ANDROID USER INTERFACE

CS 175 Mobile Software Dev
by Dr. Angus Yeung

ANDROID USER INTERFACE

A collection of visual objects arranged on the screen that the user can see and interact with

Can be created in java code or created in an external XML layout file

Each screen in an Android app is identified as a layout resource

LAYOUTS

CS 175 Mobile Software Dev
by Dr. Angus Yeung

LAYOUTS

The term Layout denotes the visual architecture of the application

There are six standard root layouts as follows:

- RelativeLayout
- LinearLayout
- TableLayout
- RowLayout
- GridLayout
- FrameLayout

LAYOUTS

A **RelativeLayout** is used for screen designs that require control elements to be positioned in relation to one another

A **LinearLayout** is used for simple arrangements that require elements to be displayed along either a horizontal or vertical line

A **TableLayout** is used to arrange elements into tabular rows and columns

LAYOUTS

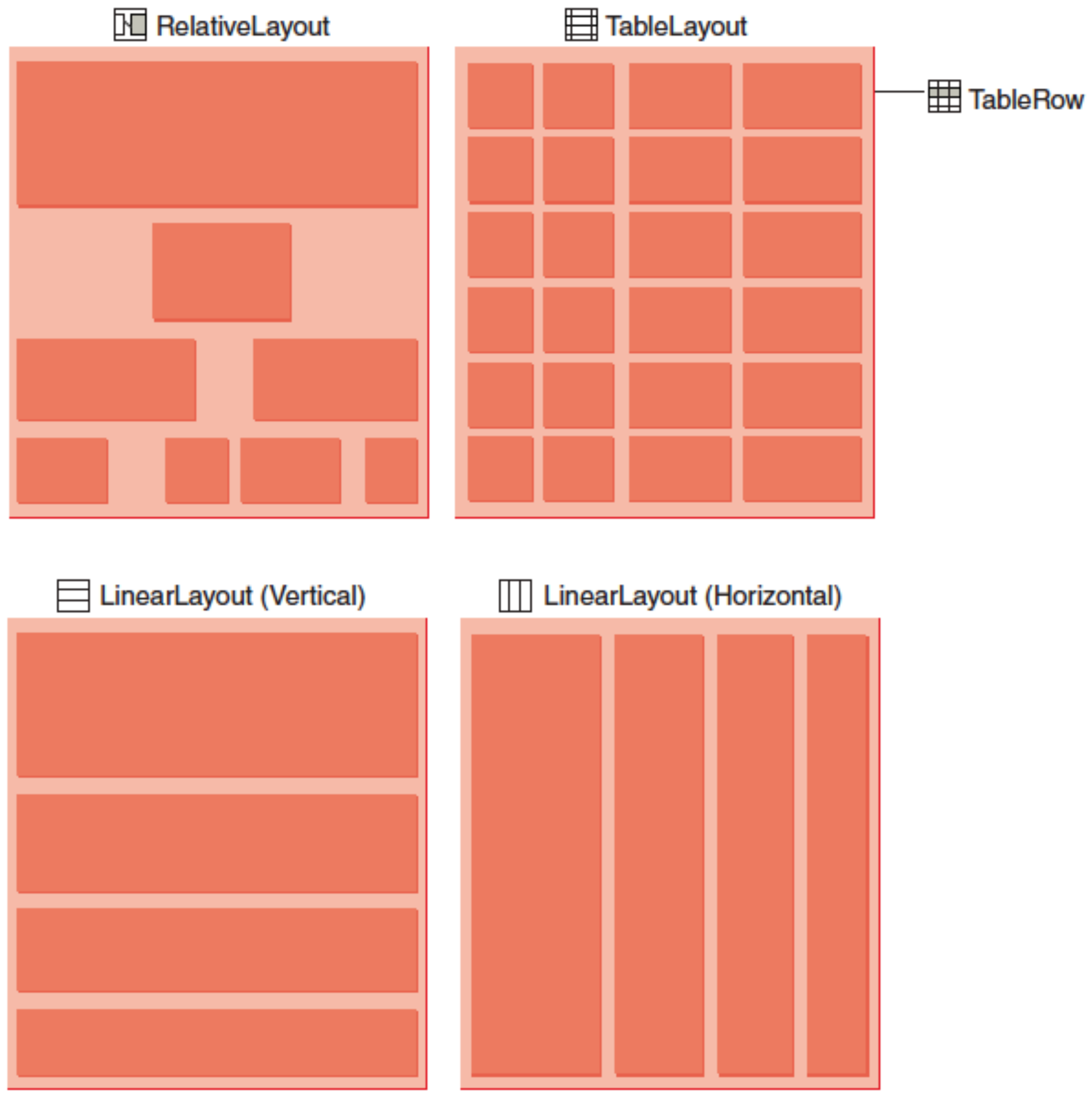


FIGURE 2-1 Standard Layout Types.

LAYOUTS

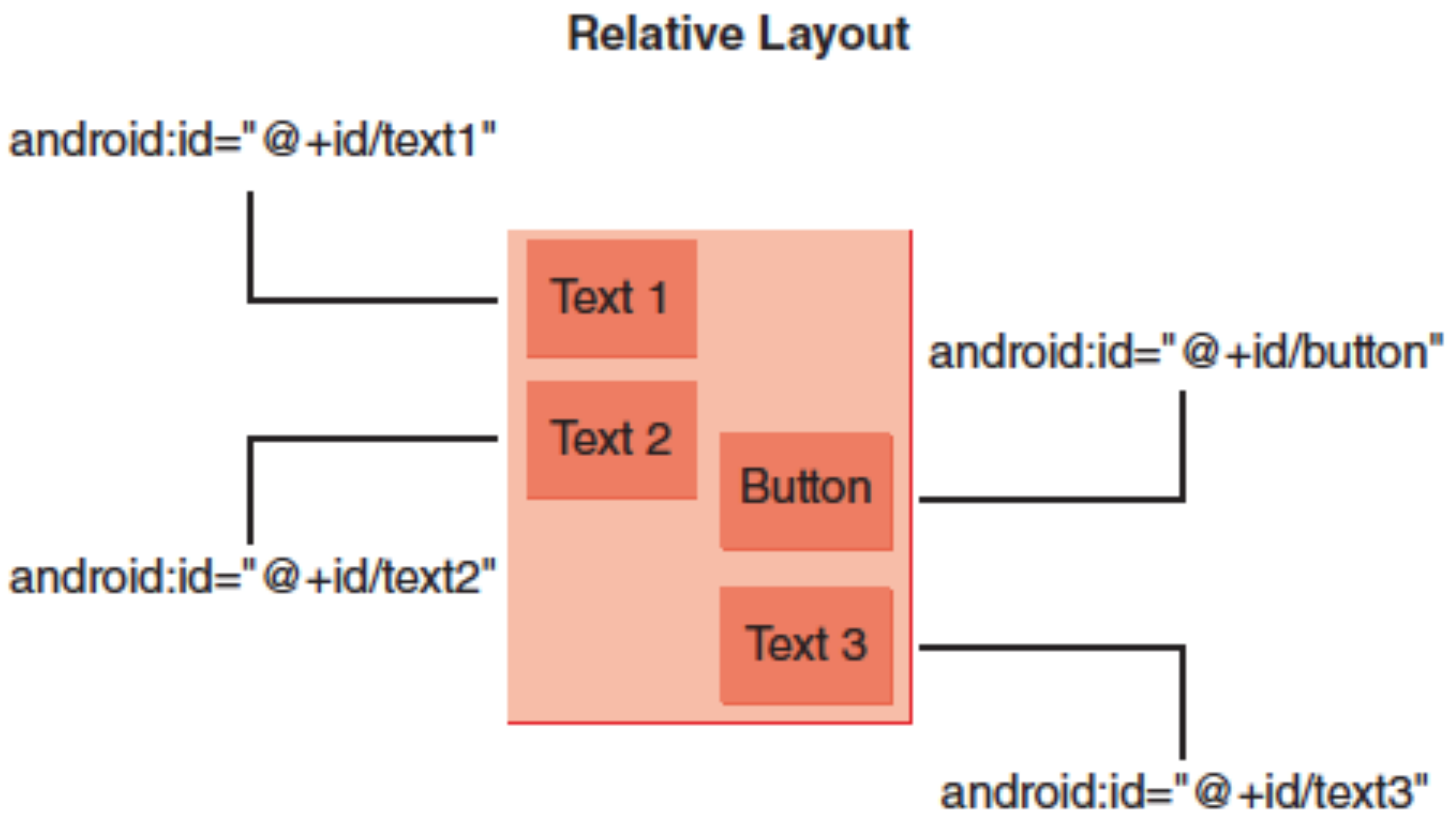


FIGURE 2-2 RelativeLayout elements are positioned relative to each other.

THE VIEW CLASS

Android user interface is built around an object called a View.

A **View** describes every interactive visual control object that appears on an application screen.

Every **control object** in an Android user interface is a subclass of the Android View class.

UI CONSTRUCTION

The user interface for your application can be built in two ways:

1) constructing it as a layout using XML code (static), or
2) building the entire layout, or pieces of the layout, programmatically at runtime (dynamic)

TEXT INPUT AND OUTPUT

CS 175 Mobile Software Dev
by Dr. Angus Yeung

TEXT INPUT AND OUTPUT

TextView and EditText are the two Android text field classes, both derived from the View super class.

TextView is used primarily for text output.

EditText allows text input and editing by the user.

SOFT KEYBOARDS

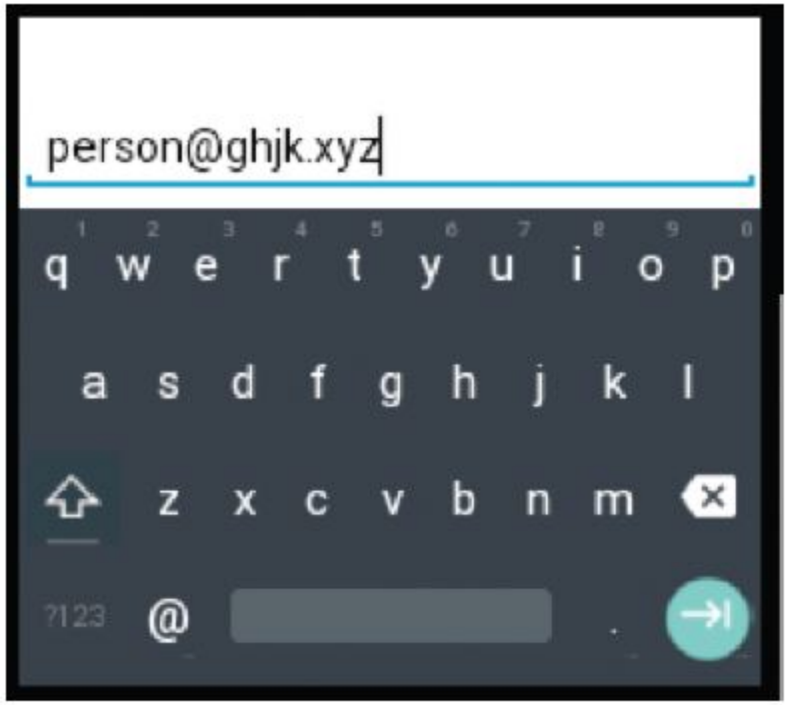
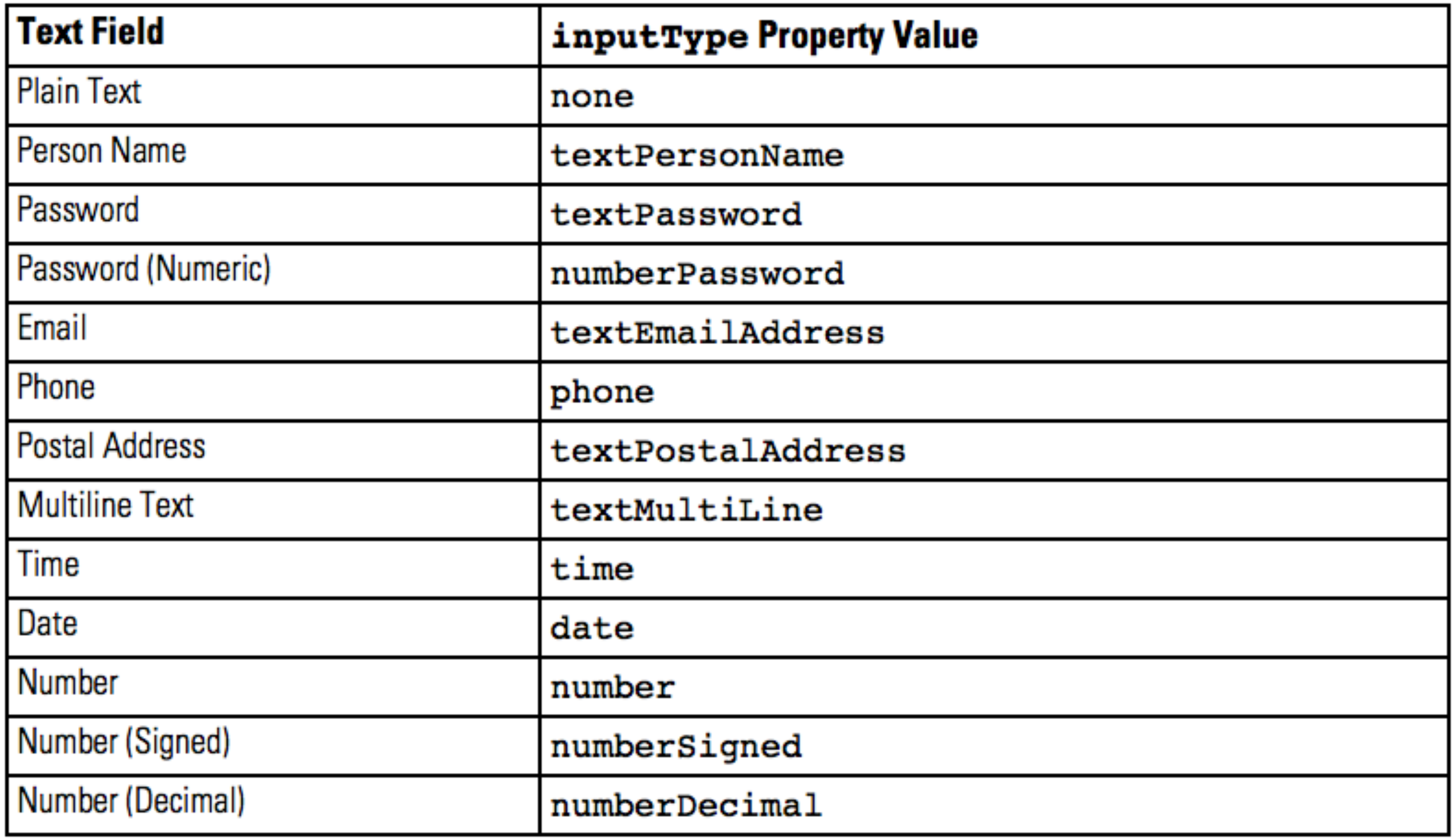


FIGURE 2-6 A soft keyboard configured for the input of an email address.

THE INPUT TYPE ATTRIBUTE

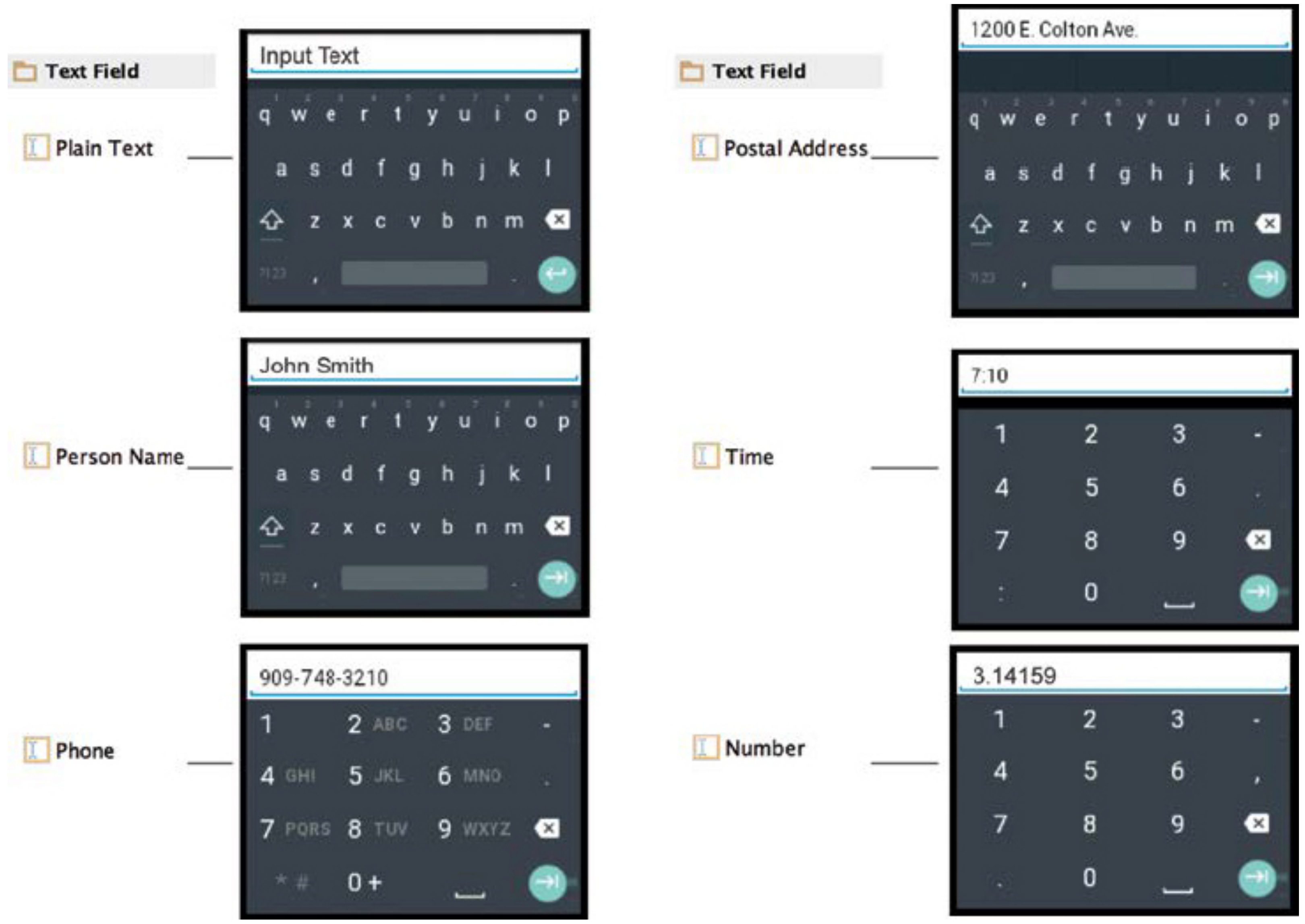
```
<EditText
    android:id="@+id/editText1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="email address"
    android:inputType="textEmailAddress" />
```

TEXT FIELDS



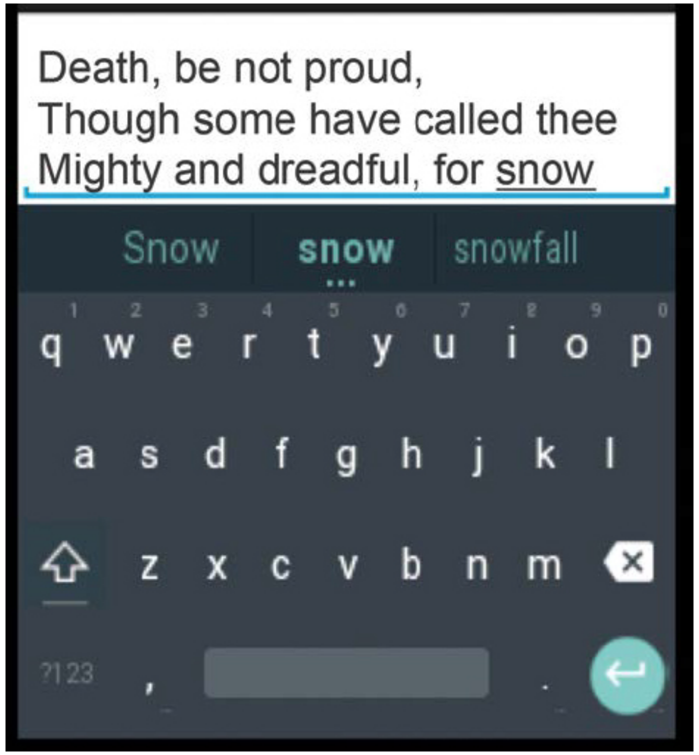
Text Field	`inputType` Property Value
Plain Text	`none`
Person Name	`textPersonName`
Password	`textPassword`
Password (Numeric)	`numberPassword`
Email	`textEmailAddress`
Phone	`phone`
Postal Address	`textPostalAddress`
Multiline Text	`textMultiLine`
Time	`time`
Date	`date`
Number	`number`
Number (Signed)	`numberSigned`
Number (Decimal)	`numberDecimal`

INPUT TYPES


Text Field
Plain Text
Input Text
Person Name
John Smith
Phone
909-748-3210
Text Field
Postal Address
1200 E. Colton Ave.
Time
7:10
Number
3.14159

TEXT FIELDS

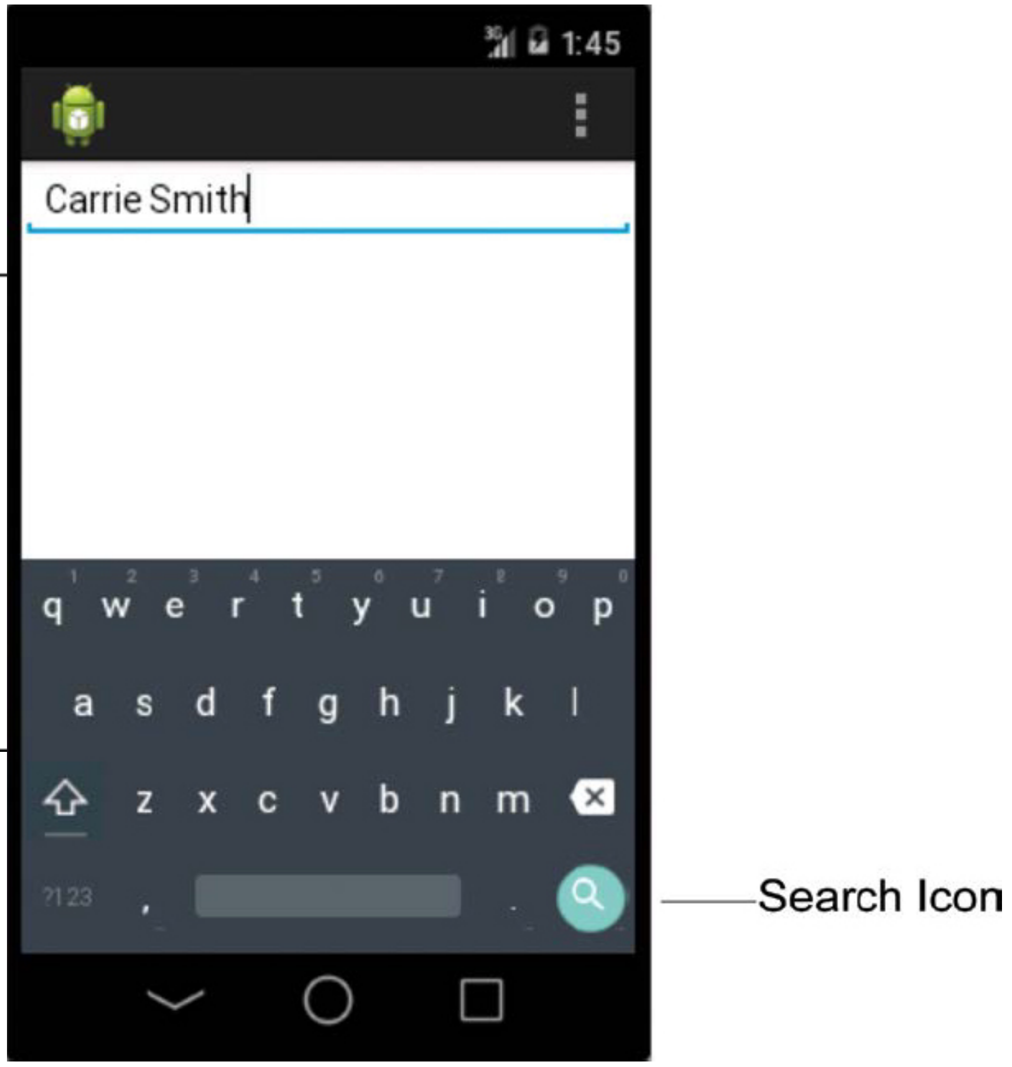
Input Types


Death, be not proud,
Though some have called thee
Mighty and dreadful, for snow

```
<EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType=
          "textMultiLine|textCapSentences
                      |textAutoComplete|textAutoCorrect"
        android:id="@+id/editText1" />
```

TEXT FIELDS

```
<EditText
    android:id="@+id/editText"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/search_hint"
    android:inputType="text"
    android:imeOptions="actionSearch" />
```



FORM WIDGETS

CS 175 Mobile Software Dev
by Dr. Angus Yeung

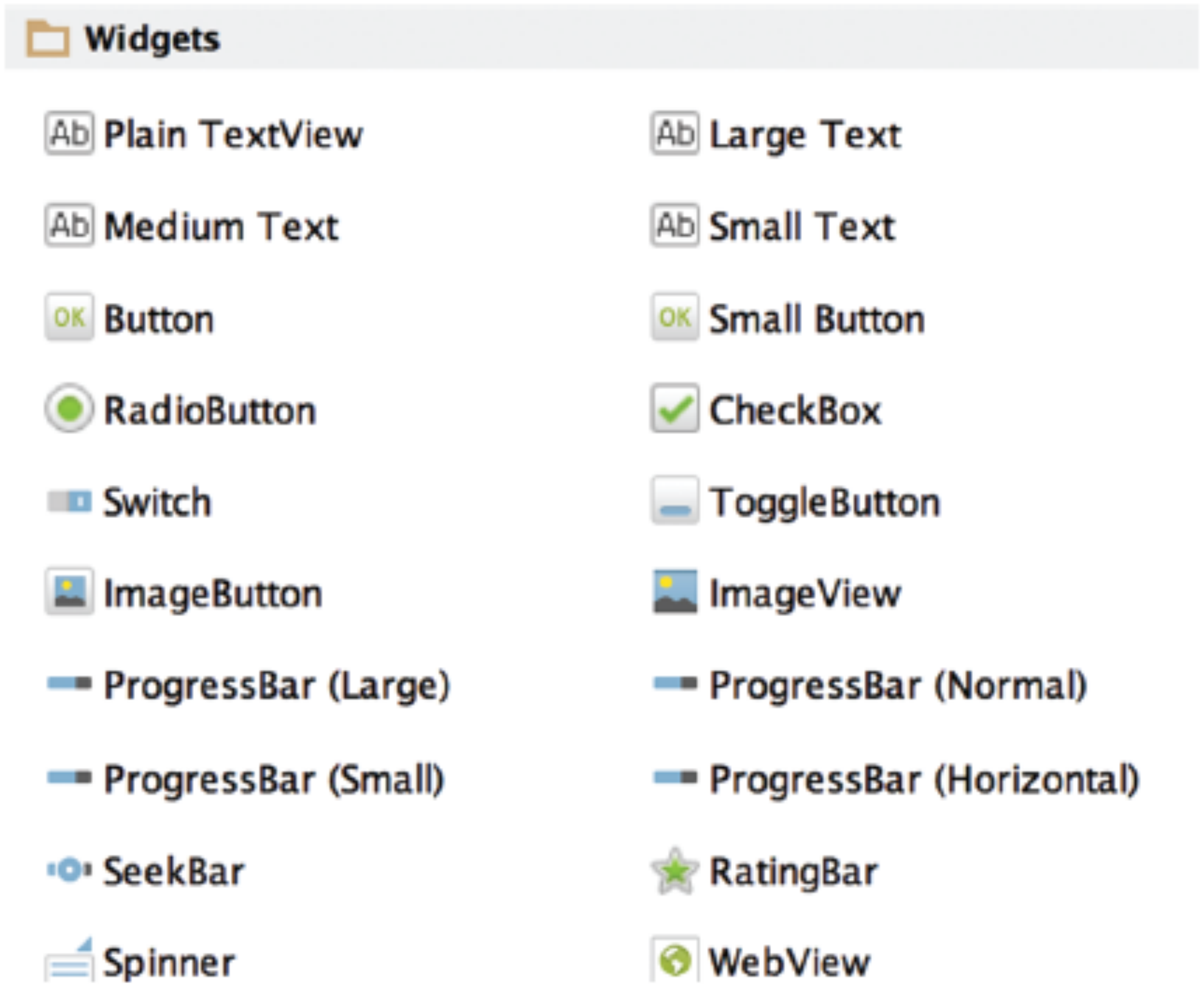
FORM WIDGETS

Android provides a wide set of input controls, (widgets), to be used in an app's user interface.

Widgets are subclasses of the View base class.

Each widget has a built-in set of properties that can be used to customize the appearance of a widget as seen by the user.

WIDGETS



Widgets

Plain TextView	Large Text
Medium Text	Small Text
Button	Small Button
RadioButton	CheckBox
Switch	ToggleButton
ImageButton	ImageView
ProgressBar (Large)	ProgressBar (Normal)
ProgressBar (Small)	ProgressBar (Horizontal)
SeekBar	RatingBar
Spinner	WebView

FIGURE 2-14 Widgets are subclasses of the `View` base class.

RADIOBUTTON AND CHECKBOX

A radio button is specifically used when a single item from a collection of items must be made.

If a radio button is already selected, it will be de-selected when another radio button in the collection is selected.

RADIOBUTTON AND CHECKBOX

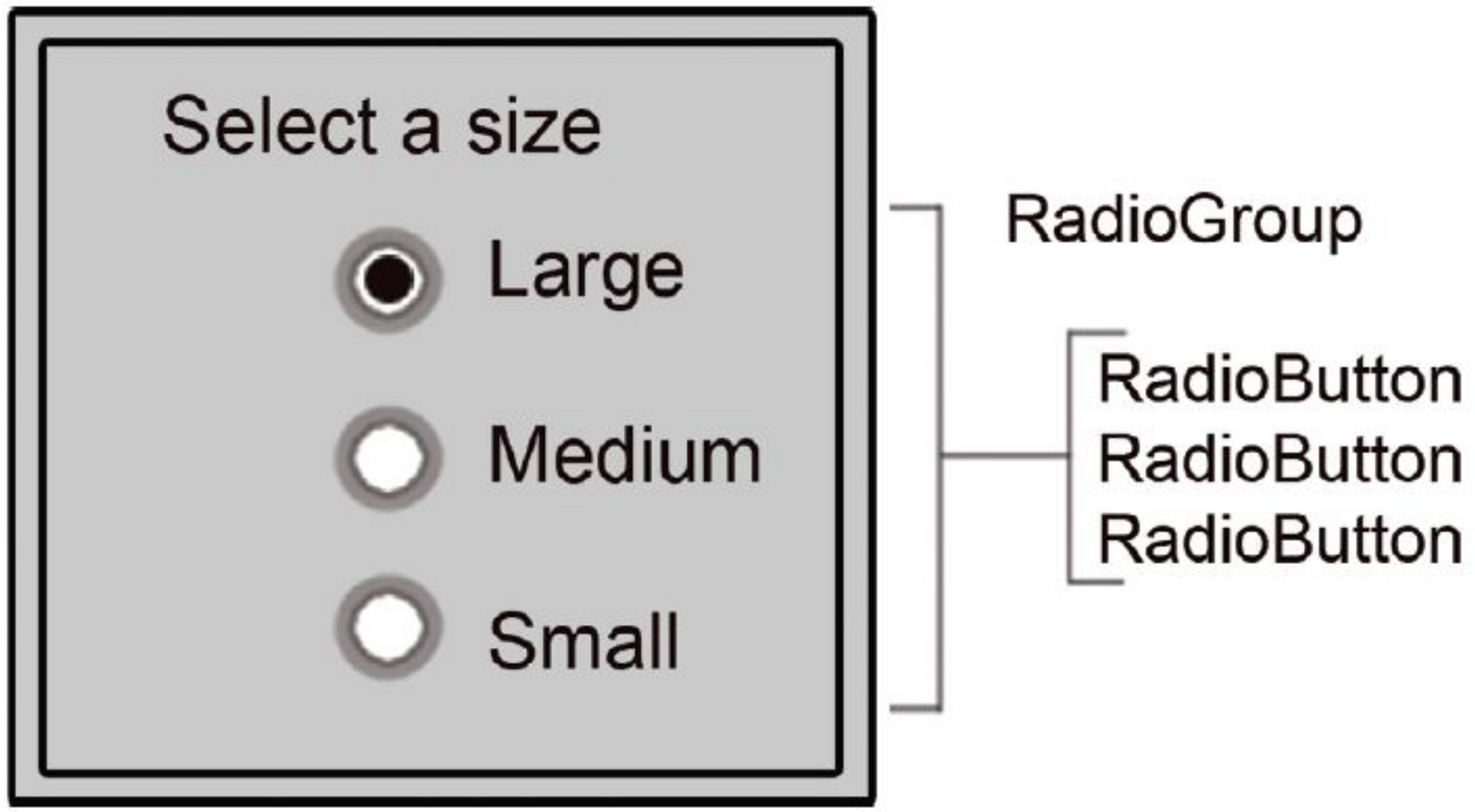
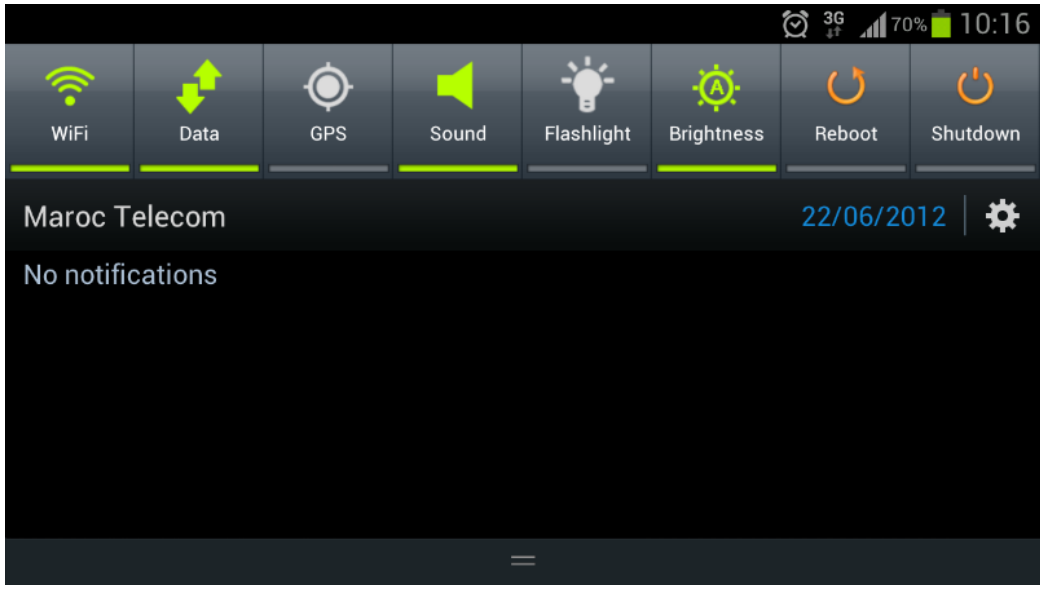


FIGURE 2-15

TOGGLEBUTTON

A toggle button allows the user to change a setting between two states, such as on or off.



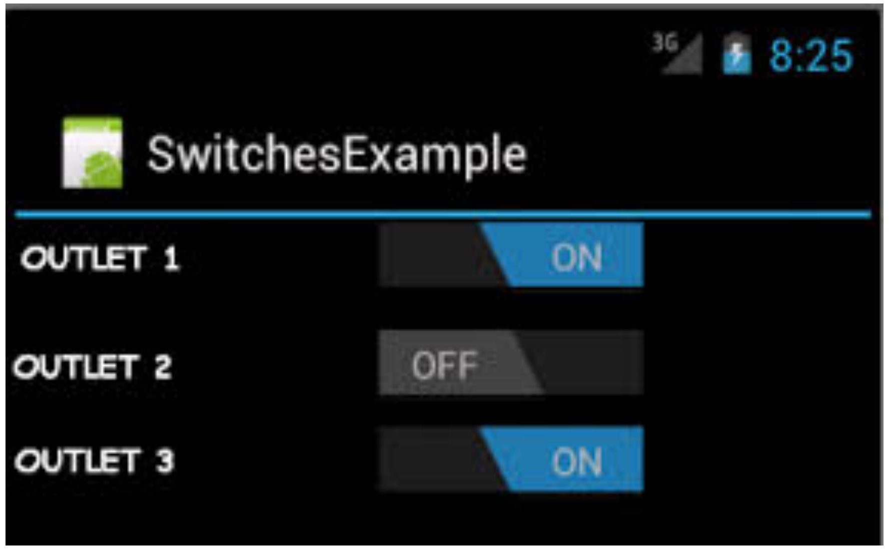
Maroc Telecom 22/06/2012

No notifications

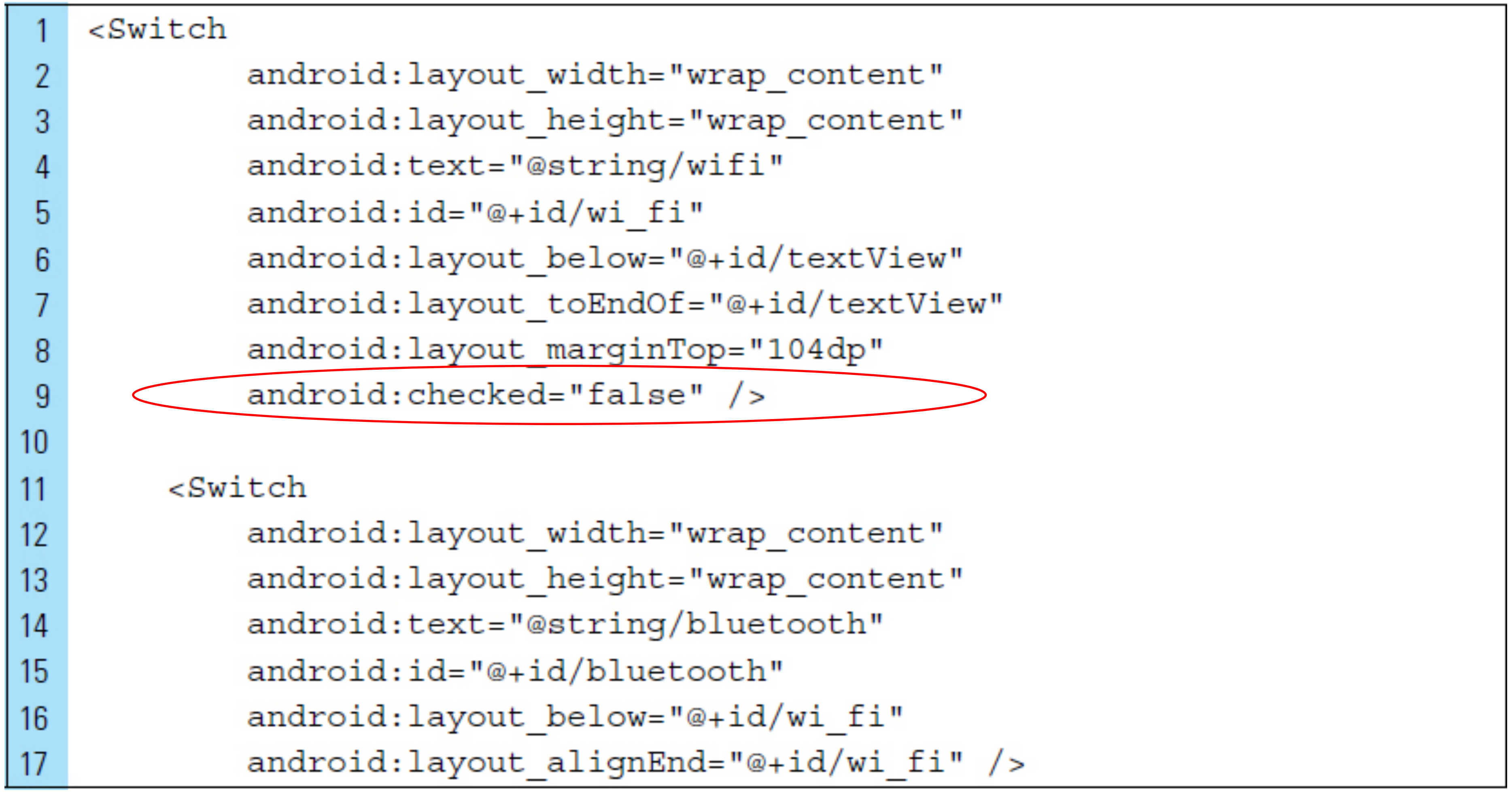
SWITCH

A Switch is a two-state toggle switch widget that can select between two options, off and on

The user can drag the "thumb" back and forth to choose the selected option, or simply tap to toggle as if it were a checkbox.



SWITCH

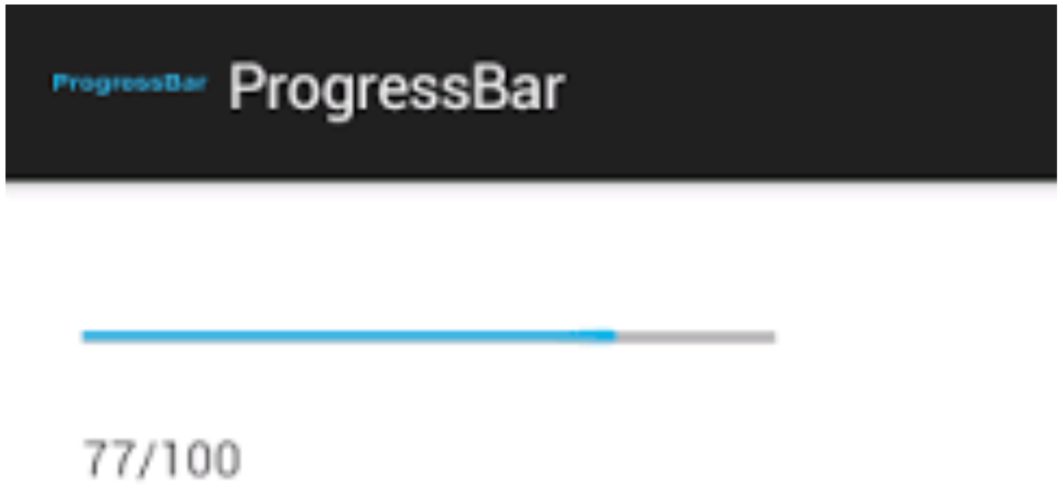


```
<Switch
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/wifi"
        android:id="@+id/wi_fi"
        android:layout_below="@+id/textView"
        android:layout_toEndOf="@+id/textView"
        android:layout_marginTop="104dp"
        android:checked="false" />

    <Switch
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/bluetooth"
        android:id="@+id/bluetooth"
        android:layout_below="@+id/wi_fi"
        android:layout_alignEnd="@+id/wi_fi" />
```

PROGRESSBAR

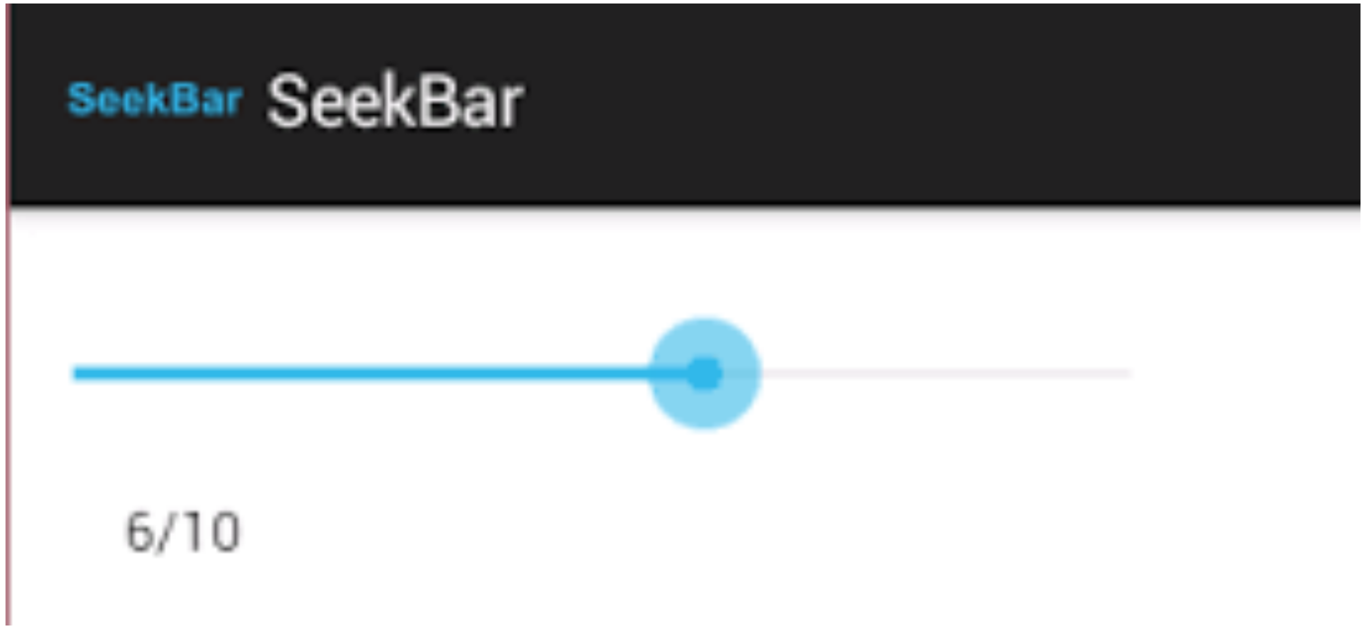
A ProgressBar is a visual indicator of progress in a given operation

A ProgressBar control can be displayed to the user representing how far an operation has progressed



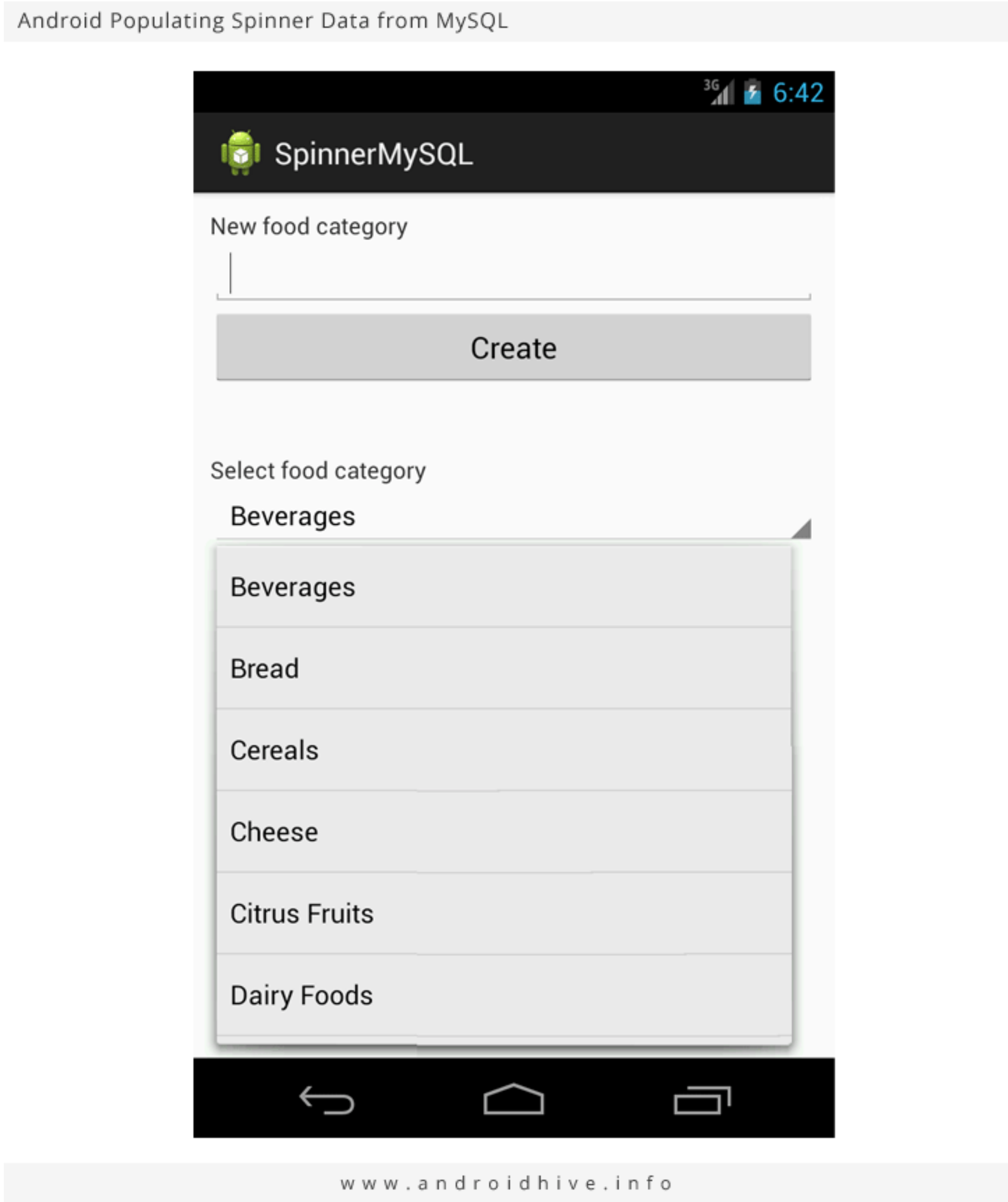
SEEKBAR

A SeekBar is an extension of ProgressBar that adds a draggable thumb.



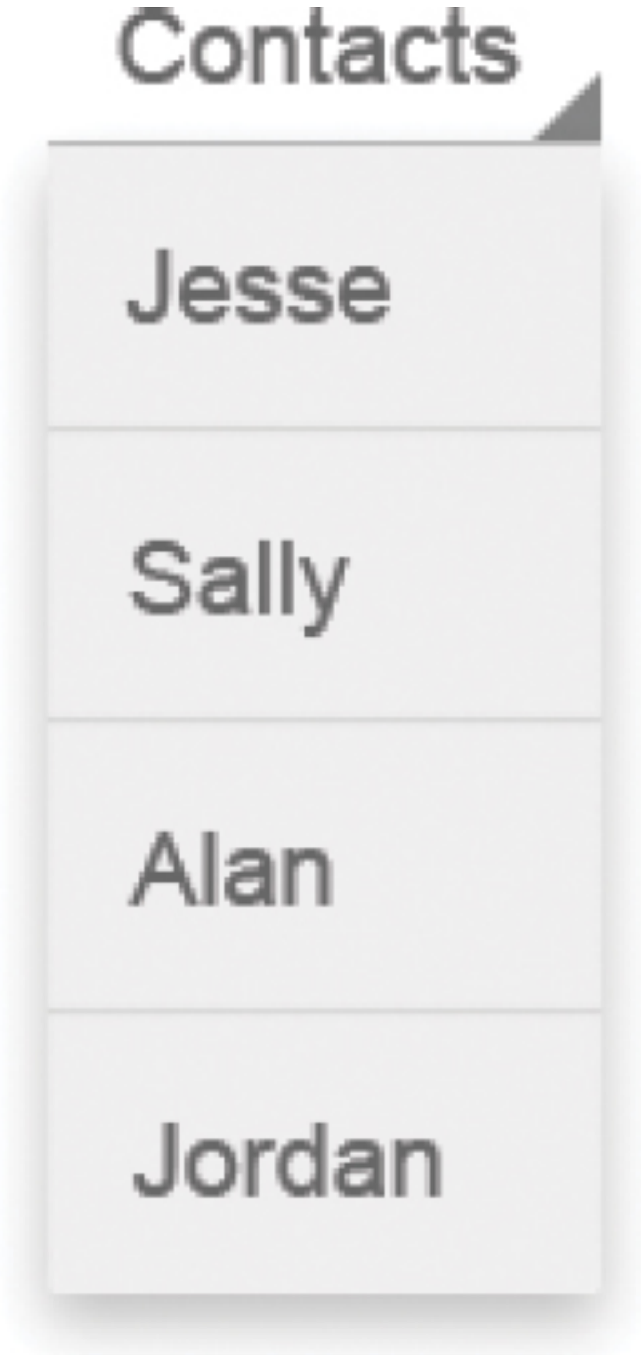
SPINNER

Spinners provide a quick way to select one value from a set of values.

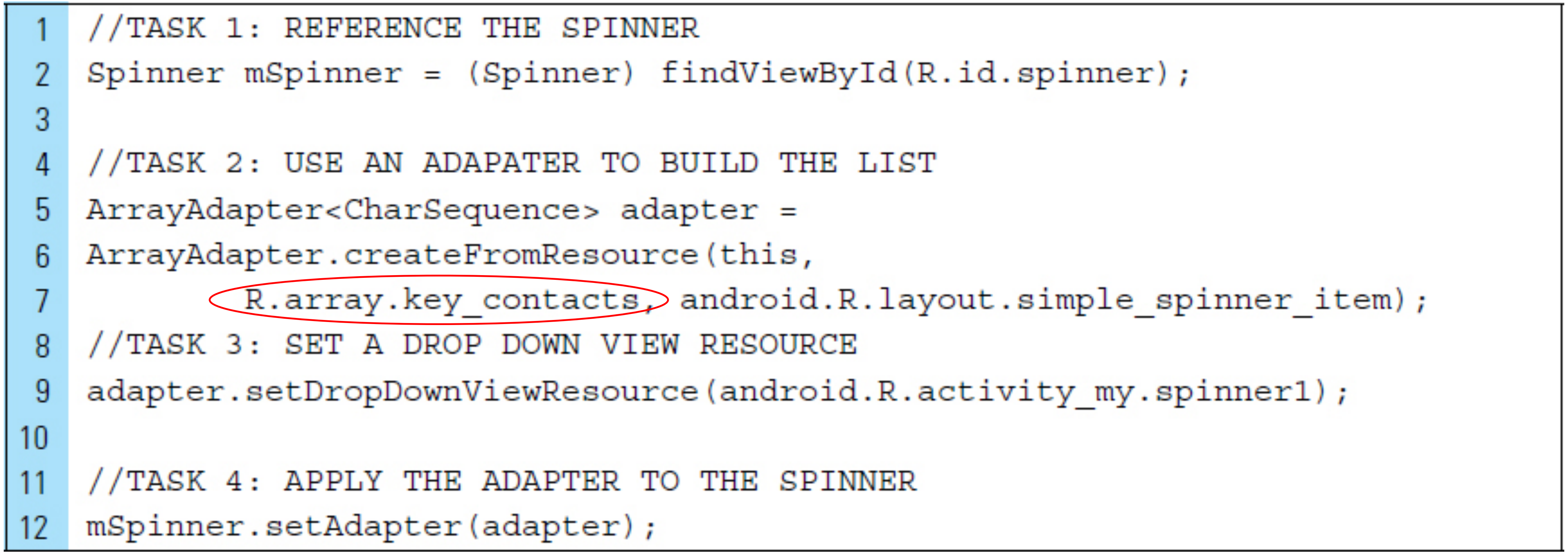


SPINNER

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="key_contacts">
        <item>Jesse</item>
        <item>Sally</item>
        <item>Alan</item>
        <item>Jordan</item>
</string-array>
</resources>
```



SPINNER



```
//TASK 1: REFERENCE THE SPINNER
Spinner mSpinner = (Spinner) findViewById(R.id.spinner);

//TASK 2: USE AN ADAPATER TO BUILD THE LIST
ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this,
        R.array.key_contacts, android.R.layout.simple_spinner_item);
//TASK 3: SET A DROP DOWN VIEW RESOURCE
adapter.setDropDownViewResource(android.R.activity_my.spinner1);

//TASK 4: APPLY THE ADAPTER TO THE SPINNER
mSpinner.setAdapter(adapter);
```

ID & R CLASS

CS 175 Mobile Software Dev
by Dr. Angus Yeung

ID AND R CLASS

One of the most important View attributes is the id attribute. Every View object shares this attribute

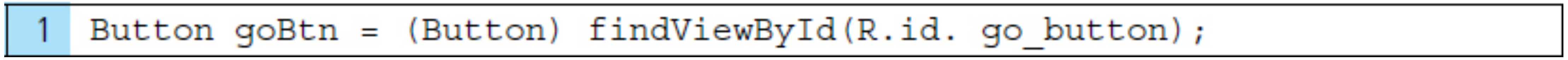
All an app’s View objects are assigned a unique integer that identifies them

Once a View can be uniquely identified, it can be referenced in Java source code

‘@’ AND ‘+’ INSIDE OF THE XML TAG

```
    <Button
        android:id="@+id/go_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="goGet"
        android:text="Button" />
```

USE R TO LOCATE THE UI ELEMENT



```
Button goBtn = (Button) findViewById(R.id. go_button);
```

VIEWGROUPS

CS 175 Mobile Software Dev
by Dr. Angus Yeung

THE VIEWGROUP

A ViewGroup is a container of View objects

All ViewGroup objects are also View objects

A ViewGroup is a special type of View that is designed to hold groups of Views

Each ViewGroup is an invisible container that organizes child Views

THE VIEWGROUP

View objects can be organized in ViewGroup containers.

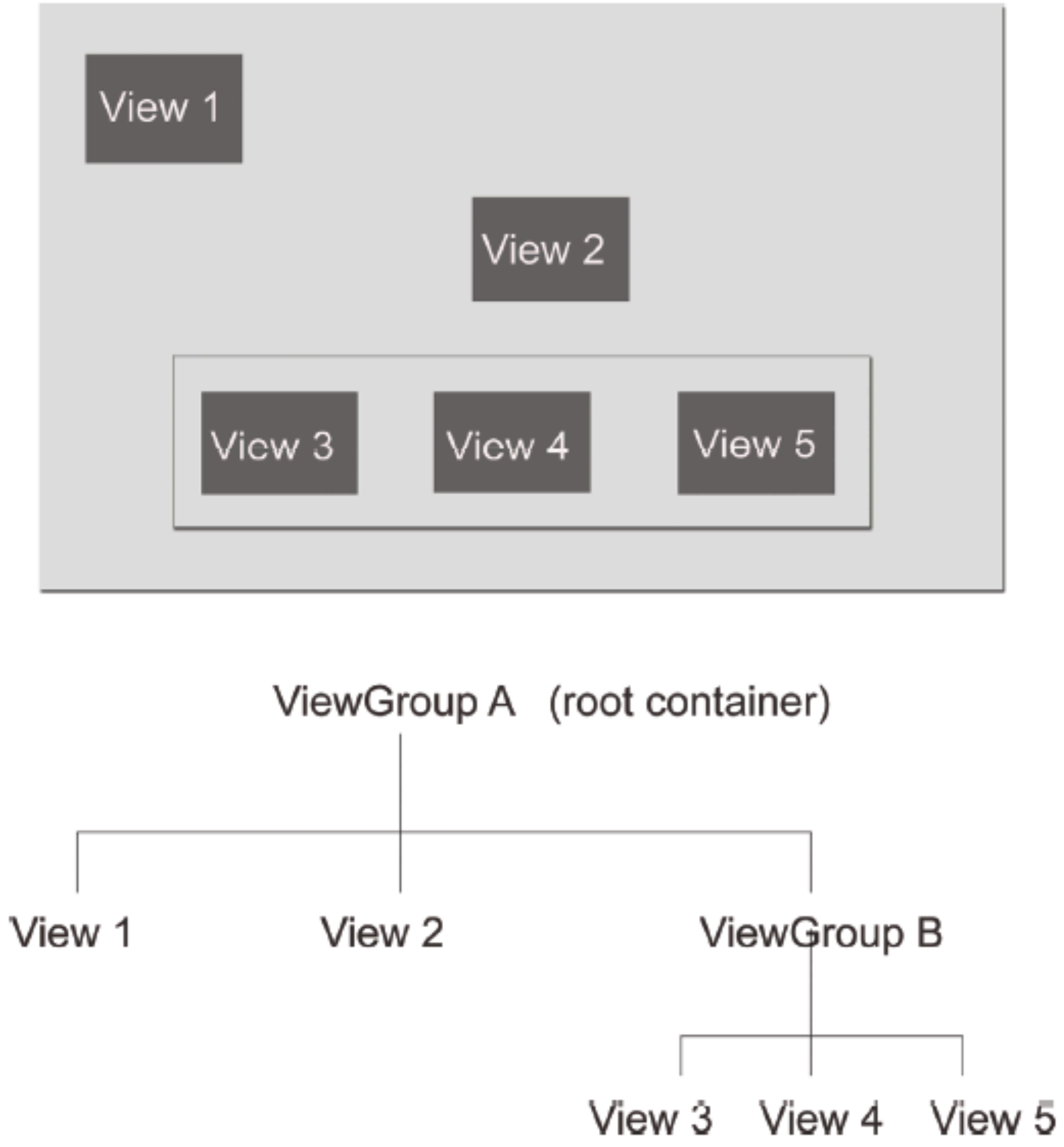
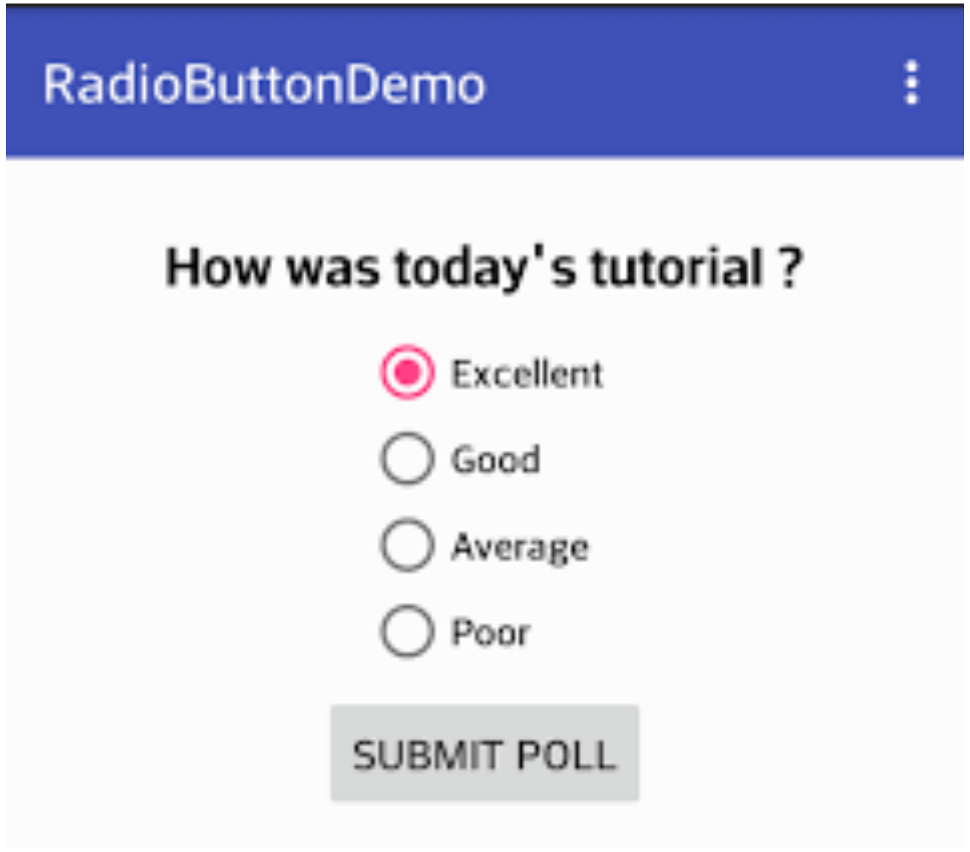


FIGURE 2-19 View objects can be organized in ViewGroup containers.

RADIOGROUP

A RadioGroup object is a ViewGroup container

As a ViewGroup, the RadioGroup is used to group together a related set of RadioButtons



SCREEN AND ORIENTATION

CS 175 Mobile Software Dev
by Dr. Angus Yeung

SCREENS AND ORIENTATION

There are many variations in screen sizes that are available on the market at any given time

Adaptive design is important to Android because it supports flexibility when designing an app that can work on multiple devices

Adaptive design refers to the adaptation of a layout design that fits an individual screen size and or orientation

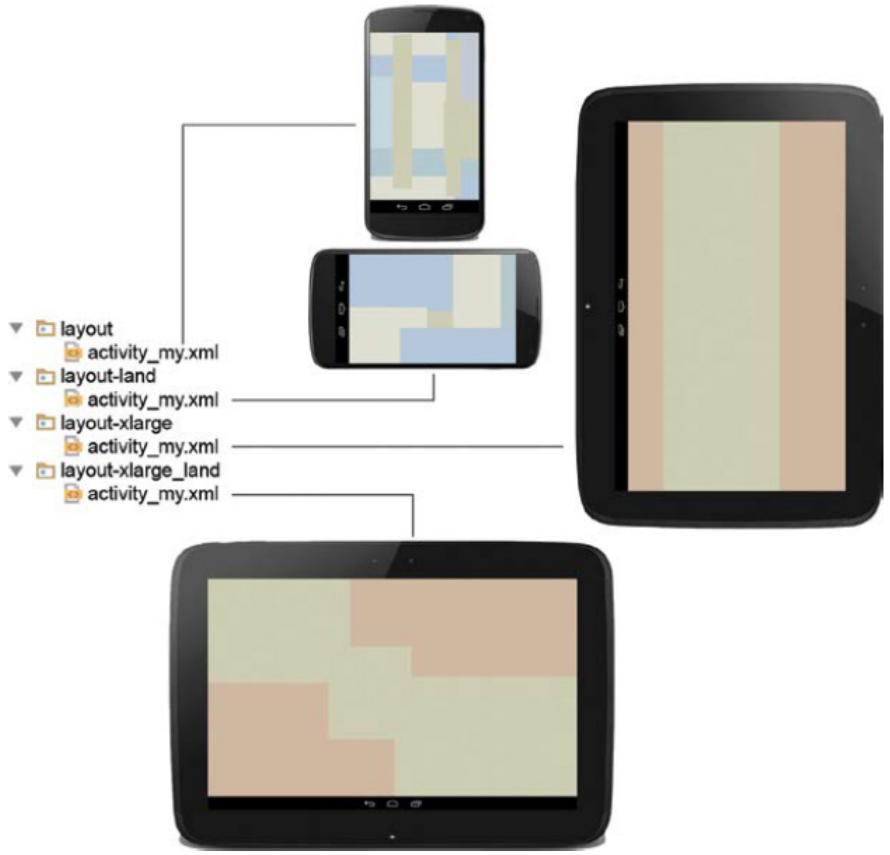
SCREENS AND ORIENTATION

There are many variations in screen sizes that are available on the market at any given time

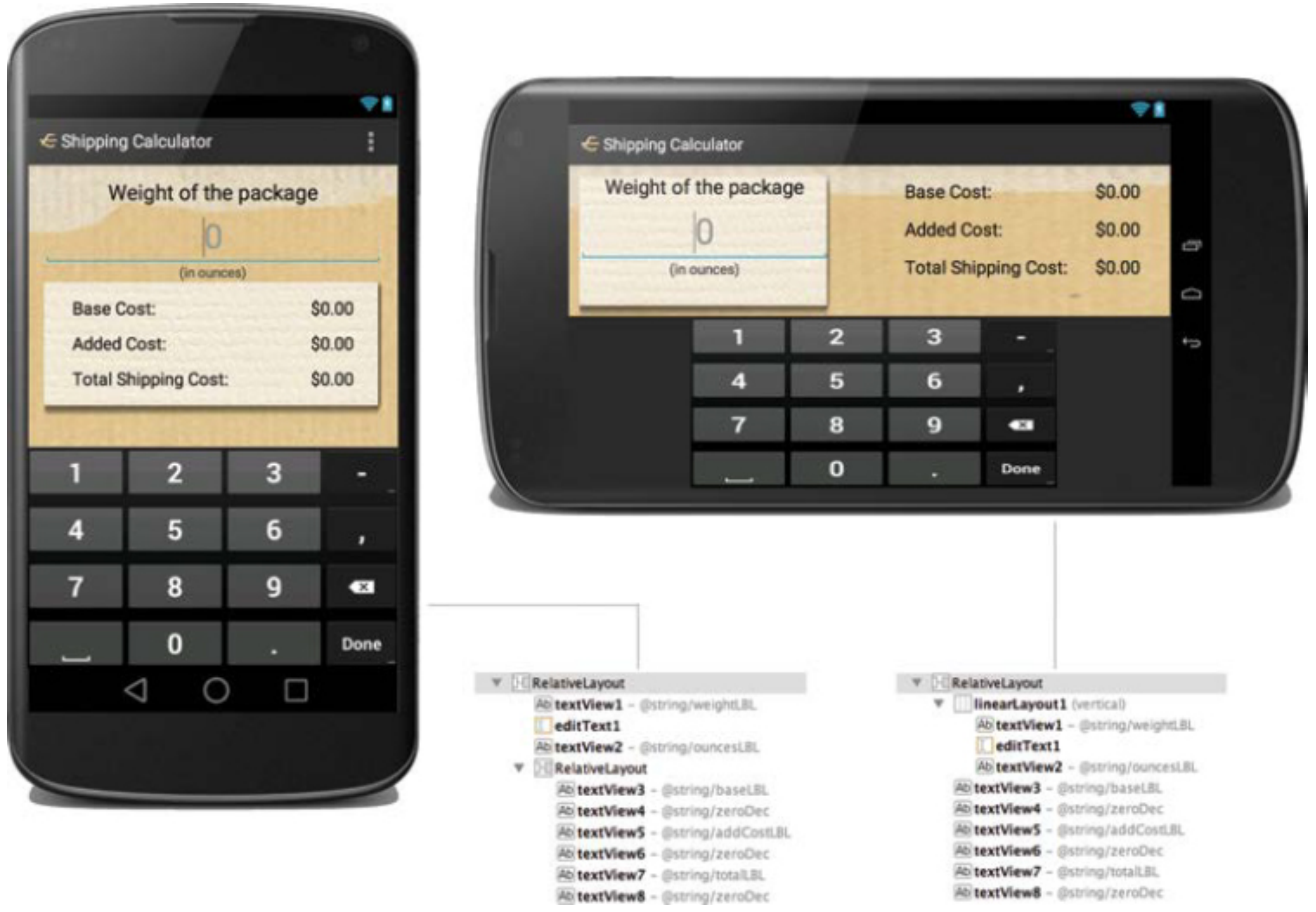
Adaptive design is important to Android because it supports flexibility when designing an app that can work on multiple devices

Adaptive design refers to the adaptation of a layout design that fits an individual screen size and or orientation

SCREENS AND ORIENTATION



```
▼ layout
    activity_my.xml
▼ layout-land
    activity_my.xml
▼ layout-xlarge
    activity_my.xml
▼ layout-xlarge_land
    activity_my.xml
```



```
▼ RelativeLayout
    textView1 - @string/weightLBL
    editText1
    textView2 - @string/ouncesLBL
  ▼ RelativeLayout
      textView3 - @string/baseLBL
      textView4 - @string/zeroDec
      textView5 - @string/addCostLBL
      textView6 - @string/zeroDec
      textView7 - @string/totalLBL
      textView8 - @string/zeroDec
```

```
▼ RelativeLayout
  ▼ linearLayout1 (vertical)
      textView1 - @string/weightLBL
      editText1
      textView2 - @string/ouncesLBL
    textView3 - @string/baseLBL
    textView4 - @string/zeroDec
    textView5 - @string/addCostLBL
    textView6 - @string/zeroDec
    textView7 - @string/totalLBL
    textView8 - @string/zeroDec
```

TABLE LAYOUT

CS 175 Mobile Software Dev
by Dr. Angus Yeung

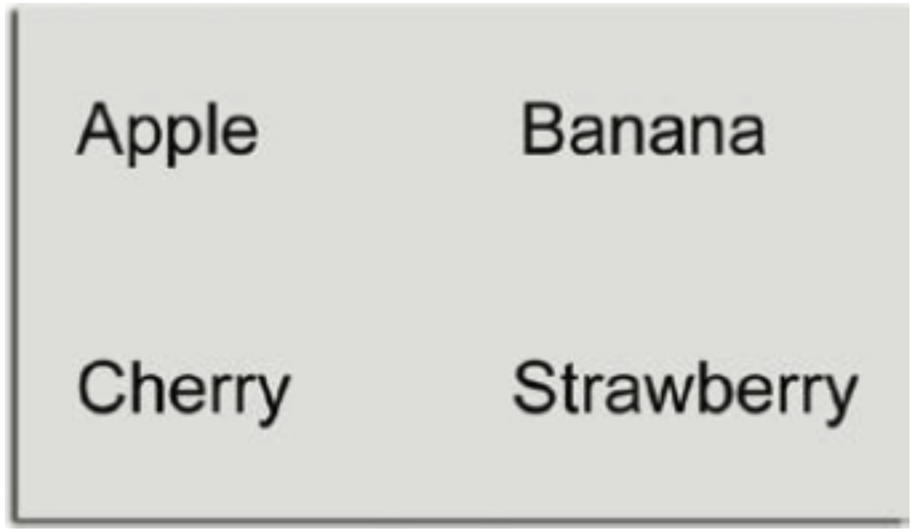
TABLELAYOUT AND TABLEROW

In Android, TableLayouts are often used for organizing data content into tabular form

Tables can be added to a layout file using the Graphical Layout Editor or programmatically using Java

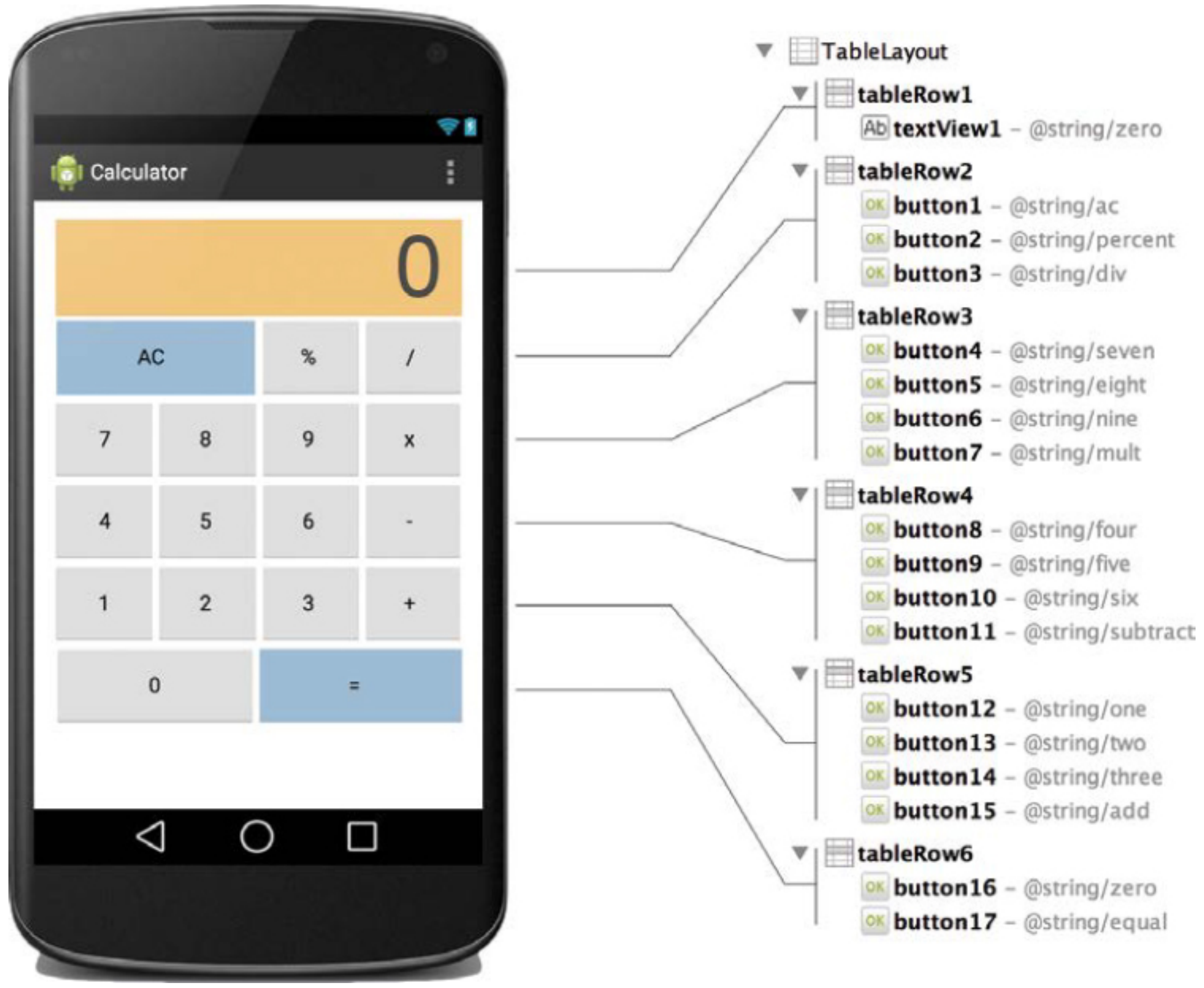
SETTING UP TABLE LAYOUT PROGRAMMATICALLY

```
TableLayout table = new TableLayout(this);

table.setStretchAllColumns(true);
table.setShrinkAllColumns(true);

TableRow row1 = new TableRow(this);
TextView fruit1 = new TextView(this);
fruit1.setText("Apple");
TextView fruit2 = new TextView(this);
fruit2.setText("Banana");
row1.addView(fruit1);
row1.addView(fruit2);

TableRow row2 = new TableRow(this);
TextView fruit3 = new TextView(this);
fruit3.setText("Cherry");
TextView fruit4 = new TextView(this);
fruit4.setText("Strawberry");
row2.addView(fruit3);
row2.addView(fruit4);
table.addView(row1);
table.addView(row2);

setContentView(table);
```



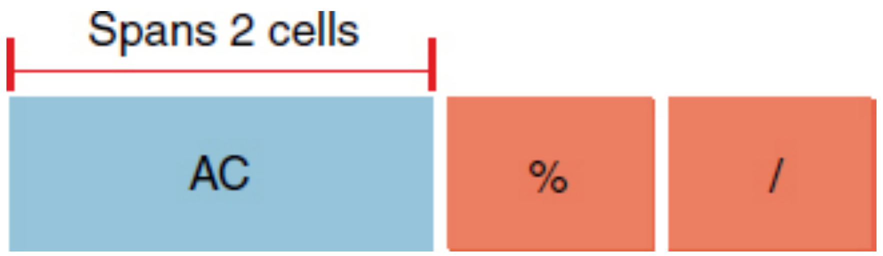
Apple	Banana
Cherry	Strawberry

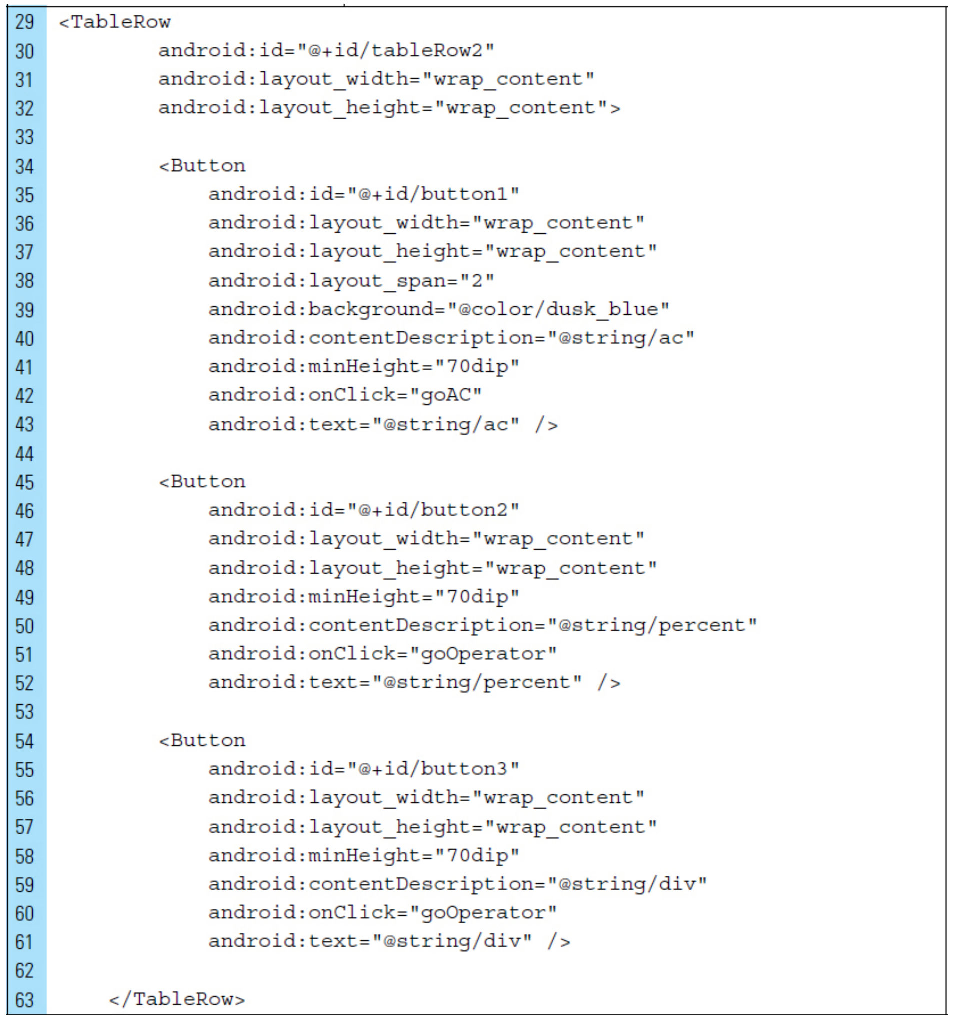
SETTING TABLE LAYOUT USING XML



```
▼ TableLayout
  ▼ tableRow1
      textView1 - @string/zero
  ▼ tableRow2
      button1 - @string/ac
      button2 - @string/percent
      button3 - @string/div
  ▼ tableRow3
      button4 - @string/seven
      button5 - @string/eight
      button6 - @string/nine
      button7 - @string/mult
  ▼ tableRow4
      button8 - @string/four
      button9 - @string/five
      button10 - @string/six
      button11 - @string/subtract
  ▼ tableRow5
      button12 - @string/one
      button13 - @string/two
      button14 - @string/three
      button15 - @string/add
  ▼ tableRow6
      button16 - @string/zero
      button17 - @string/equal
```

USING LAYOUT SPAN





```
<TableRow
        android:id="@+id/tableRow2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

        <Button
            android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_span="2"
            android:background="@color/dusk_blue"
            android:contentDescription="@string/ac"
            android:minHeight="70dip"
            android:onClick="goAC"
            android:text="@string/ac" />

        <Button
            android:id="@+id/button2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:minHeight="70dip"
            android:contentDescription="@string/percent"
            android:onClick="goOperator"
            android:text="@string/percent" />

        <Button
            android:id="@+id/button3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:minHeight="70dip"
            android:contentDescription="@string/div"
            android:onClick="goOperator"
            android:text="@string/div" />

    </TableRow>
```

CONTAINER VIEWS

CS 175 Mobile Software Dev
by Dr. Angus Yeung

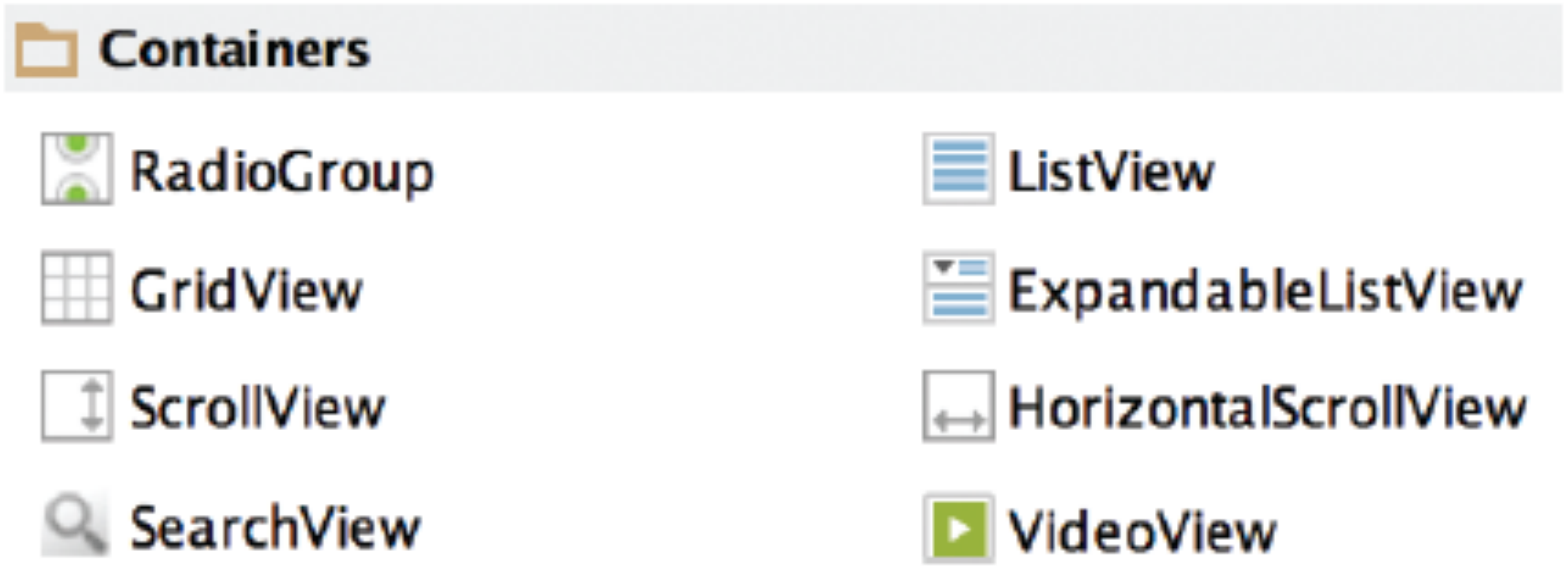
CONTAINER VIEWS

Container Views are simply ViewGroups, which are Views.

Android categorizes this group of Views as “containers” because their sole function is to act as containers for other views.

Any object that provides access to container values is referred to as a Container.

VIEWGROUP CONTAINERS


Containers

RadioGroup

ListView

GridView

ExpandableListView

ScrollView

HorizontalScrollView

SearchView

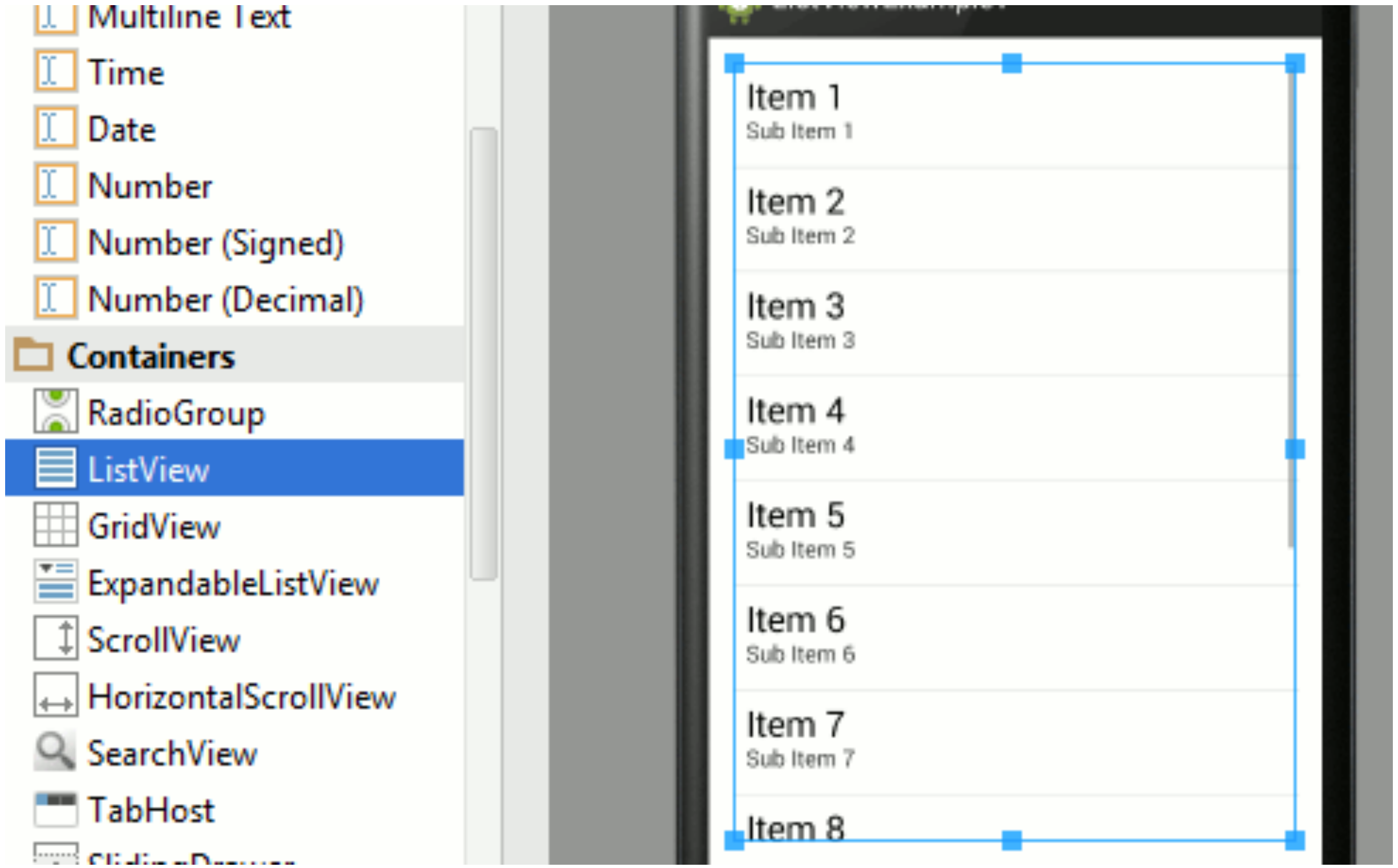
VideoView

FIGURE 2-40 ViewGroup Containers.

LISTVIEW

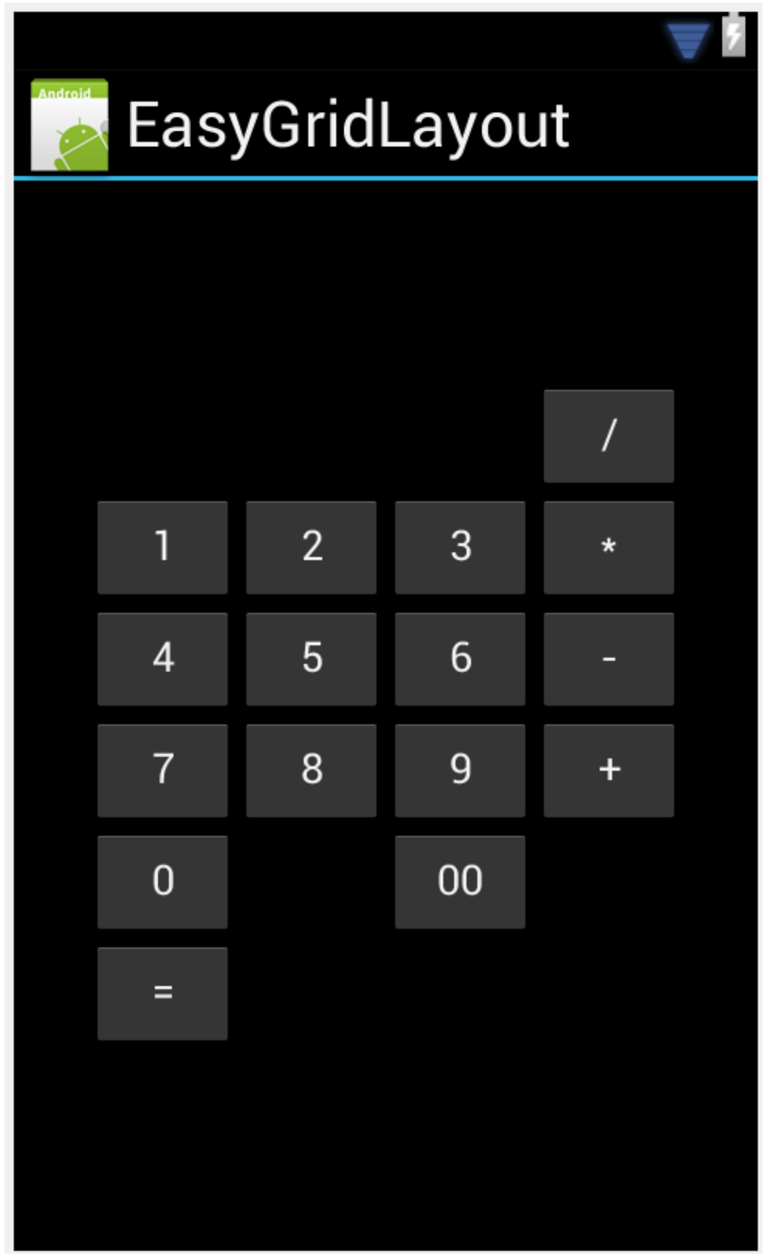
The ListView, GridView, and ExpandableListView are all AdapterViews. This means they are populated with Views that are identified by an Adapter.

A ListView object displays items in a vertically scrolling list.



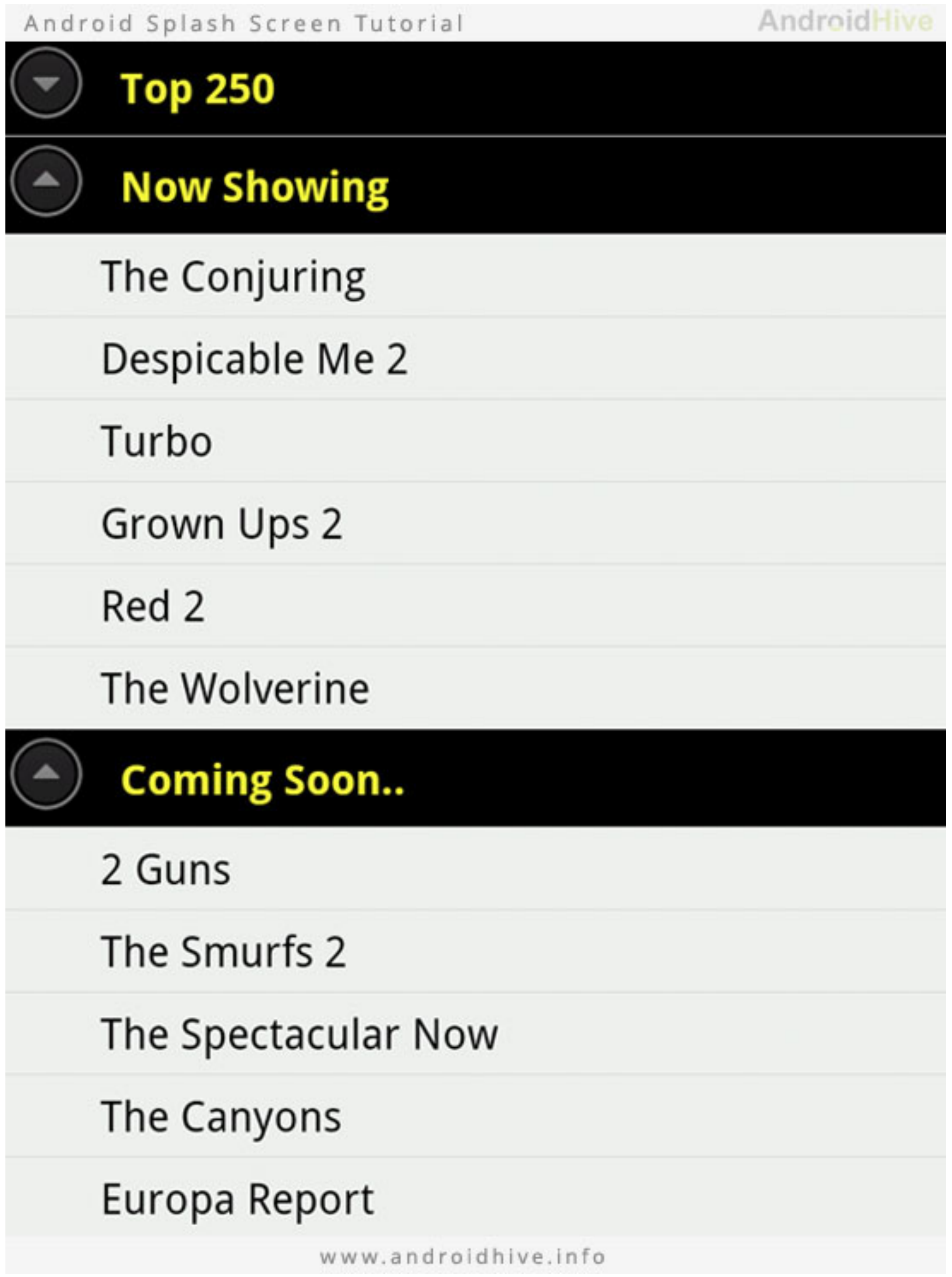
GRIDVIEW

GridView displays contain items in a two-dimensional scrolling grid.



EXPANDABLELISTVIEW

A ExpandableListView is an extension of a ListView. This type of container displays items in a vertically scrolling list that supports two levels

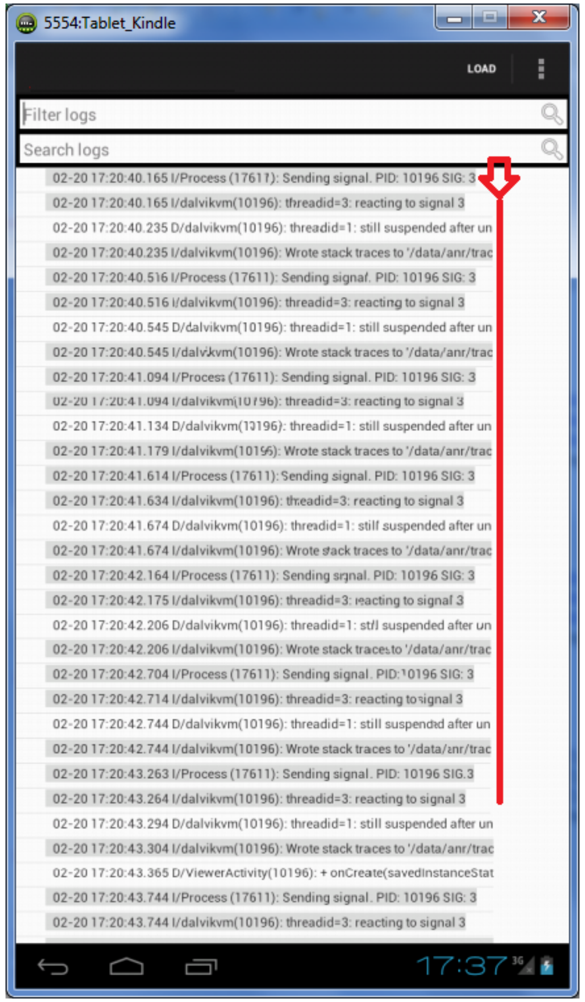


SCROLLVIEW AND HORIZONTALSCROLLVIEW

The ScrollView and the HorizontalScrollView are containers specifically designed for scrolling.

Both these containers are extensions of the FrameLayout.

Once a View has been placed in either of these containers, the view can be made scrollable.



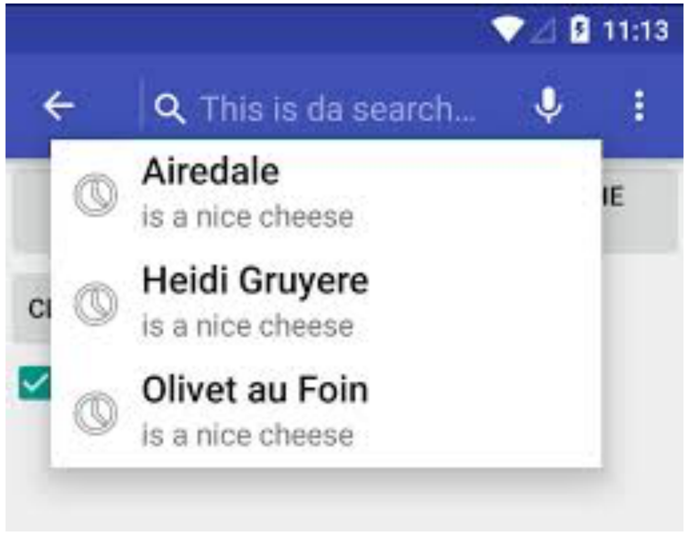
```
02-20 17:20:40.165 I/Process (17611): Sending signal. PID: 10196 SIG: 3
02-20 17:20:40.165 I/dalvikvm(10196): threadid=3: reacting to signal 3
02-20 17:20:40.235 D/dalvikvm(10196): threadid=1: still suspended after un
02-20 17:20:40.235 I/dalvikvm(10196): Wrote stack traces to '/data/anr/trac
02-20 17:20:40.516 I/Process (17611): Sending signal. PID: 10196 SIG: 3
02-20 17:20:40.516 I/dalvikvm(10196): threadid=3: reacting to signal 3
02-20 17:20:40.545 D/dalvikvm(10196): threadid=1: still suspended after un
02-20 17:20:40.545 I/dalvikvm(10196): Wrote stack traces to '/data/anr/trac
02-20 17:20:41.094 I/Process (17611): Sending signal. PID: 10196 SIG: 3
02-20 17:20:41.094 I/dalvikvm(10196): threadid=3: reacting to signal 3
02-20 17:20:41.134 D/dalvikvm(10196): threadid=1: still suspended after un
02-20 17:20:41.179 I/dalvikvm(10196): Wrote stack traces to '/data/anr/trac
02-20 17:20:41.614 I/Process (17611): Sending signal. PID: 10196 SIG: 3
02-20 17:20:41.634 I/dalvikvm(10196): threadid=3: reacting to signal 3
02-20 17:20:41.674 D/dalvikvm(10196): threadid=1: still suspended after un
02-20 17:20:41.674 I/dalvikvm(10196): Wrote stack traces to '/data/anr/trac
02-20 17:20:42.164 I/Process (17611): Sending signal. PID: 10196 SIG: 3
02-20 17:20:42.175 I/dalvikvm(10196): threadid=3: reacting to signal 3
02-20 17:20:42.206 D/dalvikvm(10196): threadid=1: still suspended after un
02-20 17:20:42.206 I/dalvikvm(10196): Wrote stack traces to '/data/anr/trac
02-20 17:20:42.704 I/Process (17611): Sending signal. PID: 10196 SIG: 3
02-20 17:20:42.714 I/dalvikvm(10196): threadid=3: reacting to signal 3
02-20 17:20:42.744 D/dalvikvm(10196): threadid=1: still suspended after un
02-20 17:20:42.744 I/dalvikvm(10196): Wrote stack traces to '/data/anr/trac
02-20 17:20:43.263 I/Process (17611): Sending signal. PID: 10196 SIG.3
02-20 17:20:43.264 I/dalvikvm(10196): threadid=3: reacting to signal 3
02-20 17:20:43.294 D/dalvikvm(10196): threadid=1: still suspended after un
02-20 17:20:43.304 I/dalvikvm(10196): Wrote stack traces to '/data/anr/trac
02-20 17:20:43.365 D/ViewerActivity(10196): + onCreate(savedInstanceStat
02-20 17:20:43.744 I/Process (17611): Sending signal. PID: 10196 SIG: 3
02-20 17:20:43.744 I/dalvikvm(10196): threadid=3: reacting to signal 3
```

SEARCHVIEW

The SearchView is typically added to the menu and provides an easy way to incorporate a standard search into the header of any activity.

The Android system controls all search events.

A SearchView object can be placed anywhere in your layout and will behave like a standard EditText View.

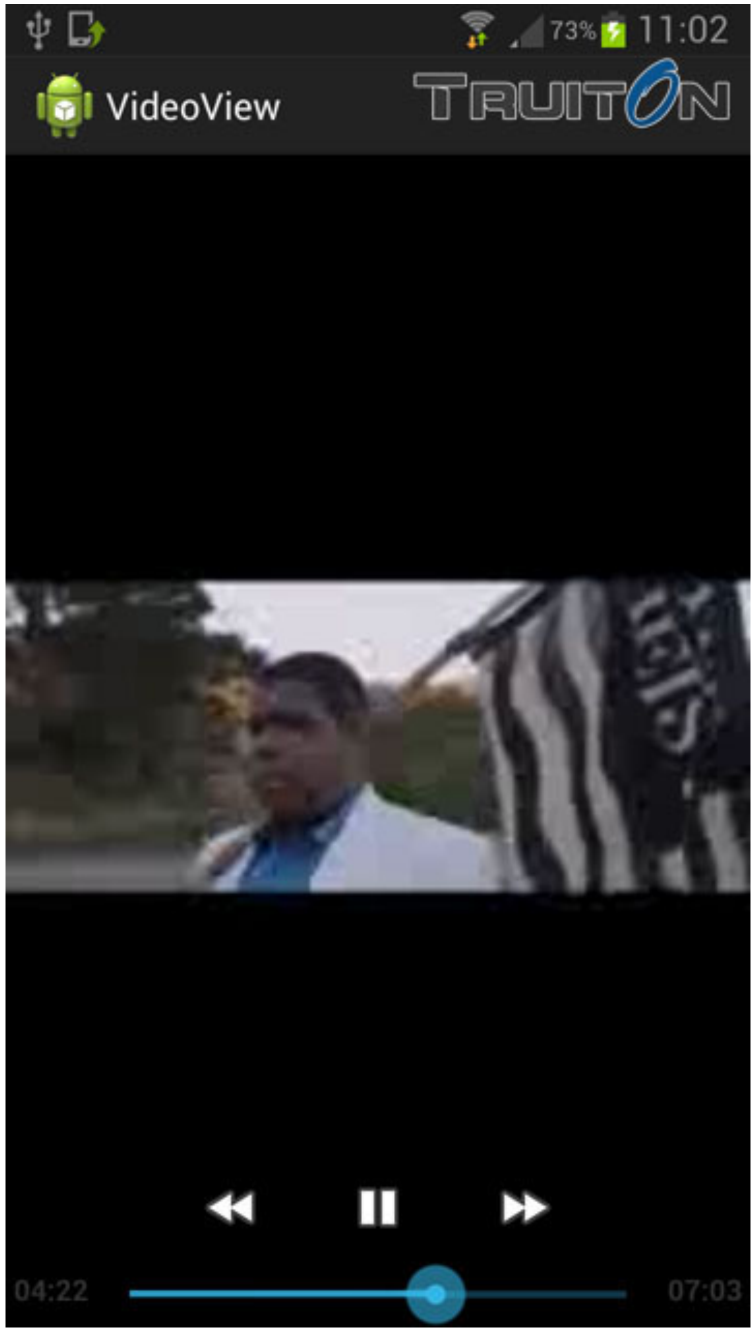


VIDEOVIEW

A VideoView is an extension of a **SurfaceView**.

Is used to display a video file.

This means that it can load images from various sources (such as resources or content providers), and provides various display options such as scaling and tinting.



USING AN ADAPTER

CS 175 Mobile Software Dev
by Dr. Angus Yeung

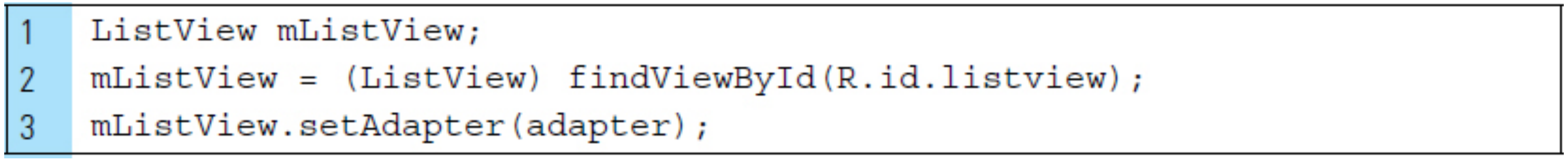
SIMPLE ADAPTER

In Android, an Adapter provides a common interface to the data model behind an AdapterView, such as a ListView object.

An Adapter is the control that is responsible for accessing the data to be supplied to a container widget and converting the individual elements of data into a specific.

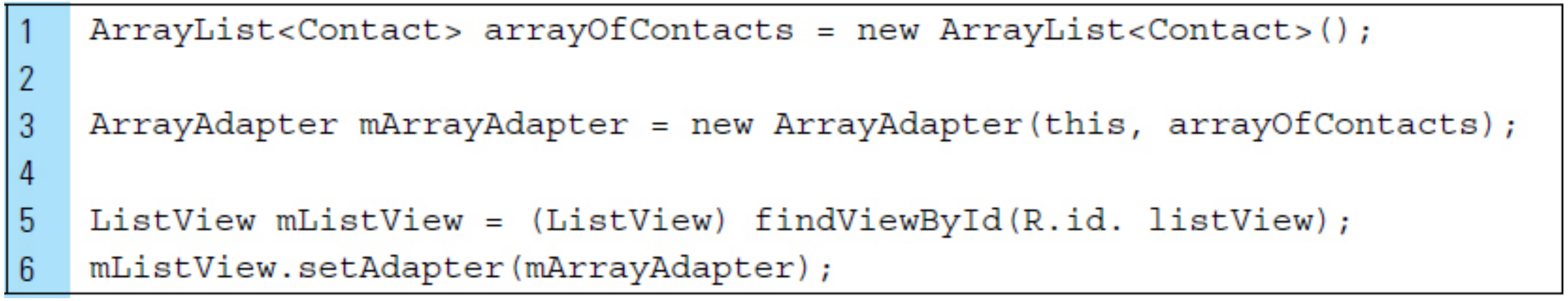
The Adapter behaves as a middleman between the data source and the AdapterView layout.

BIND AN ADAPTER TO LISTVIEW



```
ListView mListView;
mListView = (ListView) findViewById(R.id.listview);
mListView.setAdapter(adapter);
```

USING ARRAY ADAPTER



```java
ArrayList<Contact> arrayOfContacts = new ArrayList<Contact>();

ArrayAdapter mArrayAdapter = new ArrayAdapter(this, arrayOfContacts);

ListView mListView = (ListView) findViewById(R.id. listView);
mListView.setAdapter(mArrayAdapter);
```