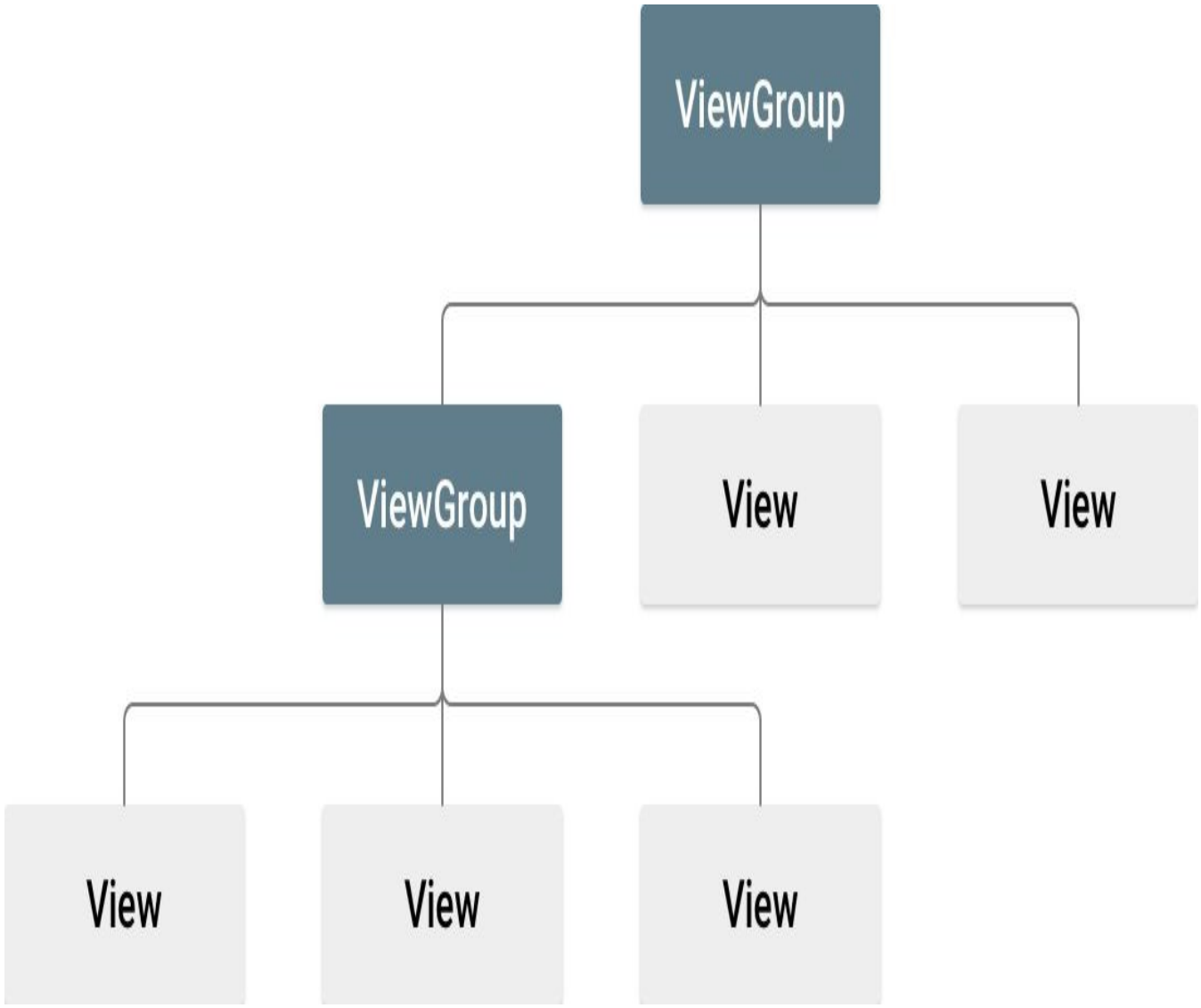
Layout

- A layout defines the structure for a user interface in your app, such as in an activity.
- 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
- The View objects are usually called "widgets" and can be one of many subclasses, such as Button or TextView.
- The ViewGroup objects are usually called "layouts" can be one of many types that provide a different layout structure, such as LinearLayout or ConstraintLayout .


ViewGroup
ViewGroup
View
View
View
View
View

Layout Code : res/layout/

```
<?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_conte
        nt"
        android:layout_height="wrap_conte
        nt"  android:text="Hello, I am a
        TextView" />
  <Button android:id="@+id/button"
       android:layout_width="wrap_conte
       nt"
       android:layout_height="wrap_cont
       ent"  android:text="Hello, I am a
       Button" />
</LinearLayout>
```

Different Types of Layout

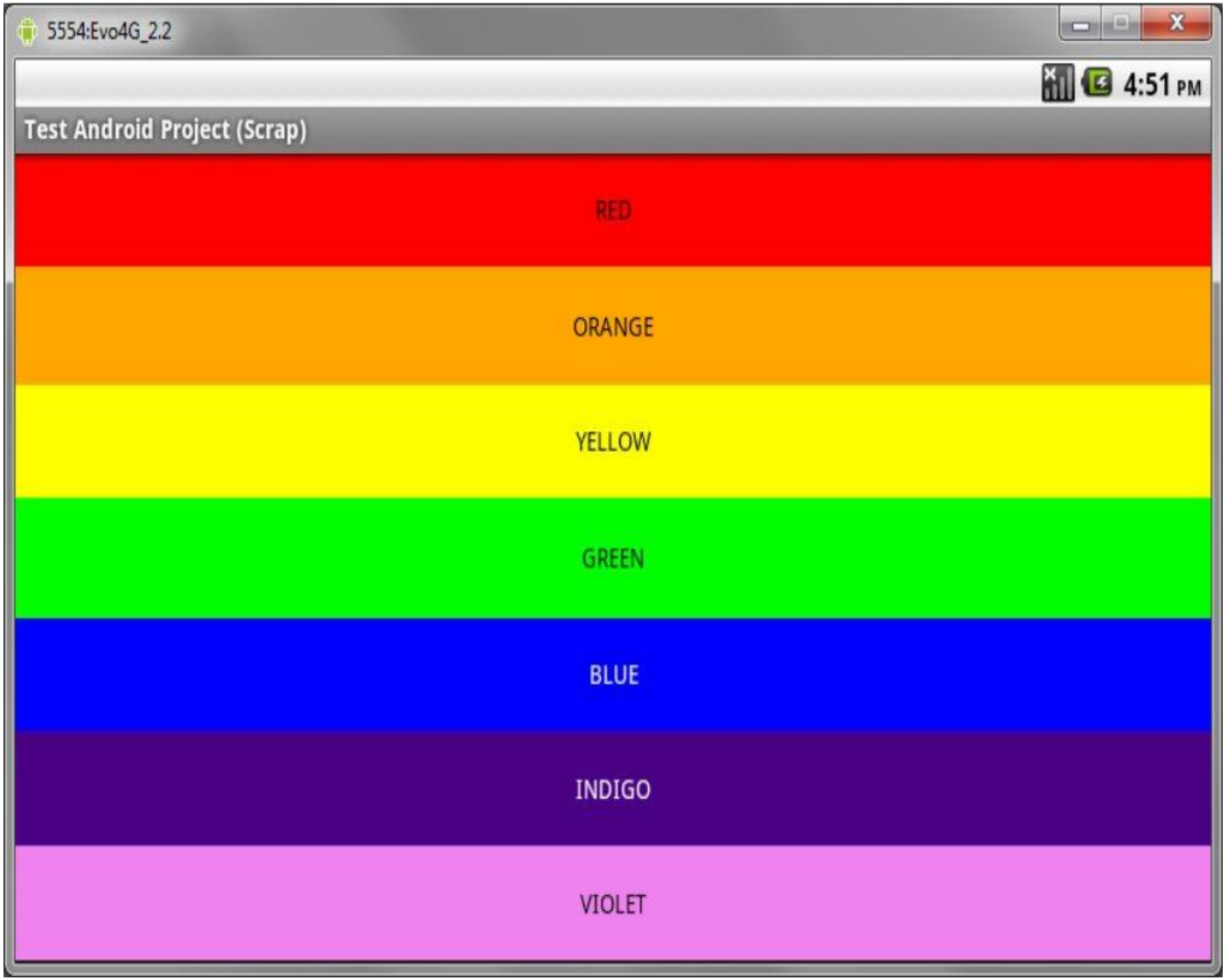
There are six different layouts that can be used in android applications.

1) **LinearLayout**
2) **TableLayout**
3) **TableRow**
4) **RelativeLayout**
5) **GridLayout**
6) **FrameLayout**

Linear Layout

- Linear layouts are one of the simplest and most common types of layouts used by Android developers to organize controls within their user interfaces.
- The linear layout works much as its name implies: it organizes controls linearly in either a vertical or horizontal fashion.
- When the layout's orientation is set to vertical, all child controls within it are organized in a single column; when the layout's orientation is set to horizontal, all child controls within it are organized in a single row.

Linear Layout-Example



```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="https://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="fill_parent"
    android:orientation="vertical">
    <TextView android:text="RED" android:id="@+id/TextView01"
        android:layout_height="wrap_content" android:background="#f00"
        android:layout_width="fill_parent" android:layout_weight=".14"
        android:gravity="center" android:textColor="#000"></TextView>
    <TextView android:text="ORANGE" android:id="@+id/TextView02"
        android:layout_height="wrap_content" android:layout_width="fill_parent"
        android:layout_weight=".15" android:background="#ffa500"
        android:gravity="center" android:textColor="#000"></TextView>
    <TextView android:text="YELLOW" android:id="@+id/TextView03"
        android:layout_height="wrap_content" android:layout_width="fill_parent"
        android:layout_weight=".14" android:background="#ffff00"
        android:gravity="center" android:textColor="#000"></TextView>
    <TextView android:text="GREEN" android:id="@+id/TextView04"
        android:layout_height="wrap_content" android:layout_width="fill_parent"
        android:layout_weight=".15" android:background="#0f0" android:gravity="center"
        android:textColor="#000"></TextView>
    <TextView android:text="BLUE" android:id="@+id/TextView05"
        android:layout_height="wrap_content" android:layout_width="fill_parent"
        android:layout_weight=".14" android:background="#00f" android:gravity="center"
        android:textColor="#fff"></TextView>
    <TextView android:text="INDIGO" android:id="@+id/TextView06"
        android:layout_height="wrap_content" android:layout_width="fill_parent"
        android:layout_weight=".14" android:background="#4b0082"
        android:gravity="center" android:textColor="#fff"></TextView>
    <TextView android:text="VIOLET" android:id="@+id/TextView07"
        android:layout_height="wrap_content" android:layout_width="fill_parent"
        android:layout_weight=".14" android:background="#ee82ee"
        android:gravity="center" android:textColor="#000"></TextView>
</LinearLayout>
```

Recall that, from within the Activity, only a single line of code within the onCreate() method is necessary to load and display a layout resource on the screen.
If the layout resource was stored in the /res/layout/rainbow.xml file, that line of code would be:

Inside Activity:

setContentView(R.layout.rainbow);

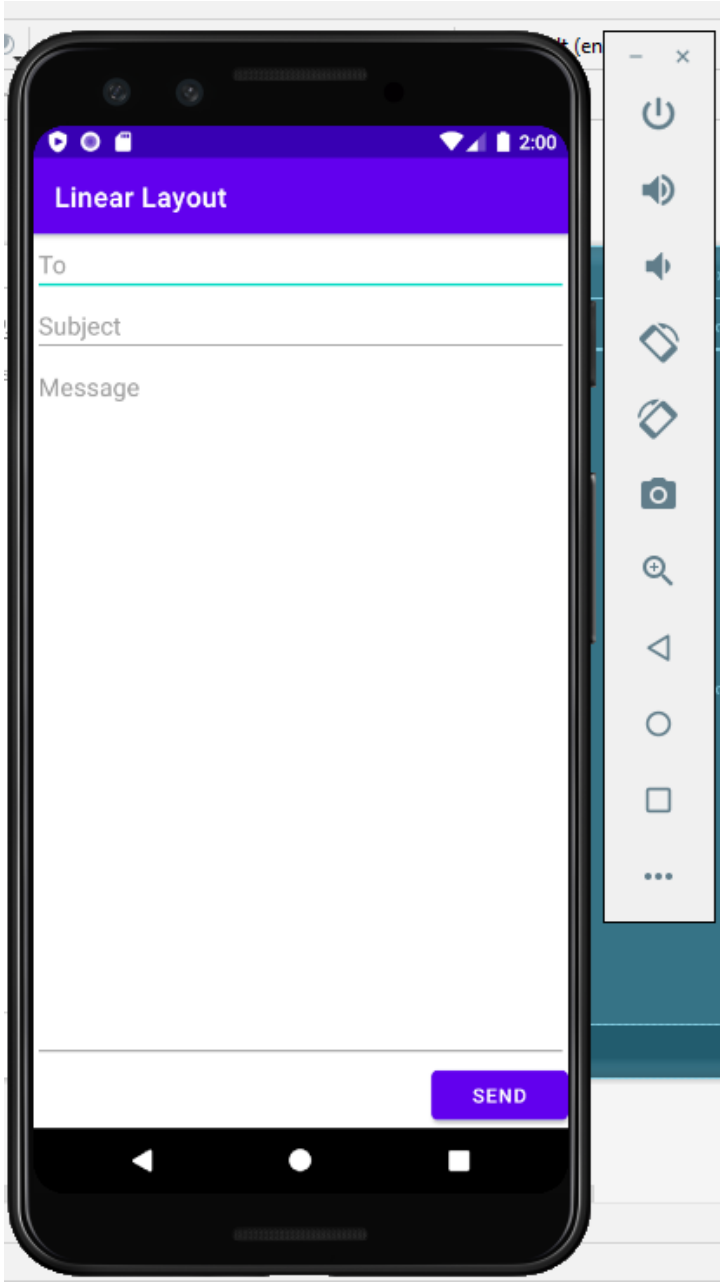
Important Linear Layout Properties and Attributes

Some specific attributes apply to linear layouts. Some of the most important attributes you'll use with linear layouts include:

- The **orientation** attribute (required), which can be set to vertical or horizontal (class: LinearLayout)
- The **gravity** attribute (optional), which controls how all child controls are aligned and displayed within the linear layout (class: LinearLayout)
- The **layout_weight** attribute (optional, applied to each child control) specifies each child control's relative importance within the parent linear layout (class: LinearLayout.LayoutParams)
- This attribute assigns an "importance" value to a view in terms of how much space it should occupy on the screen.

- **Equal distribution**
- To create a linear layout in which each child uses the same amount of space on the screen, set the android:layout_height of each view to "0dp" (for a vertical layout) or the android:layout_width of each view to "0dp" (for a horizontal layout). Then set the android:layout_weight of each view to "1".
- **Unequal distribution**
- You can also create linear layouts where the child elements use different amounts of space on the screen: If there are three text fields and two of them declare a weight of 1, while the other is given no weight, the third text field without weight doesn't grow. Instead, this third text field occupies only the area required by its content.
- The other two text fields, on the other hand, expand equally to fill the space remaining after all three fields are measured.
- If there are three text fields and two of them declare a weight of 1, while the third field is then given a weight of 2 (instead of 0), then it's now declared more important than both the others, so it gets half the total remaining space, while the first two share the rest equally.

Design the following User Interface using Linear Layout



Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
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"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity">
<EditText
    android:inputType="text"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/to"
    android:paddingLeft="10dp"
    />
```

```
<EditText
        android:inputType="text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Subject"
        android:paddingLeft="10dp"/>
 <EditText
        android:inputType="textMultiLine"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Message"
        android:layout_weight="1"
        android:paddingLeft="10dp"
        android:gravity="top"/>
  <Button
   android:layout_width="100dp"
   android:layout_height="wrap_content"
   android:layout_gravity="right"
   android:backgroundTint="@color/ButtonColor"
   android:text="SEND"
   app:cornerRadius="24dp"
   app:strokeColor="@color/white"
   app:strokeWidth="1dp" />


</LinearLayout>
```

string.xml

```
<resources>
        <string name="app_name">Linear Layout</string>
        <string name="to">To</string>
        <string name="subject">Subject</string>
        <string name="message">Message</string>
        <string name="button">Send</string>
    </resources>
```