Laporan Resmi Prak 14

Praktikum Konsep Jaringan



Nama : Muhammad zaid abdillah

Kelas : 2 D4 Teknik Informatika A

NRP : 2110191013

1. Percobaan 1
2. Listing program

EchoServer.java

import java.net.\*;

import java.io.\*;

public class EchoServer

{

 public static void main(String[] args) throws IOException

   {

    ServerSocket serverSocket = null;

    try {

         serverSocket = new ServerSocket(10007);

        }

    catch (IOException e)

        {

         System.err.println("Could not listen on port: 10007.");

         System.exit(1);

        }

    Socket clientSocket = null;

    System.out.println ("Waiting for connection.....");

    try {

         clientSocket = serverSocket.accept();

        }

    catch (IOException e)

        {

         System.err.println("Accept failed.");

         System.exit(1);

        }

    System.out.println ("Connection successful");

    System.out.println ("Waiting for input.....");

    PrintWriter out = new PrintWriter(clientSocket.getOutputStream(),

                                      true);

    BufferedReader in = new BufferedReader(

            new InputStreamReader( clientSocket.getInputStream()));

    String inputLine;

    while ((inputLine = in.readLine()) != null)

        {

         System.out.println ("Server: " + inputLine);

         out.println(inputLine);

         if (inputLine.equals("Bye."))

             break;

        }

    out.close();

    in.close();

    clientSocket.close();

    serverSocket.close();

   }

}

EchoClient.java

import java.io.\*;

import java.net.\*;

public class EchoClient {

    public static void main(String[] args) throws IOException {

        String serverHostname = new String ("127.0.0.1");

        if (args.length > 0)

           serverHostname = args[0];

        System.out.println ("Attemping to connect to host " +

        serverHostname + " on port 10007.");

        Socket echoSocket = null;

        PrintWriter out = null;

        BufferedReader in = null;

        try {

            // echoSocket = new Socket("taranis", 7);

            echoSocket = new Socket(serverHostname, 10007);

            out = new PrintWriter(echoSocket.getOutputStream(), true);

            in = new BufferedReader(new InputStreamReader(

                                        echoSocket.getInputStream()));

        } catch (UnknownHostException e) {

            System.err.println("Don't know about host: " + serverHostname);

            System.exit(1);

        } catch (IOException e) {

            System.err.println("Couldn't get I/O for "

                               + "the connection to: " + serverHostname);

            System.exit(1);

        }

    BufferedReader stdIn = new BufferedReader(

                                   new InputStreamReader(System.in));

    String userInput;

        System.out.print ("input: ");

    while ((userInput = stdIn.readLine()) != null) {

        out.println(userInput);

        System.out.println("echo: " + in.readLine());

            System.out.print ("input: ");

    }

    out.close();

    in.close();

    stdIn.close();

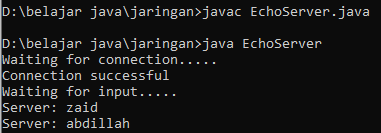
    echoSocket.close();

    }

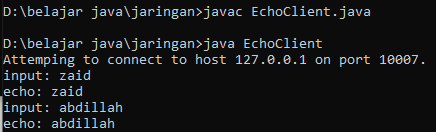
}

1. Output

EchoServer.java



EchoClient.java



1. Analisa

Dalam percobaan diatas, kita melakukan percobaan, kita menjalankan server.php sebagai server untuk diakses oleh client. Setelah kita menjalankan server.php maka server akan terbuka dan dapat diakses oleh client. Setelah itu, kita bisa menjalankan clientnya. Setelah itu, kita bisa mengakses server melalui client. Dengan saya mengetikkan inputan berupa “dadang” maka dalam client akan muncul “zaid” dan di server “zaid“ juga. Kesimpulannya yang kita inputkan akan muncul di server dan akan muncul lagi di client.

1. Percobaan 2
2. Listing Program

EchoServer2.java

import java.net.\*;

import java.io.\*;

public class EchoServer2 extends Thread

{

 protected Socket clientSocket;

 public static void main(String[] args) throws IOException

   {

    ServerSocket serverSocket = null;

    try {

         serverSocket = new ServerSocket(10008);

         System.out.println ("Connection Socket Created");

         try {

              while (true)

                 {

                  System.out.println ("Waiting for Connection");

                  new EchoServer2 (serverSocket.accept());

                 }

             }

         catch (IOException e)

             {

              System.err.println("Accept failed.");

              System.exit(1);

             }

        }

    catch (IOException e)

        {

         System.err.println("Could not listen on port: 10008.");

         System.exit(1);

        }

    finally

        {

         try {

              serverSocket.close();

             }

         catch (IOException e)

             {

              System.err.println("Could not close port: 10008.");

              System.exit(1);

             }

        }

   }

 private EchoServer2 (Socket clientSoc)

   {

    clientSocket = clientSoc;

    start();

   }

 public void run()

   {

    System.out.println ("New Communication Thread Started");

    try {

         PrintWriter out = new PrintWriter(clientSocket.getOutputStream(),

                                      true);

         BufferedReader in = new BufferedReader(

                 new InputStreamReader( clientSocket.getInputStream()));

         String inputLine;

         while ((inputLine = in.readLine()) != null)

             {

              System.out.println ("Server: " + inputLine);

              out.println(inputLine);

              if (inputLine.equals("Bye."))

                  break;

             }

         out.close();

         in.close();

         clientSocket.close();

        }

    catch (IOException e)

        {

         System.err.println("Problem with Communication Server");

         System.exit(1);

        }

    }

}

EchoClient2.java

import java.io.\*;

import java.net.\*;

public class EchoClient2 {

    public static void main(String[] args) throws IOException {

        String serverHostname = new String ("127.0.0.1");

        if (args.length > 0)

           serverHostname = args[0];

        System.out.println ("Attemping to connect to host " +

                serverHostname + " on port 10008.");

        Socket echoSocket = null;

        PrintWriter out = null;

        BufferedReader in = null;

        try {

            echoSocket = new Socket(serverHostname, 10008);

            out = new PrintWriter(echoSocket.getOutputStream(), true);

            in = new BufferedReader(new InputStreamReader(

                                        echoSocket.getInputStream()));

        } catch (UnknownHostException e) {

            System.err.println("Don't know about host: " + serverHostname);

            System.exit(1);

        } catch (IOException e) {

            System.err.println("Couldn't get I/O for "

                               + "the connection to: " + serverHostname);

            System.exit(1);

        }

    BufferedReader stdIn = new BufferedReader(

                                   new InputStreamReader(System.in));

    String userInput;

        System.out.println ("Type Message (\"Bye.\" to quit)");

    while ((userInput = stdIn.readLine()) != null)

           {

        out.println(userInput);

            // end loop

            if (userInput.equals("Bye."))

                break;

        System.out.println("echo: " + in.readLine());

       }

    out.close();

    in.close();

    stdIn.close();

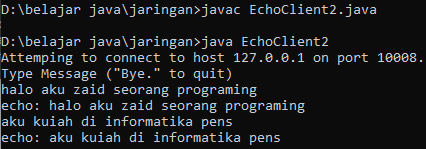
    echoSocket.close();

    }

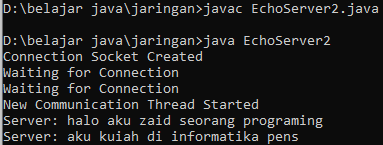
}

1. Output

EchoClient2.java



EchoServer2.java



1. Analisa

Pada percobaan ini, secara umum sama dengan percobaan 1 tetapi bedanya disini dalam satu server terdapat 2 klient. Sehingga kita bisa mengakses server dengan 2 klien. Dalam percobaan tersebut saya memakai 2 cmd yang menjalankan EchoClient2.java dengan secara bersamaan. Setelah itu kita input kata yang berbeda di klien tersebut. Maka dua kata tersebut akan muncul dalam satu server.

1. Percobaan 3
2. Listing program

UDPServer.java

import java.io.\*;

import java.net.\*;

class UDPServer {

  public static void main(String args[]) throws Exception

    {

     try

     {

      DatagramSocket serverSocket = new DatagramSocket(9876);

      byte[] receiveData = new byte[1024];

      byte[] sendData  = new byte[1024];

      while(true)

        {

          receiveData = new byte[1024];

          DatagramPacket receivePacket =

             new DatagramPacket(receiveData, receiveData.length);

          System.out.println ("Waiting for datagram packet");

          serverSocket.receive(receivePacket);

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

          InetAddress IPAddress = receivePacket.getAddress();

          int port = receivePacket.getPort();

          System.out.println ("From: " + IPAddress + ":" + port);

          System.out.println ("Message: " + sentence);

          String capitalizedSentence = sentence.toUpperCase();

          sendData = capitalizedSentence.getBytes();

          DatagramPacket sendPacket =

             new DatagramPacket(sendData, sendData.length, IPAddress,

                               port);

          serverSocket.send(sendPacket);

        }

     }

      catch (SocketException ex) {

        System.out.println("UDP Port 9876 is occupied.");

        System.exit(1);

      }

    }

}

UDPClient.java

import java.io.\*;

import java.net.\*;

class UDPClient {

    public static void main(String args[]) throws Exception

    {

     try {

        String serverHostname = new String ("127.0.0.1");

        if (args.length > 0)

           serverHostname = args[0];

      BufferedReader inFromUser =

        new BufferedReader(new InputStreamReader(System.in));

      DatagramSocket clientSocket = new DatagramSocket();

      InetAddress IPAddress = InetAddress.getByName(serverHostname);

      System.out.println ("Attemping to connect to " + IPAddress +

                          ") via UDP port 9876");

      byte[] sendData = new byte[1024];

      byte[] receiveData = new byte[1024];

      System.out.print("Enter Message: ");

      String sentence = inFromUser.readLine();

      sendData = sentence.getBytes();

      System.out.println ("Sending data to " + sendData.length +

                          " bytes to server.");

      DatagramPacket sendPacket =

         new DatagramPacket(sendData, sendData.length, IPAddress, 9876);

      clientSocket.send(sendPacket);

      DatagramPacket receivePacket =

         new DatagramPacket(receiveData, receiveData.length);

      System.out.println ("Waiting for return packet");

      clientSocket.setSoTimeout(10000);

      try {

           clientSocket.receive(receivePacket);

           String modifiedSentence =

               new String(receivePacket.getData());

           InetAddress returnIPAddress = receivePacket.getAddress();

           int port = receivePacket.getPort();

           System.out.println ("From server at: " + returnIPAddress +

                               ":" + port);

           System.out.println("Message: " + modifiedSentence);

          }

      catch (SocketTimeoutException ste)

          {

           System.out.println ("Timeout Occurred: Packet assumed lost");

      }

      clientSocket.close();

     }

   catch (UnknownHostException ex) {

     System.err.println(ex);

    }

   catch (IOException ex) {

     System.err.println(ex);

    }

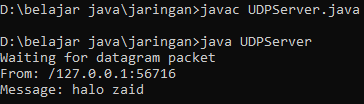
  }

}

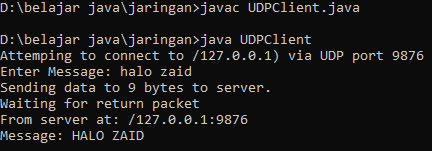
1. Output

Percobaan pertama

UDPServer.java



UDPClient.java



1. Analisa

Dalam percobaan ke 3 ini, kita menjalankan server UDPServer.java untuk menjalankan server yang bisa diakses oleh client. Setelah itu kita jalankan clien, dan input kalimat atau kata. Setelah kita menginputkan, maka kita langsung keluar dari program yang dijalankan. Artinya kita hanya dapat menjalankan client hanya satu kali. Setelah itu, di server muncul kata yang diinput dan muncul alamat ip atau port.

1. Percobaan 4
2. Listing program

UDPServer.java

import java.io.\*;

import java.net.\*;

class UDPServer {

  public static void main(String args[]) throws Exception

    {

     try

     {

      DatagramSocket serverSocket = new DatagramSocket(9876);

      byte[] receiveData = new byte[1024];

      byte[] sendData  = new byte[1024];

      while(true)

        {

          receiveData = new byte[1024];

          DatagramPacket receivePacket =

             new DatagramPacket(receiveData, receiveData.length);

          System.out.println ("Waiting for datagram packet");

          serverSocket.receive(receivePacket);

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

          InetAddress IPAddress = receivePacket.getAddress();

          int port = receivePacket.getPort();

          System.out.println ("From: " + IPAddress + ":" + port);

          System.out.println ("Message: " + sentence);

          String capitalizedSentence = sentence.toUpperCase();

          sendData = capitalizedSentence.getBytes();

          DatagramPacket sendPacket =

             new DatagramPacket(sendData, sendData.length, IPAddress,

                               port);

          serverSocket.send(sendPacket);

        }

     }

      catch (SocketException ex) {

        System.out.println("UDP Port 9876 is occupied.");

        System.exit(1);

      }

    }

}

UDPClient2.java

import java.io.\*;

import java.net.\*;

import java.util.\*;

class UDPClient2

{

 private InetAddress IPAddress;

 boolean done;

 boolean keepGoing;

 public UDPClient2(String sHostName)

  {

   String s1;

   ArrayList lines = new ArrayList();

   int size;

   BufferedReader br;

   try {

        IPAddress = InetAddress.getByName(sHostName);

        System.out.println ("Attemping to connect to " + IPAddress +

                          ") via UDP port 9876");

       }

   catch (UnknownHostException ex)

       {

        System.err.println(ex);

        System.exit(1);

       }

  // set up the buffered reader to read from the keyboard

  try {

       br = new BufferedReader (new FileReader ("UDPInputFile.txt"));

       s1 = br.readLine();

       while (s1 != null)

         {

          lines.add(s1);

          s1 = br.readLine ();

         }

       size = lines.size();

       System.out.println ("ArrayList lines has size of: " + size);

       done = false;

       DatagramSocket clientSocket = new DatagramSocket();

       for (int i = 0; i < size ; i++)

          {

           byte[] sendData = new byte[1024];

           s1 = (String) lines.get(i);

           sendData = s1.getBytes();

           System.out.println ("Sending data to " + sendData.length +

                               " bytes to server from line " + (i + 1));

           DatagramPacket sendPacket =

              new DatagramPacket(sendData, sendData.length, IPAddress, 9876);

           clientSocket.send(sendPacket);

          }

       done = true;

       byte[] receiveData = new byte[1024];

       keepGoing = true;

       DatagramPacket receivePacket =

            new DatagramPacket(receiveData, receiveData.length);

       System.out.println ("Waiting for return packet");

       clientSocket.setSoTimeout(10000);

       while (keepGoing)

          {

           try {

              clientSocket.receive(receivePacket);

              String modifiedSentence =

                  new String(receivePacket.getData());

              //InetAddress returnIPAddress = receivePacket.getAddress();

              //int port = receivePacket.getPort();

              //System.out.println ("From server at: " + returnIPAddress +

              //                    ":" + port);

              System.out.println("Message: " + modifiedSentence);

             }

         catch (SocketTimeoutException ste)

             {

              System.out.println ("Timeout Occurred: Packet assumed lost");

              if (done)

                keepGoing = false;

             }

          }

       clientSocket.close();

      }

  catch (IOException ex)

     {

      System.err.println(ex);

     }

  }

 public static void main(String args[]) throws Exception

   {

    String serverHostname = new String ("127.0.0.1");

    if (args.length > 0)

       serverHostname = args[0];

    new UDPClient2 (serverHostname);

   }

}

       DatagramSocket clientSocket = new DatagramSocket();

       for (int i = 0; i < size ; i++)

          {

           byte[] sendData = new byte[1024];

           s1 = (String) lines.get(i);

           sendData = s1.getBytes();

           System.out.println ("Sending data to " + sendData.length +

                               " bytes to server from line " + (i + 1));

           DatagramPacket sendPacket =

              new DatagramPacket(sendData, sendData.length, IPAddress, 9876);

           clientSocket.send(sendPacket);

          }

       done = true;

       byte[] receiveData = new byte[1024];

       keepGoing = true;

       DatagramPacket receivePacket =

            new DatagramPacket(receiveData, receiveData.length);

       System.out.println ("Waiting for return packet");

       clientSocket.setSoTimeout(10000);

       while (keepGoing)

          {

           try {

              clientSocket.receive(receivePacket);

              String modifiedSentence =

                  new String(receivePacket.getData());

              //InetAddress returnIPAddress = receivePacket.getAddress();

              //int port = receivePacket.getPort();

              //System.out.println ("From server at: " + returnIPAddress +

              //                    ":" + port);

              System.out.println("Message: " + modifiedSentence);

             }

         catch (SocketTimeoutException ste)

             {

              System.out.println ("Timeout Occurred: Packet assumed lost");

              if (done)

                keepGoing = false;

             }

          }

       clientSocket.close();

      }

  catch (IOException ex)

     {

      System.err.println(ex);

     }

  }

 public static void main(String args[]) throws Exception

   {

    String serverHostname = new String ("127.0.0.1");

    if (args.length > 0)

       serverHostname = args[0];

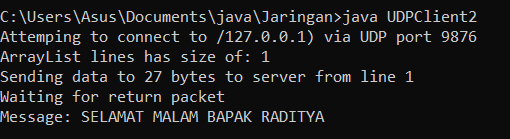
    new UDPClient2 (serverHostname);

   }

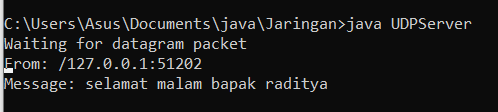
}

1. Output

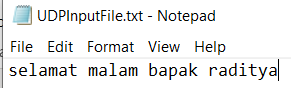
UDPClient2.java



UDPServer.java



UDPInputFile.txt



1. Analisa

Dalam percobaan ini, secara kesuluruhan sama seperti percobaan sebelumnya. Bedanya kita tidak bisa menginputkan secara langsung. Jadi disini output yang keluar berasal dari file UDPInputFile.txt yang akan ditampilkan. Jadi secara teknis, file UDPClient2.java mengirim file UDPInputFile.txt kepada UDPServer.java yang akan ditampilkan di server dan dibawa kembali pada client dan ditampilkan di client.

1. Percobaan 5
2. Listing program

UDPPortScanner.java

import java.net.\*;

import java.io.IOException;

public class UDPPortScanner {

  public static void main(String[] args) {

    // first test to see whether or not we can bind to ports

    // below 1024

    boolean rootaccess = false;

    for (int port = 1; port < 1024; port += 50) {

      try {

        ServerSocket ss = new ServerSocket(port);

        // if successful

        rootaccess = true;

        ss.close();

        break;

      } catch (IOException ex) {

      }

    }

    int startport = 1;

    if (!rootaccess)

      startport = 1024;

    int stopport = 65535;

    System.out.println("Checking UDP ports " + startport + " to " + stopport);

    for (int port = startport; port <= stopport; port++) {

      try {

        DatagramSocket ds = new DatagramSocket(port);

        ds.close();

      } catch (IOException ex) {

        System.out.println("UDP Port " + port + " is occupied.");

      }

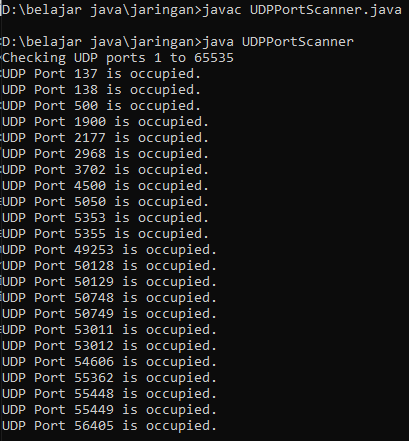
    }

  }

}

1. Output

UDPPortScanner.java



1. Analisa

Dalam percobaan tersebut berguna untuk mengecek UDP port yang telah terpakai pada percobaan sebelumnya. Dalam output tersebut menampilkan banyak port yang digunakan dengan rentang port 1 sampai 65535. Jadi dalam output tersebut menampilkan port yang telah digunakan dari port 1 sampai 65535.

1. Percobaan 6
2. Listing Program

WebServer.java

import java.io.\*;

import java.net.\*;

import java.util.\*;

class WebServer {

  public static void main(String args[]) throws Exception {

    String requestMessageLine;

    String fileName;

    // check if a port number is given as the first command line argument

    // if not argument is given, use port number 6789

    int myPort = 6789;

    if (args.length > 0) {

      try {

        myPort = Integer.parseInt(args[0]);

      } catch (ArrayIndexOutOfBoundsException e) {

        System.out.println("Need port number as argument");

        System.exit(-1);

      } catch (NumberFormatException e) {

        System.out.println("Please give port number as integer.");

        System.exit(-1);

      }

    }

    // set up connection socket

    ServerSocket listenSocket = new ServerSocket(myPort);

    // listen (i.e. wait) for connection request

    System.out.println("Web server waiting for request on port " + myPort);

    Socket connectionSocket = listenSocket.accept();

    // set up the read and write end of the communication socket

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

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

    // retrieve first line of request and set up for parsing

    requestMessageLine = inFromClient.readLine();

    System.out.println("Request: " + requestMessageLine);

    StringTokenizer tokenizedLine = new StringTokenizer(requestMessageLine);

    // check for GET request

    if (tokenizedLine.nextToken().equals("GET")) {

      fileName = tokenizedLine.nextToken();

      // remove leading slash from line if exists

      if (fileName.startsWith("/") == true)

        fileName = fileName.substring(1);

      // access the requested file

      File file = new File(fileName);

      // convert file to a byte array

      int numOfBytes = (int) file.length();

      FileInputStream inFile = new FileInputStream(fileName);

      byte[] fileInBytes = new byte[numOfBytes];

      inFile.read(fileInBytes);

      // Send reply

      outToClient.writeBytes("HTTP/1.0 200 Document Follows\r\n");

      if (fileName.endsWith(".jpg"))

        outToClient.writeBytes("Content-Type: image/jpeg\r\n");

      if (fileName.endsWith(".gif"))

        outToClient.writeBytes("Content-Type: image/gif\r\n");

      outToClient.writeBytes("Content-Length: " + numOfBytes + "\r\n");

      outToClient.writeBytes("\r\n");

      outToClient.write(fileInBytes, 0, numOfBytes);

      // read and print out the rest of the request

      requestMessageLine = inFromClient.readLine();

      while (requestMessageLine.length() >= 5) {

        System.out.println("Request: " + requestMessageLine);

        requestMessageLine = inFromClient.readLine();

      }

      System.out.println("Request: " + requestMessageLine);

      connectionSocket.close();

    } else {

      System.out.println("Bad Request Message");

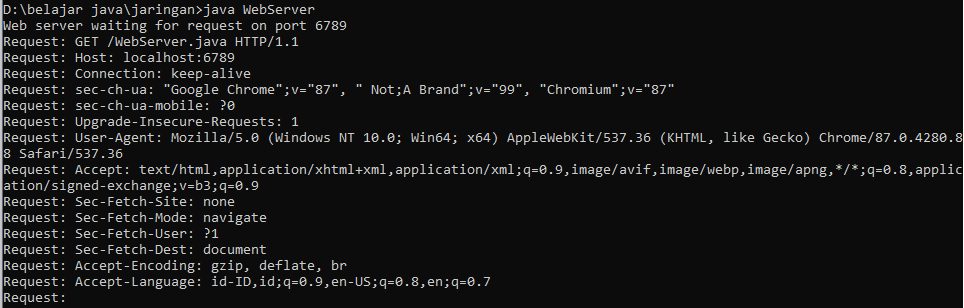
    }

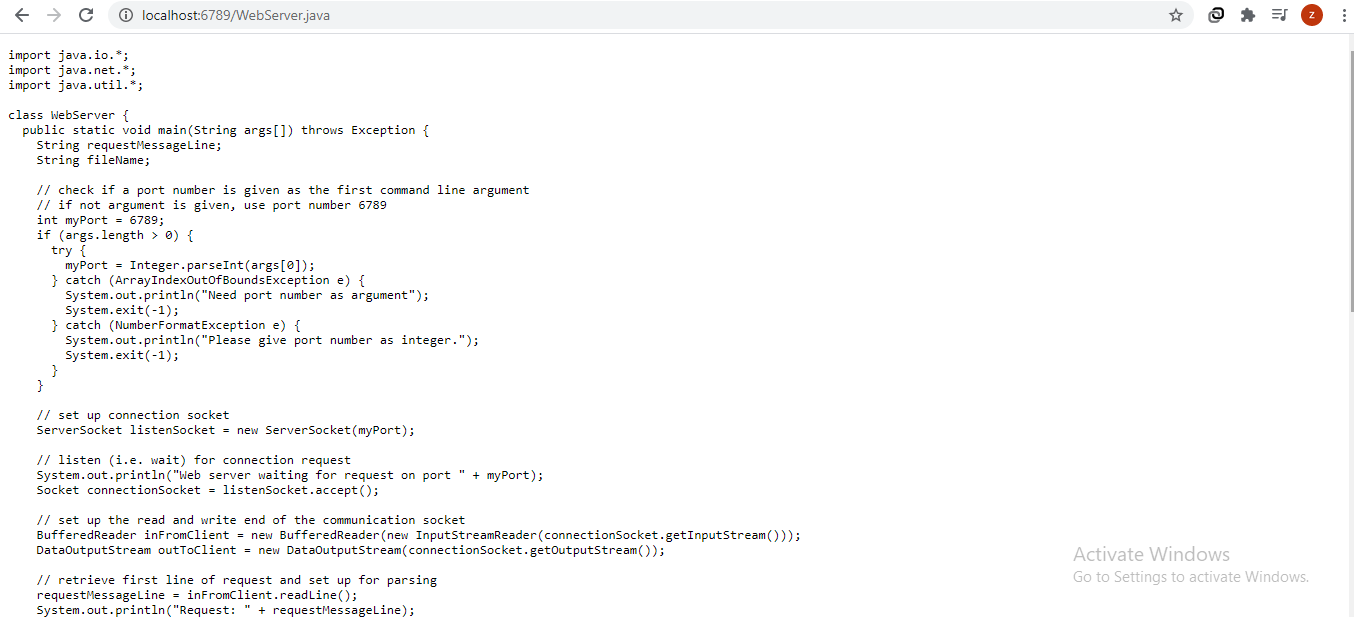
  }

}

1. Output

WebServer.java





1. Analisis

Dalam percobaan tersebut berbeda dari sebelumnya, dalam percobaan tersebut kita menggunakan server melalui browser sehingga client mengakses server dengan cara melalui browser. Setelah kita menjalankan Server maka kita lihat tampilannya masih kosong hanya ada port 6789. Cara mengakses server tersebut kita buka chrome dan mengakses alamat localhost:6789/WebServer.java. Setelah itu, kita bisa melihat tampilan browser akan tampil source code yang ada dalam WebServer.java. dan setelah kita lihat kembali di cmd server maka akan muncul tampilan penjelasan alur pengiriman fileWebServer.java