Android UI Development

Android Studio

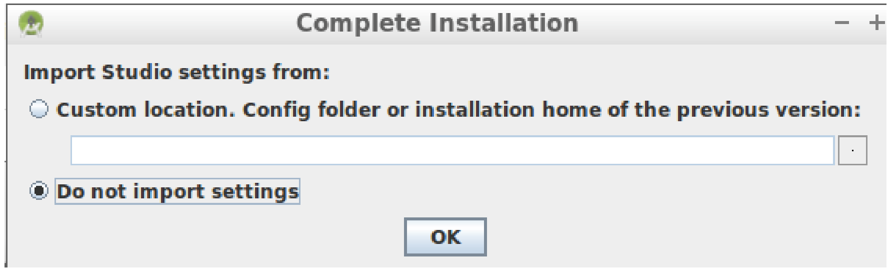
Widget

Layout

Android Development Environment

1. Install Android Studio, and run it
   https://developer.android.com/studio/install.html

2. Don't import previous settings



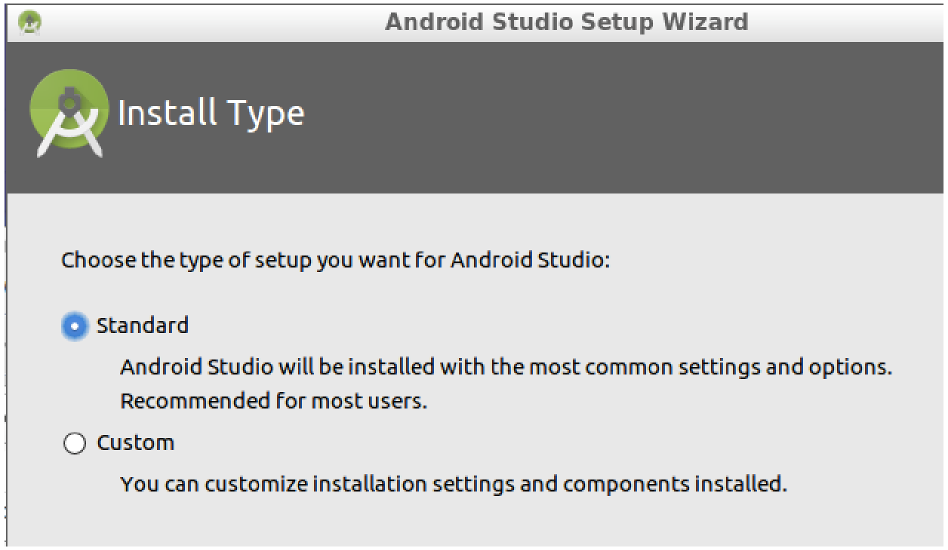
Complete Installation

Import Studio settings from:

( ) Custom location. Config folder or installation home of the previous version:

(●) Do not import settings

1. Choose "install standard type"



Android Studio Setup Wizard

Install Type

Choose the type of setup you want for Android Studio:

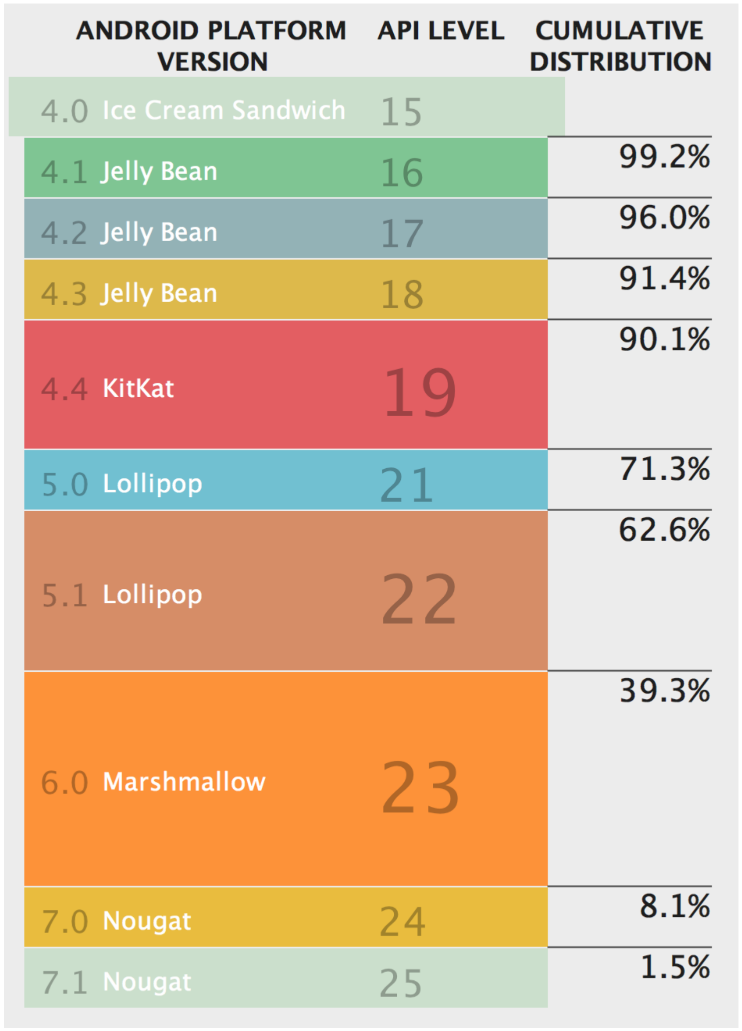
(●) Standard
Android Studio will be installed with the most common settings and options.
Recommended for most users.

( ) Custom
You can customize installation settings and components installed.

Create new project

- Domain
  - only important if you release your app, can just use something like: `cosc341.ubco.ca`

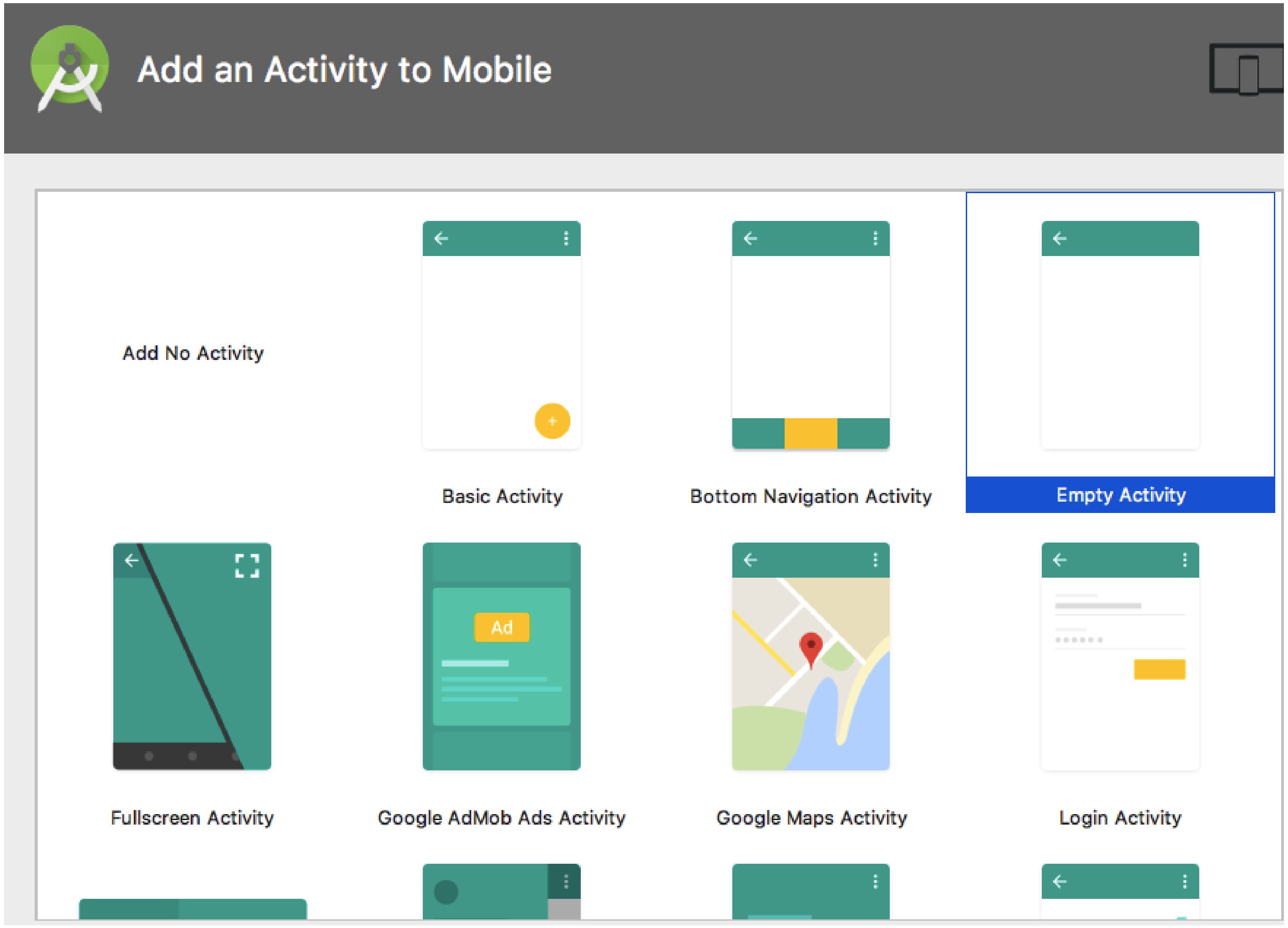
- Need to choose which API to target



ANDROID PLATFORM VERSION		API LEVEL	CUMULATIVE DISTRIBUTION
4.0	Ice Cream Sandwich	15	
4.1	Jelly Bean	16	99.2%
4.2	Jelly Bean	17	96.0%
4.3	Jelly Bean	18	91.4%
4.4	KitKat	19	90.1%
5.0	Lollipop	21	71.3%
5.1	Lollipop	22	62.6%
6.0	Marshmallow	23	39.3%
7.0	Nougat	24	8.1%
7.1	Nougat	25	1.5%

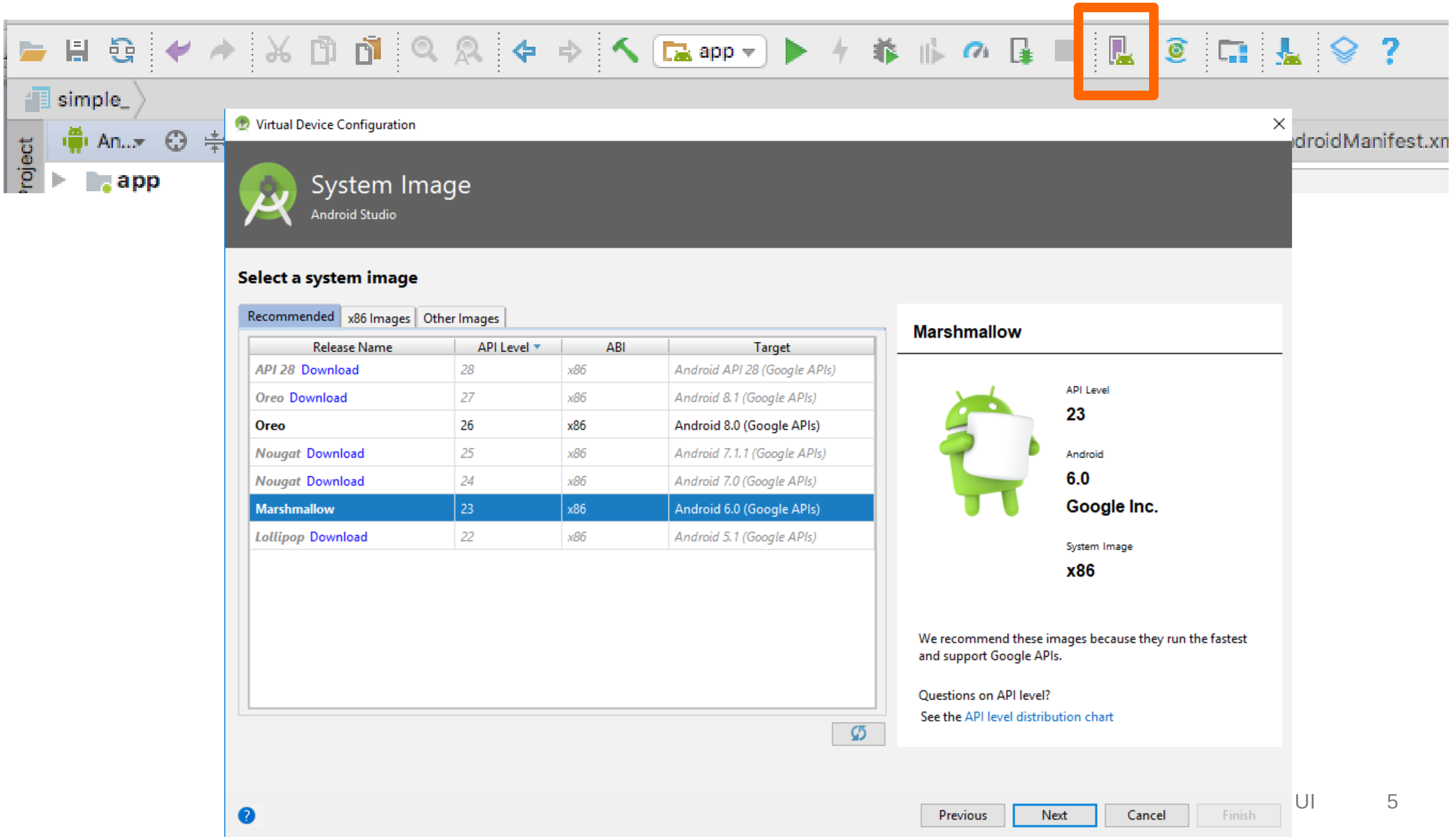
Create new project - Activity templates -

- For the assignment, choose “Empty Activity”



Create Android Virtual Device (ADV) For testing

- Device: Pixel
- System: Android API 23 (Marshmallow)



Virtual Device Configuration

System Image
Android Studio

Select a system image

Release Name	API Level	ABI	Target
API 28 Download	28	x86	Android API 28 (Google APIs)
Oreo Download	27	x86	Android 8.1 (Google APIs)
Oreo	26	x86	Android 8.0 (Google APIs)
Nougat Download	25	x86	Android 7.1.1 (Google APIs)
Nougat Download	24	x86	Android 7.0 (Google APIs)
Marshmallow	23	x86	Android 6.0 (Google APIs)
Lollipop Download	22	x86	Android 5.1 (Google APIs)

Marshmallow

API Level
23

Android
6.0
Google Inc.

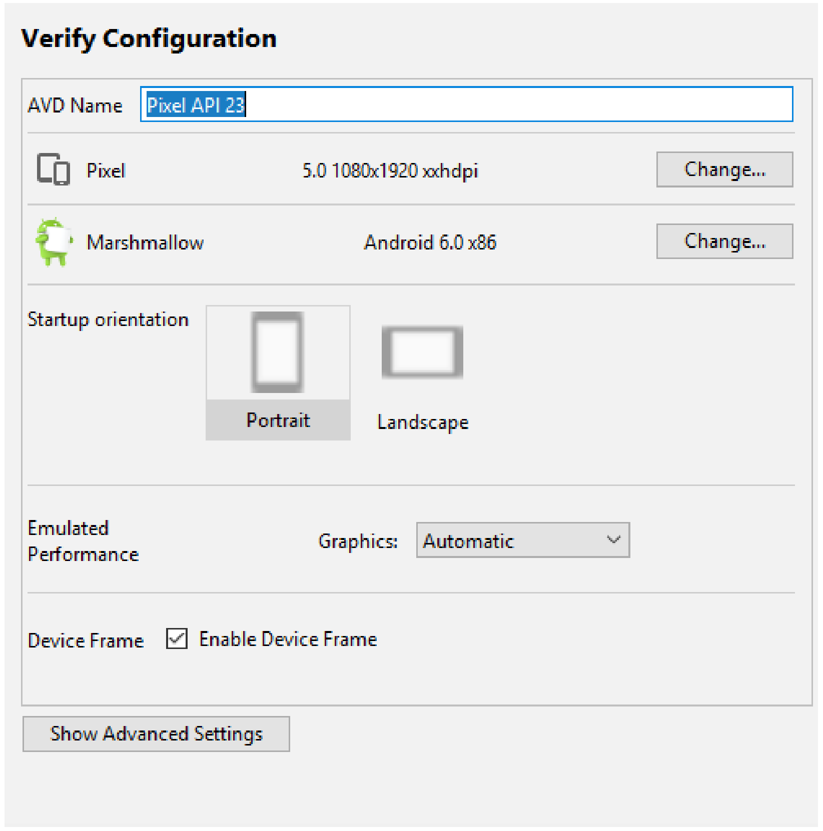
System Image
x86

We recommend these images because they run the fastest and support Google APIs.

Questions on API level?
See the API level distribution chart

Android Virtual Device (AVD)

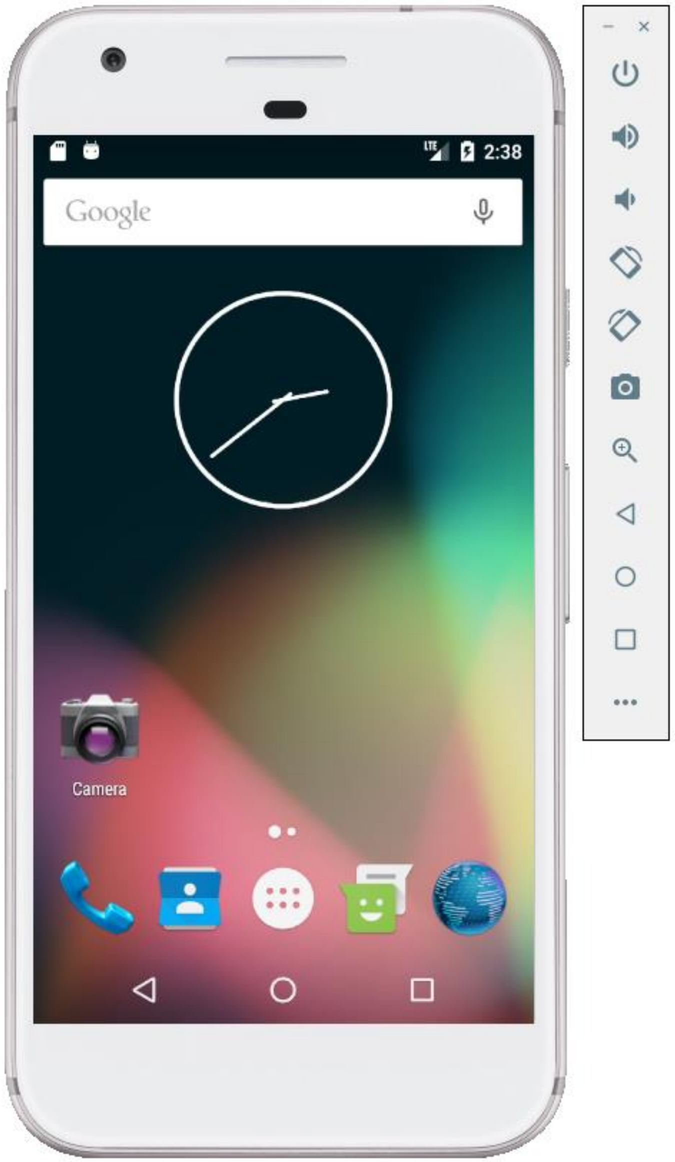
- The AVD is slow to launch, so keep it running in the background while you're programming / debugging.



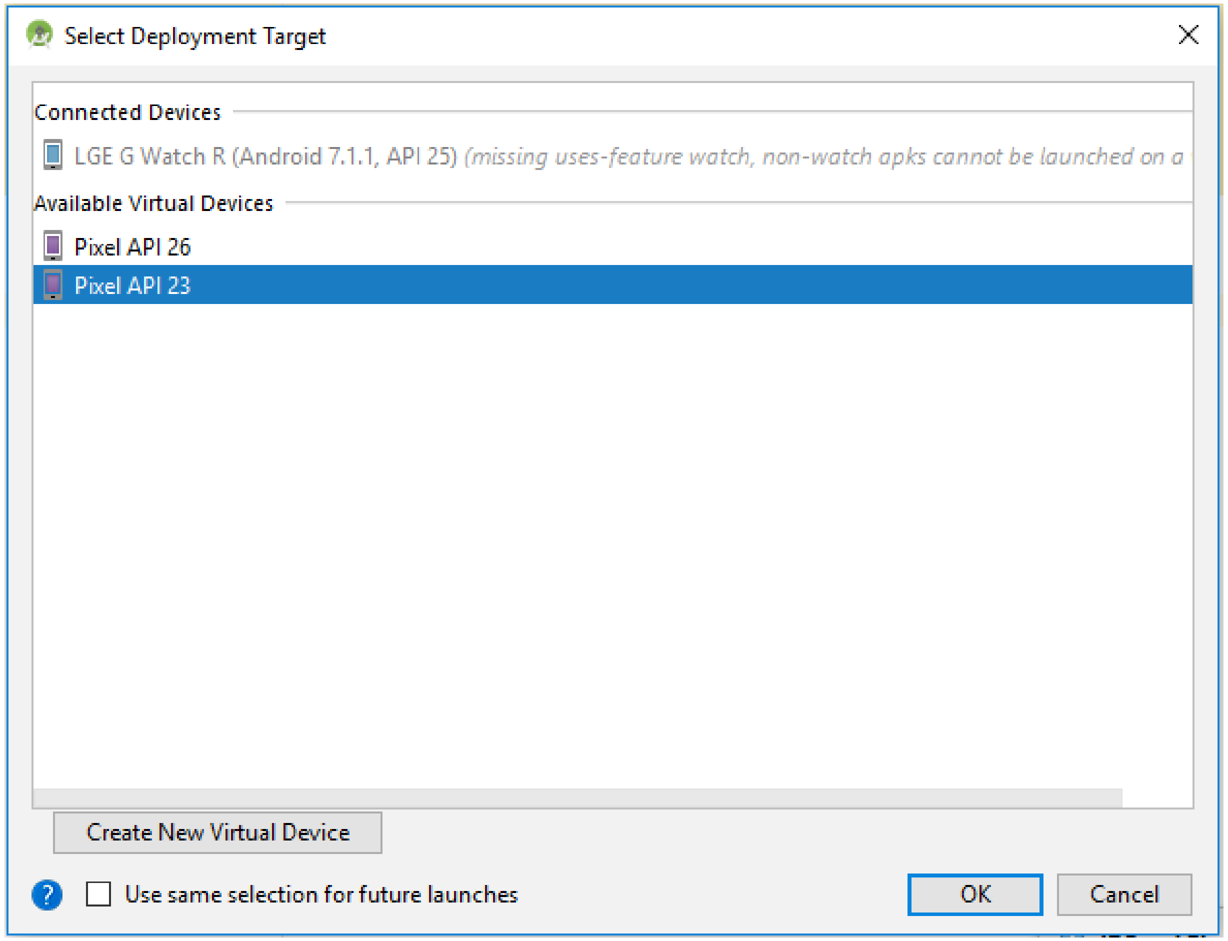
Verify Configuration

AVD Name	Pixel API 23		
Pixel	5.0 1080x1920 xxhdpi		Change...
Marshmallow	Android 6.0 x86		Change...
Startup orientation	Portrait	Landscape	
Emulated Performance	Graphics:	Automatic	
Device Frame	☑ Enable Device Frame		

Show Advanced Settings



Run sample code

- Run
- Select Deployment Target -> Pixel API 23 (the one just created)
- 

Select Deployment Target

Connected Devices

LGE G Watch R (Android 7.1.1, API 25) *(missing uses-feature watch, non-watch apks cannot be launched on a*

Available Virtual Devices

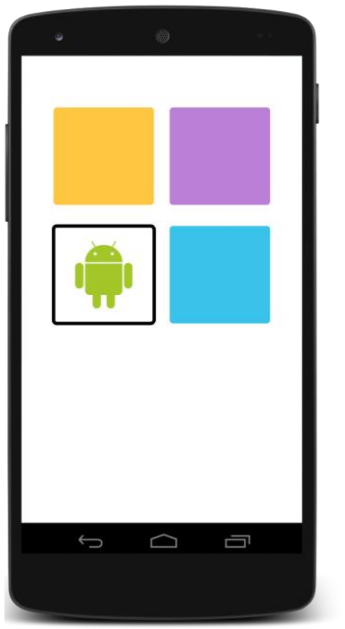
Pixel API 26

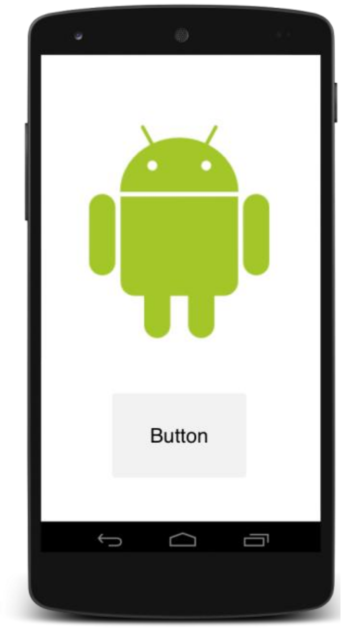
Pixel API 23

Create New Virtual Device
Use same selection for future launches
OK
Cancel

Activities

- A standard application component is an Activity
  - Typically represents a single screen
  - Main entry point (equivalent to main() method)
  - For most purposes, this is your application class

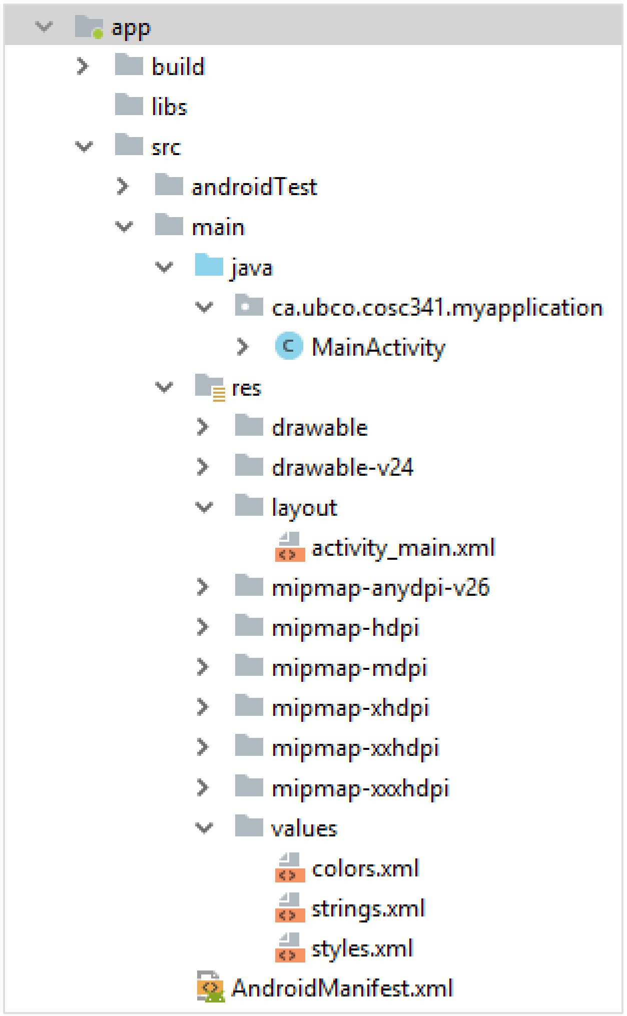

Activity 1



Activity 2

Android Project Files

- Manifest (app/manifests/)
  - Application setting
- Java (app/java/)
  - **(*.java) source code**
- Resources (app/res/)
  - **layout: (*.xml) UI layout and View definitions**
  - **values: (*.xml) constants** like strings, colours, …
  - also bitmaps and SVG images (mipmap*, drawable*, ….)



```
app
  build
  libs
  src
    androidTest
    main
      java
        ca.ubco.cosc341.myapplication
          MainActivity
      res
        drawable
        drawable-v24
        layout
          activity_main.xml
        mipmap-anydpi-v26
        mipmap-hdpi
        mipmap-mdpi
        mipmap-xhdpi
        mipmap-xxhdpi
        mipmap-xxxhdpi
        values
          colors.xml
          strings.xml
          styles.xml
      AndroidManifest.xml
```

Manifest

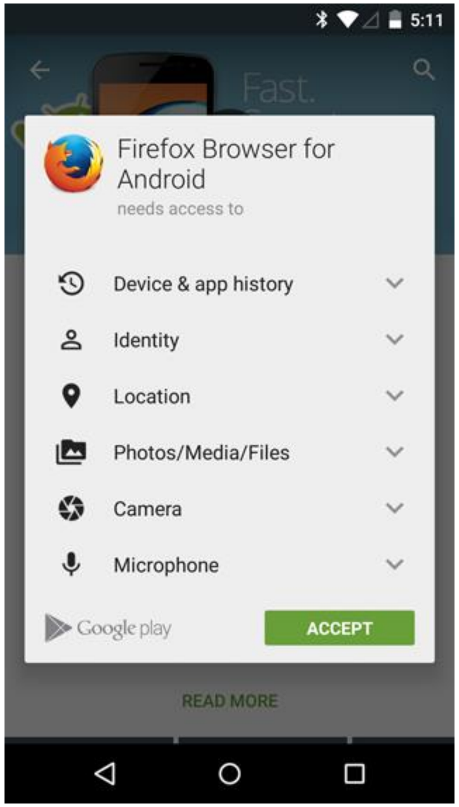
- Metadata about the app
- App components, Intent filters

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
        <intent-filter>
          <action android:name="android.intent.action.MAIN" />
          <category
              android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

Manifest – Permissions -

```
<manifest>
   <uses-permission
       android:name="android.permission.INTERNET" />
   <uses-permission
       android:name="android.permission.ACCESS_FINE_LOCATION" />
   <uses-permission
       android:name="android.permission.SEND_SMS" />
</manifest>
```

- Android must request permission to access sensitive user data


Firefox Browser for Android

needs access to

- Device & app history
- Identity
- Location
- Photos/Media/Files
- Camera
- Microphone

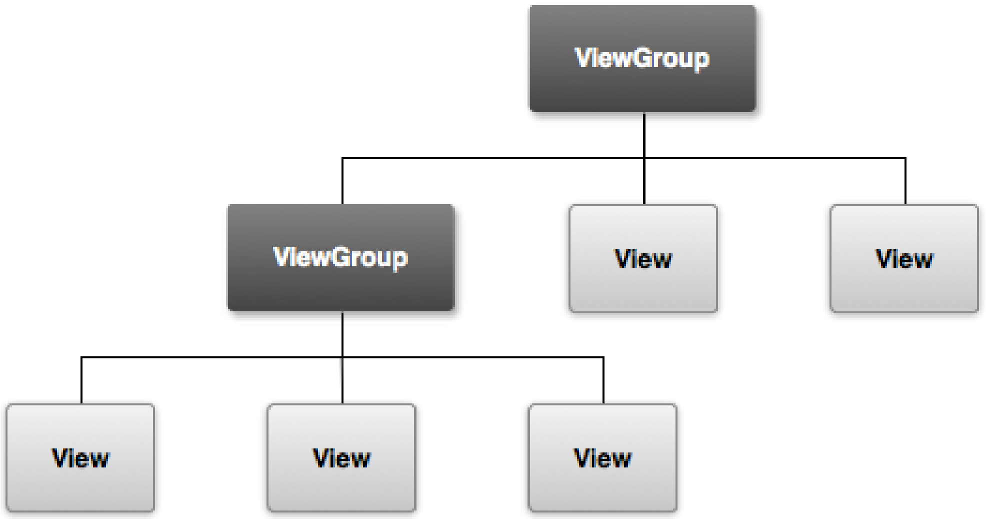
App Resources

- Each type of resource in a specific subdirectory of your project's res/ directory
- Access them using resource IDs that are generated in the project's R class

```
app/
      manifest/
      java/
      res/
            drawable/
                        graphic.png
            layout/
                        activity_main.xml
            mipmap/
                        icon.png
            values/
                        strings.xml
```

Layouts

- Defines the structure for a user interface
- Built using a hierarchy of View and ViewGroup
- A View usually draws something the user can see and interact with
- A ViewGroup is an invisible container that defines the layout structure for View and other ViewGroup objects



https://developer.android.com/guide/topics/ui/declaring-layout.html

Views (what android calls a widget)

`android.view.View`

- Base widget class (drawing and event handling)
- Subclasses:

```
android.widget.Button
android.widget.ImageView
android.widget.ProgressBar
Android.widget.TextView
...
```

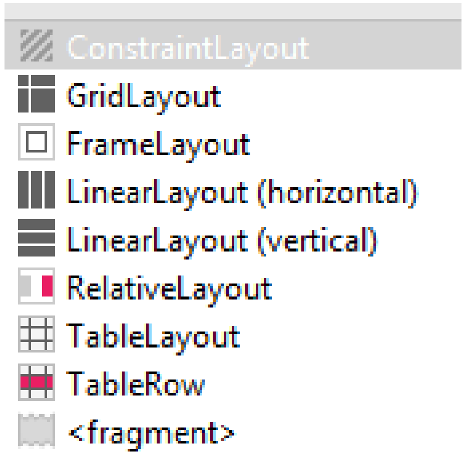
`android.view.ViewGroup`

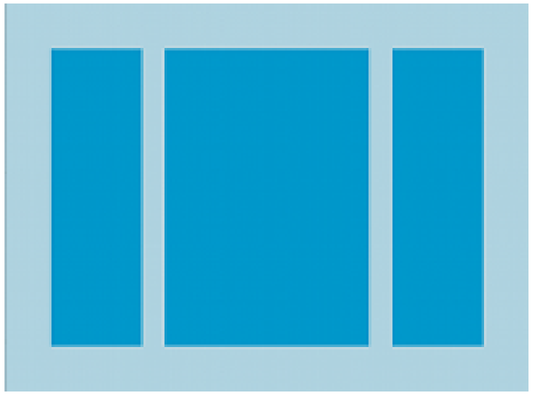
- Abstract container class
- Includes layout functionality directly
- Subclasses:

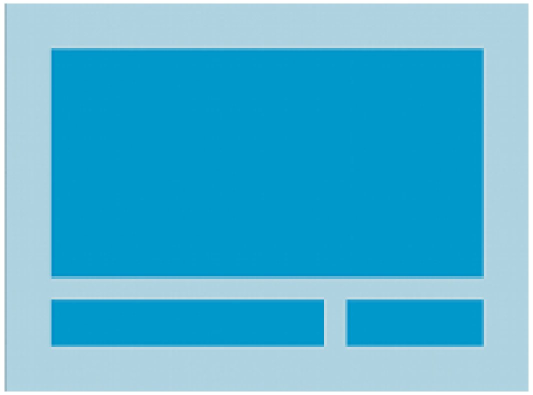
```
LinearLayout, RelativeLayout, GridLayout, …
```

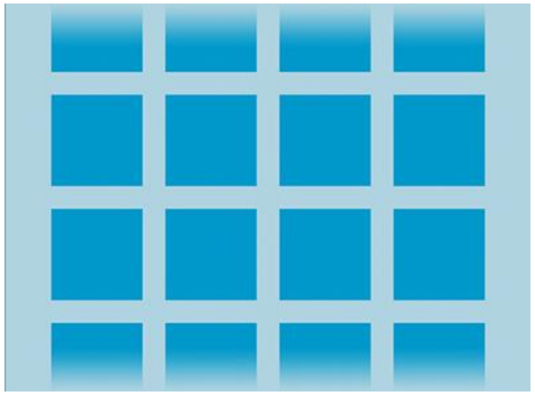
Common Layouts

- Defines the structure for a user interface in your app
- Can be nested







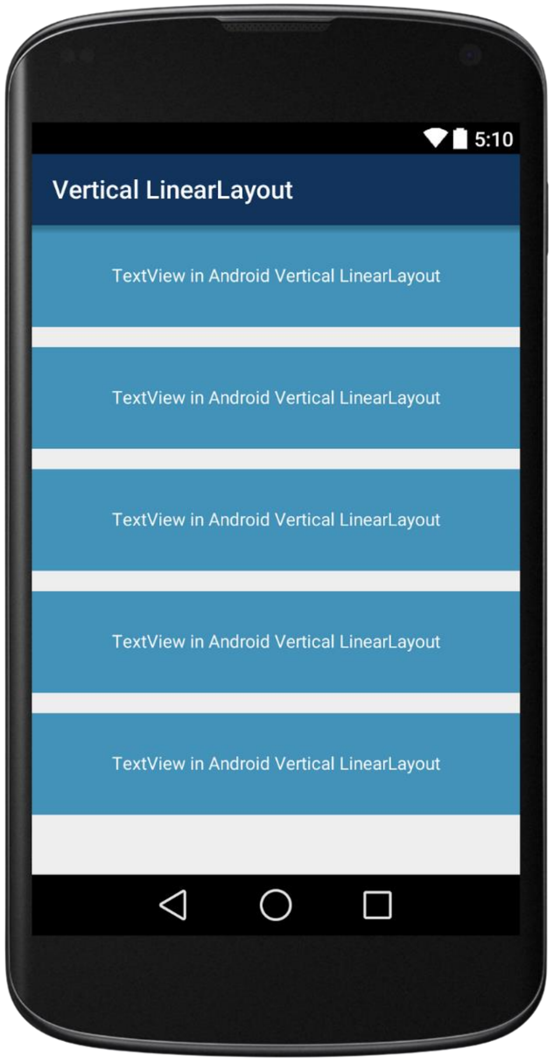


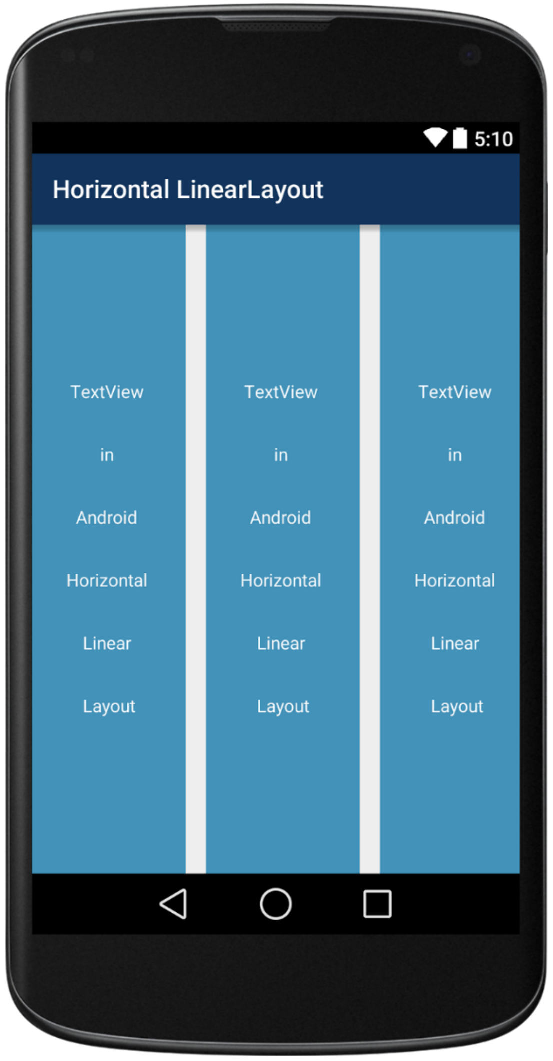
Linear Layout
A layout that organizes its children into a single horizontal or vertical row

Relative Layout
Enables us to specify the location of child objects relative to each other or to the parent.

Grid View
Displays items in a two-dimensional, scrollable grid

Common Layouts

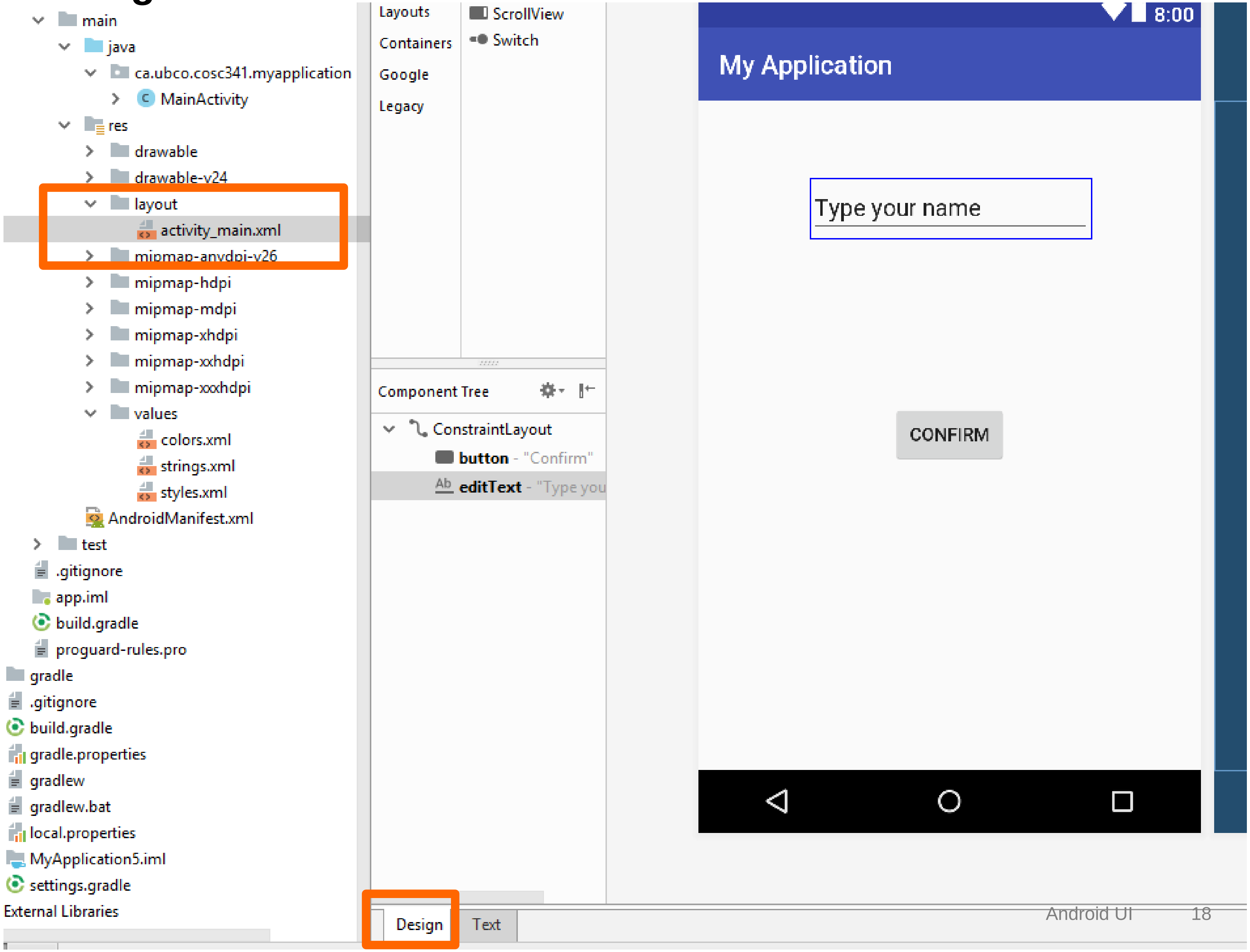

5:10
Vertical LinearLayout
TextView in Android Vertical LinearLayout
TextView in Android Vertical LinearLayout
TextView in Android Vertical LinearLayout
TextView in Android Vertical LinearLayout
TextView in Android Vertical LinearLayout


5:10
Horizontal LinearLayout
TextView in Android Horizontal Linear Layout
TextView in Android Horizontal Linear Layout
TextView in Android Horizontal Linear Layout

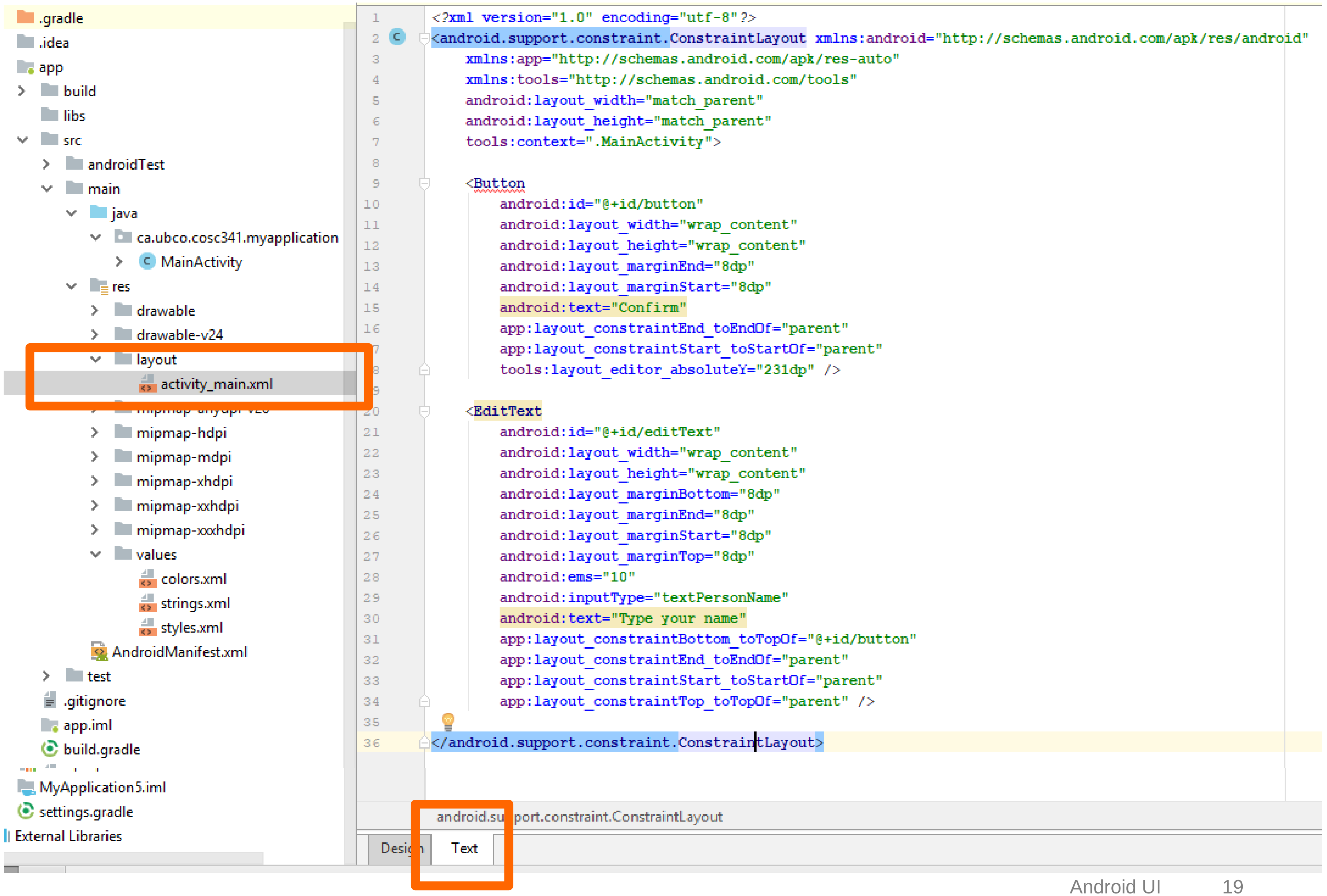
UI Definition and Layout

- Layout can be handled in one of two ways:
  - **Programmatically**. You write code to instantiate ViewGroups, Views and bind them together (like in Java Swing).
  - **Use XML to describe your layout**. In XML describe the screen elements (view groups and views) along with properties, and then tell your application to dynamically load it.
- Using XML is the preferred way
  - Android Studio includes a GUI builder to make this easier!

Editing XML


main
java
ca.ubco.cosc341.myapplication
MainActivity
res
drawable
drawable-v24
layout
activity_main.xml
mipmap-anydpi-v26
mipmap-hdpi
mipmap-mdpi
mipmap-xhdpi
mipmap-xxhdpi
mipmap-xxxhdpi
values
colors.xml
strings.xml
styles.xml
AndroidManifest.xml
test
.gitignore
app.iml
build.gradle
proguard-rules.pro
gradle
.gitignore
build.gradle
gradle.properties
gradlew
gradlew.bat
local.properties
MyApplication5.iml
settings.gradle
External Libraries
Layouts
Containers
Google
Legacy
ScrollView
Switch
Component Tree
ConstraintLayout
button - "Confirm"
editText - "Type you
8:00
My Application
Type your name
CONFIRM
Design
Text

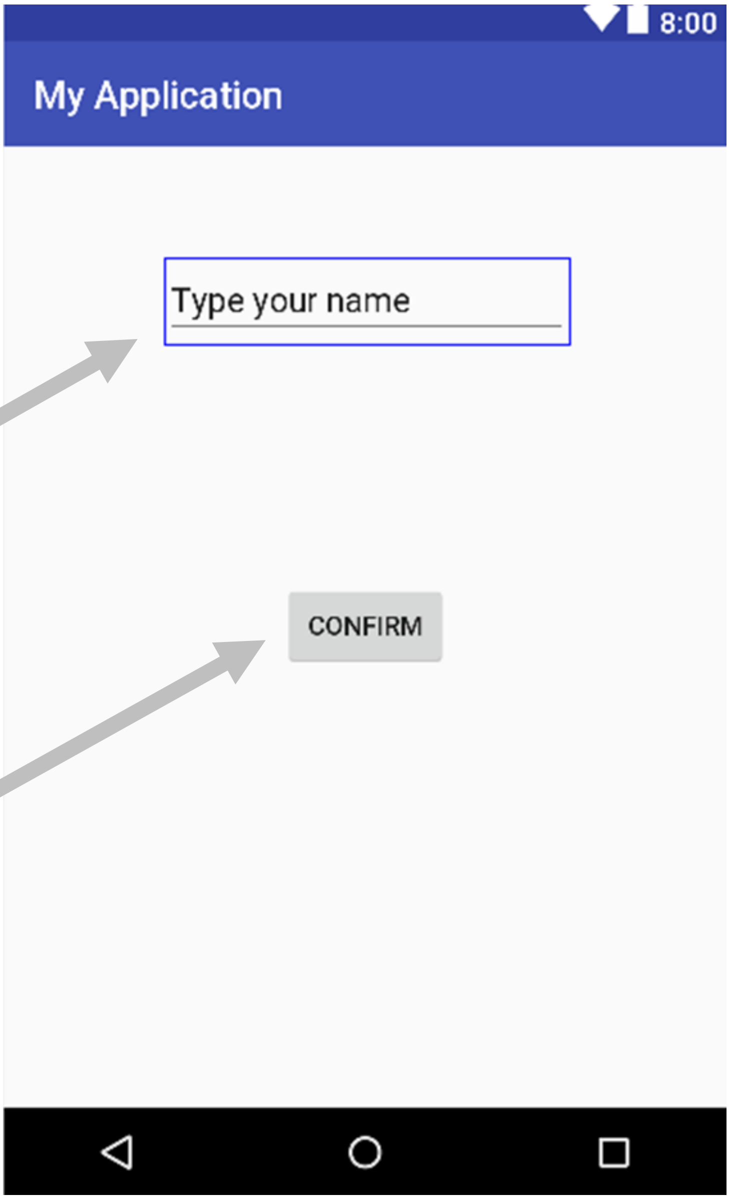
Editing XML: XML Version



```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    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"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:text="Confirm"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        tools:layout_editor_absoluteY="231dp" />

    <EditText
        android:id="@+id/editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Type your name"
        app:layout_constraintBottom_toTopOf="@+id/button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

Layout Example

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    …
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:text="Type your name“
        ….
        />

   <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Confirm"
        …/>


</android.support.constraint.ConstraintLayout>
```



Layout

- When you compile your app, each XML layout file is compiled into a View resource
- calling setContentView(), passing it the reference to your layout resource in the form of: R.layout.*layout_file_name*.
- app/java/MainActivity.java

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
```

View (Widget)

Properties:

- Background color, text, font, alignment, size, padding, margin, etc

Event Listeners :

- respond to various events such as: click, long-click, focus change, etc.

Set focus:

- Set focus on a specific view requestFocus() or use XML tag <requestFocus />

Visibility:

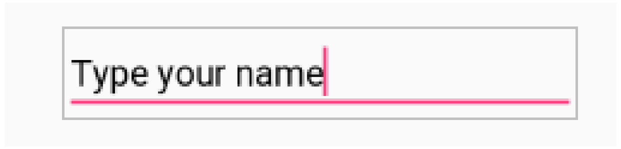
- You can hide or show views using setVisibility(…).

Views: TextViews

```
<TextView
    android:id="@+id/txtHello"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!" />
```

```
TextView helloTextView = findViewById(R.id.txtHello);
helloTextView.setText(“COSC 341");
```

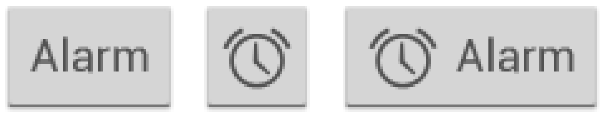
Views: EditText



```
<EditText
    android:id="@+id/name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content”
    android:inputType="textPersonName"
    android:text=”@string/name” >
    <requestFocus/>
 <EditText/>
```

```
EditText nameView = findViewById(R.id.name);
Text name = nameView.getText().toString();
```

Views: Buttons



```
<Button
    android:id="@+id/btnAlarm"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/alarm" />
```

https://developer.android.com/guide/topics/ui/controls/button.html

Responding to Button Events

- *Option 1*

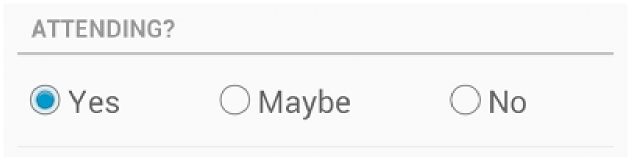
```
<Button
    android:id="@+id/btnAlarm"
    ……
    android:onClick="sendMessage"/>

 /** Called when the user touches the button */
 public void sendMessage(View view) {
     // Do something in response to button click
 }
```

- *Option 2*

```
Button button = (Button) findViewById(R.id.btnAlarm);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // Do something in response to button click
    }
});
```

Radio Buttons

```
<RadioGroup
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <RadioButton
       android:id="@+id/radio_yes"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       …
       android:onClick="onRadioButtonClicked"
       android:text="@string/yes" />
    <RadioButton
       android:id="@+id/radio_no"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       …
       android:onClick="onRadioButtonClicked"
       android:text="@string/no" />
</RadioGroup>
```

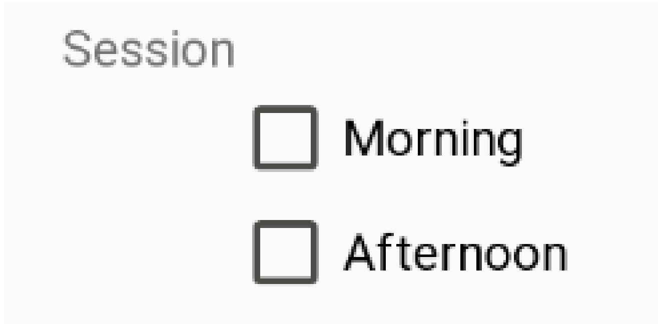


Radio Buttons

```java
public void onRadioButtonClicked(View view) {
    // Is the button now checked?
    boolean checked = ((RadioButton) view).isChecked();

    // Check which radio button was clicked
    switch (view.getId()) {
        case R.id.radio_yes:
            if (checked)
                // code for yes
                break;
        case R.id.radio_maybe:
            if (checked)
                // code for may be
                break;
        case R.id.radio_no:
            if (checked)
                // code for no
                break;
    }
}
```

Checkboxes

```
<CheckBox
    android:id="@+id/checkbox_morning"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onCheckboxClicked"
    android:text="@string/morning" />

<CheckBox
    android:id="@+id/checkbox_afternoon"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onCheckboxClicked"
    android:text="@string/afternoon" />
```



https://developer.android.com/guide/topics/ui/controls/checkbox.html

Checkboxes

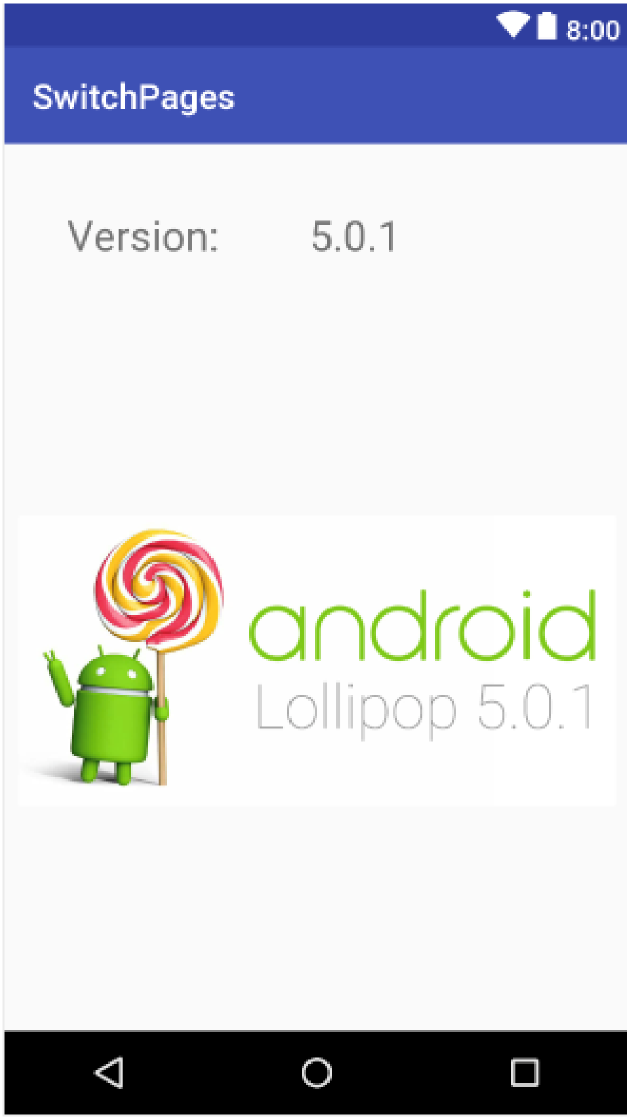
```
public void onCheckboxClicked(View view) {
    // Is the view now checked?
    boolean checked = ((CheckBox) view).isChecked();

    // Check which checkbox was clicked
    switch (view.getId()) {
        case R.id.checkbox_morning:
            if (checked)
            // Add morning session
        else
            // Remove morning session
            break;
        case R.id.checkbox_afternoon:
            if (checked)
            // Add afternoon session
        else
            // Remove afternoon session
            break;
    }
}
```

https://developer.android.com/guide/topics/ui/controls/checkbox.html

ImageView

- Save image resources to drawable folder
  - app/src/main/res/drawable/

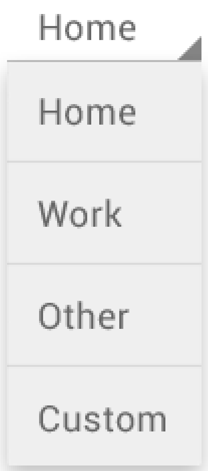
```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content”
    android:src="@drawable/lollipop" />
```



Spinners

- Spinners provide a quick way to select one value from a set
- Touching the spinner displays a dropdown menu with all other available values

```
<Spinner
    android:id="@+id/planets_spinner"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="my_array">
        <item>Home</item>
        <item>Work</item>
        <item>Other</item>
        <item>Custom</item>
    </string-array>
</resources>
```



Spinners

```
Spinner spinner = (Spinner) findViewById(R.id.spinner);

// Create an ArrayAdapter using the string array and a default
spinner layout
ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this,
        R.array.my_array, android.R.layout.simple_spinner_item);

// Specify the layout to use when the list of choices appears
adapter.setDropDownViewResource(android.R.layout.simple_spinner_
dropdown_item);

// Apply the adapter to the spinner
spinner.setAdapter(adapter);
```

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
  - 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

https://developer.android.com/guide/topics/ui/layout/linear.html

Key Attributes

- **Orientation**

Should the layout be a column or a row? Use "horizontal" for a row, "vertical" for a column.

- **Fill model**

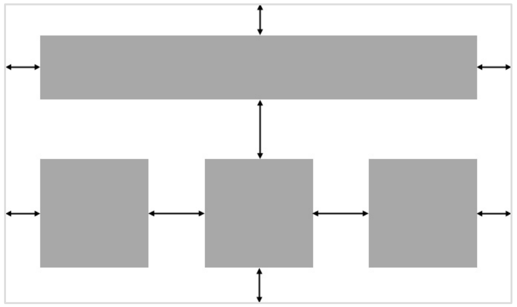
MATCH_PARENT: the view wants to be as big as its parent

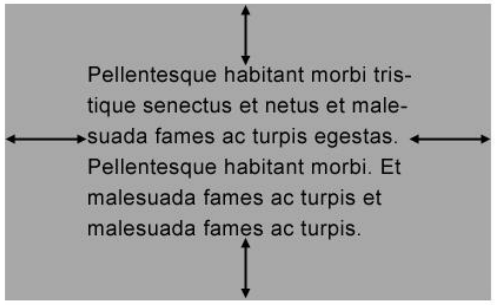
WRAP_CONTENT: the view wants to be just large enough to fit its own internal content

Key Attributes

- **Padding/margin**

Setting padding/margin



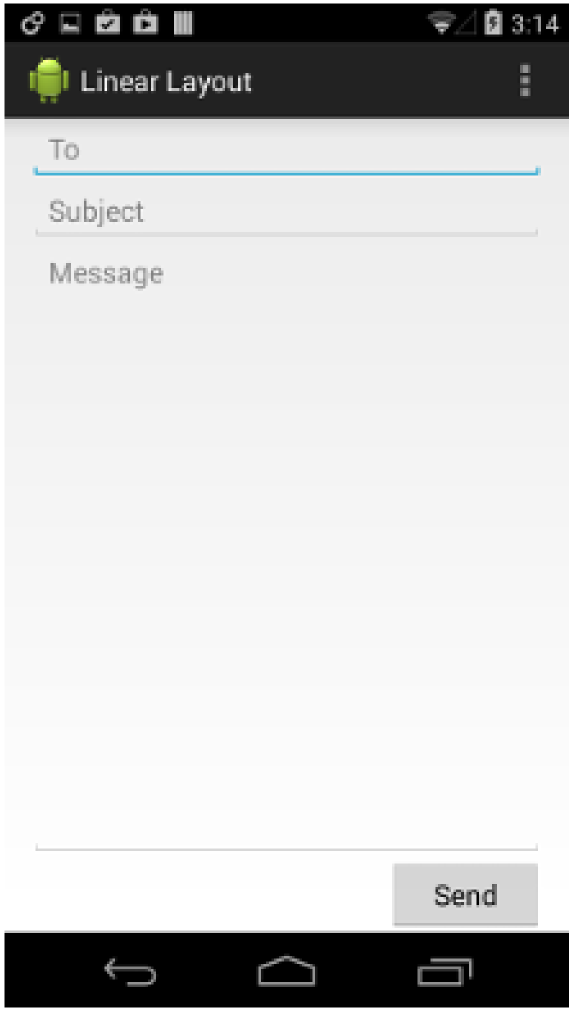

Pellentesque habitant morbi tris-
tique senectus et netus et male-
suada fames ac turpis egestas.
Pellentesque habitant morbi. Et
malesuada fames ac turpis et
malesuada fames ac turpis.

- **Weight**

android:layout_weight attribute assigns an "importance" value to a view in terms of how much space it should occupy on the screen.

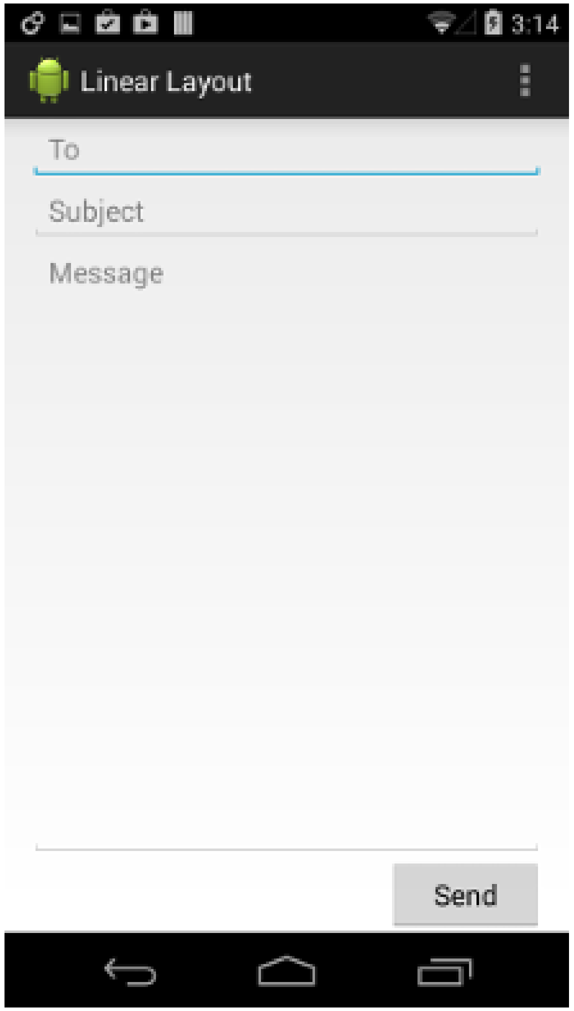
LinearLayout

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        …
     />
     <EditText
        …
     />
    <Button
        …
     />
</LinearLayout>
```



LinearLayout

```
<LinearLayout …>
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```



Relative Layout

- RelativeLayout is a view group that displays child views in relative positions.
- The position of each view can be specified as
  - relative to sibling elements (such as to the left-of or below another view)
  - in positions relative to the parent RelativeLayout area (such as aligned to the bottom, left or center).

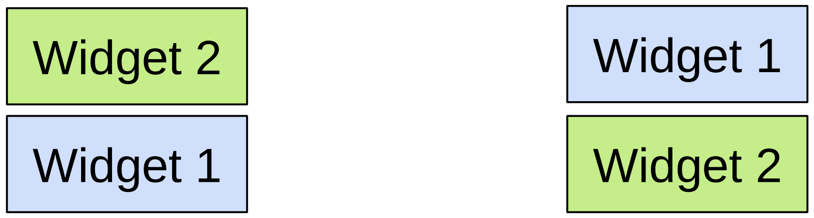
https://developer.android.com/guide/topics/ui/layout/linear.html

View Positioning

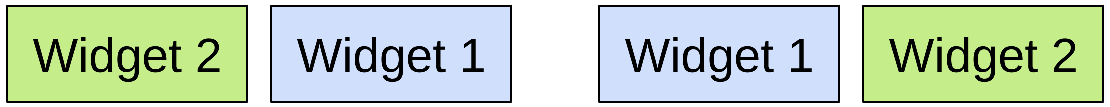
- RelativeLayout lets child views specify their position relative to the parent view or to each other (specified by ID).
- By default, all child views are drawn at the top-left of the layout
- Example of some layout properties :
  - android:layout_alignParentTop
  - android:layout_centerVertical
  - android:layout_below
  - android:layout_toRightOf
  - More: RelativeLayout.LayoutParams

View Positioning in Relative Layout

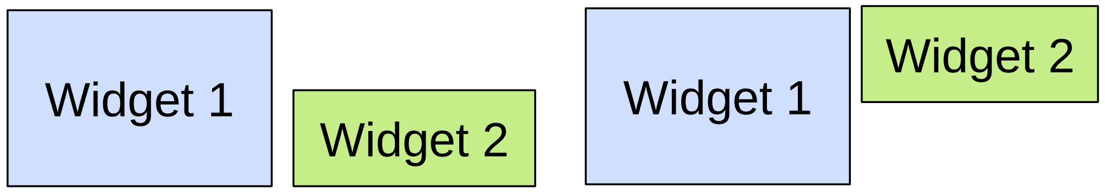
android:layout_above
android:layout_below



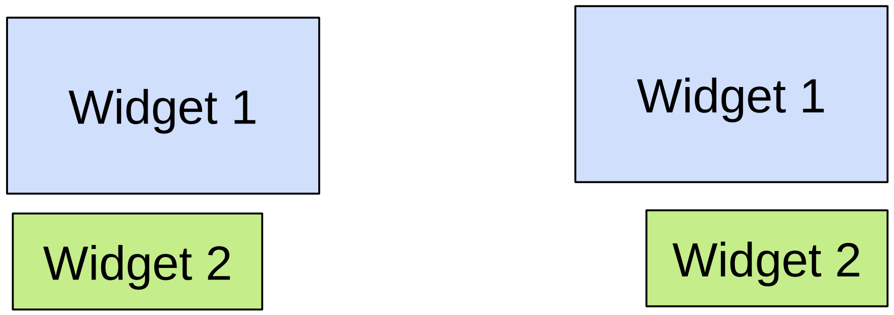
android:layout_toLeftOf
android:layout_toRightOf



android:layout_alignBottom
android:layout_alignTop



android:layout_alignLeft
android:layout_alignRight



Relative layout alignment parameters

android:layout_alignParentTop

android:layout_alignParentBottom

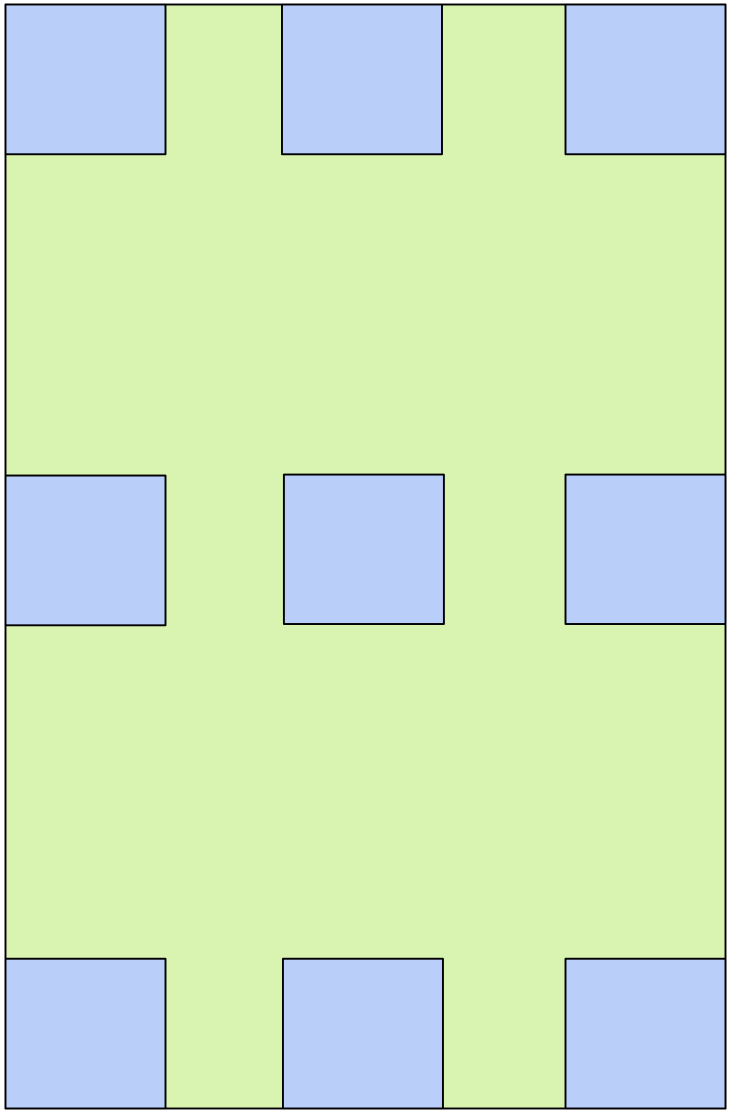
android:layout_alignParentLeft

android:layout_alignParentRight

android:layout_centerInParent

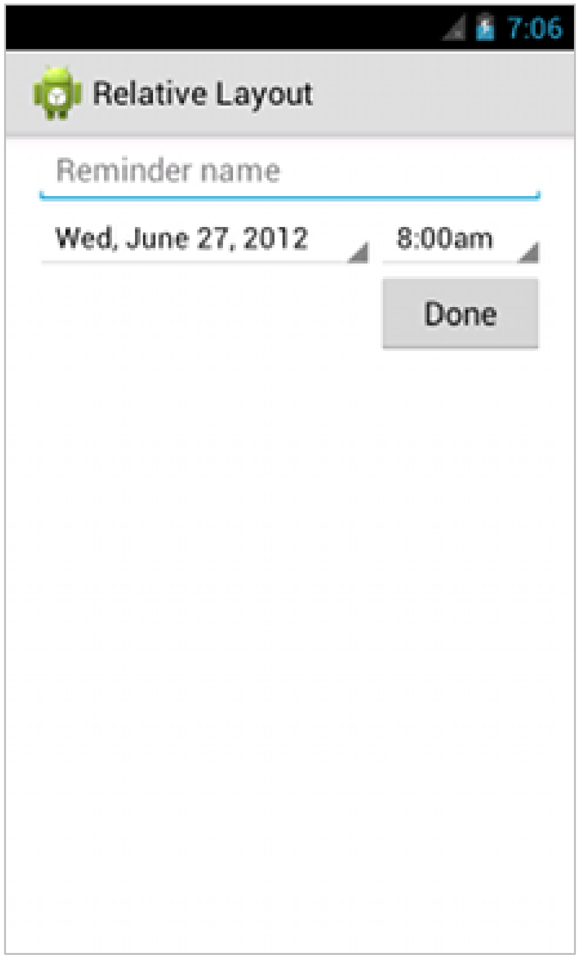
android:layout_centerVertical

android:layout_centerHorizontal

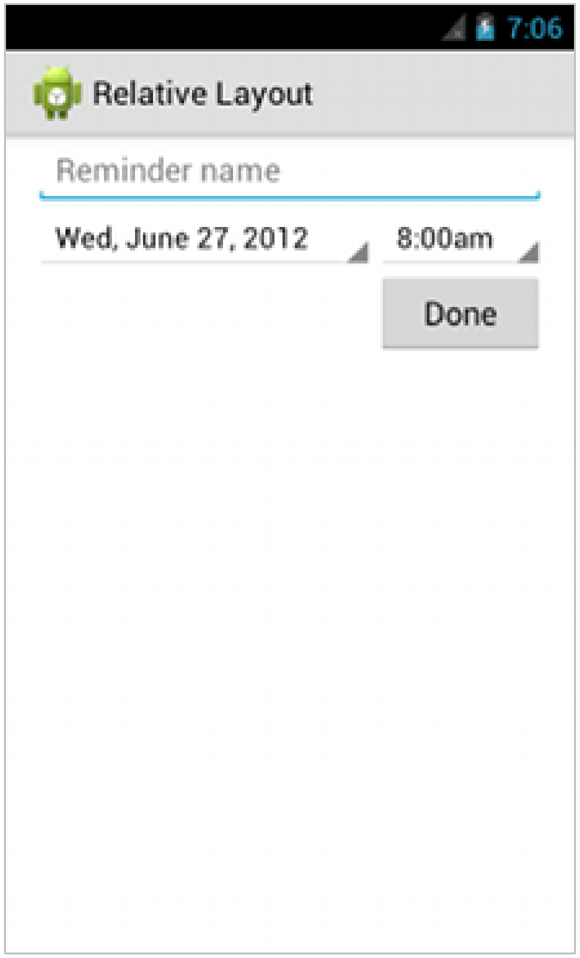


Relative Layout

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        …
    />
    <Spinner
        …
    />
   <Spinner
        …
    />
    <Button
        …
    />
</RelativeLayout>
```



```
<RelativeLayout …
    <EditText
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content” />
    <Spinner
        android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Spinner
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
         android:layout_toLeftOf="@+id/times”
        android:layout_alignParentLeft="true” />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```

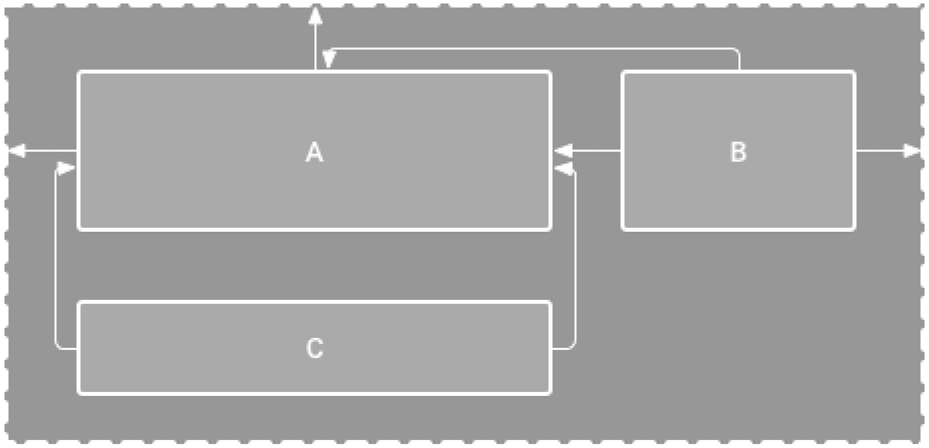


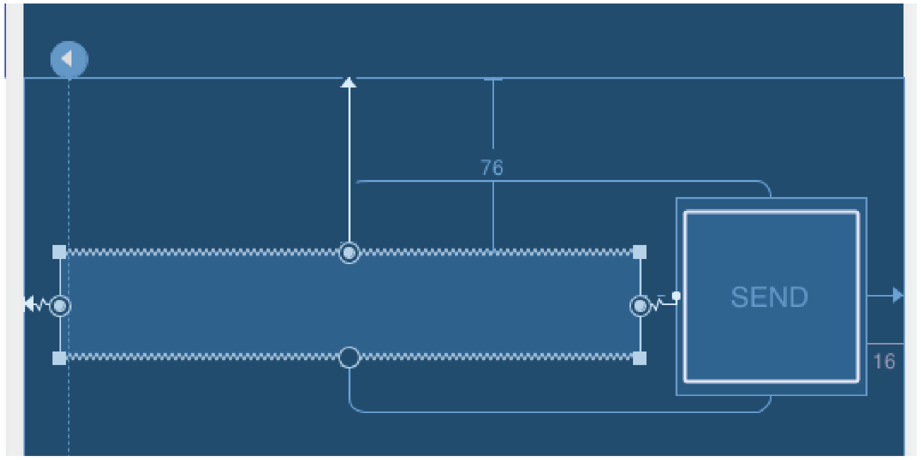
Nested Layout

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
       android:id="@+id/name_title"                        Views
       android:text=“@string/name">
       ……
    </EditText>
       ……
    <LinearLayout
       android:layout_below="@+id/session">
       ………
        <CheckBox
            android:id="@+id/checkbox_morning"
            android:text="@string/morining”                    Views
           ……  />
        <CheckBox
           ……  />
    </LinearLayout>
</RelativeLayout>
```

Constraint Layout

- Similar to RelativeLayout
  - All views are laid out according to relationships with others
- Easier to use with Android Studio’s Layout Editor
  - Default layout for a new layout

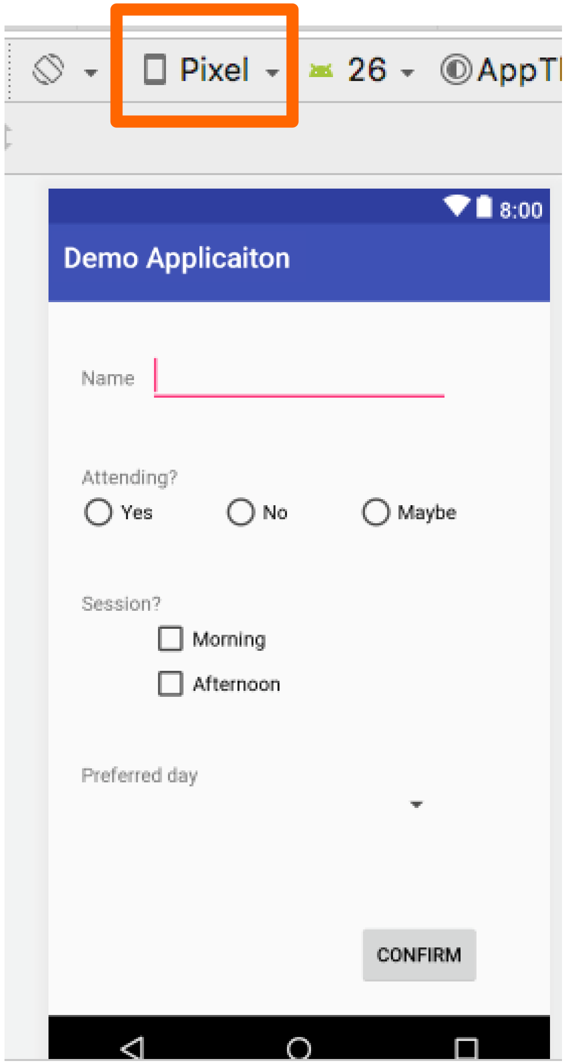


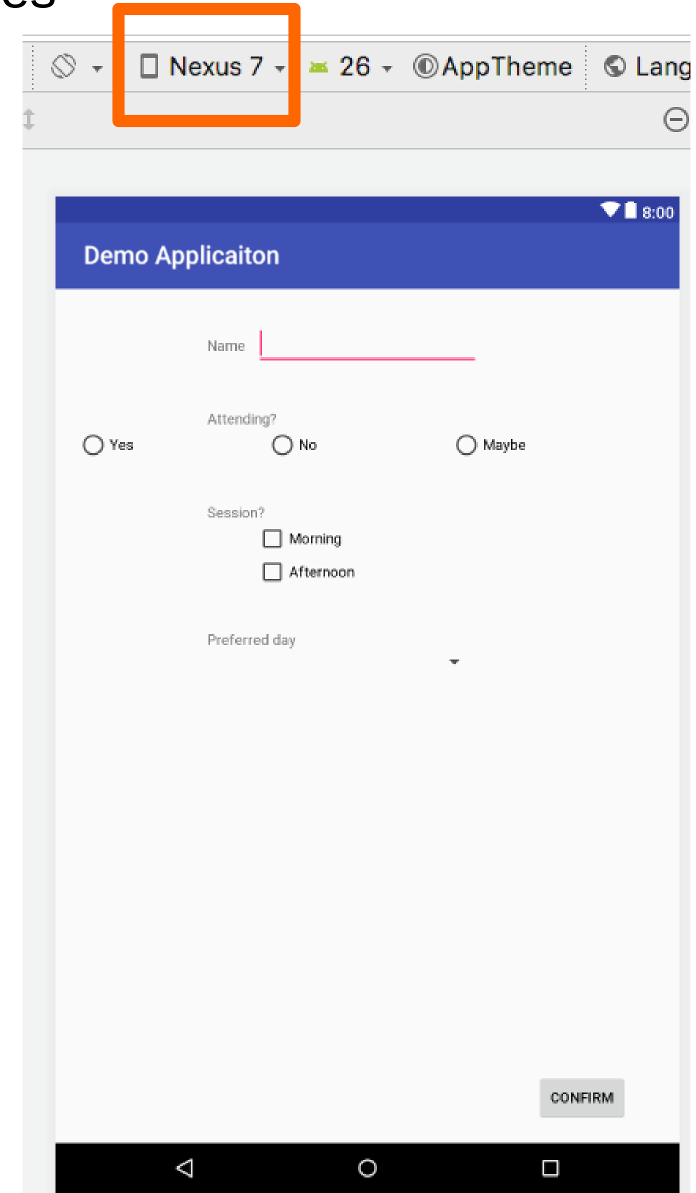


https://developer.android.com/training/constraint-layout/index.html

Layout test

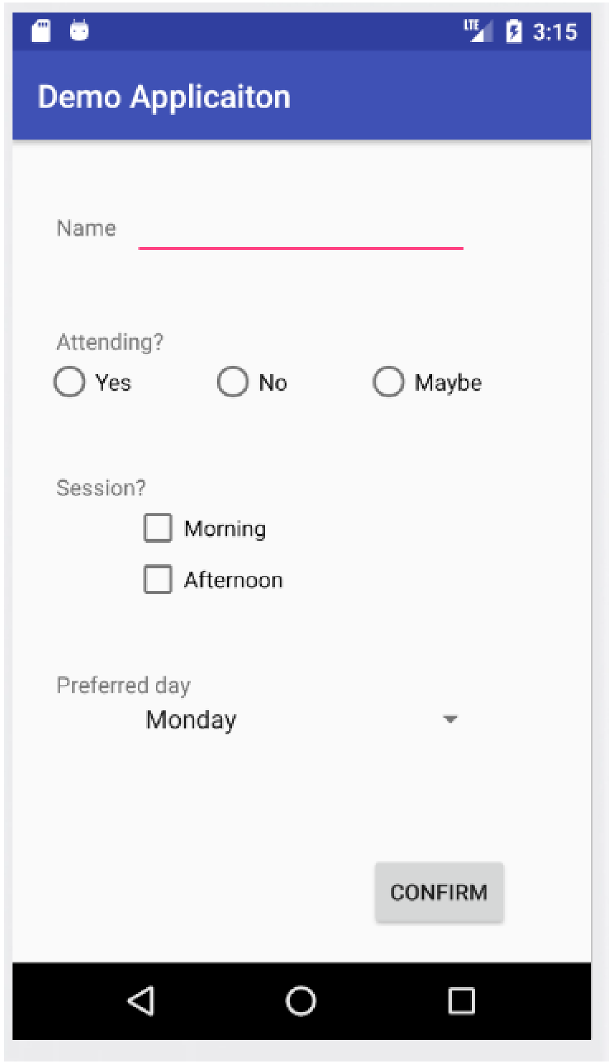
- Check the layout with multiple screen sizes





Demo Application

- Notes
  - TextView
  - EditText
  - RadioButton
  - CheckBox
  - Spinners



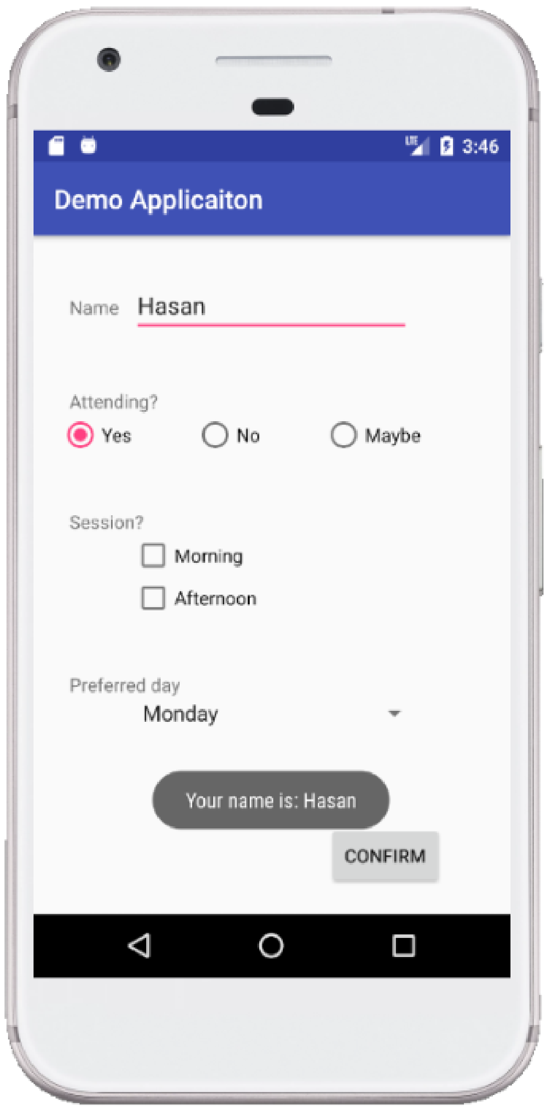
Events

- Android uses the Java event model with additional mobile events
  - Event listener: interface for specific type of event
  - Event handler: registered callback method to handle the event

Event Listener	Event Handler	Type of event
`OnClickListener()`	`onClick()`	Touch, click
`OnLongClickListener()`	`onLongClick()`	Press and hold
`onTouchListener()`	`onTouch()`	Generic touch events can be used for touch_up second_touch

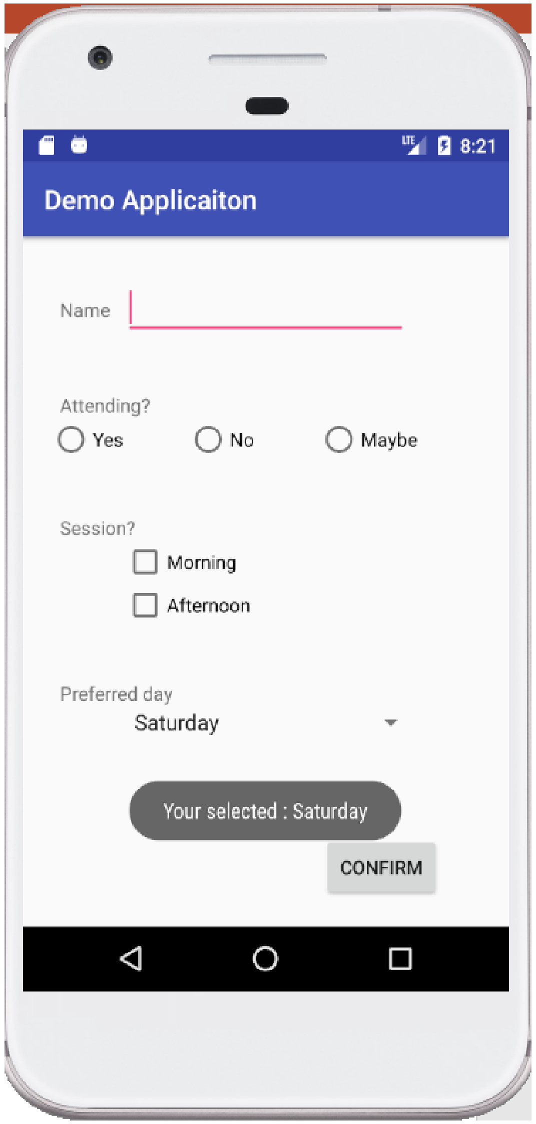
Try-It Activity

- Download the code and run on your machine



```
Toast.makeText(getApplicationContext(),
"Your text: " + variable,
Toast.LENGTH_SHORT).show();
```

Try-It Activity



```
 spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void
onItemSelected(AdapterView<?> parentView, View
selectedItemView, int position, long id) {
                // your code here
                   }

            @Override
            public void
onNothingSelected(AdapterView<?> parentView) {
                // your code here
            }

        });
```

```
parentView.getItemAtPosition(position)
```

Questions?