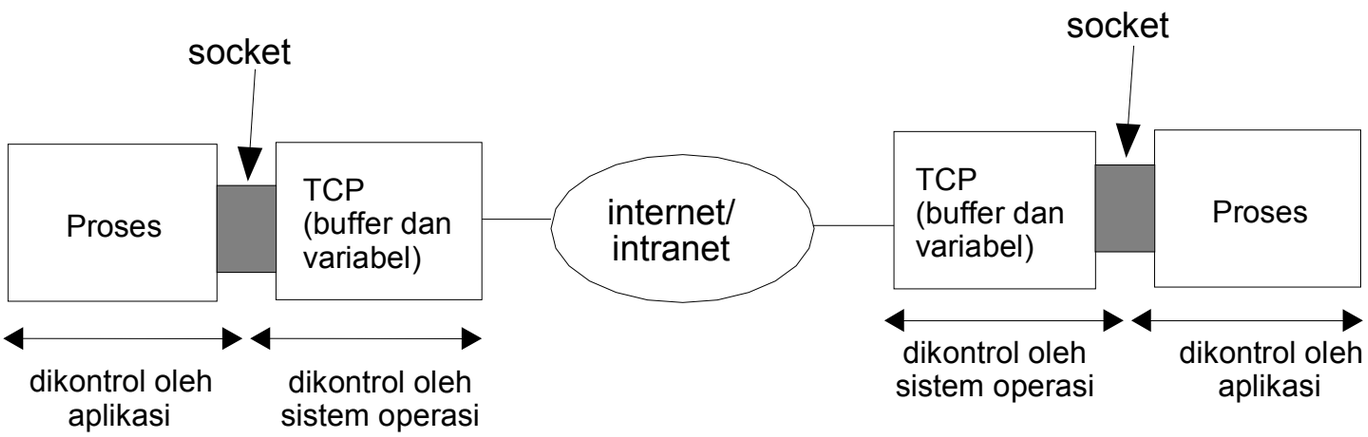
Pemrograman Client/Server dengan Java Socket

budi susanto (budsus@ukdw.ac.id)

Java Socket

- Socket adalah sebuah abstraksi perangkat lunak yang digunakan sebagai suatu "terminal" dari suatu hubungan antara dua mesin atau proses yang saling berinterkoneksi.
- Di tiap mesin yang saling berinterkoneksi, harus terpasang socket.
- Pada J2SE telah disediakan paket java.net yang berisi kelas-kelas dan interface yang menyediakan API (Application Programming Interface) level rendah (Socket, ServerSocket, DatagramSocket) dan level tinggi (URL, URLConnection).

INGAT: Socket akan membangun komunikasi antar proses yang sama-sama aktif.



Kelas Socket

Socket(InetAddress address, int port)

membuat sebuah stream socket dan koneksi ke suatu nomor port pada sebuah komputer yang memiliki alamat IP.

Socket(String host, int port)

membuat sebuah stream socket dan juga koneksi ke suatu port tertentu pada sebuah komputer berdasar namanya.

Socket(InetAddress address, int port, InetAddress localAddr, int localPort);

Socket(String host, int port, InetAddress localAddr, int localPort);

membuat sebuah socket dan mengkoneksikannya ke port yang dituju pada alamat IP yang disebutkan pada parameter address atau nama host. Selain itu juga akan dilakukan bind socket ke alamat lokal dan port lokal. (Hal ini dilakukan jika koneksi antara client dan server membutuhkan nomor port yang sudah ditentukan.

getInetAddress()

untuk mendapatkan nama host yang dituju dan alamat IPnya

getPort()

untuk mendapatkan nomor remote host

getLocalPort()

untuk mendapatkan nomor port localhost

getLocalAddress()

untuk mendapatkan alamat local dimana socket digunakan

getInputStream()

mengembalikan objek input stream dari socket

getOutputStream()

mengembalikan objek output stream ke socket

setSoTimeout(int timeout)

getSoTimeOut()

Kedua method tersebut digunakan untuk memberi (set) dan

mengambil (get) nilai opsi Socket untuk time out block (dalam milidetik) reading dari socket (SO_TIMEOUT). Jika dalam waktu timeout tidak mendapat suatu nilai maka, akan dilemparkan ke exception **java.net.SocketTimeoutException**. Nilai default timeoutnya adalah 0, yang berarti tanpa batas.

setTCPNoDelay(boolean on)

getTCPNoDelay()

Kedua method ini digunakan untuk memberi dan mengambil nilai opsi Socket TCP_NODELAY, yaitu untuk mengaktifkan atau menonaktifkan Algoritma Nagle (RFC 896), yaitu algoritma yang membuat TCP lebih efisien dalam konsumsi bandwidth dengan cara memperlambat penulisan data dalam ukuran yang kecil sehingga data-data yang ada dapat terkirimkan dalam suatu paket dengan ukuran besar. Nilai default opsi ini adalah aktif. Namun jika diinginkan adanya pengurangan network latency (waktu delay dalam pengiriman paket) dan meningkatkan unjuk kerja, maka opsi ini harus di nonaktifkan (di set dengan nilai false), namun akibatnya konsumsi bandwidth akan bertambah besar.

setSoLinger(boolean on, int linger)

getSoLinger()

Method tersebut akan mengaktifkan (true) atau menonaktifkan (false) opsi SO_LINGER dengan nilai waktu linger dalam milidetik. Opsi ini berpengaruh ketika socket ditutup, yaitu menentukan nilai waktu maksikum koneksi yang masih akan dipertahankan sampai socket koneksi benar-bernar ditutup. Hal ini berguna untuk mengirim dan memberikan ACK (acknowledge) terhadap data yang belum terkirim.

setSendBufferSize(int size)

getSendBufferSize()

Method ini akan mengatur dan mengambil informasi tentang ukuran buffer SO_SNDBUF, yaitu buffer untuk mengiriman. Ukuran ini juga harus disesuaikan ukuran buffer pada level

network.

setReceiveBufferSize(int size)

getReceiveBufferSize()

Method ini digunakan jika Anda ingin mengatur ukuran buffer SO_RCVBUF, yaitu buffer yang digunakan untuk menampung paket yang masuk. Ukuran buffer ini juga digunakan untuk mengatur ukuran window yang diterapkan oleh TCP untuk flow controlnya (sliding window). Dalam pemanfaatan opsi ini, perlu dipastikan antara RCVBUF client dengan server ada sinkronisasi, sehingga sebelum server membind port yang akan digunakan ke socket (TCP) ataupun sebelum client membuka koneksi ke server, terlebih dahulu opsi ini harus di atur, jika Anda ingin mengaturnya.

Kelas-kelas Exception yang dibangkitkan Socket, ketika ada kesalahan :

SocketException

Kelas ini merupakan kelas yang diturunkan dari kelas IOException. Kelas exception ini dipanggil atau dipicu ketika ada kegagalan dalam pemakaian socket, sebagai contoh adalah kegagalan dalam protokol TCP.

Salah satu penyebabnya yang mungkin terjadi adalah ketika port yang akan digunakan sudah digunakan sebelumnya pada lokalhost. Penyebab yang lain adalah user tidak dapat melakukan bind ke port yang dituju. Misalnya saja, Anda ingin menggunakan port 80 untuk aplikasi Anda, namun ternyata pada komputer Anda tersebut sudah berjalan HTTP Server yang juga menggunakan port 80. Bila hal ini terjadi, maka JVM akan melemparkan kegagalan yang ada ke kelas exception SocketException.

BindException

Exception ini akan dipanggil ketika ada port lokal yang akan digunakan sudah terpakai oleh yang lain, atau ada kegagalan dalam permintaan untuk menggunakan alamat.

ConnectException

Exception ini akan dipanggil ketika sebuah koneksi ditolak oleh host yang dituju, oleh karena tidak ada proses yang siap menerima data pada port yang dituju.

NoRouteToHostException

Koneksi yang akan dibangun tidak dapat dipenuhi oleh karena melebihi waktu timeout yang tersedia atau host yang dituju tidak dapat dicapai (unreachable).

Contoh :

```
import java.io.*;
import java.net.*;

public class ExHTTPClient {
  public static void main(String args[]) {
     try {
        Socket clientSocket =
               new Socket(args[0], 80);
        System.out.println("Client: " +
               clientSocket);
        getHTML(clientSocket, args[1]);
     }
     catch (UnknownHostException e)
     { System.out.println(e); }
     catch (IOException e)
     { System.err.println(e); }
  }

  public static void getHTML(Socket clientSocket,
                             String fileName)
  {
     try {
       DataOutputStream outbound =
```

```
            new DataOutputStream(
                clientSocket.getOutputStream() );
        DataInputStream inbound =
          new DataInputStream(
                clientSocket.getInputStream() );
        outbound.writeBytes("GET " + fileName +
                " HTTP/1.0\r\n\r\n");
        String responseLine;
        while ((responseLine = inbound.readLine())
                != null) {
          System.out.println(responseLine);
        }
        outbound.close();
        inbound.close();
        clientSocket.close();
      }
      catch (IOException e)
      { System.out.println(e); }
   }
}
```

Contoh Hasil :

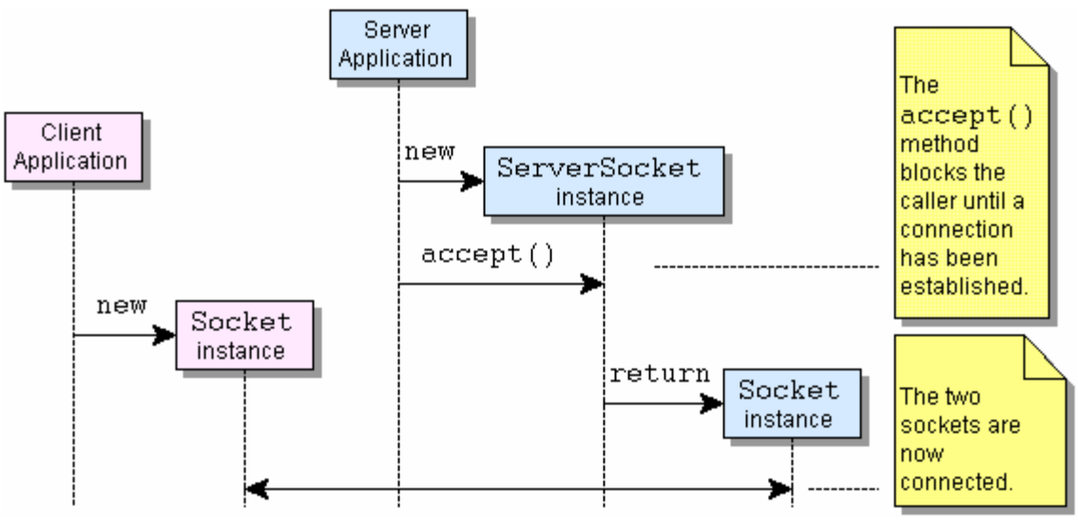
```
# java ExHTTPClient localhost /test.html
Client1:Socket[addr=localhost/127.0.0.1,port=80,localport=32777]
HTTP/1.1 200 OK
Date: Sat, 01 Mar 2003 13:10:34 GMT
Server: Apache/1.3.23 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.7
OpenSSL/0.9.6b DAV/1.0.3 PHP/4.1.2 mod_perl/1.26
Last-Modified: Sat, 01 Mar 2003 13:02:20 GMT
ETag: "932fe-44-3e60af5c"
Accept-Ranges: bytes
Content-Length: 68
Connection: close
Content-Type: text/html
```

```
<HTML>
<BODY>
Dokumen ini diakses dari Apache HTTP.
</BODY>
</HTML>
```

Kelas ServerSocket

ServerSocket(int port [, int backlog [, InetAddress bindAddress]])

membuat sebuah server dengan port tertentu, batasan jumlah antrian (backlog), dan alamat IP bindAddress.



InfoServer.java

```
import java.io.*;
import java.net.*;
import java.util.*;

public class InfoServer {
  private final int INFO_PORT=50000;
  private String datafromClient;

  public InfoServer() {
    BufferedReader inFromClient;
```

```
DataOutputStream outToClient;
Socket serverSocket;

try {
  ServerSocket infoServer =
       new ServerSocket(INFO_PORT);

  System.out.println("Server telah siap...");

  while (true) {
    serverSocket = infoServer.accept();
    System.out.println("Ada client " +
                       "yang terkoneksi!");

    inFromClient =
         new BufferedReader(
         new InputStreamReader(
             serverSocket.getInputStream()));

    outToClient =
          new DataOutputStream(
              serverSocket.getOutputStream());

   outToClient.writeBytes("InfoServer versi 0.1\n"+
                          "hanya untuk testing..\n"+
  "Silahkan berikan perintah TIME | NET | QUIT\n");

   boolean isQUIT = false;
   while (!isQUIT) {
     datafromClient = inFromClient.readLine();

     if (datafromClient.startsWith("TIME")) {
       outToClient.writeBytes(new
```

```
                    Date().toString() + "\n");
            } else if (datafromClient.startsWith("NET")) {
              outToClient.writeBytes(
          InetAddress.getByName("budsusothie").toString() +
                    "\n");
            } else if (datafromClient.startsWith("QUIT"))
            {
               isQUIT = true;
            }
          }
          outToClient.close();
          inFromClient.close();
          serverSocket.close();

          System.out.println("Koneksi client tertutup..");
        }
      }
      catch (IOException ioe) {
        System.out.print("error: " + ioe);
      }
      catch (Exception e) {
        System.out.print("error: " + e);
      }
    }

    /* program utama */
    public static void main(String[] args) {
       new InfoServer();
    }
}
```

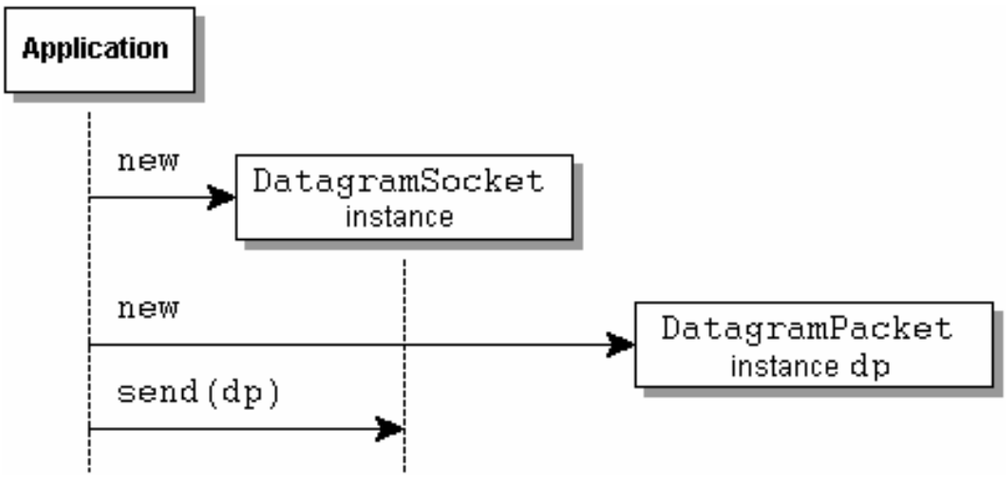
InfoClient.java

```
import java.net.*;
import java.io.*;
import java.util.*;

public class InfoClient {
  private final int INFO_PORT=50000;
  private final String TargetHost = "localhost";
  private final String QUIT = "QUIT";

  public InfoClient() {
    try {
      BufferedReader inFromUser =
         new BufferedReader(new
             InputStreamReader(System.in));

      Socket clientSocket = new
           Socket(TargetHost, INFO_PORT);
      DataOutputStream outToServer =
         new DataOutputStream(
             clientSocket.getOutputStream());

      BufferedReader inFromServer =
         new BufferedReader(
             new InputStreamReader(
                 clientSocket.getInputStream()));

      System.out.println(inFromServer.readLine());
      System.out.println(inFromServer.readLine());
      System.out.println(inFromServer.readLine());
      System.out.println("");

      boolean isQuit = false;
```

```
      while (!isQuit) {
        System.out.print("Perintah Anda : ");
        String cmd = inFromUser.readLine();

        cmd = cmd.toUpperCase();
        if (cmd.equals(QUIT)) {
           isQuit = true;
        }

        outToServer.writeBytes(cmd + "\n");
        String result = inFromServer.readLine();
        System.out.println("Dari Server: " + result);
      }

      outToServer.close();
      inFromServer.close();
      clientSocket.close();
    }
    catch (IOException ioe) {
       System.out.println("Error:" + ioe);
    }
    catch (Exception e) {
       System.out.println("Error:" + e);
    }
  }

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

Kelas **DatagramSocket**

Digunakan untuk membangun koneksi connectionless dengan protokol UDP.



DatagramSocket(int port)

Kelas ini dapat digunakan untuk menyatakan penggunaan suatu nomor port sebagai "pintu" untuk menerima koneksi dari client.

DatagramSocket(int port, InetAddress laddr)

Kelas ini membentuk koneksi dengan protokol UDP pada alamat IP lokal tertentu dan pada nomor port tertentu.

DatagramSocket()

Kelas ini membentuk koneksi dengan protokol UDP pada alamat IP lokal host dengan penentuan nomor portnya secara random berdasar tersedianya nomor port yang dapat digunakan.

DatagramPacket(byte[] buf, int length)

Kelas ini dapat digunakan untuk mengambil informasi. Constructor ini membutuhkan sebuah array byte yang menjadi parameter pertama, yang berfungsi untuk menyimpan data dan informasi ukuran data yang diterima.

DatagramPacket(byte[] buf, int length,

InetAddress address, int port)

Constructor ini digunakan untuk membuat paket Datagram yang akan mengirim data. Constructor ini memerlukan informasi array byte yang akan dikirim dan panjangnya, serta alamat dan port yang dituju.

InfoServerUDP.java

```
import java.io.*;
import java.net.*;
import java.util.*;

public class InfoServerUDP {
  private final int INFO_PORT=50000;
  private String datafromClient;

  public InfoServerUDP() {
    DatagramSocket serverSocket;

    try {
      serverSocket = new DatagramSocket(INFO_PORT);
      System.out.println("Server telah siap...");

      while (true) {
        boolean isQUIT = false;
        while (!isQUIT) {
          byte[] byteFromClient = new byte[1024];
          byte[] byteToClient = new byte[1024];

          DatagramPacket receivePacket =
            new DatagramPacket(
               byteFromClient, byteFromClient.length);
```

```
        serverSocket.receive(receivePacket);

        InetAddress IPAddress =
             receivePacket.getAddress();
        int port = receivePacket.getPort();

        String data =
             new String(receivePacket.getData());

        if (data.startsWith("TIME")) {
          String DateNow =
              new String(new Date().toString());
          byteToClient = DateNow.getBytes();

        } else if (data.startsWith("NET")) {
          String hostname = new String(
            InetAddress.getByName("xxx").toString());
          byteToClient = hostname.getBytes();

        } else if (data.startsWith("QUIT")) {
          isQUIT = true;
          String thanks =
              new String("Terima kasih!");
          byteToClient = thanks.getBytes();
        }
        DatagramPacket sendPacket =
          new DatagramPacket(byteToClient,
             byteToClient.length, IPAddress, port);
        serverSocket.send(sendPacket);
      }
      System.out.println("Hub. client tertutup..");
    }
}
```

```
    catch (IOException ioe) {
      System.out.print("error: " + ioe);
    }
    catch (Exception e) {
      System.out.print("error: " + e);
    }
  }
  public static void main(String[] args) {
    new InfoServerUDP();
  }
}
```

InfoClientUDP.java

```
import java.net.*;
import java.io.*;
import java.util.*;

public class InfoClientUDP {
  private final int INFO_PORT=50000;
  private final String TargetHost = "localhost";
  private final String QUIT = "QUIT";
  private DatagramSocket clientSocket;

  public InfoClientUDP() {
    try {
      BufferedReader inFromUser =
         new BufferedReader(
           new InputStreamReader(System.in));

      clientSocket = new DatagramSocket();

      InetAddress IPAddress =
         InetAddress.getByName("localhost");
```

```
  boolean isQuit = false;
  while (!isQuit) {
    byte[] byteFromServer = new byte[1024];
    byte[] byteToServer = new byte[1024];

    System.out.print("Perintah Anda : ");
    String cmd = inFromUser.readLine();

    cmd = cmd.toUpperCase();
    isQuit = cmd.equals(QUIT);
    byteToServer = cmd.getBytes();

    DatagramPacket sendPacket =
       new DatagramPacket(byteToServer,
       byteToServer.length, IPAddress, INFO_PORT);
    clientSocket.send(sendPacket);

    DatagramPacket receivePacket =
      new DatagramPacket(byteFromServer,
                    byteFromServer.length);

    clientSocket.receive(receivePacket);
    String result =
       new String(receivePacket.getData());
    System.out.println("Dari Server: " + result);
  }

  clientSocket.close();
}
catch (IOException ioe) {
  System.out.println("Error:" + ioe);
}
```

```
    catch (Exception e) {
      System.out.println("Error:" + e);
    }
  }

  /* program utama */
  public static void main(String[] args) {
    new InfoClientUDP();
  }
}
```