Software Practice 3

- Today’s lecture
- Today’s Task
- Porting Android App. in real device

Prof. Hwansoo Han
T.A. Jeonghwan Park 43
BASED ON IMPLEMENTING AN APPLICATION

LAYOUT
View & Layout

- **Layout** declares UI elements in XML
- All elements in the **Layout** are built using a hierarchy of **View** and **ViewGroup** objects
  - A **View** usually draws something the user can see and interact with
  - A **ViewGroup** is an invisible container that defines the layout structure for View and other **ViewGroup** objects
Inheritance of Layout & View

- **Layout (= ViewGroup)**
  
  ```java
  java.lang.Object
  ↘  android.view.View
  ↘  android.view.ViewGroup
  ```

- **View**
  
  ```java
  java.lang.Object
  ↘  android.view.View
  ```
XML of Layout

- Make it easier to design UI Layouts and the screen elements they contain
- Each Layout must contain exactly one root element
  - Root element must be a View or ViewGroup

- More information of Layout; [link](#)
Attributes of Layout

- ID
  - Any View object may have an integer ID
  - android:id="@+id/my_button"
    - @ symbol at the beginning of the string indicates that the XML parser should parse and expand the rest of ID string and identify it as an ID resource
    - + symbol means that this is a new resource name that must be created and added to our resources (in R.java file)
  - Each View object can be accessed with ID
    - Button myButton = (Button) findViewById(R.id.my_button);
Attributes of Layout

- **Layout Parameters**
  - Help defining other attributes with relative values
    - `wrap_content` tells your view to size itself to the dimensions required by its content
    - `match_parent` tells your view to become as big as its parent view group will allow

- **Size, Padding and Margin**
  - `android:layout_height="30dp"
  - `android:layout_width="wrap_content"
  - `android:layout_height="wrap_content"

**"dp" stands for density-independent pixel unit**
Attributes of Layout

- All attributes of Layout are also able to be defined in Activity, except predefined attributes such as ID
- Must be initialized in XML file
Common Layouts

- Each subclass of **ViewGroup** class provides a unique way to display the views

**Linear Layout**
A layout that organizes its children into a single horizontal or vertical row. It creates a scrollbar if the length of the window exceeds the length of the screen.

**Relative Layout**
Enables you to specify the location of child objects relative to each other (child A to the left of child B) or to the parent (aligned to the top of the parent).

**Web View**
Displays web pages.
## Find View in an Activity

### public methods of View class

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>final &lt;T extends View&gt; T findViewById(int id)</code></td>
<td>Finds the first descendant view with the given ID, the view itself if the ID matches <code>getId()</code>, or null if the ID is invalid (&lt; 0) or there is no matching view in the hierarchy.</td>
</tr>
<tr>
<td><code>final &lt;T extends View&gt; T findViewByIdWithTag(Object tag)</code></td>
<td>Look for a child view with the given tag.</td>
</tr>
<tr>
<td><code>void findViewsWithText(ArrayList&lt;View&gt; outViews, CharSequence searched, int flags)</code></td>
<td>Finds the Views that contain given text.</td>
</tr>
</tbody>
</table>
## Find View in an Activity

- **public methods of ViewGroup class**

<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>View</td>
<td><code>getChildAt(int index)</code></td>
<td>Returns the view at the specified position in the group.</td>
</tr>
<tr>
<td>int</td>
<td><code>getChildCount()</code></td>
<td>Returns the number of children in the group.</td>
</tr>
</tbody>
</table>
Events with View

- **onClickListener()**
  - View.OnClickListener
  - the user touches the item

- **onLongClickListener()**
  - View.OnLongClickListener
  - the user touches and holds the item

- **onFocusChangeListener()**
  - View.OnFocusChangeListener
  - the user navigates onto or away from the item
Events with View

- **onKey()**
  - View.OnKeyListener
  - the user is focused on the item and presses or releases a hardware key on the device

- **onTouch()**
  - View.OnTouchListener
  - the user performs an action qualified as a touch event, including a press, a release, or any movement gesture on the screen

- **onCreateContextMenu()**
  - View.onCreateContextMenuListener
  - a Context Menu is being built
Event Handling

- An example below shows how to register an on-click listener for a Button in common way

```java
// Create an anonymous implementation of OnClickListener
private OnClickListener mCorkyListener = new OnClickListener() {
    public void onClick(View v) {
        // do something when the button is clicked
    }
};

protected void onCreate(Bundle savedValues) {
    ...
    // Capture our button from layout
    Button button = (Button) findViewById(R.id.corky);
    // Register the onClick listener with the implementation above
    button.setOnClickListener(mCorkyListener);
    ...
}
```
Event Handling

- An example below is used when most events in an Activity performs the same things

```java
public class ExampleActivity extends Activity implements OnClickListener {
    protected void onCreate(Bundle savedInstanceState) {
        ...
        Button button = (Button) findViewById(R.id.corky);
        button.setOnClickListener(this);
    }

    // Implement the OnClickListener callback
    public void onClick(View v) {
        // do something when the button is clicked
    }
    ...
}
```
IN ANDROID STUDIO

GENERATE IMAGE
IN ANDROID STUDIO

EXPORT APK
Implement your application satisfying the following conditions

- You have to use relative layout
- Make two buttons; one for changing text, other for changing image
- text should be changed like below
  - id -> name -> name, id -> id -> …
- image should be changed like below
  - android -> android with coffee -> android in mobile phone -> android -> …
[Lab – Practice #2]

- Design of the application
example source code of relative layout

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp">
    <EditText
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder"
    />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```