❖ App Development Using Java.

• Android apps written in Java 6+ – Everything we've learned still holds

• Apps use four main components: – Activity: A "single screen" that's visible to user – Service: Long-running background "part" of app (not separate process or thread) – ContentProvider: Manages app data (usually stored in database) and data access for queries – BroadcastReceiver: Component that listens for particular Android system "events", e.g., "found wireless device", and responds accordingly 46 App Manifest

• Every Android app must include an AndroidManifest.xml file describing functionality

• The manifest specifies: – App's Activities, Services, etc. – Permissions requested by app – Minimum API required – Hardware features required, e.g., camera with autofocus 47 Activity Lifecycle

• Activity: key building block of Android apps

• Extend Activity class, override onCreate(), onPause(), onResume() methods

• Dalvik VM can stop any Activity without warning, so saving state is important!
• Activities need to be “responsive”, otherwise Android shows user “App Not Responsive” warning: – Place lengthy operations in Runnable Threads, AsyncTasks Source: [12] 48 App Creation Checklist
• If you own an Android device: – Ensure drivers are installed – Enable developer options on device under Settings, specifically USB Debugging
• Android 4.2+: Go to Settings→About phone, press Build number 7 times to enable developer options
• For Android Studio: – Under File→Settings→Appearance, enable “Show tool window bars”, “Widescreen tool window layout” – Programs should log states via android.util.Log’s Log.d(APP_TAG_STR, “debug”), where APP_TAG_STR is a final String tag denoting your app – Other commands: Log.e() (error); Log.i() (info); Log.w() (warning); Log.v() (verbose) – same parameters 49 Creating Android App
• Creating Android app project (Android Studio): – Go to File→New Project – Select what kind of

Activity to create (we'll use Empty activity) – Choose package name using "reverse DNS" style (e.g., edu.osu.myapp) – Choose APIs for app – Click Finish to create "Hello World" app 50 Deploying the App

- Two choices for deployment: – Real Android device – Android virtual device
- Plug in your real device; otherwise, create an Android virtual device
- Emulator is slow. Try Intel accelerated version, or perhaps http://www.genymotion.com/
- Run the app: press "Run" button in toolbar

❖ Introduction & Core Building Block.

An android **component** is simply a piece of code that has a well defined life cycle e.g. Activity, Receiver, Service etc.

The **core building blocks** or **fundamental components** of android are activities, views, intents, services, content providers, fragments and AndroidManifest.xml.

ACTIVITY :-

An activity is a class that represents a single screen. It is like a Frame in AWT.

VIEW :-

A view is the UI element such as button, label, text field etc. Anything that you see is a view.

INTENT :-

Intent is used to invoke components. It is mainly used to:

- Start the service
- Launch an activity
- Display a web page
- Display a list of contacts
- Broadcast a message
- Dial a phone call etc.

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);

SERVICE :-

Service is a background process that can run for a long time.

There are two types of services local and remote. Local service is accessed from within the application whereas remote service is accessed remotely from other applications running on the same device.

CONTENT PROVIDER :-

Content Providers are used to share data between the applications.

FRAGMENT :-

Fragments are like parts of activity. An activity can display one or more fragments on the screen at the same time.

ANDROIDMANIFEST.XML :-

It contains informations about activities, content providers, permissions etc. It is like the web.xml file in Java EE.

ANDROID VIRTUAL DEVICE (AVD) :-

It is used to test the android application without the need for mobile or tablet etc. It can be created in different configurations to emulate different types of real devices.

❖ UI Widget

The views or the controls that we want to display in an application are arranged in an order or sequence by placing them in the desired layout. The layouts also known as Containers or ViewGroups. These are used for organizing the

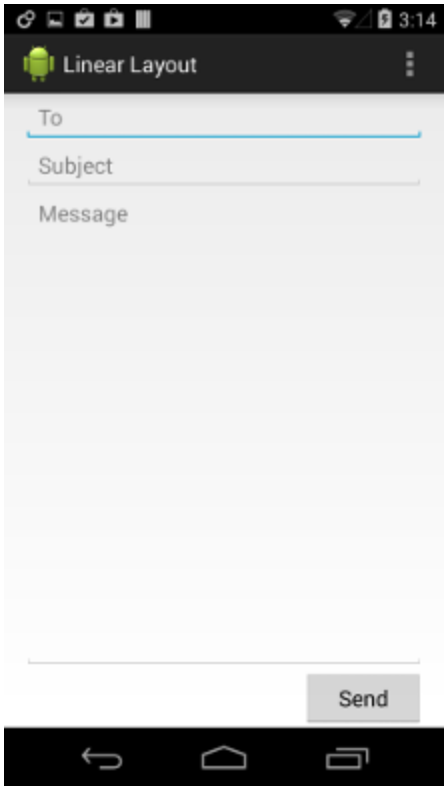
views or controls in the required format. Android provides the following layouts

• **LinearLayout**:

In this layout, all elements are arranged in a descending column from top to bottom or left to right. Each element contains properties to specify how much of the screen space it will consume. Depending on the orientation parameter, elements are either arranged in row or column format.

```
<LinearLayout
xmlns:android="http://schemas.android
.com/apk/res/android"
    android:layout_width="match_paren
t"
    android:layout_height="match_pare
nt"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_p
arent"
        android:layout_height="wrap_c
```

```
ontent"
        android:hint="@string/to" />
    <EditText
        android:layout_width="match_p
arent"
        android:layout_height="wrap_c
ontent"
        android:hint="@string/subject
" />
    <EditText
        android:layout_width="match_p
arent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message
" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_c
ontent"
        android:layout_gravity="right
"
        android:text="@string/send"
/>
</LinearLayout>
```



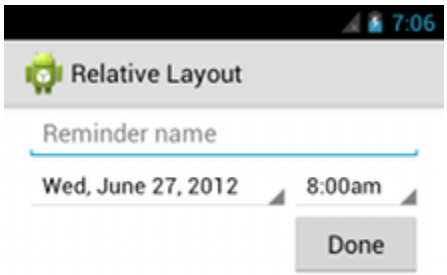
- **RelativeLayout**:

In this layout, each child element is laid out in relation to other child elements. That is, a child element appears in relation to the previous child. Also, the elements are laid out in relation to the parent.

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/
android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
```

```
android:paddingLeft="16dp"
android:paddingRight="16dp" >
<EditText
   android:id="@+id/name"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:hint="@string/reminder" />
<Spinner
   android:id="@+id/dates"
   android:layout_width="0dp"
   android:layout_height="wrap_content"
   android:layout_below="@id/name"
   android:layout_alignParentLeft="true"
   android:layout_toLeftOf="@+id/times" />
<Spinner
   android:id="@id/times"
   android:layout_width="96dp"
   android:layout_height="wrap_content"
   android:layout_below="@id/name"
   android:layout_alignParentRight="true" />
<Button
   android:layout_width="96dp"
   android:layout_height="wrap_content"
```

```
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```



- **FrameLayout**:

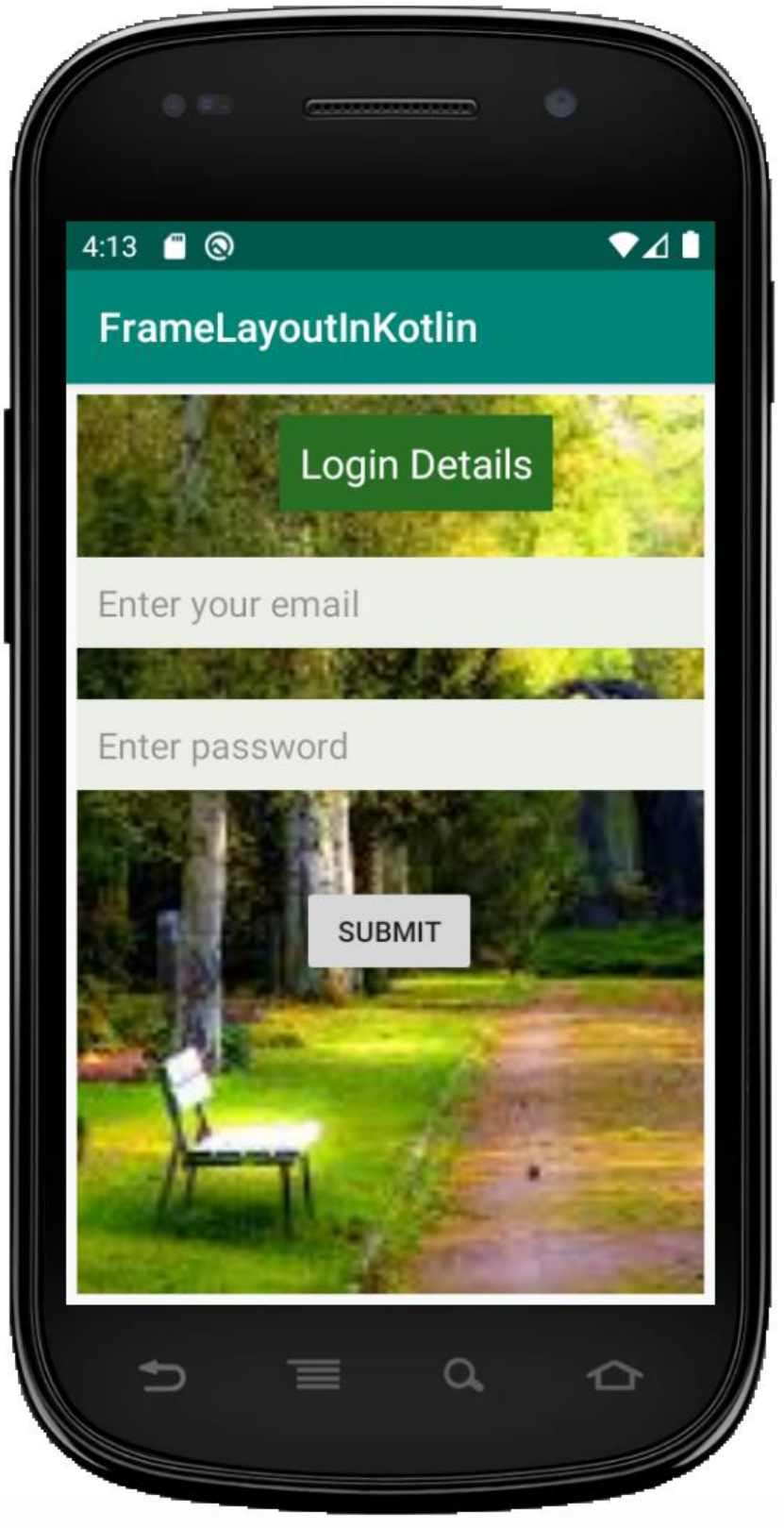
This is a layout used to display a single view. Views added to this are always placed at the top left of the layout. Any other view that is added to the FrameLayout overlaps the previous view; that is, each view stacks on top of the previous one.

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/
android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:padding="5dp">

   <ImageView
      android:id="@+id/imgvw1"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:scaleType="centerCrop"
      android:src="@drawable/img" />

   <TextView
      android:id="@+id/txtvw1"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:background="#286F24"
    android:padding="10dp"
    android:text="Login Details"
    android:textColor="#FFFFFF"
    android:textSize="20sp"
    android:layout_marginLeft="100dp"/>

<EditText
    android:id="@+id/editText1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="80dp"
    android:background="#ECEEE8"
```

```
    android:padding="10dp"
    android:hint="Enter your email" />

<EditText
    android:id="@+id/editText2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="150dp"
    android:background="#ECEEE8"
    android:padding="10dp"
    android:hint="Enter password"/>

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text= "Submit"
```

```
        android:layout_marginTop="240dp"
        android:layout_marginLeft="110dp"/>


</FrameLayout>
```


4:13
FrameLayoutInKotlin
Login Details
Enter your email
Enter password
SUBMIT

- **TableLayout**:

In this layout, the screen is assumed to be divided in table rows, and each of the child elements is arranged in a specific row and column.

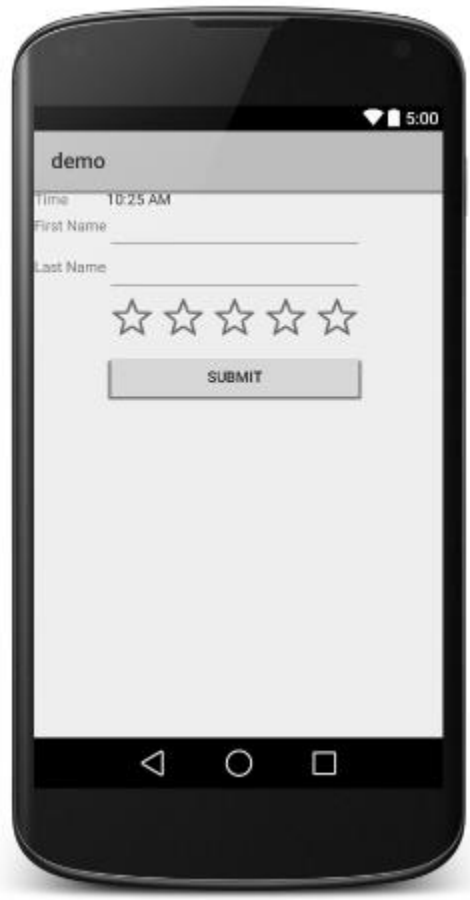
```
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/
android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">

  <TableRow
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <TextView
      android:text="Time"
      android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:layout_column="1" />

  <TextClock
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/textClock"
    android:layout_column="2" />

</TableRow>

<TableRow>

  <TextView
    android:text="First Name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:layout_column="1" />

  <EditText
    android:width="200px"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
</TableRow>

<TableRow>

  <TextView
    android:text="Last Name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="1" />

  <EditText
```

```
    android:width="100px"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
</TableRow>

<TableRow
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">

  <RatingBar
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/ratingBar"
    android:layout_column="2" />
</TableRow>

<TableRow
```

```
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"/>

  <TableRow
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Submit"
      android:id="@+id/button"
      android:layout_column="2" />
  </TableRow>

</TableLayout>
```



- **GridLayout**:

In this layout, child views are arranged in a grid format, that is, in the rows and columns pattern. The views can be placed at the specified row and column location. Also, more than one view can be placed at the given row and column position. The following list highlights some of the controls commonly used in Android applications:

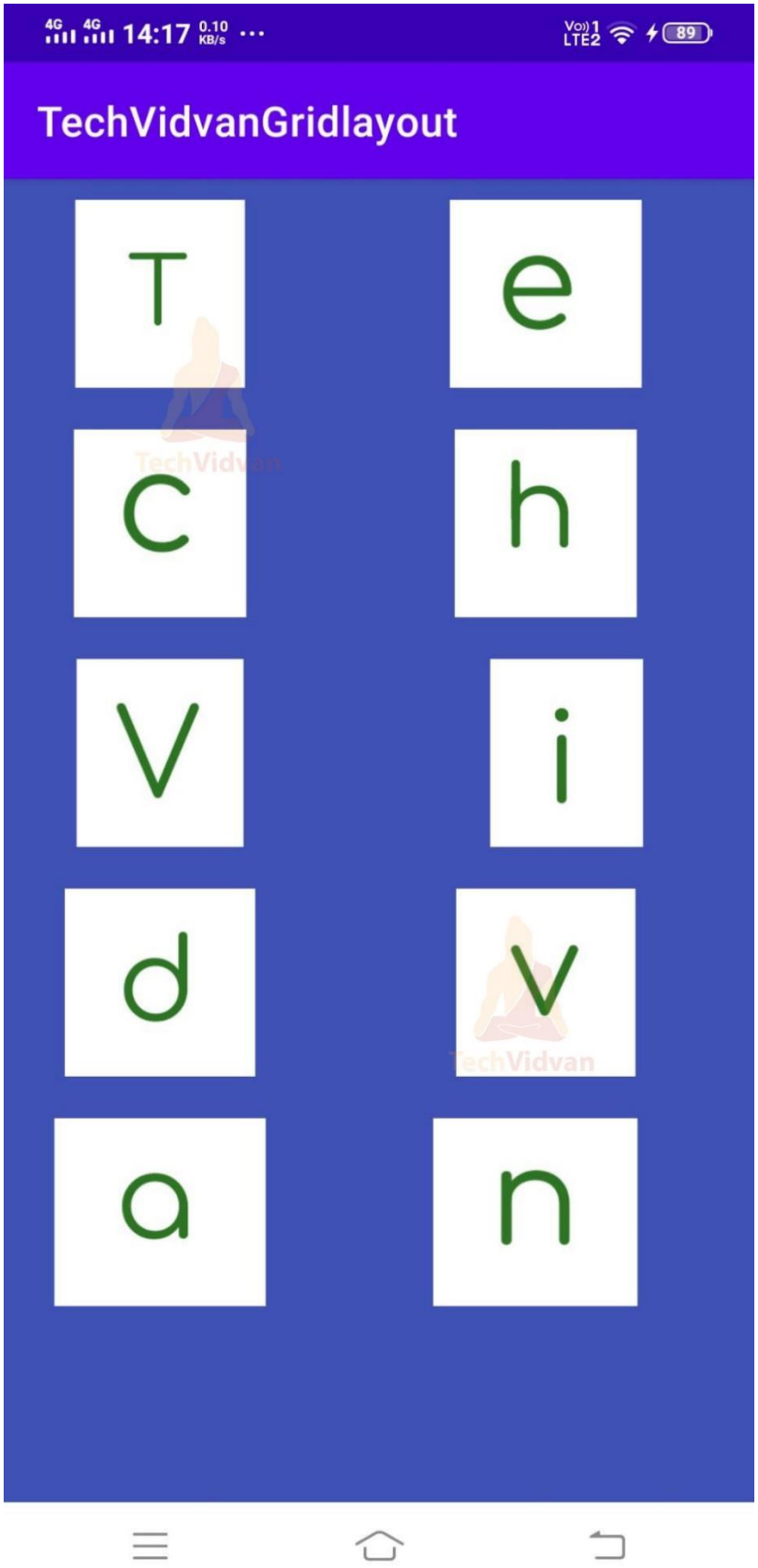
```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
xmlns:android="http://schemas.android.com/apk/res/androi
d"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:rowCount="4"
android:columnCount="2"
android:background="#3F51B5"
tools:context=".MainActivity">
<ImageView
android:layout_height="90dp"
android:layout_width="130dp"
android:layout_margin="10dp"
android:src="@drawable/image_1"
/>
<ImageView
```

```
android:layout_height="90dp"
android:layout_width="200dp"
android:layout_margin="10dp"
android:src="@drawable/image_2"
/>
<ImageView
android:layout_height="90dp"
android:layout_width="130dp"
android:layout_margin="10dp"
android:src="@drawable/image_3"
/>
<ImageView
android:layout_height="90dp"
android:layout_width="200dp"
android:layout_margin="10dp"
android:src="@drawable/image_4"
/>
<ImageView
```

```
android:layout_height="90dp"
android:layout_width="130dp"
android:layout_margin="10dp"
android:src="@drawable/image_5"
/>
<ImageView
android:layout_height="90dp"
android:layout_width="220dp"
android:layout_margin="10dp"
android:src="@drawable/image_6"
/>
<ImageView
android:layout_height="90dp"
android:layout_width="130dp"
android:layout_margin="10dp"
android:src="@drawable/image_7"
/>
<ImageView
```

```
android:layout_height="90dp"
android:layout_width="200dp"
android:layout_margin="10dp"
android:src="@drawable/image_8"
/>
<ImageView
android:layout_height="90dp"
android:layout_width="130dp"
android:layout_margin="10dp"
android:src="@drawable/image_9"
/>
<ImageView
android:layout_height="90dp"
android:layout_width="130dp"
android:layout_margin="10dp"
android:layout_marginLeft="40dp"
android:src="@drawable/image_10"
/>
```

```
</GridLayout>
```


14:17
TechVidvanGridlayout
T
e
C
h
V
i
d
v
a
n

- **TextView**:

A read-only text label. It supports multiline display, string formatting, and automatic word wrapping.

```
<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_m
argin"

android:paddingLeft="@dimen/activity_horizontal_ma
rgin"

android:paddingRight="@dimen/activity_horizontal_m
argin"
```

```
android:paddingTop="@dimen/activity_vertical_margi
n"
  tools:context=".MainActivity" >

  <TextView
    android:id="@+id/text_id"
    android:layout_width="300dp"
    android:layout_height="200dp"
    android:capitalize="characters"
    android:text="hello_world"

android:textColor="@android:color/holo_blue_dark"

android:textColorHighlight="@android:color/primary_
text_dark"
    android:layout_centerVertical="true"
    android:layout_alignParentEnd="true"
```

```
    android:textSize="50dp"/>


</RelativeLayout>
```

- **EditText**:

An editable text box that also accepts multiline entry and word-wrapping.

```
<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:layout_alignParentLeft="true"
  android:layout_alignParentTop="true"
  android:layout_marginLeft="14dp"
  android:layout_marginTop="18dp"
  android:text="@string/example_edittext" />

<Button
  android:id="@+id/button"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_alignLeft="@+id/textView1"
  android:layout_below="@+id/textView1"
```

```
    android:layout_marginTop="130dp"
    android:text="@string/show_the_text" />

  <EditText
    android:id="@+id/edittext"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/button"
    android:layout_below="@+id/textView1"
    android:layout_marginTop="61dp"
    android:ems="10"
    android:text="@string/enter_text"
android:inputType="text" />

</RelativeLayout>
```

- **Button**:

A standard command button.

```
<Button
    android:id="@+id/button_id"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
```

```
android:text="@string/self_destruct" />


button.setOnClickListener(new
View.OnClickListener() {
        public void onClick(View v) {
          // Code here executes on main thread after
user presses button
        }
    });
```

- **CheckBox**:

A button allowing a user to select (check) or unselect (uncheck).

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout x
mlns:android="http://schemas.android.com/ap
k/res/android"
```

```
    xmlns:app="http://schemas.android.com/apk
/res-auto"
    xmlns:tools="http://schemas.android.com/to
ols"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="example.javatpoint.com.check
box.MainActivity">


    <CheckBox
        android:id="@+id/checkBox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="144dp"
        android:layout_marginTop="68dp"
```

```
        android:text="Pizza"
        app:layout_constraintStart_toStartOf="par
ent"
        app:layout_constraintTop_toTopOf="paren
t" />


    <CheckBox
        android:id="@+id/checkBox2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="144dp"
        android:layout_marginTop="28dp"
        android:text="Coffee"
        app:layout_constraintStart_toStartOf="par
ent"
        app:layout_constraintTop_toBottomOf="@
+id/checkBox" />
```

```
    <CheckBox
        android:id="@+id/checkBox3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="144dp"
        android:layout_marginTop="28dp"
        android:text="Burger"
        app:layout_constraintStart_toStartOf="par
ent"
        app:layout_constraintTop_toBottomOf="@
+id/checkBox2" />


    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
```

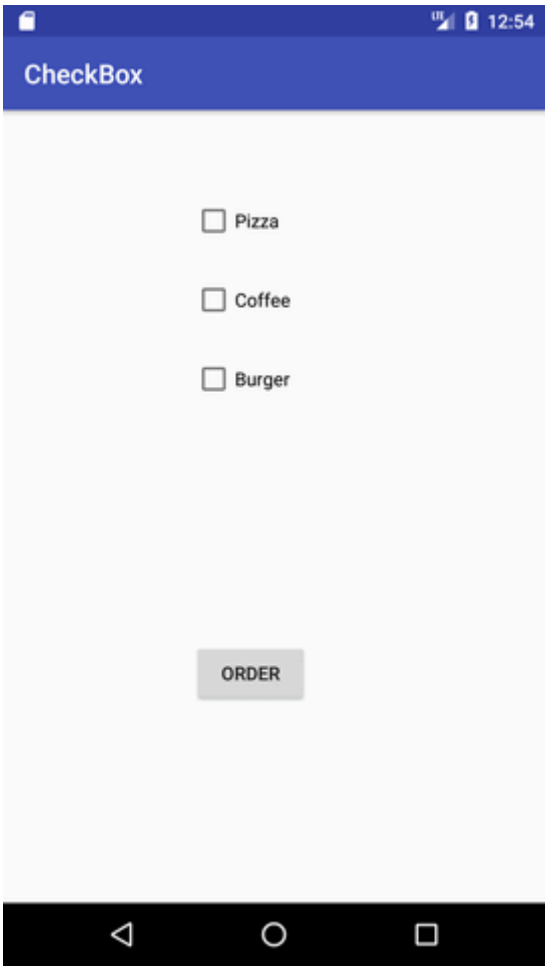
```
        android:layout_height="wrap_content"
        android:layout_marginLeft="144dp"
        android:layout_marginTop="184dp"
        android:text="Order"
        app:layout_constraintStart_toStartOf="par
ent"
        app:layout_constraintTop_toBottomOf="@
+id/checkBox3" />

</android.support.constraint.ConstraintLayout>

package example.javatpoint.com.checkbox;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
```

```
import android.widget.CheckBox;
import android.widget.Toast;


public class MainActivity extends AppCompatActivity
{
   CheckBox pizza,coffe,burger;
   Button buttonOrder;
   @Override
   protected void onCreate(Bundle savedInstanceSta
te) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      addListenerOnButtonClick();
   }
   public void addListenerOnButtonClick(){
      //Getting instance of CheckBoxes and Button fro
m the activty_main.xml file
      pizza=(CheckBox)findViewById(R.id.checkBox);
```

```
        coffe=(CheckBox)findViewById(R.id.checkBox2);


        burger=(CheckBox)findViewById(R.id.checkBox3
);
        buttonOrder=(Button)findViewById(R.id.button)
;


        //Applying the Listener on the Button click
        buttonOrder.setOnClickListener(new View.OnCli
ckListener(){


            @Override
            public void onClick(View view) {
                int totalamount=0;
                StringBuilder result=new StringBuilder();
                result.append("Selected Items:");
                if(pizza.isChecked()){
                    result.append("\nPizza 100Rs");
```

```
            totalamount+=100;
        }
        if(coffe.isChecked()){
            result.append("\nCoffe 50Rs");
            totalamount+=50;
        }
        if(burger.isChecked()){
            result.append("\nBurger 120Rs");
            totalamount+=120;
        }
        result.append("\nTotal: "+totalamount+"Rs
");
        //Displaying the message on the toast
        Toast.makeText(getApplicationContext(), re
sult.toString(), Toast.LENGTH_LONG).show();
    }
```

```
        });
    }
}
```


12:54
CheckBox
Pizza
Coffee
Burger
ORDER

- **RadioButton**:

A mutually exclusive button, which, when selected, unselects all other buttons in the group.

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

   <TextView
      android:id="@+id/textView1"
      android:layout_width="fill_parent"
```

```
    android:layout_height="wrap_content"
    android:layout_marginTop="30dp"
    android:gravity="center_horizontal"
    android:textSize="22dp"
    android:text="Single Radio Buttons" />



<!--   Default RadioButtons  -->

<RadioButton
    android:id="@+id/radioButton1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:text="Radio Button 1"
    android:layout_marginTop="20dp"
```

```
    android:textSize="20dp" />
<RadioButton
    android:id="@+id/radioButton2"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Radio Button 2"
    android:layout_marginTop="10dp"

    android:textSize="20dp" />


<View
    android:layout_width="fill_parent"
    android:layout_height="1dp"
    android:layout_marginTop="20dp"
    android:background="#B8B894" />
```

```
<TextView
   android:id="@+id/textView2"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:layout_marginTop="30dp"
   android:gravity="center_horizontal"
   android:textSize="22dp"
   android:text="Radio button inside RadioGroup" />



<!--   Customized RadioButtons  -->


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

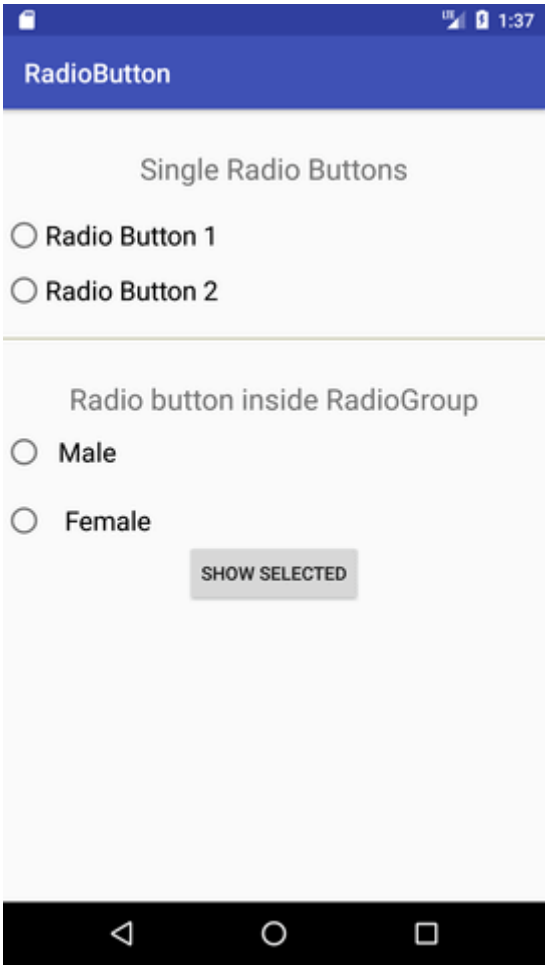
```
android:layout_height="wrap_content"
android:id="@+id/radioGroup">

<RadioButton
   android:id="@+id/radioMale"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="  Male"
   android:layout_marginTop="10dp"
   android:checked="false"
   android:textSize="20dp" />

<RadioButton
   android:id="@+id/radioFemale"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="   Female"
```

```
        android:layout_marginTop="20dp"
        android:checked="false"

        android:textSize="20dp" />
    </RadioGroup>

    <Button
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="Show Selected"
       android:id="@+id/button"
       android:onClick="onclickbuttonMethod"
       android:layout_gravity="center_horizontal" />


</LinearLayout>
```

```
package example.javatpoint.com.radiobutton;


import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;


public class MainActivity extends AppCompatActivity
{
    Button button;
    RadioButton genderradioButton;
    RadioGroup radioGroup;
    @Override
```

```
    protected void onCreate(Bundle savedInstanceSta
te) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        radioGroup=(RadioGroup)findViewById(R.id.radi
oGroup);
    }
    public void onclickbuttonMethod(View v){
        int selectedId = radioGroup.getCheckedRadioBu
ttonId();
        genderradioButton = (RadioButton) findViewByI
d(selectedId);
        if(selectedId==-1){
            Toast.makeText(MainActivity.this,"Nothing sel
ected", Toast.LENGTH_SHORT).show();
        }
        else{
```

```
            Toast.makeText(MainActivity.this,genderradio
Button.getText(), Toast.LENGTH_SHORT).show();
        }


    }
}
```



❖ Spinner and Dialog

Spinner :-

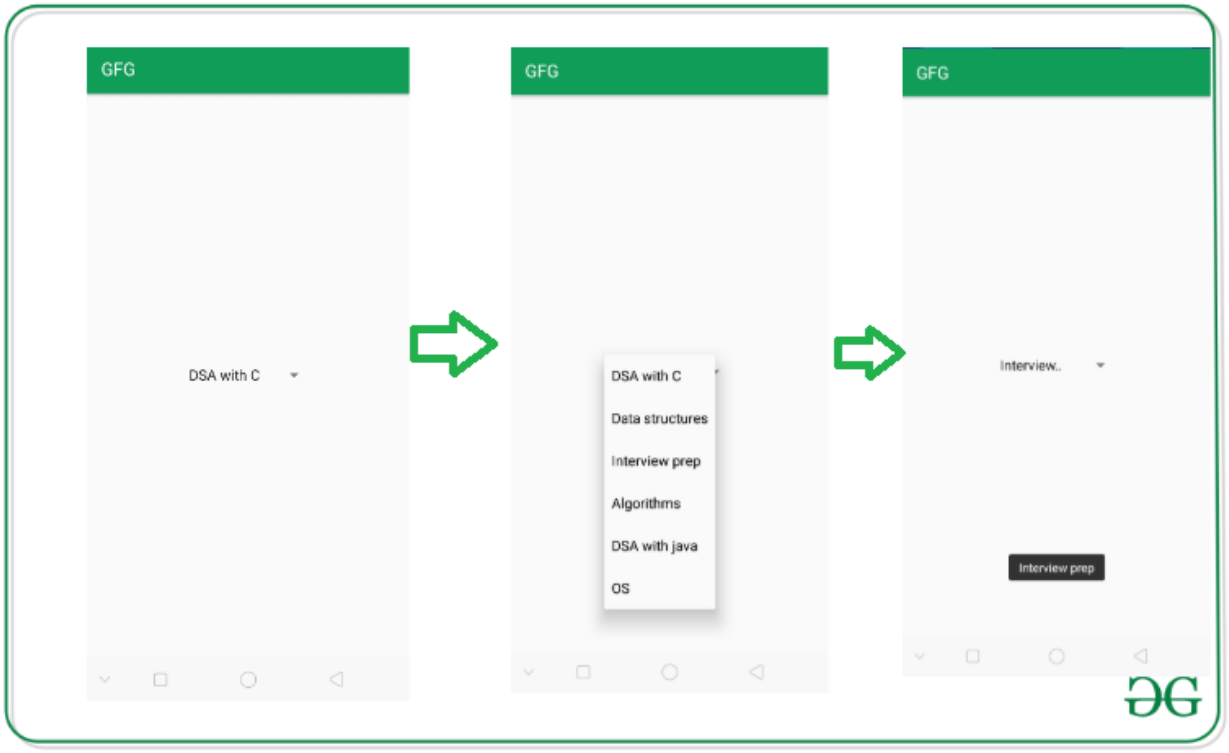
Android Spinner is a view similar to the dropdown list which is used to select one option from the list of options. It provides an easy way to select one item from the list of items and it shows a dropdown list of all values when we click on it. The default value of the android spinner will be the currently selected value and by using Adapter we can easily bind the items to the spinner objects. Generally, we populate our Spinner control with a list of items by using an ArrayAdapter in our Kotlin file.

Different Attributes for Spinner Widget :

XML attributes	Description
android: id	Used to specify the id of the view.
Android: textAlignment	Used to the text alignment in the dropdown list.
android: background	Used to set the background of the view.
android: padding	Used to set the padding of the view.
android: visibility	Used to set the visibility of the view.
android: gravity	Used to specify the gravity of the view like center, top, bottom, etc.

```
<Spinner
        android:id="@+id/coursesspinner"
        android:layout_height="50dp"
        android:layout_width="160dp"
        android:layout_marginEnd="10dp"
        android:layout_marginStart="10dp"
        android:layout_marginBottom="10dp"
        android:layout_marginTop="10dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>

        String[] courses = { "C", "Data structures",
                             "Interview prep", "Algorithms",
                             "DSA with java", "OS" };
        ArrayAdapter ad
```

```
    = new ArrayAdapter(
        this,
        android.R.layout.simple_spinner_item,
        courses);

// set simple layout resource file
// for each item of spinner
ad.setDropDownViewResource(
    android.R.layout
        .simple_spinner_dropdown_item);

// Set the ArrayAdapter (ad) data on the
// Spinner which binds data to spinner
spino.setAdapter(ad);
```



Dialogs :-

Widgets make use of RemoteViews to display their user interface. RemoteViews can be executed by another process with the same permissions as the original application. This way the Widget runs with the permissions of its defining application. This is form the official Android Documentation : ” RemoteView is a class that describes a view hierarchy that can be displayed in another process. The hierarchy is inflated from a layout resource file,

and this class provides some basic operations for modifying the content of the inflated hierarchy."

The user interface of a Widget is defined by an `BroadcastReceiver`.

This `BroadcastReceiver` inflates the layout into the `RemoteViews` of the Widget. Then `RemoteViews` is delivered to Android, which updates the user interface at the home screen application.

Widgets have limited functionality and styles compared to Activities. So you may need to thing some workarounds to do more complex stuff, like we want to do. And so to launch a pop up dialogue from our widget here's what we do: When the widget is clicked we simply launch a new `Activity` with `android:theme="@android:style/Theme.Dialog"` property set in the configuration of the `Activity` in `AndroidManifest.xml`.

To sum up the basic steps, we are going to:

- Create an Android Widget.

- Create an Intent that when sent to the BroadcastReceiver it marks the launching of the new Activity.
- Register a ClickListener to the widget. Thus when the widget is pressed the above Intent will be sent to the BroadcastReceiver.
- When this Intent is received, a new Activity that will look like a pop up dialog box, will be lauched.

- **Types of Dialogs :-**

  - Alert Dialog
  - Custom Dialog
  - Rating Dialog

❖ Listview, Webview, and Gridview.

○ ListView :-

Android ListView is a view which contains the group of items and displays in a scrollable list. ListView is implemented by

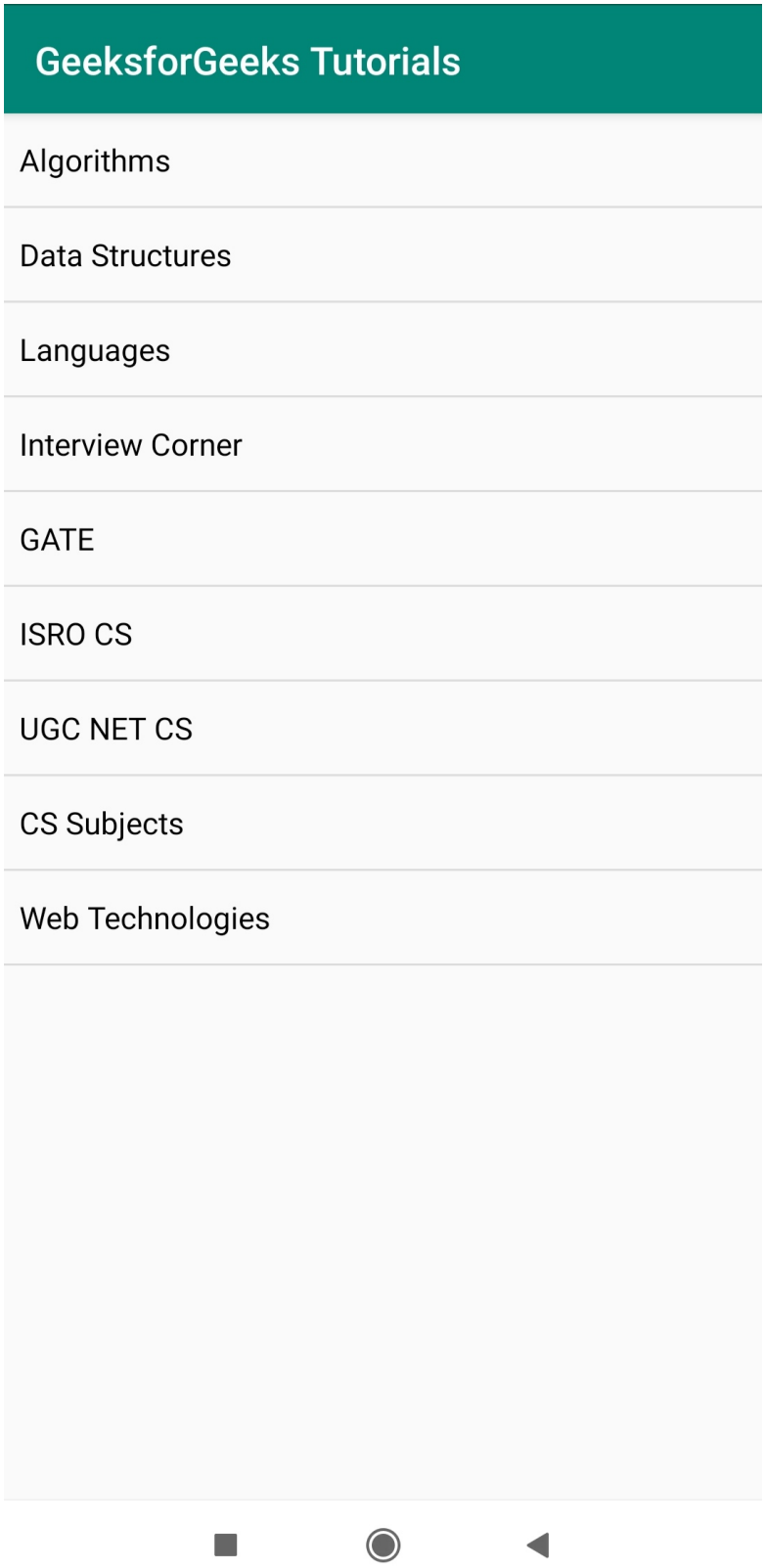
importing *android.widget.ListView* class.
ListView is a default scrollable which does not use other scroll view.

ListView uses Adapter classes which add the content from data source (such as string array, array, database etc) to ListView. Adapter bridges data between an *AdapterViews* and other Views (ListView, ScrollView etc).

```
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
```

```
    tools:context=".MainActivity">


    <ListView
        android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
</LinearLayout>


String tutorials[]
        = { "Algorithms", "Data Structures",
            "Languages", "Interview Corner",
            "GATE", "ISRO CS",
            "UGC NET CS", "CS Subjects",
            "Web Technologies" };


        ArrayAdapter<String> arr;
```

```
        arr
            = new ArrayAdapter<String>(
                this,
                R.layout.support_simple_spinner_dropdown
_item,
                tutorials);
        l.setAdapter(arr);
```


GeeksforGeeks Tutorials
Algorithms
Data Structures
Languages
Interview Corner
GATE
ISRO CS
UGC NET CS
CS Subjects
Web Technologies

- **WebView :-**

Android WebView is a view component which displays the web pages in the application. It uses a WebKit engine to show the web pages. The android.webkit.WebView class is the subclass of AbsoluteLayout class.

The loadUrl() and loadData() methods of WebView are used to load and display the web pages. To learn more about Android WebView go to https://www.google.com

There are different ways to load the web page in WebView such as:

Load the HTML content as a string in the class:

```
val wedData: String =  "<html><body><h1>
Hello, Javatpoint!</h1></body></html>"
val mimeType: String = "text/html"
val utfType: String = "UTF-8"
webView.loadData(wedData,mimeType,utf
Type)
```

Load the web page (.html, .jsp, etc.) from within the application. In such case, web pages are placed in assets directory.

```
webView.loadUrl("file:///android_asset/index.html")
```

Load the web URL inside WebView as:

```
webView.loadUrl("https://www.javatpoint.com/")
```

- **GridView** :-

GridView has existed in android sdk since the beginning of android. Through this widget you are able to show items in a grid and you have complete control over the number of columns shown. In this tutorial we will look at several android gridview customization examples.

A GridView is a type of AdapterView that displays items in a two-dimensional scrolling grid. Items are inserted into this grid layout from a database or from an array. The adapter is used for displaying this data, **setAdapter()** method is used to join the adapter with GridView. The main function of the adapter in GridView is to fetch data from a database or array and insert each piece of data in an appropriate item that will be displayed in GridView. This is how the GridView structure looks like. Note that we are going to implement this project using the **Java** language

```
<GridView
    android:id="@+id/idGVcourses"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:horizontalSpacing="6dp"
    android:numColumns="2"
    android:verticalSpacing="6dp" />
```

```
<?xml version="1.0" encoding="utf-8"?>
<!--XML implementation of Card Layout-->
<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="match_parent"
  android:layout_height="120dp"
  android:layout_gravity="center"
```

```
android:layout_margin="5dp"
app:cardCornerRadius="5dp"
app:cardElevation="5dp">

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

    <ImageView
        android:id="@+id/idIVcourse"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_gravity="center"
        android:src="@mipmap/ic_launcher" />

    <TextView
        android:id="@+id/idTVCourse"
```

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/app_name"
            android:textAlignment="center" />

    </LinearLayout>

</androidx.cardview.widget.CardView>


public class CourseModel {

    // string course_name for storing course_name
    // and imgid for storing image id.
    private String course_name;
    private int imgid;

    public CourseModel(String course_name, int imgid)
{
```

```
    this.course_name = course_name;
    this.imgid = imgid;
}

public String getCourse_name() {
    return course_name;
}

public void setCourse_name(String course_name) {
    this.course_name = course_name;
}

public int getImgid() {
    return imgid;
}

public void setImgid(int imgid) {
    this.imgid = imgid;
```

```
    }
}


import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import java.util.ArrayList;


public class CourseGVAdapter extends ArrayAdapter<CourseModel> {
    public CourseGVAdapter(@NonNull Context context, ArrayList<CourseModel> courseModelArrayList) {
```

```
    super(context, 0, courseModelArrayList);
  }


  @NonNull
  @Override
  public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
    View listitemView = convertView;
    if (listitemView == null) {
      // Layout Inflater inflates each item to be displayed in GridView.
      listitemView = LayoutInflater.from(getContext()).inflate(R.layout.card_item, parent, false);
    }
    CourseModel courseModel = getItem(position);
    TextView courseTV = listitemView.findViewById(R.id.idTVCourse);
    ImageView courseIV = listitemView.findViewById(R.id.idIVcourse);
```

```
        courseTV.setText(courseModel.getCourse_name(
));
        courseIV.setImageResource(courseModel.getImg
id());
        return listitemView;
    }
}


import android.os.Bundle;
import android.widget.GridView;
import androidx.appcompat.app.AppCompatActivity;
import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    GridView coursesGV;

    @Override
```

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


        coursesGV = findViewById(R.id.idGVcourses);


        ArrayList<CourseModel> courseModelArrayList = new ArrayList<CourseModel>();
        courseModelArrayList.add(new CourseModel("DSA", R.drawable.ic_gfglogo));
        courseModelArrayList.add(new CourseModel("JAVA", R.drawable.ic_gfglogo));
        courseModelArrayList.add(new CourseModel("C++", R.drawable.ic_gfglogo));
        courseModelArrayList.add(new CourseModel("Python", R.drawable.ic_gfglogo));
        courseModelArrayList.add(new CourseModel("Javascript", R.drawable.ic_gfglogo));
```

```
        courseModelArrayList.add(new
CourseModel("DSA", R.drawable.ic_gfglogo));


        CourseGVAdapter adapter = new
CourseGVAdapter(this, courseModelArrayList);
        coursesGV.setAdapter(adapter);
    }
}
```


10:23 PM
0.2KB/s
GFG GridView Example
GeeksforGeeks
DSA
GeeksforGeeks
JAVA
GeeksforGeeks
C++
GeeksforGeeks
Python
GeeksforGeeks
Javascript
GeeksforGeeks
DSA

❖ Intent(Implicit and Explicit Intent).

The intent is a messaging object which tells what kind of action to be performed. The intent's most significant use is the launching of the activity. The intent is a passive data structure holding an abstract description of an action to be performed.

There are two important things in intent

- **action:** The general action to be performed, such as ACTION_VIEW, ACTION_EDIT, ACTION_MAIN, etc.
- **data:** The data to operate on, such as a person record in the contacts database, is expressed as a Uri

1. Implicit Intent

Implicit Intent doesn't specify the component. In such a case, intent provides information on available components provided by the system that is to be invoked. For example, you may write the following code to view the webpage.

```
Intent intent=new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("https://www.geeksforgeeks.org"));
startActivity(intent)
;
```

2. Explicit Intent

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

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

❖ How to Create Multiple Activity with Passing Data.

This article aims to tell and show about how to "Send the data from one activity to second activity using Intent". In this Example, we have

two activities, **activity_first** which is the source activity and **activity_second** which is the destination activity. We can send the data using putExtra() method from one activity and get the data from the second activity using getStringExtra() methods.

In this Example, one EditText is used to input the text. This text is sent to the second activity when the "Send" button is clicked. For this, Intent will start and the following methods will run:

- **putExtra()** method is used for send the data, data in key value pair **key** is variable name and **value** can be Int, String, Float etc.

- **getStringExtra()** method is for getting the data(key) which is send by above method. according the data type of value there are other methods like **getIntExtra(), getFloatExtra()**

❖ How to Create Splash Screen and Toolbar.

- **Splash Screen :-**

SPLASH SCREEN IS THE FIRST SCREEN VISIBLE TO THE USER WHEN THE APPLICATION'S LAUNCHED. IT'S A VITAL SCREEN THAT WILL ALLOW THE USERS TO HAVE FIRST IMPRESSIONS OF THE APPLICATION.

A **Splash Screen** is a screen that appears when you open an app on your mobile device. Sometimes it's referred to as a launch screen or startup screen and shows up when your app is loading after you've just opened it.

When the loading is finished, you'll be taken to a more *functional screen* where you can complete actions.

Splash screens appear on your screen for a **fleeting moment** — look away and you might miss them. The

splash screen is one of the most vital screens in the application since it's the *user's first experience with the application*

- **Toolbar :-**

In Android applications, **Toolbar** is a kind of **ViewGroup** that can be placed in the **XML layouts** of an activity. It was introduced by the Google Android team during the release of **Android Lollipop(API 21)**. The Toolbar is basically the advanced successor of the **ActionBar**. It is much more flexible and customizable in terms of appearance and functionality. Unlike ActionBar, its position is not hardcoded i.e., not at the top of an activity. Developers can place it anywhere in the activity according to the need just like any other **View** in android. Toolbar use **material design theme** features of Android and thus it provides backward compatibility up to **API 7(Android 2.1)**. One can use the Toolbar in the following two ways:

1. **Use as an ActionBar:** In an app, the toolbar can be used as an ActionBar in order to provide more customization and a better appearance. All the features of ActionBar such as menu inflation, ActionBarDrawerToogle, etc. are also supported in Toolbar.

2. **Use a standalone Toolbar:** It can be used to implement a certain kind of design to an application that can not be fulfilled by an ActionBar. For example, showing a Toolbar at a position other than the top of the activity or showing multiple Toolbars in an activity.

Features supported by the Toolbar are much more focused and customizable than the ActionBar. Following are the components that can be added to make a user-appealing Toolbar:

- **Navigation button:** This element is aligned vertically with respect to the minimum height of the Toolbar. It is used as a guide for switching between other destinations within an app. The appearance of this element can be in many forms

such as navigation menu toggle, close, collapse, done, a simple up arrow, or any other kind of glyph required by the app.

- **Brand logo/App icon:** It is one of the most important aspects of an application as it provides a kind of identity. The height of the logo/icon is generally up to the height of the Toolbar but it can be extended as per the need.
- **Title and Subtitle:** The purpose of providing a title to the Toolbar is to give information regarding the current position within the navigation hierarchy of an app. The Subtitle is used to indicate any kind of extended information about the content.
- **ActionMenuView:** It is similar to the **Action Buttons** in ActionBar that display some important actions/elements of the app which may be required to the users frequently. Elements of this menu are placed to the end(rightmost side) of the Toolbar.
- **Multiple Custom Views:** Android allows developers to add one or more Views such as ImageView, TextView, etc. within the layout of the Toolbar. These views are treated as children of

the Toolbar layout and their position can be adjusted according to the need

❖ Advanced View (Cardview, Circleimageview, and Library integration).

We can easily use an Android ImageView to show Image in our App. But what if we want to show a circular image on our App screen, how will we do it? It is not possible to do it using a simple ImageView as it is not made for doing this type of stuffs, so to do so we have to use an external library.

One such external library is the **hdodenhof** module which provides the **hdodenhof CircleImageView** using which we can create a simple CircleImageView, but or for this tutorial we will use **CardView** to create a circular image view our our app in android studio.

```
<androidx.cardview.widget.CardView
    app:cardElevation = "16dp"
    app:cardCornerRadius = "160dp"
    android:layout_margin = "16dp"
    android:foregroundGravity = "center"
    android:layout_centerInParent = "true"
    android:layout_width = "300dp"
    android:layout_height = "300dp">
<ImageView
    android:scaleType = "centerCrop"
    android:src = "@drawable/img"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"/>


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

Implementation:-

'de.hdodenhof:circleimageview:3.1.0'

for circular image dependencies

❖ Image Loader Library(Glide, Piccaso, and Universal).

- Glide

Glide, like Picasso, can load and display images from many sources, while also taking care of caching and keeping a low memory impact when doing image manipulations. Official Google apps (like the app for Google I/O 2015) are using Glide. In this article, we're going to explore the functionalities of Glide and why it's superior in certain aspects.

Glide is an Image Loader Library for Android developed by bumptech and is a library that is recommended by Google. It has been used in many

Google open source projects including Google I/O 2014 official application. It provides animated GIF support and handles image loading/caching. Animated GIF support is currently not implemented in Picasso. Yes, images play a major role in making the UI of an App more interactive and user-friendly too. So, as an Android Developer, we should take care of using images in App. We should handle the different aspects of an image like the slow unresponsive image, memory issues, loading errors, and many more. If you are not handling these aspects in your application, then your app will make a bad impression on your users.

```
dependencies {
                                    implementation
'com.github.bumptech.glide:glide:4.11.0'
                                    annotationProcessor
'com.github.bumptech.glide:compiler:4.11.0'
}
```

```
<uses-permission
android:name=”android.permission.INTERNET”/>
```

```
Glide.with(context)
        .load(“YOUR IMAGE URL HERE”)
        .into(imageView)
```

Piccaso:-

Picasso is open source and one of the widely used image download libraries in Android. It is created and maintained by Square. It is among the powerful image download and caching library for Android. Picasso simplifies the process of loading images from external URLs and displays them on your application. For example, downloading an image from the server is one of the most common tasks in any application. And it needs quite a larger amount of code to achieve this via android networking API. By using Picasso, you can achieve this with a few lines of code.

```
implementation
'com.squareup.picasso:picasso:2.5.2'

<users-permission
android:name="android.permission.INTERNET"/>
```

```
Picasso.with(this)
        .load("https://media.geeksforgeeks.org/wp-content/cdn-uploads/logo-new-2.svg")
        .into(imageView);
```

Universal:-

UIL (Universal Image Loader) is a similar library to that of Picasso and Glide which performs loading images from any web URL into ImageView of Android. This image loading library has been created to provide a powerful, flexible, and customizable solution to load images in Android from Server. This image loading library is being created by an indie developer and it is present in the top list of GitHub. Features of UIL (universal Image Loader) library:

- This library provides Multi-thread image loading.
- Image caching can be done in memory and the user's device.
- Listening loading process (including downloading progress).
- Many customizable features are available for every display image call.

implementation 'com.nostra13.universalimageloader:universal-image-loader:1.9.5'

❖ Menu(Option Menu) & Shared Preference.

There are 3 types of menus in Android:

1. Option Menu
2. Context Menu
3. Pop-up Menu

OPTION MENU

The options menu is the primary collection of menu items for an activity. It's where you should place actions that have a overall impact on the app, such as Search, Compose Email and Settings.

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

```
 <menu
xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/i1"
        android:title="item"
        android:icon="@drawable/item" >

       <!-- "item" submenu -->
       <menu>
          <item android:id="@+id/a"
              android:title="subitem a"
              android:icon="@drawable/subitem_a"/>
          <item android:id="@+id/b"
              android:title="subitem b"
              android:icon="@drawable/subitem_b"
/>
```

```
        </menu>
    </item>
</menu>
```

- **android:id**

  A resource ID that's unique to the item, which allows the application to recognize the item when the user selects it.

- **android:icon**

  A reference to a drawable to use as the item's icon.

- **android:title**

  A reference to a string to use as the item's title.

```
@Override
 public boolean onCreateOptionsMenu(Menu menu) {
```

```
 MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu_file, menu);
    return true;
 }


@Override
public boolean onOptionsItemSelected(MenuItem item) {
   //Handle item selection
   switch (item.getItemId()) {
      case R.id.i1:
         //perform any action;
         return true;
      case R.id.a:
          //perform any action;
         return true;
```

```
        case R.id.b:
            //perform any action;
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

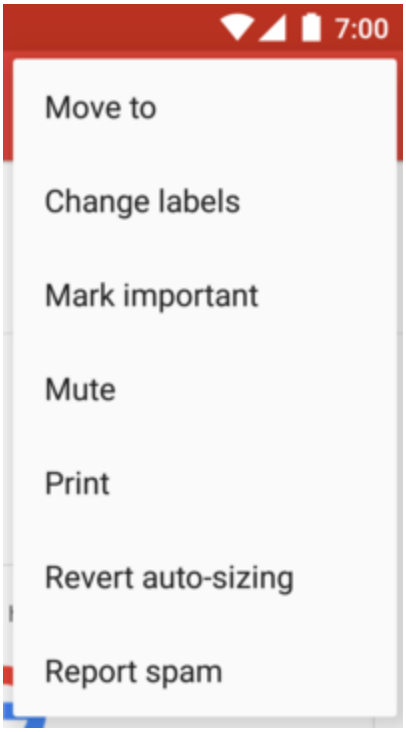
CONTEXT MENU

A context menu is a floating menu that appears when the user performs a long-click on an element. It provides actions that affect the selected content or context frame.

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                                ContextMenuInfo menuInfo) {
  super.onCreateContextMenu(menu, v, menuInfo);
  MenuInflater inflater = getMenuInflater();
  inflater.inflate(R.menu.menu_file, menu);
}

@Override
 public boolean onContextItemSelected(MenuItem item) {
```

```
    switch (item.getItemId()) {
        case R.id.i1:
            //Perform any action;
            return true;
        case R.id.a:
            //Perform any action;
            return true;
        case R.id.b:
            //Perform any action;
            return true;
        default:
            return super.onContextItemSelected(item);
    }
}
```

POPUP MENU



A popup menu displays a list of items in a vertical list that is anchored(sticked) to the view that invoked the menu. It's good for providing an overflow of actions that relate to specific content or to provide options for a second part of a command.

```
public void pop(View v){

        PopupMenu popup = new PopupMenu(this,v);

        MenuInflater inflater = getMenuInflater();


inflater.inflate(R.menu.menu_file,popup.getMenu()
);
```

```
        popup.show();
    }


@Override
 public boolean onMenuItemClick(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.i1:
            //Perform any action;
            return true;
        case R.id.a:
            //Perform any action;
            return true;
        case R.id.b:
            //Perform any action;
            return true;
        default:
```

```
        return false;
    }
}
```

❖ Image Access (Camera & Gallery).

- **Camera**

This image shows the Image clicked by the camera and set in Imageview. When the app is opened, it displays the “Camera” Button to open the camera. When pressed, ACTION_IMAGE_CAPTURE Intent gets started by the MediaStore class. When the image is captured, it is displayed in the imageview.

```
Intent camera_intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(camera_intent, pic_id);
```

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) { }

Bitmap photo = (Bitmap) data.getExtras().get("data");
clicked_image_id.setImageBitmap(photo);
```

- **Gallery**

Selecting an image from a gallery in Android is required when the user has to upload or set their image as a profile picture or the user wants to send a pic to the other. So in this article, it's been discussed step by step how to select an image from the gallery and preview the selected image. Have a look at the following image what's been discussed further in this article.

```
Intent i = new Intent();
    i.setType("image/*");
    i.setAction(Intent.ACTION_GET_CONTENT);
```

```
        // pass the constant to compare it
          // with the returned requestCode
        startActivityForResult(Intent.createChooser(i,
"Select Picture"), SELECT_PICTURE);

public void onActivityResult(int requestCode, int
resultCode, Intent data) {
        super.onActivityResult(requestCode,
resultCode, data);

        if (resultCode == RESULT_OK) {

            // compare the resultCode with the
             // SELECT_PICTURE constant
            if (requestCode == SELECT_PICTURE) {
                // Get the url of the image from data
                Uri selectedImageUri = data.getData();
                if (null != selectedImageUri) {
                    // update the preview image in the
layout
                    IVPreviewImage.setImageURI(selectedI
mageUri);
                }
```

```
        }
    }
}
```

❖ recycler view with Model Class.

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.

exam_card.xml

```
   <!-- 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>
```

ExamViewHolder.java

```
// ViewHolder code for RecyclerView
package com.example.admin.example;

import android.support.v7.widget.RecyclerView;
import android.view.View;
import android.widget.TextView;
```

```
public class examViewHolder
   extends RecyclerView.ViewHolder {
   TextView examName;
   TextView examMessage;
   TextView examDate;
   View view;

   examViewHolder(View itemView)
   {
         super(itemView);
         examName
               = (TextView)itemView
                     .findViewById(R.id.examName);
         examDate
               = (TextView)itemView
                     .findViewById(R.id.examDate);
         examMessage
               = (TextView)itemView
                     .findViewById(R.id.examMessage);
         view = itemView
   }
}
```

ExamData.java

```
public class examData {
   String name;
   String date;
   String message;

   examData(String name,
         String date,
         String message)
   {
      this.name = name;
      this.date = date;
      this.message = message;
   }
}
```

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(recyclerVie
w);
    }


}
```

Activity_exam.java

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

    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
android:layout_marginTop="56dp"
android:background="#FFFFFF"
tools:context=".exam" >

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

    <android.support.v7.widget.RecyclerView
        android:nestedScrollingEnabled="false"
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:overScrollMode="never"
        android:layout_height="wrap_content"/>
```

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

ExamActivity.java

```
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;
    }
}
```