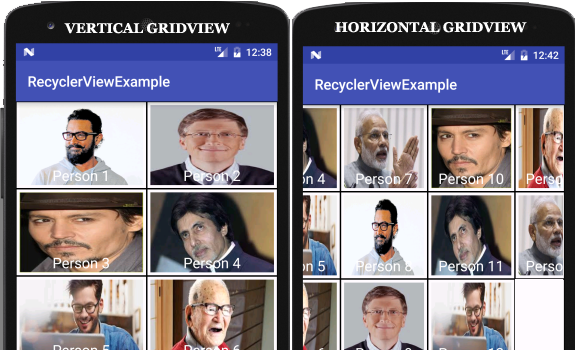
RecyclerView Using GridLayoutManager With Example In Android Studio

In Android, [RecyclerView](http://abhiandroid.com/materialdesign/recyclerview-as-listview.html) is an advance and flexible version of [ListView](http://abhiandroid.com/ui/listview) and [GridView](http://abhiandroid.com/ui/gridview). It is a container used to display large amount of data sets that can be scrolled very efficiently by maintaining a limited number of views. [RecyclerView](http://abhiandroid.com/materialdesign/recyclerview-as-listview.html) was introduced in Material Design in API level 21 (Android 5.0 i.e Lollipop).

Material Design brings lot of new features in Android that changed a lot the visual design patterns regarding the designing of modern Android applications. This new widget is a big step for displaying data because the [GridView](http://abhiandroid.com/ui/gridview) is one of the most commonly used UI widget. In RecyclerView android provides a lot of new features which are not present in existing [ListView](http://abhiandroid.com/ui/listview) or GridView.  
  
**Basic RecyclerView**[**XML**](http://abhiandroid.com/ui/xml)**code:**

<?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="abhiandroid.com.recyclerviewexample.MainActivity">

<android.support.v7.widget.RecyclerView

android:id="@+id/recyclerView"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" />

</RelativeLayout>

**Gradle Dependency to use RecyclerView:**

The RecyclerView widget is a part of separate library valid for API 7 level or higher. Add the following dependency in your Gradle build file to use recyclerview.

dependencies {

...

compile "com.android.support:recyclerview-v7:23.23.4.0"

}

**Table Of Contents**[[hide](http://abhiandroid.com/materialdesign/recyclerview-gridview.html)]

* [1 RecyclerView As GridView In Android:](http://abhiandroid.com/materialdesign/recyclerview-gridview.html#RecyclerView_As_GridView_In_Android)
* [2 Comparison between RecyclerView and GridView](http://abhiandroid.com/materialdesign/recyclerview-gridview.html#Comparison_between_RecyclerView_and_GridView)
* [3 Example Of RecyclerView As Vertical GridView In Android Studio:](http://abhiandroid.com/materialdesign/recyclerview-gridview.html#Example_Of_RecyclerView_As_Vertical_GridView_In_Android_Studio)
* [4 Example of RecyclerView As Horizontal GridView In Android Studio](http://abhiandroid.com/materialdesign/recyclerview-gridview.html#Example_of_RecyclerView_As_Horizontal_GridView_In_Android_Studio)

**RecyclerView As GridView In Android:**

In this article we will discuss how to use a RecyclerView As GridView. For that we need to understand LayoutManager component of RecyclerView. Layout Manager is a very new concept introduced in RecyclerView for defining the type of Layout which RecyclerView should use. It Contains the references for all the views that are filled by the data of the entry. We can create a Custom Layout Manager by extending RecyclerView.LayoutManager [Class](http://abhiandroid.com/java/class-objects) but RecyclerView Provides three types of in-built Layout Managers.

**1. LinearLayoutManager:**It is used for displaying Vertical or Horizontal List. To understand it please read [RecyclerView as Listview](http://abhiandroid.com/materialdesign/recyclerview-as-listview.html)  
**2. GridLayoutManager:** It is used for displaying the items in the form of Grids.  
**3. StaggeredGridLayoutManager:** It is used to show the items in staggered Grid.

In this article our main focus is on GridLayoutManager because it is used to display the data in the form of grids. By using this Layout Manager we can easily create grid items. A common example of grid items is our phone’s [gallery](http://abhiandroid.com/ui/gallery) in which all the images are displayed in the form of grids.

**GridLayoutManager** is used for displaying the data items in grid format and we can easily define the orientation for the items. In Simple words we can say that we use the GridLayoutManager for displaying RecyclerView as a GridView.

**Public constructors for GridLayoutManager:**Below we define the public [constructor](http://abhiandroid.com/java/constructor) for GridLayoutManager that should be used for defining orientation(vertical or horizontal) of RecyclerView.

**1- GridLayoutManager (Context context, int spanCount):** It is used to create a Vertical GridLayoutManager. In this [constructor](http://abhiandroid.com/java/constructor) first parameter is used to set the current context and second parameter is used to set the span Count means the number of columns in the grid.  
**Example:** In below code snippet we show how to use this constructor in Android.

**With Default Vertical Orientation:**

// get the reference of RecyclerView

RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recyclerView);

// set a GridLayoutManager with default vertical orientation and 3 number of columns

GridLayoutManager gridLayoutManager = new GridLayoutManager(getApplicationContext(),3);

recyclerView.setLayoutManager(gridLayoutManager); // set LayoutManager to RecyclerView

**With Horizontal Orientation:**

// get the reference of RecyclerView

RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recyclerView);

// set a GridLayoutManager with 3 number of columns

GridLayoutManager gridLayoutManager = new GridLayoutManager(getApplicationContext(),3);

gridLayoutManager.setOrientation(LinearLayoutManager.HORIZONTAL); // set Horizontal Orientation

recyclerView.setLayoutManager(gridLayoutManager); // set LayoutManager to RecyclerView

**2- GridLayoutManager (Context context, int spanCount, int orientation, boolean reverseLayout):** In this constructor first parameter is used to set the current context, second parameter is used to set the span Count means the number of columns in the grid, third parameter is used to set the Layout Orientation should be vertical or horizontal and last param is a boolean value when sets to true layout from end to start means grids are arranged from end to start.

**Example:** In below code snippet we show how to use this constructor in Android.

// get the reference of RecyclerView

RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recyclerView);

// set a GridLayoutManager with 3 number of columns , horizontal gravity and false value for reverseLayout to show the items from start to end

GridLayoutManager gridLayoutManager = new GridLayoutManager(getApplicationContext(),3,LinearLayoutManager.HORIZONTAL,false);

recyclerView.setLayoutManager(gridLayoutManager); // set LayoutManager to RecyclerView

**Comparison between RecyclerView and GridView**

There are a lot of new features in RecyclerView that are not present in existing GridView. The RecyclerView is more flexible, powerful and a major enhancement over GridView. I will try to give you a detailed insight into it. Below we discuss some important features of RecyclerView that should clear the reason why RecyclerView is better than GridView.

**1. Custom Item Layouts:** GridView can only layout the grid items in Vertical Arrangement in which we set the number of columns and rows are automatically creates according to number of items. GridView cannot be customized according to our requirements. Suppose we need to show items in horizontal arrangement in which we want to fix number of rows and columns are automatically creates according to number of items but that thing is not feasible with default GridView. But with introduction of Recyclerview we can easily create a horizontal or Vertical arrangement for grid items. By using GridLayoutManager component of RecyclerView we can easily define the orientation of grid items and spanCount used for number of columns if orientation is vertical or number of rows if orientation is horizontal.

**2. Use Of ViewHolder Pattern:** GridView Adapters do not require the use of ViewHolder but RecyclerView require the use of ViewHolder that is used to store the reference of View’s. In GridView it is recommended to use the ViewHolder but it is not compulsion but in RecyclerView it is mandatory to use ViewHolder that is the main difference between RecyclerView and GridView. ViewHolder is a static inner [class](http://abhiandroid.com/java/class-objects) in our [Adapter](http://abhiandroid.com/ui/adapter) which holds references to the relevant view’s. By using these references our code can avoid time consuming findViewById() [method](http://abhiandroid.com/java/method) to update the widgets with new data.

**3. GridView Adapters:** In GridView we use many [Adapter](http://abhiandroid.com/ui/adapter)‘s like [ArrayAdapter](http://abhiandroid.com/ui/arrayadapter-tutorial-example.html) for displaying simple [array](http://abhiandroid.com/java/arrays) data, Base and SimpleAdapters for custom grids with images and text. In RecyclerView we only use RecyclerView.Adapter to setting the grid items. In Below code snippet we show how our CustomAdapter looks when we extends RecyclerView.Adapter class and use ViewHolder in it.

package abhiandroid.com.recyclerviewexample;

import android.support.v7.widget.RecyclerView;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.TextView;

public class CustomAdapter extends RecyclerView.Adapter {

@Override

public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

// infalte the item Layout

View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.rowlayout, parent, false);

// set the view's size, margins, paddings and layout parameters

MyViewHolder vh = new MyViewHolder(v); // pass the view to View Holder

return vh;

}

@Override

public void onBindViewHolder(MyViewHolder holder, int position) {

}

@Override

public int getItemCount() {

return 0;

}

public class MyViewHolder extends RecyclerView.ViewHolder {

TextView textView;// init the item view's

public MyViewHolder(View itemView) {

super(itemView);

// get the reference of item view's

textView = (TextView) itemView.findViewById(R.id.textView);

}

}

}

**4. Item Animator:**GridView are lacking in support of Good Animation’s. RecyclerView brings a new dimensions in it. By using RecycleView.ItemAnimator class we can easily animate the view’s.



**5. Item Decoration:** In GridView’s Dynamically decorating items like Adding divider or border was never easy but in RecyclerView by using RecycleView.ItemDecorator class we have a huge control on it.

**Conclusion:** At the end we can say that RecyclerView is much more customizable than the existing GridView and gives a lot of control and power to its developers.

**Example Of RecyclerView As Vertical GridView In Android Studio:**

Below is the example of RecyclerView As GridView in which we display grids of Person Names with their images with default vertical orientation by using RecyclerView. In this example we are using GridLayoutManager with vertical orientation and 2 span count value to display the items. Firstly we declare a RecyclerView in our XML file and then get the reference of it in our Activity. After that we creates two [ArrayList](http://abhiandroid.com/java/arraylist)‘s for Person Names and Images. After that we set a GridLayoutManager and finally we set the Adapter to show the grid items in RecyclerView. Whenever a user clicks on an item the full size image will be displayed on the next screen.

[Download Code (Password: abhiandroid)](https://www.dropbox.com/s/cnxhjrr8ukbve5j/RecyclerViewExample.zip?dl=0)

  
**Step 1**: [Create a New Project](http://abhiandroid.com/androidstudio/start-create-project) And Name It RecyclerViewExample.

**Step 2:** Open Gradle Scripts > build.gradle and add RecyclerView Library dependency in it.

apply plugin: 'com.android.application'

android {

compileSdkVersion 24

buildToolsVersion "24.0.1"

defaultConfig {

applicationId "abhiandroid.com.recyclerviewexample"

minSdkVersion 16

targetSdkVersion 24

versionCode 1

versionName "1.0"

}

buildTypes {

release {

minifyEnabled false

proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'

}

}

}

dependencies {

compile fileTree(dir: 'libs', include: ['\*.jar'])

testCompile 'junit:junit:4.12'

compile 'com.android.support:appcompat-v7:24.1.1'

compile "com.android.support:recyclerview-v7:23.4.0" // dependency file for RecyclerView

}

**Step 3:** Open res -> layout -> activity\_main.xml (or) main.xml and add following code:  
In this step we create a RecyclerView in our XML file.

<?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="abhiandroid.com.recyclerviewexample.MainActivity">

<android.support.v7.widget.RecyclerView

android:id="@+id/recyclerView"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" />

</RelativeLayout>

**Step 4:** Create a new drawable XML file in Drawable folder and name it custom\_item\_layout.xml and add the following code in it for creating a custom grid item.  
In this step we create a drawable XML file in which we add the code for creating custom grid items.

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

<shape xmlns:android="http://schemas.android.com/apk/res/android">

<!--

stroke with color and width for creating outer line

-->

<stroke

android:width="1dp"

android:color="#000" />

</shape>

**Step 5:** Create a new XML file rowlayout.xml for grid item of RecyclerView and paste the following code in it.  
In this step we create a new xml file for item row in which we creates a [TextView](http://abhiandroid.com/ui/textview) and [ImageView](http://abhiandroid.com/ui/imageview) to show the data in grid format.

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

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="120dp"

android:background="@drawable/custom\_item\_layout"

android:padding="5dp">

<!--

grid items for RecyclerView

-->

<ImageView

android:id="@+id/image"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:scaleType="fitXY"

android:src="@mipmap/ic\_launcher" />

<TextView

android:id="@+id/name"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:layout\_centerHorizontal="true"

android:text="ABCD"

android:textSize="20sp"

android:textColor="#fff" />

</RelativeLayout>

**Step 6 :** Now open app -> java -> package -> MainActivity.java and add the below code.

In this step firstly we get the reference of RecyclerView. After that we creates two [ArrayList](http://abhiandroid.com/java/arraylist)‘s for Person Names and Images. After that we set a GridLayoutManager and finally we set the Adapter to show the grid items in RecyclerView.

package abhiandroid.com.recyclerviewexample;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.support.v7.widget.GridLayoutManager;

import android.support.v7.widget.RecyclerView;

import java.util.ArrayList;

import java.util.Arrays;

public class MainActivity extends AppCompatActivity {

// ArrayList for person names

ArrayList personNames = new ArrayList<>(Arrays.asList("Person 1", "Person 2", "Person 3", "Person 4", "Person 5", "Person 6", "Person 7","Person 8", "Person 9", "Person 10", "Person 11", "Person 12", "Person 13", "Person 14"));

ArrayList personImages = new ArrayList<>(Arrays.asList(R.drawable.person1, R.drawable.person2, R.drawable.person3, R.drawable.person4, R.drawable.person5, R.drawable.person6, R.drawable.person7,R.drawable.person1, R.drawable.person2, R.drawable.person3, R.drawable.person4, R.drawable.person5, R.drawable.person6, R.drawable.person7));

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// get the reference of RecyclerView

RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recyclerView);

// set a GridLayoutManager with default vertical orientation and 2 number of columns

GridLayoutManager gridLayoutManager = new GridLayoutManager(getApplicationContext(),2);

recyclerView.setLayoutManager(gridLayoutManager); // set LayoutManager to RecyclerView

// call the constructor of CustomAdapter to send the reference and data to Adapter

CustomAdapter customAdapter = new CustomAdapter(MainActivity.this, personNames,personImages);

recyclerView.setAdapter(customAdapter); // set the Adapter to RecyclerView

}

}

**Step 7:** Create a new class CustomAdapter.java inside package and add the following code.  
In this step we create a CustomAdapter class and extends RecyclerView.Adapter class with Viewholder in it. After that we implement the overrided [methods](http://abhiandroid.com/java/method) and create a constructor for getting the data from Activity. In this custom Adapter two [methods](http://abhiandroid.com/java/method) are more important first is onCreateViewHolder in which we inflate the layout item xml and pass it to View Holder and other is onBindViewHolder in which we set the data in the view’s with the help of ViewHolder. Finally we implement the setOnClickListener event on itemview and on click of item we display the selected image in full size in the next Activity.

package abhiandroid.com.recyclerviewexample;

import android.content.Context;

import android.content.Intent;

import android.support.v7.widget.RecyclerView;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageView;

import android.widget.TextView;

import java.util.ArrayList;

public class CustomAdapter extends RecyclerView.Adapter {

ArrayList personNames;

ArrayList personImages;

Context context;

public CustomAdapter(Context context, ArrayList personNames, ArrayList personImages) {

this.context = context;

this.personNames = personNames;

this.personImages = personImages;

}

@Override

public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

// infalte the item Layout

View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.rowlayout, parent, false);

// set the view's size, margins, paddings and layout parameters

MyViewHolder vh = new MyViewHolder(v); // pass the view to View Holder

return vh;

}

@Override

public void onBindViewHolder(MyViewHolder holder, final int position) {

// set the data in items

holder.name.setText(personNames.get(position));

holder.image.setImageResource(personImages.get(position));

// implement setOnClickListener event on item view.

holder.itemView.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

// open another activity on item click

Intent intent = new Intent(context, SecondActivity.class);

intent.putExtra("image", personImages.get(position)); // put image data in Intent

context.startActivity(intent); // start Intent

}

});

}

@Override

public int getItemCount() {

return personNames.size();

}

public class MyViewHolder extends RecyclerView.ViewHolder {

// init the item view's

TextView name;

ImageView image;

public MyViewHolder(View itemView) {

super(itemView);

// get the reference of item view's

name = (TextView) itemView.findViewById(R.id.name);

image = (ImageView) itemView.findViewById(R.id.image);

}

}

}

**Step 8:** Create a new XML file activity\_second.xml and add below code in it.  
In this step we create a [ImageView](http://abhiandroid.com/ui/imageview) in our XML file to show the selected image in full size.

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

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

android:background="#fff">

<ImageView

android:id="@+id/selectedImage"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_centerInParent="true"

android:scaleType="fitXY" />

</RelativeLayout>

**Step 9:** Create a new Activity and name it SecondActivity.class and add the below code in it.  
In this step we get the reference of ImageView and then get Intent which was set from adapter of Previous Activity and then finally we set the image in ImageView.

package abhiandroid.com.recyclerviewexample;

import android.content.Intent;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.widget.ImageView;

public class SecondActivity extends AppCompatActivity {

ImageView selectedImage;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_second);

selectedImage = (ImageView) findViewById(R.id.selectedImage); // init a ImageView

Intent intent = getIntent(); // get Intent which was set from adapter of Previous Activity

selectedImage.setImageResource(intent.getIntExtra("image", 0)); // get image from Intent and set it in ImageView

}

}

**Output:**

Now [run the App](http://abhiandroid.com/androidstudio/run-app-avd-emulator-android-studio) and you will see person name which can be scrolled in vertical direction created using RecyclerView as Gridview.

**Example of RecyclerView As Horizontal GridView In Android Studio**

Below is the example of RecyclerView As GridView in which we display the grids of Person Names with their images with horizontal orientation by using RecyclerView. In this example we are using GridLayoutManager with horizontal orientation and 3 span count value which defines the number of rows. Firstly we declare a RecyclerView in our XML file and then get the reference of it in our Activity. After that we creates two ArrayList’s for Person Names and Images. After that we set a GridLayoutManager and finally we set the Adapter to show the grid items in RecyclerView. Whenever a user clicks on an item the full size image will be displayed on the next screen.

[Download Code (Password: abhiandroid)](https://www.dropbox.com/s/w5mk3dqs6fhi4p0/RecyclerViewExample.zip?dl=0)

  
**Step 1:** [Create a New Project](http://abhiandroid.com/androidstudio/start-create-project) and name it RecyclerViewExample.

**Step 2:** Open Gradle Scripts > build.gradle and add RecyclerView Library dependency in it.

apply plugin: 'com.android.application'

android {

compileSdkVersion 24

buildToolsVersion "24.0.1"

defaultConfig {

applicationId "abhiandroid.com.recyclerviewexample"

minSdkVersion 16

targetSdkVersion 24

versionCode 1

versionName "1.0"

}

buildTypes {

release {

minifyEnabled false

proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'

}

}

}

dependencies {

compile fileTree(dir: 'libs', include: ['\*.jar'])

testCompile 'junit:junit:4.12'

compile 'com.android.support:appcompat-v7:24.1.1'

compile "com.android.support:recyclerview-v7:23.4.0" // dependency file for RecyclerView

}

**Step 3:** Open res -> layout -> activity\_main.xml (or) main.xml and add following code:  
In this step we create a RecyclerView in our XML file.

<?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="abhiandroid.com.recyclerviewexample.MainActivity">

<android.support.v7.widget.RecyclerView

android:id="@+id/recyclerView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

</RelativeLayout>

**Step 4:** Create a new drawable XML file in Drawable folder and name it custom\_item\_layout.xml and add the following code in it for creating a custom grid item.  
In this step we create a drawable XML file in which we add the code for creating custom grid items.

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

<shape xmlns:android="http://schemas.android.com/apk/res/android">

<!--

stroke with color and width for creating outer line

-->

<stroke

android:width="1dp"

android:color="#000" />

</shape>

**Step 5:** Create a new XML file rowlayout.xml for grid item of RecyclerView and paste the following code in it.  
In this step we create a new xml file for item row in which we creates a [TextView](http://abhiandroid.com/ui/textview) and ImageView to show the data in grid format.

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

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="120dp"

android:layout\_height="120dp"

android:background="@drawable/custom\_item\_layout"

android:padding="5dp">

<!--

grid items for RecyclerView

-->

<ImageView

android:id="@+id/image"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:scaleType="fitXY"

android:src="@mipmap/ic\_launcher" />

<TextView

android:id="@+id/name"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:layout\_centerHorizontal="true"

android:text="ABCD"

android:textSize="20sp"

android:textColor="#fff" />

</RelativeLayout>

**Step 6 :** Now open app -> java -> package -> MainActivity.java and add the below code.

In this step firstly we get the reference of RecyclerView. After that we creates two ArrayList’s for Person Names and Images. After that we set a GridLayoutManager with horizontal orientation and false value for reverseLayout to show the grids to show the items from start to end and then finally we set the Adapter to show the grid items in RecyclerView.

package abhiandroid.com.recyclerviewexample;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.support.v7.widget.GridLayoutManager;

import android.support.v7.widget.RecyclerView;

import java.util.ArrayList;

import java.util.Arrays;

public class MainActivity extends AppCompatActivity {

// ArrayList for person names

ArrayList personNames = new ArrayList<>(Arrays.asList("Person 1", "Person 2", "Person 3", "Person 4", "Person 5", "Person 6", "Person 7","Person 8", "Person 9", "Person 10", "Person 11", "Person 12", "Person 13", "Person 14"));

ArrayList personImages = new ArrayList<>(Arrays.asList(R.drawable.person1, R.drawable.person2, R.drawable.person3, R.drawable.person4, R.drawable.person5, R.drawable.person6, R.drawable.person7,R.drawable.person1, R.drawable.person2, R.drawable.person3, R.drawable.person4, R.drawable.person5, R.drawable.person6, R.drawable.person7));

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// get the reference of RecyclerView

RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recyclerView);

// set a GridLayoutManager with 3 number of columns , horizontal gravity and false value for reverseLayout to show the items from start to end

GridLayoutManager gridLayoutManager = new GridLayoutManager(getApplicationContext(),3, LinearLayoutManager.HORIZONTAL,false);

recyclerView.setLayoutManager(gridLayoutManager); // set LayoutManager to RecyclerView

// call the constructor of CustomAdapter to send the reference and data to Adapter

CustomAdapter customAdapter = new CustomAdapter(MainActivity.this, personNames,personImages);

recyclerView.setAdapter(customAdapter); // set the Adapter to RecyclerView

}

}

**Step 7:** Create a new class CustomAdapter.java inside package and add the following code.  
In this step we create a CustomAdapter class and extends RecyclerView.Adapter class with ViewHolder in it. After that we implement the overrided methods and create a constructor for getting the data from Activity, in this custom Adapter two methods are more important first is onCreateViewHolder in which we inflate the layout item xml and pass it to View Holder and other is onBindViewHolder in which we set the data in the view’s with the help of ViewHolder. Finally we implement the setOnClickListener event on itemview and on click of item we display the selected image in full size in the next Activity.

package abhiandroid.com.recyclerviewexample;

import android.content.Context;

import android.content.Intent;

import android.support.v7.widget.RecyclerView;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageView;

import android.widget.TextView;

import java.util.ArrayList;

public class CustomAdapter extends RecyclerView.Adapter {

ArrayList personNames;

ArrayList personImages;

Context context;

public CustomAdapter(Context context, ArrayList personNames, ArrayList personImages) {

this.context = context;

this.personNames = personNames;

this.personImages = personImages;

}

@Override

public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

// infalte the item Layout

View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.rowlayout, parent, false);

// set the view's size, margins, paddings and layout parameters

MyViewHolder vh = new MyViewHolder(v); // pass the view to View Holder

return vh;

}

@Override

public void onBindViewHolder(MyViewHolder holder, final int position) {

// set the data in items

holder.name.setText(personNames.get(position));

holder.image.setImageResource(personImages.get(position));

// implement setOnClickListener event on item view.

holder.itemView.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

// open another activity on item click

Intent intent = new Intent(context, SecondActivity.class);

intent.putExtra("image", personImages.get(position)); // put image data in Intent

context.startActivity(intent); // start Intent

}

});

}

@Override

public int getItemCount() {

return personNames.size();

}

public class MyViewHolder extends RecyclerView.ViewHolder {

// init the item view's

TextView name;

ImageView image;

public MyViewHolder(View itemView) {

super(itemView);

// get the reference of item view's

name = (TextView) itemView.findViewById(R.id.name);

image = (ImageView) itemView.findViewById(R.id.image);

}

}

}

**Step 8:** Create a new XML file activity\_second.xml and add below code in it.  
In this step we create a ImageView in our XML file to show the selected image in full size.

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

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

android:background="#fff">

<ImageView

android:id="@+id/selectedImage"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_centerInParent="true"

android:scaleType="fitXY" />

</RelativeLayout>

**Step 9:** Create a new Activity and name it SecondActivity.class and add the below code in it.  
In this step we get the reference of ImageView and then get Intent which was set from adapter of Previous Activity and then finally we set the image in ImageView.

package abhiandroid.com.recyclerviewexample;

import android.content.Intent;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.widget.ImageView;

public class SecondActivity extends AppCompatActivity {

ImageView selectedImage;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_second);

selectedImage = (ImageView) findViewById(R.id.selectedImage); // init a ImageView

Intent intent = getIntent(); // get Intent which was set from adapter of Previous Activity

selectedImage.setImageResource(intent.getIntExtra("image", 0)); // get image from Intent and set it in ImageView

}

}