

Pemograman Jaringan

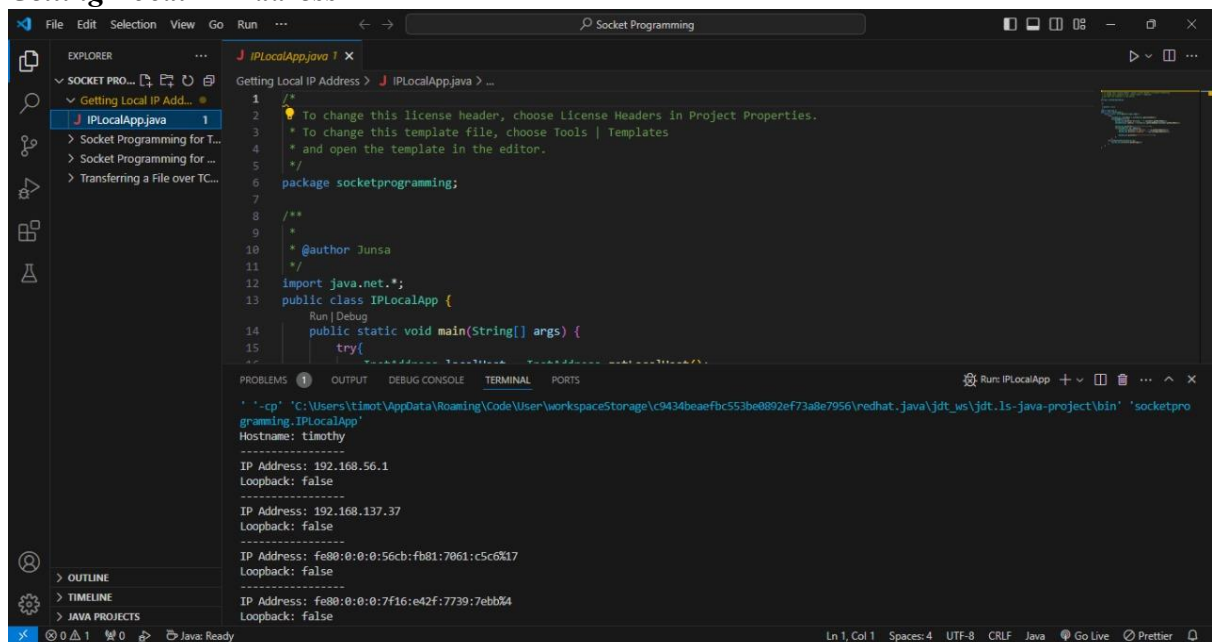
Tugas Pengujian Socket Programming

Kelompok 3

1. Tania Resubun (21013044) Ketua Kelompok
2. Monica Mary Poluakan (21013001)
3. Julinda Mikha Rondonuwu (21013043)
4. Timothy Daniel Rooroh (21013028)

Hasil output dan penjelasannya:

1. *Getting Local IP Address*



The screenshot shows a code editor with a Java file named `IPLocalApp.java`. The code is as follows:

```
1  /*  
2  * To change this license header, choose License Headers in Project Properties.  
3  * To change this template file, choose Tools | Templates  
4  * and open the template in the editor.  
5  */  
6  package socketprogramming;  
7  
8  /**  
9  *  
10 * @author Junsu  
11 */  
12 import java.net.*;  
13 public class IPLocalApp {  
14     Run | Debug  
15     public static void main(String[] args) {  
16         try {  
17             // ...  
18         }  
19     }  
20 }
```

The terminal output at the bottom shows the following results:

```
Run: IPLocalApp  
-cp 'C:\Users\timot\AppData\Roaming\Code\User\workspaceStorage\c9434beaefbc553be0892ef73a8e7956\redhat.java\jdt_ws\jdt.ls-java-project\bin' 'socketpro  
gramming.IPLocalApp'  
Hostname: timothy  
-----  
IP Address: 192.168.56.1  
Loopback: false  
-----  
IP Address: 192.168.137.37  
Loopback: false  
-----  
IP Address: fe80:0:0:0:56cb:fb81:7061:c5c6%17  
Loopback: false  
-----  
IP Address: fe80:0:0:0:7f16:e42f:7739:7ebb%4  
Loopback: false
```

Output diatas merupakan :

Hostname: timothy: Ini menunjukkan nama *host* dari perangkat tempat program dijalankan, yaitu "timothy".

- *IP Address: 192.168.56.1, Loopback: false*: Ini adalah alamat *IP* dari perangkat yang memiliki koneksi jaringan aktif. Alamat *IP* ini adalah 192.168.56.1, dan *Loopback* dinyatakan sebagai "*false*", yang berarti ini bukanlah alamat *loopback* (*localhost*).

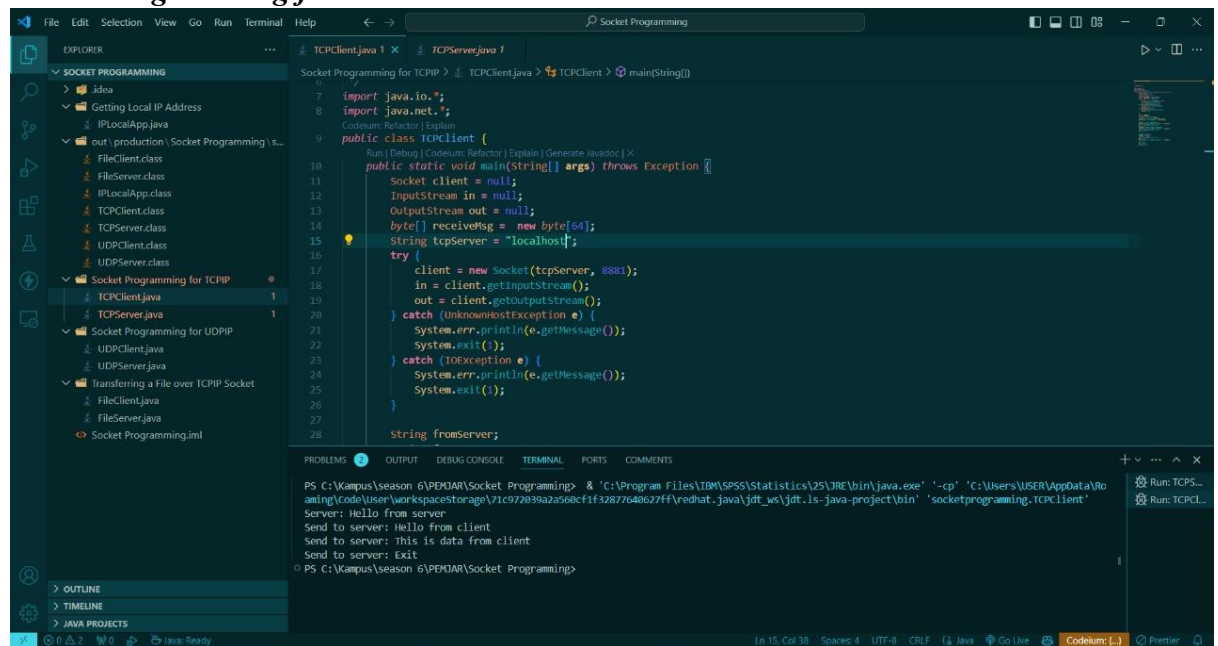
- *IP Address: 192.168.137.37, Loopback: false*: Ini adalah alamat *IP* lain dari perangkat yang mungkin terhubung ke jaringan. Alamat *IP* ini adalah 192.168.137.37, dan seperti sebelumnya, *Loopback* dinyatakan sebagai "*false*".

- *IP Address: fe80:0:0:0:56cb:fb81:7061:c5c6%17, Loopback: false*: Ini adalah alamat *IP* dalam format *IPv6*. Alamat ini adalah fe80:0:0:0:56cb:fb81:7061:c5c6 dengan *Loopback* yang dinyatakan sebagai "*false*".

- *IP Address*: fe80:0:0:0:7f16:e42f:7739:7ebb%4, *Loopback*: false: Ini juga merupakan alamat IPv6. Alamat ini adalah fe80:0:0:0:7f16:e42f:7739:7ebb dengan *Loopback* yang dinyatakan sebagai "false".

Dari output ini, terlihat bahwa perangkat "timothy" memiliki beberapa alamat *IP* yang terhubung ke jaringan, baik dalam format *IPv4* maupun *IPv6*. Dan mereka semua bukan alamat *loopback*.

2. Socket Programming for TCP/IP



- Socket Programming for TCP/IP

File: TCPClient.java

Kode ini menjelaskan implementasi sederhana dari klien TCP di Java. Program dimulai dengan mengimpor pustaka yang diperlukan dan mendefinisikan kelas TCPClient dengan metode utama. Metode utama menginisialisasi variabel dan mencoba membuat koneksi ke server. Setelah koneksi dibuat, klien membaca pesan dari server, mencetaknya, dan mengirim dua pesan ke server. Klien kemudian mengambil jeda sejenak, mengirimkan pesan "Exit" terakhir, dan menutup koneksi.

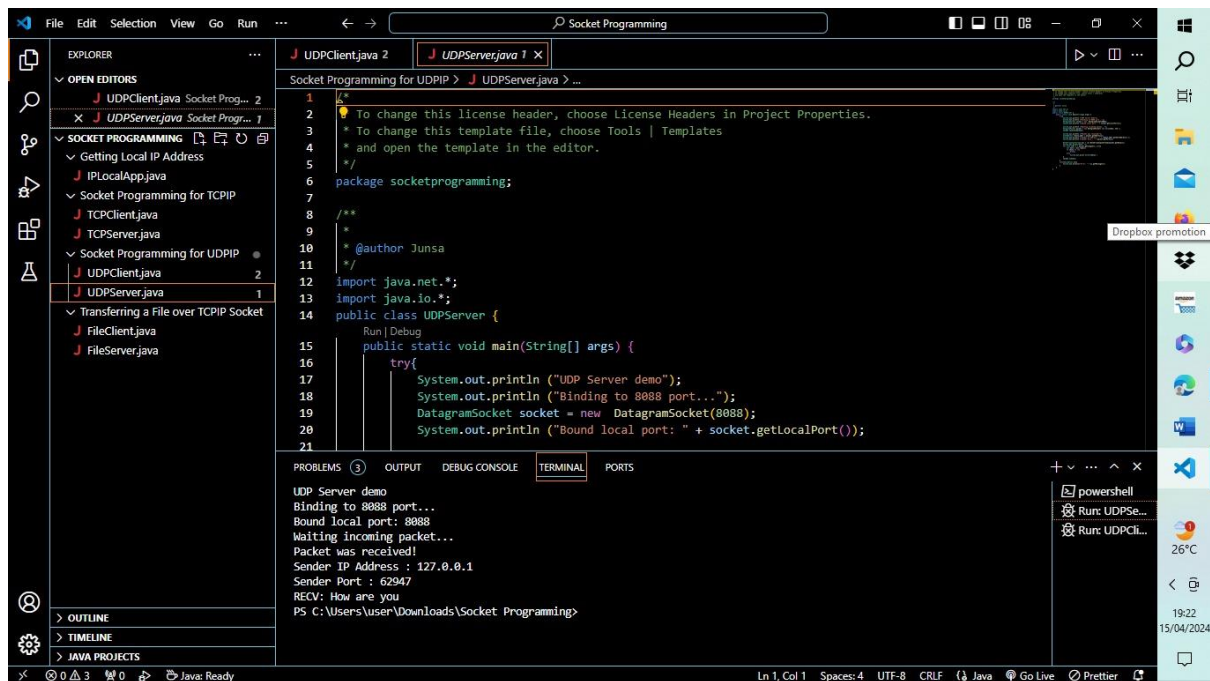
File: TCPServer.java

Program Java ini menyediakan implementasi sederhana dari server TCP. Program ini mengimpor pustaka yang diperlukan untuk operasi jaringan dan I/O. Kelas TCPServer didefinisikan dengan metode utama sebagai titik masuk program. Metode utama menginisialisasi variabel dan mencoba membuat soket server pada port 8881. Jika berhasil, maka akan menunggu klien untuk melakukan koneksi. Server kemudian mengirimkan pesan "Halo dari server" ke klien menggunakan "Input Stream". Server memasuki sebuah loop untuk membaca pesan dari klien menggunakan "Input Stream". Jika pesan diterima, pesan tersebut diubah menjadi string dan dicetak. Jika pesan berisi string "Exit", server

mengirimkan pesan "Exit" kembali ke klien dan keluar dari perulangan. Terakhir, server "Output Stream" dan socket.

3. Socket Programming for UDP/IP

a. UDP Server



Gambar 1. Dokumentasi running file UDP Server

Penjelasan:

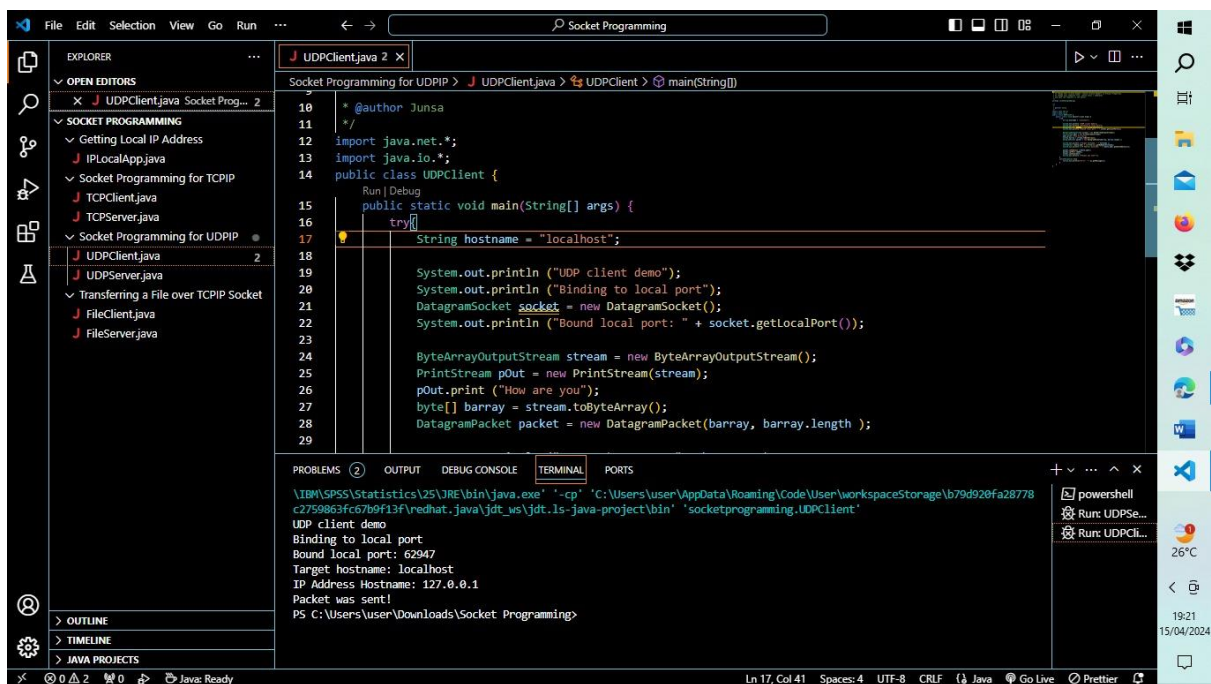
Output yang dihasilkan pada file *UDP Server* ini memberikan informasi tentang status dan interaksi antara server dan klien dalam *socket programming*. Ini menunjukkan langkah-langkah yang dilakukan oleh *server* untuk menerima dan memproses data yang dikirim oleh klien melalui koneksi *UDP*.

- 1) *UDP Server demo*: adalah pesan pertama yang ditampilkan oleh *server* saat program dimulai. Ini hanya merupakan pesan informasi bahwa program *server UDP* telah dimulai.
- 2) *Binding to 8088 port...*: Pesan ini menunjukkan bahwa *server* sedang mencoba untuk mengikat (*bind*) pada *port* 8088. Proses mengikat inilah yang memungkinkan *server* untuk mendengarkan koneksi yang masuk pada *port* tertentu.
- 3) *Bound local port: 8088*: Ini adalah konfirmasi bahwa *server* telah berhasil mengikat (*bind*) pada *port* 8088. Ini menunjukkan bahwa *server* sekarang siap untuk menerima koneksi pada *port* tersebut.

- 4) *Waiting incoming packet...*: Setelah *server* berhasil mengikat port, pesan ini menandakan bahwa *server* sekarang sedang menunggu kedatangan paket dari klien. Ini menunjukkan bahwa *server* telah siap untuk menerima data dari klien.
- 5) *Packet was received!*: Pesan ini menandakan bahwa *server* telah menerima sebuah paket dari klien. Ini menunjukkan bahwa koneksi antara klien dan *server* berhasil, dan data telah berhasil dikirim dari klien ke *server*.
- 6) *Sender IP Address: 127.0.0.1*: Ini adalah alamat IP pengirim paket. Dalam hal ini, "127.0.0.1" adalah alamat *loopback* yang menunjukkan bahwa klien dan *server* berjalan di komputer yang sama.
- 7) *Sender Port: 62947*: Ini adalah *port* pengirim paket. Ini menunjukkan *port* yang digunakan oleh klien untuk mengirim paket kepada *server*.
- 8) *RECV: How are you* : Ini adalah pesan yang dikirim oleh klien dan berhasil diterima oleh *server*. Ini menunjukkan bahwa data yang dikirim oleh klien ("How are you.") berhasil diterima dan diproses oleh *server*.

Berdasarkan *output* yang ditampilkan *code UDP Server* tersebut berhasil berjalan.

b. UDP Client



```
10  * @author Junsu
11  */
12  import java.net.*;
13  import java.io.*;
14  public class UDPClient {
15      Run | Debug
16      public static void main(String[] args) {
17          try {
18              String hostname = "localhost";
19
20              System.out.println ("UDP client demo");
21              System.out.println ("Binding to local port");
22              DatagramSocket socket = new DatagramSocket();
23              System.out.println ("Bound local port: " + socket.getLocalPort());
24
25              ByteArrayOutputStream stream = new ByteArrayOutputStream();
26              PrintStream pOut = new PrintStream(stream);
27              pOut.print ("How are you");
28              byte[] barray = stream.toByteArray();
29              DatagramPacket packet = new DatagramPacket(barray, barray.length );
```

PROBLEMS (2) OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
\IBM\SPSS\Statistics\25\JRE\bin\java.exe -cp "C:\Users\User\AppData\Roaming\Code\User\workspaceStorage\b79d920fa28778
c2759863fc67b9f13f\redhat.java\jdt_ws\jdt.ls-java-project\bin" 'socketprogramming.UDPClient'
UDP client demo
Binding to local port
Bound local port: 62947
Target hostname: localhost
IP Address Hostname: 127.0.0.1
Packet was sent!
PS C:\Users\User\Downloads\Socket Programming>
```

Gambar 2. Dokumentasi *running file UDP Client*

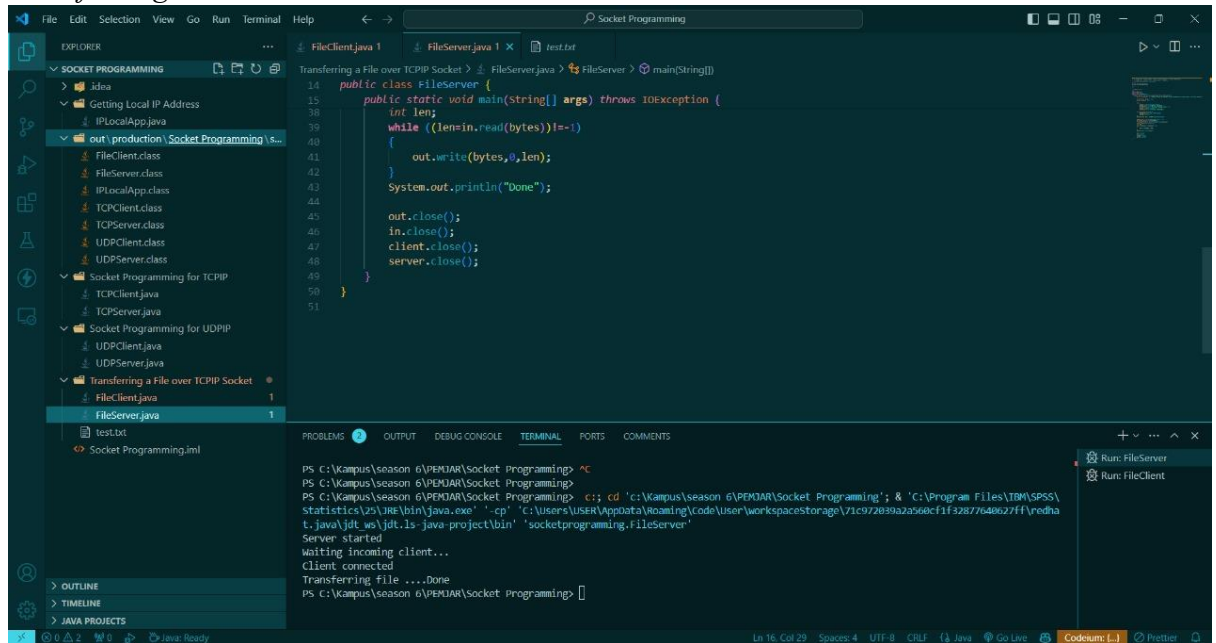
Penjelasan:

Output tersebut menunjukkan bahwa klien berhasil mengirimkan paket data ke *server* melalui protokol *UDP*. Kesimpulannya, dengan berhasilnya pengiriman paket, implementasi *socket* programming pada klien telah berhasil dalam mengirim data ke *server*.

- 1) *UDP client demo*: Ini adalah pesan pertama yang ditampilkan oleh klien saat program dimulai. Ini hanya merupakan pesan informasi bahwa program klien *UDP* telah dimulai.
- 2) *Binding to local port*: Pesan ini menunjukkan bahwa klien sedang melakukan proses binding (pengikatan) pada *port* lokal. Dalam hal ini, klien tidak secara eksplisit menentukan *port* tertentu, sehingga sistem secara otomatis memilih *port* yang tersedia.
- 3) *Bound local port: 62947*: Ini adalah konfirmasi bahwa klien telah berhasil mengikat (*bind*) ke *port* lokal. Nomor *port* yang ditampilkan adalah nomor *port* yang digunakan oleh klien untuk komunikasi. *Port* ini akan digunakan oleh sistem operasi untuk mengarahkan paket yang keluar dari klien.
- 4) *Target hostname: localhost*: Ini menunjukkan bahwa klien akan mengirim paket ke *host* 'localhost'. Hostname ini merupakan alamat IP lokal dari komputer tempat program dijalankan.
- 5) *IP Address Hostname: 127.0.0.1*: Ini adalah alamat IP yang sesuai dengan hostname 'localhost'. Dalam kasus ini, 'localhost' mengarah ke alamat IP *loopback* standar (127.0.0.1), yang menunjukkan bahwa klien dan *server* berjalan di komputer yang sama.
- 6) *Packet was sent!*: Pesan ini menunjukkan bahwa paket data telah berhasil dikirim oleh klien ke *server* melalui jaringan menggunakan protokol UDP. Ini menegaskan bahwa koneksi antara klien dan *server* berhasil, dan data berhasil dikirim dari klien ke *server*.

Dengan demikian, *output* tersebut memberikan informasi tentang langkah-langkah yang dilakukan oleh klien dalam proses pengiriman paket data menggunakan *socket programming*. Ini menunjukkan proses komunikasi yang terjadi antara klien dan *server* dalam lingkungan jaringan.

4. Transferring a File over TCP/IP Socket



The screenshot shows an IDE with the following components:

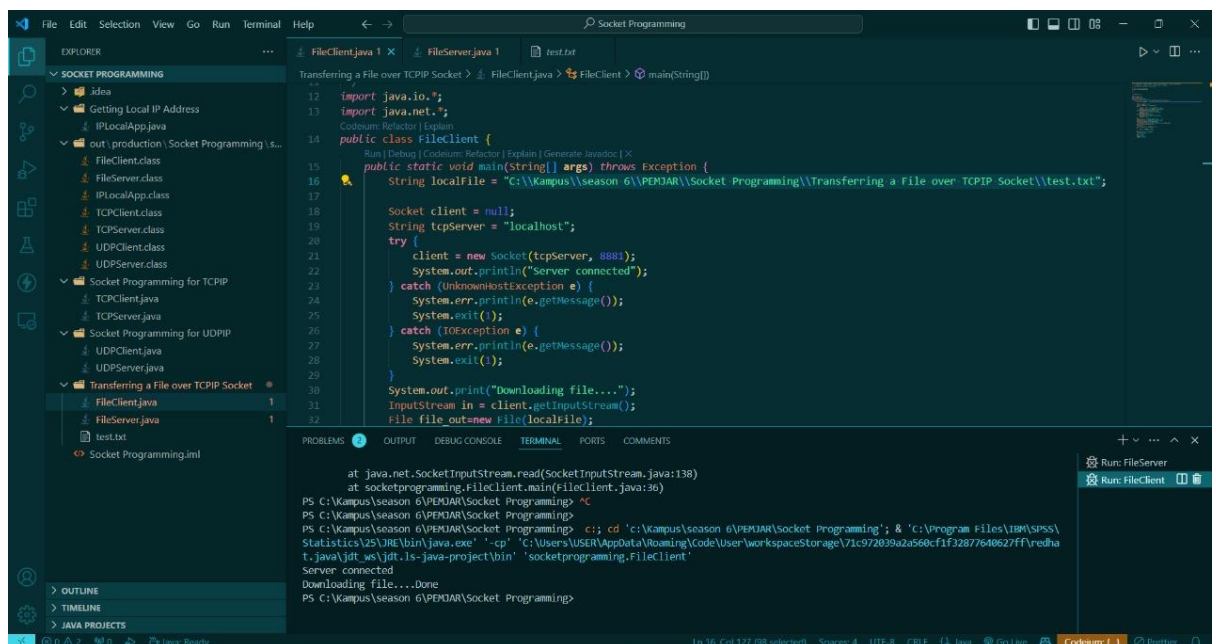
- EXPLORER:** A project named "Socket Programming" with a sub-project "Transferring a File over TCP/IP Socket". It contains files like FileClient.class, FileServer.class, and test.txt.
- Code Editor:** Displays the `FileServer.java` file. The code is as follows:

```
14 public class FileServer {
15     public static void main(String[] args) throws IOException {
16         int len;
17         while ((len=in.read(bytes))!=1)
18         {
19             out.write(bytes,0,len);
20             System.out.println("Done");
21         }
22         out.close();
23         in.close();
24         client.close();
25         server.close();
26     }
27 }
```
- TERMINAL:** Shows the execution of the `FileServer` class. The output is:

```
PS C:\Kampus\season 6\PEMJAR\Socket Programming> ^C
PS C:\Kampus\season 6\PEMJAR\Socket Programming>
PS C:\Kampus\season 6\PEMJAR\Socket Programming> cd 'c:\Kampus\season 6\PEMJAR\Socket Programming'; & 'c:\Program Files\IBM\SPSS\
Statistics\25\JRE\bin\java.exe' -cp 'c:\Users\USER\AppData\Roaming\Code\User\workspaceStorage\71c972039a2a560cf1f32877648627ff\redha
t_java\jdt_ws\jdt.ls-java-project\bin' 'socketprogramming.FileServer'
Server started
Waiting incoming client...
Client connected
Transferring file ....Done
PS C:\Kampus\season 6\PEMJAR\Socket Programming> ^C
```

a. FileServer.java

Kode di atas adalah contoh sederhana dari sebuah server file dalam bahasa pemrograman Java yang menggunakan socket untuk mentransfer file dari server ke klien.



The screenshot shows an IDE with the following components:

- EXPLORER:** The same project structure as the previous screenshot, but with `FileClient.java` selected.
- Code Editor:** Displays the `FileClient.java` file. The code is as follows:

```
12 import java.io.*;
13 import java.net.*;
14 public class FileClient {
15     public static void main(String[] args) throws Exception {
16         String localFile = "c:\\Kampus\\season 6\\PEMJAR\\Socket Programming\\Transferring a File over TCP/IP Socket\\test.txt";
17
18         Socket client = null;
19         String tcpServer = "localhost";
20         try {
21             client = new Socket(tcpServer, 8881);
22             System.out.println("Server connected");
23         } catch (UnknownHostException e) {
24             System.err.println(e.getMessage());
25             System.exit(1);
26         } catch (IOException e) {
27             System.err.println(e.getMessage());
28             System.exit(1);
29         }
30         System.out.print("Downloading file....");
31         InputStream in = client.getInputStream();
32         File file = new File(localFile);
```
- TERMINAL:** Shows the execution of the `FileClient` class. The output is:

```
at java.net.SocketInputStream.read(SocketInputStream.java:138)
at socketprogramming.FileClient.main(FileClient.java:36)
PS C:\Kampus\season 6\PEMJAR\Socket Programming> ^C
PS C:\Kampus\season 6\PEMJAR\Socket Programming>
PS C:\Kampus\season 6\PEMJAR\Socket Programming> cd 'c:\Kampus\season 6\PEMJAR\Socket Programming'; & 'c:\Program Files\IBM\SPSS\
Statistics\25\JRE\bin\java.exe' -cp 'c:\Users\USER\AppData\Roaming\Code\User\workspaceStorage\71c972039a2a560cf1f32877648627ff\redha
t_java\jdt_ws\jdt.ls-java-project\bin' 'socketprogramming.FileClient'
Server connected
Downloading file....Done
PS C:\Kampus\season 6\PEMJAR\Socket Programming>
```

b. FileClient.java

Kode di atas adalah contoh implementasi klien file dalam bahasa pemrograman Java yang menggunakan socket untuk mengunduh file dari server dan menyimpannya secara lokal.

