

Mobile Computing

# Mobile Networking

Ramos Somya

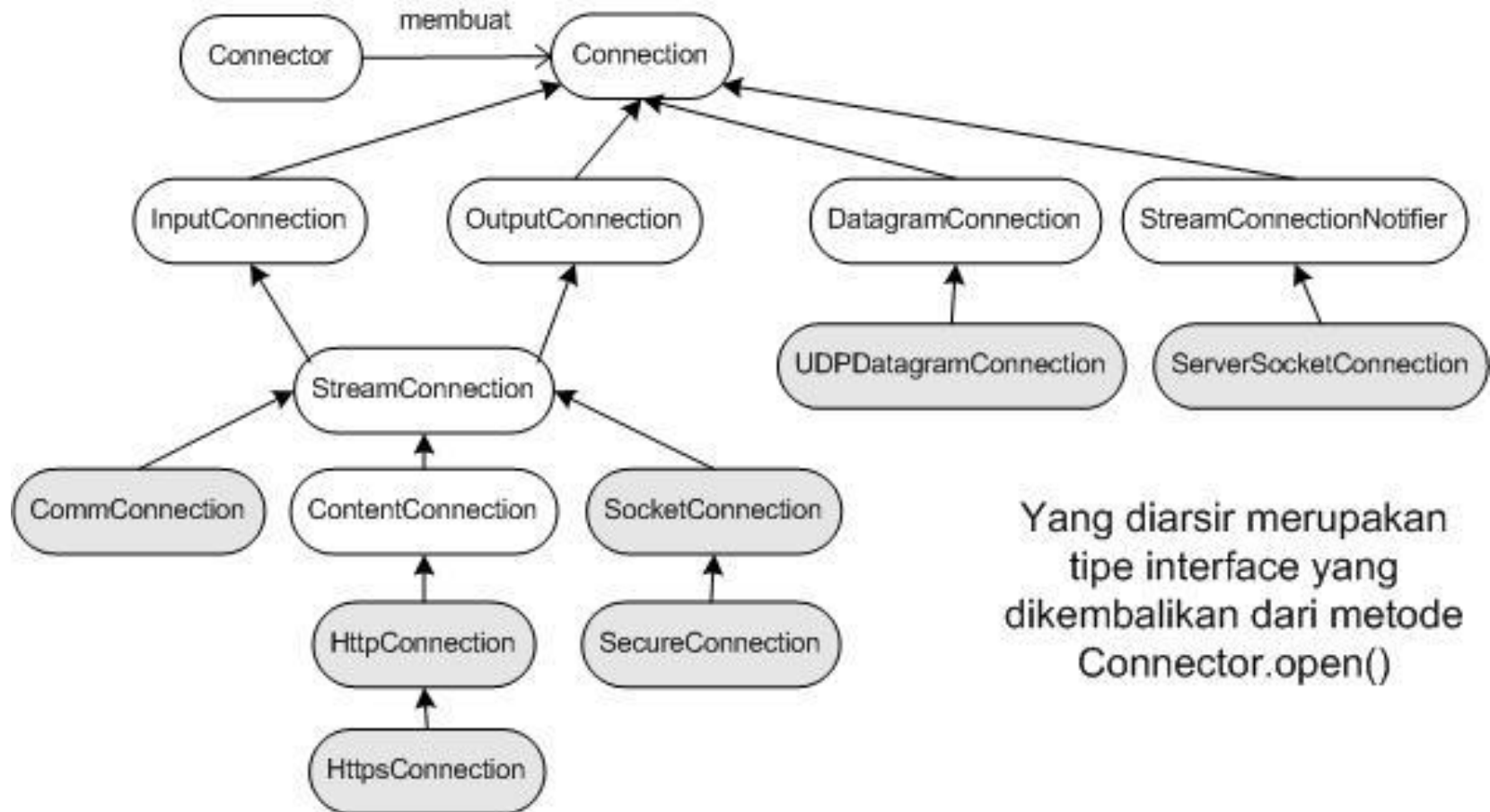
## Generic Connection Framework (GCF)

- CLDC mempunyai kelas-kelas yang diturunkan dari J2SE dan kelas-kelas yang spesifik pada CLDC, yaitu GCF.
- MIDP menggunakan Generic Connection Framework (GCF) dari CLDC untuk mendukung networking dan I/O.
- Package : `javax.microedition.io`

# package

- **javax.microedition.io**
- **Kelas : Connector, PushRegistry**
- **Interface:**
  - CommConnection**
  - HttpConnection**
  - HttpsConnection**
  - SecurityConnection**
  - SecurityInfo**
  - ServerSocketConnection**
  - SocketConnection**
  - UDPDatagramConnection**

# Hirarki Interface Connection



# Membuka koneksi

- Menggunakan metode `open()` dari kelas `Connector`.
- Definisinya :
  - `Connection open(String url)`
  - `Connection open(String url, int mode)`
  - `Connection open(String url, int mode, boolean timeout)`

# Format url

- Format URL : `{scheme}:[{target}][{params}]`
- `{scheme}` merupakan tipe connection protocol
- `{target}` merupakan network address atau host name
- `{params}` merupakan daftar dari parameter koneksi, bentuknya : `;key=value`

# url yang disupport

- `http://host:port` – Membuat koneksi HTTP ke remote server
- `https://host:port` – Membuat koneksi HTTPS ke remote server
- `ssl://host:port` – Membuat secure socket connection ke remote server
- `socket://:port` – “Mendengar” koneksi socket pada local port
- `comm://IR#` - Membuat koneksi ke IR (Infra red)
- `comm://COM#` - Membuat koneksi serial port
- `datagram://host:port` – Mengirim/menerima UDP datagram

## Metode open()

- Saat menggunakan metode open(), harus dilakukan casting ke tipe interface yang sesuai.
- Hubungan antara protokol dan tipe interface:
  - URLConnection – http://host:port
  - URLConnection – https://host:port
  - URLConnection – socket://host:port
  - URLConnection – ssl://host:port
  - UDPDatagramConnection – datagram://host:port
  - ServerSocketConnection – socket://:port
  - CommConnection – comm://IR# atau comm://COM#
    - Tanda # menunjukkan nomor com atau irda port

## contoh

- `HttpConnection hc = (HttpConnection) Connector.open("http://localhost");`
- `HttpsConnection hsc = (HttpConnection) Connector.open(https://localhost);`

# Kelas Connector

- Metode-metode penting :
- `Connection open(String url)`
- `Connection open(String url, int mode)`
- `Connection open(String url, int mode, boolean timeout)`
- `DataInputStream openDataInputStream(String name)`
- `DataOutputStream openDataOutputStream(String name)`
- `InputStream openInputStream(String name)`
- `OutputStream openOutputStream(String name)`

# Menutup koneksi

---


- Menggunakan metode `close()`

# Contoh Program // Class ClientSocket

```
import javax.microedition.io.*;
import java.io.*;
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;

public class ClientSocket extends MIDlet implements CommandListener {


    Display display;
    Form formClient;
    Command exitCommand = new Command("Exit", Command.EXIT, 0);
    SocketConnection connection = null;
    OutputStream oStream = null;
```



```
public ClientSocket() {
    formClient = new Form("Client Socket");
    formClient.addCommand(exitCommand);
    formClient.setCommandListener(this);
    try {
        connection = (SocketConnection)
            Connector.open("socket://localhost:3887");
        connection.setSocketOption(connection.DELAY, 0);
        oStream = connection.openOutputStream();
        oStream.write("This is message form client\n".getBytes());
        oStream.close();
        connection.close();
    } catch (Exception e) {
        formClient.append(e.toString());
    }
}
```



```
public void startApp() {  
    if (display == null) {  
        display = Display.getDisplay(this);  
        display.setCurrent(formClient);  
    }  
}  
  
public void pauseApp() {  
}  
  
public void destroyApp(boolean d) {  
}
```




```
public void commandAction(Command c, Displayable d) {
    if (c == exitCommand) {
        try {
            if (connection != null) {
                connection.close();
            }
            if (oStream != null) {
                oStream.close();
            }
        } catch (Exception e) {
        }
        destroyApp(true);
        notifyDestroyed(); // Exit
    }
}
}
```

# Class ServerSocket

```
import javax.microedition.io.*;
import java.io.*;
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;

public class ServerSocket extends MIDlet implements Runnable,
CommandListener {


    Display display;
    Form formServer;
    Command exitCommand = new Command("Exit", Command.EXIT, 0);
    ServerSocketConnection connection = null;
    SocketConnection conn = null;
    InputStream iStream = null;
```



```
public ServerSocket() {  
    formServer = new Form("Server Socket");  
    formServer.addCommand(exitCommand);  
    formServer.setCommandListener(this);  
}
```

```
public void startApp() {  
    if (display == null) {  
        display = Display.getDisplay(this);  
        display.setCurrent(formServer);  
    }  
    Thread t = new Thread(this);  
    t.start();  
}
```

```
public void pauseApp() {  
}
```



```
public void destroyApp(boolean d) {  
    }  
  
    public void commandAction(Command c, Displayable d) {  
        if (c == exitCommand) {  
            try {  
                if (connection != null) {  
                    connection.close();  
                }  
                if (conn != null) {  
                    conn.close();  
                }  
                if (iStream != null) {  
                    iStream.close();  
                }  
            } catch (Exception e) {  
            }  
            destroyApp(true);  
            notifyDestroyed(); // Exit  
        }  
    }  
}
```



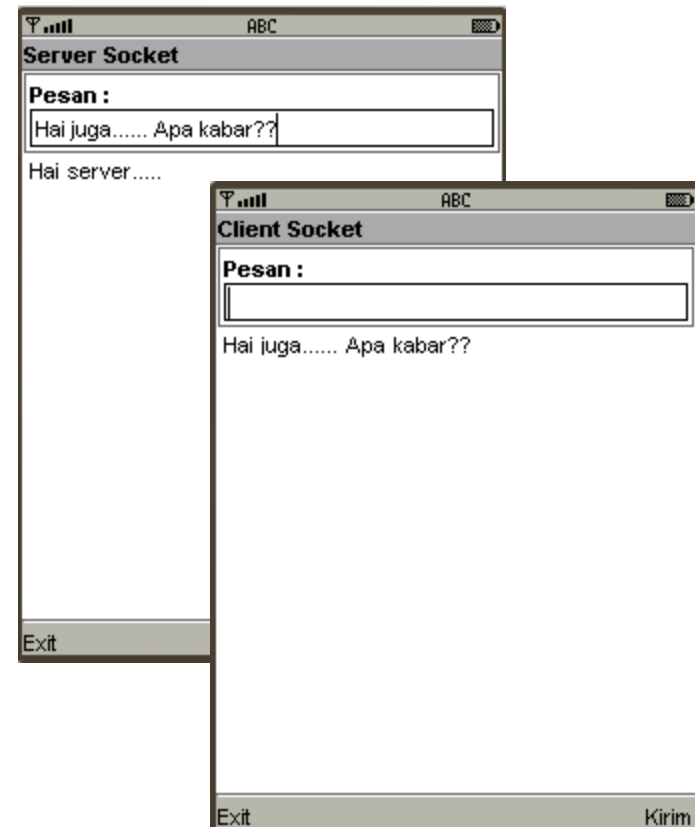
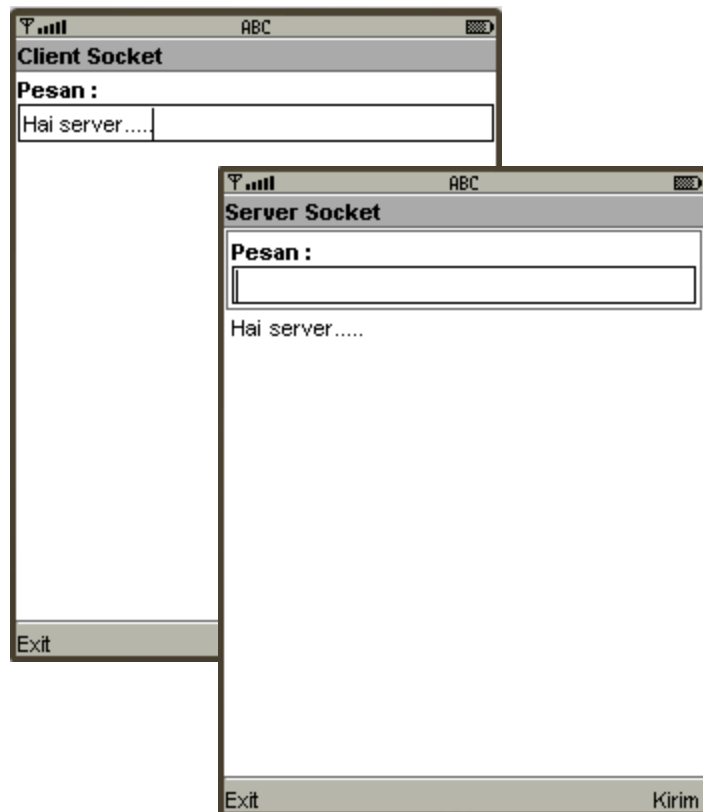
```
public void start() {  
    }  
  
    public void run() {  
        try {  
            connection = (ServerSocketConnection) Connector.open("socket://:3887");  
            conn = (SocketConnection) connection.acceptAndOpen();  
            conn.setSocketOption(conn.DELAY, 0);  
            iStream = conn.openInputStream();  
            int c = 0;  
            String data = "";  
            while ((c = iStream.read()) != -1) {  
                data += (char) c;  
            }  
            formServer.append(data);  
        } catch (Exception e) {  
            formServer.append(e.toString());  
        }  
    }  
  
    public void stop() {  
    }  
}
```

# hasil



# Tugas Kelas

- Tambahkan textfield pada server dan client supaya dapat saling mengirimkan pesan.



---

**See You Next Week**