Activities in Android

Activities as Application Context and View/GUI Frame

Activity as a Major App Component

Activity

- Activity is an application component that
  - Provides a GUI screen for interaction of user with the app
  - It also implements a **Context** that is a container of global info/services of the application.

- An application may have one or multiple activities
  - One of them is specified as the main activity (the app's launching window)

- The application activities are loosely bound to each other
  - Since they are basically designed as standalone components
  - Remind that android is based on a **Component Oriented** software architecture
  - Each activity can start other activities to do a requested action

Activity Life Cycle

- When a new Activity starts:
  - The previous activity is stopped
  - But android system preserves it in a stack (**Back Stack**)
- When and activity task is done and user presses Back button:
  - It (current activity) is popped from Back Stack and destroyed.
  - The previous activity is resumed from the stopped state.
- When an activity is created, started, stopped (because a new one starts) or resumed:
  - It is notified of this state change through its **life cycle callback methods**
  - Example: onCreate(), onStart(), onPause(), onStop(), onResume(), onDestroy()

Building/Defining an Activity

- To define an activity
  - You must inherit a class from **Activity** or one of **Activity subclasses**.
  - Then you need to implement callback methods that android system will call when the activity transitions between its life cycle states

- The most important callback methods are
  - **onCreate()**
    - Called by system when creating the activity
    - You must initialize the activity contents/components there
    - You call **setContentView()** method to set the View hierarchy which either
      - You have designed in the designer as an XML resource file
      - Or you have defined dynamically in java code

  - **onPause()**
    - Called by system when user is leaving from your activity
    - It does not mean that, the Actvity will be destroyed necessarily.
    - Here is the time that you must save/commit changes/data provided by user.
    - The user may or may not return to this Activity instance again.

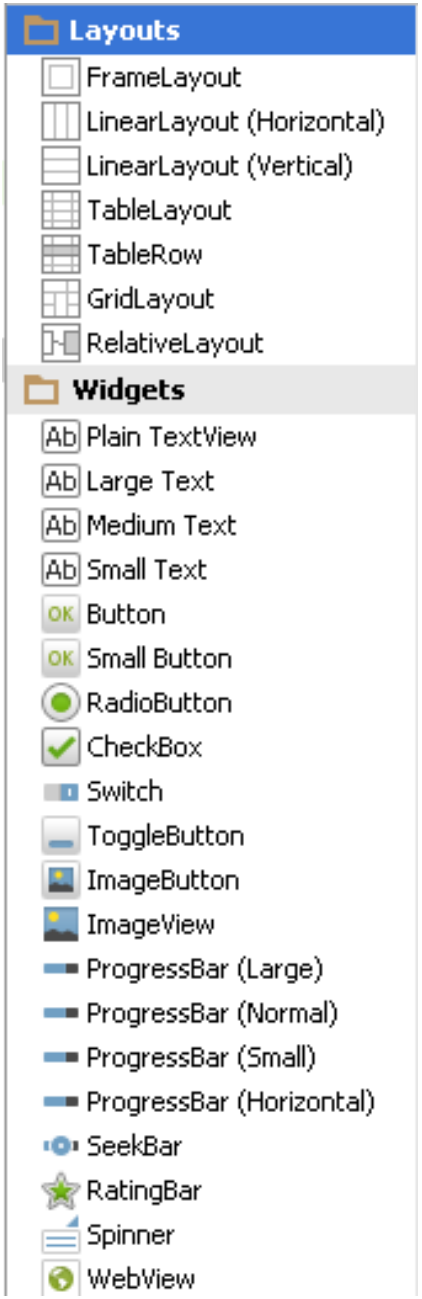
Designing a User Interface

- The UI is a hierarchy of android Views
  - All GUI components in android are derived from **View** calss
  - Each View controls a **rectangular space** within the activity window
  - It also responds to user interactions within its region
  - View examples: Button, TextView, ImageView, EditView, ViewGroup, …

- Widgets
  - Are Views that provide visual interactive input elements
  - Examples: Button, Text, Checkbox, …

- Layouts
  - Are Views that are derived from ViewGroup
  - Can hold multiple child views and provide a layout model to organize them on screen
  - Various built-in layout models exists and you may define new one if you need
  - Examples: RelativeLayout, GridLayout, TableLayout, ScrollLayout…

- You can subclass the View, ViewGroup or their existing views/widgets/layouts to define your modified views.



Designing a User Interface (...)

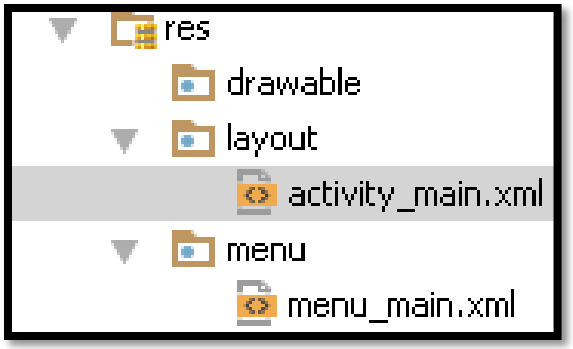
- Methods for specifying a UI layout

  - Writing/designing an XML layout file which is saved in the app resources
    - This is the default and encouraged method
    - Separates the user interface View from its Behaviour
    - Then you can set layout of your Activity via **setActivityContent(int)** by passing the layout file resource id:
      - **setActivityContent(xml_layout_resource_id)**
  - Instantiating View/ViewGroups in Java code and to build a view hierarchy
    - You create Views and arrange them in ViewGroup(s)
    - Then you add the root ViewGroup of the created view hierarchy to the activity via overloaded method **setContentView(ViewGroup v)**

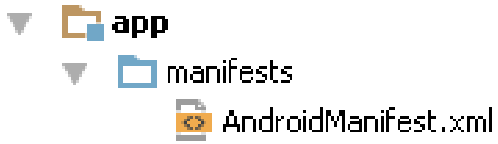
Designing a User Interface (...)

- The XML Layout Resource File

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity">

    <TextView android:text="@string/hello_world" android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="center" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello SHU"
        android:id="@+id/hello_shu_btn"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />

</RelativeLayout>
```



Declaring the activity in the manifest




```
<manifest … >
   <application … >
     <activity android:name=".ExampleActivity" />
     …
   </application … >
   …
</manifest >
```

Declaring the activity in the manifest

Using Intent Filters:

```
<activity android:name=".ExampleActivity“ android:icon="@drawable/app_icon">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Starting an Activity

```
// Intent is used as a message to the Android system/other components
Intent intent = new Intent(this, SignInActivity.class);
startActivity(intent);
```

```
// Passing extra parameters via Intent
Intent intent = new Intent(Intent.ACTION_SEND);
intent.putExtra(Intent.EXTRA_EMAIL, recipientArray);
startActivity(intent);
```

Starting an Activity for a Result

```
private void pickContact() {
    // Create an intent to "pick" a contact, as defined by the content provider URI
    Intent intent = new Intent(Intent.ACTION_PICK, Contacts.CONTENT_URI);
    startActivityForResult(intent, PICK_CONTACT_REQUEST);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // If the request went well (OK) and the request was PICK_CONTACT_REQUEST
    if (resultCode == Activity.RESULT_OK && requestCode == PICK_CONTACT_REQUEST) {
        // Perform a query to the contact's content provider for the contact's name
        Cursor cursor = getContentResolver().query(data.getData(),
        new String[] {Contacts.DISPLAY_NAME}, null, null, null);
        if (cursor.moveToFirst()) { // True if the cursor is not empty
            int columnIndex = cursor.getColumnIndex(Contacts.DISPLAY_NAME);
            String name = cursor.getString(columnIndex);
            // Do something with the selected contact's name...
        }
    }
}
```

Shutting Down an Activity

- In the current Actvivity just call:
  - finish()

- To stop another activity that you have started:
  - finishActivity(int requestCode)

Shutting Down an Activity

- In the current Actvivity just call:
  - finish()

- To stop another activity that you have started:
  - finishActivity(int requestCode)

Managing Activity Lifecycle

- Activity is essentially in one of 3 major states:
  - **Resumed**
  - **Paused**
  - **Stopped**

- When it transits between this and other minor states LifeCycle methods is called.
  - onCreate()
  - onStart()
  - onResume()
  - onPause()
  - onStop()
  - onDestroy()

Activity Life time

- Three nested loops in the Activity LifeCycle:onCreate()
  - Entire Lifetime
  - Visible Lifetime
  - Foreground Lifetime

- Entire Life time of an Activity
  - Happens between **onCreate() – onDestroy()**

Implementing the Lifecycle methods

```
public class ExampleActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // The activity is being created.
    }
    @Override
    protected void onStart() {
        super.onStart();
        // The activity is about to become visible.
    }
    @Override
    protected void onResume() {
        super.onResume();
        // The activity has become visible (it is now "resumed").
    }
    @Override
    protected void onPause() {
        super.onPause();
        // Another activity is taking focus (this activity is about to be "paused").
    }
```

Implementing the Lifecycle methods(...)

```
    @Override
    protected void onStop() {
        super.onStop();
        // The activity is no longer visible (it is now "stopped")
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        // The activity is about to be destroyed.
    }
}
```