APLIKASI *REMOTE AND MONITORING* BERBASIS *JAVA REMOTE METHOD INVOCATION (RMI)*

SKRIPSI

Diajukan Kepada Fakultas Teknik
Universitas Negeri Yogyakarta
Untuk Memenuhi Sebagian Persyaratan
Guna Memperoleh Gelar Sarjana Pendidikan Teknik

Oleh

AGUS SETIAWAN

NIM. 07520244054

PROGRAM STUDI PENDIDIKAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS NEGERI YOGYAKARTA
JUNI 2011

LEMBAR PERSETUJUAN

LAPORAN TUGAS AKHIR SKRIPSI

APLIKASI *REMOTE AND MONITORING BERBASIS JAVA REMOTE METHOD INVOCATION (RMI)*

Oleh :

Agus Setiawan
NIM. 07520244054

Telah Diperiksa dan Disetujui oleh Dosen Pembimbing.

Untuk Diuji

Yogyakarta, 20 Mei 2011

Mengetahui	Menyetujui
Ketua Program Studi	Dosen Pembimbing
Pendidikan Teknik Informatika,	Tugas Akhir Skripsi,
Umi Rochayati, M.T	**Dr. Eko Marpanaji**
NIP. 19630528 198710 2 001	NIP. 19670608 199303 1 001

LEMBAR PENGESAHAN

TUGAS AKHIR SKRIPSI

APLIKASI *REMOTE AND MONITORING BERBASIS JAVA REMOTE METHOD INVOCATION (RMI)*

Dipersiapkan dan Disusun Oleh:

AGUS SETIAWAN
NIM. 07520244054

Telah Dipertahankan di depan Panitia Penguji Tugas Akhir Skripsi
FAKULTAS TEKNIK UNIVERSITAS NEGERI YOGYAKARTA
Pada tanggal 6 Juni 2011
Dinyatakan Telah Memenuhi Syarat Guna Memperoleh Gelar
Sarjana Pendidikan Teknik Informatika

SUSUNAN PANITIA PENGUJI TUGAS AKHIR SKRIPSI

Jabatan	Nama Lengkap dan Gelar	Tanda Tangan	Tanggal
Ketua Penguji	**Dr. Eko Marpanaji**	.....	16-6-2011
Sekretaris	**Masduki Zakaria, M.T**	.....	16-6-2011
Penguji Utama	**Drs. Kadarisman Tejo Yuwono**	.....	16-6-2011

Yogyakarta, 2011
Dekan FT UNY

Wardan Suyanto, Ed.D
NIP. 19540810 197803 1 001

LEMBAR PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini :

Nama	:	Agus Setiawan
NIM	:	07520244054
Program Studi	:	Pendidikan Teknik Informatika
Judul Tugas Akhir Skripsi	:	Aplikasi *Remote and Monitoring* berbasis Java *Remote Method Invocation (RMI)*

Dengan ini peneliti menyatakan bahwa dalam Tugas Akhir Skripsi ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar Sarjana atau gelar lainnya disuatu Perguruan Tinggi. Sepanjang pengetahuan peneliti juga tidak terdapat karya atau pendapat yang pernah ditulis oleh orang lain, kecuali tertulis dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, 2011
Peneliti,

Agus Setiawan
NIM. 07520244054

APLIKASI *REMOTE AND MONITORING* BERBASIS *JAVA REMOTE METHOD INVOCATION (RMI)*

Oleh
Agus Setiawan
NIM. 07520244054

ABSTRAK

Penelitian ini bertujuan untuk mengembangkan perangkat lunak aplikasi *Remote and Monitoring* berbasis Java *Remote Method Invocation(RMI).* Perangkat lunak aplikasi ini dapat dimanfaatkan sebagai pengendalian dan pengawasan jarak jauh pada komputer yang terhubung jaringan yang tidak terbatas sistem operasi yang dipakai.

Penelitian ini menggunakan metode pengembangan perangkat lunak berorientasi objek dan dipadukan dengan model proses perangkat lunak *waterfall* yang melalui 4 tahapan. Tahapan pertama, analisis kebutuhan, obeservasi terhadap aplikasi *remote system* yang sudah ada, dan studi literatur. Tahapan kedua, desain sistem meliputi *use case diagram* yang menggambarkan kegiatan actor terhadap aplikasi dan *sequence diagram* menggambarkan urutan eksekusi aplikasi. Tahapan ketiga, pengkodean meliputi, implementasi pengkodean dari desain *sequence diagram* dan pengujian unit atau lebih sering disebut *white-box testing.* Tahapan keempat, Pengujian terintegrasi meliputi pengujian *black-box,* pengujian *alpha* yang nantinya akan digunakan untuk mengetahui unjuk kerja aplikasi, dan pengujian *beta* terhadap pengguna *Focus Group Discussion Digital Networks and Multimedia PUSKOM UNY.*

Berdasarkan hasil pengujian perangkat lunak Aplikasi *Remote and Monitoring* berbasis *Java Remote Method Invocation(RMI)* menunjukkan bahwa: 1) Pengembangan aplikasi *Remote* and *Monitoring* berbasis *Java Remote Method Invocation* melalui tahapan analisis kebutuhan, desain sistem, pengkodean dan pengujian. 2) Unjuk kerja Aplikasi *Remote and Monitoring* berbasis *Java Remote and Monitoring* berbasis *Java Remote Method Invocation* memiliki unjuk kerja yang baik semua system yang diujikan dapat berjalan dan bekerja sesuai dengan spesifikasi yang diinginkan. 3) Kelayakan aplikasi *Remote and Monitoring* berbasis Java *Remote Method Invocation* dari segi *usability* adalah sangat layak dengan persentase 82,14%.

Kata kunci: aplikasi, *remote and monitoring, remote system,* pengendalia jarak jauh, *java, remote method invocation*

MOTTO DAN PERSEMBAHAN

MOTTO :

- *Sesungguhnya sesudah kesulitan itu ada kemudahan, maka apabila kamu sudah selesai (dari suatu urusan) kerjakanlah dengan sungguh-sungguh (urusan) yang lain. Dan hanya kepada Tuhan-mulah hendaknya kamu berharap (QS. Al-Insyirah : 6 - 8).*
- *Hidup artinya berani menghadapi tantangan, tantangan terbesar dari hidup adalah tetap tumbuh dalam situasi apapun.*
- *Doa adalah senjata ampuh orang mukmin.*

PERSEMBAHAN

Dengan cinta dan kasih sayang yang tulus karya kecil ini kupersembahkan untuk:

1. *Bapak, dan Ibu tercinta yang selalu memberi do'a, kasih sayang dan semangat terus maju.*
2. *Keluarga dan sahabat-sahabatku.*
3. *Teman-teman S1 PTI '07 kelas G.*
4. *Bonita Destiana ☺*
5. *Almamaterku tercinta.*
6. *Nusa, bangsa, dan agama.*

KATA PENGANTAR

Alhamdulillah penyusun panjatkan kepada Allah SWT yang telah memberikan rahmat dan petunjuk kepada penyusun sehingga dapat menyelesaikan Tugas Akhir Skripsi dan laporan ini dengan judul "Aplikasi *Remote and Monitoring* berbasis *Java Remote Method Invocation (RMI)*". Pembuatan tugas akhir ini sebagai salah satu syarat untuk memperoleh gelar Sarjana Pendidikan Teknik Fakultas Teknik Universitas Negeri Yogyakarta.

Penyusun mengucapkan terima kasih kepada semua pihak atas bantuan dan bimbingan dalam pembuatan tugas akhir skripsi ini, sehingga penyusun dapat menyelesaikan laporan tugas akhir skripsi ini tepat waktu. Dengan kerendahan hati, pada kesempatan ini penyusun mengucapkan rasa terima kasih kepada:

1. Prof. Dr. Rochmat Wahab M.Pd, MA, selaku Rektor Universitas Negeri Yogyakarta.
2. Wardan Suyanto, Ed.D, selaku Dekan Fakultas Teknik UNY.
3. Masduki Zakaria, M.T, selaku Ketua Jurusan Pendidikan Teknik Elektronika Fakultas Teknik UNY.
4. Dr. Eko Marpanaji, selaku Dosen Pembimbing Tugas Akhir Skripsi.
5. Keluarga penyusun yang selalu memberikan doa, semangat dan bantuan yang tiada henti, serta telah menjadi guru dan sandaran terbaik dalam hidup.

6. Teman-teman Prodi PT. Informatika UNY angkatan 2007, atas semangat dan dukungan yang diberikan.
7. Bonita Destiana yang selalu mendukung dan menemani di perpustakaan.
8. Semua pihak yang tidak dapat penyusun sebutkan satu persatu yang telah membantu penyusun hingga tersusunnya laporan tugas akhir skripsi ini.

Berbagai upaya telah penyusun lakukan untuk menyelesaikan tugas akhir ini, namun penyusun menyadari bahwa dalam penyusunan laporan tugas akhir skripsi ini masih jauh dari kesempurnaan. Semoga laporan tugas akhir akhir ini dapat memberikan manfaat bagi seluruh pembaca.

وَالسَّلَامُ عَلَيْكُمْ وَرَحْمَةُ اللهِ وَبَرَكَاتُهُ

Yogyakarta, 2011

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

A. Latar Belakang Masalah

Ilmu Pengetahuan dan Teknologi (IPTEK) merupakan ilmu yang terus berkembang pesat. Proses perkembangan IPTEK selalu diikuti dengan tingkah laku manusia yang mengikuti tren. Dahulu sebelum alat komunikasi *handphone* muncul, warung telekomunikasi (Wartel) masih ramai dikunjungi. Hadirnya *handphone* menjadikan bukan wartel yang ramai dikunjungi, tetapi warung pulsa dengan kebutuhan pulsa seakan menjadi kebutuhan pokok. Perkembangan teknologi yang tidak kalah pesat adalah perkembangan teknologi komputer yang semakin lama semakin *portable* dan *mobile.*

Perkembangan teknologi komunikasi menjadikan ruang sempit komunikasi yang menyebabkan adanya teknologi jaringan komputer. Jaringan komputer dikembangkan untuk memenuhi kebutuhan komunikasi antar komputer yang satu dengan lainnya. Komunikasi antar jaringan komputer memungkinkan juga bertukar sumber daya yang bisa berbentuk *hardware* atau *software.* Keuntungan yang didapat dari teknologi jaringan komputer membuat pengguna jaringan komputer memanfaatkan perangkat – perangkat yang dipunyai dari *hardware* maupun *software* untuk memaksimalkannya.

Sebuah jaringan komputer yang tangguh tentunya tidak hanya didukung oleh perangkat keras yang berteknologi tinggi saja, akan tetapi juga perlu didukung dengan penggunaan perangkat lunak yang sesuai dengan

kebutuhan. Kombinasi dari penggunaan perangkat keras yang tangguh yang dipadu dengan perangkat lunak yang sesuai dengan kebutuhan serta berdayaguna akan membuat sebuah jaringan komputer menjadi lebih bermanfaat bagi pemakai jaringan.

Kemampuan jaringan komputer yang sudah sudah dirasa cukup tentu tidak akan lepas dari masalah. Jaringan komputer minimal terdiri dari dua komputer. Pengawasan dan pengendalian jarak jauh perlu dilakukan untuk mengantisipasi berbagai masalah. Jumlah komputer yang lebih dari dua puluh itu sudah banyak, dengan adanya pengawasan dan pengendalian jarak jauh tentu akan lebih sangat membantu. Pengawasan dan pengendalian jarak jauh tentu akan mempercepat penanganan daripada harus mendatangi komputer yang bermasalah satu persatu. Pengawasan dan pengendalian jarak jauh akan membantu *administrator* jaringan tetap berada dikursi nyamannya saat ada masalah pada komputer – komputer yang terhubung dalam sebuah jaringan.

Penelitian tentang *Remote and Monitoring* diantaranya adalah Irwan Pribadi dan Mukhammad Andri Setiawan dengan judul Manajemen Pengelolaan *LAN* dengan *Remote System Application (2005),* yang diteliti adalah *Remote System Application* pada jaringan *Local Area Network (LAN)* menggunakan *software development* Visual Basic. Peneliti lain adalah You, Xiang-bai Liu, Yi-min Xu, Wang-ming dengan judul *The Design of a Remote Monitoring System based On Java(2010).* Penelitian ini mengkhususkan *Remote and Monitoring* dalam jaringan internet dan *Code Division Multiple Access (CDMA).* Peneliti lain adalah Yunus Kurniawan dengan judul

Pembangunan Aplikasi *Remote Task Manager* pada Jaringan Komputer Berbasis Windows(2010). Penelitian ini menggunakan bahasa pemrograman Microsoft C# dalam lingkungan minimal NET Framework 2.0.

Berdasarkan hasil penelitian tersebut, nampak bahwa pemanfaatan *Java Remote Method Invocation (RMI)* masih sedikit yang meneliti untuk keperluan pembuatan *Remote System and Monitoring.* Dengan demikian, penelitian tentang *Remote System and Monitoring* masih diperlukan. Penelitian ini akan mengkaji *Remote System and Monitoring* yang *opensource* dan bisa dijalankan pada berbagai system operasi *(multiplatform).* Salah satu bahasa pemrograman yang *multiplatform* adalah java. Peneliti menyimpulkan untuk mengusulkan penelitian yang berjudul "Aplikasi *Remote and Monitoring* berbasis *Java Remote Method Invocation (RMI)"*.

B. Identifikasi Masalah

Berdasarkan latar belakang masalah di atas, maka dapat diidentifikasikan beberapa masalah yang ada antara lain :

1. Penanganan banyak komputer yang bermasalah akan melelahkan untuk mendatangi komputer – komputer tersebut.
2. Pemakai komputer yang belum mengerti akan aturan penggunaan komputer *User Workstation* dan aturan hak akses membuat komputer butuh pengawasan yang lebih dari administrator jaringan.
3. Java *Remote Method Invocation* masih sedikit digunakan dalam pengembangan aplikasi *remote system.*

C. Batasan Masalah

Berdasarkan identifikasi masalah di atas, maka pembatasan masalahnya adalah pengendalian dan pengawasan jarak jauh terhadap komputer yang terhubung dengan jaringan. Semua komputer bisa diawasi secara bersamaan dari sebuah komputer yang digunakan oleh *network* administrator. Komputer yang dimaksud dibatasi untuk *Local Area Network (LAN)*. Batasan lain dalam penelitian ini juga akan disesuaikan dengan analisis kebutuhan dan desain sistem.

D. Rumusan Masalah

Berdasarkan latar belakang masalah, identifikasi masalah dan batasan masalah maka dibuat rumusan masalah sebagai berikut:

1. Bagaimana pengembangan aplikasi *Remote and Monitoring* berbasis *Java RMI*?
2. Bagaimana unjuk kerja aplikasi *Remote and Monitoring* berbasis *Java RMI*?
3. Bagaimana kelayakan aplikasi *Remote and Monitoring* berbasis *Java RMI* dari segi *usability*?

E. Tujuan Penelitian

Tujuan yang ingin dicapai dalam tugas akhir skripsi *Remote and Monitoring* berbasis *Java RMI* adalah sebagai berikut :

1. Mengetahui bagaimana pengembangan aplikasi *Remote and Monitoring* berbasis *Java RMI*.

2. Mengetahui unjuk kerja aplikasi *Remote and Monitoring* berbasis *Java RMI.*
3. Mengetahui kelayakan aplikasi *Remote and Monitoring* berbasis *Java RMI* dari segi *usability* .

F. Manfaat Penelitian

Hasil penelitian ini diharapkan dapat memberikan manfaat:

1. Bagi Peneliti, diharapkan dapat menambah dan meningkatkan wawasan, pengetahuan serta sebagai ajang latihan dalam menerapkan teori-teori yang pernah dipelajari di bangku kuliah.
2. Bagi penyelenggara pendidikan, penelitian ini dapat digunakan sebagai masukan atau memberi gambaran tentang rekayasa perangkat lunak dari tahap analisis kebutuhan, desain, pengkodean, dan pengujian.
3. Bagi masyarakat umum, khususnya administrator jaringan, penelitian ini dapat digunakan sebagai aplikasi pengendali dan pengawas jarak jauh untuk komputer yang menjadi tanggung jawabnya.

BAB II

KAJIAN PUSTAKA

A. Kerangka Teori

1. *Remote* dan *Monitoring*

Pengendali jarak jauh (Inggris: *remote control*) adalah sebuah alat elektronik yang digunakan untuk mengoperasikan sebuah mesin dari jarak jauh (Wikipedia, 2011).

Istilah *remote control* juga sering disingkat menjadi *"remote"* saja. *Remote* dalam bahasa Indonesia berarti jauh dan *control* berarti kendali. *Remote* juga sering kali mengacu pada istilah *"controller"*, *"donker"*, *"doofer"*, *"zapper" "click-buzz"*, *"box"*, *"flipper"*, *"zippity"*, *"clicker"*, atau *"changer"*. Umumnya, pengendali jarak jauh digunakan untuk memberikan perintah dari kejauhan kepada televisi atau barang – barang elektronik lainnya seperti sistem stereo dan pemutar DVD. *Remote control* untuk perangkat – perangkat elektronik ini biasanya berupa benda kecil nirkabel yang dipegang dalam tangan dengan sederetan tombol untuk menyesuaikan berbagai *setting*, misalnya saluran televisi, nomor *trek*, dan volume suara. Kebanyakan piranti modern dengan kontrol seperti ini, *remote control*-nya memiliki segala kontrol fungsi, sementara perangkat yang dikendalikan itu sendiri hanya mempunyai sedikit kontrol utama yang mendasar. Kebanyakan *remote* berkomunikasi dengan perangkatnya masing-masing melalui sinyal – sinyal infra merah dan beberapa saja melalui sinyal radio.

Sebuah *remote system* merupakan dua buah titik yang saling berhubungan, dimana satu titik menjadi pengendali dan satu titik lainnya menjadi target. Titik target akan menjadi *passive* atau dalam istilah komunikasi disebut dengan status *listen*. Titik pengendali akan bersifat *active* dengan mengirimkan sebuah data atau lebih yang nantinya akan diterjemahkan oleh titik target sebagai proses/perintah yang harus dijalankan oleh target yang merupakan proses *remoting*.

Monitoring adalah proses rutin pengumpulan data dan pengukuran kemajuan atas objektif program. Memantau perubahan, yang fokus pada proses dan keluaran (Hafidz, 2009:1).

Remote and Monitoring adalah pengendalian dan pengawasan jarak jauh melalui jaringan komputer. Komputer – komputer yang terhubung dapat dikendalikan dan diawasi oleh administrator jaringan.

2. Jaringan Komputer

Tahun 1940-an di Amerika ada sebuah penelitian yang ingin memanfaatkan sebuah perangkat komputer secara bersama. Tahun 1950-an ketika jenis komputer mulai membesar sampai terciptanya super komputer. Mahalnya harga perangkat komputer menjadikan tuntutan sebuah komputer harus melayani beberapa terminal. Dari sinilah maka muncul konsep distribusi proses berdasarkan waktu yang dikenal dengan nama TSS (*Time Sharing System*), bentuk pertama kali jaringan (network) komputer diaplikasikan. Pada sistem TSS beberapa terminal terhubung secara seri ke sebuah *host* komputer.

TSS berkembang menjadi proses distribusi (*Distributed Processing*). Dalam proses ini beberapa host komputer mengerjakan sebuah pekerjaan besar secara paralel untuk melayani beberapa terminal yang tersambung secara seri disetiap *host* komputer.

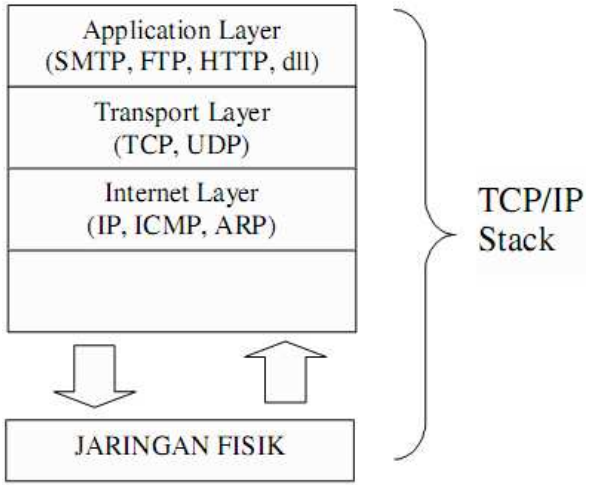
Harga - harga komputer kecil sudah mulai menurun dan konsep proses distribusi sudah matang menjadikan penggunaan komputer dan jaringannya sudah mulai beragam dari mulai menangani proses bersama maupun komunikasi antar komputer (*Peer to Peer System*) saja tanpa melalui komputer pusat. Komunikasi komputer *peer to peer* adalah awal teknologi jaringan sebelum jaringan lokal yang dikenal dengan sebutan LAN (Local Area Network) muncul. Demikian pula ketika Internet mulai diperkenalkan, maka sebagian besar LAN yang berdiri sendiri mulai berhubungan dan terbentuklah jaringan raksasa ditingkat dunia yang disebut dengan istilah WAN (Word Area Network).

a. TCP/IP

Inti dari TCP/IP (Tranmission Control Protocol/Internet Protocol) yang merupakan kombinasi dari dua protokol yang bekerja bersama – sama untuk memberikan koneksi pada internet/intranet. IP dipakai untuk mendefinisikan dan mengirimkan datagram (unit data internet) dan menyatakan skema pengalamatan. TCP bertanggung jawab terhadap sevis-servis level atas.

TCP/IP terdiri dari bagian-bagian tertentu dari komunikasi data. Protokol TCP/IP dapat diterapkan dengan mudah di setiap jenis

komputer dan interface jaringan, karena sebagian besar kumpulan protokol ini tidak spesifik terhadap satu komputer atau peralatan jaringan tertentu. Agar TCP/IP dapat berjalan di atas interface jaringan tertentu, hanya perlu dilakukan perubahan pada protokol yang berhubungan dengan antarmuka jaringan saja. Sekumpulan protocol TCP/IP ini dimodelkan dengan empat layer TCP/IP, sebagaimana diperlihatkan Gambar 1.



Gambar 1. *Layer TCP/IP*

1) Network Interface Layer

Bertanggung jawab mengirimdan menerima data dari media fisik. Media fisik ini berupa kabel, serat optik, atau gelombang radio. Protokol pada layer ini harus mampu menerjemahkan sinyal listrik menjadi data digital yang dimengerti komputer, yang berasal dari peralatan sejenisnya.

2) Internet Layer

Bertanggung jawab dalam proses pengiriman paket ke alamat yang tepat. Layer ini terdapat tiga macam protocol, yaitu IP

(Internet Protocol), ARP *(Address Resolution Protocol)* dan ICMP *(Internet Control Message Protocol).*

3) Transport Layer

Bertanggung jawab untuk mengadakan komunikasi antar dua host/komputer. Kedua protocol tersebut ialah TCP *(Transmission Control Protocol)* dan UDP *(User Datagram Protocol).*

4) Application Layer

Terletaknya semua aplikasi yang menggunakan protocol TCP/IP.

b. Socket

Adalah sebuah resource yang disediakan untuk melewatkan pengiriman dan penerimaan data pada komunikasi data. Pemograman socket adalah bagian dari pemograman tingkat rendah. Terdapat tiga jenis koneksi socket yaitu:

1) Koneksi client

Dimulai oleh client dan menghubungkan socket client lokal dengan sebuah socket server remote. Socket client harus menyatakan server yang akan dihubungi dengan memberikan nama host atau alamat IP dan portnya

2) Koneksi ‘mendengarkan’

Socket server bersifat pasif dan menunggu client. Jika client memberikan request baru server membuat socket baru khusus untuk koneksi tersebut dan ‘mendengarkan’ lagi. Socket server ini harus menyatakan port yang menyatakan servis yang diberikan.

3) Koneksi server

Koneksi yang diaktifkan oleh server pada saat server menerima request dari client.

Tipe-tipe koneksi tersebut hanya penting untuk membangun hubungan antara client dan server. Setelah hubungan terbentuk, kedua sisi bebas untuk membuat request dan menerima data ke sisi lain.

c. Port

Untuk dapat berkomunikasi pada protokol TCP/IP, diperlukanlah sebuah pintu yang sama pada titik-titik yang melakukan proses komunikasi. Pintu inilah yang disebut dengan port komunikasi. Port komunikasi sangat beragam nilainya, dimana jumlah port yang tersedia sangat banyak, dan agar dapat melakukan proses komunikasi data, protokol TCP/IP harus menunjuk sebuah port tertentu.

Tabel 1. Daftar *Port* Standar

Nilai *Port*	**Standar Penggunaan *Port***
139	Koneksi
80	*Web*
21	*FTP*
8080, 3128	*Proxy, Firewall*
6667	*IRC, Chatting*
23	*Telnet*

3. Rekayasa Perangkat Lunak Berorientasi Objek

a. Perangkat Lunak

1) Pengertian Perangkat Lunak

Perangkat lunak adalah perintah (program komputer) yang bila dieksekusi memberikan fungsi dan unjuk kerja seperti yang di

inginkan (Pressman, 2002:10). Perangkat lunak dapat berupa program atau prosedur. Perangkat lunak mempunyai spesifikasi dan tujuan yang disesuaikan dengan fungsinya.

2) Karakteristik Perangkat Lunak

Perangkat lunak lebih merupakan elemen logika dan bukan merupakan elemen sistem fisik. Perangkat lunak memiliki ciri yang berbeda dari perangkat keras (Pressman, 2002:10):

a) Perangkat lunak dibangun dan dikembangkan, tidak dibuat dalam bentuk yang klasik.
b) Perangkat lunak tidak pernah usang.
c) Sebagian besar perangkat lunak dibuat secara *custom-built*, serta tidak dapat dirakit dari komponen yang sudah ada.

3) Komponen Perangkat Lunak

Komponen perangkat lunak dibangun dengan bahasa pemrograman yang memiliki kosakata yang terbatas, sebuah tata bahasa yang dibatasi secara eksplisit, serta aturan - aturan syntax dan semantik yang dibentuk secara baik. Pada tingkat yang paling rendah, bahasa – bahasa tersebut mencerminkan serangkaian instruksi perangkat keras. Pada tingkat sedang, bahasa pemrograman seperti *C* atau *Smaltalk*, dipakai untuk membuat deskripsi prosedural dari program. Pada tingkat yang paling tinggi, bahasa-bahasa tersebut menggunakan ikon grafik atau simbol lain untuk mewakili

kebutuhan akan sebuah pemecahan. Instruksi-instruksi yang dapat dieksekusi dibuat secara otomatis.

Kode mesin, bahasa *assembly* (tingkat mesin), bahasa pemrograman tingkat menengah, sering disebut tiga generasi bahasa komputer yang pertama. Dengan bahasa-bahasa tersebut, pemrogram harus melihat dengan baik kekhususan struktur informasi maupun kontrol pemrograman itu sendiri. Bahasa di dalam tiga generasi yang pertama dimasukan ke dalam jenis bahasa prosedural.

Bahasa generasi keempat disebut juga sebagai bahasa nonprosedural. Bahasa ini menggerakan pengembang perangkat lunak untuk mengkhususkan pada detail prosedural. Bahasa nonprosedural secara tidak langsung menyatakan sebuah program melalui spesifikasi hasil yang diharapkan, dan tidak pada aksi yang dibutuhkan untuk mencapai hasil tersebut. Perangkat lunak penopang menerjemahkan spesifikasi hasil ke dalam program mesin yang dapat dieksekusi.

Komponen perangkat lunak kualitas tinggi memiliki ciri penting yakni *reusability*. Ini berarti sebuah komponen perangkat lunak harus didesain dan diimplementasi sehingga dapat dipakai lagi pada berbagai program yang berbeda. Komponen - komponen *reusable* modern mengenkapsulasi data dan pemrosesan yang diaplikasikan ke data yang memungkinkan insinyur perangkat lunak membuat aplikasi baru dari bagian-bagian yang dapat digunakan

kembali. Sebagai contoh, *interface* interaktif saat ini dibuat dengan menggunakan komponen *reusable* yang mampu membuat jendela-jendela grafis, menu-menu *pull-down*, dan mekanisme interaktif dengan variasi yang sangat luas. Struktur data dan detail pemrosesan yang dibutuhkan untuk membangun interface, diisikan ke dalam sebuah pustaka komponen *reusable* untuk konstruksi *interface*.

4) Aplikasi Perangkat Lunak

Perangkat lunak dapat diaplikasikan ke berbagai situasi di mana serangkaian langkah prosedural (seperti algoritma) telah didefinisikan (pengecualian – pengecualian yang dapat dicatat pada aturan ini adalah sistem pakar dan perangkat lunak jaringan syaraf kecerdasan buatan). Kandungan (*content*) informasi dan determinasi merupakan faktor penting dalam menentukan sifat aplikasi perangkat lunak. *Content* mengarah kepada arti bentuk dari informasi yang masuk dan keluar. Sebagai contoh, banyak aplikasi bisnis memakai data input yang terstruktur secara tinggi (sebuah *database*) dan menghasilkan laporan yang sudah terformat.

Area perangkat lunak berikut menunjukan luasnya aplikasi potensial (Pressman, 2002:16):

a) Perangkat Lunak Sistem

Perangkat lunak sistem merupakan sekumpulan program yang ditulis untuk melayani program-program yang lain. Area perangkat lunak sistem ditandai dengan eratnya interaksi dengan

perangkat keras komputer, penggunaan oleh banyak pemakai, operasi konkuren yang membutuhkan penjadawalan, tukar menukar sumber, dan pengaturan proses yang canggih.

b) Perangkat Lunak *Real-Time*

Perangkat lunak *real-time* merupakan program-program yang memonitor atau menganalisis kejadian dunia nyata pada saat terjadinya. *Real-time* berbeda dengan interaksi atau *timesharing*. Sistem *real-time* harus merespon di dalam suatu rentang waktu yang tetap. Waktu respon sebuah sistem interaktif (atau *timesharing*) secara normal dapat diperpanjang tanpa memberikan risiko kerusakan pada hasil.

c) Perangkat Lunak Bisnis

Pemrosesan informasi bisnis merupakan area aplikasi perangkat lunak yang paling luas. Aplikasi dalam area ini menyusun kembali struktur data yang ada dengan suatu cara tertentu untuk memperlancar operasi bisnis atau pengambilan keputusan manajemen. Selain itu, aplikasi perangkat lunak bisnis juga meliputi penghitungan klien/server seta penghitungan interaktif.

d) Perangkat Lunak Teknik dan Ilmu Pengetahuan

Perangkat lunak teknik dan ilmu pengetahuan ditandai dengan algoritma *number crunching*. Perangkat lunak ini memiliki jangkauan aplikasi mulai dari astronomi sampai

vulakanologi, dari analisis otomotif sampai dinamika orbit pesawat ruang angkasa, dan dari biologi molekuler sampai pabrik yang sudah diotomatisasi.

e) *Embedded Software*

Produk pintar telah menjadi bagian yang umum bagi hampir semua konsumen dan pasar industri. *Embedded software* ada dalam *read-only memory* dan dipakai untuk mengontrol hasil serta sistem untuk keperluan konsumen dan pasar industri.

f) Perangkat Lunak Komputer Personal

Pasar perangkat lunak komputer personal telah berkembang selama dekade terakhir. Pengolah kata, *spreadsheet*, multimedia, hiburan, manajemen database, jaringan eksternal atau akses database hanya merupakan beberapa saja dari ratusan aplikasi yang ada.

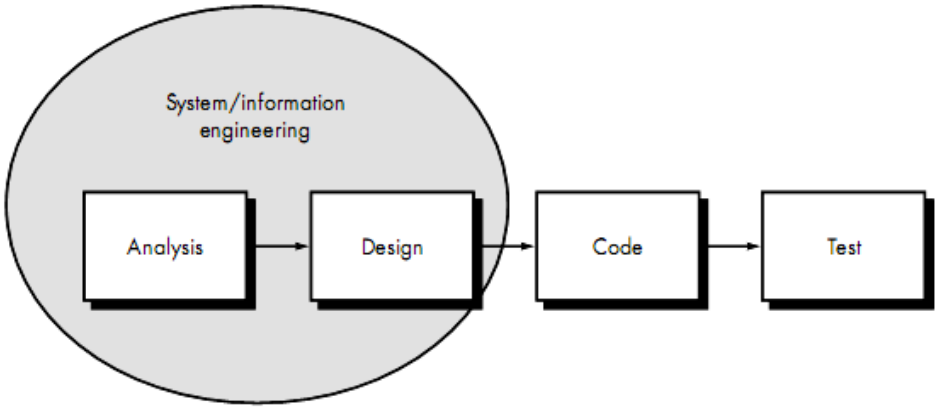
g) Perangkat Lunak Kecerdasan Buatan

Perangkat lunak kecerdasan buatan (*Artificial Inteligent* (AI)) menggunakan algoritma non-numeris atau memecahkan masalah kompleks yang tidak sesuai untuk perhitungan atau analisis secara langsung. Area kecerdasan buatan yang aktif adalah sistem pakar, disebut juga sistem bebasis ilmu pengetahuan. Tetapi area aplikasi lainnya untuk perangkat lunak kecerdasan buatan adalah pengakuan pola (*image* dan *voice*), pembuktian teorema, dan permainan *game*.

5) Model Proses Perangkat Lunak *Linear Sequential*

Model proses perangkat lunak (atau disebut juga paradigma rekayasa perangkat lunak) adalah suatu strategi pengembangan yang memadukan lapisan proses, metode, dan alat serta tahap-tahap generik. Model proses untuk rekayasa perangkat lunak dipilih berdasarkan sifat proyek dan aplikasi, metode dan alat yang digunakan, serta pengendalian dan hasil yang diinginkan. Berikut adalah beberapa model proses pengembangan perangkat lunak (Nugroho, 2009:18):

Linear sequential model (atau disebut juga *"classic life cycle"* atau *"waterfall model")* adalah metode pengembangan perangkat lunak dengan pendekatan sekuensial dengan cakupan aktivitas:



Gambar 2. *Waterfall* menurut *Pressman*

a) *Analysis*

Proses pengumpulan kebutuhan diintensifkan dan difoukuskan, khususnya pada perangkat lunak. Untuk memahami sifat program yang dibangun, perekayasa perangkat lunak (analis) harus memahami domain informasi,

tingkah laku, unjuk kerja, dan antarmuka (interface) yang diperlukan. Kebutuhan baik untuk system maupun perangkat lunak didokumentasikan dan dilihat lagi dengan pelanggan.

b) *Design*

Desain perangkat lunak sebenarnya adalah proses multi langkah yang berfokus pada empat atribut sebuah program yang berbeda; struktur data, arsitektur perangkat lunak, representasi interface, dan detail (algoritma) procedural. Proses desain menerjemahkan syarat/kebutuhan ke dalam sebuah representasi perangkat lunak yang dapat diperkirakan demi kualitas sebelum dimulai pemunculan kode. Sebagaimana persyaratan, desain didokumentasikan dan menjadi bagian dari konfigurasi perangkat lunak.

c) *Code*

Desain harus diterjemahkan ke dalam bentuk mesin yang bisa dibaca. Jika desain dilakukan dengan cara yang lengkap maka pembuatan dapat diselesaikan secara mekanis.

d) *Test*

Sekali kode dibuat, pengujian program dimulai. Proses pengujian berfokus pada logika internal perangkat lunak, memastikan bahwa semua pernyataan sudah diuji, dan pada eksternal fungsional yaitu mengarahkan pengujian untuk menemukan kesalahan – kesalahan dan memastikan bahwa

input yang dibatasi akan memberikan hasil actual yang sesuai dengan hasil yang dibutuhkan.

White-box testing (kadang-kadang disebut sebagai *glass-box testing*) adalah metode desain kasus uji yang menggunakan struktur kontrol desain prosedural untuk memperoleh kasus uji. Penggunaan metode pengujian *white-box*, perekayasa sistem dapat melakukan kasus uji yang:

(1) Memberikan jaminan bahwa semua jalur independen pada suatu modul telah digunakan paling tidak satu kali.
(2) Menggunakan semua keputusan logis pada sisi true dan false.
(3) Mengeksekusi semua loop pada batasan mereka dan pada batas operasional mereka.
(4) Menggunakan struktur data internal untuk menjamin validitasnya.

Black-box testing fokus pada persyaratan fungsional perangkat lunak. Pengujian ini memungkinkan pelaku RPL mendapatkan serangkaian kondisi input yang memenuhi persyaratan fungsional suatu program. Pengujian ini berusaha menemukan kesalahan dengan kategori sebagai berikut:

(1) Fungsi-fungsi yang salah atau hilang.
(2) Kesalahan antarmuka.
(3) Kesalahan struktur data atau akses basisdata eksternal.

(4) Kesalahan kinerja.

(5) Kesalahan inisialisasi atau terminasi

Verifikasi dan validasi merupakan dua istilah yang sering dikaitkan dengan tahapan pengujian perangkat lunak. Verifikasi mengacu pada serangkaian aktivitas untuk memastikan bahwa perangkat lunak mengimplementasikan fungsi tertentu secara benar, sedangkan validasi mengacu pada serangkaian aktivitas untuk memastikan bahwa perangkat lunak yang telah dibuat sesuai dengan kebutuhan konsumen.

Validation testing dilakukan setelah perangkat lunak selesai dirangkai sebagai suatu kesatuan dan semua kesalahan interfacing telah ditemukan dan dikoreksi. Validasi dikatakan berhasil jika perangkat lunak berfungsi sesuai dengan kebutuhan konsumen.

Validasi perangkat lunak diperoleh melalui sederetan pengujian *Black-Box* yang memperlihatkan kesesuaian dengan kebutuhan perangkat lunak. Semua kasus uji dirancang untuk memastikan apakah semua persyaratan fungsional telah dipenuhi, semua persyaratan kinerja tercapai, semua dokumentasi telah benar, dan persyaratan lainnya dipenuhi.

Alpha testing, yakni pengujian yang dilakukan pada perangkat lunak oleh pengguna dengan adanya supervisi dan kontrol dari pengembang perangkat lunak. Berdsasarkan pengujian *alpha* maka didapatkan revisi untuk dilanjutkan ke pengujian selanjutnya.

Beta testing, yakni pengujian yang dilakukan pada perangkat lunak oleh end-user tanpa adanya supervisi dan kontrol dari pengembang perangkat lunak. Jika ditemukan masalah, maka pengguna akan melaporkannya kepada pengembang perangkat lunak tersebut. Berdasarkan pengujian *beta* maka didapatkan revisi akhir.

Beberapa kelemahan *linear sequential model* atau metode "*classic life cycle*" atau "*waterfall model*":

a) Proyek yang sebenarnya jarang mengikuti alur sekuensial, sehingga perubahan yang terjadi dapat menyebabkan hasil yang sudah didapat tim harus diubah kembali.
b) *Linear sequential* model mengharuskan semua kebutuhan pemakai sudah dinyatakan secara eksplisit di awal proses, tetapi kadang-kadang hal ini tidak dapat terlaksana karena kesulitan yang dialami pemakai saat akan mengungkapkan semua kebutuhannya tersebut.
c) Pemakai harus bersabar karena versi dari program tidak akan didapat sampai akhir rentang waktu proyek.

d) Adanya waktu menganggur bagi pengembang, karena harus menunggu anggota tim proyek lainnya menuntaskan pekerjaannya.

b. Rekayasa Perangkat Lunak Berorientasi Objek

Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan sistem (Shalahudin, 2008). Pendekatan berorientasi objek akan memandang sistem yang akan dikembangkan sebagai suatu kumpulan objek yang berkorespondensi dengan objek-objek dunia nvata. Ada banyak cara untuk mengabstraksikan dan memodelkan objek-objek tersebut, mulai dari abstraksi objek, kelas, hubungan antar kelas sampai abstraksi sistem. Saat mengabstraksikan dan memodelkan objek ini, data dan proses-proses yang dipunyai oleh objek akan dienkapsulasi (dibungkus) menjadi satu kesatuan.

Konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancangan, pemrograman, dan pengujian perangkat lunak. Ada berbagai teknik yang dapat digunakan pada masing-masing tahap tersebut, dengan aturan dan alat bantu pemodelan tertentu.

Sistem berorientasi objek merupakan sebuah sistem yang dibangun dengan berdasarkan metode berorientasi objek adalah sebuah sistem yang komponennya dibungkus (dienkapsulasi) menjadi kelompok data dan fungsi. Setiap komponen dalam sistem tersebut

dapat mewarisi atribut dan sifat dari komponen lainnya, dan dapat berinteraksi satu sama lain.

Karakteristik sebuah sistem berorientasi objek sebagai berikut (Shalahudin, 2008):

1) Abstraksi
   Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek - aspek lain yang tidak sesuai dengan permasalahan.
2) Enkapsulasi
   Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek. Untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.
3) Pewarisan (*inheritance*)
   Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.
4) *Reusabilily*
   Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.
5) Generalisasi dan Spesialisasi
   Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus.
6) Komunikasi Antar Objek
   Komunikasi antar objek dilakukan lewat pesan (*message*) yang dikirim dan satu objek ke objek lainnya.
7) *Polymorphism*
   Kemampuan suatu objek untuk digunakan di banyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

1) Metodologi Berorientasi Objek

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnva. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui

pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek.

Pada saat ini, metode berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentransformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekatan terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan pada saat ini sangat beragam (aplikasi bisnis, *real-time*, *utility*, dan sebagainya) dengan platform yang berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi tersebut.

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut (Shalahudin, 2008):

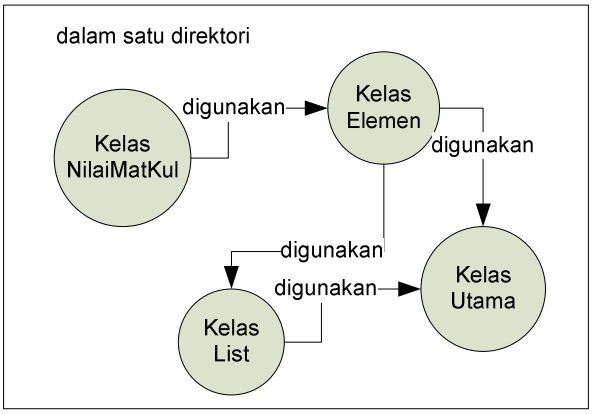
a) Meningkatkan produktivitas
   Kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnva yang melibatkan objek tersebut (*reusable*).
b) Kecepatan pengembangan
   Sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengkodean.
c) Kemudahan pemeliharaan
   Dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah

d) Adanya konsistensi
   Sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.
e) Meningkatkan kualitas perangkat lunak
   Pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengembanganna, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

2) Pengertian Objek dan Kelas

Kelas adalah kumpulan dan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dan himpunan objek yang sama yang mungkin lahir atau diciptakan dan kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut). Kelakuan (operasi/metode), hubungan (*relationship*) dan arti. Suatu kelas dapat diturunkan dan kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru.

Secara teknis, kelas adalah sebuah struktur tertentu dalam pembuatan perangkat lunak. Kelas merupakan bentuk struktur pada kode program yang menggunakan metodologi berorientasi objek. Ilustrasi dari sebuah kelas dapat dilihat pada gambar berikut.

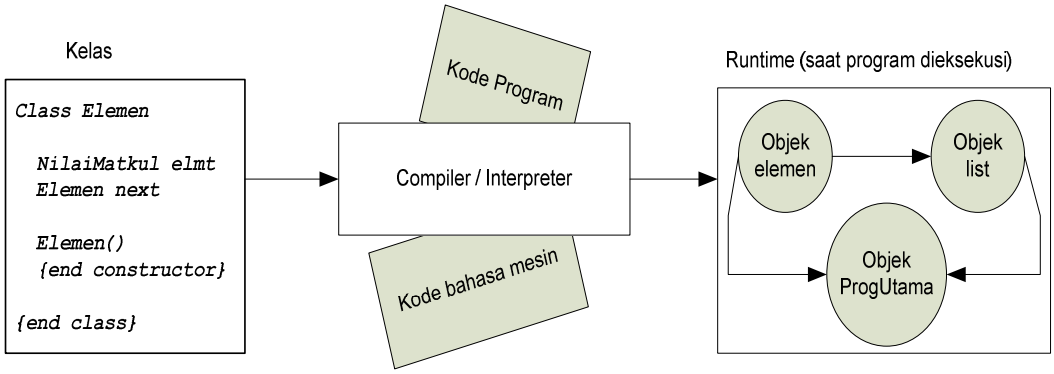


Gambar 3. Ilustrasi Kelas

Sebuah kelas lebih fleksibel untuk digunakan oleh kelas lain.

Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunvai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.

Sebuah kelas saat program dieksekusi maka akan dibuat sebuah objek. Objek dilihat dari segi teknis adalah elemen pada saat runtime yang akan diciptakan, dimanipulasi, dan dihancurkan saat eksekusi sehingga sebuah objek hanya ada saat sebuah program dieksekusi, jika masih dalam bentuk kode, disebut sebagai kelas jadi pada saat runtime (saat sebuah program dieksekusi), yang kita punya adalah objek, di dalam teks program yang kita lihat hanyalah kelas. Ilustrasi kelas dan objek dapat dilihat pada gambar berikut.



Gambar 4. Ilustrasi Kelas dan Objek

3) Enkapsulasi

Enkapsulasi dapat dianggap sebagai sebuah bungkusan. Enkapsulasi inilah yang diimplementasikan dalam sebuah kelas dimana di dalam sebuah kelas terdiri dari atribut dan metode yang dibungkus dalam suatu kelas. Enkapsulasi pada sebuah kelas bertujuan untuk melindungi atribut dan metode-metode yang ada di dalam kelas agar tidak sembarangan diakses oleh kelas lain.

4) Atribut

Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya.

5) Operasi atau Metode (Method)

Operasi atau metode atau method pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek. Metode atau operasi dapat berasal dari:

a) Event

b) Aktivitas atau aksi keadaan

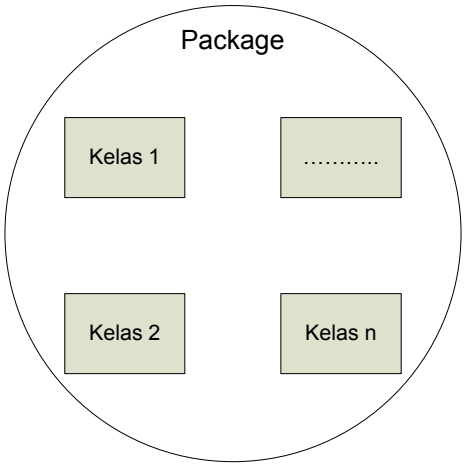
c) Fungsi

d) Kelakuan dunia nyata

Contoh metode atau operasi misalnya *Read, Write, Move, Copy,* dan sebagainya.

6) Pengertian Package

Package adalah sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam package yang berbeda. Ilustrasi dari sebuah package dapat dilihat pada gambar berikut.



Gambar 5. *Package*

7) Pengertian Antarmuka *(Interface)*

Antarmuka atau interface sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah interface dapat diimplementasikan oleh kelas lain. Sebuah kelas dapat mengimplementasikan lebih dari

satu antarmuka dimana kelas ini akan mendeklarasikan metode pada antarmuka yang dibutuhkan oleh kelas itu sekaligus mendefinisikan isinya pada kode program kelas itu. Metode pada antarmuka yang diimplementasikan pada suatu kelas harus sama persis dengan yang ada pada antarmuka, misalnya pada antarmuka terdapat deklarasi metode *printAnimal()* maka pada kelas yang mengimplementasikan metode itu harus ditulis sama. Antarmuka atau interface biasanya digunakan agar kelas yang lain tidak mengakses langsung ke suatu kelas, mengakses antarmukanya.

4. *Unified Modelling Language (*UML)

a. Pengertian UML

Pada perkembangan teknologi perangkat lunak, diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar orang di berbagai negara dapat mengerti pemodelan perangkat lunak. Seperti yang kita ketahui bahwa menyatukan banyak kepala untuk menceritakan sebuah ide dengan tujuan untuk memahami hal yang sama tidaklah mudah, oleh karena itu diperlukan sebuah bahasa pemodelan perangkat lunak yang dapat dimengerti oleh banyak orang.

Banyak orang yang telah membuat bahasa pemodelan pembangunaan perangkat lunak sesuai dengan teknologi pemrograman yang berkembang pada saat itu, misalnya yang sempat berkembang dan digunakan oleh banyak pihak adalah *Data Flow Diagram* (*DFD*) untuk

memodelkan perangkat lunak yang menggunakan pemrogramana prosedural atau struktural, kemudian juga ada *State Transition Diagram* (STD) yang digunakan untuk memodelkan sistem real time (waktu nyata).

Pada perkembangan teknik pemrograman berorientasi objek, muncullah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modeling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.

Seperti yang kita ketahui bahwa banyak hal di dunia sistem informasi yang tidak dapat dibakukan, semua tergantung kebutuhan, lingkungan dan konteksnya. Begitu juga dengan perkembangan penggunaan UML bergantung pada level abstraksi penggunaannya. Jadi belum tentu pandangan yang berbeda dalam penggunaan UML adalah suatu yang salah, tapi perlu ditelaah dimanakah UML digunakan dan hal apa yang ingin divisualkan. Secara analogi jika dengan bahasa yang kita gunakan sehari-hari, belum tentu penyampaian bahasa dengan puisi adalah hal yang salah. Sistem informasi bukanlah ilmu pasti,

maka jika ada banyak perbedaan dan interpretasi di dalam bidang sistem informasi merupakan hal yang sangat wajar.

b. Sejarah Singkat UML

Bahasa pemrograman berorientasi objek yang pertama dikembangkan dikenal dengan nama *Simula-67* yang dikembangkan pada tahun 1967. Bahasa pemrograman ini kurang berkembang dan dikembangkan lebih lanjut, namun dengan kemunculannya telah memberikan sumbangan yang besar pada developer pengembang bahasa pemrograman berorientasi objek selanjutnya.

Perkembangan aktif dari pemrograman berorientasi objek mulai menggeliat ketika berkembangnya bahasa pemrograman *Smalltalk* pada awal 1980-an yang kemudian diikuti dengan perkembangan bahasa pemrograman beorientasi objek yang lainnya seperti *C* objek, *C++*, *Eiffel*, dan *CLOS*. Secara aktual, penggunaan bahasa pemrograman berorientasi objek pada saat itu masih terbatas, namun telah banyak menarik perhatian di saat itu. Sekitar lima tahun setelah *Smalltalk* berkembang, maka berkembang pula metode pengembangan berorientasi objek. Metode yang pertama diperkenalkan oleh *Sally Shlaer* dan *Stephen Mellor* (*Shlaer-Mellor, 1988*) dan Peter Coad dan Edward Yourdon (*Coad-Yourdon, 1991*), diikuti oleh Grady Booch (Booch, 1991), James R. Rumbaugh, Michael R. Blaha, William Lorensen, Frederick Eddy, William Premerlani (Rumbaugh-Blaha-Premerlani-Eddy-Lorensen, 1991), dan masih banyak lagi. Buku

terkenal yang juga berkembang selanjutnya adalah karangan Ivar Jacobson (Jacobson, 1992) yang menerangkan perbedaan pendekatan yang fokus pada use case dan proses pengembangan. Sekitar lima tahun kemudian muncul buku yang membahas mengenai metodologi berorientasi objek yang diikuti dengan buku-buku yang lainnya. Di dalamnya juga membahas mengenai konsep, definisi, notasi, terminologi, dan proses mengenai metodologi berorientasi objek.

Karena banyaknya metodologi – metodologi yang berkembang pesat saat itu, maka muncullah ide untuk membuat sebuah bahasa yang dapat dimengerti semua orang. Usaha penyatuan ini banyak mengambil dari metodologi-metodologi yang berkembang saat itu. Maka dibuat bahasa yang merupakan gabungan dari beberapa konsep seperti konsep *Object Modelling Technique* (OMT) dari Rumbaugh dan Booch (1991), konsep *The Classes, Responsibilities, Collaborators* (CRC) dari Rebecca Wirfs-Brock (1990), konsep pemikiran Ivar Jacobson, dan beberapa konsep lainnya dimana James R. Rumbaigh, Grady Booch, dan Ivar Jacobson bergabung dalam sebuah perusahaan yang bernama *Rational Software Corporation* menghasilkan bahasa yang disebut dengan *Unified Modeling Language* (UML). Pada 1996, *Object Management Group* (OMG) mengajukan proposal agar adanya standardisasi pemodelan berorientasi objek dan pada bulan September 1997 UML diakomodasi oleh OMG sehingga sampai saat ini UML

telah memberikan kontribusinya yang cukup besar di dalam metodologi berorientasi objek dan hal-hal yang terkait di dalamnya.

c. View dan Diagram UML

Tidak ada batasan yang jelas antara aneka ragam konsep dan konstruksi di dalam UML, tapi untuk pemahaman yang lebih mudah, UML dibagi menjadi beberapa *view* atau pandangan. *View* atau pandangan adalah bagian yang simpel dari konstruksi pemodelan UML yang merepresentasikan aspek dari sebuah sistem. Pembagian menjadi *view* atau pandangan yang berbeda bukanlah sesuatu yang baku tergantung dari kebutuhan, tapi diharapkan dengan adanya *view* akan memudahkan kosntruksi UML. Satu atau lebih diagram merepresentasikan konsep notasi visual pada setiap view atau padangan.

Pada level atas, *view* atau pandangan dapat dibagi menjadi tiga area:

1) Klasifikasi struktural *(structural clasification)*
   mendeskripsikan hubungan segala hal yang ada di dalam sistem
2) Kelakuan dinamik *(dynamic behavior)*
   mendeskripsikan kelakuan sistem, atau urutan perubahan yang dialami sistem
3) Pengelolaan model *(model management)*
   mendeskripsikan keterkaitan organisasi dengan hirarki unit yang ada di dalam sistem

Tabel 2. Keterkaitan antara view dan diagram di dalam UML

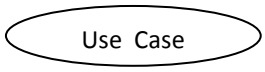
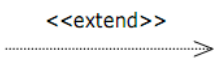
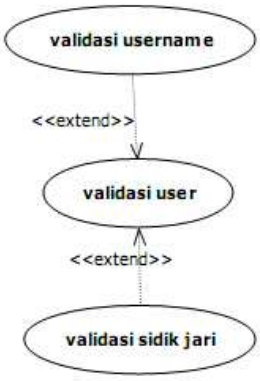
Area Mayor	View	Diagram
Struktural	Static view view atau pandangan yang tidak bergantung pada waktu	diagram kelas
	use case view view atau pandangan dari segi fungsionalitas sistem	diagram use case
	implementation view view atau pandangan dari segi komponen implementasi sistem	diagram komponen
	deployment view view atau pandangan dari segi node tempat komponen di-deploy	diagram deployment
dinamik	state machine view view atau pandangan dari segi status yang dialami sistem berdasarkan objek-objek sistem	diagram status
	activity view view atau pandangan dari segi aktivitas yang dilakukan oleh sistem	diagram aktivitas
	Diagram interaksi	diagram sekuen
		diagram kolaborasi
pengelolaan model (model-management)	model-management view view atau pandangan dari segi pengelolaan model sistem	diagram kelas

UML merupakan diagram yang saling terkait oleh karena itu perlu adanya kekonsistenan rancangan diagram yang satu dengan lainnya, bukan asal menggambar.

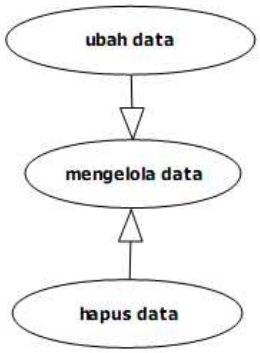
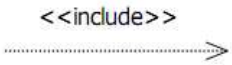
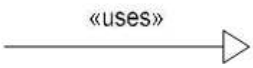
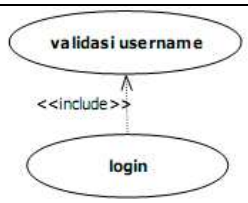
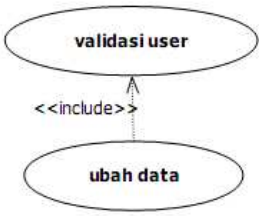
d. *Use Case Diagram*

Use case atau diagram *use case* merupakan pemodelan untuk kelakukan *(behavior)* suatu sistem. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah system dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Tabel 3. Simbol - simbol pada *Use case*

Simbol	Deskripsi
Use Case 	Fungsionalitas yang disediakan system sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal di awal frase nama *use case*
Aktor / *actor*  nama aktor	orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Asosiasi / *association*	komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan actor
Ekstensi / *extend* 	 relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan, misal arah panah mengarah pada *use case* yang ditambahkan

Tabel 3. Simbol - simbol pada *Use case (lanjutan)*

Simbol	Deskripsi
Generalisasi / *generalization* 	 Hubungan generalisasi dan spesialisasi (umum - khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya: arah panah mengarah pada *use case* yang menjadi generalisasinya (umum)
Menggunakan / *include* / *uses*  	 include berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* tambahan dijalankan, misal pada kasus berikut:  Kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan. arah panah include mengarah pada *use case* yang dipakai

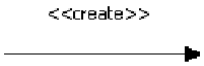
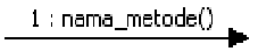
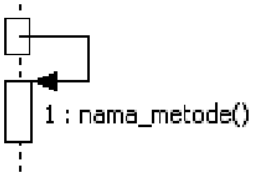
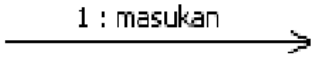
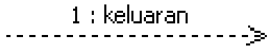
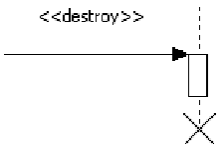
Arah panah relasi pada *use case* mengarah pada *use case* yang lebih besar kontrolnya atau yang dipakai.

e. *Sequence Diagram*

Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Banyaknya diagram sekuen yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah

didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.

Tabel 4. Simbol - simbol pada *Sequence Diagram*

Simbol	Deskripsi
Aktor 	orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar system informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Garis hidup / *lifeline*	menyatakan kehidupan suatu objek
Waktu aktif 	menyatakan objek yang berinteraksi
Pesan tipe create 	menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat
Pesan tipe call 	menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri,  arah panah mengarah pada objek yang memiliki operasi / metode, karena ini memanggil operasi / metode maka operasi / metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.
Pesan tipe send 	menyatakan bahwa suatu objek mengirimkan data / masukan / informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi
Pesan tipe return 	menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
Pesan tipe destroy 	menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka ada destroy

Penomoran pesan berdasarkan urutan interaksi pesan. Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu.

5. Java

a. Sejarah Java

Pada 1991, sekelompok insinyur Sun dipimpin oleh Patrick Naughton dan James Gosling ingin merancang bahasa komputer untuk perangkat konsumer seperti cable TV Box. Karena perangkat tersebut tidak memiliki banyak memori, bahasa harus berukuran kecil dan mengandung kode yang liat. Juga karena manufaktur – manufaktur berbeda memilih processor yang berbeda pula, maka bahasa harus bebas dari manufaktur manapun. Proyek diberi nama kode ”Green”.

Kebutuhan untuk fleksibilitas, kecil, liat dan kode yang netral terhadap platform mengantar tim mempelajari implementasi Pascal yang pernah dicoba. Niklaus Wirth, pencipta bahasa Pascal telah merancang bahasa portabel yang menghasilkan intermediate code untuk mesin hipotesis. Mesin ini sering disebut dengan mesin maya (virtual machine). Kode ini kemudian dapat digunakan di sembarang mesin yang memiliki interpreter. Proyek Green menggunakan mesin maya untuk mengatasi isu utama tentang netral terhadap arsitektur mesin.

Karena orang – orang di proyek Green berbasis C++ dan bukan Pascal maka kebanyakan sintaks diambil dari C++, serta mengadopsi orientasi objek dan bukan prosedural. Mulanya bahasa yang diciptakan

diberi nama ”Oak” oleh James Gosling yang mendapat inspirasi dari sebuah pohon yang berada pada seberang kantornya, namun dikarenakan nama Oak sendiri merupakan nama bahasa pemrograman yang telah ada sebelumnya, kemudian SUN menggantinya dengan JAVA. Nama JAVA sendiri terinspirasi pada saat mereka sedang menikmati secangkir kopi di sebuah kedai kopi yang kemudian dengan tidak sengaja salah satu dari mereka menyebutkan kata JAVA yang mengandung arti asal bijih kopi. Akhirnya mereka sepakat untuk memberikan nama bahasa pemrograman tersebut dengan nama Java.

Produk pertama proyek Green adalah Star 7 (*7), sebuah kendali jarak jauh yang sangat cerdas. Dikarenakan pasar masih belum tertarik dengan produk konsumer cerdas maka proyek Green harus menemukan pasar lain dari teknologi yang diciptakan. Pada saat yang sama, implementasi WWW dan Internet sedang mengalami perkembangan pesat. Di lain pihak, anggota dari proyek Green juga menyadari bahwa Java dapat digunakan pada pemrograman internet, sehingga penerapan selanjutnya mengarah menjadi teknologi yang berperan di web.

Bahasa/Alat pengembangan	Arsitektur Program			
	Modul Web Server	Scripting Web Server	Modul Web Browser	Scripting Web Browser
Java	servlet	JSP	Applet	Javascript
C++	CGI exe		ActiveX*	
Perl	CGI script			
Phyton	CGI script			
PHP		PHP script		
Visual Basic		ASP*	ActiveX*	VB Script*

*) Hanya di landasan Windows, tidak bisa di Linux.

Gambar 6. Arsitektur Program

Java telah mengakomodasi hampir seluruh fitur penting bahasa – bahasa pemrograman yang ada semenjak perkembangan komputasi modern manusia:

1) Dari SIMULA, bahasa pada tahun 65-an, bahasa yang paling mempengaruhi Java sekaligus C++. Dari bahasa ini diadopsi bentukan – bentukan dasar dari pemrograman berorientasi objek.
2) Dari LISP – bahasa tahun 55-an. Diadopsi fasilitas garbage collection, serta kemampuan untuk meniru generic list processing, meski fasilitas ini jarang yang memanfaatkannya.
3) Dari Algol – bahasa pada tahun 60-an, diambil struktur kendali yang dimilikinya.
4) Dari C++, diadopsi sintaks, sebagian semantiks dan exception handling.
5) Dari bahasa Ada, diambil strongly type, dan exception handling.
6) Dari Objective C, diambil fasilitas interface.
7) Dari bahasa SmallTalk, diambil pendekatan single-root class hiérarchie, dimana objek adalah satu kesatuan hirarki pewarisan.
8) Dari bahasa Eiffel, fasilitas assertion yang mulai diterapkan di sebagian JDK 1.4.

b. Teknologi Java

1) Sebuah Bahasa Pemrograman

Sebagai sebuah bahasa pemrograman, Java dapat membuat seluruh bentuk aplikasi, desktop, web dan lainnya, sebagaimana

dibuat dengan menggunakan bahasa pemrograman konvensional yang lain.

Java adalah bahasa pemrograman yang berorientasi objek (OOP) dan dapat dijalankan pada berbagai platform sistem operasi. Perkembangan Java tidak hanya terfokus oada satu sistem operasi, tetapi dikembangkan untuk berbagai sistem operasi dan bersifat open source.

2) Sebuah *Development* Environtment

Sebagai sebuah peralatan pembangun, teknologi Java menyediakan banyak tools : compiler, interpreter, penyusun dokumentasi, paket kelas dan sebagainya.

3) Sebuah Aplikasi

Aplikasi dengan teknologi Java secara umum adalah aplikasi serba guna yang dapat dijalankan pada seluruh mesin yang memiliki Java Runtime Environment (JRE).

4) Sebuah Deployment *Environtment*

Terdapat dua komponen utama dari Deployment Environment. Yang pertama adalah JRE, yang terdapat pada paket J2SDK, mengandung kelas – kelas untuk semua paket teknologi Java yang meliputi kelas dasar dari Java, komponen GUI dan sebagainya. Komponen yang lain terdapat pada *Web Browser*. Hampir seluruh *Web Browser* komersial menyediakan interpreter dan runtime *environment* dari teknologi Java.

c. Karakteristik Java

Berdasarkan white paper resmi dari SUN, Java memiliki karakteristik berikut:

1) Sederhana (*Simple*)

Bahasa pemrograman Java menggunakan Sintaks mirip dengan C++ namun sintaks pada Java telah banyak diperbaiki terutama menghilangkan penggunaan pointer yang rumit dan *multiple inheritance*. Java juga menggunakan *automatic memory allocation* dan *memory garbage collection*.

2) Berorientasi objek *(Object Oriented)*

Java mengunakan pemrograman berorientasi objek yang membuat program dapat dibuat secara modular dan dapat dipergunakan kembali. Pemrograman berorientasi objek memodelkan dunia nyata kedalam objek dan melakukan interaksi antar objek-objek tersebut.

3) Terdistribusi *(Distributed)*

Java dibuat untuk membuat aplikasi terdistribusi secara mudah dengan adanya libraries networking yang terintegrasi pada Java.

4) Interpreted

Program Java dijalankan menggunakan interpreter yaitu Java Virtual Machine (JVM). Hal ini menyebabkan source code Java

yang telah dikompilasi menjadi Java bytecodes dapat dijalankan pada platform yang berbeda-beda.

5) Robust

Java mempuyai reliabilitas yang tinggi. Compiler pada Java mempunyai kemampuan mendeteksi error secara lebih teliti dibandingkan bahasa pemrograman lain. Java mempunyai runtime-Exception handling untuk membantu mengatasi error pada pemrograman.

6) Secure

Sebagai bahasa pemrograman untuk aplikasi internet dan terdistribusi, Java memiliki beberapa mekanisme keamanan untuk menjaga aplikasi tidak digunakan untuk merusak sistem komputer yang menjalankan aplikasi tersebut.

7) Architecture Neutral

Program Java merupakan platform independent. Program cukup mempunyai satu buah versi yang dapat dijalankan pada platform berbeda dengan Java Virtual Machine.

8) Portable

Program Java dapat dengan mudah dibawa ke platform yang berbeda-beda tanpa harus dikompilasi ulang.

9) Performance

Performance pada Java sering dikatakan kurang tinggi. Namun performance Java dapat ditingkatkan menggunakan kompilasi Java lain seperti buatan *Inprise*, *Microsoft* ataupun *Symantec* yang menggunakan Just *In Time Compilers* (JIT).

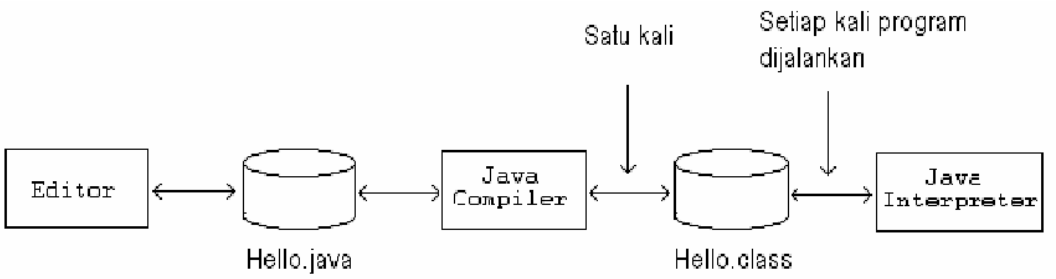
10) Multithreaded

Java mempunyai kemampuan untuk membuat suatu program yang dapat melakukan beberapa pekerjaan secara sekaligus dan simultan.

11) Dynamic

Java didesain untuk dapat dijalankan pada lingkungan yang dinamis. Perubahan pada suatu class dengan menambahkan properties ataupun method dapat dilakukan tanpa menggangu program yang menggunakan class tersebut.

d. Fase – fase Pemrograman Java



Gambar 7. Aliran proses kompilasi dan eksekusi sebuah program Java

Langkah pertama dalam pembuatan sebuah program berbasis Java adalah menuliskan kode program pada text editor. Contoh text

editor yang dapat digunakan antara lain : *notepad, vi, emacs* dan lain sebagainya. Kode program yang dibuat kemudian tersimpan dalam sebuah berkas berekstensi *.java.*

Setelah membuat dan menyimpan kode program, kompilasi file yang berisi kode program tersebut dengan menggunakan *Java Compiler*. Hasil dari adalah berupa berkas *bytecode* dengan ekstensi *.class.*

Berkas yang mengandung *bytecode* tersebut kemudian akan dikonversikan oleh *Java Interpreter* menjadi bahasa mesin sesuai dengan jenis dan platform yang digunakan.

Tabel 5. Pemrosesan Pemrograman Java

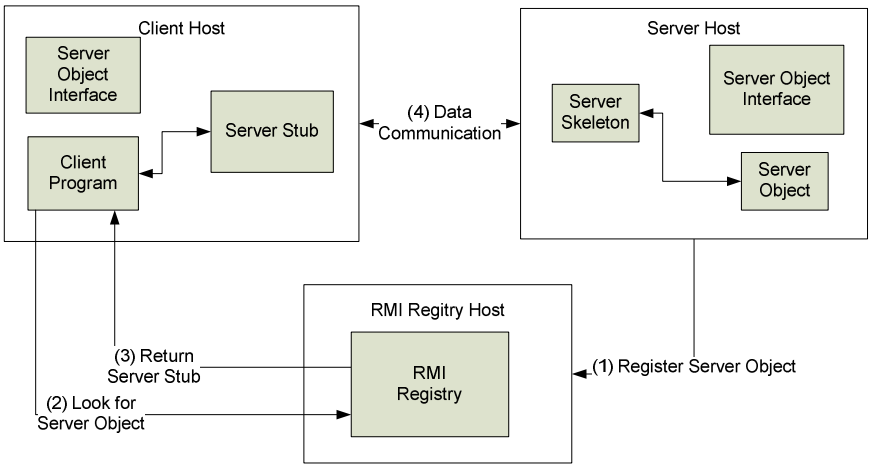
Proses	**Tool**	**Hasil**
Menulis kode program	Text editor	Berkas berekstensi .java
Kompilasi program	Java Compiler	Berkas berekstensi .class (Java Bytecodes)
Menjalankan program	Java Interpreter	Program Output

6. *Remote Method Invocation (RMI)*

Teknologi *Remote Method Invocation* (RMI) memberikan sebuah kerangka untuk membentuk *Java System* terdistribusi. Dengan RMI, sebuah Java object pada satu sistem dapat invoke sebuah method di sebuah *object* pada system lain di jaringan. Sebuah Java System terdistribusi dapat

didefinisikan sebagaimana sebuah koleksi dari kerjasama object terdistribusi pada jaringan (Liang, 2007:1244).

RMI merupakan Objek Terdistribusi Model Java untuk memfasilitasi komunikasi antara object terdistribusi. RMI merupakan sebuah Level Tinggi API built on top dari soket. Pemrograman Level-Soket mengijinkan anda untuk melewatkan data melalui soket antara komputer. RMI tidak hanya sekedar melewatkan data antar object dalam system yang berbeda, tetapi juga meng-invoke method di sebuah remote object. Remote object dapat dimanipulasi sebagaimana terletak pada localhost. Pengiriman data antar mesin yang berbeda ditangani oleh Java Virtual Machine secara transparan (Liang, 2007:1244).



Gambar 8. Diagram RMI

Gambar di atas adalah diagram Java RMI menggunakan sebuah registry untuk penamaan service guna remote object, dan menggunakan stub dan skeleton untuk memfasilitasi komunikasi antara client dan server. RMI bekerja mengikuti urutan:

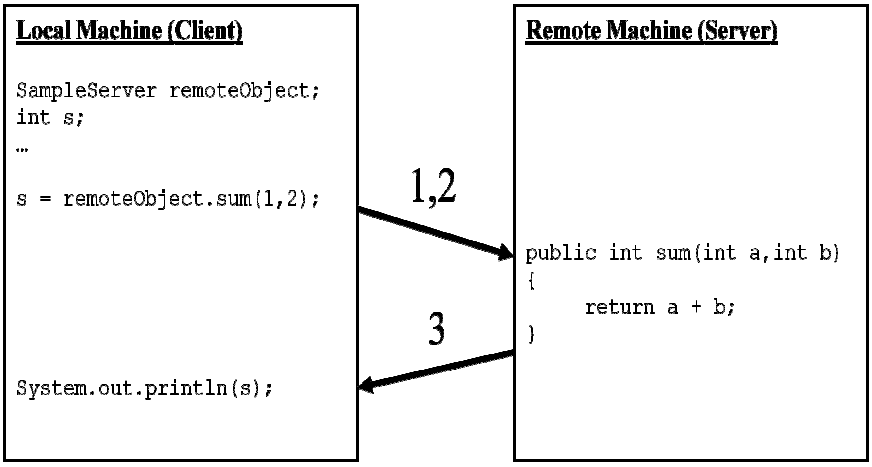
1) Sebuah object server teregistrasi oleh RMI registry.

2) Sebuah client terlihat melewati RMI registry untuk remote object.

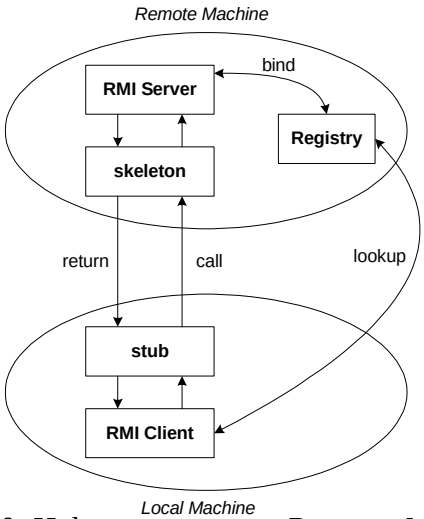
3) Sekali remote object ditempatkan, stub ini dikembalikan pada client.

4) Remote object dapat digunakan di alur yang sama sebagaimana object local. Komunikasi antara client dan server dilewatkan stub dan skeleton.

Java RMI memungkinkan seorang programmer untuk mengeksekusi sebuah fungsi kelas remote dengan semantik yang sama dengan memanggil fungsi local.



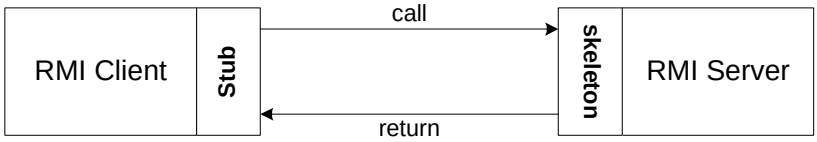
Gambar 9. Hubungan antara *Client – Server*



Gambar 10. Hubungan antara *Remote Machine* dan *Local Machine*

Proses yang terjadi pada gambar 10 di atas adalah hubungan yang terjadi antara *Remote Machine* dan *Local Machine*, dan yang terjadi adalah sebagai berikut:

1) *Server* harus menge-”bind” namanya ke *registry* (*rmiregistry service*).
2) *Client* melihat ke *name server* (*rmiregistry*) untuk mendapatkan referensi objek *remote*.
3) *Stub* melakukan serialisasi parameter ke *skeleton*, dan *skeleton* memanggil *method remote* dan mengserialisasi hasilnya ke *stub*.



Gambar 11. Hubungan antara Stub dan Skeleton

Proses yang terjadi pada gambar 11 di atas adalah hubungan yang terjadi antara Remote Machine dan Local Machine, dan yang terjadi adalah sebagai berikut:

1) *Client* memanggil *method remote*, pemanggilan pertama – tama disampaikan ke stub.
2) *Stub* bertanggung jawab untuk mengirimkan pemanggilan *remote* ke skeleton sisi *server*.
3) *Stub* membuka soket ke *server remote*, melakukan “marshalling” parameter objek dan menyampaikan data stream ke *skeleton*.

4) *Skeleton* berisi *method* untuk menerima pemanggilan remote, melakukan “unmarshalling” paramter dan memanggil implementasi aktual dari objek *remote*.

7. *Usability*

Usability (daya guna) adalah tingkat produk dapat digunakan yang ditetapkan oleh *user* untuk mencapai tujuan secara efektif dan tingkat kepuasan dalam menggunakannya (ISO 9241).

Usability dapat dibagi menjadi empat bagian aspek, yaitu: *learn ability, efficiency of use, error handling,* dan *acceptability (Lethbridge, 2002:245). Learnability* adalah sebuah pengukuran dari kecepatan dari seorang pengguna baru untuk menjadi ahli terhadap system. *Learnability* dapat ditingkatkan dengan dua cara: dengan mengurangi hal – hal yang dipelajari, atau dengan membuat proses pemahaman lebih *intuitive (Lethbridge, 2002:245). Efficiency of use* adalah sebuah pengukuran dari seberapa cepat pengguna yang ahli dapat melakukan tugasnya dengan menggunakan *system*. *Error handling* adalah sebuah antisipasi *system* untuk memberitahu kepada pengguna dalam kondisi *system* yang tidak berjalan normal. *Acceptability* adalah pengukuran terhadap tingkat kesukaan pengguna terhadap system.

Usability dapat diukur tetapi juga sangat subjektif. Ukuran us*ability* berubah dalam kepentingannya. *Usability* sangat bergantung kepada pengguna, yakni: pengguna pemula perlu *learnability,* pengguna yang

jarang memakai perlu *memorability*, dan para ahli perlu efisiensi (Adhikara, 2011). Ukuran *usability* juga tidak dapat disamakan sebagai pemula atau ahli. Pengguna yang ahli dan pemula akan beda persepsi dengan pengalaman – pengalamnya, diantaranya adalah: pengalaman *domain*, pengalaman aplikasi, dan pengalaman *feature.*

Menurut Mandel (1994) yang dikutip oleh Najmuddin (2011) masalah – masalah *usability* yang umum terjadi antara lain :

a. Visualisasi yang buruk.
b. Informasi yang bisa dibaca mengalami kerusakan.
c. Komponen yang tidak dapat dimengerti.
d. Selingan yang mengganggu.
e. Navigasi yang membingungkan.
f. Navigasi yang tidak efisien.
g. Operasi yang tidak efisien.
h. Scrolling yang berlebihan atau yang tidak efisien.
i. Informasi yang berlebihan atau terlalu banyak.
j. Rancangan yang tidak konsisten.
k. Informasi yang tidak ter-update.

B. Penelitian yang Relevan

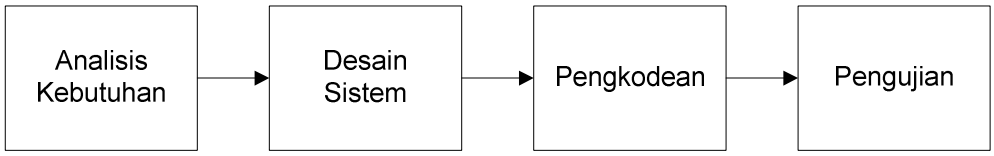
Penelitian tentang *Remote and Monitoring* diantaranya adalah Irwan Pribadi dan Mukhammad Andri Setiawan dengan judul Manajemen Pengelolaan *LAN* dengan *Remote System Application (2005),* yang diteliti adalah *Remote System Application* pada jaringan *Local Area Network (LAN)* menggunakan *software development* Visual Basic.

Peneliti lain adalah You, Xiang-bai Liu, Yi-min Xu, Wang-ming dengan judul *The Design of a Remote Monitoring System based On Java(2010).* Penelitian ini mengkhususkan *Remote and Monitoring* dalam jaringan internet dan *Code Division Multiple Access (CDMA).*

Peneliti lain adalah Yunus Kurniawan dengan judul Pembangunan Aplikasi *Remote Task Manager* pada Jaringan Komputer Berbasis Windows(2010). Penelitian ini menggunakan bahasa pemrograman Microsoft C# dalam lingkungan minimal NET Framework 2.0,. Hasil dari penelitian ini adalah sebuah aplikasi yang diinstal disisi server dan aplikasi untuk mengontrol *client* yang diletakkan disisi client.

C. Kerangka Berpikir

Perangkat lunak aplikasi yang akan dikembangkan dalam penelitian ini adalah aplikasi *Remote and Monitoring* berbasis Java *Remote Method Invocation.* Aplikasi dirancang untuk mengawasi dan mengendalikan komputer *User Workstation* oleh seorang *Network Admin.* Tanpa perlu mendatangi komputer *User Workstation,* seorang *Network Admin* dapat mengendalikan dan mengawasi komputer *User Workstation.*



Gambar 12. Bagan kerangka berpikir

Pengembangan Aplikasi *Remote and Monitoring* berbasis Java RMI dalam penelitian ini menggunakan metode pengembangan perangkal lunak model *waterfall.* Tahap pengembangan aplikasi meliputi: (1). Analsis Kebutuhan, (2). Desain Sistem, (3). Pengkodean, (4). Pengujian.

Proses pengujian aplikasi ini melalui tahap verifikasi dan validasi. Verifikasi adalah proses mengevaluasi *software* untuk menentukan apakah

produk telah sesuai dengan tahap pengembangan mulai dari fase awal. Verifikasi berbasis proses, menjawab pertanyaan "*Have we built the software right?"*. Validasi adalah proses mengevaluasi *software* selama atau pada akhir fase pengembangan untuk membandingkan apakah *software* telah sesuai dengan *requirement*. Validasi berbasis produk, menjawab pertanyaan "*Have we built the right software?"*

BAB III

METODE PENELITIAN

A. Desain Penelitian

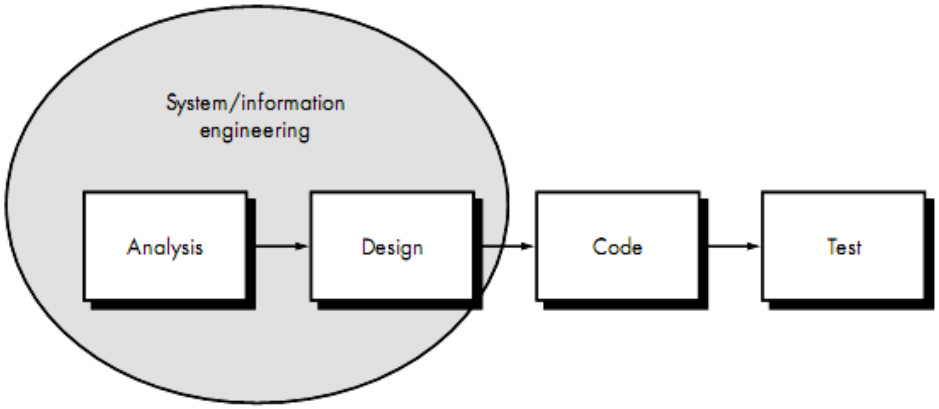
1. Metode Penelitian

Metode yang digunakan dalam penelitian ini menggunakan pendekatan penelitian pengembangan *(Research and Development).* Metode penelitian dan pengembangan adalah metode penelitian yang digunakan untuk menghasilkan produk tertentu, dan menguji keefektifan produk tersebut (Sugiyono, 2009:407).

"… Tetapi perlu diketahui bahwa tidak setiap penelitian harus merumuskan hipotesis. Penelitian yang bersifat eksploratif dan deskriptif sering tidak perlu merumuskan hipotesis." (Sugiyono, 2009:96).

"Penelitian deskriptif ini hanya berusaha menggambarkan secara jelas dan sekuensial terhadap pertanyaan penelitian yang telah ditentukan sebelum para peneliti terjun ke lapangan dan mereka tidak menggunakan hipotesis sebagai petunjuk arah atau *guide* dalam penelitian." (Sukardi, 2004:14).

Peneliti membangun perangkat lunak ini menggunakan metode pengembangan perangkat lunak berorientasi objek dipadukan dengan model proses *waterfall*. Model proses *waterfall* adalah model klasik yang bersifat sistematis, berurutan dalam membangun suatu software. Fase-fase dalam ***Waterfall Model*** (Pressman, 2001 : 29):



Gambar 13. *Waterfall* menurut *Pressman*

a. Analisis kebutuhan *(Analysis)*

Analisis kebutuhan merupakan tahap awal dari sebuah tahapan rekayasa perangkat lunak. Peneliti menganalisis kebutuhan fungsional, *interface* perangkat lunak, dan membangun batasan yang harus dipenuhi oleh suatu perangkat lunak. Berdasarkan analisis kebutuhan diharapkan perangkat lunak yang dibangun dapat memenuhi kebutuhan pengguna.

Peneliti melakukan observasi terhadap aplikasi – aplikasi yang sudah ada dan studi literatur guna mendapatkan spesifikasi dan kebutuhan – kebutuhan secara teknis untuk membangun perangkat lunak *Remote and Monitoring.*

b. Desain *(design)*

Peneliti merancang sistem berdasarkan analisis kebutuhan yang sudah dilakukan pada tahap pertama. Pemodelan desain sistem dari perangkat lunak dapat menggunakan beberapa macam jenis pemodelan, salah satunya adalah dengan UML.

c. Pengkodean *(Code)*

Implementasi pengkodean dilakukan berdasarkan analisis kebutuhan dan desain yang sudah dilakukan sebelumnya. Peneliti menguji dari tiap unit program secara langsung pada saat pengkodean. Pengujian pada tahap ini disebut *white-box.*

d. Pengujian *(Test)*

Pengujian perangkat lunak adalah elemen kritis dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari spesifikasi, desain, dan pengkodean (Pressman, 2002:525). Pengujian perangkat lunak meliputi, *alpha,* dan *beta.*

1) *Alpha Testing*

Uji *alpha* meliputi uji *white-box dan black-box.* Pengujian *alpha* dilakukan bersama ahli rekayasa perangkat lunak guna menguji unjuk kerja dan menganalisa alur kerja perangkat lunak. Berdasarkan pengujian *alpha* ini kemudian didapatkan revisi sesuai masukan ahli.

(a) *White-box Testing*

Peneliti melakukan pengujian *white-box* dengan menganalisa algoritma dan menjalankan setiap perintah pada bahasa pemrograman. Pengujian ini dilakukan saat pengkodean berlangsung, sehingga *method* yang dibuat langsung diuji dan langsung diimplementasikan.

(b)*Black-box Testing*

Peneliti menguji semua persyaratan fungsional perangkat lunak yang dibangun pada pengujian *black-box.* Berdasarkan pengujian *black-box* diharapakan dapat mengantisipasi kesalahan fungsi, kesalahan antarmuka, kesalahan struktur data, kesalahan kinerja, dan kesalahan inisialisasi.

2) *Beta Testing*

Tahapan lanjut setelah pengujian *alpha* adalah uji *beta.* Uji *beta* dilakukan terhadap kelompok tertentu yang dianggap mengetahui seluk beluk perangkat lunak yang dibangun. Uji *beta* dilakukan guna mendapatkan masukan evaluasi teknis terhada perangkat lunak yang telah dibangun. Uji *beta* ditindaklanjuti dengan revisi akhir. Berdasarkan tahapan rekayasa perangkat lunak maka tahapan pengembangan perangkat lunak selesai setelah tahap uji *beta.*

2. Obyek Penelitian

Obyek yang diteliti pada penelitian ini adalah Aplikasi *Remote and Monitoring* berbasis *Java Remote Method Invocation.*

3. Tempat dan Waktu Penelitian

Penelitian dilaksanakan di Laboratorium Digital Networks and Multimedia Pusat Komputer Universitas Negeri Yogyakarta. Ada pun pelaksanaan dimulai bulan November 2010 sampai selesai.

4. Sampel Penelitian

Teknik *Sampling* menggunakan *Nonprobability Sampling*. *Nonprobability Sampling* adalah teknik pengambilan sampel yang tidak memberi peluang/kesempatan sama bagi setiap unsur atau anggota populasi untuk dipilih menjadi sampel (Sugiyono, 2010:122). Teknik Sampling yang akan penulis tindak lanjuti adalah *Sampling Purposive*. *Sampling purposive* adalah teknik penentuan sampel dengan pertimbangan tertentu (Sugiyono, 2010:124). Misalnya sumber data adalah orang yang ahli dibidangnya. Sampel ini lebih cocok digunakan untuk penelitian kualitatif atau penelitian-penelitian yang tidak melakukan generalisasi. Ukuran sampel yang dipakai peneliti adalah sebanyak 10 orang. Sampel yang digunakan peneliti adalah sebagai berikut:

a. Ahli rekayasa perangkat lunak

Peneliti memilih satu orang ahli rekayasa perangkat lunak untuk pengujian *alpha.* Pengujian *alpha* dilakukan guna mengetahui unjuk kerja dari aplikasi yang telah dibangun.

b. Pengguna khusus

Peneliti memilih sembilan orang pengguna khusus yang dianggap mengerti dan memahami dari aplikasi yang telah dibangun. Peneliti memilih *Focus Group Discussion Digital Networks and Multimedia* Pusat Komputer UNY untuk pengujian *beta* guna mendapatkan evaluasi – evaluasi secara teknis untuk pengembangan perangkat lunak *Remote and Monitoring* berbasis Java RMI.

5. Alat dan Bahan Penelitian

Fasilitas atau perangkat pendukung yang digunakan dalam penelitian ini yaitu:

a. Perangkat Komputer

Satu buah perangkat notebook Intel Pentium Dual Core dengan prosesor 2,16 Ghz, memori DDR2 2 GB, Hardisk 120 GB, Soundcard card, VGA, DVD/CD-RW, Keyboard, Mouse.

b. Printer

Printer yang digunakan adalah HP Deskjet 2466. Printer ini digunakan untuk mencetak data berupa tulisan/teks, gambar dan laporan.

c. Jaringan Internet

Jaringan internet digunakan untuk mengakses internet.

d. Perangkat Lunak

Proses pengembangan aplikasi *Remote and Monitoring* menggunakan beberapa perangkat lunak. Perangkat lunak tersebut adalah *NetBeans IDE 6.9.1., Java(TM) SE Runtime Environment (build 1.6.0_17-b04), Altova Umodel 2011* serta program perangkat lunak pendukung lainnya

B. Pengembangan Perangkat Lunak

1. Analisis Kebutuhan

Peneliti menganalisis kebutuhan dengan dua macam analisis, yaitu mencari informasi dengan observasi terhadap aplikasi – aplikasi *remote*

system yang sudah ada dan mencari informasi dengan studi literatur dari berbagai sumber media cetak maupun elektronik.

a. Observasi

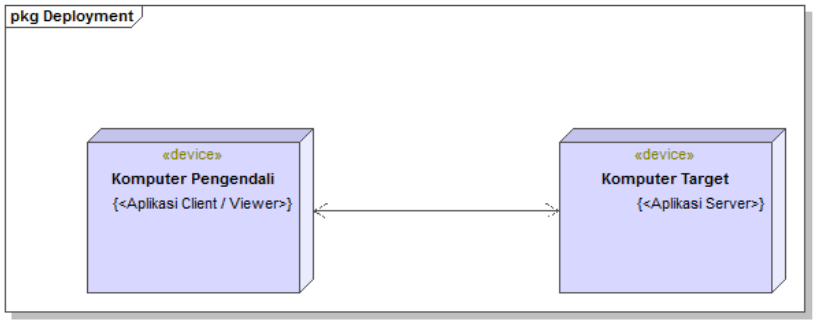
Peneliti melakukan observasi langsung terhadap tiga aplikasi *remote system* yang sudah ada. Tiga aplikasi tersebut adalah *RealVNC Remote Control*, *NetSupport School*, dan *RAdmin*. Peneliti mendapatkan kesimpulan bahwa:

1) Komputer target *(Server)* dan komputer pengendali (*Viewer* atau *Client* atau *admin*) harus terhubung dengan metode jaringan komputer dan bisa berbagi sumber *(Sharing).*
2) Tiap aplikasi *remote* harus menyediakan aplikasi untuk komputer target (Server) dan aplikasi untuk komputer pe-*remote (Viewer* atau *Client).*
3) Komputer target yang sudah di pasang aplikasi *Server* harus di *start* dulu atau di-*running,* kemudian baru komputer target bisa diawasi dan dikendalikan.
4) Komputer *Viewer* atau *client* yang sudah dipasang aplikasi *Viewer* atau *client,* untuk bisa mengendalikan komputer target harus tahu alamat komputer target, baru kemudian dikoneksikan ke komputer target.

b. Studi Literatur

Peneliti melakukan analisisa kebutuhan tidak hanya melalui observasi, tetapi juga dengan metode studi pustaka. Peneliti melakukan

studi pustaka terhadap konsep – konsep dasar sebuah *remote.* Dan dalam studi pustaka tersebut peneliti mendapatkan spesifikasi minimal dari sebuah aktifitas *remote.*



Gambar 14. Arsitektur Pengendalian jarak jauh

Gambar di atas menunjukkan hubungan antara komputer target (Aplikasi *Server)* dan komputer pengendali (Aplikasi *Client / Viewer*). Dan dari gambar tersebut peneliti menyimpulkan bahwa:

1) Aktifitas *remote* minimal harus terdiri dari dua objek.
2) Hubungan yang terjadi antara objek satu dengan objek yang lainnya adalah saling keterkaitan.
3) Jika hubungan terputus antara objek satu dengan objek lainnya maka proses *remote* tidak dapat terjadi.

Kesimpulan dari observasi dan studi literatur adalah dibuatnya spesifikasi/kebutuhan fungsional. Dengan demikian, secara ringkas kebutuhan fungsional yang harus dibuat dalam implementasi Aplikasi *remote and monitoring* adalah:

1) Aplikasi memiliki fitur *login* untuk membedakan hak akses antar *Network Admin* dan *User Workstation.*

2) Jika pengguna aplikasi login sebagai admin*(Network Admin)* maka:
   a) Pengguna dapat memulai *Remote Server*.
   b) Pengguna dapat mengakhiri *Remote Server.*
   c) Pengguna dapat mengkonfigurasi *Remote Server.*
   d) Pengguna dapat mengganti *password.*
   e) Pengguna dapat memulai *Chat Server.*
   f) Pengguna dapat mengakhiri *Chat Server.*
   g) Pengguna dapat mengirim pesan ke semua *Client Chat.*
   h) Pengguna dapat menerima pesan dari *Client Chat.*
   i) Pengguna dapat mengawasi dan mengendalikan komputer *User Workstation* yang sudah dipasang aplikasi *Remote and Monitoring.*
   j) Pengguna dapat keluar dari aplikasi *Remote and Monitoring*.
3) Jika pengguna aplikasi *login* sebagai *user(User Workstation)* maka:
   a) Pengguna dapat memulai *Remote Server.*
   b) Pengguna dapat menghubungkan *Chat Client* ke *Chat Server.*
   c) Pengguna dapat mengakhiri koneksi *Chat Client* ke *Chat Server.*
   d) Pengguna dapat mengirim pesan ke *Chat Server (Network Admin).*
   e) Pengguna dapat menerima pesan dari semua *Chat Client* dan *Network Admin.*
   f) Pengguna tidak dapat keluar dari aplikasi *Remote and Monitoring*.

Peneliti membuat instrumen untuk pengujian *Alpha* berdasarkan pengembangan dari kebutuhan fungsional. Kebutuhan fungsional harus terpenuhi sehingga aplikasi *Remote and Monitoring* dapat dikatakan mempunyai unjuk kerja yang baik.

Selain itu ada pula kebutuhan non-fungsional yang harus terpenuhi, antara lain membuat sistem interaksi (*user interface)* yang menarik dan *friendly*, serta membuat sistem navigasi aplikasi *remote and monitoring* yang mudah dipahami.

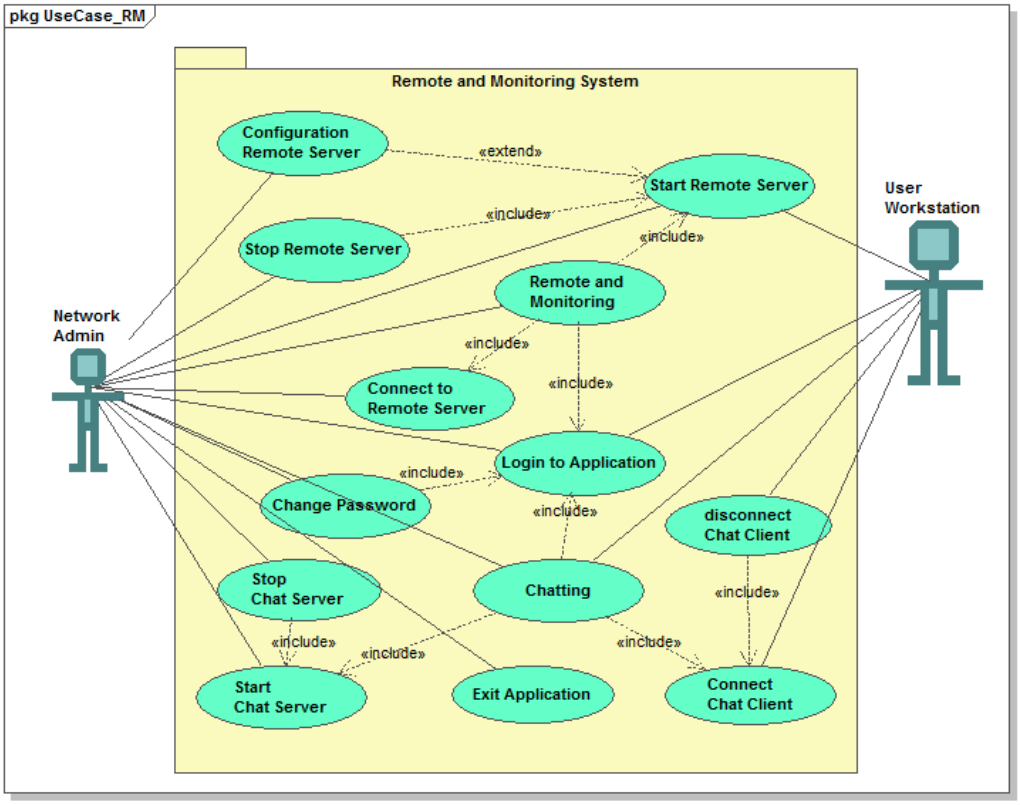
2. Desain Sistem dan *Software*

Dari analisis kebutuhan yang sudah ada, maka tahap selanjutnya adalah desain sistem. Informasi – informasi yang ada pada analisis kebutuhan dimodelkan dengan metode UML.

a. *Use Case Diagram*

Analisis kebutuhan yang sudah dilakukan akan dimodelkan dalam *use case* diagram. Kebutuhan fungsional yang ada dari analisis kebutuhan dimasukkan dalam *case – case* tersendiri. Peneliti membuat *use case* utama menjadi tiga belas *use case.* Pemeran atau *actor* dalam hal ini adalah *Network Admin* dan *User Workstation*.

Gambar di bawah ini menunjukkan hubungan antar *use case* dengan *actor*. *User (User Workstation)* mempunyai hak akses yang terbatas. *Admin(Network Admin)* mempunyai hak akses penuh atas segala aktifitas yang tidak dimiliki oleh *user. User* hanya mengoperasikan komputer di bawah pengawasan dan kendali *Admin*.



Gambar 15. *Use case diagram*

Tabel 6. *Use case description* dari *Login to Application*

Use case name	*Login to application*
Primary actor	*Network Admin*
Supporting actor(s)	*Aplikasi Remote and Monitoring Viewer.*
Summary	*Netwok Admin login* ke *Aplikasi Remote and Monitoring Viewer* dengan memasukkan *password* dari *username admin* kemudian klik tombol *login.* Aplikasi mengecek kebenaran dari *password* yang telah dimasukkan.
Pre-condition	*Network Admin* memiliki *password* dari *username admin.* *Aplikasi Remote and Monitoring Viewer* beroperasi.
Normal flow of event	1. *Network Admin* mengeksekusi Aplikasi. 2. Aplikasi menampilkan *Login Dialog*. 3. *Network Admin* memasukkan *password* dari *username admin*. 4. Klik tombol *login*. 5. Aplikasi mengecek kebenaran *password*. 6. Aplikasi *Remote and Monitoring Viewer* terbuka. 7. Aplikasi *Chat Server* terbuka.
Extensions	3a. *Network Admin* tidak memasukkan *password.* 3b. *Network Admin* salah memasukkan *password.*
Post-condition	1. Aplikasi *Remote and Monitoring Viewer* terbuka. 2. Aplikasi *Chat Server* terbuka.

Tabel 7. *Use case description* dari *Change Password*

Use case name	*Change Password*
Primary actor	*Network Admin*
Supporting actor(s)	*Aplikasi Remote and Monitoring Viewer*
Summary	*Netwok Admin* mengganti *password login* Aplikasi *Remote and Monitoring Viewer* dengan memasukkan *password* baru kemudian klik tombol change password. Aplikasi kemudian menyimpan password yang baru.
Pre-condition	*Network Admin* telah *login*. Aplikasi *Remote and Monitoring Viewer* sudah terbuka.
Normal flow of event	1. *Network Admin* memilih tombol *"Change Password"*. 2. Aplikasi menampilkan jendela *"Change Password"*. 3. *Network Admin* memasukkan *password* yang baru. 4. Klik *Change Password.* 5. Aplikasi menyimpan *password* yang baru
Extensions	-
Post-condition	1. Aplikasi *Remote and Monitoring Viewer* menyimpan *password* yang baru.

Tabel 8. *Use case description* dari *Start Remote Server*

Use case name	*Start Remote Server*
Primary actor	*User Workstation*
Supporting actor(s)	Aplikasi *Remote and Monitoring Server*
Summary	*User Workstation login* ke *Aplikasi Remote and Monitoring Server* dengan memilih *username user* tanpa memasukkan password kemudian klik tombol *login.* Aplikasi mengecek *username. Remote Server* secara otomatis mulai *listening port.*
Pre-condition	*User workstation* sudah *login Remote and Monitoring*. Aplikasi *Remote and Monitoring Server* beroperasi.
Normal flow of event	1. *User Workstation* mengeksekusi Aplikasi. 2. Aplikasi menampilkan *Login Dialog*. 3. *User Workstation* memilih *username user*. 4. Klik tombol *login*. 5. Aplikasi mengecek *username*. 6. Aplikasi *Remote and Monitoring Server* terbuka. 7. Remote Server status *"Running … at 127.0.0.1:6666"* 8. Jendela *Chat Client* terbuka.
Extensions	-
Post-condition	1. Aplikasi *Remote and Monitoring Server* telah listening port dengan status "Running … at 127.0.0.1:6666" 2. Jendela *Chat Client terbuka.*

Tabel 9. *Use case description* dari *Stop Remote Server*

Use case name	*Stop Remote Server*
Primary actor	*Network Admin*
Supporting actor(s)	Aplikasi *Remote and Monitoring Server*
Summary	*Netwok Admin* mengakhiri *listeing port* dari aplikasi *Remote and Monitoring Server* pada komputer *User Workstation* dengan memasukkan *password* setelah memilih tombol *"Exit"*.
Pre-condition	*Network Admin* sudah *login Remote and Monitoring*. *Network Admin* mempunyai *password* dari *username admin.*
Normal flow of event	1. *Network Admin* memilih tombol *"Exit"*. 2. Aplikasi menampilkan jendela autentifikasi untuk keluar. 3. *Network Admin* diminta untuk memasukkan *password* dari *username admin* untuk keluar dari aplikasi. 4. Klik *"Exit"*. 5. Aplikasi *Remote and Monitoring Server* mengakhiri listening port. 6. Jendela *Chat Client* tertutup. 7. Aplikasi *Remote and Monitoring Server* tertutup.
Extensions	3a. *Network Admin* tidak memasukkan *password.* 3b. *Network Admin* salah memasukkan *password.*
Post-condition	1. Jendela Chat Client tertutup. 2. Aplikasi *Remote and Monitoring Server* tertutup.

Tabel 10. *Use case description* dari *Configuration Remote Server*

Use case name	*Configuration Remote Server*
Primary actor	*Network Admin*
Supporting actor(s)	Aplikasi *Remote and Monitoring Server*
Summary	*Netwok Admin* mengganti konfigurasi dari *Remote and Monitoring Server* pada komputer *user workstation* dengan memilih tombol "Configuration". Network Admin memilih *IP Address untuk Remote Server* dan *port.*
Pre-condition	Aplikasi *Remote and Monitoring Server* terbuka. Pengguna sudah *login Remote and Monitoring*.
Normal flow of event	1. *Network Admin* memilih tombol *"Configuration"*. 2. Aplikasi menampilkan jendela *"Server Configuration"*. 3. *Network Admin* memilih *IP Address* dan *port* untuk *Remote Server*. 4. Klik *OK.* 5. Aplikasi menyimpan *konfigurasi untuk Remote Server.*
Extensions	-
Post-condition	1. Aplikasi *Remote and Monitoring Server* menyimpan konfigurasi yang baru.

Tabel 11. *Use case description* dari *Connect to Remote Server*

Use case name	*Connect to Remote Server*
Primary actor	*Network Admin*
Supporting actor(s)	Aplikasi *Remote and Monitoring Viewer*
Summary	*Netwok Admin* memilih koneksi ke komputer *User Workstation* dengan memasukkan *IP address* dan *port Remote Server.*
Pre-condition	*Network Admin* sudah *login* ke aplikasi *Remote and Monitoring.* Aplikasi *Remote and Monitoring Viewer* sudah terbuka .
Normal flow of event	1. *Network Admin* memilih tombol *Connect to Server.* 2. Aplikasi meminta masukan *IP Address* dan *port Remote Server.* *3. Network Admin* memasukkan *IP Address* dan *Port Remote Server.* 4. Aplikasi *Remote and Monitoring Viewer* menghubungkan koneksi ke aplikasi *Remote and Monitoring Server* yang ada di komputer User Workstation. 5. Aplikasi *Remote and Monitoring Viewer* memunculkan jendela tampilan computer Remote Server
Extensions	3a. Network Admin memasukkan *IP Address* dan *port* yang salah. 4a. *Remote and Monitoring Server* belum dijalankan. 5a. Aplikasi *Remote and Monitoring Viewer* tidak memunculkan jendela tampilan computer Remote Server.
Post-condition	1. *Remote and Monitoring Viewer* terhubung ke Remote Server 2. Muncul jendela tampilan komputer *Remote and Monitoring Server*

Tabel 12. *Use case description* dari *Start Chat Server*

Use case name	*Start Chat Server*
Primary actor	*Network Admin*
Supporting actor(s)	*Chat Server*
Summary	*Netwok Admin* dapat menerima koneksi dan pesan *Chatting* dari *Chat Client* dengan memilih tombol *Start Service.*
Pre-condition	*Network Admin* sudah *login Remote and Monitoring.* Chat Server sudah terbuka
Normal flow of event	1. *Network Admin* memilih tombol *Start Service.* 2. Notifikasi bahwa *Chat Server* sudah dapat menerima koneksi dari *Chat Client.* 3. *Chat Server* menunggu koneksi dari *Chat Client*
Extensions	-
Post-condition	1. *Chat Server* siap menerima koneksi dari *Chat Client.* 2. Muncul notifikasi *"Ready to accept Chat Clients"*

Tabel 13. *Use case description* dari *Remote and Monitoring*

Use case name	*Remote and Monitoring*
Primary actor	*Network Admin*
Supporting actor(s)	Aplikasi *Remote and Monitoring Viewer*
Summary	*Netwok Admin* mengawasi dan mengendalikan *Remote and Monitoring Server* dari jendela *Remote and Monitoring Viewer*. *Network Admin* dapat memasukkan input dari *keyboard* ataupun *mouse* dari *Remote and Monitoring Viewer* untuk mengendalikan komputer *Remote and Monitoring Server.*
Pre-condition	Pengguna sudah *login Remote and Monitoring* sebagai admin. Aplikasi *Remote and Monitoring Viewer* sudah terhubung ke *Remote and Monitoring Server.* Jendela tampilan *desktop* komputer *User Workstation* sudah muncul.
Normal flow of event	1. *Network Admin* mengawasi *Remote and Monitoring Server* dari *Remote and Monitoring Viewer*. 2. *Network Admin* memberikan masukan melalui *keyboard* ataupun *mouse.* 3. Aplikasi *Remote and Monitoring Viewer* mengirimkan masukan ke aplikasi *Remote and Monitoring Server* yang berada di komputer User Workstation. 4. Aplikasi *Remote and Monitoring Server* mengolah data dari *Remote and Monitoring Viewer* untuk dieksekusi di komputer *User Workstation.*
Extensions	1a. *Remote and Monitoring Server* berhenti merespon. 4a. *Remote and Monitoring Server* berhenti merespon.
Post-condition	1. *Remote and Monitoring Server* menerima masukan dari *Remote and Monitoring Viewer.* 2. Komputer *User Workstation* mengikuti kendali dari *Network Admin* melalui aplikasi *Remote and Monitoring Viewer.*

Tabel 14. *Use case description* dari *disconnect Chat Client*

Use case name	*Disconnect Chat Client*
Primary actor	*User Workstation*
Supporting actor(s)	*Chat Client*
Summary	*User Workstation* mengakhiri koneksi Chat Client dari Chat Server dengan memilih tombol *disconnect.*
Pre-condition	*User Workstation* sudah login *Remote and Monitoring.* *Chat Client* sudah terbuka. *Chat Client* terhubung dengan *Chat Server.*
Normal flow of event	1. *User Workstation* memilih tombol *disconnect.* 2. *Chat Client* mengirim data pemutusan koneksi ke *Chat Server.* 3. *Chat Client* mengakhiri koneksi dari *Chat Server.*
Extensions	-
Post-condition	1. Chat Client sudah terputus koneksi dari chat server 2. Chat Server memberikan notifikasi

Tabel 15. *Use case description* dari *Connect Chat Client*

Use case name	*Connect Chat Client*
Primary actor	*User Workstation*
Supporting actor(s)	*Chat Client*
Summary	*User Workstation* memasukkan *IP Address server* dan *username* untuk bisa terhubung ke *Chat Server*. *Chat server* akan mengecek ketersediaan *username* yang dimasukkan *User Workstation*.
Pre-condition	*User Workstation* sudah login *Remote and Monitoring.* *Chat Client* sudah terbuka. *Chat Server* sudah aktif atau dalam kondisi *listening port.*
Normal flow of event	1. *User Workstation* memasukkan *IP Address Server* dan *username.* 2. Klik connect. 3. *Chat Server* mengecek ketersediaan username dari *user workstation*.
Extensions	1a. *User Workstation* salah memasukkan *IP Address server.* 3a. *Username* yang dimasukkan sudah ada yang menggunakan.
Post-condition	1. Chat Client sudah terhubung dengan *Chat Server.* 2. Chat Client menerima notifikasi dari *Chat Server.*

Tabel 16. *Use case description* dari *Chatting*

Use case name	*Chatting*
Primary actor	*Network Admin, User workstation*
Supporting actor(s)	*Chat Server, Chat Client*
Summary	*Network Admin* dan *User Workstation* dapat mengirim dan menerima pesan.
Pre-condition	*Chat Client* sudah terhubung dengan *Chat Server.*
Normal flow of event	1. *User Workstation* mengirim pesan kepada *Chat Server.* 2. *Chat Client* mengirimkan data pesan ke *Chat Server*. 3. *Chat Server* menerima data pesan dan menampilkan di layar komputer *User Workstation.* 4. *Network Admin* membalas pesan dari *User Workstation.* 5. *Chat Server* mengirimkan data pesan ke *Chat Client.* 6. *Chat Client* menerima data pesan dan menampilkan di layar komputer *Network Admin.*
Extensions	-
Post-condition	1. Pesan diterima oleh *User Workstation* dari *Chat Server.* 2. Pesan diterima oleh *Chat Server* dari *User Workstation.*

Tabel 17. *Use case description* dari *Stop Chat Server*

Use case name	*Stop Chat Server*
Primary actor	*Network Admin*
Supporting actor(s)	*Chat Server*
Summary	*Network Admin* mengakhiri *Chatting* dengan memilih tombol *"Stop Service"*, semua *online Chat Client* akan terputus.
Pre-condition	*Chat Server* dalam kondisi *"Start Service"*
Normal flow of event	1. *Network Admin* memilih tombol *"Stop Service"* atau memilih tombol *"X" (Close).* 2. *Chat Server* akan menghentikan proses *listening port.* 3. *Chat Server* memberikan notifikasi kepada *Chat Client* yang terhubung dengan *"Server is Shuttingdown"*
Extensions	-
Post-condition	1. Semua *Chat Client* yang terhubung ke *Chat Server* akan mendapatkan notifikasi bahwa *"Server is shuttingdown"*. 2. Semua *Chat Client* yang terhubung ke *Chat Server* akan terputus koneksi.

Tabel 18. *Use case description* dari *Exit Application*

Use case name	*Exit Application*
Primary actor	*Network Admin*
Supporting actor(s)	Aplikasi *Remote and Monitoring Viewer*
Summary	*Network Admin* menutup aplikasi *Remote and Monitoring viewer* dengan memilih tombol *"Exit"*, semua aplikasi yang termasuk di dalamnya akan berhenti beroperasi dan tertutup.
Pre-condition	Aplikasi *Remote and Monitoring Viewer* terbuka. Chat Server dalam kondisi *"Server Closed"*.
Normal flow of event	1. Network admin memilih tombol "Exit". 2. Aplikasi Remote and Monitoring akan memberikan konfirmasi. 3. Network admin memilih "OK". 4. *Chat Server* tertutup. 5. Aplikasi *Remote and Monitoring Viewer.*
Extensions	-
Post-condition	1. *Chat Server* tertutup. 2. Aplikasi *Remote and Monitoring Viewer tertutup.*

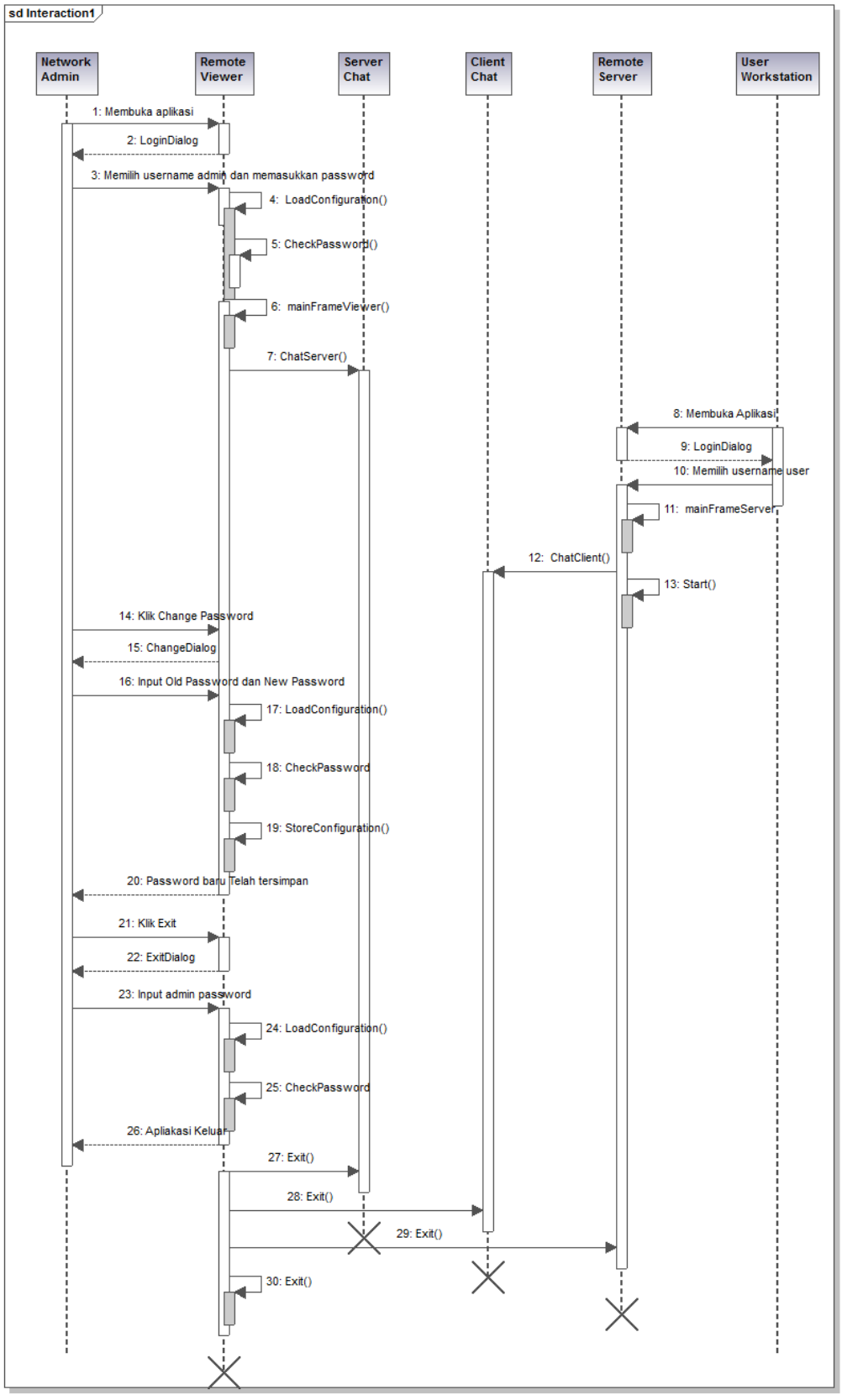
b. *Sequence Diagram*

Sequence diagram adalah tahapan selanjutnya setelah *use case diagram.* Di sini objek – objek di modelkan dalam keterkaitannya. Di sini ada enam objek, yaitu: *Network Admin, Remote Server, Remote Viewer, Chat Server, Chat Client,* dan *User Workstation*. Pemodelan

dalam *sequence diagram* disesuiakan dengan alur kerja sistem pada *use case diagram*. Peneliti membuat tiga jenis *sequence diagram*, yakni: *sequence diagram* bagian otentikasi*, sequence diagram* bagian *remote and monitoring*, dan *sequence diagram* bagian *chatting.*

Sequence diagram bagian otentikasi merupakan bagian awal pada aplikasi *Remote and Monitoring. Sequence diagram* bagian otentikasi ini meliputi *login to application*, *change password*, dan *exit application*. Pemodelan dari bagian otentikasi ini secara berurutan, sehingga dengan membuat *sequence diagram* ini dapat mengetahui bagaimana alur kerja sistem yang akan dibuat nanti.

Gambar 16, menunjukan alur kerja sistem bagian otentikasi. Pengguna dapat menggunakan aplikasi dengan diawali menjalankan atau mengeksekusi aplikasi. Aplikasi akan memunculkan *login dialog* yang kemudian pengguna harus memilih *username* dan memasukkan *password.* Berdasarkan *username* dan *password* yang telah dimasukkan maka program akan mengecek kebenarannya dengan *method CheckPassword(). Method – method* inilah yang nantinya akan dikembangkan dan dikodekan ke dalam bahasa pemrograman. Inti dari bagian ontentikasi ini adalah untuk memberikan kewenangan yang berbeda antara “user” dan “admin”. Aplikasi dapat diakhiri hanya oleh “admin” dengan memasukkan *password* yang sama seperti saat *login.* Kelas dan *method* yang nantinya dibangun merupakan pengembangan dari *sequence diagram.*

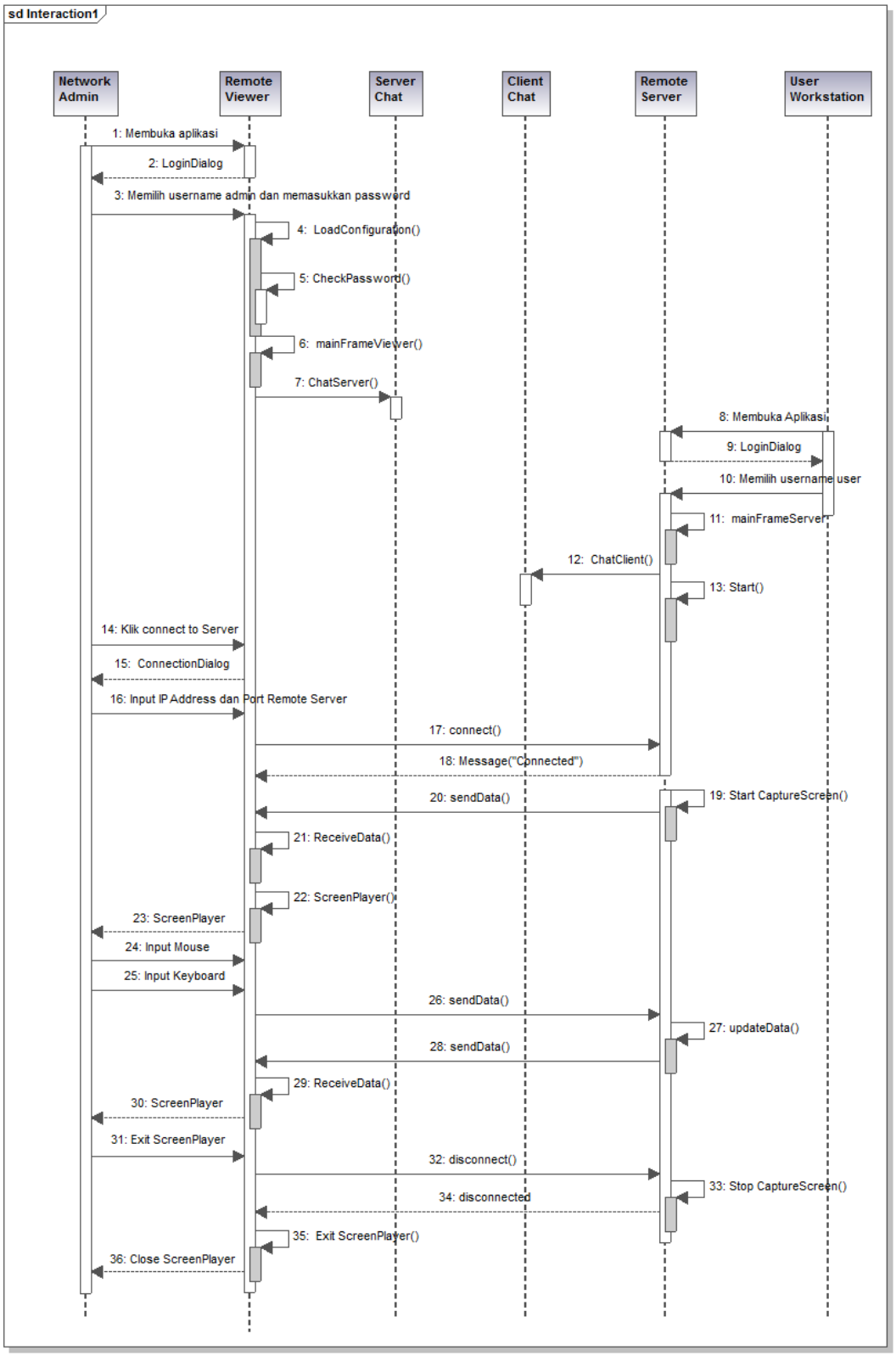


Gambar 16. *Sequence Diagram* Otentikasi

Gambar 17 menunjukan *sequence diagram* bagian *remote and monitoring*. Pengguna dapat melakukan aktifitas *remote and monitoring* harus dengan melalui tahap otentikasi yaitu *login to application*. *Network Admin* maupun *User Workstatillon* harus melalui proses *login to application* untuk dapat melakukan aktifitas *remote and monitoring.*

Aktifitas *remote and monitoring* ini adalah pengawasan dan pengendalian komputer *User Workstation* oleh *Network Admin*. *User Workstation* pasif dalam aktifitas *remote and monitoring. Sequence diagram* bagian *remote and monitoring* ini menunjukan urutan kerja sistem pada saat aktifitas pengendalian dan pengawasan komputer *User Workstation*. Aktifitas pengendalian dan pengawasan ini hanya dapat diakhiri oleh pengguna yang login sebagai “admin” atau *Network Admin*.

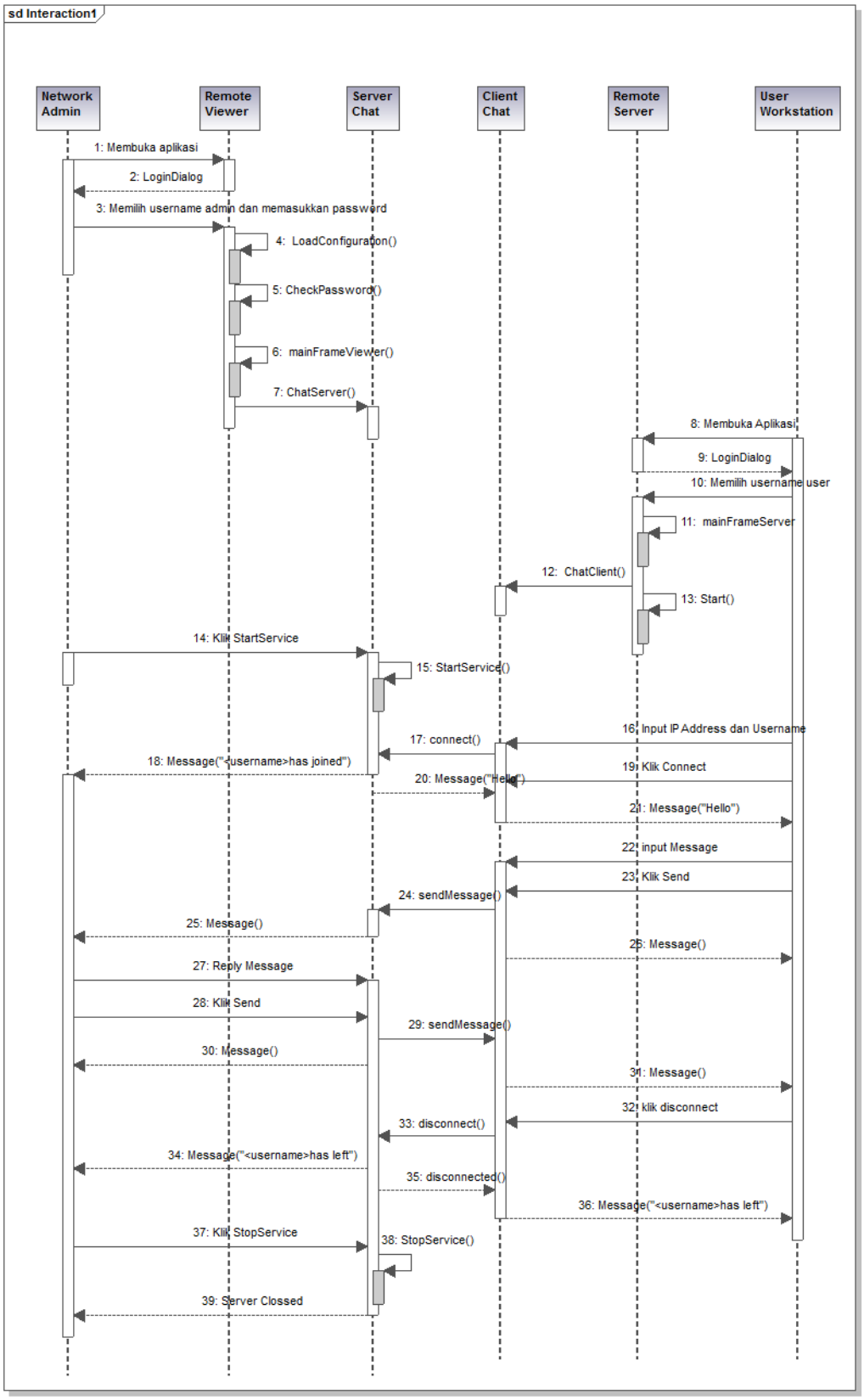
Bagian *remote and monitoring* merupakan bagian inti dari aplikasi. Pada bagian ini lah *Remote Method Invovation* diterapkan. Kelas yang menjadi ciri dari *Remote Method Invocation* adalah adanya kelas *implements* dan *interface.* Kelas *interface* adalah kelas yang berisi dengan deklarasi dari *method – method* yang digunakan pada kelas *implements.*



Gambar 17. *Sequence Diagram Remote and Monitoring*

Sequence diagram yang ketiga adalah *chatting.* Gambar 18 menunjukkan *sequence diagram* dari *chatting.* Sama seperti bagian *remote and monitoring,* untuk dapat melakukan aktifitas *chatting* maka pengguna harus melalui tahap *login to application.* Bagian *chatting* ini adalah alur kerja sistem untuk proses pengiriman pesan antara "user" dan "admin". Penghubung antara "user" dan "admin" ini adalah *Server Chat* untuk "admin" dan *Client Chat* untuk "user". "user" untuk dapat memulai *chatting* harus melakukan koneksi ke *Server Chat* dahulu dengan memasukkan *IP Address* dan *username*.

Kelas – kelas yang dibuat pun sesuai dengan *diagram sequence,* yaitu ada kelas untuk *Client Chat,* dan kelas untuk *Server Chat.* Dalam kelas – kelas tersebut ditempatkan *method – method. Method – method* yang diigunakan pada bagian ini diantaranya adalah *connect(), disconnect(), sendMessage(), startService(),* dan *stopService(). Method* nantinya dikembangkan dan dikodekan ke dalam bahasa pemrograman.



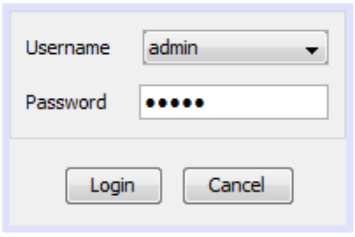
Gambar 18. *Sequence Diagram Chatting*

c. Desain *Interface*

Desain *interface* merupakan desain untuk tampilan aplikasi *Remote and Monitoring.*

1) *Form Login*

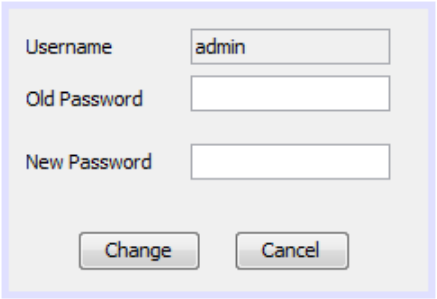
Form login merupakan tampilan awal untuk masuk ke aplikasi *Remote and Monitoring*. *Form login* akan muncul pada saat awal aplikasi dieksekusi.



Gambar 19. *Form Login*

2) Form Change Password

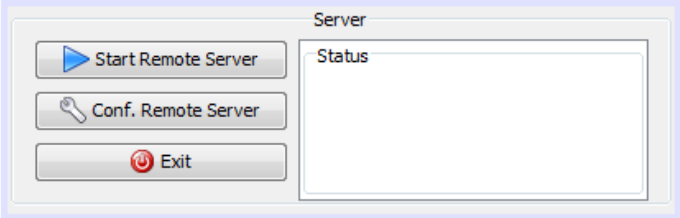
Form change password berguna untuk membantu pengguna mengganti *password login* untuk *admin*. *Form change password* akan muncul apabila pengguna menekan tombol *"Change Password"* pada *Form mainframe Viewer.*



Gambar 20. *Form Change Password*

3) Form mainframe Server

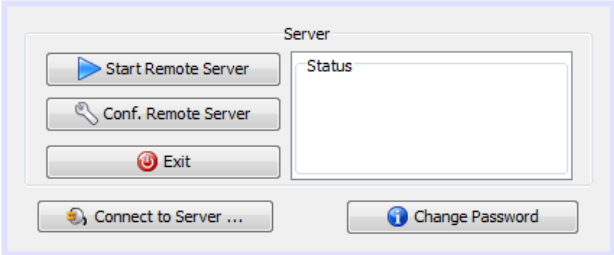
Form mainframe Server adalah tampilan utama untuk pengguna yang *login* sebagai *"user"*.



Gambar 21. *Form mainframe Server*

4) Form mainframe Viewer

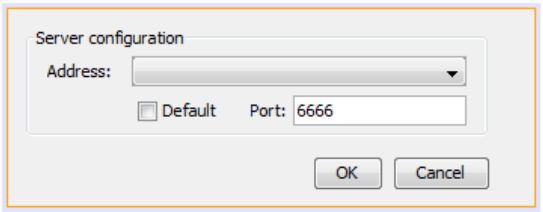
Form mainframe Viewer adalah tampilan utama untuk pengguna yang *login* sebagai *"admin"*.



Gambar 22. *Form mainframe Viewer*

5) Form Server Configuration

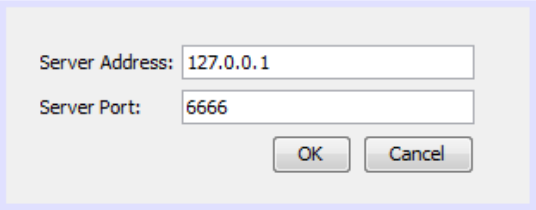
Form Server Configuration adalah form yang akan muncul apabila pengguna menekan tombol "*Conf. Remote Server*" pada tampilan utama *Remote and Monitoring.*



Gambar 23. *Form Server Configuration*

6) Form Connection Dialog

Form Connection Dialog adalah form yang akan muncul apabila pengguna menekan tombol "*Connect to Server …*" pada tampilan *Form mainframe Viewer.*



Gambar 24. *Form Connection Dialog*

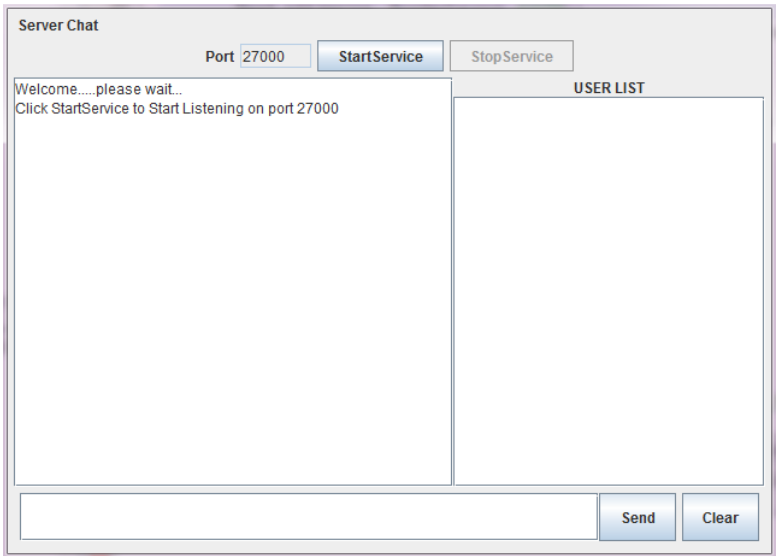
7) Form Remote and Monitoring Viewer

Form Remote and Monitoring Viewer ini akan muncul apabila *Remote Viewer* berhasil terhubung ke *Remote Server*. *Form* ini nantinya akan memunculkan gambar bergerak yang berhasil di-*capture* oleh aplikasi *Remote Server* pada komputer *User Workstation. Form* ini hanya dimiliki oleh pengguna yang login sebagai "*admin*".

Gambar 25. *Form Remote and Monitoring Viewer*

8) Form Server Chat

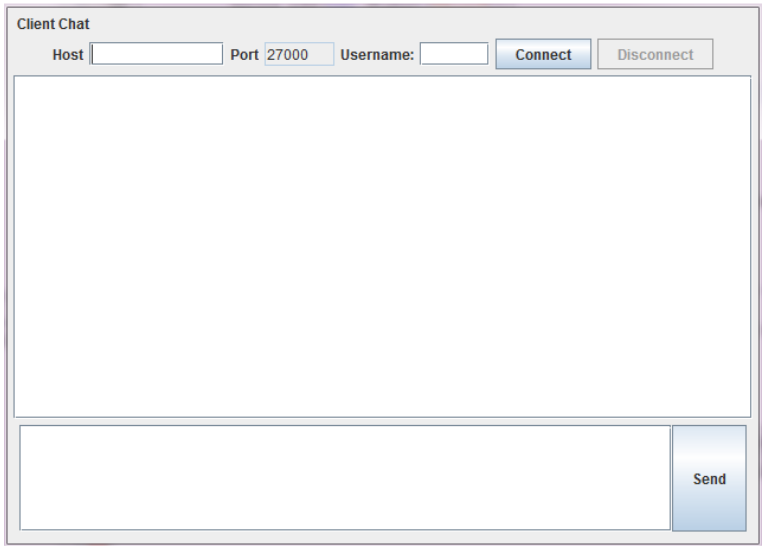
Form Server Chat adalah tampilan *chatting* untuk pengguna yang *login* sebagai *"admin"*.



Gambar 26. *Form Server Chat*

9) Form Client Chat

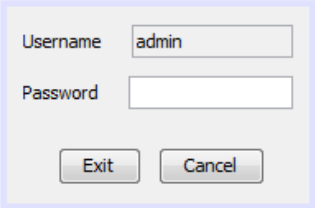
Form Client Chat adalah tampilan *chatting* untuk pengguna yang *login* sebagai *"user"*.



Gambar 27. *Form Client Chat*

10) Form Exit Dialog

Form Exit Dialog adalah tampilan yang akan muncul saat pengguna akan mengakhiri aplikasi dengan menekan tombol *"Exit"* pada tampilan utama *Remote and Monitoring.*



Gambar 28. *Form Exit Dialog*

C. Instrumen dan Pengumpulan Data

1. Instrumen Penelitian

Instrumen penelitian adalah suatu alat yang digunakan mengukur fenomena alam maupun sosial yang diamati. Secara spesifik semua fenomena ini disebut variable penelitian (Sugiyono, 2009:148).

Instrumen penelitian dibuat dari penurunan analisis kebutuhan. Analisis kebutuhan akan menghasilkan spesifikasi aplikasi *Remote and Monitoring.* Instrumen digunakan untuk mengidentifikasi ketercapaian spesifikasi terhadap hasil jadi produk aplikasi *Remote and Monitoring.*

a. *Alpha Testing*

Alpha testing merupakan bagian dari validasi perangkat lunak yang sudah dibangun. Pengujian ini dilakukan oleh ahli *(expert judgment)* untuk mendapatkan penilaian unjuk kerja dari aplikasi *Remote and Monitoring.*

Tabel 19. Pengujian Aplikasi bagian Otentikasi

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Login* - *Network Admin*	1. Pengguna dapat masuk ke aplikasi *Remote and Monitoring Viewer* dengan memilih *username "admin"* dan memasukkan *password* yang sesuai. 2. Jendela *Remote and Monitoring Viewer* terbuka. 3. Jendela *Chat Server* terbuka.		
	- *User Workstation*	1. Pengguna dapat masuk ke aplikasi *Remote and Monitoring Server* dengan memilih *username "user"* dan *password* kosong. 2. Jendela *Remote and Monitoring Server* terbuka. 3. Jendela *Chat Client* terbuka.		
2.	*Change Password*	1. Pengguna yang hanya *login* sebagai *"admin"* dapat mengganti *password* lama dengan *password* baru untuk *login* aplikasi *Remote and Monitoring*. 2. Aplikasi menyimpan konfigurasi *password* yang baru pada file *"user.config"*.		
3.	*Exit Application*	1. Pengguna yang hanya mempunyai *password "admin"* dapat menutup/ mengakhiri aplikasi. 2. Aplikasi *Remote and Monitoring* tertutup.		

Tabel 20. Pengujian Aplikasi bagian *Remote and Monitoring*

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Start Remote Server*	1. Pengguna dapat memulai *Remote Server* secara otomatis setelah pengguna *login* memilih *username "user"*. 2. Aplikasi dapat menerima koneksi dari *Remote Viewer*.		
2.	*Configuration Remote Server*	1. Pengguna dapat mengkonfigurasi *IP Address* dan *Port* yang digunakan untuk *Remote Server.* 2. Aplikasi menyimpan konfigurasi yang baru pada file *"server.config"*.		
3.	*Stop Remote Server*	1. Pengguna yang hanya mempunyai *password "admin"* dapat mengakhiri *Remote Server*. 2. Aplikasi tidak dapat menerima koneksi dari *Remote Viewer.*		
4.	*Connect to Remote Server*	1. Pengguna yang hanya *login* sebagai *"admin"* dapat menghubungkan ke komputer *Remote Server*. 2. Aplikasi meminta masukan *IP Address* dan *Port* dari komputer *Remote Server*.		
5.	*Remote and Monitoring*	*1.* Pengguna yang hanya *login* sebagai *"admin"* dapat mengawasi dan mengendalikan komputer *Remote Server.* 2. Aplikasi menampilkan tampilan komputer dari *Remote Server*.		

Tabel 21. Pengujian Aplikasi bagian *Chatting*

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Start Chat Server*	1. Pengguna yang hanya *login* sebagai *"admin"* dapat memulai *Chat Server*. 2. Aplikasi menampilkan notifikasi telah siap menerima koneksi dari *Chat Client*. 3. Aplikasi dapat menerima koneksi dari *Chat Client.*		
2.	*Stop Chat Server*	1. Pengguna yang hanya *login* sebagai *"admin"* dapat mengakhiri *Chat Server*. 2. Aplikasi menampilkan notifikasi telah menutup koneksi dari *Chat Client*. 3. Aplikasi mengakhiri koneksi dari semua *Chat Client.*		
3.	*Chatting*	1. Pengguna baik *"admin"* maupun *"user"* dapat berkomunikasi melalui aplikasi *Chat.* 2. Pengguna baik *"admin"* maupun *"user"* dapat bertukar pesan. 3. Aplikasi dapat menampilkan pesan dari *"admin"* dan *"user"*.		
4.	*Disconnect Chat Client*	1. Pengguna *Client Chat* dapat mengakhiri koneksi dari *Server Chat* dengan memilih *disconnect chat client.* 2. Aplikasi mengakhiri koneksi dari *Server Chat.*		
5.	*Connect Chat Client*	1. Pengguna *Client Chat* dapat menghubungi *"admin"* dengan terlebih dahulu menghubungkan *Client Chat* ke *Server Chat.* 2. Pengguna dapat konek ke *Server Chat* dengan memasukkan *IP Address Server Chat* dan *username.* 3. Aplikasi dapat menampilkan notifikasi telah terhubung dengan *Server Chat.*		

Tabel 22. Unjuk Kerja Sistem *Remote and Monitoring*

No.	Unjuk Kerja Sistem	Sesuai	
		Ya	Tidak
1.	Kemampuan aplikasi untuk memberikan kewenangan pada pengguna dalam menggunakan aplikasi dan fasilitas – fasilitas yang dipunyai dari aplikasi berdasarkan dari *username* saat *login* (Otentikasi).		
2.	Kemampuan aplikasi memproses pengendalian dan pengawasan komputer yang terhubung dalam jaringan komputer *(Remote and Monitoring).*		
3.	Kemampuan aplikasi mengelola komunikasi antara *Network Admin* dan *User Workstation* yang terhubung dalam jaringan komputer *(Chatting).*		

b. *Beta Testing*

Tabel 23. Kisi – kisi Penilaian dari segi *Usability* (Timothy, 2002:245)

No	Aspek	Indikator	Butir
1	*Learnability*	Kemudahan mempelajari produk.	1
		Kecepatan untuk menguasai sampai menjadi mahir.	2
2	*Efficiency of use*	Kecepatan menyelesaikan tugas.	3
		Kecepatan membenahi kesalahan.	4
3	*Error Handling*	Pesan kesalahan.	5
4	*Acceptability*	Kebermanfaatan produk.	6
		Kepuasan terhadap produk.	7, 8

Jawaban instrument penelitian untuk *beta* menggunakan skala *Likert*. Skala *Likert* yang digunakan untuk pengukuran pada *beta-testing* mempunyai gradasi dari sangat positif sampai sangat negatif. Berikut adalah jawaban untuk pengukuran pada *beta-testing.*

Tabel 24. Skor pertanyaan

No	Jawaban	Skor
1	SS (Sangat setuju)	4
2	S (Setuju)	3
3	TS (Tidak setuju)	2
4	STS (Sangat tidak setuju)	1

2. Teknik Analisis Data

Penelitian ini merupakan penelitian deskriptif yang bersifat *developmental* sehingga dalam penelitian ini tidak dimaksudkan untuk menguji hipotesis tertentu, tetapi hanya menggambarkan apa adanya tentang suatu keadaan (Suharsimi Arikunto 2009: 234).

Teknis analisis data yang dilakukan pada tahap pertama adalah menggunakan deskriptif kualitatif yaitu memaparkan produk hasil pengembangan aplikasi *Remote and Monitoring* setelah diimplementasikan dalam bentuk produk jadi dan menguji unjuk kerja produk. Tahap kedua menggunakan deskriptif kuantitatif, yaitu memaparkan mengenai kelayakan produk dari segi *usability.* Data kuantitatif yang diperoleh kemudian diubah menjadi data kualitatif dengan menggunakan skala *Likert*. Skala *Likert* memiliki gradasi dari sangat positif sampai sangat negatif yang dapat diwujudkan dalam beragam kata-kata. Tingkatan bobot nilai yang digunakan sebagai skala pengukuran adalah 4, 3, 2, 1.

Dari data instrumen penelitian, kemudian dengan melihat bobot tiap tanggapan yang dipilih atas tiap pernyataan, selanjutnya menghitung skor rata-rata hasil penilaian dengan menggunakan rumus:

$$\overline{X} = \frac{\sum X}{n}$$

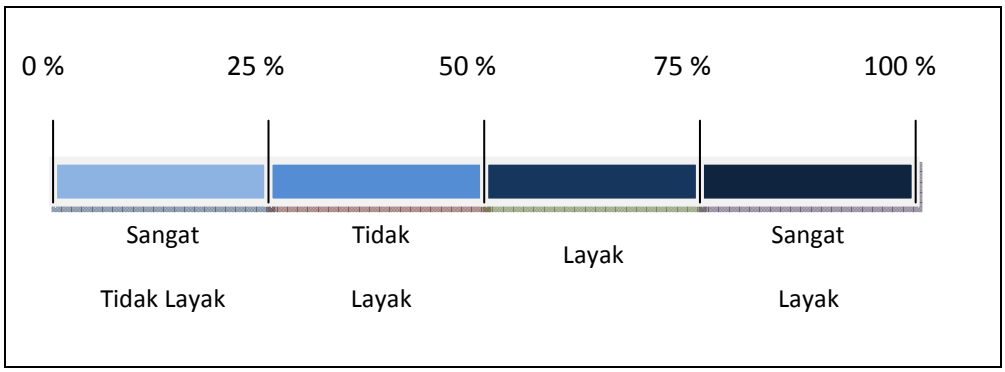
Keterangan:

$\overline{X}$ = skor rata-rata
n = jumlah penilai
$\sum X$ = skor total masing-masing penilai

Rumus perhitungan persentase skor ditulis dengan rumus berikut :

$$Persentase\ Kelayakan\ (\%) = \frac{Skor\ yang\ diobservasi}{Skor\ yang\ diharapkan}\ x\ 100\%$$

Setelah persentase didapatkan maka nilai tersebut dirubah dalam pernyataan predikat yang menunjuk pada pernyataan keadaaan, ukuran kualitas. Data yang terkumpul dianalisis dengan analisis deskriptif kuantitatif yang diungkapkan dalam distribusi skor dan presentase terhadap kategori skala penilaian yang telah ditentukan. Setelah penyajian dalam bentuk presentase, untuk menentukan kategori kelayakan dari aplikasi *Remote and Monitoring* ini, dipakai skala pengukuran skala *Likert*.



Gambar 29. Skala Pengukuran

Selanjutnya, kategori kelayakan digolongkan menggunakan skala sebagai berikut:

Tabel 25. Tabel Kategori Kelayakan berdasarkan skala *Likert.*

No	Skor dalam Persen (%)	Kategori Kelayakan
1	0% - 25%	Sangat Tidak Layak
2	>25% - 50%	Tidak Layak
3	>50% - 75%	Layak
4	>75% - 100%	Sangat Layak

BAB IV

HASIL PENELITIAN DAN PEMBAHASAN

A. Hasil Penelitian

1. Implementasi Pengkodean

Implementasi dilakukan setelah proses analisis kebutuhan sampai dengan desain sistem. Desain system yang sudah ada diterjemahkan ke dalam bahasa mesin sehingga nantinya dapat dieksekusi oleh komputer.

Kelas – kelas yang dibuat berdasarkan masing – masing objek, dan juga *utility* pendukung. Kelas – kelas ini dikelompokkan menjadi beberapa *package. Package* utama adalah *"remote_and_monitoring_b",* dan di dalamnya terdapat *package "ikon", "remoter", "viewer", "messanger*" dan *"utility"*.

Implementasi pengkodean adalah kelanjutan proses setelah desain. Fungsi atau *method* yang diperlukan diubah ke dalam bahasa pemrograman kemudian tiap unit langsung diuji. Pengujian langsung pada tahap ini disebut *white-box testing.* Ini dimaksudkan untuk menghindari kesalahan – kesalahan yang semakin besar. Pengkodean dilakukan secara bertahap sesuai dengan kebutuhan sistem.

a. *Package remote_and_monitoring_b*

Peneliti mengawali pengkodean dengan membuat kelas main. Setelah kelas *main* dibuat maka kelas untuk *mainFrame* dan *systray* dibuat.

Tabel 26. Fungsi atau *method* pada *package remote_and_monitoring*

method	class	Package
main(String args[])	main	Remote_and_monitoring_b
startServer(int port)	main	Remote_and_monitoring_b
startViewer(String server, int port)	main	Remote_and_monitoring_b
getCurrentDirectory ()	main	Remote_and_monitoring_b
exit()	main	Remote_and_monitoring_b
main(String[] args)	mainFrame	Remote_and_monitoring_b
mainFrame()	mainFrame extends javax.swing.JFrame	Remote_and_monitoring_b
jButtonStartStopActionPerformed(java.awt.event.ActionEvent evt)	mainFrame extends javax.swing.JFrame	Remote_and_monitoring_b
jButtonConfigActionPerformed(java.awt.event.ActionEvent evt)	mainFrame extends javax.swing.JFrame	Remote_and_monitoring_b
jButtonExitActionPerformed(java.awt.event.ActionEvent evt)	mainFrame extends javax.swing.JFrame	Remote_and_monitoring_b
jButton1ActionPerformed(java.awt.event.ActionEvent evt)	mainFrame extends javax.swing.JFrame	Remote_and_monitoring_b
formWindowClosing(java.awt.event.WindowEvent evt)	mainFrame extends javax.swing.JFrame	Remote_and_monitoring_b
formWindowActivated(java.awt.event.WindowEvent evt)	mainFrame extends javax.swing.JFrame	Remote_and_monitoring_b
isServerRunning()	mainFrame extends javax.swing.JFrame	Remote_and_monitoring_b
updateStatus()	mainFrame extends javax.swing.JFrame	Remote_and_monitoring_b
updateServerStatus(int msgType)	SysTray	Remote_and_monitoring_b
displayViewer(String viewer, int size, boolean connected)	SysTray	Remote_and_monitoring_b
isSupported()	SysTray	Remote_and_monitoring_b
isEnabled()	SysTray	Remote_and_monitoring_b
isServerRunning()	SysTray	Remote_and_monitoring_b
Hide()	SysTray	Remote_and_monitoring_b
Show()	SysTray	Remote_and_monitoring_b

b. *Package* Server.rmi

Package rmi adalah inti dari aplikasi untuk server yang berguna untuk membuka *port* dan *listening port* pada komputer target yang akan dikendalikan nantinya.

Tabel 27. Fungsi atau *method* pada *package server.rmi*

method	class	Package
Start()	Server extends Thread	Server.rmi
Stop()	Server extends Thread	Server.rmi
isRunning()	Server extends Thread	Server.rmi
isIdle()	Server extends Thread	Server.rmi
updateData(byte[] data, int index)	Server extends Thread	Server.rmi
updateData(int index)	Server extends Thread	Server.rmi
AddObject(Object object)	Server extends Thread	Server.rmi
addViewer(InetAddress inetAddress)	Server extends Thread	Server.rmi
removeViewer(int index)	Server extends Thread	Server.rmi
disconnectAllViewers()	Server extends Thread	Server.rmi
getViewersAds ()	Server extends Thread	Server.rmi
getViewersCount ()	Server extends Thread	Server.rmi
getStatus()	Server extends Thread	Server.rmi
startViewer(InetAddress inetAddress) throws RemoteException	ServerImpl extends UnicastRemoteObject implements ServerInterface	Server.rmi
stopViewer(int index) throws RemoteException	ServerImpl extends UnicastRemoteObject implements ServerInterface	Server.rmi
updateData(byte[] data, int index)	ServerImpl extends UnicastRemoteObject implements ServerInterface	Server.rmi
updateData(int index)	ServerImpl extends UnicastRemoteObject implements ServerInterface	Server.rmi

c. *Package* Server

Package server berisi juga *package rmi*. *Package* ini berisi kelas – kelas untuk interface dari server. Konfigurasi server juga dibuatkan pada package ini.

Tabel 28. Fungsi atau *method* pada dari *server*

method	class	Package
loadConfiguration()	Config	Server
storeConfiguration ()	Config	Server
SetConfiguration(int Port)	Config	Server
robot()	robot	Server
BufferedImage captureScreen()	robot	Server
CaptureScreenByteArray()	robot	Server
getScreenRect()	robot	Server
updateData(Object object)	robot	Server
applyMouseEvent(MouseEvent evt)	robot	Server
applyKeyEvent(KeyEvent evt)	robot	Server
main(String args[])	ConfigGUI extends javax.swing.JFrame	Server
ConfigGUI()	ConfigGUI extends javax.swing.JFrame	Server
jButtonCancelActionPerformed(java.awt.event.ActionEvent evt)	ConfigGUI extends javax.swing.JFrame	Server
jButtonOKActionPerformed(java.awt.event.ActionEvent evt)	ConfigGUI extends javax.swing.JFrame	Server

d. *Package remoter*

Package remoter didalamnya terdapat *package rmi. Package remoter* ini berisi kelas – kelas untuk interface untuk pengendalian ke komputer target. *Package remoter* tidak terdapat pada aplikasi "a".

Tabel 29. Fungsi atau *method* pada *package remoter*

method	class	Package
loadConfiguration()	Config	remoter
storeConfiguration ()	Config	remoter
SetConfiguration(String Address, int Port)	Config	remoter
main(String[] args)	ConnectionDialog extends javax.swing.JFrame	remoter
ConnectionDialog()	ConnectionDialog extends javax.swing.JFrame	remoter
jButtonCancelActionPerformed(java.awt.event.ActionEvent evt)	ConnectionDialog extends javax.swing.JFrame	remoter
jButtonOKActionPerformed(java.awt.event.ActionEvent evt)	ConnectionDialog extends javax.swing.JFrame	remoter
Recorder(Viewer viewer)	Recorder extends Thread	remoter
run()	Recorder extends Thread	remoter
Wait()	Recorder extends Thread	remoter
Notify()	Recorder extends Thread	remoter
Stop()	Recorder extends Thread	remoter
Start()	Recorder extends Thread	remoter
isRecording ()	Recorder extends Thread	remoter
updateData(ArrayList objects)	Recorder extends Thread	remoter
ScreenPlayer(Recorder recorder)	ScreenPlayer extends JLabel	remoter

Tabel 29. Fungsi atau *method* pada *package remoter (lanjutan)*

method	class	Package
addAdapters()	ScreenPlayer extends JLabel	remoter
removeAdapters()	ScreenPlayer extends JLabel	remoter
paint(Graphics g)	ScreenPlayer extends JLabel	remoter
setScteenRect(Rectangle rect)	ScreenPlayer extends JLabel	remoter
UpdateScreen(byte[] data)	ScreenPlayer extends JLabel	remoter
clearScreen()	ScreenPlayer extends JLabel	remoter
isScreenRectChanged ()	ScreenPlayer extends JLabel	remoter
ViewerGUI(Recorder recorder)	ViewerGUI extends javax.swing.JFrame	remoter
Start()	ViewerGUI extends javax.swing.JFrame	remoter
formWindowClosing(java.awt.event.WindowEvent evt)	ViewerGUI extends javax.swing.JFrame	remoter

e. *Package remoter.rmi*

Package rmi yang merupakan bagian dari *package remoter* hanya berisi satu kelas. Kelas ini berguna untuk menerima proses yang dikirim dari komputer target.

Tabel 30. Fungsi atau *method* pada *package remoter.rmi*

method	class	Package
Viewer ()	Viewer extends Thread	Remoter.rmi
isConnected()	Viewer extends Thread	Remoter.rmi
Start()	Viewer extends Thread	Remoter.rmi
Stop()	Viewer extends Thread	Remoter.rmi
connect()	Viewer extends Thread	Remoter.rmi
disconnect()	Viewer extends Thread	Remoter.rmi
updateData(Object object)	Viewer extends Thread	Remoter.rmi
AddObject(Object object)	Viewer extends Thread	Remoter.rmi
sendData()	Viewer extends Thread	Remoter.rmi
recieveData()	Viewer extends Thread	Remoter.rmi
displayStatus()	Viewer extends Thread	Remoter.rmi

f. *Package Messanger*

Package messsanger berisi dua kelas untuk aplikasi *remote and monitoring* sebagai pendukung *chatting*. Dua kelas tersebut mempunyai fungsi untuk menjadi *Server Chatting* dan *Client Chatting*.

Tabel 31. Fungsi atau *method* pada *package messanger*

method	class	Package
main(String[] args)	ChatClientGUI	messanger.client
Connect()	ChatClientGUI	messanger.client
Disconnect()	ChatClientGUI	messanger.client
Send()	ChatClientGUI	messanger.client
run()	threadrecieve	messanger.client
main(String[] args)	ChatServerGUI	messanger.server
windowClosing(WindowEvent we)	ChatServerGUI	messanger.server
StartService()	ChatServerGUI	messanger.server
StopService()	ChatServerGUI	messanger.server
Send()	ChatServerGUI	messanger.server
run()	threadserver	messanger.server
Run()	threadclient	messanger.server

g. *Package Utility*

Package utility berisi tiga kelas pendukung untuk aplikasi *remote and monitoring* ini. Tiga kelas tersebut mempunyai fungsi untuk pengalamatan komputer target, *image utility* untuk menampung data dalam bentuk image , *dan zip utility* untuk mengompress data.

Tabel 32. Fungsi atau *method* pada *package utility*

method	class	Package
read(InputStream in) throws IOException	ImageUtility	Server.rmi
read(byte[] bytes)	ImageUtility	Server.rmi
toByteArray(BufferedImage image)	ImageUtility	Server.rmi
InetAddress getLocalAdr()	InetAdrUtility	Server.rmi
String[] getLocalIPAdresses()	InetAdrUtility	Server.rmi
Object byteArraytoObject(byte[] data) throws Exception	ZipUtility	Server.rmi
byte[] objecttoByteArray(Object obj) throws IOException	ZipUtility	Server.rmi

2. Hasil Pengujian Terintegrasi

a. *Black-box Testing*

Pengujian *black-box* adalah pengujian terintegrasi yang dilakukan oleh peneliti untuk memastikan bahwa sistem sudah siap untuk diuji *alpha.* Pengujian *black-box* wajib dilakukan sebelum pengujian *alpha.* Peneliti melakukan pengujian *black-box* aplikasi *Remote and Monitoring* dengan membagi menjadi 13 bagian sebagaimana *use case* pada analisis kebutuhan dan perancangan sistem.

Tabel 33. Hasil Pengujian *Black-box*

No.	Use Case	Aplikasi Bagian	Hasil Pengujian
1.	*Login* - *Network Admin* - *User Workstation*	Otentikasi	Sesuai
2.	*Change Password*	Otentikasi	Sesuai
3.	*Exit Application*	Otentikasi	Sesuai
4.	*Start Remote Server*	*Remote and Monitoring*	Sesuai
5.	*Configuration Remote Server*	*Remote and Monitoring*	Sesuai
6.	*Stop Remote Server*	*Remote and Monitoring*	Sesuai
7.	*Connect to Remote Server*	*Remote and Monitoring*	Sesuai
8.	*Remote and Monitoring*	*Remote and Monitoring*	Sesuai
9.	*Start Chat Server*	*Chatting*	Sesuai
10.	*Stop Chat Server*	*Chatting*	Sesuai
11.	*Chatting*	*Chatting*	Sesuai
12.	*Disconnect Chat Client*	*Chatting*	Sesuai
13.	*Connect Chat Client*	*Chatting*	Sesuai

b. *Alpha Testing*

Uji *alpha* adalah proses validasi produk yang telah dibangun. Ahli melakukan validasi dengan mengoreksi kesalahan – kesalahan dan kekurangan yang ada dalam produk, kemuadian memberikan saran dan komentar serta rekomendasi untuk perbaikan. Hasil dari koreksi tersebut menjadi data yang akan digunakan untuk merevisi produk aplikasi *Remote and Monitoring.* Berikut adalah hasil pengujian yang dilakukan peneliti dengan seorang ahli rekayasa perangkat lunak:

Tabel 34. Pengujian Aplikasi bagian Otentikasi

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Login* - *Network Admin*	1. Pengguna dapat masuk ke aplikasi *Remote and Monitoring Viewer* dengan memilih *username "admin"* dan memasukkan *password* yang sesuai. 2. Jendela *Remote and Monitoring Viewer* terbuka. 3. Jendela *Chat Server* terbuka.	√	
	- *User Workstation*	1. Pengguna dapat masuk ke aplikasi *Remote and Monitoring Server* dengan memilih *username "user"* dan *password* kosong. 2. Jendela *Remote and Monitoring Server* terbuka. 3. Jendela *Chat Client* terbuka.	√	
2.	*Change Password*	1. Pengguna yang hanya *login* sebagai *"admin"* dapat mengganti *password* lama dengan *password* baru untuk *login* aplikasi *Remote and Monitoring*. 2. Aplikasi menyimpan konfigurasi *password* yang baru pada file *"user.config"*.	√	
3.	*Exit Application*	1. Pengguna yang hanya mempunyai *password "admin"* dapat menutup/ mengakhiri aplikasi. 2. Aplikasi *Remote and Monitoring* tertutup.	√	

Tabel 35. Pengujian Aplikasi bagian *Remote and Monitoring*

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Start Remote Server*	1. Pengguna dapat memulai *Remote Server* secara otomatis setelah pengguna *login* memilih *username "user"*. 2. Aplikasi dapat menerima koneksi dari *Remote Viewer*.	√	
2.	*Configuration Remote Server*	1. Pengguna dapat mengkonfigurasi *IP Address* dan *Port* yang digunakan untuk *Remote Server.* 2. Aplikasi menyimpan konfigurasi yang baru pada file *"server.config"*.	√	
3.	*Stop Remote Server*	1. Pengguna yang hanya mempunyai *password "admin"* dapat mengakhiri *Remote Server.* 2. Aplikasi tidak dapat menerima koneksi dari *Remote Viewer.*	√	
4.	*Connect to Remote Server*	1. Pengguna yang hanya *login* sebagai *"admin"* dapat menghubungkan ke komputer *Remote Server*. 2. Aplikasi meminta masukan *IP Address* dan *Port* dari komputer *Remote Server.*	√	
5.	*Remote and Monitoring*	*1.* Pengguna yang hanya *login* sebagai *"admin"* dapat mengawasi dan mengendalikan komputer *Remote Server.* 2. Aplikasi menampilkan tampilan komputer dari *Remote Server.*	√	

Tabel 36. Pengujian Aplikasi bagian *Chatting*

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Start Chat Server*	1. Pengguna yang hanya *login* sebagai *"admin"* dapat memulai *Chat Server*. 2. Aplikasi menampilkan notifikasi telah siap menerima koneksi dari *Chat Client*. 3. Aplikasi dapat menerima koneksi dari *Chat Client.*	√	
2.	*Stop Chat Server*	1. Pengguna yang hanya *login* sebagai *"admin"* dapat mengakhiri *Chat Server*. 2. Aplikasi menampilkan notifikasi telah menutup koneksi dari *Chat Client.* 3. Aplikasi mengakhiri koneksi dari semua *Chat Client.*	√	
3.	*Chatting*	1. Pengguna baik *"admin"* maupun *"user"* dapat berkomunikasi melalui aplikasi *Chat.* 2. Pengguna baik *"admin"* maupun *"user"* dapat bertukar pesan. 3. Aplikasi dapat menampilkan pesan dari *"admin"* dan *"user"*.	√	
4.	*Disconnect Chat Client*	1. Pengguna *Client Chat* dapat mengakhiri koneksi dari *Server Chat* dengan memilih *disconnect chat client.* 2. Aplikasi mengakhiri koneksi dari *Server Chat.*	√	
5.	*Connect Chat Client*	1. Pengguna *Client Chat* dapat menghubungi *"admin"* dengan terlebih dahulu menghubungkan *Client Chat* ke *Server Chat.* 2. Pengguna dapat konek ke *Server Chat* dengan memasukkan *IP Address Server Chat* dan *username.* 3. Aplikasi dapat menampilkan notifikasi telah terhubung dengan *Server Chat.*	√	

Tabel 37. Unjuk Kerja Sistem *Remote and Monitoring*

No.	Unjuk Kerja Sistem	Sesuai	
		Ya	Tidak
1.	Kemampuan aplikasi untuk memberikan kewenangan pada pengguna dalam menggunakan aplikasi dan fasilitas – fasilitas yang dipunyai dari aplikasi berdasarkan dari username saat login (Otentikasi).	√	
2.	Kemampuan aplikasi memproses pengendalian dan pengawasan komputer yang terhubung dalam jaringan komputer *(Remote and Monitoring).*	√	
3.	Kemampuan aplikasi mengelola komunikasi antara *Network Admin* dan *User Workstation* yang terhubung dalam jaringan komputer *(Chatting).*	√	

Ahli rekayasa perangkat lunak memberikan kesimpulan bahwa aplikasi *Remote and Monitoring* yang telah peneliti bangun adalah mempunyai unjuk kerja yang baik. Ahli rekayasa perangkat lunak juga memberikan saran untuk menambah fasilitas *chatting* agar dibuat *full-duplex*, supaya *Network Admin* maupun *User Workstation* dapat saling bertukar informasi.

c. *Beta Testing*

Pengujian *beta* dilakukan dengan melibatkan sembilan pengguna yang telah ditentukan sebelumnya. Berikut adalah hasil pengujian *beta* dari pengguna khusus.

Tabel 38. Hasil Uji Beta Daftar Penguji

No	Aspek *usability* yang diuji	Penguji								
		1	2	3	4	5	6	7	8	9
1	Tombol – tombol pada aplikasi *Remote and Monitoring* familiar.	3	4	4	4	4	3	3	3	4
2	Aplikasi *Remote and Monitoring* dengan cepat dikuasai.	4	4	4	4	4	4	3	2	4
3	Prosedur penggunaan aplikasi *Remote and Monitoring* tidak berbelit – belit.	2	3	3	4	2	3	3	3	3
4	Tidak membutuhkan waktu lama untuk membenahi kesalahan.	3	3	3	4	3	3	3	3	4
5	Pesan kesalahan mudah untuk dipahami.	3	2	3	4	3	2	3	3	4
6	Aplikasi *Remote and Monitoring* sangat membantu untuk seorang pengelola jaringan komputer.	2	4	4	4	4	3	3	3	4
7	Aplikasi *Remote and Monitoring* menjadikan jarak yang jauh antara *User Workstation* dan *Network Admin* bukan suatu masalah.	3	4	4	4	4	3	3	3	4
8	Aplikasi *Remote and Monitoring* menjadikan komunikasi antara *User Workstation* dan *Network Admin* lebih cepat.	4	3	4	4	3	3	3	3	4

Keterangan:

4 = Sangat Setuju **2 = Tidak Setuju**

3 = Setuju **1 = Sangat Tidak Setuju**

Para pengguna penilai perangkat lunak juga memberikan masukan atau saran sebagai berikut:

1) Perlu ada dokumentasi yang lengkap.
2) Pengaturan tata letak window sebaiknya ditampilkan ditengah.
3) User interface dipermanis.
4) Semua fasilitas yang ada dijadikan satu frame.
5) Perlu menu Help.
6) Ditambah menu log.
7) Menu – menu dibuat pull-down.

B. Pembahasan

1. *Alpha Testing*

Uji *alpha* yang dilakukan oleh ahli rekayasa perangkat lunak adalah untuk mendapatkan unjuk kerja dari aplikasi yang telah dibangun. Unjuk kerja didapat dengan menganalisa aplikasi dari spesifikasi yang diharapkan saat analisis kebutuhan.

Unjuk kerja yang diharapkan dari aplikasi *Remote and Monitoring* ini meliputi 3 bagian, yakni bagian otentikasi, bagian *remote and monitoring*, dan bagian *chatting.* Keseluruhan dari reviu oleh ahli rekayasa perangkat lunak menjadi bahan pertimbangan dan masukan yang berharga bagi peneliti.

Bagian otentikasi secara keseluruhan sudah sesuai dengan spesifikasi harapan. Semua spesifikasi harapan tercapai dan ahli sudah setuju dengan unjuk kerja bagian otentikasi pada aplikasi ini.

Bagian *Remote and Monitoring* merupakan inti dari sistem aplikasi. Secara keseluruhan spesifikasi harapan pada bagian *Remote and \monitoring* ini sudah tercapai, ahli sudah setuju dengan unjuk kerja bagian *remote and monitoring.*

Bagian *Chatting* secara keseluruhan sudah memenuhi spesifikasi harapan, hanya beberapa saran yang harus diperhatikan sebagai landasan untuk revisi produk.

Berdasarkan pengujian yang sudah dilakukan ahli, maka secara teoritis aplikasi *Remote and Monitoring* berbasis Java RMI ini mempunyai unjuk kerja yang baik.

a. Revisi untuk *alpha*

Revisi awal dilakukan sesuai dengan saran ahli rekayasa perangkat lunak. Sesuai dengan hasil pengujian ahli rekayasa perangkat lunak, peneliti merevisi aplikasi *Remote and Monitoring* ini dengan menambahkan fitur *"Send Warning"*. Fitur *"Send Warning"* ini dapat memberikan peringatan dari *Network Admin* kepada *User Workstation* berupa pesan singkat yang akan muncul di layar komputer *User Workstation.*

2. *Beta Testing*

Uji *beta* merupakan pengujian lanjutan dari uji *alpha.* Revisi awal dari uji *alpha* dilanjutkan dengan uji *beta.* Uji *beta* aplikasi *Remote and Monitoring* dilakukan oleh sembilan pengguna khusus. Uji *beta* ini meliputi beberapa aspek dari *usability. Usability* dapat dibagi menjadi empat bagian

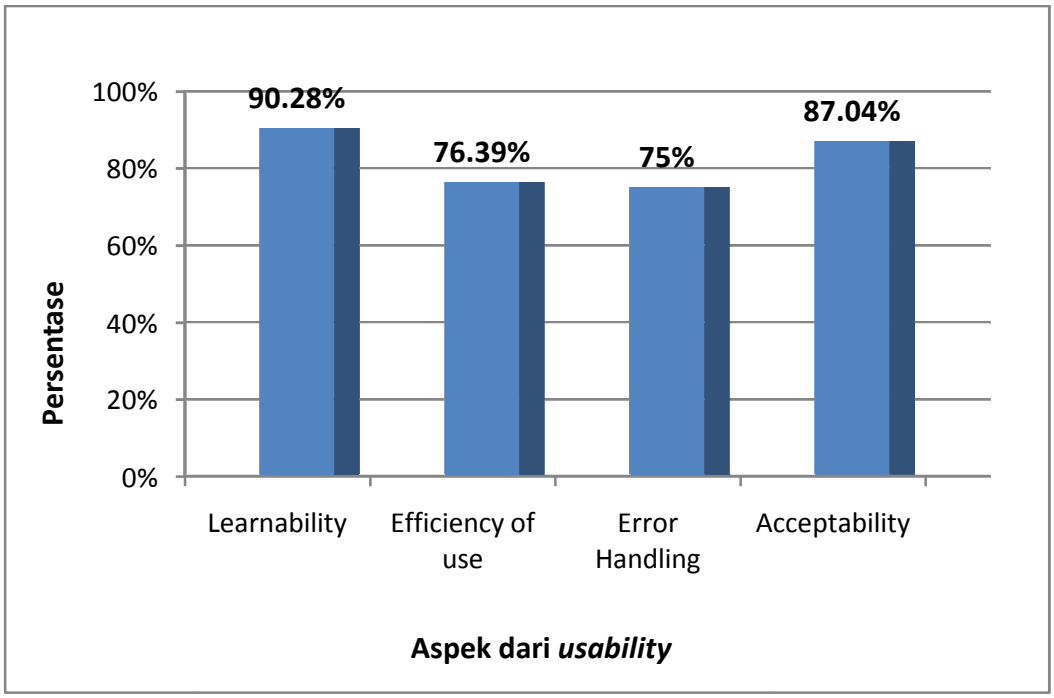
aspek: *learnability*, *efficiency of use*, *error handling* dan *acceptability* (Timothy C. Lethbridge, 2002:245).

Berdasarkan hasil pengujian *beta* yang dilakukan peneliti terhadap aplikasi *Remote and Monitoring* berbasis *Java RMI* kepada pengguna, kemudian dilakukan perhitungan untuk mendapatkan hasil penilaian pengguna secara keseluruhan. Hasil perhitungan tersebut ditabelkan sebagaimana pada tabel 42.

Tabel 39. Skor kelayakan dari segi *usability*

Pengguna	***Learnability***	***Efficiency of use***	***Error Handling***	***Acceptability***	
1	7	5	3	9	**R E R A T A**
2	8	6	2	11	
3	8	6	3	12	
4	8	8	4	12	
5	8	5	3	11	
6	7	6	2	9	
7	6	6	3	9	
8	5	6	3	9	
9	8	7	4	12	
Hasil skor	65	55	27	94	
Hasil maks	72	72	36	108	
Rerata item	3,61	3,06	3,00	3,48	3,29
Persen (%)	**90,28**	**76,39**	**75**	**87,04**	**82,18**

Berdasarkan hasil perhitungan, maka dapat di gambarkan dengan diagram batang sebagai berikut:

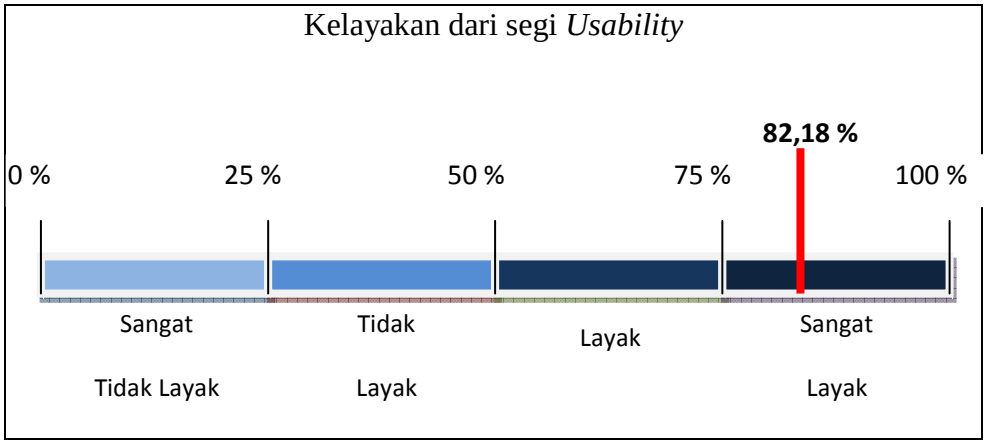


Gambar 30. Persentase dari aspek *usability*

Tabel 40. Skor masing – masing aspek dari *usability*

No.	Aspek	Persentase	Kategori Kelayakan
1	*Learnability*	90,28 %	Sangat Layak
2	*Efficiency of use*	76,39 %	Sangat Layak
3	*Error Handling*	75,00 %	Layak
4	*Acceptability*	87,04 %	Sangat Layak
Rata – rata keseluruhan		**82,18 %**	**Sangat Layak**

Berdasarkan perhitungan rata – rata dari segi *usability*, maka dapat di gambarkan posisi tingkat *usability* dari aplikasi *Remote and Monitoring* sebagai berikut:



Gambar 31.Skor penilaian dari pengguna

Berdasarkan posisi yang telah digambarkan, dapat dijabarkan hasil penilaian dari 9 orang pengguna khusus dari *Focus Group Discussion Digital Networks and Multimedia* PUSKOM UNY bahwa aplikasi ini mempunyai tingkat kelayakan 82,18 %. Berdasarkan hasil penilaian tersebut, dapat disimpulkan bahwa aplikasi *Remote and Monitoring* berbasis Java RMI adalah sangat layak.

a. Revisi Akhir *(beta)*

Aplikasi *Remote and Monitoring* berbasis Java RMI dikembangkan dengan menggunakan program utama *NetBeans IDE 6.9.1,*. Aplikasi *Remote and Monitoring* ini telah mengikuti tahap – tahap dalam pengembangan sesuai dengan tahap pengembangan perangkat lunak menurut Pressman. Aplikasi *Remote and Monitoring* ini telah selesai divalidasi oleh ahli rekayasa perangkat lunak.

Aplikasi divalidasi oleh ahli rekayasa perangkat lunak, kemudian dilakukan berbagai revisi terhadap aplikasi *Remote and Monitoring* sesuai dengan saran ahli yang bersangkutan sampai diperoleh hasil aplikasi Remote and Monitoring yang diharapkan. Penilaian dilakukan

oleh pengguna khusus *Focus Group Discussion Digital Networks and Multimedia* PUSKOM UNY.

Aplikasi *Remote and Monitoring* yang dihasilkan dari penelitian pengembangan ini memiliki kelebihan dan kelemahan. Kelebihan dari aplikasi *Remote and Monitoring* adalah sebagai berikut:

1) Aplikasi ini dapat dijalankan di semua jenis sistem operasi *(Multiplatform).*
2) Aplikasi simpel dan mudah digunakan.

Kelemahan aplikasi *Remote and Monitoring*:

1) Adanya *delay* antara komputer target dan komputer remoter. Untuk koneksi yang lemah tidak akan bisa menampilkan *video* dari komputer yang di-*remote.*
2) Membutuhkan *Java(TM) SE Runtime Environment* 1.6.0 ke atas.

C. Hasil Akhir Produk

Aplikasi *Remote and Monitoring* berbasis Java RMI ini merupakan aplikasi yang digunakan untuk mengawasi dan mengendalikan komputer yang saling terhubung jaringan. Aplikasi *Remote and Monitoring* ini mempunyai tiga bagian, yakni: otentikasi, *remote and monitoring,* dan *chatting.*

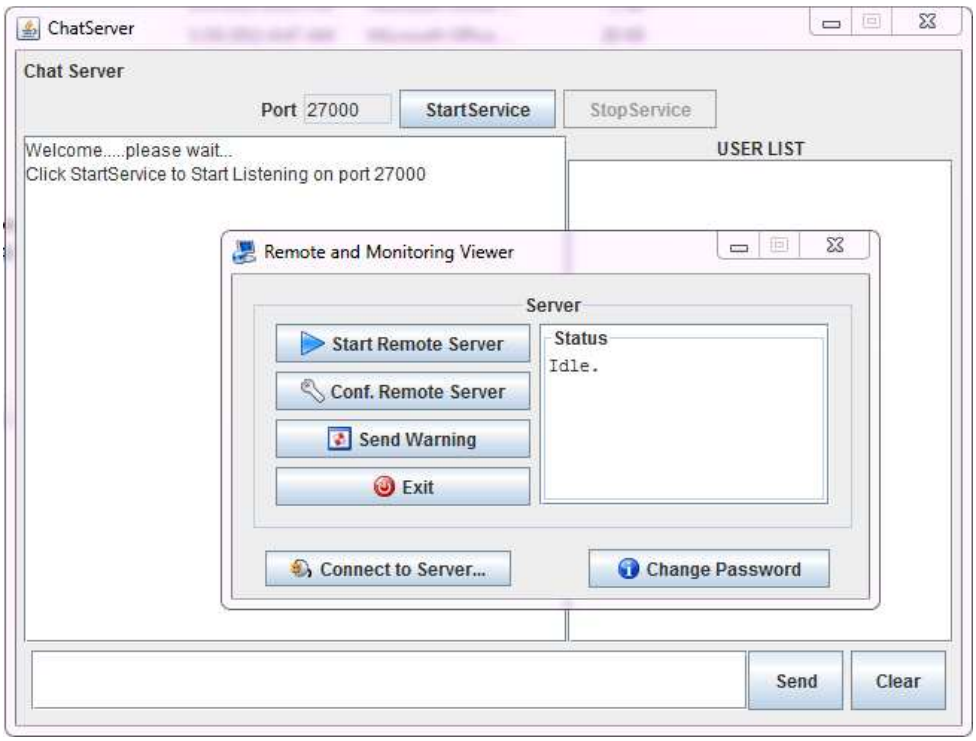
1. Otentikasi

Aplikasi bagian otentikasi merupakan bagian untuk mengatur keamanan sistem. Otentikasi ini meliputi *login* aplikasi, *change password,* dan otentikasi untuk keluar dari aplikasi. *Login* aplikasi berguna untuk

membatasi kewenangan terhadap aplikasi. Pengguna aplikasi dapat login sebagai admin atau user.

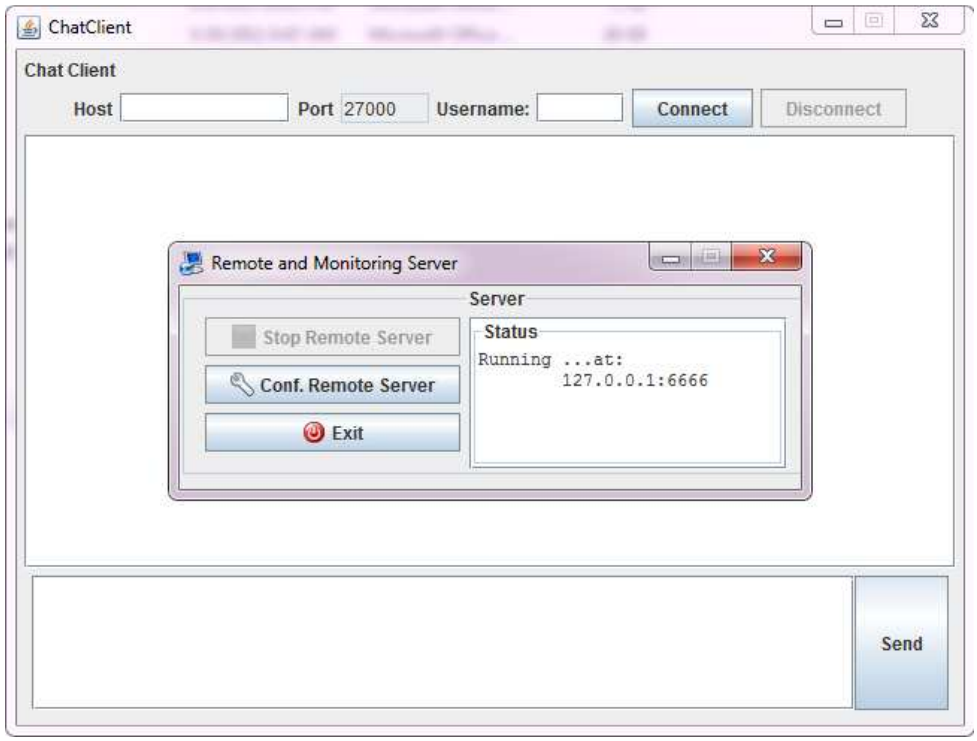
Gambar 32. Otentikasi pada saat login

"*admin*" mempunyai kewenangan penuh terhadap aplikasi. "*admin*" dapat mengawasi dan mengendalikan komputer *User Workstation* yang sedang dalam keadaan "*on*".



Gambar 33. Tampilan utama untuk pengguna"*admin*"

"user" adalah untuk pengguna *User Workstation*. *"user"* tidak mempunyai kewenangan penuh seperti *"admin"*. *"user"* hanya dapat menghubungi *"admin"* melalui aplikasi *Client Chat*.



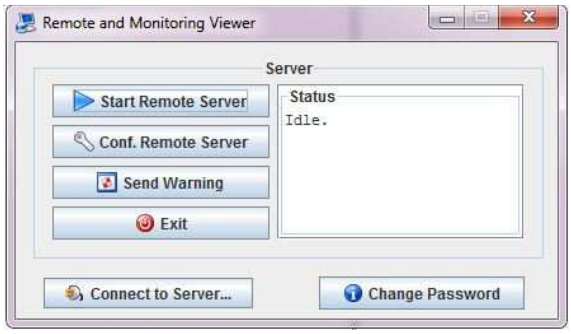
Gambar 34. Tampilan utama untuk pengguna *"user"*

2. Remote and Monitoring

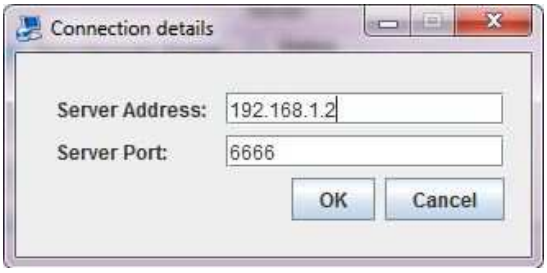
Aplikasi bagian *Remote and Monitoring* merupakan bagian inti dari aplikasi. Bagian ini berfungsi untuk mengawasi dan mengendalikan komputer *"user"* yang dalam kondisi *"on"* dan terhubung jaringan.

Gambar 35. *Remote Server Started*

"admin" dapat mengawasi dan mengendalikan komputer *"user"* dengan memilih menu *"Connect to Server"*. *"admin"* harus memasukkan *IP Address* dan *port* komputer *Remote Server* milik *"user"* untuk dapat menggunakan fasilitas *Remote and Monitoring* ini.

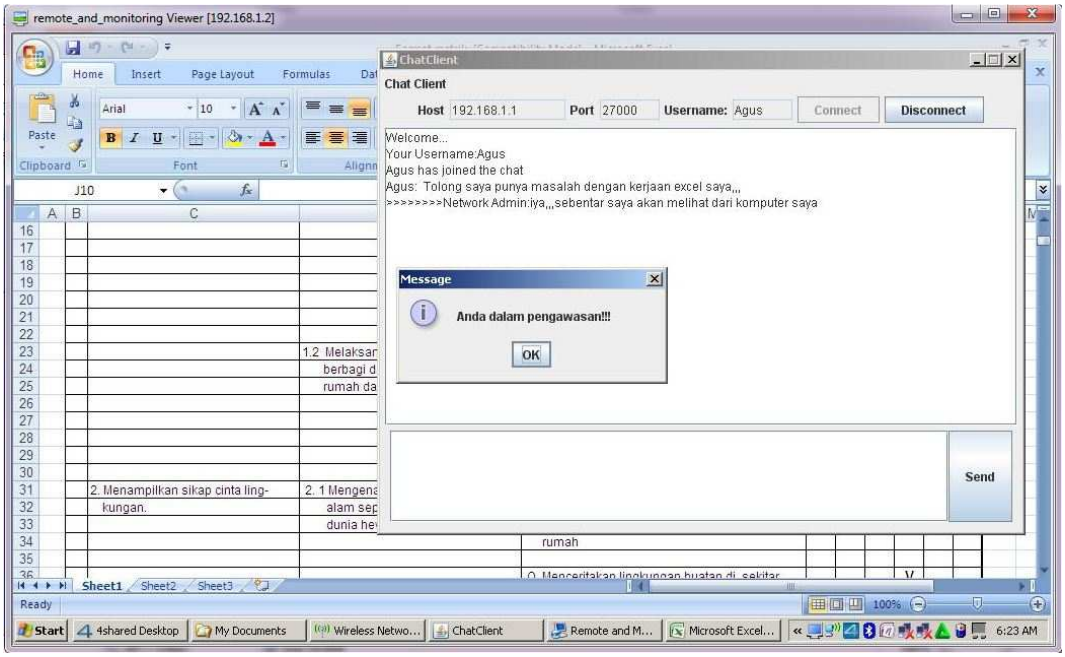


Gambar 36. *Form* utama *Remote Viewer*



Gambar 37. *Form* untuk memasukkan *IP Address Remote Server*

"admin" mempunyai kewenangan untuk mengendalikan komputer *User Workstation* melalui komputernya dengan aplikasi ini. Fitur – fitur dari aktifitas *Remote and Monitoring* yang dapat dilakukan oleh *"admin"* adalah dapat menggerakkan atau mengendalikan gerak *pointer mouse User Workstation,* dapat melihat aktifitas secara *real-time* dari komputer *User Workstation,* dan dapat memberikan input dari keyboard komputer *Network Admin* terhadap komputer *User Workstation.*



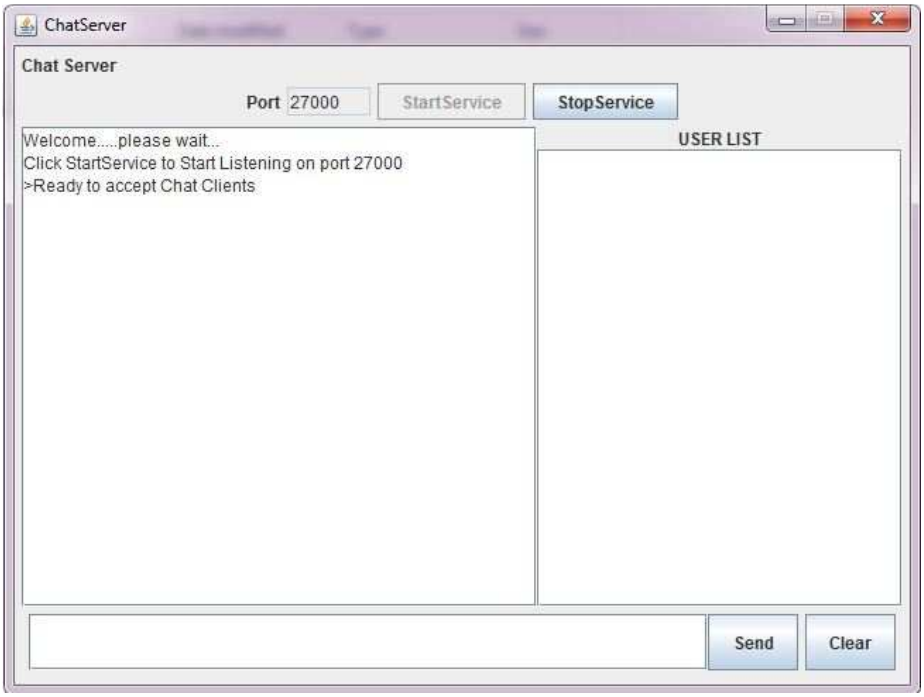
Gambar 38. Tampilan pada komputer *User Workstation* yang ditangkap oleh komputer *Network Admin.*

3. Chatting

Aplikasi bagian Chatting berguna untuk kirim pesan antar *"user"* dan *"admin"*. Bagian ini dapat mempercepat komunikasi antara *User Workstation* dan *Network Admin*, sehingga *User Workstation* dapat lebih mudah menghubungi *Network Admin* untuk mendapatkan bantuan ataupun mendapatkan informasi dari *Network Admin*. *Network Admin* bertugas memantau Chat Server untuk menunggu apakah ada *"user"* yang masuk *chatting.*

Pengguna dapat melakukan aktifitas *chatting* dengan beberapa langkah sebagai beriku:

a. Start Service pada aplikasi *Chat Server* bagi *username admin.*

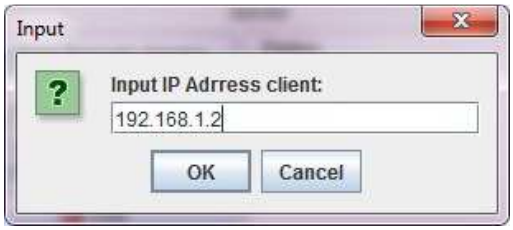


Gambar 39. Tampilan *Chat Server*

b. Masukkan IP Address dan Username untuk *Chat Client* bagi *username user*

Gambar 40. Tampilan *Chat Client*

c. Pengguna yang *login* sebagai *admin* dapat memberikan pesan peringatan kepada *user* dengan memilih dan klik tombol "*Send Warning*" pada tampilan utama. Kemudian pengguna perlu memasukkan *IP Address* target dan Pesan yang akan dikirim secara berurutan.



Gambar 41. Masukkan IP Address target

Gambar 42. Masukkan pesan *warning*

d. Setelah berhasil dikirim maka dapat dilihat dari tampilan *User Workstation* muncul pesan *warning* yang terkirim.

Gambar 43. Tampilkan pesan *warning* untuk *User Workstation*

BAB V

KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan hasil penelitian pengembangan yang dilakukan pada pengembangan perangkat lunak yang dibangun tentang "Aplikasi *Remote and Monitoring* berbasis Java RMI" maka dapat disimpulkan sebagai berikut.

1. Pengembangan perangkat lunak *Remote System* untuk Aplikasi *Remote and Monitoring* melalui tahapan analisis kebutuhan, desain sistem dan pengkodean menggunakan skrip pemrograman Java sebagai pengawasan dan pengendalian komputer. Berdasarkan hasil pengujian *white Box* dan *black Box*, program dapat bekerja sesuai dengan spesifikasi yang telah ditentukan yaitu dapat mengawasi dan mengendalikan komputer yang terhubung melalui jaringan komputer.
2. Berdasarkan pengujian *alpha* yang dilakukan oleh ahli rekayasa perangkat lunak, aplikasi *Remote and Monitoring* berbasis *Java RMI* memiliki unjuk kerja yang baik, semua sistem yang diujikan dapat berjalan dan bekerja sesuai dengan spesifikasi yang diharapkan.
3. Berdasarkan pengujian *beta* yang dilakukan oleh pengguna *Focus Group Discussion Digital Networks and Multimedia* PUSKOM UNY, kelayakan aplikasi *Remote and Monitoring* berbasis *Java RMI* dari segi *usability* adalah sangat layak dengan persentase sebanyak 82,14%.

B. Saran

Peneliti menyarankan untuk pengembangan lebih lanjut :

1. *Delay* yang terjadi pada pengiriman data proses dari komputer target ke komputer pengendali dapat dikurangi dengan mengkompres data yang dikirim.
2. Aplikasi *Remote and Monitoring* berbasis Java RMI ini dapat dikembangkan menjadi aplikasi *Remote System* yang lebih komplek dengan menambah fitur – fitur lain.

DAFTAR PUSTAKA

Adikara, Putra Pandu. (2011). *Daya Guna (Usability).* *http://hikaruyuuki.lecture.ub.ac.id/files/2011/02/04-Daya-Guna-Usability.pdf*, (20 Maret 2011).

Benyammi, Bachir., Hassani,Mustapha. & Bensaad, Mohammed Lahcen. (2008). *Java Remote Desktop.* *http://jrdesktop.sourceforge.net*, (1 Mei 2010).

Fikri, Rijalul., Adam, Ipam Fuadian., Prakoso, Imam. (2005). *Pemrograman Java.* Yogyakarta:Andi.

Gandjar Kiswanto dan Abdurrasyid Mujahid. (2007). *Analisis dan Desain Pengembangan Modul Roughing 3-Axis pada Sistem CAM (Computer Aided Manufacturing) Berbasis Model Faset 3D.* Jakarta: Departemen Teknik Mesin Universitas Indonesia.

Hafidz, Firdaus. (2009). *PENGERTIAN MONITORING DAN EVALUASI.* http://hafidzf.wordpress.com/2009/06/16/pengertian-monitoring-dan-evaluasi/, (1 Januari 2010).

Irwan Pribadi dan Mukhammad Andri Setiawan. (2005). *Manajemen Pengelolaan LAN dengan Remote System Application.* Yogyakarta:SNATI.

Jammal, Ziad. (2003). *Remote Command.* *http://www.planet-source-code.com*, (1 Mei 2010).

Kurniawan, Yunus. (2010). *Pembangunan Aplikasi Remote Task Manager* pada Jaringan Komputer Berbasis Windows. Yogyakarta:STIE AMIKOM.

Lethbridge, Timothy C., & Laganière, Robert. (2002). *Object-Oriented Software Engineering:Practical software development using UML and Java.* England:Mc Graw Hill International.

Liang, Y. Daniel. (2007).*Introduction to Java programming:comprehensive version*. Pearson Education, Inc:USA.

Merz, M., Lamersdorf, W., (1993). *Generic Interfaces to Remote Applications in Open Systems.* Department of Computer Science:University of Hamburg.

Najmuddin, Muhammad. (2011). *IMK – Usability Prinsiples.* *http://mumurangkas.blogspot.com/2010/02/imk-usability-principles.html*, (9 Maret 2011).

Naveed, Tahir. (2002). *Remote Access.* http://www.planet-source-code.com, (1 Mei 2010).

Nugroho, Adi. (2009). *Rekayasa Perangkat Lunak.* Yogyakarta:Andi.

Nugroho, Eddy Prasetyo. et al. (2009). *Rekayasa Perangkat Lunak.* Bandung:Politeknik Telkom.

Pressman, S Roger, (2002). *Rekayasa Perangkat Lunak: pendekatan praktisi (Buku I) / Roger S. Pressman; Diterjemahkan oleh: LN Harnaningrum,* ***Ed. II –*** Yogyakarta : Andi.

Pressman, S Roger, (2002). *Rekayasa Perangkat Lunak: pendekatan praktisi (Buku II) / Roger S. Pressman; Diterjemahkan oleh: LN Harnaningrum,* ***Ed. II –*** Yogyakarta : Andi.

Saqeeb, Md Tanzim. (2005). *ExtremePC 2.0 – Remote Control.* http://www.planet-source-code.com, (1 Mei 2010).

Shalahudin, M., A, S, Rosa. (2008). *Analisis dan Desain Sistem Informasi.* Bandung:Politeknik Telkom.

Sinaga, Benyamin L. (2005). *Pemrograman Berorientasi Objek dengan Java.* Yogyakarta:Penerbit Gava Media.

Sugiyono. (2009). *Metode Penelitian Pendidikan*. Bandung: Alfabeta.

Sukardi. (2004). *Metodologi Penelitian Pendidikan*. Jakarta: Bumi Aksara.

Wijono, Matius Soesilo., Wijono, G. Sri Hartati. et al. (2005). *Java 2 SE dengan JBuilder.* Yogyakarta:Andi.

Wikipedia. (2011). *Pengendali Jarak Jauh.* http://id.wikipedia.org/wiki/Pengendali_jarak_jauh, (1 Mei 2011).

Yatin, Chautan. (2004). *YQwe RMI Bank.* http://www.planet-source-code.com, (1 Mei 2010).

You, Xiang-bai Liu. et al. (2010). *The Design of a Remote Monitoring System Based On Java.* China:IEEE International Conference.

L A M P I R A N

```
package remote_and_monitoring_b;

import java.io.File;
import java.io.IOException;
import java.net.URL;
import remote_and_monitoring_b.server.rmi.Server;
import remote_and_monitoring_b.remoter.rmi.Viewer;

/**
 *
 * @author Agus Setiawan
 */
public class main {

    public static final URL IDLE_ICON =
main.class.getResource("ikon/idle.png");
    public static final URL ALIVE_ICON =
main.class.getResource("ikon/background.png");
    public static final URL WAIT_ICON =
main.class.getResource("ikon/display.png");
    public static final URL START_ICON =
main.class.getResource("ikon/player_play.png");
    public static final URL STOP_ICON =
main.class.getResource("ikon/player_stop.png");

    public static String SERVER_CONFIG_FILE;
    public static String VIEWER_CONFIG_FILE;
    public static String USER_CONFIG_FILE;

    public static void main(String args[]) {

        SERVER_CONFIG_FILE = getCurrentDirectory() +
"server.config";
        VIEWER_CONFIG_FILE = getCurrentDirectory() +
"viewer.config";
        USER_CONFIG_FILE = getCurrentDirectory() +
"user.config";


System.getProperties().remove("java.rmi.server.hostna
me");
        loginDialog.main(null);
    }

    public static void startViewer(String server, int
port) {

remote_and_monitoring_b.remoter.Config.SetConfigurati
on(server, port);
        new Viewer().Start();
    }

    public static String getCurrentDirectory () {
        String currentDirectory = null;
        try {
            currentDirectory = new
File(".").getCanonicalPath() + File.separatorChar;
        } catch (IOException e) {
            e.getStackTrace();
        }
        return currentDirectory;
    }

    public static void exit() {
        if (Server.isRunning())
            Server.Stop();
        System.exit(0);
    }
}
```

```
package remote_and_monitoring_b;

import java.awt.event.WindowEvent;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;

import remote_and_monitoring_b.server.ConfigGUI;
import remote_and_monitoring_b.server.rmi.Server;
import
remote_and_monitoring_b.remoter.ConnectionDialog;
import remote_and_monitoring_b.messanger.ChatClient;

public class mainFrameViewer extends
javax.swing.JFrame {
    private static mainFrameViewer frame = new
mainFrameViewer();

    /** Membuat Interface dari Remote */
    public mainFrameViewer() {
        initComponents();
        updateStatus();
    }

    /** This method is called from within the
constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content
of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed"
desc="Generated Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jButtonStartStop = new javax.swing.JButton();
        jButtonConfig = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTextAreaStatus = new
javax.swing.JTextArea();
        jButtonExit = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();


setDefaultCloseOperation(javax.swing.WindowConstants.
DO_NOTHING_ON_CLOSE);
        setTitle("Remote and Monitoring Viewer");
        setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        setIconImage(new
ImageIcon(main.IDLE_ICON).getImage());
        setResizable(false);
        addWindowListener(new
java.awt.event.WindowAdapter() {
            public void
windowActivated(java.awt.event.WindowEvent evt) {
                formWindowActivated(evt);
            }
            public void
windowClosing(java.awt.event.WindowEvent evt) {
                formWindowClosing(evt);
            }
        });


jPanel1.setBorder(javax.swing.BorderFactory.createTit
ledBorder(null, "Server",
javax.swing.border.TitledBorder.CENTER,
javax.swing.border.TitledBorder.DEFAULT_POSITION));

        jButtonStartStop.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/remote
_and_monitoring_b/ikon/player_play.png"))); // NOI18N
        jButtonStartStop.setText("Start Remote
Server");
        jButtonStartStop.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonStartStopActionPerformed(evt);
            }
        });

        jButtonConfig.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/remote
_and_monitoring_b/ikon/configure.png"))); // NOI18N
        jButtonConfig.setText("Conf. Remote Server");
        jButtonConfig.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonConfigActionPerformed(evt);
            }
        });

        jTextAreaStatus.setColumns(20);
        jTextAreaStatus.setEditable(false);
        jTextAreaStatus.setFont(new
java.awt.Font("Courier New", 0, 12));
        jTextAreaStatus.setRows(4);

jTextAreaStatus.setBorder(javax.swing.BorderFactory.c
reateTitledBorder("Status"));

jScrollPane1.setViewportView(jTextAreaStatus);

        jButtonExit.setIcon(new
```

```
javax.swing.ImageIcon(getClass().getResource("/remote
_and_monitoring_b/ikon/exit.png"))); // NOI18N
        jButtonExit.setText("Exit");
        jButtonExit.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonExitActionPerformed(evt);
            }
        });

        jButton3.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/remote
_and_monitoring_b/ikon/props.png"))); // NOI18N
        jButton3.setText("Send Warning");
        jButton3.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.TRAILING)
                    .addComponent(jButtonStartStop,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 164,
Short.MAX_VALUE)
                    .addComponent(jButtonConfig,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 164,
Short.MAX_VALUE)
                    .addComponent(jButton3,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 164,
Short.MAX_VALUE)
                    .addComponent(jButtonExit,
javax.swing.GroupLayout.DEFAULT_SIZE, 164,
Short.MAX_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.RELATED)
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 195,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jButtonStartStop)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.RELATED)
                        .addComponent(jButtonConfig)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.RELATED)
                        .addComponent(jButton3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.RELATED)
                        .addComponent(jButtonExit))
                    .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 118,
Short.MAX_VALUE))
                .addContainerGap())
        );
```

```
        jButton1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/remote
_and_monitoring_b/ikon/connect_creating.png"))); //
NOI18N
        jButton1.setText("Connect to Server ...");
        jButton1.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/remote
_and_monitoring_b/ikon/info.png"))); // NOI18N
        jButton2.setText("Change Password");
        jButton2.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()
                        .addGap(10, 10, 10)
                        .addComponent(jButton1,
javax.swing.GroupLayout.PREFERRED_SIZE, 165,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.RELATED, 41, Short.MAX_VALUE)
                        .addComponent(jButton2,
javax.swing.GroupLayout.PREFERRED_SIZE, 162,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(29, 29, 29))

.addGroup(layout.createSequentialGroup()
                        .addComponent(jPanel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addContainerGap())))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel1,
javax.swing.GroupLayout.DEFAULT_SIZE, 149,
Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.BASELINE)
                    .addComponent(jButton1,
javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButton2))
                .addContainerGap())
        );

        java.awt.Dimension screenSize =
java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-433)/2,
(screenSize.height-245)/2, 433, 245);
    }// </editor-fold>

private void
jButtonStartStopActionPerformed(java.awt.event.Action
```

```
Event evt) {
    if (Server.isRunning())
        Server.Stop();
    else
        Server.Start();

    updateStatus();
}

private void
jButtonConfigActionPerformed(java.awt.event.ActionEve
nt evt) {
    ConfigGUI.main(null);
}

private void
jButtonExitActionPerformed(java.awt.event.ActionEvent
evt) {
    ExitDialog.main(null);
}

private void
jButton1ActionPerformed(java.awt.event.ActionEvent
evt) {
    ConnectionDialog.main(null);
}

private void
formWindowClosing(java.awt.event.WindowEvent evt) {
    if (evt.getID() == WindowEvent.WINDOW_CLOSING) {
            if (!Server.isRunning() &&
(!SysTrayViewer.isEnabled() ||
!SysTrayViewer.isSupported())) {
                if
(JOptionPane.showConfirmDialog(this, "Are you sure to
exit ?",
                        "Confirm Dialog",
JOptionPane.OK_CANCEL_OPTION) ==
                    JOptionPane.OK_OPTION)
                        main.exit();
            }
            else
                dispose();
    }
    else
        super.processWindowEvent(evt);
}

private void
formWindowActivated(java.awt.event.WindowEvent evt) {
    updateStatus();
}

private void
jButton2ActionPerformed(java.awt.event.ActionEvent
evt) {
    ChangeDialog.main(null);   // TODO add your
handling code here:
}

private void
jButton3ActionPerformed(java.awt.event.ActionEvent
evt) {
    ChatClient.main(null);    // TODO add your
handling code here:
}

    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        if (frame.isDisplayable()) return;
        java.awt.EventQueue.invokeLater(new
Runnable() {
            public void run() {
                frame.setVisible(true);
            }
        });
    }

    public static boolean isServerRunning() {
        boolean bool = Server.isRunning();
        if (!bool)
            JOptionPane.showMessageDialog(null,
                "Server is not running !!",
                "Information",
JOptionPane.INFORMATION_MESSAGE);
        return bool;
    }
```

```
    public void updateStatus() {
        if (Server.isRunning()) {

jTextAreaStatus.setText(Server.getStatus());
            jButtonStartStop.setText("Stop Remote
Server");
            jButtonStartStop.setIcon(new
ImageIcon(main.STOP_ICON));
        }
        else {
            if (Server.isIdle())
                jTextAreaStatus.setText("Idle.");
            else
                jTextAreaStatus.setText("Stopped.");
            jButtonStartStop.setText("Start Remote
Server");
            jButtonStartStop.setIcon(new
ImageIcon(main.START_ICON));
        }
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButtonConfig;
    private javax.swing.JButton jButtonExit;
    private javax.swing.JButton jButtonStartStop;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextArea jTextAreaStatus;
    // End of variables declaration

}
```

```
package remote_and_monitoring_b;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

import remote_and_monitoring_b.server.ConfigGUI;
import remote_and_monitoring_b.server.rmi.Server;

public class SysTrayViewer {

    final static public int SERVER_STARTED = 1;
    final static public int SERVER_STOPPED = 2;
    final static public int CONNECTION_FAILED = 3;
    final static public int SERVER_RUNNING = 4;
    final static public int SERVER_NOT_RUNNING = 5;
    private static MenuItem serverItem;
    private static TrayIcon trayIcon;
    private static boolean enabled = false;

    public static void updateServerStatus(int
msgType) {
        if (!SystemTray.isSupported() || enabled ==
false) return;

        switch (msgType) {
            case SERVER_RUNNING:
                serverItem.setLabel("Stop Server");
                if (Server.isRunning()) {
                    if (Server.getViewersCount() !=
0)
                        trayIcon.setImage(new
ImageIcon(main.ALIVE_ICON).getImage());
                    else
                        trayIcon.setImage(new
ImageIcon(main.WAIT_ICON).getImage());
                }

trayIcon.setToolTip("remote_and_monitoring [Server
running]\n" +

remote_and_monitoring_b.server.Config.server_address)
;
                break;
            case SERVER_NOT_RUNNING:
                serverItem.setLabel("Start");
                trayIcon.setImage(new
ImageIcon(main.IDLE_ICON).getImage());

trayIcon.setToolTip("remote_and_monitoring [Server
stopped]\n" +

remote_and_monitoring_b.server.Config.server_address)
;
                break;
```

```
            case SERVER_STARTED:
                serverItem.setLabel("Stop");
                trayIcon.displayMessage("Connection
status", "Server Started !!",
                        TrayIcon.MessageType.INFO);
                trayIcon.setImage(new
ImageIcon(main.WAIT_ICON).getImage());

trayIcon.setToolTip("remote_and_monitoring [Server
running]\n" +

remote_and_monitoring_b.server.Config.server_address)
;
                break;
            case CONNECTION_FAILED:
                trayIcon.displayMessage("Connection
status", "Connection Failed !!",
                        TrayIcon.MessageType.ERROR);
                break;
            case SERVER_STOPPED:
                serverItem.setLabel("Start");
                trayIcon.displayMessage("Connection
status", "Server Stopped !!",
                        TrayIcon.MessageType.INFO);
                trayIcon.setImage(new
ImageIcon(main.IDLE_ICON).getImage());

trayIcon.setToolTip("remote_and_monitoring [Server
stopped]\n" +

remote_and_monitoring_b.server.Config.server_address)
;
                break;
        }
        serverItem.setEnabled(true);
    }
    public static void displayViewer(String viewer,
int size, boolean connected) {
        if (!SystemTray.isSupported() || enabled ==
false) return;

        if (connected) {
            trayIcon.displayMessage("Viewer details",
viewer + " connected !!",
                    TrayIcon.MessageType.INFO);
            if (size == 0) {
                trayIcon.setImage(new
ImageIcon(main.ALIVE_ICON).getImage());
            }
        } else {
            trayIcon.displayMessage("Viewer details",
viewer + " disconnected !!",
                    TrayIcon.MessageType.INFO);
            if (size == 0) {
                trayIcon.setImage(new
ImageIcon(main.WAIT_ICON).getImage());
            }
        }
    }
    public static boolean isSupported() {
        return SystemTray.isSupported();
    }

    public static boolean isEnabled() {
        return enabled;
    }

    public static boolean isServerRunning() {
        boolean bool = Server.isRunning();
        if (!bool) {
            JOptionPane.showMessageDialog(null,
                    "Server is not running !!",
                    "Information",
JOptionPane.INFORMATION_MESSAGE);
        }
        return bool;
    }

    public static void Hide() {
        enabled = false;
        if (!SystemTray.isSupported()) return;
        final SystemTray tray =
SystemTray.getSystemTray();
        tray.remove(trayIcon);
    }

    public static void Show() {
        if (!SystemTray.isSupported()) return;
        enabled = true;
        Runnable runner = new Runnable() {
```

```
            public void run() {
                final SystemTray tray =
SystemTray.getSystemTray();
                PopupMenu popup = new PopupMenu();
                trayIcon = new TrayIcon(new
ImageIcon(main.IDLE_ICON).getImage(),
                        "remote_and_monitoring",
popup);
                trayIcon.addActionListener(new
ActionListener() {

                    public void
actionPerformed(ActionEvent e) {
                        mainFrameViewer.main(null);
                    }
                });

                MenuItem item = new MenuItem("Open
remote_and_monitoring");
                item.setFont(new Font(null,
Font.BOLD, 12));
                item.addActionListener(new
ActionListener() {
                    public void
actionPerformed(ActionEvent e) {
                        mainFrameViewer.main(null);
                    }
                });

                popup.add(item);

                item = new MenuItem("-");
                popup.add(item);

                Menu menu = new Menu("Server");

                serverItem = new MenuItem("Start");

                serverItem.addActionListener(new
ActionListener() {

                    public void
actionPerformed(ActionEvent e) {
                        serverItem.setEnabled(false);
                        if (Server.isRunning()) {
                            Server.Stop();
                        } else {
                            Server.Start();
                        }
                    }
                });
                menu.add(serverItem);

                item = new MenuItem("Configuration
...");
                item.addActionListener(new
ActionListener() {

                    public void
actionPerformed(ActionEvent e) {
                        ConfigGUI.main(null);
                    }
                });
                menu.add(item);


                menu.add(item);
                popup.add(menu);

                item = new MenuItem("-");
                popup.add(item);

                popup.add(item);

                item = new MenuItem("-");
                popup.add(item);

                popup.add(item);

                item = new MenuItem("Exit");
                item.addActionListener(new
ActionListener() {

                    public void
actionPerformed(ActionEvent e) {
                       ExitDialog.main(null);
                    }
                });
                popup.add(item);
```

```
                try {
                    tray.add(trayIcon);
                } catch (AWTException e) {
                    System.err.println("Can't add to
tray");
                }

            if (Server.isRunning())

SysTrayViewer.updateServerStatus(SERVER_RUNNING);
            else {
                if (!Server.isIdle())

SysTrayViewer.updateServerStatus(SERVER_NOT_RUNNING);
            }
            }
        };
        EventQueue.invokeLater(runner);
    }
}
```

```
package remote_and_monitoring_b;

import javax.swing.JOptionPane;
import remote_and_monitoring_b.messanger.ChatServer;
import
remote_and_monitoring_b.messanger.Client.ChatClientGU
I;
import
remote_and_monitoring_b.messanger.Server.ChatServerGU
I;
import remote_and_monitoring_b.server.rmi.Server;

/**
 *
 * @author Agus
 */
public class loginDialog extends javax.swing.JDialog
{

    /** Creates new form loginDialog */
    public loginDialog(java.awt.Frame parent, boolean
modal) {
        super(parent, modal);
        initComponents();
    }

    /** This method is called from within the
constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content
of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed"
desc="Generated Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jPasswordField1 = new
javax.swing.JPasswordField();
        jComboBox1 = new javax.swing.JComboBox();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();


setDefaultCloseOperation(javax.swing.WindowConstants.
DISPOSE_ON_CLOSE);
        setTitle("Login Remote and Monitoring");
        setAlwaysOnTop(true);
        setIconImage(null);
        setIconImages(null);
        setResizable(false);

        jLabel1.setText("Username");

        jLabel2.setText("Password");

        jComboBox1.setModel(new
javax.swing.DefaultComboBoxModel(new String[] {
"admin", "user" }));

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
```

```
jPanel1Layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()
                        .addComponent(jLabel1)
                        .addGap(18, 18, 18)
                        .addComponent(jComboBox1, 0,
106, Short.MAX_VALUE))

.addGroup(jPanel1Layout.createSequentialGroup()
                        .addComponent(jLabel2)
                        .addGap(18, 18, 18)

.addComponent(jPasswordField1,
javax.swing.GroupLayout.DEFAULT_SIZE, 108,
Short.MAX_VALUE)))
                .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(jComboBox1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.UNRELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(jPasswordField1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE
, Short.MAX_VALUE))
        );

        jButton1.setText("Login");
        jButton1.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("Cancel");
        jButton2.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(31, 31, 31)
                .addComponent(jButton1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.UNRELATED)
                .addComponent(jButton2))
            .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE)
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.RELATED, 15, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.BASELINE)
                    .addComponent(jButton2)
                    .addComponent(jButton1))
                .addContainerGap())
        );

        java.awt.Dimension screenSize =
java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-208)/2,
(screenSize.height-160)/2, 208, 160);
    }// </editor-fold>

    private void
jButton1ActionPerformed(java.awt.event.ActionEvent
evt) {
        Config.loadConfiguration();
        String user = "user";
        if
(jComboBox1.getSelectedItem().equals(user)) {
            this.setVisible(false);
            Server.Start();
            SysTrayServer.Show();
            mainFrameServer.main(null);
            ChatClientGUI.main(null);
            ChatServer.main(null);
        }
        if
(jComboBox1.getSelectedItem().equals(Config.username)
&&
(Encrypt.reverse(jPasswordField1.getText()).equals(Co
nfig.password))) {
            this.setVisible(false);
            SysTrayViewer.Show();
            mainFrameViewer.main(null);
            ChatServerGUI.main(null);
        } else {
            JOptionPane.showMessageDialog(this,
"Wrong password or You are login with account of
user!!", "Warning", JOptionPane.WARNING_MESSAGE,
null);
            jPasswordField1.setText("");
        }       // TODO add your handling code here:
    }

    private void
jButton2ActionPerformed(java.awt.event.ActionEvent
evt) {
        System.exit(0);// TODO add your handling code
here:
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new
Runnable() {

            public void run() {
                loginDialog dialog = new
loginDialog(new javax.swing.JFrame(), true);
                dialog.addWindowListener(new
java.awt.event.WindowAdapter() {

                    public void
windowClosing(java.awt.event.WindowEvent e) {
                        System.exit(0);
                    }
                });
                dialog.setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JComboBox jComboBox1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPasswordField
jPasswordField1;
    // End of variables declaration
}
```

```
package remote_and_monitoring_b;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.Properties;

public class Config {

    public static String username = "admin";
    public static String password = "{vzqn";

    public static void loadConfiguration() {
        if (new
File(main.USER_CONFIG_FILE).canRead())
            try {
                Properties properties = new
Properties();
                properties.load(new
FileInputStream(main.USER_CONFIG_FILE));

                username =
properties.get("username").toString();
                password =
properties.get("password").toString();
            }
            catch (Exception e) {
                e.getStackTrace();
            }
       else
            storeConfiguration();
    }

    public static void storeConfiguration () {
        try {
            new
File(main.USER_CONFIG_FILE).createNewFile();
            Properties properties = new Properties();
            properties.put("username", username);
            properties.put("password", password);

            properties.store(new
FileOutputStream(main.USER_CONFIG_FILE),
                "remote_and_monitoring viewer
configuration file");
        } catch (Exception e) {
            e.getStackTrace();
        }
    }

    public static void SetConfiguration(String User,
String Pass) {
        username = User;
        password = Pass;

        storeConfiguration();
    }
}
```

```
package remote_and_monitoring_b;

import java.io.*;

class Encrypt {

    public static void main(String[] args) throws
IOException {
        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

        System.out.println("Please enter a word to be
encrypted:");
        String encrypt = br.readLine();//Stores word

        String Encrypted = reverse(encrypt);//Applies
reverse method to word
```

```
        System.out.println(Encrypted);
    }
//This method reverses the word
    public static String reverse(String a) {
        String word = "";
        for (int i = a.length(); i > 0; i--) {
            int b=a.charAt(i - 1)+13;
            word = word +(char) b;
        }
        return word;
    }
}
```

```
package remote_and_monitoring_b;

import javax.swing.JOptionPane;

/**
 *
 * @author Agus
 */
public class ChangeDialog extends javax.swing.JDialog
{

    /** Creates new form ChangeDialog */
    public ChangeDialog(java.awt.Frame parent,
boolean modal) {
        super(parent, modal);
        initComponents();
    }

    /** This method is called from within the
constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content
of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed"
desc="Generated Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jPasswordField1 = new
javax.swing.JPasswordField();
        jLabel4 = new javax.swing.JLabel();
        jPasswordField2 = new
javax.swing.JPasswordField();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();


setDefaultCloseOperation(javax.swing.WindowConstants.
DO_NOTHING_ON_CLOSE);
        setTitle("Change Password");
        setResizable(false);

        jLabel1.setText("Username");

        jLabel2.setText("New Password");

        jTextField1.setEditable(false);
        jTextField1.setText("admin");
        jTextField1.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField1ActionPerformed(evt);
            }
        });

        jLabel4.setText("Old Password");

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel2)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel1)
                            .addComponent(jLabel4))
                        .addGap(26, 26, 26)

.addGroup(jPanel1Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.TRAILING)

.addComponent(jPasswordField1,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 111,
Short.MAX_VALUE)

.addComponent(jPasswordField2,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 111,
Short.MAX_VALUE)

.addComponent(jTextField1,
javax.swing.GroupLayout.DEFAULT_SIZE, 111,
Short.MAX_VALUE))
                        .addGap(1, 1, 1)
                        .addComponent(jLabel3)))
                .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.BASELINE)
                    .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel1))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.RELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel4)
                    .addComponent(jPasswordField2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(jPasswordField1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(36, 36, 36)
                .addComponent(jLabel3))
        );

        jButton1.setText("Change");
        jButton1.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("Cancel");
        jButton2.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
```

```
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()
                        .addGap(38, 38, 38)
                        .addComponent(jButton1)
                        .addGap(18, 18, 18)
                        .addComponent(jButton2))
                    .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE
, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 112,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.BASELINE)
                    .addComponent(jButton1)
                    .addComponent(jButton2))
                .addContainerGap())
        );

        java.awt.Dimension screenSize =
java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-249)/2,
(screenSize.height-195)/2, 249, 195);
    }// </editor-fold>

    private void
jTextField1ActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
    }

    private void
jButton1ActionPerformed(java.awt.event.ActionEvent
evt) {
        Config.loadConfiguration();
        if
(jTextField1.getText().equals(Config.username) &&
(Encrypt.reverse(jPasswordField2.getText()).equals(Co
nfig.password))) {
            JOptionPane.showMessageDialog(this,
"Sukses");
            Config.password =
Encrypt.reverse(jPasswordField1.getText());
            Config.storeConfiguration();
            this.setVisible(false);
        } else {
            JOptionPane.showMessageDialog(this,
"Invalid Passowrd!!", "Warning",
JOptionPane.WARNING_MESSAGE, null);
            jPasswordField1.setText(null);
            jPasswordField2.setText(null);
        }       // TODO add your handling code here:
    }

    private void
jButton2ActionPerformed(java.awt.event.ActionEvent
evt) {
        this.setVisible(false);// TODO add your
handling code here:
    }

    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new
Runnable() {
            public void run() {
                ChangeDialog dialog = new
ChangeDialog(new javax.swing.JFrame(), true);
                dialog.addWindowListener(new
java.awt.event.WindowAdapter() {
                    public void
windowClosing(java.awt.event.WindowEvent e) {
                        System.exit(0);
                    }
                });
                dialog.setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPasswordField
jPasswordField1;
    private javax.swing.JPasswordField
jPasswordField2;
    private javax.swing.JTextField jTextField1;
    // End of variables declaration

}
```

```
package remote_and_monitoring_b;

import javax.swing.JOptionPane;
/**
 *
 * @author Agus
 */
public class ExitDialog extends javax.swing.JDialog {

    /** Creates new form loginDialog */
    public ExitDialog(java.awt.Frame parent, boolean
modal) {
        super(parent, modal);
        initComponents();
    }

    /** This method is called from within the
constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content
of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed"
desc="Generated Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jPasswordField1 = new
javax.swing.JPasswordField();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();


setDefaultCloseOperation(javax.swing.WindowConstants.
DISPOSE_ON_CLOSE);
        setTitle("Confirm Exit Remote and
Monitoring");
        setAlwaysOnTop(true);
        setResizable(false);

        jLabel1.setText("Username");

        jLabel2.setText("Password");

        jTextField1.setEditable(false);
        jTextField1.setText("admin");

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
```

```
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.LEADING, false)

.addGroup(jPanel1Layout.createSequentialGroup()
                        .addComponent(jLabel1)
                        .addGap(18, 18, 18)
                        .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, 98,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(jPanel1Layout.createSequentialGroup()
                        .addComponent(jLabel2)
                        .addGap(18, 18, 18)

.addComponent(jPasswordField1)))

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE
, Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.UNRELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(jPasswordField1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE
, Short.MAX_VALUE))
        );

        jButton1.setText("Exit");
        jButton1.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("Cancel");
        jButton2.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGroup(layout.createSequentialGroup()
                .addGap(31, 31, 31)
                .addComponent(jButton1)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.UNRELATED)
                .addComponent(jButton2))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.RELATED, 12, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.BASELINE)
                    .addComponent(jButton2)
                    .addComponent(jButton1))
                .addContainerGap())
        );

        java.awt.Dimension screenSize =
java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-200)/2,
(screenSize.height-157)/2, 200, 157);
    }// </editor-fold>

    private void
jButton1ActionPerformed(java.awt.event.ActionEvent
evt) {
        Config.loadConfiguration();
        if
(jTextField1.getText().equals(Config.username) &&
(Encrypt.reverse(jPasswordField1.getText()).equals(Co
nfig.password))) {
            System.exit(0);
        } else {
            JOptionPane.showMessageDialog(this,
"Invalid Passowrd!!", "Warning",
JOptionPane.WARNING_MESSAGE, null);
            jPasswordField1.setText("");
        }        // TODO add your handling code here:
    }

    private void
jButton2ActionPerformed(java.awt.event.ActionEvent
evt) {
        this.setVisible(false);// TODO add your
handling code here:
    }

    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new
Runnable() {
            public void run() {
                ExitDialog dialog = new
ExitDialog(new javax.swing.JFrame(), true);
                dialog.addWindowListener(new
java.awt.event.WindowAdapter() {
                    public void
windowClosing(java.awt.event.WindowEvent e) {
                        System.exit(0);
                    }
                });
                dialog.setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPasswordField
jPasswordField1;
    private javax.swing.JTextField jTextField1;
    // End of variables declaration

}
```

```
package remote_and_monitoring_b.messanger.Client;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.net.*;
import java.io.*;

public class ChatClientGUI extends JFrame implements
ActionListener {

    JLabel header;
    JButton send, clear, connect, disconnect;
    JTextField ip, port, username;
    Socket S;
    threadrecieve tr;
    JTextArea msg = new JTextArea(), main = new
JTextArea();

    public ChatClientGUI() {

        setTitle("ChatClient");

        JPanel panel = new JPanel();
        JPanel msgP = new JPanel(new BorderLayout());
        JPanel mains = new JPanel(new FlowLayout());
        JPanel Head = new JPanel(new BorderLayout());
        JPanel sends = new JPanel(new
BorderLayout());
        JScrollPane mainscroll = new
JScrollPane(main);
        JScrollPane msgscroll = new JScrollPane(msg);

panel.setBorder(BorderFactory.createEmptyBorder(5, 5,
5, 5));
        panel.setLayout(new BorderLayout());
        main.setLineWrap(true);
        msg.setLineWrap(true);
        mains.add(new JLabel("Host"));
        mains.add(ip = new JTextField(10));
        mains.add(new JLabel("Port"));
        mains.add(port = new JTextField(5));
        mains.add(new JLabel("Username:"));
        mains.add(username = new JTextField(5));
        port.setText("27000");
        port.setEditable(false);
        mains.add(connect = new JButton("Connect"));
        mains.add(disconnect = new
JButton("Disconnect"));
        Head.add(mains, BorderLayout.SOUTH);
        Head.add(header = new JLabel("Chat Client"),
BorderLayout.NORTH);
        panel.add(Head, BorderLayout.NORTH);
        msgP.add(mainscroll, BorderLayout.CENTER);
        panel.add(msgP, BorderLayout.CENTER);

sends.setBorder(BorderFactory.createEmptyBorder(5, 5,
5, 5));
        sends.add(msgscroll, BorderLayout.CENTER);
        sends.add(send = new JButton("Send"),
BorderLayout.EAST);
        panel.add(sends, BorderLayout.SOUTH);

        sends.setPreferredSize(new Dimension(100,
100));

        main.setEditable(false);
        disconnect.setEnabled(false);
        connect.addActionListener(this);
        send.addActionListener(this);
        disconnect.addActionListener(this);

        addWindowListener(new WindowAdapter() {

       /*     public void windowClosing(WindowEvent
we) {
                try {
                     PrintWriter pw = new
PrintWriter(S.getOutputStream());
                     pw.println("0x007");
                     pw.flush();

                connect.setEnabled(true);
                disconnect.setEnabled(false);
                username.setEditable(true);
                ip.setEditable(true);
                } catch (Exception E) {
                }
            } */
        });
        add(panel);

        setSize(640, 480);
        this.setResizable(false);

setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        setLocationRelativeTo(null);
        setVisible(true);


    }

    public void actionPerformed(ActionEvent ae) {
        String str = ae.getActionCommand();

        if (str.equals("Connect") &&
username.getText().length() >= 1) {

            try {
                S = new Socket(ip.getText(), 27000);
                PrintWriter pw = new
PrintWriter(S.getOutputStream());
                pw.println(username.getText());
                pw.flush();
                BufferedReader br = new
BufferedReader(new
InputStreamReader(S.getInputStream()));
                String ms = br.readLine();
                if (ms.equals("ok")) {
                    tr = new threadrecieve(S);
                    tr.start();
                    connect.setEnabled(false);
                    disconnect.setEnabled(true);
                    username.setEditable(false);
                    ip.setEditable(false);
                } else {
                    S.close();

JOptionPane.showMessageDialog(null, "Username already
in use", "Error", JOptionPane.ERROR_MESSAGE);
                }

            } catch (Exception E) {
                JOptionPane.showMessageDialog(null,
E.toString(), "Error", JOptionPane.ERROR_MESSAGE);
            }
        }

        if (str.equals("Disconnect")) {
            try {

                PrintWriter pw = new
PrintWriter(S.getOutputStream());
                pw.println("0x007");
                pw.flush();
                connect.setEnabled(true);
                disconnect.setEnabled(false);
                username.setEditable(true);
                ip.setEditable(true);
            } catch (Exception E) {
            }
        }

        if (str.equals("Send")) {
            try {
                PrintWriter pw = new
PrintWriter(S.getOutputStream());
                pw.println(msg.getText());
                pw.flush();
                msg.setText("");
            } catch (Exception E) {
            }
        }
    }

    public class threadrecieve extends Thread {

        Socket S;
        String msg;
        BufferedReader br;

        threadrecieve(Socket r) {
            S = r;
        }

        public void run() {
            try {
                System.out.println("hello");
                br = new BufferedReader(new
InputStreamReader(S.getInputStream()));
```

```
                while ((msg = br.readLine()) != null)
{
                    System.out.println(msg);

ChatClientGUI.this.main.append(msg + "\n");
                    if (msg.equals("Server is
Shutting Down")) {
                        username.setEditable(true);
                        ip.setEditable(true);
                        disconnect.setEnabled(false);
                        connect.setEnabled(true);
                        break;
                    }

                }
                S.close();
                br.close();
            } catch (Exception E) {
            }
        }
    }

    public static void main(String[] args) {
        new ChatClientGUI();
    }
}
```

```
package remote_and_monitoring_b.messanger.Server;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.net.*;
import java.io.*;
import java.util.*;

public class ChatServerGUI extends JFrame implements
ActionListener {

    private static final long serialVersionUID =
7526472295622776147L;
    JLabel header;
    JButton send, clear, start, stop;
    JTextField port;
    JList userListBox;
    DefaultListModel userList;
    threadserver ts;
    int no_of_users;
    boolean serverState = false;
    String com;
    Vector<Socket> SocketClients = new
Vector<Socket>();
    JTextArea msg = new JTextArea(), main = new
JTextArea();
    ServerSocket SS;
    Socket C;


    public ChatServerGUI() {

        setTitle("ChatServer");

        JPanel panel = new JPanel();
        JPanel msgP = new JPanel(new BorderLayout());
        JPanel mains = new JPanel(new FlowLayout());
        JPanel Head = new JPanel(new BorderLayout());
        JPanel sends = new JPanel(new
BorderLayout());
        JPanel userPane = new JPanel(new
BorderLayout());
        userList = new DefaultListModel();
        userListBox = new JList(userList);

        JPanel msgbuttons = new JPanel(new
GridLayout(0, 2, 5, 5));
        JScrollPane mainscroll = new
JScrollPane(main);
        JScrollPane msgscroll = new JScrollPane(msg);

        userPane.add(new JScrollPane(userListBox),
BorderLayout.CENTER);
        userPane.add(new JLabel("USER LIST",
JLabel.CENTER), BorderLayout.NORTH);

panel.setBorder(BorderFactory.createEmptyBorder(5, 5,
5, 5));
        panel.setLayout(new BorderLayout());

        panel.add(userPane, BorderLayout.EAST);
        main.setLineWrap(true);
        msg.setLineWrap(true);
        mains.add(new JLabel("Port"));
        mains.add(port = new JTextField(5));
        mains.add(start = new
JButton("StartService"));
        mains.add(stop = new JButton("StopService"));
        stop.setEnabled(false);
        port.setText("27000");
        port.setEditable(false);
        Head.add(mains, BorderLayout.SOUTH);
        Head.add(header = new JLabel("Chat Server"),
BorderLayout.NORTH);
        panel.add(Head, BorderLayout.NORTH);
        msgP.add(mainscroll, BorderLayout.CENTER);
        panel.add(msgP, BorderLayout.CENTER);

sends.setBorder(BorderFactory.createEmptyBorder(5, 5,
5, 5));
        sends.add(msgscroll, BorderLayout.CENTER);
        sends.add(msgbuttons, BorderLayout.EAST);
        msgbuttons.add(send = new JButton("Send"));

        msgbuttons.add(clear = new JButton("Clear"));

        panel.add(sends, BorderLayout.SOUTH);

        sends.setPreferredSize(new Dimension(50,
50));
        main.setEditable(false);

        add(panel);

setDefaultCloseOperation(javax.swing.WindowConstants.
DO_NOTHING_ON_CLOSE);
        setSize(640, 480);
        this.setResizable(false);
        start.addActionListener(this);
        stop.addActionListener(this);
        setLocationRelativeTo(null);
        addWindowListener(new WindowAdapter() {

            @Override
            public void windowClosing(WindowEvent we)
{
                try {
                    Socket dummy;
                    PrintWriter pw;
                    serverState = false;
                    main.setText(main.getText() +
"\n===SERVER CLOSED===\n");
                    for (int i = 0; i <
SocketClients.size(); i++) {
                        dummy = (Socket)
SocketClients.get(i);
                        pw = new
PrintWriter(dummy.getOutputStream());
                        pw.println("Server is
Shutting Down");
                        pw.flush();
                    }
                    userList.clear();
                    SocketClients.clear();

                    SS.close();
                    start.setEnabled(true);
                    stop.setEnabled(false);
                } catch (Exception E) {
                }
            }
        });
        setVisible(true);
        try {
            Thread.sleep(1000);
            main.append("Welcome.....");
            main.append("please wait...\n");
            Thread.sleep(2000);
            main.append("Click StartService to Start
Listening on port 27000\n");
        } catch (Exception E) {
        }

        send.addActionListener(this);
    }

    public void actionPerformed(ActionEvent ae) {
        String str = ae.getActionCommand();

        if (str.equals("StartService")) {
            try {
                JOptionPane.showMessageDialog(null,
```

```
"Running!", "Ready to accept new clients",
JOptionPane.INFORMATION_MESSAGE);
                serverState = true;
                SS = new ServerSocket(27000);
                ts = new threadserver(SS);
                ts.start();
                stop.setEnabled(true);
                start.setEnabled(false);
            } catch (Exception E) {
            }
        }

        if (str.equals("StopService")) {
            try {
                Socket dummy;
                PrintWriter pw;
                serverState = false;
                main.setText(main.getText() +
"\n===SERVER CLOSED===\n");
                for (int i = 0; i <
SocketClients.size(); i++) {
                    dummy = (Socket)
SocketClients.get(i);
                    pw = new
PrintWriter(dummy.getOutputStream());
                    pw.println("Server is Shutting
Down");
                    pw.flush();
                }

                userList.clear();
                SocketClients.clear();

                SS.close();
                start.setEnabled(true);
                stop.setEnabled(false);

            } catch (Exception E) {
            }
        }
        if (str.equals("Send")) {
            PrintWriter pw;

            try {
                Socket dummy;
                for (int i = 0; i <
SocketClients.size(); i++) {
                    dummy = (Socket)
SocketClients.get(i);
                    pw = new
PrintWriter(dummy.getOutputStream());
                    pw.println(">>>>>>>>Network
Admin:" + msg.getText());
                    main.append(">>>>>>>>Network
Admin:" + msg.getText() + "\n");
                    msg.setText("");
                    pw.flush();
                }
            } catch (Exception E) {
            }
        }
    }

    public class threadserver extends Thread {

        ServerSocket s;
        Socket C;

        threadserver(ServerSocket k) {
            s = k;
        }

        @Override
        public void run() {
            boolean proceed = true;
            no_of_users = 0;
            try {

ChatServerGUI.this.main.append(">Ready to accept Chat
Clients\n");
                while (true) {
                    C = s.accept();
                    no_of_users++;
                    BufferedReader br = new
BufferedReader(new
InputStreamReader(C.getInputStream()));
                    PrintWriter pw = new
PrintWriter(C.getOutputStream());
                    String ms = br.readLine();
                    for (int i = 0; i <
userList.size(); i++) {
                        if
(ms.equalsIgnoreCase((String) userList.get(i))) {
                            pw.println("not ok");
                            pw.flush();
                            C.close();
                            proceed = false;
                            break;
                        }
                        proceed = true;
                    }

                    if (proceed) {
                        pw.println("ok");
                        pw.flush();
                        SocketClients.add(C);
                        threadclient client;
                        client = new threadclient(C);
                        client.setName(ms);
                        client.start();
                    }
                }
            } catch (Exception E) {
            }
        }
    }

    public class threadclient extends Thread {

        Socket c;
        Socket dummy;
        String msg;
        BufferedReader br;
        PrintWriter pw;
        boolean clientStatus;

        threadclient(Socket S) {
            c = S;
        }

        @Override
        public void run() {
            try {

                System.out.println("texting");
                br = new BufferedReader(new
InputStreamReader(c.getInputStream()));
                pw = new
PrintWriter(c.getOutputStream());
                pw.println("Welcome...");
                pw.println("Your Username:" +
this.getName());
                pw.flush();

                for (int j = 0; j <
SocketClients.size(); j++) {
                    dummy = (Socket)
SocketClients.get(j);
                    pw = new
PrintWriter(dummy.getOutputStream());
                    pw.println(this.getName() + " has
joined the chat");
                    pw.flush();

                }
                System.out.println(c);
                userList.addElement((String)
this.getName()+ "@" + dummy);
                while
(ChatServerGUI.this.serverState) {
                    if ((msg = br.readLine()) !=
null) {

                        if (msg.equals("0x007")) {
                            pw.println("You have been
Disconnected from the Chat Server");
                            pw.flush();
                            com = this.getName();

SocketClients.removeElement(c);
                            for (int i = 0; i <
SocketClients.size(); i++) {

                                dummy = (Socket)
SocketClients.get(i);

                                pw = new
PrintWriter(dummy.getOutputStream());
```

```
pw.println(ChatServerGUI.this.com + " has left the
chat");
                                pw.flush();
                            }
                            break;
                        } else if
(msg.equals("**users")) {

pw.println("\n****USERS****");
                            for (int i = 0; i <
userList.size(); i++) {
                                pw.println((String)
userList.get(i));
                            }
                            pw.println();
                            pw.flush();
                        } else {
                            System.out.println(msg);
                            msg = this.getName() + ":
" + msg;
                            System.out.println(msg);

ChatServerGUI.this.main.append(msg + "\n");
                            for (int i = 0; i <
ChatServerGUI.this.SocketClients.size(); i++) {
                                dummy = (Socket)
SocketClients.get(i);
                                pw = new
PrintWriter(dummy.getOutputStream());
                                pw.println(msg);

//System.out.println(msg);
                                pw.flush();
                            }
                        }
                    }
                }
                System.out.println("exiting " +
this.getName());
                c.close();
                br.close();

                ChatServerGUI.this.main.append(">>" +
this.getName() + " has left the chat\n");
                for (int j = 0; j <
SocketClients.size(); j++) {

System.out.println(SocketClients.get(j));
                }

userList.removeElement(this.getName()+"@"+dummy);
            } catch (Exception E) {
                E.printStackTrace();
            }
        }
    }

    public static void main(String[] args) {
        new ChatServerGUI();
    }
}
```

```
" + ioe.getMessage());
        }
        //String line = "";
        String line =
JOptionPane.showInputDialog("Input Pesan: ");
            try {
                //line = console.readLine();
                streamOut.writeUTF(line);
                streamOut.flush();
                console.close();
                streamOut.close();
                socket.close();
            } catch (IOException ioe) {
                System.out.println("Sending error: "
+ ioe.getMessage());
            }
    }

    public void start() throws IOException {
        console = new DataInputStream(System.in);
        streamOut = new
DataOutputStream(socket.getOutputStream());
    }

    public void stop() {
        try {
            if (console != null) {
                console.close();
            }
            if (streamOut != null) {
                streamOut.close();
            }
            if (socket != null) {
                socket.close();
            }
        } catch (IOException ioe) {
            System.out.println("Error closing ...");
        }
    }

  /*  public static void main(String args[]) {
        ChatClient client = null;
        if (args.length != 2) {
            System.out.println("Usage: java
ChatClient host port");
        } else {
            client = new ChatClient(args[0],
Integer.parseInt(args[1]));
        }
    } */

        public static void main(String args[]) {
   //     ChatServer a = new ChatServer();

        String ip =
JOptionPane.showInputDialog("Input IP Adrress client:
");

        new ChatClient(ip, 1234);

    }
}
```

```
package remote_and_monitoring_b.messanger;

import java.net.*;
import java.io.*;
import java.util.Scanner;
import javax.swing.JOptionPane;

public class ChatClient {

    private Socket socket = null;
    private DataInputStream console = null;
    private DataOutputStream streamOut = null;

    public ChatClient(String serverName, int
serverPort) {
        System.out.println("Establishing connection.
Please wait ...");
        try {
            socket = new Socket(serverName,
serverPort);
            System.out.println("Connected: " +
socket);
            start();
        } catch (UnknownHostException uhe) {
            System.out.println("Host unknown: " +
uhe.getMessage());
        } catch (IOException ioe) {
            System.out.println("Unexpected exception:
```

```
package remote_and_monitoring_b.messanger;

/**
 *
 * @author Agus
 */
import java.net.*;
import java.io.*;
import java.util.Scanner;
import javax.swing.JOptionPane;

public class ChatServer implements Runnable {

    private Socket socket = null;
    private ServerSocket server = null;
    private Thread thread = null;
    private DataInputStream streamIn = null;

    public ChatServer(int port) {
        try {
            System.out.println("Binding to port " +
port + ", please wait  ...");
            server = new ServerSocket(port);
            System.out.println("Server started: " +
server);
            start();
        } catch (IOException ioe) {
```

```
            System.out.println(ioe);
        }
    }

    public void run() {
        while (thread != null) {
            try {
                System.out.println("Waiting for a
client ...");
                socket = server.accept();
                System.out.println("Client accepted:
" + socket);
                open();
                boolean done = false;
                while (!done) {
                    try {
                        String line =
streamIn.readUTF();
                        System.out.println(line);

JOptionPane.showMessageDialog(null, line);
                        done = line.equals(".bye");
                    } catch (IOException ioe) {
                        done = true;
                    }
                }
                close();
            } catch (IOException ie) {
                System.out.println("Acceptance Error:
" + ie);
            }
        }
    }

    public void start() {
        if (thread == null) {
            thread = new Thread(this);
            thread.start();
        }
    }

    public void stop() {
        if (thread != null) {
            thread.stop();
            thread = null;
        }
    }

    public void open() throws IOException {
        streamIn = new DataInputStream(new
BufferedInputStream(socket.getInputStream()));
    }

    public void close() throws IOException {
        if (socket != null) {
            socket.close();
        }
        if (streamIn != null) {
            streamIn.close();
        }
    }

 /*   public static void main(String args[]) {
        ChatServer server = null;
        if (args.length != 1) {
            System.out.println("Usage: java
ChatServer port");
        } else {
            server = new
ChatServer(Integer.parseInt(args[0]));
        }
    } */

    public static void main(String args[]) {
          new ChatServer(1234);
    }
}
```

```
package remote_and_monitoring_b.remoter;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.Properties;

import remote_and_monitoring_b.main;

public class Config {

    public static String server_address =
"127.0.0.1";
    public static int server_port = 6666;

    public static void loadConfiguration() {
        if (new
File(main.VIEWER_CONFIG_FILE).canRead())
            try {
                Properties properties = new
Properties();
                properties.load(new
FileInputStream(main.VIEWER_CONFIG_FILE));

                server_address =
properties.get("server-address").toString();
                server_port =
Integer.valueOf(properties.get("server-
port").toString());
            }
            catch (Exception e) {
                e.getStackTrace();
            }
        else
            storeConfiguration();
    }

    public static void storeConfiguration () {
        try {
            new
File(main.VIEWER_CONFIG_FILE).createNewFile();
            Properties properties = new Properties();
            properties.put("server-address",
server_address);
            properties.put("server-port",
String.valueOf(server_port));

            properties.store(new
FileOutputStream(main.VIEWER_CONFIG_FILE),
                "remote_and_monitoring viewer
configuration file");
        } catch (Exception e) {
            e.getStackTrace();
        }
    }

    public static void SetConfiguration(String
Address, int Port) {
        server_address = Address;
        server_port = Port;

        storeConfiguration();
    }
}
```

```
package remote_and_monitoring_b.remoter;

import javax.swing.ImageIcon;

import remote_and_monitoring_b.main;
import remote_and_monitoring_b.remoter.rmi.Viewer;

public class ConnectionDialog extends
javax.swing.JFrame {

    /** Creates new form ServerConnection */
    public ConnectionDialog() {
        initComponents();
        Config.loadConfiguration();


jTextFieldIPAdr.setText(Config.server_address);

jTextFieldPort.setText(String.valueOf(Config.server_p
ort));
    }

    /** This method is called from within the
constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content
of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed"
desc="Generated Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
```

```
        jTextFieldIPAdr = new
javax.swing.JTextField();
        jTextFieldPort = new
javax.swing.JTextField();
        jButtonCancel = new javax.swing.JButton();
        jButtonOK = new javax.swing.JButton();


setDefaultCloseOperation(javax.swing.WindowConstants.
DISPOSE_ON_CLOSE);
        setTitle("Connection details");
        setIconImage(new
ImageIcon(main.IDLE_ICON).getImage());
        setResizable(false);

        jLabel1.setText("Server Address:");

        jLabel2.setText("Server Port:");

        jTextFieldIPAdr.setText("127.0.0.1");

        jTextFieldPort.setText("6666");
        jTextFieldPort.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jTextFieldPortActionPerformed(evt);
            }
        });

        jButtonCancel.setText("Cancel");
        jButtonCancel.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonCancelActionPerformed(evt);
            }
        });

        jButtonOK.setText("OK");
        jButtonOK.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonOKActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel1)
                            .addComponent(jLabel2))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.RELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.LEADING)

.addComponent(jTextFieldPort,
javax.swing.GroupLayout.DEFAULT_SIZE, 170,
Short.MAX_VALUE)

.addComponent(jTextFieldIPAdr,
javax.swing.GroupLayout.DEFAULT_SIZE, 170,
Short.MAX_VALUE)))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                        .addComponent(jButtonOK)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.RELATED)
.addComponent(jButtonCancel)))
                .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(jTextFieldIPAdr,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.RELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(jTextFieldPort,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.RELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.BASELINE)
                    .addComponent(jButtonCancel)
                    .addComponent(jButtonOK))

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE
, Short.MAX_VALUE))
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(22,
Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE
, Short.MAX_VALUE)
                .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(77, 77, 77))
        );

        java.awt.Dimension screenSize =
java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-320)/2,
(screenSize.height-152)/2, 320, 152);
    }// </editor-fold>

    private void
jButtonCancelActionPerformed(java.awt.event.ActionEve
nt evt) {
        dispose();
    }

    private void
jButtonOKActionPerformed(java.awt.event.ActionEvent
evt) {
```

```
        dispose();
        main.startViewer(jTextFieldIPAdr.getText(),

Integer.parseInt(jTextFieldPort.getText()));
    }

    private void
jTextFieldPortActionPerformed(java.awt.event.ActionEv
ent evt) {
        // TODO add your handling code here:
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new
Runnable() {
            public void run() {
                new
ConnectionDialog().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButtonCancel;
    private javax.swing.JButton jButtonOK;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JTextField jTextFieldIPAdr;
    private javax.swing.JTextField jTextFieldPort;
    // End of variables declaration

}
```

```
package remote_and_monitoring_b.remoter;

import java.awt.Rectangle;
import java.util.ArrayList;

import remote_and_monitoring_b.remoter.rmi.Viewer;

public class Recorder extends Thread {

    private boolean recording = false;

    public Viewer viewer;
    public ViewerGUI viewerGUI;
    public ScreenPlayer screenPlayer;

    public Recorder(Viewer viewer) {
        this.viewer = viewer;
        start();

        screenPlayer = new ScreenPlayer(this);
        viewerGUI = new ViewerGUI(this);
    }

    @Override
    public void run()
    {
        while (true) {
            Wait();

            while (recording) {
                viewer.sendData();
                viewer.recieveData();
            }
        }
    }

    public void Wait() {
        try {
            synchronized(this) {
                wait();
            }
        }
        catch (Exception e) {
            e.getStackTrace();
        }
    }

    public void Notify() {
        try {
            synchronized(this){
                notify();
            }
        }
        catch (Exception e) {
            e.getStackTrace();
        }
    }

    public void Stop() {
        recording = false;
        screenPlayer.removeAdapters();
        screenPlayer.clearScreen();
        viewer.disconnect();
    }

    public void Start() {
        if (!viewer.isConnected())
            if (viewer.connect() == -1) return;
        screenPlayer.addAdapters();
        recording = true;
        Notify();
    }

    public boolean isRecording () {
        return recording;
    }

    public void updateData(ArrayList objects){
        screenPlayer.UpdateScreen((byte[])
objects.get(0));
        screenPlayer.setScteenRect((Rectangle)
objects.get(1));
    }
}
```

```
package remote_and_monitoring_b.remoter;

import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.Rectangle;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionAdapter;
import java.awt.event.MouseWheelEvent;
import java.awt.event.MouseWheelListener;
import javax.swing.JLabel;

import remote_and_monitoring_b.utility.ImageUtility;

public class ScreenPlayer extends JLabel {

    private Recorder recorder;

    private Image img;

    private Rectangle screenRect = new Rectangle(0,
0, 0, 0);
    private Rectangle oldScreenRect = new Rectangle(-
1, -1, -1, -1);

    private KeyAdapter keyAdapter;
    private MouseAdapter mouseAdapter;
    private MouseWheelListener mouseWheelListener;
    private MouseMotionAdapter mouseMotionAdapter;

    public ScreenPlayer(Recorder recorder) {
        this.recorder = recorder;

        keyAdapter = new KeyAdapter() {
            @Override
            public void keyPressed(KeyEvent e){

ScreenPlayer.this.recorder.viewer.AddObject(e);
            }

            @Override
            public void keyReleased(KeyEvent e){

ScreenPlayer.this.recorder.viewer.AddObject(e);
            }
        };

        mouseWheelListener = new MouseWheelListener()
{
            @Override
            public void
mouseWheelMoved(MouseWheelEvent e) {

ScreenPlayer.this.recorder.viewer.AddObject(e);
            }
```

```
        };

        mouseMotionAdapter = new MouseMotionAdapter()
{
            @Override
            public void mouseMoved(MouseEvent e) {

ScreenPlayer.this.recorder.viewer.AddObject(e);
            }

            @Override
            public void mouseDragged(MouseEvent e) {

ScreenPlayer.this.recorder.viewer.AddObject(e);
            }
        };

        mouseAdapter = new MouseAdapter()
        {
            @Override
            public void mousePressed(MouseEvent e) {

ScreenPlayer.this.recorder.viewer.AddObject(e);
            }

            @Override
            public void mouseReleased(MouseEvent e) {

ScreenPlayer.this.recorder.viewer.AddObject(e);
            }
        };

        setFocusable(true);
    };

    public void addAdapters() {
        addKeyListener(keyAdapter);
        addMouseWheelListener(mouseWheelListener);
        addMouseMotionListener(mouseMotionAdapter);
        addMouseListener(mouseAdapter);
    }

    public void removeAdapters() {
        removeKeyListener(keyAdapter);
        removeMouseWheelListener(mouseWheelListener);

removeMouseMotionListener(mouseMotionAdapter);
        removeMouseListener(mouseAdapter);
    }

    @Override
    public void paint(Graphics g) {
        g.drawImage(img, 0, 0, screenRect.width,
screenRect.height, this);
    }

    public void setScteenRect(Rectangle rect) {
        screenRect = rect;
    }

    public void UpdateScreen(byte[] data) {
        if (!screenRect.equals(oldScreenRect)) {
               oldScreenRect = screenRect;
               setSize(screenRect.getSize());

setPreferredSize(screenRect.getSize());
        }

        img = ImageUtility.read(data);
        repaint();
    }

    public void clearScreen() {
        setSize(new Dimension(1, 1));
        setPreferredSize(new Dimension(1, 1));
        img = createImage(getWidth(), getHeight());
        repaint();
        oldScreenRect = new Rectangle(-1, -1, -1, -
1);
    }

    public boolean isScreenRectChanged () {
        boolean bool =
(!screenRect.equals(oldScreenRect));
        oldScreenRect = screenRect;
        return bool;
    }
}
```

```
package remote_and_monitoring_b.remoter;

import java.awt.event.WindowEvent;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;

import remote_and_monitoring_b.SysTrayViewer;
import remote_and_monitoring_b.main;

public class ViewerGUI extends javax.swing.JFrame {

    private Recorder recorder;

    /** Creates new form MainFrame */
    public ViewerGUI(Recorder recorder) {
        this.recorder = recorder;
        initComponents();

jScrollPane1.setViewportView(recorder.screenPlayer);
        setVisible(true);
    }

    public void Start() {
        if (recorder.isRecording())
            recorder.Stop();
        else
            recorder.Start();

        if (recorder.isRecording()) {
            setIconImage(new
ImageIcon(main.ALIVE_ICON).getImage());
            setTitle("remote_and_monitoring Viewer ["
+Config.server_address + "]");
        } else {
            setTitle("remote_and_monitoring Viewer");
            setIconImage(new
ImageIcon(main.WAIT_ICON).getImage());
        }
    }

    /** This method is called from within the
constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content
of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed"
desc="Generated Code">
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();


setDefaultCloseOperation(javax.swing.WindowConstants.
DO_NOTHING_ON_CLOSE);
        setTitle("Remote and Monitoring Viewer");
        setIconImage(new
ImageIcon(main.WAIT_ICON).getImage());
        addWindowListener(new
java.awt.event.WindowAdapter() {
            public void
windowClosing(java.awt.event.WindowEvent evt) {
                formWindowClosing(evt);
            }
        });

        jScrollPane1.setFocusable(false);

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 884,
Short.MAX_VALUE)
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addComponent(jScrollPane1,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 462,
Short.MAX_VALUE)
        );

        java.awt.Dimension screenSize =
```

```
java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-900)/2,
(screenSize.height-500)/2, 900, 500);
    }// </editor-fold>

    private void
formWindowClosing(java.awt.event.WindowEvent evt) {
        if (evt.getID() ==
WindowEvent.WINDOW_CLOSING) {
            if (JOptionPane.showConfirmDialog(null,
"Exit Viewer ?", "Confirm Dialog",
                JOptionPane.OK_CANCEL_OPTION) ==
JOptionPane.OK_OPTION) {
                if (recorder.isRecording())
                    recorder.viewer.Stop();
                    if (SysTrayViewer.isSupported())
                        dispose();
                    else
                        main.exit();
            }
        }
        else
            super.processWindowEvent(evt);
    }


    // Variables declaration - do not modify
    private javax.swing.JScrollPane jScrollPane1;
    // End of variables declaration

}
```

```
package remote_and_monitoring_b.remoter.rmi;

import
remote_and_monitoring_b.server.rmi.ServerInterface;
import remote_and_monitoring_b.remoter.Recorder;
import
remote_and_monitoring_b.utility.InetAdrUtility;
import remote_and_monitoring_b.utility.ZipUtility;
import remote_and_monitoring_b.remoter.Config;

import java.io.IOException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;
import javax.swing.JOptionPane;

public class Viewer extends Thread {

    private int index = -1;
    private Recorder recorder;

    private Registry registry;
    private ServerInterface rmiServer;

    private String server = "127.0.0.1";
    private int port = 6666;

    private boolean connected = false;

    private ArrayList<Object> Objects;

    public Viewer () {
        Config.loadConfiguration();
        server = Config.server_address;
        port = Config.server_port;
    }

    public boolean isConnected() {
        return connected;
    }

    public void Start() {
        connect();
        if (connected) {
            recorder = new Recorder(this);
            recorder.viewerGUI.Start();
        }
        else Stop();
    }

    public void Stop() {
        if (recorder != null) {
            recorder.Stop();
            recorder.interrupt();
        }
        disconnect();
        interrupt();
    }

    public int connect() {
        connected = false;

        try {
                registry =
LocateRegistry.getRegistry(server, port);

            rmiServer = (ServerInterface)
registry.lookup("ServerImpl");

            index =
rmiServer.startViewer(InetAdrUtility.getLocalAdr());

            displayStatus();
            Objects = new ArrayList<Object>();
            connected = true;
            return index;
       } catch (Exception e) {
            JOptionPane.showMessageDialog(null,
e.getMessage(), "Error !!",
                    JOptionPane.ERROR_MESSAGE);
            return -1;
      }
    }

    public void disconnect() {
        connected = false;
        try {
            if (rmiServer != null) {
                    rmiServer.stopViewer(index);

UnicastRemoteObject.unexportObject(rmiServer, true);
            }
        }
        catch (Exception e) {
            e.getStackTrace();
        }
      rmiServer = null;
      registry = null;
    }

    public void updateData(Object object) {
        byte[] data;
        try {
             data =
ZipUtility.objecttoByteArray(object);

            updateData(data);
        }
        catch (IOException e) {
            e.getStackTrace();
        }
    }

    public void updateData(byte[] data) {
        try {rmiServer.updateData(data, index);}
        catch (Exception re) {
            re.getStackTrace();
        }
    }

    public void AddObject(Object object) {
        Objects.add(object);
    }

    public void sendData() {
        ArrayList SendObjects;
        synchronized(Objects){

            SendObjects = Objects;
            Objects = new ArrayList<Object>();
        }
        updateData(SendObjects);
    }

    public void recieveData() {
        Object object = null;
        try {
            byte[] data =
rmiServer.updateData(index);
            object =
ZipUtility.byteArraytoObject(data);

            recorder.updateData((ArrayList) object);
        }
        catch (Exception e) {
            e.getStackTrace();
        }
```

```
    }

    public void displayStatus() {
        System.out.println("Viewer connected to " +
rmiServer);
    }
}
```

```
package remote_and_monitoring_b.server;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.Properties;

import remote_and_monitoring_b.main;
import
remote_and_monitoring_b.utility.InetAdrUtility;

public class Config {

    public static String server_address =
"127.0.0.1";
    public static int server_port = 6666;
    public static boolean default_address = false;

    public static void loadConfiguration() {
        if (new
File(main.SERVER_CONFIG_FILE).canRead())
            try {
                Properties properties = new
Properties();
                properties.load(new
FileInputStream(main.SERVER_CONFIG_FILE));

                server_address =
properties.get("server-address").toString();
                server_port =
Integer.valueOf(properties.get("server-
port").toString());

                default_address =
Boolean.valueOf(properties.getProperty("default-
address"));
            }
            catch (Exception e) {
                e.getStackTrace();
            }
       else
            storeConfiguration();
    }

    public static void storeConfiguration () {
        try {
            new
File(main.SERVER_CONFIG_FILE).createNewFile();
            Properties properties = new Properties();
            properties.put("server-address",
server_address);
            properties.put("server-port",
String.valueOf(server_port));
            properties.put("default-address",
String.valueOf(default_address));

            properties.store(new
FileOutputStream(main.SERVER_CONFIG_FILE),
                "remote_and_monitoring server
configuration file");
        } catch (Exception e) {
            e.getStackTrace();
        }
    }

    public static void SetConfiguration(int Port) {
        server_address =
InetAdrUtility.getLocalAdr().getHostAddress();
        server_port = Port;

        storeConfiguration();
    }
}
```

```
package remote_and_monitoring_b.server;

import javax.swing.ImageIcon;
import remote_and_monitoring_b.main;

import
remote_and_monitoring_b.utility.InetAdrUtility;
```

```
public class ConfigGUI extends javax.swing.JFrame {

    /** Creates new form ServerConnection */
    public ConfigGUI() {
        initComponents();
        Config.loadConfiguration();


jComboBoxLocalAdrs.setSelectedItem(Config.server_addr
ess);

jTextFieldPort.setText(String.valueOf(Config.server_p
ort));


jCheckBoxDefaultAdr.setSelected(Config.default_addres
s);
    }

    /** This method is called from within the
constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content
of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed"
desc="Generated Code">
    private void initComponents() {

        jButtonCancel = new javax.swing.JButton();
        jButtonOK = new javax.swing.JButton();
        jPanel2 = new javax.swing.JPanel();
        jComboBoxLocalAdrs = new
javax.swing.JComboBox(InetAdrUtility.getLocalIPAdress
es());
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jTextFieldPort = new
javax.swing.JTextField();
        jCheckBoxDefaultAdr = new
javax.swing.JCheckBox();


setDefaultCloseOperation(javax.swing.WindowConstants.
DISPOSE_ON_CLOSE);
        setTitle("Server configuration");
        setAlwaysOnTop(true);
        setIconImage(new
ImageIcon(main.IDLE_ICON).getImage());
        setResizable(false);

        jButtonCancel.setText("Cancel");
        jButtonCancel.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonCancelActionPerformed(evt);
            }
        });

        jButtonOK.setText("OK");
        jButtonOK.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonOKActionPerformed(evt);
            }
        });


jPanel2.setBorder(javax.swing.BorderFactory.createTit
ledBorder("Server configuration"));

        jComboBoxLocalAdrs.setMaximumSize(new
java.awt.Dimension(28, 20));
        jComboBoxLocalAdrs.setMinimumSize(new
java.awt.Dimension(28, 20));

        jLabel1.setText("Address:");

        jLabel2.setText("Port:");

        jTextFieldPort.setText("6666");
        jTextFieldPort.setMaximumSize(new
java.awt.Dimension(28, 20));
        jTextFieldPort.setMinimumSize(new
java.awt.Dimension(28, 20));
        jTextFieldPort.setPreferredSize(new
java.awt.Dimension(28, 20));
```

```
        jCheckBoxDefaultAdr.setText("Default");

        javax.swing.GroupLayout jPanel2Layout = new
javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1)
                .addGap(13, 13, 13)

.addGroup(jPanel2Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addComponent(jCheckBoxDefaultAdr)
                        .addGap(18, 18, 18)
                        .addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.RELATED)
                        .addComponent(jTextFieldPort,
javax.swing.GroupLayout.DEFAULT_SIZE, 115,
Short.MAX_VALUE))
                    .addComponent(jComboBoxLocalAdrs,
0, 222, Short.MAX_VALUE))
                .addContainerGap())
        );
        jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addGroup(jPanel2Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(jComboBoxLocalAdrs,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.RELATED)

.addGroup(jPanel2Layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.BASELINE)

.addComponent(jCheckBoxDefaultAdr)
                    .addComponent(jLabel2)
                    .addComponent(jTextFieldPort,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.TRAILING)

.addGroup(layout.createSequentialGroup()
                        .addComponent(jButtonOK)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.RELATED)
                        .addComponent(jButtonCancel))
                    .addComponent(jPanel2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(29,
Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
```

```
layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPla
cement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.BASELINE)
                    .addComponent(jButtonCancel)
                    .addComponent(jButtonOK))

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE
, Short.MAX_VALUE))
        );

        java.awt.Dimension screenSize =
java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-365)/2,
(screenSize.height-168)/2, 365, 168);
    }// </editor-fold>

    private void
jButtonCancelActionPerformed(java.awt.event.ActionEve
nt evt) {
        dispose();
    }

    private void
jButtonOKActionPerformed(java.awt.event.ActionEvent
evt) {
        Config.server_address =
jComboBoxLocalAdrs.getSelectedItem().toString();
        Config.server_port =
Integer.valueOf(jTextFieldPort.getText());

        Config.default_address =
jCheckBoxDefaultAdr.isSelected();

        Config.storeConfiguration();
        dispose();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new
Runnable() {
            public void run() {
                new ConfigGUI().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButtonCancel;
    private javax.swing.JButton jButtonOK;
    private javax.swing.JCheckBox
jCheckBoxDefaultAdr;
    private javax.swing.JComboBox jComboBoxLocalAdrs;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JTextField jTextFieldPort;
    // End of variables declaration

}
```

```
package remote_and_monitoring_b.server;

/* @author Benyamin Bachir
 */

import java.awt.AWTException;
import java.awt.Rectangle;
import java.awt.Robot;
import java.awt.Toolkit;
import java.awt.event.InputEvent;
import java.awt.event.KeyEvent;
import java.awt.event.MouseEvent;
import java.awt.event.MouseWheelEvent;
import java.awt.image.BufferedImage;
import java.util.ArrayList;
```

```
import remote_and_monitoring_b.utility.ImageUtility;

public class robot {

    private Robot rt;
    private Toolkit tk = null;
    private Rectangle screenRect;

    public robot() {
        tk = Toolkit.getDefaultToolkit();
        screenRect = new
Rectangle(tk.getScreenSize());
        try {
            rt = new Robot();
        } catch (AWTException awte) {
            awte.getStackTrace();
        }
    }

    public BufferedImage captureScreen() {
        screenRect = new
Rectangle(tk.getScreenSize());
        return rt.createScreenCapture(screenRect);
    }

    public byte[] CaptureScreenByteArray() {
        return
ImageUtility.toByteArray(captureScreen());
    }

    public Rectangle getScreenRect() {
        return screenRect;
    }

    public void updateData(Object object) {
        ArrayList Objects = (ArrayList) object;
        for (int i = 0; i < Objects.size(); i++) {
            Object obj = Objects.get(i);

            if (obj instanceof MouseEvent) {
                applyMouseEvent((MouseEvent) obj);
            } else if (obj instanceof KeyEvent) {
                applyKeyEvent((KeyEvent) obj);
            }
        }
    }

    public void applyMouseEvent(MouseEvent evt) {
        rt.mouseMove(evt.getX(), evt.getY());
        int buttonMask = 0;
        int buttons = evt.getButton();
        if ((buttons == MouseEvent.BUTTON1)) {
            buttonMask = InputEvent.BUTTON1_MASK;
        }
        if ((buttons == MouseEvent.BUTTON2)) {
            buttonMask |= InputEvent.BUTTON2_MASK;
        }
        if ((buttons == MouseEvent.BUTTON3)) {
            buttonMask |= InputEvent.BUTTON3_MASK;
        }
        switch (evt.getID()) {
            case MouseEvent.MOUSE_PRESSED:
                rt.mousePress(buttonMask);
                break;
            case MouseEvent.MOUSE_RELEASED:
                rt.mouseRelease(buttonMask);
                break;
            case MouseEvent.MOUSE_WHEEL:
                rt.mouseWheel(
                        ((MouseWheelEvent)
evt).getUnitsToScroll());
                break;
        }
    }

    public void applyKeyEvent(KeyEvent evt) {
        switch (evt.getID()) {
            case KeyEvent.KEY_PRESSED:
                rt.keyPress(evt.getKeyCode());
                break;
            case KeyEvent.KEY_RELEASED:
                rt.keyRelease(evt.getKeyCode());
                break;
        }
    }
}
```

```
package remote_and_monitoring_b.server.rmi;
```

```
import java.io.IOException;
import java.net.InetAddress;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;
import java.util.Enumeration;
import java.util.Hashtable;
import javax.swing.JOptionPane;

import remote_and_monitoring_b.SysTrayViewer;
import remote_and_monitoring_b.server.Config;
import remote_and_monitoring_b.server.robot;
import remote_and_monitoring_b.utility.ZipUtility;

public class Server extends Thread {

    private static boolean idle = true;
    private static boolean running = false;

    private static Registry registry;
    private static ServerImpl serverImpl;

    private static robot rt = new robot();

    private static ArrayList<Object> Objects = new
ArrayList<Object>();
    public static Hashtable<Integer, InetAddress>
viewers = new Hashtable<Integer, InetAddress>();

    public static void Start() {
        idle = false;
        running = false;
        Config.loadConfiguration();

        if (Config.default_address)

System.setProperty("java.rmi.server.hostname",
                    Config.server_address);
        else

System.getProperties().remove("java.rmi.server.hostna
me");

        try{
            serverImpl = new ServerImpl();
            registry =
LocateRegistry.createRegistry(Config.server_port);
            registry.rebind("ServerImpl",
serverImpl);
        } catch (Exception e) {
            e.getStackTrace();
            Stop();

            JOptionPane.showMessageDialog(null,
e.getMessage(), "Error !!",
                    JOptionPane.ERROR_MESSAGE);
            return;
        }

        System.out.println(getStatus());
        running = true;

SysTrayViewer.updateServerStatus(SysTrayViewer.SERVER
_STARTED);
    }

    public static void Stop() {
        if (running) {
            running = false;
            disconnectAllViewers();

SysTrayViewer.updateServerStatus(SysTrayViewer.SERVER
_STOPPED);
        }
        else

SysTrayViewer.updateServerStatus(SysTrayViewer.CONNEC
TION_FAILED);
        try {
            if (registry != null)

UnicastRemoteObject.unexportObject(registry, true);
        } catch (Exception e) {
            e.getStackTrace();
        }
        registry = null;
        serverImpl = null;
    }
```

```
    public static boolean isRunning() {
        return running;
    }

    public static boolean isIdle() {
        return idle;
    }

    public static void updateData(byte[] data, int
index) {
        Object object;
        try {
            object =
ZipUtility.byteArraytoObject(data);
            rt.updateData(object);
        }
        catch (Exception e) {
            e.getStackTrace();
        }
    }

    public static byte[] updateData(int index) {
        byte[] data = null;
        Objects.add(rt.CaptureScreenByteArray());
        Objects.add(rt.getScreenRect());

        synchronized(Objects) {
            try {
                data =
ZipUtility.objecttoByteArray(Objects);
            } catch (IOException ioe) {
                ioe.getStackTrace();
            }
            Objects = new ArrayList<Object>();
        }

        return data;
    }

    public static void AddObject(Object object) {
        Objects.add(object);
    }

    public static synchronized int
addViewer(InetAddress inetAddress) {
        int index = viewers.size();
        viewers.put(index, inetAddress);


SysTrayViewer.displayViewer(inetAddress.toString(),
index, true);
        return index;
    }

    public static synchronized int removeViewer(int
index) {
        String viewer =
viewers.get(index).toString();

        viewers.remove(index);

        SysTrayViewer.displayViewer(viewer,
viewers.size(), false);
        return index;
    }

    public static void disconnectAllViewers() {
        Enumeration<Integer> viewerEnum =
viewers.keys();
        while (viewerEnum.hasMoreElements())
            removeViewer(viewerEnum.nextElement());
    }

    public static ArrayList<InetAddress>
getViewersAds () {
        ArrayList<InetAddress> viewersAds = new
ArrayList<InetAddress>();
        for (int i=0; i<viewers.size(); i++)
            viewersAds.add(viewers.get(i));
        return viewersAds;
    }

    public static int getViewersCount () {
        return viewers.size();
    }

    public static String getStatus() {
        String status = "Running ...at:\n\t" +
            Config.server_address + ":" +
Config.server_port;
        return status;
    }
}
```

```
package remote_and_monitoring_b.server.rmi;

import java.net.InetAddress;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class ServerImpl extends UnicastRemoteObject
implements ServerInterface {

    public ServerImpl () throws RemoteException {}

    @Override
    public int startViewer(InetAddress inetAddress)
throws RemoteException {
        return Server.addViewer(inetAddress);
    }

    @Override
    public void stopViewer(int index) throws
RemoteException {
        Server.removeViewer(index);
    }

    @Override
    public void updateData(byte[] data, int index) {
       Server.updateData(data, index);
    }

    @Override
    public byte[] updateData(int index) {
        return Server.updateData(index);
    }
}
```

```
package remote_and_monitoring_b.server.rmi;

import java.net.InetAddress;
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface ServerInterface extends Remote {
    public void updateData(byte[] data, int index)
throws RemoteException;
    public byte[] updateData(int index) throws
RemoteException;
    public void stopViewer(int index) throws
RemoteException;
    public int startViewer(InetAddress inetAddress)
throws RemoteException;
}
```

```
package remote_and_monitoring_b.utility;

/* @author Benyamin Bachir
 */

import java.awt.image.*;
import java.io.*;
import javax.imageio.*;

public class ImageUtility {

    static {ImageIO.setUseCache(false);}

    public static BufferedImage read(InputStream in)
throws IOException {
        BufferedImage image = null;
        image = ImageIO.read(in);
        if (image == null)
            throw new IOException("Read fails");
        return image;
    }

    public static BufferedImage read(byte[] bytes) {
        try {
            return read(new
ByteArrayInputStream(bytes));
        } catch (IOException e) {
            e.getStackTrace();
            return null;
        }
    }

    public static byte[] toByteArray(BufferedImage
image) {
        try {
```

```
            ByteArrayOutputStream out = new
ByteArrayOutputStream();
            ImageIO.write(image, "jpeg", out); //
write without compression
            return out.toByteArray();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

```
package remote_and_monitoring_b.utility;

/* @author Benyamin Bachir
 */

import java.net.Inet4Address;
import java.net.InetAddress;
import java.net.NetworkInterface;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.Enumeration;

public class InetAdrUtility {

   public static InetAddress getLocalAdr() {
        try{
            return (InetAddress.getLocalHost());
        }
        catch(UnknownHostException uhe){
            uhe.getStackTrace();
            return null;
        }
    }

    public static String[] getLocalIPAdresses() {
        try {
            InetAddress inetAddress;
            ArrayList<String> hosts = new
ArrayList<String>();
            Enumeration<NetworkInterface> ifaces =

NetworkInterface.getNetworkInterfaces();
            while (ifaces.hasMoreElements()) {
                NetworkInterface iface =
ifaces.nextElement();
                Enumeration<InetAddress> addrs =
iface.getInetAddresses();
                while (addrs.hasMoreElements()) {
                    inetAddress =
addrs.nextElement();
                    if (inetAddress instanceof
Inet4Address)

hosts.add(inetAddress.getHostAddress());
                }
            }
            return (String[]) hosts.toArray(new
String[hosts.size()]);
        } catch (Exception e) {
            e.getStackTrace();
            return null;
        }
    }
}
```

```
package remote_and_monitoring_b.utility;

/* @author Benyamin Bachir
 */

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class ZipUtility {

    public static Object byteArraytoObject(byte[]
data) throws Exception {
        ByteArrayInputStream bais = new
ByteArrayInputStream(data);
        ObjectInputStream ois = new
ObjectInputStream(bais);
        ois.close();
        bais.close();
        return ois.readObject();
     }
```

```
    public static byte[] objecttoByteArray(Object
obj) throws IOException {
        ByteArrayOutputStream bos = new
ByteArrayOutputStream();
        ObjectOutputStream oos = new
ObjectOutputStream(bos);
        oos.writeObject(obj);
        oos.flush();
        oos.close();
        bos.close();
        return bos.toByteArray();
    }
}
```

Tabel 1. Pengujian Integrasi Sistem Remote and Monitoring berbasis Java RMI

No.	Unjuk Kerja Sistem	Sesuai	
		Ya	Tidak
1.	Kemampuan aplikasi untuk memberikan kewenangan pada pengguna dalam menggunakan aplikasi dan fasilitas – fasilitas yang dipunyai dari aplikasi berdasarkan dari username saat login (Otentikasi).	√	
2.	Kemampuan aplikasi memproses pengendalian dan pengawasan komputer yang terhubung dalam jaringan komputer *(Remote and Monitoring).*	√	
3.	Kemampuan aplikasi mengelola komunikasi antara *Network Admin* dan *User Workstation* yang terhubung dalam jaringan komputer *(Chatting).*	√	

Tabel 2. Pengujian Sistem bagian Otentikasi

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Login* - *Network Admin*	1. Pengguna dapat masuk ke aplikasi *Remote and Monitoring Viewer* dengan memilih *username "admin"* dan memasukkan *password* yang sesuai. 2. Jendela *Remote and Monitoring Viewer* terbuka. 3. Jendela *Chat Server* terbuka.	√	
	- *User Workstation*	1. Pengguna dapat masuk ke aplikasi *Remote and Monitoring Server* dengan memilih *username "user"* dan *password* kosong. 2. Jendela *Remote and Monitoring Server* terbuka. 3. Jendela *Chat Client* terbuka.	√	
2.	*Change Password*	1. Pengguna yang hanya *login* sebagai "*admin"* dapat mengganti *password* lama dengan *password* baru untuk *login* aplikasi *Remote and Monitoring*. 2. Aplikasi menyimpan konfigurasi *password* yang baru pada file "*user.config"*.	√	
3.	*Exit Application*	1. Pengguna yang hanya mempunyai *password "admin"* dapat menutup/ mengakhiri aplikasi. 2. Aplikasi *Remote and Monitoring* tertutup.	√	

Tabel 3. Pengujian Sistem bagian *Remote and Monitoring*

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Start Remote Server*	1. Pengguna dapat memulai *Remote Server* secara otomatis setelah pengguna *login* memilih *username "user"*. 2. Aplikasi dapat menerima koneksi dari *Remote Viewer*.	√	
2.	*Configuration Remote Server*	1. Pengguna dapat mengkonfigurasi *IP Address* dan *Port* yang digunakan untuk *Remote Server*. 2. Aplikasi menyimpan konfigurasi yang baru pada file *"server.config"*.	√	
3.	*Stop Remote Server*	1. Pengguna yang hanya mempunyai *password "admin"* dapat mengakhiri *Remote Server.* 2. Aplikasi tidak dapat menerima koneksi dari *Remote Viewer.*	√	
4.	*Connect to Remote Server*	1. Pengguna yang hanya *login* sebagai *"admin"* dapat menghubungkan ke komputer *Remote Server.* 2. Aplikasi meminta masukan *IP Address* dan *Port* dari komputer *Remote Server.*	√	
5.	*Remote and Monitoring*	*1.* Pengguna yang hanya *login* sebagai *"admin"* dapat mengawasi dan mengendalikan komputer *Remote Server.* 2. Aplikasi menampilkan tampilan komputer dari *Remote Server.*	√	

Tabel 4. Pengujian Sistem bagian *Chatting*

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Start Chat Server*	1. Pengguna yang hanya *login* sebagai *"admin"* dapat memulai *Chat Server*. 2. Aplikasi menampilkan notifikasi telah siap menerima koneksi dari *Chat Client*. 3. Aplikasi dapat menerima koneksi dari *Chat Client.*	√	
2.	*Stop Chat Server*	1. Pengguna yang hanya *login* sebagai *"admin"* dapat mengakhiri *Chat Server*. 2. Aplikasi menampilkan notifikasi telah menutup koneksi dari *Chat Client*. 3. Aplikasi mengakhiri koneksi dari semua *Chat Client.*	√	
3.	*Chatting*	1. Pengguna baik *"admin"* maupun *"user"* dapat berkomunikasi melalui aplikasi *Chat.* 2. Pengguna baik *"admin"* maupun *"user"* dapat bertukar pesan. 3. Aplikasi dapat menampilkan pesan dari *"admin"* dan *"user"*.	√	
4.	*Disconnect Chat Client*	1. Pengguna *Client Chat* dapat mengakhiri koneksi dari *Server Chat* dengan memilih *disconnect chat client.* 2. Aplikasi mengakhiri koneksi dari *Server Chat.*	√	
5.	*Connect Chat Client*	1. Pengguna *Client Chat* dapat menghubungi *"admin"* dengan terlebih dahulu menghubungkan *Client Chat* ke *Server Chat.* 2. Pengguna dapat konek ke *Server Chat* dengan memasukkan *IP Address Server Chat* dan *username.* 3. Aplikasi dapat menampilkan notifikasi telah terhubung dengan *Server Chat.*	√	

KEPUTUSAN DEKAN
FAKULTAS TEKNIK
UNIVERSITAS NEGERI YOGYAKARTA
NOMOR : 173/ELK/Q-I/X/2010

TENTANG
PENGANGKATAN PEMBIMBING TUGAS AKHIR SKRIPSI
BAGI MAHASISWA FAKULTAS TEKNIK UNIVERSITAS NEGERI YOGYAKARTA

DEKAN FAKULTAS TEKNIK
UNIVERSITAS NEGERI YOGYAKARTA

Menimbang : 1. Bahwa sehubungan dengan telah dipenuhi syarat untuk penulisan Tugas Akhir Skripsi bagi mahasiswa Fakultas Teknik Universitas Negeri Yogyakarta, perlu diangkat pembimbing.
2. Bahwa untuk keperluan dimaksud perlu ditetapkan dengan Keputusan Dekan.

Mengingat : 1. Undang-undang Nomor 20 tahun 2003.
2. Peraturan Pemerintah RI Nomor 60 tahun 1999.
3. Keputusan Presiden RI: a. Nomor 93 tahun 1999; b. 305/M tahun 1999.
4. Keputusan Menteri Pendidikan dan Kebudayaan RI: Nomor 274/O/1999.
5. Keputusan Mendiknas RI Nomor 003/O/2001.
6. Keputusan Rektor UNY Nomor : 529/H34/KP/2007.

M E M U T U S K A N

Menetapkan

Pertama : Mengangkat Pembimbing Tugas Akhir Skripsi bagi mahasiswa Fakultas Teknik Universitas Negeri Yogyakarta sebagai berikut :

Nama Pembimbing : Dr. Eko Marpanaji

Bagi mahasiswa :
Nama/No.Mahasiswa : Agus Setiawan / 07520244054
Jurusan/ Prodi : Pendidikan Teknik Elektronika / Pendidikan Teknik Informatika

Kedua : Dosen pembimbing diserahi tugas membimbing penulisan Tugas Akhir Skripsi sesuai dengan Pedoman Tugas Akhir Skripsi.

Ketiga : Keputusan ini berlaku sejak ditetapkan

Keempat : Segala sesuatu akan diubah dan dibetulkan sebagaimana mestinya apabila di kemudian hari ternyata terdapat kekeliruan dalam Keputusan ini.

Ditetapkan : di Yogyakarta
Pada tanggal : 27 Oktober 2010
Dekan

Wardan Suyanto, Ed.D
NIP. 19540810 197803 1 001

Tembusan Yth :
1. Pembantu Dekan I, II, III FT UNY
2. Ketua Jurusan Pendidikan Teknik Elektronika
3. Ka Bag Tata Usaha FT UNY
4. Yang bersangkutan

Wir/26/10/2010/11:33:50

Yogyakarta, 17 April 2011

Kepada:
Yth. Drs. Kadarisman Tejo Yuwono
di tempat

Mohon dengan hormat kepada Bapak, untuk menjadi penilai ahli rekayasa perangkat lunak terhadap perangkat lunak sebagai hasil dari skripsi saya dengan judul "**APLIKASI *REMOTE AND MONITORING* BERBASIS JAVA *REMOTE METHOD INVOCATION***".

User : **admin** (login sebagai *Network Admin*)
dan **user** (login sebagai *User Workstation*)
Password : **admin** (login sebagai *Network Admin*)
dan **<kosong>** (login sebagai *User Workstation*)

Demikian surat permohonan saya, atas kesempatan yang diberikan untuk mengevaluasi perangkat lunak tersebut, kami ucapkan terima kasih.

Mengetahui
Dosen Pembimbing,

Dr. Eko Marpanaji
NIP. 19670608 199303 1 001

Hormat saya,

Agus Setiawan
NIM. 07520244054

SURAT KETERANGAN

Yang bertanda tangan di bawah ini :

Nama : Drs. Kadarisman Tejo Yuwono
NIP : 19600505 198702 1 001

Menyatakan bahwa perangkat lunak sebagai hasil dari skripsi dengan judul "**APLIKASI *REMOTE AND MONITORING* BERBASIS JAVA *REMOTE METHOD INVOCATION***" dari mahasiswa :

Nama : Agus Setiawan
NIM : 07520244054

Telah (siap / ~~belum~~)* diujicobakan dengan menambahkan beberapa saran sebagai berikut :

1. chatting dibuat full duplex agar admin maupun user dpt saling kirim berita
2. keterbatasan gambar video hanya yang resolusi rendah
3. .....

Demikian surat keterangan ini saya buat untuk dapat digunakan seperlunya.

Yogyakarta, 26/4/ 2011

Drs. Kadarisman Tejo Yuwono
NIP. 19600505 198702 1 001

*) coret yang tidak perlu

LEMBAR PENGUJIAN AHLI REKAYASA PERANGKAT LUNAK

Berilah tanda centang (√) pada pilihan "Ya" atau "Tidak" pada kolom "Tercapai" yang disediakan sesuai dengan penilaian untuk pengujian perangkat lunak sebagai hasil dari skripsi: "**APLIKASI *REMOTE AND MONITORING* BERBASIS JAVA *REMOTE METHOD INVOCATION***" yang disusun oleh Agus Setiawan.

A. PENGUJIAN

Tabel 1. Pengujian Aplikasi bagian Otentikasi

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Login* - *Network Admin*	1. Pengguna dapat masuk ke aplikasi *Remote and Monitoring Viewer* dengan memilih *username "admin"* dan memasukkan *password* yang sesuai. 2. Jendela *Remote and Monitoring Viewer* terbuka. 3. Jendela *Chat Server* terbuka.	✓	
	- *User Workstation*	1. Pengguna dapat masuk ke aplikasi *Remote and Monitoring Server* dengan memilih *username "user"* dan *password* kosong. 2. Jendela *Remote and Monitoring Server* terbuka. 3. Jendela *Chat Client* terbuka.	✓	
2.	*Change Password*	1. Pengguna yang hanya *login* sebagai *"admin"* dapat mengganti *password* lama dengan *password* baru untuk *login* aplikasi *Remote and Monitoring*. 2. Aplikasi menyimpan konfigurasi *password* yang baru pada file *"user.config"*.	✓	
3.	*Exit Application*	1. Pengguna yang hanya mempunyai *password "admin"* dapat menutup/ mengakhiri aplikasi. 2. Aplikasi *Remote and Monitoring* tertutup.	✓	

Tabel 2. Pengujian Aplikasi bagian *Remote and Monitoring*

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Start Remote Server*	1. Pengguna dapat memulai *Remote Server* secara otomatis setelah pengguna *login* memilih *username "user"*. 2. Aplikasi dapat menerima koneksi dari *Remote Viewer*.	✓	
2.	*Configuration Remote Server*	1. Pengguna dapat mengkonfigurasi *IP Address* dan *Port* yang digunakan untuk *Remote Server*. 2. Aplikasi menyimpan konfigurasi yang baru pada file *"server.config"*.	✓	
3.	*Stop Remote Server*	1. Pengguna yang hanya mempunyai *password "admin"* dapat mengakhiri *Remote Server*. 2. Aplikasi tidak dapat menerima koneksi dari *Remote Viewer*.	✓	
4.	*Connect to Remote Server*	1. Pengguna yang hanya *login* sebagai *"admin"* dapat menghubungkan ke komputer *Remote Server*. 2. Aplikasi meminta masukan *IP Address* dan *Port* dari komputer *Remote Server*.	✓	
5.	*Remote and Monitoring*	*1.* Pengguna yang hanya *login* sebagai *"admin"* dapat mengawasi dan mengendalikan komputer *Remote Server*. 2. Aplikasi menampilkan tampilan komputer dari *Remote Server*.	✓	

Tabel 3. Pengujian Aplikasi bagian *Chatting*

No.	Aktifitas/menu	Hasil yang diharapkan	Tercapai	
			Ya	Tidak
1.	*Start Chat Server*	1. Pengguna yang hanya *login* sebagai *"admin"* dapat memulai *Chat Server*. 2. Aplikasi menampilkan notifikasi telah siap menerima koneksi dari *Chat Client*. 3. Aplikasi dapat menerima koneksi dari *Chat Client*.	✓	
2.	*Stop Chat Server*	1. Pengguna yang hanya *login* sebagai *"admin"* dapat mengakhiri *Chat Server*. 2. Aplikasi menampilkan notifikasi telah menutup koneksi dari *Chat Client*. 3. Aplikasi mengakhiri koneksi dari semua *Chat Client*.	✓	
3.	*Chatting*	1. Pengguna baik *"admin"* maupun *"user"* dapat berkomunikasi melalui aplikasi *Chat*. 2. Pengguna baik *"admin"* maupun *"user"* dapat bertukar pesan. 3. Aplikasi dapat menampilkan pesan dari *"admin"* dan *"user"*.	✓	
4.	*Disconnect Chat Client*	1. Pengguna *Client Chat* dapat mengakhiri koneksi dari *Server Chat* dengan memilih *disconnect chat client*. 2. Aplikasi mengakhiri koneksi dari *Server Chat*.	✓	
5.	*Connect Chat Client*	1. Pengguna *Client Chat* dapat menghubungi *"admin"* dengan terlebih dahulu menghubungkan *Client Chat* ke *Server Chat*. 2. Pengguna dapat konek ke *Server Chat* dengan memasukkan *IP Address Server Chat* dan *username*. 3. Aplikasi dapat menampilkan notifikasi telah terhubung dengan *Server Chat*.	✓	

Tabel 4. Unjuk Kerja Sistem *Remote and Monitoring*

No.	Unjuk Kerja Sistem	Sesuai	
		Ya	Tidak
1.	Kemampuan aplikasi untuk memberikan kewenangan pada pengguna dalam menggunakan aplikasi dan fasilitas – fasilitas yang dipunyai dari aplikasi berdasarkan dari username saat login (Otentikasi).	✓	
2.	Kemampuan aplikasi memproses pengendalian dan pengawasan komputer yang terhubung dalam jaringan komputer *(Remote and Monitoring).*	✓	
3.	Kemampuan aplikasi mengelola komunikasi antara *Network Admin* dan *User Workstation* yang terhubung dalam jaringan komputer *(Chatting).*	✓	

B. KESIMPULAN

Perangkat lunak ini dinyatakan : (*lingkari salah satu*)

(1) Memiliki unjuk kerja yang baik.

2. Memiliki unjuk kerja yang buruk.

3. Tidak berfungsi.

C. SARAN

...Chatting...dibuat full duplex.....

.....

.....

.....

.....

Penguji,

Drs. Kadarisman Tejo Yuwono

NIP. 19600505 198702 1 001

SURAT KETERANGAN VALIDASI

Yang bertanda tangan di bawah ini :

Nama : **Drs. Kadarisman Tejo Yuwono**

NIP : 19600505 198702 1 001

Menyatakan bahwa instrumen penilitian dari skripsi dengan judul **"APLIKASI *REMOTE AND MONITORING* BERBASIS JAVA *REMOTE METHOD INVOCATION*"** dari mahasiswa :

Nama : **Agus Setiawan**

Nim : 07520244054

(sudah layak/~~belum layak~~)* dipergunakan untuk penelitian dengan menambahkan beberapa saran sebagai berikut :

1. Indikator untuk frekuensi kesalahan kurang tepat dihapus saja
2. Indikator untuk kepuasan produk disesuaikan dgn tujuan
3. .....

.....

Demikian surat keterangan ini kami buat untuk dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 21 April 2011

Validator

Drs. Kadarisman Tejo Yuwono

Tabel 1. Kisi – kisi Penilaian Pengguna dari segi *Usability* (Timothy, 2002:245)

No	Aspek	Indikator	Butir
1	*Learnability*	Kemudahan mempelajari produk.	1
		Kecepatan untuk menguasai sampai menjadi mahir.	2
2	*Efficiency of use*	Kecepatan menyelesaikan tugas.	3
		Kecepatan membenahi kesalahan.	4
3	*Error Handling*	Pesan kesalahan.	5
4	*Acceptability*	Kebermanfaatan produk.	6
		Kepuasan terhadap produk.	7, 8

Tabel 1. Rekap hasil penilaian pengguna dalam uji *beta*

No.	Nama Penilai uji beta	Penilaian							
		Learnability		***Efficeiency of use***		***Error Handling***	***Acceptability***		
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	Anton	3	4	2	3	3	2	3	4
2	Adi	4	4	3	3	2	4	4	3
3	Arwan	4	4	4	4	4	4	4	4
4	Anjar	4	4	3	3	3	4	4	4
5	Banu	4	4	2	3	3	4	4	3
6	Yudan	3	4	3	3	2	3	3	3
7	Lukman	3	3	3	3	3	3	3	3
8	Rizam	3	2	3	3	3	3	3	3
9	Sigit	4	4	3	4	4	4	4	4

Keterangan:

4 = Sangat Setuju **2 = Tidak Setuju**

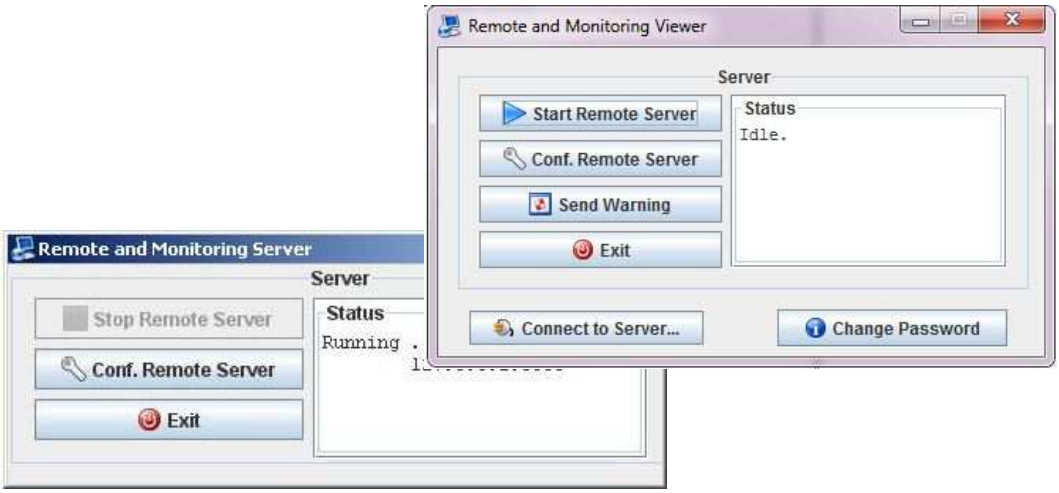
3 = Setuju **1 = Sangat Tidak Setuju**

Tabel 2. Rekap hasil komentar/saran dari uji *beta*

No.	Nama Penilai uji *beta*	Komentar/Saran
1	Anton	- Perlu dokumentasi yang lengkap. - Pengaturan tata letak window sebaiknya ditampilkan ditengah layar.
2	Adi	- *User Interface* dipermanis.
3	Arwan	- Hendaknya menu aplikasi dibuat satu kotak dialog.
4	Anjar	-
5	Banu	- Perlu adanya dokumentasi. - Pesan kesalahan dipermudah.
6	Yudan	- Ditambah menu log dan monitoring multipleworkstation. - Error fault yang mudah dipahami.
7	Lukman	- Sebaiknya menu-menu dibuat pulldown.
8	Rizam	-
9	Sigit	-

MANUAL APLIKASI

Remote and Monitoring berbasis Java *Remote Method Invocation*



Oleh
Agus Setiawan
07520244054

PROGRAM STUDI PENDIDIKAN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI YOGYAKARTA

MEI 2011

A. Deskripsi Aplikasi

Aplikasi *Remote and Monitoring* berbasis Java RMI ini merupakan aplikasi yang digunakan untuk mengawasi dan mengendalikan komputer yang saling terhubung jaringan. Aplikasi *Remote and Monitoring* ini mempunyai tiga bagian, yakni: otentikasi, *remote and monitoring,* dan *chatting.*

1. Otentikasi

Aplikasi bagian otentikasi merupakan bagian untuk mengatur keamanan sistem. Otentikasi ini meliputi *login* aplikasi, *change password*, dan otentikasi untuk keluar dari aplikasi. *Login* aplikasi berguna untuk membatasi kewenangan terhadap aplikasi. Pengguna aplikasi dapat login sebagai admin atau user.

"*admin*" mempunyai kewenangan penuh terhadap aplikasi. "*admin*" dapat mengawasi dan mengendalikan komputer *User Workstation* yang sedang dalam keadaan "*on*".

"*user*" adalah untuk pengguna *User Workstation*. "*user*" tidak mempunyai kewenangan penuh seperti "*admin*". "*user*" hanya dapat menghubungi "*admin*" melalui aplikasi *Client Chat*.

2. Remote and Monitoring

Aplikasi bagian *Remote and Monitoring* merupakan bagian inti dari aplikasi. Bagian ini berfungsi untuk mengawasi dan mengendalikan komputer "*user*" yang dalam kondisi "*on*" dan terhubung jaringan.

"*admin*" dapat mengawasi dan mengendalikan komputer "*user*" dengan memilih menu "*Connect to Server*". "*admin*" harus memasukkan *IP Address* dan *port* komputer *Remote Server* milik "*user*" untuk dapat menggunakan fasilitas *Remote and Monitoring* ini.

3. Chatting

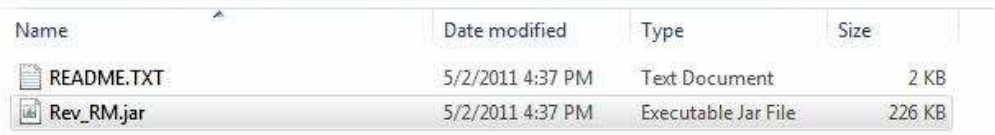
Aplikasi bagian Chatting berguna untuk kirim pesan antar "*user*" dan "*admin*". Bagian ini dapat mempercepat komunikasi antara *User Workstation* dan *Network Admin*, sehingga *User Workstation* dapat lebih mudah menghubungi *Network Admin* untuk mendapatkan bantuan ataupun mendapatkan informasi dari *Network Admin*. *Network Admin* bertugas memantau Chat Server untuk menunggu apakah ada "*user*" yang masuk *chatting.*

B. Spesifikasi Minimal

Aplikasi *Remote and Monitoring* berbasis *Java Remote Method Invocation* ini adalah dapat dijalankan di semua system operasi *(multiplatform)* dan minimal *Java Runtime Environtment* diatas jre 1.6.0,.

C. Instalasi

Aplikasi *Remote and Monitoring* berbasis *Java Remote Method Invocation* ini tidak memerlukan instalasi yang rumit, hanya perlu diklik dua kali pada *file* aplikasi *Remote and Monitoring.*



Gambar 1. File aplikasi *Remote and Monitoring*

D. Manual Aplikasi

1. Otentikasi

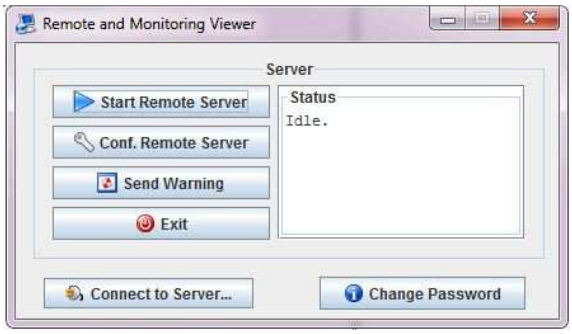
Tiap aplikasi *Remote and Monitoring* di-eksekusi akan selalu melewati proses otentikasi, dimana proses otentikasi ini yang akan menentukan kewenangan pengguna. Untuk sebagai *Network Admin* maka dapat login dengan menggunakan *username: admin*, dan *password: admin.*

Gambar 2. Login untuk *admin*

Komputer *User Workstation* dipasangi aplikasi dengan login menggunakan *username:user*, dan *password: <kosong>*.

Gambar 3. Login untuk *user*

Aplikasi juga dapat mengganti *password* untuk login *admin.* Pengguna dapat mengganti *password* dengan memilih dan klik tombol *Change Password.*



Gambar 4. Form utama *Remote and Monitoring Viewer*

Gambar 5. *Form* ganti *password admin*

Setelah pengguna mengganti *password* untuk *admin* maka untuk *password login* maupun keluar adalah sama dengan *password* yang baru. Pengguna dapat keluar dari aplikasi juga harus dengan proses otentikasi. Pengguna dapat keluar dengan memilih dan klik tombol *Exit,* kemudian masukkan *password admin.*

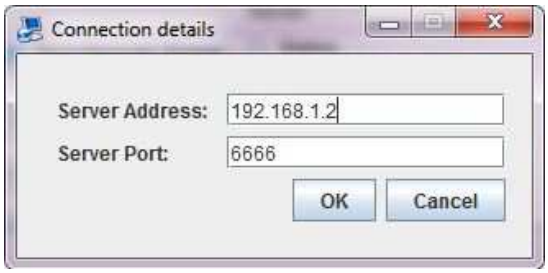
Gambar 6. *form* untuk *Exit*

2. *Remote and Monitoring*

Pengawasan dan pengendalian hanya dapat dilakukan oleh pengguna yang login dengan *username admin.* Pastikan sebelum mulai pengawasan dan pengendalian, aplikasi untuk *User Workstation* sudah dieksekusi.

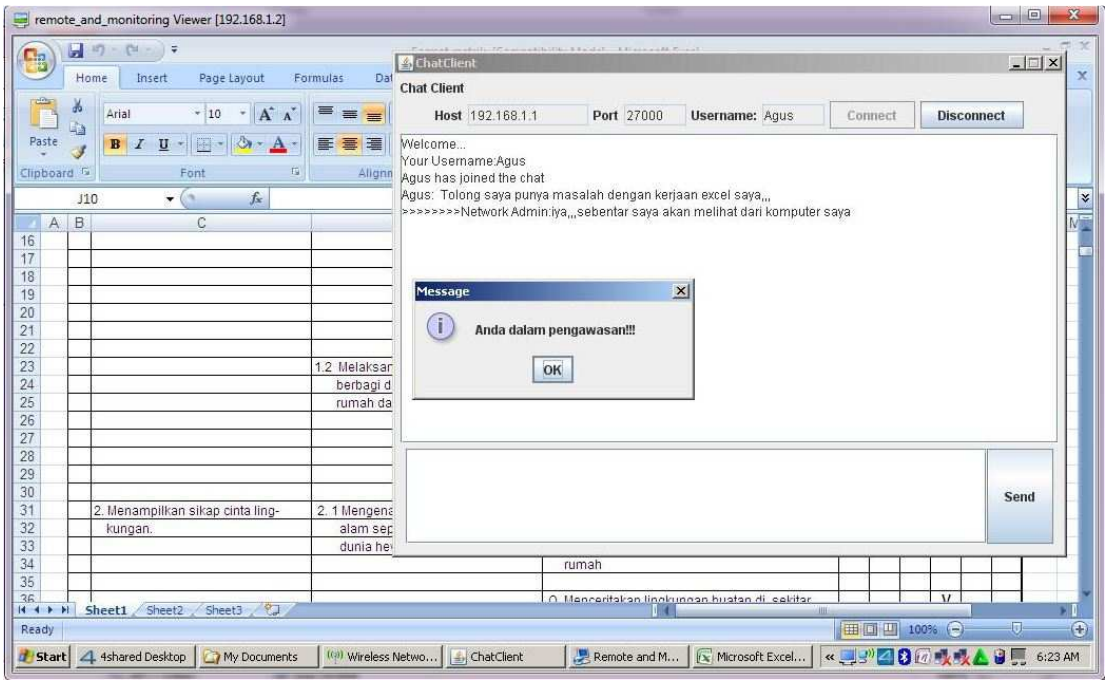
Gambar 7. *Remote and Monitoring Server*

Gambar 7 menunjukkan bahwa aplikasi *Remote Server* pada komputer *User Workstation* sudah dieksekusi dan sudah siap untuk dikendalikan dan diawasi. Untuk langkah selanjutnya, pengguna *admin* dapat memilih dan klik "*Connect to Server …*" pada tampilan utama *Remote and Monitoring Viewer.* Kemudian akan muncul tampilan *"Connection Details"*, masukkan IP Address dan Port komputer *User Workstation* yang akan diawasi dan dikendalikan.



Gambar 8. *Form* untuk *Connect to Server*

Jika koneksi dari *Remote and Monitoring Viewer* ke *Remote and Monitoring Server* maka akan muncul seperti contoh tampilan pada Gambar 9. Dengan muncul tampilan tersebut maka pengguna sudah dapat mengendalikan komputer *User Workstation* dengan memasukkan input dari *mouse* maupun *keyboard.*

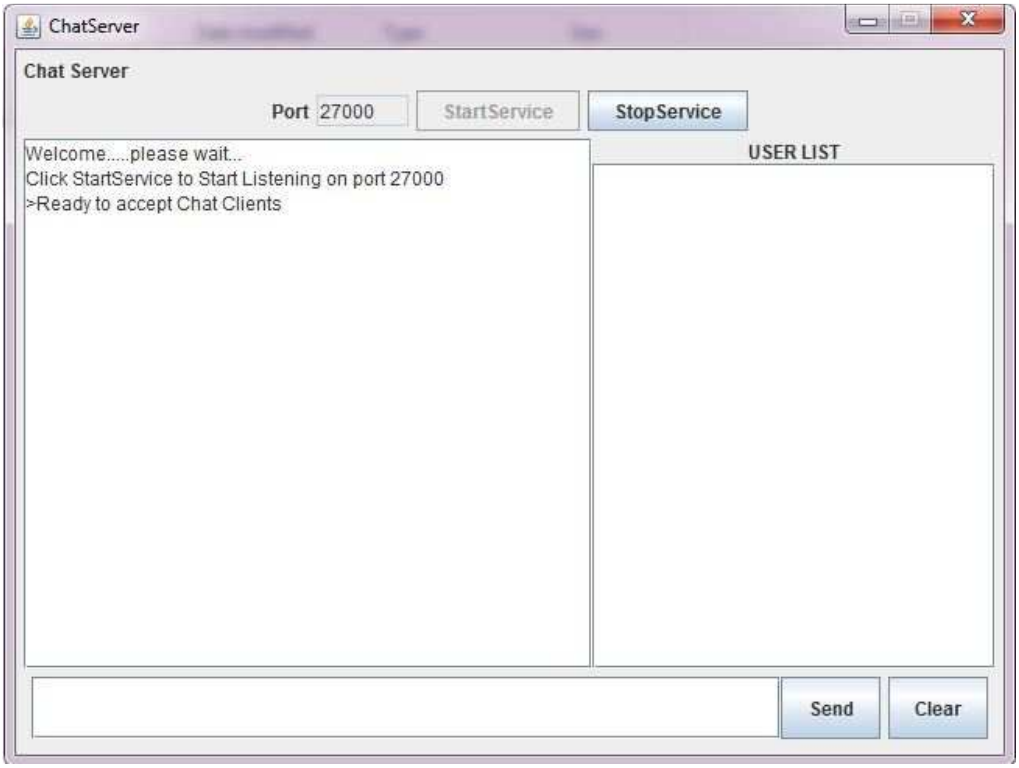


Gambar 9. *Pengawasan dan Pengendalian Remote Server*

3. *Chatting*

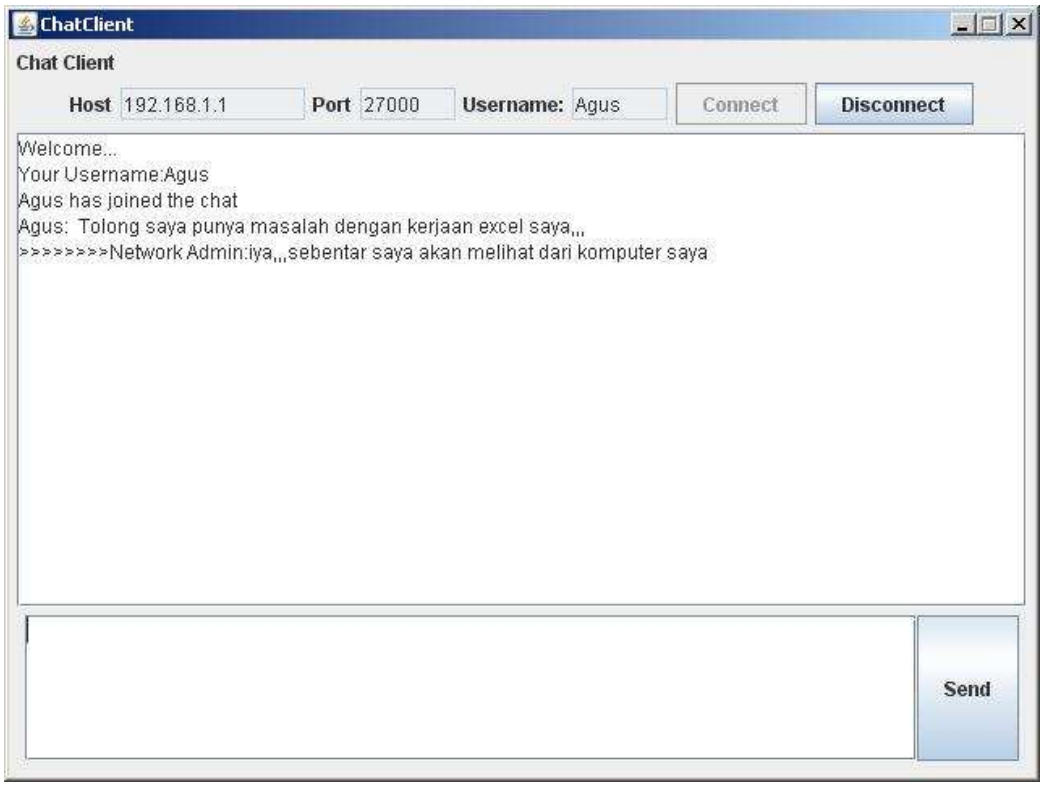
Pengguna dapat melakukan aktifitas *chatting* dengan beberapa langkah sebagai beriku:

a. Start Service pada aplikasi *Chat Server* bagi *username admin.*



Gambar 10. Tampilan *Chat Server*

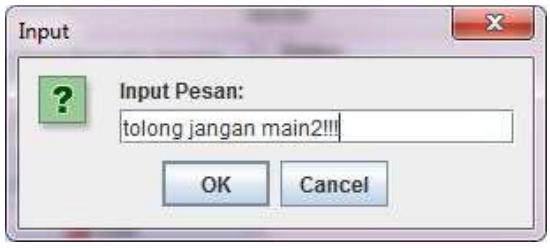
b. Masukkan IP Address dan Username untuk *Chat Client* bagi *username user*



Gambar 11. Tampilan *Chat Client*

c. Pengguna yang login sebagai *admin* dapat memberikan pesan peringatan kepada *user* dengan memilih dan klik tombol *"Send Warning"* pada tampilan utama. Kemudian pengguna perlu memasukkan IP Address target dan Pesan yang akan dikirim secara berurutan.

Gambar 12. Masukkan IP Address target



Gambar 13. Masukkan pesan *warning*

d. Setelah berhasil dikirim maka dapat dilihat dari tampilan *User Workstation* muncul pesan *warning* yang terkirim.

Gambar 14. Tampilkan pesan *warning* untuk *User Workstation*