KONFERENSI NASIONAL SISTEM & INFORMATIKA

PROCEEDINGS

KONFERENSI NASIONAL SISTEM & INFORMATIKA 2012

AUDITORIUM STMIK STIKOM BALI
17 November 2012

Editor :
Yudi Agusta, PhD

Dipublikasikan Tahun 2012 Oleh:
STMIK STIKOM BALI
Denpasar – Indonesia
www.stikom-bali.ac.id

Editor:
Yudi Agusta, PhD

Asisten Editor:
Desy Tri Puspasari, S.Kom
Tubagus Mahendra Kusuma, S.E

Disain Cover:
Tubagus Mahendra Kusuma, S.E

Dicetak di Denpasar – Indonesia
PERCETAKAN RYZQUNA PRINTING

ISSN: 1979-9845

DAFTAR REVIEWER

Agus Fanar Syukri, PhD (Lembaga Ilmu Pengetahuan Indonesia)

Agus Pribadi, ST, MSc (STMIK Bumigora Mataram)

Andreas Handojo, MT (Universitas Kristen Petra Surabaya)

Anto Satriyo Nugroho, DR.Eng (Badan Pengkajian dan Penerapan Teknologi)

Dahliyusmanto, MSc (Universiti Teknologi Malaysia)

Edhy Sutanta, ST, MKom (AKPRIND Yogyakarta)

Indra Adji Sulistijono,ST,M.Eng (Institut Teknologi Sepuluh Nopember)

Khairul Munadi (Universitas Syiah Kuala)

Lintang Yuniar Bonosuwari, DR (Universitas Gunadarma)

Marvin Candra Wijaya, MT (Universitas Kristen Maranatha)

Muhammad Arhami (Politeknik Negeri Lhokseumawe)

Muhamad Said Hasibuan, M.Kom (Institut Informatika dan Bisnis Darmajaya)

Rudi Adipranata, MEng (Universitas Kristen Petra Surabaya)

Son Kuswadi, DR (Institut Teknologi Sepuluh Nopember)

Surya Sumpeno (Institut Teknologi Sepuluh Nopember)

Wahju Sediono, DR.Eng (Badan Pengkajian dan Penerapan Teknologi)

Warnia Nengsih Sikumbang, SKom, MKom (Politeknik Caltex Riau)

Widodo, SKom, MKom (Universitas Negeri Jakarta)

Yudho Giri Sucahyo, DR (Universitas Indonesia)

Yudi Agusta, PhD (STMIK STIKOM Bali)

DAFTAR ISI

KATA PENGANTAR i

DAFTAR REVIEWER ii

DAFTAR ISI iii

KEYNOTE SPEAKER: *Augmented Reality and Ubiquitous Multimedia: The Next Digital Media* v
Mochamad Hariadi, PhD
Electrical Engineering Department, Sepuluh Nopember Institute of Technology, Surabaya

DAFTAR MAKALAH

[KNS&I12-001]	Aplikasi Dinas Pariwisata dan Kebudayaan Jawa Barat Berbasis Mobile Web	1
[KNS&I12-002]	Implementasi Class Robot Java Pada Aplikasi Remote Desktop	9
[KNS&I12-003]	Perancangan dan Implementasi Pemantauan Bandwidth Usage Jaringan Komputer	18
[KNS&I12-004]	Pemantauan Performa Perangkat Keras Pada Server Dengan Psutil Python Library dan Restful Web Service Menggunakan Deploy Engine	24
[KNS&I12-005]	Strategi Pemanfaatan Sistem Penerimaan Mahasiswa Berbasis Web Untuk Meningkatkan Keunggulan Kompetitif Perguruan Tinggi di Bangka Belitung (Studi Kasus: STMIK Atma Luhur)	30
[KNS&I12-006]	Pemanfaatan Squid Sebagai Web Proxy Server Untuk Mempercepat Koneksi Internet dan Penghematan Pemakaian Bandwidth	36
[KNS&I12-007]	Rancang Bangun Sistem Informasi Rawat Jalan Menggunakan Metode Berorientasi Objek: Studi Kasus Klinik Sehat Sungailiat Bangka Belitung	41
[KNS&I12-008]	Rancangan Sistem Informasi Perpustakaan Berbasis Web Studi Kasus STMIK Atma Luhur Pangkalpinang	47
[KNS&I12-009]	Model Sistem Informasi Penagihan Retribusi Sampah: Studi Kasus BLH Sungailiat	53
[KNS&I12-010]	Membangun Sistem Informasi Persediaan Barang Pada Instansi Pemerintah	60
[KNS&I12-011]	Membangun Rancangan Sistem Informasi Penjualan Pada Toko Sinar Buana Mebel Dengan Metodologi Berorientasi Objek	66
KNS&I12-012]	Rancangan Pengembangan Sistem Basisdata Peminjaman Buku Dengan Kartu Anggota dan Buku Ber-Barcode Studi Kasus: Perpustakaan Kota Pangkalpinang	72
[KNS&I12-013]	Membangun Rancangan Sistem Informasi Perpustakaan: Studi Kasus SD N 1 Pangkalpinang	77
[KNS&I12-014]	Membangun Sistem Informasi Administrasi Pinjaman Pakai Senjata Api (Senpi) Studi Kasus: Kepolisian Resort Kota Pangkalpinang Dengan Metodologi Berorientasi Objek	83
[KNS&I12-015]	Rancang Bangun Sistem Informasi Surat Keterangan Catatan Kepolisian (SKCK) Studi Kasus: Kepolisian Resort Kota Pangkalpinang Dengan Metodologi Berorientasi Objek	88
[KNS&I12-016]	Sistem Pendukung Keputusan Untuk Menentukan Model Pengembangan Sistem Pembelajaran Berbasis Internet	93
[KNS&I12-017]	Perancangan dan Implementasi Sistem Pakar Prediksi Penyakit Jantung Berdasarkan Metode Backward Chaining dan Fuzzy Logic	99
[KNS&I12-018]	Membangun Sistem Informasi Pasien Rawat Jalan Pada Puskesmas Taman Sari Pangkalpinang	106
[KNS&I12-019]	Sistem Pakar Untuk Diagnosa Penyakit Vertigo Dengan Metode Forward Chaining	112
[KNS&I12-020]	Perencanaan Strategis Sistem Informasi Studi Kasus PT Prime Capital Securities	116
[KNS&I12-021]	Analisis Information Visualization Pada Website Bandar Udara Hartsfield-Jackson, Heathrow, dan Beijing	123
[KNS&I12-022]	Perancangan Algoritma Kriptografi Rivest Shamir Adleman (RSA) Untuk Keamanan Data di Oracle 10g	129
[KNS&I12-023]	Aplikasi Pembagian Kelompok Kelas Menggunakan Algoritma Genetik Pada SMA Budi Mulia Tangerang	135
[KNS&I12-024]	Pengontrolan dan Monitoring Ruang Kelas Dengan Menggunakan Controller Board ARM 2368	141
[KNS&I12-025]	Explaining Behavioral Intention On Information Technology: A Case Study of CIMB Niaga Internet Banking in Surabaya	147
[KNS&I12-026]	Implementasi Data Mining Untuk Menemukan Association Rule Pada Data Perbankan	154
[KNS&I12-027]	Implementasi Trust Negotiation Pada E-Commerce Dengan Manajemen Identitas Menggunakan Metode Enkripsi Asimetrik Riverst Shamir Adleman (RSA)	160

IMPLEMENTASI CLASS ROBOT JAVA PADA APLIKASI *REMOTE DESKTOP*

Jasman Pardede
Teknik Informatika, Institut Teknologi Nasional Bandung
jasman@itenas.ac.id

ABSTRACT

Remote Desktop is a term to represent a device that is located in a particular location used to access other device from a different location. Some existing Remote Desktop applications have been created is Windows and Macintosh OS. Windows Remote Desktop has a functionality to transmit a copy of the screen (screen captured), responsibility keyboard and mouse events, and voice. Macintosh Remote Desktop provides additional features such as remote installation. The applications can only be used remotely on each platform by internet device. However, there is a condition where people want to access data and applications in multiplatform. To solve the problem, researcher developed a multiplatform Remote Desktop Application. Programming language that supports multiplatform is Java. To send screen of server PC and handle mouse and keyboard events are performed by user, researcher then implements the class Robot provided in Java.awt package. The client server communication implements Java socket programming using TCP/IP protocol.
Keywords: *Remote Desktop, Screen Captured, Keyboard and Mouse Events, Multiplatform, Class Robot.*

1. Pendahuluan

Sub bab ini membahas tentang latar belakang perlunya mengembangkan aplikasi *Remote Desktop* berbasis Java, rumusan masalah, tujuan penelitian, dan batasan masalah dalam pengembangan aplikasi.

1.1 Latar Belakang

Remote Desktop adalah sebuah istilah untuk mewakili kejadian di mana sebuah PC atau perangkat komputer yang ada di suatu lokasi tertentu, dapat diakses dan digunakan dari tempat yang berbeda[1]. Penggunaan komputer yang di *remote* tidak hanya terbatas pada tampilan teks tetapi tampilan *desktop* komputer seperti saat menggunakannya secara langsung. *Remote Desktop* dalam mengendalikan dan menampilkan salinan gambar layar (*screen captured*) komputer yang dikendalikan dipengaruhi oleh kualitas jaringan komunikasi antar kedua komputer dalam interval waktu tertentu. Perangkat lunak *Remote Desktop* pada komputer yang akan di*remote* mentransmisikan dan mengendalikan aktivitas *keyboard* atau *mouse* ke komputer yang me*remote*. Teknologi *Remote Desktop* bukanlah teknologi baru, teknologi ini sudah cukup lama dikenal dalam dunia komputerisasi.

Beberapa aplikasi *Remote Desktop* yang sudah pernah dibuat di antaranya *Remote Desktop* Windows dan Macintosh. *Remote Desktop* bawaan Windows memiliki fungsionalitas mentransmisikan salinan layar (*screen captured*), *event keyboard* dan *mouse*, dan suara[2]. *Remote Desktop* Macintosh disamping fungsionalitas yang dimiliki oleh Windows juga menyediakan fungsionalitas tambahan seperti peng*install*an secara *remote*[1]. Tetapi kedua *Remote Desktop* tersebut hanya dapat melakukan *remote* pada platformnya masing-masing dan jaringan komunikasi menggunakan jaringan internet. Untuk menyelesaikan masalah tersebut, peneliti akan mengembangkan aplikasi *Remote Desktop* yang *multiplatform*. Bahasa pemrograman yang mendukung *multiplatform* yaitu bahasa pemrograman Java. Untuk dapat mengirimkan *screen captured*, PC komputer yang akan di*remote* serta penanganan *event mouse* dan *keyboard* yang dilakukan oleh pengguna pada *screen captured* yang di*remote*, maka peneliti akan mengimplementasikan *method-method* yang dimiliki oleh *class Robot* Java yang berada pada *package Java.awt*. Jaringan komunikasi antara *client* dan *server* dilakukan dengan mengimplementasikan pemrograman *socket* Java berbasis TCP/IP.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah dinyatakan pada subbab 1.1 maka rumusan masalah pada penelitian ini adalah sebagai berikut:

1. Bagaimana aplikasi *server* dapat merespon *request*-an *client*, melakukan pengaturan pengiriman dan penerimaan data pada *Remote Desktop Client* berbasis PC?
2. Bagaimana aplikasi *client* berbasis PC me*request* ke *Remote Desktop Server*, menerima data dari *Remote Desktop Server* dan menampilkannya?

1.3 Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah untuk mengembangkan aplikasi *Remote Desktop* berbasis PC tanpa memperhatikan *platform* dengan mengimplementasikan *class Robot* Java.

1.4 Batasan Masalah

Pada penelitian ini penulis membatasi masalah sebagai berikut:

1. Aplikasi *Remote Desktop* yang akan dikembangkan tidak memperhatikan *delay* pengiriman data.
2. Pengembangan aplikasi dalam proses pengiriman dan penerimaan data menggunakan pemrograman *socket* Java.

2. Landasan Teori

Sub bab ini membahas tentang pengertian *Remote Desktop*, *remote dekstop* sistem operasi windows dan macintos bahasa pemrograman Java, class Robot Java, dan *socket*.

2.1 Pengertian *Remote Desktop*

Remote Desktop adalah sebuah istilah untuk mewakili kejadian dimana sebuah PC atau perangkat komputer yang ada suatu lokasi tertentu, diakses dan digunakan dari tempat yang berbeda lokasi[1]. Teknologi *Remote Desktop* sebenarn bukanlah teknologi yang baru. Kehadiran teknologi ini sudah cukup lama di dunia komputerisasi. Umumnya teknolo yang digunakan untuk *Remote Desktop* hanya untuk pengendalian dari PC ke PC.

Dalam komputasi, *Remote Desktop* merujuk pada sebuah perangkat lunak atau fungsionalitas dari sistem operasi yan memungkinkan untuk menjalankan *server* secara jarak jauh untuk ditampilkan secara lokal. Cara kerja dari *Remo Desktop* cukup sederhana yaitu dengan cara mengirimkan *captured screen* komputer yang di*remote* ke komputer yan me*remote* saat bekerja mengendalikan komputer yang di*remote* sehingga komputer yang me*remote* dapat melihat has eksekusi dari komputer yang di*remote* tersebut. Secara umum komputer yang me*remote* menggirimkan *event-event* yan terjadi di komputer yang me*remote* dan meng*copy*kan *event-event* tersebut ke komputer yang akan di*remote*. Beberap fitur yang biasanya ada pada sebuah *Remote Desktop* adalah menampilkan *screen* komputer yang dikendalika mengirimkan dan mengeksekusi *event-event* dari *keyboard* dan *mouse*, *network printer redirection*, *redirection of loc drives*, *copy and paste files*, serta *audio redirection*.

2.2 *Remote Desktop* Sistem Operasi Windows dan Macintosh

Pada sistem operasi yang berbasis Windows pada umumnya fitur *Remote Desktop* yang disediakan adalah menampilka *screen* komputer yang dikendalikan, penangangan *event-event keyboard* dan *mouse*, *network printer redirect*, *redirect local drivers*, *copy* dan *paste files*, *audio redirection*, *support for over 256 colors*, *redirection of windows ke combinations*, *shared clipboard*, 128 bit *encryption*, *port redirection*, dan *connection bar*. *Remote Desktop* pada siste operasi Windows menggunakan protokol RDP (*Remote Desktop Protocol*). Windows menanam protokol tersebut dala level kernel sehingga *Remote Desktop* menjadi lebih aman dan lebih cepat dalam pengeksekusian[2]. Versi RDP terba saat ini adalah RDP versi 7.0 yang sudah ter*include* dalam Windows *Server* 2008 dan Windows 7.

Pada sistem operasi Macintosh terdapat *software* Apple *Remote Desktop* 3 yang memiliki beberapa fitur yaitu *Remot Spotlight Search*, *Dashboard Widget*, *Automator Action*, *Auto Install*, *Curtain Mode*, *RemoteDrag and Drop*, *Powe Copy*, *System Status Indicators*, *Application Usage Report*, *User History Reports*, *Smart Computer List*, dan *Tas Templates*[1].

2.3 Bahasa Pemrograman Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Java berdiri d atas sebuah mesin *interpreter* yang diberi nama *Java Virtual Machine* (JVM)[3]. JVM inilah yang akan membac *bytecode* dalam *file .class* dari suatu program sebagai representasi langsung program yang berisi bahasa mesin. *Platfor* Java terdiri dari kumpulan library, JVM, kelas-kelas *loader* yang dipaket dalam sebuah lingkungan rutin Java dan sebua kompiler, *debugger* dan kakas lain yang dipaket dalam *Java Development Kit* (JDK). Java 2 adalah generasi yan sekarang sedang berkembang dari *platform Java*. Agar sebuah program Java dapat berjalan dengan baik, maka *fil* dengan eksistensi Java harus dikompilasi menjadi *file bytecode*. Untuk menjalankan *bytecode* tersebut dibutuhkan *Jav Runtime Environtment* (JRE) yang memungkinkan pemakai untuk menjalankan program Java, hanya menjalankan, tida untuk membuat kode baru lagi. JRE berisi JVM dan *library* Java yang digunakan[4].

2.4 *Class Robot* Java

Class Robot pada Java berada pada *package Java.awt*. *Class Robot* digunakan untuk membangkitkan *event-event inpu native system* untuk tujuan pengujian secara otomatisasi, *self-running demos*, dan aplikasi lain yang pengontrolan kerj aplikasi dilakukan oleh *event mouse* dan *keyboard* tanpa memperhatikan *platform* dimana aplikasi dijalankan[5]. Pembangkitan *event* secara otomatis yang dilakukan oleh *class Robot* dalam melakukan *screen captured*, mengendalikan *mouse* dan *keyboard* dilakukan secara *software*. *Class Robot* Java memiliki beberapa *method*, seperti yang dinyatakan pada Tabel 1.

Tabel 1. Method Class Robot Java

Return Type	*Method Class Robot*
BufferedImage	**createScreenCapture (Rectangle screenRect)** Menciptakan sebuah *image* yang mengandung *pixel* yang dibaca dari layar secara *software*.
void	**delay(int ms)** *Sleep* atau menghentikan aplikasi dalam waktu tertentu dalam satuan *milliseconds*.
int	**getAutoDelay( )** Mengembalikan waktu *sleep* dari *class Robot* dalam *milliseconds* setelah pembangkitan suatu *event*.
Color	**getPixelColor(int x, int y)** Mengembalikan warna *pixel* pada titik koordinat x dan y tertentu secara *software*.
boolean	**isAutoWaitForIdle( )** Mengembalikan status *class Robot* apakah secara otomatis dapat memintak *waitForIdle*( ) setelah pembangkitan suatu *event*.
void	**keyPress (int keycode)** Menekankan tombol *keyboard* secara *software*.
void	**keyRelease (int keycode)** Melepaskan tombol *keyboard* yang ditekan secara *software*.
void	***mouse*Move (int x, int y)** Memindahkan *pointer mouse* ke koordinat layar yang diberikan.
void	***mouse*Press (int buttons)** Menekankan tombol *mouse* secara *software*.
void	***mouse*Release (int buttons)** Melepaskan tombol *mouse* yang ditekan secara *software*.
void	***mouse*Wheel (int wheelAmt)** Menggerakkan *scroll mouse* secara *software*.
void	**setAutoDelay (int ms)** Memberikan waktu dalam *millisecond class Robot* melakukan *sleep* secara otomatisasi setelah pembangkitan suatu *event* secara *software*.
void	**setAutoWaitForIdle(boolean isOn)** Menentukan apakah *class Robot* dapat meminta waitForIdle secara otomatisasi setelah pembangkitan suatu *event* secara *software*
void	**waitForIdle( )** Menunggu sampai semua *event-event* yang sedang berjalan yang ada pada *event* antrian selesai diproses.

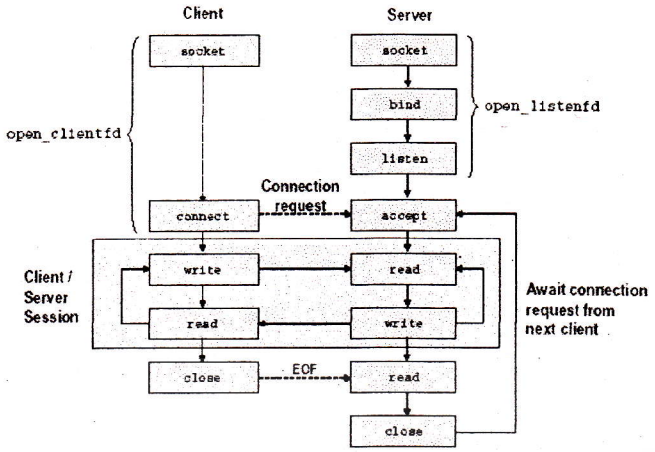
Dengan menggunakan method **createScreenCapture (Rectangle screenRect)** *class Robot*, dalam mengambil dan menyimpan gambar screen komputer secara software dapat dilakukan seperti pada Listing code 1.

Listing code 1.

```
public class RobotCaptureScreenExmp {
  public static void main(String args[]){
    try {
      // capture the whole screen
      BufferedImage screencapture = new Robot().createScreenCapture(
          new Rectangle(Toolkit.getDefaultToolkit().getScreenSize()) );
      int[] dataInt = ((DataBufferInt)screencapture.getData().getDataBuffer()).getData();
      int width = screencapture.getWidth();
      int height = screencapture.getHeight();
      BufferedImage image = new BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
      WritableRaster wr = image.getRaster();
      wr.setDataElements(0,0,width,height,dataInt);
      // Save as JPEG
      File file = new File("screencapture.jpg");
      ImageIO.write(screencapture, "jpg", file);
    } catch (IOException ex) {
      Logger.getLogger(RobotCaptureScreenExmp.class.getName()).log(Level.SEVERE, null, ex);
    } catch (AWTException ex) {
      Logger.getLogger(RobotCaptureScreenExmp.class.getName()).log(Level.SEVERE, null, ex);
    }
  }
}
```

2.5 *Socket*

Socket adalah mekanisme komunikasi yang memungkinkan terjadinya pertukaran data antar program atau proses, ba dalam satu mesin maupun antar mesin[6][7]. *Socket* dibuat dengan menyambungkan dua buah alamat IP melalui *p* tertentu. Secara umum *socket* digunakan dalam sistem *client/server*, dimana sebuah *server* akan menunggu *client* pa *port* tertentu. Saat *client* me*request server* maka *server* akan berkomunikasi dengan *client* melalui *socket* yang dibangu Salah satu fungsi *socket* adalah *interface socket* yang digunakan untuk menghubungkan komputer ke jaringan atau anta *client* dengan *server*[7]. *System call* pada *interface socket* dapat memudahkan suatu aplikasi untuk membuat *local sock* dan menghubungkannya ke *remote socket*. Dengan menghubungkan komputer ke *socket*, maka komunikasi an komputer dapat dilakukan. Adapun mekanisme komunikasi *socket* seperti yang dinyatakan pada Gambar 1.



Gambar 1. Interface Socket

3. Metodologi Penelitian

Metodologi penelitian yang digunakan dalam menyelesaikan penelitian adalah sebagai berikut:

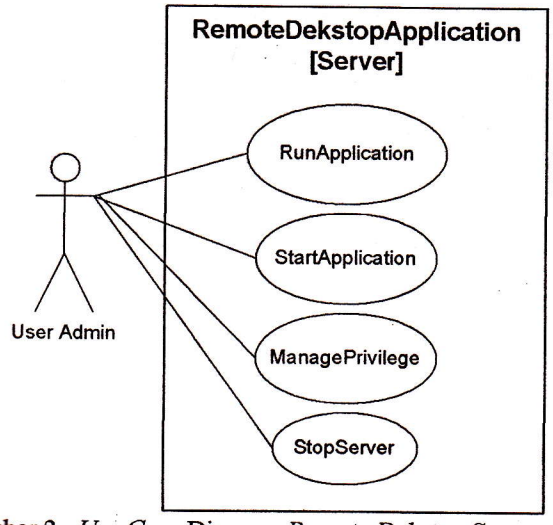
1. **Studi Pustaka** (Literatur), mencari sumber referensi yang berhubungan dengan pengembangan aplikasi *Remo Desktop*.
2. **Studi Sistem**, dilakukan dengan cara observasi dan pengambilan data yang berhubungan dengan pengembanga aplikasi *Remote Desktop* serta melakukan analisis dan rekayasa terhadap sistem yang berjalan sebelumny khususnya *Remote Desktop* Sistem Operasi Windows dan Macintosh.
3. **Pengembangan Sistem**, model pengembangan perangkat lunak yang dilakukan adalah menggunakan mod *waterfall* dengan mengikuti tahap-tahap analisis, desain, pengkodean, *testing*, dan *deploy* aplikasi.

4. Hasil Penelitian

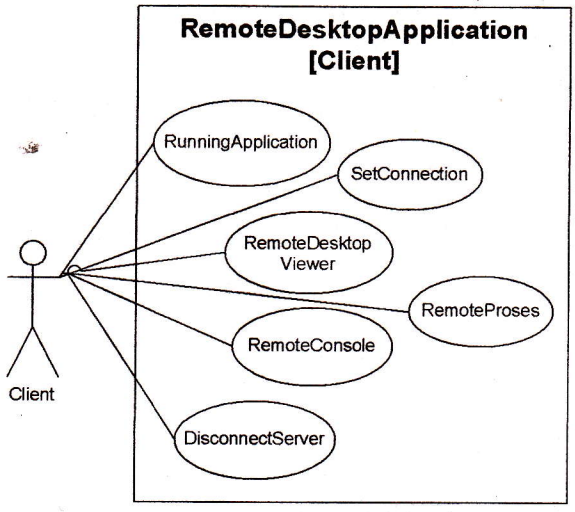
Sub bab berikut ini membahas tentang analisis kebutuhan sistem yang dilakukan, perancangan sistem, impelementas sistem, dan pengujian terhadap aplikasi yang telah dikembangkan.

4.1 Analisis Kebutuhan Sistem

Berdasarkan permasalahan yang terjadi, peneliti menemukan requirement pada bagian *server* seperti yang dinyataka pada Gambar 2 dan bagian *client* seperti yang dinyatakan pada Gambar 3.



Gambar 2. *Use Case* Diagram *Remote Dekstop Server*



Gambar 3. *Use Case* Diagram *Remote Dekstop Client*

4.1.1 Analisis Pengiriman *Capture Screen Remote Desktop*

Proses pengiriman screen *desktop server* ke *client* dilakukan dengan menggunakan *class Robot* Java, yaitu dengan menggunakan *method* **createScreenCapture**(Rectangle rect). Aplikasi *server* terlebih dahulu akan membagi *screen server* ke dalam **NUMBER_OF_REGION** wilayah, kemudian setiap wilayah akan diambil *screen desktop* yang akan diremote seperti yang dinyatakan pada Listing code 2. Aplikasi *Remote Desktop server* di dalam proses pengiriman *screen desktop* terlebih dahulu menentukan apakah terdapat perubahan screen. Jika terdapat perubahan screen pada wilayah tertentu maka wilayah yang mengalami perubahan akan dikirimkan oleh *server* ke *client* seperti yang dinyatakan pada Listing code 3. Pada sisi *client* akan menangkap informasi perubahan wilayah dan nilai **BufferedImage** yang dikirim oleh *server*, kemudian *client* akan melakukan perubahan tampilan screen di aplikasi *client*, seperti yang dinyatakan pada penggalan Listing code 4.

Listing code 2.

```
private CapturedScreenFactory( ) {
    setPriority(Thread.MAX_PRIORITY);
    try{
      setUpRectangle();
      robot = new Robot();
      // get image desktop
      fullImage = robot.createScreenCapture(new Rectangle(Toolkit.getDefaultToolkit().getScreenSize()));
      for(int i=0; i<NUMBER_OF_REGION; i++){
        try{
          image[i] = robot.createScreenCapture(rectangles[i]);
          int[] dataInt = ((DataBufferInt)image[i].getData().getDataBuffer()).getData();
          byte[] data = toByte(dataInt);
          capturedScreens[i] = new CapturedScreen(i,data);
          capturedScreens[i].setWidth(image[i].getWidth());
          capturedScreens[i].setHeight(image[i].getHeight());
        }
        catch(Exception e) {
          e.printStackTrace();
        }
      }
    }
    catch(AWTException e) {
      e.printStackTrace();
    }
  }
```

Listing code 3.

```
public void run( ){
    while(true) {
      try{
```

```
            CapturedScreen capturedScreen = capturedScreenFactory.getCapturedScreen(regionNumber);
            capturedScreens[regionNumber] = capturedScreen;

            if(capturedScreens[regionNumber].getByteImage()!=capturedScreen.getByteImage()){
               byte[] byteImage = capturedScreens[regionNumber].getByteImage();
               oos.writeInt(regionNumber);
               oos.flush();
               oos.writeInt(capturedScreens[regionNumber].getWidth());
               oos.flush();
               oos.writeInt(capturedScreens[regionNumber].getHeight());
               oos.flush();
               int length = byteImage.length;
               oos.writeInt(length);
               oos.flush();
               ByteArrayInputStream bais = new ByteArrayInputStream(byteImage);
               byte[] buffer = new byte[ServiceConstants.BUFFER_SIZE];
               int readLength;
               while((readLength=bais.read(buffer))!=-1) {
                  oos.write(buffer,0,readLength);
                  oos.flush();
               }
               oos.reset();
               System.gc();
            }
            regionNumber = (regionNumber+1)%NUMBER_OF_REGION;
         }
         catch(IOException ioe) {
            ioe.printStackTrace();
            break;
         }
      }
      System.gc();
      capturedScreenFactory.minNumberOfRequestCapturedScreen();
   }
```

Listing code 4.

```
image = new BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
 wr = image.getRaster();
 wr.setDataElements(0,0,width,height,data);
if(remoteDesktopViewer!=null && remoteDesktopViewer.isVisible()){
            remoteDesktopViewer.setCaptureScreen(capturedScreen.getRegionNumber(), image);
}
 myCanvas.setImageAt(capturedScreen.getRegionNumber(), image);
 myCanvas.repaint();
```

4.1.2 Analisis Pengiriman *Event Mouse* dan *Keyboard Remote Desktop*

Setiap *event requestService* yang dibangkitkan oleh *client* akan dikirimkan ke *server* untuk diproses *server*, seperti yang dinyatakan pada Listing code 5. Aplikasi *Remote Desktop client* dalam menangani permintaan *client* terhadap *event mouse* dan *keyboard* dilakukan dengan mengimplementasikan *interface ComponentListener, KeyListener, MouseListener, MouseMotionListener, MouseWheelListener, DropTargetListener* pada class *Remote Desktop Viewer client*. Dalam menjalankan *event mouse* dan *keyboard*, aplikasi *server* terlebih dahulu akan menentukan objek *event* yang diminta oleh *client*, kemudian *server* akan mengeksekusi objek *class Robot* secara *software* sesuai dengan permintaan objek *event client* seperti yang dinyatakan pada Listing code 6, sehingga aplikasi *client* dapat mengendalikan komputer *server* secara langsung.

Listing code 5.

```
int requestService = ois.readInt();
if(requestService==RDServiceConstants.COMPUTER_INFORMATION_SERVICE) {
         new ComputerInformationService(socket, ois, oos).start();
}
else if(requestService==RDServiceConstants.CAPTURED_SCREEN_SENDER){
         new CapturedScreenSender(socket, ois, oos).start();
```

```
}
else if(requestService==RDServiceConstants.KEYBOARD_AND_MOUSE_EVENT_SENDER) {
        new KeyboardAndMouseEventReceiver(socket, ois, oos).start();
}
else if(requestService==RDServiceConstants.PROCESS_RUNNING_SERVICE) {
        new ProcessRunningService(socket, ois, oos).start();
}
```

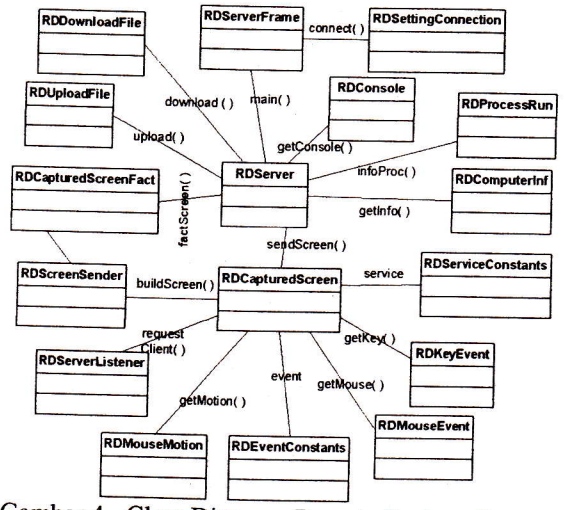
Listing code 6.

```
public void run(){
    while(true) {
      try{
        final Object obj = ois.readObject();
        Thread executeMouseThread = new Thread(){
          @Override
          public void run(){
            if(obj instanceof RDKeyEvent) {
              RDKeyEvent event = (RDKeyEvent)obj;
              if(event.getType()==RDEventConstants.KEY_PRESSED) {
                robot.keyPress(event.getKeyCode());
              }
              else if(event.getType()==RDEventConstants.KEY_RELEASED) {
                robot.keyRelease(event.getKeyCode());
              }
            }
            else if(obj instanceof RDMouseMotionEvent) {
              RDMouseMotionEvent event = (RDMouseMotionEvent)obj;
              Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
              int x = (int)((screenSize.getWidth()/event.getWidth())*event.getX());
              int y = (int)((screenSize.getHeight()/event.getHeight())*event.getY());
              robot.mouseMove(x,y);
            }
            else if(obj instanceof RDMouseEvent) {
              RDMouseEvent event = (RDMouseEvent)obj;
              if(event.getType()==RDEventConstants.MOUSE_PRESSED) {
                robot.mousePress(event.getModifiers());
              }
              else if(event.getType()==RDEventConstants.MOUSE_RELEASED) {
                robot.mouseRelease(event.getModifiers());
              }
            }
            System.gc();
          }
        };
        executeMouseThread.setPriority(Thread.MAX_PRIORITY);
        executeMouseThread.start();
      }
      catch(ClassNotFoundException e) {
        e.printStackTrace();
        break;
      }
      catch(IOException e) {
        e.printStackTrace();
        break;
      }
    }
  }
```

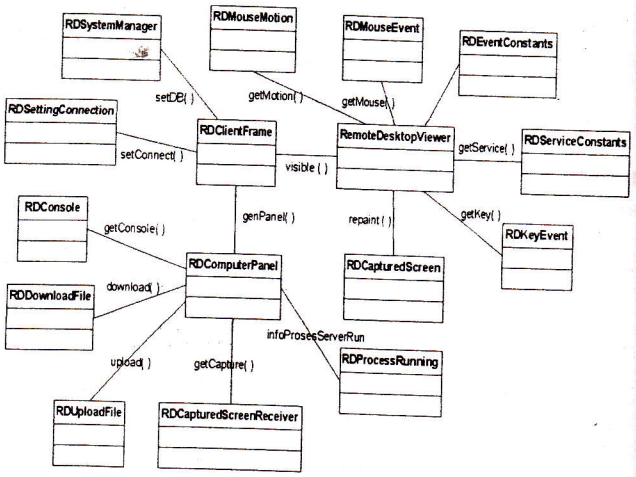
4.2 Perancangan Sistem

Berdasarkan analisis kebutuhan sistem dan kebutuhan fungsionalitas yang harus dipenuhi dalam memfasilitasi kebutuhan perangkat lunak seperti yang dinyatakan pada use case diagram pada Gambar 2 dan Gambar 3, maka peneliti

mendapatkan *class diagram* dari aplikasi *Remote Desktop* pada bagian *server* seperti yang dinyatakan pada Gambar sedangkan pada bagian *client* diperoleh *class diagram* seperti yang dinyatakan pada Gambar 5.



Gambar 4. Class Diagram *Remote Desktop Server*



Gambar 5. Class Diagram *Remote Desktop Client*

4.3 Implementasi Sistem

Untuk mengimplementasikan perancangan sistem yang dinyatakan pada Gambar 4 dan Gambar 5, membutuhkan bahas pemrograman Java JDK1.6.7 atau versi yang lebih tinggi. Selain itu juga menggunakan *software* pendukung lainnya seperti *database* MySQL *Server* 5.0 untuk pengaturan hak akses pengguna melalui login, dan *NetBeans* 6.8 IDE.

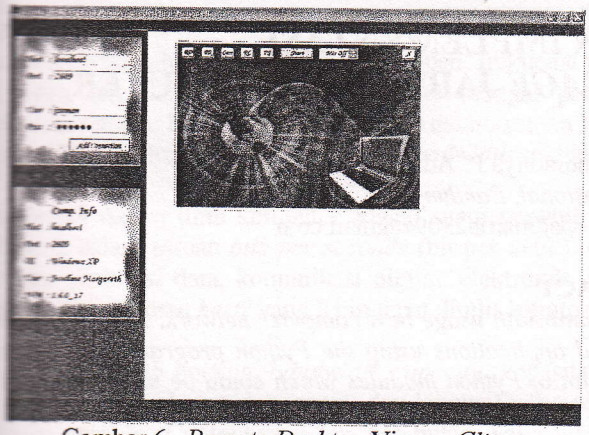
4.4 Teknik Pengujian

Teknik pengujian yang digunakan adalah teknik pengujian *black box testing*. Pengujian ini memungkinkan pemrogram untuk memperoleh sekumpulan kondisi masukan (*input*) yang akan secara penuh menjalankan semua kebutuha fungsional untuk sebuah program. Dalam hal ini diambil salah satu butir uji, yaitu *Remote Desktop Viewer*, seperti yan ditunjukkan pada Tabel 2.

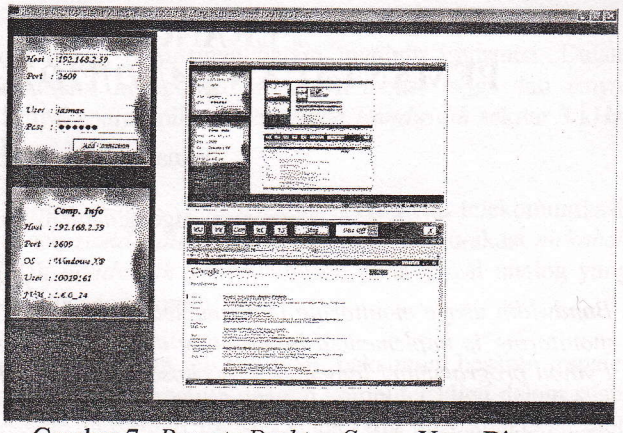
Tabel 2 Pengujian *Remote Desktop Viewer*

Identifikasi	ARDV -03		
Nama Butir Uji	*Remote Desktop Viewer*		
Tujuan	Media dalam memfasilitasi pengguna untuk menampilkan tampilan *Remote Desktop* pada PC *client*		
Deskripsi	Komunikasi dengan teknologi *wireless* atau LAN sudah terhubung dengan baik, pengguna sudah berada pada menu utama *client*, kemudian pengguna akan memilih menu tombol ***Start*** dari ***Remote Desktop Viewer***		
Kondisi Awal	Pengguna sudah terkoneksi dengan komputer *server* yang akan di *remote* dan sudah berada pada menu utama *client*		
Pengujian			
Skenario Uji			
1. Pilih Tombol ***Start Remote Desktop Viewer***			
Kriteria Evaluasi Hasil			
Tunggu beberapa detik (tergantung dari spesifikasi komputer *server*) akan ditampilkan *desktop server* yang akan di *remote*			
Kasus dan Hasil Uji (Data Normal)			
Masukan	Harapan	Pengamatan	Kesimpulan
Action Event click Button Start	Menampilkan *remote desktop server* yang akan di *remote* oleh pengguna ke PC pengguna, tanpa harus me*lock* komputer *server* yang di*remote*	Menampilkan *remote desktop server* yang akan di*remote* oleh pengguna ke PC pengguna, tanpa harus me*lock* komputer *server* yang di*remote*	**[X]** Terima **[ ]** Tolak

Berdasarkan hasil pengujian yang dilakukan pengguna terhadap butir uji *Remote Desktop Viewer* dengan mengikuti skenario yang dinyatakan pada Tabel 2 untuk dua *server* yang akan di*remote* dengan menu utama *client* seperti Gambar 6, diperoleh hasil pengujian seperti yang ditunjukkan pada Gambar 7.



Gambar 6. *Remote Desktop* Viewer *Client*



Gambar 7. *Remote Desktop Server* Yang Di*remote*

5. Kesimpulan

Berdasarkan penelitian yang dilakukan, beberapa kesimpulan dapat diambil sebagai berikut:

1) Pada penelitian ini telah berhasil dibangun aplikasi *Remote* Dekstop berbasis PC yang dapat diakses melalui berbagai *device* (*multiplatform*) dan sudah mengimplementasikan fitur minimal aplikasi *Remote Desktop*.
2) Aplikasi *Remote Desktop* dikembangkan menggunakan *class Robot* Java dengan pertukaran data antara *client* dan *server* dengan dukungan pemrograman *Socket* Java.
3) Proses *Remote Desktop* PC *server* yang berjalan pada sisi *client*, dilakukan dengan pengiriman objek *event mouse* dan *keyboard* oleh *client* ke *server*, dijalankan secara *software* pada sisi *server* dengan mengimplementasikan *method-method class Robot* Java.
4) Aplikasi *Remote Dekstop* yang dikembangkan dapat memfasilitasi pengguna dalam melakukan *Remote Desktop* beberapa *server* (dua *server* atau lebih) secara bersamaan pada satu *client*, seperti yang dinyatakan pada Gambar 7.

Daftar Pustaka

[1] Apple. Apple - *Remote Desktop 3* - New Features, http://www.apple.com/remote*desktop* /newfeatures.html, diakses terakhir tanggal 17 Juli 2012.

[2] Microsoft. *The Features of the Remote Desktop Client in Windows XP*, http://support.microsoft.com/kb/300698, diakses terakhir tanggal 17 Juli 2012.

[3] Garrido., J., M. (2003). *Object-Oriented Programming: From Problem Solving to Java*, Charles River Media, Inc.Hingham, Massachusetts.

[4] Poo, D., Kiong, D., Ashok, S. (2008). *Object-Oriented Programming and Java*, Second Edition, Springer-Verlag, London.

[5] Class Robot Java, http://docs.oracle.com/Javase/1.5.0/docs/api, diakses terakhir tanggal 17 Juli 2012.

[6] Dov Bulka. (2001). *Java™ Performance and Scalability Volume 1: Server-Side Programming Techniques*, Addison Wesley.

[7] Robert Orfali and Dan Harkey. (1998). *Client/Server Programming with Java and CORBA*, 2nd Edition, John Wiley & Sons, Inc.