Android Lecture 1: Android Basics

Agenda

- Getting Started
  - Intro to the Android platform
  - Android programming in Eclipse
  - Anatomy of a project
- Basic Android Components
  - Layouts
  - Views and Widgets
  - Menus

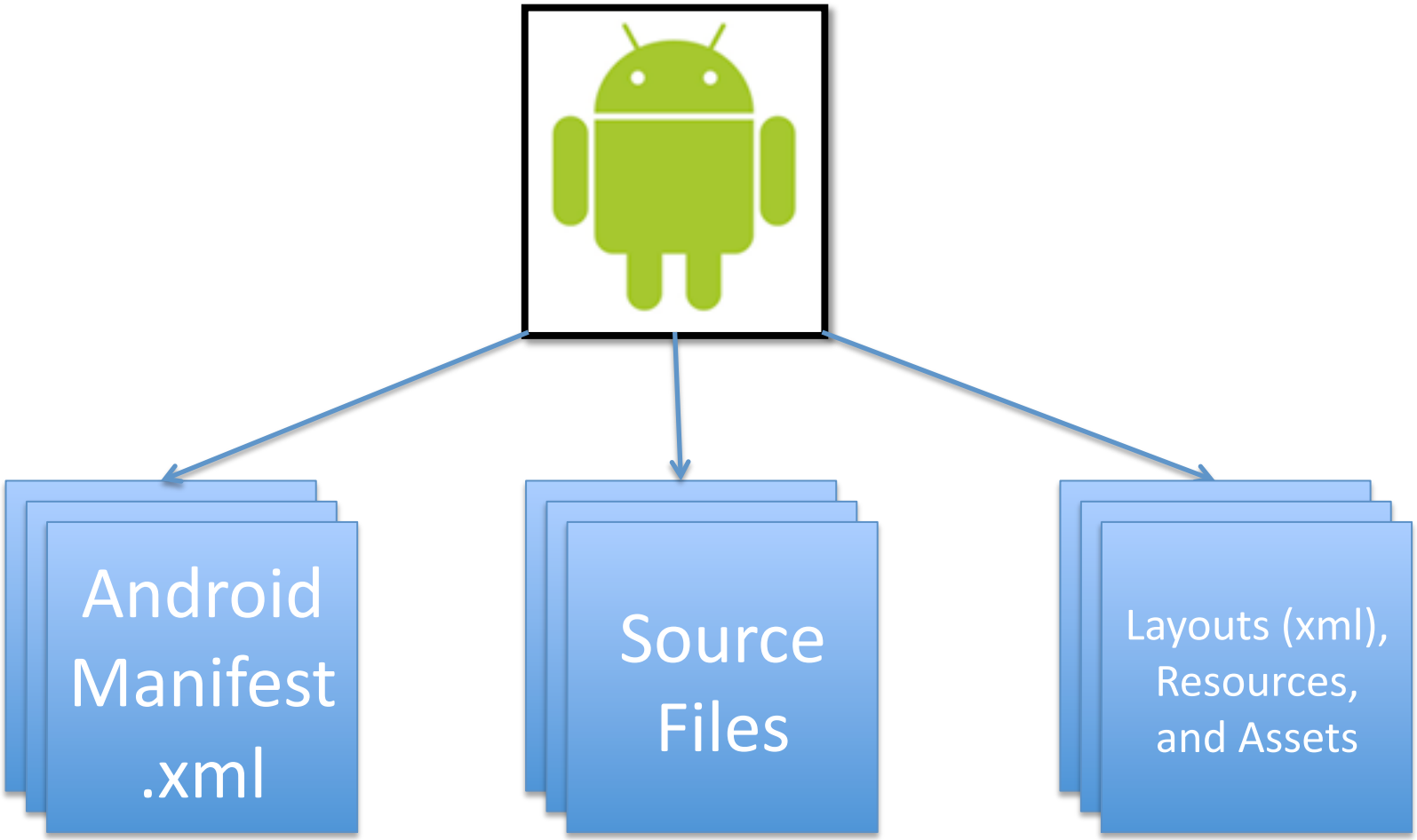
Agenda

- Getting Started
  - Intro to the Android platform
  - Android programming in Eclipse
  - Anatomy of a project
- Basic Android Components
  - Layouts
  - Views and Widgets
  - Menus

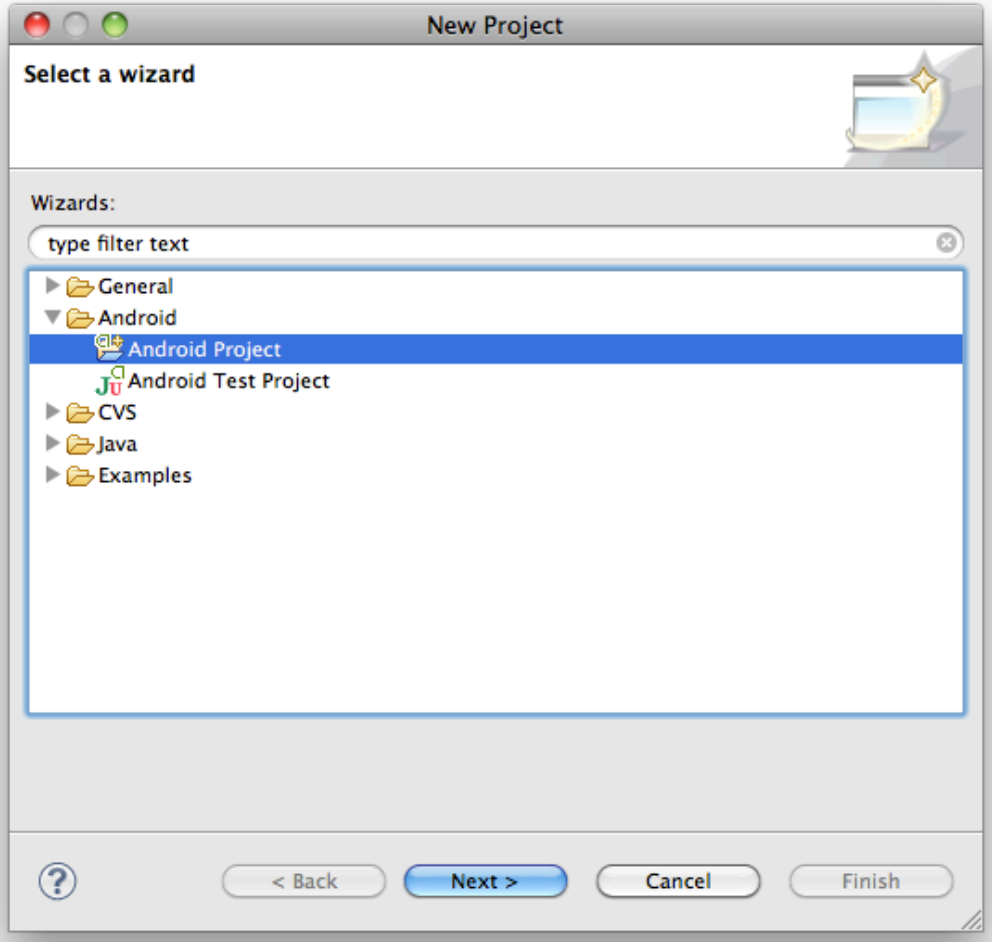
Android Platform

- Linux-based OS
- Platform components: GPS, WiFi, Camera, Audio/Video recording + playback, Sensors (acceleration, temperature, proximity, gyroscope, magnetic, …)
- SQLite Local Database Storage
- Built-in Applications (Home, Contacts, Phone, Browser, Voice Recognition, Camera, …)

Structure of an Android App

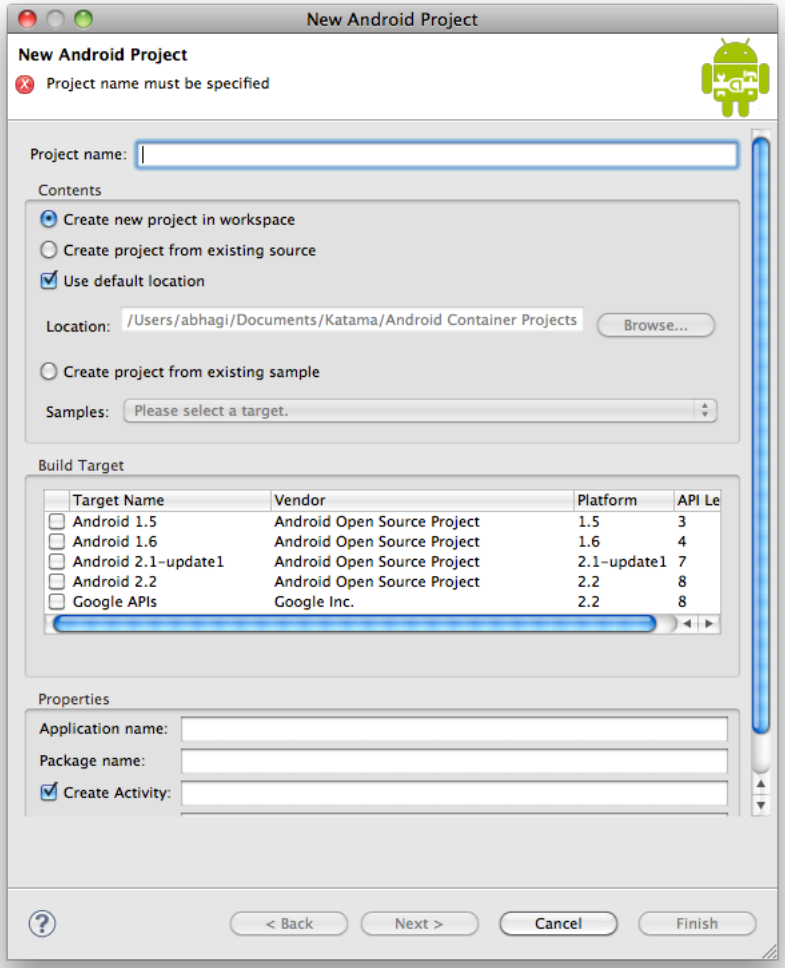


Creating Android Projects, Step 1:

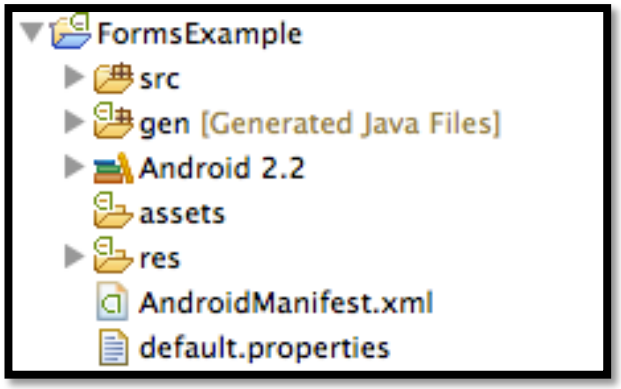


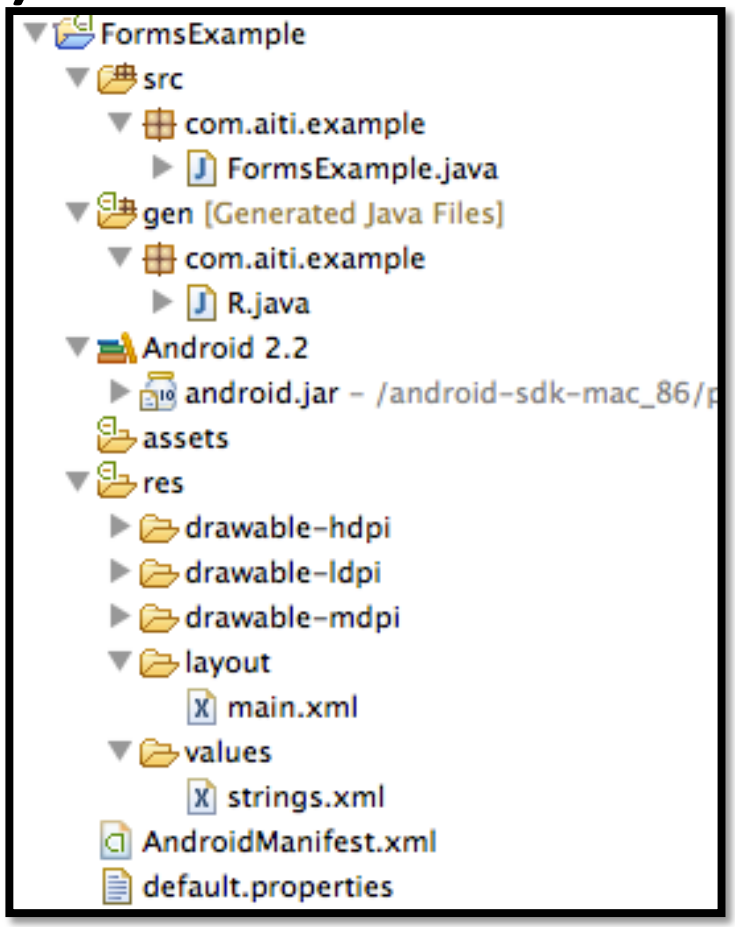
Creating Android Projects, Step 2:

- Project Name
- Build Target
- Application name
- Package name
- Activity name
- Min SDK Version



Anatomy of Android Project (in Eclipse)



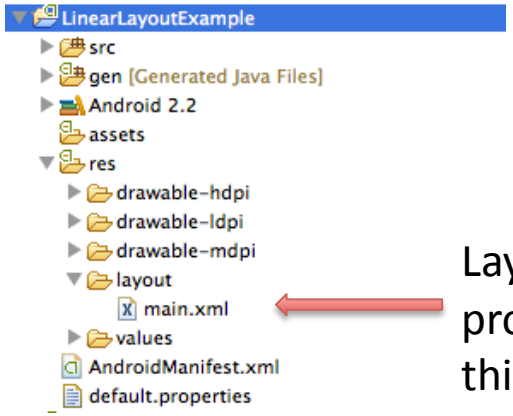


Agenda

- Getting Started
  - Intro to the Android platform
  - Android programming in Eclipse
  - Anatomy of a project
- Basic Android Components
  - Layouts
  - Views and Widgets
  - Menus

Layouts

- Defined in two ways
  - XML layout files (Declarative)



Layout file main.xml is auto-generated when an Android project is created in Eclipse. App layout can be defined in this file in XML.

  - using code

    (Programmatic, e.g. in the onCreate() method)

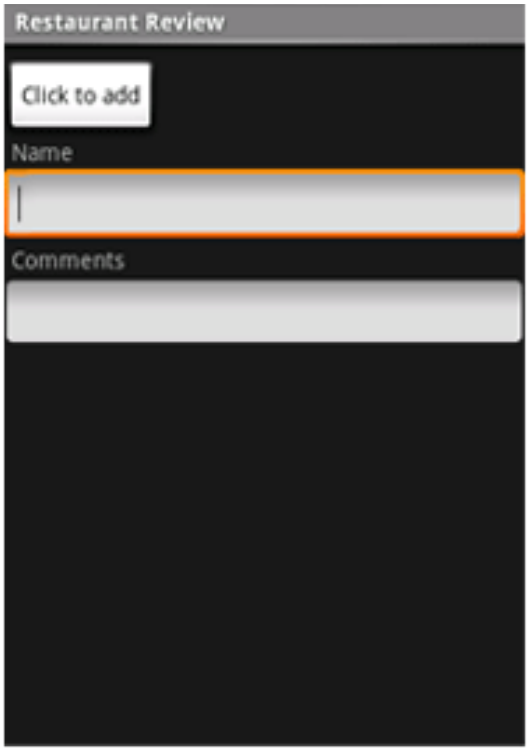
Which to use?

Declarative or Programmatic

Best practice: Use both!

- Declarative (XML) to define static UI components
  - Look and feel
  - Layout, widgets, etc.
- Programmatic (Java) to define interaction
  - What UI does

Some Layouts



LinearLayout

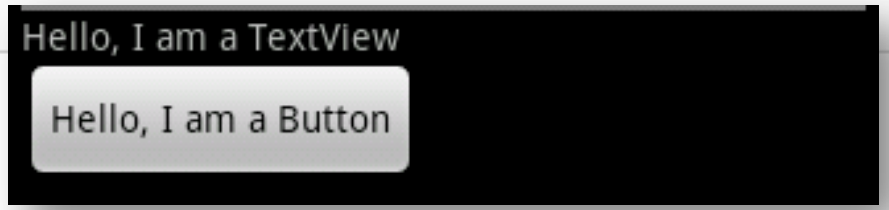
RelativeLayout

TableLayout

LinearLayout

- Arrange components one after another, left-to-right, top-to-bottom:

```
<?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="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```



RelativeLayout

- Position and align components relative to other components:

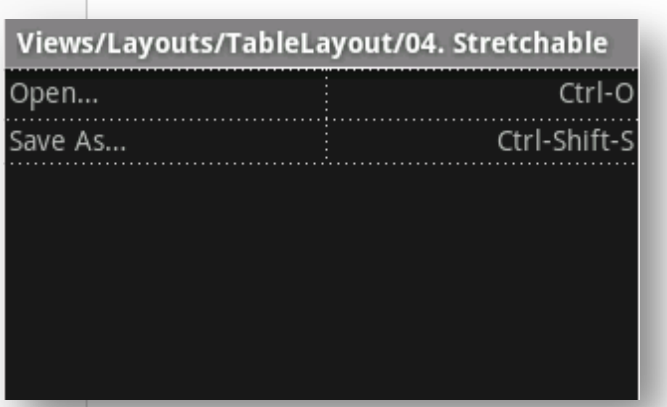
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:background="@drawable/blue"
                android:padding="10px" >

    <TextView android:id="@+id/label"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:text="Type here:" />

    <EditText android:id="@+id/entry"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:background="@android:drawable/editbox_background"
              android:layout_below="@id/label" />

</RelativeLayout>
```

`android:layout_below` is an attribute that can be used only with RelativeLayout. Other such attributes include **`layout_alignParentRight`**, and **`layout_toLeftOf`**.

TableLayout

- Position components in rows and columns:

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="1">
    <TableRow>
        <TextView
            android:text="@string/table_layout_4_open"
            android:padding="3dip" />
        <TextView
            android:text="@string/table_layout_4_open_shortcut"
            android:gravity="right"
            android:padding="3dip" />
    </TableRow>

    <TableRow>
        <TextView
            android:text="@string/table_layout_4_save"
            android:padding="3dip" />
        <TextView
            android:text="@string/table_layout_4_save_shortcut"
            android:gravity="right"
            android:padding="3dip" />
    </TableRow>
</TableLayout>
```



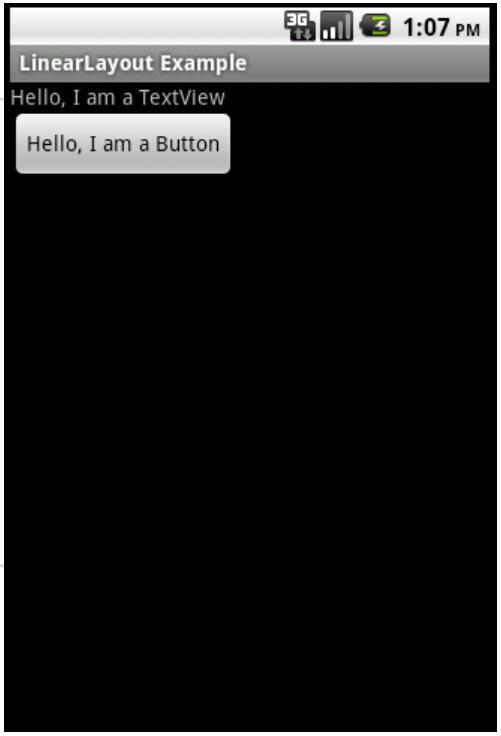
Views

- What they are: UI components
- Some common views and widgets:
  - Button
  - EditText (a text box)
  - TextView (a text label)
  - ListView
  - GridView
  - TabView
  - Spinner (a drop-down menu)
  - CheckBox
  - RadioButton
  - ToggleButton
  - RatingBar
  - MapView (for embedding Google Maps objects in applications)
  - WebView (for embedding web browsers in applications)

Adding Views to Layouts

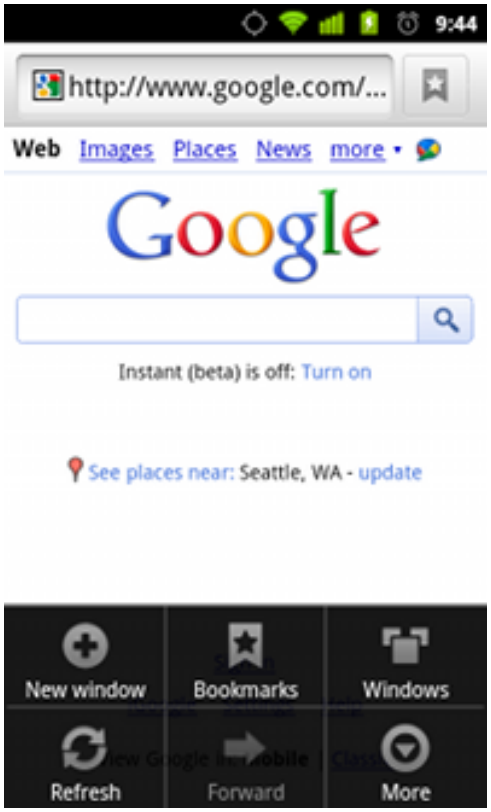
- Example: adding a button and text label to a LinearLayout:

```
<?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:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```



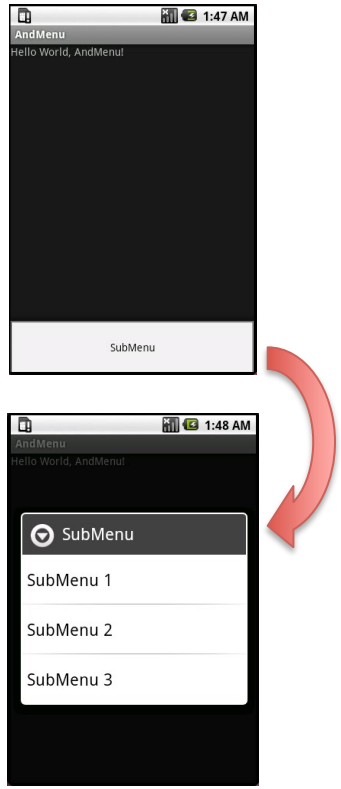
Menus

Options Menu



Context Menu

SubMenu



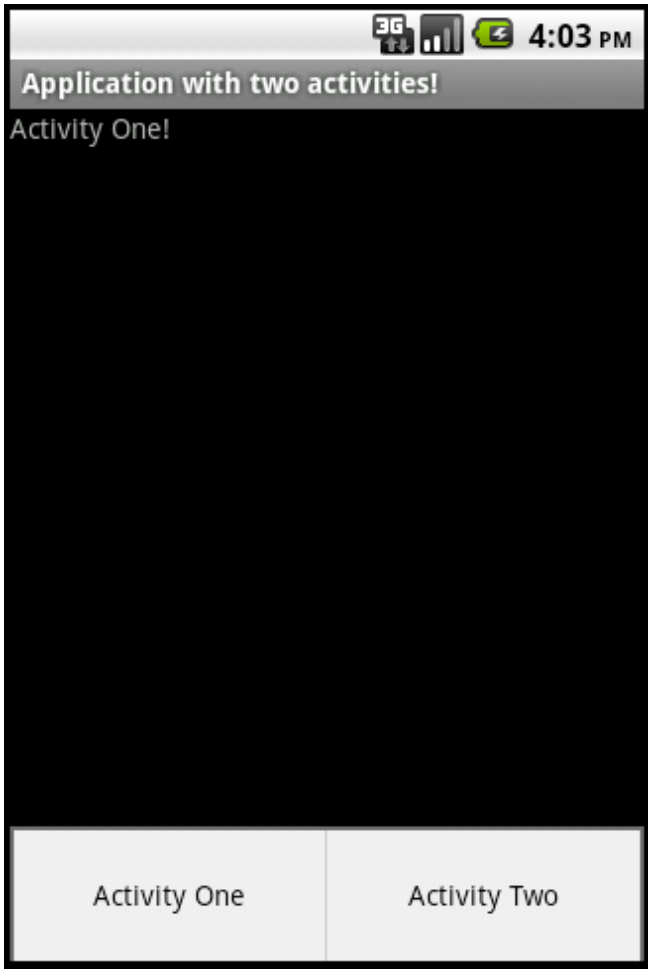
OptionsMenu Example

- Step 1: Implement **`onCreateOptionsMenu()`** method

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    boolean result = super.onCreateOptionsMenu(menu);

    menu.add(Menu.NONE, 0, 0, "Activity One");
    menu.add(Menu.NONE, 1, 1, "Activity Two");

    return result;
}
```

- Step 2: Implement **`onOptionsItemSelected()`** method

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int itemIndex = item.getItemId();

    if (itemIndex == 0){
        //first menu button pressed. do something here
    }
    else if (itemIndex == 1){
        // second menu button pressed. do something here
    }

    return super.onOptionsItemSelected(item);
}
```



Handling events

Listen to events using callback methods:

- onClick()
- onLongClick()
- onFocusChange()
- onKey()
- onTouch()
- onCreateContextMenu()

Example: Event-handling with Buttons

Method 1: Define call-backs using code

```
// 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);
    ...
}
```

Method 2: Define call-backs in Layout XML files

```
<Button android:id="@+id/button1" android:layout_width="80px"
    android:layout_height="fill_parent" android:onClick="clickhandler"
    android:text="1">

</Button>
```

```
public void clickhandler(View clickedobject) {
    int idofclickedobject = clickedobject.getId();

    switch (idofclickedobject) {
    case R.id.button1:
        //do something
        break;
    }
}
```