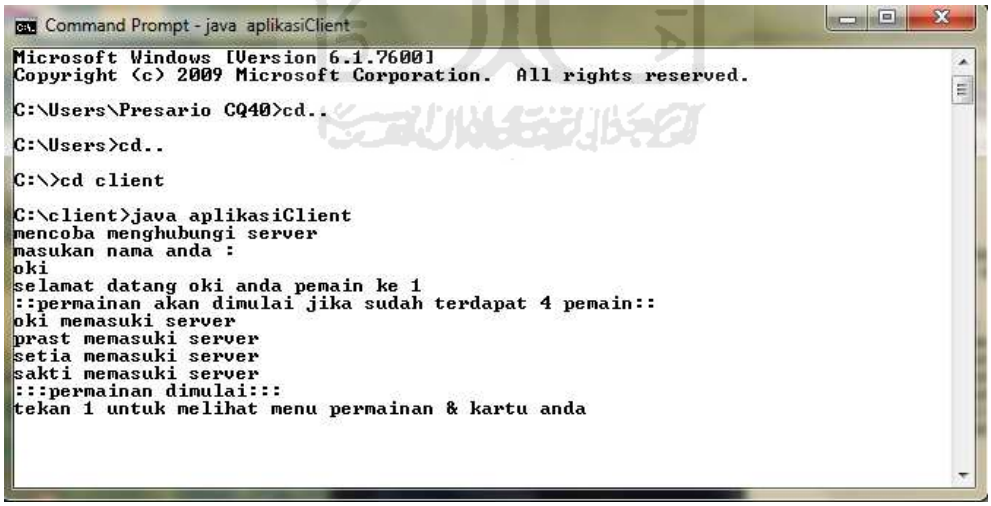
Pemain yang sudah memasuki server, yang sedang menunggu pemain lain bergabung ke server, akan mendapatkan informasi dari server jika ada pemain yang baru memasuki server, seperti terlihat pada gambar 4.4.



Gambar 4.4 Tampilan client ketika pemain yang lain bergabung ke server

4.4.5 Tampilan ketika empat pemain sudah memasuki server

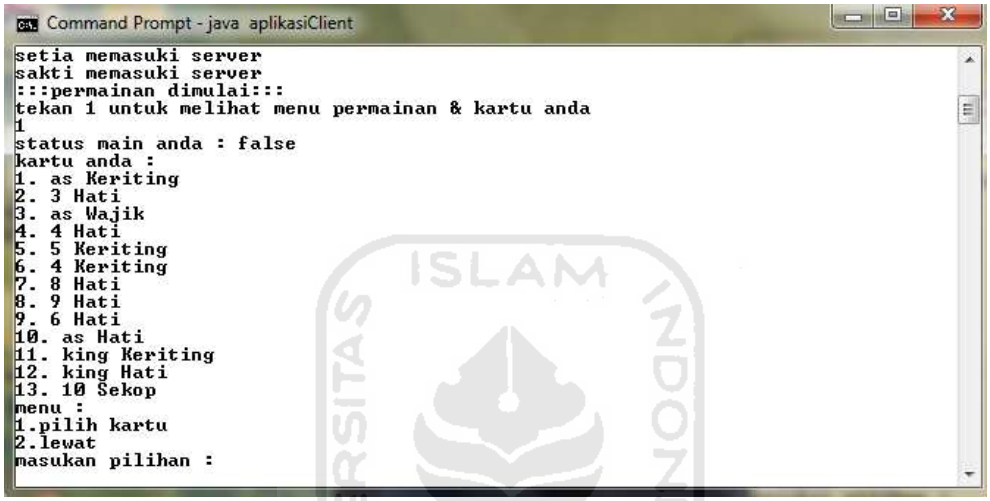
Ketika pemain ke empat sudah memasuki server, maka server akan segera memulai permainan, server akan mengirim kartu yang sudah di acak ke setiap pemain. Pemain diminta oleh server untuk menekan angka 1 untuk melihat kartu yang diberikan oleh server dan juga menu permainan.



Gambar 4.5 Tampilan ketika ke empat pemain memasuki server

4.4.6 Tampilan client saat permainan dimulai

Ketika jumlah pemain yang masuk ke *server* sudah berjumlah empat pemain, maka *server* membagikan kartu dan menampilkan menu permaianan ke seluruh pemain. Permainan dimulai dengan urutan yang melempar kartu pertama adalah pemain yang terhubung ke *server* pertama kali.



Gambar 4.6 Tampilan client memulai permainan

4.4.7 Tampilan client saat memilih menu pilih kartu

Setelah mendapat kartu dan menu permainan, pemain dapat melempar kartu dengan cara memilih menu "1. Pilih kartu". Jika *server* memperbolehkan pemain itu melempar kartu maka *server* akan memberikan informasi "silahkan lempar kartu". Setelah itu pemain melempar kartu dengan cara memasukan index dari kartu yang dibagikan oleh server, seperti pada gambar 4.6, pemain dengan nama oki memasukan index 1 yang berarti melempar kartu as Keriting.



Gambar 4.7 Tampilan client saat memilih menu pilih kartu

4.4.8 Tampilan client saat belum boleh melempar kartu

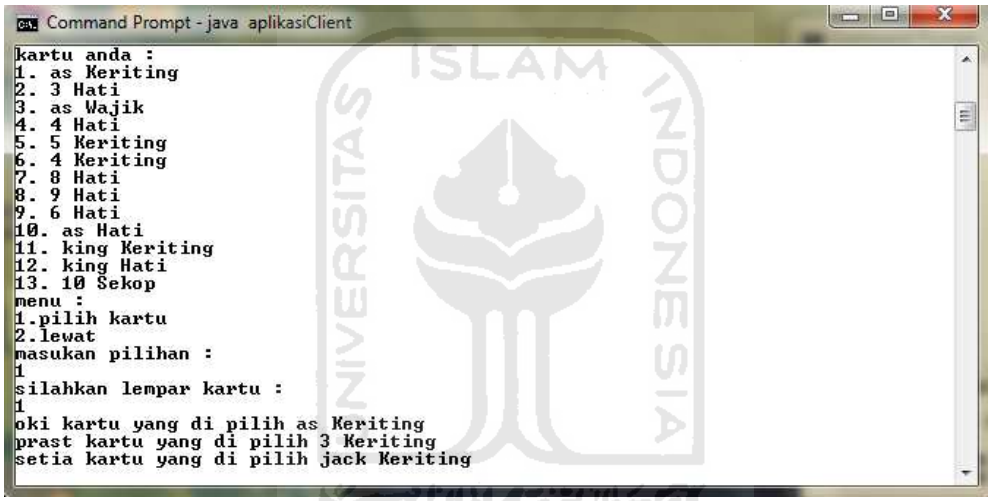
Ketika pemain memilih menu "pilih kartu" tetapi belum diperbolehkan karena bukan giliran pemain tersebut untuk melempar kartu, maka *server* akan mengirimkan informasi bahwa pemain tersebut belum diperbolehkan melempar kartu.



Gambar 4.8 Tampilan client belum diperbolehkan melempar kartu

4.4.9 Tampilan client saat menerima informasi kartu yang dilempar oleh pemain lain.

Setiap pemain yang melempar kartu, kartu yang dilempar oleh setiap pemain, server akan mengirimkan informasi berupa nama pemain dan kartu yang dilemparnya ke semua pemain, seperti terlihat pada gambar 4.9, informasi ini sangat berguna bagi pemain lain untuk memutuskan ikut melempar kartu atau tidak. Aturan *game go-fish* menyatakan bahwa pemain harus melempar kartu dengan jenis yang sama dengan jenis kartu yang dilempar oleh pemain yang pertama melempar kartu.



Gambar 4.9 Informasi dari server tentang kartu yang dilempar setiap pemain

4.4.10 Tampilan client ketika melempar jenis kartu yang berbeda

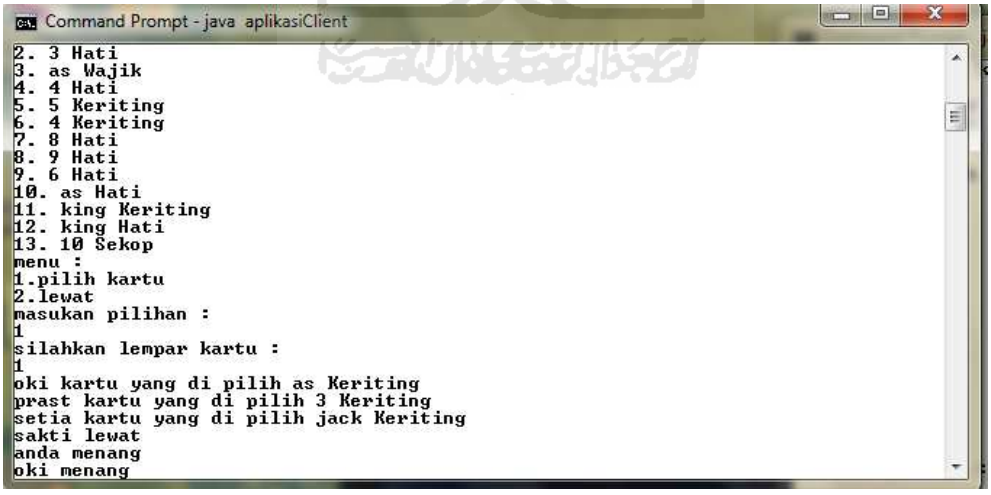
Aturan dalam permainan go-fish adalah mempertandingkan jenis kartu yang sama, jika ada pemain yang melempar jenis kartu yang berbeda dengan jenis kartu yang ditandingkan, maka server akan memberitahukan pada pemain tersebut bahwa kartu yang dipilihnya berbeda jenis dengan jenis kartu yang dilempar oleh pemain yang pertama melempar kartu.



Gambar 4.10 Tampilan client ketika jenis kartu yang dilempar berbeda

4.4.11 Tampilan client mendapatkan informasi pemenang putaran

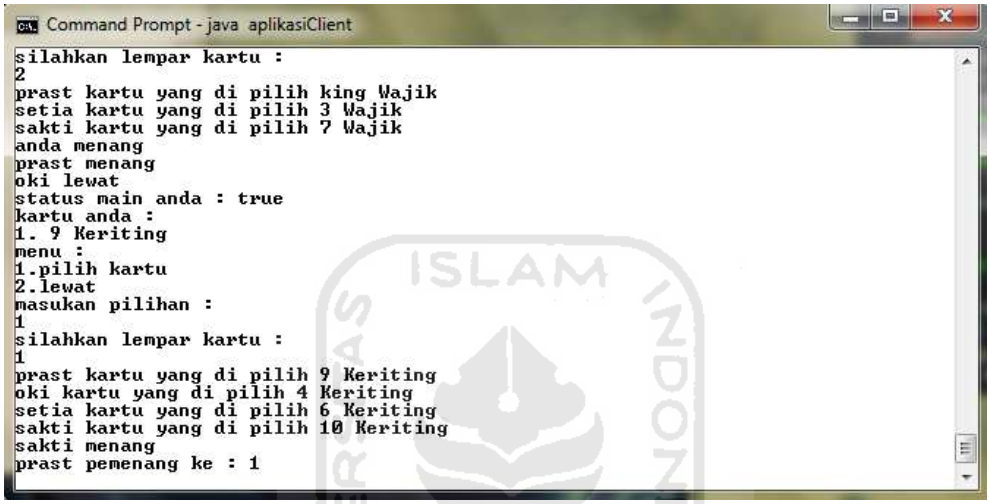
Setelah semua pemain melempar kartu, tiap-tiap kartu yang dilemparkan oleh setiap pemain akan dicek oleh server. Pemain dengan kartu yang memiliki nilai tertinggi di antara kartu pemain yang lain dinyatakan sebagai pemenang pada putaran itu. Pemenang tersebut berhak melempar kartu pertama kali diputaran berikutnya. Informasi pemenang akan dikirimkan oleh *server* ke semua pemain.



Gambar 4.11 Tampilan client ketika pengumuman pemenang putaran

4.4.12 Tampilan client saat kartunya sudah habis

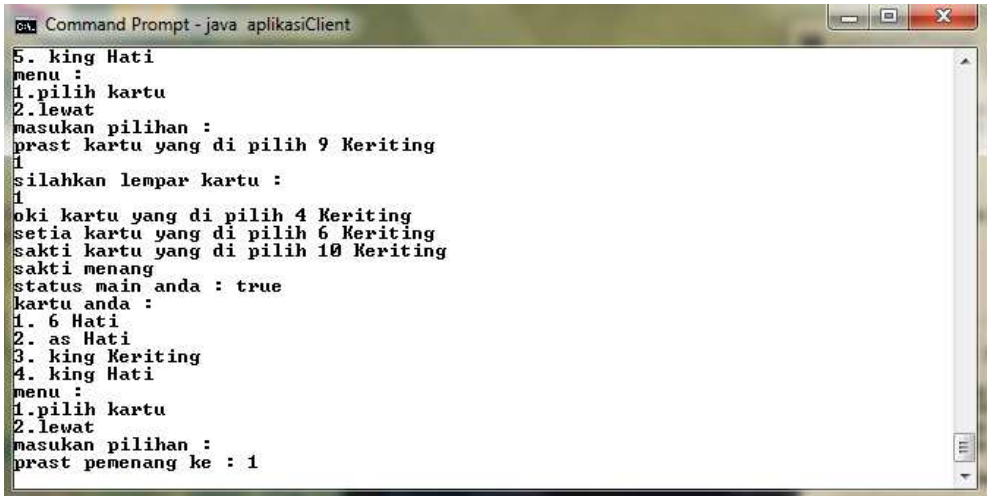
pemain akan mendapat informasi ketika kartunya sudah habis dan keluar sebagai pemenang seperti terlihat pada gambar 4.12, pemain dengan nama prast telah menghabiskan semua kartu yang dibagikan diawal oleh server dan keluar sebagai pemenang pertama.



Gambar 4.12 Tampilan ketika kartu pemain sudah habis

4.4.13 Tampilan client saat salah satu pemain menghabis kartu

Ketika salah satu pemain dinyatakan sebagai pemenang permainan, semua pemain yang lain akan mendapatkan informasi dari *server* tentang urutan pemenang permainan. Meskipun pemenang permainan sudah ada, pemain yang lain yang belum menghabiskan kartunya akan terus bermain sampai kartu tiap pemain habis.



Gambar 4.13 Tampilan client mendapatkan informasi pemenang permainan

4.4.14 Tampilan client saat pemain terakhir menghabiskan kartunya

Permainan selesai saat semua pemain sudah menghabiskan semua kartu yang dibagikan oleh *server*. Ketika semua pemain kartunya sudah habis maka *server* memberikan informasi urutan bahwa permainan telah selesai.



Gambar 4.14 Tampilan client saat permainan selesai.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah melalui berbagai tahapan-tahapan pembuatan dan pengujian *game* Jaringan *go-fish* menggunakan Java socket, maka dapat diambil kesimpulan sebagai berikut :

1. Agar dua program (*client* dan *server*) bisa saling berkomunikasi masing-masing harus membuat socket.
2. Sistem yang dirancang menggunakan konsep multithreading sehingga tidak terlalu boros dalam penggunaan *resource* memori.
3. Bahasa pemograman Java menyediakan fasilitas yang lengkap dalam pemograman jaringan yang tersedia di paket Java.net.*.

5.2 Saran

Setelah melihat hasil yang dicapai dalam tugas akhir ini, maka ada beberapa saran untuk pengembangan tugas akhir ini, sebagai berikut :

1. Menggunakan sinkronisasi antar thread sehingga hak akses antar thread tidak menjadi rebutan.
2. Aplikasi *client* menggunakan GUI pada tampilan game agar tampilan ke *user* terlihat menarik.

DAFTAR PUSTAKA

[PB00] Purbo,O. 2000. Buku Pintar Internet TCP/IP, PT Elex Media Komputindo, Jakarta.

[KS11] Kusnadi, pemograman socket di Java, http://www.scribd.com/doc/48135346/tutorial1 diakses 11 januari 2012.

[JN07] Java Education Network Indonesia. 2007. Pemograman Socket, http://www.scribd.com/doc/7594449/modul9-pemrograman-socket diakses 9 januari 2012.

[SE04] Sutanta,E. 2004. Komunikasi data dan jaringan komputer, Graha Ilmu, Yogyakarta.

[DS03] Dharwiyanti, S. 2003. Pengantar Unified Modeling Language (UML). http://www.ikc.dinus.ac.id/umum/yanti/yanti-uml.doc diakses 15 januari 2012.

[HB10] Harianto, B. 2010. Esensi-esensi bahasa pemograman Java, Informatika Bandung, Bandung.

[WS11] Wibisono, S. 2011. Pemograman Socket TCP, http://setyawan-wibisono.com/458/socket-tcp diakses 10 januari 2012.

[RR10] Rosyadi, R. 2010. Referensi model TCP/IP, http://rizalrosyadi.com/2012/02/referensi-model-tcpip/ diakses 8 januari 2012.

[NP10] Nasirin, P. 2010. Komunikasi antar socket, http://puteranasirin.students-blog.undip.ac.id/files/2010/11/ProgJar-modul-7-Socket.pdf diakses 11 januari 2012.