These **layout containers** help you structure the user interface of your Android app. Here's an overview of how each one works and how to use them in **Java**:

**1. LinearLayout**

**LinearLayout** arranges child elements either in a row (horizontal) or in a column (vertical).

* **XML (LinearLayout Example)**:

<LinearLayout

android:id="@+id/linearLayout"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="First item" />

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Click Me" />

</LinearLayout>

**2. RelativeLayout**

**RelativeLayout** positions elements relative to each other. You can position elements by specifying attributes like android:layout\_toEndOf, android:layout\_below, etc.

* **XML (RelativeLayout Example)**:

<RelativeLayout

android:id="@+id/relativeLayout"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

<TextView

android:id="@+id/textView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Above button"

android:layout\_centerHorizontal="true" />

<Button

android:id="@+id/button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Click Me"

android:layout\_below="@id/textView"

android:layout\_centerHorizontal="true" />

</RelativeLayout>

**3. ConstraintLayout**

**ConstraintLayout** is the most flexible layout container, allowing you to define complex layouts with flexible positioning and relationships between elements. It's recommended for modern UI design.

* **XML (ConstraintLayout Example)**:

<androidx.constraintlayout.widget.ConstraintLayout

android:id="@+id/constraintLayout"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<TextView

android:id="@+id/textView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hello, world!"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintStart\_toStartOf="parent" />

<Button

android:id="@+id/button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Click Me"

app:layout\_constraintTop\_toBottomOf="@id/textView"

app:layout\_constraintStart\_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

**4. FrameLayout**

**FrameLayout** is used to stack child elements on top of each other. This layout is commonly used for displaying one element at a time (like showing a fragment or image).

* **XML (FrameLayout Example)**:

<FrameLayout

android:id="@+id/frameLayout"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<TextView

android:id="@+id/textView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="This is a text" />

<Button

android:id="@+id/button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Click Me"

android:layout\_gravity="center" />

</FrameLayout>