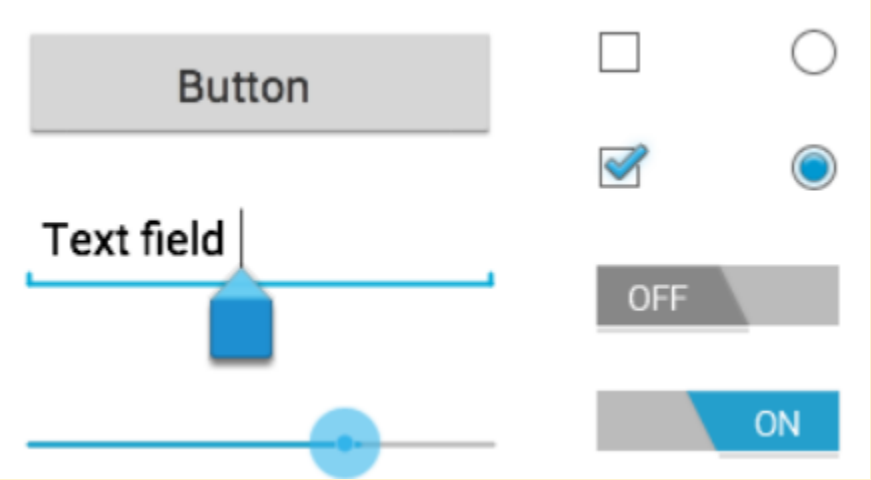
P P SAVANI UNIVERSITY

Module 3
Exploring User Interface Screen Elements



Topics to be covered

P P SAVANI UNIVERSITY

- Introducing Android Views and Layouts
- Displaying Text with TextView
- Retrieving Data From Users
- Using Buttons, Check Boxes and Radio Groups
- Getting Dates and Times from Users,
- Using Indicators to Display and Data to Users
- Adjusting Progress with SeekBar
- Providing Users with Options and Context Menus
- Handling User Events
- Working with Dialogs
- Working with Styles
- Working with Themes.

The Android View

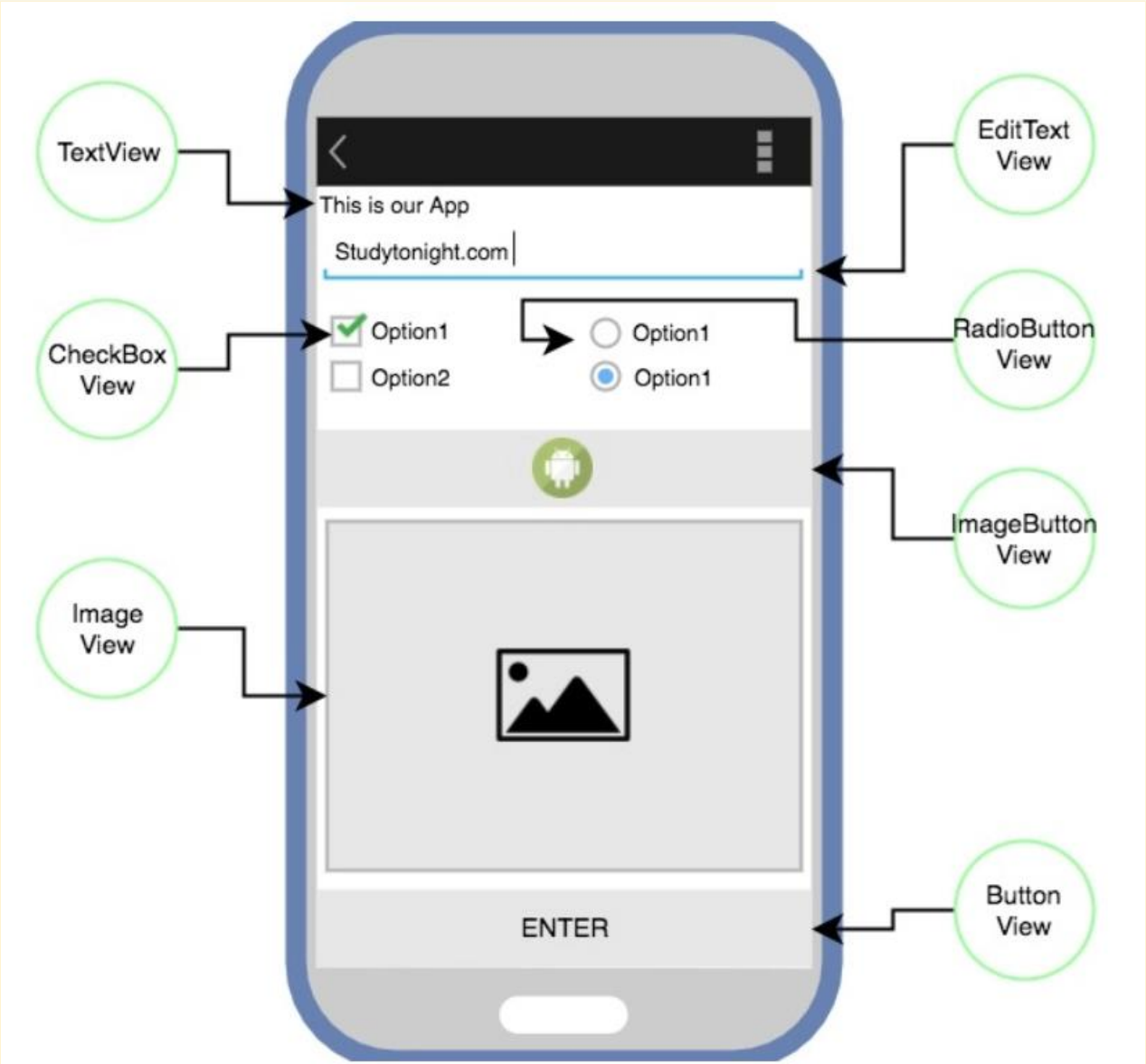
P P SAVANI UNIVERSITY

- The Android SDK has a Java package named android.view.
- This package contains a number of interfaces and classes related to drawing on the screen.
- View object refers to one of the classes within this package:

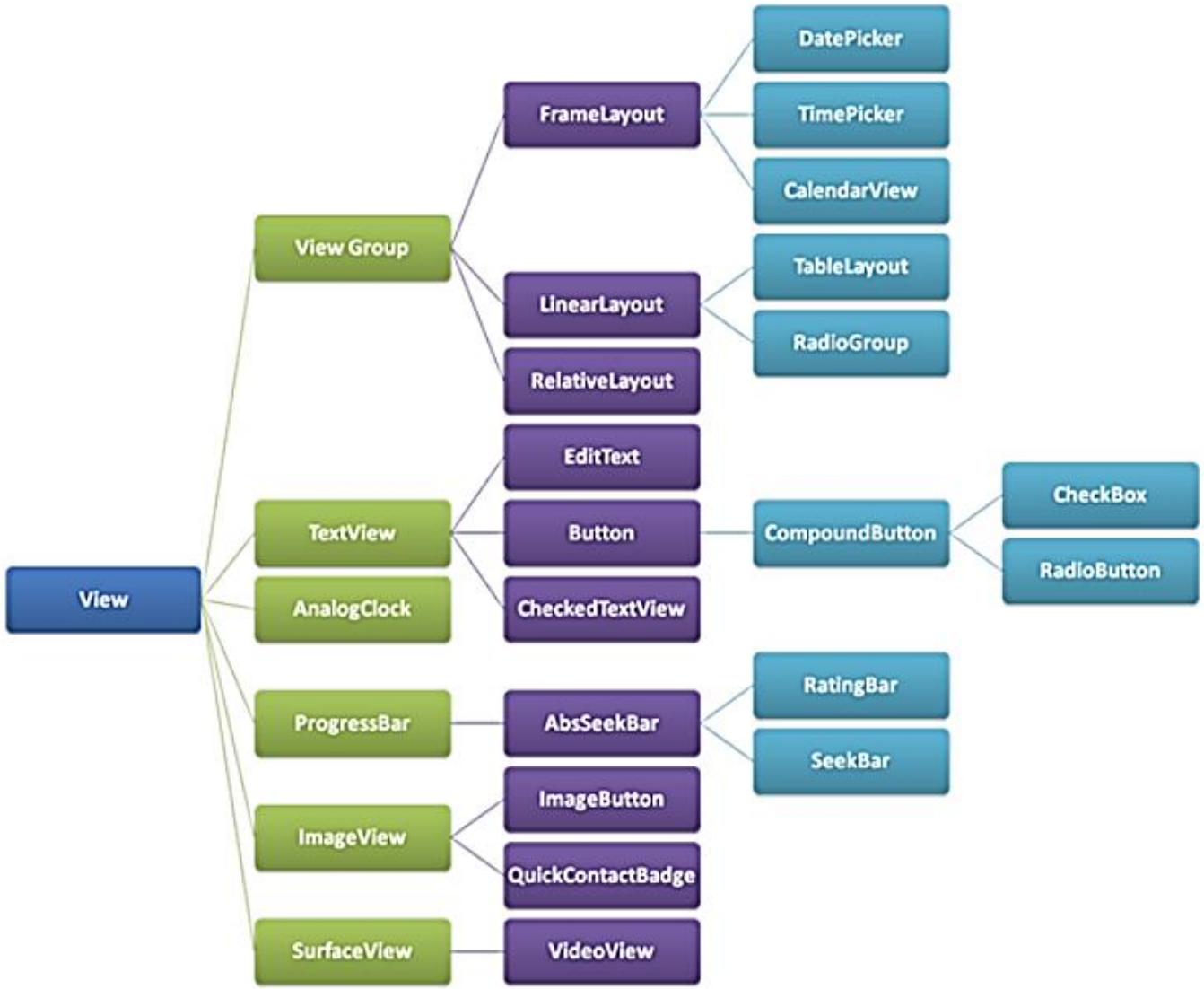
  Android.view.View

- The View class is the basic user interface building block within Android.
- It represents a rectangular portion of the screen. The View class servers as the base class for nearly all the user interface controls and layouts within the Android SDK.

Commonly used Views



View Class Hierarchy



The Android Controls

P P SAVANI UNIVERSITY

- The Android SDK contains a Java package named android.widget.
- When we refer to controls, we are typically referring to a class within this package.
- The Android SDK includes classes to draw most common objects, including ImageView, FrameLayout, EditText and Button classes.
- All these controls are derived from the View class.

The Android Controls

P P SAVANI UNIVERSITY

- User Interface Controls can be static or programmatically.
- Each control you want to be able to access programmatically must have a unique identifier specified using the android:id attribute.
- You use this identifier to access the control with the findViewById() method in your Activity class.

```
TextView tv = (TextView) findViewById(R.id.textview01);
```

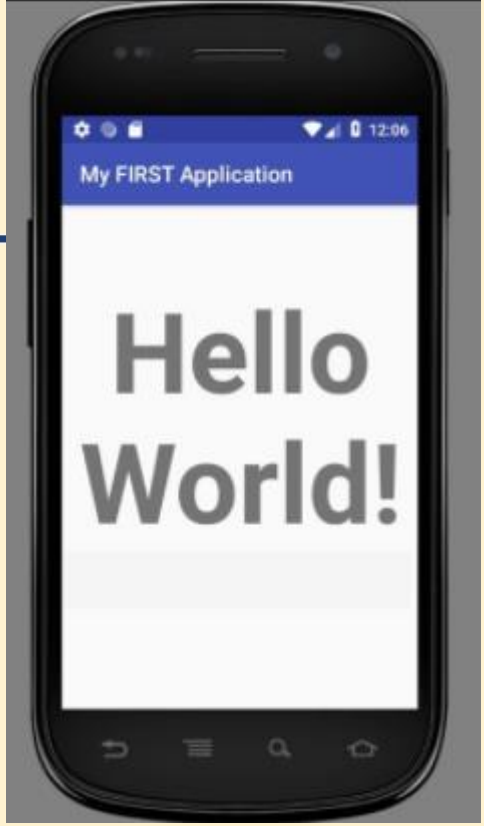
The Android Layout

P P SAVANI UNIVERSITY

- Found in android.widget package.
- It is a View object but it doesn’t actually draw anything specific on the screen. Instead it is a parent container for organizing other controls(children).
- Each type of layout control draws its children using particular rules.
- For instance, the LinearLayout controls draw its child controls in a single horizontal row or a single vertical column.
- Details will be in Module 4.

Displaying Text with TextView

```
public class TextView
extends View implements ViewTreeObserver.OnPreDrawListener

java.lang.Object
  ↳ android.view.View
      ↳ android.widget.TextView
```



- **A user interface element that displays text to the user.**

```
<TextView
    android:id="@+id/text_view_id"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:text="@string/hello" />
```

```
public class MainActivity extends Activity {

   protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final TextView helloTextView = (TextView) findViewById(R.id.text_view_id);
        helloTextView.setText(R.string.user_greeting);
    }
}
```

Creating Contextual Links in Text

P P SAVANI UNIVERSITY

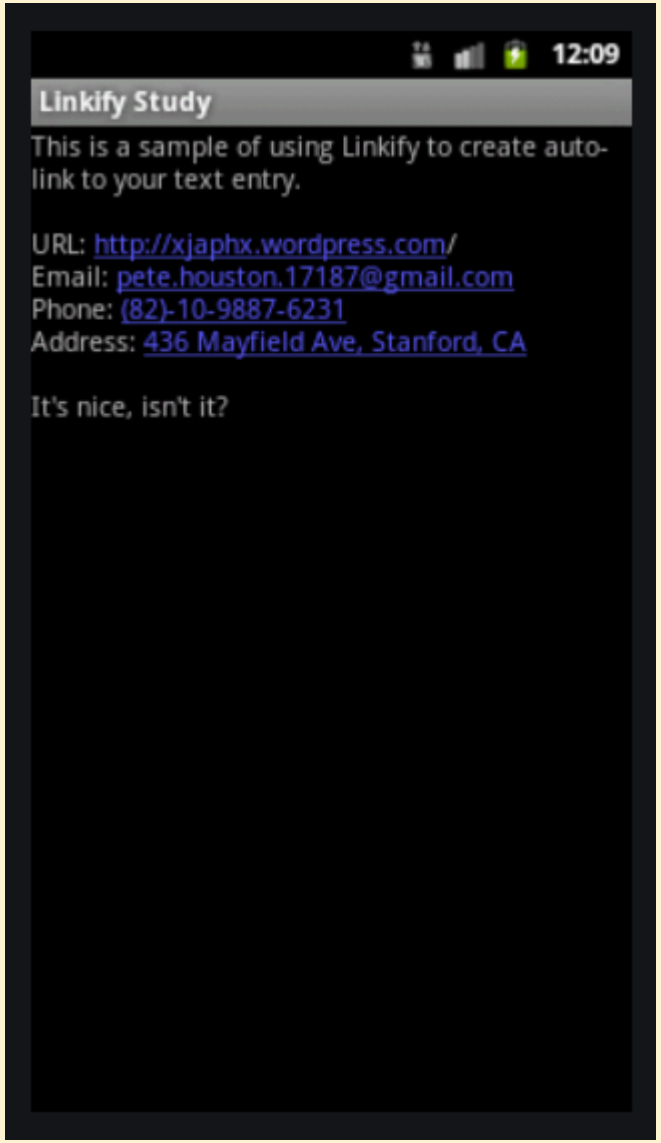
- **andorid:autoLink** => **By using this attribute, android can controls whether links(such as urls, emails, phone and address) are automatically found and converted to clickable links.**

1. **android:autoLink=”none”**
2. **android:autoLink=”email”**
3. **android:autoLink=”phone”**
4. **android:autoLink=”web”**
5. **android:autoLink=”map”**
6. **android:autoLink=”all”**

android:autoLink=”email”

P P SAVANI UNIVERSITY

```
<TextView
        android:id="@+id/txtViewEmail"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Email id: abc@gmail.com"
        android:autoLink="email"
        android:textSize="16sp"
        android:layout_margin="5dip">
</TextView>
```



Retrieving Data from Users with Text Fields

EditText

P P SAVANI UNIVERSITY

- In Android, EditText is a standard entry widget in android apps.
- EditText is a subclass of TextView with text editing operations. We often use EditText in our applications in order to provide an input or text field, especially in forms. The most simple example of EditText is Login or Sign-in form.

Using EditText Controls

P P SAVANI UNIVERSITY

```
public class EditText
extends TextView

java.lang.Object
  ↳ android.view.View
      ↳ android.widget.TextView
          ↳ android.widget.EditText
```

To retrieve data entered through Android EditText we do the following:

```
EditText simpleEditText = (EditText) findViewById(R.id.simpleEditText);
String editTextValue = simpleEditText.getText().toString();
```

```
<EditText
        android:id="@+id/Nametext"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="Enter your Name"
        android:inputType="text" />
```

activity_main.xml

Checking User Inputs with Input Filters

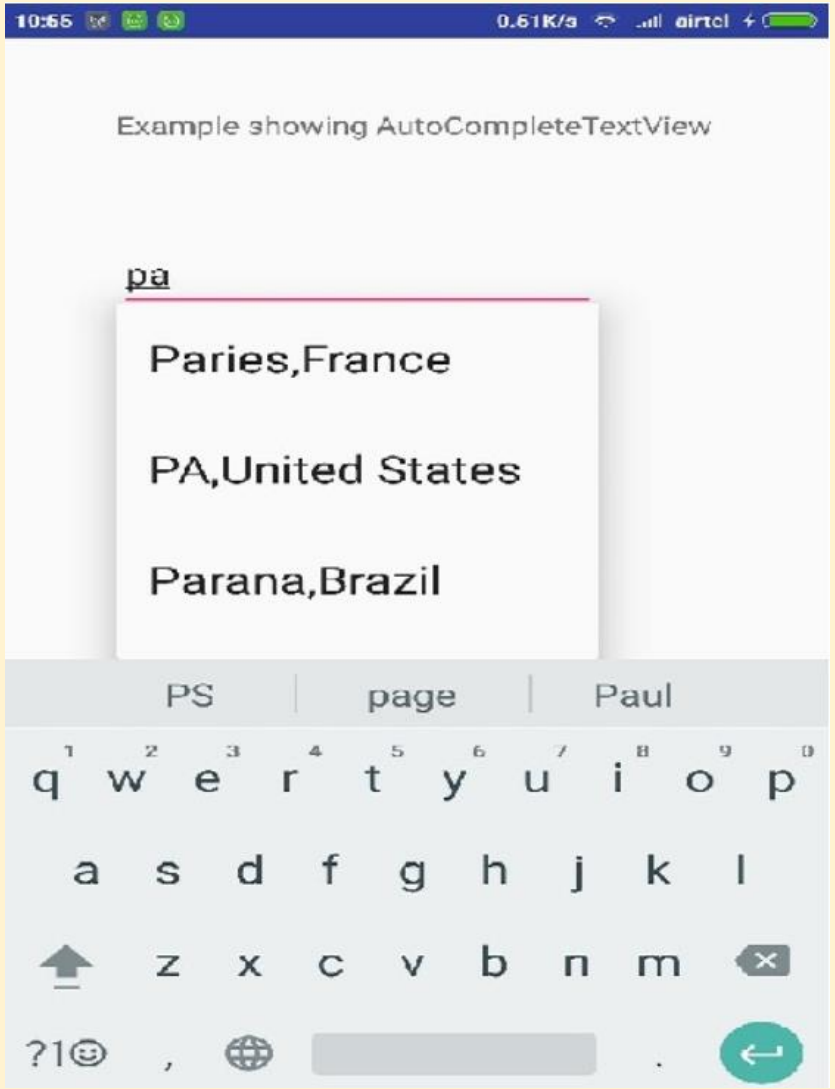
P P SAVANI UNIVERSITY

- Sometimes requirement comes in a way when one needs to restrict characters to be entered in Edit Text. Scenario can be “username should be greater than 6 characters”, “accepts only decimal number with 2 digits after fraction”, etc.

```
 EditText et = (EditText) findViewById(R.id.myEditText);
et.setFilters(new InputFilter[]{ new InputFilterMinMax("1", "12")});
```

```
 EditText editText = new EditText(this);
int maxLength = 3;
editText.setFilters(new InputFilter[] {new InputFilter.LengthFilter(maxLength)});
```

Helping the User with Autocompletion



AutoCompleteTextView

```
<AutoCompleteTextView
…
/>
```

“MainActivity.java” for AutoCompletionTextView

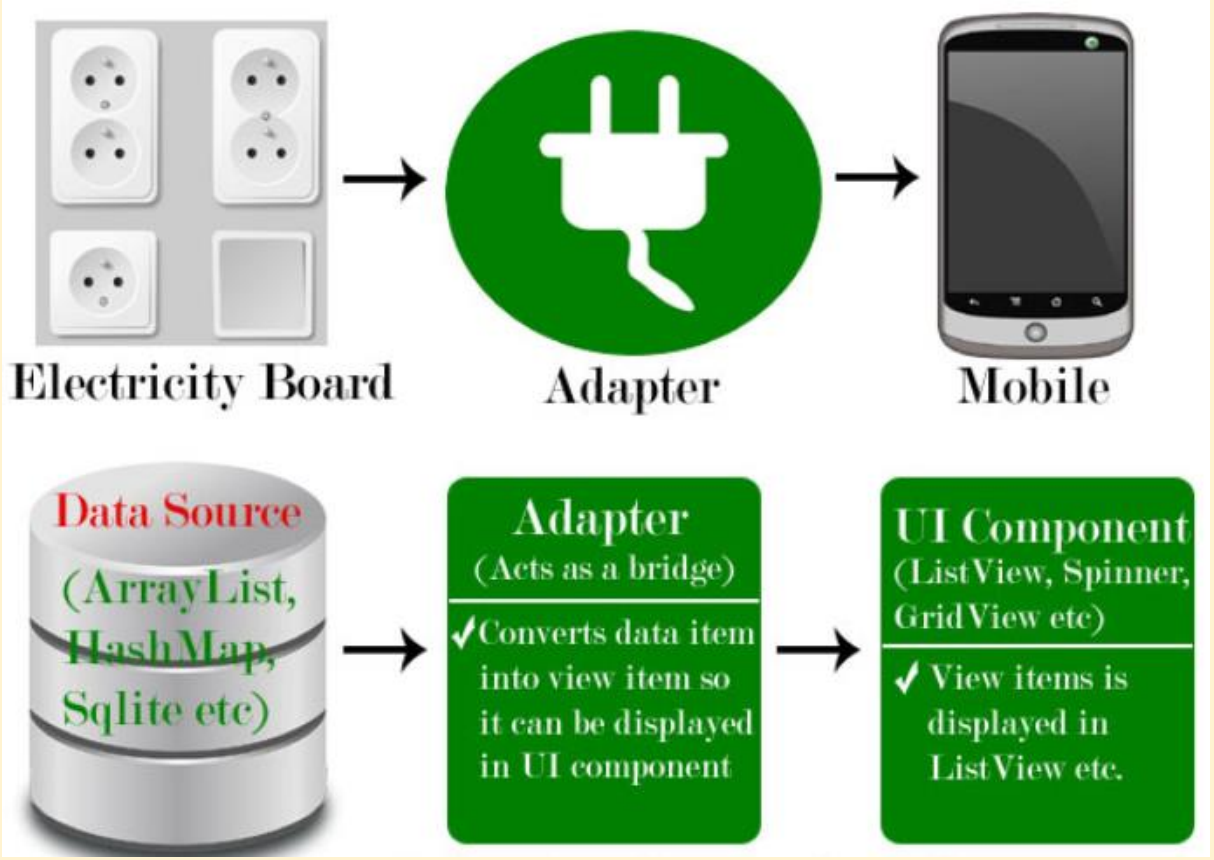
P P SAVANI UNIVERSITY

```
public class MainActivity extends Activity {
   AutoCompleteTextView autocomplete;

   String[] arr = { "Paries,France", "PA,United States","Parana,Brazil",
      "Padua,Italy", "Pasadena,CA,United States"};

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      autocomplete = (AutoCompleteTextView)
      findViewById(R.id.autoCompleteTextView1);

      ArrayAdapter<String> adapter = new ArrayAdapter<String>
      (this,android.R.layout.select_dialog_item, arr);

      autocomplete.setThreshold(2);
      autocomplete.setAdapter(adapter);
   }
}
```

Adapter in Android

P P SAVANI UNIVERSITY

- In Android, Adapter is a bridge between UI component and data source that helps us to fill data in UI component. It holds the data and send the data to an Adapter view then view can takes the data from the adapter view and shows the data on different views like as ListView, GridView, Spinner e tc



ArrayAdapter In Android

P P SAVANI UNIVERSITY

- **Whenever we have a list of single items which is backed by an Array, we can use ArrayAdapter. For instance, list of phone contacts, countries or names.**

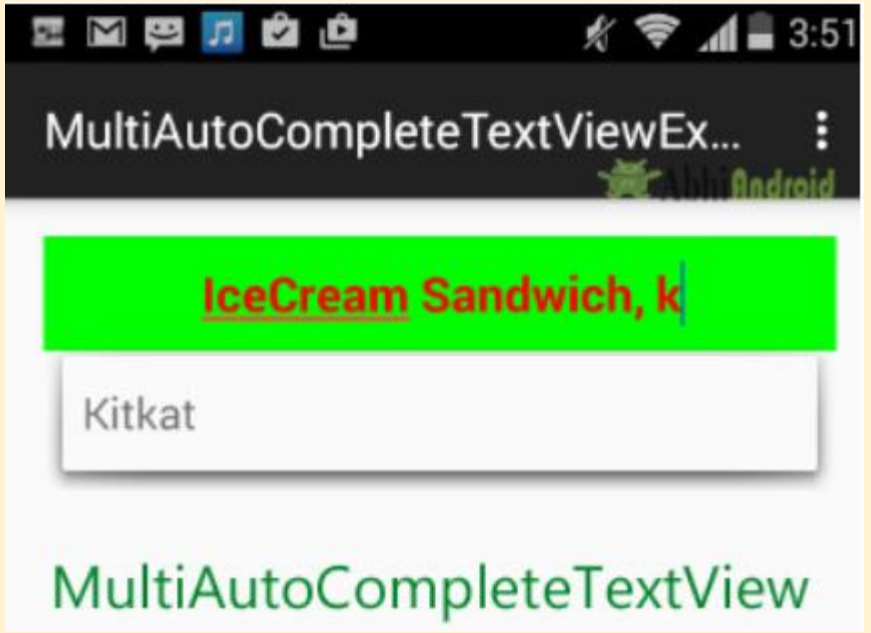
```
ArrayAdapter(Context context, int resource, int textViewResourceId, T[] objects)
```

```
String animalList[] = {"Lion","Tiger","Monkey","Elephant","Dog","Cat","Camel"};
ArrayAdapter arrayAdapter = new ArrayAdapter(this, R.layout.list_view_items, R.id.textView, animalList);
```

MultiAutoCompleteTextView

P P SAVANI UNIVERSITY

- One most important difference between MultiAutoCompleteTextView and AutoCompleteTextView is that AutoCompleteTextView can hold or select only single value at single time but MultiAutoCompleteTextView can hold multiple string words value at single time. These all values are separated by comma(,).



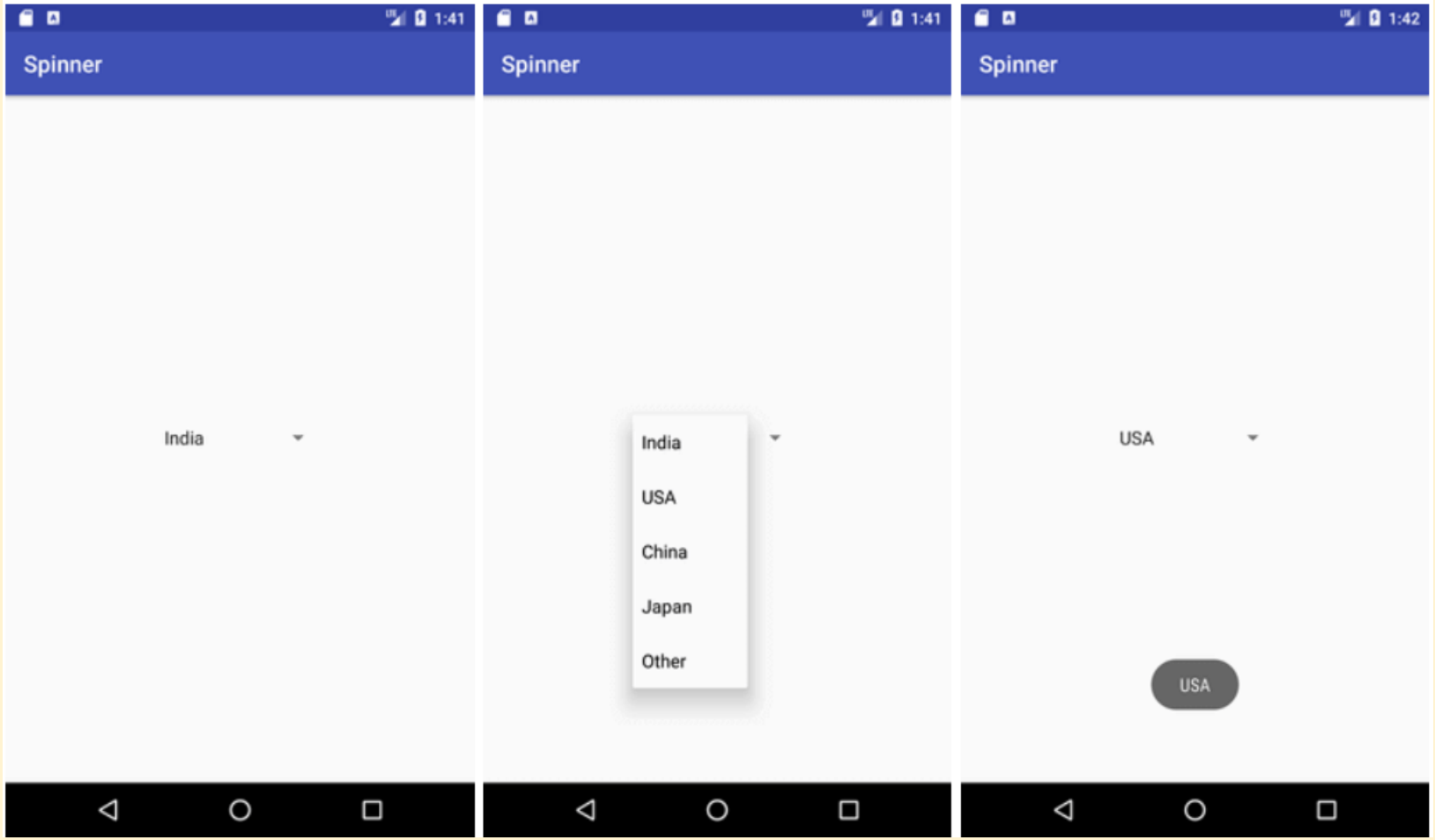
MultiAutoCompleteTextView Example


```
String[] androidVersionNames = {"Aestro", "Blender", "CupCake", "Donut", "Eclair",
"Froyo", "Gingerbread", "HoneyComb", "IceCream Sandwich", "Jellibean", "Kitkat", "
Lollipop", "MarshMallow"};


@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_main);

// initiate a MultiAutoCompleteTextView

MultiAutoCompleteTextView simpleMultiAutoCompleteTextView = (MultiAutoCompleteText
View) findViewById(R.id.simpleMultiAutoCompleteTextView);

// set adapter to fill the data in suggestion list

ArrayAdapter<String> versionNames = new ArrayAdapter<String>(this, android.R.layou
t.simple_list_item_1, androidVersionNames);

simpleMultiAutoCompleteTextView.setAdapter(versionNames);


// set threshold value 1 that help us to start the searching from first character

simpleMultiAutoCompleteTextView.setThreshold(1);

// set tokenizer that distinguish the various substrings by comma

simpleMultiAutoCompleteTextView.setTokenizer(new MultiAutoCompleteTextView.CommaTo
kenizer());

}
```

Giving Users Choices Using Spinner Controls

- It can be used to display the multiple options to the user in which only one item can be selected by the user.
- Android spinner is like the drop down menu with multiple values from which the end user can select only one value.
- Android spinner is associated with AdapterView. So you need to use one of the adapter classes with spinner..

Giving Users Choices Using Spinner Controls



```
String[] country = { "India", "USA", "China", "Japan", "Other"};

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
   //Getting the instance of Spinner and applying OnItemSelectedListener on it
    Spinner spin = (Spinner) findViewById(R.id.spinner);
    spin.setOnItemSelectedListener(this);

    //Creating the ArrayAdapter instance having the country list
    ArrayAdapter aa = new ArrayAdapter(this,android.R.layout.simple_spinner_item,country);
    aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    //Setting the ArrayAdapter data on the Spinner
    spin.setAdapter(aa);

}

//Performing action onItemSelected and onNothing selected
@Override
public void onItemSelected(AdapterView<?> arg0, View arg1, int position, long id) {
    Toast.makeText(getApplicationContext(),country[position] , Toast.LENGTH_LONG).show();
}
```

P P SAVANI UNIVERSITY

File: MainActivity.java

The code to display item on the spinner and perform event handling

User Selections with Buttons and Switches

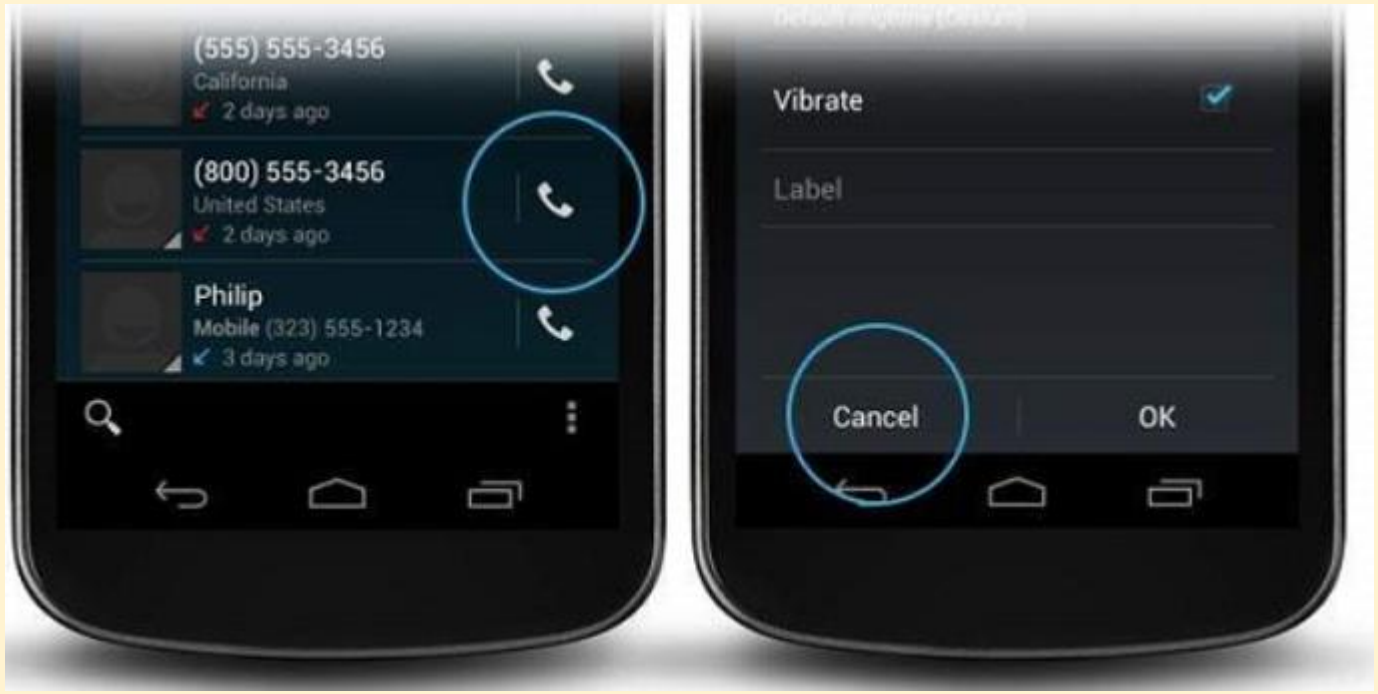
P P SAVANI UNIVERSITY

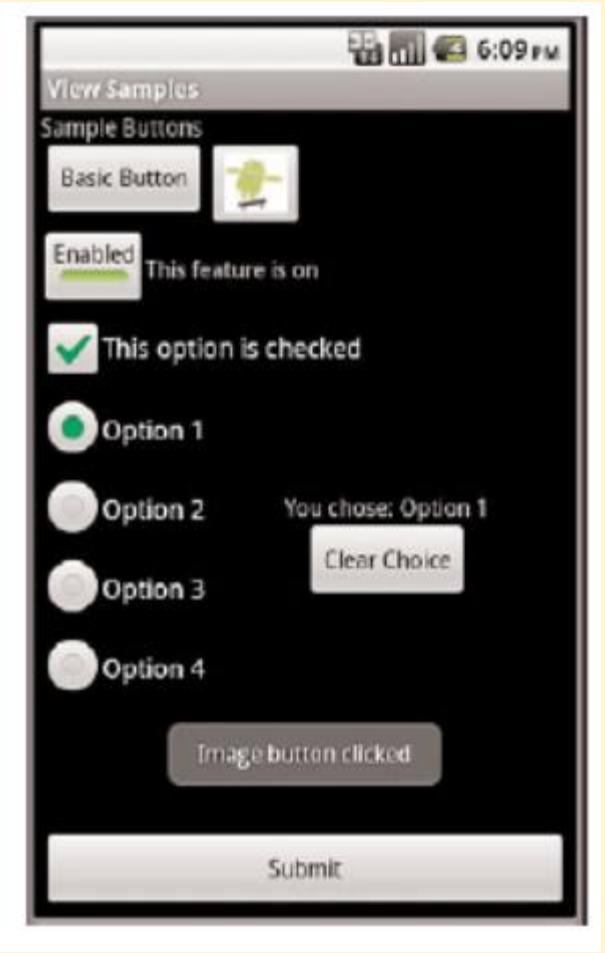
- Using Basic Buttons
- Using Checkbox and ToggleButton Controls
- Using RadioGroup and RadioButton

Using Basic Buttons

P P SAVANI UNIVERSITY

- A basic Button is often used to perform some sort of action, such as submitting a form or confirming a selection.
- A basic Button control can contain a text or image label.
- The android.widget.Button class provides a basic Button implementation in the Android SDK.
- Common Button Text such as “Yes”,”No”,”OK”,”Cancel” or “Submit”.


(555) 555-3456
California
2 days ago
(800) 555-3456
United States
2 days ago
Philip
Mobile (323) 555-1234
3 days ago
Vibrate
Label
Cancel
OK


6:09 PM
View Samples
Sample Buttons
Basic Button
Enabled
This feature is on
This option is checked
Option 1
Option 2
You chose: Option 1
Clear Choice
Option 3
Option 4
Image button clicked
Submit

A Button Listener

P P SAVANI UNIVERSITY

- **A button won't do anything, other than animate, without some code to handle the click event.**

```
setContentView(R.layout.buttons);

final Button basic_button = (Button) findViewById(R.id.basic_button);

basic_button.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

Toast.makeText(ButtonsActivity.this,

"Button clicked", Toast.LENGTH_SHORT).show();

}

});
```

ImageButton

P P SAVANI UNIVERSITY

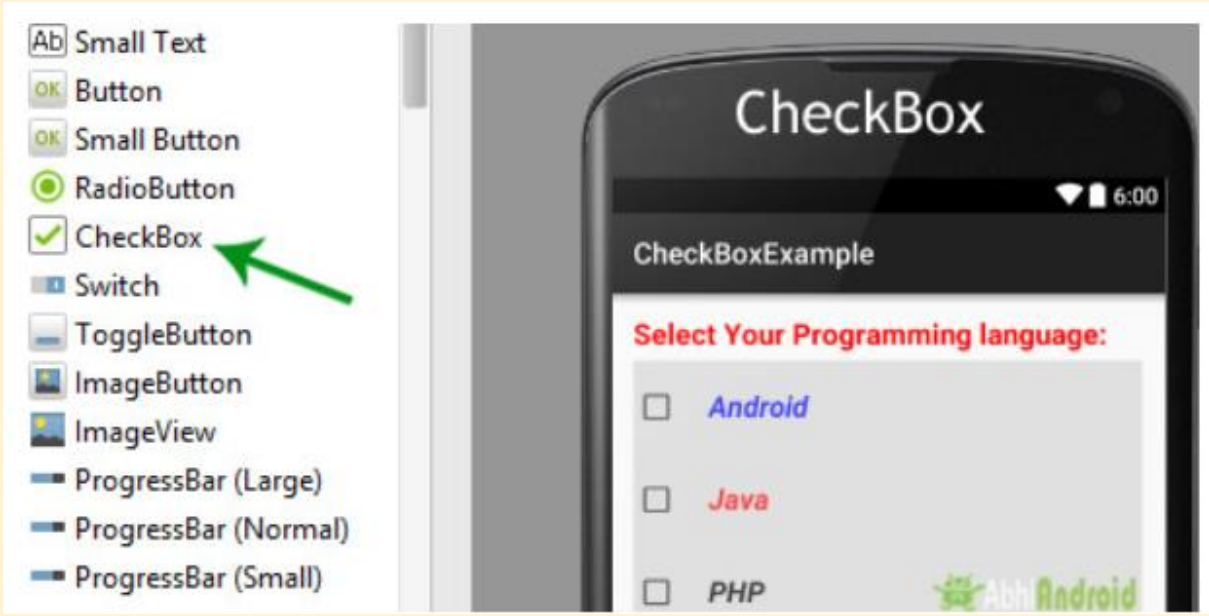
- A button with its primary label as an image is an ImageButton.
- An ImageButton is, for most purposes, almost exactly like a basic button.
- Click actions are handled in the same way. The primary difference is that you can set its src attribute to be an image

```
<ImageButton

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:id="@+id/image_button"

android:src="@drawable/droid" />
```

Checkbox Control

P P SAVANI UNIVERSITY

- The check box button is often used in lists of items where the user can select multiple items.
- The Android check box contains a text attribute that appears to the side of the check box.
- This is used in a similar way to the label of a basic button. In fact, it's basically a TextView next to the button.



Checkbox Control

P P SAVANI UNIVERSITY

- In Android, CheckBox is a type of two state button either unchecked or checked in Android. Or you can say it is a type of on/off switch that can be toggled by the users.
- You should use checkbox when presenting a group of selectable options to users that are not mutually exclusive. CompoundButton is the parent class of CheckBox class.
- In android there is a lot of usage of check box. For example, to take survey in Android app we can list few options and allow user to choose using CheckBox.
- The user will simply checked these checkboxes rather than type their own option in EditText. Another very common use of CheckBox is as remember me option in Login form.

Listener on Checkbox

P P SAVANI UNIVERSITY

```
final CheckBox check_button = (CheckBox) findViewById(R.id.checkbox);

check_button.setOnClickListener(new View.OnClickListener() {

public void onClick (View v) {

TextView tv = (TextView)findViewById(R.id.checkbox);

tv.setText(check_button.isChecked() ?

“This option is checked” :

“This option is not checked”);

}

});
```

Example


```
    android = (CheckBox) findViewById(R.id.androidCheckBox);
    android.setOnClickListener(this);
    java = (CheckBox) findViewById(R.id.javaCheckBox);
    java.setOnClickListener(this);
    python = (CheckBox) findViewById(R.id.pythonCheckBox);
    python.setOnClickListener(this);
    php = (CheckBox) findViewById(R.id.phpCheckBox);
    php.setOnClickListener(this);
    unity3D = (CheckBox) findViewById(R.id.unityCheckBox);
    unity3D.setOnClickListener(this);
}

@Override
public void onClick(View view) {

    switch (view.getId()) {
        case R.id.androidCheckBox:
            if (android.isChecked())
                Toast.makeText(getApplicationContext(), "Android", Toast.LENGTH_LONG).show();
            break;
        case R.id.javaCheckBox:
            if (java.isChecked())
                Toast.makeText(getApplicationContext(), "Java", Toast.LENGTH_LONG).show();
            break;
        case R.id.phpCheckBox:
            if (php.isChecked())
                Toast.makeText(getApplicationContext(), "PHP", Toast.LENGTH_LONG).show();
            break;
        case R.id.pythonCheckBox:
            if (python.isChecked())
                Toast.makeText(getApplicationContext(), "Python", Toast.LENGTH_LONG).show();
            break;
        case R.id.unityCheckBox:
            if (unity3D.isChecked())
                Toast.makeText(getApplicationContext(), "Unity 3D", Toast.LENGTH_LONG).show()
            break;
    }
}
```

ToggleButton

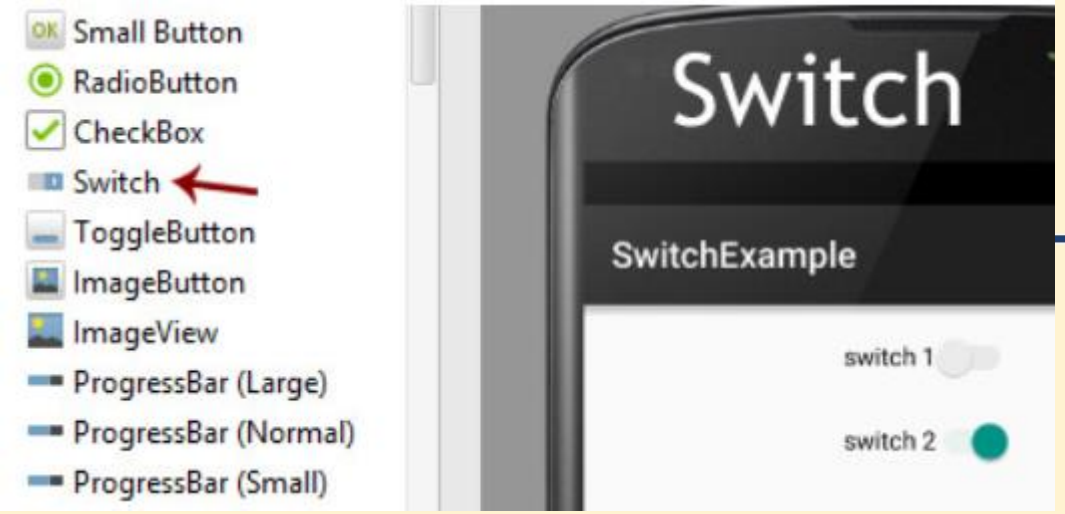
P P SAVANI UNIVERSITY

- A Toggle Button is similar to a check box in behavior but is usually used to show or alter the on or off state of something. Like the CheckBox, it has a state (checked or not).
- Also like the check box, the act of changing what displays on the button is handled for us.
- Unlike the CheckBox, it does not show text next to it.
- Instead, it has two text fields. The first attribute is textOn, which is the text that displays on the button when its checked state is on. The second attribute is textOff, which is the text that displays on the button when its checked state is off. The default text for these is “ON” and “OFF,” respectively.
- The most simple example of ToggleButton is doing on/off in sound, Bluetooth, wifi, hotspot etc

ToggleButton (On/Off)

P P SAVANI UNIVERSITY

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // initiate toggle button's
    simpleToggleButton1 = (ToggleButton) findViewById(R.id.simpleToggleButton1);
    simpleToggleButton2 = (ToggleButton) findViewById(R.id.simpleToggleButton2);
    submit = (Button) findViewById(R.id.submitButton);
    submit.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String status = "ToggleButton1 : " + simpleToggleButton1.getText() + "\n" + "ToggleButton2 : " + simpleToggleButton2.getText();
            Toast.makeText(getApplicationContext(), status, Toast.LENGTH_SHORT).show(); // display the current state of toggle button's
        }
    });
}
```



Switch (On/Off)

P P SAVANI UNIVERSITY

- In Android, Switch is a two-state toggle switch widget that can select between two options.
- It is used to display checked and unchecked state of a button providing slider control to user.
- Switch is a subclass of CompoundButton. It is basically an off/on button which indicate the current state of Switch.
- It is commonly used in selecting on/off in Sound, Bluetooth, WiFi etc.

Switch (On/Off)

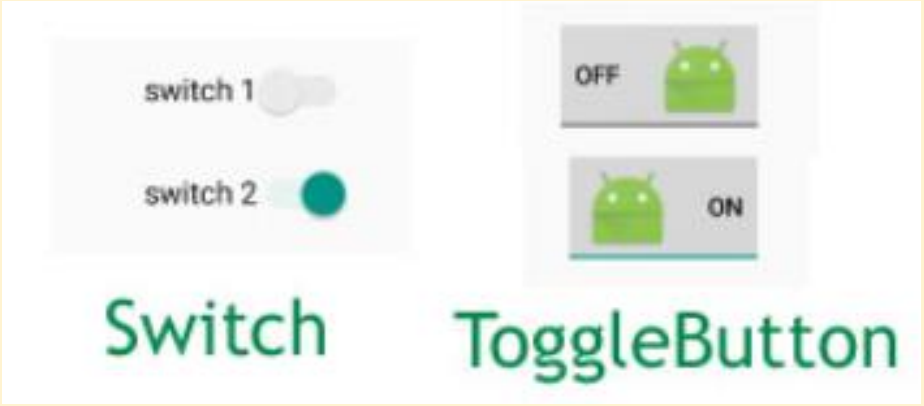
P P SAVANI UNIVERSITY


```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // initiate view's
        simpleSwitch1 = (Switch) findViewById(R.id.simpleSwitch1);
        simpleSwitch2 = (Switch) findViewById(R.id.simpleSwitch2);
        submit = (Button) findViewById(R.id.submitButton);
        submit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String statusSwitch1, statusSwitch2;
                if (simpleSwitch1.isChecked())
                    statusSwitch1 = simpleSwitch1.getTextOn().toString();
                else
                    statusSwitch1 = simpleSwitch1.getTextOff().toString();
                if (simpleSwitch2.isChecked())
                    statusSwitch2 = simpleSwitch2.getTextOn().toString();
                else
                    statusSwitch2 = simpleSwitch2.getTextOff().toString();
                Toast.makeText(getApplicationContext(), "Switch1 :" + statusSwitch1 + "\n" + "Switch2 :" + statusSwitch2, Toast.LENGTH_LONG).show();
                // display the current state for switch's
            }
        });

    }
```

ToggleButton Vs Switch In Android

P P SAVANI UNIVERSITY

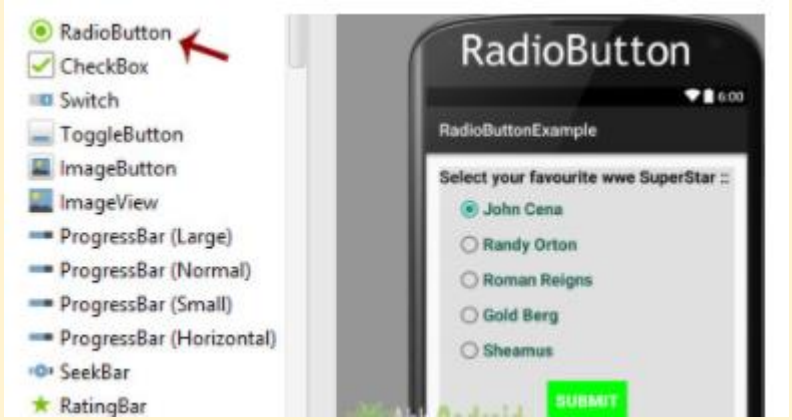
- **ToggleButton allow the users to change the setting between two states like turn on/off your wifi, Bluetooth etc from your phone's setting menu.**
- **Since, Android 4.0 version (API level 14) there is an another kind of ToggleButton called Switch which provide the user slider control.**



RadioButton and RadioGroup

P P SAVANI UNIVERSITY

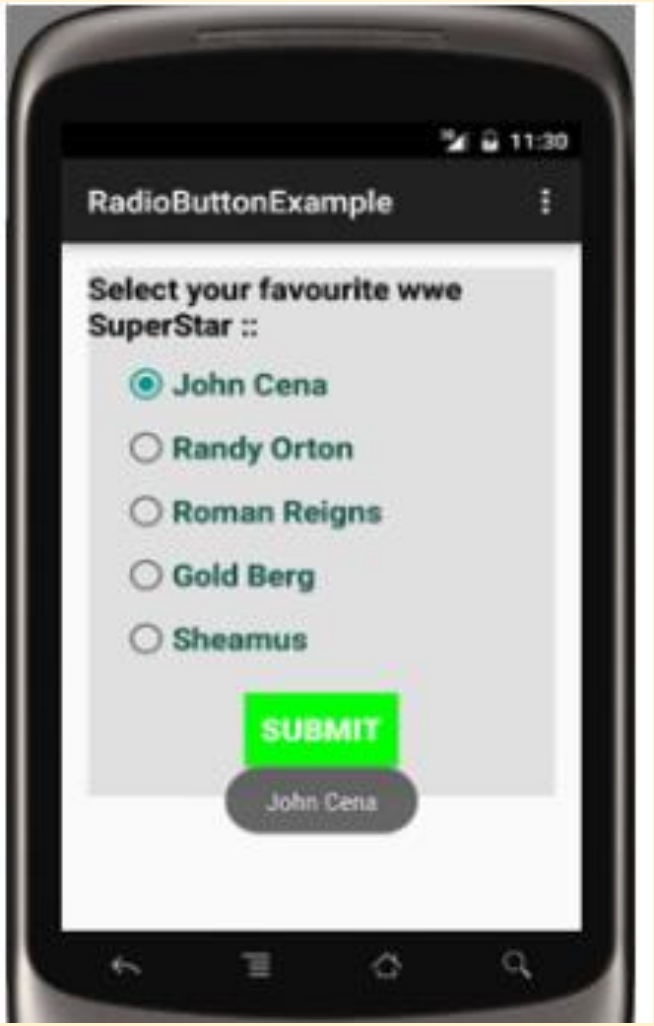
- RadioButton are mainly used together in a RadioGroup.
- In RadioGroup checking the one radio button out of several radio button added in it will automatically unchecked all the others.
- It means at one time we can checked only one radio button from a group of radio buttons which belong to same radio group.
- The most common use of radio button is in Quiz Android App code.



RadioGroup and RadioButton(Example)

P P SAVANI UNIVERSITY

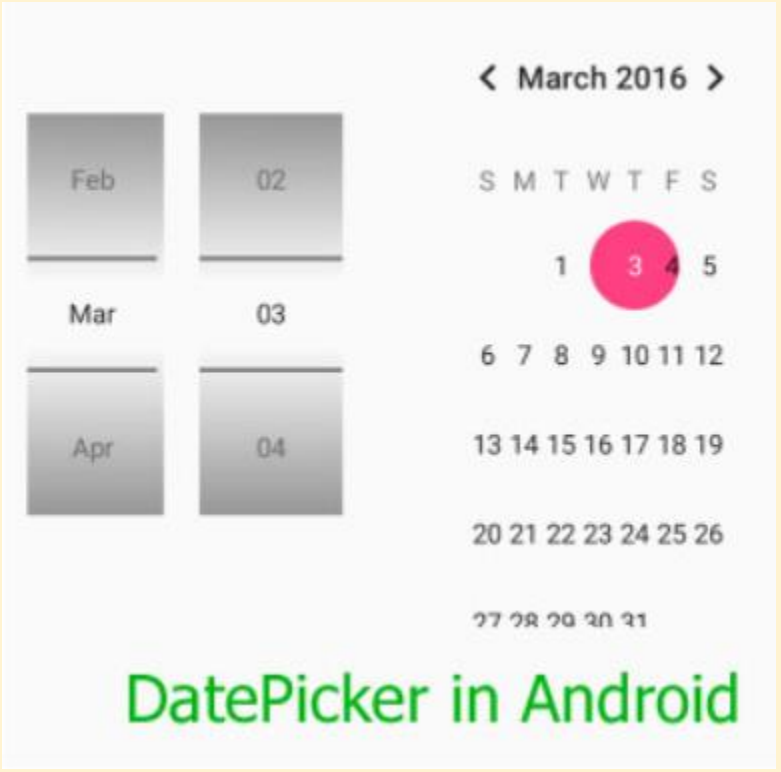
```
RadioButton johnCena, randyOrton, goldBerg, romanReigns, sheamus;
String selectedSuperStar;
Button submit;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    johnCena = (RadioButton) findViewById(R.id.johnCena);
    randyOrton = (RadioButton) findViewById(R.id.randyOrton);
    goldBerg = (RadioButton) findViewById(R.id.goldBerg);
    romanReigns = (RadioButton) findViewById(R.id.romanReigns);
    sheamus = (RadioButton) findViewById(R.id.sheamus);
    submit = (Button) findViewById(R.id.submitButton);
    submit.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (randyOrton.isChecked()) {
                selectedSuperStar = randyOrton.getText().toString();
            } else if (sheamus.isChecked()) {
                selectedSuperStar = sheamus.getText().toString();
            } else if (johnCena.isChecked()) {
                selectedSuperStar = johnCena.getText().toString();
            } else if (romanReigns.isChecked()) {
                selectedSuperStar = romanReigns.getText().toString();
            } else if (goldBerg.isChecked()) {
                selectedSuperStar = goldBerg.getText().toString();
            }
            Toast.makeText(getApplicationContext(), selectedSuperStar, Toast.LENGTH_LONG).show();
        }
    });
}
```



DatePicker

P P SAVANI UNIVERSITY

- In Android, DatePicker is a widget used to select a date. It allows to select date by day, month and year in your custom UI (user interface).
- If we need to show this view as a dialog then we have to use a DatePickerDialog class.
- For selecting time Android also provides timepicker to select time



Methods of DatePicker

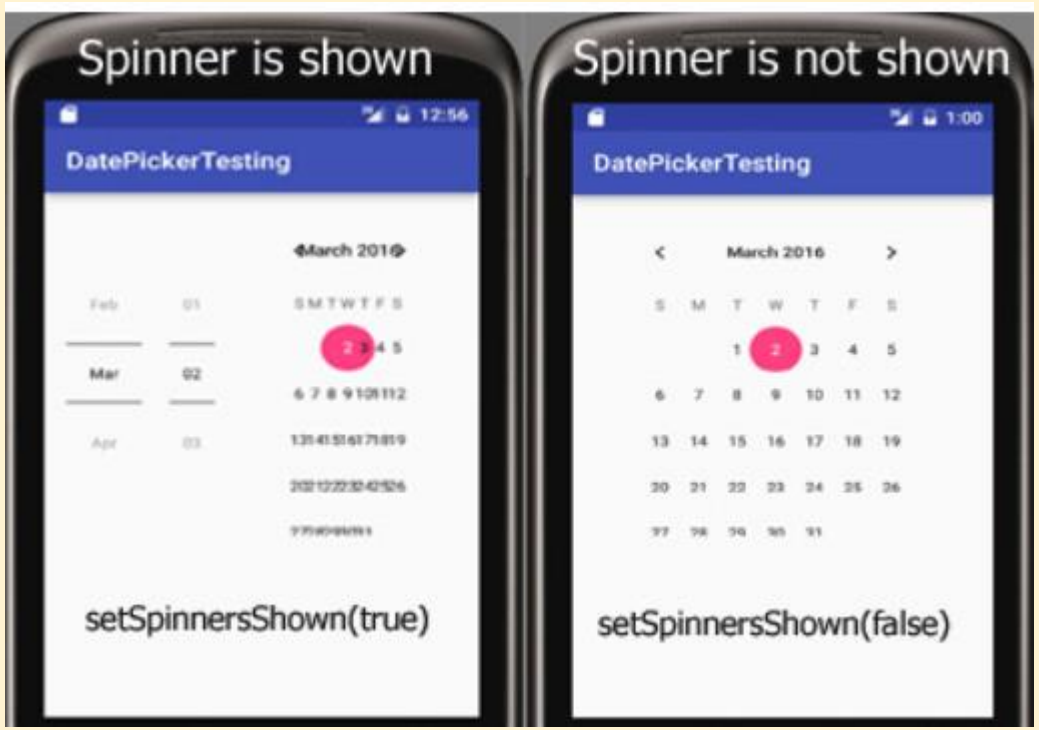
P P SAVANI UNIVERSITY

- **getDayOfMonth():** to get the selected day of the month from a date picker.
- **getMonth():**to get the selected month from a date picker
- **getYear():**to get the selected year from a date picker.
- **getFirstDayOfWeek()**:to get the first day of the week.
- **setSpinnersShown(boolean shown) :** to set whether the spinner of the date picker in shown or not

setSpinnersShown(boolean shown) for DatePicker

P P SAVANI UNIVERSITY

```
DatePicker simpleDatePicker = (DatePicker)findViewById(R.id.simpleDatePicker); // initiate a date picker

simpleDatePicker.setSpinnersShown(false); // set false value for the spinner shown function
```



DatePicker Example in Android Studio

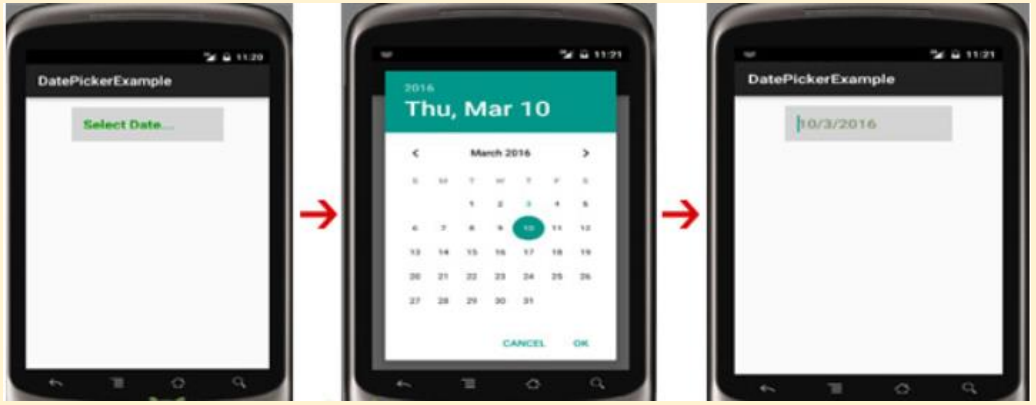
P P SAVANI UNIVERSITY


```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // initiate the date picker and a button
    simpleDatePicker = (DatePicker) findViewById(R.id.simpleDatePicker);
    submit = (Button) findViewById(R.id.submitButton);
    // perform click event on submit button
    submit.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // get the values for day of month , month and year from a date picker
            String day = "Day = " + simpleDatePicker.getDayOfMonth();
            String month = "Month = " + (simpleDatePicker.getMonth() + 1);
            String year = "Year = " + simpleDatePicker.getYear();
            // display the values by using a toast
            Toast.makeText(getApplicationContext(), day + "\n" + month + "\n" + year, Toast.LENGTH_LONG).sh
        }
    });
```

DatePicker Example 2 in Android Studio

P P SAVANI UNIVERSITY

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // initiate the date picker and a button
        date = (EditText) findViewById(R.id.date);
        // perform click event on edit text
        date.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // calender class's instance and get current date , month and year from calender
                final Calendar c = Calendar.getInstance();
                int mYear = c.get(Calendar.YEAR); // current year
                int mMonth = c.get(Calendar.MONTH); // current month
                int mDay = c.get(Calendar.DAY_OF_MONTH); // current day
                // date picker dialog
                datePickerDialog = new DatePickerDialog(MainActivity.this,
                        new DatePickerDialog.OnDateSetListener() {

                            @Override
                            public void onDateSet(DatePicker view, int year,
                                                  int monthOfYear, int dayOfMonth) {
                                // set day of month , month and year value in the edit text
                                date.setText(dayOfMonth + "/"
                                        + (monthOfYear + 1) + "/" + year);

                            }
                        }, mYear, mMonth, mDay);
                datePickerDialog.show();
            }
        });

    }


}
```



TimePicker

P P SAVANI UNIVERSITY

- In Android, TimePicker is a widget used for selecting the time of the day in either AM/PM mode or 24 hours mode.
- The displayed time consist of hours, minutes and clock format.
- If we need to show this view as a Dialog then we have to use a TimePickerDialog class.

Methods of TimePicker

P P SAVANI UNIVERSITY

1. setCurrentHour(Integer currentHour)
2. setCurrentMinute(Integer currentMinute)
3. getCurrentHour()
4. getCurrentMinute()
5. setIs24HourView(Boolean is24HourView)
6. is24HourView()
7. setOnTimeChangedListener(TimePicker.OnTimeChangedListener onTimeChangedListener):

```
TimePicker simpleTimePicker = (TimePicker)findViewById(R.id.simpleTimePicker); // initiate a time picker

simpleTimePicker.setOnTimeChangedListener(new TimePicker.OnTimeChangedListener() {
@Override
public void onTimeChanged(TimePicker view, int hourOfDay, int minute) {

}
});
```

Example of TimePicker in Android Studio

P P SAVANI UNIVERSITY


```
TextView time;
TimePicker simpleTimePicker;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //  initiate the view's
    time = (TextView) findViewById(R.id.time);
    simpleTimePicker = (TimePicker) findViewById(R.id.simpleTimePicker);
    simpleTimePicker.setIs24HourView(false); // used to display AM/PM mode
    // perform set on time changed listener event
    simpleTimePicker.setOnTimeChangedListener(new TimePicker.OnTimeChangedListener() {
        @Override
        public void onTimeChanged(TimePicker view, int hourOfDay, int minute) {
            // display a toast with changed values of time picker
            Toast.makeText(getApplicationContext(), hourOfDay + "  " + minute, Toast.LENGTH_SHORT).show(
            time.setText("Time is :: " + hourOfDay + " : " + minute); // set the current time in text vi
        }
    });
}
```

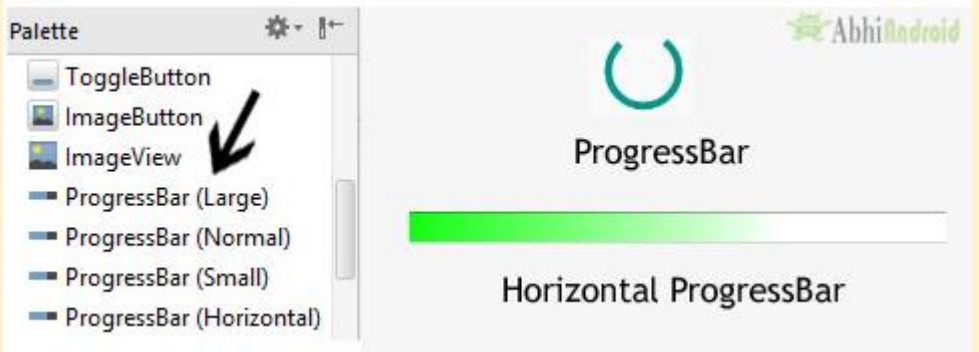
Example2 of TimePickerDialog in Android Studio

P P SAVANI UNIVERSITY

```
EditText time;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //  initiate the edit text
    time = (EditText) findViewById(R.id.time);
    // perform click event listener on edit text
    time.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Calendar mcurrentTime = Calendar.getInstance();
            int hour = mcurrentTime.get(Calendar.HOUR_OF_DAY);
            int minute = mcurrentTime.get(Calendar.MINUTE);
            TimePickerDialog mTimePicker;
            mTimePicker = new TimePickerDialog(MainActivity.this, new TimePickerDialog.OnTimeSetListener()
                        @Override
                        public void onTimeSet(TimePicker timePicker, int selectedHour, int selectedMinute)
                            time.setText(selectedHour + ":" + selectedMinute);
                    }
            }, hour, minute, true);//Yes 24 hour time
            mTimePicker.setTitle("Select Time");
            mTimePicker.show();

        }
    });
}
```

Indicating Progress with ProgressBar

- In Android, ProgressBar is used to display the status of work being done like analyzing status of work or downloading a file etc.
- In Android, by default a progress bar will be displayed as a spinning wheel but If we want it to be displayed as a horizontal bar then we need to use style attribute as horizontal.
- It mainly use the “android.widget.ProgressBar” class.



Important Methods Used In ProgressBar

1. getMax() – returns the maximum value of progress bar
2. getProgress() – returns current progress value

```
ProgressBar simpleProgressBar=(ProgressBar)findViewById(R.id.simpleProgressBar); // initiate the progress
int progressValue=simpleProgressBar.getProgress(); // get progress value from the progress bar
```

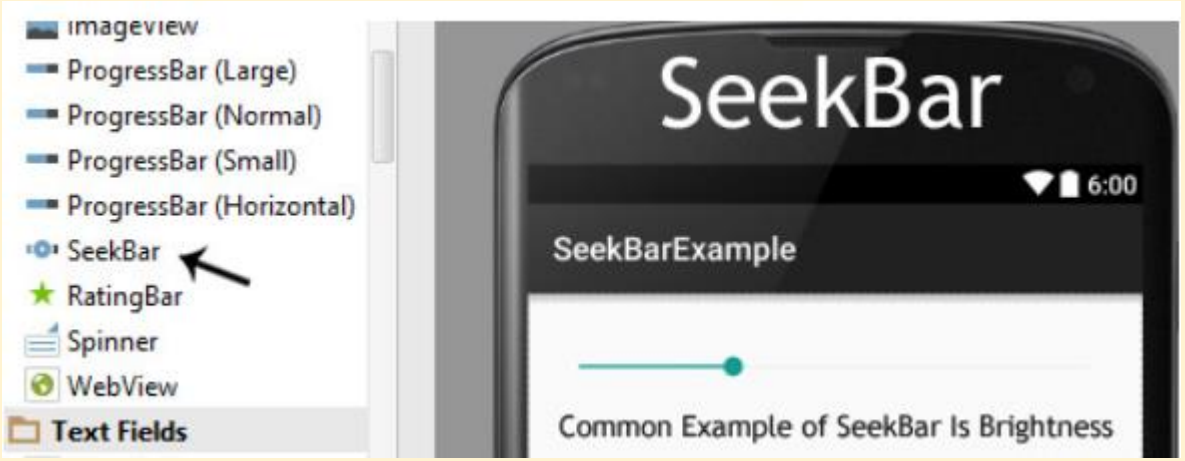
Example


```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // initiate progress bar and start button
    final ProgressBar simpleProgressBar = (ProgressBar) findViewById(R.id.simpleProgressBar);
    Button startButton = (Button) findViewById(R.id.startButton);
    // perform click event on button
    startButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // visible the progress bar
            simpleProgressBar.setVisibility(View.VISIBLE);
        }
    });
}
```

Adjusting Progress with Seek Bars

P P SAVANI UNIVERSITY

- In Android, SeekBar is an extension of ProgressBar that adds a draggable thumb, a user can touch the thumb and drag left or right to set the value for current progress.
- SeekBar is one of the very useful user interface element in Android that allows the selection of integer values using a natural user interface. An example of SeekBar is your device's brightness control and volume control



Listener To Notify The Changes In SeekBar:

P P SAVANI UNIVERSITY

- **SeekBar.OnSeekBarChangeListener is a listener used as a callback that notifies client when the progress level of seekbar has been changed.**
- **seekBarInstanceVariable.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {…}**
  - **This method is used to notify the user changes/actions in the SeekBar.**

Methods Needs To Be Implemented

P P SAVANI UNIVERSITY

1. public void onProgressChanged (SeekBar seekBar, int progresValue, boolean fromUser) {…} –
This listener method will be invoked if any change is made in the SeekBar.

2. public void onStartTrackingTouch(SeekBar seekBar) {…} –
This listener method will be invoked at the start of user's touch event. Whenever a user touch the thumb for dragging this method will automatically called.

3. public void onStopTrackingTouch(SeekBar seekBar) {…} –
This listener method will be invoked at the end of user touch event. Whenever a user stop dragging the thump this method will be automatically called.

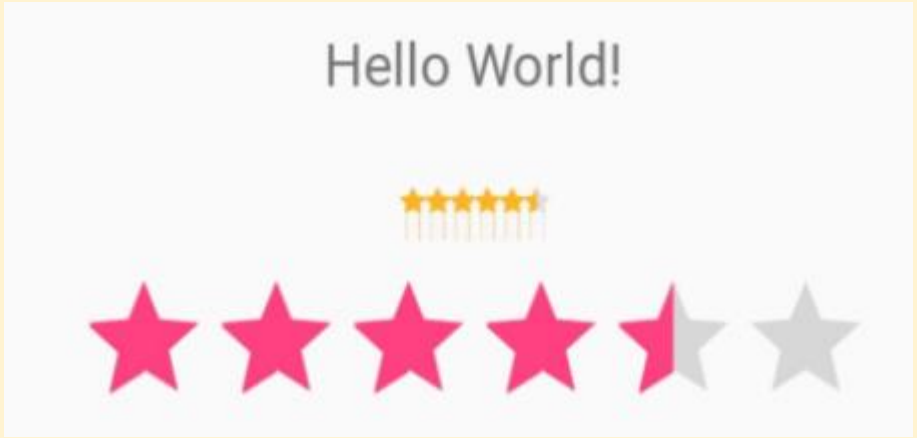
Example


```java
public class MainActivity extends AppCompatActivity {

    Button submitButton;
    SeekBar simpleSeekBar;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // initiate  views
        simpleSeekBar=(SeekBar)findViewById(R.id.simpleSeekBar);
        // perform seek bar change listener event used for getting the progress value
        simpleSeekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
            int progressChangedValue = 0;

            public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
                progressChangedValue = progress;
            }

            public void onStartTrackingTouch(SeekBar seekBar) {
                // TODO Auto-generated method stub
            }

            public void onStopTrackingTouch(SeekBar seekBar) {
                Toast.makeText(MainActivity.this, "Seek bar progress is :" + progressChangedValue,
                        Toast.LENGTH_SHORT).show();
            }
        });

    }
```

Displaying Rating Data with RatingBar

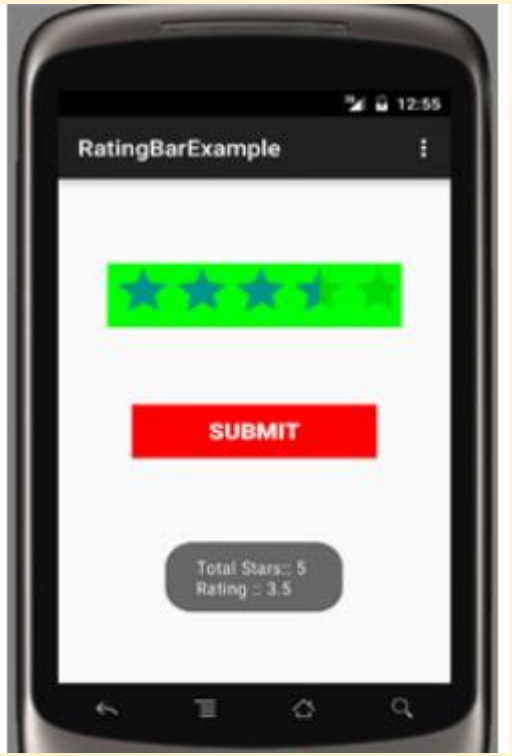
P P SAVANI UNIVERSITY

- Although the SeekBar is useful for allowing a user to set a value, such as the volume, the RatingBar has a more specific purpose: showing ratings or getting a rating from a user.
- By default, this progress bar uses the star paradigm with five stars by default. A user can drag across this horizontal to set a rating



RatingBar Example

P P SAVANI UNIVERSITY

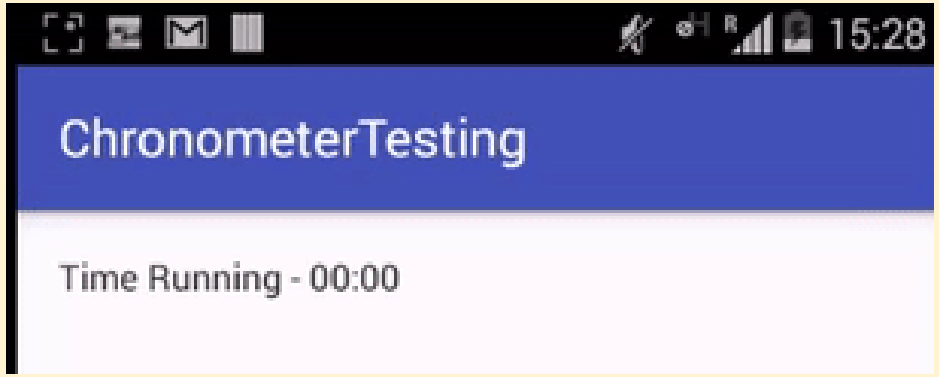



```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // initiate rating bar and a button
        final RatingBar simpleRatingBar = (RatingBar) findViewById(R.id.simpleRatingBar);
        Button submitButton = (Button) findViewById(R.id.submitButton);
        // perform click event on button
        submitButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // get values and then displayed in a toast
                String totalStars = "Total Stars:: " + simpleRatingBar.getNumStars();
                String rating = "Rating :: " + simpleRatingBar.getRating();
                Toast.makeText(getApplicationContext(), totalStars + "\n" + rating, Toast.LENGTH_LONG).sho
            }
        });
    }


}
```

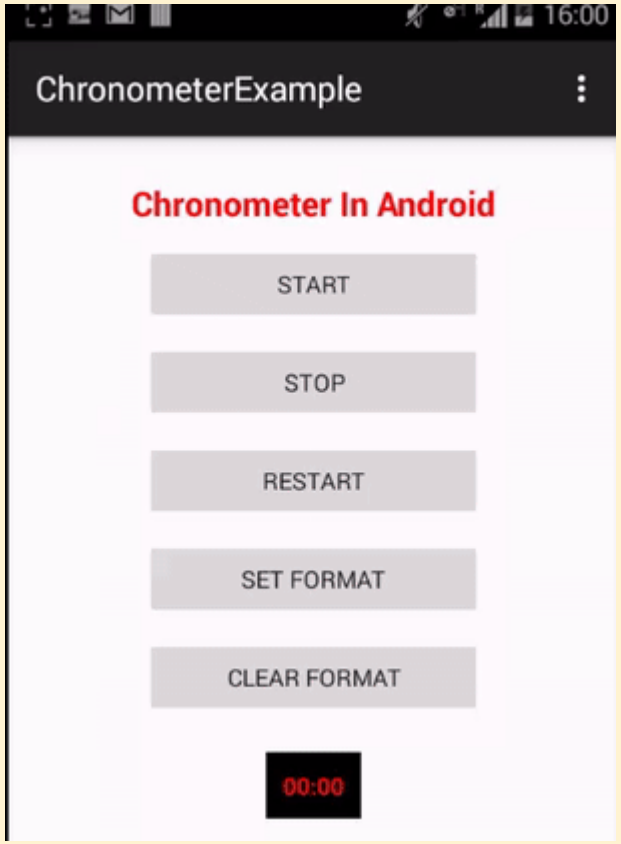
Showing Time Passage with Chronometer

P P SAVANI UNIVERSITY

- In Android, Chronometer is a class that implements a simple timer. Chronometer is a subclass of TextView. This class helps us to add a timer in our app

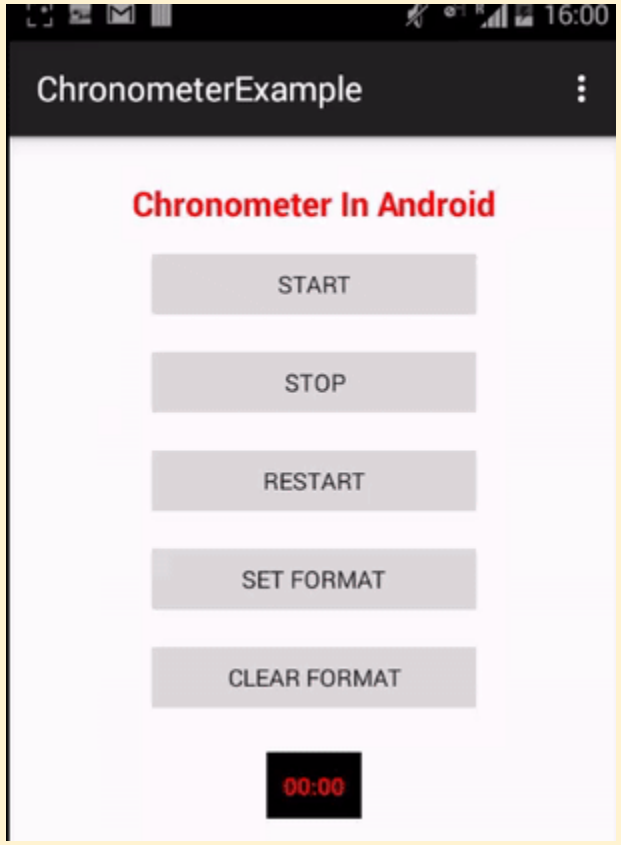


Chronometer Example




```
Chronometer simpleChronometer;
Button start, stop, restart, setFormat, clearFormat;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // initiate views
    simpleChronometer = (Chronometer) findViewById(R.id.simpleChronometer);
    start = (Button) findViewById(R.id.startButton);
    stop = (Button) findViewById(R.id.stopButton);
    restart = (Button) findViewById(R.id.restartButton);
    setFormat = (Button) findViewById(R.id.setFormat);
    clearFormat = (Button) findViewById(R.id.clearFormat);
    // perform click  event on start button to start a chronometer
    start.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub

            simpleChronometer.start();
        }
    });

    // perform click  event on stop button to stop the chronometer
    stop.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub

            simpleChronometer.stop();
        }
    });
```

Chronometer Example

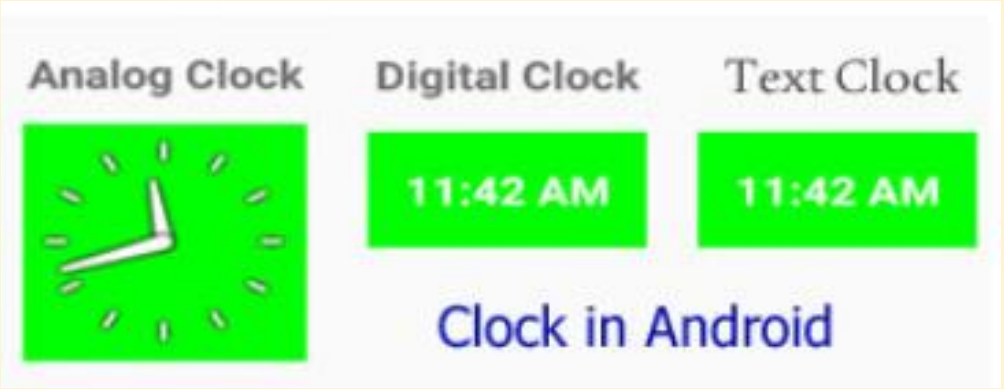



```
    // perform click  event on restart button to set the base time on chronometer
    restart.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub

            simpleChronometer.setBase(SystemClock.elapsedRealtime());
        }
    });

    // perform click  event on set Format button to set the format of chronometer
    setFormat.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub

            simpleChronometer.setFormat("Time (%s)");
        }
    });

    // perform click  event on clear button to clear the current format of chronmeter as you set thro
    clearFormat.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub

            simpleChronometer.setFormat(null);
        }
    });
}
```

Displaying the Time

P P SAVANI UNIVERSITY

- **Displaying the time in an application is often not necessary because Android devices have a status bar to display the current time. However, there are two clock controls available to display this information: the DigitalClock and AnalogClock controls.**



Analog and Digital Clock

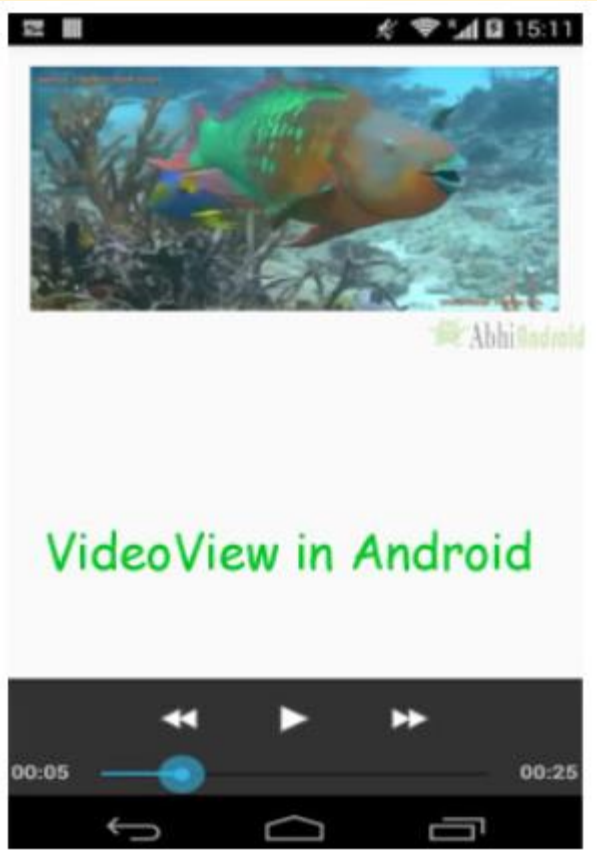
P P SAVANI UNIVERSITY

```
<AnalogClock

    android:layout_marginTop="20dp"
    android:layout_marginLeft="120dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
 <DigitalClock

     android:layout_marginLeft="140dp"
     android:textSize="25dp"
     android:layout_marginTop="300dp"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content" />
```

Playing Video with VideoView

P P SAVANI UNIVERSITY

- In Android, VideoView is used to display a video file.
- It can load images from various sources (such as content providers or resources) taking care of computing its measurement from the video so that it can be used for any layout manager, providing display options such as scaling and tinting.



MediaController In VideoView

P P SAVANI UNIVERSITY

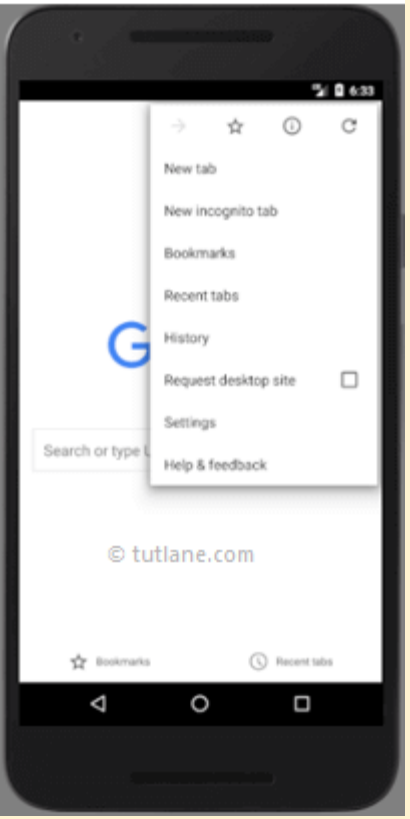
- MediaController is a class which is used to provide the controls for the video playback. If a video is simply played using the VideoView class then the user will not be given any control over the playback of the video which will run until the end of the video is reached.
- This issue can be addressed by attaching an instance of the MediaController class to the VideoView instance.
- The Media Controller will then provide a set of controls allowing the user to manage the playback (such as seeking backwards/forwards and pausing in the video timeline).

Example


```
// Find your VideoView in your video_main.xml layout
simpleVideoView = (VideoView) findViewById(R.id.simpleVideoView);

if (mediaControls == null) {
    // create an object of media controller class
    mediaControls = new MediaController(MainActivity.this);
    mediaControls.setAnchorView(simpleVideoView);
}
// set the media controller for video view
simpleVideoView.setMediaController(mediaControls);
// set the uri for the video view
simpleVideoView.setVideoURI(Uri.parse("android.resource://" + getPackageName() + "/" + R.raw.fishvideo));
// start a video
simpleVideoView.start();

// implement on completion listener on video view
simpleVideoView.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
    @Override
    public void onCompletion(MediaPlayer mp) {
        Toast.makeText(getApplicationContext(), "Thank You...!!!", Toast.LENGTH_LONG).show(); // display
    }
});
simpleVideoView.setOnErrorListener(new MediaPlayer.OnErrorListener() {
    @Override
    public boolean onError(MediaPlayer mp, int what, int extra) {
        Toast.makeText(getApplicationContext(), "Oops An Error Occur While Playing Video...!!!", Toast.LE
        return false;
    }
});
```

Encouraging Action(Menus)

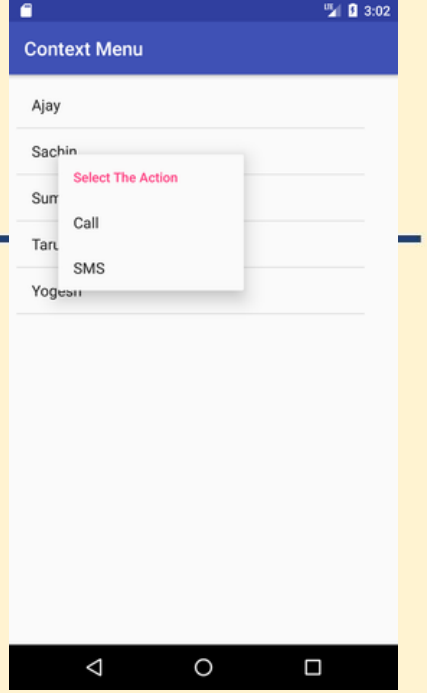
P P SAVANI UNIVERSITY

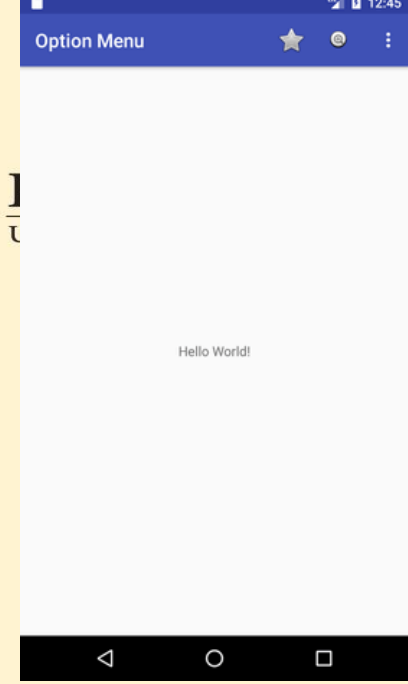


- To prompt users to take a particular action.
- Menus were from API Level 1 to API Level 11
- From API Level 11,menus have been replaced with the ActionBar that represents actions to users.
- API Level 21 added a generalized version of an ActionBar called a ToolBar.

Types of Menus

- **Options Menu:**
- **This menu is usually found at the top of your application and in it, you should place actions that affect the application as a whole. These could be the application's settings or a search box.**
- **Contextual Menu:**
- **This menu appears when a user performs a long click on one of your UI elements.**
- **Popup Menu:**
- **Android Popup Menu displays the menu below the anchor text if space is available otherwise above the anchor text. It disappears if you click outside the popup menu.**





ToolBars

P P SAVANI UNIVERSITY

- The toolbar was introduced from Android Lollipop when Material Design came into existence.
- Before Toolbar we were using ActionBar, the position of ActionBar was fix at the top of the screen.
- But in case of Toolbar, it is more flexible; you can place Toolbar anywhere in the Activity. The toolbar is useful for displaying App Icon, Title, Navigation Menu, etc.

Toolbars

- The key functions of the app bar are as follows:
- A dedicated space for giving your app an identity and indicating the user's location in the app.
- Access to important actions in a predictable way, such as search.
- Support for navigation and view switching (with tabs or drop-down lists).

Using Toolbar as ActionBar

P P SAVANI UNIVERSITY

To use Toolbar as an ActionBar, first ensure the AndroidX support library is added to your application `build.gradle` (Module:app) file:

```
<resources>
  <!-- Base application theme. -->
  <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
  </style>
</resources>
```

Second, let's disable the theme-provided ActionBar. The easiest way is to have your theme extend from `Theme.AppCompat.NoActionBar` (or the light variant) within the `res/values/styles.xml` file:

```
<resources>
  <!-- Base application theme. -->
  <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
  </style>
</resources>
```

Using Toolbar as ActionBar

P P SAVANI UNIVERSITY

Now you need to add a `Toolbar` to your Activity layout file

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    android:orientation="vertical">

    <androidx.appcompat.widget.Toolbar
      android:id="@+id/toolbar"
      android:minHeight="?attr/actionBarSize"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      app:titleTextColor="@android:color/white"
      android:background="?attr/colorPrimary">
    </androidx.appcompat.widget.Toolbar>

    <!-- Layout for content is here. This can be a RelativeLayout  -->

</LinearLayout>
```

MyActivity.java for Toolbar

P P SAVANI UNIVERSITY

```java
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

public class MyActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my);

        // Find the toolbar view inside the activity layout
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        // Sets the Toolbar to act as the ActionBar for this Activity window.
        // Make sure the toolbar exists in the activity and is not null
        setSupportActionBar(toolbar);
    }

    // Menu icons are inflated just as they were with actionbar
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
}
```

`res/menu/menu_main.xml` which is inflated above in `onCreateOptionsMenu`

P P SAVANI UNIVERSITY

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/miCompose"
        android:icon="@drawable/ic_compose"
        app:showAsAction="ifRoom"
        android:title="Compose">
    </item>
    <item
        android:id="@+id/miProfile"
        android:icon="@drawable/ic_profile"
        app:showAsAction="ifRoom|withText"
        android:title="Profile">
    </item>
</menu>
```

Reusing the Toolbar

P P SAVANI UNIVERSITY

In many apps, the same toolbar can be used across multiple activities or in alternative layout resources for the same activity. In order to easily reuse the toolbar, we can leverage the layout include tag as follows. First, define your toolbar in a layout file in res/layout/toolbar_main.xml:

```
<androidx.appcompat.widget.Toolbar
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="?attr/colorPrimary"/>
```

`<include />` tag to load the toolbar into our activity layout XML:

P P SAVANI UNIVERSITY

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    android:orientation="vertical">

    <!-- Load the toolbar here -->
    <include
        layout="@layout/toolbar_main"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <!-- Rest of content for the activity -->

</LinearLayout>
```

access the `Toolbar` by the `include` id instead:

```
public class MyActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my);

        // Find the toolbar view inside the activity layout
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar_main);
        // Sets the Toolbar to act as the ActionBar for this Activity window.
        // Make sure the toolbar exists in the activity and is not null
        setSupportActionBar(toolbar);
    }
```

Styling the Toolbar

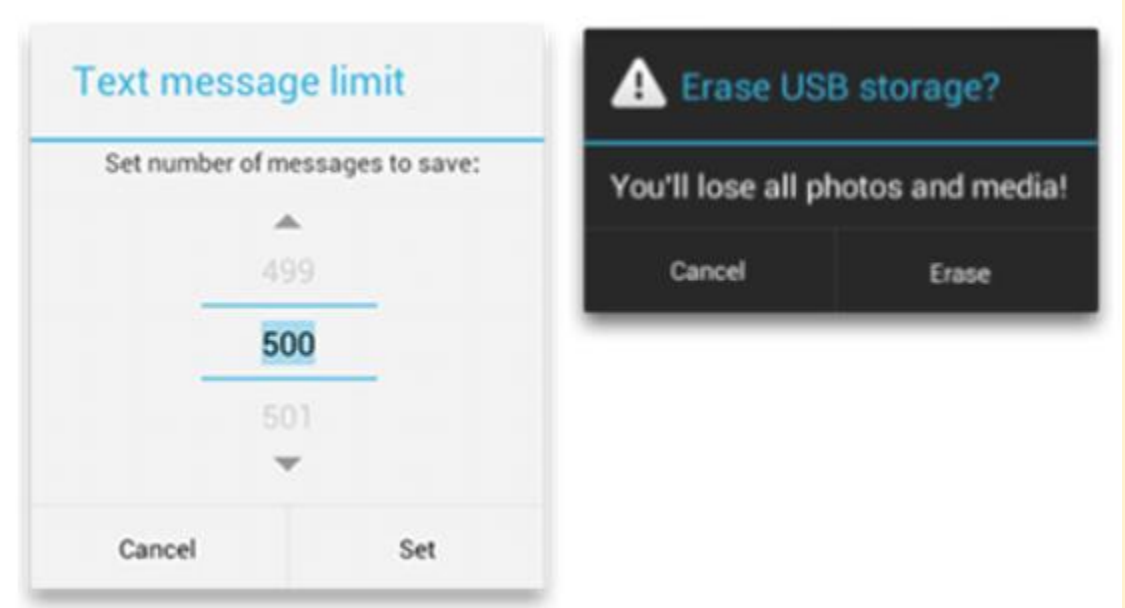
create the custom styles in `res/values/styles.xml` with:


```xml
<!-- Base application theme. -->
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
  <!-- Customize your theme here. -->
  <item name="colorPrimary">@color/colorPrimary</item>
  <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
  <item name="colorAccent">@color/colorAccent</item>
</style>

<style name="ToolbarTheme" parent="@style/ThemeOverlay.AppCompat.Dark.ActionBar">
  <!-- android:textColorPrimary is the color of the title text in the Toolbar  -->
  <item name="android:textColorPrimary">@android:color/holo_blue_light</item>
  <!-- actionMenuTextColor is the color of the text of action (menu) items  -->
  <item name="actionMenuTextColor">@android:color/holo_green_light</item>
  <!-- Tints the input fields like checkboxes and text fields -->
  <item name="colorAccent">@color/cursorAccent</item>
  <!-- Applies to views in their normal state. -->
  <item name="colorControlNormal">@color/controlNormal</item>
  <!-- Applies to views in their activated state (i.e checked or switches) -->
  <item name="colorControlActivated">@color/controlActivated</item>
  <!-- Applied to framework control highlights (i.e ripples or list selectors) -->
  <item name="colorControlHighlight">@color/controlActivated</item>

  <!-- Enable these below if you want clicking icons to trigger a ripple effect -->
  <!--
  <item name="selectableItemBackground">?android:selectableItemBackground</item>
  <item name="selectableItemBackgroundBorderless">?android:selectableItemBackground</ite
  -->
</style>

<!-- This configures the styles for the title within the Toolbar  -->
<style name="Toolbar.TitleText" parent="TextAppearance.Widget.AppCompat.Toolbar.Title">
    <item name="android:textSize">21sp</item>
    <item name="android:textStyle">italic</item>
</style>
```

Dialogs

P P SAVANI UNIVERSITY

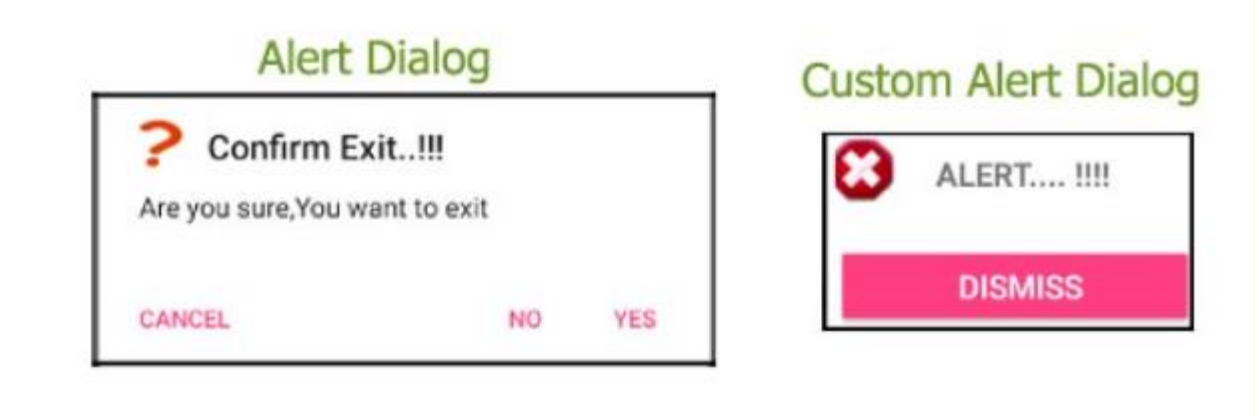
- A dialog is a small window that prompts the user to make a decision or enter additional information. A dialog does not fill the screen and is normally used for modal events that require users to take an action before they can proceed.



AlertDialog

P P SAVANI UNIVERSITY

- **Alert Dialog in an android UI prompts a small window to make decision on mobile screen. Sometimes before making a decision it is required to give an alert to the user without moving to next activity. To solve this problem alert dialog came into practise. For example you have seen this type of alert when you try to exit the App and App ask you to confirm exiting.**



Example

P P SAVANI UNIVERSITY

```
package com.example.alertdialogexample;

import android.content.DialogInterface;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void exit(View view){
        AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
        // Setting Alert Dialog Title
        alertDialogBuilder.setTitle("Confirm Exit..!!!");
        // Icon Of Alert Dialog
        alertDialogBuilder.setIcon(R.drawable.question);
        // Setting Alert Dialog Message
        alertDialogBuilder.setMessage("Are you sure,You want to exit");
        alertDialogBuilder.setCancelable(false);

                alertDialogBuilder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {

                        @Override
                        public void onClick(DialogInterface arg0, int arg1) {
                            finish();
                        }
                    });
```

Example

P P SAVANI UNIVERSITY

```
                alertDialogBuilder.setNegativeButton("No", new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                Toast.makeText(MainActivity.this,"You clicked over No",Toast.LENGTH_SHORT
                            }
                        });
                alertDialogBuilder.setNeutralButton("Cancel", new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                Toast.makeText(getApplicationContext(),"You clicked on Cancel",Toast.LENG
                            }
                         });

        AlertDialog alertDialog = alertDialogBuilder.create();
        alertDialog.show();
    }
}
```

ProgressDialog

P P SAVANI UNIVERSITY

- Android Progress Dialog is a UI which shows the progress of a task like you want user to wait until the previous lined up task is completed and for that purpose you can use progress dialog.
  The best example is you see
  when downloading or uploading any file.

ProgressDialog


```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    b1 = (Button) findViewById(R.id.button);
    b2 = (Button) findViewById(R.id.button2);

    b1.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            progressDialog = new ProgressDialog(MainActivity.this);
            progressDialog.setMessage("Loading..."); // Setting Message
            progressDialog.setTitle("ProgressDialog"); // Setting Title
            progressDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
            progressDialog.show(); // Display Progress Dialog
            progressDialog.setCancelable(false);
            new Thread(new Runnable() {
                public void run() {
                    try {
                        Thread.sleep(10000);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                    progressDialog.dismiss();
                }
            }).start();
        }
    });
```

RingProgress

ProgressDialog

P P SAVANI UNIVERSITY

```
b2.setOnClickListener(new View.OnClickListener() {
    Handler handle = new Handler() {
        public void handleMessage(Message msg) {
            super.handleMessage(msg);
            progressDialog.incrementProgressBy(2); // Incremented By Value 2
        }
    };

    @Override
    public void onClick(View v) {
        progressDialog = new ProgressDialog(MainActivity.this);
        progressDialog.setMax(100); // Progress Dialog Max Value
        progressDialog.setMessage("Loading..."); // Setting Message
        progressDialog.setTitle("ProgressDialog"); // Setting Title
        progressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL); // Progress Dialog
        progressDialog.show(); // Display Progress Dialog
        progressDialog.setCancelable(false);
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    while (progressDialog.getProgress() <= progressDialog.getMax()) {
                        Thread.sleep(200);
                        handle.sendMessage(handle.obtainMessage());
                        if (progressDialog.getProgress() == progressDialog.getMax()) {
                            progressDialog.dismiss();
                        }
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }
});
```

Styles and Themes

P P SAVANI UNIVERSITY

- Styles and themes on Android allow you to separate the details of your app design from the UI structure and behavior, similar to stylesheets in web design.
- A style is a collection of attributes that specify the appearance for a single View. A style can specify attributes such as font color, font size, background color, and much more.
- A theme is a collection of attributes that's applied to an entire app, activity, or view hierarchy—not just an individual view. When you apply a theme, every view in the app or activity applies each of the theme's attributes that it supports. Themes can also apply styles to non-view elements, such as the status bar and window background.
- Styles and themes are declared in a style resource file in res/values/, usually named styles.xml.

Styles and Themes

P P SAVANI UNIVERSITY

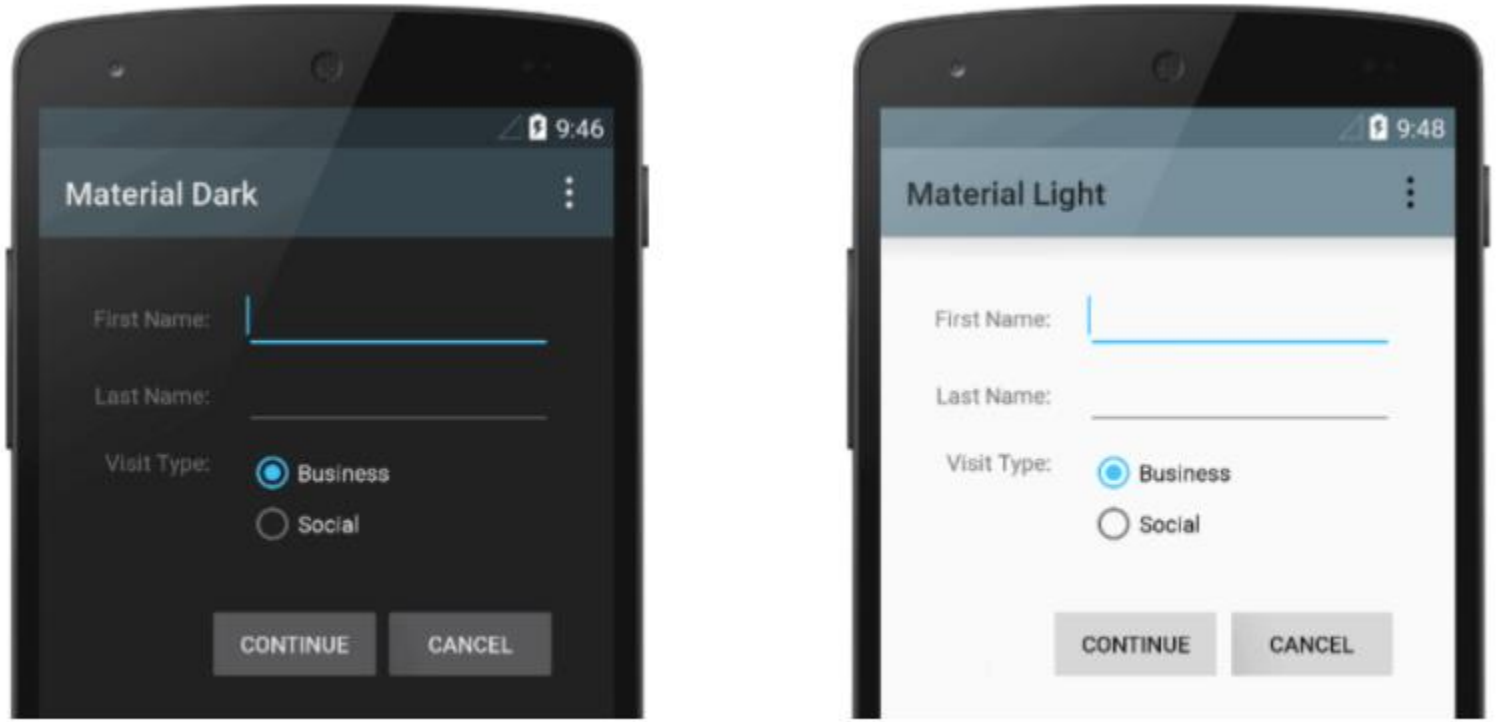


Figure 1. Two themes applied to the same activity: `Theme.AppCompat` (left) and `Theme.AppCompat.Light` (right)

Create and apply a style

To create a new style or theme, open your project's res/values/styles.xml file

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="GreenText" parent="TextAppearance.AppCompat">
        <item name="android:textColor">#00FF00</item>
    </style>
</resources>
```

You can apply the style to a view as follows:

```xml
<TextView
    style="@style/GreenText"
    ... />
```

Apply a style as a theme

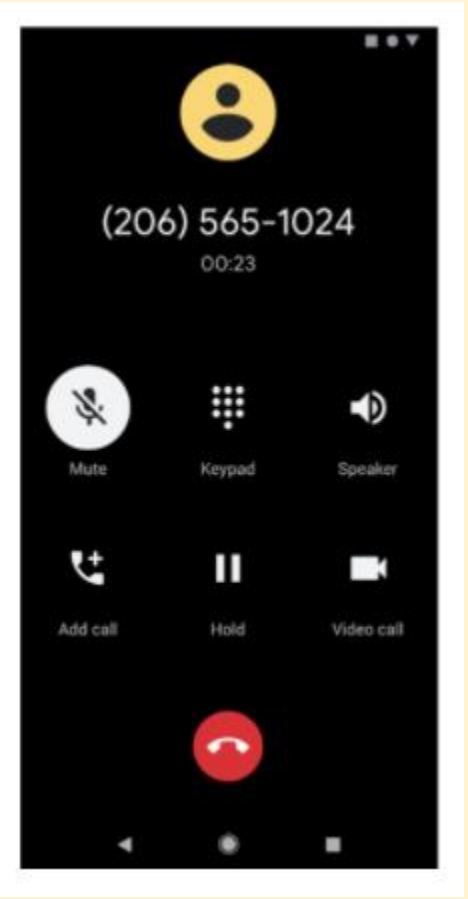
P P SAVANI UNIVERSITY

- You can create a theme the same way you create styles. The difference is how you apply it: instead of applying a style with the style attribute on a view, you apply a theme with the android:theme attribute on either the <application> tag or an <activity> tag in the AndroidManifest.xml file.

```
<manifest ... >
    <application android:theme="@style/Theme.AppCompat" ... >
    </application>
</manifest>
```

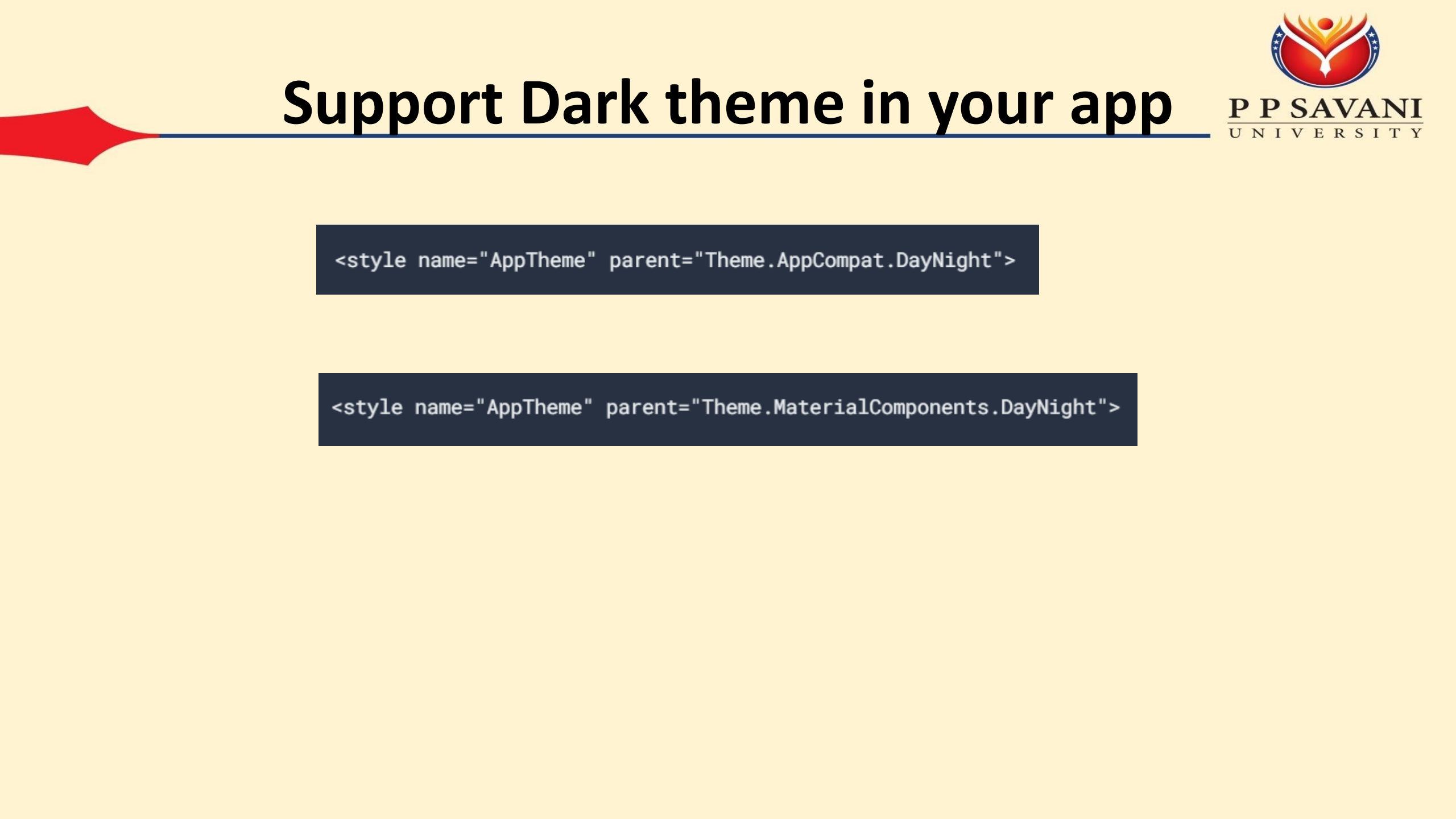
```
<manifest ... >
    <application ... >
        <activity android:theme="@style/Theme.AppCompat.Light" ... >
        </activity>
    </application>
</manifest>
```

Dark theme



- Dark theme is available in Android 10 (API level 29) and higher. It has many benefits:
- Can reduce power usage by a significant amount (depending on the device's screen technology).
- Improves visibility for users with low vision and those who are sensitive to bright light.
- Makes it easier for anyone to use a device in a low-light environment.

Support Dark theme in your app




```
<style name="AppTheme" parent="Theme.AppCompat.DayNight">
```

```
<style name="AppTheme" parent="Theme.MaterialComponents.DayNight">
```

P P SAVANI
UNIVERSITY

THANK YOU

P P SAVANI
UNIVERSITY

Module 4
Designing User Interfaces with Layouts

Creating User Interfaces in Android

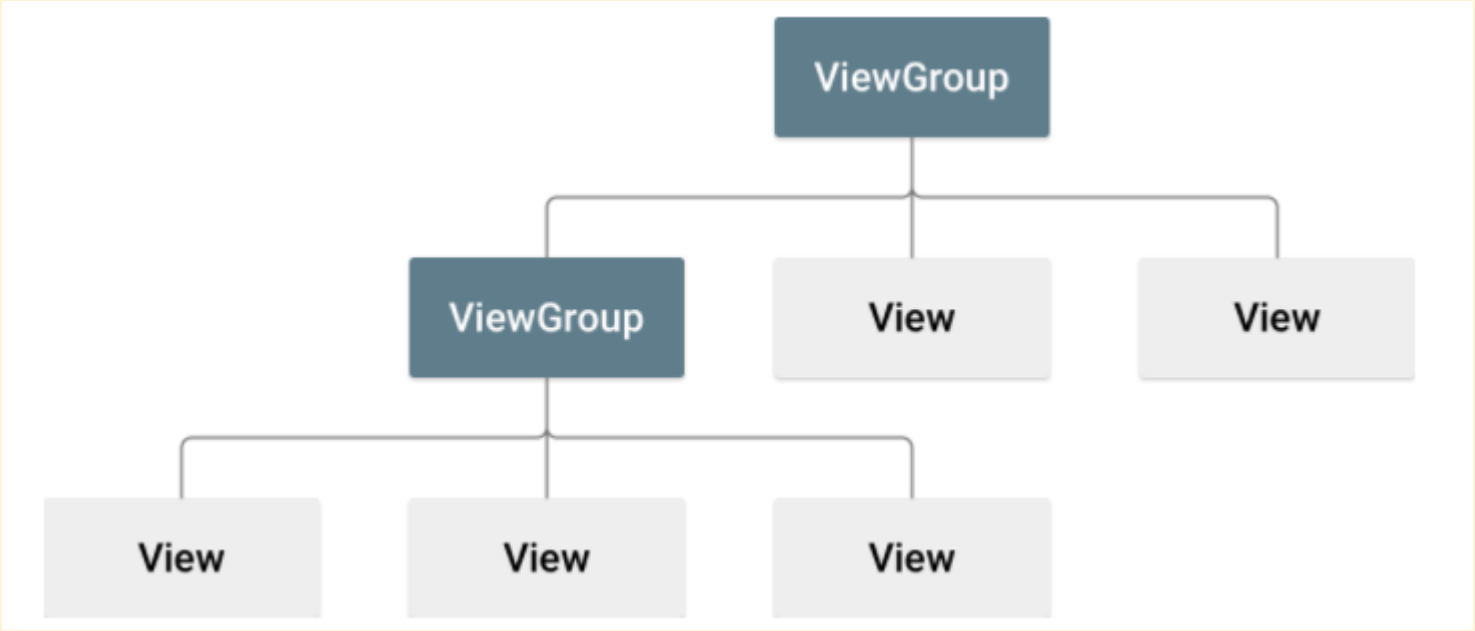
P P SAVANI UNIVERSITY

- Application user interfaces can be simple or complex, involving many different screens or only a few.
- A layout defines the structure for a user interface in your app, such as in an activity.
- All elements in the layout are built using a hierarchy of View and ViewGroup objects.

View vs ViewGroup

P P SAVANI UNIVERSITY

- A View usually draws something the user can see and interact with. Whereas a ViewGroup is an invisible container that defines the layout structure for View and other ViewGroup objects, as shown in figure



View vs ViewGroup

P P SAVANI UNIVERSITY

- The View objects are usually called "widgets" and can be one of many subclasses, such as Button or TextView.
- The ViewGroup objects are usually called "layouts" can be one of many types that provide a different layout structure, such as LinearLayout or ConstraintLayout .

Layout

P P SAVANI UNIVERSITY

- You can declare a layout in two ways:

- Declare UI elements in XML. Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses, such as those for widgets and layouts.
- You can also use Android Studio's Layout Editor to build your XML layout using a drag-and-drop interface.
- Instantiate layout elements at runtime. Your app can create View and ViewGroup objects (and manipulate their properties) programmatically.

- Declaring your UI in XML allows you to separate the presentation of your app from the code that controls its behavior. Using XML files also makes it easy to provide different layouts for different screen sizes and orientations

Write the XML and Load the XML Resource

P P SAVANI UNIVERSITY

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

```java
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_layout);
}
```

Attributes

P P SAVANI UNIVERSITY

- Every View and ViewGroup object supports their own variety of XML attributes.
- ID: Any View object may have an integer ID associated with it, to uniquely identify the View within the tree

```
<Button android:id="@+id/my_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/my_button_text"/>
```

```
Button myButton = (Button) findViewById(R.id.my_button);
```

Defining IDs for view objects is important when creating a RelativeLayout. In a relative layout, sibling views can define their layout relative to another sibling view, which is referenced by the unique ID.

- *wrap_content* tells your view to size itself to the dimensions required by its content.
- *match_parent* tells your view to become as big as its parent view group will allow.

Common Layouts

P P SAVANI UNIVERSITY

Linear Layout

A layout that organizes its children into a single horizontal or vertical row. It creates a scrollbar if the length of the window exceeds the length of the screen.

Relative Layout

Enables you to specify the location of child objects relative to each other (child A to the left of child B) or to the parent (aligned to the top of the parent).

Web View

Displays web pages.

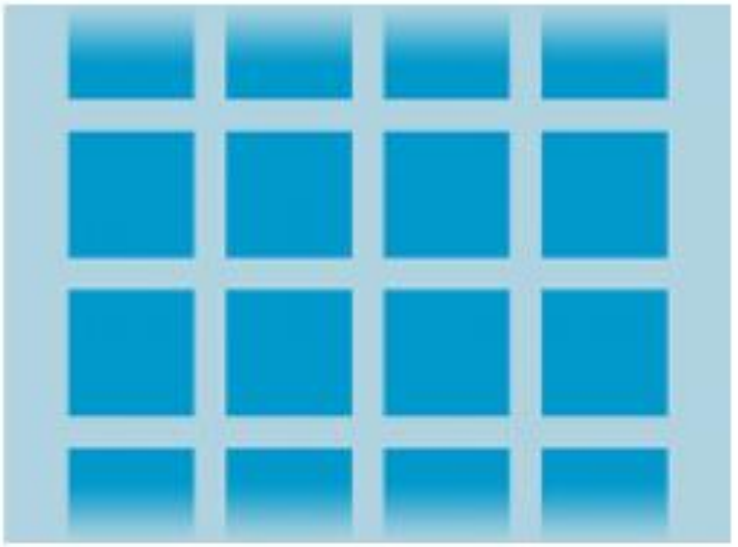
Building Layouts with an Adapter

P P SAVANI UNIVERSITY

List View

Displays a scrolling single column list.

Grid View



Displays a scrolling grid of columns and rows.

Filling an adapter view with data

P P SAVANI UNIVERSITY

- You can populate an AdapterView such as ListView or GridView by binding the AdapterView instance to an Adapter, which retrieves data from an external source and creates a View that represents each data entry.
- Android provides several subclasses of Adapter that are useful for retrieving different kinds of data and building views for an AdapterView. The two most common adapter is ArrayAdapter and SimpleCursorAdapter

ArrayAdapter

P P SAVANI UNIVERSITY

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, myStringArray);
```

```
ListView listView = (ListView) findViewById(R.id.listview);
listView.setAdapter(adapter);
```

```
// Create a message handling object as an anonymous class.
private OnItemClickListener messageClickedHandler = new OnItemClickListener() {
    public void onItemClick(AdapterView parent, View v, int position, long id) {
        // Do something in response to the click
    }
};

listView.setOnItemClickListener(messageClickedHandler);
```

SimpleCursorAdapter

P P SAVANI UNIVERSITY

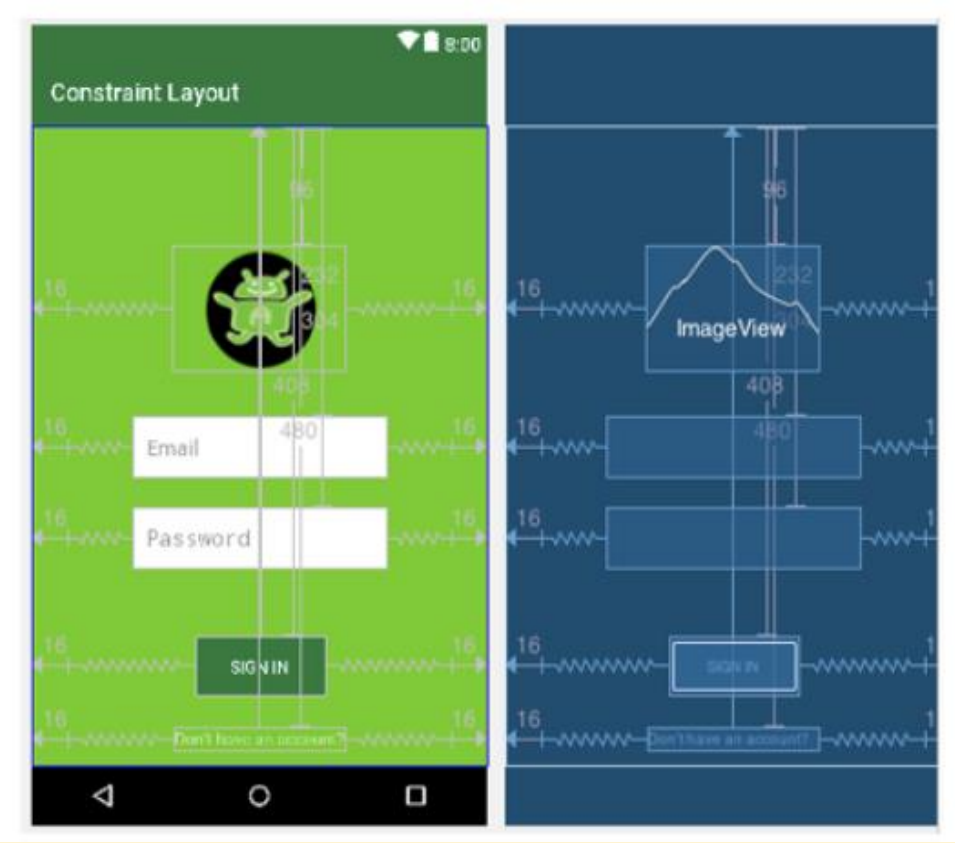
```
String[] fromColumns = {ContactsContract.Data.DISPLAY_NAME,
                        ContactsContract.CommonDataKinds.Phone.NUMBER};
int[] toViews = {R.id.display_name, R.id.phone_number};
```

```
SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
        R.layout.person_name_and_number, cursor, fromColumns, toViews, 0);
ListView listView = getListView();
listView.setAdapter(adapter);
```

Build a Responsive UI with ConstraintLayout

P P SAVANI UNIVERSITY

- Constraint Layout is a ViewGroup (i.e. a view that holds other views) which allows you to create large and complex layouts with a flat view hierarchy, and also allows you to position and size widgets in a very flexible way.
- It was created to help reduce the nesting of views and also improve the performance of layout files.



ConstraintLayout

P P SAVANI UNIVERSITY

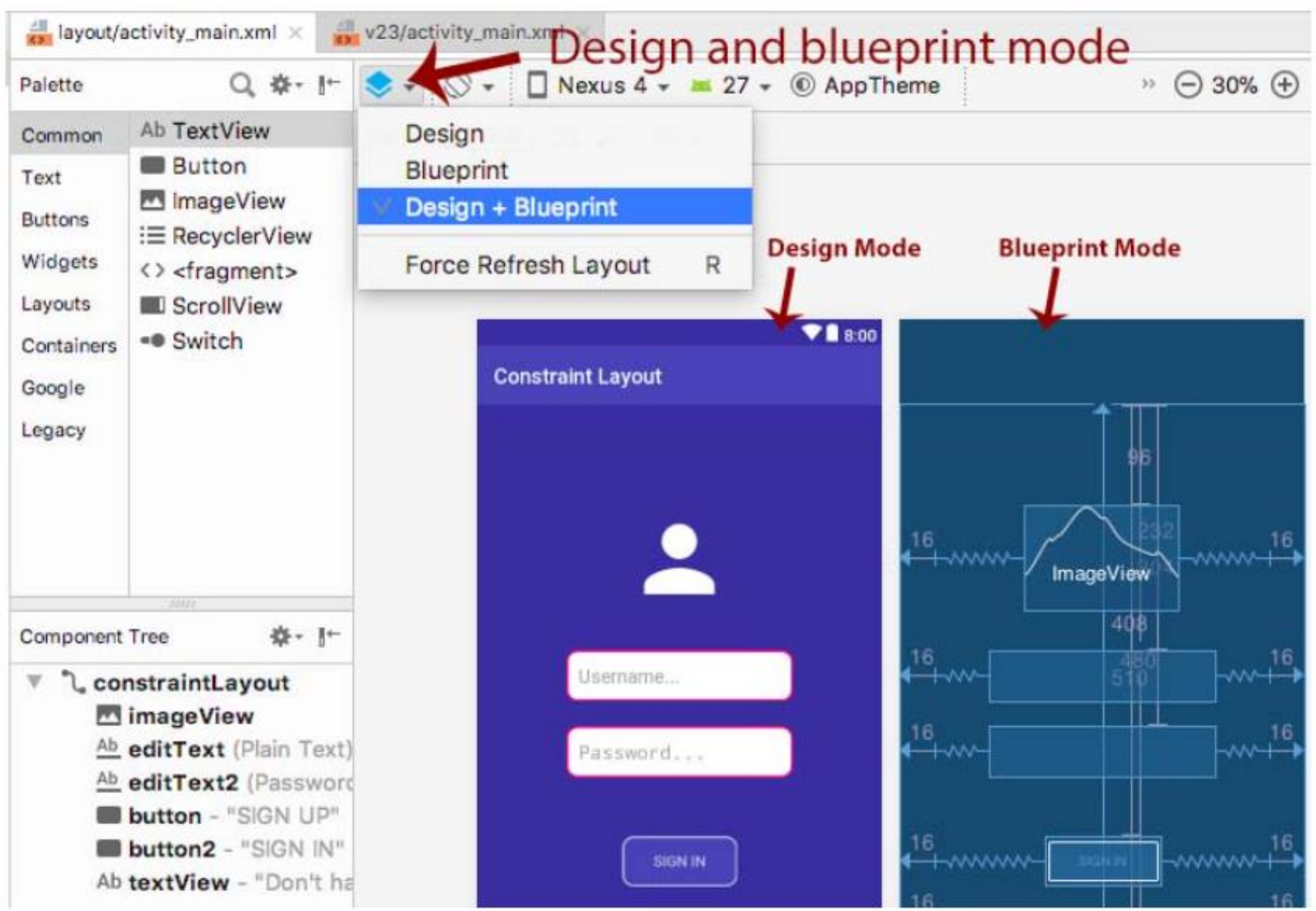
- ConstraintLayout is very similar to RelativeLayout in such a way because, views are laid out according to relationships between sibling views and the parent layout yet it's a lot more flexible and works better with the Layout Editor of the Android Studio's.
- It was released at Google I/O 2016.
- Since it came into existence (i.e. as at Android studio 2.3), it has become a wildly used viewgroup and supports Android 2.3 or higher.

Advantages Of Constraint Layout Over Other Layouts

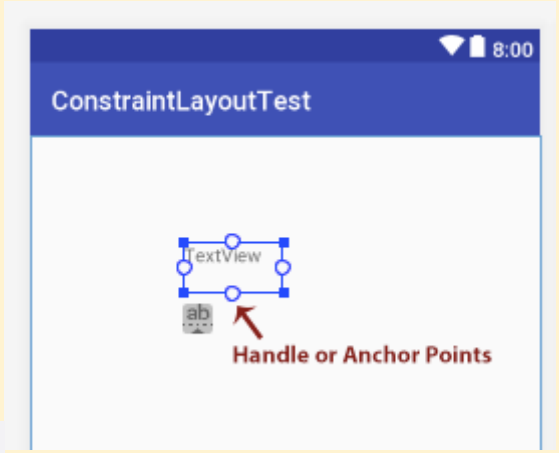
1. One great advantage of the constraintlayout is that you can perform animations on your ConstraintLayout views with very little code.
2. You can build your complete layout with simple drag-and-drop on the Android Studio design editor.
3. You can control what happens to a group of widgets through a single line of code.
4. Constraint Layout improve performance over other layout

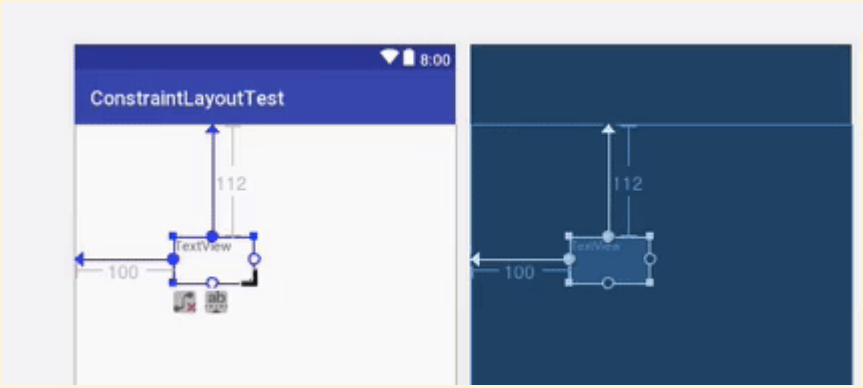
Design Or Blueprint Mode In Android Studio

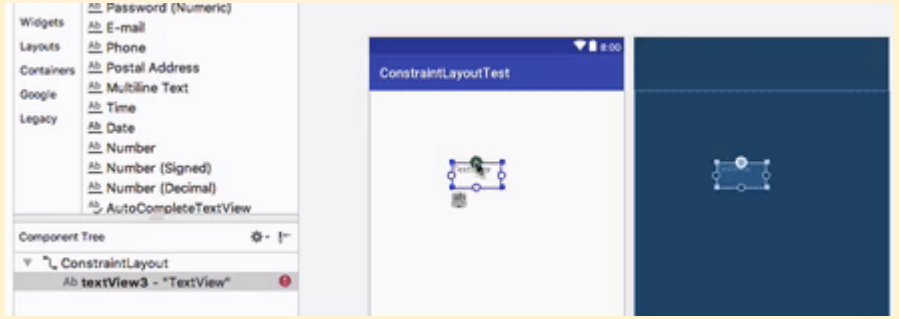
P P SAVANI UNIVERSITY

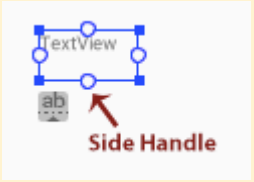


Resize Handle



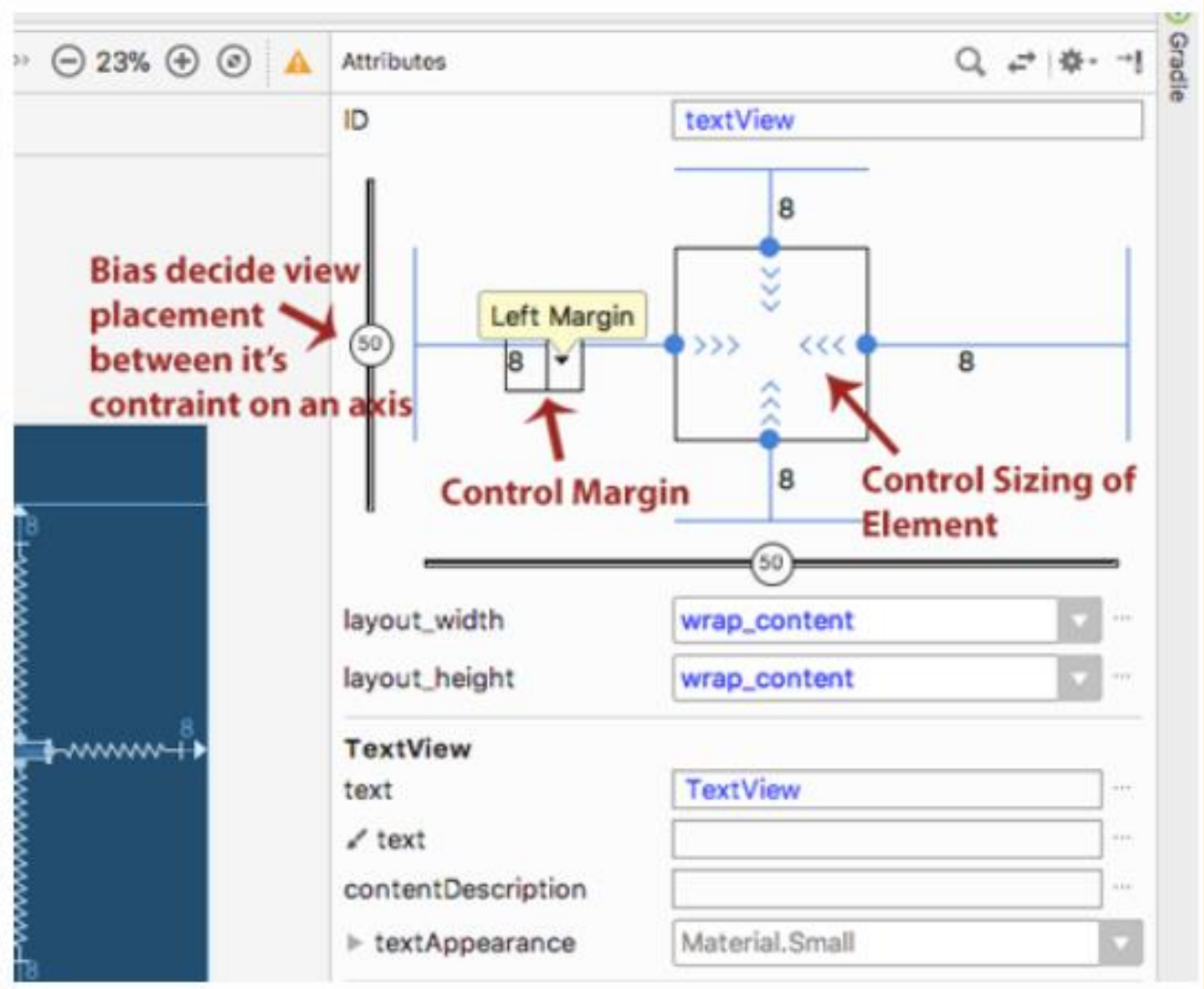






Attribute Window For ConstraintLayout

P P SAVANI UNIVERSITY



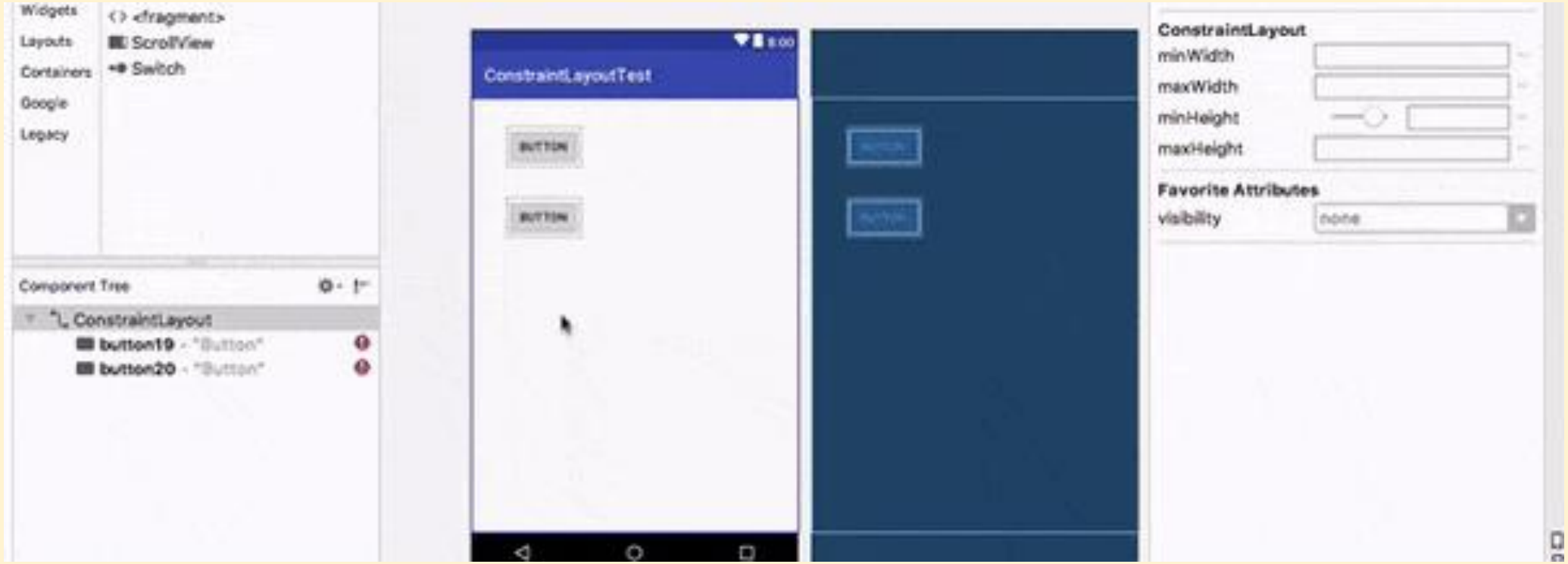
Relative Positioning In Constraint Layout

P P SAVANI UNIVERSITY

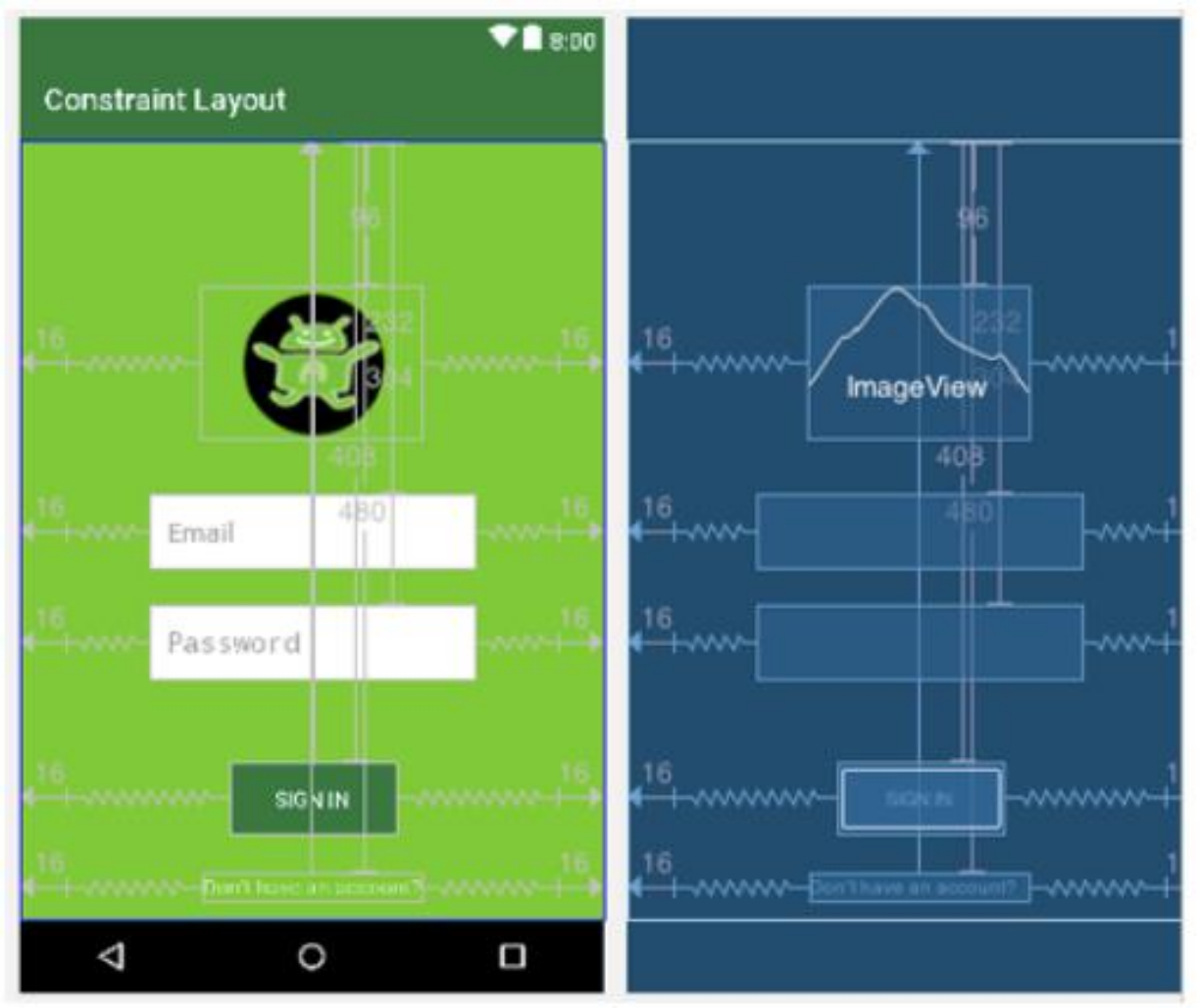
- Relative Positioning is the most important type of Constraint Layout and considered as the basic block building in it.
- The different constraint option it offers works in relation/relative to position of one another.
- Those relative positioning works only in vertical and horizontal axis only.
- layout_constraintLeft_toLeftOf : the left border of the element is positioned relative to the left border of another element
- layout_constraintLeft_toRightOf : the left border of the element is positioned relative to the right border of another element
- layout_constraintRight_toLeftOf: the right border of the element is positioned relative to the left border of another element
- layout_constraintRight_toRightOf: the right border of the element is positioned relative to the right border of another element.

Group In ConstraintLayout

P P SAVANI UNIVERSITY



Constraint Layout Example in Android Studio



Important Points About Constraint Layout

- **To use ConstraintLayout,**

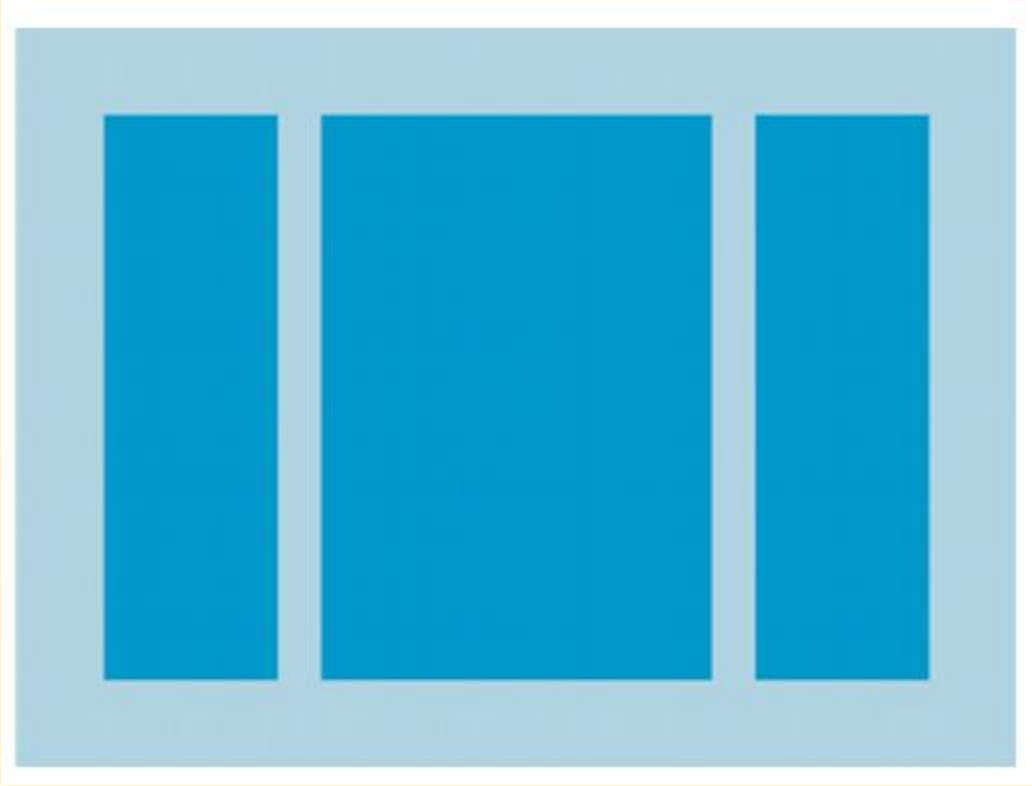
1. **Your Android Studio version must be 2.3 or higher.**
2. **You must first include the ConstraintLayout's support library.**
3. **You must have at least one vertical and one horizontal constraints.**

- **In conclusion, ConstraintLayout is a faster, better and much more efficient choice to designing large and aesthetic layouts in your Android UI.**

Linear Layout

P P SAVANI UNIVERSITY

- **LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally. You can specify the layout direction with the android:orientation attribute.**



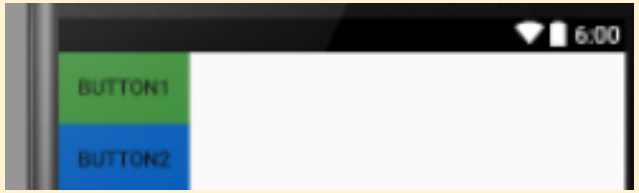
Types Of Linear Layout Orientation

P P SAVANI UNIVERSITY

- **There are two types of linear layout orientation:**

1. **Vertical**

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"> <!-- Vertical Orientation set -->

    <!-- Child Views(In this case 2 Button) are here -->


    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button1"
        android:id="@+id/button"
        android:background="#358a32" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button2"
        android:id="@+id/button2"
        android:background="#0058b6" />


</LinearLayout>
```

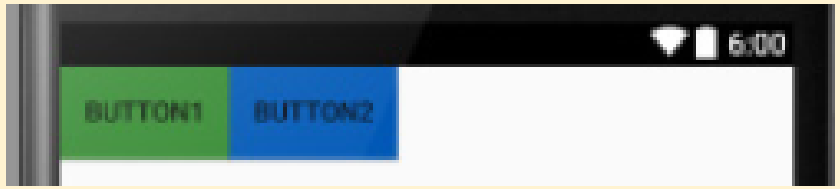


Types Of Linear Layout Orientation

P P SAVANI UNIVERSITY

2. Horizontal:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"> <!-- Horizontal Orientation set -->

    <!-- Child Views(In this case 2 Button) are here -->


    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button1"
        android:id="@+id/button"
        android:background="#358a32" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button2"
        android:id="@+id/button2"
        android:background="#0058b6" />


</LinearLayout>
```

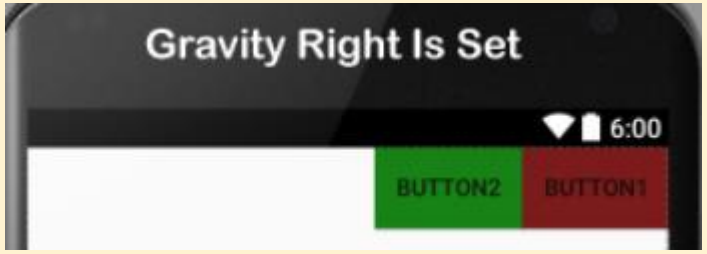


Main Attributes In Linear Layout

P P SAVANI UNIVERSITY

1. **Orientation**
2. **gravity: The gravity attribute is an optional attribute which is used to control the alignment of the layout like left, right, center, top, bottom etc.**

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="right"
    android:orientation="horizontal">
```

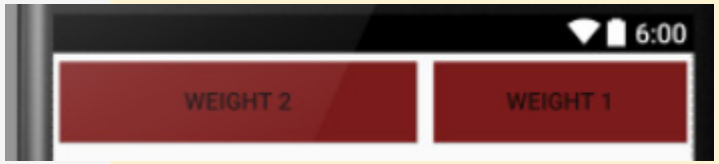


Main Attributes In Linear Layout

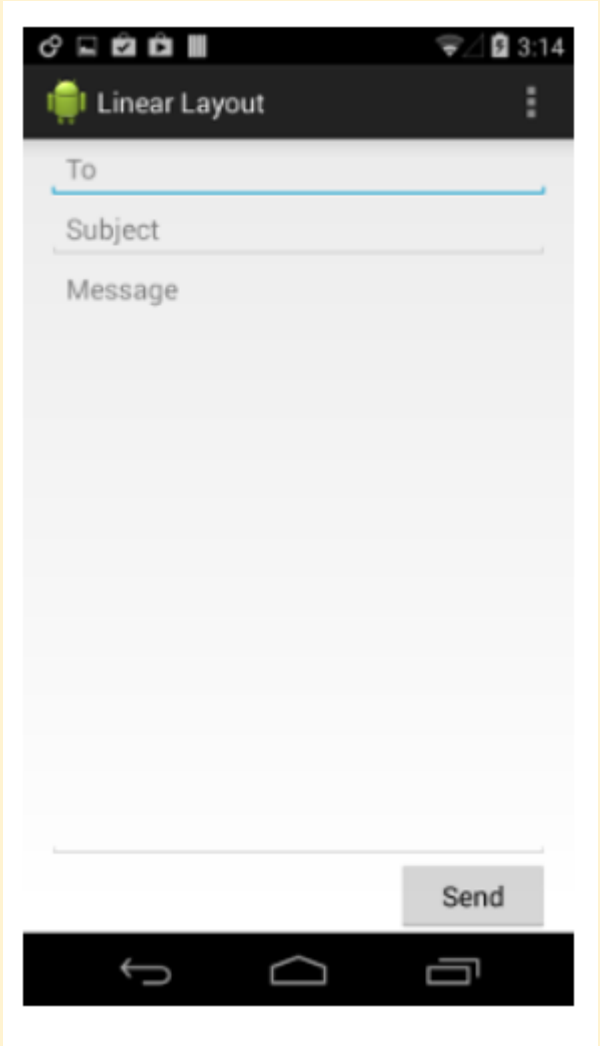
P P SAVANI UNIVERSITY

3. layout_weight: The layout weight attribute specify each child control's relative importance within the parent linear layout.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">


<!--Add Child View Here--->

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Weight 2"
        android:background="#761212"
        android:layout_margin="5dp"
        android:id="@+id/button"
        android:layout_weight="2" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#761212"
        android:layout_margin="5dp"
        android:layout_weight="1"
        android:text="Weight 1" />
</LinearLayout>
```

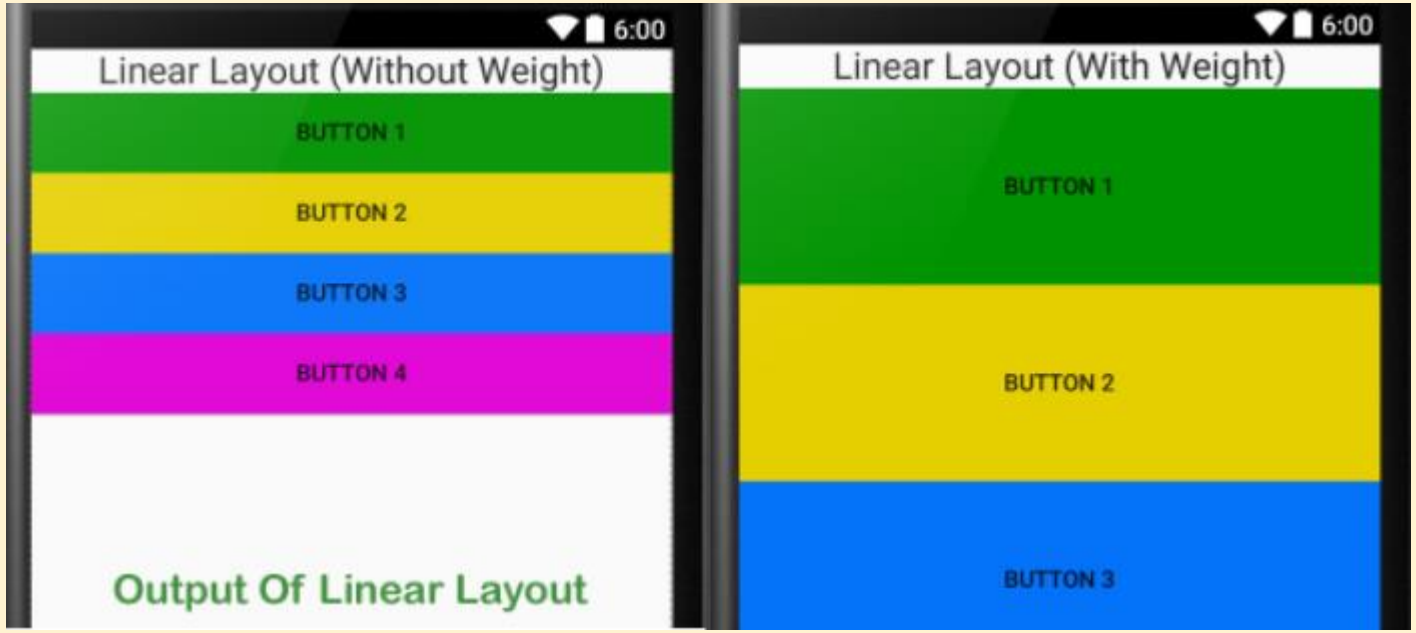


Layout Weight




```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```

Example of Linear Layout:



Example of Linear Layout:

P P SAVANI UNIVERSITY

```xml
<!-- Vertical Orientation is set -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <!-- Text Displayed At Top -->

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Linear Layout (Without Weight)"
        android:id="@+id/textView"
        android:layout_gravity="center_horizontal" />

    <!-- Button Used -->

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 1"
        android:background="#009300" />

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 2"
        android:background="#e6cf00" />
```

```xml
<!-- Vertical Orientation is set  with weightSum-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:weightSum="5"
    android:orientation="vertical">

    <!-- Text Displayed At Top -->

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Linear Layout (With Weight)"
        android:id="@+id/textView"
        android:layout_gravity="center_horizontal"
        android:layout_weight="0"/>

    <!-- Button Used -->

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 1"
        android:background="#009300"
        android:layout_weight="1"/>

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 2"
        android:background="#e6cf00"
        android:layout_weight="1"/>
```

Relative Layout

P P SAVANI UNIVERSITY

- The Relative Layout is very flexible layout used in android for custom layout designing. It gives us the flexibility to position our component/view based on the relative or sibling component's position.
- Just because it allows us to position the component anywhere we want so it is considered as most flexible layout.
- For the same reason Relative layout is the most used layout after the Linear Layout in Android.
- It allow its child view to position relative to each other or relative to the container or another container.

Relative Layout

P P SAVANI UNIVERSITY

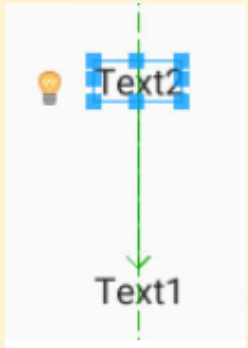
- In Relative Layout, you can use “above, below, left and right” to arrange the component’s position in relation to other component. For example, in the below image you can see content is placed in related to Heading.

Attributes of Relative layout

P P SAVANI UNIVERSITY

1.above: Position the bottom edge of the view above the given anchor view ID and must be a reference of the another resource in the form of id. Example,
android:layout_above="@+id/textView"

```
<!-- textView2 is placed above textView-->
<TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Text2"
        android:id="@+id/textView2"
        android:layout_above="@+id/textView"
        android:layout_marginBottom="100dp"
        android:layout_centerHorizontal="true"/>
```

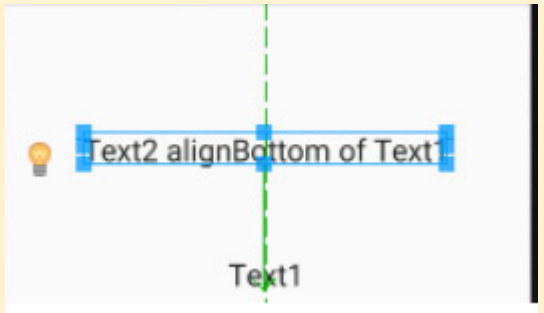


Attributes of Relative layout

P P SAVANI UNIVERSITY

2. alignBottom: alignBottom is used to makes the bottom edge of the view match the bottom edge of the given anchor view ID and it must be a reference to another resource, in the form of id. Example: android:layout_ alignBottom =”@+id/button1”

```
<!-- textView2 alignBottom of textView -->
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:layout_centerHorizontal="true"
    android:id="@+id/textView2"
    android:layout_alignBottom="@+id/textView"
    android:text="Text2 alignBottom of Text1"
    android:layout_marginBottom="90dp"
/>
```

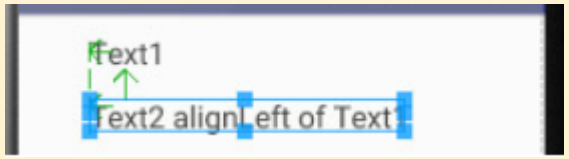


Attributes of Relative layout

P P SAVANI UNIVERSITY

3. alignLeft: alignLeft is used to make the left edge of the view match the left edge of the given anchor view ID and must be a reference to another resource, in the form of Example: android:layout_ alignLeft =”@+id/button1”.

```
<!-- textView2 alignLeft of textView -->
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:id="@+id/textView2"
    android:layout_alignLeft="@+id/textView"
    android:text="Text2 alignLeft of Text1"
    android:layout_below="@+id/textView"
    android:layout_marginTop="20dp"/>
```

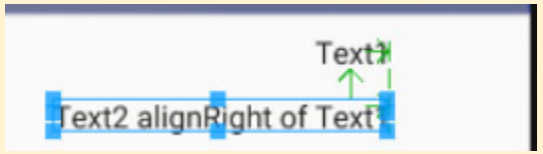


Attributes of Relative layout

P P SAVANI UNIVERSITY

4. alignRight: alignRight property is used to make the right edge of this view match the right edge of the given anchor view ID and must be a reference to another resource, in the form like this example: android:layout_alignRight=”@+id/button1”

```
<!-- textView2 alignRight of textView-->
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:id="@+id/textView2"
    android:layout_alignRight="@+id/textView"
    android:text="Text2 alignRight of Text1"
    android:layout_below="@+id/textView"
    android:layout_marginTop="20dp"/>
```



Attributes of Relative layout

P P SAVANI UNIVERSITY

5.alignStart: alignStart property is used to makes the start edge of this view match the start edge of the given anchor view ID and must be a reference to another resource, in the form of like this example: android:layout_alignStart=”@+id/button1”

```
<!-- Text2 alignStart-->
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:id="@+id/textView2"
    android:text="Text2 align start of Text1"
    android:layout_alignStart="@+id/textView"
 />
```

Attributes of Relative layout

P P SAVANI UNIVERSITY

6. alignTop: alignTop property is used to makes the top edge of this view match the top edge of the given anchor view ID and must be a reference to another resource, in the form like this example: android:layout_alignTop=”@+id/button1”.

```
<!--text is align top on Image-->
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:id="@+id/textView"
    android:layout_alignTop="@+id/imageView"
    android:text="Text Here is AlignTop on Image"
     />
```

Attributes of Relative layout

P P SAVANI UNIVERSITY

7.alignParentBottom: If alignParentBottom property is true, makes the bottom edge of this view match the bottom edge of the parent. The value of align parent bottom is either true or false. Example: android:layout_alignParentBottom=”true”

alignParentBottom and alignBottom are two different properties. In alignBottom we give the reference of another view in the form of id that the view is aligned at the bottom of referenced view but in alignParentBottom the bottom edge of the view matches the bottom edge of the parent.

P P SAVANI UNIVERSITY

8. alignParentEnd: If alignParentEnd property is true, then it makes the end edge of this view match the end edge of the parent. The value of align parent End is either true or false. Example: android:layout_alignParentEnd=”true”.

```
<!-- Text displayed in the end of parent image-->
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:id="@+id/textView"
    android:text="Text in Parent End"
    android:layout_alignBottom="@+id/imageView"
    android:layout_alignParentEnd="true"
 />
```

P P SAVANI UNIVERSITY

9. alignParentLeft: If alignParentLeft property is true, makes the left edge of this view match the left edge of the parent. The value of align parent left is either true or false. Example: android:layout_alignParentLeft=”true”.

```
<!-- align parent left in Android -->
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:id="@+id/textView"
    android:text="Text in Parent Left"
    android:layout_alignBottom="@+id/imageView"
    android:layout_alignParentLeft="true"
     />
```

P P SAVANI UNIVERSITY

10. alignParentRight: If alignParentRight property is true, then it makes the right edge of this view match the right edge of the parent. The value of align parent right is either true or false. Example: android:layout_alignParentRight=”true”.

```
<!-- alignRightParent Example -->
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:id="@+id/textView"
    android:text="Text in Parent Right"
    android:layout_alignBottom="@+id/imageView"
    android:layout_alignParentRight="true"
  />
```

P P SAVANI UNIVERSITY

11.alignParentStart: If alignParentStart is true, then it makes the start edge of this view match the start edge of the parent. The value of align parent start is either true or false. Example: android:layout_alignParentStart=”true”.

```
<!-- alignParentStart Example -->
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:id="@+id/textView"
    android:text="Text in Parent Start"
    android:layout_alignTop="@+id/imageView"
    android:layout_alignParentStart="true"
     />
```

P P SAVANI UNIVERSITY

12.alignParentTop: If alignParentTop is true, then it makes the top edge of this view match the top edge of the parent. The value of align parent Top is either true or false. Example: android:layout_alignParenTop=”true”.

```
<!-- alignParentTop example -->
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Text1 align parent top"
    android:layout_alignParentTop="true"
    android:layout_margin="20dp"
    android:textSize="20sp"
    android:textColor="#000"/>
```

13.centerInParent: If center in parent is true, makes the view in the center of the screen vertically and horizontally. The value of center in parent is either true or false. Example: android:layout_centerInParent=”true”.

```
<!-- centerInParent example -->
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Text1 center in parent"
    android:layout_centerInParent="true"
    android:textSize="20sp"
    android:textColor="#000"
/>
```

14.centerHorizontal: If centerHorizontal property is true, makes the view horizontally center. The value of centerHorizontal is either true or false.Example: android:layout_centerHorizontal=”true”.

```
<!-- centerHorizontal example -->
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Text1 center Horizontal"
    android:layout_centerHorizontal="true"
    android:textSize="20sp"
    android:textColor="#000"
/>
```

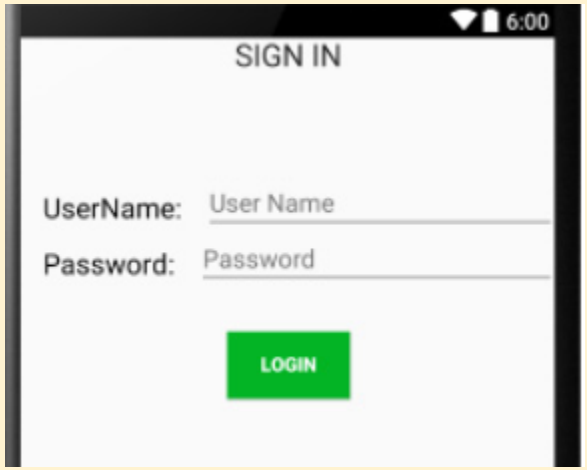
15.centerVertical: If centerVertical property is true, make the view vertically center. The value of align parent bottom is either true or false. Example: android:layout_centerVertical=”true”.

```
<!-- centerVertical example -->
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Text1 center vertical"
    android:layout_centerVertical="true"
    android:textSize="20sp"
    android:textColor="#000"
/>
```

Relative Layout Example

P P SAVANI UNIVERSITY

- **Assignment 2 for LAB = Login Application**



Difference between Linear And Relative Layout:

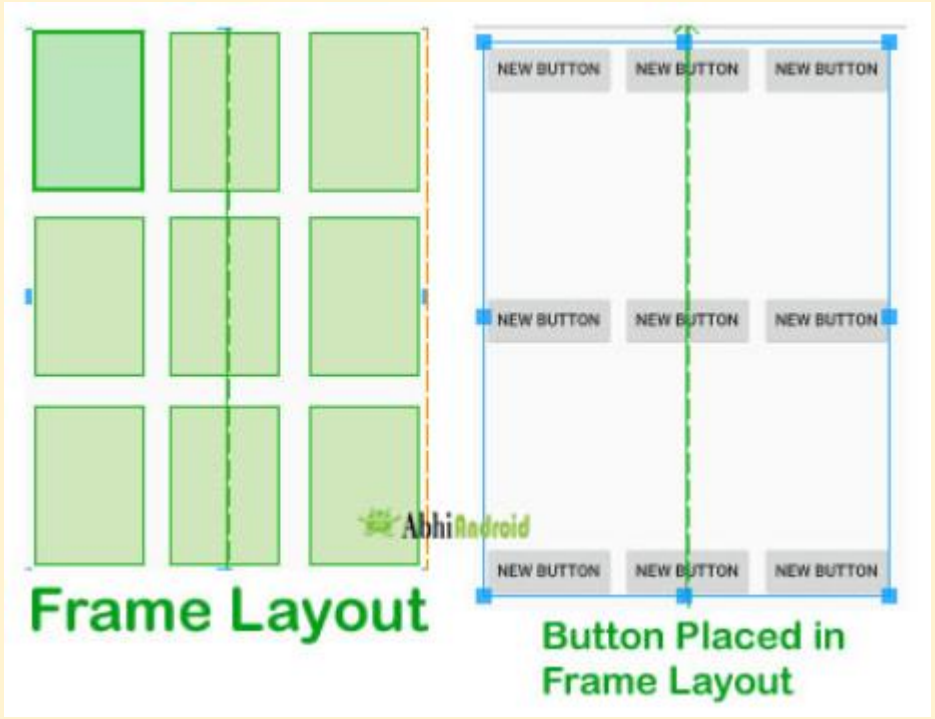
- **RELATIVE LAYOUT:**
  - **Every element of relative layout arranges itself to the other element or a parent element.**
  - **It is helpful while adding views one next to other etc**
  - **In a relative layout you can give each child a Layout Property that specifies exactly where it should go in relative to the parent or relative to other children.**
  - **Views can be layered on top of each other.**
- **LINEAR LAYOUT:**
  - **In a linear layout, like the name suggests, all the elements are displayed in a linear fashion either vertically or horizontally.**
  - **Either Horizontally or Vertically this behavior is set in android:orientation which is an property of the node Linear Layout.**
  - **android:orientation=”horizontal” or android:orientation=”vertical”**
  - **Linear layouts put every child, one after the other, in a line, either horizontally or vertically.**

Frame Layout

P P SAVANI UNIVERSITY

- Frame Layout is one of the simplest layout to organize view controls. They are designed to block an area on the screen. Frame Layout should be used to hold child view, because it can be difficult to display single views at a specific area on the screen without overlapping each other.

- We can add multiple children to a FrameLayout and control their position by assigning gravity to each child, using the android:layout_gravity attribute.



Attributes of Frame Layout

P P SAVANI UNIVERSITY

1. android:id : This is the unique id which identifies the layout in the R.java file.

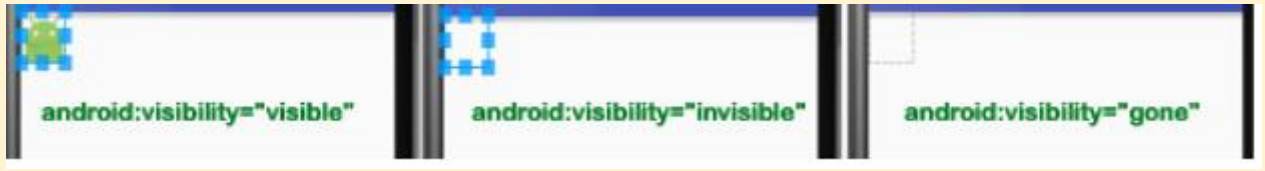
2. android:foreground : Foreground defines the drawable to draw over the content and this may be a color value. Possible color values can be in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb". This all are different color code model used

Attributes of Frame Layout

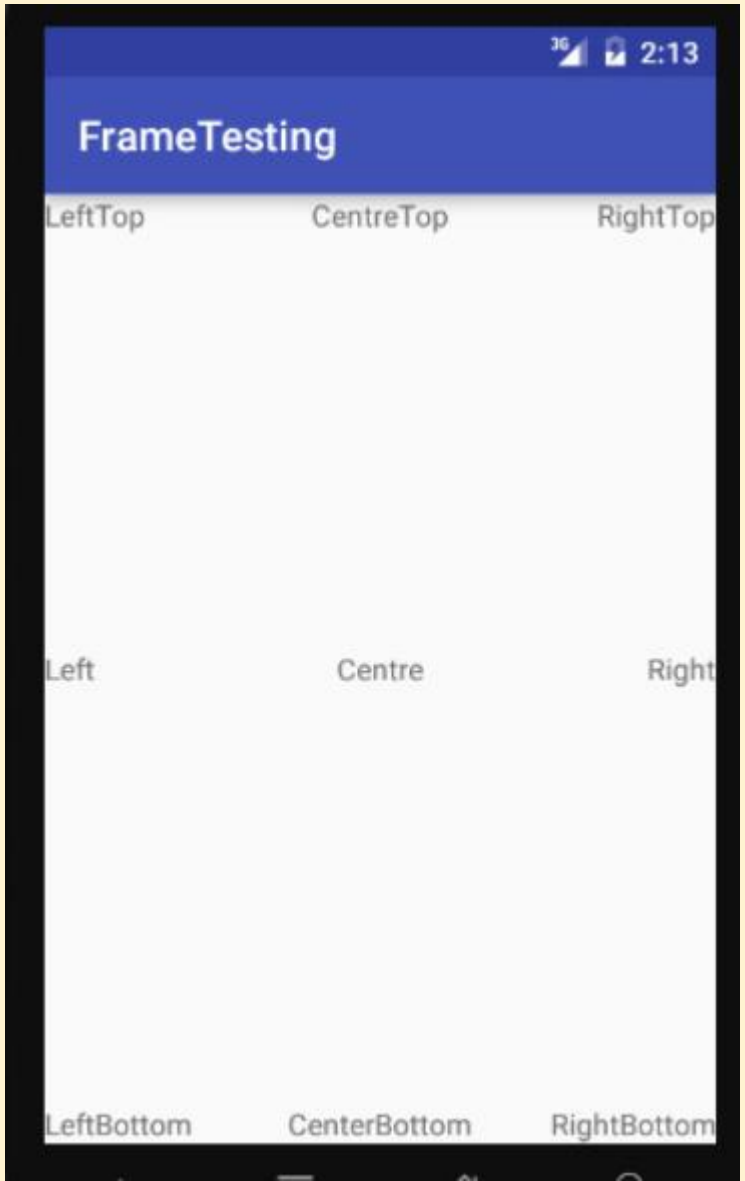
P P SAVANI UNIVERSITY

3. android:visibility : This determine whether to make the view visible, invisible or gone.

- **visible –** the view is present and also visible
- **invisible –** The view is present but not visible
- **gone –** The view is neither present nor visible



Example Of Frame Layout in Android Studio




```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    >
    <TextView android:text="LeftTop"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="RightTop"
        android:layout_gravity="top|right" />
    <TextView android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="CentreTop"
        android:layout_gravity="top|center_horizontal" />
    <TextView android:text="Left"
        android:layout_gravity="left|center_vertical"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="Right"
        android:layout_gravity="right|center_vertical" />
    <TextView android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="Centre"
        android:layout_gravity="center" />
    <TextView android:text="LeftBottom"
        android:layout_gravity="left|bottom"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="RightBottom"
        android:layout_gravity="right|bottom" />
    <TextView android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="CenterBottom"
        android:layout_gravity="center|bottom" />
</FrameLayout>
```

Example 2: Frame Layout without using layout gravity

P P SAVANI UNIVERSITY

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/frameLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="fitXY"
        android:src="@drawable/img_name" /><!--Change image as per your name of image saved in drawable-->
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:gravity="center"
        android:text="abhiAndroid"
        android:textSize="30sp"
        android:textColor="#f3f3f3"
        android:textStyle="bold" />
</FrameLayout>
```

Table Layout

P P SAVANI UNIVERSITY

- In Android, Table Layout is used to arrange the group of views into rows and columns. Table Layout containers do not display a border line for their columns, rows or cells. A Table will have as many columns as the row with the most cells.

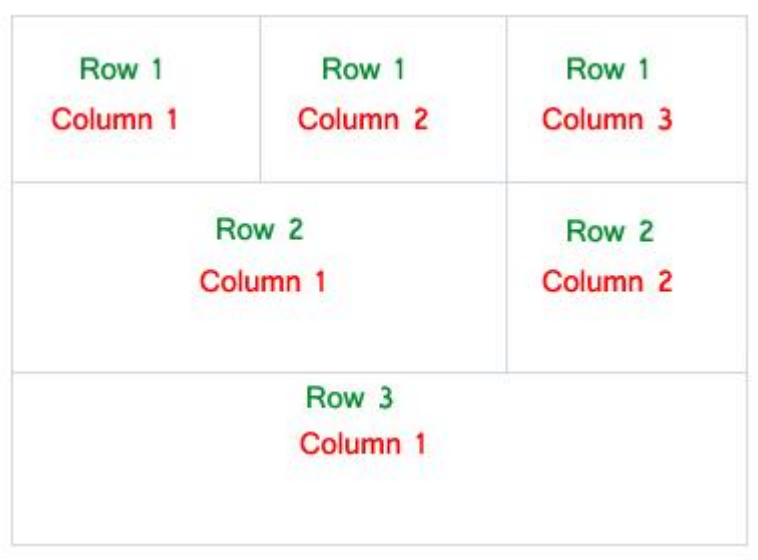


Table Layout In Android

P P SAVANI UNIVERSITY

- For building a row in a table we will use the <TableRow> element. Table row objects are the child views of a table layout.
- Each row of the table has zero or more cells and each cell can hold only one view object like ImageView, TextView or any other view.
- Total width of a table is defined by its parent container
- Column can be both stretchable and shrinkable. If shrinkable then the width of column can be shrunk to fit the table into its parent object and if stretchable then it can expand in width to fit any extra space available.

Basic Table Layout code in XML:

P P SAVANI UNIVERSITY

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:collapseColumns="0"> <!-- collapse the first column of the table row-->


    <!-- first row of the table layout-->
    <TableRow
        android:id="@+id/row1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">

        <!-- Add elements/columns in the first row-->

    </TableRow>
</TableLayout>
```

Attributes of TableLayout in Android

P P SAVANI UNIVERSITY

1. **id:** id attribute is used to uniquely identify a Table Layout.
2. **stretchColumns:** Stretch column attribute is used in Table Layout to change the default width of a column which is set equal to the width of the widest column but we can also stretch the columns to take up available free space by using this attribute. The value that assigned to this attribute can be a single column number or a comma delimited list of column numbers (1, 2, 3…n).

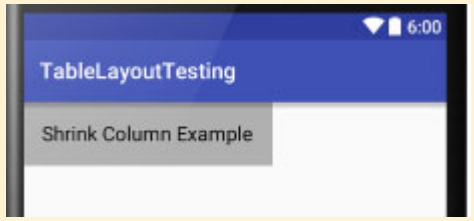
```
<?xml version="1.0" encoding="utf-8"?>

    <TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/simpleTableLayout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:stretchColumns="1"> <!-- stretch the second column of the layout-->
```

Attributes of TableLayout in Android

P P SAVANI UNIVERSITY

3. shrinkColumns: Shrink column attribute is used to shrink or reduce the width of the column‘s. We can specify either a single column or a comma delimited list of column numbers for this attribute. The content in the specified columns word-wraps to reduce their width.

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:shrinkColumns="0"> <!-- shrink the first column of the layout-->
```



Attributes of TableLayout in Android

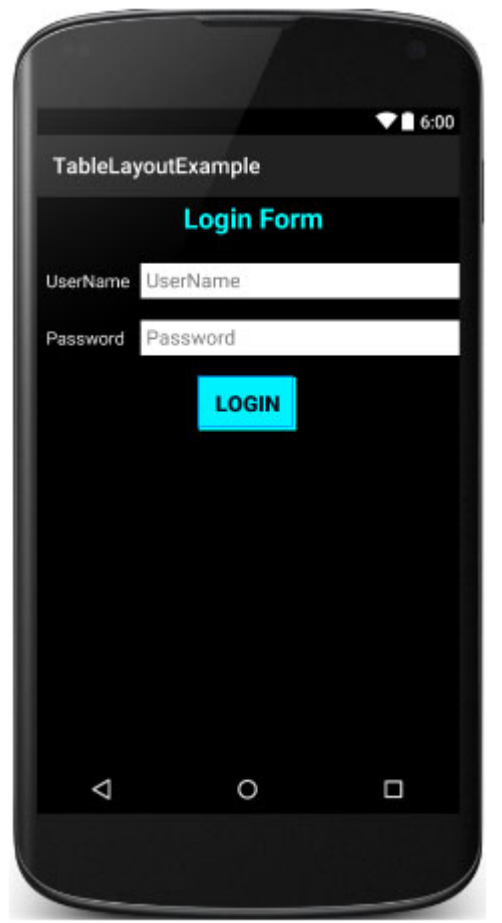
P P SAVANI UNIVERSITY

4. collapseColumns: collapse columns attribute is used to collapse or invisible the column's of a table layout. These columns are the part of the table information but are invisible.

- If the values is 0 then the first column appears collapsed, i.e it is the part of table but it is invisible.

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:collapseColumns="0"> <!-- collapse the first column of the table row-->


    <!-- first row of the table layout-->
    <TableRow
        android:id="@+id/simpleTableLayout"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">

        <!-- first element of the row that is the part of table but it is invisible-->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="#b0b0b0"
            android:padding="18dip"
            android:text="Columns 1"
            android:textColor="#000"
            android:textSize="18dp" />

        <!-- second element of the row that is shown in the screenshot-->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="#b0b0b0"
            android:padding="18dip"
            android:text="Columns 2"
            android:textColor="#000"
            android:textSize="18dp" />
    </TableRow>
</TableLayout>
```

TableLayout Example




```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000"
    android:orientation="vertical"
    android:stretchColumns="1">

    <TableRow android:padding="5dip">

        <TextView
            android:layout_height="wrap_content"
            android:layout_marginBottom="20dp"
            android:layout_span="2"
            android:gravity="center_horizontal"
            android:text="@string/loginForm"
            android:textColor="#0ff"
            android:textSize="25sp"
            android:textStyle="bold" />
    </TableRow>

    <TableRow>

        <TextView
            android:layout_height="wrap_content"
            android:layout_column="0"
            android:layout_marginLeft="10dp"
            android:text="@string/userName"
            android:textColor="#fff"
            android:textSize="16sp" />

        <EditText
            android:id="@+id/userName"
            android:layout_height="wrap_content"
            android:layout_column="1"
            android:layout_marginLeft="10dp"
            android:background="#fff"
            android:hint="@string/userName"
            android:padding="5dp"
            android:textColor="#000" />
    </TableRow>
```

TableLayout Example

P P SAVANI UNIVERSITY

```
    <TableRow>

        <TextView
            android:layout_height="wrap_content"
            android:layout_column="0"
            android:layout_marginLeft="10dp"
            android:layout_marginTop="20dp"
            android:text="@string/password"
            android:textColor="#fff"
            android:textSize="16sp" />

        <EditText
            android:id="@+id/password"
            android:layout_height="wrap_content"
            android:layout_column="1"
            android:layout_marginLeft="10dp"
            android:layout_marginTop="20dp"
            android:background="#fff"
            android:hint="@string/password"
            android:padding="5dp"
            android:textColor="#000" />
    </TableRow>


    <TableRow android:layout_marginTop="20dp">

        <Button
            android:id="@+id/loginBtn"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_span="2"
            android:background="#0ff"
            android:text="@string/login"
            android:textColor="#000"
            android:textSize="20sp"
            android:textStyle="bold" />
    </TableRow>
</TableLayout>
```

```
<resources>
    <string name="app_name">TableLayoutExample</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="loginForm">Login Form</string>
    <string name="userName">UserName</string>
    <string name="password">Password</string>
    <string name="login">LogIn</string>
</resources>
```

Using multiple Layouts and Views to design a GUI

Using multiple Layouts and Views to design a GUI

P P SAVANI UNIVERSITY

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FFF59D"
    tools:context="com.example.android.myapplication.MainActivity">
    <!--Light Yellow Color-->

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:id="@+id/l1"
        android:background="#FF9E80">
        <!--Light Pink Color-->
```

```
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Studytonight"
            android:textAllCaps="true"
            android:textSize="40dp"
            android:textColor="#F44336"/>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Android"
            android:textAllCaps="true"
            android:textSize="40dp"
            android:textColor="#F44336"/>

    </LinearLayout>
```

Using multiple Layouts and Views to design a GUI

P P SAVANI UNIVERSITY

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_below="@id/l1"
    android:background="#039BE5"
    android:id="@+id/l2">
    <!--Light Blue Color-->

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Studytonight"
        android:textAllCaps="true"
        android:textSize="30dp"
        android:textColor="#76FF03"
        />
```

```
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Android"
            android:textAllCaps="true"
            android:textSize="30dp"
            android:textColor="#76FF03"
            />

</LinearLayout>
```

Using multiple Layouts and Views to design a GUI

P P SAVANI
UNIVERSITY

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/l2"
    android:background="#7E57C2"
    >
    <!--Light Purple Color-->

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Toast 1"
        android:id="@+id/b1"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Toast 2"
        android:layout_toRightOf="@id/b1"
        android:id="@+id/b2"
        />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Toast 3"
            android:layout_below="@id/b1"
            android:id="@+id/b3"
            />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Toast 4"
            android:layout_below="@id/b2"
            android:layout_toRightOf="@id/b3"
            />

    </RelativeLayout>
```

Using multiple Layouts and Views to design a GUI

- In the UI design above, we have a root element which is RelativeLayout. This means that all its children, whether it is layouts or views will be arranged in a relative fashion. Thus, it is very important to choose our root layout carefully.
- Next, we have a Linear Layout, whose orientation is set to vertical. This means that all the elements inside this Linear Layout will be arranged in a vertical fashion.
- Next, we have another Linear Layout, whose orientation is set to horizontal. Thus, as you can see, we can have different layouts placed inside a root layout.
- Next, we have another Relative Layout placed inside the root Relative Layout. This Relative Layout has 4 buttons inside it, arranged in a relative fashion to each other.
- Remember, all these layouts are placed inside a Relative Layout. Thus, all these layouts are relative to each other.

Data-Driven Containers

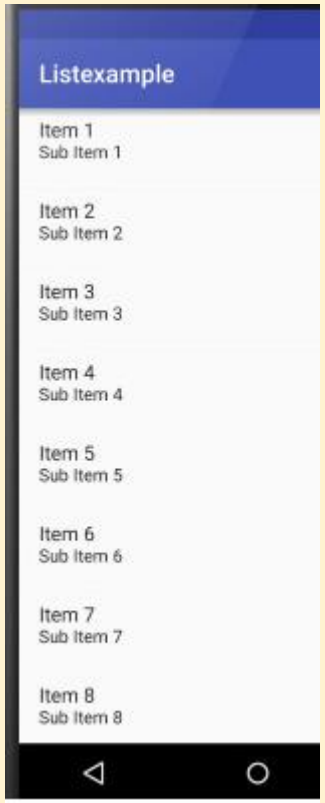
P P SAVANI UNIVERSITY

- ListView
- GridView

ListView

P P SAVANI UNIVERSITY

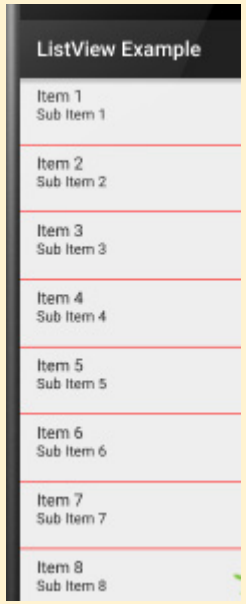
- List of scrollable items can be displayed in Android using ListView. It helps you to displaying the data in the form of a scrollable list.
- Users can then select any list item by clicking on it. ListView is default scrollable so we do not need to use scroll View or anything else with ListView.
- ListView is widely used in android applications. A very common example of ListView is your phone contact book, where you have a list of your contacts displayed in a ListView and if you click on it then user information is displayed.



Attributes of ListView

P P SAVANI UNIVERSITY

1. **id:** id is used to uniquely identify a ListView.
2. **divider:** This is a drawable or color to draw between different list items.



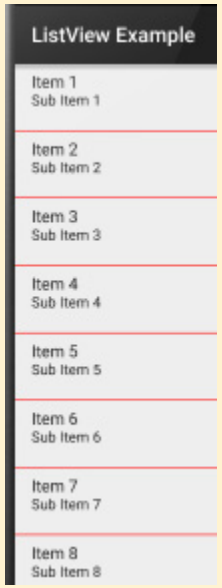

```
<!--Divider code in ListView-->
<ListView
android:id="@+id/simpleListView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:divider="#f00"
android:dividerHeight="1dp"
/>
```

Attributes of ListView

P P SAVANI UNIVERSITY

3. dividerHeight: This specify the height of the divider between list items. This could be in dp(density pixel),sp(scale independent pixel) or px(pixel).

```
<!--Divider code in ListView-->
<ListView
android:id="@+id/simpleListView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:divider="#f00"
android:dividerHeight="1dp"
/>
```



Attributes of ListView

P P SAVANI UNIVERSITY

4. listSelector: listSelector property is used to set the selector of the listView. It is generally orange or Sky blue color mostly but you can also define your custom color or an image as a list selector as per your design.

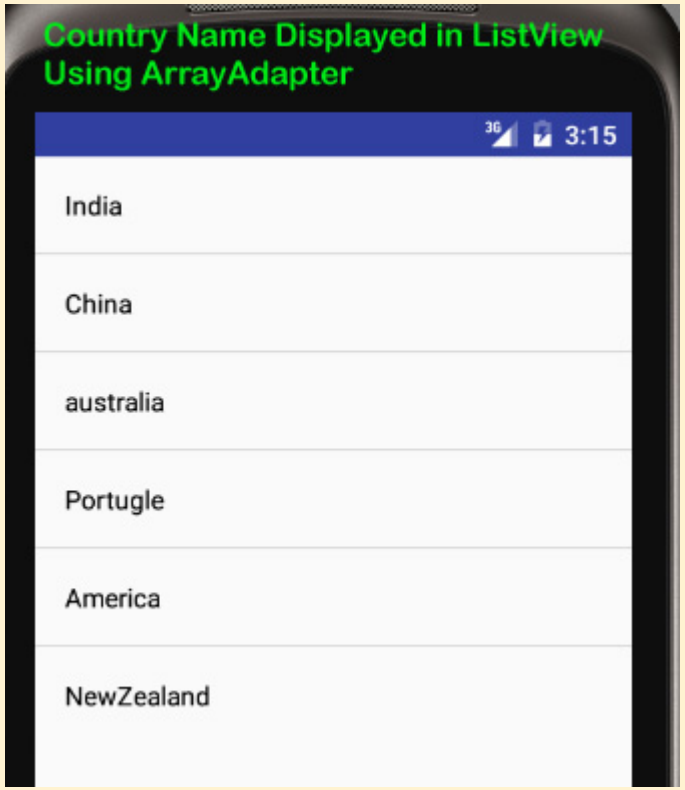
```
<!-- List Selector Code in ListView -->
<ListView
android:id="@+id/simpleListView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:divider="#f00"
android:dividerHeight="1dp"
android:listSelector="#0f0"/> <!--list selector in green color-->
```

Adapters Use in ListView

P P SAVANI UNIVERSITY

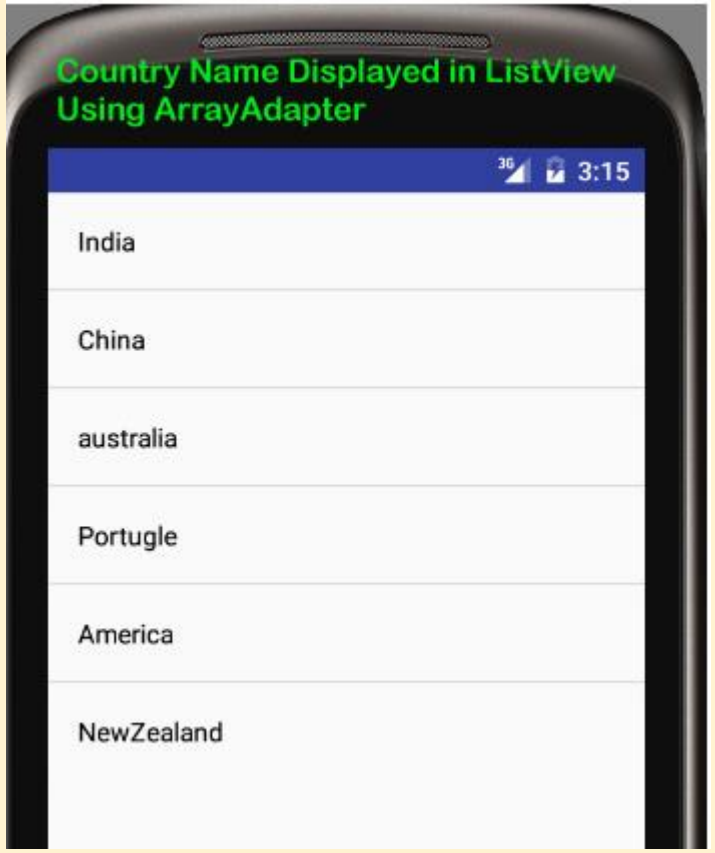
- **Array Adapter:**

```
ArrayAdapter adapter = new ArrayAdapter<String>(this,R.layout.ListView,R.id.textView,StringArray);
```



Example of list view using Array Adapter

P P SAVANI UNIVERSITY



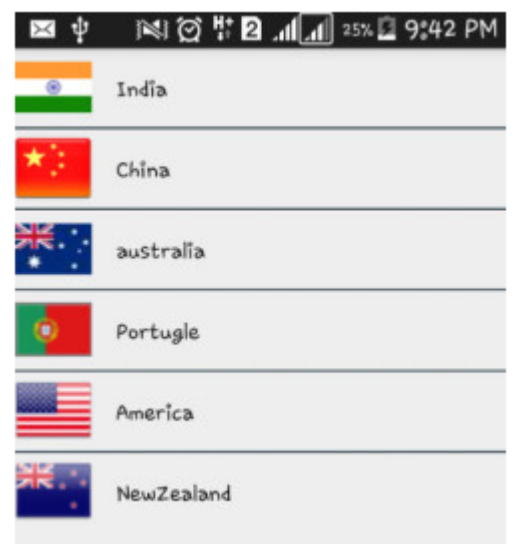

```
package abhiandroid.com.listexample;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.widget.ArrayAdapter;import android.widget.ListView;

public class MainActivity extends Activity
{
    // Array of strings...
    ListView simpleList;
    String countryList[] = {"India", "China", "australia", "Portugle", "America", "NewZealand"};

    @Override   protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);      setContentView(R.layout.activity_main);
        simpleList = (ListView)findViewById(R.id.simpleListView);
        ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(this, R.layout.activity_listview, R.i
        simpleList.setAdapter(arrayAdapter);
    }
}
```

Adapters Use in ListView

P P SAVANI UNIVERSITY

- **Base Adapter:**
- **BaseAdapter is a common base class of a general implementation of an Adapter that can be used in ListView. Whenever you need a customized list you create your own adapter and extend base adapter in that.**
- **Base Adapter can be extended to create a custom Adapter for displaying a custom list item. ArrayAdapter is also an implementation of BaseAdapter.**

Example of list view using Base Adapter



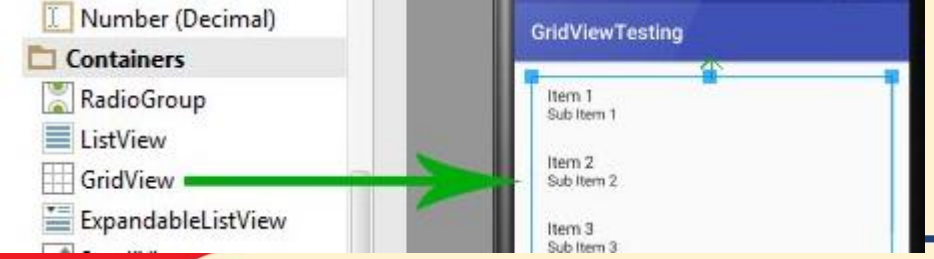

```
package com.abhiandroid.listbaseexample;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ListView;

public class MainActivity extends Activity {

ListView simpleList;
String countryList[] = {"India", "China", "australia", "Portugle", "America", "NewZealand"};
int flags[] = {R.drawable.india, R.drawable.china, R.drawable.australia, R.drawable.portugle,

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
simpleList = (ListView) findViewById(R.id.simpleListView);
CustomAdapter customAdapter = new CustomAdapter(getApplicationContext(), countryList, flags);
simpleList.setAdapter(customAdapter);
}
}
```

CustomAdapter.java

P P SAVANI UNIVERSITY

```java
package com.abhiandroid.listbaseexample;

import android.content.Context;
import android.media.Image;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.TextView;

import java.util.zip.Inflater;

public class CustomAdapter extends BaseAdapter {
Context context;
String countryList[];
int flags[];
LayoutInflater inflter;

public CustomAdapter(Context applicationContext, String[] countryList, int[] flags) {
this.context = context;
this.countryList = countryList;
this.flags = flags;
inflter = (LayoutInflater.from(applicationContext));
}

@Override
public int getCount() {
return countryList.length;
}

@Override
public Object getItem(int i) {
return null;
}

@Override
public long getItemId(int i) {
return 0;
}
```

```java
@Override
public View getView(int i, View view, ViewGroup viewGroup) {
view = inflter.inflate(R.layout.activity_listview, null);
TextView country = (TextView)           view.findViewById(R.id.textView);
ImageView icon = (ImageView) view.findViewById(R.id.icon);
country.setText(countryList[i]);
icon.setImageResource(flags[i]);
return view;
}
```



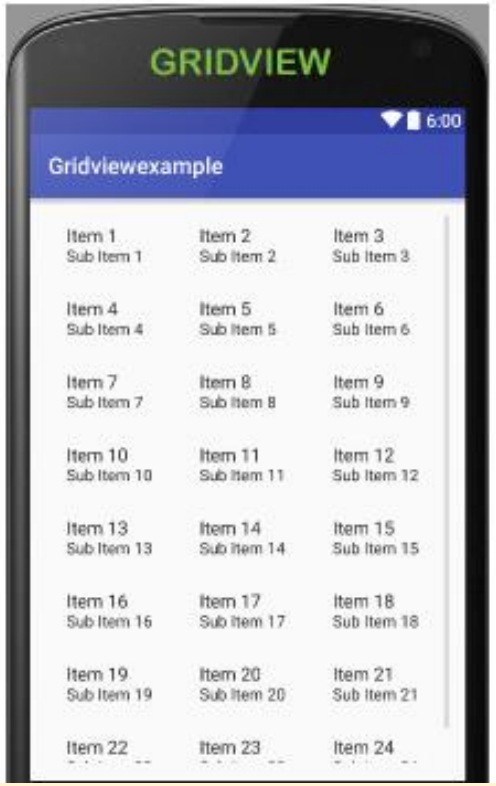
GridView Tutorial

P P SAVANI UNIVERSITY

- In android GridView is a view group that display items in two dimensional scrolling grid (rows and columns), the grid items are not necessarily predetermined but they are automatically inserted to the layout using a ListAdapter.
- Users can then select any grid item by clicking on it. GridView is default scrollable so we don’t need to use ScrollView or anything else with GridView.
- GridView is widely used in android applications.
- An example of GridView is your default Gallery, where you have number of images displayed using grid.

Basic GridView code in XML

```
<GridView
android:id="@+id/simpleGridView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:numColumns="3"/>
```



Attributes of GridView

P P SAVANI UNIVERSITY

1.id: id is used to uniquely identify a GridView.

2.numColumns: numColumn define how many columns to show. It may be a integer value, such as “5” or auto_fit.

3. horizontalSpacing: horizontalSpacing property is used to define the default horizontal spacing between columns. This could be in pixel(px),density pixel(dp) or scale independent pixel(sp).


```
<!--Horizontal space example code in grid view-->>
<GridView
android:id="@+id/simpleGridView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:numColumns="3"
android:horizontalSpacing="50dp"/><!--50dp horizontal space between grid items-->
```

Attributes of GridView

P P SAVANI UNIVERSITY

4.verticalSpacing: verticalSpacing property used to define the default vertical spacing between rows. This should be in px, dp or sp.

```
<!-- Vertical space between grid items code -->
<GridView
android:id="@+id/simpleGridView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:numColumns="3"
android:verticalSpacing="50dp"/><!--50dp vertical space set between grid items-->
```

Attributes of GridView

P P SAVANI UNIVERSITY

5.columnWidth: columnWidth property specifies the fixed width of each column. This could be in px, dp or sp.

- Below is the columnWidth example code. Here column width is 80dp and selected item's background color is green which shows the actual width of a grid item.

```
<!--columnWidth in Grid view code-->
<GridView
android:id="@+id/simpleGridView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:numColumns="3"
android:columnWidth="80dp"
android:listSelector="#0f0"/><!--define green color for selected item-->
```

GridView Example

P P SAVANI UNIVERSITY

```
ArrayAdapter adapter = new ArrayAdapter<String>(this,R.layout.ListView,R.id.textView,StringArray);
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <!--
    GridView with 3 value for numColumns attribute
    -->
    <GridView
        android:id="@+id/simpleGridView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:footerDividersEnabled="false"
        android:padding="1dp"
        android:numColumns="3" />
</LinearLayout>
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="1dp"
    android:orientation="vertical">
    <ImageView
        android:id="@+id/icon"
        android:layout_width="match_parent"
        android:layout_height="120dp"
        android:scaleType="fitXY"
        android:layout_gravity="center_horizontal"
        android:src="@drawable/logo1" />
</LinearLayout>
```

activity_gridview.xml

MainActivity.java

P P SAVANI UNIVERSITY

```java
package com.example.gourav.GridViewExample;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.AdapterView;
import android.widget.GridView;
public class MainActivity extends AppCompatActivity {
    GridView simpleGrid;
    int logos[] = {R.drawable.logo1, R.drawable.logo2, R.drawable.logo3, R.drawable.logo4,
            R.drawable.logo5, R.drawable.logo6, R.drawable.logo7, R.drawable.logo8, R.drawable.logo9,
            R.drawable.logo10, R.drawable.logo11, R.drawable.logo12, R.drawable.logo13};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        simpleGrid = (GridView) findViewById(R.id.simpleGridView); // init GridView
        // Create an object of CustomAdapter and set Adapter to GirdView
        CustomAdapter customAdapter = new CustomAdapter(getApplicationContext(), logos);
        simpleGrid.setAdapter(customAdapter);
        // implement setOnItemClickListener event on GridView
        simpleGrid.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                // set an Intent to Another Activity
                Intent intent = new Intent(MainActivity.this, SecondActivity.class);
                intent.putExtra("image", logos[position]); // put image data in Intent
                startActivity(intent); // start Intent
            }
        });
    }
}
```

CustomAdapter.java

P P SAVANI UNIVERSITY

```java
package com.example.gourav.GridViewExample;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
public class CustomAdapter extends BaseAdapter {
    Context context;
    int logos[];
    LayoutInflater inflter;
    public CustomAdapter(Context applicationContext, int[] logos) {
        this.context = applicationContext;
        this.logos = logos;
        inflter = (LayoutInflater.from(applicationContext));
    }
    @Override
    public int getCount() {
        return logos.length;
    }
    @Override
    public Object getItem(int i) {
        return null;
    }
    @Override
    public long getItemId(int i) {
        return 0;
    }
    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        view = inflter.inflate(R.layout.activity_gridview, null); // inflate the layout
        ImageView icon = (ImageView) view.findViewById(R.id.icon); // get the reference of ImageView
        icon.setImageResource(logos[i]); // set logo images
        return view;
    }
}
```

activity_second.xml

P P SAVANI UNIVERSITY

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:background="#fff"
    tools:context="com.example.gourav.GridViewExample.SecondActivity">
    <ImageView
        android:id="@+id/selectedImage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:scaleType="fitXY" />
</RelativeLayout>
```

SecondActivity.java

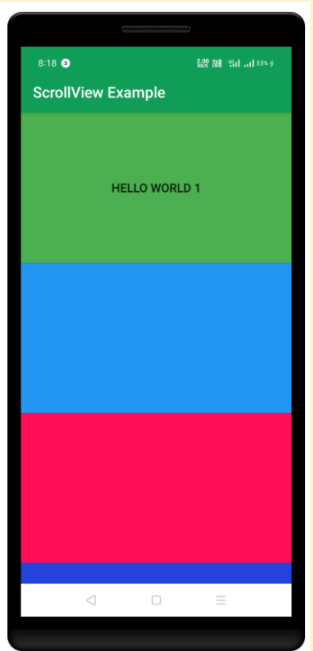
P P SAVANI UNIVERSITY

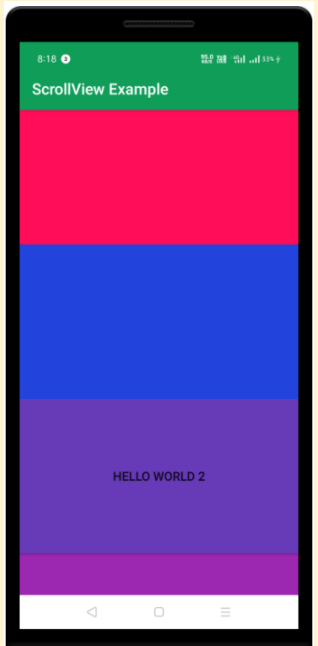
```
package com.example.gourav.GridViewExample;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.ImageView;
public class SecondActivity extends AppCompatActivity {
    ImageView selectedImage;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
        selectedImage = (ImageView) findViewById(R.id.selectedImage); // init a ImageView
        Intent intent = getIntent(); // get Intent which we set from Previous Activity
        selectedImage.setImageResource(intent.getIntExtra("image", 0)); // get image from Intent and s
    }
}
```

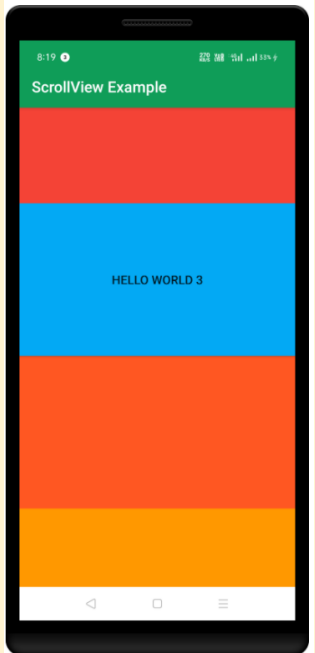
Adding Scrolling Support

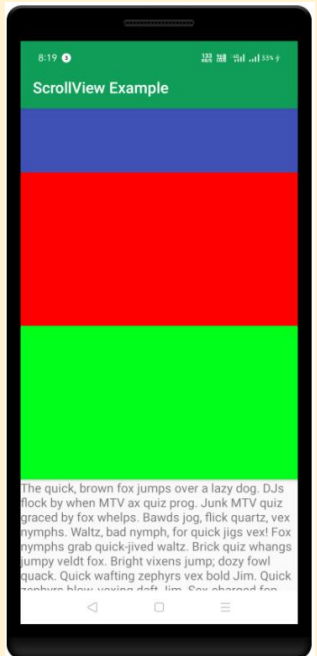
P P SAVANI UNIVERSITY

- A ScrollView is an Android view group which is used to make vertically scrollable views in our andrid app . A ScrollView only contains a direct single child.
- The main disadvantage of ScrollView is that it only support vertical scroll, but to add horizontal scroll we have many other options such as HorizontalScrollView









```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="wrap_content">
```

P P SAVANI UNIVERSITY

Adding Scrolling Support

```xml
            <TextView
                android:text="The quick, brown fox jumps over a lazy dog. DJs flock by when MTV
                android:layout_width="match_parent"
                android:layout_height="wrap_content"/>

        </LinearLayout>
    </ScrollView>
</RelativeLayout>
```

P P SAVANI
UNIVERSITY

THANK YOU

P P SAVANI
UNIVERSITY

Module 5
Drawing and Working with Animation

Working with Different Types of Resources

- Working with String Resources
- Working with Colors
- Working with Canvases and Paints
- Working with Images
- Working with Shapes
- Working with Animation

What are Resources?

P P SAVANI UNIVERSITY

- Android applications are composed of two things and they are:
  - Android Code instructions
  - Android Data / resources
- Functionality is in the instruction coded which determines the way our application should behave. Other section of importance is resources i.e. text, images, icons, audio files, videos, etc.
- Android resource files are stored separately from the rest of java files of application. Generally resources are stored as xml files. Graphics and raw data can be stored as resources.

Introduction to Android resources

P P SAVANI UNIVERSITY

Serial Number	Resource sub directory	Usage
1	/res/drawable-ldpi, /res/drawable-mdpi, /res/drawable-hdpi, /res/drawable-xhdpi	Graphics resources
2	/res/layout	User interface
3	/res/values	Simple data like strings, color values, etc

Accessing Resources Programmatically

P P SAVANI UNIVERSITY

- Developers access specific application resources using the R.java class file and its subclasses, which are automatically generated when you add resources to your project.
- You can refer to any resource identifier in your project by its name.
- For example, a String named strHello defined within the resource file called res/values/strings.xml is accessed in the code as follows:

```
R.string.strHello
String mystring = getResources().getString(R.string.strHello);
```

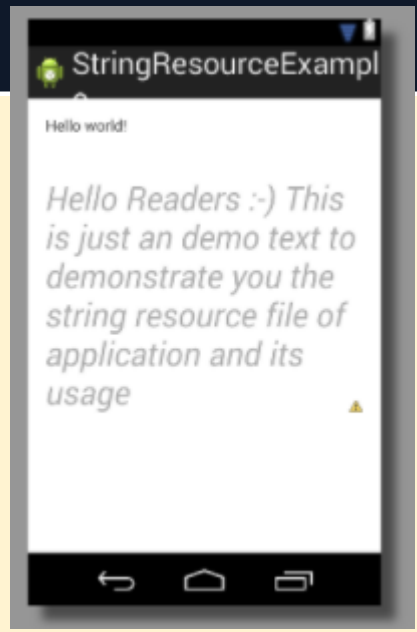
Android Resource Example

Figure - activity_main.xml

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/textView2"
        android:layout_below="@+id/textView2"
        android:layout_marginTop="40dp"
        android:text="@string/intro"
        android:textColor="@android:color/darker_gray"
        android:textSize="30dip"
        android:textStyle="italic" />
</RelativeLayout>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">StringResourceExample</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="intro">Hello Readers :-) This is just an demo text to demonstrate
</resources>
```

Figure - strings.xml file



String Resources

P P SAVANI UNIVERSITY

- A string resource provides text strings for your application with optional text styling and formatting. There are three types of resources that can provide your application with strings:
- String
- XML resource that provides a single string.
- String Array
- XML resource that provides an array of strings.

String

P P SAVANI UNIVERSITY

- **A single string that can be referenced from the application or from other resource files (such as an XML layout).**

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string
        name="app_name">Choose Your Own</string>
    <string
        name="race_orc">Orc</string>
    <string
        name="race_elf">Elf</string>
    <string
        name="race_troll">Troll</string>
    <string
        name="race_human">Human</string>
    <string
        name="race_halfling">Halfling</string>
    <string
        name="race_goblin">Goblin</string>
</resources>
```

```
String src =
getResources().getString(R.string.race_orc);
```

String Array

P P SAVANI UNIVERSITY

- **An array of strings that can be referenced from the application.**

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array
        name="races_array">
        <item>@string/race_goblin</item>
        <item>@string/race_orc</item>
        <item>@string/race_elf</item>
        <item>@string/race_human</item>
        <item>@string/race_troll</item>
        <item>@string/race_halfling</item>
    </string-array>
</resources>
```

```
String[] cRaces = getResources().getStringArray(R.array.races_array);
```

Array Resources in Simple Spinner Controls

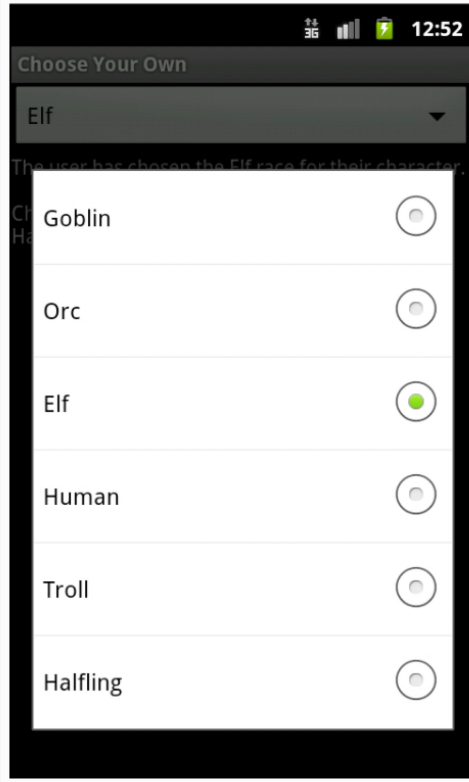
P P SAVANI UNIVERSITY

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array
        name="races_array">
        <item>@string/race_goblin</item>
        <item>@string/race_orc</item>
        <item>@string/race_elf</item>
        <item>@string/race_human</item>
        <item>@string/race_troll</item>
        <item>@string/race_halfling</item>
    </string-array>
</resources>
```

```
String[] cRaces = getResources().getStringArray(R.array.races_array);
```

```
<Spinner
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:id="@+id/spinnerOfCharacterRaces"
    android:entries="@array/races_array">
</Spinner>
```

```
Spinner cRaceSpinner = (Spinner) findViewById(R.id.spinnerOfCharacterRaces);
cRaceSpinner.setOnItemSelectedListener(new OnItemSelectedListener() {
   public void onItemSelected(AdapterView<?> arg0, View arg1,arg2, long arg3) {
      String strChosenRace = (String) arg0.getItemAtPosition(arg2);
   }
   public void onNothingSelected(AdapterView<?> arg0) {}
});
```



Formatting and Styling

P P SAVANI UNIVERSITY

- **Escaping apostrophes and quotes**

```
<string name="good_example">"This'll work"</string>
<string name="good_example_2">This\'ll also work</string>
<string name="bad_example">This doesn't work</string>
<string name="bad_example_2">XML encodings don&amp;apos;t work</string>
```

- **Styling with HTML markup**

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="welcome">Welcome to <b>Android</b>!</string>
</resources>
```

Working with Colors

P P SAVANI UNIVERSITY

- Android Application can store RGB color values,which can then be applied to other screen elements.
- Color values can be used to set the color of text or the screen background.
- Defined in res/values
- Android uses standard RGB (red, green and blue) color model. Each primary color value is usually represented by hexadecimal number. At the beginning of such a color definition you have to put a pound character (#).
- The simplest is just #RGB format, where #000 is black and #FFF is white. But in this format we have only 16 values per color so it gives 4096 combinations. That's why #RRGGBB format is mainly used. In this format we have 256 values per primary color, so 16 777 216 colors in total.
- Black is #000000 and white is #FFFFFF.

Example

```
android:textColor="#AARRGGBB"
```

```
your_text_view.setTextColor(0xAARRGGBB);
```

```
your_text_view.setTextColor(Color.parseColor("#AARRGGBB");
```

```
your_text_view.setTextColor(Color.rgb(R, G, B));
```

```
 your_text_view.setTextColor(Color.argb(A, R, G, B));
```

```
your_text_view. setTextColor(Color.COLOR_NAME);
```

The best solutions is to define your own colors set in the colors.xml file

```
<?xml version="1.0" encoding="utf-8"?>

<resources>

<color name="opaque_orange">#FF8800</color>
<color name="translucent_orange">#80FF8800</color>
<color name="text_color">#0099CC</color>
<color name="background_color">#F0FF00</color>

</resources>
```

```
android:textColor="@color/your_color_name"
```

```
 your_text_view.setTextColor(getResources().getColor(R.color.your_color_name));
```

Android Units of Measurements

P P SAVANI UNIVERSITY

- **px (Pixels)** - Actual pixels or dots on the screen.
- **in (Inches)** - Physical size of the screen in inches.
- **mm (Millimeters)** - Physical size of the screen in millimeters.
- **pt (Points)** - 1/72 of an inch.
- **dp (Density-independent Pixels)** - An abstract unit that is based on the physical density of the screen. These units are relative to a 160 dpi screen, so one dp is one pixel on a 160 dpi screen. The ratio of dp-to-pixel will change with the screen density, but not necessarily in direct proportion. "dip" and "dp" are same.
- **sp (Scale-independent Pixels)** - Similar to dp unit, but also scaled by the user's font size preference.

NOTE: Always use sp for font sizes and dip for everything else.

Android Drawable Resources

P P SAVANI UNIVERSITY

- A drawable resource represents a graphic file or xml drawable file that can be drawn on screen. In Android, these files are stored in res/drawable folder.
- To prevent blurred images, drawable resources for different screen resolutions are provided in screen resolution specific drawable folders such as drawable-mdpi, drawable-hdpi, drawable-xhdpi and drawable-xxhdpi.
- Different types of drawables are bitmap, xml bitmap, nine-patch, layer list, state list, level list, transition drawable, clip drawable, scale drawable and shape drawable.

Bitmap

P P SAVANI UNIVERSITY

- A bitmap (or raster graphic) is **a digital image composed of a matrix of dots**. When viewed at 100%, each dot corresponds to an individual pixel on a display. In a standard bitmap image, each dot can be assigned a different color.**Bitmap files are .png, .jpg and .gif. The preferred bitmap file for android is .png. Bitmap file needs to be saved in res/drawable folder and can be accessed from xml and java code.**

```
<ImageButton android:id="@+id/panda"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:src="@drawable/panda"/>
```

```
((ImageButton)findViewById(R.id.panda)).setImageResource(R.drawable.panda);
```

```
Drawable panda = getResources().getDrawable(R.drawable.panda);
```

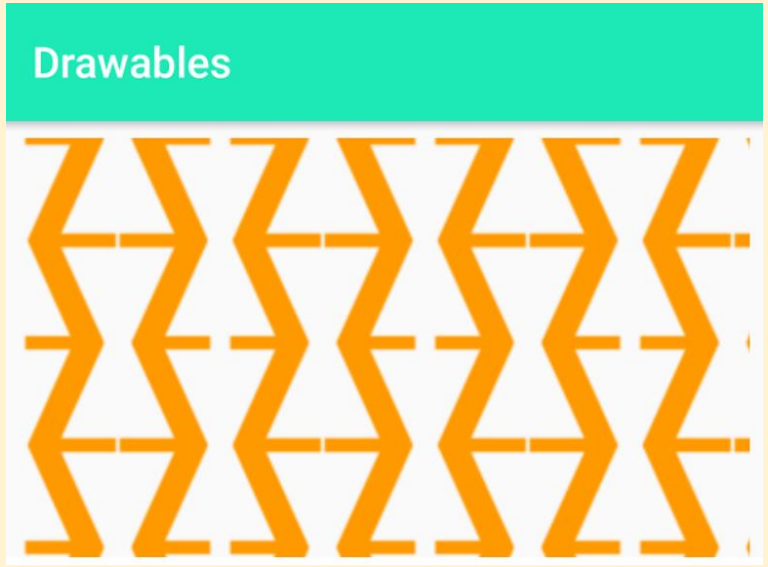
XML Bitmap

P P SAVANI UNIVERSITY

- You can add properties to bitmap in xml and use it as drawable. Some properties which can be added to bitmap are dither, used when image and screen pixel don't match, filter, used to smooth bitmap appearance when it's stretched or shrunken, and tile mode, used to repeat the image.

```
<?xml version="1.0" encoding="utf-8"?>
<bitmap xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@drawable/zoftino"
    android:tileMode="mirror" />
```

```
<ImageView
   android:id="@+id/img"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@drawable/xml_bitmap"/>
```



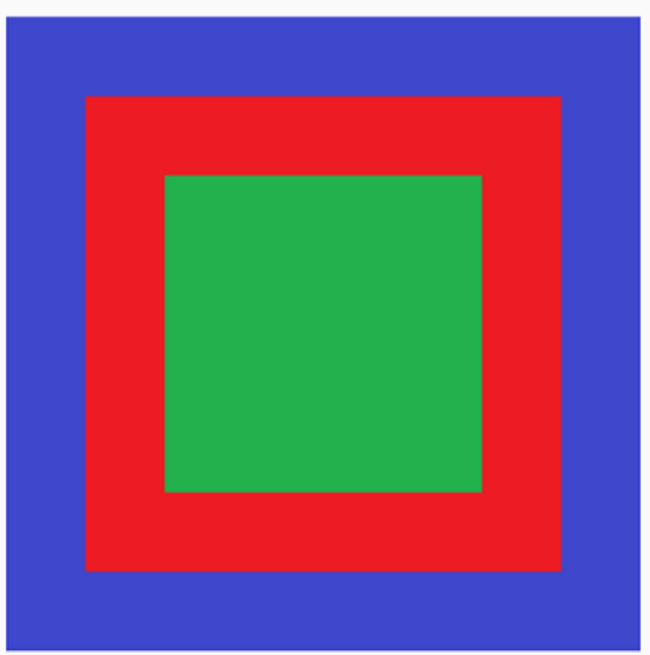
Layer List

P P SAVANI UNIVERSITY

- **You can define a list of drawables in xml and use it as drawable resource in your android app. This drawable is called layer list drawable. Layer list drawable draws drawable items in the order they are defined in the xml and each drawable item is drawn on top of previous drawable item. The last drawable item is drawn on the**

```xml
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <bitmap android:src="@drawable/square_blue"
            android:gravity="center" />
    </item>
    <item>
        <bitmap android:src="@drawable/square_red"
            android:gravity="center" />
    </item>
    <item>
        <bitmap android:src="@drawable/square_green"
            android:gravity="center" />
    </item>
</layer-list>
```

```xml
<ImageView
    android:id="@+id/img"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:src="@drawable/layer_list"/>
```



Transition Drawable

P P SAVANI UNIVERSITY

- **Transition drawable allows you to define two drawable items in xml and it transitions from first drawable to second drawable on calling startTransition method on it**

```xml
 <?xml version="1.0" encoding="utf-8"?>
<transition xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/square_red" />
    <item android:drawable="@drawable/square_green" />
</transition>
```

```
img = findViewById(R.id.img);
findViewById(R.id.button).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        TransitionDrawable drawable = (TransitionDrawable) img.getDrawable();
        drawable.startTransition(1000);
    }
});
```

Scale Drawable

P P SAVANI UNIVERSITY

- **Using scale drawable, you can specify how a drawable can be scaled depending on the level by setting scale height, scale width and scale gravity.**

```xml
<?xml version="1.0" encoding="utf-8"?>
<scale xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/zoftino"
    android:scaleGravity="center_vertical|center_horizontal"
    android:scaleHeight="60%"
    android:scaleWidth="60%" />
```

```
img = findViewById(R.id.img);
findViewById(R.id.button).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        ScaleDrawable drawable = (ScaleDrawable) img.getDrawable();
        levelInt = levelInt +800;
        drawable.setLevel(levelInt);
    }
});
```

Working with Canvases and Paints

P P SAVANI UNIVERSITY

- The android.graphics.Canvas can be used to draw graphics in android. It provides methods to draw oval, rectangle, picture, text, line etc.
- The android.graphics.Paint class is used with canvas to draw objects. It holds the information of color and style.
- The drawing of canvas happens in Bitmap, where we draw the outline and then the Paint API helps to fill color and whatever style we need.

Canvas helps to create the skeleton and Paint helps in beautifying the design.

Working with Canvases and Paints

P P SAVANI UNIVERSITY

- For 2D graphics we usually opt for any of the two following options:

1. Graphics or animation object is drawn into View object from layout.
2. We can draw graphics directly onto the canvas.

- Android Canvas class encapsulates the bitmaps used as surface. It exposes the draw methods which can be used for designing. Let us first clear the following terms:
- **Bitmap**: The surface being drawn on.
- **Paint**: It lets us specify how to draw the primitives on bitmap. It is also referred to as “Brush”.
- **Canvas**: It supplies the draw methods used to draw primitives on underlying bitmap.

Working with Canvases and Paints

P P SAVANI UNIVERSITY

- Each drawing object specifies a paint object to render:

1. drawArc: This draws an arc between the two angles bounded by an area of rectangle.
2. drawBitmap: It draws an bitmap on canvas.
3. drawRGB/drawARGB/drawColor: This fills the canvas with a single color. Alpha RGB.
4. drawBitmapMesh: It draws a bitmap using a mesh. It manipulates the appearance of target by moving points on it.
5. drawCircle: This draws a circle on a specified radius centered on a given point.
6. drawLine(s):it draws a line (or series of lines) between points.
7. drawOval: it draws an oval which is bounded by the area of rectangle.
8. drawPaint: It fills the entire canvas with a specific paint.
9. drawPath: It draws a path as per specification.
10. drawPicture: It draws a picture specified on a rectangular area.
11. drawPosText: it draws a text string specifying the offset of each character.
12. drawRect: It draws a rectangle.
13. drawRoundRect: it draws a rectangle with round edges.
14. drawText: It draws a text string on canvas.

Paint class

P P SAVANI UNIVERSITY

- The Paint class consists of a paint brush and a palette. It lets us choose how to render the primitives drawn into canvas by draw methods.
- We can control the color, style, font, special effects etc can be modified by modifying the paint object.
- For instance, setColor method can be used to select the color of Paint. Paint class supports transparency so it can be used to control variety of shades or effects, etc.

Example

P P SAVANI UNIVERSITY

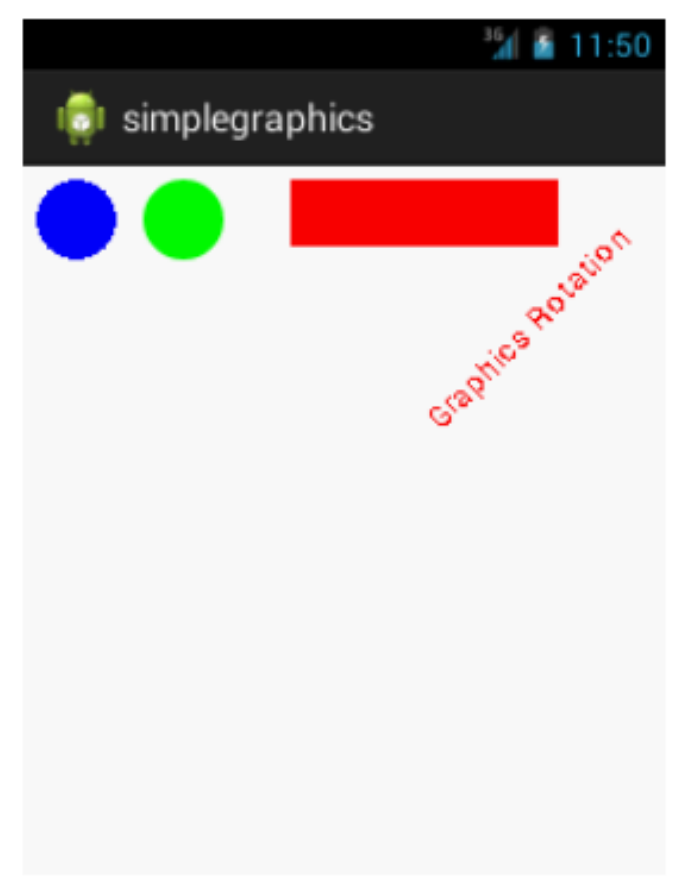
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
</RelativeLayout>
```

```
package com.android.tution.Graphics;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new CanvasView(this));
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

CanvasView.java


```java
package com.android.tution.Graphics;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Matrix;
import android.graphics.Paint;
import android.graphics.Paint.Style;
import android.view.View;
public class CanvasView extends View {
    public CanvasView(Context context) {
        super(context);
    }
    @Override
    protected void onDraw(Canvas canvas) {
        // TODO Auto-generated method stub
        super.onDraw(canvas);
        drawCirle(canvas);
        drawRotateImage(canvas);
    }
    private void drawRotateImage(Canvas canvas) {
        // TODO Auto-generated method stub
        Bitmap bitmap = BitmapFactory.decodeResource(getResources(),
                R.drawable.picture);
        bitmap = Bitmap.createScaledBitmap(bitmap, 100, 100, false);
        Matrix matrix = new Matrix();
        matrix.postRotate(70);
        matrix.postTranslate(200, 180);
        canvas.drawBitmap(bitmap, matrix, null);
    }
    private void drawCirle(Canvas canvas) {
        // TODO Auto-generated method stub
        Paint paint = new Paint();
        paint.setAntiAlias(true);
        paint.setColor(Color.YELLOW);
        paint.setStyle(Style.FILL);
        canvas.drawCircle(50, 60, 50, paint);
        paint.setStyle(Style.STROKE);
        canvas.drawCircle(200, 60, 50, paint);
```

Example




```
<RelativeLayout xmlns:androclass="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context=".MainActivity" >

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/hello_world" />

</RelativeLayout>
```

Activity class

File: MainActivity.java

P P SAVANI UNIVERSITY

```
package com.example.simplegraphics;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.view.View;

public class MainActivity extends Activity {

    DemoView demoview;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        demoview = new DemoView(this);
        setContentView(demoview);
    }

    private class DemoView extends View{
        public DemoView(Context context){
            super(context);
        }

        @Override protected void onDraw(Canvas canvas) {
            super.onDraw(canvas);

            // custom drawing code here
            Paint paint = new Paint();
            paint.setStyle(Paint.Style.FILL);

            // make the entire canvas white
            paint.setColor(Color.WHITE);
            canvas.drawPaint(paint);

            // draw blue circle with anti aliasing turned off
            paint.setAntiAlias(false);
            paint.setColor(Color.BLUE);
            canvas.drawCircle(20, 20, 15, paint);

            // draw green circle with anti aliasing turned on
            paint.setAntiAlias(true);
            paint.setColor(Color.GREEN);
            canvas.drawCircle(60, 20, 15, paint);

            // draw red rectangle with anti aliasing turned off
            paint.setAntiAlias(false);
            paint.setColor(Color.RED);
            canvas.drawRect(100, 5, 200, 30, paint);

            // draw the rotated text
            canvas.rotate(-45);

            paint.setStyle(Paint.Style.FILL);
            canvas.drawText("Graphics Rotation", 40, 180, paint);

            //undo the rotate
            canvas.restore();
        }
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

P P SAVANI
UNIVERSITY

THANK YOU