1. **Android Notifications**

(BigTextStyle, BigPictureStyle, InboxStyle)

1. Which class, object and properties are used for notification? Write a sample code using these.

To create a notification, we are going to use **NotificationCompat** class to implement a notification in our android application. We need to specify the UI content and required actions with a **NotificationCompat.Builder** object. To display an icon, title and detailed text of notification we need to set following properties in Builder object.

**setSmallIcon()** - It is used to set the small icon for our notification.

**setContentTitle()** - It is used to set the title of our notification.

**setContentText()** - It is used to set the detailed text to display in notification.

The above mentioned properties are necessary to display a notification and we can set a different type of properties to our notification like setStyle, setSound, setLights, setLargeIcon, etc. based on our requirements using Builder object.

1. **Explain the activities of NotificationCompat, PendingIntent and Notificationmanager** methods.

The **NotificationCompat** class supports a different type of notification views, such as normal view, big view and it provides a best support for a wide range of platforms. In android, we can define an action inside of notification by using **PendingIntent** object which contains an [Intent](https://www.tutlane.com/tutorial/android/android-intents-implicit-explicit) that starts an [Activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle) of our app. However, once we are done with creation of notification, we need to pass a notification to the system by using **NotificationManager.notify()** method and we need to specify a **ID** in the notification to use this **ID** to update a notification later if required.

1. What are the different notification style?

Android Big Text Style Notification

Android Inbox Style Notification

Android Big Picture Style Notification

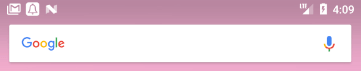
**Extra**

In android, **Notification** is a message which is used to alert the users about some events that happening in our app.

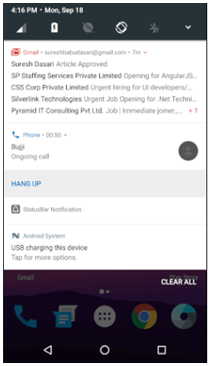
Generally, the android Notifications will be displayed outside of our app’s normal UI and alert the users without interrupting their current activities.

In android, we can alert the users about our app notifications in different forms like flash the LED or make a sounds or display an icon in status bar, etc.

When we tell the system to issue a notification, first it will display an icon in notification bar like as shown below.



To see the details of our android app notification, we need to open the notification drawer like as shown below.



Now we will see how to create and issue a notifications in android applications with examples.

Here we are going to use **NotificationCompat** class to implement a notification in our android application.

## **Create a Notification in Android**

To create a notification, we need to specify the UI content and required actions with a **NotificationCompat.Builder** object. To display an icon, title and detailed text of notification we need to set following properties in Builder object.

* **setSmallIcon()** - It is used to set the small icon for our notification.
* **setContentTitle()** - It is used to set the title of our notification.
* **setContentText()** - It is used to set the detailed text to display in notification.

The above mentioned properties are necessary to display a notification and we can set a different type of properties to our notification like setStyle, setSound, setLights, setLargeIcon, etc. based on our requirements using Builder object.

Following is the example of creating a notification using **NotificationCompat.Build** object and setting the notification properties.

|  |
| --- |
| NotificationCompat.Builder nBuilder =  new NotificationCompat.Builder(this)                 .setSmallIcon(R.drawable.notification\_icon)                 .setContentTitle("Sample notification")                 .setContentText("Hi, Welcome to Sarker.com"); |

## **Define the Android Notification Actions**

If we assign an action to the notification, it will allow users to go directly from the notification to an [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle) of our app. We can also add buttons to the notification to perform an additional actions such as hang up the call or responding immediately to a text message; this feature is available as of Android 4.1.

Following is the example of defining an action inside of notification using **PendingIntent** object.

|  |
| --- |
| NotificationCompat.Builder nBuilder =  new NotificationCompat.Builder(this) …... Intent resultIntent = new Intent(this, MainActivity.class); PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, resultIntent, 0); nBuilder.setContentIntent(pendingIntent); |

## **Issue the Android Notification**

Following is the example of sending a notification to the system using

**Notificationmanager.notify** method.

|  |
| --- |
| NotificationCompat.Builder nBuilder =  new NotificationCompat.Builder(this); .... int mNotificationId = 999; NotificationManager mNotifyMgr = (NotificationManager)getSystemService(NOTIFICATION\_SERVICE); // Builds the notification and issues it. mNotifyMgr.notify(mNotificationId, nBuilder.build()); |

Now we will see how to create and show the notification in android application with example.

## **Android Notification Example**

Following is the example of implementing a **Notifications** in android application.

Create a new android application using android studio and give names as **NotificationExample-1**.

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

**Project Name: NotificationExample-1**

## **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="match\_parent"     android:orientation="vertical" >      <Button         android:id="@+id/btnShow"         android:layout\_width="wrap\_content"         android:layout\_height="wrap\_content"         android:text="Show Notification"         android:layout\_marginTop="200dp"  android:layout\_marginLeft="100dp"/> </LinearLayout> |

If we observe above code we created a one [Button](https://www.tutlane.com/tutorial/android/android-button-with-examples) control in XML Layout file to show the notification in notification bar when we click on [Button](https://www.tutlane.com/tutorial/android/android-button-with-examples).

Once we are done with creation of layout with required controls, we need to load the XML layout resource from our [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle) **onCreate()** callback method, for that open main activity file **MainActivity.java** from **\java\com.sarker.notificationexample** path and write the code like as shown below.

## **MainActivity.java**

|  |
| --- |
| package com.sarker.notificationexample-1; import android.app.NotificationManager; import android.app.PendingIntent; import android.content.Intent; import android.support.v4.app.NotificationCompat; import android.support.v7.app.AppCompatActivity; import android.os.Bundle; import android.view.View; import android.widget.Button;  public class MainActivity extends AppCompatActivity {      @Override     protected void onCreate(Bundle savedInstanceState) {         super.onCreate(savedInstanceState);         setContentView(R.layout.activity\_main);         Button btnNotify = (Button)findViewById(R.id.btnShow);         btnNotify.setOnClickListener(new View.OnClickListener() {             @Override             public void onClick(View v) {                 NotificationCompat.Builder mBuilder =  new NotificationCompat.Builder(MainActivity.this)                                 .setSmallIcon(R.drawable.ic\_notification)                                 .setContentTitle("Sarker Send New Message")                                 .setContentText("Hi, Welcome to Sarker Notes");                 // Set the intent to fire when the user taps on notification.                 Intent resultIntent = new Intent(MainActivity.this, MainActivity.class);                 PendingIntent pendingIntent = PendingIntent.getActivity(MainActivity.this, 0, resultIntent, 0);                 mBuilder.setContentIntent(pendingIntent);                 // Sets an ID for the notification                 int mNotificationId = 001;                 NotificationManager notificationManager = (NotificationManager) getSystemService(NOTIFICATION\_SERVICE);                 // It will display the notification in notification bar                 notificationManager.notify(mNotificationId, mBuilder.build());             }         });     } } |

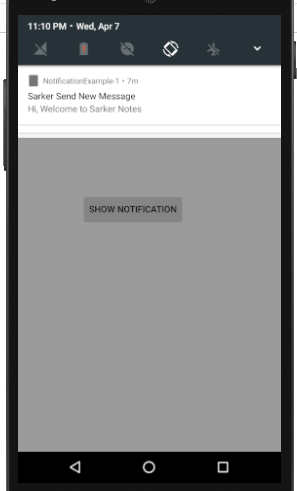
If we observe above code we are creating a notification, adding action inside of notification using [intent](https://www.tutlane.com/tutorial/android/android-intents-implicit-explicit) object to open the [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle) in our app and showing notification on [Button](https://www.tutlane.com/tutorial/android/android-button-with-examples) click using **NotificationManager**.

Here we added an **ic\_notification** image in **drawable** folder to show it as a notification icon so please add a required image in your **drawable** folder and use it in our application.

Generally, during the launch of our [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle), **onCreate()** callback method will be called by android framework to get the required layout for an [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle).

## **Output of Android Notification Example**

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



 If we observe above result we created a notification and shown it on Button click using NotificationCompat.Builder based on our requirements.

## **Android Big Text Style Notification Example**

Create a new android application using android studio and give names as **NotificationExample-2**.

We learned how to show the android notification in normal view, in case if we want to show the large icon and large text in notification, then that can be achieved using **NotificationCompat.BigTextStyle**.

**Project Name: NotificationExample-2**

## **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="match\_parent"     android:orientation="vertical" >      <Button         android:id="@+id/btnShow"         android:layout\_width="wrap\_content"         android:layout\_height="wrap\_content"         android:text="Show Notification"         android:layout\_marginTop="200dp"  android:layout\_marginLeft="100dp"/> </LinearLayout> |

We need to modify our main [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle) file **MainActivity.java** like as shown below.

## **MainActivity.java**

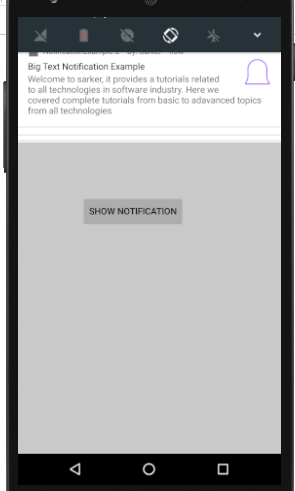
|  |
| --- |
| package com.sarker.notificationexample-2; import android.app.NotificationManager; import android.app.PendingIntent; import android.content.Intent; import android.graphics.Bitmap; import android.graphics.BitmapFactory; import android.support.v4.app.NotificationCompat; import android.support.v7.app.AppCompatActivity; import android.os.Bundle; import android.view.View; import android.widget.Button;  public class MainActivity extends AppCompatActivity {     @Override     protected void onCreate(Bundle savedInstanceState) {         super.onCreate(savedInstanceState);         setContentView(R.layout.activity\_main);         Button btnNotify = (Button)findViewById(R.id.btnShow);         btnNotify.setOnClickListener(new View.OnClickListener() {             @Override             public void onClick(View v) {                 //To set large icon in notification                 Bitmap licon = BitmapFactory.decodeResource(getResources(), R.drawable.ic\_notification);                 //Assign BigText style notification                 NotificationCompat.BigTextStyle bigText = new NotificationCompat.BigTextStyle();                 bigText.bigText("Welcome to sarker, it provides a tutorials related to all technologies in software industry. Here we covered complete tutorials from basic to adavanced topics from all technologies");                 bigText.setSummaryText("By: Sarker");                  NotificationCompat.Builder mBuilder =  new NotificationCompat.Builder(MainActivity.this)                                 .setSmallIcon(R.drawable.ic\_notification)                                 .setContentTitle("Big Text Notification Example")                                 .setLargeIcon(licon)                                 .setStyle(bigText);                 // Set the intent to fire when the user taps on notification.                 Intent resultIntent = new Intent(MainActivity.this, MainActivity.class);                 PendingIntent pendingIntent = PendingIntent.getActivity(MainActivity.this, 0, resultIntent, 0);                 mBuilder.setContentIntent(pendingIntent);                 // Sets an ID for the notification                 int mNotificationId = 001;                 NotificationManager notificationManager = (NotificationManager) getSystemService(NOTIFICATION\_SERVICE);                 // It will display the notification in notification bar                 notificationManager.notify(mNotificationId, mBuilder.build());             }         });     } } |

If we observe above code we are creating a big text style notification using **NotificationCompat.BigTextStyle** and appended to notification using **setStyle()** property.

Here we added a **ic\_notification**, **notification\_icon** (**bmp**) images in **drawable** folder to show it in notification icon so please add a required images in your **drawable** folder and use it in your application.

## **Output of Android Big Text Style Notification Example**

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



If we observe above result we created a big text style notification with large image and shown it on [Button](https://www.tutlane.com/tutorial/android/android-button-with-examples) click based on our requirements.

## **Android Inbox Style Notification Example**

Create a new android application using android studio and give names as **NotificationExample-3**.

**Project Name: NotificationExample-3**

## **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="match\_parent"     android:orientation="vertical" >      <Button         android:id="@+id/btnShow"         android:layout\_width="wrap\_content"         android:layout\_height="wrap\_content"         android:text="Show Notification"         android:layout\_marginTop="200dp"  android:layout\_marginLeft="100dp"/> </LinearLayout> |

By using **NotificationCompat.InboxStyle** object we can implement inbox style notification.

We need to modify our main [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle) file **MainActivity.java** like as shown below.

## **MainActivity.java**

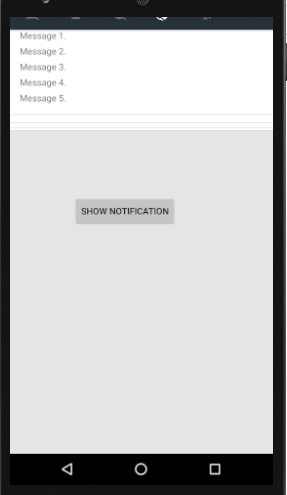
|  |
| --- |
| package com.sarker.notificationexample-3; import android.app.NotificationManager; import android.app.PendingIntent; import android.content.Intent; import android.support.v4.app.NotificationCompat; import android.support.v7.app.AppCompatActivity; import android.os.Bundle; import android.view.View; import android.widget.Button;  public class MainActivity extends AppCompatActivity {      @Override     protected void onCreate(Bundle savedInstanceState) {         super.onCreate(savedInstanceState);         setContentView(R.layout.activity\_main);         Button btnNotify = (Button)findViewById(R.id.btnShow);         btnNotify.setOnClickListener(new View.OnClickListener() {             @Override             public void onClick(View v) {                 //Implement inbox style notification                 NotificationCompat.InboxStyle iStyle =  new NotificationCompat.InboxStyle();                 iStyle.addLine("Message 1.");                 iStyle.addLine("Message 2.");                 iStyle.addLine("Message 3.");                 iStyle.addLine("Message 4.");                 iStyle.addLine("Message 5.");                 iStyle.setSummaryText("+2 more");                  NotificationCompat.Builder mBuilder =  new NotificationCompat.Builder(MainActivity.this)                                 .setSmallIcon(R.drawable.ic\_notification)                                 .setContentTitle("Inbox Style Notification Example")                                 .setStyle(iStyle);                 // Set the intent to fire when the user taps on notification.                 Intent resultIntent = new Intent(MainActivity.this, MainActivity.class);                 PendingIntent pendingIntent = PendingIntent.getActivity(MainActivity.this, 0, resultIntent, 0);                 mBuilder.setContentIntent(pendingIntent);                 // Sets an ID for the notification                 int mNotificationId = 001;                 NotificationManager notificationManager = (NotificationManager) getSystemService(NOTIFICATION\_SERVICE);                 // It will display the notification in notification bar                 notificationManager.notify(mNotificationId, mBuilder.build());             }         });     } } |

If we observe above code we are creating a inbox style notification using **NotificationCompat.InboxStyle** and appended to notification using **setStyle()** property.

Here we added a **ic\_notification** in **drawable** folder to show it in notification icon so please add a required images in your **drawable** folder and use it in your application.

## **Output of Android Inbox Style Notification Example**

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



If we observe above result we created an inbox style notification and shown it on [Button](https://www.tutlane.com/tutorial/android/android-button-with-examples) click based on our requirements.

## **Android Big Picture Notification Example**

Create a new android application using android studio and give names as **NotificationExample-4**.

**Project Name: NotificationExample-4**

## **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="match\_parent"     android:orientation="vertical" >      <Button         android:id="@+id/btnShow"         android:layout\_width="wrap\_content"         android:layout\_height="wrap\_content"         android:text="Show Notification"         android:layout\_marginTop="200dp"  android:layout\_marginLeft="100dp"/> </LinearLayout> |

By using **NotificationCompat.BigPictureStyle** object we can implement inbox style notification.

We need to modify our main [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle) file **MainActivity.java** like as shown below.

## **MainActivity.java**

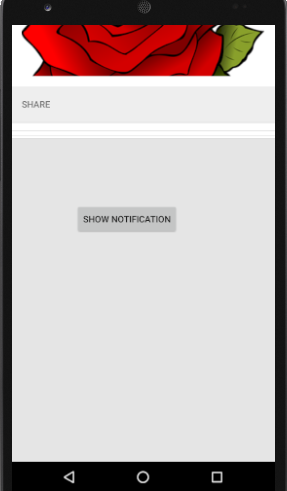
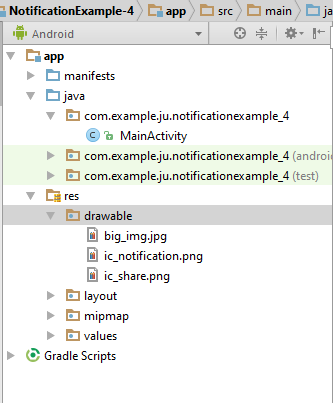
|  |
| --- |
| package com.sarker.notificationexample-4; import android.app.NotificationManager; import android.app.PendingIntent; import android.content.Intent; import android.graphics.BitmapFactory; import android.net.Uri; import android.support.v4.app.NotificationCompat; import android.support.v7.app.AppCompatActivity; import android.os.Bundle; import android.view.View; import android.widget.Button;  public class MainActivity extends AppCompatActivity {      @Override     protected void onCreate(Bundle savedInstanceState) {         super.onCreate(savedInstanceState);         setContentView(R.layout.activity\_main);         Button btnNotify = (Button)findViewById(R.id.btnShow);         btnNotify.setOnClickListener(new View.OnClickListener() {             @Override             public void onClick(View v) {                 // Assign big picture notification                 NotificationCompat.BigPictureStyle bpStyle = new NotificationCompat.BigPictureStyle();                 bpStyle.bigPicture(BitmapFactory.decodeResource(getResources(), R.drawable.big\_img)).build();                 // Set the intent to fire when the user taps on notification.                 Intent rIntent = new Intent(Intent.ACTION\_VIEW, Uri.parse("http://sarker.com/"));                 PendingIntent pendingIntent = PendingIntent.getActivity(MainActivity.this, 0, rIntent, 0);                 NotificationCompat.Builder mBuilder =  new NotificationCompat.Builder(MainActivity.this)                                 .setSmallIcon(R.drawable.ic\_notification)                                 .setContentTitle("Big Picture Notification Example")                                 .addAction(R.drawable.ic\_share, "Share", pendingIntent)                                 .setStyle(bpStyle);                  mBuilder.setContentIntent(pendingIntent);                 // Sets an ID for the notification                 int mNotificationId = 001;                 NotificationManager notificationManager = (NotificationManager) getSystemService(NOTIFICATION\_SERVICE);                 // It will display the notification in notification bar                 notificationManager.notify(mNotificationId, mBuilder.build());             }         });     } } |

If we observe above code we are creating a inbox style notification using **NotificationCompat.BigPicture** and appended to notification using **setStyle()** property.

Here we added a **ic\_share, big\_img** images in **drawable** folder to show it in notification so please add a required images in your **drawable** folder and use it in your application.

## **Output of Android Big Picture Style Notification Example**

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

If we observe above result we created a big picture style notification and shown it on [Button](https://www.tutlane.com/tutorial/android/android-button-with-examples) click based on our requirements.

This is how we can create and show the notifications in android applications based on our requirements.

**2. Android Toast with Examples**

1. What is Toast and Toast method? What is its three parameters and display method?

**Toast** is a small popup notification which is used to display an information about the operation which we performed in our app. The Toast will show the message for a small period of time and it will disappear automatically after a timeout.

Generally, the size of Toast will be adjusted based on the space required for the message and it will be displayed on the top of main content of an [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle) for a short period of time.

| **Parameter** | **Description** |
| --- | --- |
| context | It’s our application context. |
| message | It’s our custom message which we want to show in Toast notification. |
| duration | It is used to define the duration for notification to display on screen. |

In android, we can create a Toast by instantiating an **android.widget.Toast** object using **makeText()** method. The **makeText()** method will take three parameters: application context, text message and the duration for the toast. We can display the **Toast notification** by using **show()** method.

1. What are the methods to change the position of Toast notification and three parameter for it?

By default, the android Toast notification will always appear near the bottom of the screen, centred horizontally.

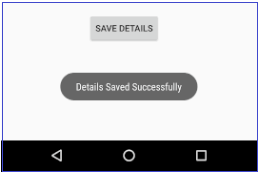
In case if we want to change the position of Toast notification, we can do it by using

**setGravity(int, int, int)** method. The **setGravity()** method will accepts three parameters: a **Gravity** constant, an **x-position** offset, and a **y-position** offset.

EXTRA

For example, some of the apps will show a message like “**Press again to exit**” in toast, when we pressed a back button in the home page or showing a message like “**saved successfully**” toast, when we click on button to save the details.

Following is the pictorial representation of using **Toast** in android applications.



**Create a Toast in Android**

Following is the syntax of creating a **Toast** in android applications.

|  |
| --- |
| Toast.makeText(context, "message", duration).show(); |

If we observe above syntax, we defined a Toast **notification** using **makeText()** method with three parameters, those are

We have a two ways to define the Toast **duration**, either in **LENGTH\_SHORT** or

**LENGTH\_LONG** to display the toast notification for short or longer period of time.

Following is the example of defining a Toast in android applications.

|  |
| --- |
| Toast.makeText(MainActivity.this, "Details Saved Successfully.", Toast.LENGTH\_SHORT).show(); |

Now we will see how to implement a Toast notification in android applications with examples.

**Android Toast Notification Example**

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

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"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"     android:layout\_width="match\_parent"     android:layout\_height="match\_parent"     android:orientation="vertical" >      <Button         android:id="@+id/btnShow"         android:layout\_width="wrap\_content"         android:layout\_height="wrap\_content"         android:text="Show Toast"         android:layout\_marginTop="200dp"  android:layout\_marginLeft="140dp"/> </LinearLayout> |

If we observe above code we created a one [Button](https://www.tutlane.com/tutorial/android/android-button-with-examples) control in XML Layout file to show the toast **notification** when we click on [Button](https://www.tutlane.com/tutorial/android/android-button-with-examples).

Once we are done with creation of layout with required controls, we need to load the XML layout resource from our [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle) **onCreate()** callback method, for that open main [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle) file

**MainActivity.java** from **\java\com.sarker.toastexample** path and write the code like as shown below.

**MainActivity.java**

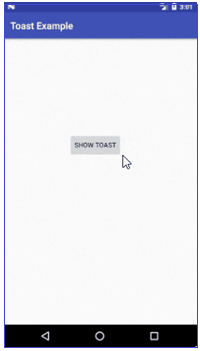
|  |
| --- |
| package com.sarker.toastexample; import android.support.v7.app.AppCompatActivity; import android.os.Bundle; import android.view.View; import android.widget.Button; import android.widget.Toast;  public class MainActivity extends AppCompatActivity {      @Override     protected void onCreate(Bundle savedInstanceState) {         super.onCreate(savedInstanceState);         setContentView(R.layout.activity\_main);         Button btn = (Button)findViewById(R.id.btnShow);         btn.setOnClickListener(new View.OnClickListener() {             @Override             public void onClick(View v) {                 Toast.makeText(MainActivity.this, "You Clicked on Button..", Toast.LENGTH\_SHORT).show();             }         });     } } |

If we observe above code we are created a toast notification using **makeText()** method and showing a toast notification on [Button](https://www.tutlane.com/tutorial/android/android-button-with-examples) click.

Generally, during the launch of our [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle), **onCreate()** callback method will be called by android framework to get the required layout for an [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle).

**Output of Android Toast Notification Example**

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



If we observe above result we created a toast notification and shown it on Button click based on our requirements.

**Change the Position of Android Toast Notification**

**Android Toast Positioning Example**

Following is the example of changing the position of android toast notification to top-right side using **setGravity()**method.

We need to modify our main [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle) file **MainActivity.java** code like as shown below.

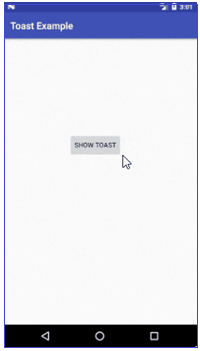
**MainActivity.java**

|  |
| --- |
| package com.sarker.toastexample; import android.support.v7.app.AppCompatActivity; import android.os.Bundle; import android.view.Gravity; import android.view.View; import android.widget.Button; import android.widget.Toast;  public class MainActivity extends AppCompatActivity {      @Override     protected void onCreate(Bundle savedInstanceState) {         super.onCreate(savedInstanceState);         setContentView(R.layout.activity\_main);         Button btn = (Button)findViewById(R.id.btnShow);         btn.setOnClickListener(new View.OnClickListener() {             @Override             public void onClick(View v) {                 //display toast message in top right side               Toast toast =  Toast.makeText(MainActivity.this, "You Clicked on Button..", Toast.LENGTH\_SHORT);                 toast.setGravity(Gravity.TOP|Gravity.RIGHT, 100, 250);                 toast.show();             }         });     } } |

If we observe above code we are changing the position of android toast notification using **setGravity()** property.

**Output of Android Toast Positioning Example**

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



If we observe above result we changed the position of toast notification to top right side on [Button](https://www.tutlane.com/tutorial/android/android-button-with-examples) click based on our requirements.

In case if we want to change the style of toast notification, then we can do it by creating a custom XML layout file.

**3. Android Custom Toast with Examples**

1. What is custom Toast and its method?

Generally, the size of Toast will be adjusted based on the space required for the message and it will be displayed on the top of main content of an [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle) for a short period of time.

Generally, the Toast notification in android will be displayed with simple text like as shown in above image. In android, we can customize the layout of our toast notification to change the appearance of based on requirements like include images in toast notification or change the background color of toast notification, etc.

To customize the appearance of Toast notification, we need to create a custom layout in our XML or application code and pass the root View object to the **setView(View)** method.

To create a custom Toast notification in android, we need to define a custom [View](https://www.tutlane.com/tutorial/android/android-view-and-viewgroup-with-examples) layout in XML, for that create a custom XML file (**custom\_toast.xml**) in layout (**/layout**) folder.

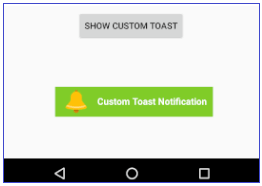
1. Write the role of Inflate and configure it for Toast.

To use this custom layout as [Toast](https://www.tutlane.com/tutorial/android/android-toast-with-examples) notification in our android application, we need to inflate the layout. We create an instance of **LayoutInflater** with **getLayoutInflater()**, and then inflate our XML layout using **inflate(int, ViewGroup)**. Here the first parameter is the **layout resource ID** and the second is the **root View** and this inflated layout will help us to find the [View](https://www.tutlane.com/tutorial/android/android-view-and-viewgroup-with-examples) objects in the layout. After that we created a new [Toast](https://www.tutlane.com/tutorial/android/android-toast-with-examples) with **Toast(Context)** and set required properties of the [toast](https://www.tutlane.com/tutorial/android/android-toast-with-examples), then we call **setView(View)** and pass it to the inflated layout. [… code: inflate code, below, page 2]

Once we are done with required configurations, then we can show the custom toast notification by calling **show()** method.

In android, [**Toast**](https://www.tutlane.com/tutorial/android/android-toast-with-examples) is a small popup notification which is used to display an information about the operation which we performed in our app. The Toast will show the message for a small period of time and it will disappear automatically after a timeout.

Following is the pictorial representation of using **Custom Toast** notification in android applications.



**Create a Custom Toast in Android**

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"     android:id="@+id/custom\_toast\_container"     android:orientation="horizontal"     android:layout\_width="match\_parent"     android:layout\_height="match\_parent"     android:paddingLeft="10dp"     android:paddingRight="10dp"     android:background="#80CC28">     <ImageView android:src="@drawable/ic\_notification"         android:layout\_width="wrap\_content"         android:layout\_height="wrap\_content"         android:layout\_marginRight="10dp" />     <TextView android:id="@+id/txtvw"         android:layout\_width="wrap\_content"         android:layout\_height="wrap\_content"         android:layout\_marginTop="13dp"         android:textColor="#FFF"         android:textStyle="bold"         android:textSize="15dp" /> </LinearLayout> |

**Inflate code.**

|  |
| --- |
| LayoutInflater inflater = getLayoutInflater(); View layout = inflater.inflate(R.layout.custom\_toast, (ViewGroup) findViewById(R.id.custom\_toast\_layout)); TextView tv = (TextView) layout.findViewById(R.id.txtvw); tv.setText("Custom Toast Notification"); Toast toast = new Toast(getApplicationContext()); toast.setDuration(Toast.LENGTH\_LONG); toast.setView(layout); toast.show(); |

If we observe above code,

Now we will see how to implement a custom Toast [notification](https://www.tutlane.com/tutorial/android/android-notifications-bigtextstyle-bigpicturestyle-inboxstyle) in android applications with examples.

**Android Custom Toast Example**

Create a new android application using android studio and give names as **ToastExample**. In case if you are not aware of creating an app in android studio check this article [Android Hello World App](https://www.tutlane.com/tutorial/android/android-hello-world-app-example).

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"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"     android:layout\_width="match\_parent"     android:layout\_height="match\_parent"     android:orientation="vertical" >      <Button         android:id="@+id/btnShow"         android:layout\_width="wrap\_content"         android:layout\_height="wrap\_content"         android:text="Show Custom Toast"         android:layout\_marginTop="150dp"  android:layout\_marginLeft="110dp"/> </LinearLayout> |

If we observe above code we created a one [Button](https://www.tutlane.com/tutorial/android/android-button-with-examples) control in XML Layout file to show the custom toast [notification](https://www.tutlane.com/tutorial/android/android-notifications-bigtextstyle-bigpicturestyle-inboxstyle) when we click on [Button](https://www.tutlane.com/tutorial/android/android-button-with-examples).

Now we need to create a custom layout for our toast notification, for that create a new xml file (**custom\_toast.xml**) in **/layout** folder and write the code like as shown below.

**Custom\_toast.xml**

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"     android:id="@+id/custom\_toast\_layout"     android:orientation="horizontal"     android:layout\_width="match\_parent"     android:layout\_height="match\_parent"     android:paddingLeft="10dp"     android:paddingRight="10dp"     android:background="#80CC28">      <ImageView android:src="@drawable/ic\_notification"         android:layout\_width="wrap\_content"         android:layout\_height="wrap\_content"         android:layout\_marginRight="10dp" />     <TextView android:id="@+id/txtvw"         android:layout\_width="wrap\_content"         android:layout\_height="wrap\_content"         android:layout\_marginTop="13dp"         android:textColor="#FFF"         android:textStyle="bold"         android:textSize="15dp" /> </LinearLayout> |

If we observe above code we are loading image (**ic\_notification**) from **drawable** folder so we need to add your icon in **drawable** folder to show it in notification.

Once we are done with creation of layout with required controls, we need to load the XML layout resource from our [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle) **onCreate()** callback method, for that open main activity file **MainActivity.java** from **\java\com.sarker.toastexample** path and write the code like as shown below.

**MainActivity.java**

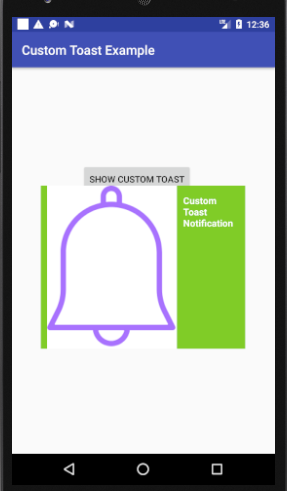
|  |
| --- |
| package com.sarker.customtoastexample; import android.support.v7.app.AppCompatActivity; import android.os.Bundle; import android.view.Gravity; import android.view.LayoutInflater; import android.view.View; import android.view.ViewGroup; import android.widget.Button; import android.widget.TextView; import android.widget.Toast;  public class MainActivity extends AppCompatActivity {     @Override     protected void onCreate(Bundle savedInstanceState) {         super.onCreate(savedInstanceState);         setContentView(R.layout.activity\_main);         Button btn = (Button)findViewById(R.id.btnShow);         btn.setOnClickListener(new View.OnClickListener() {             @Override             public void onClick(View v) {                 LayoutInflater inflater = getLayoutInflater();                 View layout = inflater.inflate(R.layout.custom\_toast, (ViewGroup) findViewById(R.id.custom\_toast\_layout));                 TextView tv = (TextView) layout.findViewById(R.id.txtvw);                 tv.setText("Custom Toast Notification");                 Toast toast = new Toast(getApplicationContext());                 toast.setGravity(Gravity.CENTER\_VERTICAL, 0, 100);                 toast.setDuration(Toast.LENGTH\_LONG);                 toast.setView(layout);                 toast.show();             }         });     } } |

If we observe above code we are calling our custom toast notification by using **LayoutInflater** object and showing a toast notification on [Button](https://www.tutlane.com/tutorial/android/android-button-with-examples) click.

Generally, during the launch of our [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle), **onCreate()** callback method will be called by android framework to get the required layout for an [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle).

**Output of Android Custom Toast Example**

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



If we observe above result we created a custom toast notification and showing it on [Button](https://www.tutlane.com/tutorial/android/android-button-with-examples) click based on our requirements.

This is how we can create custom toast notifications in android applications based on our requirements.

**4. Android Progress Notification with Examples**

1. What is Progress Notification and its types and method/class?

In android, **Progress Notification** is used to show the progress of an ongoing operation in notification bar. By using progress notification, we can easily know that how much percentage of the current operation completed and how long the operation will run to complete the remaining operation.

In android, two types of progress indicators available, one is **determinate** and another one is **indeterminate**. If we are known about how long the operation will take to complete, then we can use **determinate** form of the indicator. In case if we are not aware of how long the operation will run then we can use **indeterminate** form of indicator.

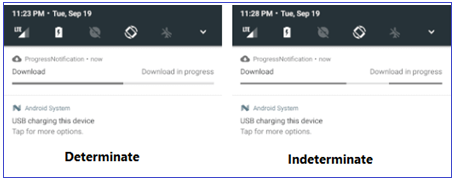
1. Explain the method setProgress(max, progress, false).

Generally, the progress indicators in android are implemented by using [ProgressBar](https://www.tutlane.com/tutorial/android/android-progressbar-with-examples) class. To display the progress indicators in our app, we need to add the [progress bar](https://www.tutlane.com/tutorial/android/android-progressbar-with-examples) to our notification by calling setProgress(max, progress, false)method and then issue the notification.

1. Write the meaning of setProgress(0,0, false).

Third argument in **setProgress()** method is used to indicate whether the [progress bar](https://www.tutlane.com/tutorial/android/android-progressbar-with-examples) is **determinate** (false) or **indeterminate** (true). Once the operation is done leave the [progress bar](https://www.tutlane.com/tutorial/android/android-progressbar-with-examples) showing or remove it by calling **setProgress(0,0, false)** and update the notification text to show that the operation is complete.

Following is the pictorial representation of using different type of android progress indicators (**determinate** and **indeterminate**) to show the progress of current operation in notification.



Here the As our operation proceeds, we need to increase the value of **progress**, and update the status of [notification](https://www.tutlane.com/tutorial/android/android-notifications-bigtextstyle-bigpicturestyle-inboxstyle). At the end of operation, the **progress** value must be equal to **max** value. The better way to call **setProgress()** is to set **max** value to **100** and then increment **progress** as a percent complete value for the operation.

Now we will see how to create and show the progress bar in android notification bar with examples.

**Android Progress Notification Example**

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

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"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"     android:layout\_width="match\_parent"     android:layout\_height="match\_parent"     android:orientation="vertical" >     <Button         android:id="@+id/btnShow"         android:layout\_width="wrap\_content"         android:layout\_height="wrap\_content"         android:text="Show Notification"         android:layout\_marginTop="100dp"  android:layout\_marginLeft="120dp"/> </LinearLayout> |

If we observe above code we created a one [Button](https://www.tutlane.com/tutorial/android/android-button-with-examples) control in XML Layout file to show the progress indicator in notification when we click on [Button](https://www.tutlane.com/tutorial/android/android-button-with-examples).

Once we are done with creation of layout with required controls, we need to load the XML layout resource from our [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle) **onCreate()** callback method, for that open main activity file **MainActivity.java** from **\java\com.sarker.progressnotification** path and write the code like as shown below.

**MainActivity.java**

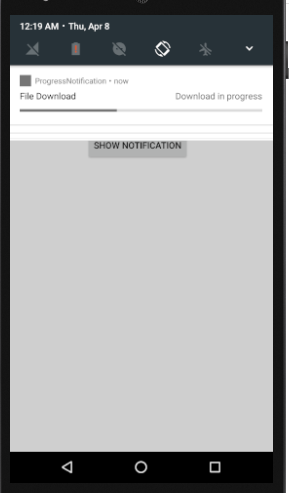
|  |
| --- |
| package com.sarker.progressnotification; import android.app.NotificationManager; import android.content.Context; import android.support.v7.app.AppCompatActivity; import android.os.Bundle; import android.support.v7.app.NotificationCompat; import android.util.Log; import android.view.View; import android.widget.Button;  public class MainActivity extends AppCompatActivity {     private NotificationManager mNotifyManager;     private NotificationCompat.Builder mBuilder;     int id = 1;     @Override     protected void onCreate(Bundle savedInstanceState) {         super.onCreate(savedInstanceState);         setContentView(R.layout.activity\_main);         Button b1 = (Button) findViewById(R.id.btnShow);         b1.setOnClickListener(new View.OnClickListener() {             @Override             public void onClick(View v) {                 mNotifyManager = (NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE);                 mBuilder = new NotificationCompat.Builder(MainActivity.this);                 mBuilder.setContentTitle("File Download")                         .setContentText("Download in progress")                         .setSmallIcon(R.drawable.download);                 // Start a the operation in a background thread                 new Thread(                         new Runnable() {                             @Override                             public void run() {                                 int incr;                                 // Do the "lengthy" operation 20 times                                 for (incr = 0; incr <= 100; incr+=5) {                                     // Sets the progress indicator to a max value, the current completion percentage and "determinate" state                                     mBuilder.setProgress(100, incr, false);                                     // Displays the progress bar for the first time.                                     mNotifyManager.notify(id, mBuilder.build());                                     // Sleeps the thread, simulating an operation                                     try {                                         // Sleep for 1 second                                         Thread.sleep(1\*1000);                                     } catch (InterruptedException e) {                                         Log.d("TAG", "sleep failure");                                     }                                 }                                 // When the loop is finished, updates the notification                                 mBuilder.setContentText("Download completed")                                         // Removes the progress bar                                         .setProgress(0,0,false);                                 mNotifyManager.notify(id, mBuilder.build());                             }                         }                 // Starts the thread by calling the run() method in its Runnable                 ).start();             }         });     } } |

If we observe above code we are created a progress notification using **setProgress()** method and showing the progress notification on [Button](https://www.tutlane.com/tutorial/android/android-button-with-examples) click.

Generally, during the launch of our [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle), **onCreate()** callback method will be called by android framework to get the required layout for an [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle).

**Output of Android Progress Notification Example**

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



If we observe above result we created a progress notification and shown it on [Button](https://www.tutlane.com/tutorial/android/android-button-with-examples) click based on our requirements.

This is how we can use progress notification in our android applications based on our requirements.

**5. Android Push Notifications using Firebase Cloud Messaging**

1. What is Firebase Cloud Messaging?

**Firebase Cloud Messaging** (FCM) is a cross platform (Android, iOS, Mobile Web) messaging solution which is used to send notification messages to the mobile devices at no cost.

By using firebase we can easily send messages to any device or schedule a messages to send in user’s local time zone based on our requirements.

We don’t need to write much coding to send a notifications and firebase will provide a detailed engagement and conversion tracking for notification messages.

EXTRA

In android, **Push Notification** is used to send an important messages to the mobile devices about our mobile app. By using Push Notifications we can let people know about the events happening in our app by sending a notifications to the mobile devices which contains our app. For example, shopping app’s which will send a notifications about the sale happening in their sites.

In android, by using Firebase Cloud Messaging (FCM) we can easily push a notifications to the mobile devices which will contain our mobile app.

**What is Firebase Cloud Messaging?**

Following is the pictorial representation of how firebase will work with android applications to send or receive messages.



Now we will see how to use firebase cloud messaging in our android applications to push notifications based on our requirements.

**Android Push Notifications Example**

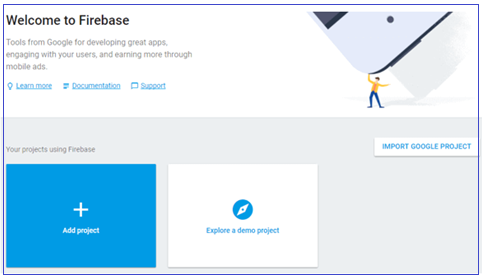
Create a new android application using android studio and give names as **PushNotification**.

Once we are done with creation of new app, we need to integrate a Firebase in our android applications to push notifications.

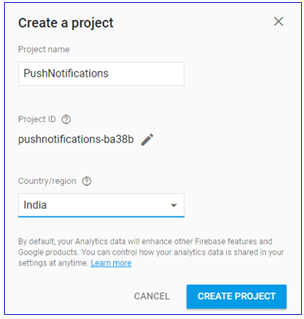
**Add Firebase to our Android App**

To add Firebase to our android app, first we need to create a Firebase project and Firebase configuration file for our app.

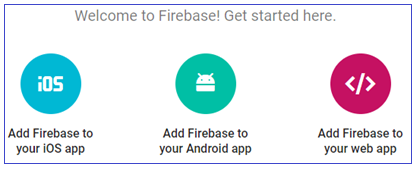
To create a Firebase project. Open [Firebase Console](https://console.firebase.google.com/?authuser=0) and it will ask us to login with Google mail account. Once we are logged in, select **Add Project** like as shown below.



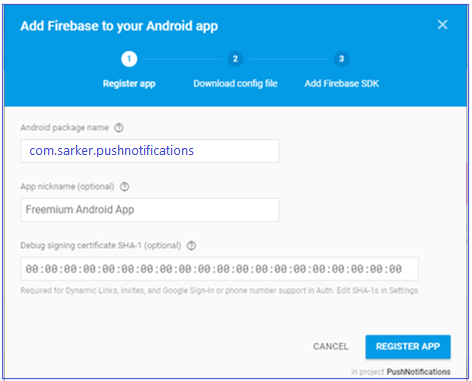
Once we click on **Add Project**, a new popup will open in that enter required details like project name, country and click on **Create Project** like as shown below.



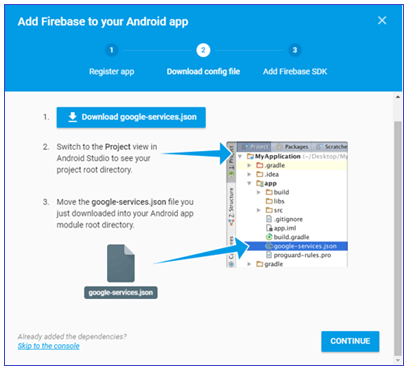
Once we click on Create Project, our project will be created. Now we need to add Firebase to our android app for that we need to select **Add Firebase to your Android App** option from the available options like as shown below.



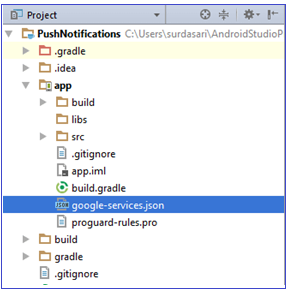
Once we select the required platform, a new popup will open in that we need to enter our app **package name**. This package name must be same as which we defined in our android **AndroidMaifest.xml** file and click **Register App** button like as shown below.



Once we register our app, we will get **google-services.json** file to download and add it to our project **app** folder like as shown below.



Once we add the downloaded **google-services.json** file in our project **app** folder that will be like as shown below.



Now we need to modify our application **build.gradle** file in project level (**<project>/build.gradle**) to include our firebase plugin file (**google-services.json**) like as shown below.

|  |
| --- |
| // Top-level build file where you can add configuration options common to all sub-projects/modules.  buildscript {     repositories {         jcenter()     }     dependencies {         classpath 'com.android.tools.build:gradle:2.3.3'            // Add this line         classpath 'com.google.gms:google-services:3.1.0'      } }  Now open **app** level **build.gradle** (<project>/<app-module>/build.gradle) and make following changes at the bottom of file.  apply plugin: 'com.android.application'  android { …. }  dependencies {     compile fileTree(dir: 'libs', include: ['\*.jar'])     ….   // Add this line compile 'com.google.firebase:firebase-messaging:9.4.0' }  // Add this line to the bottom of file apply plugin: 'com.google.gms.google-services' |

After adding the required dependencies click on **Sync Now** option in the right side to sync our project with required files.

Once we are done with registering our app with firebase, we need to integrate firebase cloud messaging in our android application to send notifications.

**Integrate Firebase Cloud Messaging**

To integrate FCM (firebase cloud messaging) in android app, we need to create a new class file **FireBaseMessaging.java**in **\java\com.sarker.pushnotifications** path and write the code like as shown below.

**FireBaseMessaging.java**

|  |
| --- |
| package com.sarker.pushnotifications; import android.app.NotificationManager; import android.app.PendingIntent; import android.content.Context; import android.content.Intent; import android.media.RingtoneManager; import android.net.Uri; import android.support.v4.app.NotificationCompat; import com.google.firebase.messaging.FirebaseMessagingService; import com.google.firebase.messaging.RemoteMessage;  /\*\*  \* Created by surdasari on 21-09-2017.  \*/  public class FireBaseMessaging extends FirebaseMessagingService {     @Override     public void onMessageReceived(RemoteMessage rMsg) {         sendNotification(rMsg.getNotification());     }      private void sendNotification(RemoteMessage.Notification rNotfy) {         Uri soundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE\_NOTIFICATION);         Intent rintent = new Intent(this, MainActivity.class);         PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, rintent, 0);         NotificationCompat.Builder builder = new NotificationCompat.Builder(this)                 .setContentTitle(rNotfy.getTitle())                 .setContentText(rNotfy.getBody())                 .setAutoCancel(true)                 .setSmallIcon(R.drawable.ic\_notification)                 .setSound(soundUri)                 .setContentIntent(pendingIntent);         NotificationManager notificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE);         notificationManager.notify(0, builder.build());     } } |

If we observe above code, we are extending our class (**FireBaseMessaging.java**) file behaviour using **FirebaseMessagingService** class. The **FirebaseMessagingService** class is the base class to communicate with firebase messaging and it provide a functionality to display the notifications automatically.

Now we need to register our newly created class files in **AndroidManifest.xml** file like as shown below

**AndroidManifest.xml**

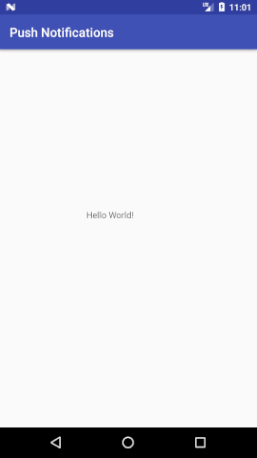
|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?> <manifest xmlns:android="http://schemas.android.com/apk/res/android"     package="com.sarker.pushnotifications">     <uses-permission android:name="android.permission.INTERNET"/>     <application         android:allowBackup="true"         android:icon="@mipmap/ic\_launcher"         android:label="@string/app\_name"         android:roundIcon="@mipmap/ic\_launcher\_round"         android:supportsRtl="true"         android:theme="@style/AppTheme">         <activity android:name=".MainActivity">             <intent-filter>                 <action android:name="android.intent.action.MAIN" />                 <category android:name="android.intent.category.LAUNCHER" />             </intent-filter>         </activity>          <service             android:name=".FireBaseMessaging">             <intent-filter>                 <action android:name="com.google.firebase.MESSAGING\_EVENT"/>             </intent-filter>         </service>      </application> </manifest> |

If we observe above code, we registered our **FireBaseMessaging** class in **AndroidManifest.xml** file and added a permission to access **INTERNET** because firebase cloud messaging required an INTERNET access to show the notifications.

Generally, during the launch of our [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle), **onCreate()** callback method will be called by android framework to get the required layout for an [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle).

**Output of Android Push Notification Example**

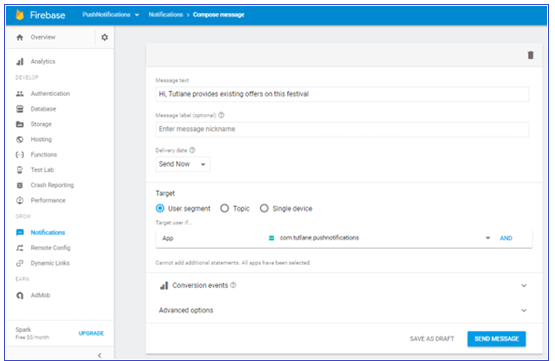
When we run above example using android virtual device (AVD) we will get a result like as shown below.



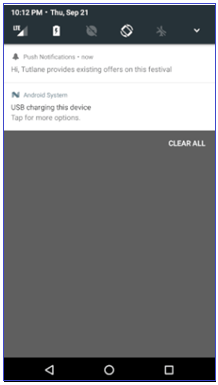
**Push Notifications using Firebase Console**

To publish or push notifications, we need to visit **Firebase Console** and open our app by clicking on it. After that, click on [Notifications](https://www.tutlane.com/tutorial/android/android-notifications-bigtextstyle-bigpicturestyle-inboxstyle) tab in the left panel. In case, if you visit first time, click on **Send Your First Message** otherwise click on **New Message**.

After that enter **Message Text**, select **Target** as **User Segment** then select your app from the list and click on **Send Message** like as shown below.



Once we click on Send Message button, we will get a notification in our app like as shown below.



This is how we can push notifications using firebase cloud messaging in android applications based on our requirements.