Summer 2, 2019 - CS 4520/CS5520 – Mobile Application Development

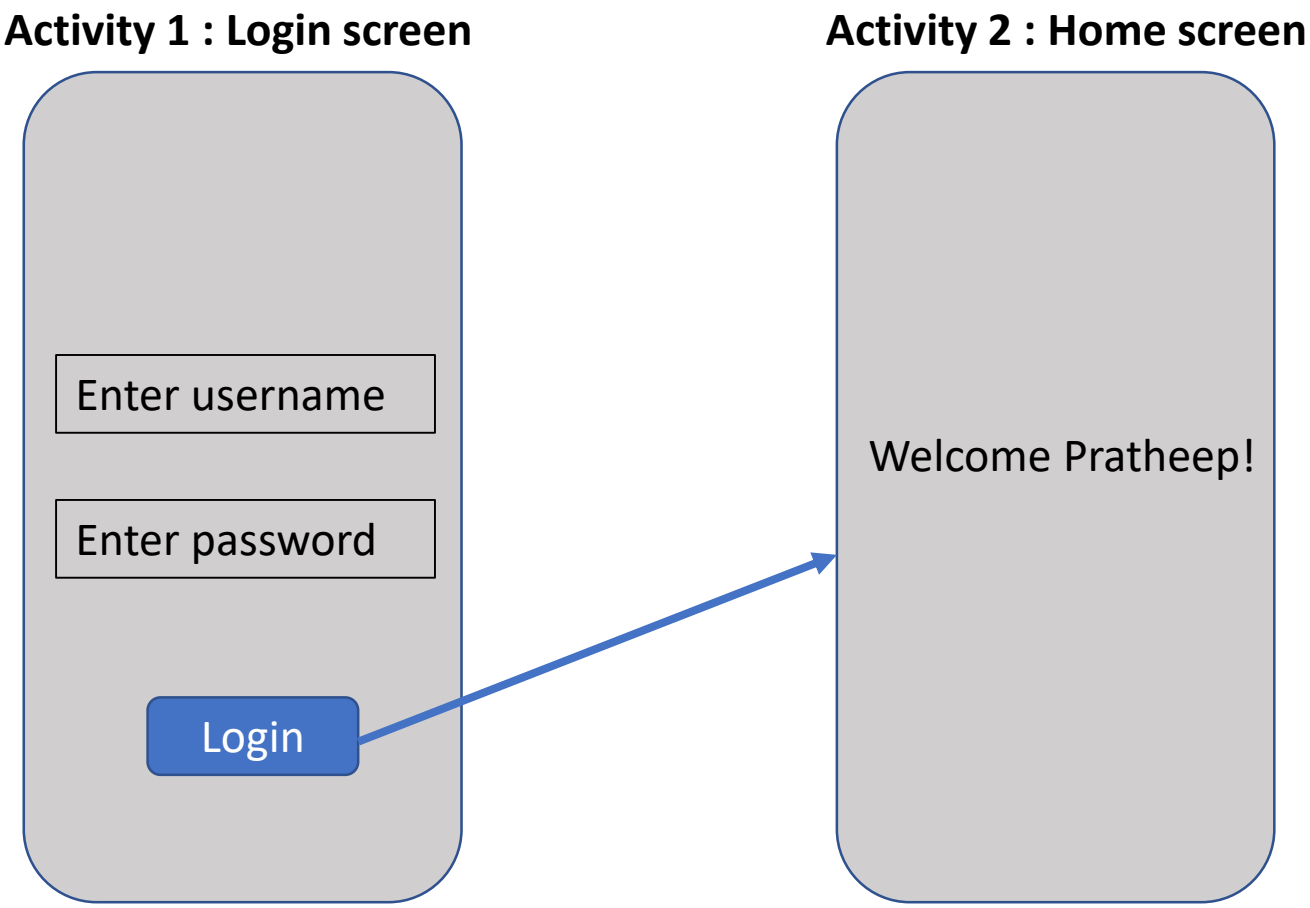
Pratheep Kumar Paranthaman, Ph.D.,

Types of Intents

- **Explicit Intent:**
  - Activity that you want to call is known
  - Typically used to start a component(activity/service) in your app

[interviewquestionanswer.com/android-questions/what-is-an-explicit-intent]

Example – Explicit Intent



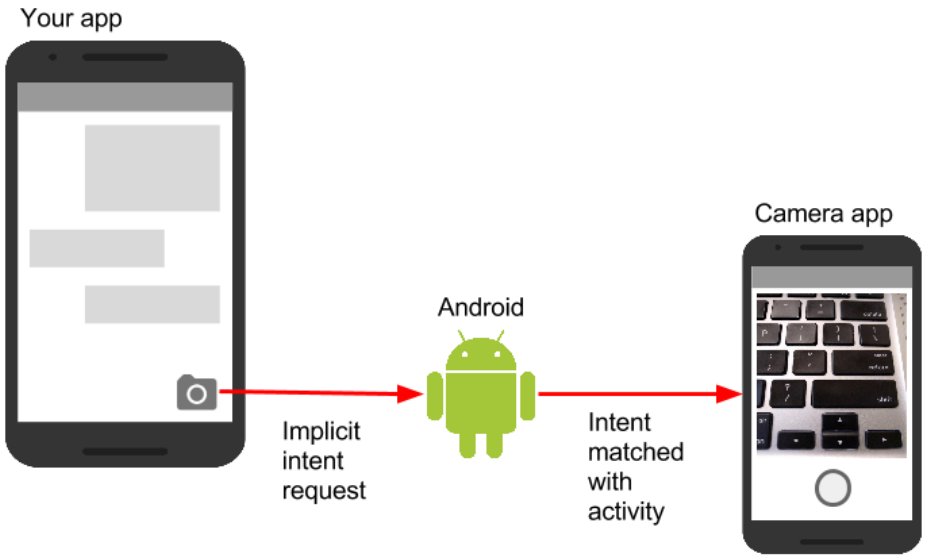
Passing data between activities

Passing data

- putExtra(Key,Value)
- getIntent().getExtra(Key)

Types of Intents

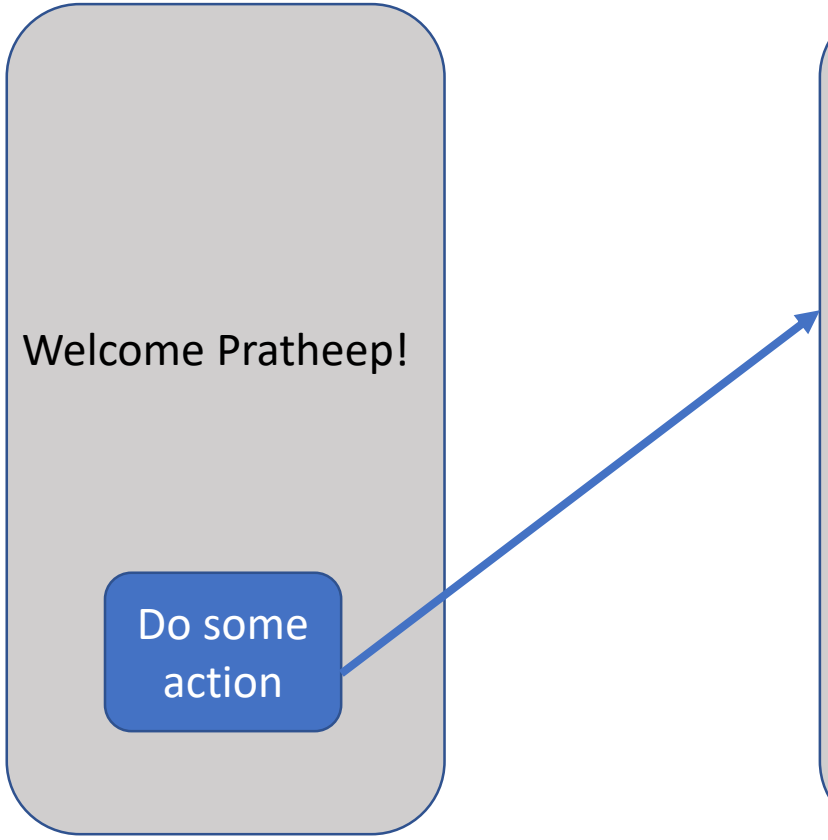
- **Implicit Intent:**
  - Exact name of activity/service to run is unknown
  - **Declare general action** to perform and the component in another app will handle it.



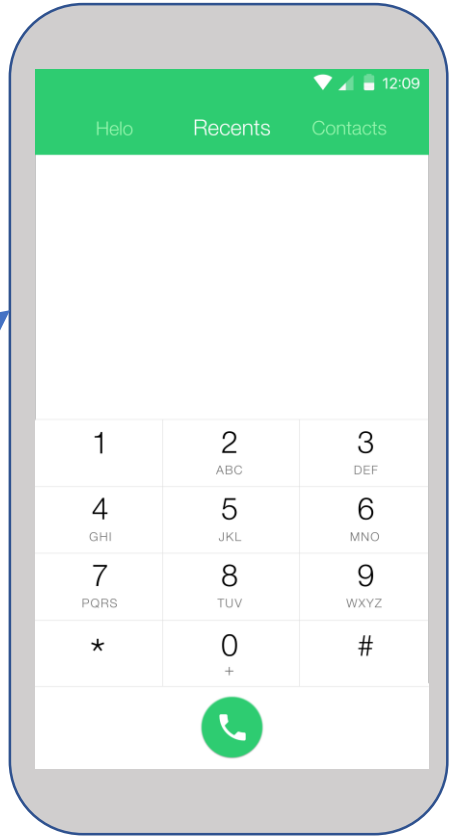
[google-developer-training.github.io/android-developer-fundamentals-course-concepts/en/Unit%201/23_c_activities_and_implicit_intents.html]

Example – Implicit Intent

Activity 1 : Home Screen



Activity 2 : Action Page

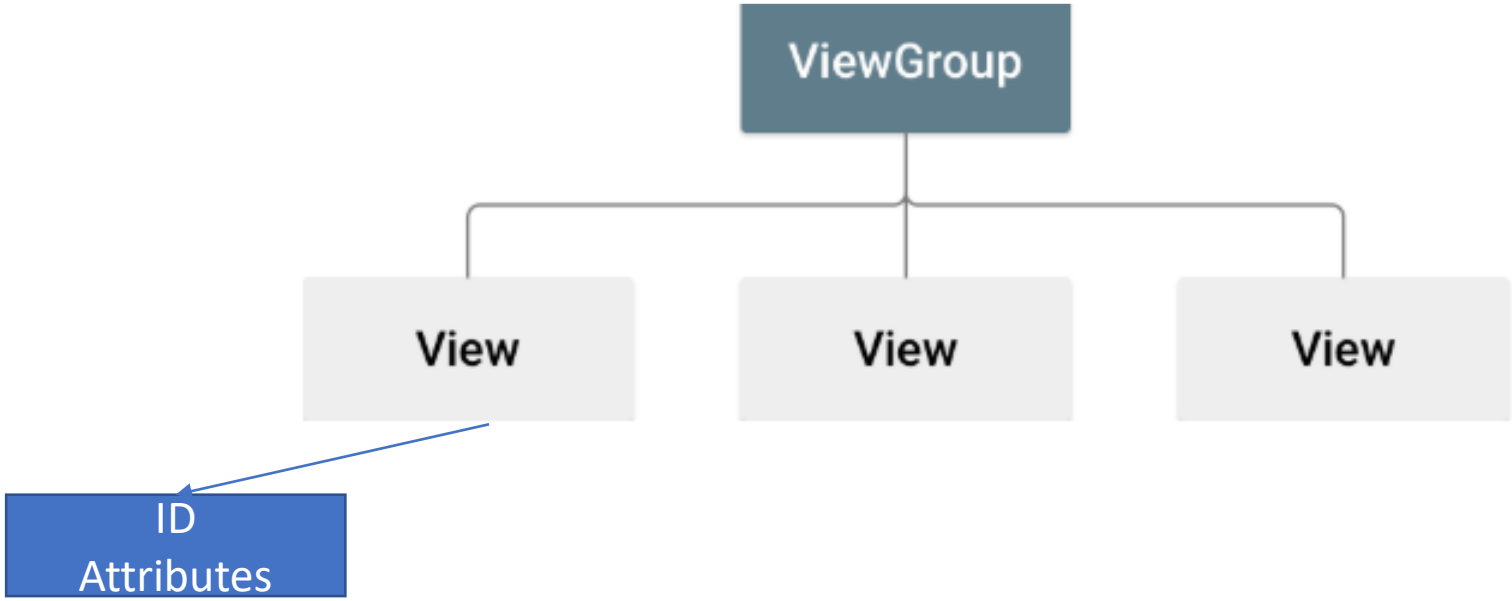


Layouts

Layout

UI Layout with view Hierarchy

- Create them using the XML
- Instantiate during the runtime

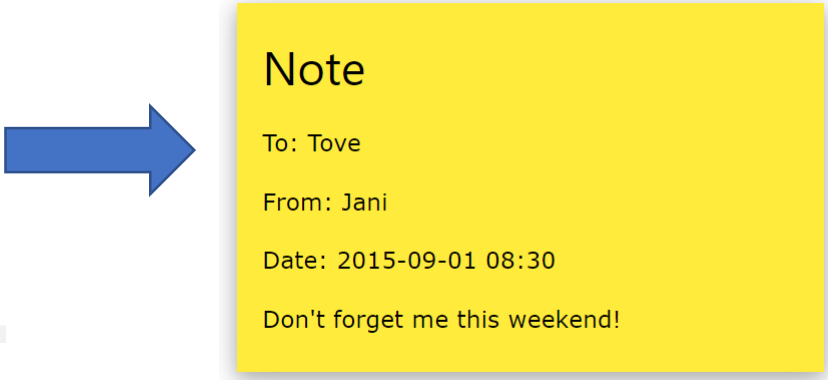


[developer.android.com/guide/topics/ui/declaring-layout]

XML in Android

- XML – Extensible Markup Language
  - Lightweight language – doesn't make your layout heavy
  - Separates UI from logic

```
<note>
  <date>2015-09-01</date>
  <hour>08:30</hour>
  <to>Tove</to>
  <from>Jani</from>
  <body>Don't forget me this weekend!</body>
</note>
```



Layouts

- **Units of measurements**

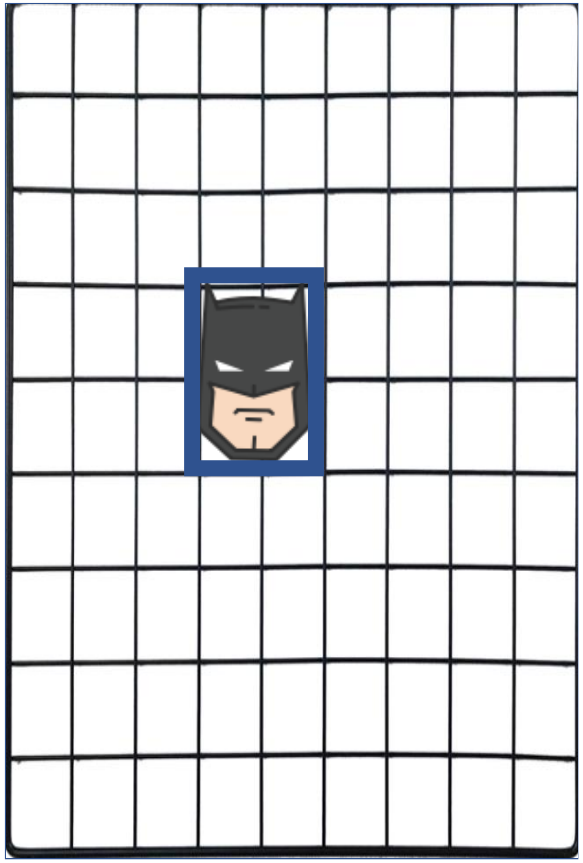
- dp – Density-independent pixel –
  - Specifying dimension of view in your layout
    - Layout_height, Layout_width

- sp – Sacle – independent pixel – similar to dp
  - Specifying font sizes

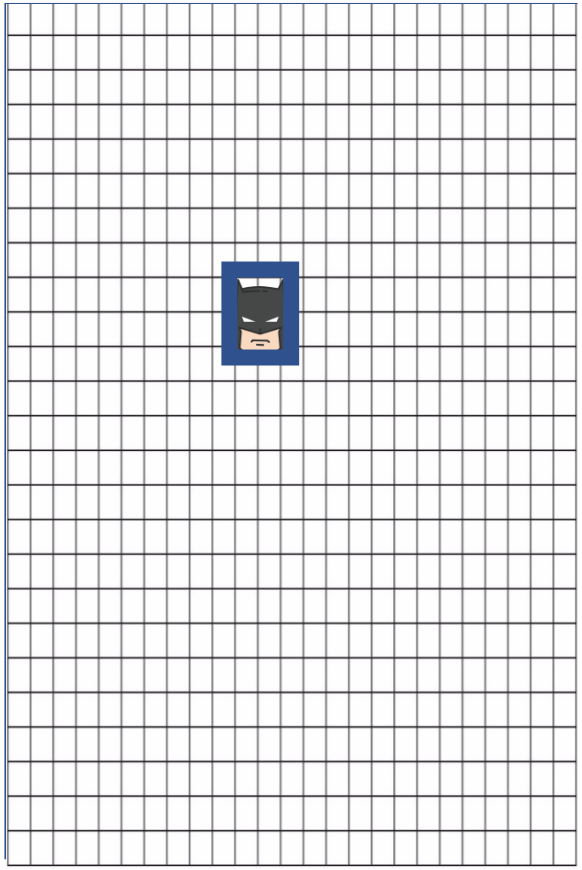
- px – pixel corresponds to actual pixel on Screen
  - Using this is not recommended , as your UI might not render correctly on devices with different screen resolution

Issues with pixels?

Layouts – Density Independent Pixel



MDPI



XXHDPI

Layout Types

- Constraint Layout
- Linear Layout
- Relative Layout
- Frame Layout
- Grid Layout

- Layout using Adapter
  - List View

XML structure

- Let’s start with the smallest component of the layout -“Views”
- Understand how to create view with XML
- Investigate the attributes
- Study the various Layout types

XML Structure in Android

```
<TextView
    android:id="@+id/myTextView"
    android:text="@string/app_name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

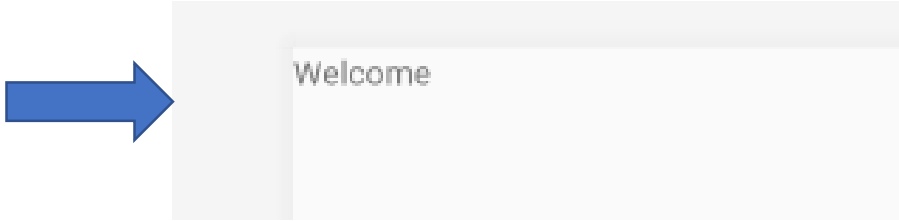
Opening Tag

Attributes

Closing Tag

XML structure in Android

```
<TextView
    android:id="@+id/myTextView"
    android:text="@string/welcomeText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```



XML Structure in Android

- ID

```
android:id="@+id/my_button"
```

- Attributes

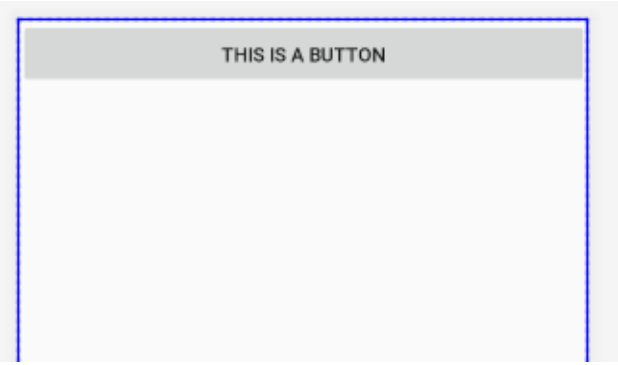
```
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
```

XML attributes for views

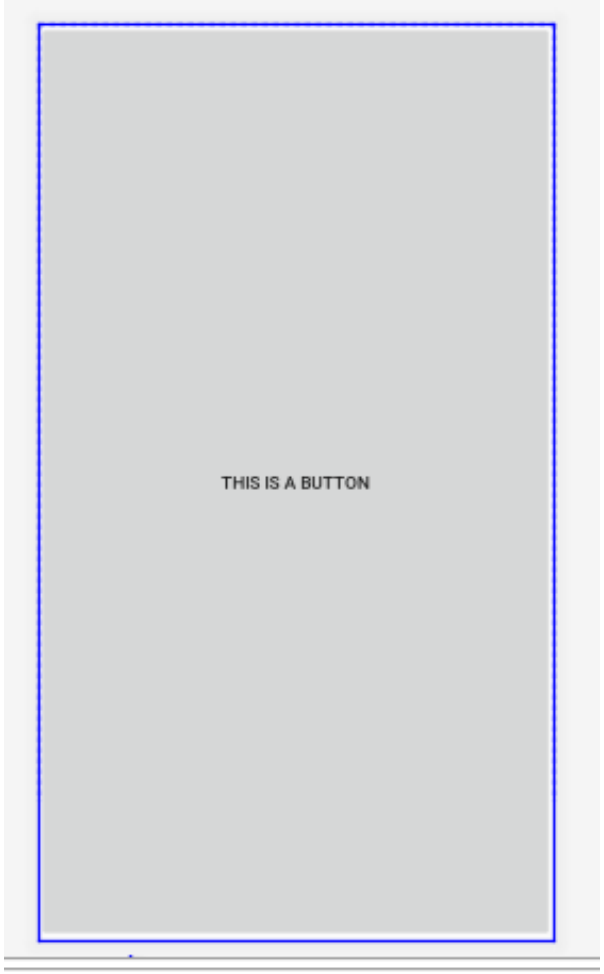
```
<TextView
    android:id="@+id/myTextView"
    android:text="@string/welcomeText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

```
<Button
    android:id="@+id/btn1"
    android:text="This is a Button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

```
<Button
    android:id="@+id/btn1"
    android:text="This is a Button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```



```
<Button
    android:id="@+id/btn1"
    android:text="This is a Button"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

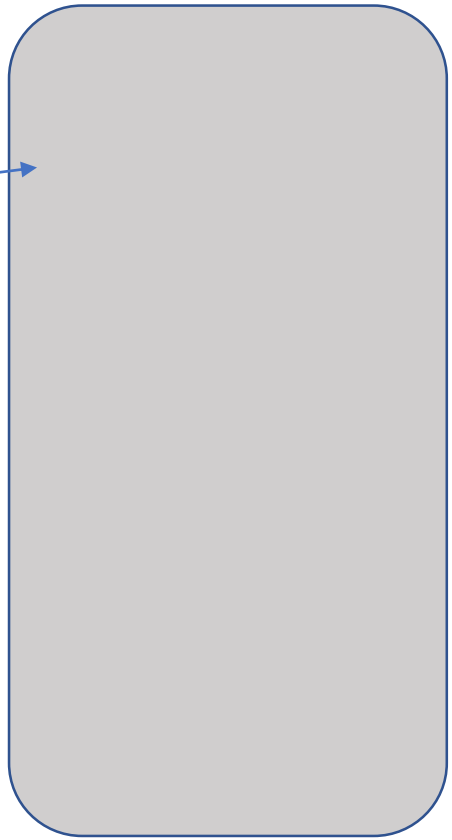


XML attributes for android views

- android:id
- android:layout_width
- android:layout_height
- android:text
- android:background
- android:onClick
- android:padding
- android:margin
- android:textSize

XML structure

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

</LinearLayout>
```



XML structure

Root element - ViewGroup

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              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>
```

View

View

Hello, I am a TextView

Hello, I am a Button

XML Structure in Android

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              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>
```

Let’s try it!

Task

- Change the Constraint layout to Linear layout
- Create a button using XML
- Change the text of the button
- Include an ID for the button
- Set the width of the button to match the parent
- Set the height of the button to wrap content

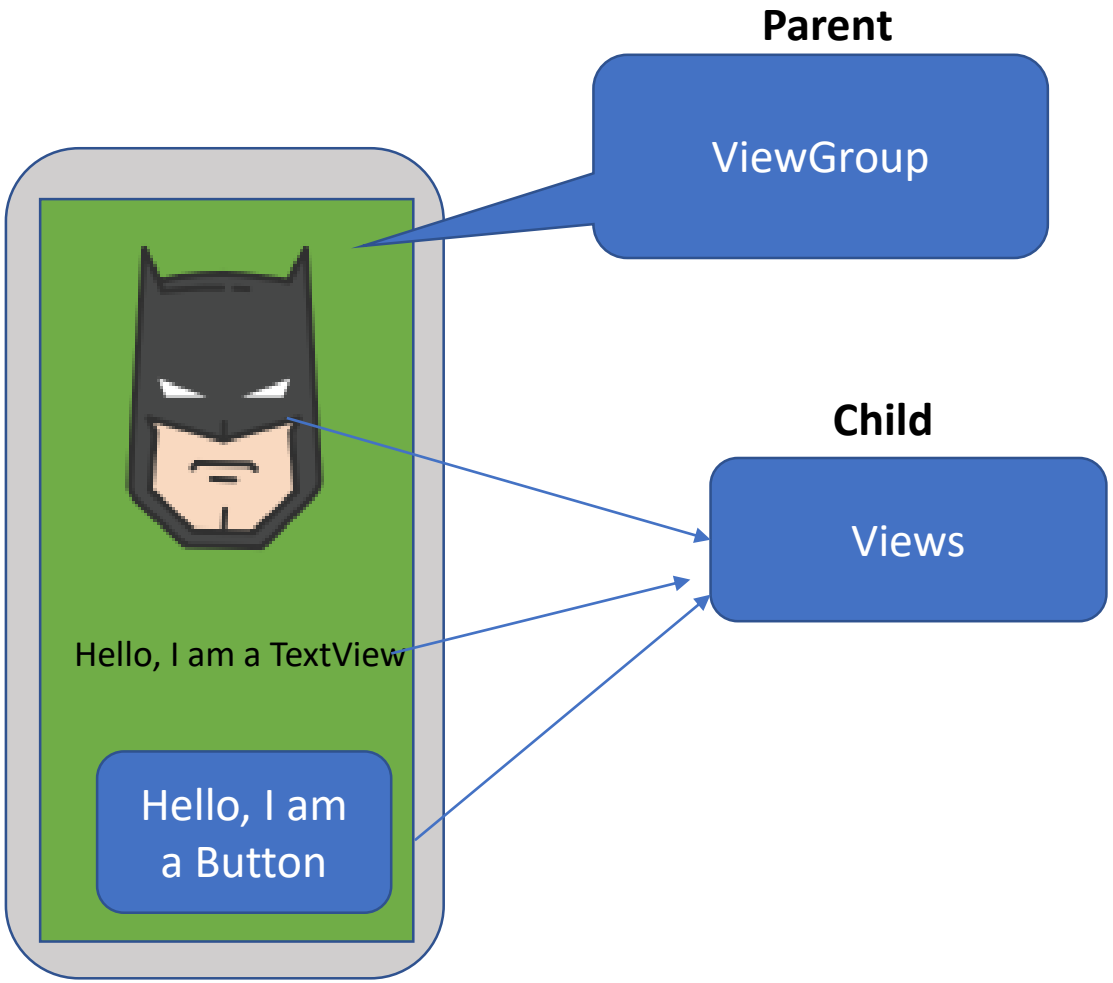
Loading XML in Activity

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


}
```

XML outside layouts

- Manifest
- Strings
- Color
- Style

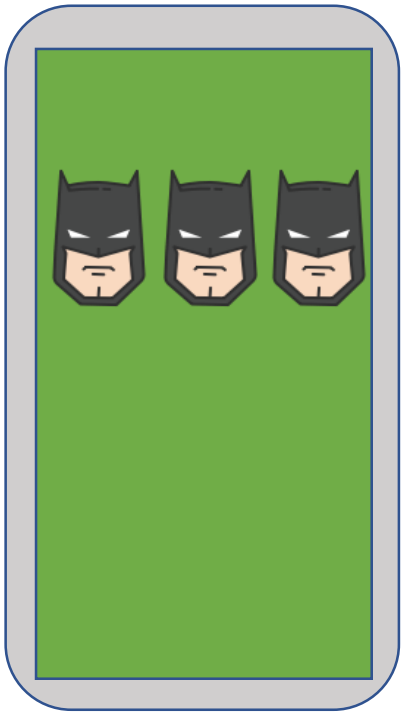
ViewGroups – Layout type



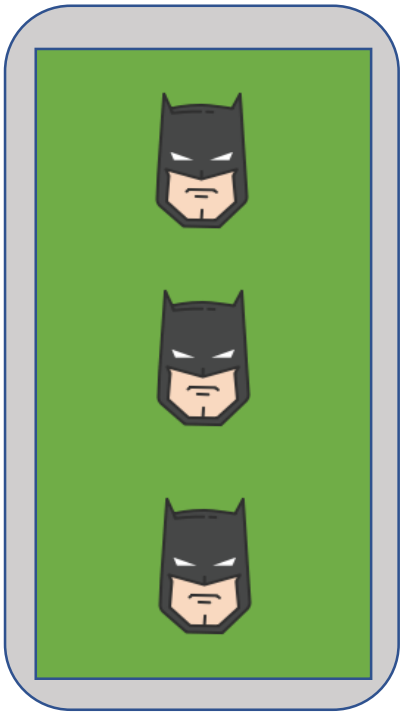
Linear Layout

- Aligns all its children in a single horizontal or vertical row, stacking them one after the other.
- Specify the orientation

android:orientation =“horizontal”



android:orientation =“vertical”

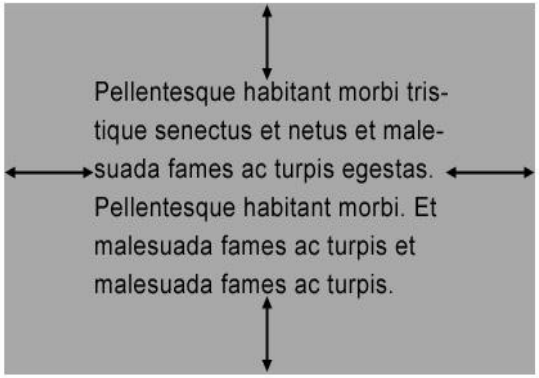


Attributes

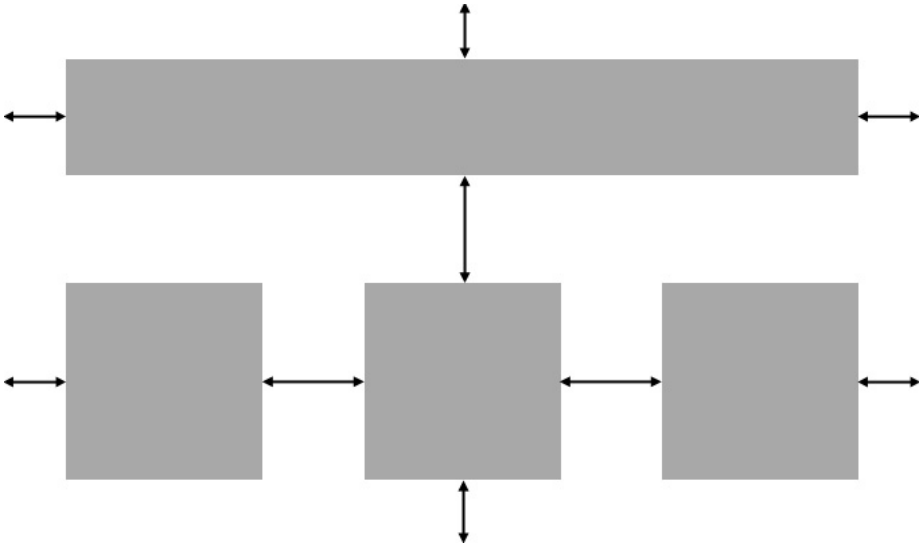
- Gravity
- Layout weight
- Padding
- Margin

Layout Padding Vs. Margin

Padding



Margin

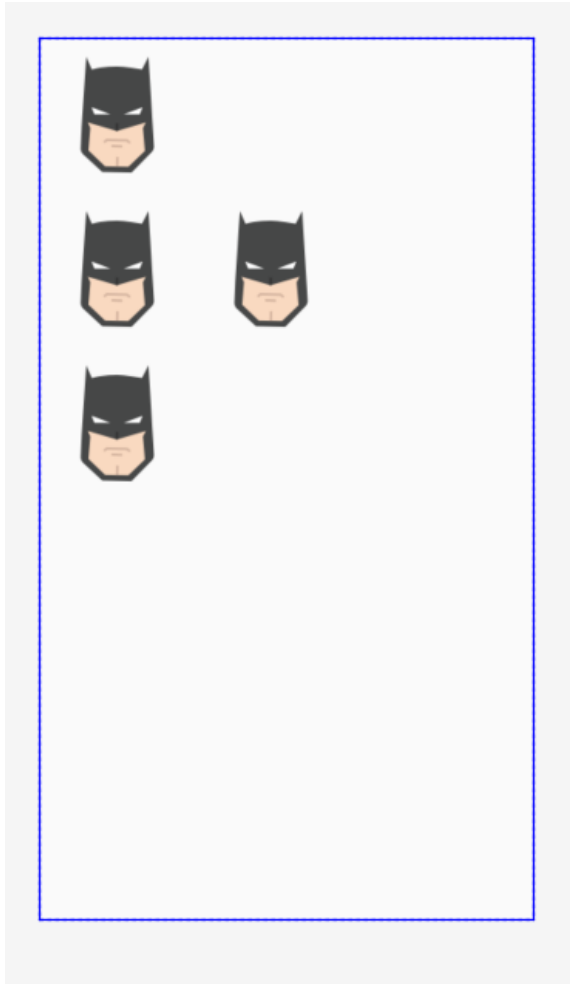


[stackoverflow.com/questions/21959050/android-beginner-difference-between-padding-and-margin]

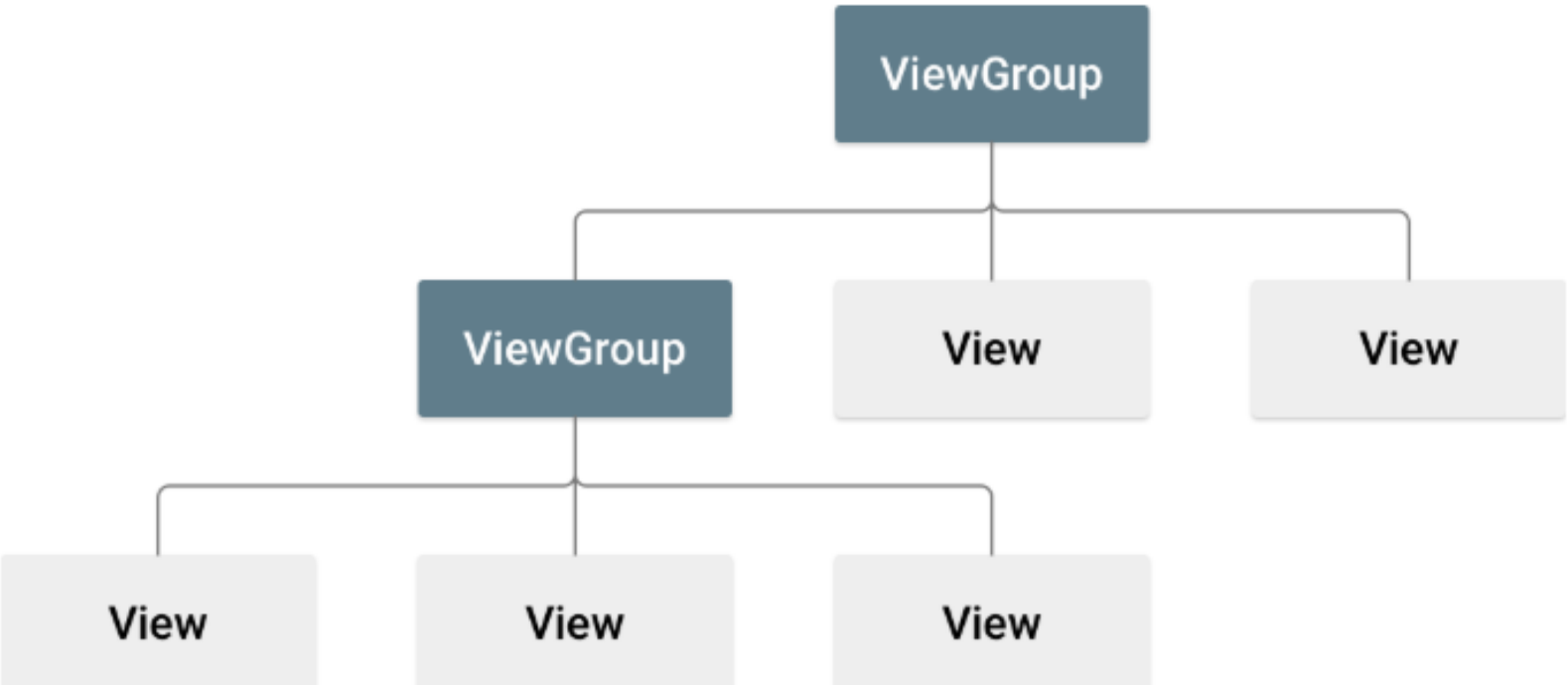
Linear Layout

ut

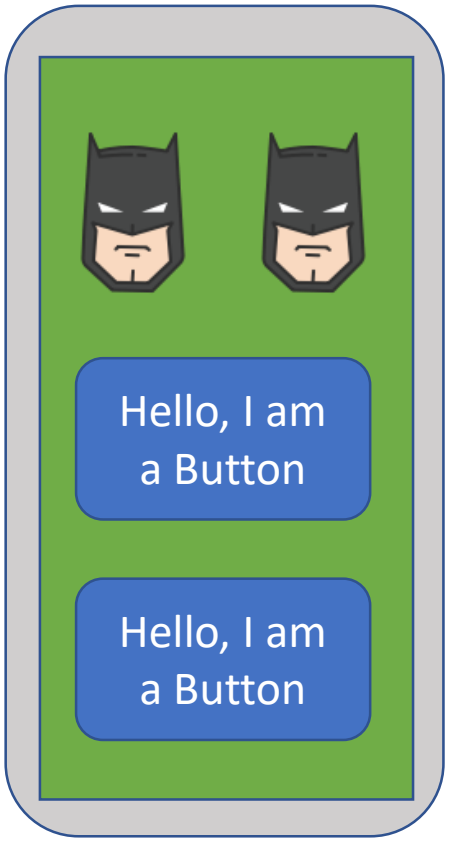
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/andro
id"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">


<ImageView
    android:id="@+id/img1"
    android:src="@drawable/batman"
    android:layout_width="wrap_content"

    android:layout_height="wrap_content" />

   <LinearLayout
       android:orientation="horizontal"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content">


       <ImageView
           android:id="@+id/img2"
           android:src="@drawable/batman"

           android:layout_width="wrap_content"
           android:layout_height="wrap_content" />

       <ImageView
           android:id="@+id/img4"
           android:src="@drawable/batman"

           android:layout_width="wrap_content"
           android:layout_height="wrap_content" />


   </LinearLayout>

    <ImageView
        android:id="@+id/img3"

        android:src="@drawable/batman"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
```



Nested Viewgroups

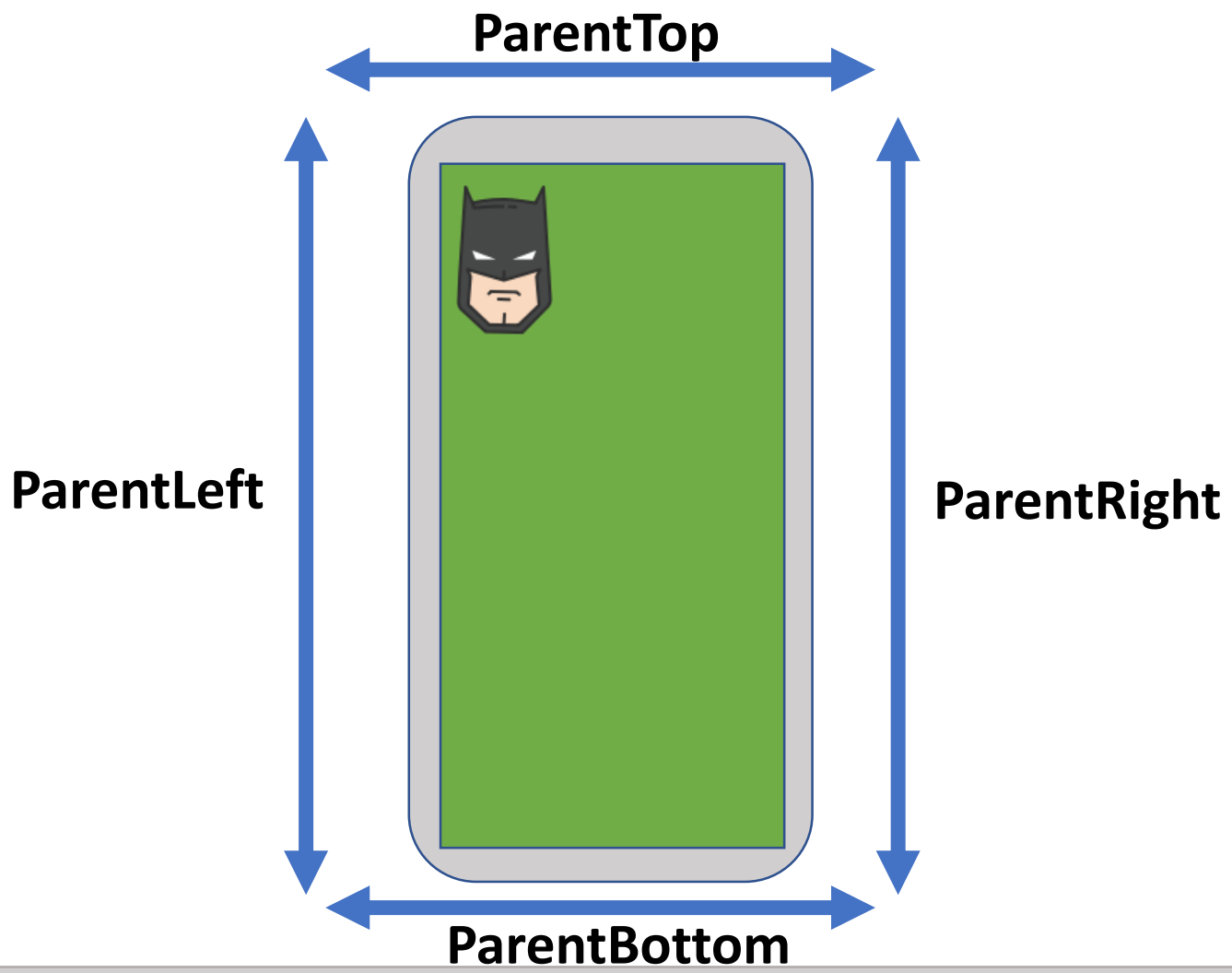


Relative Layout

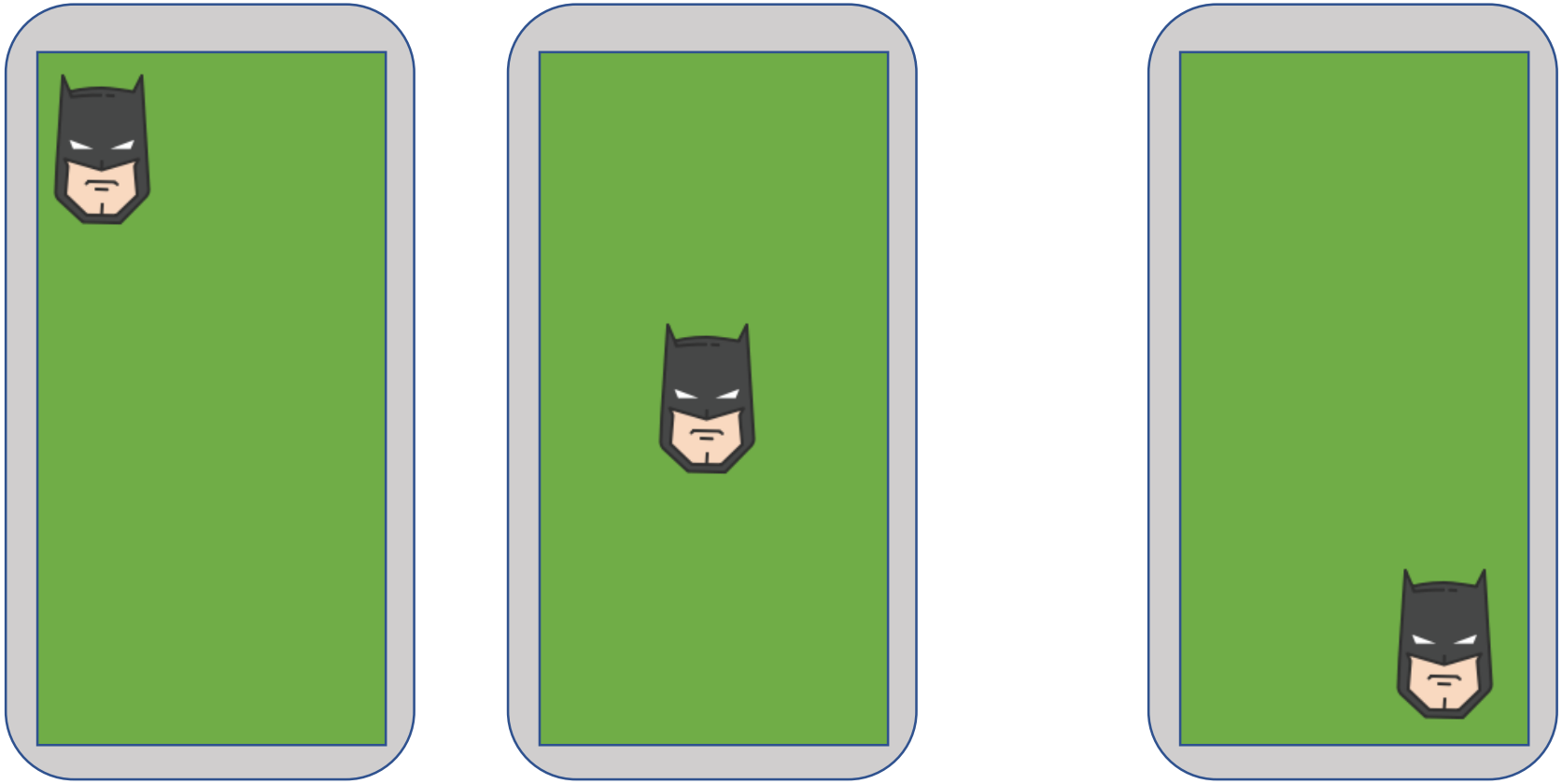


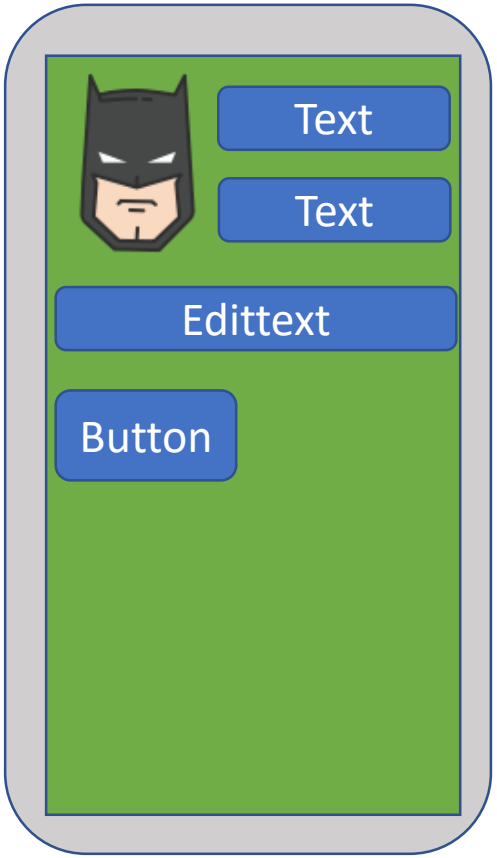
- Higher level of control compared to Linear Layout
- Align views (position child)
  - Relative to parent view
  - Relative to other child views

Align relative to parent



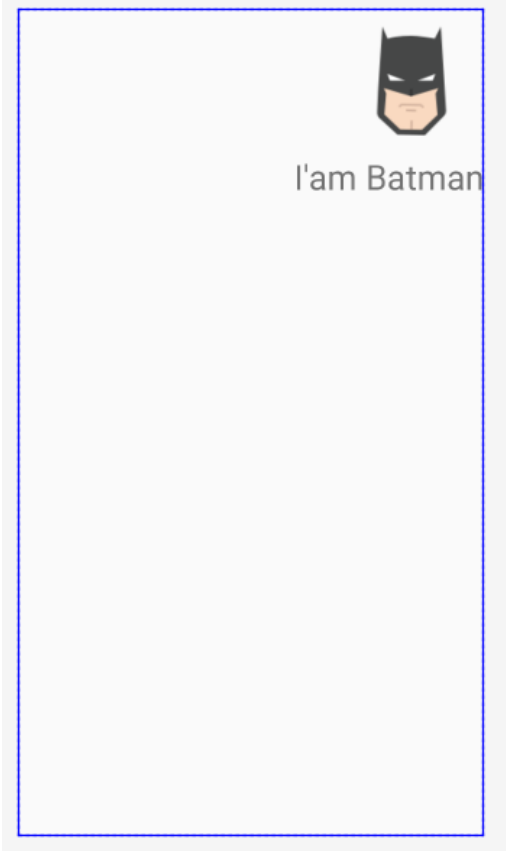
Align relative to parent




Text
Text
Edittext
Button

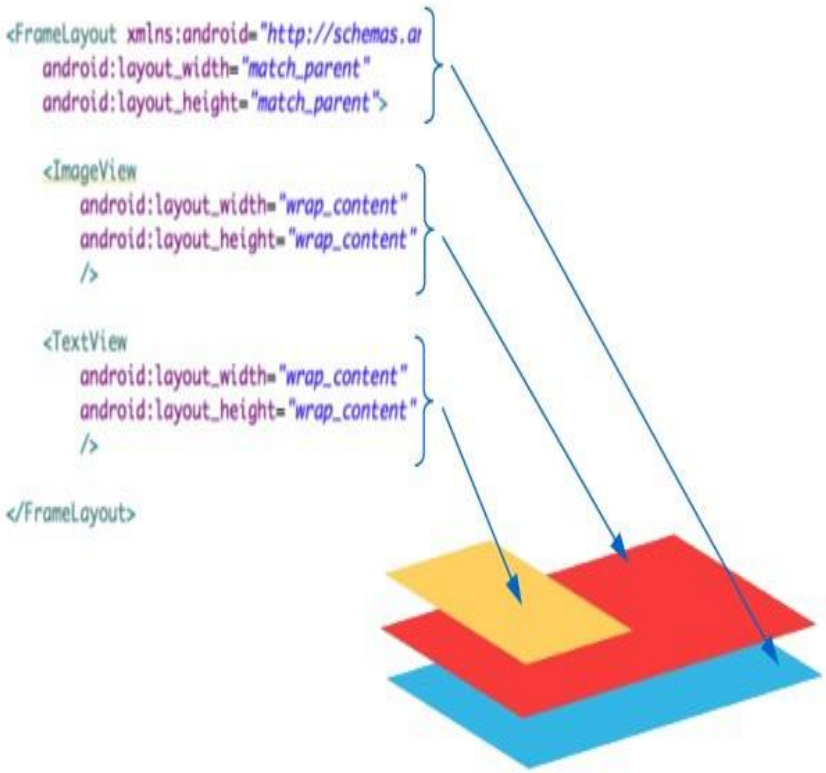
Example Relative Layout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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=".Main3Activity">


    <ImageView
        android:id="@+id/img1"
        android:src="@drawable/batman"
        android:layout_alignParentRight="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/text1"
        android:layout_below="@+id/img1"
        android:layout_alignParentRight="true"
        android:textSize="30sp"
        android:text="I'am Batman"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</RelativeLayout>
```



Frame Layout

- Stack child views on top of each other
- Multiple children can be added and they can be controlled using **layout:gravity**
- The most recent child is added on top



[medium.com/@geekanamika/android-beginners-views-layouts-657a5bbeebe2]

Example Frame Layout

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
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=".Main2Activity">

    <ImageView
        android:id="@+id/img1"
        android:src="@drawable/batman"
        android:layout_gravity="center"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/text1"
        android:text="I'm Batman"
        android:layout_gravity="center"
        android:textSize="30sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</FrameLayout>
```

Exercise 4 - Layouts

Layout 1

Layout 2