Android App Development

User Interface

Sabih Agbaria
View

- View is the base class of UI components.
- View occupies a rectangular area, and is responsible for it.
- ViewGroup is a View subclass, and it’s a base class for layouts
- Example of View method: setVisibility()
**ViewGroup**

- ViewGroup is a container that holds other Views
- Views are arranged in a **tree**, with parent-child relationship
- Each ViewGroup type lays out its children in a specific way
Common Views

- **TextView** displays a formatted text label
- **ImageView** displays an image resource
- **Button** can be clicked to perform an action
- **ImageButton** displays a clickable image
- **EditText** is an editable text field for user input
- **ListView** is a scrollable list of items containing other views
Building the layout (View Tree)

Two ways:

1. Declare UI elements in XML, Benefits: Separation between logic and gui, simplicity and visual viewer

   <Button android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="press me"
          android:layout_centerHorizontal="true" />

2. Instantiate UI at runtime

   ViewGroup rootLayout = (ViewGroup) findViewById(R.id. rootLayout);
   Button testButton = new Button(this);
   testButton.setText("test me");
   rootLayout.addView(testButton);
Let's add buttons using the two ways

Notice that in order to see the effect of UI added in code we need to run the app.

while in XML, it is enough to refresh the design viewer.
**View Properties**

- View properties can be given as XML element attribute or set in Java code
- **id**: to uniquely identify the View within the tree
- mandatory view properties: `layout_width` and `layout_height`, possible values:
  - **match_parent**: view should be as big as its parent
  - **wrap_content**: view should be only big enough to enclose its content
  - numerical dp.
MARGIN

A space between views.
Padding

A space between view’s border and its content.
App Resources

- Separate static data and application resources from code.
  - Easier to change
  - Enables dynamic resource loading (will see later)
- Resource types:
  - Graphics: saved in res/drawable/
  - Layouts: saved in res/layout/
  - Strings: saved in res/values/strings.xml
  - Colors: saved in res/values/colors.xml

- Access from xml: @[resource type]/name
- Access from code: R.[resource type].name
Let’s use some resources

- String resources for buttons
- color resource
- some images resources
Let's Add Event Listeners

- same onClick handler
- use nested layout
More Event Listeners

- Use SeekBar
- Add Listener in code
- show Log.x
Common Layouts

Each layout provides a unique way to display views you nest within it.

Linear Layout

Relative Layout

places children objects relative to each other or relative to parent
Linear Layout

- single direction: horizontal or vertical
- supports weight property for children
  - “importance” of the child
  - extra space are divided between children proportional to their weight.
LINEAR LAYOUT DEMO