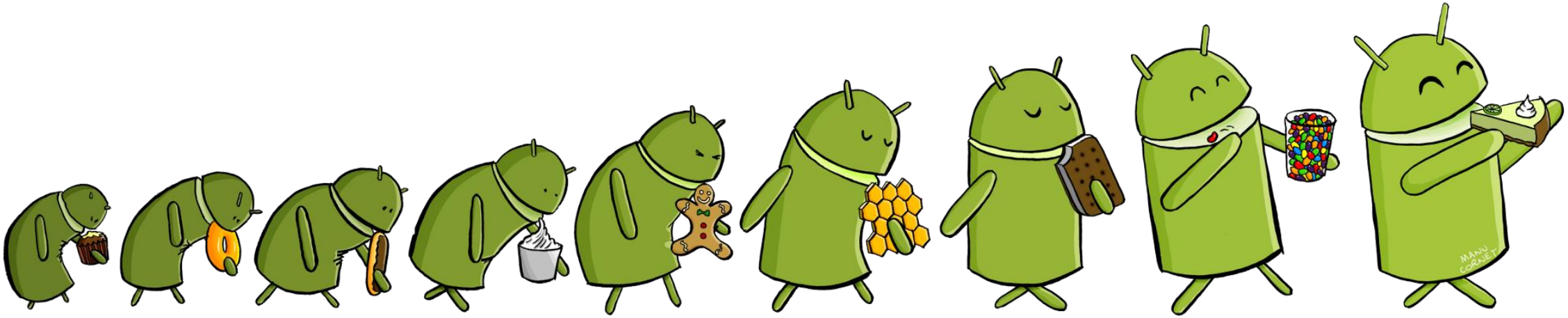


In the name of Allah

Layouts (part one)

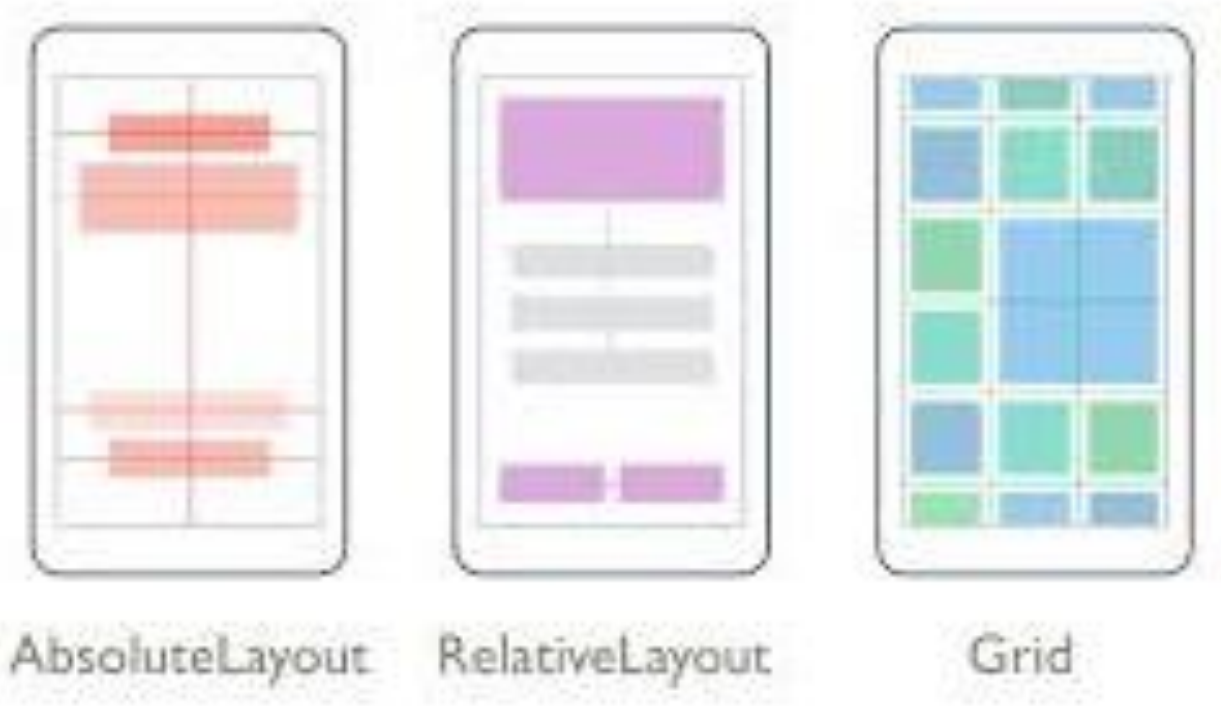
Lecture #07

Subject: Mobile App Development
Instructor: Ehsan Hasin {ehsanhasin@gmail.com}
Date: Sunday, April 21, 2024

Table of Content

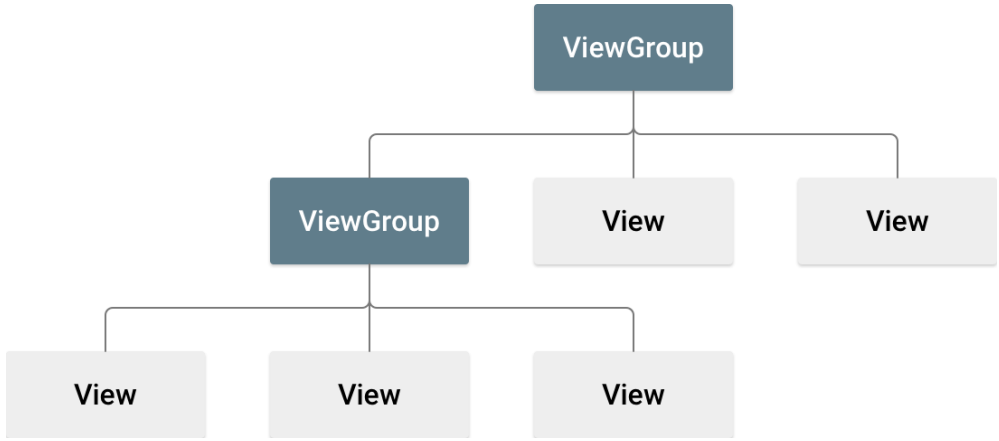
What is a Layout?

- A layout defines the **structure** for a user interface in your app, such as in an activity.
- A Layout is a special type of view that acts as a **container** for other views or layouts.



Structure of a Layouts

- Each Layout is a subclass of **ViewGroup** class, which derives from **View** class.
- All elements in the layout are built using a hierarchy of View and ViewGroup objects.
- A **View** usually draws something the user can see and interact with.
- Whereas a **ViewGroup** is an invisible container that defines the layout structure for View and other ViewGroup objects, as shown in figure.



Layout – Creation

- You can declare a layout in two ways:
  - **Declare UI elements in XML**
    - XML
  - **Instantiate layout elements at runtime**.
    - JAVA

Layout – Write the XML

- Using Android's XML vocabulary, you can quickly design UI layouts and the screen elements they contain.
- Each layout file must contain exactly one **root element**, which must be a View or ViewGroup object. Once you've defined the root element, you can add additional layout objects or widgets as **child elements** to gradually build a View hierarchy that defines your layout.
- **For example**, here's an XML layout that uses a vertical LinearLayout to hold a TextView and a Button:

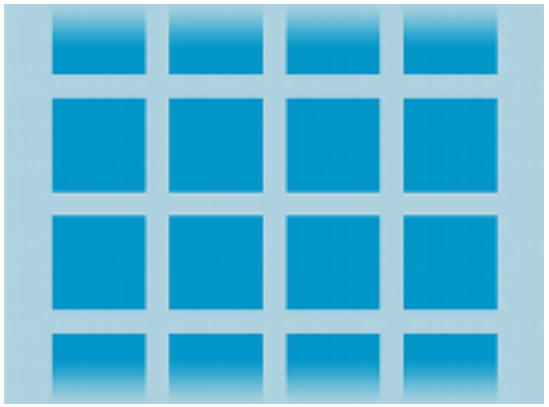
```
<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, 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>
```

Android Layout Types

- There are number of Layouts provided by Android which you will use in almost all the Android applications to provide different view, look and feel.

1. ***Android Linear Layout***
2. ***Android Constraint Layout***
3. ***Android Frame Layout***
4. ***Android Table Layout***
5. ***Android List View***

Linear Layout



Grid Layout

List Layout

Relative layout

Basic Attributes

- There are some basic attributes which can be used commonly in every view.
  - Attributes for **sizing**
  - Attributes for **Margin**
  - Attributes for **Padding**
  - Attributes for **Gravity**

Basic Attributes – Sizing

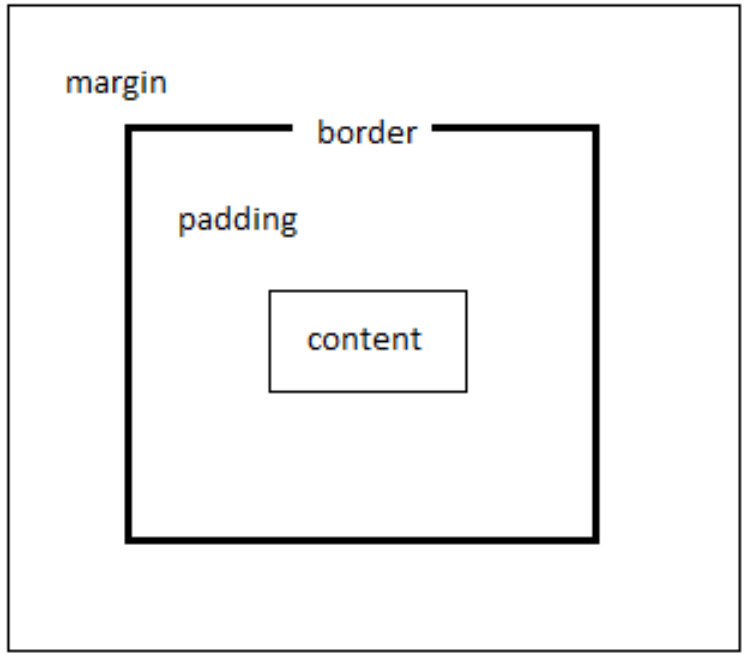
- android:**layout_height**:
  - Specifies the basic **height** of the view and is **required** for any view.
  - Its **value** may be
    1. a *dimension* (such as "12dip") for a constant height.
       - If it is a dimension value then available units are: **px** (pixels), **dp** (density-independent pixels), **sp** (scaled pixels based on preferred font size), **in** (inches), and **mm** (millimeters).
    2. or one of the special *constants*.
       - **match_parent**: The view should be as big as its parent (minus padding).
       - **wrap_content**: The view should be only big enough to enclose its content (plus padding).

Cont.

- android:**layout_width**:
  - Specifies the basic **width** of the view and is **required** for any view.
  - Its **value** may be
    1. a *dimension* (such as "12dip") for a constant height.
       - If it is a dimension value then available units are: **px** (pixels), **dp** (density-independent pixels), **sp** (scaled pixels based on preferred font size), **in** (inches), and **mm** (millimeters).
    2. or one of the special *constants*.
       - **match_parent**: The view should be as big as its parent (minus padding).
       - **wrap_content**: The view should be only big enough to enclose its content (plus padding).

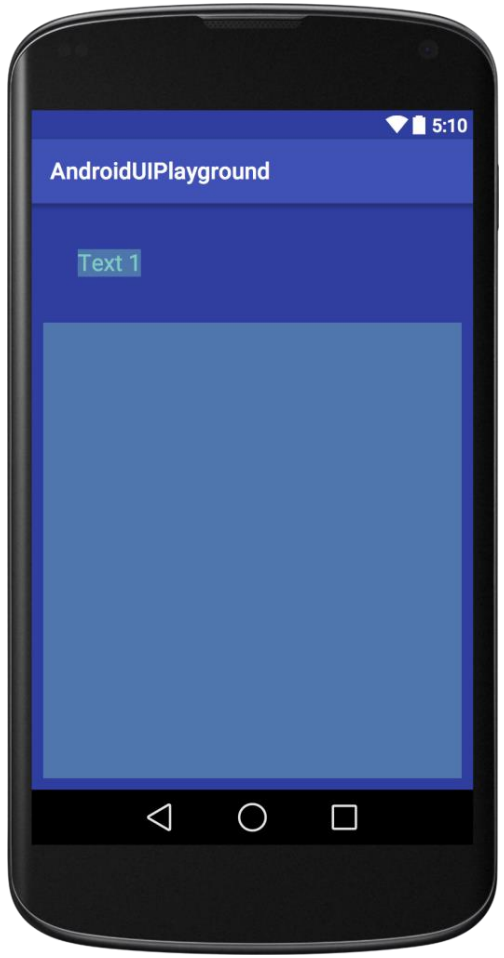
Basic Attributes - Margin

- **Margins** are the spaces outside the border, between the border and the other elements next to this view.
- Attributes for Margin include:
  - **margin** – keep distance on all the four sides
  - **marginLeft** – keep distance from left side of the view
  - **marginRight** – keep distance from right side of the view
  - **marginTop** – keep distance from top of the view
  - **marginBottom** – keep distance from bottom of the view



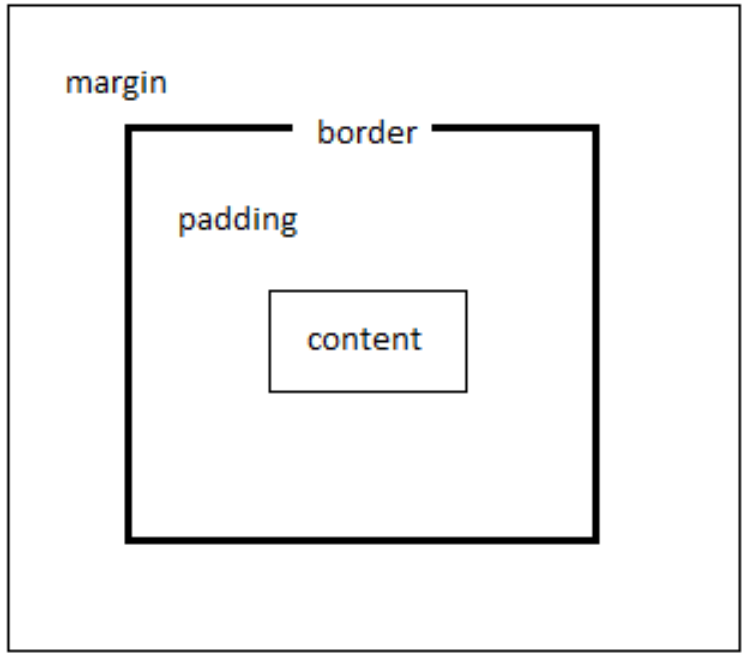
Basic Attributes – Margin - Example

```
<TextView
    android:background="@color/highlighted_text_material_dark"
    android:id="@+id/text1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20dp"
    android:layout_margin="40dp"
    android:text="Text 1"
    android:textColor="@color/accent_material_dark"/>

  <LinearLayout
    android:orientation="vertical"
    android:background="@color/highlighted_text_material_dark"
    android:id="@+id/layout1"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp"
    android:layout_marginBottom="10dp"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:textColor="@color/accent_material_dark"/>
```



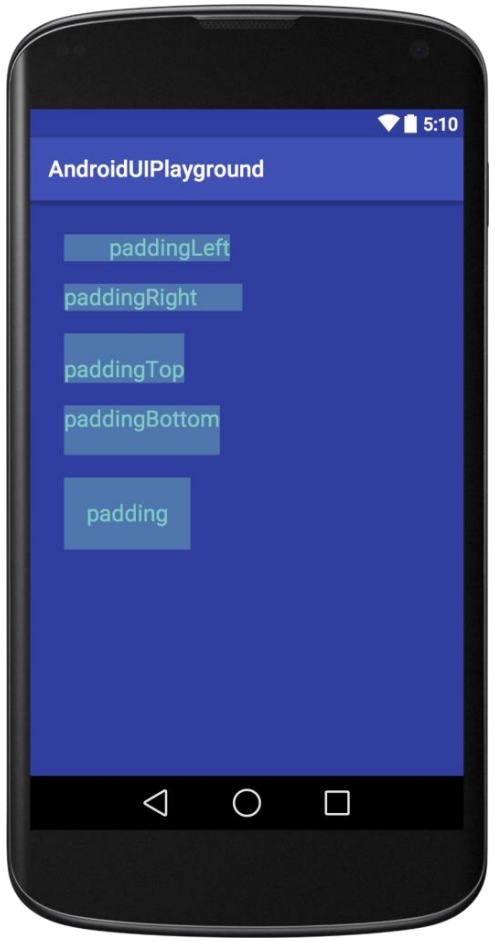
Basic Attributes - Padding

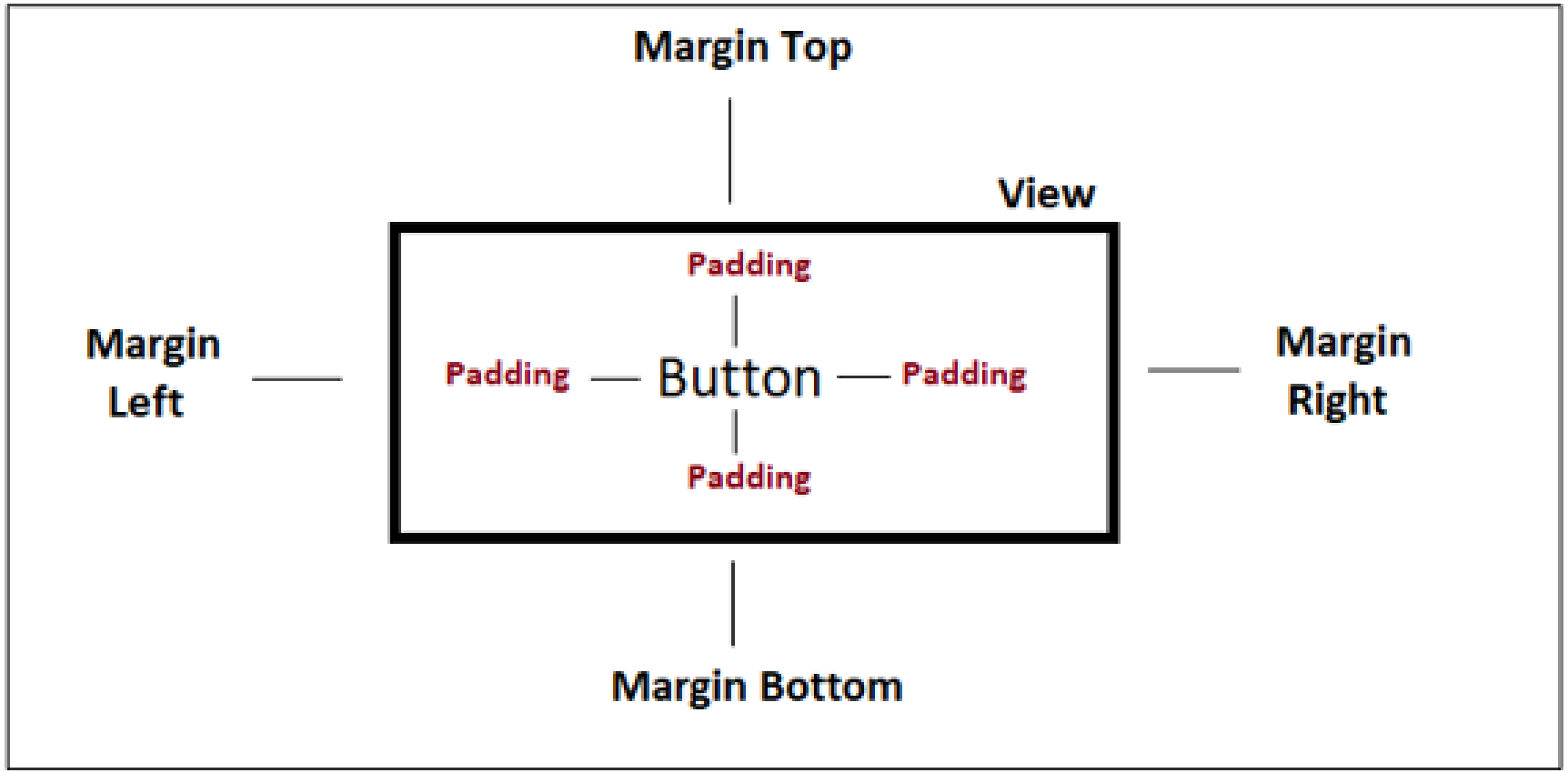
- **Padding** is the space inside the border, between the border and the actual view's content.
- Attributes for Padding include:
  - **padding** – keep distance from all the inner boundaries
  - **paddingLeft** – keep distance from the left inner boundary
  - **paddingRight** – keep distance from the right inner boundary
  - **paddingTop** – keep distance from the top inner boundary
  - **paddingBottom** – keep distance from the bottom inner boundary



Basic Attributes – Padding - Example

```
…
<TextView
        android:background="@color/highlighted_text_material_dark"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20dp"
        android:padding="20dp"
        android:text="padding"
        android:layout_margin="10dp"
        android:textColor="@color/accent_material_dark"/>
…
```

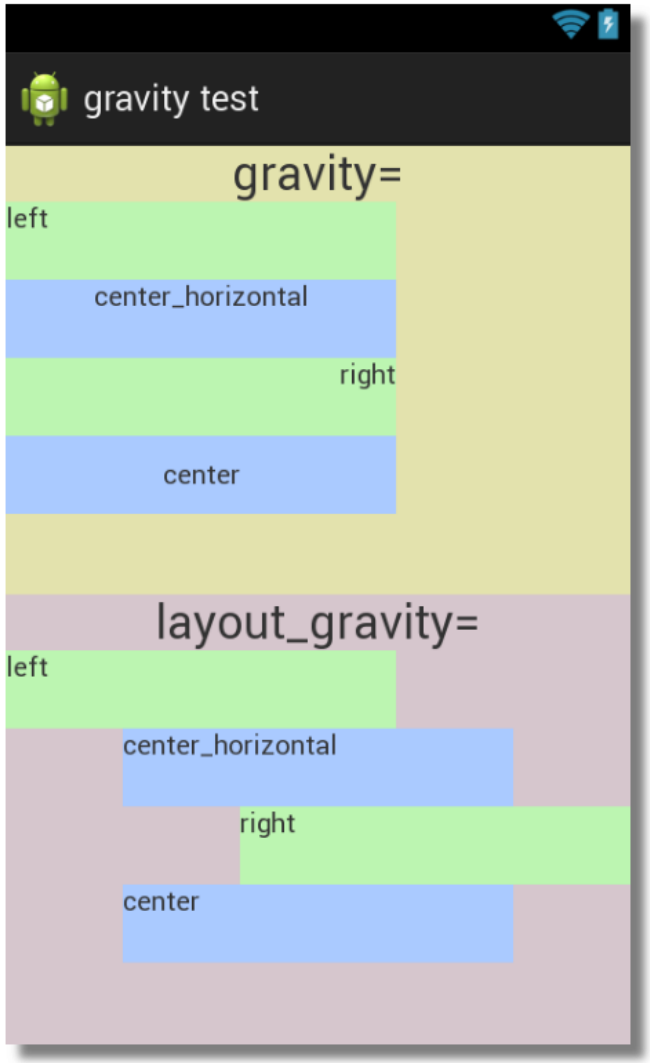



Margin Top
View
Padding
Margin Left
Padding
Button
Padding
Margin Right
Padding
Margin Bottom

Basic Attributes – Gravity

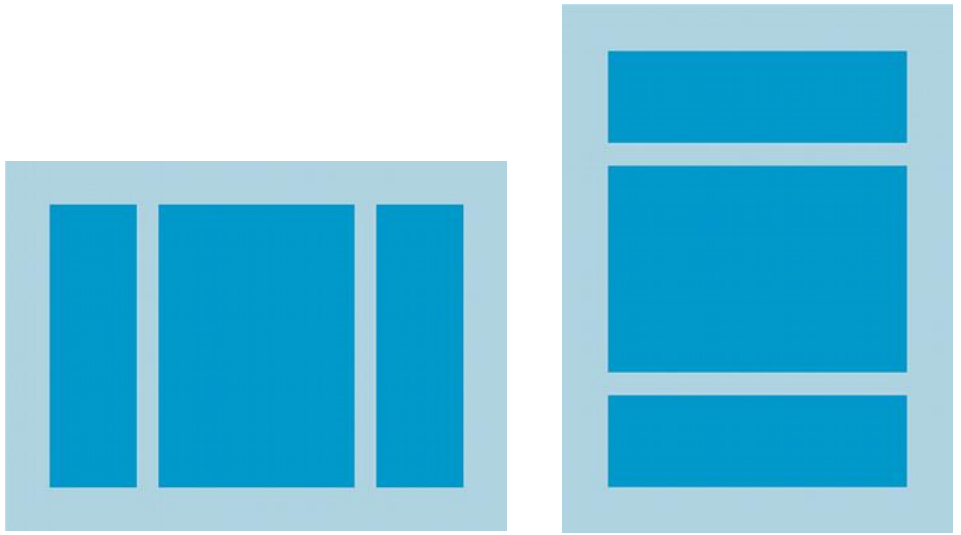
- Gravity adjusts view and content position.
- Android supports both gravity and layout_gravity.
- Using gravity we can do alignment of view as displayed.
- Values of gravity attributes can be center, top, bottom, left, right, …
- Example:

```
<TextView
  android:id = "@+id/button"
  android:layout_width = "match_parent"
  android:layout_height = "wrap_content"
  android:gravity = "center"/>

< TextView
  android:id = "@+id/editext"
  android:layout_width = "wrap_content"
  android:layout_height = "wrap_content"
  android:layout_gravity = "right“/>
```



Linear Layout

- **LinearLayout** is a view group that aligns all children in a single direction, vertically or horizontally.
- You can specify the layout direction with the **android:orientation** attribute.
- All children of a LinearLayout are stacked one after the other, so
- a vertical list will only have one child per row, no matter how wide they are.
- a horizontal list will only be one row high.



Horizontal *Vertical*

Linear Layout - Example

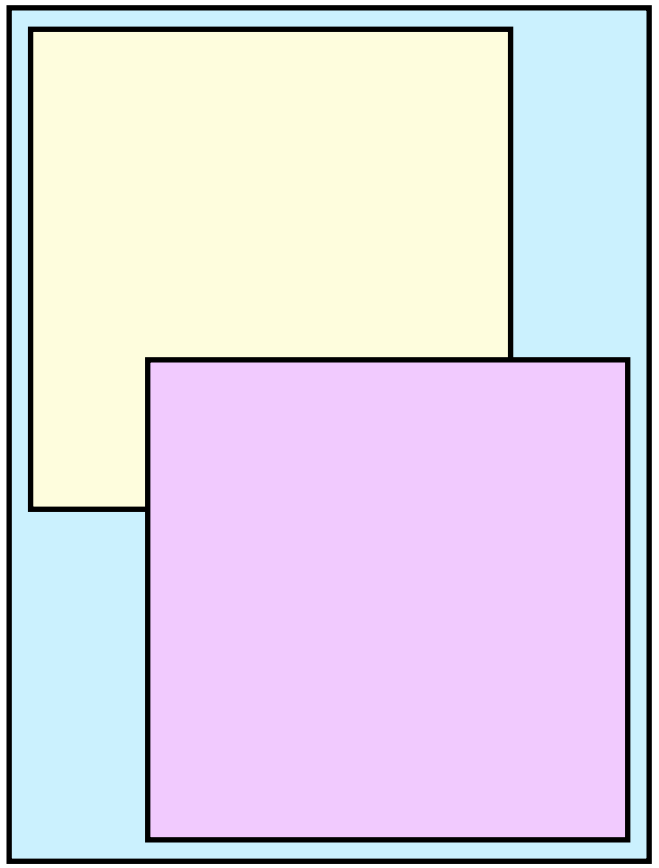
```
<LinearLayout
   …
   android:gravity="center"
   android:orientation="vertical"
   tools:context=".MainActivity">

      <Button
         android:layout_width="250dp"...
         android:textColor="@android:color/white"/>
      <Button
         android:layout_width="250dp"...
         android:textColor="@android:color/white"/>
      <Button
         android:layout_width="250dp"...
         android:textColor="@android:color/white"/>

</LinearLayout>
```

Linear Layout - Attributes

1	android:**id** This is the ID which uniquely identifies the layout.
2	android:**gravity** This specifies how an object should position its content, on both the X and Y axes. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.
3	android:**orientation** This specifies the direction of arrangement and you will use "horizontal" for a row, "vertical" for a column. The default is horizontal.
4	android:**weightSum** Sum up of child weight

Frame Layout

- Framelayout is a ViewGroup that is used to specify the position of View instances it contains on the top of each other to display only single View inside it.
- We can say FrameLayout is designed to block out an area on the screen to display a single item.



Frame Layout

Frame Layout - Example

```
<FrameLayout
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical">
   <ImageView
      android:id="@+id/imgvw1"
      ...
      android:src="@drawable/flimg" />
   <TextView
      android:id="@+id/txtvw1"
      ...
      android:textSize="20sp" />
   <TextView
      android:id="@+id/txtvw2"
      ...
      android:textSize="18sp" />
</FrameLayout>
```

Summary

- A **Layout** is a special type of view that acts as a **container** for other views or layouts.
- You can declare a layout in **two ways**:
  - Declare UI elements in XML
  - Instantiate layout elements at runtime
- **Layout Types** include: Linear Layout, Frame Layout, Constraint Layout, Table Layout, …
- There are some **basic attributes** which can be used commonly in every view. For, sizing, Margin, Padding, Gravity
- **LinearLayout** is a view group that aligns all children in a single direction, vertically or horizontally.
- **FrameLayout** is designed to block out an area on the screen to display a single item.

Assignment #2

- Design and develop a Calculator application.
  - *I will describe the app details orally for you.*

- *Score: 3*
- *Deadline: until end of next week*

The End

Thank You