Lecture 6

User Interface: Linear Layout

Course: **Mobile App Development**

By: Tho C. Mai
Nha Trang University

User Interface

- How ViewGroups and Layouts can be used to lay out your views and organize your application screen
- How to adapt and manage changes in screen orientation
- How to create the UI programmatically

Components of a Screen

- The basic unit of an Android application is an activity, which displays the UI of your application using *views* and *ViewGroups*
- The activity may contain widgets such as buttons, labels, textboxes, etc.
- Typically, you define your UI using an XML file
  - located in the res/layout folder of your project
- During runtime, you load the XML UI in the onCreate() method handler in your Activity class, using the **setContentView**() method of the Activity class
- During compilation, each element in the XML file is compiled into its equivalent Android GUI 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: buttons, labels, and text boxes
  - A view derives from the base class *android.view.View*
- A ViewGroup (which is itself a special type of view) is to group views logically—such as a group of buttons with a similar purpose
  - Examples: *RadioGroup* and *ScrollView*
  - A ViewGroup derives from the base class *android.view.ViewGroup*
- Another type of ViewGroup is a Layout used to group and arrange views visually on the screen
  - Also derives from android.view.ViewGroup
  - FrameLayout, LinearLayout, TableLayout, TableRow, GridLayout, RelativeLayout, Contrainst Layout

Common Attributes of Views & ViewGroups

ATTRIBUTE	DESCRIPTION
`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 should 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

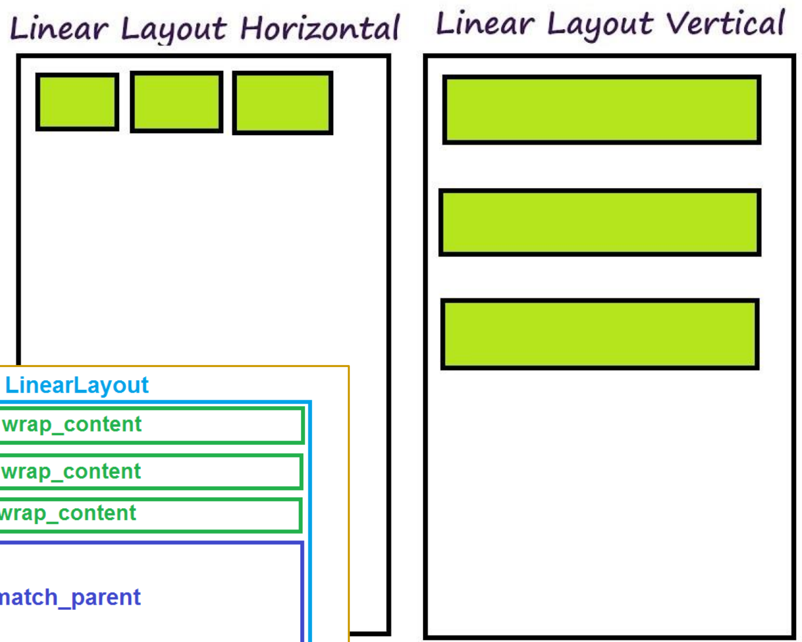
Units of Measurement

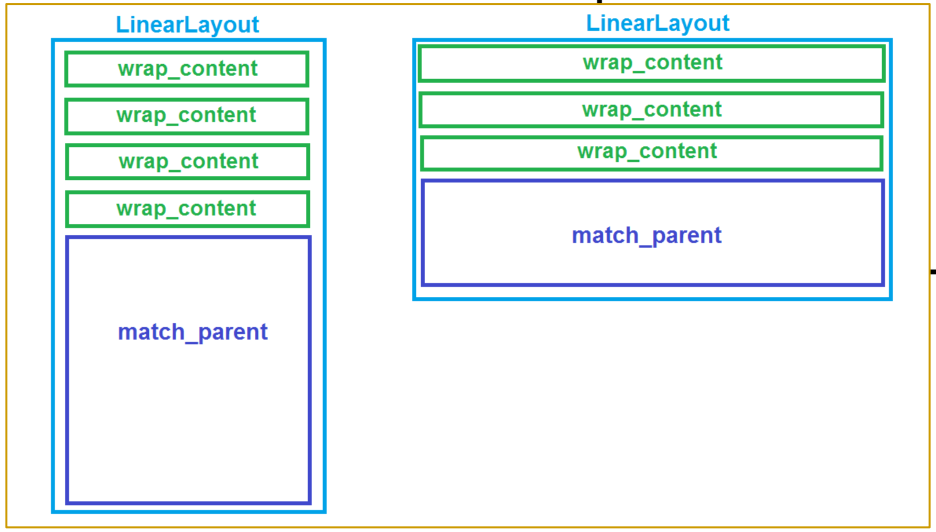
- Size of an element on an Android UI
  - dp—Density-independent pixel. 1 dp is equivalent to one pixel on a 160 dpi screen
  - sp—Scale-independent pixel. This is similar to dp and is recommended for specifying font sizes
  - pt—Point. A point is defined to be 1/72 of an inch, based on the physical screen size
  - px—Pixel. Corresponds to actual pixels on the screen. Using this unit is not recommended

LinearLayout

- The LinearLayout arranges views in a single column or a single row
- Child views can be arranged either horizontally or vertically

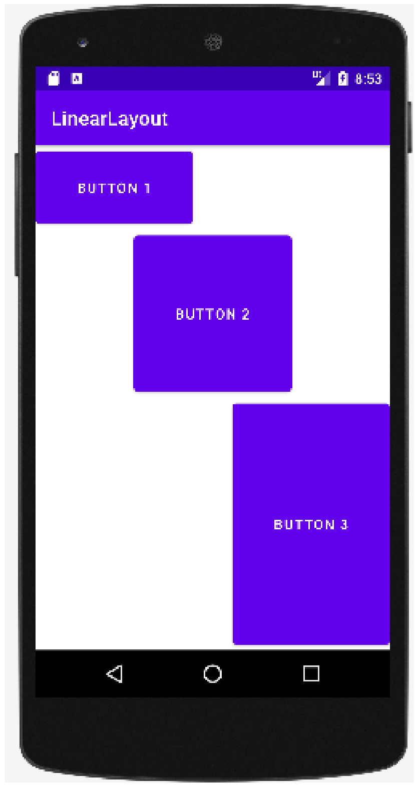
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout    xmlns:android="http://schemas.android.com/apk/res/android
   xmlns:tools="http://schemas.android.com/tools"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">
   <TextView
      android:layout_width="fill_parent"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





Layout Weight & Gravity

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout    xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <Button
    android:layout_width="160dp"
    android:layout_height="0dp"
    android:text="Button 1"
    android:layout_gravity="left"
    android:layout_weight="1" />
  <Button
    android:layout_width="160dp"
    android:layout_height="0dp"
    android:text="Button 2"
    android:layout_gravity="center"
    android:layout_weight="2" />
  <Button
    android:layout_width="160dp"
    android:layout_height="0dp"
    android:text="Button 3"
    android:layout_gravity="right"
    android:layout_weight="3" />
</LinearLayout>
```

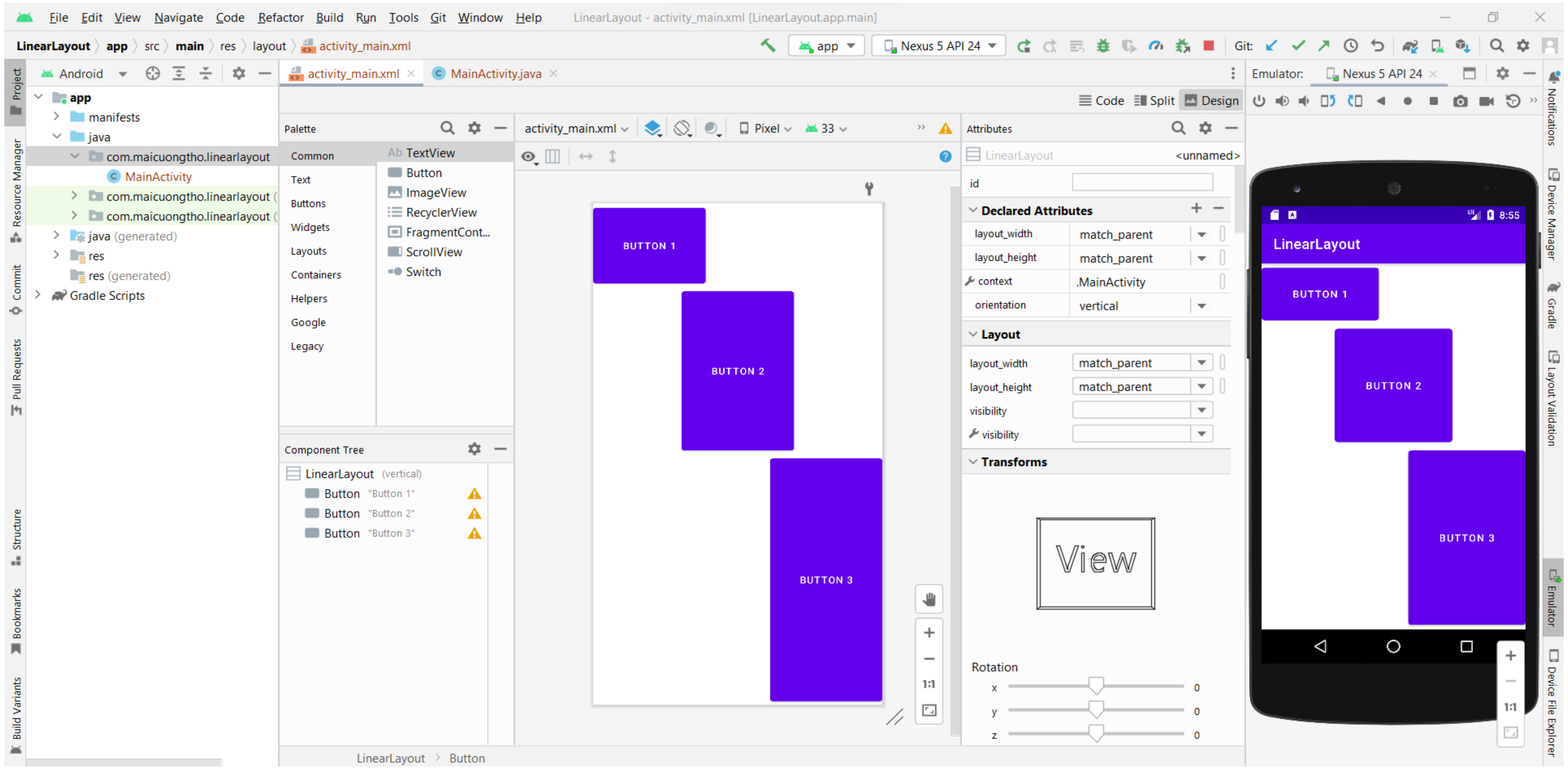


The **layout_gravity** attribute indicates the positions the views should gravitate toward, whereas the **layout_weight** attribute specifies the distribution of available space

The three buttons occupy about 16.6 percent (1/(1+2+3) * 100), 33.3 percent (2/(1+2+3) * 100), and 50 percent (3/(1+2+3) * 100) of the available height, respectively

The height of each button is set to 0dp because the layout orientation is vertical

WYSIWYG (Design view): Drag, Drop, set Attribute



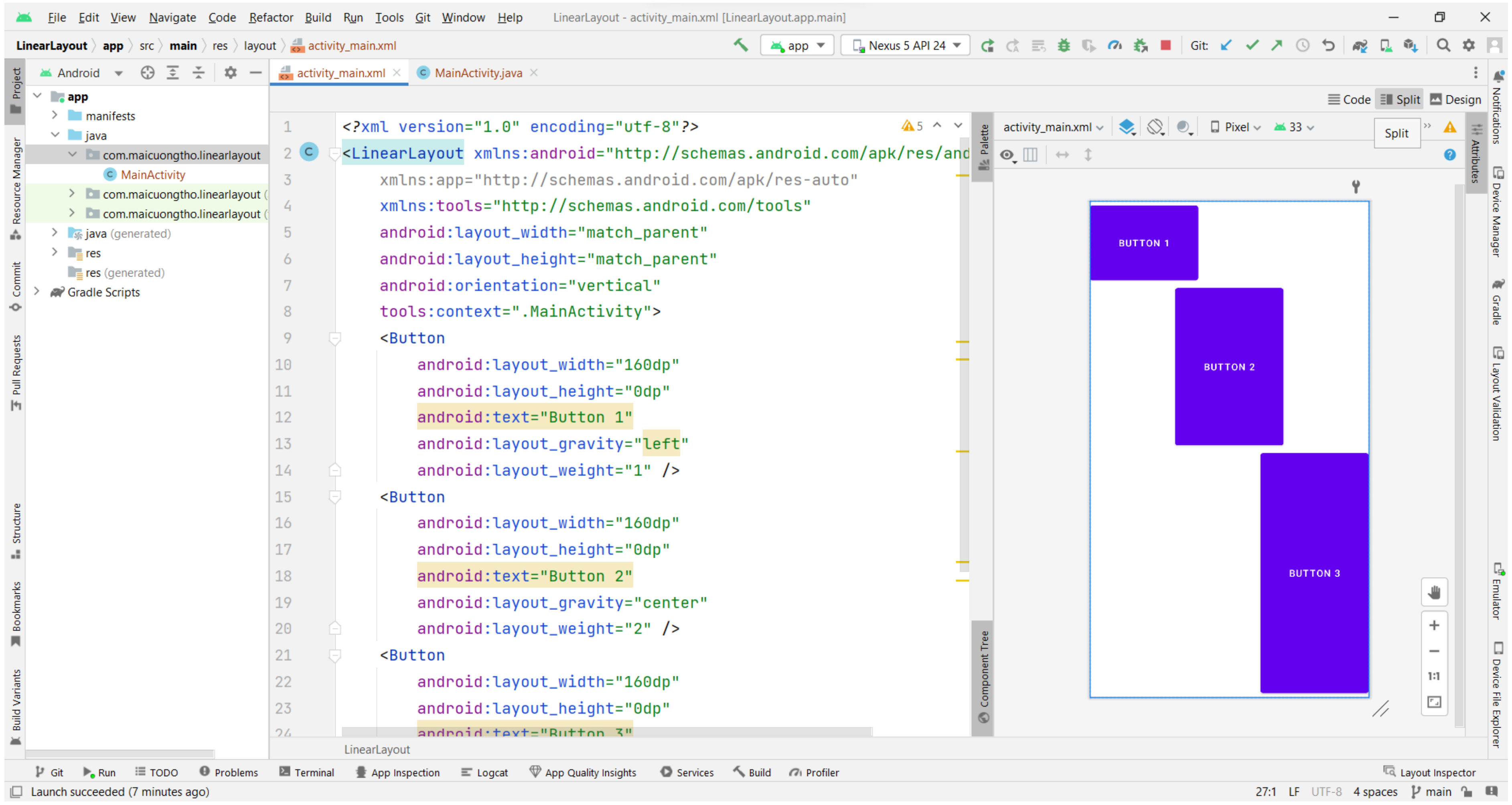
Declared Attributes

layout_width	match_parent
layout_height	match_parent
context	.MainActivity
orientation	vertical

Layout

layout_width	match_parent
layout_height	match_parent
visibility	
visibility	

XML code and Preview (Split code and design)



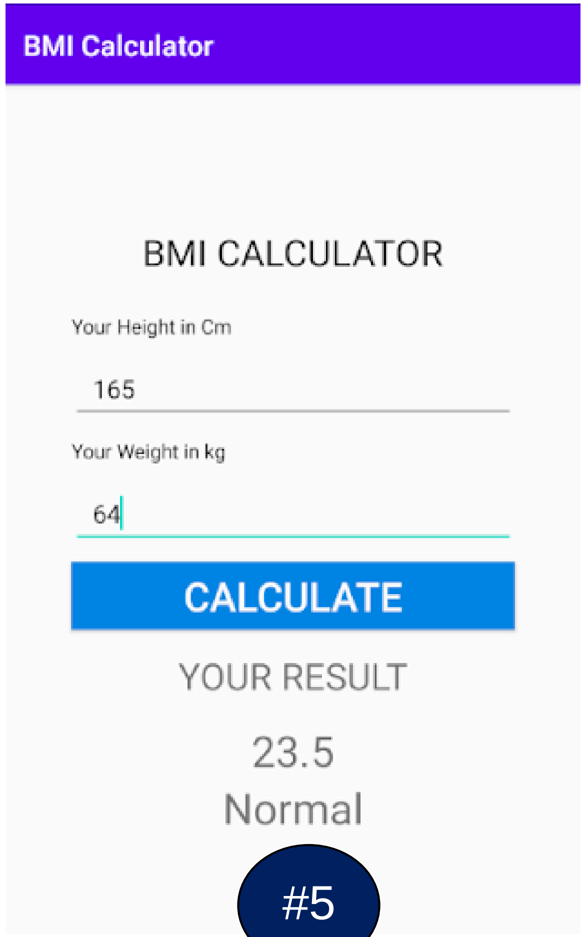
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <Button
        android:layout_width="160dp"
        android:layout_height="0dp"
        android:text="Button 1"
        android:layout_gravity="left"
        android:layout_weight="1" />
    <Button
        android:layout_width="160dp"
        android:layout_height="0dp"
        android:text="Button 2"
        android:layout_gravity="center"
        android:layout_weight="2" />
    <Button
        android:layout_width="160dp"
        android:layout_height="0dp"
        android:text="Button 3"
```

Your turn

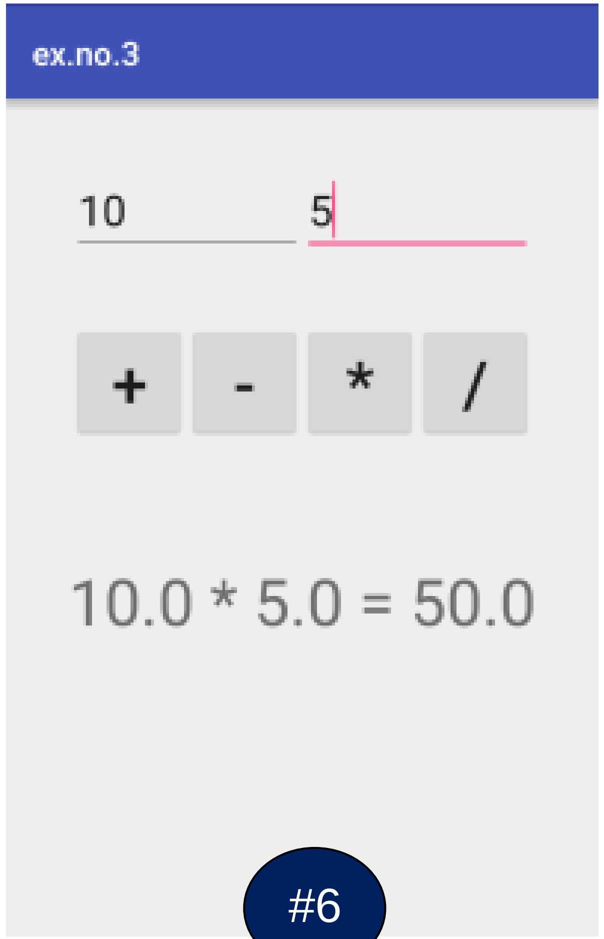
I fear not the man who has practiced 10,000 kicks once,
but I fear the man who has practiced one kick 10,000 times.
Bruce Lee

- Practice 11/ Excercie 11
  1) Repeat the examples by yourself
  2) Push it to your github repository
     - ✓ With a report with screenshots of the final app in action, data structures used/class design, and the implementation logic.
- ❑ Practice 12/ Excercie 12 (Homework #5)
- ❑ Practice 13/ Excercie 13 (Homework #6)
- ❑ Practice 14/ Excercie 14 (Homework #7)

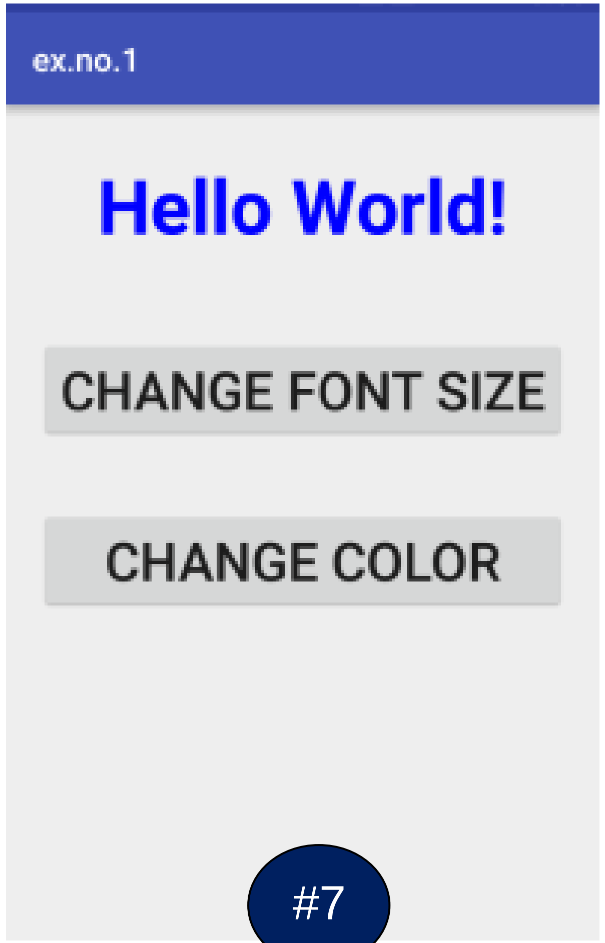
Homework #5,6,7



#5

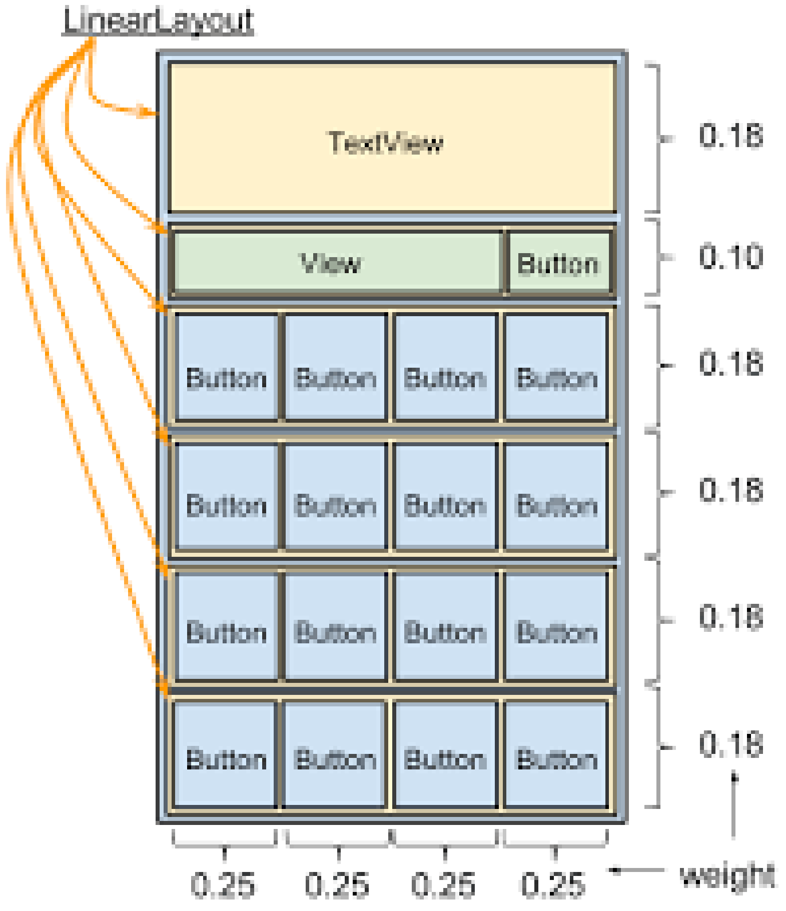


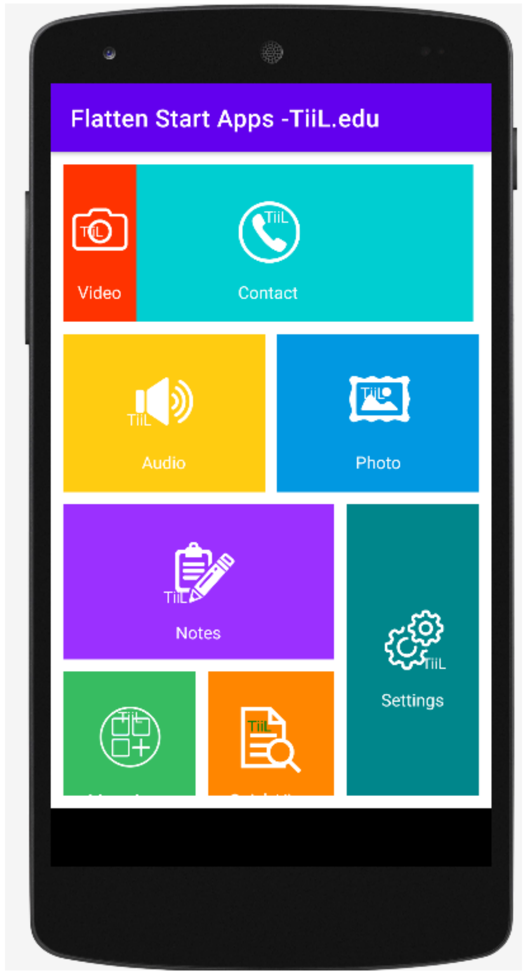
#6



#7

Homework 8*, 9*


LinearLayout
TextView
View
Button
Button
0.18
0.10
0.18
0.18
0.18
0.18
0.25
0.25
0.25
0.25
weight


Flatten Start Apps -TiiL.edu
Video
Contact
Audio
Photo
Notes
Settings