## User Interface Basics

Views and ViewGroups are the fundamental building blocks for creating user interfaces (UIs) in Android applications.

### Views:

* Views represent the basic UI elements that users can interact with or see on the screen. These elements can be simple (like a text view) or complex (like a button or an image).
* Each view is a subclass of the View class in the Android framework.
* Common examples of views include:
  + TextView: Displays text content.
  + Button: A clickable button that triggers an action.
  + EditText: Allows users to enter text.
  + ImageView: Displays images.
  + LinearLayout: Arranges views in a linear fashion (horizontal or vertical).

**Properties of Views:**

* Views have various properties that define their appearance and behavior, such as:
  + Layout (position and size on the screen)
  + Background color
  + Text content (for text views)
  + Image source (for image views)
  + Clickable or not (responds to user taps)

### ViewGroups:

* ViewGroups are special views that do not represent UI elements themselves, but instead act as containers for other views and ViewGroups.
* They are responsible for arranging and managing the layout of their child views (the views they contain).
* ViewGroups are subclasses of the ViewGroup class in the Android framework.
* Common examples of ViewGroups include:
  + LinearLayout: Arranges views in a linear fashion (horizontal or vertical).
  + RelativeLayout: Positions views relative to each other or the parent view.
  + FrameLayout: Stacks views on top of each other.
  + ConstraintLayout: Offers flexible layout options using constraints.

### Relationship Between Views and ViewGroups:

* An app's UI hierarchy is a tree structure.
* A ViewGroup can contain multiple child views (other Views or ViewGroups).
* These child views can also be ViewGroups themselves, further containing other views or ViewGroups.
* The ViewGroup acts as the parent, and the child views are its descendants.
* The root of the UI hierarchy is typically a ViewGroup (e.g., ConstraintLayout) that holds the entire layout for the activity.

### Layout Examples – View Groups

**Linear Layout (Horizontal):** This layout arranges views in a single row, from left to right.

A screen shot of a computer code

Description automatically generated

* android:orientation="horizontal": Defines the layout as horizontal.
* android:layout\_weight: Used to distribute remaining space proportionally between views.

**Linear Layout (Vertical):** This layout arranges views in a single column, from top to bottom.

A screen shot of a computer code

Description automatically generated

**Relative Layout:** This layout positions views relative to each other or the parent view.

**A screenshot of a computer program

Description automatically generated**

* android:layout\_centerInParent="true": Centers the view within the parent.
* android:layout\_alignBottom="@+id/text\_view": Aligns the button's bottom with the text view's bottom.
* android:layout\_toRightOf="@+id/text\_view": Positions the button to the right of the text view.

**Constraint Layout:** This layout offers more flexible positioning using constraints (horizontal/vertical guidelines, margins to edges).

A screenshot of a computer program

Description automatically generated