Understanding User Interface in Android

Layouts

Android Screen UI Components

- the basic unit of an Android application is an Activity.
- An Activity displays the user interface of your application, which may contain widgets like buttons, labels, text boxes, etc.
- Typically, you define your UI using an XML file (for example, the main.xml file located in the res/layout folder)

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

Android Screen UI Components

- During runtime, you load the XML UI in the onCreate() event handler in your Activity class, using the setContentView() method of the Activity class:

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

Views and ViewGroups

- An Activity contains Views and ViewGroups.
- A View is a widget that has an appearance on screen. Examples of widgets are buttons, labels, text boxes, etc. A View derives from the base class android.view.View.
- One or more Views can be grouped together into a ViewGroup. A ViewGroup (which is by itself is a special type of View) provides the layout in which you can order the appearance and sequence of views.
- Examples of Viewgroups are LinearLayout, FrameLayout, etc. A ViewGroup derives from the base class android.view.ViewGroup.

Views and ViewGroups

- Android supports the following ViewGroups:
  - LinearLayout
  - AbsoluteLayout
  - TableLayout
  - RelativeLayout
  - FrameLayout
  - ScrollView

LinearLayout

- The LinearLayout arranges views in a single column or single row. Child views can either be arranged vertically or horizontally.
- let's modify the main.xml

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

- In the main.xml file, observe that the root element is <LinearLayout> and it has a <TextView> element contained within it. The <LinearLayout> element controls the order in which the views contained within it appear.

Views and ViewGroups attributes

- layout_width
  Specifies the width of the View or ViewGroup
- layout_height
  Specifies the height of the View or ViewGroup
- layout_marginTop
  Specifies extra space on the top side of the View or ViewGroup
- layout_marginBottom
  Specifies extra space on the bottom side of the View or ViewGroup
- layout_marginLeft
  Specifies extra space on the left side of the View or ViewGroup
- layout_marginRight
  Specifies extra space on the right side of the View or ViewGroup
- layout_gravity
  Specifies how child Views are positioned
- layout_weight
  Specifies how much of the extra space in the layout to be allocated to the View
- layout_x
  Specifies the x-coordinate of the View or ViewGroup
- layout_y
  Specifies the y-coordinate of the View or ViewGroup

LinearLayout

- For example, the <TextView> element above has its width filling up the entire width of its parent (which is the screen in this case) using the fill_parent constant. Its height is indicated by the wrap_content constant, which means that its height is the height of its content (in this case, the text contained within it).

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

- set the width to an absolute value

```
<TextView
   android:layout_width="105px"
   android:layout_height="wrap_content"
   android:text="@string/hello"
   />
```

Modify the main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   xmlns:android="http://schemas.android.com/apk/res/android"
   >
   <TextView
      android:layout_width="105px"
      android:layout_height="wrap_content"
      android:text="@string/hello"
      />
   <Button
      android:layout_width="100px"
      android:layout_height="wrap_content"
      android:text="Button"
      />
</LinearLayout>
```

LinearLayout

- The default orientation of LinearLayout is set to horizontal.
- If you want to change its orientation to vertical, set the orientation attribute to vertical

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

LinearLayout

- In LinearLayout, you can apply the **layout_weight** and **layout_gravity** attributes to views contained within it

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="vertical"
   >
   <TextView
      android:layout_width="105px"
      android:layout_height="wrap_content"
      android:text="@string/hello"
      />
   <Button
      android:layout_width="100px"
      android:layout_height="wrap_content"
      android:text="Button"
      android:layout_gravity="right"
      android:layout_weight="0.2"
      />
   <EditText
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:textSize="18sp"
      android:layout_weight="0.8"
      />
</LinearLayout>
```

LinearLayout

- the button is aligned to the right of its parent (which is the LinearLayout) using the layout_gravity attribute. At the same time, you use the layout_weight attribute to specify the ratio in which the Button and EditText views occupy the remaining space on the screen. The total value for the layout_weight attribute must be equal to 1.

AbsoluteLayout

- The AbsoluteLayout lets you specify the exact location of its children. Consider the following UI defined in main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   xmlns:android="http://schemas.android.com/apk/res/android"
   >
   <Button
      android:layout_width="188px"
      android:layout_height="wrap_content"
      android:text="Button"
      android:layout_x="126px"
      android:layout_y="361px"
      />
   <Button
      android:layout_width="113px"
      android:layout_height="wrap_content"
      android:text="Button"
      android:layout_x="12px"
      android:layout_y="361px"
      />
</AbsoluteLayout>
```

- the two Button views located at their specified positions using the android_layout_x and android_layout_y attributes

TableLayout

- The TableLayout groups views into rows and columns.
- You use the <TableRow> element to designate a row in the table.
- Each row can contain one or more views. Each view you place within a row forms a cell.
- The width for each column is determined by the largest width of each cell in that column.

TableLayout

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout

    xmlns:android="http://schemas.andr
    oid.com/apk/res/android"
   android:layout_height="fill_parent"
   android:layout_width="fill_parent"
   android:background="#000044">
   <TableRow>
      <TextView
         android:text="User Name:"
         android:width ="120px"
         />
      <EditText
         android:id="@+id/txtUserName"
         android:width="200px" />
   </TableRow>
```

```
<TableRow>
      <TextView
         android:text="Password:"
         />
      <EditText
         android:id="@+id/txtPassword"
         android:password="true"
         />
   </TableRow>
   <TableRow>
      <TextView />
      <CheckBox android:id="@+id/chkRememberPassword"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="Remember Password"
         />
   </TableRow>
   <TableRow>
      <Button
         android:id="@+id/buttonSignIn"
         android:text="Log In" />
   </TableRow>
</TableLayout>
```

TableLayout

TableLayout

- Note that in the above example, there are two columns and four rows in the TableLayout. The cell directly under the Password TextView is populated with an empty element. If you don't do this, the Remember Password checkbox will then appear under the Password TextView

RelativeLayout

- The RelativeLayout lets you specify how child views are positioned relative to each other. Consider the following main.xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
   android:id="@+id/RLayout"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   xmlns:android="http://schemas.android.com/apk/res/android"
   >
   <TextView
      android:id="@+id/lblComments"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Comments"
      android:layout_alignParentTop="true"
      android:layout_alignParentLeft="true"
      />
   <EditText
      android:id="@+id/txtComments"
      android:layout_width="fill_parent"
      android:layout_height="170px"
      android:textSize="18sp"
      android:layout_alignLeft="@+id/lblComments"
      android:layout_below="@+id/lblComments"
      android:layout_centerHorizontal="true"
      />
```

```
<Button
      android:id="@+id/btnSave"
      android:layout_width="125px"
      android:layout_height="wrap_content"
      android:text="Save"
      android:layout_below="@+id/txtComments"
      android:layout_alignRight="@+id/txtComments"
      />
   <Button
      android:id="@+id/btnCancel"
      android:layout_width="124px"
      android:layout_height="wrap_content"
      android:text="Cancel"
      android:layout_below="@+id/txtComments"
      android:layout_alignLeft="@+id/txtComments"
      />
</RelativeLayout>
```

RelativeLayout

- Notice that each view embedded within the RelativeLayout have attributes that allow them to align with another view. These attributes are:
  - layout_alignParentTop
  - layout_alignParentLeft
  - layout_alignLeft
  - layout_alignRight
  - layout_below
  - layout_centerHorizontal

FrameLayout

- The FrameLayout is a placeholder on screen that you can use to display a single view. Views that you add to a FrameLayout is always anchored to the top left of the layout. Consider the following content in main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
   android:id="@+id/widget68"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   xmlns:android="http://schemas.android.com/apk/res/android"
   >
   <FrameLayout
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_x="40px"
      android:layout_y="35px"
      >
      <ImageView
         android:src = "@drawable/androidlogo"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         />
   </FrameLayout>
</AbsoluteLayout>
```

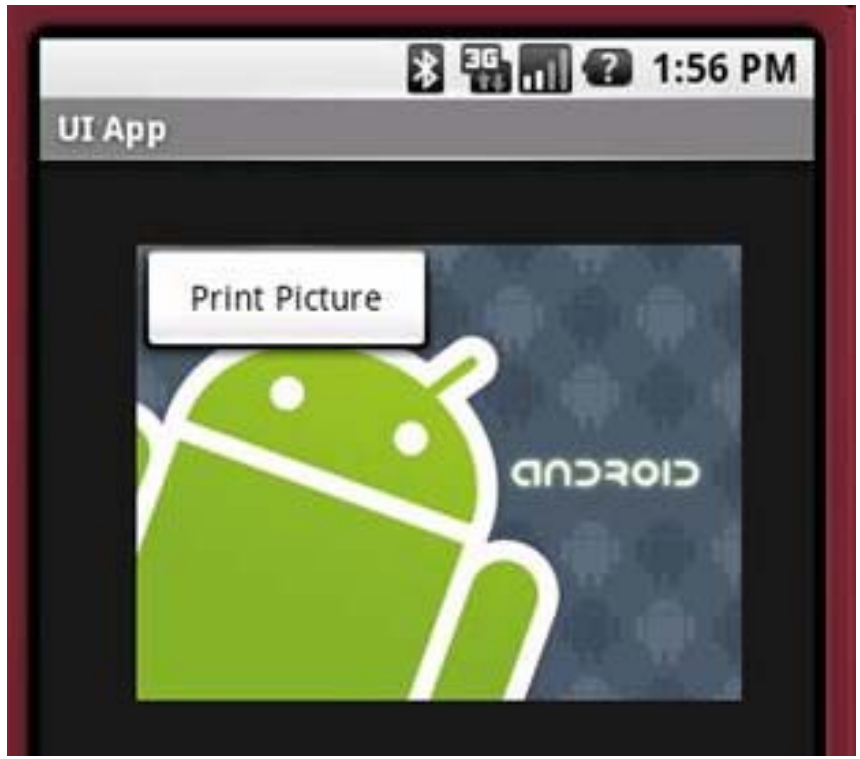
FrameLayout

- Here, you have a FrameLayout within an AbsoluteLayout. Within the FrameLayout, you embed an ImageView view.
- Note: This example assumes that the res/drawable folder has an image named androidlogo.png.

FrameLayout

- If you add another view (such as a Button view) within the FrameLayout, the view will overlap the previous view

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
   android:id="@+id/widget68"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   xmlns:android="http://schemas.android.com/apk/res/android"
   >
   <FrameLayout
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_x="40px"
      android:layout_y="35px"
      >
      <ImageView
         android:src = "@drawable/androidlogo"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         />
      <Button
         android:layout_width="124px"
         android:layout_height="wrap_content"
         android:text="Print Picture"
         />
   </FrameLayout>
</AbsoluteLayout>
```



ScrollView

- A ScrollView is a special type of FrameLayout in that it allows users to scroll through a list of views that occupy more space than the physical display. The ScrollView can contain only one child view or ViewGroup, which normally is a LinearLayout.

- Note: Do not use a ListView together with the ScrollView. The ListView is designed for showing a list of related information and is optimized for dealing with large lists.

ScrollView

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
   android:id="@+id/widget54"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"

      xmlns:android="http://schemas.android.com/apk/res/and
      roid"
   >
   <LinearLayout
      android:layout_width="310px"
      android:layout_height="wrap_content"
      android:orientation="vertical"
      >
      <Button
         android:id="@+id/button1"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="Button 1"
         />
      <Button
         android:id="@+id/button2"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="Button 2"
         />
```

```
<Button
         android:id="@+id/button3"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="Button 3"
         />
      <EditText
         android:id="@+id/txt"
         android:layout_width="fill_parent"
         android:layout_height="300px"
         />
      <Button
         android:id="@+id/button4"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="Button 4"
         />
      <Button
         android:id="@+id/button5"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="Button 5"
         />
   </LinearLayout>
</ScrollView>
```

ScrollView


Android Emulator (5554)
1:56 PM
UI App
Button 1
Button 2
Button 3
MENU
Android Emulator (5554)
1:56 PM
UI App
Button 4
Button 5
MENU