가천대학교
Gachon University

학생 여러분 반갑습니다.

다른 친구들이 입장할 때까지 조금 기다려 주십시오.

곧 모바일 프로그래밍 수업을 시작합니다.

음소거( )가 되었는지 확인 바랍니다.

모바일 프로그래밍
화목(1,2교시)/ 화목(3,4교시)
정윤현 (AI/소프트웨어학부)

Mobile Programming

Android Programming

Chap 8-1. Data Persistence, Networking

영속성.

Prof. Younhyun Jung

Email) younhyun.jung@gachon.ac.kr

SharedPreferences.
SQLite
Network
File I/O.

가천대학교 소프트웨어학과
Gachon University

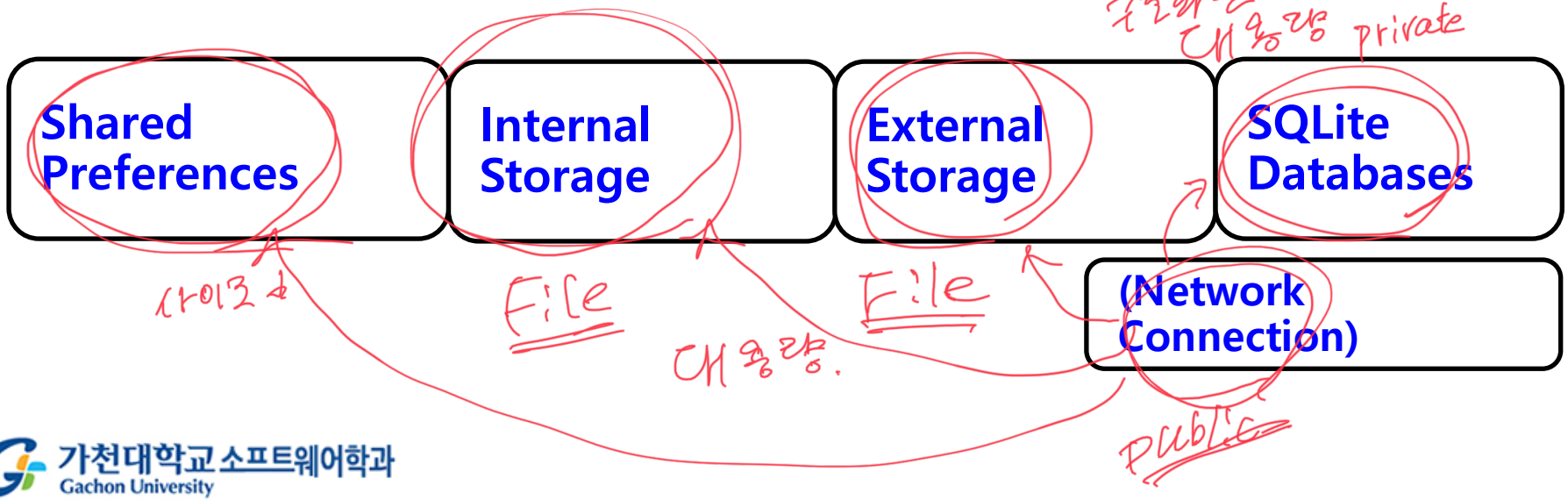
Objective

- Learn How to **persist data** in your Android apps.
- In particular, you will learn
  - How to save simple data using Shared Preference objects
  - How to write and read files in internal and external **storage** (Traditional local file system)
- Later
  - How to create and use a SQLite database
- http://developer.android.com/guide/topics/data/data-storage.html

Choosing a Persistent Environment

- Your permanent data storage destination is usually determined by parameters such as:
  - size (small/large),
  - location (internal/external),
  - accessibility (private/public).

- Depending on your situation the following options are available:



Android Data Storage

- Android provides several options to save persistent application data:
  - **Shared Preferences :** To save small chunks of data (device's private memory area) in key-value pairs
  - **Internal Storage, *External Storage* :** Traditional file systems on the device's (internal) main memory or shared external storage
  - **SQLite Databases:** Store structured data through the support of SQLite DBs

Preferences

Preferences

- Suppose, for example,
  - your application may have an option that enables users to specify the font size of text → need to remember, where??
  - To do so, there are several solutions
    - 1) using file
    - 2) using database
    - 3) preference

☻ saving simple data to a database is overkill

- Android provides the SharedPreferences object to help you save simple application data.

Preferences (SharedPreferences)

KEY	VALUE

- A *lightweight* mechanism to store and retrieve *<key-value>* pairs of primitive data types
- *Preferences are typically used to*
  - keep state information and
  - shared data among several activities of an application.

- In each entry of the form *<key-value>* the *key* is a string and the *value* must be a primitive data type.

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<map>
    <int name="textSize" value="20" />
    <string name="textStyle">bold</string>
    <string name="DateLastExecution">Oct 26, 2011 6:23:00 PM</string>
    <int name="layoutColor" value="-16711936" />
    <int name="backColor" value="-16776961" />
</map>
```

Preferences API calls (con't)

```
import android.content.SharedPreferences;
```

- Shared preference API
  - The shared preferences can be used by other application components in the application
    - from within your Activity to access application-level preferences
  - To be stored in a file specified : getSharedPreferences()

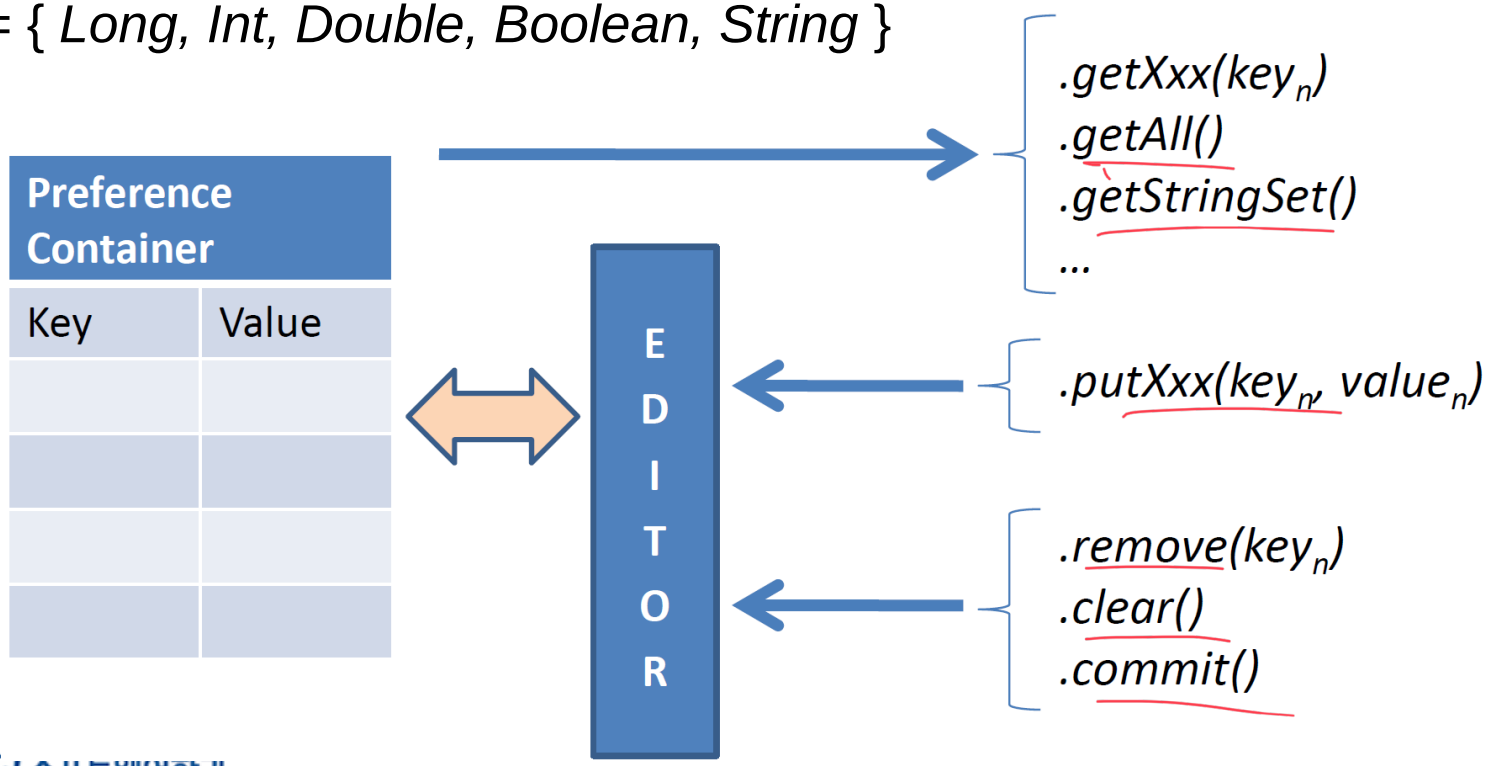
Singleton

- Example
  - getSharedPreferences()

```
public static final String PREF_FILE_NAME = "PrefFile"; …
  SharedPreferences preferences = getSharedPreferences(PREF_FILE_NAME, Activity.MODE_PRIVATE)
```

Con't

- **Using Preferences API calls**
  - Each of the Preference mutator methods carries a typed-value content that can be manipulated by an ***editor*** that allows *putXxx… and getXxx…* commands to place data in and out of the Preference container.
- *Xxx = { Long, Int, Double, Boolean, String }*



Example: Shared Preferences

- Good for a few items saved as <Name, Value>

```
private void usingPreferences(){
  // Save data in a SharedPreferences container
  // We need an Editor object to make preference changes.

  SharedPreferences settings = getSharedPreferences("my_preferred_Choices",
                                                    Context.MODE_PRIVATE);

  SharedPreferences.Editor editor = settings.edit();
          editor.putString("favorite_color", "#ff0000ff");
          editor.putInt("favorite_number", 101);
  editor.commit();

  // retrieving data from SharedPreferences container
  String favColor = settings.getString("favorite_color", "default black");
  int favNumber = settings.getInt("favorite_number", 0);

  Toast.makeText(this, favColor + " " + favNumber, 1).show();

}
```

EDITOR

STORE

READ

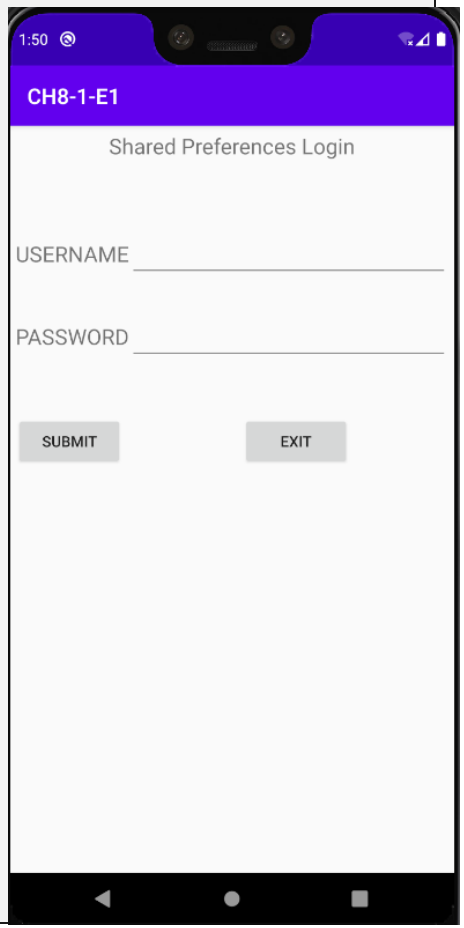
default value

In this example the user selects a preferred 'color' and 'number'. Both values are stored in a SharedPreferences file.

Exercise

- activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:padding="5dp" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:text="Shared Preferences Login"
        android:textSize="20dp" />

    <TextView
        android:id="@+id/usertext"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="100dp"
        android:text="USERNAME"
        android:textSize="20dp" />

    <EditText
        android:id="@+id/userinput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/usertext"
        android:layout_toRightOf="@+id/usertext" />
```



Exercise

```
 <TextView
        android:id="@+id/userpass"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/usertext"
        android:layout_marginTop="50dp"
        android:text="PASSWORD"
        android:textSize="20dp" />

    <EditText
        android:id="@+id/passinput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"

android:layout_alignBaseline="@+id/userpass"

android:layout_toRightOf="@+id/userpass"
        android:inputType="textPassword" />
```

```
 <Button
        android:id="@+id/submit"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_below="@+id/userpass"
        android:layout_marginTop="60dp"
        android:text="Submit" />

    <Button
        android:id="@+id/exit"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_below="@+id/userpass"
        android:layout_marginLeft="110dp"
        android:layout_marginTop="60dp"
        android:layout_toRightOf="@+id/submit"
        android:text="Exit" />

</RelativeLayout>
```

MainActivity.java

Exercise

```
package org.android.customview.myapplication;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
```

```
public class MainActivity extends AppCompatActivity {
    Button submit, exit;
    String username, password;
    EditText userinput, passinput;
    SharedPreferences sh_Pref;
    SharedPreferences.Editor toEdit;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        submit = findViewById(R.id.submit);
        exit = findViewById(R.id.exit);
        userinput = findViewById(R.id.userinput);
        passinput = findViewById(R.id.passinput);
        submit.setOnClickListener(new Button.OnClickListener(){
            @Override
            public void onClick(View view) {
                username = userinput.getText().toString();
                password = passinput.getText().toString();
                sharedPreference();
                Toast.makeText(getApplicationContext(), "Details are saved", Toast.LENGTH_LONG).show();
            }
        });
        exit.setOnClickListener(new Button.OnClickListener(){
             @Override
             public void onClick(View view) {
                 finish();
             }
        });

        applySharedPreference();
    }
```

앱 종료

SharedPref. 저장된 값 로드

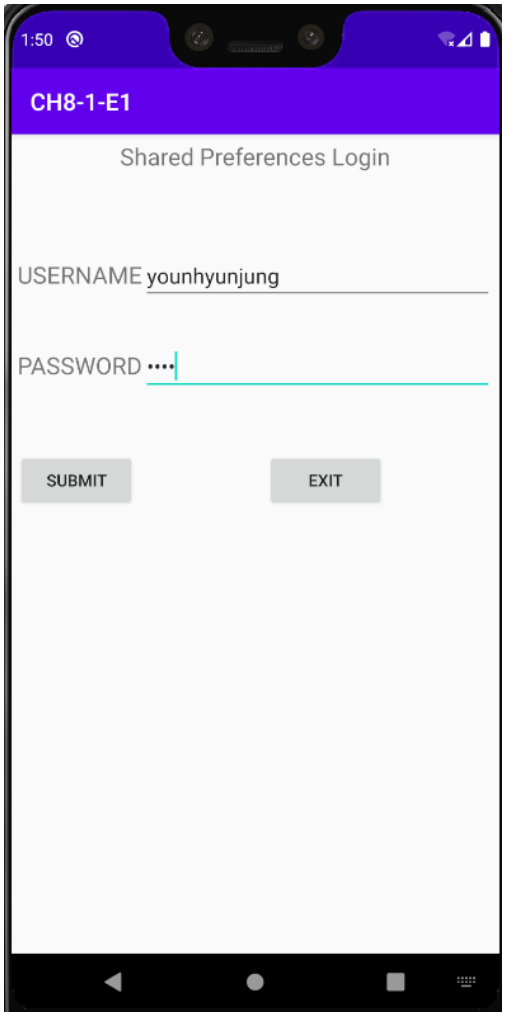
Exercise

```
public void sharedPreference() {
        sh_Pref = getSharedPreferences("Login Credentials", MODE_PRIVATE);
        toEdit = sh_Pref.edit();
        toEdit.putString("Username", username);
        toEdit.putString("Password", password);
        toEdit.commit();
    }

    public void applySharedPreference(){
        sh_Pref = getSharedPreferences("Login Credentials", MODE_PRIVATE);
        if (sh_Pref!=null && sh_Pref.contains("Username")){
            String name = sh_Pref.getString("Username", "noname");
            userinput.setText(name);
        }

    }
}
```

Exercise

- Run!
  - Enter your name & password
  - Click the submit button
  - Click the Exit
  - Reopen the application

  - You may see the USERNAME



Persisting Data to Files

-Saving to Internal/External Storage

Android Files

- Android uses the same *file* constructions found in a typical Java application.
- Files can be stored in the device's (small) main memory or in the much larger SD card.
  - They can also be obtained from the network
- Files stored in the device's memory, stay together with other application's resources (such as icons, pictures, music, …).
- Internal files are called: *Resource Files.*

Using the Internal Storage

- You can save files directly on the device's internal storage.
  - By default, files saved to the internal storage are private to your application and other applications cannot access them
  - When the user uninstalls your application, these files are removed.

Using the Internal Storage

- To create and write a private file to the internal storage:
  1. Call openFileOutput() with the name of the file and the operating mode. (안드로이드에서 정의 된 method → internal storage에 접근 한정)
     This returns a FileOutputStream.
  2. Write bytes to the file with write().
  3. Close the stream with close().

- To read a file from internal storage:
  1. Call openFileInput() and pass the name of the file to be read.
     This returns a FileInputStream.
  2. Read bytes from the file with read().
  3. Close the stream with close().

Using the Internal Storage

- `FileOutputStream openFileOutput (String name, int mode)`

```
FileOutputStream fOut = openFileOutput("textfile.txt", MODE_WORLD_READABLE);
```

- **mode**
  - **MODE_PRIVATE** (file can only be accessed by the application that created it),
  - **MODE_APPEND** (for appending to an existing file),
  - **MODE_WORLD_READABLE** (all other applications have read access to the file),
  - **MODE_WORLD_WRITEABLE** (all other applications have write access to the file).

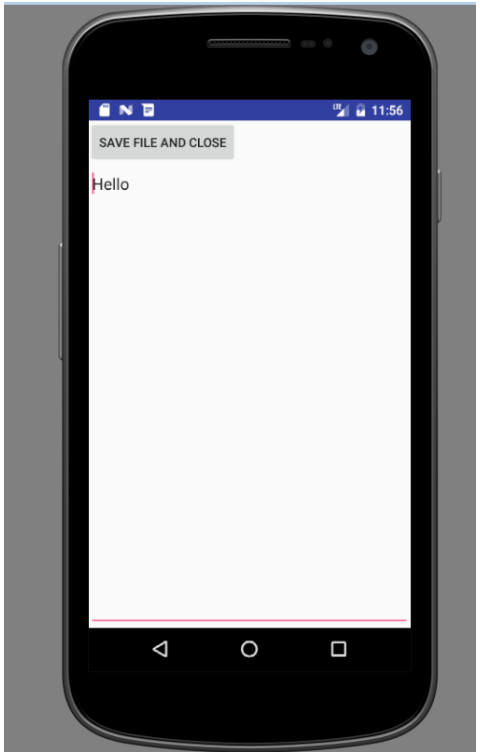
- `FileInputStream openFileInput (String name)`

```
FileInputStream fIn = openFileInput("textfile.txt");
```

Exercise

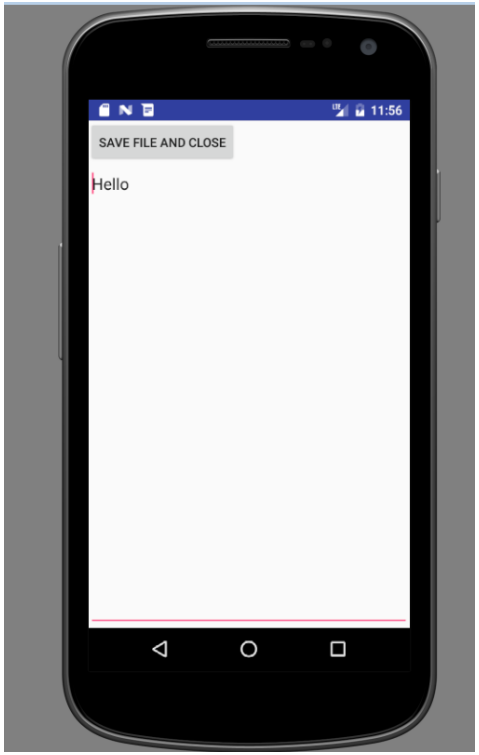
- In this example an application collects data from the UI and saves it to a persistent data file into the (limited) internal Android System space area.

- Next time the application is executed the *Resource File* will be read and its data shown on the UI



Exercise

- Grab from screen, save to file, retrieve from file.
  - OnPause → save text, OnResume → restore text




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <Button android:id="@+id/close"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Save File and Close" />
    <EditText
        android:id="@+id/txtUIData"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="top"
        />
</LinearLayout>
```

Exercise

- Java code

```
package org.androidtown.listview.filesystem;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

public class MainActivity extends Activity {
    private final static String NOTES = "notes.txt";
    private EditText txtUIData;

    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.activity_main);
        txtUIData = (EditText) findViewById(R.id.txtUIData);
        Button btn = (Button) findViewById(R.id.close);

        btn.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                finish();
            }
        });
    }
```

Exercise

```
@Override
    public void onPause() {
        super.onPause();
        try {
            OutputStreamWriter out = new OutputStreamWriter(openFileOutput(NOTES, 0));
            out.write(txtUIData.getText().toString());
            out.close();
        } catch (Throwable t) {
            Toast.makeText(this, "Exception: " + t.toString(), Toast.LENGTH_SHORT).show();
        }
    }

    @Override
    public void onResume() {
        super.onResume();
        try {
            InputStream in = openFileInput(NOTES);
            if (in != null) {
                BufferedReader reader = new BufferedReader(new InputStreamReader(in));
                String str = "";
                StringBuffer buf = new StringBuffer();

                while ((str = reader.readLine()) != null) {
                    buf.append(str + "\n");
                }
                in.close();
                txtUIData.setText(buf.toString());
            }//if
        } catch (java.io.FileNotFoundException e) {
        } catch (Throwable t) {
            Toast.makeText(this, "Exception: " + t.toString(), Toast.LENGTH_SHORT).show();
        }
    }
}
```

문자단위 출력 스트림 (보조출력스트림)

Use 0 or MODE_PRIVATE for the default operation

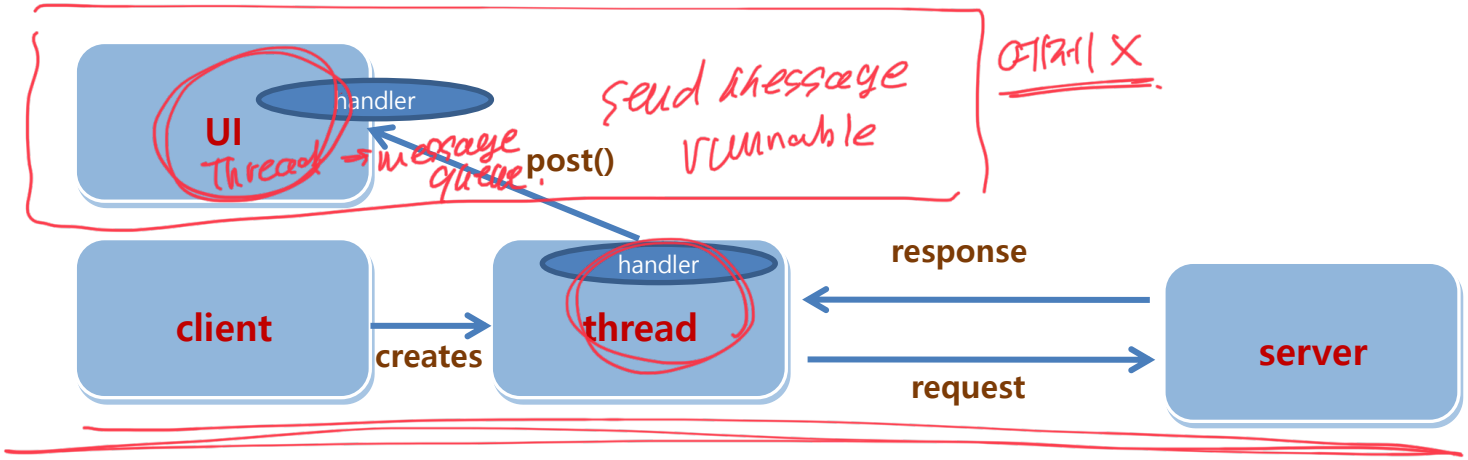
문자단위 입력 스트림

버퍼(라인별) 읽음.

Using client-server networking model in Android

- Have to Manipulate threads for networking support in Android

- UI updates via Handler (We already learned this in thread!)
  - post() method is recommended as simple approach for updating UI



Using client-server networking model in Android

- Making a client socket

```
Socket sock = new Socket("localhost", portNumber);
```

- Making a server socket

```
ServerSocket aServerSocket = new ServerSocket(portNumber);
```

Example

- Java example
  - Server side

```
[11:14:59]서버가 준비되었습니다.
[11:14:59]연결요청을 기다립니다.
[11:18:02]/127.0.0.1로부터 연결요청이 들어왔습니다.
[11:18:02]데이터를 전송했습니다.
```

출처: http://adgw.tistory.com/entry/JAVA-TCP소켓-프로그래밍의-이해-및-채팅프로그램-예제 [앗뜨거워]


```
1
2
3  import java.io.DataOutputStream;
10
11 public class JavaSocketServer {
12     public static void main(String args[]) {
13         ServerSocket serverSocket = null;
14         try {
15             // 서버소켓을 생성하여 7777번 포트와 결합(bind)시킨다.
16             serverSocket = new ServerSocket(7777);
17             System.out.println(getTime() + "서버가 준비되었습니다.");
18         } catch (IOException e) {
19             e.printStackTrace();
20         }
21         while (true) {
22             try {
23                 System.out.println(getTime() + "연결요청을 기다립니다.");
24                 // 서버소켓은 클라이언트의 연결요청이 올 때까지
25                 // 실행을 멈추고 계속 기다린다.
26                 // 클라이언트의 연결요청이 오면 클라이언트 소켓과 통신할
27                 // 새로운 소켓을 생성한다.
28                 Socket socket = serverSocket.accept();
29                 System.out.println(getTime() + socket.getInetAddress()
30                         + "로부터 연결요청이 들어왔습니다.");
31                 // 소켓의 출력스트림을 얻는다.
32                 OutputStream out = socket.getOutputStream();
33                 DataOutputStream dos = new DataOutputStream(out);
34                 // 원격 소켓(remote socket)에 데이터를 보낸다.
35                 dos.writeUTF("[Notice] Test Message1 from Server.");
36                 System.out.println(getTime() + "데이터를 전송했습니다.");
37                 // 스트림과 소켓을 닫아준다.
38                 dos.close();
39                 socket.close();
40             } catch (IOException e) {
41                 e.printStackTrace();
42             }
43         } // while
44     } // main
45     // 현재시간을 문자열로 반환하는 함수
46     static String getTime() {
47         SimpleDateFormat f = new SimpleDateFormat("[hh:mm:ss]");
48         return f.format(new Date());
49     }
50 }
51
```

Example

- Java example
  - Client side

```
서버에 연결중입니다. 서버IP :localhost
서버로부터 받은 메세지 :[Notice] Test
Message1 from Server.
연결을 종료합니다.
연결이 종료되었습니다.
```

```
1
2
3  import java.io.DataInputStream;
8
9  public class JavaSocketClient {
10     public static void main(String args[]) {
11         try {
12             String serverIp = "localhost";
13             System.out.println("서버에 연결중입니다. 서버IP :" + serverIp);
14             // 소켓을 생성하여 연결을 요청한다.
15             Socket socket = new Socket(serverIp, 7777);
16             // 소켓의 입력스트림을 얻는다.
17             InputStream in = socket.getInputStream();
18             DataInputStream dis = new DataInputStream(in);
19             // 소켓으로 부터 받은 데이터를 출력한다.
20             System.out.println("서버로부터 받은 메세지 :" + dis.readUTF());
21             System.out.println("연결을 종료합니다.");
22             // 스트림과 소켓을 닫는다.
23             dis.close();
24             socket.close();
25             System.out.println("연결이 종료되었습니다.");
26         } catch (ConnectException ce) {
27             ce.printStackTrace();
28         } catch (IOException ie) {
29             ie.printStackTrace();
30         } catch (Exception e) {
31             e.printStackTrace();
32         }
33     } // main
34 }
35
```

출처: http://adgw.tistory.com/entry/JAVA-TCP소켓-프로그래밍의-이해-및-채팅프로그램-예제 [앗뜨거워]

Exercise


```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView01"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:text="버튼을 눌르면 소켓 연결 됩니다. 메시지는 로그를
확인하세요" />

    <Button
        android:id="@+id/button01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="소켓 연결하기"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/textView01" />

    <EditText
        android:id="@+id/editText01"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/button01"
        android:text="10.0.2.2"/>

</RelativeLayout>
```

localhost ⇒ 안됨!

↳ 에뮬레이터 네트워크 서버 주소

Exercise

- Android example
  - Client side (cont.)

```
public void onCreate(Bundle savedInstanceState)
  ... 중략 ...

  button01.setOnClickListener(new OnClickListener()
    public void onClick(View v)
      String addr = input01.getText().toString().trim();

          ConnectThread thread = new ConnectThread(addr);
          thread.start();

  );


class ConnectThread extends Thread
  String hostname;

  public ConnectThread(String addr)
    hostname = addr;
```

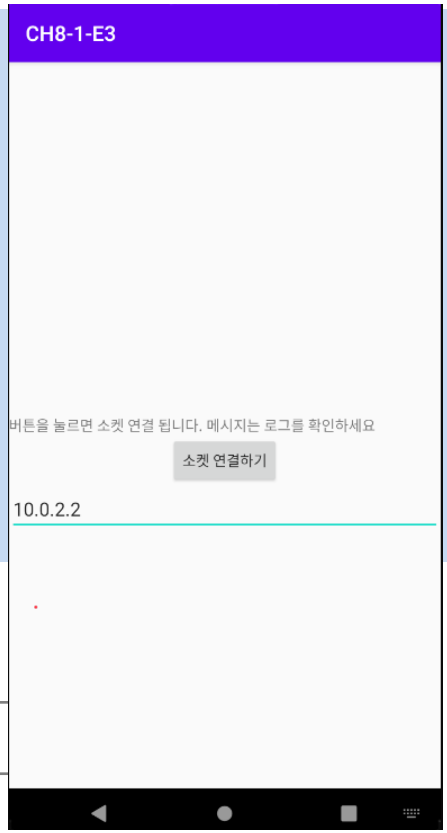
Exercise

- Android example
  - Client side (cont.)

Java와 동일!

```
public void run()
    try  {
          int port = 7777;

          Socket sock = new Socket(hostname, port);

          InputStream in = sock.getInputStream();
          DataInputStream dis = new DataInputStream(in);
          Log.d("MainActivity", "서버에서 받은 메시지 : " + dis.readUTF());
          dis.close();
          sock.close();

    } catch(Exception ex) {
          ex.printStackTrace();
    }
```

UI를 수정 ⇒ AsyncTask or handler




```
<uses-permission android:name="android.permission.INTERNET" />
```

Managing network connectivity

- There are APIs for controlling network settings and connections

- **`ConnectivityManager`** (a network connectivity service)
  - Monitor the state of network connections
  - Configure failover settings
  - Control network radios

- **`NetworkInfo`**
  - Describe the status of a network interface

Managing network connectivity

- Accessing connectivity manager

```
ConnectivityManager connMgr = (ConnectivityManager)
        getSystemService(Context.CONNECTIVITY_SERVICE);
```

- App needs read/write access permission on network states

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>

<uses-permission android:name="android.permission.INTERNET" />
```

↳ 네트워크 상태 확인 시 필요한 권한!

Accessing network status

- Accessing NetworkInfo

```
private static final String DEBUG_TAG = "NetworkStatusExample";
...
ConnectivityManager connMgr = (ConnectivityManager)
        getSystemService(Context.CONNECTIVITY_SERVICE);
// NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
NetworkInfo networkInfo =
                    connMgr.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
boolean isWifiConn = networkInfo.isConnected();

networkInfo = connMgr.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);
boolean isMobileConn = networkInfo.isConnected();

Log.d(DEBUG_TAG, "Wifi connected: " + isWifiConn);
Log.d(DEBUG_TAG, "Mobile connected: " + isMobileConn);
```

Preferred networks

- Preferred network configuration:
  - getNetworkPreference()
  - setNetworkPreference()

```
int networkPreference = connMgr.getNetworkPreference();
connMgr.setNetworkPreference(NetworkPreference.PREFER_WIFI);
```

- If the preferred connection is unavailable, or connectivity on this network is lost, Android will automatically attempt to connect to the secondary network.
- Control availability of the network types using the setRadio() method.

```
connMgr.setRadio(NetworkType.WIFI, false);
connMgr.setRadio(NetworkType.MOBILE, true);
```