Andorid Programming

Week 2

Mina Jung

EECS, Syracuse University

Spring 2017

Part I

Activity, User Interface - Layout, UI Controls, Styles and Themes

Outline

Main Components I

- Activities
  - dictate UI and handle user interaction
  - one activity represents a single screen with one user interface (layout), and performs actions on the screen
- Services
  - handle background processing
  - a service runs in the background to perform long-running operations without blocking user interaction with an activity
- Broadcast Receivers
  - handle communication between Android OS and applications

Main Components II

  - simply respond to broadcast messages from other applications or from the system

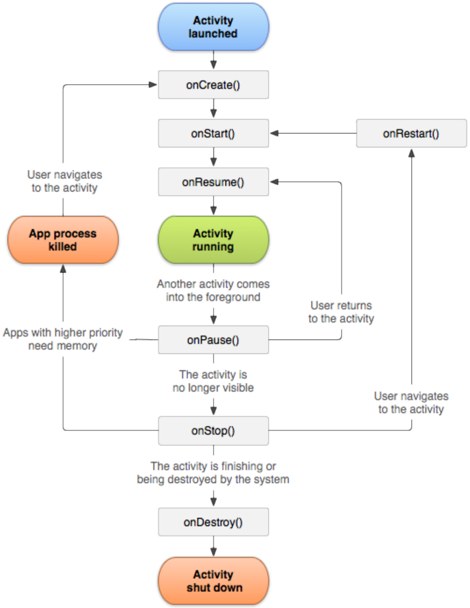
- Content Providers
  - handle data and database management issues

- Additional Components
  - Fragments
    - represent a portion of UI in an Activity (Discuss later)
  - Views
    - UI elements on screen
  - Layouts

Main Components III

    - control screen format and appearance of the views

- Intents
    - Messages wiring components together
- Resources
- Manifest
    - configuration file

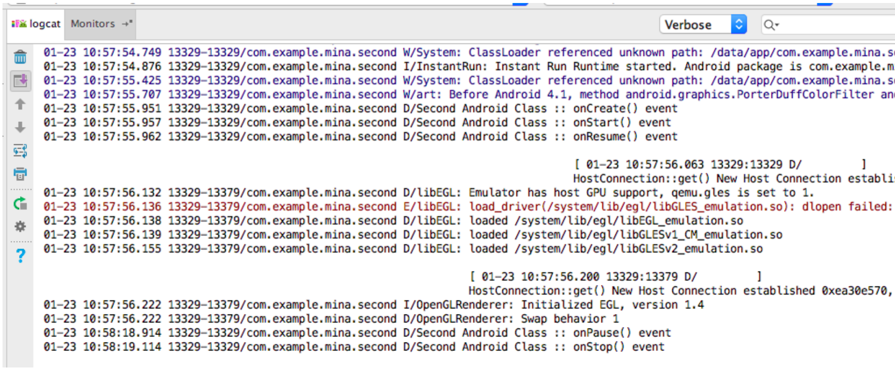

Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

```
package com.example.mina.second;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;

public class MainActivity extends AppCompatActivity {
    String msg = "Second Android Class : ";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d(msg, "onCreate() event");
    }

    @Override
    protected void onStart() {
        super.onStart();
        Log.d(msg, "onStart() event");
    }

    @Override
    protected void onResume(){
        super.onResume();
        Log.d(msg, "onResume() event");
    }

    @Override
    protected void onPause(){
        super.onPause();
        Log.d(msg, "onPause() event");
    }

    @Override
    protected void onStop(){
        super.onStop();
        Log.d(msg, "onStop() event");
    }

    @Override
    protected void onDestroy(){
        super.onDestroy();
        Log.d(msg, "onDestroy() event");
    }
}
```

MainActivity.java

- includes each of fundamental life cycle methods
- loads UI components from *res/layout/activity_main.xml* file
- Log.d() method is used to generate log messages displayed on LogCat window in Android Studio
- Click Log Class

Write and View Logs with LogCat



```
01-23 10:57:54.749 13329-13329/com.example.mina.second W/System: ClassLoader referenced unknown path: /data/app/com.example.mina.s
01-23 10:57:54.876 13329-13329/com.example.mina.second I/InstantRun: Instant Run Runtime started. Android package is com.example.m
01-23 10:57:55.425 13329-13329/com.example.mina.second W/System: ClassLoader referenced unknown path: /data/app/com.example.mina.s
01-23 10:57:55.707 13329-13329/com.example.mina.second W/art: Before Android 4.1, method android.graphics.PorterDuffColorFilter an
01-23 10:57:55.951 13329-13329/com.example.mina.second D/Second Android Class :: onCreate() event
01-23 10:57:55.957 13329-13329/com.example.mina.second D/Second Android Class :: onStart() event
01-23 10:57:55.962 13329-13329/com.example.mina.second D/Second Android Class :: onResume() event

                                                                           [ 01-23 10:57:56.063 13329:13329 D/         ]
                                                                           HostConnection::get() New Host Connection establi
01-23 10:57:56.132 13329-13379/com.example.mina.second D/libEGL: Emulator has host GPU support, qemu.gles is set to 1.
01-23 10:57:56.136 13329-13379/com.example.mina.second E/libEGL: load_driver(/system/lib/egl/libGLES_emulation.so): dlopen failed:
01-23 10:57:56.138 13329-13379/com.example.mina.second D/libEGL: loaded /system/lib/egl/libEGL_emulation.so
01-23 10:57:56.139 13329-13379/com.example.mina.second D/libEGL: loaded /system/lib/egl/libGLESv1_CM_emulation.so
01-23 10:57:56.155 13329-13379/com.example.mina.second D/libEGL: loaded /system/lib/egl/libGLESv2_emulation.so

                                                            [ 01-23 10:57:56.200 13329:13379 D/         ]
                                                            HostConnection::get() New Host Connection established 0xea30e570,
01-23 10:57:56.222 13329-13379/com.example.mina.second I/OpenGLRenderer: Initialized EGL, version 1.4
01-23 10:57:56.222 13329-13379/com.example.mina.second D/OpenGLRenderer: Swap behavior 1
01-23 10:58:18.914 13329-13329/com.example.mina.second D/Second Android Class :: onPause() event
01-23 10:58:19.114 13329-13329/com.example.mina.second D/Second Android Class :: onStop() event
```

- An application can have one or more activities
- Each activity must be declared in *AndrioidManifest.xml*
- Main activity for the app must be declared with $< intent - filter >$ including both **MAIN** action and **LAUNCHER** category
  - if not correctly declared, app icon will not appear

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.mina.second">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

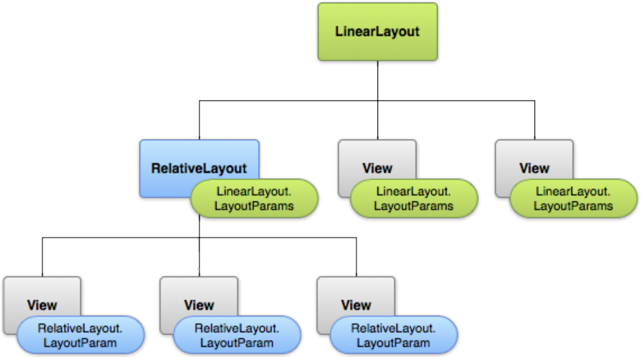
AndroidManifest.xml

Layouts I

- Define the visual structure for UI
  - a View object (Click for View Class)
    - basic building block
    - occupies a rectangular area on the screen
    - responsible for drawing and event handling
  - ViewGroups (Click for ViewGroup Class)
    - subclass of View
    - invisible container holding other Views or ViewGroups
    - define layout properties
  - subclass of ViewGroup

Layouts II

- View hierarchy with layout parameters associated with each view


LinearLayout
RelativeLayout
LinearLayout. LayoutParams
View
LinearLayout. LayoutParams
View
LinearLayout. LayoutParams
View
RelativeLayout. LayoutParam
View
RelativeLayout. LayoutParam
View
RelativeLayout. LayoutParam

Layouts III

- Declared in two ways
  - declare UI elements in XML file
  - instantiate layout elements at runtime (programmatically)
  - use either or both of the above methods for declaring and managing UI

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <TextView android:id="@+id/text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="This is a TextView" />

   <Button android:id="@+id/button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="This is a Button" />

```

Layouts IV

```
    <!-- More GUI components go here  -->

</LinearLayout>
```

- Once a layout has created, load the layout resource in Activity.onCreate() callback

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
```

View Identification I

- ID
  - uniquely identify the View
  - syntax for a unique ID of a view in XML layout file
    `android:id=''@+id/my_button''`
    - @ (at-symbol)
    - + (plus-symbol) : a new resource is created and added

- Create Views and Reference them
  1. Define a view/widget in the layout file and assign a unique ID

```
<Button android:id="@+id/my_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/my_button_text"/>
```

View Identification II

2. Create an instance of the view object and capture it from the layout

```
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);

   Button myButton = (Button) findViewById(R.id.my_button);
}
```

Layout Attributes I

No.	Attribute	Description
1	android:id	ID uniquely identifies the view
2	android:layout_width	width of the layout
3	android:layout_height	height of the layout
4	android:layout_marginTop	extra space on the top side of the layout
5	android:layout_marginBottom	extra space on the bottom side of the layout
6	android:layout_marginLeft	extra space on the left side of the layout
7	android:layout_marginRight	extra space on the right side of the layout
8	android:layout_x	x-coordinate of the layout
9	android:layout_y	y-coordinate of the layout
10	android:layout_gravity	how child Views are positioned
11	android:layout_weight	how much of the extra space in the layout should be allocated to the View
12	android:paddingLeftt	left padding filled for the layout
13	android:paddingRight	right padding filled for the layout

Layout Attributes II

14	android:paddingTop	top padding filled for the layout
15	android:paddingBottom	bottom padding filled for the layout

- Click Layout Paramters
- Click Layout Resource
- Click Gravity

Unit of Measurement	
dp	Density-independent Pixels
sp	Scale-independent Pixels
pt	Points (1/72 of an inch)
in	Inches

No.	Layout	Description
1	LinearLayout	viewgroup that aligns all children in a single direction, vertically or horizontally
2	RelativeLayout	viewgroup that displays child views in relative positions
3	TableLayout	view that groups views into rows and columns
4	AbsoluteLayout	specify the exact location of its children
5	FrameLayout	placeholder on the screen to be used to display a single view
6	ListView	viewgroup that displays a list of scrollable items
7	GridView	viewgroup that displays items in a 2-D, scrollable grid

1. LinearLayout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />
</LinearLayout>
```

2. RelativeLayout

```
<RelativeLayout
    android:id="@+id/RLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android" >
</RelativeLayout>
```

3. TableLayout

```
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent" >

    <TableRow>
        <TextView
            android:text="User Name:"
            android:width ="120dp"
        />

        <EditText
            android:id="@+id/txtUserName"
            android:width="200dp" />
    </TableRow>

</TableLayout>
```

4. AbsoluteLayout

```
<AbsoluteLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <Button
        android:layout_width="188dp"
        android:layout_height="wrap_content"
        android:text="Button"
        android:layout_x="126px"
        android:layout_y="361px" />
</AbsoluteLayout>
```

5. FrameLayout

```
<FrameLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/lblComments"
    android:layout_below="@+id/lblComments"
    android:layout_centerHorizontal="true" >

    <ImageView
        android:src = "@drawable/droid"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</FrameLayout>
```

No.	UI Control	Description
1	TextView	display text to the user
2	EditText	predefined subclass of TextView with rich editing capabilities
3	AutoCompleteTextView	similar to EditText, with a list of completion suggestions automatically
4	Button	pusb-button clicked by the user to perform an action
5	ImageButton	button with an image, an AbsoluteLayout
6	CheckBox	on/off switch toggled by the user, group of slectable options
7	ToggleButton	on/off button with a light indicator
8	RadioButton	with two states: checked or unchecked
9	RadioGroup	used to group together one or more RadioButtons
10	ProgressBar	provides visual feedback about ongoing tasks
11	Spinner	drop-down list

12	TimePicker	
13	DatePicker	
14	ImageView	

- Create UI control in layout XML and Instantiate the Control object

from the layout programmatically

- Event Listeners
  - interface in the View class containg a single callback method
  - called by Android Framework triggered by user interaction
- Event Listeners Registration
  - Event Handler gets registered with an Event Listener
  - handler is called when the event listener fires the event
- EventHandlers
  - actually handle the event

Event Handler	Event Listener Interface	Description
onClick()	OnClickListener()	called when the user either clicks or touches or focuses upon any widget like button, text, image etc.
onLongClick()	OnLongClickListener()	called when the user either clicks or touches or focuses upon any widget like button, text, image etc. for one or more seconds
onFocusChange()	onFocusChangeListener()	called when the widget looses its focus ie. user goes away from the view item
onKey()	OnKeyListener()	called when the user is focused on the item and presses or releases a hardware key on the device
onTouch()	OnTouchListener()	called when the user presses the key, releases the key, or any movement gesture on the screen
onMenuItemClick()	OnMenuItemClickListener()	called when the user selects a menu item
onCreateContextMenu()	OnCreateContextMenuListener()	called when the context menu is being built

Event Registration I

- An Event Handler gets registered with an Event Listener so that the handler is called when the Event Listener fires the event

1. Using an Anonymous Inner Class

```java
// Create an anonymous implementation of OnClickListener
private OnClickListener myButtonListener = 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.button);
    // Register the onClick listener with the implementation above
    button.setOnClickListener(myButtonListener);
    ...
}
```

Event Registration II

2. Activity class implements the Listener interface

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

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

Event Registration III

3. Using Layout file (such as activity_main.xml) to specify event handler directly

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal">

    <Button android:id="@+id/button_send"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send"
        android:onClick="sendMessage" />
</LinearLayout>
```

```
/** Called when the user touches the button */
public void sendMessage(View view) {
    // Do something in response to button click
    ...
}
```

- Style is a collection of properties that specify the look and format for a View
  - specify properties such as height, padding, font color, font size, background color, and etc.

- Define Styles in res/values/styles.xml

```
<resources>
    <style name="CodeFont" parent="@android:style/TextAppearance.Medium">
        <item name="android:layout_width">fill_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#00FF00</item>
        <item name="android:typeface">monospace</item>
    </style>
</resources>
```

  - Applied to a View element in Layout XML

```
<TextView
    style="@style/CodeFont"
    android:text="@string/hello" />
```

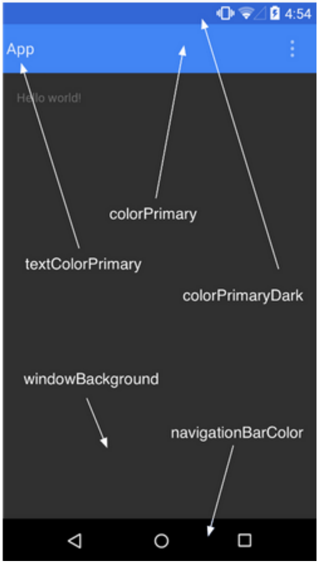
- Inheritance
  - **parent** attribute to specify a style to inherit its properties

```
<style name="GreenText" parent="@android:style/TextAppearance">
        <item name="android:textColor">#00FF00</item>
</style>
```

  - prefix the name of the style you want to inherit to the name of your new style, separated by a period

```
<style name="CodeFont.Red">
        <item name="android:textColor">#FF0000</item>
</style>
```

- Style properties are defined by <item> element
  - View attributes such as "android:textColor" can be defined by <item> elements of a new style



- Theme is a style applied to an entire Activity or application

```
<color name="custom_theme_color">#b0b0ff</color>
<style name="CustomTheme"
    parent="android:Theme.Light">
  <item
      name="android:windowBackground">@color/custom_th
  <item
      name="android:colorBackground">@color/custom_the
</style>
```

- AndroidManifest.xml file
  - application

    `<application android:theme="@style/CustomTheme">`
  - Activity

    `<activity android:theme="@style/CustomTheme">`
- Click Material Theme Info
  Click Color Palette

Part II

Examples

Outline I

- native action bar doesn't support material design
- Should use **Toolbar** class to implement activities' app bars

1. Make sure the activity extends AppCompatActivity

```
public class MainActivity extends AppCompatActivity {
    // .....
}
```

2. Set the <application> element of manifest to use one of appcompat's NoActionBar themes

```
<application
    android:theme="@style/Theme.AppCompat.Light.NoActionBar"
    />
```

3. Add a Toolbar to the activity's layout

```
<android.support.v7.widget.Toolbar
   android:id="@+id/my_toolbar"
   android:layout_width="match_parent"
   android:layout_height="?attr/actionBarSize"
   android:background="?attr/colorPrimary"
   android:elevation="4dp"
   android:theme="@style/ThemeOverlay.AppCompat.ActionBar"
   app:popupTheme="@style/ThemeOverlay.AppCompat.Light"/>
```

4. In the activity's onCreate() method, call the activity's setSupportActionBar() method, and pass the activity's toolbar

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_my);
    Toolbar myToolbar = (Toolbar) findViewById(R.id.my_toolbar);
    setSupportActionBar(myToolbar);
}
```

5. Add actions (menu items in /res/menu/xxx_menu.xml)

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item android:id="@+id/action_one"
        android:orderInCategory="100"
        android:title="@string/action_one"
        app:showAsAction="never"/>

    <item android:id="@+id/action_two"
        android:orderInCategory="100"
        android:title="@string/action_two"
        app:showAsAction="never"/>

    <item .....  />
</menu>
```

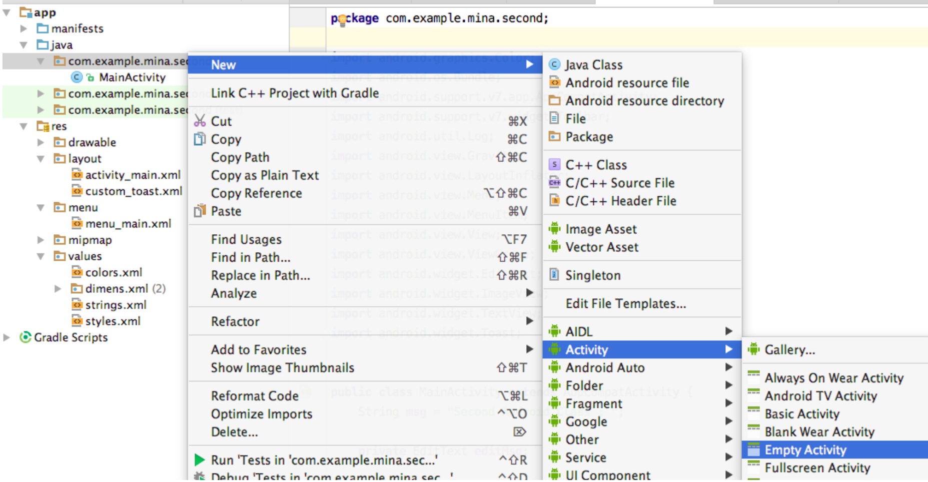
6. Handling actions

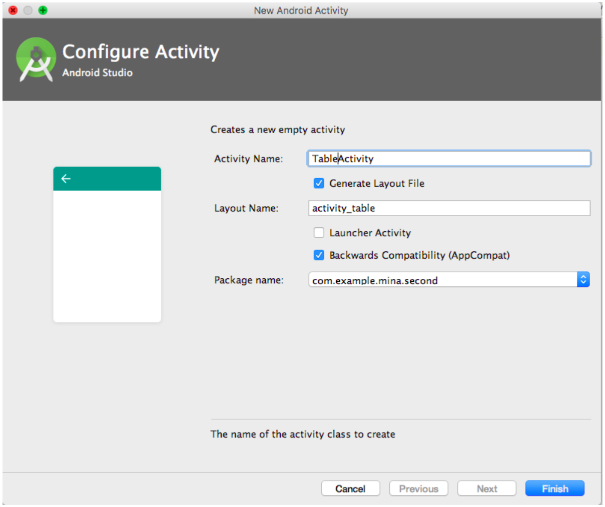
```
@Override
public boolean onCreateOptionsMenu(Menu menu){
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_one:

            return true;

        case R.id.action_two:

            return true;

        default:
            // If we got here, the user's action was not recognized.
            // Invoke the superclass to handle it.
            return super.onOptionsItemSelected(item);
    }
}
```

- Toast message

```
//display in short period of time
Toast.makeText(getApplicationContext(), "short TOAST msg",
    Toast.LENGTH_SHORT).show();

//display in long period of time
Toast.makeText(getApplicationContext(), "long TOAST msg",
    Toast.LENGTH_LONG).show();
```

- Create a New Activity and its Layout




New Android Activity

Configure Activity

Android Studio

Creates a new empty activity

Activity Name: TableActivity

[✓] Generate Layout File

Layout Name: activity_table

[ ] Launcher Activity

[✓] Backwards Compatibility (AppCompat)

Package name: com.example.mina.second

The name of the activity class to create

- NewActivity.java and activity_new.xml files (automatically created)

- Start a New Activity from Main Activity when a button (or an menu) is clicked and the event is handled

```java
Intent intent = new Intent(MainActivity.this,
    NewActivity.class);

startActivity(intent);
```

- Custom Toast Layout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/custom_toast_layout_id"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#FFF"
    android:orientation="horizontal"
    android:padding="5dp" >

    <ImageView
        android:id="@+id/toast_image"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:layout_marginRight="5dp" />

    <TextView
        android:id="@+id/toast_text"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:textColor="#000" />
</LinearLayout>
```

- Display

```
//Toast Message 2 with Custom Toast Layout
// get your custom_toast.xml Layout
LayoutInflater inflater = getLayoutInflater();

View layout = inflater.inflate(R.layout.custom_toast,
     (ViewGroup) findViewById(R.id.custom_toast_layout_id));

// set a dummy image
ImageView image = (ImageView)
     layout.findViewById(R.id.toast_image);
image.setImageResource(R.drawable.toast);

 // set a message
 TextView text = (TextView)
      layout.findViewById(R.id.toast_text);
  text.setText(R.string.toast_msg);

 // Toast...
 Toast toast = new Toast(getApplicationContext());
  toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
  toast.setDuration(Toast.LENGTH_LONG);
  toast.setView(layout);
  toast.show();
```

```java
LinearLayout layout =
     (LinearLayout)findViewById(R.id.scrollVertical);
ImageView img = new ImageView(this);
img.setImageResource(R.drawable.wise);
LinearLayout.LayoutParams lp = new
     LinearLayout.LayoutParams(
     ViewGroup.LayoutParams.WRAP_CONTENT,
     ViewGroup.LayoutParams.WRAP_CONTENT);
lp.setMargins(30,20,30,0);
layout.addView(img, lp);
```

- SnackBar

```
Snackbar.make(v, "This is Snackbar", Snackbar.LENGTH_LONG).show();
// or LENGTH_SHORT
```

- SnackBar with button

```
Snackbar.make(v, "SnackBar Button
     Clicked",Snackbar.LENGTH_INDEFINITE).setAction("Done", new
     View.OnClickListener(){
                    @Override
                    public void onClick(View v){

                    }
}).show();
```

```
<HorizontalScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="10dp"
    android:background="@drawable/round"
    android:scrollbarSize="10dp"
    android:scrollbarStyle="outsideInset"
    android:scrollbarThumbHorizontal="@drawable/scrollbar_bg1"
    android:scrollbarTrackHorizontal="@drawable/scrollbar_bg2"
    android:fadeScrollbars="false">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal">
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Horizontal Scroll (-->)"/>

            <LinearLayout
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:orientation="horizontal">

```

```
                    <ImageView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_margin="10dp"
                        android:src="@drawable/martian"/>
                    <ImageView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_margin="10dp"
                        android:src="@drawable/star"/>
                    <ImageView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_margin="10dp"
                        android:src="@drawable/xmen"/>
                    <ImageView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_margin="10dp"
                        android:src="@drawable/wise"/>
                </LinearLayout>
            </LinearLayout>

        </LinearLayout>

    </HorizontalScrollView>
```

- /res/drawable/ xml files for shape
  - round.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <solid android:color="#99e7e7e7"/>
    <corners android:radius="15dip"/>
</shape>
```

  - scrollbar_bg1.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <gradient android:startColor="#FFE400"
        android:endColor="#ffffff" android:angle="0"/>
    <corners android:radius="10dp"/>
</shape>
```

  - scrollbar_bg2.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <gradient android:startColor="#ff0000"
        android:endColor="#ffffff" android:angle="0"/>
    <corners android:radius="10dp"/>
</shape>
```

```
// for same kind of event, listener registration
        findViewById(R.id.coin1).setOnClickListener(mClickListener);
        ...
        findViewById(R.id.coin12).setOnClickListener(mClickListener);



    // listener declaration
    Button.OnClickListener mClickListener = new
         Button.OnClickListener() {
        public void onClick(View v) {
             switch (v.getId()) {
                case R.id.coin1:
                   break;
                   .....
                case R.id.coin12:
                    break;
            }
        }
    };
```

```
findViewById(R.id.btn_clear).setOnLongClickListener(new
     View.OnLongClickListener() {
    @Override
    public boolean onLongClick(View v) {

        return true;    // return true --  consumed
                             // return false -- not consumed
    }
});
```

```
// Swipe handling
        findViewById(R.id.activity_movie).setOnTouchListener(new
            View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                switch (event.getAction()){
                    case MotionEvent.ACTION_DOWN:
                        downX = event.getX();
                        downY = event.getY();
                        break;
                    case MotionEvent.ACTION_MOVE:
                        double deltaX = downX - event.getX();
                        double deltaY = downY - event.getY();
                        // horizontal swipe detection
                        if (Math.abs(deltaX) > 40) {
                            // left or right
                            }
                            if (deltaX > 0) {
                            // right to left
                            }
                        }
                }
                return true;
            }
        });
```

```
SeekBar sb = (SeekBar) findViewById(R.id.seekBar);
sb.setProgress(50);
sb.setOnSeekBarChangeListener(new
     SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int
         progress, boolean fromUser) {
     // Your code
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {

    }
});
```