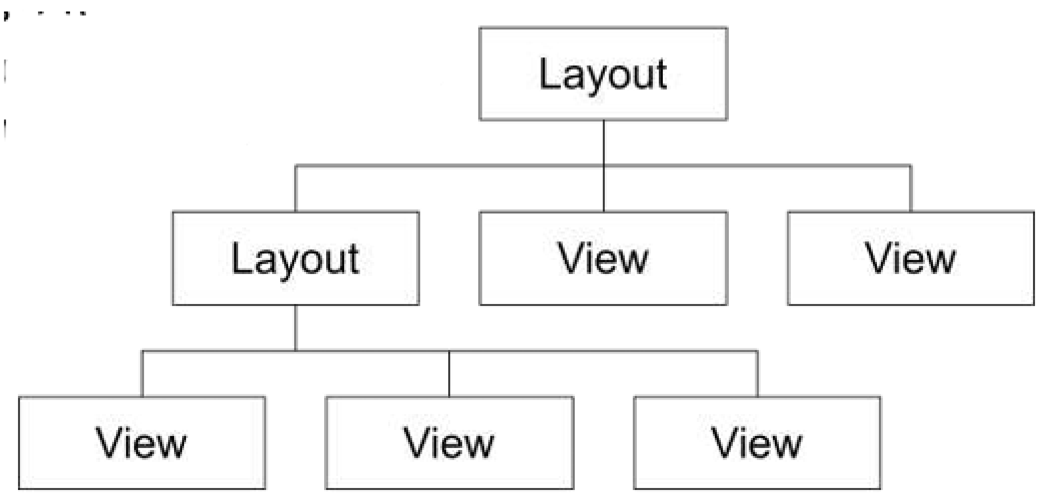
GUI - Basic widgets (1)

The View class



- The View class is the Android's most basic component from which user interfaces can be created. It acts as a container of displayable elements
- A View occupies a rectangular area on the screen and is responsible for drawing and event handling
- Widgets are subclasses of View. They are used to create interactive UI components such as buttons, checkboxes, labels, text fields, etc.
- Layouts are invisible structured containers used for holding other Views and nested layouts.

Nesting XML Layouts

- An Android's XML view file consists of a layout design holding a hierarchical arrangement of its contained elements
- The inner elements could be basic widgets or user-defined nested layouts holding their own viewgroups
- An Activity uses the `setContentView(R.layout.xmlfilename)` method to render a view on the device's screen

Setting Views to Work

Dealing with widgets & layouts typically involves the following operations

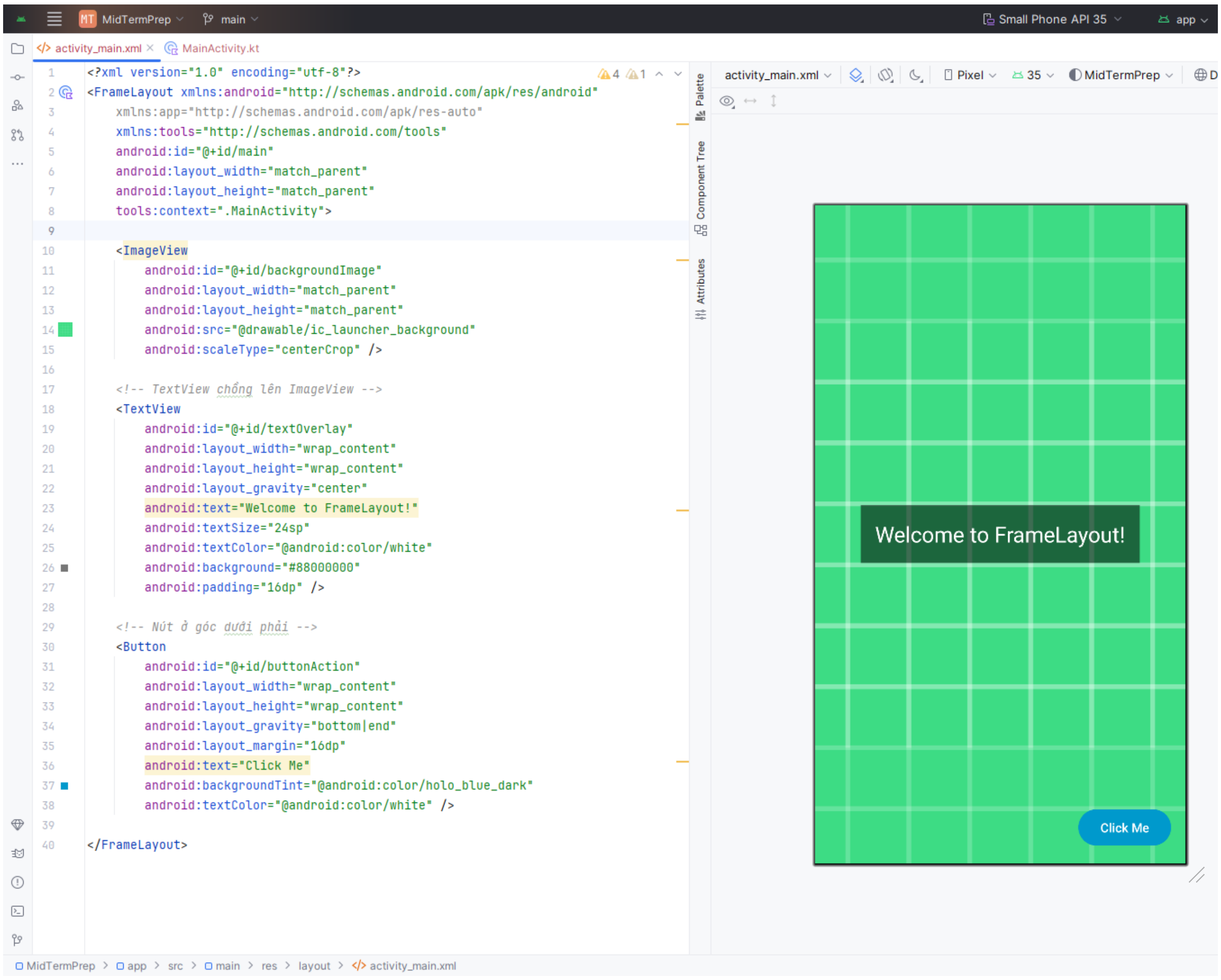
- Set properties: For instance, when working with a TextView you set the background color, text, font, alignment, size, padding, marging, etc.
- Set up listeners: For example, an image could be programmed to respond to various events such as: click, long-tap, mouseover, etc.

The Layout

FrameLayout

- The FrameLayout is the simplest type of GUI container.
- It is useful as an outermost container holding a window.

- Allows you to define how much of the screen (high, width) is to be used.
- All its children elements are aligned to the top left corner of the screen



```xml
<?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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/backgroundImage"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:src="@drawable/ic_launcher_background"
        android:scaleType="centerCrop" />

    <!-- TextView chồng lên ImageView -->
    <TextView
        android:id="@+id/textOverlay"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Welcome to FrameLayout!"
        android:textSize="24sp"
        android:textColor="@android:color/white"
        android:background="#88000000"
        android:padding="16dp" />

    <!-- Nút ở góc dưới phải -->
    <Button
        android:id="@+id/buttonAction"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="16dp"
        android:text="Click Me"
        android:backgroundTint="@android:color/holo_blue_dark"
        android:textColor="@android:color/white" />

</FrameLayout>
```

LinearLayout

- The LinearLayout supports a filling strategy in which new elements are stacked either in a **horizontal** or **vertical** fashion
- If the layout has a vertical orientation new rows are placed one on top of the other
- A horizontal layout uses a side-by-side column placement policy

Attributes

Attribute	Value
orientation	vertical, horizontal
fill model	match_parent, wrap_contents
weight	0, 1, 2, ..., n
gravity	top, bottom, center
padding	dp - dev. independent pixels

Attribute	Value
margin	dp - dev. independent pixels

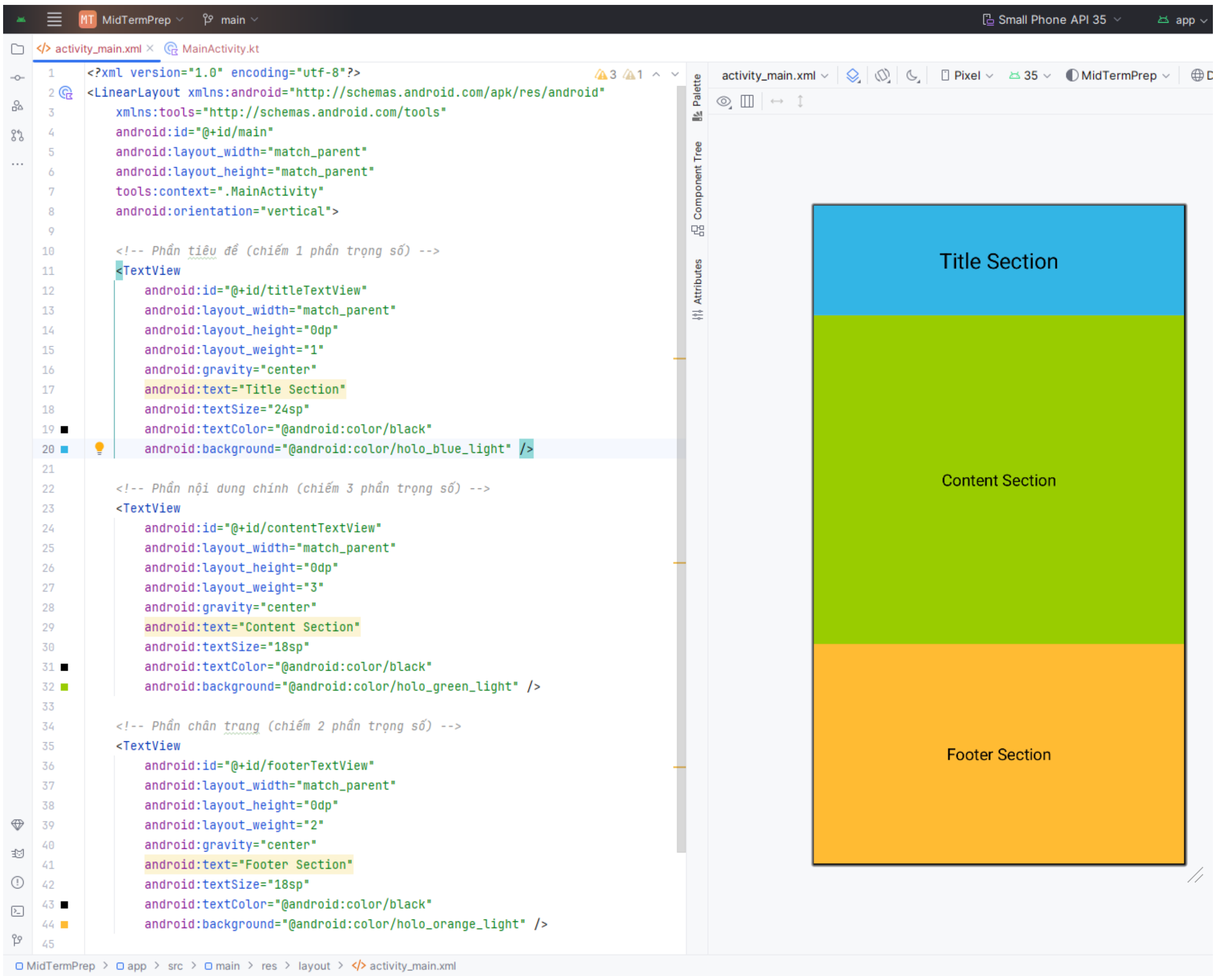
LinearLayout: Fill model

All widgets inside a LinearLayout **must** include `android:layout_width` and `android:layout_height` attributes. Values can be:

- A specific dimension such as 125dp (devide independent pixels dip)
- `wrap_content` indicates the widget should just fill up its natural space
- `match_parent` (previous version: `fill_parent`) indicates the widget wants to be as big as the enclosing parent

LinearLayout: Weight

The extra space left unclaimed in a layout could be assigned to any of its inner components by setting its Weight attribute. Use 0 if the view should not be stretched. The bigger the weight the larger the extra space given to that widget.



```xml
<?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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical">

    <!-- Phần tiêu đề (chiếm 1 phần trọng số) -->
    <TextView
        android:id="@+id/titleTextView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="center"
        android:text="Title Section"
        android:textSize="24sp"
        android:textColor="@android:color/black"
        android:background="@android:color/holo_blue_light" />

    <!-- Phần nội dung chính (chiếm 3 phần trọng số) -->
    <TextView
        android:id="@+id/contentTextView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="3"
        android:gravity="center"
        android:text="Content Section"
        android:textSize="18sp"
        android:textColor="@android:color/black"
        android:background="@android:color/holo_green_light" />

    <!-- Phần chân trang (chiếm 2 phần trọng số) -->
    <TextView
        android:id="@+id/footerTextView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="2"
        android:gravity="center"
        android:text="Footer Section"
        android:textSize="18sp"
        android:textColor="@android:color/black"
        android:background="@android:color/holo_orange_light" />

```

LinearLayout: Gravity

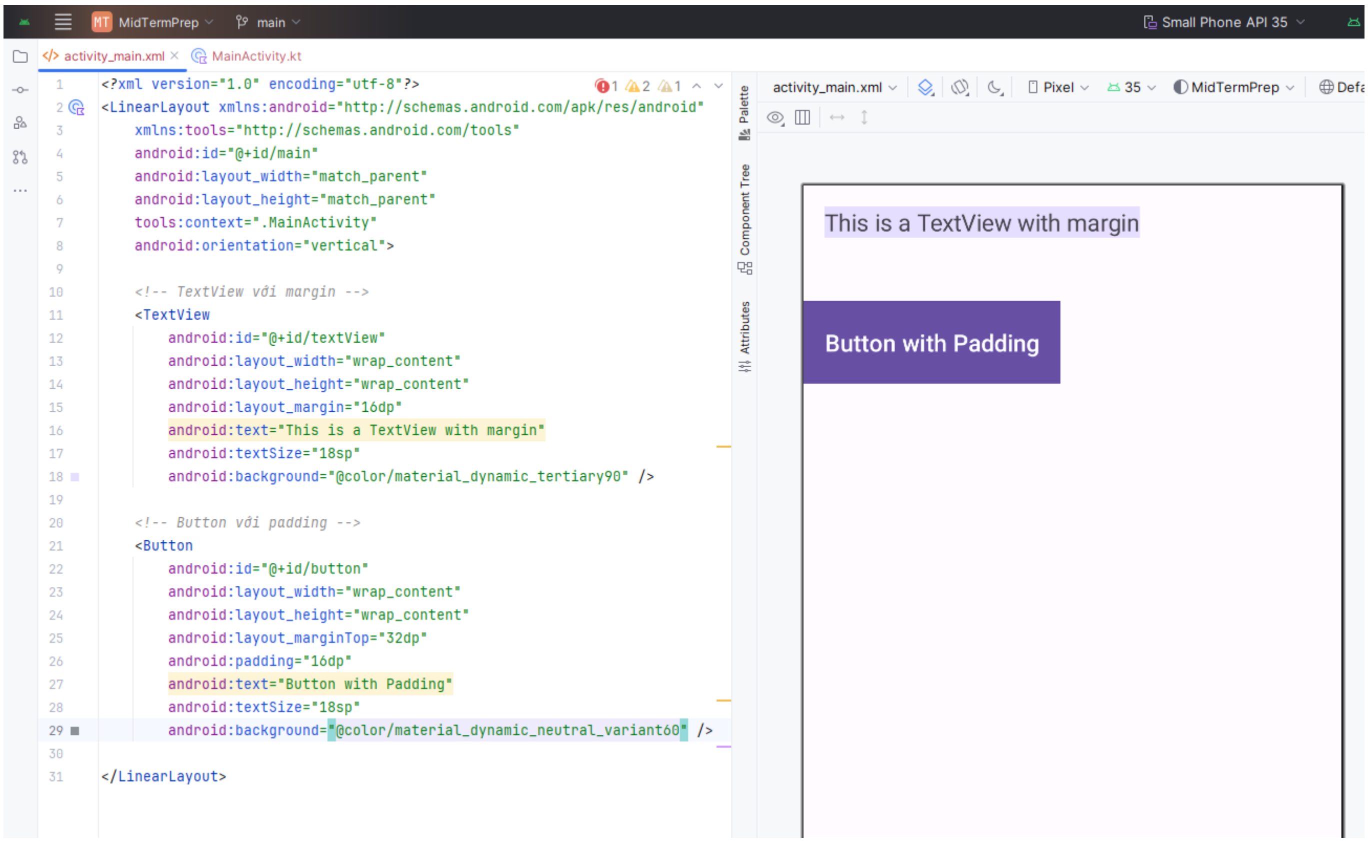
Gravity is used to indicate how a control will align on the screen

> Default: left and top align

Other possible arrangements: `top`, `bottom`, `left`, `right`, `center_vertical`, `fill_vertical`, `center_horizontal`, `fill_horizontal`, `center`, `fill`, `clip_vertical` (Nội dung bị cắt hoặc giới hạn theo chiều dọc nếu nó vượt quá vùng chứa), `clip_horizontal`, `start` (Căn nội dung vào mép bắt đầu theo hướng ngôn ngữ, Tiếng Anh: Bên trái, Tiếng Ả Rập: Bên phải), `end`

Multiple values: `android:layout_gravity="end|bottom"`

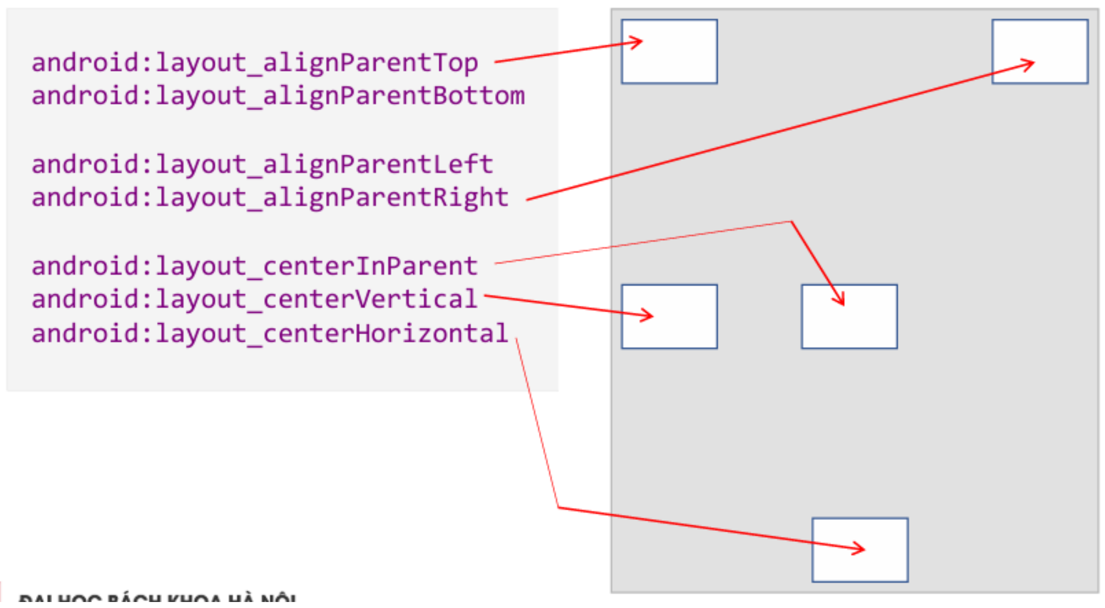
LinearLayout: Margin and Padding



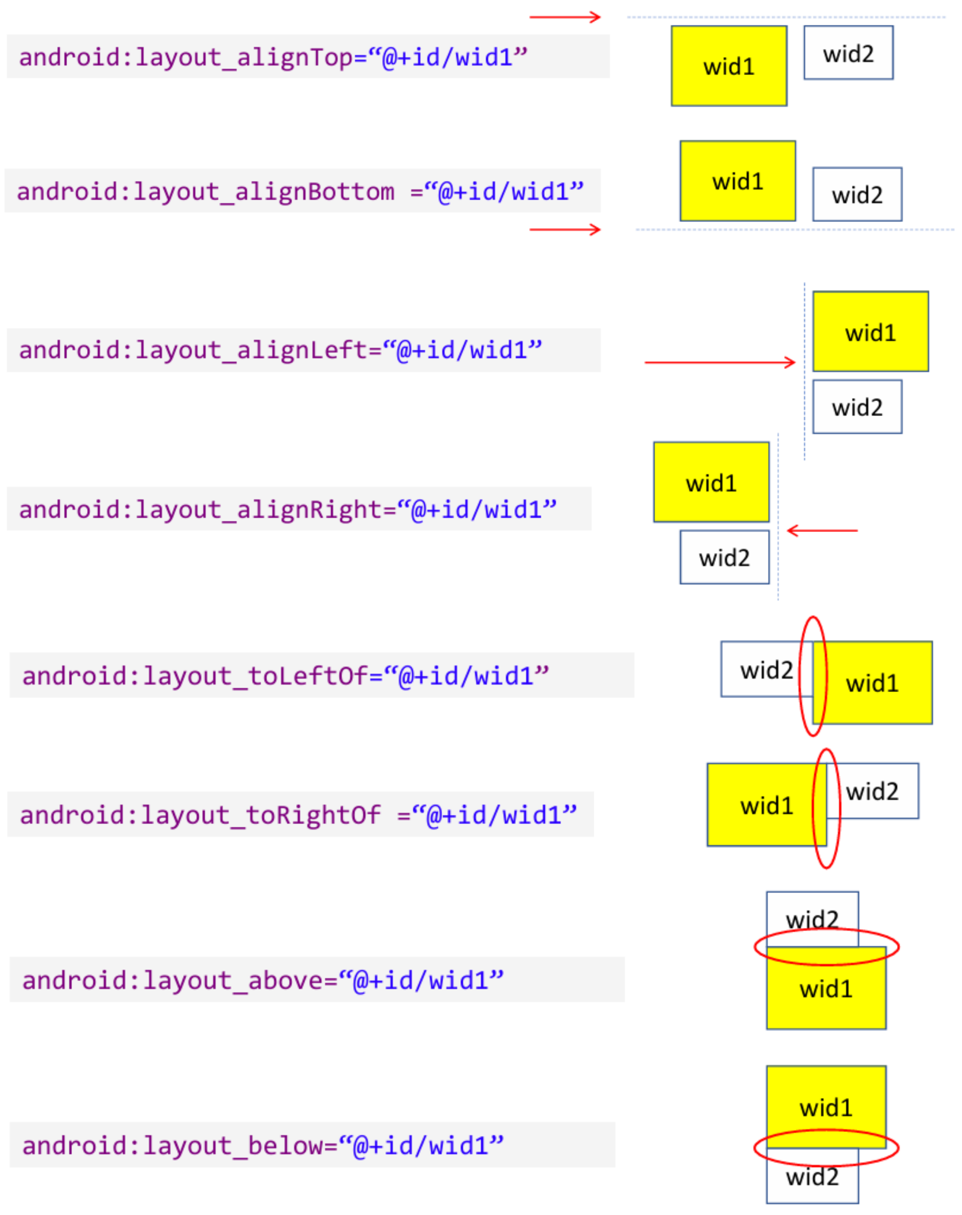
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"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

    <!-- TextView với margin -->
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:text="This is a TextView with margin"
        android:textSize="18sp"
        android:background="@color/material_dynamic_tertiary90" />

    <!-- Button với padding -->
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:padding="16dp"
        android:text="Button with Padding"
        android:textSize="18sp"
        android:background="@color/material_dynamic_neutral_variant60" />

</LinearLayout>
```

Relative Layout

The placement of a widget in a RelativeLayout is based on its positional relationship to other widgets in the container as well as the parent container



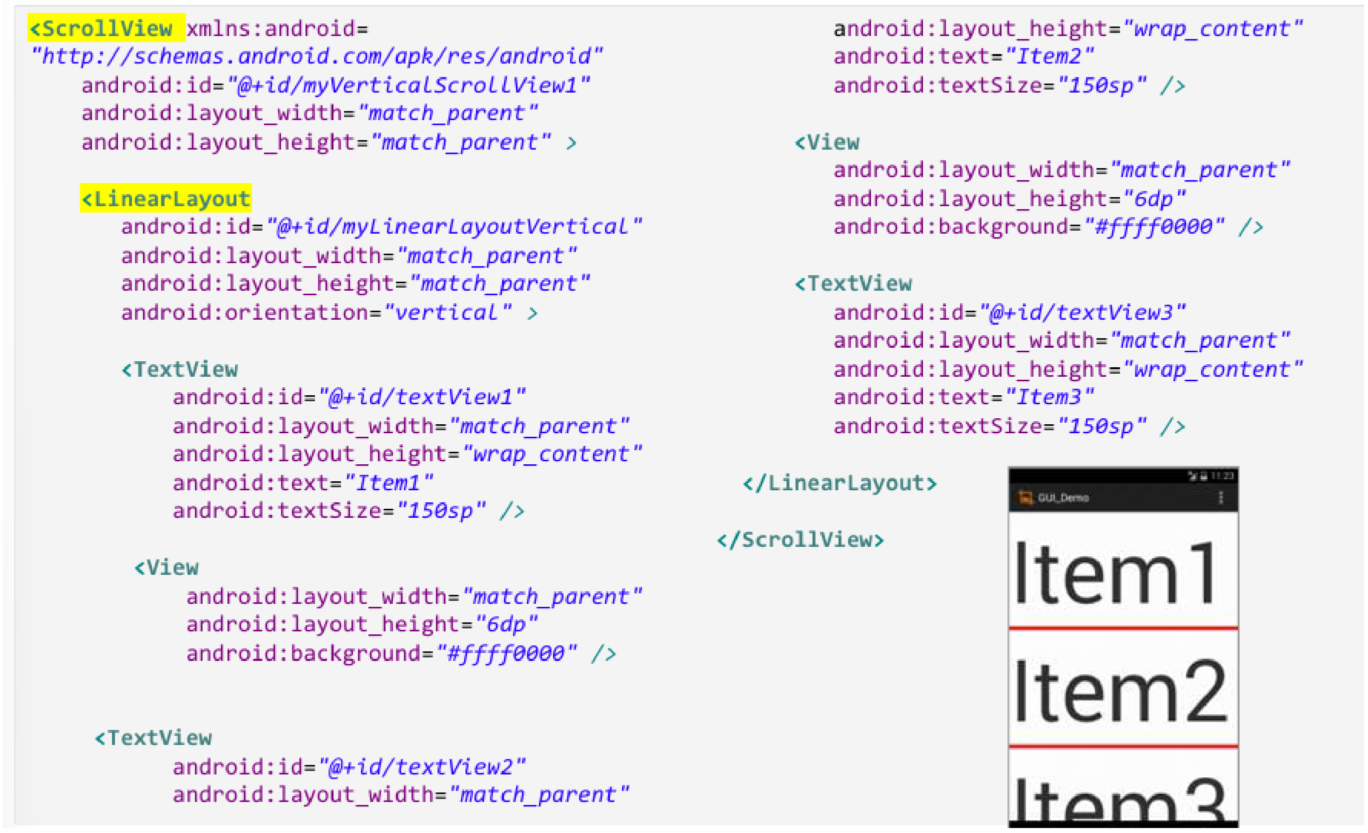
```
android:layout_alignParentTop
android:layout_alignParentBottom

android:layout_alignParentLeft
android:layout_alignParentRight

android:layout_centerInParent
android:layout_centerVertical
android:layout_centerHorizontal
```



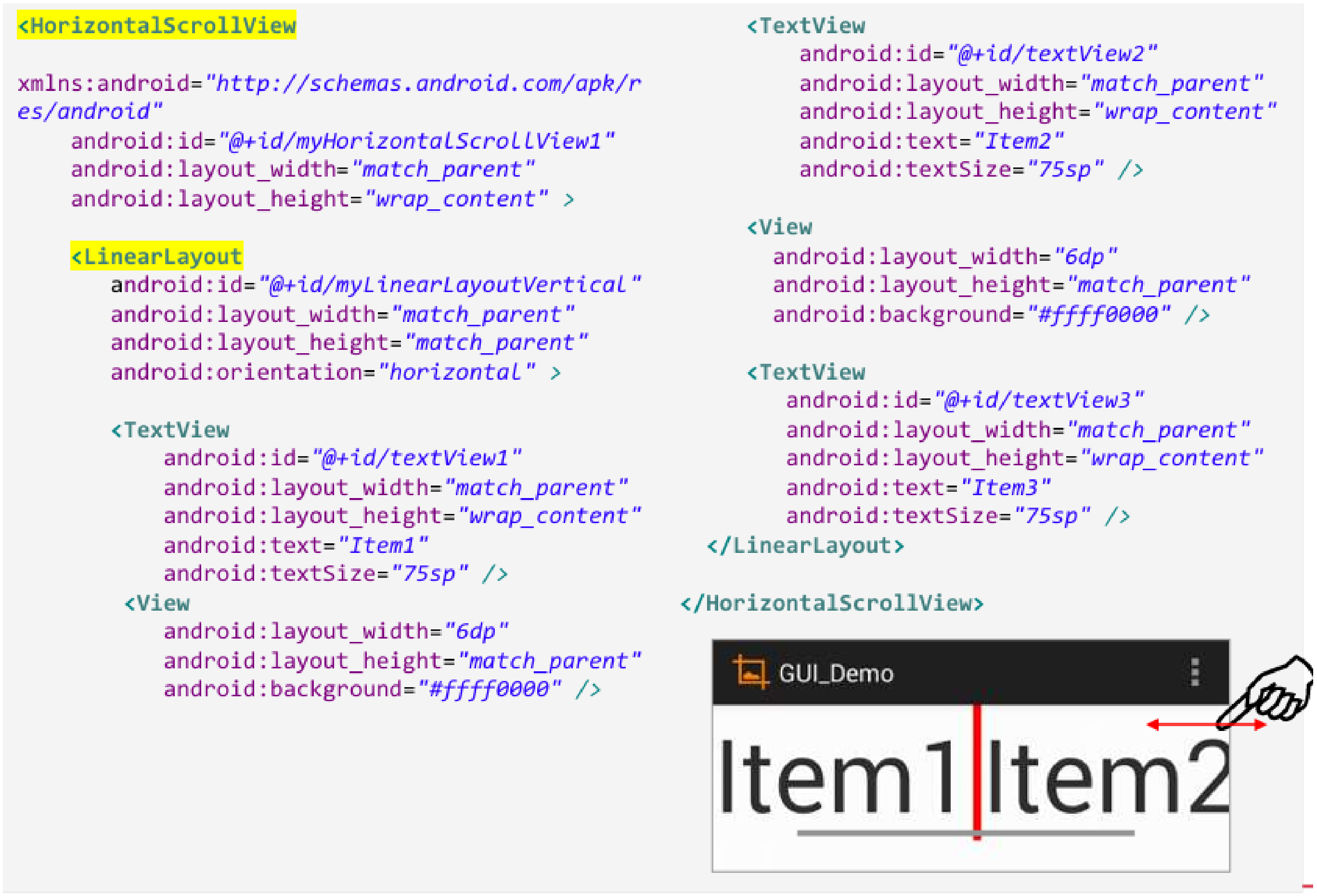
```
android:layout_alignTop="@+id/wid1"
android:layout_alignBottom ="@+id/wid1"
android:layout_alignLeft="@+id/wid1"
android:layout_alignRight="@+id/wid1"
android:layout_toLeftOf="@+id/wid1"
android:layout_toRightOf ="@+id/wid1"
android:layout_above="@+id/wid1"
android:layout_below="@+id/wid1"
```

ScrollView Layout

- ScrollViews provide a vertical sliding (up/down) access to the data
- The HorizontalScrollView provides a similar left/right sliding mechanism



```xml
<ScrollView xmlns:android=
"http://schemas.android.com/apk/res/android"
    android:id="@+id/myVerticalScrollView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <LinearLayout
        android:id="@+id/myLinearLayoutVertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/textView1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Item1"
            android:textSize="150sp" />

         <View
             android:layout_width="match_parent"
             android:layout_height="6dp"
             android:background="#ffff0000" />


      <TextView
            android:id="@+id/textView2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Item2"
            android:textSize="150sp" />

        <View
            android:layout_width="match_parent"
            android:layout_height="6dp"
            android:background="#ffff0000" />

        <TextView
            android:id="@+id/textView3"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Item3"
            android:textSize="150sp" />

    </LinearLayout>

  </ScrollView>
```



```xml
<HorizontalScrollView

xmlns:android="http://schemas.android.com/apk/r
es/android"
    android:id="@+id/myHorizontalScrollView1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <LinearLayout
        android:id="@+id/myLinearLayoutVertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal" >

        <TextView
            android:id="@+id/textView1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Item1"
            android:textSize="75sp" />
         <View
            android:layout_width="6dp"
            android:layout_height="match_parent"
            android:background="#ffff0000" />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Item2"
            android:textSize="75sp" />

        <View
          android:layout_width="6dp"
          android:layout_height="match_parent"
          android:background="#ffff0000" />

        <TextView
           android:id="@+id/textView3"
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:text="Item3"
           android:textSize="75sp" />
    </LinearLayout>

  </HorizontalScrollView>
```

Constraint Layout

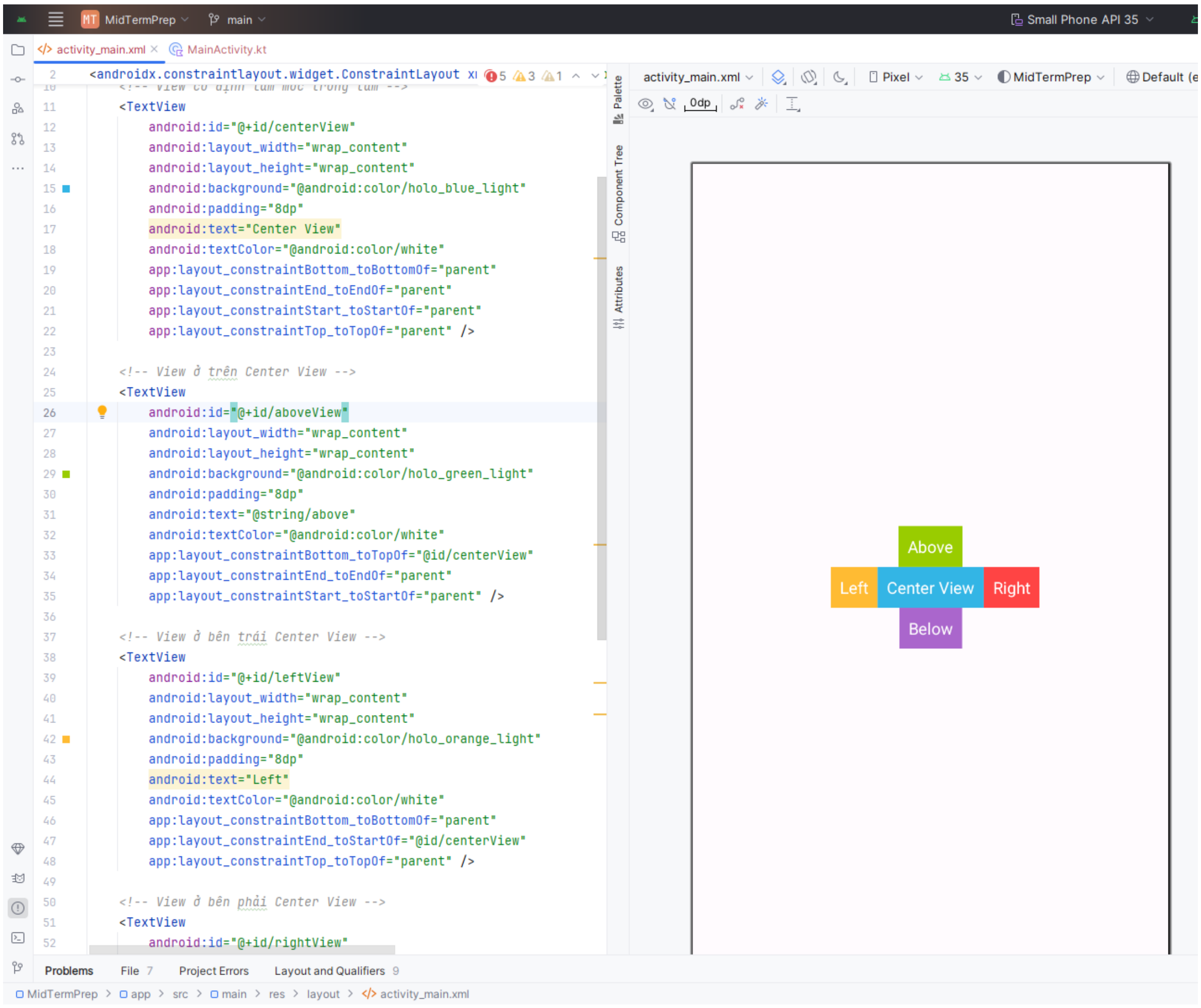
Deeply nested layouts are costly

- Deeply nested ViewGroups require more computation

- Views may be measured multiple times
- Can cause UI slowdown and lack of responsiveness Use ConstraintLayout to avoid some of these issues

ConstraintLayout:

- Recommended default layout for Android
- Solves costly issue of too many nested layouts, while allowing complex behavior
- Position and size views within it using a set of constraints



```xml
<androidx.constraintlayout.widget.ConstraintLayout xı
    <!-- View cố định làm mốc trong tâm -->
    <TextView
        android:id="@+id/centerView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@android:color/holo_blue_light"
        android:padding="8dp"
        android:text="Center View"
        android:textColor="@android:color/white"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <!-- View ở trên Center View -->
    <TextView
        android:id="@+id/aboveView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@android:color/holo_green_light"
        android:padding="8dp"
        android:text="@string/above"
        android:textColor="@android:color/white"
        app:layout_constraintBottom_toTopOf="@id/centerView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <!-- View ở bên trái Center View -->
    <TextView
        android:id="@+id/leftView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@android:color/holo_orange_light"
        android:padding="8dp"
        android:text="Left"
        android:textColor="@android:color/white"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@id/centerView"
        app:layout_constraintTop_toTopOf="parent" />

    <!-- View ở bên phải Center View -->
    <TextView
        android:id="@+id/rightView"
```