CIS 470
Mobile App Development

Lecture 6

Wenbing Zhao
Department of Electrical Engineering and Computer Science
Cleveland State University
wenbing@ieee.org

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
- How to listen for UI notifications

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, ach 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);
}
```

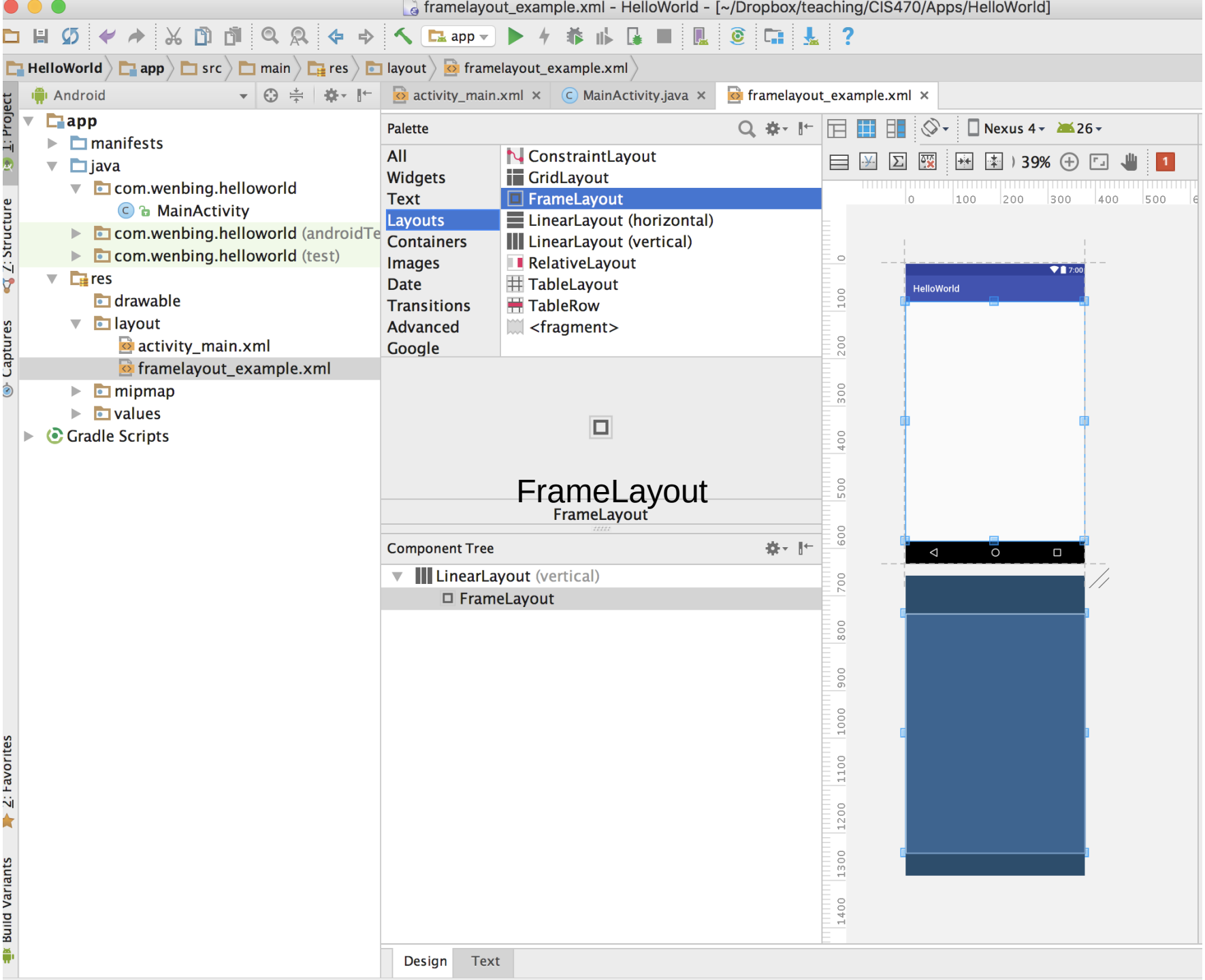
Views and ViewGroups

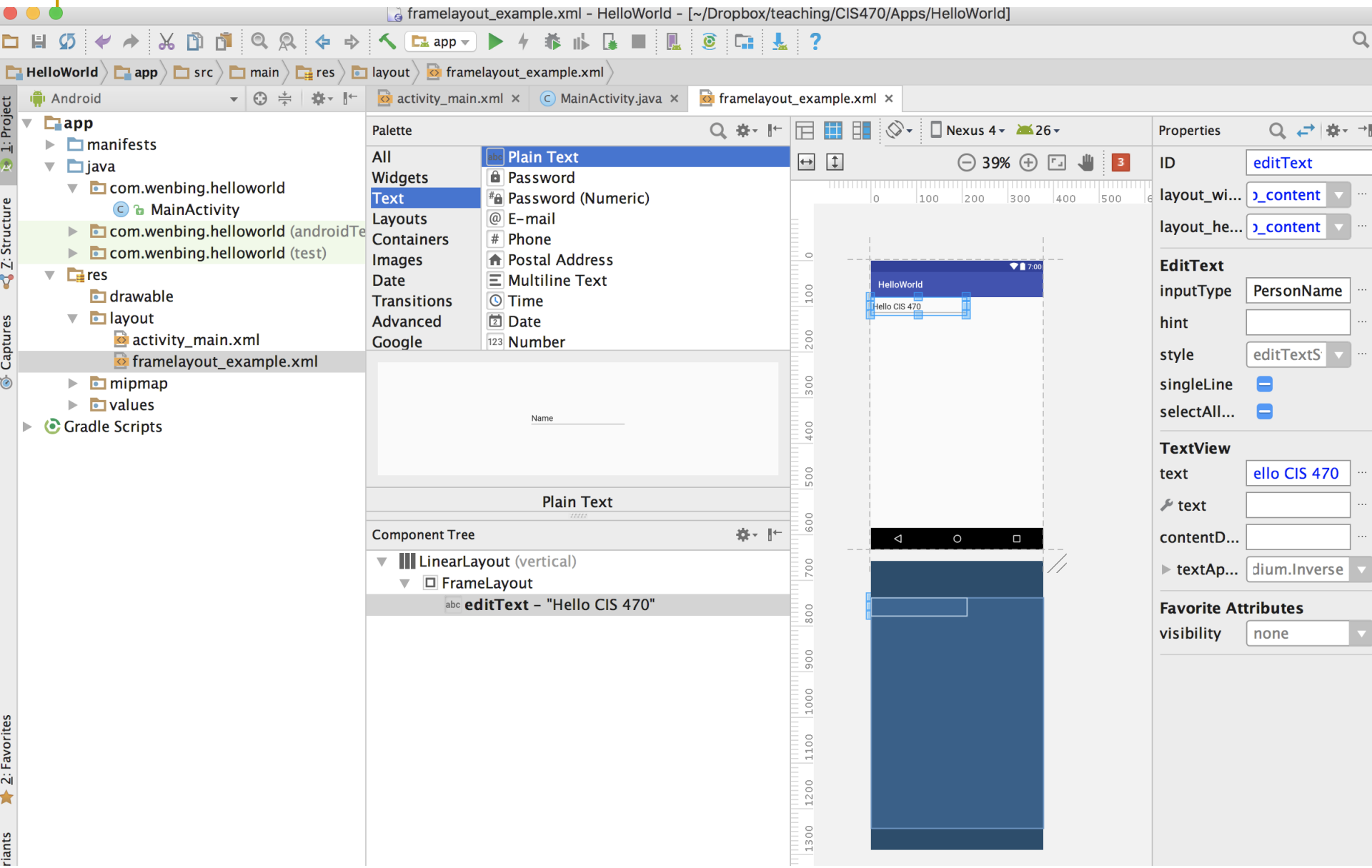
- An activity contains views and ViewGroups
- A view is a widget that has an appearance on screen
  - Examples: buttons, labels, and text boxes
  - A view derives from the base class *android.view.View*
- A ViewGroup (which is itself a special type of view) is to group views logically—such as a group 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

FrameLayout

- The FrameLayout is the most basic of the Android layouts.
  - FrameLayouts are built to hold one View
- The FrameLayout is used to help you control the stacking of single views as the screen is resized (or screens with different resolutions)
- Try-it-out: Place a TextView Within a FrameLayout
  - Open/create the HelloWorld project in Android Studio.
  - Right-click the res/layout folder and add a new layout resource file. Name the file
  - framelayout_example.xml.
  - Using the design panel, drag the FrameLayout and place it anywhere on the device screen.
  - Using the design panel, drag a Plain TextView and place it in the FrameLayout.
  - Type some text into the Plain TextView.

The above is missing a critical step. See if you can figure it out!


framelayout_example.xml - HelloWorld - [~/Dropbox/teaching/CIS470/Apps/HelloWorld]
app
HelloWorld
app
src
main
res
layout
framelayout_example.xml
Android
activity_main.xml
MainActivity.java
framelayout_example.xml
app
manifests
java
com.wenbing.helloworld
MainActivity
com.wenbing.helloworld (androidTe
com.wenbing.helloworld (test)
res
drawable
layout
activity_main.xml
framelayout_example.xml
mipmap
values
Gradle Scripts
Palette
All
Widgets
Text
Layouts
Containers
Images
Date
Transitions
Advanced
Google
ConstraintLayout
GridLayout
FrameLayout
LinearLayout (horizontal)
LinearLayout (vertical)
RelativeLayout
TableLayout
TableRow
<fragment>
FrameLayout
FrameLayout
Component Tree
LinearLayout (vertical)
FrameLayout
Nexus 4
26
39%
HelloWorld
Design
Text
1: Project
7: Structure
Captures
2: Favorites
Build Variants


framelayout_example.xml - HelloWorld - [~/Dropbox/teaching/CIS470/Apps/HelloWorld]
HelloWorld
app
src
main
res
layout
framelayout_example.xml
Android
activity_main.xml
MainActivity.java
framelayout_example.xml
app
manifests
java
com.wenbing.helloworld
MainActivity
com.wenbing.helloworld (androidTe
com.wenbing.helloworld (test)
res
drawable
layout
activity_main.xml
framelayout_example.xml
mipmap
values
Gradle Scripts
Palette
All
Widgets
Text
Layouts
Containers
Images
Date
Transitions
Advanced
Google
Plain Text
Password
Password (Numeric)
E-mail
Phone
Postal Address
Multiline Text
Time
Date
Number
Name
Plain Text
Component Tree
LinearLayout (vertical)
FrameLayout
editText – "Hello CIS 470"
Nexus 4
26
39%
HelloWorld
Hello CIS 470
Properties
ID
editText
layout_wi...
layout_he...
EditText
inputType
PersonName
hint
style
singleLine
selectAll...
TextView
text
ello CIS 470
contentD...
textAp...
Favorite Attributes
visibility
none

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

fill_parent: width of the<TextView> element fills the entire width of its parent

wrap_content: its height is the height of its content

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

Layout Weight & Gravity

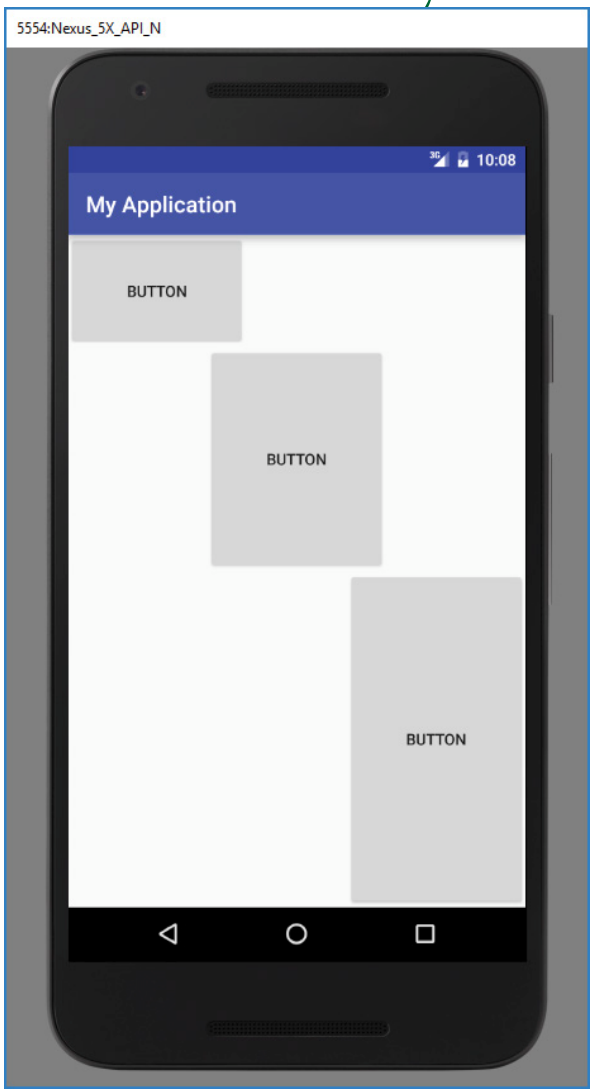
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">
    <Button
        android:layout_width="160dp"
        android:layout_height="0dp"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

LinearLayout



Exercises:

1. Create an app that produces the left output
2. Create another app that produces the right output (combination of vertical and horizontal layouts)

TableLayout

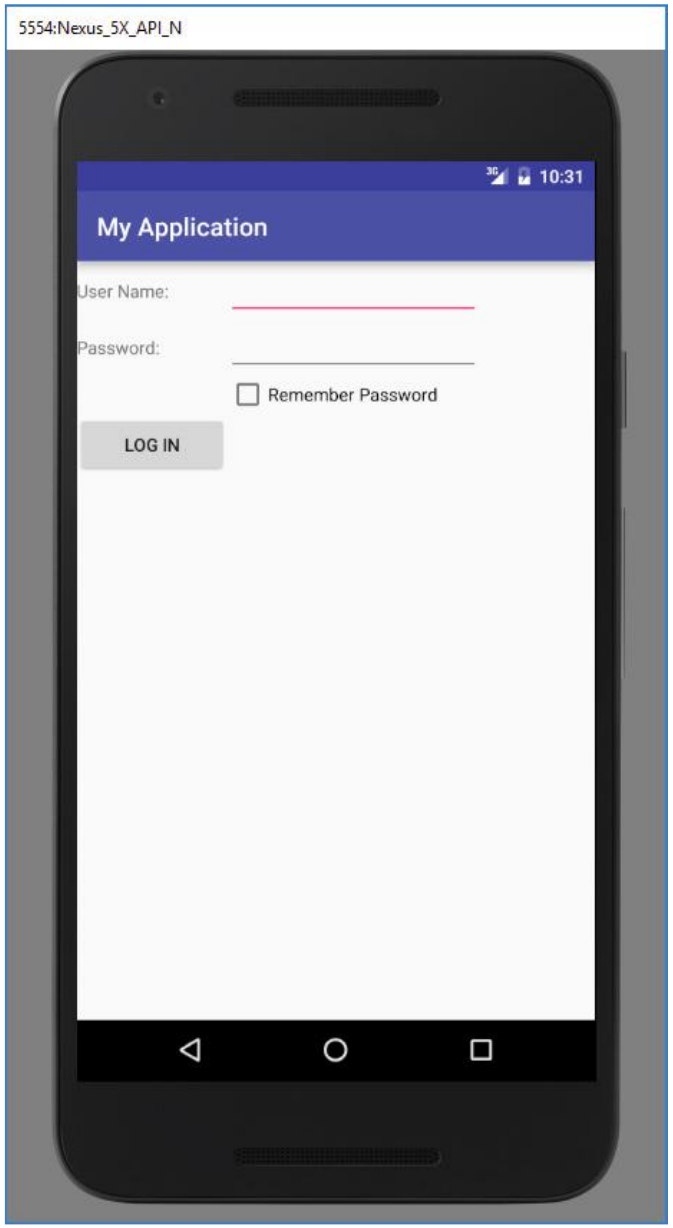
- The TableLayout Layout groups views into rows and columns
- Use the <TableRow> element to designate a row in the table
- Each row can contain one or more views. Each view you place within a row forms a cell.
- The width of each column is determined by the largest width of each cell in that column

two columns and four rows in the TableLayout

```
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent">
   <TableRow>
      <TextView
            android:text="User Name:"
            android:width ="120dp"
       />
      <EditText
            android:id="@+id/txtUserName"
            android:width="200dp" />
   </TableRow>
   <TableRow>
      <TextView
            android:text="Password:"
      />
      <EditText
         android:id="@+id/txtPassword"
         android:inputType="textPassword"
      />
</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

- Exercise: create an app with the TableLayout as described earlier and shown on the right



RelativeLayout

- The RelativeLayout enables you to specify how child views are positioned relative to each other
- RelativeLayout has attributes that enable it to align with another view. These attributes are as follows:
  - layout_alignParentTop: If true, makes the top edge of this view match the top edge of the parent
  - layout_alignParentStart
  - layout_alignStart: Makes the start edge of this view match the start edge of the given anchor view ID
  - layout_alignEnd
  - layout_below: Positions the top edge of this view below the given anchor view ID. Accommodates top margin of this view and bottom margin of anchor view.
  - layout_centerHorizontal: If true, centers this child horizontally within its parent.

RelativeLayout

```
<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_alignParentStart="true" />
    <EditText
        android:id="@+id/txtComments"
        android:layout_width="fill_parent"
        android:layout_height="170dp”
        android:textSize="18sp"
        android:layout_alignStart="@+id/lblComments"
        android:layout_below="@+id/lblComments"
        android:layout_centerHorizontal="true" />
```

```
<Button
    android:id="@+id/btnSave"
    android:layout_width="125dp"
    android:layout_height="wrap_content"
    android:text="Save"
    android:layout_below="@+id/txtComments"
    android:layout_alignEnd="@+id/txtComments" />
```

```
    <Button
        android:id="@+id/btnCancel"
        android:layout_width="124dp"
        android:layout_height="wrap_content"
        android:text="Cancel"
        android:layout_below="@+id/txtComments"
        android:layout_alignStart="@+id/txtComments" />
</RelativeLayout>
```

RelativeLayout

- Exercise: create an app that produces the display on the right