Introduce Layout in Android

Dac Hoang Nguyen

Introduction

- Your role
- Your background and experience in the subject:
  - Java.
  - Basic Android.
- What do you want from this course

Course Objectives

- At the end of the course, you will have acquired sufficient knowledge to:
  - Understand Android Layout.
  - Understand Android Fragment.
  - Be able to design and implement layout in Android.

Agenda

I. Android Layout

II. Android Fragment

Set Up Environment

- To complete the course, your PC must install:
  - Eclipse with Android plugins
  - Android SDK

Course Administration

- In order to complete the course you must:
  - Sign in the Class Attendance List
  - Participate in the course
  - Provide your feedback in the End of Course Evaluation

Assessment Disciplines

- Class Participation: 40%
- Assignment: 60%
- Final Exam: 0%
- Passing Scores: 70%

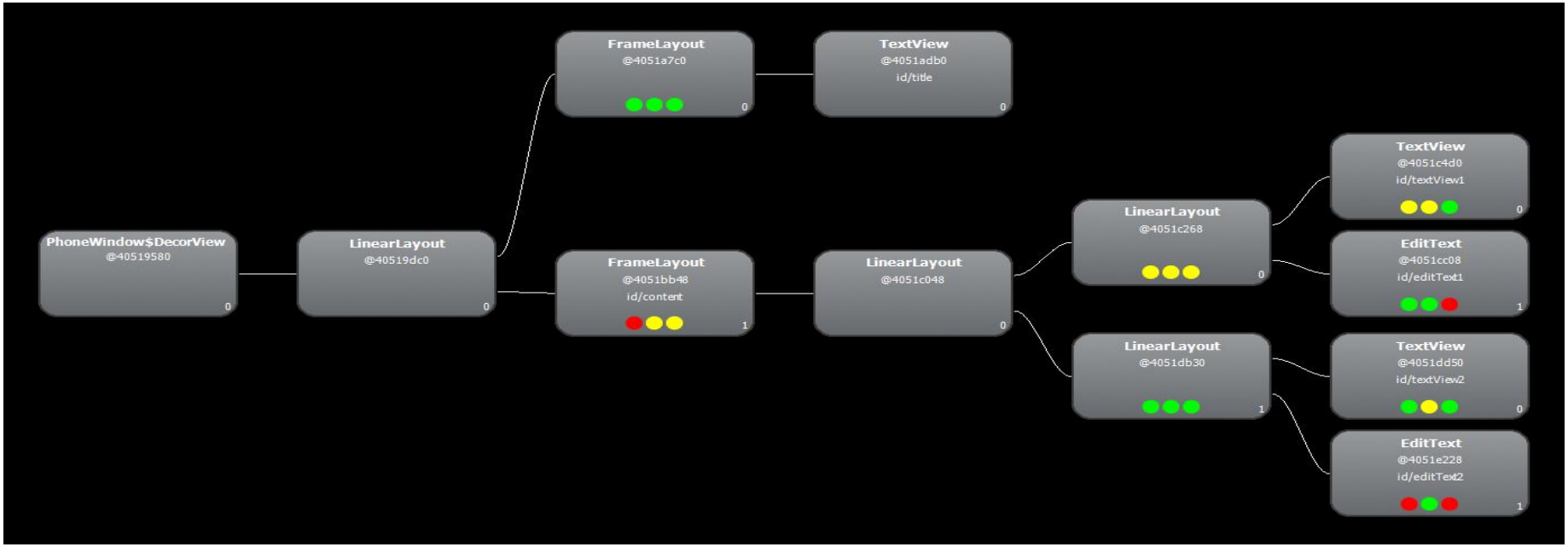
Android Layout

Android Layout

- A layout defines the visual structure for a user interface
- Declare a layout in two ways:
  - **Declare UI elements in XML:**
    - Android separated UI in the text file.
    - XML layout is human-readable (intuitive).
  - **Instantiate layout elements at runtime**
- Layout, View, Widgets can be accessed by the application using *findViewById(…).*
- View's attribute can be set in design, XML text, Java code.

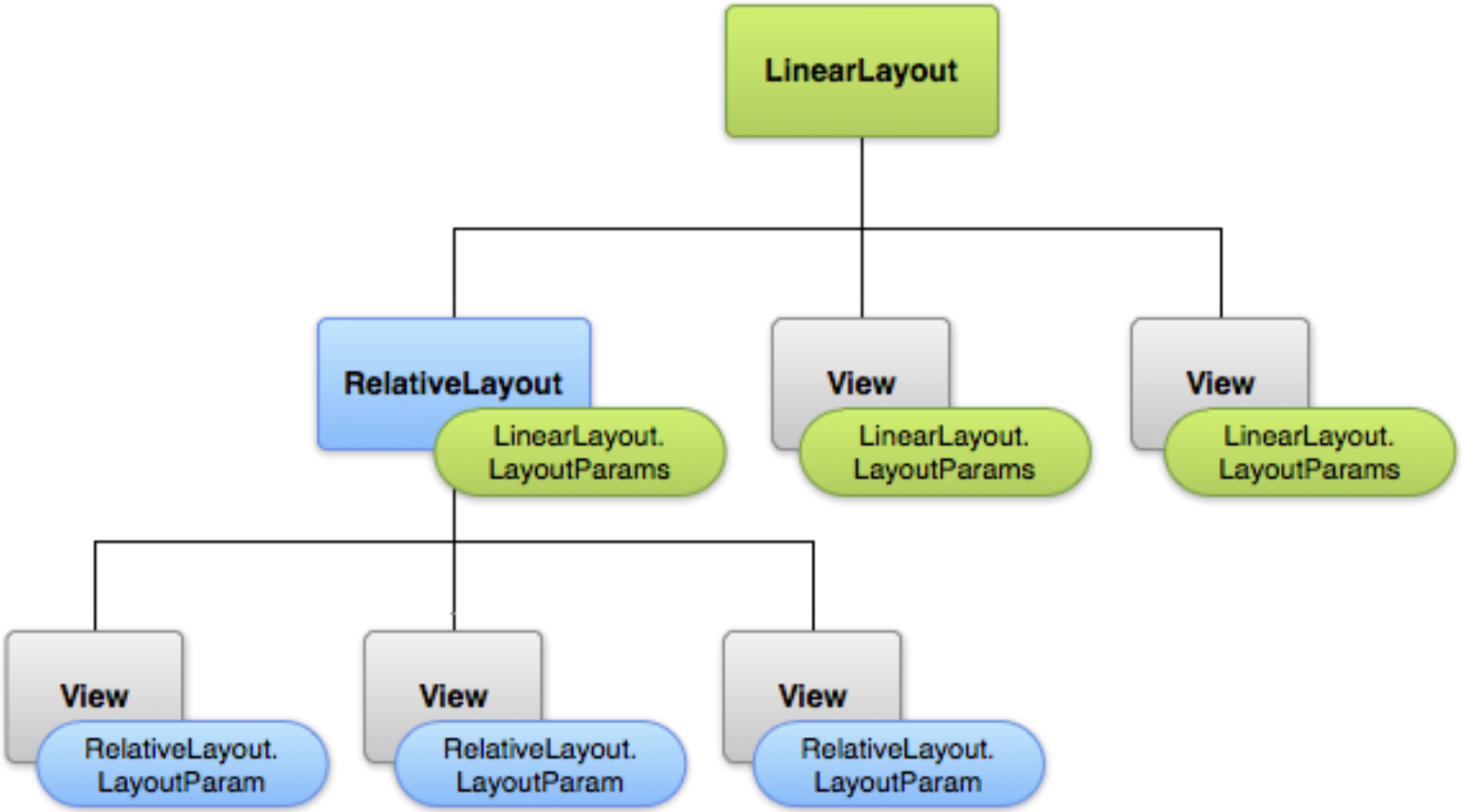
Android Layout

- The Android UI Framework paints the screen by walking the View tree and ask each component draws it self and then, in turn, each component ask its children to do the same.
- To view the tree represented for layout: Run a application and select <sdk>/tools/ hierarchyviewer.bat

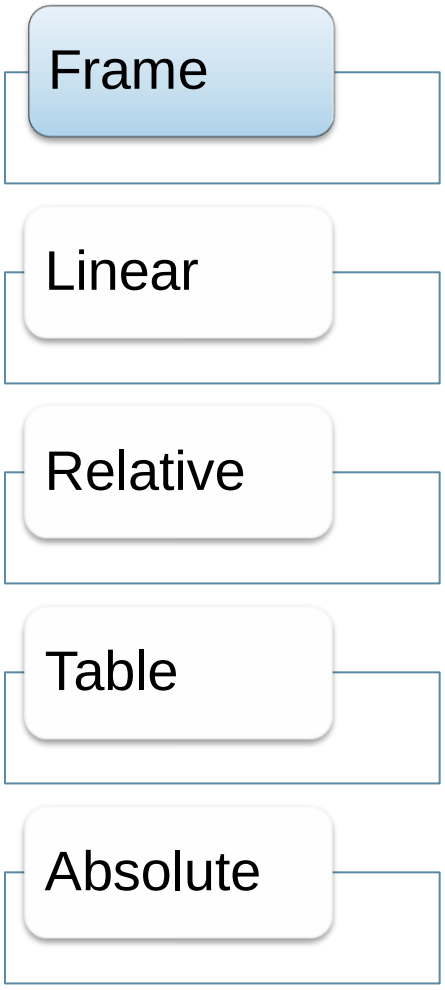


Android Layout

- Example:

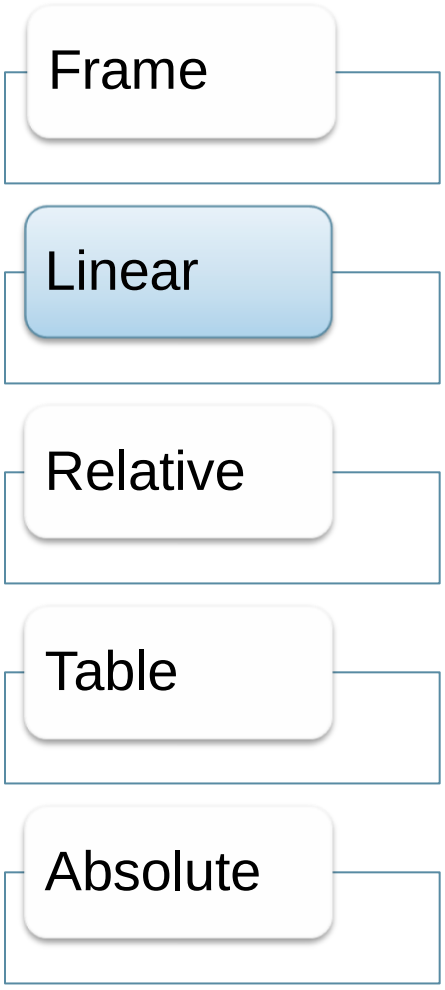


Android Layout: Common layout



- Frame is the simplest type of layout. Basically, it's a blank space on screen that we can fill with single object.
- One frame can just hold a single View object, if it has more than 1 View object, it will display just a latest View.
- Use: Display Image, custom view….

Android Layout: Common layout

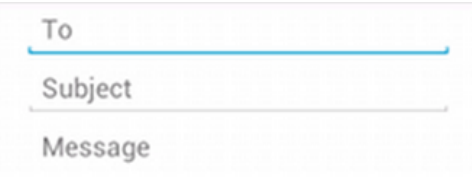


- Linear layout aligns all children in a single direction (vertical or horizontal), set in orientation attribute.
- Common attribute:
  - Margins: Specifies extra space on the left, top, right and bottom sides of layout.
  - Gravity: Specifies how to place the content of an object, both on the x- and y-axis, within the object itself.
  - Weight: gives an "importance" value to a view, and allows it to expand to fill any remaining space in the parent view.
  - Padding: set whitespace between widgets.
  - Orientation: vertical or horizontal.

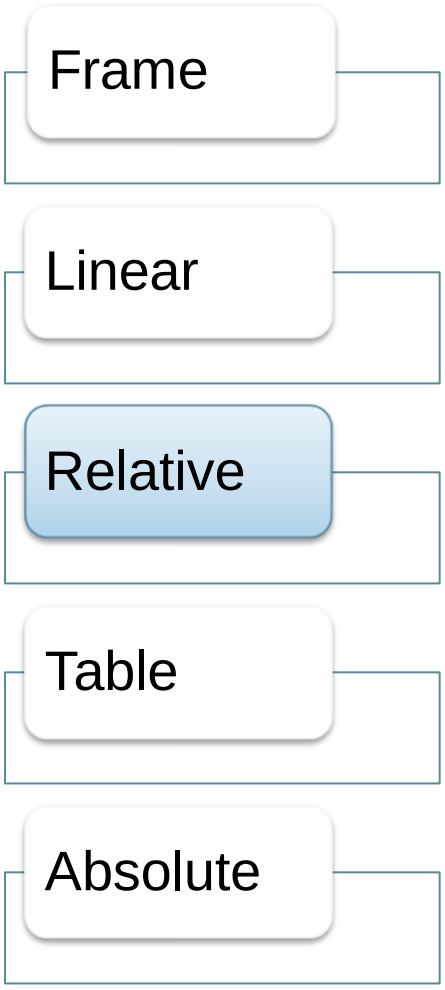
Android Layout: Common layout

- Frame
- Linear
- Relative
- Table
- Absolute

```
<LinearLayout xmlns:android="http://schemas.android.com/
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
```



Android Layout: Common layout

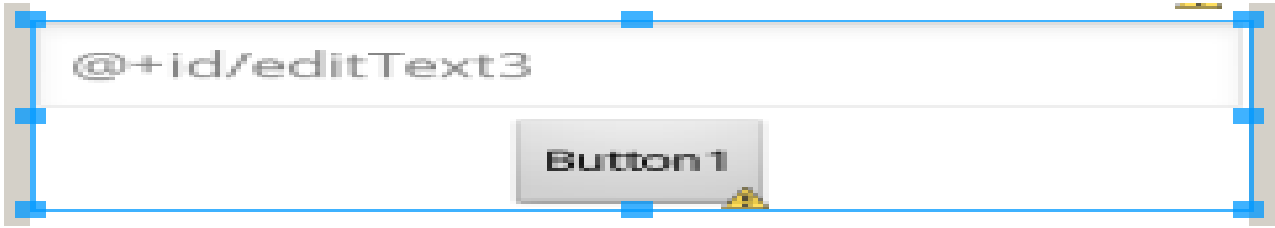


- Relative Layout lets child views specify their *position relative to the parent view or to each other.*
- It can eliminate nested view groups and keep your layout hierarchy flat, which improves performance

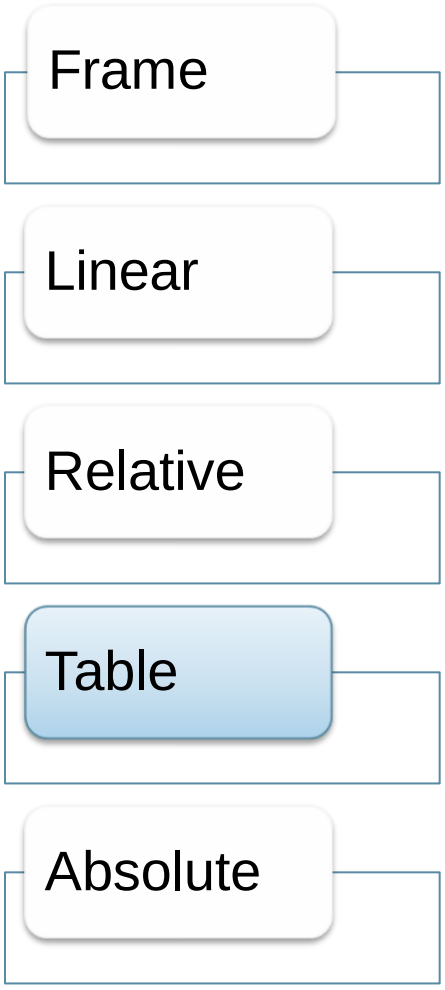
Android Layout: Common layout

- Frame
- Linear
- Relative
- Table
- Absolute

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <EditText
        android:id="@+id/editText3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@+id/editText3"
        android:ems="10" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText3"
        android:layout_centerHorizontal="true"
        android:text="Button1" />

</RelativeLayout>
```



Android Layout: Common layout

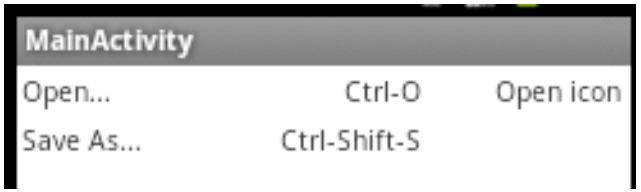


- Table layout positions its children into rows and columns.
- Table layout don’t display border lines and its cell cannot span columns.
- Table layout content TableRow, each TableRow defines a single row in the table.
- The table will have as many columns as the row with the most cells.

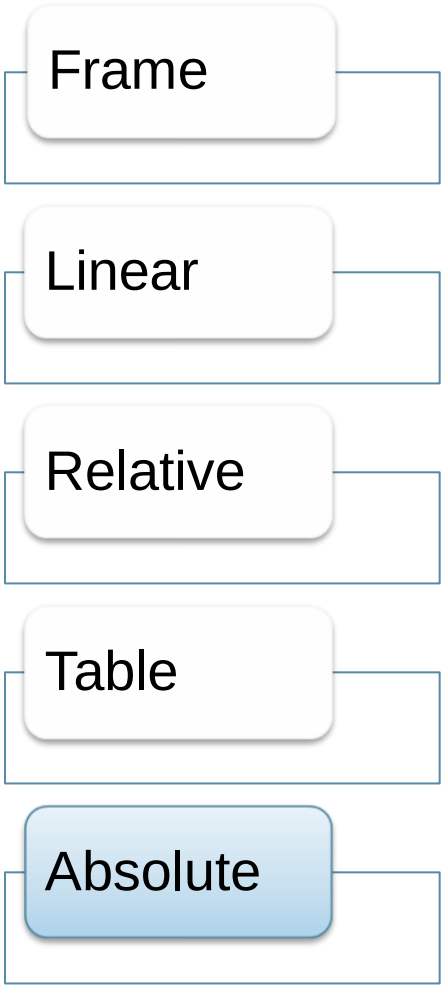
Android Layout: Common layout

- Frame
- Linear
- Relative
- Table
- Absolute

```
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:stretchColumns="*">
<TableRow>
<TextView
    android:text="Open…"
    android:padding="3dip" />
<TextView
    android:text="Ctrl-O"
    android:gravity="right"
    android:padding="3dip" />
<TextView
    android:text="Open icon"
    android:gravity="right"
    android:padding="3dip" />
</TableRow>
<TableRow>
<TextView
    android:text="Save As…"
    android:padding="3dip" />
<TextView
    android:text="Ctrl-Shift-S"
    android:gravity="right"
    android:padding="3dip" />
</TableRow>

</TableLayout>
```



Android Layout: Common layout

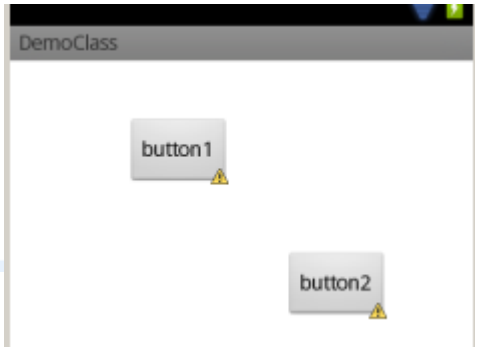


- Absolute layout allow we specify exact locations (x/y coordinates) of its children.
- Less flexible and hard to maintain.

Android Layout: Common layout

- Frame
- Linear
- Relative
- Table
- Absolute

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

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="81dp"
        android:layout_y="39dp"
        android:text="button1" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="190dp"
        android:layout_y="131dp"
        android:text="button2" />

</AbsoluteLayout>
```



Android Layout: Demo and Pracetice

- Demo and practice

Android Fragment

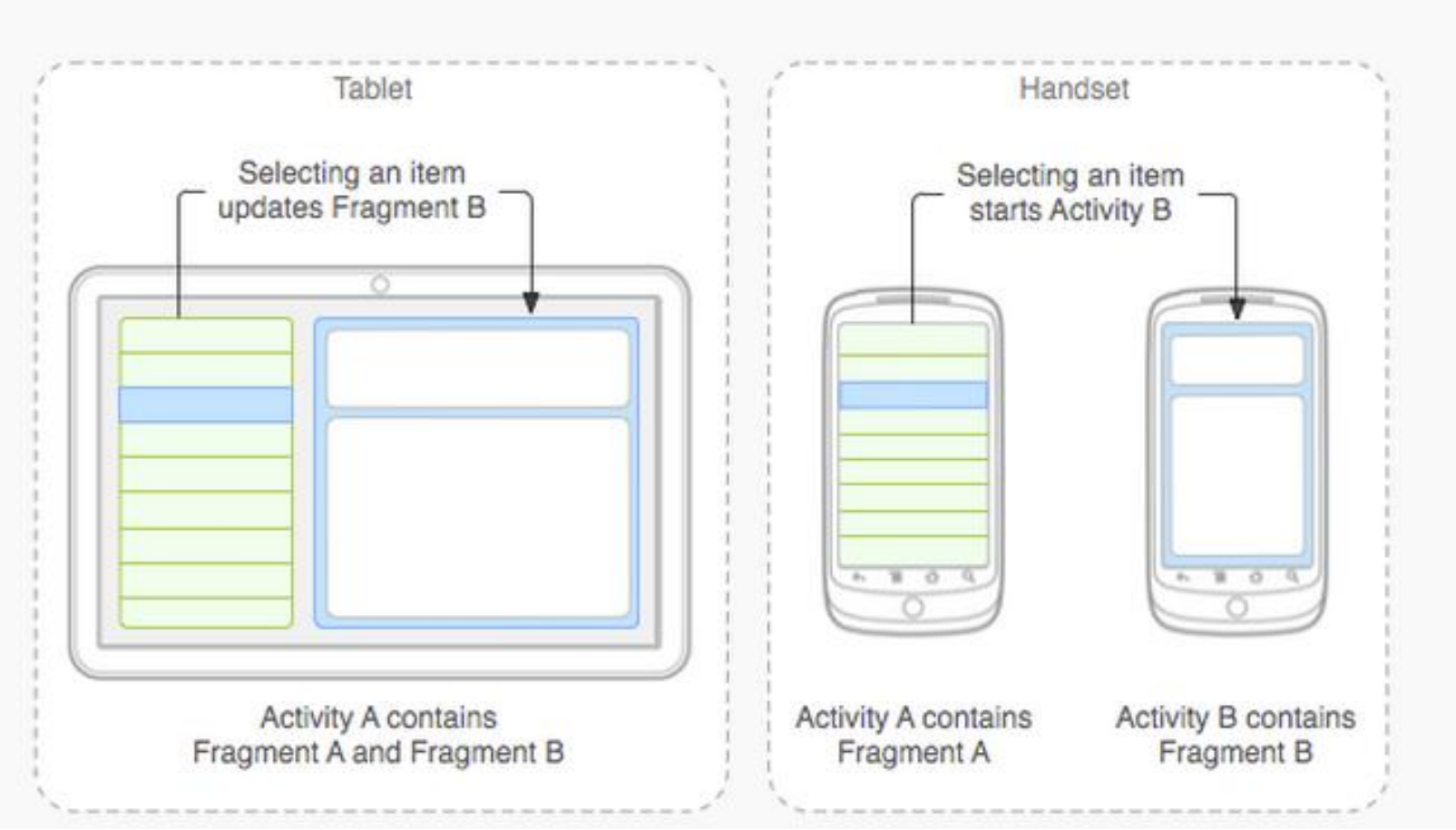
Introduce Fragment

- Activity is a container for views. With large screen device like tablet, the UI smart phone is too simple.
- At Android 3.0 (Honeycomb), developer can divide an Activity into sub-Activity call Fragment.
- You can think the fragment like a mini Activity which can be embedded in an Activity and can reuse in other Activity.

Properties of Fragment

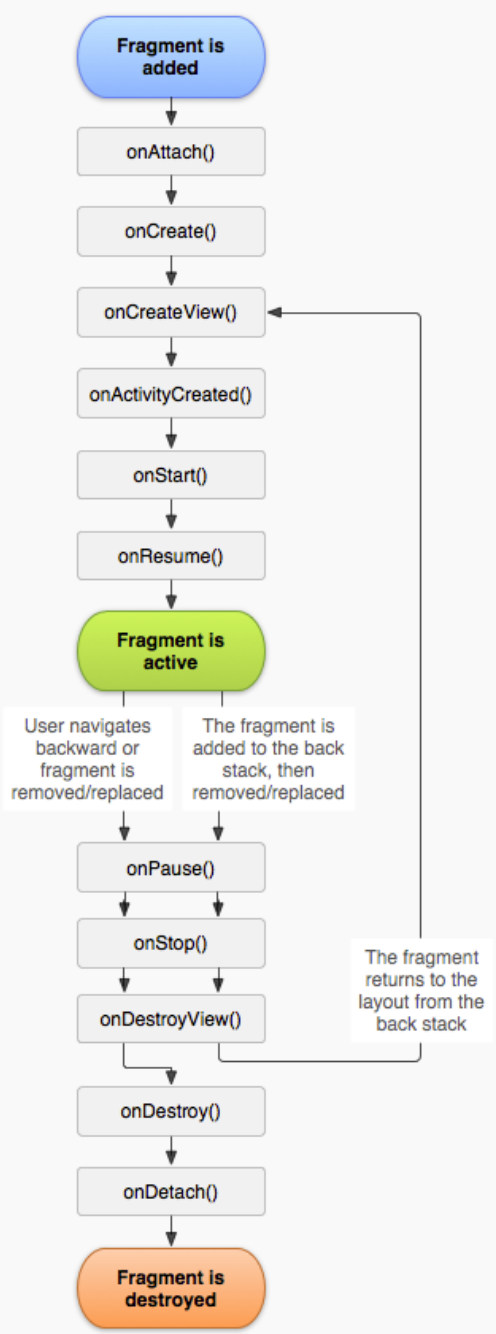
- Each fragment has its own set of views, own lifecycle and receives its own input events.
- Activity can contain one or many fragments. Developer can combine multiple fragments to build a multi-pane UI. Fragment also can be reused in multiple Activities.
- Can control fragment base on device type (phone or tablet) or base on orientation.
- Fragment's lifecycle is affected by the host Activity's lifecycle. For example: if activity is paused, all fragment in this activity is also paused.
- Fragments can be invisible.
- Fragment can force host Activity to implement specific callback methods. Host Activity can get and execute its fragment's methods.

Fragment design.



Fragment Lifecycle

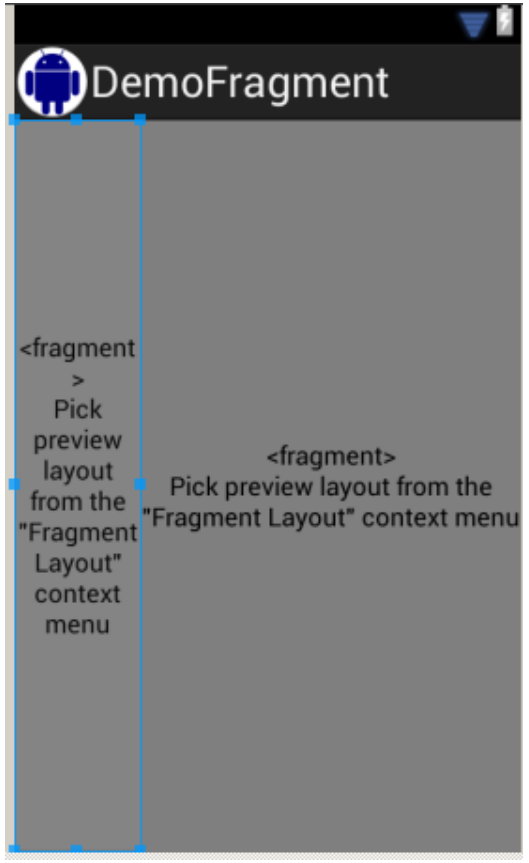
- Common lifecycle methods:
  - onCreate: called when system creating the fragment. Initialize essential components.
  - onCreateView: called when system draw views for Fragment. Return the root View of fragment layout, also can return null if fragment doesn't have UI.
  - onPause: called when user leaving the fragment. Use to commit changes of current user session.



Fragment Layout with XML:

- Each fragment have its own XML layout file and defined in onCreateView of Fragment class
- Main Activity layout contain fragment with specific fragment class.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="horizontal"
         >
  <fragment
        android:id="@+id/fragmentList"
        android:name="com.demo.demofragment.ListFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="3"
         />

    <fragment
        android:id="@+id/fragmentDetail"
        android:name="com.demo.demofragment.DetailFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
         />

 </LinearLayout>
```



Fragment class

- Define view and view's behavior.

```
public class ListFragment extends Fragment {


  @Override
  public void onCreate(Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
  }

  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
      Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    View v = inflater.inflate(R.layout.list_fragment, container, false);
    return v;
  }


}
```

Fragment demo and practice

- Demo and practice

Q&A

Thank You


CSC
BUSINESS SOLUTIONS
TECHNOLOGY
OUTSOURCING