Android Resources

ArrayList

```
import java.util.ArrayList; // import the ArrayList class


ArrayList<String> cars = new ArrayList<String>(); // Create an
ArrayList object

import java.util.ArrayList;


public class Main {

  public static void main(String[] args) {

    ArrayList<String> cars = new ArrayList<String>();

    cars.add("Volvo");

    cars.add("BMW");

    cars.add("Ford");

    cars.add("Mazda");

    System.out.println(cars);

  }

}

cars.get(0);

cars.set(0, "Opel");

cars.remove(0);

cars.clear();

cars.size();
```

Toast

```
Context context = getApplicationContext();
CharSequence text = "Hello toast!";
int duration = Toast.LENGTH_SHORT;

Toast toast = Toast.makeText(context, text, duration);
toast.show();
Toast.makeText(context, text, duration).show();
```

```
Toast toast = Toast.makeText(this, "I am custom Toast!", Toast.LENGTH_LONG);
View toastView = toast.getView(); // This'll return the default View of the Toast.

/* And now you can get the TextView of the default View of the Toast. */
TextView toastMessage = (TextView) toastView.findViewById(android.R.id.message);
toastMessage.setTextSize(25);
toastMessage.setTextColor(Color.RED);
toastMessage.setCompoundDrawablesWithIntrinsicBounds(R.mipmap.ic_fly, 0, 0, 0);
toastMessage.setGravity(Gravity.CENTER);
toastMessage.setCompoundDrawablePadding(16);
toastView.setBackgroundColor(Color.CYAN);
toast.show();
```

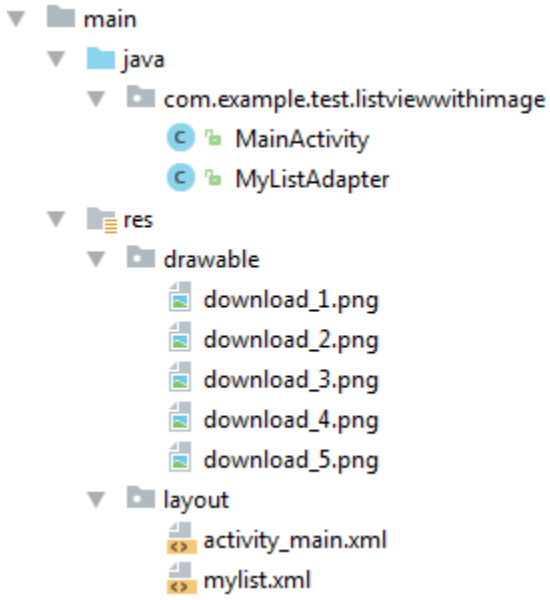
ListView

```
public class ListDisplay extends Activity {
   // Array of strings...
   String[] mobileArray =
{"Android","IPhone","WindowsMobile","Blackberry",
      "WebOS","Ubuntu","Windows7","Max OS X"};

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

      ArrayAdapter adapter = new ArrayAdapter<String>(this,
         R.layout.activity_listview, mobileArray);

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

Example of Custom ListView

In this custom listview example, we are adding image, text with title and its sub-title.

Structure of custom listview project



activity_main.xml

Create an activity_main.xml file in layout folder.

File: activity_main.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"
    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="com.example.test.listviewwithimage.MainActivity">

    <ListView
        android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="50dp">
    </ListView>
</RelativeLayout>
```

Create an additional mylist.xml file in layout folder which contains view components displayed in listview.

mylist.xml

File: mylist.xml

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

3.      android:layout_width="match_parent"
4.      android:layout_height="match_parent"
5.      android:orientation="horizontal" >
6.
7.      <ImageView
8.          android:id="@+id/icon"
9.          android:layout_width="60dp"
10.         android:layout_height="60dp"
11.         android:padding="5dp" />
12.
13.     <LinearLayout android:layout_width="wrap_content"
14.         android:layout_height="wrap_content"
15.         android:orientation="vertical">
16.
17.     <TextView
18.         android:id="@+id/title"
19.         android:layout_width="wrap_content"
20.         android:layout_height="wrap_content"
21.         android:text="Medium Text"
22.         android:textStyle="bold"
23.         android:textAppearance="?android:attr/textAppearanceMedium"
24.         android:layout_marginLeft="10dp"
25.         android:layout_marginTop="5dp"
26.         android:padding="2dp"
27.         android:textColor="#4d4d4d" />
28.     <TextView
29.         android:id="@+id/subtitle"
30.         android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="TextView"
        android:layout_marginLeft="10dp"/>
    </LinearLayout>
</LinearLayout>
```

Place the all required images in drawable folder.

Activity class

File: MainActivity.java

```
package com.example.test.listviewwithimage;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    ListView list;

    String[] maintitle ={
            "Title 1","Title 2",
            "Title 3","Title 4",
            "Title 5",
    };

    String[] subtitle ={
            "Sub Title 1","Sub Title 2",
            "Sub Title 3","Sub Title 4",
            "Sub Title 5",
    };

    Integer[] imgid={
            R.drawable.download_1,R.drawable.download_2,
```

```
27.             R.drawable.download_3,R.drawable.download_4,
28.             R.drawable.download_5,
29.     };
30.     @Override
31.     protected void onCreate(Bundle savedInstanceState) {
32.         super.onCreate(savedInstanceState);
33.         setContentView(R.layout.activity_main);
34.
35.         MyListAdapter adapter=new MyListAdapter(this, maintitle, subtitle,imgid
    );
36.         list=(ListView)findViewById(R.id.list);
37.         list.setAdapter(adapter);
38.
39.
40.         list.setOnItemClickListener(new AdapterView.OnItemClickListener() {
41.
42.             @Override
43.             public void onItemClick(AdapterView<?> parent, View view,int positio
    n, long id) {
44.                 // TODO Auto-generated method stub
45.                 if(position == 0) {
46.                     //code specific to first list item
47.                     Toast.makeText(getApplicationContext(),"Place Your First Option
    Code",Toast.LENGTH_SHORT).show();
48.                 }
49.
50.                 else if(position == 1) {
51.                     //code specific to 2nd list item
52.                     Toast.makeText(getApplicationContext(),"Place Your Second Opti
    on Code",Toast.LENGTH_SHORT).show();
53.                 }
54.
55.                 else if(position == 2) {
56.
57.                     Toast.makeText(getApplicationContext(),"Place Your Third Option
    Code",Toast.LENGTH_SHORT).show();
58.                 }
```

```
59.             else if(position == 3) {
60.
61.                 Toast.makeText(getApplicationContext(),"Place Your Forth Option
    Code",Toast.LENGTH_SHORT).show();
62.             }
63.             else if(position == 4) {
64.
65.                 Toast.makeText(getApplicationContext(),"Place Your Fifth Option
    Code",Toast.LENGTH_SHORT).show();
66.             }
67.
68.         }
69.     });
70. }
71. }
```

Customize Our ListView

Create another java class MyListView.java which extends ArrayAdapter class. This class customizes our listview.

MyListView.java

```
1.  package com.example.test.listviewwithimage;
2.
3.  import android.app.Activity;
4.
5.  import android.view.LayoutInflater;
6.  import android.view.View;
7.  import android.view.ViewGroup;
8.  import android.widget.ArrayAdapter;
9.  import android.widget.ImageView;
10. import android.widget.TextView;
11.
12. public class MyListAdapter extends ArrayAdapter<String> {
13.
14.   private final Activity context;
```

```
15.    private final String[] maintitle;
16.    private final String[] subtitle;
17.    private final Integer[] imgid;
18.
19.    public MyListAdapter(Activity context, String[] maintitle,String[] subtitle, Int
    eger[] imgid) {
20.        super(context, R.layout.mylist, maintitle);
21.        // TODO Auto-generated constructor stub
22.
23.        this.context=context;
24.        this.maintitle=maintitle;
25.        this.subtitle=subtitle;
26.        this.imgid=imgid;
27.
28.    }
29.
30.    public View getView(int position,View view,ViewGroup parent) {
31.        LayoutInflater inflater=context.getLayoutInflater();
32.        View rowView=inflater.inflate(R.layout.mylist, null,true);
33.
34.        TextView titleText = (TextView) rowView.findViewById(R.id.title);
35.        ImageView imageView = (ImageView) rowView.findViewById(R.id.icon);
36.        TextView subtitleText = (TextView) rowView.findViewById(R.id.subtitle);
37.
38.        titleText.setText(maintitle[position]);
39.        imageView.setImageResource(imgid[position]);
40.        subtitleText.setText(subtitle[position]);
41.
42.        return rowView;
43.
44.    };
45. }
```

RecyclerView

RecyclerView is a ViewGroup added to the android studio as a successor of the GridView and ListView. It is an improvement on both of them and can be found in the latest v-7 support packages. It has been created to make possible construction of any lists with XML layouts as an item which can be customized vastly while *improving on the efficiency of ListViews and GridViews*. This improvement is achieved by recycling the views which are out of the visibility of the user. For example, if a user scrolled down to a position where items 4 and 5 are visible; items 1, 2, and 3 would be cleared from the memory to reduce memory consumption.

Implementation: To implement a basic RecyclerView three sub-parts are needed to be constructed which offer the users the degree of control they require in making varying designs of their choice.

1. **The Card Layout: The card layout is an XML layout which will be treated as an item for the list created by the RecyclerView.**
2. **The ViewHolder: The ViewHolder is a java class that stores the reference to the card layout views that have to be dynamically modified during the execution of the program by a list of data obtained either by online databases or added in some other way.**
3. **The Data Class: The Data class is a custom java class that acts as a structure for holding the information for every item of the RecyclerView.**

Below is the implementation of the RecyclerView:

- **exam_card.xml**
- **examViewHolder.java**
- **examData.java**

```
<!-- XML Code illustrating card layout usage. -->
<?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"
    android:layout_width="match_parent"
    android:layout_height="105dp">
```

```
<TextView
   android:layout_width="200dp"
   android:id="@+id/examName"
   android:textSize="16sp"
   android:layout_marginStart="20dp"
   android:text="First Exam"
   android:textColor="@color/black"
   android:layout_marginEnd="20dp"
   android:maxLines="1"
   android:layout_marginTop="15dp"
   android:layout_height="wrap_content"/>

<ImageView
   android:id="@+id/examPic"
   android:layout_width="20dp"
   android:layout_height="20dp"
   android:layout_below="@+id/examName"
   android:tint="#808080"
   android:layout_marginStart="20dp"
   android:layout_marginTop="7dp"
   app:srcCompat="@drawable/baseline_schedule_black_36dp"/>

<TextView
   android:id="@+id/examDate"
   android:layout_toEndOf="@+id/examPic"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
```

```
    android:layout_below="@+id/examName"
    android:layout_marginTop="5dp"
    android:layout_marginEnd="20dp"
    android:layout_marginStart="10dp"
    android:gravity="center"
    android:text="May 23, 2015"
    android:textSize="16sp"/>

<ImageView
    android:id="@+id/examPic2"
    android:layout_width="20dp"
    android:layout_height="20dp"
    android:layout_below="@+id/examDate"
    android:tint="#808080"
    android:layout_marginStart="20dp"
    android:layout_marginTop="7dp"
    app:srcCompat="@drawable/baseline_school_black_36dp"/>

<TextView
    android:id="@+id/examMessage"
    android:layout_toEndOf="@+id/examPic2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/examDate"
    android:layout_marginEnd="20dp"
    android:layout_marginTop="5dp"
    android:layout_marginStart="10dp"
    android:gravity="center"
```

```
        android:text="Best Of Luck"
        android:textSize="16sp"/>


    <TextView
        android:id="@+id/border2"
        android:layout_width="match_parent"
        android:layout_height="1dp"
        android:layout_marginStart="15dp"
        android:layout_marginEnd="15dp"
        android:layout_alignParentBottom="true"
        android:background="#808080"/>


  </RelativeLayout>
```

To click on recycler item:

To click on item of recycler view pass the instance of click interface in constructor of adapter

```
  public class ClickListiner{


  // here index is index
  // of item clicked
  public click(int index);


  }
```

The Adapter: The adapter is the main code responsible for RecyclerView. It holds all the important methods dealing with the implementation of RecylcerView. The basic methods for a successful implementation are:

- ***onCreateViewHolder*: which deals with the inflation of the card layout as an item for the RecyclerView.**
- ***onBindViewHolder*: which deals with the setting of different data and methods related to clicks on particular items of the RecyclerView.**
- ***getItemCount*: which Returns the length of the RecyclerView.**
- ***onAttachedToRecyclerView*: which attaches the adapter to the RecyclerView.**

Below program illustrates an example of a custom adapter:

ImageGalleryAdapter2.java

```
  private class ImageGalleryAdapter2
     extends RecyclerView.Adapter<examViewHolder> {


     List<examData> list
        = Collections.emptyList();


     Context context;
     ClickListiner listiner;


     public ImageGalleryAdapter2(List<examData> list,
                         Context context,ClickListiner listiner)
     {
        this.list = list;
        this.context = context;
        this.listiner = listiner;
     }


     @Override
     public examViewHolder
     onCreateViewHolder(ViewGroup parent,
                  int viewType)
```

```
{

   Context context
      = parent.getContext();
   LayoutInflater inflater
      = LayoutInflater.from(context);

   // Inflate the layout

   View photoView
      = inflater
          .inflate(R.layout.card_exam,
                parent, false);

   examViewHolder viewHolder
      = new examViewHolder(photoView);
   return viewHolder;
}

@Override
public void
onBindViewHolder(final examViewHolder viewHolder,
            final int position)
{
   final index = viewHolder.getAdapterPosition();
   viewHolder.examName
      .setText(list.get(position).name);
   viewHolder.examDate
```

```
            .setText(list.get(position).date);
        viewHolder.examMessage
            .setText(list.get(position).message);
        viewHolder.view.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view)
            {
                listiner.click(index);
            }
        });
    }

    @Override
    public int getItemCount()
    {
        return list.size();
    }

    @Override
    public void onAttachedToRecyclerView(
        RecyclerView recyclerView)
    {
        super.onAttachedToRecyclerView(recyclerView);
    }


}
```

RecyclerView Implementation in an activity:

- **exam.java**
- **activity_exam.xml**

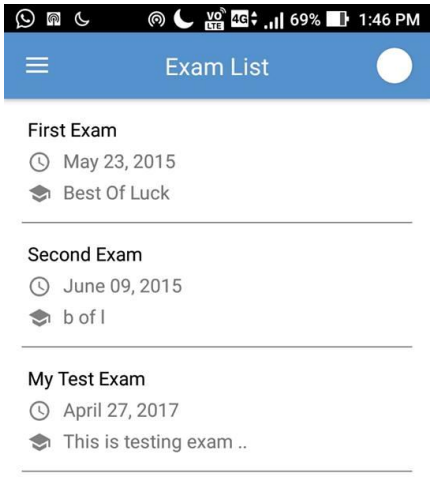
```
package com.example.admin.example;

import android.content.Context;
import android.content.Intent;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.view.LayoutInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;

import com.prolificinteractive
   .materialcalendarview
   .MaterialCalendarView;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class exam extends AppCompatActivity
   implements NavigationView
            .OnNavigationItemSelectedListener {
```

```
ImageGalleryAdapter2 adapter;
RecyclerView recyclerView;
ClickListiner listiner;

@Override
protected void onCreate(Bundle savedInstanceState)
{
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_exam);
   Toolbar toolbar
      = (Toolbar)findViewById(R.id.toolbar);
   toolbar.setTitle("");
   setSupportActionBar(toolbar);

   List<examData> list = new ArrayList<>();
   list = getData();

   recyclerView
      = (RecyclerView)findViewById(
         R.id.recyclerView);
  listiner = new ClickListiner() {
      @Override
      public void click(int index){
    Toast.makeTexT(this,"clicked item index is
"+index,Toast.LENGTH_LONG).show();
     }
  };
```

```
    adapter
        = new ImageGalleryAdapter2(
            list, getApplication(),listiner);
    recyclerView.setAdapter(adapter);
    recyclerView.setLayoutManager(
        new LinearLayoutManager(exam.this));
}

@Override
public void onBackPressed()
{
    super.onBackPressed();
}

// Sample data for RecyclerView
private List<examData> getData()
{
    List<examData> list = new ArrayList<>();
    list.add(new examData("First Exam",
                          "May 23, 2015",
                          "Best Of Luck"));
    list.add(new examData("Second Exam",
                          "June 09, 2015",
                          "b of l"));
    list.add(new examData("My Test Exam",
                          "April 27, 2017",
                          "This is testing exam .."));
```

```
        return list;
    }
}
```

Output:



Keep in mind, that the drawable mentioned in the XML layouts have to be added to the drawable folder under res of the Android Studio Project and support package v7 should be added as an implementation in the Gradle file of the project for the code to run. The above code uses ScrollView as a parent to RecyclerView and disables the scrolling of the RecyclerView hence making the whole page scroll instead of just the RecyclerView contents.

Gallery App

Step by Step Implementation

Step 1: Create a New Project

To create a new project in Android Studio please refer to How to Create/Start a New Project in Android Studio. Note that select **Java** as the programming language.

Step 2: Add the dependency in build.gradle file

Navigate to the **app > Gradle Scripts > build.gradle(:app)** and add the below dependency to it. We are using Picasso for loading images from paths in our ImageView.

Now sync your project and we will move towards adding permissions in our **AndroidManifest.xml** file.

Step 3: Adding permissions in our AndroidManifest.xml file

Navigate to the **app > AndroidManifest.xml** file and add the below permissions to it.

- XML

```
<!-- permissions for reading external storage -->

<uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

As we are loading all the images from our storage at a time so we have to add 2 attributes to our application tag in the AndroidManifest.xml file. Navigate to the AndroidManifest.xml file and add below two lines in your application tag of Manifest file.

- XML

```
android:hardwareAccelerated="false"

android:largeHeap="true"
```

Step 4: Working with the activity_main.xml file

Navigate to the **app > res > layout > activity_main.xml** and add the below code to that file. Below is the code for the **activity_main.xml** file.

- 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"
    android:layout_gravity="center"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <!--recycler view for displaying the list of images-->
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/idRVImages"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</RelativeLayout>
```

Step 5: Creating an item for displaying in a RecyclerView

Navigate to the **app > res > layout > Right-click on it > New > layout Resource file** and create a new layout resource file. Name the file as **card_layout** and add the below code to it. Comments are added in the code to get to know in more detail.

- XML

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_margin="3dp"
    android:elevation="8dp"
    app:cardCornerRadius="8dp">

    <!--Image view for displaying the image
        in our card layout in recycler view-->
    <ImageView
        android:id="@+id/idIVImage"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_gravity="center"
        android:scaleType="centerCrop" />

</androidx.cardview.widget.CardView>
```

Step 6: Creating a new activity for displaying a single image

Navigate to the **app > java > your app's package name > Right-click on it > New > Empty Activity** and name your activity as **ImageDetailActivity** and create a new activity. We will be using this activity to display our single image from the list of different images.

Step 7: Creating an adapter class for setting data to each item in our RecyclerView

Navigate to the **app > java > your app's package name > Right-click on it > New Java class** and name your class as **RecyclerViewAdapter** and add the below code to it. Comments are added in the code to get to know in more detail.

- Java

```
import android.content.Context;

import android.content.Intent;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageView;


import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;


import com.squareup.picasso.Picasso;


import java.io.File;

import java.util.ArrayList;


public class RecyclerViewAdapter extends
RecyclerView.Adapter<RecyclerViewAdapter.RecyclerViewHolder> {


    // creating a variable for our context and array list.

    private final Context context;

    private final ArrayList<String> imagePathArrayList;


    // on below line we have created a constructor.

    public RecyclerViewAdapter(Context context, ArrayList<String>
imagePathArrayList) {

        this.context = context;
```

```
        this.imagePathArrayList = imagePathArrayList;
    }

    @NonNull
    @Override
    public RecyclerViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
int viewType) {
        // Inflate Layout in this method which we have created.
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.card_layout, parent,
false);
        return new RecyclerViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull RecyclerViewHolder holder, int
position) {

        // on below line we are getting th file from the
        // path which we have stored in our list.
        File imgFile = new File(imagePathArrayList.get(position));

        // on below line we are checking if the file exists or not.
        if (imgFile.exists()) {

            // if the file exists then we are displaying that file in our image
view using picasso library.
            Picasso.get().load(imgFile).placeholder(R.drawable.ic_launcher_back
ground).into(holder.imageIV);

            // on below line we are adding click listener to our item of
recycler view.
```

```
            holder.itemView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {

                    // inside on click listener we are creating a new intent
                    Intent i = new Intent(context, ImageDetailActivity.class);

                    // on below line we are passing the image path to our new
activity.
                    i.putExtra("imgPath", imagePathArrayList.get(position));

                    // at last we are starting our activity.
                    context.startActivity(i);
                }
            });
        }
    }

    @Override
    public int getItemCount() {
        // this method returns
        // the size of recyclerview
        return imagePathArrayList.size();
    }

    // View Holder Class to handle Recycler View.
    public static class RecyclerViewHolder extends RecyclerView.ViewHolder {

        // creating variables for our views.
        private final ImageView imageIV;
```

```
        public RecyclerViewHolder(@NonNull View itemView) {
            super(itemView);
            // initializing our views with their ids.
            imageIV = itemView.findViewById(R.id.idIVImage);
        }
    }
}
```

Step 8: Working with the MainActivity.java file

Go to the **MainActivity.java** file and refer to the following code. Below is the code for the **MainActivity.java** file. Comments are added inside the code to understand the code in more detail.

- Java

```
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.os.Bundle;
import android.provider.MediaStore;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;
```

```
import static android.Manifest.permission.READ_EXTERNAL_STORAGE;

public class MainActivity extends AppCompatActivity {

    // on below line we are creating variables for
    // our array list, recycler view and adapter class.
    private static final int PERMISSION_REQUEST_CODE = 200;
    private ArrayList<String> imagePaths;
    private RecyclerView imagesRV;
    private RecyclerViewAdapter imageRVAdapter;

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

        // we are calling a method to request
        // the permissions to read external storage.
        requestPermissions();

        // creating a new array list and
        // initializing our recycler view.
        imagePaths = new ArrayList<>();
        imagesRV = findViewById(R.id.idRVImages);

        // calling a method to
        // prepare our recycler view.
        prepareRecyclerView();
    }
```

```
    private boolean checkPermission() {
        // in this method we are checking if the permissions are granted or not
and returning the result.
        int result = ContextCompat.checkSelfPermission(getApplicationContext(),
READ_EXTERNAL_STORAGE);
        return result == PackageManager.PERMISSION_GRANTED;
    }


    private void requestPermissions() {
        if (checkPermission()) {
            // if the permissions are already granted we are calling
            // a method to get all images from our external storage.
            Toast.makeText(this, "Permissions granted..",
Toast.LENGTH_SHORT).show();
            getImagePath();
        } else {
            // if the permissions are not granted we are
            // calling a method to request permissions.
            requestPermission();
        }
    }


    private void requestPermission() {
        //on below line we are requesting the rea external storage permissions.
        ActivityCompat.requestPermissions(this, new
String[]{READ_EXTERNAL_STORAGE}, PERMISSION_REQUEST_CODE);
    }


    private void prepareRecyclerView() {
```

```
        // in this method we are preparing our recycler view.
        // on below line we are initializing our adapter class.
        imageRVAdapter = new RecyclerViewAdapter(MainActivity.this,
imagePaths);


        // on below line we are creating a new grid layout manager.
        GridLayoutManager manager = new GridLayoutManager(MainActivity.this,
4);


        // on below line we are setting layout
        // manager and adapter to our recycler view.
        imagesRV.setLayoutManager(manager);
        imagesRV.setAdapter(imageRVAdapter);
    }


    private void getImagePath() {
        // in this method we are adding all our image paths
        // in our arraylist which we have created.
        // on below line we are checking if the device is having an sd card or
not.
        boolean isSDPresent =
android.os.Environment.getExternalStorageState().equals(android.os.Environment.
MEDIA_MOUNTED);


        if (isSDPresent) {


            // if the sd card is present we are creating a new list in
            // which we are getting our images data with their ids.
            final String[] columns = {MediaStore.Images.Media.DATA,
MediaStore.Images.Media._ID};


            // on below line we are creating a new
```

```
            // string to order our images by string.
            final String orderBy = MediaStore.Images.Media._ID;


            // this method will stores all the images
            // from the gallery in Cursor
            Cursor cursor =
getContentResolver().query(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
columns, null, null, orderBy);


            // below line is to get total number of images
            int count = cursor.getCount();


            // on below line we are running a loop to add
            // the image file path in our array list.
            for (int i = 0; i < count; i++) {


                // on below line we are moving our cursor position
                cursor.moveToPosition(i);


                // on below line we are getting image file path
                int dataColumnIndex =
cursor.getColumnIndex(MediaStore.Images.Media.DATA);


                // after that we are getting the image file path
                // and adding that path in our array list.
                imagePaths.add(cursor.getString(dataColumnIndex));
            }
            imageRVAdapter.notifyDataSetChanged();
            // after adding the data to our
            // array list we are closing our cursor.
            cursor.close();
```

```
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, String
permissions[], int[] grantResults) {
        // this method is called after permissions has been granted.
        switch (requestCode) {
            // we are checking the permission code.
            case PERMISSION_REQUEST_CODE:
                // in this case we are checking if the permissions are accepted
or not.
                if (grantResults.length > 0) {
                    boolean storageAccepted = grantResults[0] ==
PackageManager.PERMISSION_GRANTED;
                    if (storageAccepted) {
                        // if the permissions are accepted we are displaying a
toast message
                        // and calling a method to get image path.
                        Toast.makeText(this, "Permissions Granted..",
Toast.LENGTH_SHORT).show();
                        getImagePath();
                    } else {
                        // if permissions are denied we are closing the app and
displaying the toast message.
                        Toast.makeText(this, "Permissions denined, Permissions
are required to use the app..", Toast.LENGTH_SHORT).show();
                    }
                }
                break;
        }
    }
```

```
}
```

Step 9: Working with the activity_image_detail.xml file.

Navigate to the **app > res > layout > activity_image_detail.xml** and add the below code to that file. Below is the code for the **activity_image_detail.xml** file.

- 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=".ImageDetailActivity">

    <!--image view to display our image-->
    <ImageView
        android:id="@+id/idIVImage"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:layout_centerInParent="true" />

</RelativeLayout>
```

Step 10: Working with ImageDetailActivity.java file

Go to the **ImageDetailActivity.java** file and refer to the following code. Below is the code for the **ImageDetailActivity.java** file. Comments are added inside the code to understand the code in more detail.

- Java

```
import android.os.Bundle;

import android.view.MotionEvent;

import android.view.ScaleGestureDetector;

import android.widget.ImageView;


import androidx.appcompat.app.AppCompatActivity;


import com.squareup.picasso.Picasso;


import java.io.File;


public class ImageDetailActivity extends AppCompatActivity {


    // creating a string variable, image view variable

    // and a variable for our scale gesture detector class.

    String imgPath;

    private ImageView imageView;

    private ScaleGestureDetector scaleGestureDetector;


    // on below line we are defining our scale factor.

    private float mScaleFactor = 1.0f;


    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

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

```
        // on below line getting data which we have passed from our adapter
class.
        imgPath = getIntent().getStringExtra("imgPath");


        // initializing our image view.
        imageView = findViewById(R.id.idIVImage);


        // on below line we are initializing our scale gesture detector for
zoom in and out for our image.
        scaleGestureDetector = new ScaleGestureDetector(this, new
ScaleListener());


        // on below line we are getting our image file from its path.
        File imgFile = new File(imgPath);


        // if the file exists then we are loading that image in our image view.
        if (imgFile.exists()) {
            Picasso.get().load(imgFile).placeholder(R.drawable.ic_launcher_back
ground).into(imageView);
        }
    }


    @Override
    public boolean onTouchEvent(MotionEvent motionEvent) {
        // inside on touch event method we are calling on
        // touch event method and pasing our motion event to it.
        scaleGestureDetector.onTouchEvent(motionEvent);
        return true;
    }
```

```
    private class ScaleListener extends
ScaleGestureDetector.SimpleOnScaleGestureListener {

        // on below line we are creating a class for our scale

        // listener and  extending it with gesture listener.

        @Override

        public boolean onScale(ScaleGestureDetector scaleGestureDetector) {


            // inside on scale method we are setting scale

            // for our image in our image view.

            mScaleFactor *= scaleGestureDetector.getScaleFactor();

            mScaleFactor = Math.max(0.1f, Math.min(mScaleFactor, 10.0f));


            // on below line we are setting

            // scale x and scale y to our image view.

            imageView.setScaleX(mScaleFactor);

            imageView.setScaleY(mScaleFactor);

            return true;

        }

    }

}
```

Intent

1) Implicit Intent

Implicit Intent doesn't specifiy the component. In such case, intent provides information of available components provided by the system that is to be invoked.

For example, you may write the following code to view the webpage.

```
1. Intent intent=new Intent(Intent.ACTION_VIEW);
2. intent.setData(Uri.parse("http://www.javatpoint.com"));
3. startActivity(intent);
```

2) Explicit Intent

Explicit Intent specifies the component. In such case, intent provides the external class to be invoked.

```
1. Intent i = new Intent(getApplicationContext(), ActivityTwo.class);
2. startActivity(i);
```

To get the full code of explicit intent, visit the next page.

Android Implicit Intent Example

Let's see the simple example of implicit intent that displays a web page.

activity_main.xml

File: activity_main.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.and
   roid.com/apk/res/android"
3.     xmlns:app="http://schemas.android.com/apk/res-auto"
4.     xmlns:tools="http://schemas.android.com/tools"
5.     android:layout_width="match_parent"
6.     android:layout_height="match_parent"
7.     tools:context="example.javatpoint.com.implicitintent.MainActivity">
8.
```

```
    <EditText
      android:id="@+id/editText"
       android:layout_width="wrap_content"
      android:layout_height="wrap_content"
       android:layout_marginEnd="8dp"
      android:layout_marginStart="8dp"
       android:layout_marginTop="60dp"
      android:ems="10"
       app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintHorizontal_bias="0.575"
       app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent" />

    <Button
       android:id="@+id/button"
      android:layout_width="wrap_content"
       android:layout_height="wrap_content"
      android:layout_marginRight="8dp"
       android:layout_marginLeft="156dp"
      android:layout_marginTop="172dp"
       android:text="Visit"
      app:layout_constraintEnd_toEndOf="parent"
       app:layout_constraintHorizontal_bias="0.0"
      app:layout_constraintStart_toStartOf="parent"
       app:layout_constraintTop_toBottomOf="@+id/editText" />
</android.support.constraint.ConstraintLayout>
```

Activity class

File: MainActivity.java

```
package example.javatpoint.com.implicitintent;

import android.content.Intent;
import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
```

```
8.  import android.widget.Button;
9.  import android.widget.EditText;
10.
11. public class MainActivity extends AppCompatActivity {
12.
13.     Button button;
14.     EditText editText;
15.
16.     @Override
17.     protected void onCreate(Bundle savedInstanceState) {
18.         super.onCreate(savedInstanceState);
19.         setContentView(R.layout.activity_main);
20.
21.         button = findViewById(R.id.button);
22.         editText =  findViewById(R.id.editText);
23.
24.         button.setOnClickListener(new View.OnClickListener() {
25.             @Override
26.             public void onClick(View view) {
27.                 String url=editText.getText().toString();
28.                 Intent intent=new Intent(Intent.ACTION_VIEW, Uri.parse(url));
29.                 startActivity(intent);
30.             }
31.         });
32.     }
33. }
```

Example#

This example illustrates sending a `String` with value as `"Some data!"` from `OriginActivity` to `DestinationActivity`.

NOTE: This is the most straightforward way of sending data between two activities. See the example on using the starter pattern for a more robust implementation.

OriginActivity#

```
public class OriginActivity extends AppCompatActivity {

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_origin);

        // Create a new Intent object, containing DestinationActivity as target
Activity.
        final Intent intent = new Intent(this, DestinationActivity.class);

        // Add data in the form of key/value pairs to the intent object by using
putExtra()
        intent.putExtra(DestinationActivity.EXTRA_DATA, "Some data!");

        // Start the target Activity with the intent object
        startActivity(intent);
    }
}
```

DestinationActivity#

```
public class DestinationActivity extends AppCompatActivity {

    public static final String EXTRA_DATA = "EXTRA_DATA";

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

        // getIntent() returns the Intent object which was used to start this
Activity
        final Intent intent = getIntent();

        // Retrieve the data from the intent object by using the same key that
        // was previously used to add data to the intent object in OriginActivity.
        final String data = intent.getStringExtra(EXTRA_DATA);
    }
}
```

It is also possible to pass other `primitive` data types as well as `arrays`, `Bundle` and `Parcelable` data. Passing `Serializable` is also possible, but should be avoided as it is more than three times slower than `Parcelable`.

Serializable is a standard Java `interface`. You simply mark a class as `Serializable` by implementing the `Serializable` `interface` and Java will automatically serialize it during required situations.

Parcelable is an Android specific `interface` which can be implemented on custom data types (i.e. your own objects / POJO objects), it allows your object to be flattened and reconstruct itself without the destination needing to do anything. There is a documentation example of making an object parcelable.

Once you have a `parcelable` object you can send it like a primitive type, with an intent object:

```
intent.putExtra(DestinationActivity.EXTRA_DATA, myParcelableObject);
```
Or in a bundle / as an argument for a fragment:

```
bundle.putParcelable(DestinationActivity.EXTRA_DATA, myParcelableObject);
```
and then also read it from the intent at the destination using getParcelableExtra:

```
final MyParcelableType data = intent.getParcelableExtra(EXTRA_DATA);
```
Or when reading in a fragment from a bundle:

```
final MyParcelableType data = bundle.getParcelable(EXTRA_DATA);
```
Once you have a `Serializable` object you can put it in an intent object:

```
bundle.putSerializable(DestinationActivity.EXTRA_DATA, mySerializableObject);
```
and then also read it from the intent object at the destination as shown below:

```
final SerializableType data =
(SerializableType)bundle.getSerializable(EXTRA_DATA);
```

Notifications

```
public class MainActivity extends AppCompatActivity {

    Integer hours,minutes;
    @RequiresApi(api = Build.VERSION_CODES.O)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button set;
        EditText hourset,minuteset;
        set = findViewById(R.id.set_btn);
        hourset = findViewById(R.id.hours_et);
        minuteset = findViewById(R.id.minutes_et);

        NotificationChannel notificationChannel = new
NotificationChannel("MyNotification", "My Notification",
NotificationManager.IMPORTANCE_DEFAULT);
        NotificationManager notificationManager =
getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(notificationChannel);

        set.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                hours = Integer.parseInt(hourset.getText().toString());
                minutes = Integer.parseInt(minuteset.getText().toString());

                addNotification();
            }
        });
    }

    private void addNotification() {
        NotificationCompat.Builder builder = new
NotificationCompat.Builder(MainActivity.this, "MyNotification");
```

```
        builder.setSmallIcon(R.drawable.ic_launcher_background);
        builder.setContentTitle("Notifications Example");
        builder.setContentText("This is a test notification");
        builder.setAutoCancel(true);

//        Intent notificationIntent = new Intent(this, MainActivity.class);
//        PendingIntent contentIntent = PendingIntent.getActivity(this, 0,
notificationIntent,
//                PendingIntent.FLAG_UPDATE_CURRENT);
//        builder.setContentIntent(contentIntent);

        // Add as notification
        NotificationManagerCompat notificationManagerCompat =
NotificationManagerCompat.from(MainActivity.this);
        notificationManagerCompat.notify(1, builder.build());
    }

}
```

notification.xml

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

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="400dp"
        android:text="Hi, Your Detailed notification view goes here...." />
</LinearLayout>
```

Create and Send Notifications

You have simple way to create a notification. Follow the following steps in your application to create a notification −

Step 1 - Create Notification Builder

As a first step is to create a notification builder using *NotificationCompat.Builder.build()*. You will use Notification Builder to set various Notification properties like its small and large icons, title, priority etc.

```
NotificationCompat.Builder mBuilder = new
NotificationCompat.Builder(this)
```

Step 2 - Setting Notification Properties

Once you have **Builder** object, you can set its Notification properties using Builder object as per your requirement. But this is mandatory to set at least following −

- A small icon, set by **setSmallIcon()**

- A title, set by **setContentTitle()**
- Detail text, set by **setContentText()**

```
mBuilder.setSmallIcon(R.drawable.notification_icon);
mBuilder.setContentTitle("Notification Alert, Click Me!");
mBuilder.setContentText("Hi, This is Android Notification
Detail!");
```

You have plenty of optional properties which you can set for your notification. To learn more about them, see the reference documentation for NotificationCompat.Builder.

Step 3 - Attach Actions

This is an optional part and required if you want to attach an action with the notification. An action allows users to go directly from the notification to an **Activity** in your application, where they can look at one or more events or do further work.

The action is defined by a **PendingIntent** containing an **Intent** that starts an Activity in your application. To associate the PendingIntent with a gesture, call the appropriate method of *NotificationCompat.Builder*. For example, if you want to start Activity when the user clicks the notification text in the notification drawer, you add the PendingIntent by calling **setContentIntent()**.

A PendingIntent object helps you to perform an action on your applications behalf, often at a later time, without caring of whether or not your application is running.

We take help of stack builder object which will contain an artificial back stack for the started Activity. This ensures that navigating backward from the Activity leads out of your application to the Home screen.

```
Intent resultIntent = new Intent(this, ResultActivity.class);
TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
stackBuilder.addParentStack(ResultActivity.class);

// Adds the Intent that starts the Activity to the top of the
stack
stackBuilder.addNextIntent(resultIntent);
PendingIntent resultPendingIntent =
stackBuilder.getPendingIntent(0,PendingIntent.FLAG_UPDATE_CURRENT
);
mBuilder.setContentIntent(resultPendingIntent);
```

Step 4 - Issue the notification

Finally, you pass the Notification object to the system by calling NotificationManager.notify() to send your notification. Make sure you call **NotificationCompat.Builder.build()** method on builder object before notifying it. This method combines all of the options that have been set and return a new **Notification** object.

```
NotificationManager mNotificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
```

```
// notificationID allows you to update the notification later on.
mNotificationManager.notify(notificationID, mBuilder.build());
```

Following is the content of the modified main activity file **src/com.example.notificationdemo/MainActivity.java**. This file can include each of the fundamental lifecycle methods.

```
package com.example.notificationdemo;

import android.app.Activity;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.support.v4.app.NotificationCompat;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity {
   Button b1;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      b1 = (Button)findViewById(R.id.button);
      b1.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            addNotification();
         }
      });
   }

   private void addNotification() {
      NotificationCompat.Builder builder =
         new NotificationCompat.Builder(this)
         .setSmallIcon(R.drawable.abc)
         .setContentTitle("Notifications Example")
         .setContentText("This is a test notification");

      Intent notificationIntent = new Intent(this,
MainActivity.class);
      PendingIntent contentIntent =
PendingIntent.getActivity(this, 0, notificationIntent,
         PendingIntent.FLAG_UPDATE_CURRENT);
      builder.setContentIntent(contentIntent);
```

```
      // Add as notification
      NotificationManager manager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
      manager.notify(0, builder.build());
   }
}
```

Following will be the content of **res/layout/notification.xml** file −

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

   <TextView
      android:layout_width="fill_parent"
      android:layout_height="400dp"
      android:text="Hi, Your Detailed notification view goes
here...." />
</LinearLayout>
```

Following is the content of the modified main activity file **src/com.example.notificationdemo/NotificationView.java**.

```
package com.example.notificationdemo;

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

public class NotificationView extends Activity{
   @Override
   public void onCreate(Bundle savedInstanceState){
      super.onCreate(savedInstanceState);
      setContentView(R.layout.notification);
   }
}
```

Following will be the content of **res/layout/activity_main.xml** file −

```
<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/textView1"
      android:layout_width="wrap_content"
```

```
      android:layout_height="wrap_content"
      android:text="Notification Example"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:textSize="30dp" />

   <TextView
      android:id="@+id/textView2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Tutorials point "
      android:textColor="#ff87ff09"
      android:textSize="30dp"
      android:layout_below="@+id/textView1"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="48dp" />

   <ImageButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/imageButton"
      android:src="@drawable/abc"
      android:layout_below="@+id/textView2"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="42dp" />

   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Notification"
      android:id="@+id/button"
      android:layout_marginTop="62dp"
      android:layout_below="@+id/imageButton"
      android:layout_centerHorizontal="true" />

</RelativeLayout>
```

Following will be the content of **res/values/strings.xml** to define two new constants −

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="action_settings">Settings</string>
   <string name="app_name">tutorialspoint </string>
</resources>
```

Following is the default content of **AndroidManifest.xml** −

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.notificationdemo" >

   <application
```

```
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >

      <activity
         android:name="com.example.notificationdemo.MainActivity"
         android:label="@string/app_name" >

         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category
android:name="android.intent.category.LAUNCHER" />
         </intent-filter>

      </activity>

      <activity android:name=".NotificationView"
         android:label="Details of notification"
         android:parentActivityName=".MainActivity">
         <meta-data
         android:name="android.support.PARENT_ACTIVITY"
         android:value=".MainActivity"/>
      </activity>

   </application>
</manifest>
```

AlertDialog

Android AlertDialog Example

Let's see a simple example of android alert dialog.

activity_main.xml

You can have multiple components, here we are having only a textview.

File: activity_main.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.and
   roid.com/apk/res/android"
3.     xmlns:app="http://schemas.android.com/apk/res-auto"
4.     xmlns:tools="http://schemas.android.com/tools"
5.     android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    tools:context="example.javatpoint.com.alertdialog.MainActivity">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button"
        android:text="Close app"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

---

strings.xml

Optionally, you can store the dialog message and title in the strings.xml file.

File: strings.xml

```
<resources>
    <string name="app_name">AlertDialog</string>
    <string name="dialog_message">Welcome to Alert Dialog</string>
    <string name="dialog_title">Javatpoint Alert Dialog</string>
</resources>
```

Activity class

Let's write the code to create and show the AlertDialog.

File: MainActivity.java

```
package example.javatpoint.com.alertdialog;

import android.content.DialogInterface;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
```

```
7.  import android.widget.Button;
8.  import android.app.AlertDialog;
9.  import android.widget.Toast;
10.
11. public class MainActivity extends AppCompatActivity {
12.     Button closeButton;
13.     AlertDialog.Builder builder;
14.     @Override
15.     protected void onCreate(Bundle savedInstanceState) {
16.         super.onCreate(savedInstanceState);
17.         setContentView(R.layout.activity_main);
18.
19.         closeButton = (Button) findViewById(R.id.button);
20.         builder = new AlertDialog.Builder(this);
21.         closeButton.setOnClickListener(new View.OnClickListener() {
22.             @Override
23.             public void onClick(View v) {
24.
25.                 //Uncomment the below code to Set the message and title from the
    strings.xml file
26.                 builder.setMessage(R.string.dialog_message) .setTitle(R.string.dialog_title);
27.
28.                 //Setting message manually and performing action on button click
29.                 builder.setMessage("Do you want to close this application ?")
30.                         .setCancelable(false)
31.                         .setPositiveButton("Yes", new DialogInterface.OnClickListener()
    {
32.                             public void onClick(DialogInterface dialog, int id) {
33.                                 finish();
34.                                 Toast.makeText(getApplicationContext(),"you choose yes action f
    or alertbox",
35.                                         Toast.LENGTH_SHORT).show();
36.                             }
37.                         })
38.                         .setNegativeButton("No", new DialogInterface.OnClickListener() {
39.                             public void onClick(DialogInterface dialog, int id) {
40.                                 //  Action for 'NO' Button
```

```
41.                         dialog.cancel();
42.                       Toast.makeText(getApplicationContext(),"you choose no action fo
    r alertbox",
43.                         Toast.LENGTH_SHORT).show();
44.                     }
45.                   });
46.             //Creating dialog box
47.              AlertDialog alert = builder.create();
48.             //Setting the title manually
49.              alert.setTitle("AlertDialogExample");
50.             alert.show();
51.           }
52.       });
53.    }
54. }
```

A Dialog is small window that prompts the user to a decision or enter additional information.

Some times in your application, if you wanted to ask the user about taking a decision between yes or no in response of any particular action taken by the user, by remaining in the same activity and without changing the screen, you can use Alert Dialog.

In order to make an alert dialog, you need to make an object of AlertDialogBuilder which an inner class of AlertDialog. Its syntax is given below

```
AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(this);
```

Now you have to set the positive (yes) or negative (no) button using the object of the AlertDialogBuilder class. Its syntax is

```
alertDialogBuilder.setPositiveButton(CharSequence text,
   DialogInterface.OnClickListener listener)
alertDialogBuilder.setNegativeButton(CharSequence text,
   DialogInterface.OnClickListener listener)
```

Apart from this , you can use other functions provided by the builder class to customize the alert dialog. These are listed below

Sr.No	Method type & description
1	**setIcon(Drawable icon)**

	This method set the icon of the alert dialog box.
2	**setCancelable(boolean cancel able)** This method sets the property that the dialog can be cancelled or not
3	**setMessage(CharSequence message)** This method sets the message to be displayed in the alert dialog
4	**setMultiChoiceItems(CharSequence[] items, boolean[] checkedItems DialogInterface.OnMultiChoiceClickListener listener)** This method sets list of items to be displayed in the dialog as the content. The selecte option will be notified by the listener
5	**setOnCancelListener(DialogInterface.OnCancelListener onCancelListener)** This method Sets the callback that will be called if the dialog is cancelled.
6	**setTitle(CharSequence title)** This method set the title to be appear in the dialog

After creating and setting the dialog builder , you will create an alert dialog by calling the create() method of the builder class. Its syntax is

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

This will create the alert dialog and will show it on the screen.

Dialog fragment

Before enter into an example we should need to know dialog fragment.Dialog fragment is a fragment which can show fragment in dialog box

```
public class DialogFragment extends DialogFragment {
   @Override
   public Dialog onCreateDialog(Bundle savedInstanceState) {
      // Use the Builder class for convenient dialog construction
      AlertDialog.Builder builder = new
AlertDialog.Builder(getActivity());
      builder.setPositiveButton(R.string.fire, new
DialogInterface.OnClickListener() {
         public void onClick(DialogInterface dialog, int id) {
            toast.makeText(this,"enter a text
here",Toast.LENTH_SHORT).show();
```

```
            }
        })
        .setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                finish();
            });
            // Create the AlertDialog object and return it
            return builder.create();
        }
    }
}
```

List dialog

It has used to show list of items in a dialog box.For suppose, user need to select a list of items or else need to click a item from multiple list of items.At this situation we can use list dialog.

```
public Dialog onCreateDialog(Bundle savedInstanceState) {
    AlertDialog.Builder builder = new
AlertDialog.Builder(getActivity());
    builder.setTitle(Pick a Color)

    .setItems(R.array.colors_array, new
DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            // The 'which' argument contains the index position
            // of the selected item
        }
    });
    return builder.create();
}
```

Single-choice list dialog

It has used to add single choice list to Dialog box.We can check or uncheck as per user choice.

```
public Dialog onCreateDialog(Bundle savedInstanceState) {
    mSelectedItems = new ArrayList();
    AlertDialog.Builder builder = new
AlertDialog.Builder(getActivity());

    builder.setTitle("This is list choice dialog box");
    .setMultiChoiceItems(R.array.toppings, null,
        new DialogInterface.OnMultiChoiceClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which,
boolean isChecked) {
```

```
            if (isChecked) {
               // If the user checked the item, add it to the
selected items
               mSelectedItems.add(which);
            }

            else if (mSelectedItems.contains(which)) {
               // Else, if the item is already in the array, remove
it
               mSelectedItems.remove(Integer.valueOf(which));
            }
         }
      })

      // Set the action buttons
      .setPositiveButton(R.string.ok, new
DialogInterface.OnClickListener() {
         @Override
         public void onClick(DialogInterface dialog, int id) {
            // User clicked OK, so save the mSelectedItems results
somewhere
            // or return them to the component that opened the
dialog
            ...
         }
      })

      .setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener() {
         @Override
         public void onClick(DialogInterface dialog, int id) {
            ...
         }
      });
      return builder.create();
}
```

Example

The following example demonstrates the use of AlertDialog in android.

To experiment with this example , you need to run this on an emulator or an actual device.

Steps	Description
1	You will use Android studio to create an Android application and name it as My Applicatio under a package com.example.sairamkrishna.myapplication.

2	Modify src/MainActivity.java file to add alert dialog code to launch the dialog.
3	Modify layout XML file res/layout/activity_main.xml add any GUI component if required.
4	No need to change default string constants. Android studio takes care of default strings a values/string.xml
5	Run the application and choose a running android device and install the application on it and verify the results.

Here is the modified code of **src/MainActivity.java**

```
package com.example.sairamkrishna.myapplication;

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

public class MainActivity extends ActionBarActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }

   public void open(View view){
      AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(this);
      alertDialogBuilder.setMessage("Are you sure,
         You wanted to make decision");
      alertDialogBuilder.setPositiveButton("yes",
         new DialogInterface.OnClickListener() {
         @Override
         public void onClick(DialogInterface arg0, int arg1) {
            Toast.makeText(MainActivity.this,"You clicked yes
               button",Toast.LENGTH_LONG).show();
         }
      });

      alertDialogBuilder.setNegativeButton("No",new
DialogInterface.OnClickListener() {
         Override
         public void onClick(DialogInterface dialog, int which) {
            finish();
         }
```

```
        });

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

Here is the modified code of **res/layout/activity_main.xml**

In the below code **abc** indicates the logo of tutorialspoint.com

```
<?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:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   android:paddingBottom="@dimen/activity_vertical_margin"
   tools:context=".MainActivity">

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Alert Dialog"
      android:id="@+id/textView"
      android:textSize="35dp"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Tutorialspoint"
      android:id="@+id/textView2"
      android:textColor="#ff3eff0f"
      android:textSize="35dp"
      android:layout_below="@+id/textView"
      android:layout_centerHorizontal="true" />

   <ImageView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/imageView"
      android:src="@drawable/abc"
      android:layout_below="@+id/textView2"
      android:layout_alignRight="@+id/textView2"
      android:layout_alignEnd="@+id/textView2"
      android:layout_alignLeft="@+id/textView"
      android:layout_alignStart="@+id/textView" />
   <Button
      android:layout_width="wrap_content"
```

```
      android:layout_height="wrap_content"
      android:text="Alert dialog"
      android:id="@+id/button"
      android:layout_below="@+id/imageView"
      android:layout_alignRight="@+id/textView2"
      android:layout_alignEnd="@+id/textView2"
      android:layout_marginTop="42dp"
      android:onClick="open"
      android:layout_alignLeft="@+id/imageView"
      android:layout_alignStart="@+id/imageView" />

</RelativeLayout>
```

Here is of**Strings.xml**

```
<resources>
    <string name="app_name">My Application</string>
</resources>
```

Here is the default code of **AndroidManifest.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.sairamkrishna.myapplication" >

   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >

      <activity

android:name="com.example.sairamkrishna.myapplication.MainActivit
y"
         android:label="@string/app_name" >

         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category
android:name="android.intent.category.LAUNCHER" />
         </intent-filter>

      </activity>

   </application>
</manifest>
```

Media Player

Android provides many ways to control playback of audio/video files and streams. One of this way is through a class called **MediaPlayer**.

Android is providing MediaPlayer class to access built-in mediaplayer services like playing audio,video e.t.c. In order to use MediaPlayer, we have to call a static Method **create()** of this class. This method returns an instance of MediaPlayer class. Its syntax is as follows −

```
MediaPlayer mediaPlayer = MediaPlayer.create(this, R.raw.song);
```

The second parameter is the name of the song that you want to play. You have to make a new folder under your project with name **raw** and place the music file into it.

Once you have created the Mediaplayer object you can call some methods to start or stop the music. These methods are listed below.

```
mediaPlayer.start();
mediaPlayer.pause();
```

On call to **start()** method, the music will start playing from the beginning. If this method is called again after the **pause()** method, the music would start playing from where it is left and not from the beginning.

In order to start music from the beginning, you have to call **reset()** method. Its syntax is given below.

```
mediaPlayer.reset();
```

Apart from the start and pause method, there are other methods provided by this class for better dealing with audio/video files. These methods are listed below −

Sr.No	Method & description
1	**isPlaying()** This method just returns true/false indicating the song is playing or not
2	**seekTo(position)** This method takes an integer, and move song to that particular position millisecond

3	**getCurrentPosition()** This method returns the current position of song in milliseconds
4	**getDuration()** This method returns the total time duration of song in milliseconds
5	**reset()** This method resets the media player
6	**release()** This method releases any resource attached with MediaPlayer object
7	**setVolume(float leftVolume, float rightVolume)** This method sets the up down volume for this player
8	**setDataSource(FileDescriptor fd)** This method sets the data source of audio/video file
9	**selectTrack(int index)** This method takes an integer, and select the track from the list on that particular index
10	**getTrackInfo()** This method returns an array of track information

Example

Here is an example demonstrating the use of MediaPlayer class. It creates a basic media player that allows you to forward, backward, play and pause a song.

To experiment with this example, you need to run this on an actual device to hear the audio sound.

Steps	Description

1	You will use Android studio IDE to create an Android application under a package com.example.sairamkrishna.myapplication.
2	Modify src/MainActivity.java file to add MediaPlayer code.
3	Modify the res/layout/activity_main to add respective XML components
4	Create a new folder under MediaPlayer with name as raw and place an mp3 music file in with name as song.mp3
5	Run the application and choose a running android device and install the application on it and verify the results

Following is the content of the modified main activity file **src/MainActivity.java**.

```
package com.example.sairamkrishna.myapplication;

import android.app.Activity;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;

import android.widget.Button;
import android.widget.ImageView;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;
import java.util.concurrent.TimeUnit;


public class MainActivity extends Activity {
   private Button b1,b2,b3,b4;
   private ImageView iv;
   private MediaPlayer mediaPlayer;

   private double startTime = 0;
   private double finalTime = 0;

   private Handler myHandler = new Handler();;
   private int forwardTime = 5000;
   private int backwardTime = 5000;
   private SeekBar seekbar;
   private TextView tx1,tx2,tx3;

   public static int oneTimeOnly = 0;
   @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);
      b3 = (Button)findViewById(R.id.button3);
      b4 = (Button)findViewById(R.id.button4);
      iv = (ImageView)findViewById(R.id.imageView);

      tx1 = (TextView)findViewById(R.id.textView2);
      tx2 = (TextView)findViewById(R.id.textView3);
      tx3 = (TextView)findViewById(R.id.textView4);
      tx3.setText("Song.mp3");

      mediaPlayer = MediaPlayer.create(this, R.raw.song);
      seekbar = (SeekBar)findViewById(R.id.seekBar);
      seekbar.setClickable(false);
      b2.setEnabled(false);

      b3.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            Toast.makeText(getApplicationContext(), "Playing
               sound",Toast.LENGTH_SHORT).show();
            mediaPlayer.start();

            finalTime = mediaPlayer.getDuration();
            startTime = mediaPlayer.getCurrentPosition();

            if (oneTimeOnly == 0) {
               seekbar.setMax((int) finalTime);
               oneTimeOnly = 1;
            }

            tx2.setText(String.format("%d min, %d sec",
               TimeUnit.MILLISECONDS.toMinutes((long) finalTime),
               TimeUnit.MILLISECONDS.toSeconds((long) finalTime)
-

TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes((long)
                  finalTime)))
            );

            tx1.setText(String.format("%d min, %d sec",
               TimeUnit.MILLISECONDS.toMinutes((long) startTime),
               TimeUnit.MILLISECONDS.toSeconds((long) startTime)
-

TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes((long)
                  startTime)))
            );
```

```
            seekbar.setProgress((int)startTime);
            myHandler.postDelayed(UpdateSongTime,100);
            b2.setEnabled(true);
            b3.setEnabled(false);
         }
      });

      b2.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            Toast.makeText(getApplicationContext(), "Pausing
               sound",Toast.LENGTH_SHORT).show();
            mediaPlayer.pause();
            b2.setEnabled(false);
            b3.setEnabled(true);
         }
      });

      b1.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            int temp = (int)startTime;

            if((temp+forwardTime)<=finalTime){
               startTime = startTime + forwardTime;
               mediaPlayer.seekTo((int) startTime);
               Toast.makeText(getApplicationContext(),"You have
Jumped forward 5
                  seconds",Toast.LENGTH_SHORT).show();
            }else{
               Toast.makeText(getApplicationContext(),"Cannot
jump forward 5
                  seconds",Toast.LENGTH_SHORT).show();
            }
         }
      });

      b4.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            int temp = (int)startTime;

            if((temp-backwardTime)>0){
               startTime = startTime - backwardTime;
               mediaPlayer.seekTo((int) startTime);
               Toast.makeText(getApplicationContext(),"You have
Jumped backward 5
                  seconds",Toast.LENGTH_SHORT).show();
            }else{
               Toast.makeText(getApplicationContext(),"Cannot
jump backward 5
                  seconds",Toast.LENGTH_SHORT).show();
            }
```

```
            }
        });
    }

    private Runnable UpdateSongTime = new Runnable() {
        public void run() {
            startTime = mediaPlayer.getCurrentPosition();
            tx1.setText(String.format("%d min, %d sec",
                TimeUnit.MILLISECONDS.toMinutes((long) startTime),
                TimeUnit.MILLISECONDS.toSeconds((long) startTime) -
                TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.
                toMinutes((long) startTime)))
            );
            seekbar.setProgress((int)startTime);
            myHandler.postDelayed(this, 100);
        }
    };
}
```

Following is the modified content of the xml **res/layout/activity_main.xml**.

In the below code **abc**indicates the logo of tutorialspoint.com

```
<?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:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".MainActivity">

    <TextView android:text="Music Palyer"
android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textview"
        android:textSize="35dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Tutorials point"
        android:id="@+id/textView"
        android:layout_below="@+id/textview"
        android:layout_centerHorizontal="true"
        android:textColor="#ff7aff24"
        android:textSize="35dp" />

    <ImageView
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/imageView"
    android:layout_below="@+id/textView"
    android:layout_centerHorizontal="true"
    android:src="@drawable/abc"/>

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/forward"
    android:id="@+id/button"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/pause"
    android:id="@+id/button2"
    android:layout_alignParentBottom="true"
    android:layout_alignLeft="@+id/imageView"
    android:layout_alignStart="@+id/imageView" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/back"
    android:id="@+id/button3"
    android:layout_alignTop="@+id/button2"
    android:layout_toRightOf="@+id/button2"
    android:layout_toEndOf="@+id/button2" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/rewind"
    android:id="@+id/button4"
    android:layout_alignTop="@+id/button3"
    android:layout_toRightOf="@+id/button3"
    android:layout_toEndOf="@+id/button3" />

<SeekBar
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/seekBar"
    android:layout_alignLeft="@+id/textview"
    android:layout_alignStart="@+id/textview"
    android:layout_alignRight="@+id/textview"
    android:layout_alignEnd="@+id/textview"
    android:layout_above="@+id/button" />
```

```
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceSmall"
      android:text="Small Text"
      android:id="@+id/textView2"
      android:layout_above="@+id/seekBar"
      android:layout_toLeftOf="@+id/textView"
      android:layout_toStartOf="@+id/textView" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceSmall"
      android:text="Small Text"
      android:id="@+id/textView3"
      android:layout_above="@+id/seekBar"
      android:layout_alignRight="@+id/button4"
      android:layout_alignEnd="@+id/button4" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceMedium"
      android:text="Medium Text"
      android:id="@+id/textView4"
      android:layout_alignBaseline="@+id/textView2"
      android:layout_alignBottom="@+id/textView2"
      android:layout_centerHorizontal="true" />

</RelativeLayout>
```

Following is the content of the **res/values/string.xml**.

```
<resources>
   <string name="app_name">My Application</string>
   <string name="back"><![CDATA[<]]></string>
   <string name="rewind"><![CDATA[<<]]></string>
   <string name="forward"><![CDATA[>>]]></string>
   <string name="pause">||</string>
</resources>
```

Following is the content of **AndroidManifest.xml** file.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.sairamkrishna.myapplication" >

   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
```

```
      <activity

android:name="com.example.sairamkrishna.myapplication.MainActivit
y"
         android:label="@string/app_name" >

         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category
android:name="android.intent.category.LAUNCHER" />
         </intent-filter>

      </activity>

   </application>
</manifest>
```

In android, by using **MediaPlayer** class we can easily fetch, decode and play both audio and video files with minimal setup.

The android media framework provides built-in support for playing a variety of common media types, such as audio or video. We have multiple ways to play audio or video but the most important component of media framework is **MediaPlayer** class.

Android MediaPlayer Class

In android, by using **MediaPlayer** class we can access audio or video files from application (raw) resources, standalone files in file system or from a data stream arriving over a network connection and play audio or video files with the multiple playback options such as play, pause, forward, backward, etc.

Following is the code snippet, to play an audio that is available in our application's local raw resource (**res**/**raw**) directory.

```
MediaPlayer mPlayer = MediaPlayer.create(this, R.raw.baitikochi_chuste)
;
mPlayer.start();
```

The second parameter in **create()** method is the name of the song that we want to play from our application resource directory (**res/raw**). In case if **raw** folder not exists in your application, create a new **raw** folder under **res** directory and add a properly encoded and formatted media files in it.

In case, if we want to play an audio from a **URI** that is locally available in the system, we need to write the code like as shown below.

```
Uri myUri = ....; // initialize Uri here
MediaPlayer mPlayer = new MediaPlayer();
mPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
```

```
mPlayer.setDataSource(getApplicationContext(), myUri);
mPlayer.prepare();
mPlayer.start();
```

If we want to play an audio from a **URL** via HTTP streaming, we need to write the code like as shown below.

```
String url = "http://........"; // your URL here
MediaPlayer mPlayer = new MediaPlayer();
mPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mPlayer.setDataSource(url);
mPlayer.prepare(); // might take long! (for buffering, etc)
mPlayer.start();
```

If you observe above code snippets, we create an instance of **MediaPlayer** class and added required audio source, streaming type, audio file path, etc. to play an audio from our application.

Apart from above methods, **MediaPlayer** class provides a different type of methods to control audio and video files based on requirements.

Method	Description
getCurrentPosition()	It is used to get the current position of the song in milliseconds.
getDuration()	It is used to get the total time duration of the song in milliseconds.
isPlaying()	It returns true / false to indicate whether song playing or not.
pause()	It is used to pause the song playing.
setAudioStreamType()	it is used to specify the audio streaming type.
setDataSource()	It is used to specify the path of audio / video file to play.
setVolume()	It is used to adjust media player volume either up / down.
seekTo(position)	It is used to move song to particular position in milliseconds.

Method	Description
getTrackInfo()	It returns an array of track information.
start()	It is used to start playing the audio/video.
stop()	It is used to stop playing the audio/video.
reset()	It is used to reset the MediaPlayer object.
release()	It is used to releases the resources which are associated with MediaPlayer object

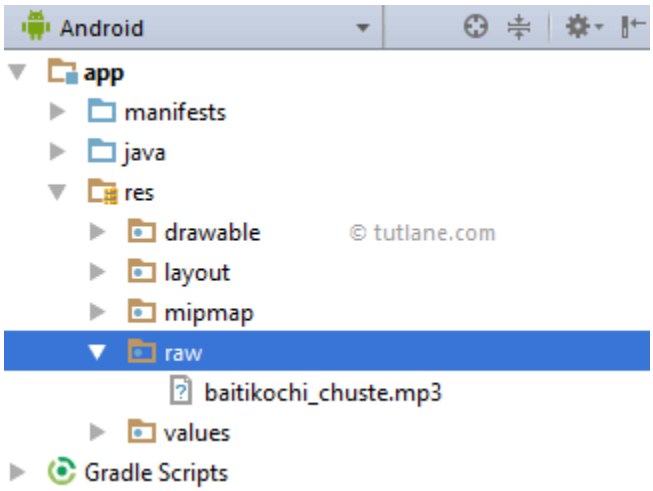
Now we will see how to implement media playing application using **MediaPlayer** to play a song or audio with multiple playback options, such as play, pause, forward, backward in android application with examples.

Android Audio Player Example

Following is the example of implementing an audio player to play a song or audio with multiple playback options using **MediaPlayer**.

Create a new android application using android studio and give names as **MediaPlayerExample**. In case if you are not aware of creating an app in android studio check this article Android Hello World App.

As discussed create a new **raw** folder in **res** directory and add one music file like as shown below to play it by using **MediaPlayer** class.



Now open **activity_main.xml** file from **\res\layout** folder 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/andro
id"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="10dp"
    android:paddingRight="10dp">
    <TextView
        android:id="@+id/txtVw1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Now Playing: "
        android:layout_marginTop="30dp"
        android:textAppearance="?android:attr/textAppearanceMedium" />
    <TextView
        android:id="@+id/txtSname"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/txtVw1"
        android:layout_toRightOf="@+id/txtVw1"
        android:text="TextView" />
    <ImageView
        android:id="@+id/imgLogo"
        android:layout_width="match_parent"
        android:layout_height="450dp"
        android:layout_below="@+id/txtVw1"
        android:src="@drawable/tutlane" />
    <ImageButton
        android:id="@+id/btnBackward"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="44dp"
        android:layout_marginLeft="20dp"
        android:src="@android:drawable/ic_media_rew" />
    <ImageButton
        android:id="@+id/btnPlay"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/btnBackward"
        android:layout_marginLeft="20dp"
        android:layout_toRightOf="@+id/btnBackward"
        android:src="@android:drawable/ic_media_play" />
    <ImageButton
        android:id="@+id/btnPause"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/btnPlay"
        android:layout_marginLeft="20dp"
        android:layout_toRightOf="@+id/btnPlay"
        android:src="@android:drawable/ic_media_pause" />
    <ImageButton
        android:id="@+id/btnForward"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/btnPause"
        android:layout_marginLeft="20dp"
        android:layout_toRightOf="@+id/btnPause"
        android:contentDescription="@+id/imageButton3"
        android:src="@android:drawable/ic_media_ff" />
      <TextView
        android:id="@+id/txtStartTime"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/sBar"
        android:text="0 min, 0 sec" />
    <SeekBar
        android:id="@+id/sBar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_above="@+id/btnBackward"
        android:layout_toLeftOf="@+id/txtSongTime"
        android:layout_toRightOf="@+id/txtStartTime" />
    <TextView
        android:id="@+id/txtSongTime"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@+id/btnForward"
        android:layout_alignTop="@+id/sBar"
        android:text="0 min, 0 sec " />
</RelativeLayout>
```

Now open your main activity file **MainActivity.java** from **\java\com.tutlane.audioplayerexample** path and write the code like as shown below.

MainActivity.java

```
package com.tutlane.mediaplayerexample;
import android.media.MediaPlayer;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageButton;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;
```

```
import java.util.concurrent.TimeUnit;

public class MainActivity extends AppCompatActivity {
    private ImageButton forwardbtn, backwardbtn, pausebtn, playbtn;
    private MediaPlayer mPlayer;
    private TextView songName, startTime, songTime;
    private SeekBar songPrgs;
    private static int oTime =0, sTime =0, eTime =0, fTime = 5000, bTim
e = 5000;
    private Handler hdlr = new Handler();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        backwardbtn = (ImageButton)findViewById(R.id.btnBackward);
        forwardbtn = (ImageButton)findViewById(R.id.btnForward);
        playbtn = (ImageButton)findViewById(R.id.btnPlay);
        pausebtn = (ImageButton)findViewById(R.id.btnPause);
        songName = (TextView)findViewById(R.id.txtSname);
        startTime = (TextView)findViewById(R.id.txtStartTime);
        songTime = (TextView)findViewById(R.id.txtSongTime);
        songName.setText("Baitikochi Chuste");
        mPlayer = MediaPlayer.create(this, R.raw.baitikochi_chuste);
        songPrgs = (SeekBar)findViewById(R.id.sBar);
        songPrgs.setClickable(false);
        pausebtn.setEnabled(false);

        playbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(MainActivity.this, "Playing Audio", Toas
t.LENGTH_SHORT).show();
                mPlayer.start();
                eTime = mPlayer.getDuration();
                sTime = mPlayer.getCurrentPosition();
                if(oTime == 0){
                    songPrgs.setMax(eTime);
                    oTime =1;
                }
                songTime.setText(String.format("%d min, %d sec", TimeUn
it.MILLISECONDS.toMinutes(eTime),
                        TimeUnit.MILLISECONDS.toSeconds(eTime) - TimeUn
it.MINUTES.toSeconds(TimeUnit.MILLISECONDS. toMinutes(eTime))) );
                startTime.setText(String.format("%d min, %d sec", TimeU
nit.MILLISECONDS.toMinutes(sTime),
                        TimeUnit.MILLISECONDS.toSeconds(sTime) - TimeUn
it.MINUTES.toSeconds(TimeUnit.MILLISECONDS. toMinutes(sTime))) );
                songPrgs.setProgress(sTime);
                hdlr.postDelayed(UpdateSongTime, 100);
                pausebtn.setEnabled(true);
                playbtn.setEnabled(false);
```

```
            }
        });
        pausebtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mPlayer.pause();
                pausebtn.setEnabled(false);
                playbtn.setEnabled(true);
                Toast.makeText(getApplicationContext(),"Pausing Audio",
Toast.LENGTH_SHORT).show();
            }
        });
        forwardbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if((sTime + fTime) <= eTime)
                {
                    sTime = sTime + fTime;
                    mPlayer.seekTo(sTime);
                }
                else
                {
                    Toast.makeText(getApplicationContext(), "Cannot jum
p forward 5 seconds", Toast.LENGTH_SHORT).show();
                }
                if(!playbtn.isEnabled()){
                    playbtn.setEnabled(true);
                }
            }
        });
        backwardbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if((sTime - bTime) > 0)
                {
                    sTime = sTime - bTime;
                    mPlayer.seekTo(sTime);
                }
                else
                {
                    Toast.makeText(getApplicationContext(), "Cannot jum
p backward 5 seconds", Toast.LENGTH_SHORT).show();
                }
                if(!playbtn.isEnabled()){
                    playbtn.setEnabled(true);
                }
            }
        });
    }
    private Runnable UpdateSongTime = new Runnable() {
        @Override
```

```
        public void run() {
            sTime = mPlayer.getCurrentPosition();
            startTime.setText(String.format("%d min, %d sec", TimeUnit.
MILLISECONDS.toMinutes(sTime),
                    TimeUnit.MILLISECONDS.toSeconds(sTime) - TimeUnit.M
INUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes(sTime))) );
            songPrgs.setProgress(sTime);
            hdlr.postDelayed(this, 100);
        }
    };
}
```

If you observe above code we used **MeidaPlayer** object properties to play, pause song and changing the song position either forward or backward based on our requirements.

Android Alarm Clock Tutorial

Last modified on June 19th, 2017 by Joe.

This Android tutorial will walk you through to create an alarm clock Android application. This alarm app is planned to be minimalistic and usable. It can set alarm for one occurrence for the coming day. You will get alarm ring sound, a notification message and a message in the app UI. This application and device can be idle or sleeping when the alarm triggers.

We will be using the `AlarmManager` API to set and ring the alarm notification. We will have a TimePicker component and a toggle switch in the UI to set the alarm time.

Android Alarm Clock Application

Application is planned to be simple as possible and you can use this as a base framework and enhance it by adding fancy features.

1. Android Manifest

AndroidManifest.xml

We need to give uses-permission for `WAKE_LOCK`, other than that the AndroidManifest.xml is pretty standard one. Just need to include the service and receiver.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.javapapers.androidalarmclock">

    <uses-permission android:name="android.permission.WAKE_LOCK" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme">
        <activity
            android:name=".AlarmActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
        <service
            android:name=".AlarmService"
            android:enabled="true" />
        <receiver android:name=".AlarmReceiver" />
    </application>
```

```
</manifest>
```

2. Android Activity

activity_my.xml

The Android Activity is designed to be simple. We have a `TimePicker` component followed by a `ToggleButton`. That's it. Choose the time to set the alarm and toggle the switch to on. The alarm will work.

```
<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:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MyActivity">

    <TimePicker
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/alarmTimePicker"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <ToggleButton
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"

        android:text="Alarm On/Off"

        android:id="@+id/alarmToggle"

        android:layout_centerHorizontal="true"

        android:layout_below="@+id/alarmTimePicker"

        android:onClick="onToggleClicked" />


    <TextView

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:textAppearance="?android:attr/textAppearanceLarge"

        android:text=""

        android:id="@+id/alarmText"

        android:layout_alignParentBottom="true"

        android:layout_centerHorizontal="true"

        android:layout_marginTop="20dp"

        android:layout_below="@+id/alarmToggle" />


</RelativeLayout>
```

AlarmActivity.java

AlarmActivity uses the AlarmManager to set the alarm and send notification on alarm trigger.

```
package com.javapapers.androidalarmclock;


import android.app.Activity;

import android.app.AlarmManager;

import android.app.PendingIntent;
```

```
import android.content.Intent;

import android.os.Bundle;

import android.util.Log;

import android.view.View;

import android.widget.TextView;

import android.widget.TimePicker;

import android.widget.ToggleButton;


import java.util.Calendar;


public class AlarmActivity extends Activity {


    AlarmManager alarmManager;

    private PendingIntent pendingIntent;

    private TimePicker alarmTimePicker;

    private static AlarmActivity inst;

    private TextView alarmTextView;


    public static AlarmActivity instance() {

        return inst;

    }


    @Override

    public void onStart() {

        super.onStart();

        inst = this;

    }
```

```
    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_my);

        alarmTimePicker = (TimePicker) findViewById(R.id.alarmTimePicker);

        alarmTextView = (TextView) findViewById(R.id.alarmText);

        ToggleButton alarmToggle = (ToggleButton)
findViewById(R.id.alarmToggle);

        alarmManager = (AlarmManager) getSystemService(ALARM_SERVICE);

    }



    public void onToggleClicked(View view) {

        if (((ToggleButton) view).isChecked()) {

            Log.d("MyActivity", "Alarm On");

            Calendar calendar = Calendar.getInstance();

            calendar.set(Calendar.HOUR_OF_DAY,
alarmTimePicker.getCurrentHour());

            calendar.set(Calendar.MINUTE,
alarmTimePicker.getCurrentMinute());

            Intent myIntent = new Intent(AlarmActivity.this,
AlarmReceiver.class);

            pendingIntent = PendingIntent.getBroadcast(AlarmActivity.this,
0, myIntent, 0);

            alarmManager.set(AlarmManager.RTC, calendar.getTimeInMillis(),
pendingIntent);

        } else {

            alarmManager.cancel(pendingIntent);

            setAlarmText("");

            Log.d("MyActivity", "Alarm Off");

        }

    }
```

```
    public void setAlarmText(String alarmText) {

        alarmTextView.setText(alarmText);

    }

}
```

3. Alarm Receiver

AlarmReceiver.java

AlarmReceiver is a `WakefulBroadcasReceiver`, this is the one that receives the alarm trigger on set time. From here we initiate different actions to notify the user as per our choice. I have given three type of notifications, first show a message to user in the activity UI, second play the alarm ringtone and third send an Android notification message. So this is the place to add enhancement for different types of user notifications.

```
package com.javapapers.androidalarmclock;


import android.app.Activity;

import android.content.ComponentName;

import android.content.Context;

import android.content.Intent;

import android.media.Ringtone;

import android.media.RingtoneManager;

import android.net.Uri;

import android.support.v4.content.WakefulBroadcastReceiver;


public class AlarmReceiver extends WakefulBroadcastReceiver {
```

```
    @Override

    public void onReceive(final Context context, Intent intent) {

        //this will update the UI with message

        AlarmActivity inst = AlarmActivity.instance();

        inst.setAlarmText("Alarm! Wake up! Wake up!");



        //this will sound the alarm tone

        //this will sound the alarm once, if you wish to

        //raise alarm in loop continuously then use MediaPlayer and
setLooping(true)

        Uri alarmUri =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_ALARM);

        if (alarmUri == null) {

            alarmUri =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);

        }

        Ringtone ringtone = RingtoneManager.getRingtone(context, alarmUri);

        ringtone.play();



        //this will send a notification message

        ComponentName comp = new ComponentName(context.getPackageName(),

                AlarmService.class.getName());

        startWakefulService(context, (intent.setComponent(comp)));

        setResultCode(Activity.RESULT_OK);

    }

}
```

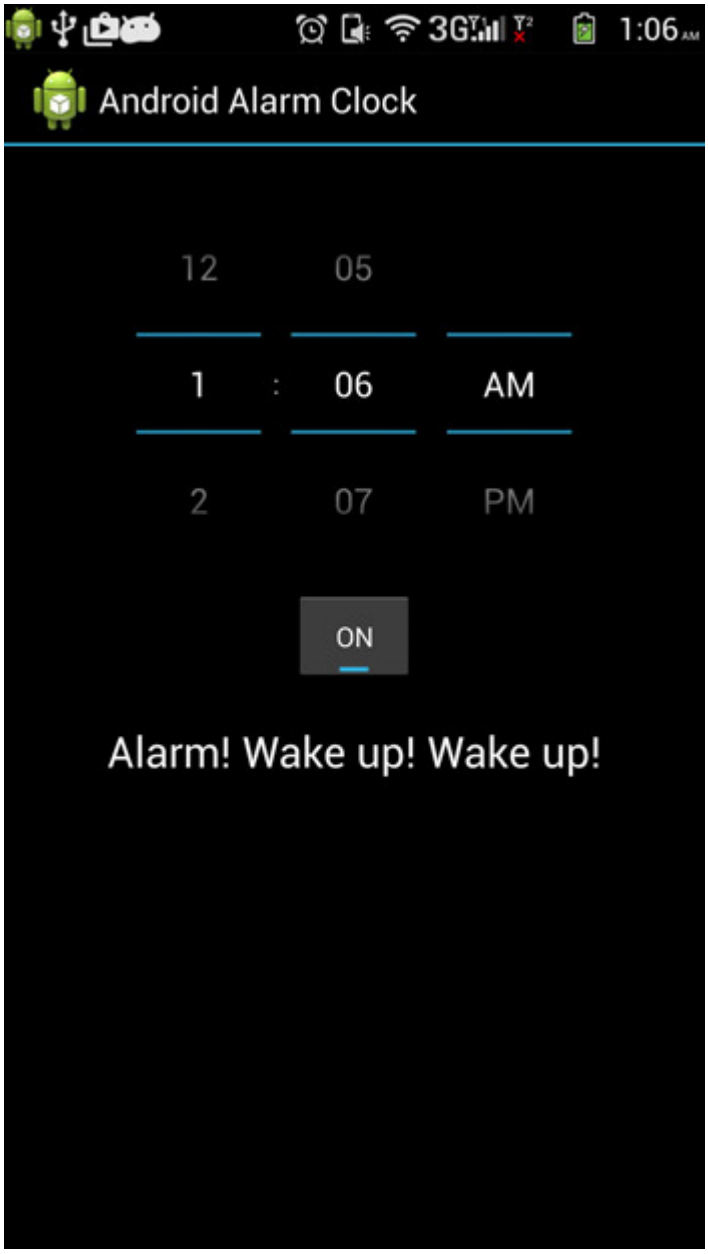
4. Alarm Notification Message

AlarmService.java

The receiver will start the following `IntentService` to send a standard notification to the user.

```
package com.javapapers.androidalarmclock;


import android.app.IntentService;

import android.app.NotificationManager;

import android.app.PendingIntent;

import android.content.Context;

import android.content.Intent;

import android.support.v4.app.NotificationCompat;

import android.util.Log;


public class AlarmService extends IntentService {

    private NotificationManager alarmNotificationManager;


    public AlarmService() {

        super("AlarmService");

    }


    @Override

    public void onHandleIntent(Intent intent) {

        sendNotification("Wake Up! Wake Up!");

    }


    private void sendNotification(String msg) {

        Log.d("AlarmService", "Preparing to send notification...: " + msg);

        alarmNotificationManager = (NotificationManager) this

                .getSystemService(Context.NOTIFICATION_SERVICE);
```

```
        PendingIntent contentIntent = PendingIntent.getActivity(this, 0,

                new Intent(this, AlarmActivity.class), 0);



        NotificationCompat.Builder alamNotificationBuilder = new
NotificationCompat.Builder(


this).setContentTitle("Alarm").setSmallIcon(R.drawable.ic_launcher)

                .setStyle(new
NotificationCompat.BigTextStyle().bigText(msg))

                .setContentText(msg);




        alamNotificationBuilder.setContentIntent(contentIntent);

        alarmNotificationManager.notify(1,
alamNotificationBuilder.build());

        Log.d("AlarmService", "Notification sent.");

    }
```



SQLite

AddressBook App

Following is the content of the modified **MainActivity.java**.

```
package com.example.sairamkrishna.myapplication;

import android.content.Context;
import android.content.Intent;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;

import android.view.KeyEvent;
import android.view.Menu;
```

```
import android.view.MenuItem;
import android.view.View;

import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;

import java.util.ArrayList;
import java.util.List;

public class MainActivity extends ActionBarActivity {
   public final static String EXTRA_MESSAGE = "MESSAGE";
   private ListView obj;
   DBHelper mydb;

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

      mydb = new DBHelper(this);
      ArrayList array_list = mydb.getAllCotacts();
      ArrayAdapter arrayAdapter=new
ArrayAdapter(this,android.R.layout.simple_list_item_1,
array_list);

      obj = (ListView)findViewById(R.id.listView1);
      obj.setAdapter(arrayAdapter);
      obj.setOnItemClickListener(new OnItemClickListener(){
         @Override
         public void onItemClick(AdapterView<?> arg0, View arg1,
int arg2,long arg3) {
            // TODO Auto-generated method stub
            int id_To_Search = arg2 + 1;

            Bundle dataBundle = new Bundle();
            dataBundle.putInt("id", id_To_Search);

            Intent intent = new
Intent(getApplicationContext(),DisplayContact.class);

            intent.putExtras(dataBundle);
            startActivity(intent);
         }
      });
   }

   @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;
   }

   @Override
   public boolean onOptionsItemSelected(MenuItem item){
      super.onOptionsItemSelected(item);

      switch(item.getItemId()) {
         case R.id.item1:Bundle dataBundle = new Bundle();
         dataBundle.putInt("id", 0);

         Intent intent = new
Intent(getApplicationContext(),DisplayContact.class);
         intent.putExtras(dataBundle);

         startActivity(intent);
         return true;
         default:
         return super.onOptionsItemSelected(item);
      }
   }

   public boolean onKeyDown(int keycode, KeyEvent event) {
      if (keycode == KeyEvent.KEYCODE_BACK) {
         moveTaskToBack(true);
      }
      return super.onKeyDown(keycode, event);
   }
}
```

Following is the modified content of display contact activity **DisplayContact.java**

```
package com.example.sairamkrishna.myapplication;

import android.os.Bundle;
import android.app.Activity;
import android.app.AlertDialog;

import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;

import android.view.Menu;
import android.view.MenuItem;
import android.view.View;

import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class DisplayContact extends Activity {
   int from_Where_I_Am_Coming = 0;
   private DBHelper mydb ;
```

```
   TextView name ;
   TextView phone;
   TextView email;
   TextView street;
   TextView place;
   int id_To_Update = 0;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_display_contact);
      name = (TextView) findViewById(R.id.editTextName);
      phone = (TextView) findViewById(R.id.editTextPhone);
      email = (TextView) findViewById(R.id.editTextStreet);
      street = (TextView) findViewById(R.id.editTextEmail);
      place = (TextView) findViewById(R.id.editTextCity);

      mydb = new DBHelper(this);

      Bundle extras = getIntent().getExtras();
      if(extras !=null) {
         int Value = extras.getInt("id");

         if(Value>0){
            //means this is the view part not the add contact
part.
            Cursor rs = mydb.getData(Value);
            id_To_Update = Value;
            rs.moveToFirst();

            String nam =
rs.getString(rs.getColumnIndex(DBHelper.CONTACTS_COLUMN_NAME));
            String phon =
rs.getString(rs.getColumnIndex(DBHelper.CONTACTS_COLUMN_PHONE));
            String emai =
rs.getString(rs.getColumnIndex(DBHelper.CONTACTS_COLUMN_EMAIL));
            String stree =
rs.getString(rs.getColumnIndex(DBHelper.CONTACTS_COLUMN_STREET));
            String plac =
rs.getString(rs.getColumnIndex(DBHelper.CONTACTS_COLUMN_CITY));

            if (!rs.isClosed())  {
               rs.close();
            }
            Button b = (Button)findViewById(R.id.button1);
            b.setVisibility(View.INVISIBLE);

            name.setText((CharSequence)nam);
            name.setFocusable(false);
            name.setClickable(false);

            phone.setText((CharSequence)phon);
            phone.setFocusable(false);
```

```
            phone.setClickable(false);

            email.setText((CharSequence)emai);
            email.setFocusable(false);
            email.setClickable(false);

            street.setText((CharSequence)stree);
            street.setFocusable(false);
            street.setClickable(false);

            place.setText((CharSequence)plac);
            place.setFocusable(false);
            place.setClickable(false);
         }
      }
   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the menu; this adds items to the action bar if
it is present.
      Bundle extras = getIntent().getExtras();

      if(extras !=null) {
         int Value = extras.getInt("id");
         if(Value>0){
            getMenuInflater().inflate(R.menu.display_contact,
menu);
         } else{
            getMenuInflater().inflate(R.menu.menu_main menu);
         }
      }
      return true;
   }

   public boolean onOptionsItemSelected(MenuItem item) {
      super.onOptionsItemSelected(item);
      switch(item.getItemId()) {
         case R.id.Edit_Contact:
         Button b = (Button)findViewById(R.id.button1);
         b.setVisibility(View.VISIBLE);
         name.setEnabled(true);
         name.setFocusableInTouchMode(true);
         name.setClickable(true);

         phone.setEnabled(true);
         phone.setFocusableInTouchMode(true);
         phone.setClickable(true);

         email.setEnabled(true);
         email.setFocusableInTouchMode(true);
         email.setClickable(true);
```

```
         street.setEnabled(true);
         street.setFocusableInTouchMode(true);
         street.setClickable(true);

         place.setEnabled(true);
         place.setFocusableInTouchMode(true);
         place.setClickable(true);

         return true;
         case R.id.Delete_Contact:

         AlertDialog.Builder builder = new
AlertDialog.Builder(this);
         builder.setMessage(R.string.deleteContact)
            .setPositiveButton(R.string.yes, new
DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog, int
id) {
                  mydb.deleteContact(id_To_Update);
                  Toast.makeText(getApplicationContext(),
"Deleted Successfully",
                     Toast.LENGTH_SHORT).show();
                  Intent intent = new
Intent(getApplicationContext(),MainActivity.class);
                  startActivity(intent);
               }
         })
         .setNegativeButton(R.string.no, new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
               // User cancelled the dialog
            }
         });

         AlertDialog d = builder.create();
         d.setTitle("Are you sure");
         d.show();

         return true;
         default:
         return super.onOptionsItemSelected(item);

      }
   }

   public void run(View view) {
      Bundle extras = getIntent().getExtras();
      if(extras !=null) {
         int Value = extras.getInt("id");
         if(Value>0){

if(mydb.updateContact(id_To_Update,name.getText().toString(),
```

```
                phone.getText().toString(),
email.getText().toString(),
                                street.getText().toString(),
place.getText().toString())){
                Toast.makeText(getApplicationContext(), "Updated",
Toast.LENGTH_SHORT).show();
                Intent intent = new
Intent(getApplicationContext(),MainActivity.class);
                startActivity(intent);
            } else{
                Toast.makeText(getApplicationContext(), "not
Updated", Toast.LENGTH_SHORT).show();
            }
         } else{
            if(mydb.insertContact(name.getText().toString(),
phone.getText().toString(),
                                email.getText().toString(),
street.getText().toString(),
                                place.getText().toString())){
                  Toast.makeText(getApplicationContext(), "done",

Toast.LENGTH_SHORT).show();
            } else{
                Toast.makeText(getApplicationContext(), "not
done",
                                        Toast.LENGTH_SHORT).show();

            }
            Intent intent = new
Intent(getApplicationContext(),MainActivity.class);
            startActivity(intent);
         }
      }
   }
}
```

Following is the content of Database class **DBHelper.java**

```
package com.example.sairamkrishna.myapplication;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Hashtable;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.DatabaseUtils;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteDatabase;

public class DBHelper extends SQLiteOpenHelper {

   public static final String DATABASE_NAME = "MyDBName.db";
   public static final String CONTACTS_TABLE_NAME = "contacts";
```

```
   public static final String CONTACTS_COLUMN_ID = "id";
   public static final String CONTACTS_COLUMN_NAME = "name";
   public static final String CONTACTS_COLUMN_EMAIL = "email";
   public static final String CONTACTS_COLUMN_STREET = "street";
   public static final String CONTACTS_COLUMN_CITY = "place";
   public static final String CONTACTS_COLUMN_PHONE = "phone";
   private HashMap hp;

   public DBHelper(Context context) {
      super(context, DATABASE_NAME , null, 1);
   }

   @Override
   public void onCreate(SQLiteDatabase db) {
      // TODO Auto-generated method stub
      db.execSQL(
         "create table contacts " +
         "(id integer primary key, name text,phone text,email
text, street text,place text)"
      );
   }

   @Override
   public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
      // TODO Auto-generated method stub
      db.execSQL("DROP TABLE IF EXISTS contacts");
      onCreate(db);
   }

   public boolean insertContact (String name, String phone,
String email, String street,String place) {
      SQLiteDatabase db = this.getWritableDatabase();
      ContentValues contentValues = new ContentValues();
      contentValues.put("name", name);
      contentValues.put("phone", phone);
      contentValues.put("email", email);
      contentValues.put("street", street);
      contentValues.put("place", place);
      db.insert("contacts", null, contentValues);
      return true;
   }

   public Cursor getData(int id) {
      SQLiteDatabase db = this.getReadableDatabase();
      Cursor res =  db.rawQuery( "select * from contacts where
id="+id+"", null );
      return res;
   }

   public int numberOfRows(){
      SQLiteDatabase db = this.getReadableDatabase();
```

```
      int numRows = (int) DatabaseUtils.queryNumEntries(db,
CONTACTS_TABLE_NAME);
      return numRows;
   }

   public boolean updateContact (Integer id, String name, String
phone, String email, String street,String place) {
      SQLiteDatabase db = this.getWritableDatabase();
      ContentValues contentValues = new ContentValues();
      contentValues.put("name", name);
      contentValues.put("phone", phone);
      contentValues.put("email", email);
      contentValues.put("street", street);
      contentValues.put("place", place);
      db.update("contacts", contentValues, "id = ? ", new
String[] { Integer.toString(id) } );
      return true;
   }

   public Integer deleteContact (Integer id) {
      SQLiteDatabase db = this.getWritableDatabase();
      return db.delete("contacts",
      "id = ? ",
      new String[] { Integer.toString(id) });
   }

   public ArrayList<String> getAllCotacts() {
      ArrayList<String> array_list = new ArrayList<String>();

      //hp = new HashMap();
      SQLiteDatabase db = this.getReadableDatabase();
      Cursor res =  db.rawQuery( "select * from contacts", null
);
      res.moveToFirst();

      while(res.isAfterLast() == false){

array_list.add(res.getString(res.getColumnIndex(CONTACTS_COLUMN_N
AME)));
         res.moveToNext();
      }
      return array_list;
   }
}
```

Following is the content of the **res/layout/activity_main.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"
   android:paddingLeft="@dimen/activity_horizontal_margin"
```

```
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".MainActivity">

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/textView"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:textSize="30dp"
      android:text="Data Base" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Tutorials Point"
      android:id="@+id/textView2"
      android:layout_below="@+id/textView"
      android:layout_centerHorizontal="true"
      android:textSize="35dp"
      android:textColor="#ff16ff01" />

   <ImageView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/imageView"
      android:layout_below="@+id/textView2"
      android:layout_centerHorizontal="true"
      android:src="@drawable/logo"/>

   <ScrollView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/scrollView"
      android:layout_below="@+id/imageView"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true"
      android:layout_alignParentBottom="true"
      android:layout_alignParentRight="true"
      android:layout_alignParentEnd="true">

      <ListView
         android:id="@+id/listView1"
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:layout_centerHorizontal="true"
         android:layout_centerVertical="true" >
      </ListView>

   </ScrollView>
```

```
</RelativeLayout>
```

Following is the content of the **res/layout/activity_display_contact.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/scrollView1"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   tools:context=".DisplayContact" >

   <RelativeLayout
      android:layout_width="match_parent"
      android:layout_height="370dp"
      android:paddingBottom="@dimen/activity_vertical_margin"
      android:paddingLeft="@dimen/activity_horizontal_margin"
      android:paddingRight="@dimen/activity_horizontal_margin"
      android:paddingTop="@dimen/activity_vertical_margin">

      <EditText
         android:id="@+id/editTextName"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignParentLeft="true"
         android:layout_marginTop="5dp"
         android:layout_marginLeft="82dp"
         android:ems="10"
         android:inputType="text" >
      </EditText>

      <EditText
         android:id="@+id/editTextEmail"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignLeft="@+id/editTextStreet"
         android:layout_below="@+id/editTextStreet"
         android:layout_marginTop="22dp"
         android:ems="10"
         android:inputType="textEmailAddress" />

      <TextView
         android:id="@+id/textView1"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignBottom="@+id/editTextName"
         android:layout_alignParentLeft="true"
         android:text="@string/name"

android:textAppearance="?android:attr/textAppearanceMedium" />

      <Button
         android:id="@+id/button1"
```

```
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignLeft="@+id/editTextCity"
         android:layout_alignParentBottom="true"
         android:layout_marginBottom="28dp"
         android:onClick="run"
         android:text="@string/save" />

      <TextView
         android:id="@+id/textView2"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignBottom="@+id/editTextEmail"
         android:layout_alignLeft="@+id/textView1"
         android:text="@string/email"

android:textAppearance="?android:attr/textAppearanceMedium" />

      <TextView
         android:id="@+id/textView5"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignBottom="@+id/editTextPhone"
         android:layout_alignLeft="@+id/textView1"
         android:text="@string/phone"

android:textAppearance="?android:attr/textAppearanceMedium" />

      <TextView
         android:id="@+id/textView4"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_above="@+id/editTextEmail"
         android:layout_alignLeft="@+id/textView5"
         android:text="@string/street"

android:textAppearance="?android:attr/textAppearanceMedium" />

      <EditText
         android:id="@+id/editTextCity"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignRight="@+id/editTextName"
         android:layout_below="@+id/editTextEmail"
         android:layout_marginTop="30dp"
         android:ems="10"
         android:inputType="text" />

      <TextView
         android:id="@+id/textView3"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignBaseline="@+id/editTextCity"
```

```
         android:layout_alignBottom="@+id/editTextCity"
         android:layout_alignParentLeft="true"
         android:layout_toLeftOf="@+id/editTextEmail"
         android:text="@string/country"

android:textAppearance="?android:attr/textAppearanceMedium" />

      <EditText
         android:id="@+id/editTextStreet"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignLeft="@+id/editTextName"
         android:layout_below="@+id/editTextPhone"
         android:ems="10"
         android:inputType="text" >

         <requestFocus />
      </EditText>

      <EditText
         android:id="@+id/editTextPhone"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignLeft="@+id/editTextStreet"
         android:layout_below="@+id/editTextName"
         android:ems="10"
         android:inputType="phone|text" />

   </RelativeLayout>
</ScrollView>
```

Following is the content of the **res/value/string.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="app_name">Address Book</string>
   <string name="action_settings">Settings</string>
   <string name="hello_world">Hello world!</string>
   <string name="Add_New">Add New</string>
   <string name="edit">Edit Contact</string>
   <string name="delete">Delete Contact</string>
   <string
name="title_activity_display_contact">DisplayContact</string>
   <string name="name">Name</string>
   <string name="phone">Phone</string>
   <string name="email">Email</string>
   <string name="street">Street</string>
   <string name="country">City/State/Zip</string>
   <string name="save">Save Contact</string>
   <string name="deleteContact">Are you sure, you want to delete
it.</string>
   <string name="yes">Yes</string>
   <string name="no">No</string>
```

```
</resources>
```

Following is the content of the **res/menu/main_menu.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
>

   <item android:id="@+id/item1"
      android:icon="@drawable/add"
      android:title="@string/Add_New" >
   </item>

</menu>
```

Following is the content of the **res/menu/display_contact.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
>
   <item
      android:id="@+id/Edit_Contact"
      android:orderInCategory="100"
      android:title="@string/edit"/>

   <item
      android:id="@+id/Delete_Contact"
      android:orderInCategory="100"
      android:title="@string/delete"/>

</menu>
```

This is the defualt **AndroidManifest.xml** of this project

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.sairamkrishna.myapplication" >

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >

      <activity
         android:name=".MainActivity"
         android:label="@string/app_name" >

         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category
android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
```

```
      </activity>

      <activity android:name=".DisplayContact"/>

   </application>
</manifest>
```

Option 2: SQLite

Example of android SQLite database

Let's see the simple example of android sqlite database.

File: Contact.java

```
1.  package example.javatpoint.com.sqlitetutorial;
2.
3.  public class Contact {
    int _id;
    String _name;
    String _phone_number;
    public Contact(){   }
    public Contact(int id, String name, String _phone_number){
        this._id = id;
        this._name = name;
        this._phone_number = _phone_number;
    }

    public Contact(String name, String _phone_number){
        this._name = name;
        this._phone_number = _phone_number;
    }
    public int getID(){
        return this._id;
    }

    public void setID(int id){
        this._id = id;
    }
```

```
    public String getName(){
        return this._name;
    }

    public void setName(String name){
        this._name = name;
    }

    public String getPhoneNumber(){
        return this._phone_number;
    }

    public void setPhoneNumber(String phone_number){
        this._phone_number = phone_number;
    }
4.  }
```

File: DatabaseHandler.java

Now, let's create the database handler class that extends SQLiteOpenHelper class and provides the implementation of its methods.

```
package example.javatpoint.com.sqlitetutorial;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import java.util.ArrayList;
import java.util.List;



public class DatabaseHandler extends SQLiteOpenHelper {
    private static final int DATABASE_VERSION = 1;
    private static final String DATABASE_NAME = "contactsManager";
    private static final String TABLE_CONTACTS = "contacts";
    private static final String KEY_ID = "id";
    private static final String KEY_NAME = "name";
```

```
private static final String KEY_PH_NO = "phone_number";

public DatabaseHandler(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
   //3rd argument to be passed is CursorFactory instance
}

// Creating Tables
@Override
public void onCreate(SQLiteDatabase db) {
   String CREATE_CONTACTS_TABLE = "CREATE TABLE " + TABLE_CONTACTS + "("
           + KEY_ID + " INTEGER PRIMARY KEY," + KEY_NAME + " TEXT,"
          + KEY_PH_NO + " TEXT" + ")";
    db.execSQL(CREATE_CONTACTS_TABLE);
}

// Upgrading database
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // Drop older table if existed
   db.execSQL("DROP TABLE IF EXISTS " + TABLE_CONTACTS);

   // Create tables again
    onCreate(db);
}

// code to add the new contact
void addContact(Contact contact) {
   SQLiteDatabase db = this.getWritableDatabase();

   ContentValues values = new ContentValues();
    values.put(KEY_NAME, contact.getName()); // Contact Name
   values.put(KEY_PH_NO, contact.getPhoneNumber()); // Contact Phone

   // Inserting Row
    db.insert(TABLE_CONTACTS, null, values);
   //2nd argument is String containing nullColumnHack
```

```
    db.close(); // Closing database connection
}

// code to get the single contact
Contact getContact(int id) {
    SQLiteDatabase db = this.getReadableDatabase();

    Cursor cursor = db.query(TABLE_CONTACTS, new String[] { KEY_ID,
                    KEY_NAME, KEY_PH_NO }, KEY_ID + "=?",
            new String[] { String.valueOf(id) }, null, null, null, null);
    if (cursor != null)
        cursor.moveToFirst();

    Contact contact = new Contact(Integer.parseInt(cursor.getString(0)),
            cursor.getString(1), cursor.getString(2));
    // return contact
    return contact;
}

// code to get all contacts in a list view
public List<Contact> getAllContacts() {
    List<Contact> contactList = new ArrayList<Contact>();
    // Select All Query
    String selectQuery = "SELECT  * FROM " + TABLE_CONTACTS;

    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(selectQuery, null);

    // looping through all rows and adding to list
    if (cursor.moveToFirst()) {
        do {
            Contact contact = new Contact();
            contact.setID(Integer.parseInt(cursor.getString(0)));
            contact.setName(cursor.getString(1));
            contact.setPhoneNumber(cursor.getString(2));
            // Adding contact to list
            contactList.add(contact);
```

```
        } while (cursor.moveToNext());
    }

    // return contact list
    return contactList;
}

// code to update the single contact
public int updateContact(Contact contact) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(KEY_NAME, contact.getName());
    values.put(KEY_PH_NO, contact.getPhoneNumber());

    // updating row
    return db.update(TABLE_CONTACTS, values, KEY_ID + " = ?",
            new String[] { String.valueOf(contact.getID()) });
}

// Deleting single contact
public void deleteContact(Contact contact) {
    SQLiteDatabase db = this.getWritableDatabase();
    db.delete(TABLE_CONTACTS, KEY_ID + " = ?",
            new String[] { String.valueOf(contact.getID()) });
    db.close();
}

// Getting contacts Count
public int getContactsCount() {
    String countQuery = "SELECT  * FROM " + TABLE_CONTACTS;
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.rawQuery(countQuery, null);
    cursor.close();

    // return count
    return cursor.getCount();
```

```
    }

}
```

File: MainActivity.java

```
1.  package example.javatpoint.com.sqlitetutorial;
2.
3.  import android.support.v7.app.AppCompatActivity;
4.  import android.os.Bundle;
5.  import android.util.Log;
6.  import java.util.List;
7.
8.  public class MainActivity extends AppCompatActivity {
9.
10.   @Override
11.   protected void onCreate(Bundle savedInstanceState) {
12.     super.onCreate(savedInstanceState);
13.      setContentView(R.layout.activity_main);
14.     DatabaseHandler db = new DatabaseHandler(this);
15.
16.     // Inserting Contacts
17.      Log.d("Insert: ", "Inserting ..");
18.     db.addContact(new Contact("Ravi", "9100000000"));
19.      db.addContact(new Contact("Srinivas", "9199999999"));
20.     db.addContact(new Contact("Tommy", "9522222222"));
21.      db.addContact(new Contact("Karthik", "9533333333"));
22.
23.      // Reading all contacts
24.     Log.d("Reading: ", "Reading all contacts..");
25.      List<Contact> contacts = db.getAllContacts();
26.
27.      for (Contact cn : contacts) {
28.        String log = "Id: " + cn.getID() + " ,Name: " + cn.getName() + " ,Phone: " +
29.                cn.getPhoneNumber();
30.        // Writing Contacts to log
31.         Log.d("Name: ", log);
32.     }
```

```
33.     }
34. }
```