Screen Layout Design: Views and Layouts

One of the most important parts of any application’s design and development is the graphical user interface (GUI) and screen layout design. Many of the most widely circulated Android applications are popular because of their visual design, animated graphics, and easy- or fun-to-use interfaces. We will explore the Java classes that provide the core foundation for all of these front-end capabilities in this chapter.

Android View Hierarchies

In Google Android, in order to interface with the smartphone screen, you use two core Java classes. These are two of the most important and often used classes in Android development:

- The `View` class
- The `ViewGroup` class

`View` and `ViewGroup` are core, high-level classes, created or subclassed from the Java `Object` class, as are all Java classes. `View` objects are created using the `View` class. The `View` class can also be used to create many lower-level, or more customized, Java classes. Those classes that are subclassed from the `View` class inherit the characteristics of their superclass.

So, the basic screen layout in Android is controlled by a `View` object, which contains a complex data structure that represents the content and layout parameters for a given rectangular section of the smartphone’s display screen.
Using the View Class

There may be one or more View objects that make up the entire display screen, depending on how you use the View and ViewGroup classes to create the UI structure for your Android application’s screen.

Each View object controls and references its own rectangular view parameters, allowing you to control many attributes. Here are just some examples of the many attributes controlled by the View class parameters available to programmers:

- Bounds (measurements)
- Layout on the screen
- Order in which its layers are drawn
- Scrolling
- Focus
- Keystroke interactions
- Gesture interactions

Finally, Views have the ability to receive events—interaction events between your application’s end user and the View object itself. For this reason, the View class is the logical Java construct to subclass to build more detailed and specific UI elements, such as buttons, check boxes, radio buttons, and text fields.

NOTE: The View class serves as the foundation for UI elements that are subclasses of the View class. Recall that in Java, a subclass is a more specific or detailed implementation of the class from which it is subclassed. For instance, the Button class is subclassed from the TextView class, which is subclassed from the View class, which is subclassed from the Object class. The Button class is subclassed from the TextView class because the Button has a TextView label and is thus a more specialized version of a TextView; that is, it is a clickable TextView with a button background appearance.

So many UI classes have been subclassed from the View class that there is a name for them: widgets. All of these widgets are contained in a package (a collection of classes) called android.widget. For example, you can access a Button class via this package using android.widget.button.

Nesting Views: Using the ViewGroup Class

One of the most useful classes subclassed from the View class is the ViewGroup class. The ViewGroup class is used to subclass layout container classes, which allow groups of View objects to be logically grouped, arranged, and cascaded onto the screen.