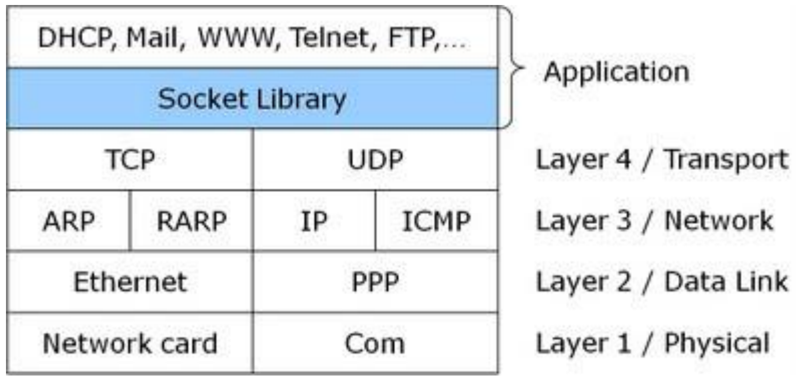
Android Socket

What is Socket?

- **Socket**：兩個互相溝通的程序(process)之間的任一端點。此兩個process可同屬一電腦系統之內，或分屬於兩個不同的電腦系統透過網路來溝通。
- This lab will try how to use “Socket "contact with another process.
- You can learn more in “Introduction to Computer Networks”
- http://developer.android.com/reference/java/net/Socket.html (API Url)

Socket in anywhere



- In C language:
  - So difficult to use
  - Ex:

    -
      ```
      #include<winsock2.h>
      SOCKET socksrv =
       socket(AF_INET,SOCK_STREAM,0);
      SOCKADDR_IN socketadd;
      socketadd.sin_addr.S_un.S_addr = htonl(INADDR_ANY);
      socketadd.sin_family = AF_INET;
      socketadd.sin_port = htons(7001)
      ```

    會涉及很多字串轉換、型別設定的問題
- In Android ≒ In Java
  - You can use java.net.ServerSocket this object

Socket in Android

- Socket like a door in your process, but you have many key to open this door.
  - Key -> Port, Room name->IP
  - Http use Port 80
  - Ftp use Port 20 ~ 21
  - BBS port (telnet) 23
  - TCP/IP 的 Port Range 只有從 0 到 65535 (Why?) 2^16-1
- So if you want contact another device, tell your socket IP and port.

Socket in Android(cont.) (TCP/IP)

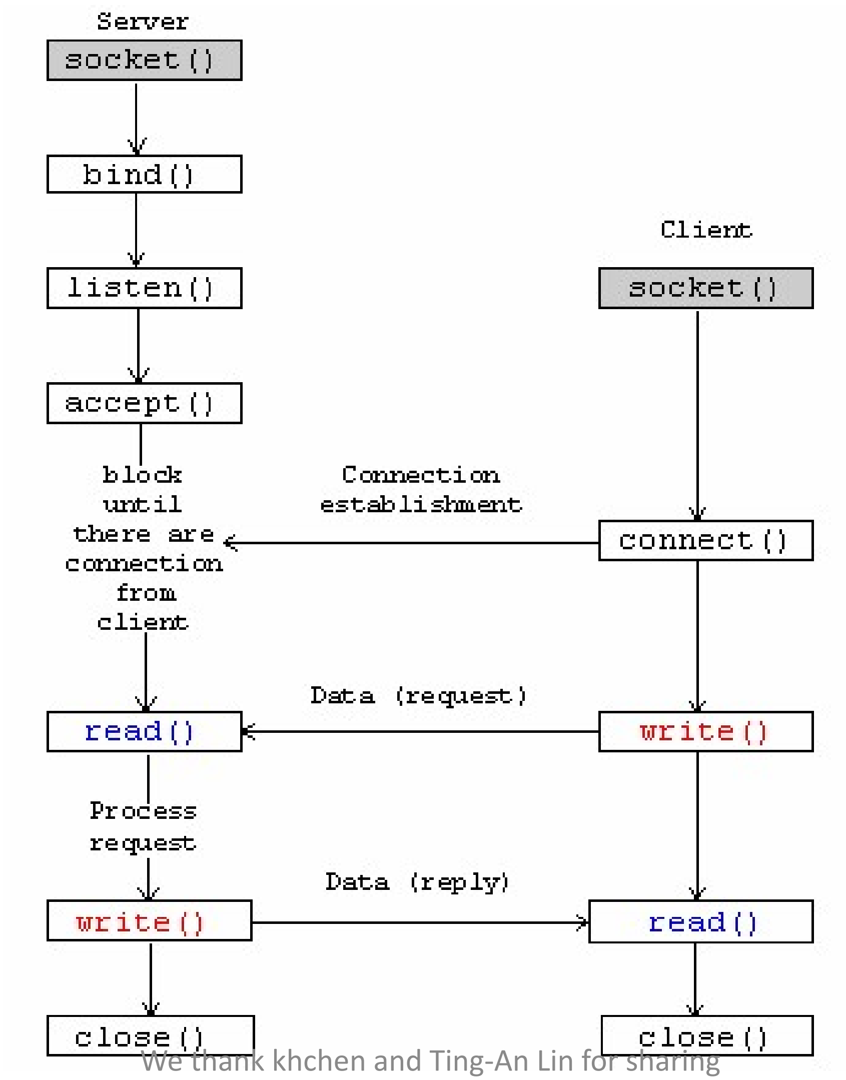
Socket 通信流程server端

- Socket construct (Your door)
- Bind() -> port (在門上裝鎖)
- Listen()
  (鑰匙插進來 要有所反應的人)
  - Constructor 幫你做好了
- Accept()
  <- wait client connect()
- Recv() & Send()
- Closesocket()
  - *serverSocket.close();*

Client端 (客戶端)

- Socket Construct
- Connect()
  - `Socket clientSocket=new Socket(serverIp,serverPort);`
- Send() & Recv()
- Closesocket()
  - clientSocket.close();

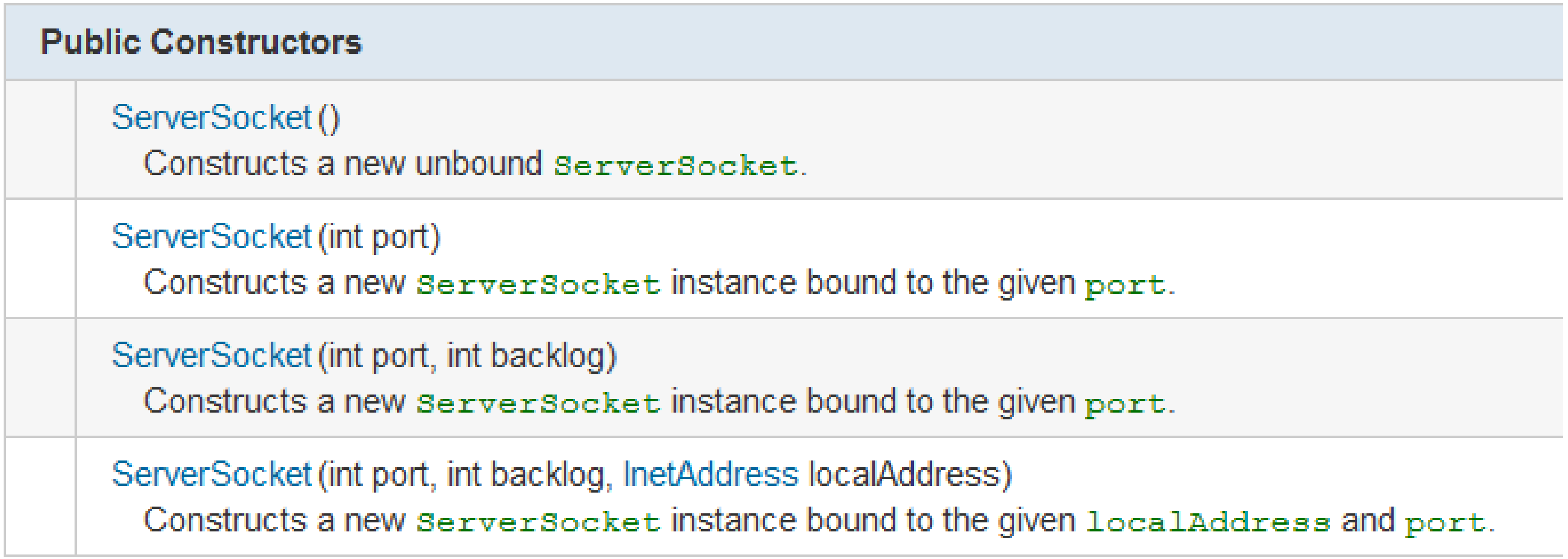
Socket in Android(cont.) (TCP/IP)



ServerSocket

- Object in java.net
- A server-side socket that waits for incoming client connections

ServerSocket()



Public Constructors
ServerSocket ()   Constructs a new unbound `ServerSocket`.
ServerSocket (int port)   Constructs a new `ServerSocket` instance bound to the given `port`.
ServerSocket (int port, int backlog)   Constructs a new `ServerSocket` instance bound to the given `port`.
ServerSocket (int port, int backlog, InetAddress localAddress)   Constructs a new `ServerSocket` instance bound to the given `localAddress` and `port`.

Serversocket(int port)

- *public ServerSocket (int port)*
- Create a new server socket that listen for client connect on given port.
- Parameters
  - *port* : the port that the server socket listen on

accept()

- *pubilc Socket accept ()*
- Waits for an incoming request and blocks until the connection is opened.
- This method returns a socket object representing the just opened connection

isClose()

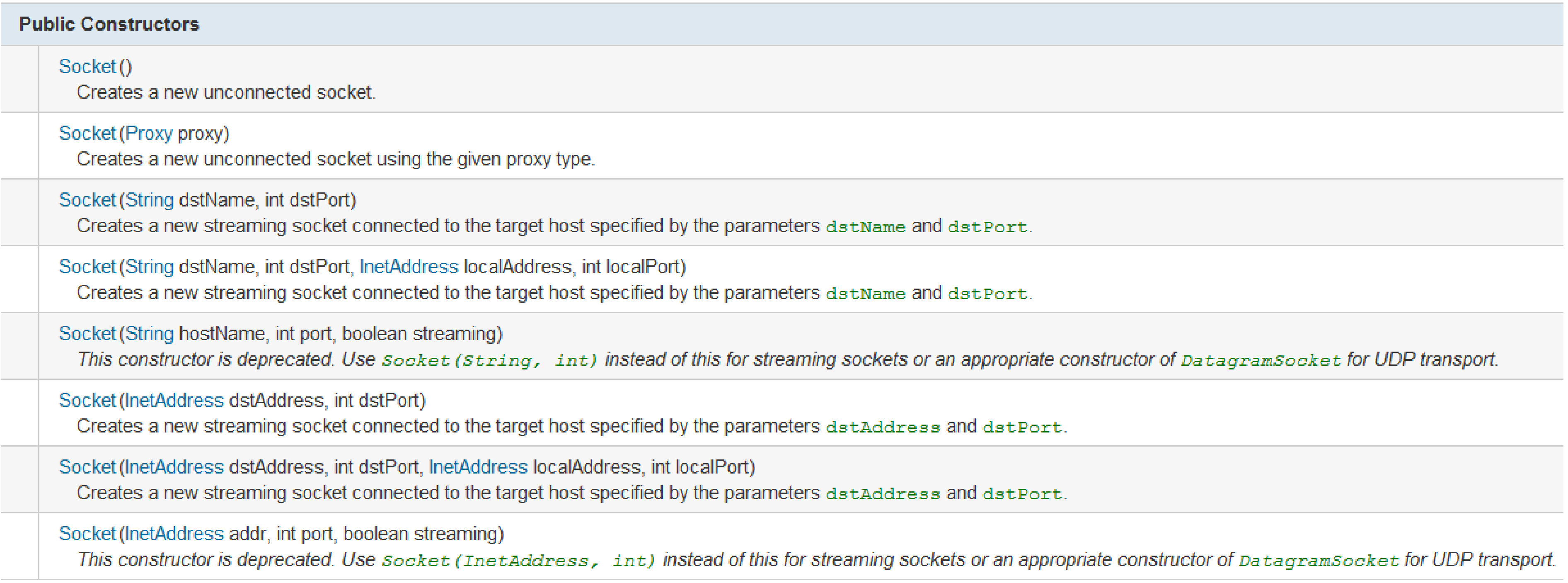
- *public boolean isClosed ()*

- Returns whether this server socket is closed or not.

Socket

- Object in java.net
- A client-side TCP socket

Socket()



Public Constructors
Socket()   Creates a new unconnected socket.
Socket(Proxy proxy)   Creates a new unconnected socket using the given proxy type.
Socket(String dstName, int dstPort)   Creates a new streaming socket connected to the target host specified by the parameters `dstName` and `dstPort`.
Socket(String dstName, int dstPort, InetAddress localAddress, int localPort)   Creates a new streaming socket connected to the target host specified by the parameters `dstName` and `dstPort`.
Socket(String hostName, int port, boolean streaming)   *This constructor is deprecated. Use `Socket(String, int)` instead of this for streaming sockets or an appropriate constructor of `DatagramSocket` for UDP transport.*
Socket(InetAddress dstAddress, int dstPort)   Creates a new streaming socket connected to the target host specified by the parameters `dstAddress` and `dstPort`.
Socket(InetAddress dstAddress, int dstPort, InetAddress localAddress, int localPort)   Creates a new streaming socket connected to the target host specified by the parameters `dstAddress` and `dstPort`.
Socket(InetAddress addr, int port, boolean streaming)   *This constructor is deprecated. Use `Socket(InetAddress, int)` instead of this for streaming sockets or an appropriate constructor of `DatagramSocket` for UDP transport.*

Socket(InetAddress dstAddress , int dstPort)

- *public Socket (InetAddress dstAddress , int dstPort)*

- Creates a new streaming socket connected to the target host.

- Parameters
  - *dstAddress* : the target host address to connect to
  - *dstPort* : the port on the target host to connect to

We thank khchen and Ting-An Lin for sharing his slides with us

getOutputStream()

- *public OutputStream getOutputStream()*
- Returns an output stream to write data into this socket

getInputStream()

- *public InputStream getInputStream()*

- Returns an input stream to read data from this socket

public boolean isConnected()

- Returns whether this socket is connected to a remote host.

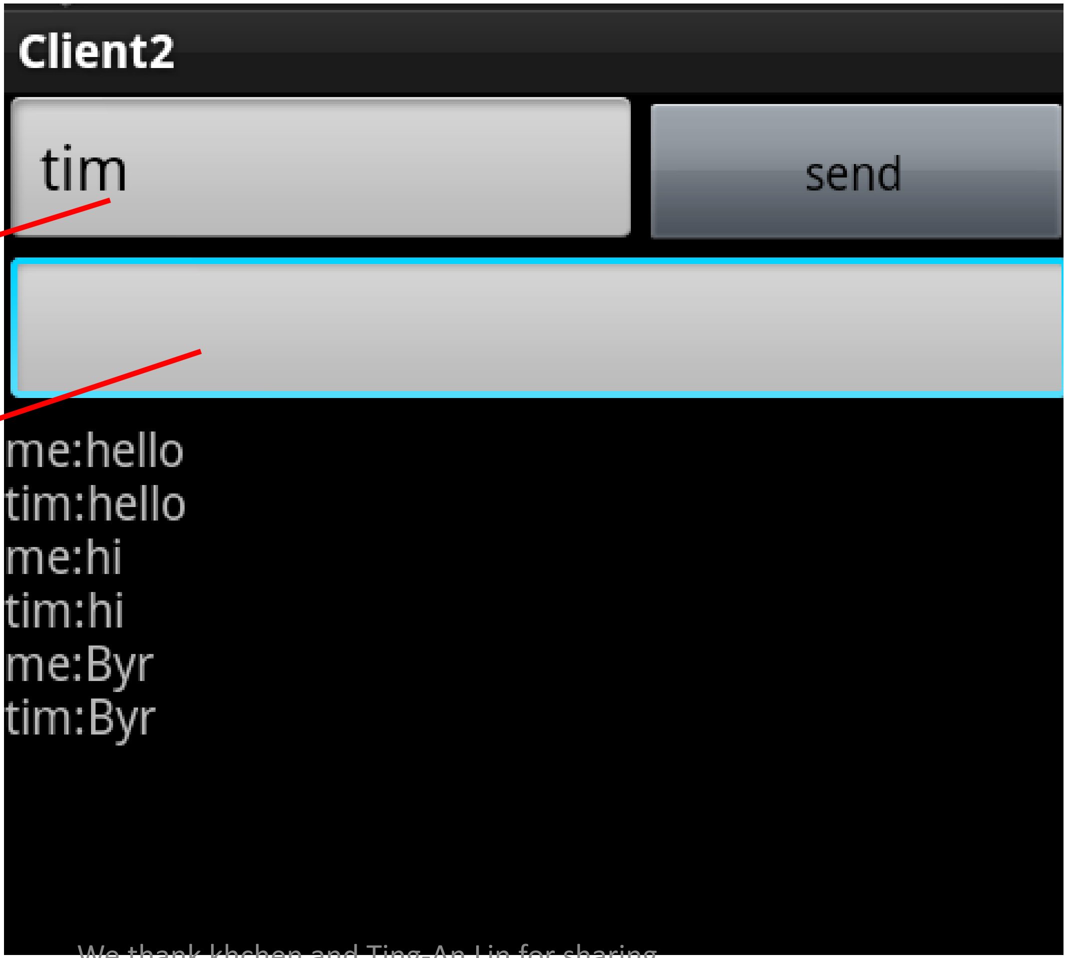
- Returns
  - true if the socket is connected, false otherwise.

Simple example

- A simple socket programming example
- A client-server architecture
- The server in running on PC, client is running on Android

We thank khchen and Ting-An Lin for sharing his slides with us

Demo

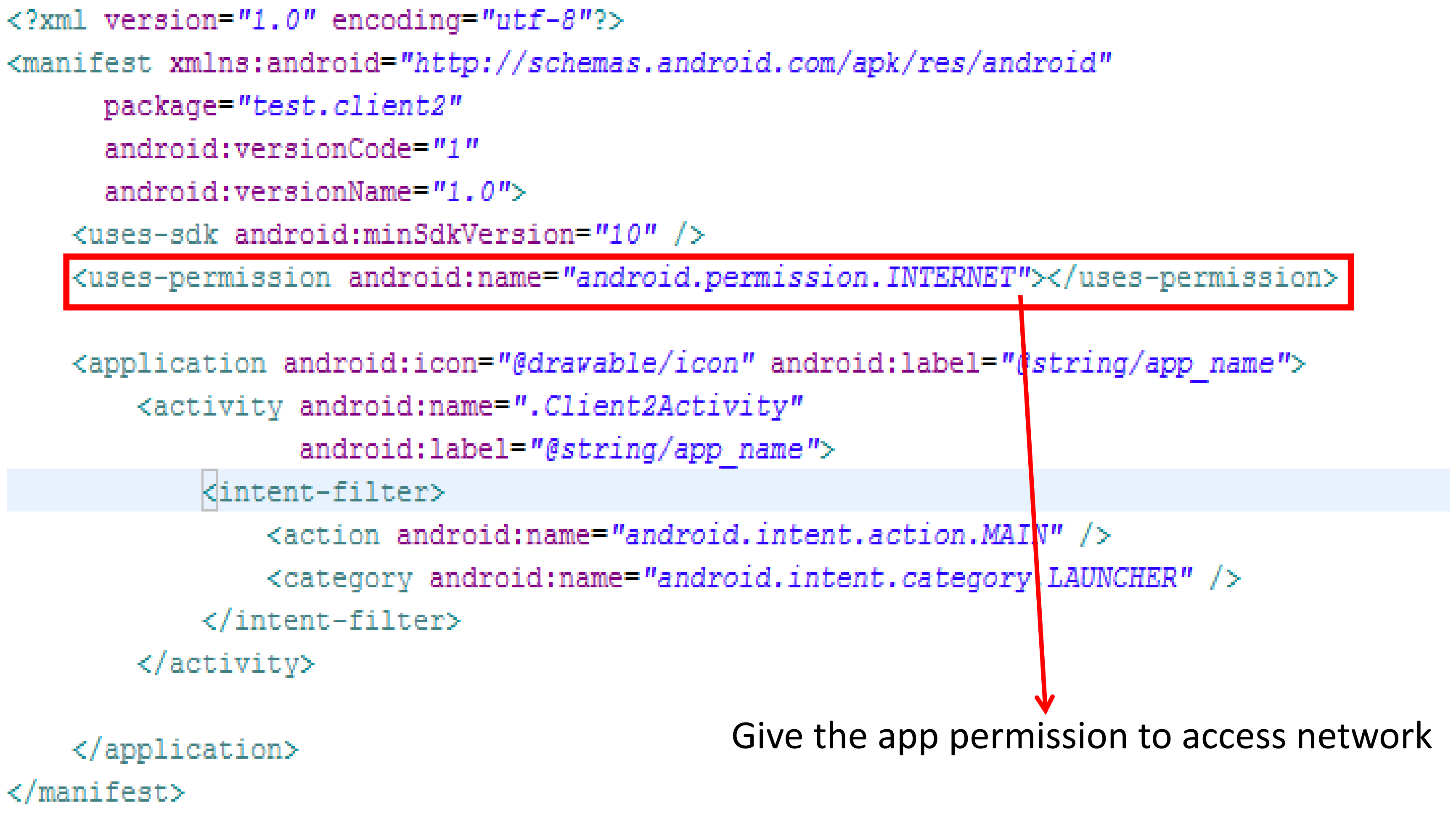


```
me:hello
tim:hello
me:hi
tim:hi
me:Byr
tim:Byr
```

User name

User message

AndroidManifest.xml



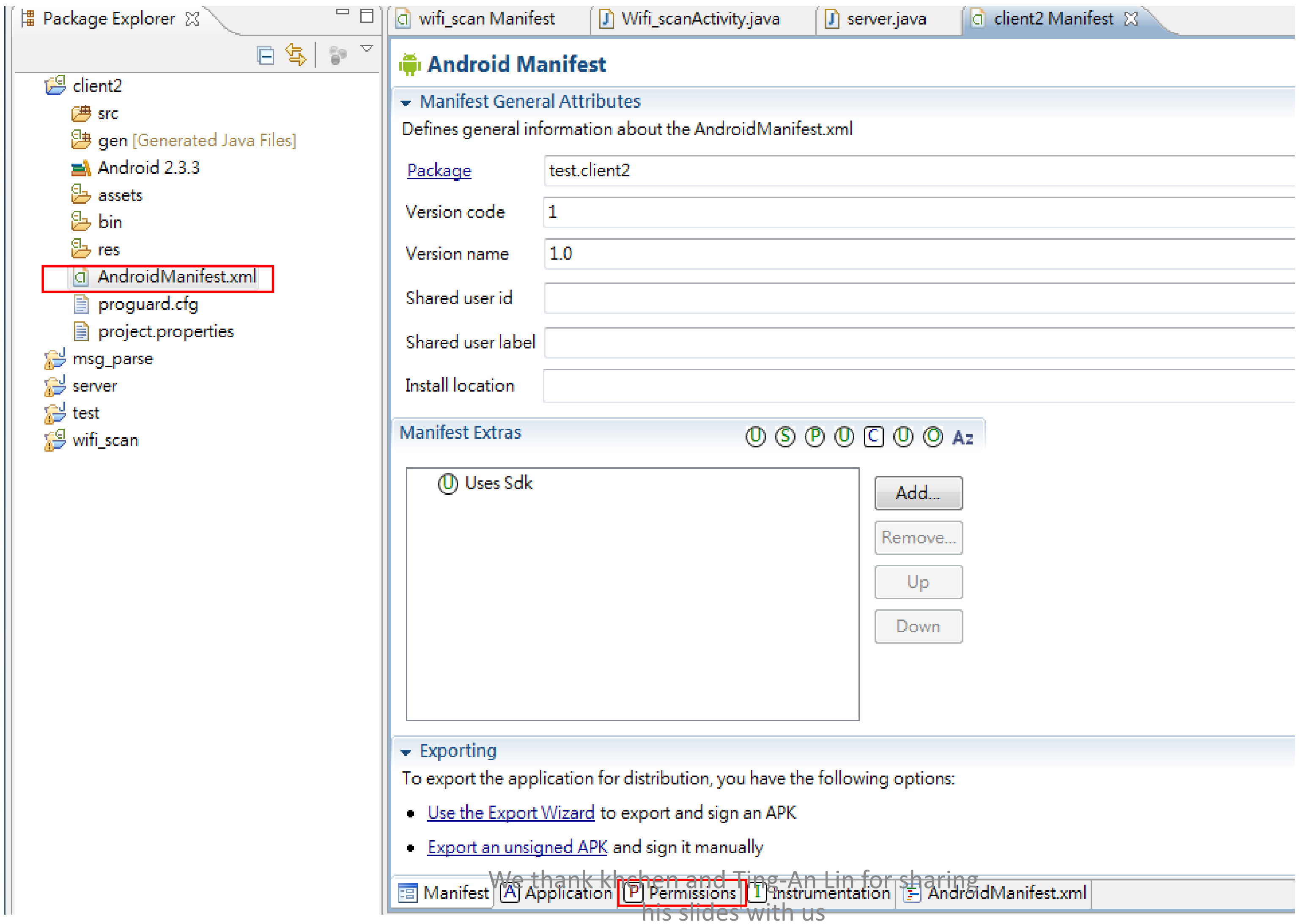
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="test.client2"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="10" />
    <uses-permission android:name="android.permission.INTERNET"></uses-permission>

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Client2Activity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
</manifest>
```

Give the app permission to access network

Add permission(1/3)



Android Manifest

Manifest General Attributes

Defines general information about the AndroidManifest.xml

Package	test.client2
Version code	1
Version name	1.0
Shared user id	
Shared user label	
Install location	

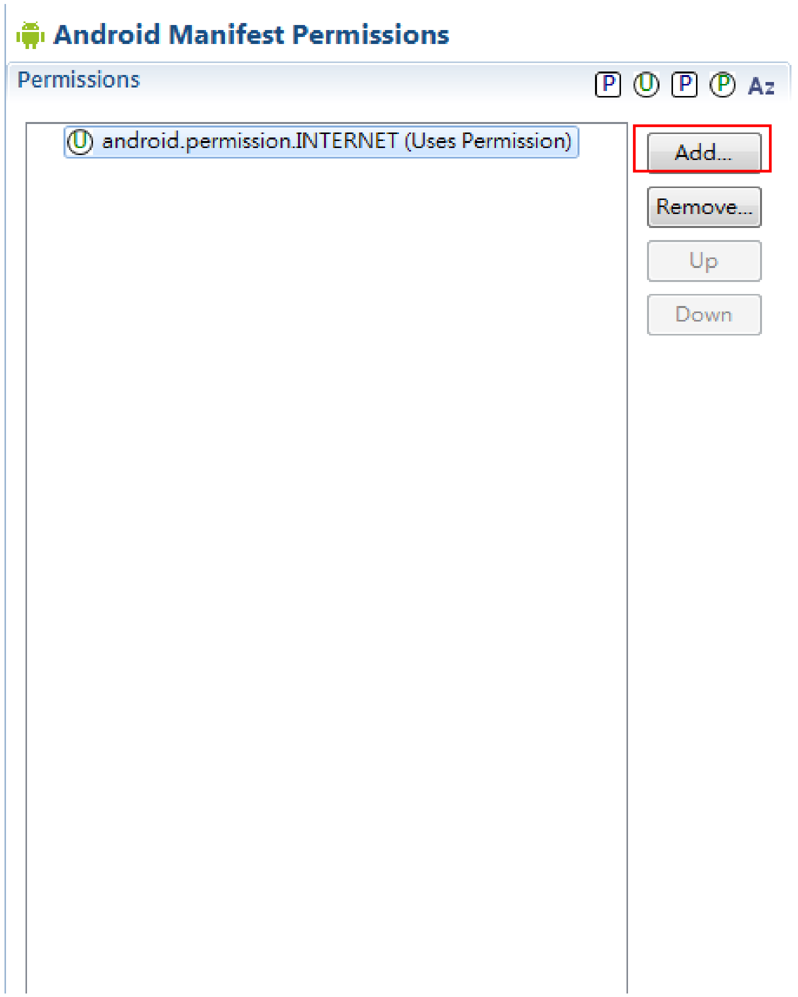
Exporting

To export the application for distribution, you have the following options:

- Use the Export Wizard to export and sign an APK
- Export an unsigned APK and sign it manually

We thank khchen and Ting-An Lin for sharing his slides with us

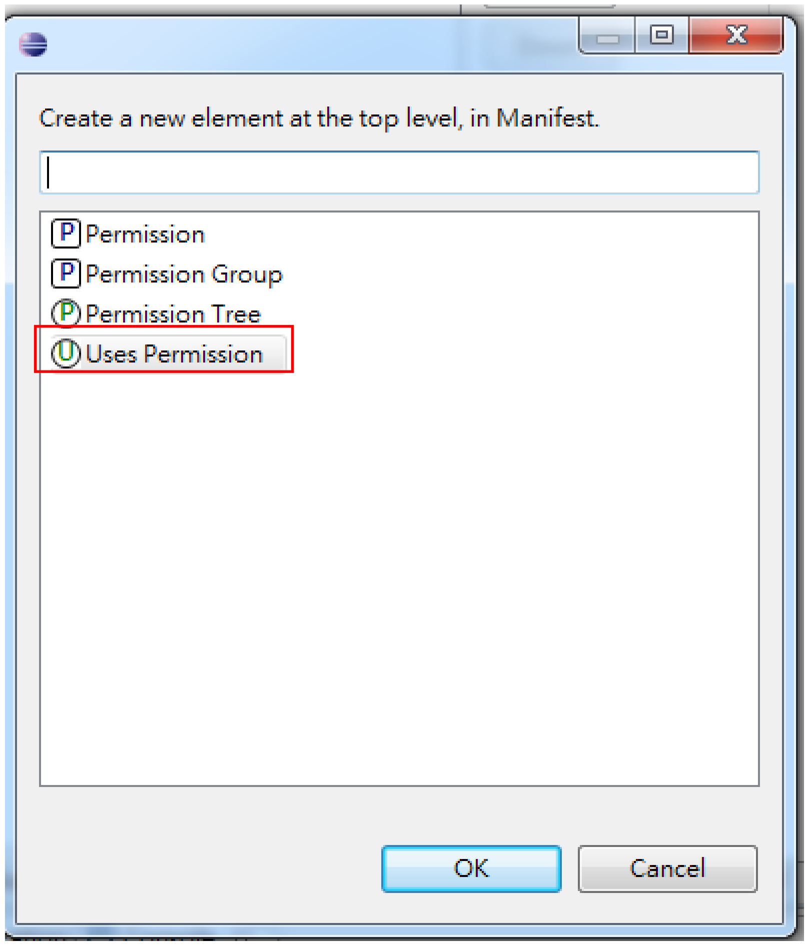
Add permission(2/3)



Android Manifest Permissions

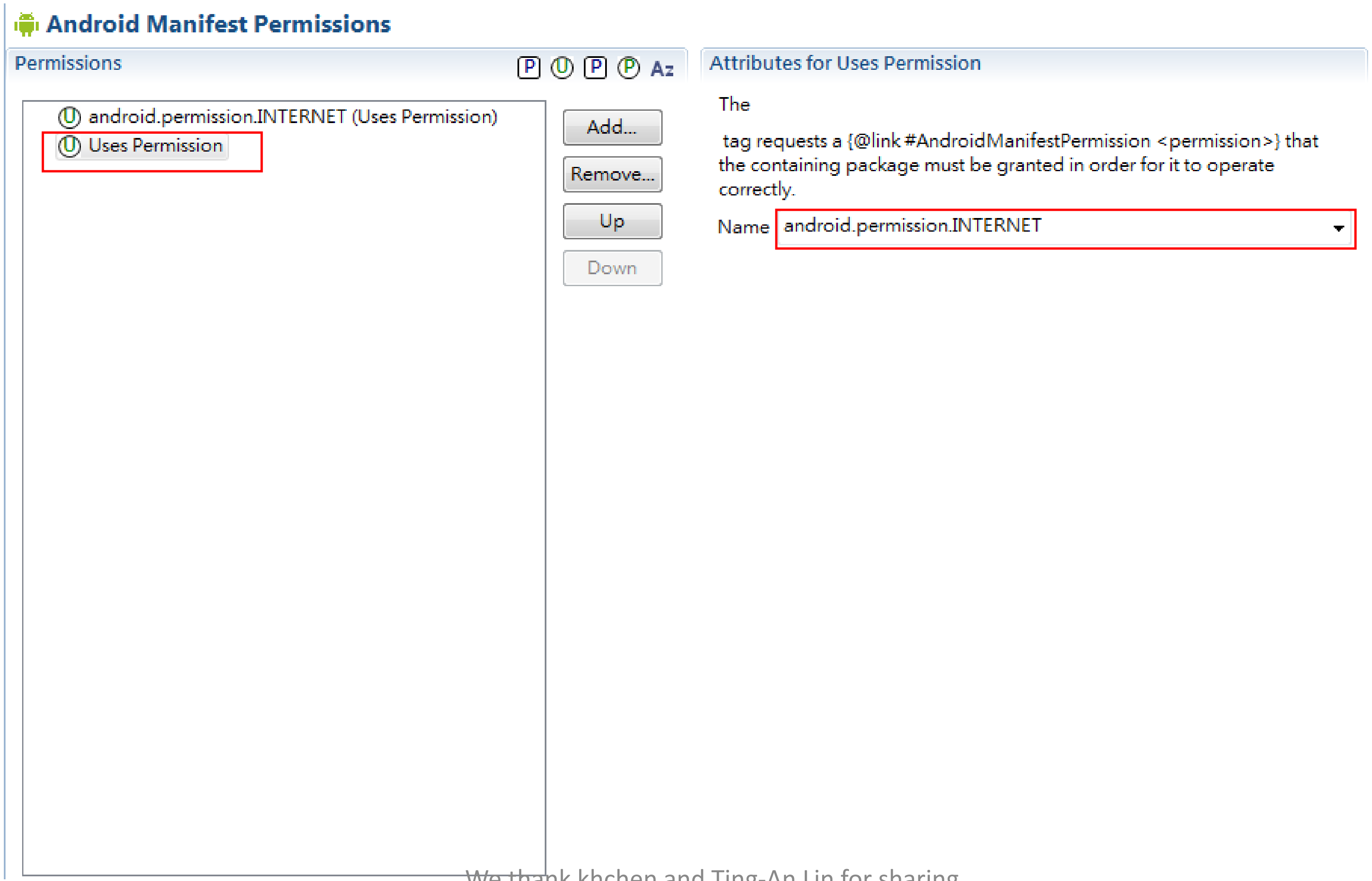
Permissions

android.permission.INTERNET (Uses Permission)



Create a new element at the top level, in Manifest.

Add permission(3/3)



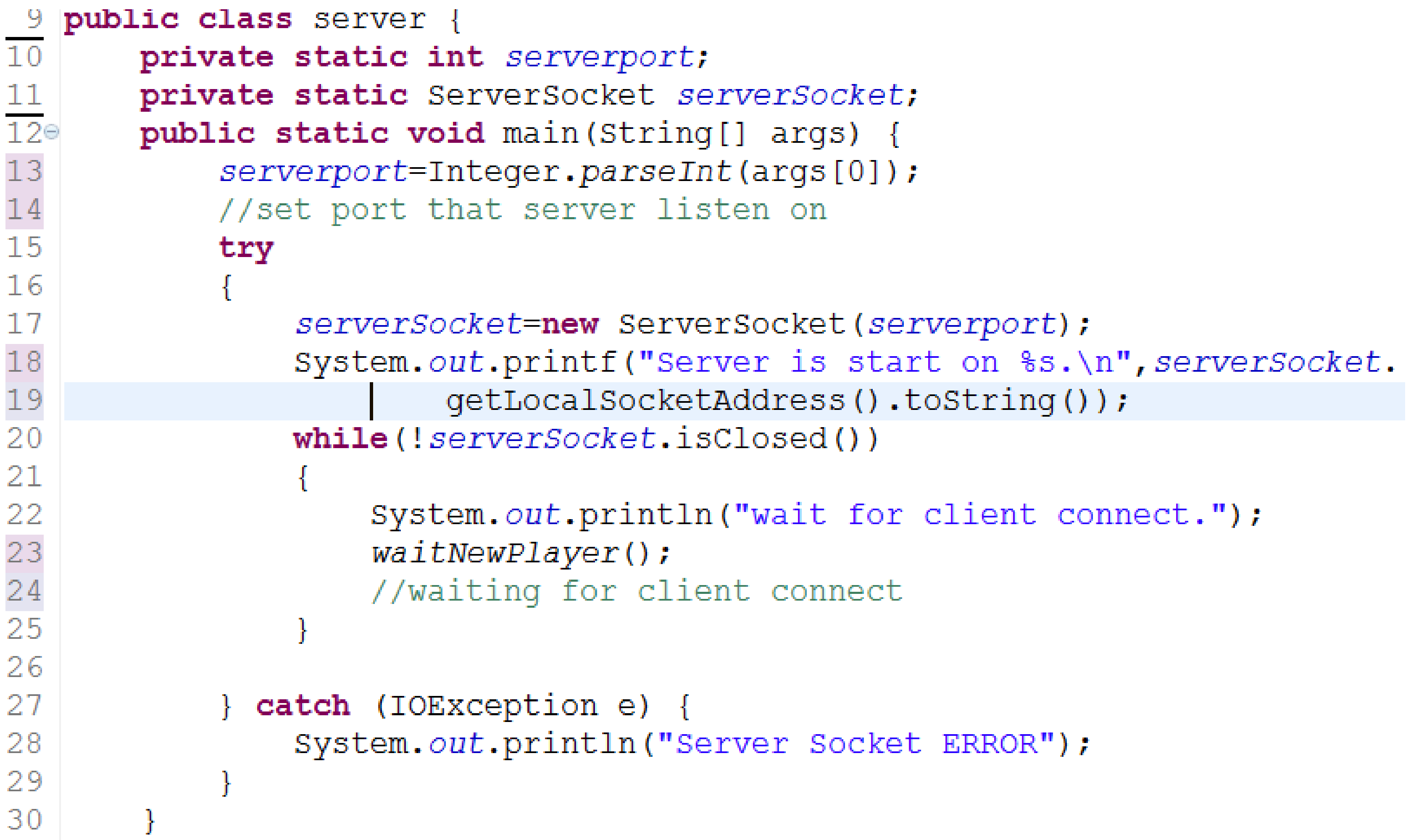
android.permission.INTERNET (Uses Permission)
Uses Permission

The

tag requests a {@link #AndroidManifestPermission <permission>} that the containing package must be granted in order for it to operate correctly.

Name: android.permission.INTERNET

Server (1/3)

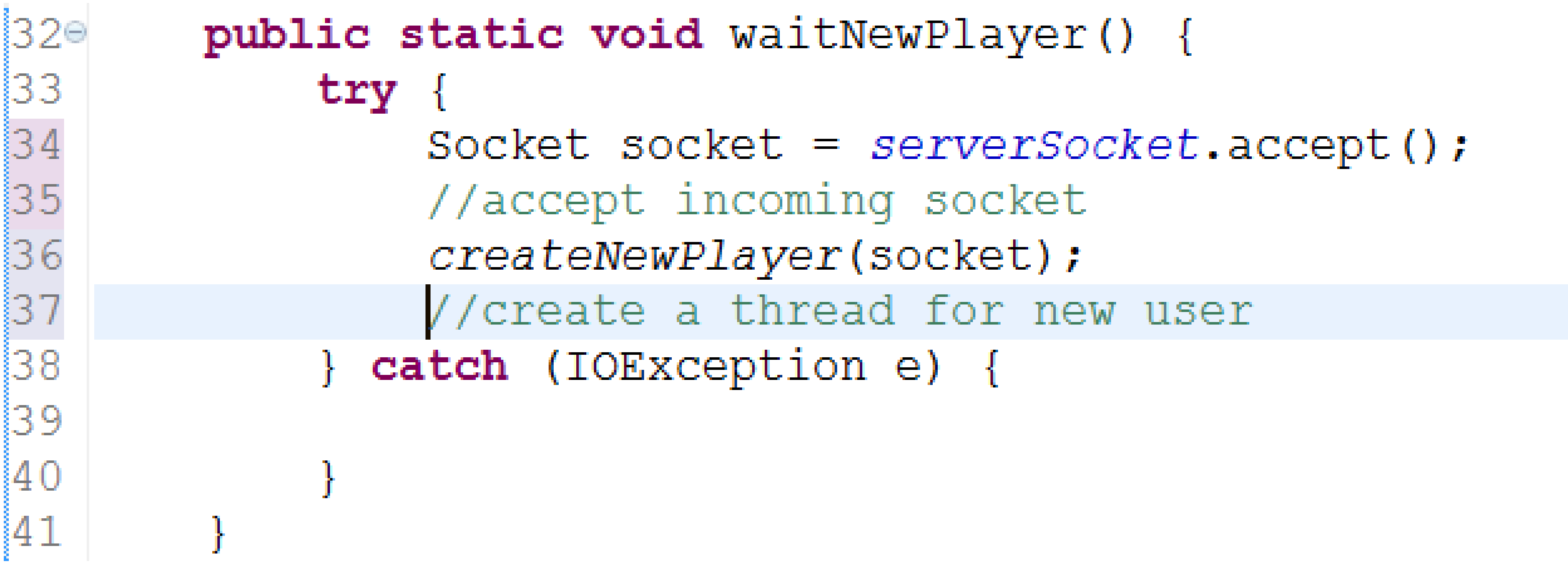


```
public class server {
    private static int serverport;
    private static ServerSocket serverSocket;
    public static void main(String[] args) {
        serverport=Integer.parseInt(args[0]);
        //set port that server listen on
        try
        {
            serverSocket=new ServerSocket(serverport);
            System.out.printf("Server is start on %s.\n",serverSocket.
                        getLocalSocketAddress().toString());
            while(!serverSocket.isClosed())
            {
                System.out.println("wait for client connect.");
                waitNewPlayer();
                //waiting for client connect
            }

        } catch (IOException e) {
            System.out.println("Server Socket ERROR");
        }
    }
```

We thank khchen and Ting-An Lin for sharing his slides with us

Server (2/3)

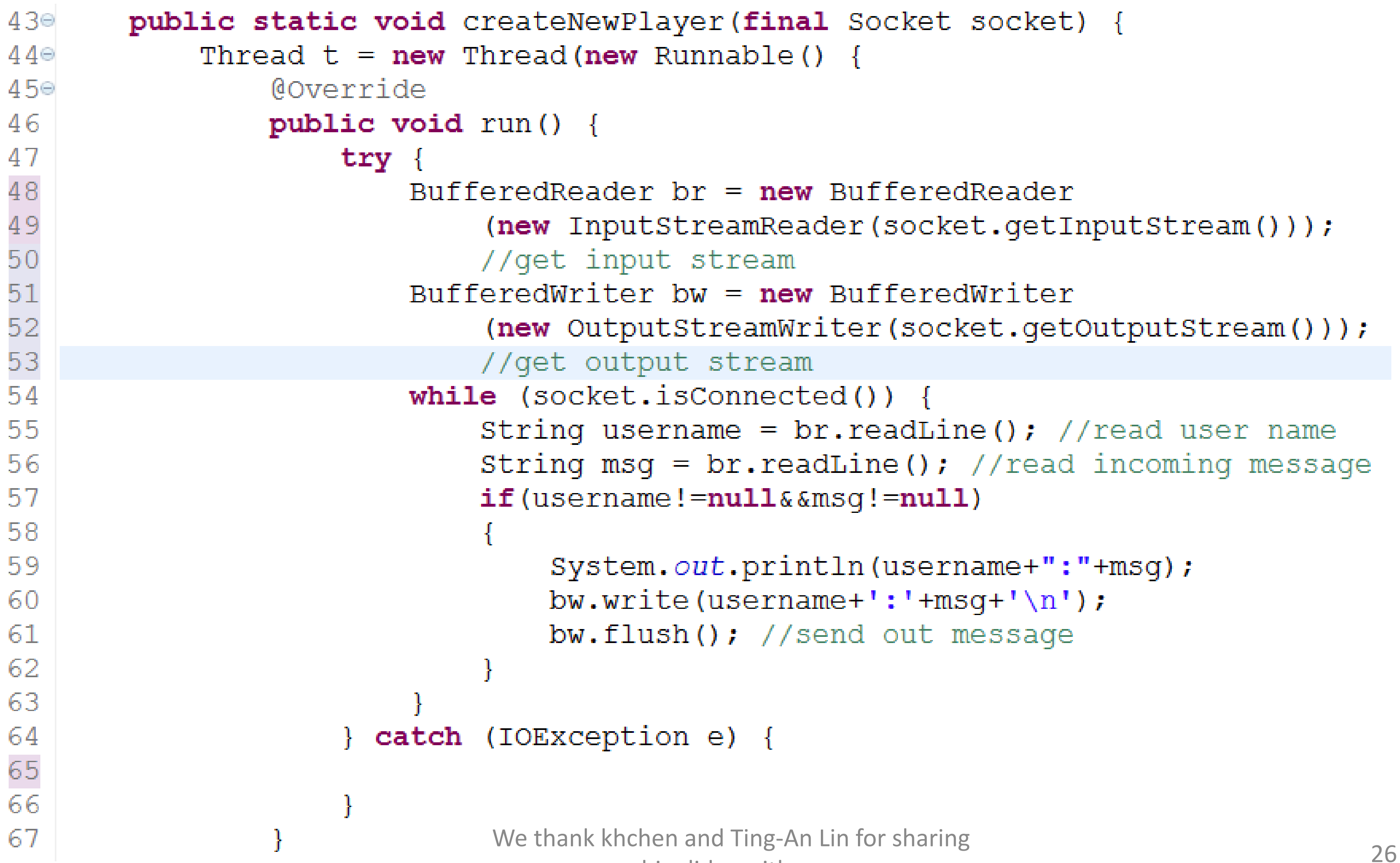


```java
    public static void waitNewPlayer() {
        try {
            Socket socket = serverSocket.accept();
            //accept incoming socket
            createNewPlayer(socket);
            //create a thread for new user
        } catch (IOException e) {

        }
    }
```

We thank khchen and Ting-An Lin for sharing his slides with us

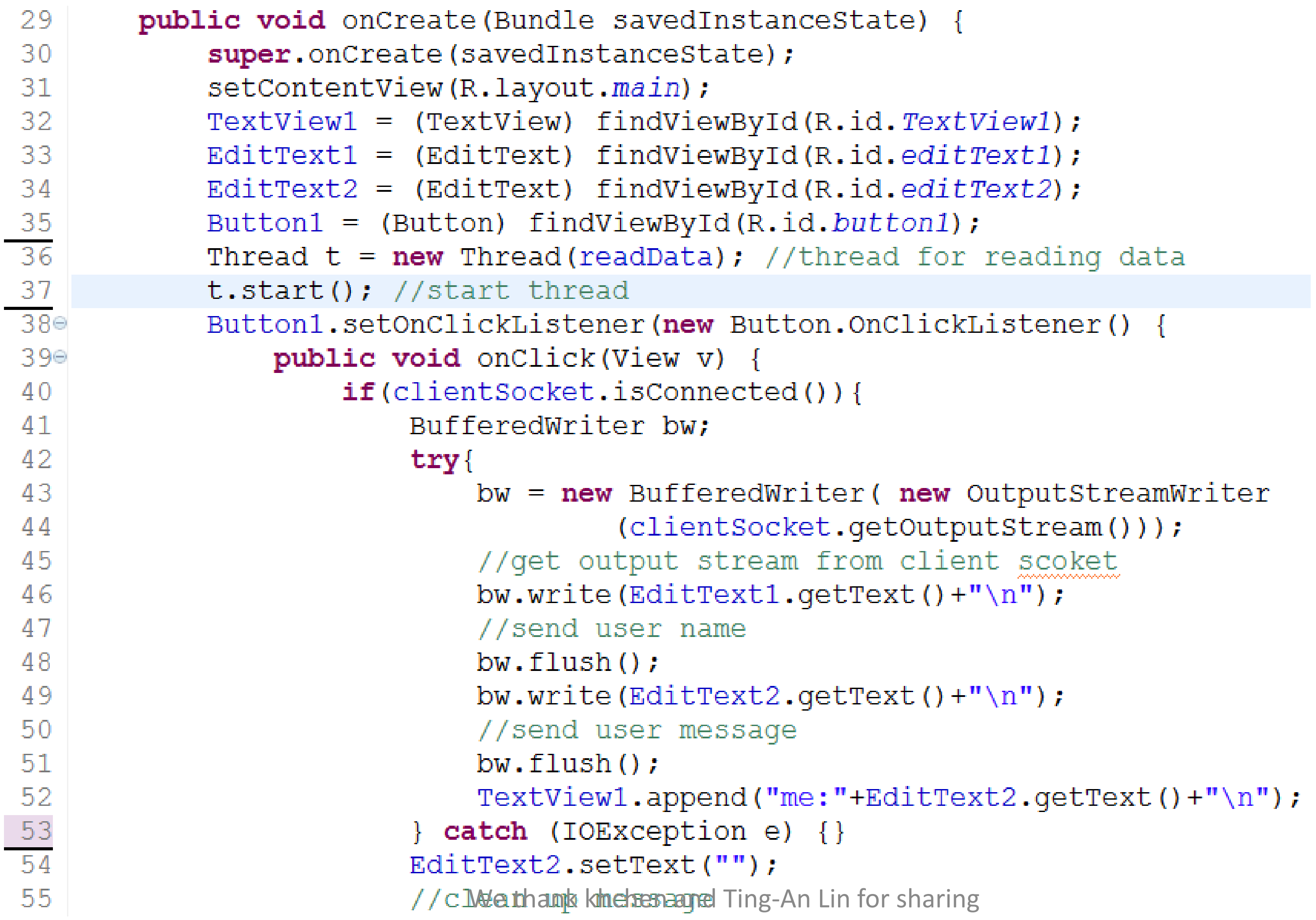
Server(3/3)



```java
    public static void createNewPlayer(final Socket socket) {
        Thread t = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    BufferedReader br = new BufferedReader
                        (new InputStreamReader(socket.getInputStream()));
                        //get input stream
                    BufferedWriter bw = new BufferedWriter
                        (new OutputStreamWriter(socket.getOutputStream()));
                        //get output stream
                    while (socket.isConnected()) {
                        String username = br.readLine(); //read user name
                        String msg = br.readLine(); //read incoming message
                        if(username!=null&&msg!=null)
                        {
                            System.out.println(username+":"+msg);
                            bw.write(username+':'+msg+'\n');
                            bw.flush(); //send out message
                        }
                    }
                } catch (IOException e) {

                }
            }
```

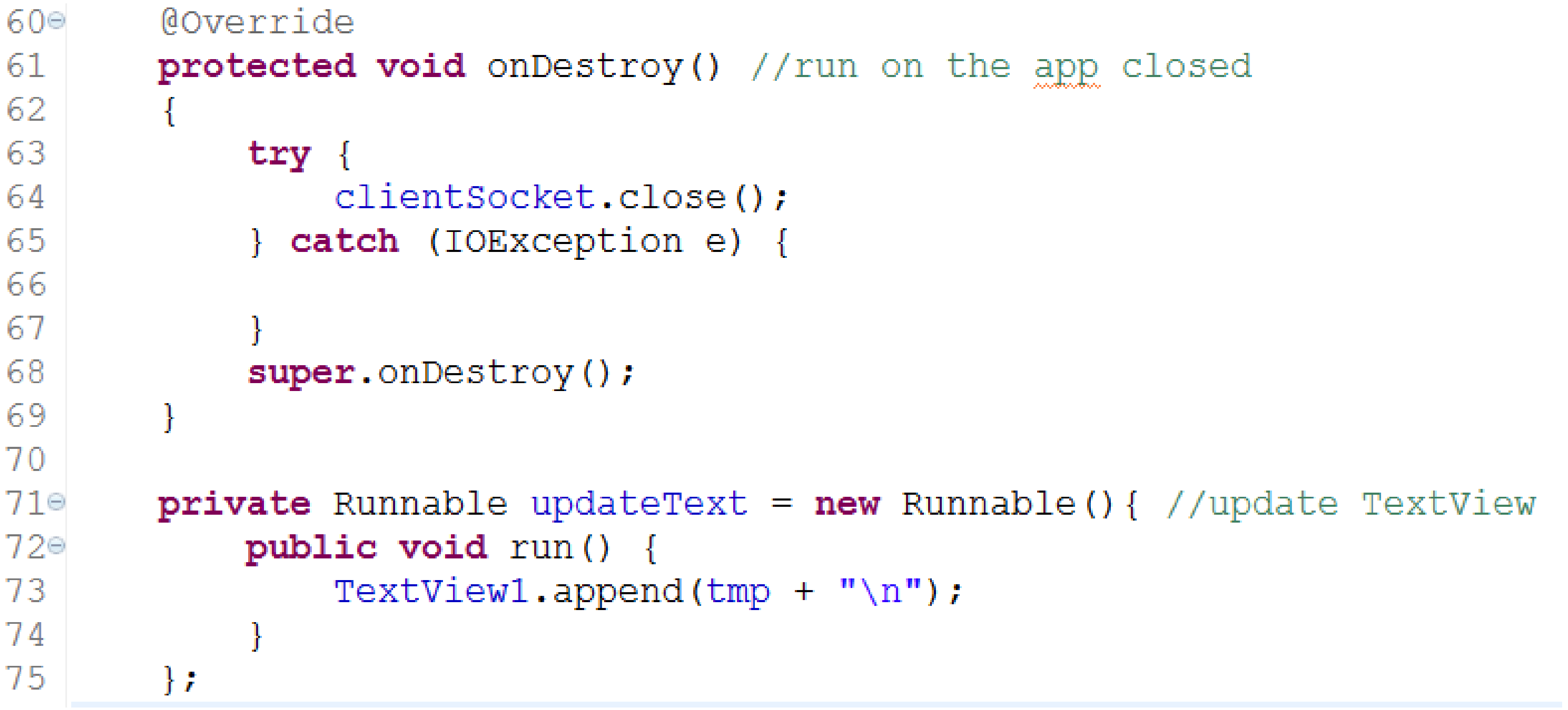
Client(1/3)



```java
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView1 = (TextView) findViewById(R.id.TextView1);
        EditText1 = (EditText) findViewById(R.id.editText1);
        EditText2 = (EditText) findViewById(R.id.editText2);
        Button1 = (Button) findViewById(R.id.button1);
        Thread t = new Thread(readData); //thread for reading data
        t.start(); //start thread
        Button1.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                if(clientSocket.isConnected()){
                    BufferedWriter bw;
                    try{
                        bw = new BufferedWriter( new OutputStreamWriter
                                (clientSocket.getOutputStream()));
                        //get output stream from client scoket
                        bw.write(EditText1.getText()+"\n");
                        //send user name
                        bw.flush();
                        bw.write(EditText2.getText()+"\n");
                        //send user message
                        bw.flush();
                        TextView1.append("me:"+EditText2.getText()+"\n");
                    } catch (IOException e) {}
                    EditText2.setText("");
                    //clear up message
```

We thank Ting-An Lin for sharing his slides with us

Client(2/3)



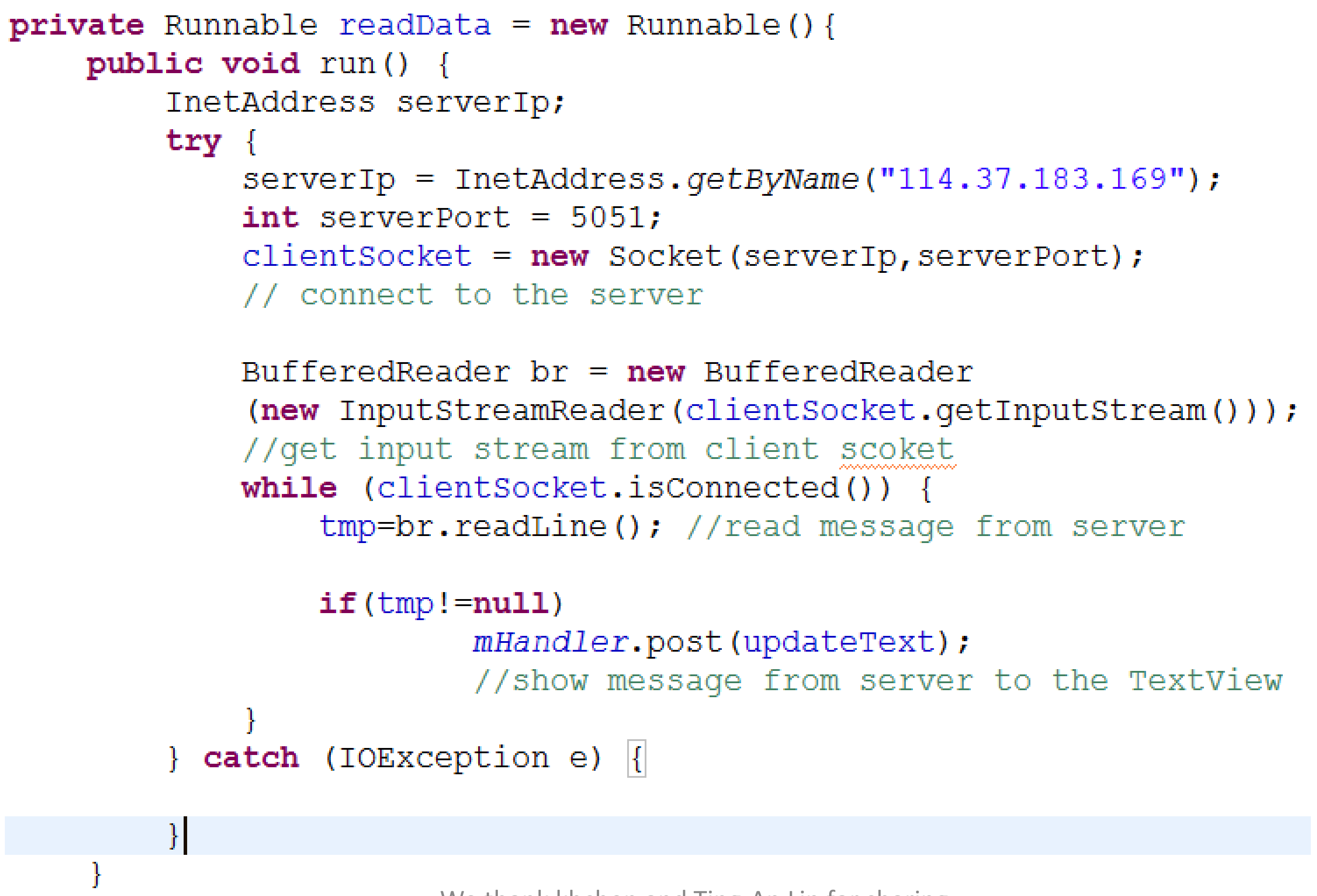
```java
    @Override
    protected void onDestroy() //run on the app closed
    {
        try {
            clientSocket.close();
        } catch (IOException e) {

        }
        super.onDestroy();
    }

    private Runnable updateText = new Runnable(){ //update TextView
        public void run() {
            TextView1.append(tmp + "\n");
        }
    };
```

We thank khchen and Ting-An Lin for sharing
his slides with us

Client(3/3)



```
private Runnable readData = new Runnable(){
    public void run() {
        InetAddress serverIp;
        try {
            serverIp = InetAddress.getByName("114.37.183.169");
            int serverPort = 5051;
            clientSocket = new Socket(serverIp,serverPort);
            // connect to the server

            BufferedReader br = new BufferedReader
            (new InputStreamReader(clientSocket.getInputStream()));
            //get input stream from client scoket
            while (clientSocket.isConnected()) {
                tmp=br.readLine(); //read message from server

                if(tmp!=null)
                        mHandler.post(updateText);
                        //show message from server to the TextView
            }
        } catch (IOException e) {

        }
    }
```

We thank khchen and Ting-An Lin for sharing his slides with us

Exercise1

- Spec
  - Modify the example to support multi-users
- Hint:
  - How to handle different user in the same time
    - multi-thread
  - How to broadcast message to all users
    - Different listen on different socket

Exercise 2

- Spec
  - Implement a simple app to transfer file between 2 android phones

- Hint:
  - Read file in SD card
    - Java.io.FileInputStream
  - Send file via network
    - Scoket
  - Write file into the reciver
    - Java.io.FileOutputStream

Write and read file in SD card (1/3)

- Need user permission

  android.permission.WRITE_EXTERNAL_STORAGE

- Get the state of SD card by *Environment.getExternalStorageState ()*

- Gets the external storage directory by *Environment.getExternalStorageDirectory ()*

We thank khchen and Ting-An Lin for sharing his slides with us

Write and read file in SD card (2/3)

- Create a file object by constructer
  Flie (getExternalStorageDirectory () , aa/bb.txt)

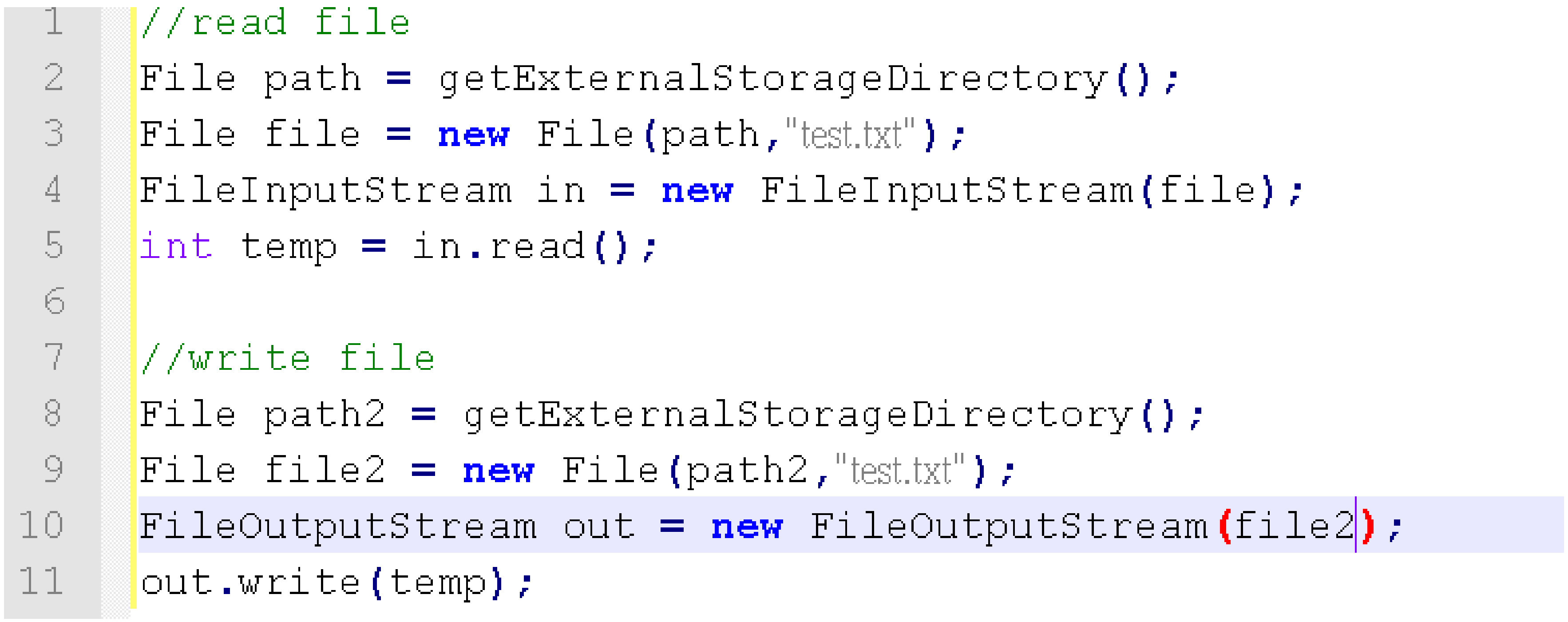
- Check the path of file is exist or not by *File.exist()*

- *Chreate* the path of file by *Flie.mkdir()*

Write and read file in SD card (3/3)

- Create new file by *File.createNewFile()*
- Write single byte data to file by *FileOutputStream.write(int oneByte)*
- Use FileInputStream.read() to get signle byte from file

Example

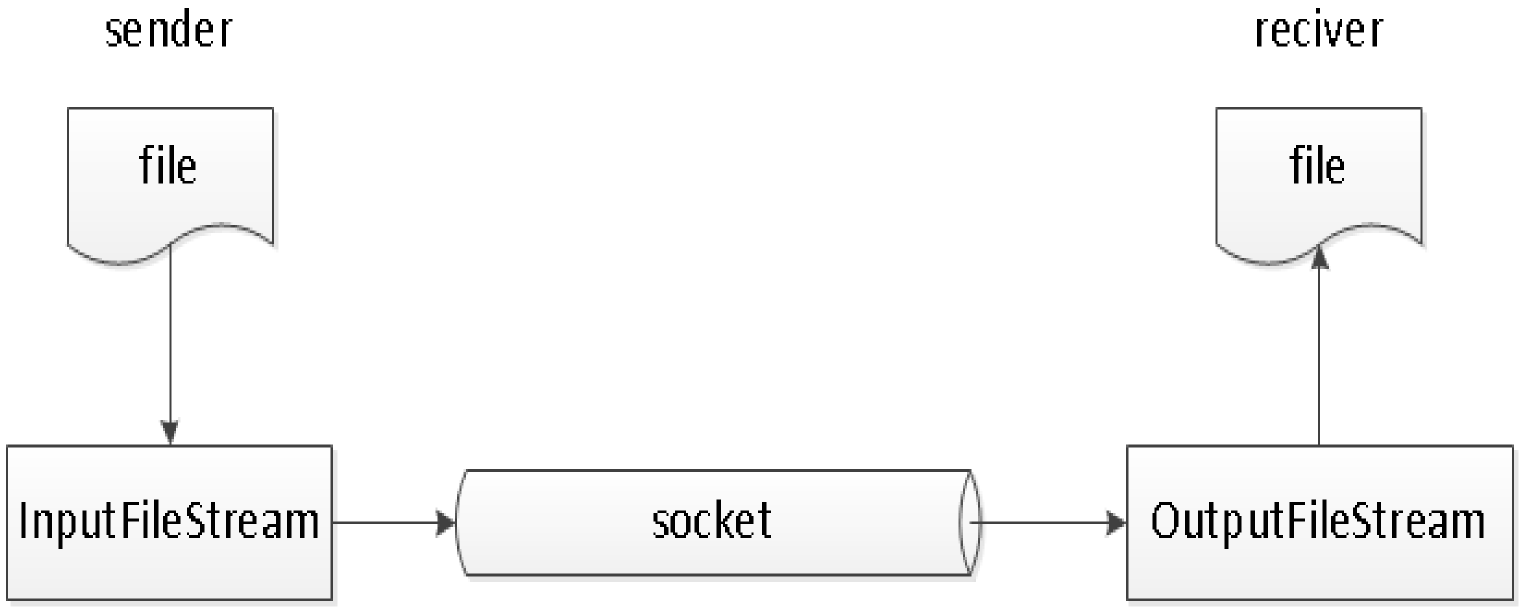


```
//read file
File path = getExternalStorageDirectory();
File file = new File(path,"test.txt");
FileInputStream in = new FileInputStream(file);
int temp = in.read();

//write file
File path2 = getExternalStorageDirectory();
File file2 = new File(path2,"test.txt");
FileOutputStream out = new FileOutputStream(file2);
out.write(temp);
```

We thank khchen and Ting-An Lin for sharing
his slides with us

Transfer file by socket



Reference

- http://developer.android.com/reference/android/net/wifi/p2p/package-summary.html