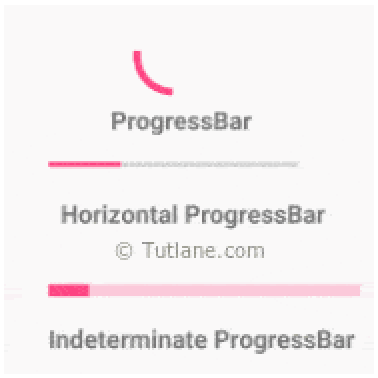
Android ProgressBar with Examples

In android, **ProgressBar** is a user interface control that is used to indicate the progress of an operation. For example, downloading a file, uploading a file.

Following is the pictorial representation of using a different type of progress bars in android applications.



By default the ProgressBar will be displayed as a spinning wheel, in case if we want to show it like a horizontal bar then we need to change the style property to horizontal like style="?android:attr/progressBarStyleHorizontal".

Create Android ProgressBar in XML Layout File

In android, we can create ProgressBar in XML layout file using <**ProgressBar**> element with different attributes like as shown below

```
<ProgressBar
   android:id="@+id/pBar3"
   style="?android:attr/progressBarStyleHorizontal"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:minHeight="50dp"
   android:minWidth="250dp"
   android:max="100"
   android:indeterminate="true"
   android:progress="1" />
```

If you observe above code snippet, we defined a progress bar (<ProgressBar>) with different attributes, those are

Attribute	Description
android:id	It is used to uniquely identify the control
android:minHeight	It is used to set the height of the progress bar.
android:minWidth	It is used to set the width of the progress bar.

Attribute	Description
android:max	It is used to set the maximum value of the progress bar.
android:progress	It is used to set the default progress value between 0 and max. It must be an integer value.

In android, the ProgressBar supports two types of modes to show the progress, those are **Determinate** and **Indeterminate**.

Android ProgressBar with Determinate Mode

Generally, we use the **Determinate** progress mode in progress bar when we want to show the quantity of progress has occurred. For example, the percentage of file downloaded, number of records inserted into a database, etc.

To use Determinate progress, we need to set the style of the progress bar to **Widget_ProgressBar_Horizontal** or **progressBarStyleHorizontal** and set the amount of progress using **android:progress** attribute.

Following is the example which shows a **Determinate** progress bar that is **50%** complete.

```
<ProgressBar
   android:id="@+id/pBar"
   style="?android:attr/progressBarStyleHorizontal"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:max="100"
   android:progress="50" />
```

By using **setProgress(int)** method, we can update the percentage of progress displayed in app or by calling **incrementProgressBy(int)** method, we can increase the value of current progress completed based on our requirements.

Generally, when the progress value reaches **100** then the progress bar is full. By using **android:max** attribute we can adjust this default value.

Android ProgressBar with Indeterminate Mode

Generally, we use the **Indeterminate** progress mode in progress bar when we don't know how long an operation will take or how much work has done.

In indeterminate mode the actual progress will not be shown, only the cyclic animation will be shown to indicate that some progress is happing like as shown in the above progress bar loading images.

By using progressBar.**setIndeterminate(true)** in activity file programmatically or using **android:indeterminate = "true"** attribute in XML layout file, we can enable **Indeterminate** progress mode.

Following is the example to set **Indeterminate** progress mode in an XML layout file.

```
<ProgressBar
    android:id="@+id/progressBar1"
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:indeterminate="true"/>
```

This is how we can define the Progress modes in **ProgressBar** based on our requirements in android applications.

Android ProgressBar Control Attributes

The following are some of the commonly used attributes related to **ProgressBar** control in android applications.

Attribute	Description
android:id	It is used to uniquely identify the control
android:max	It is used to specify the maximum value of the progress can take
android:progress	It is used to specify default progress value.
android:background	It is used to set the background color for a progress bar.
android:indeterminate	It is used to enable the indeterminate progress mode.
android:padding	It is used to set the padding for left, right, top or bottom of a progress bar.

Android ProgressBar Example

Following is the example of defining one ProgressBar control, one TextView control and one Button control in RelativeLayout to start showing the progress in the progress bar on Button click in the android application.

Create a new android application using android studio and give names as **ProgressBarExample**.

Now open an **activity_main.xml** file from **\res\layout** path and write the code like as shown below

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent">
    <ProgressBar
        android:id="@+id/pBar"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:layout_marginTop="200dp"
        android:minHeight="50dp"
        android:minWidth="200dp"
        android:max="100"
        android:indeterminate="false"
        android:progress="0" />
    <TextView
        android:id="@+id/tView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/pBar"
        android:layout_below="@+id/pBar" />
    <Button
        android:id="@+id/btnShow"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="130dp"
        android:layout_marginTop="20dp"
        android:text="Start Progress"
        android:layout_below="@+id/tView"/>
</RelativeLayout>
```

If you observe above code we created a one Progress control, one TextView control and one Button control in XML Layout file.

Once we are done with the creation of layout with required controls, we need to load the XML layout resource from our activity **onCreate()** callback method, for that open main activity file **MainActivity.java** from **\java\com.tutlane.progressbarexample** path and write the code like as shown below.

MainActivity.java

```
package com.tutlane.progressbarexample;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    private ProgressBar pgsBar;
    private int i = 0;
    private TextView txtView;
    private Handler hdlr = new Handler();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        pgsBar = (ProgressBar) findViewById(R.id.pBar);
        txtView = (TextView) findViewById(R.id.tView);
        Button btn = (Button)findViewById(R.id.btnShow);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                i = pgsBar.getProgress();
                new Thread(new Runnable() {
```

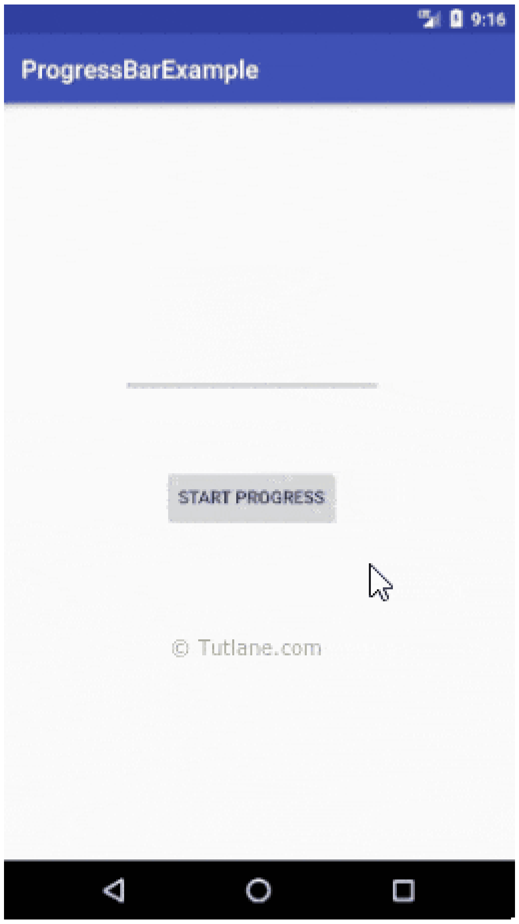
```
                public void run() {
                    while (i < 100) {
                        i += 1;
                        // Update the progress bar and display the current value in text view
                        hdlr.post(new Runnable() {
                            public void run() {
                                pgsBar.setProgress(i);
                                txtView.setText(i+"/"+pgsBar.getMax());
                            }
                        });
                        try {
                            // Sleep for 100 milliseconds to show the progress slowly.
                            Thread.sleep(100);
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                }
            }).start();
        }
    });
  }
}
```

If you observe above code we are calling our layout using **setContentView** method in the form of **R.layout.layout_file_name** in our activity file. Here our xml file name is **activity_main.xml** so we used file name **activity_main** and we are trying to show the progress of task in progress bar on Button click.

Generally, during the launch of our activity, the onCreate() callback method will be called by the android framework to get the required layout for an activity.

Output of Android ProgressBar Example

When we run the above example using an android virtual device (AVD) we will get a result like as shown below.



If you observe the above result, we are able to start showing the progress of the task in **progress bar** when we click on Button in the android application.

This is how we can use **ProgressBar** control in android applications to show the progress of tasks or work based on our requirements.