Android O

Android UI and Layouts

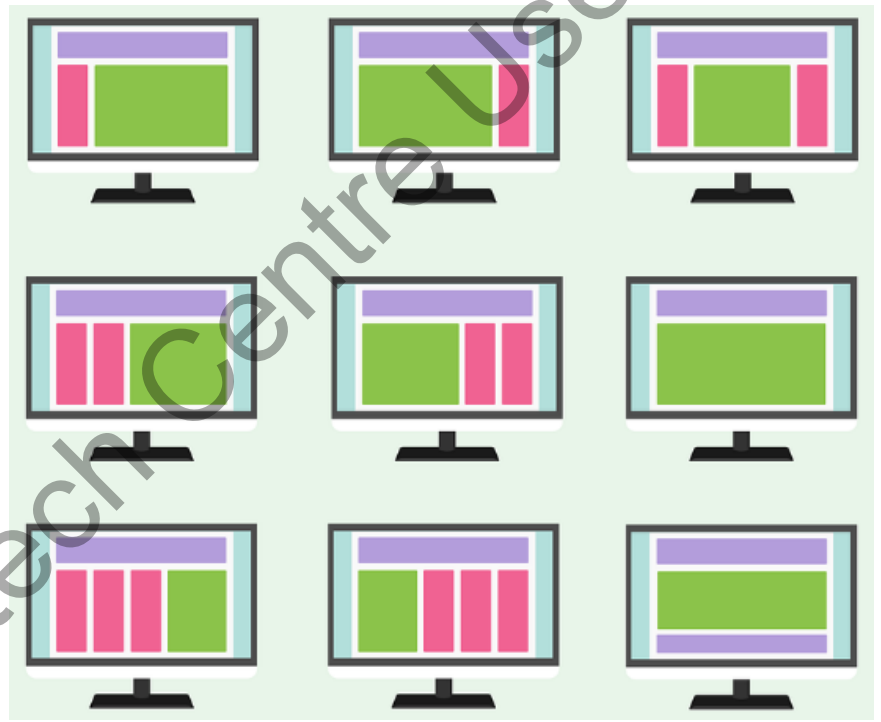
Session 03

Android UI and Layouts

- Objectives
  - Define different types of Android UI Layouts
  - Describe Android UI Layouts
  - Explain the use of different Android UI Layouts
  - Explain how to use UI Controls
  - Explain what are UI Widgets

Android User Interface (UI) Layouts

- Layout is defined as the structure of your app's user interface.
- Android supports interactive user interface elements:
  - Buttons
  - Text boxes
  - Drop-down lists
- Belong to the category of layouts.



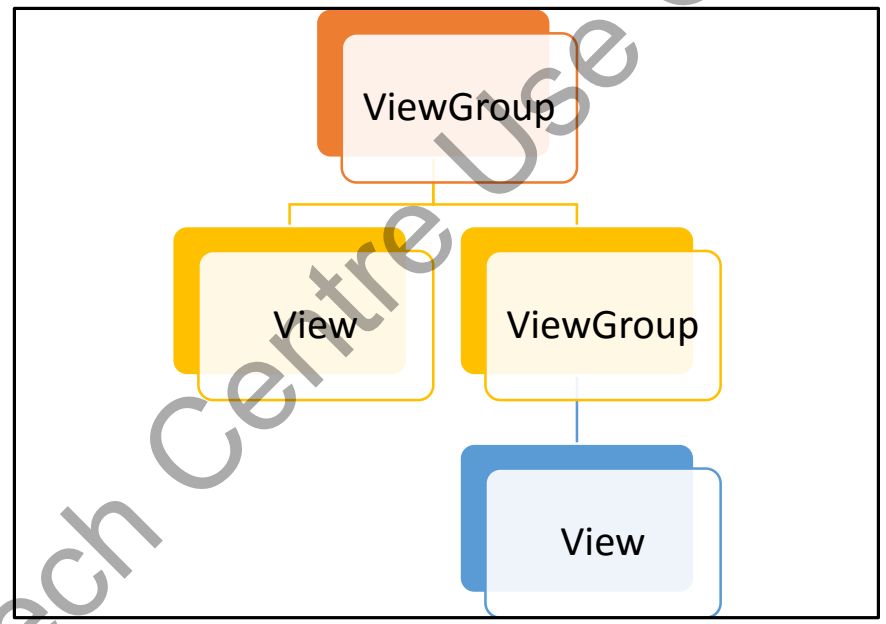
Different Layouts

View Object

- Basic building block of an Android app's user interface is the View object.
- View object is derived from the View class.
- All elements on an Android app's user interface are View objects and responsible for various activities.
- View object includes:
  - Location
  - Dimension

Types of Layouts

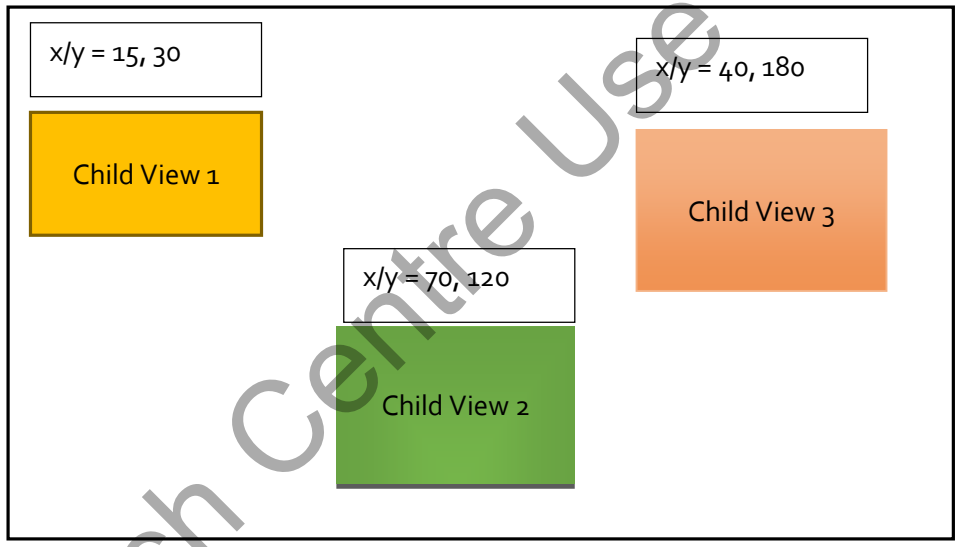
- Layouts supported by Android:
  - Absolute
  - Frame
  - Grid
  - Linear
  - List
  - Relative
  - Table
- Views can be grouped together in a ViewGroup, which acts as a container of Views and other ViewGroups.



ViewGroup and Views

AbsoluteLayout (1-2)

- AbsoluteLayout allows you to specify the exact locations of child objects or Views.
- The location is specified in x and y coordinates.



AbsoluteLayout

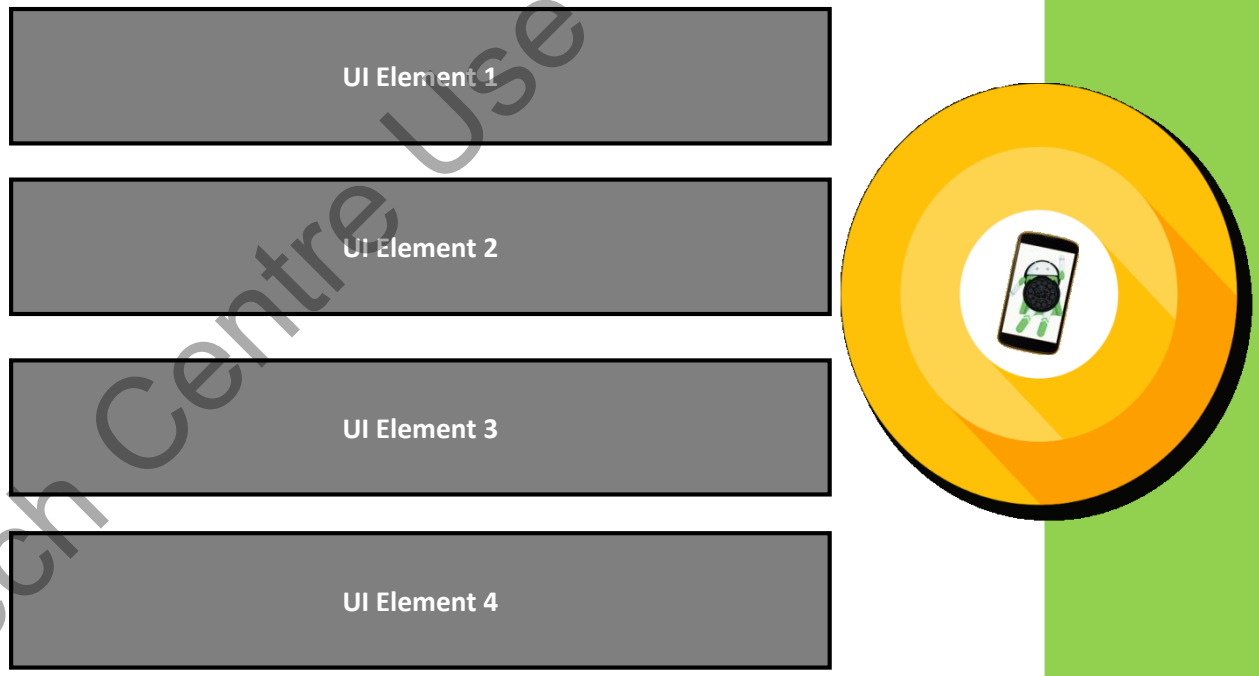
AbsoluteLayout (2-2)

- Add a button in the AbsoluteLayout:

```
<AbsoluteLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android" >
    <Button
        android:layout_width="190dp"
        android:layout_height="wrap_content"
        android:text="Test Button"
        android:layout_x="130px"
        android:layout_y="365px" />
</AbsoluteLayout>
```

FrameLayout

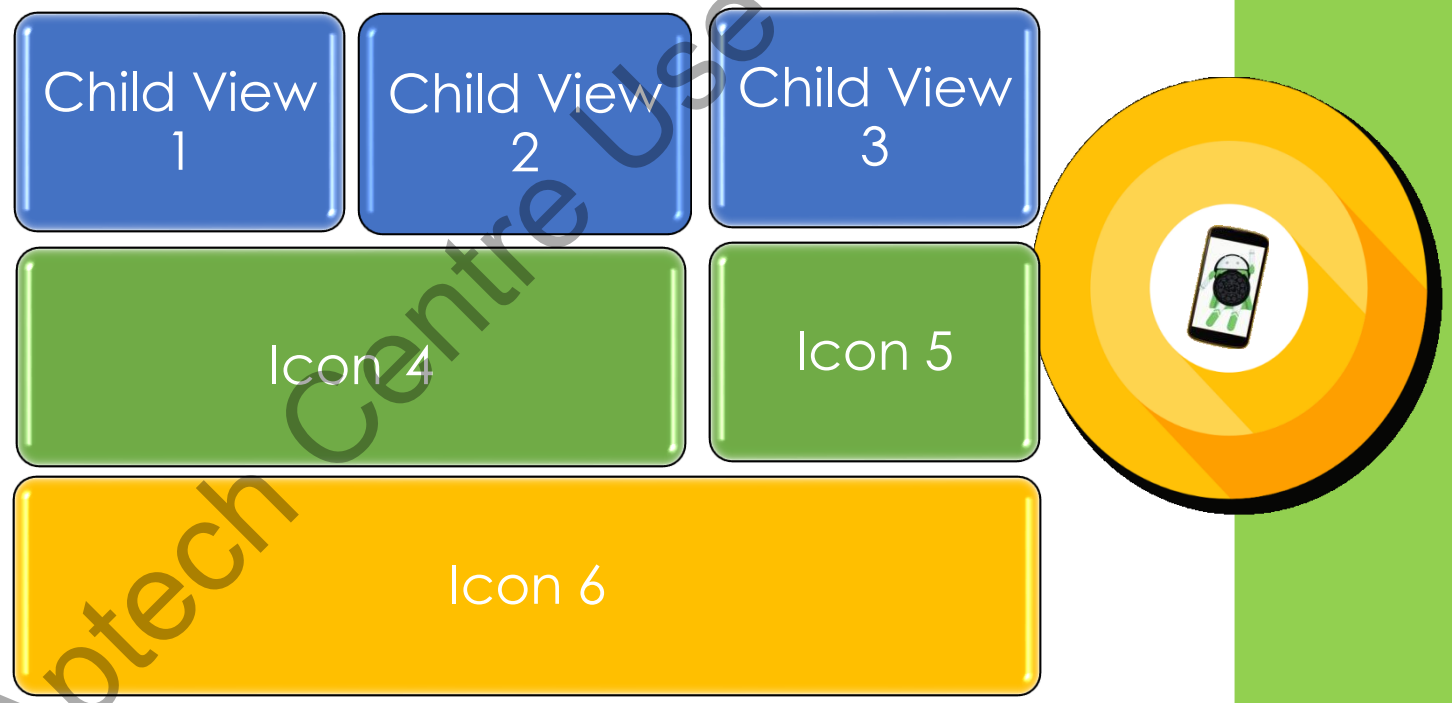
- FrameLayout display child Views in a single stack.
- FrameLayout blocks a specific area on the user interface to display a single view object of all the child.



FrameLayout

GridLayout (1-2)

- Place the child Views or objects in a rectangular grid.
- Grid is composed of a set of infinitely thin lines that separate the view area into blocks/cells



GridLayout

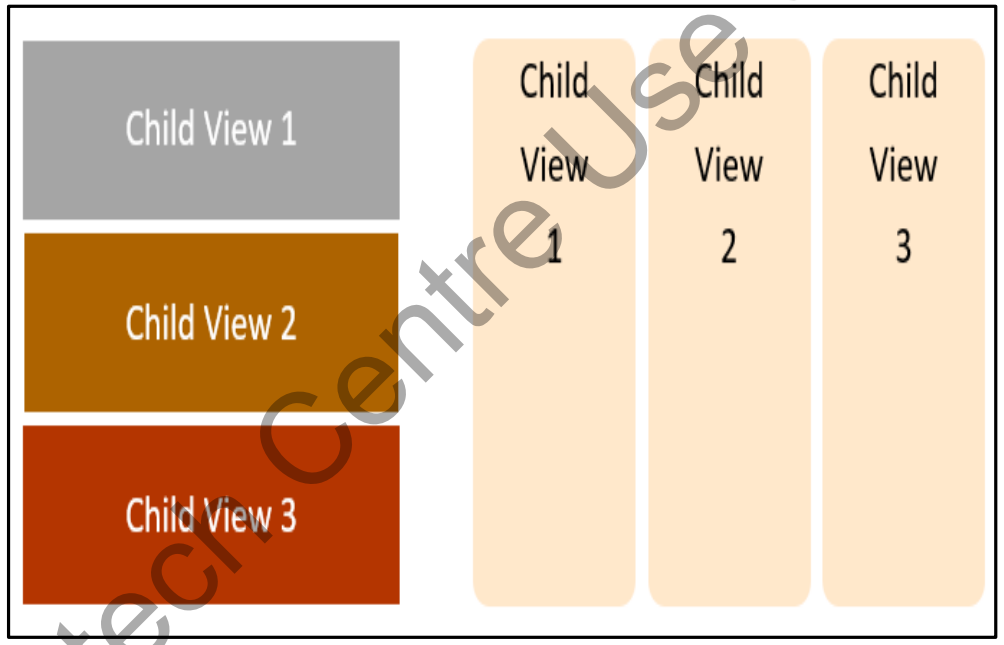
GridLayout (2-2)

- Defining a GridLayout:

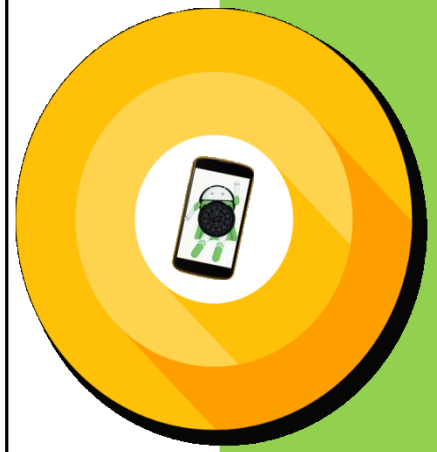
```
<?xml version="1.0" encoding="utf-8"?>
<GridView
xmlns:android="http://schemas.android.com/apk/res/and
roid"
    android:id="@+id/gridviewExampleId"
    android:layout_width="fill_parent"
    android:layout_height="“fill_parent"
     android:horizontalSpacing="10dp"
     android:verticalSpacing="10dp"
     android:columnWidth="40dp"
     android:gravity="“center"
     android:numColumns="auto_fit"
     android:stretchMode="columnWidth"
</GridView>
```

LinearLayout

- LinearLayout allows you to arrange child objects in a single row or column.
- Based on the specified orientation property horizontal or vertical.

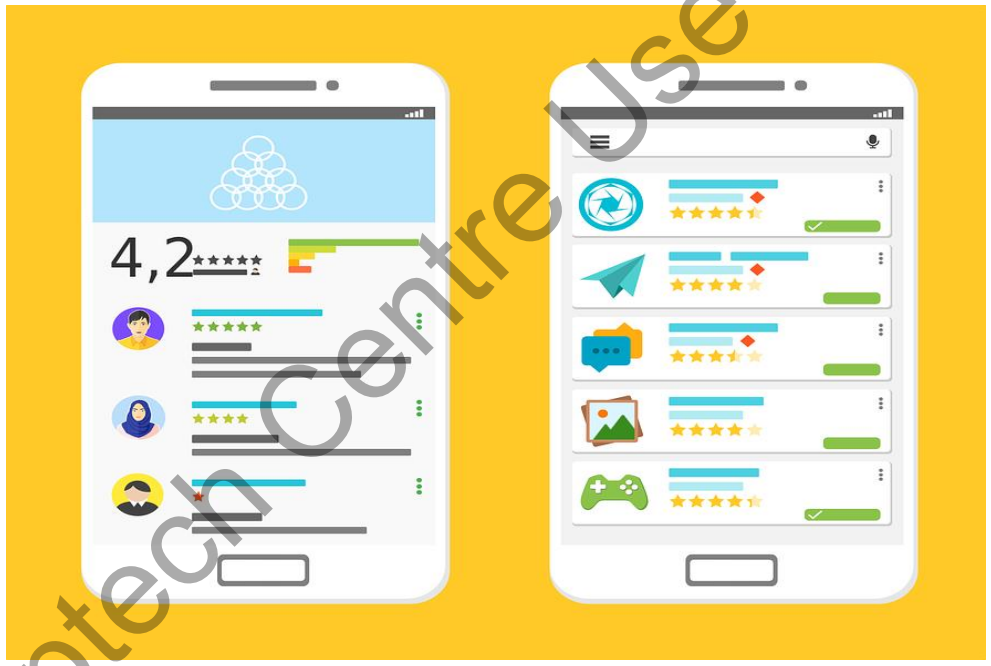


LinearLayout



ListView (1-2)

- ListView is a ViewGroup in which items are displayed in a vertically scrollable list.
- Adapter is responsible to convert content from source to view and add it to list.



ListView

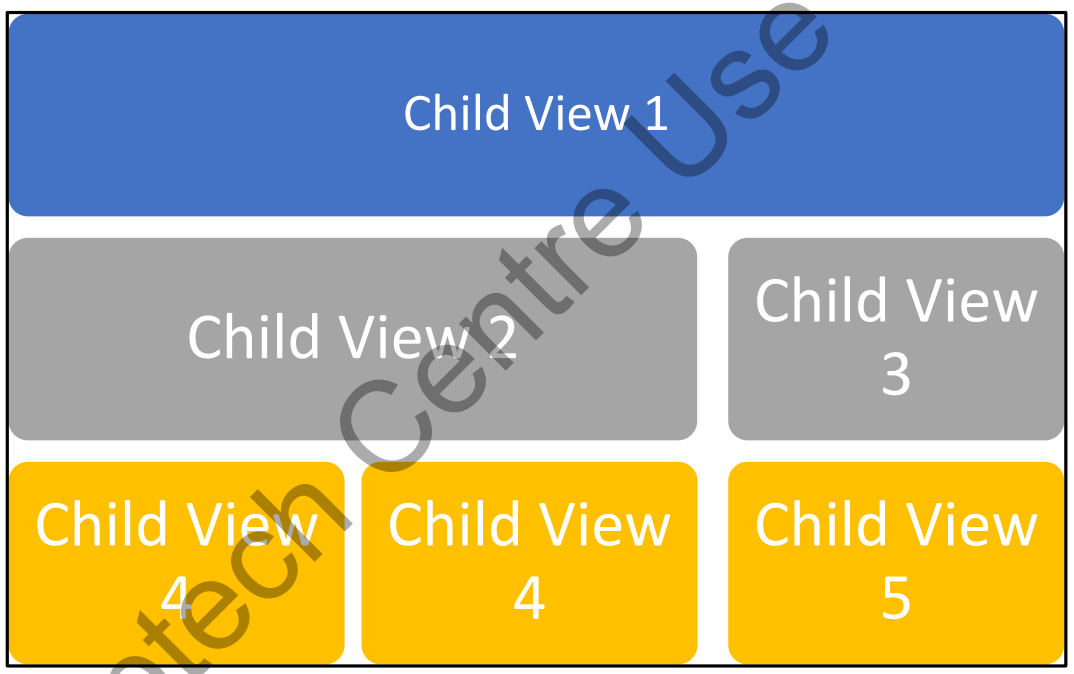
ListView (2-2)

- Defining ListView:

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
xmlns:android="http://schemas.android.com/apk/res/and
roid"
    android:id="@+id/listViewTextLabel"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="15dip"
    android:textSize="27dip"
    android:textStyle="bold"
    android:textStyle="italics" >
</TextView>
```

RelativeLayout

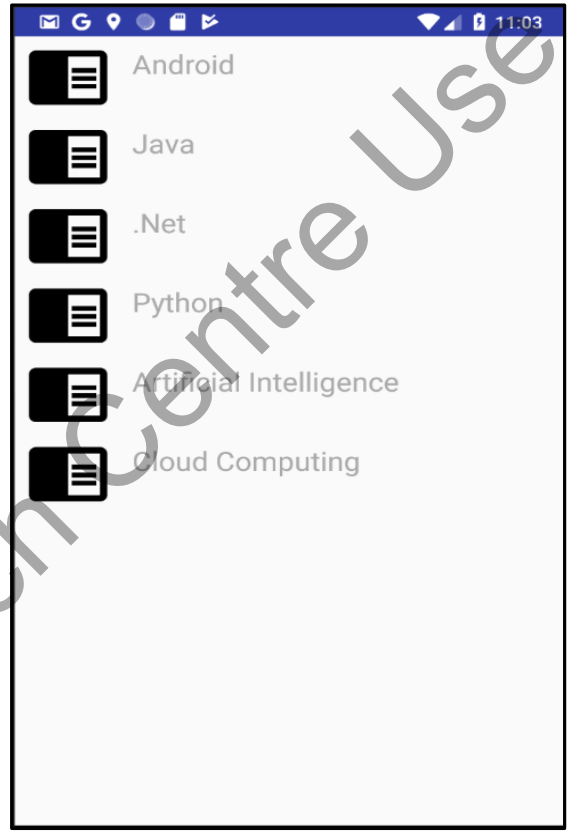
- RelativeLayout is used to specify the position of child Views or objects relative to each other.
- Relative to the parent container.



RelativeLayout

RecyclerView (1-2)

- Advanced version of the ListView
- Scrollable lists for large datasets, which might include frequently changing data.



RecyclerView Layout

RecyclerView (2-2)

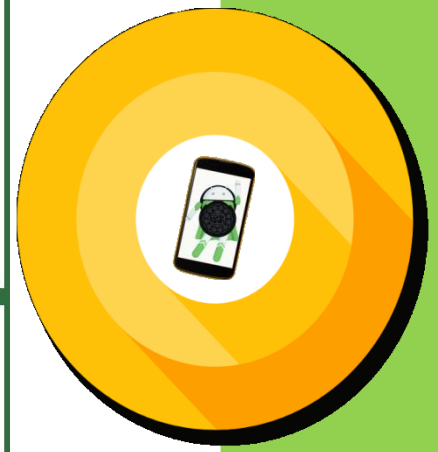
- To use RecyclerView class, you need to include the v7 support libraries in the Android project.
- Dependencies to add in build.gradle file of the app:

```
dependencies {
    implementation 'com.android.support:recyclerview-v7:27.1.1'
}
```

User Interface Controls (1-4)

- Android application user interface includes several components that bring interactivity to the application, such as:
  - Button
  - Edit text/TextView
  - Check box/Radio button
  - Date/Time picker
  - Toggle buttons
  - Progress bar

Types of UI Controls



User Interface Controls (2-4)

- Radio button:

```
<RadioButton
android:id="@+id/radioButtonExample"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
```

- Date picker:

```
<DatePicker
android:id="@+id/datePickerExample"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:datePickerMode="spinner"/>
```

User Interface Controls (3-4)

- Check box:

```
<CheckBox
android:id="@+id/checkboxExample"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="CheckBox Example"/>
```

- TextView:

```
<TextView
android:id="@+id/textViewExample"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Text View Example" />
```

User Interface Controls (4-4)

- Toggle button:

```
<ToggleButton
android:id="@+id/toggleButtonExample"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
```

- Progress bar:

```
<ProgressBar
android:id="@+id/progressBarExample"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
```

Summary (1-3)

- Layout is defined as the structure of your app's user interface and UI elements, such as buttons, text fields, check boxes, belong to the category of layouts.
- The basic building block of an Android app's user interface is the View object, which is derived from the View class.
- The View class is the base class for all UI widgets in an Android application, and is used to create interactive UI elements, such as drop-down lists, buttons, check boxes, and text boxes.

Summary (2-3)

- All the widgets that appear on the Android app's interface are View objects whereas the layout is the ViewGroup object.
- All the layout source code must be placed in the /res/layout folder.
- The Absolute layout lets you specify the exact locations of child objects or Views.
- The Frame layout is used to display child views in a single stack.
- The Grid layout allows you to place the child Views or objects in a rectangular grid, and is one of the most used layouts for Android applications.

Summary (3-3)

- The Linear layout lets you arrange child objects in a single column or row horizontally or vertically.
- The List layout allows you to create a group of several child Views or objects and display this group as a vertically scrollable list.
- The Relative layout lets you specify the position of child Views or objects relative to each other or relative to the parent container.
- Android application user interface includes several components that bring interactivity to the application, such as a text field allows a user to input text, whereas a radio buttons allows a user to select. These interactive components are called input controls or UI controls or UI widgets.