# Unit-6: ListView, GridView and RecycleView

**Unit -6**
**Listview, Gridview and Recyclerview [6 HRS]**
Listview (Introduction, Features, Implementing listview in an application)
Gridview (Introduction, Features, Implementing gridview in an application)
Recyclerview (Introduction, Features, Implementing recyclerview in an application)

## ListView:

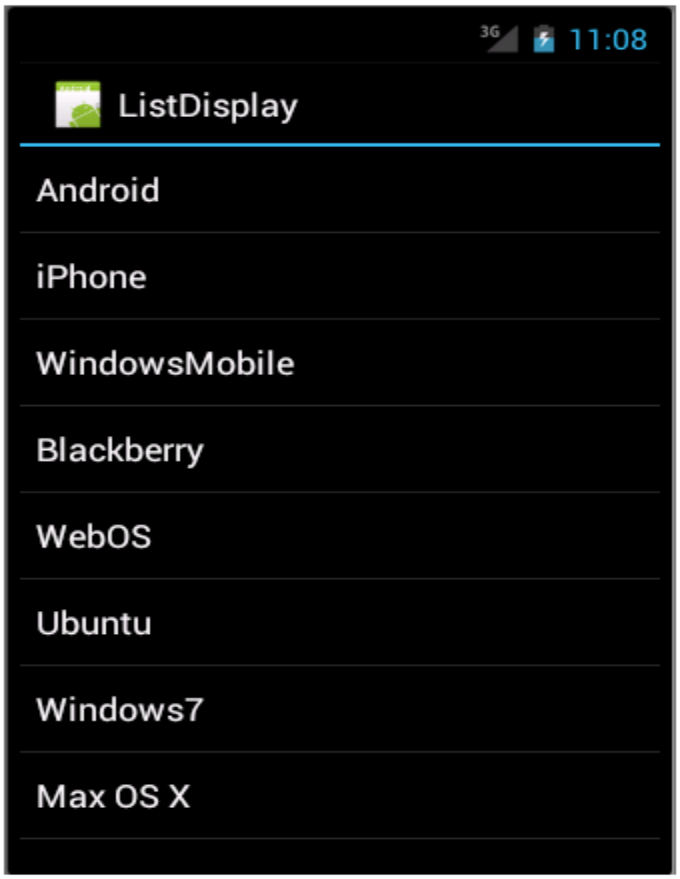
### *Introduction:*

Android ListView is a view which groups several items and display them in vertical scrollable list. Or in another words List of scrollable items can be displayed in Android using ListView. It helps you to display the data in the form of a scrollable list. Users can then select any list item by clicking on it. ListView is default scrollable so we do not need to use scroll View or anything else with ListView.

A very common example of ListView is your phone contact book, where you have a list of your contacts displayed in a ListView and if you click on it then user information is displayed.

Adapter: To fill the data in a ListView we simply use adapters. List items are automatically inserted to a list using an Adapter that pulls the content from a source such as an arraylist, array or database.

Adapter holds the data and send the data to adapter view, the view can takes the data from adapter view and shows the data on different views like as spinner, list view, grid view etc.



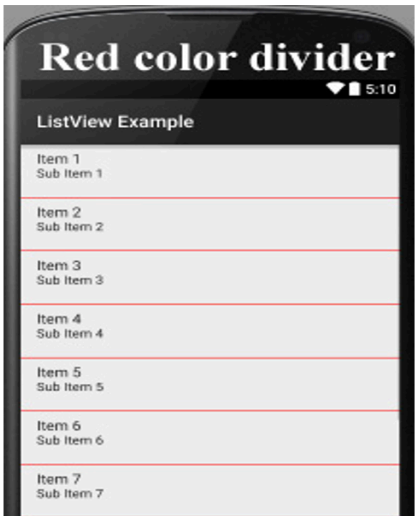

ListView

***ListView Attributes***

**1. id:** id is used to uniquely identify a ListView.

**2. divider:** This is a drawable or color to draw between different list items.
Below is the divider example code with explanation included, where we draw red color divider between different views.

```
<!--Divider code in ListView-->
<ListView
android:id="@+id/simpleListView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:divider="#f00"
android:dividerHeight="1dp"
/>
```



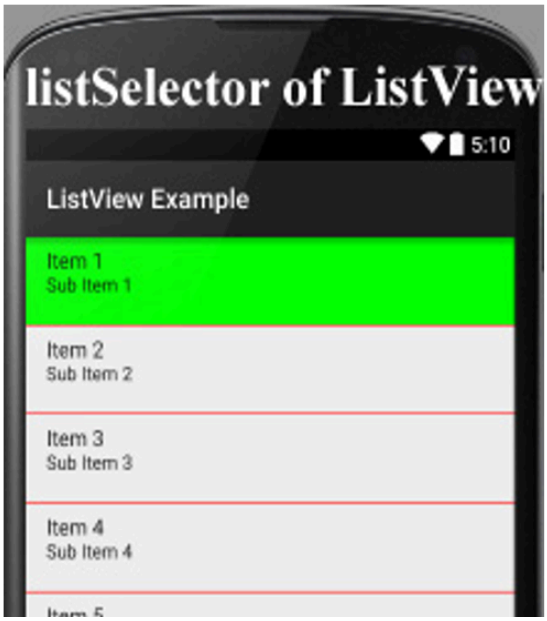

**3. dividerHeight:**

This specify the height of the divider between list items. This could be in dp(density pixel),sp(scale independent pixel) or px(pixel).

In above example of divider we also set the divider height 1dp between the list items. The height should be in dp,sp or px.

**4. listSelector:**

listSelector property is used to set the selector of the listView. It is generally orange or Sky blue color mostly but you can also define your custom color or an image as a list selector as per your design.



```
<!-- List Selector Code in ListView -->
<ListView
android:id="@+id/simpleListView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:divider="#f00"
android:dividerHeight="1dp"
android:listSelector="#0f0"/> <!--list selector in green color-->
```

**5. android:footerDividersEnabled**

When set to false, the ListView will not draw the divider before each footer view. The default value is true.

**6. android:headerDividersEnabled**

When set to false, the ListView will not draw the divider after each header view. The default value is true.

***ArrayAdapter***

You can use this adapter when your data source is an array. By default, ArrayAdapter creates a view for each array item by calling toString() on each item and placing the contents in a **TextView**. Consider you have an array of strings you want to display in a ListView, initialize a new **ArrayAdapter** using a constructor to specify the layout for each string and the string array

**Syntax:**

**ArrayAdapter adapter = new ArrayAdapter <String>**
**(this,R.layout.ListView,StringArray);**

Here are arguments for this constructor

- First argument **this** is the application context. Most of the case, keep it **this**.

- Second argument will be layout defined in XML file and having **TextView** for each string in the array.
- Final argument is an array of strings which will be populated in the text view.

Once you have array adapter created, then simply call **setAdapter()** on your **ListView** object as follows

```
ListView listView = (ListView) findViewById(R.id.listview);

listView.setAdapter(adapter);
```

You can define your list view under res/layout directory in an XML file.

***Features of ListView:***

- It displays a vertically-scrollable collection of views, where each view is positioned immediately below the previous view in the list.
- ListView uses Adapter classes which add the content from data source (such as string, array, database etc.)
- ListView is a default scrollable which does not use other scroll view..
- ListView is implemented by importing ***android.widget.ListView*** class

***Implementing ListView in an Application:***

**Step 1: Create a new project**

- Click on File, then New => New Project.
- Choose “Empty Activity” for the project template.

**Step 2: Modify activity_main.xml file**

Add a ListView in the activity_main.xml file.

**activity_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

   <ListView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/lvid"/>
</LinearLayout>
```

**Step 3: create my_layout.xml**

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

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
    android:id="@+id/textview"></TextView>

</LinearLayout>
```

**Step 3: Modify MainActivity.java file**

**MainActivity.java**

```
package com.example.myapplication;

import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

   ListView lv;
   String myarray[]={"apple","banana","grapes"};
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      lv=findViewById(R.id.lvid);
      ArrayAdapter<String> arr=new
ArrayAdapter<>(this,R.layout.my_layout,R.id.textview,myarray);
      lv.setAdapter(arr);
```

```
    }
}
```

**Note:**

In above program the content in myarry is displayed by list view in my_layout xml file's TettView widget……

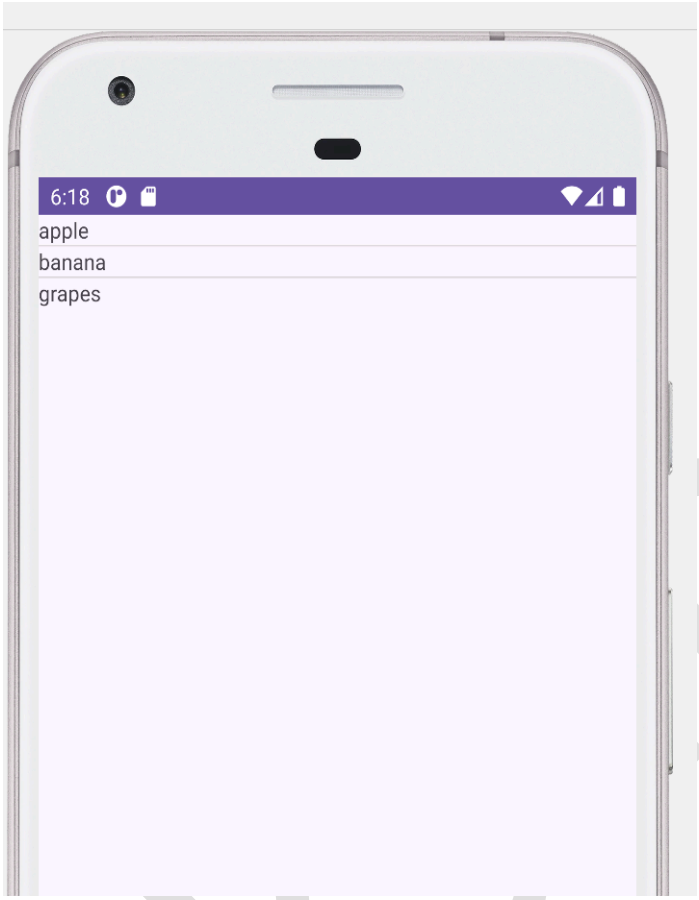
In another words…In the above provided program, the content of the myarray is displayed by the ListView using the layout defined in my_layout.xml. Specifically, each item in the myarray is displayed within the TextView defined in my_layout.xml.

Here,

**lv.setAdapter(arr);**

This line sets the adapter (arr) for the ListView (lv). The ArrayAdapter takes the data from myarray and uses the layout defined in R.layout.my_layout to display each item in the ListView.

Output:

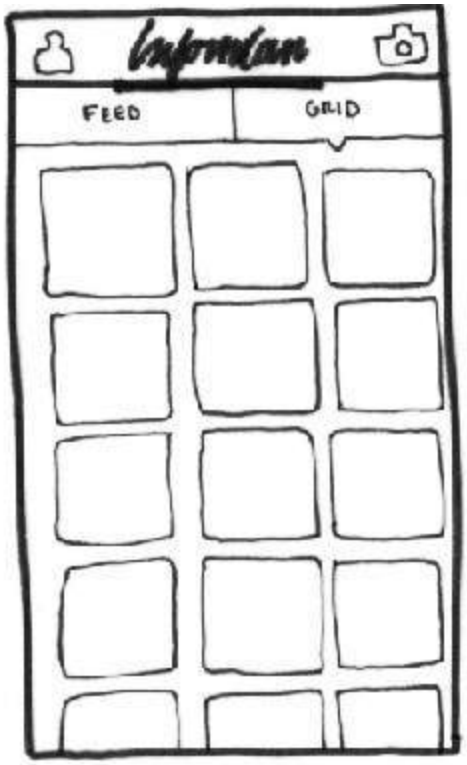



**GridView:**

Android GridView shows items in two-dimensional scrolling grid (rows & columns) and the grid items are not necessarily predetermined but they automatically inserted to the layout using a ListAdapter

An adapter actually bridges between UI components and the data source that fill data into UI Component. Adapter can be used to supply the data to like spinner, list view, grid view etc

The **ListView** and **GridView** are subclasses of **AdapterView** and they can be populated by binding them to an **Adapter**, which retrieves data from an external source and creates a View that represents each data entry.




GridView

***GridView Attributes***

Following are the important attributes specific to GridView

| SN | Attribute & Description |
|---|---|
| 1 | android:id<br>This is the ID which uniquely identifies the layout. |
| 2 | android:columnWidth<br>This specifies the fixed width for each column. This could be in px, dp, sp, in, or mm. |
| 3 | android:gravity |

| | |
|---|---|
| | Specifies the gravity within each cell. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc. |
| 4 | android:horizontalSpacing<br>Defines the default horizontal spacing between columns. This could be in px, dp, sp, in, or mm. |
| 5 | android:numColumns<br>Defines how many columns to show. May be an integer value, such as "100" or auto_fit which means display as many columns as possible to fill the available space. |
| 6 | android:stretchMode<br>Defines how columns should stretch to fill the available empty space, if any. This must be either of the values −<br>none − Stretching is disabled.<br>spacingWidth − The spacing between each column is stretched.<br>columnWidth − Each column is stretched equally.<br>spacingWidthUniform − The spacing between each column is uniformly stretched.. |
| 7 | android:verticalSpacing<br>Defines the default vertical spacing between rows. This could be in px, dp, sp, in, or mm. |

***Features of Grid View:***

- GridView displays items in two-dimensional scrolling grid.
- GridView uses Adapter classes which add the content from data source (such as string, array, database etc.)
- GridView is a default scrollable which does not need other scroll views.
- GridView is implemented by importing ***android.widget.GridView*** class.

***Implementing GridView in an Application***

This example will take you through simple steps to show how to create your own Android application using GridView.

| Step | Description |
|---|---|
| 1 | You will use Android studio IDE to create an Android application and name it as *HelloWorld* under a package *com.example.helloworld* as explained in the *Hello World Example* chapter. |
| 2 | Modify the detault content of *res/layout/activity_main.xml* file to include GridView content with the self explanatory attributes. |
| 3 | No need to change string.xml, Android studio takes care of defaults strings which are placed at string.xml |
| 4 | Let's put few pictures in *res/drawable-hdpi* folder. I have put sample0.jpg, sample1.jpg, sample2.jpg, sample3.jpg, sample4.jpg, sample5.jpg, sample6.jpg and sample7.jpg. |
| 5 | Create a new class called ImageAdapter under a package com.example.helloworld that extends BaseAdapter. This class will implement functionality of an adapter to be used to fill the view. |

| | |
|---|---|
| 6 | Run the application to launch Android emulator and verify the result of the changes done in the application. |

***activity_main.xml***

```
<?xml version="1.0" encoding="utf-8"?>
<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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Cars Brands"
        android:textStyle="bold"
        android:textColor="#000000"
        android:textAlignment="center"
        />
    <GridView
        android:id="@+id/gridView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:verticalSpacing="10dp"
        android:horizontalSpacing="10dp"
        android:layout_marginTop="20dp"
        android:numColumns="2"/>

</LinearLayout>
```

***grid_item.xml***

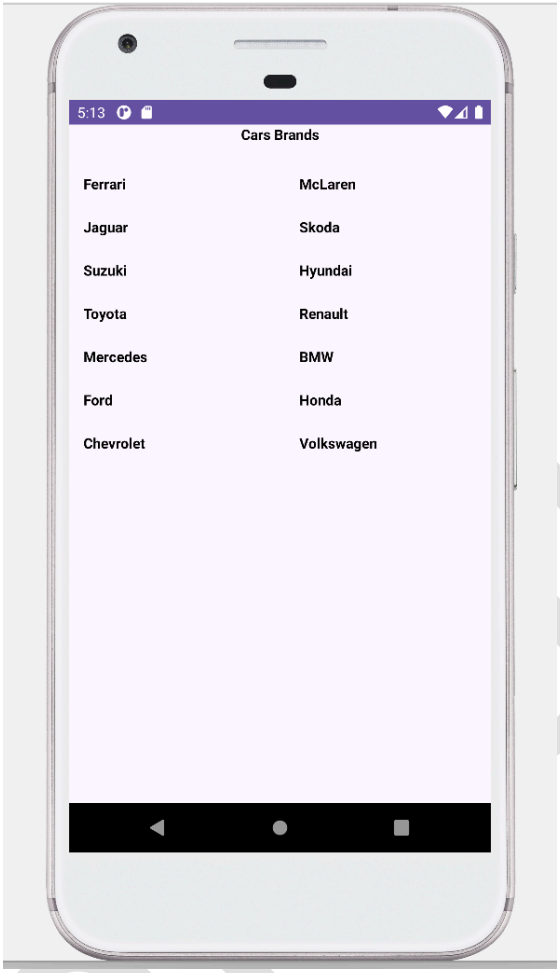
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textStyle="bold"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="5dp"
        android:padding="4dp"
        android:textColor="#000000"
        />

</LinearLayout>
```

**MainActivity.java**

```
package com.example.myapplication;

import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.GridView;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    GridView gv;
    String[] myarray = {
            "Ferrari",
            "McLaren",
            "Jaguar",
            "Skoda",
            "Suzuki",
            "Hyundai",
```

```
        "Toyota",
        "Renault",
        "Mercedes",
        "BMW",
        "Ford",
        "Honda",
        "Chevrolet",
        "Volkswagen",
    };

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

        gv = findViewById(R.id.gridView);
        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
                R.layout.grid_item,R.id.textView, myarray);

        gv.setAdapter(adapter);

        gv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view, int
position, long l) {
                String value = adapter.getItem(position)+ "  is my favorite car " ;
                Toast.makeText(getApplicationContext(), value,
Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

**Output:**




## Recycler View:

In Android, RecyclerView is an advanced and flexible version of ListView and GridView. It is a container used for displaying large amount of data sets that can be scrolled very efficiently by maintaining a limited number of views. RecyclerView was introduced in Material Design in API level 21 (Android 5.0 i.e Lollipop).

In Android, RecyclerView provides an ability to implement the horizontal, vertical and Expandable List. It is mainly used when we have data collections whose elements can change at run time based on user action or any network events. For using this widget we have to specify the Adapter and Layout Manager.

***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.

*Gradle Scripts > build.gradle and inside dependencies*

```
dependencies {
...
compile "com.android.support:recyclerview-v7:23.0.1"
}
```

***Need of RecyclerView In Android***

RecyclerView uses a ViewHolder for storing the reference of the view for one entry in the RecyclerView. When we use ListView or GridView for displaying custom items then we create a custom xml file and then use it inside our Adapter. For this we create a CustomAdapter class and then extends our Base or any other Adapter in it. In getView() method of our Adapter we inflate the item layout xml file and then give the reference of every view by using the unique id's we provide in our xml file . Once finished we pass that view to the ListView, ready to be drawn, but the truth is that ListView and GridView do only half the job of achieving true memory efficiency.

ListView/GridView recycle the item layout but don't keep the reference to the layout children, forcing us to call findViewById() for every child of our item layout for every time we call getView(). This issue causes the scrolling or non-responsive problem as it frantically tries to grab references to the view's we needed.

With the arrival of RecyclerView everything is changed. RecyclerView still uses Adapter to act as Data source but in this we have to create a ViewHolder to keep the reference of View in memory, so when we need a new view it either creates a new ViewHolder object to inflate the layout and hold those references or it recycles one from existing stack.

### *Components of RecyclerView In Android*

Below we define the mainly used components of a RecyclerView.

**1. Layout Managers:**

In Android a RecyclerView needs to have a Layout Manager and an Adapter to be instantiated. Layout Manager is a very new concept introduced in RecyclerView for defining the type of Layout which RecyclerView should use. It Contain 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 but RecyclerView Provides three types of in-built Layout Managers.

- **Linear Layout Manager :** It is used for displaying the data items in a horizontal or vertical scrolling List
- **GridLayoutManager:** It is used to show the items in grid format
- **StaggeredGridLayoutManager:** It is used to show the items in staggered Grid.

**2. ViewHolder:**

ViewHolder is used to store the reference of the View's for one entry in the RecyclerView. A ViewHolder is a static inner class in our Adapter which holds references to the relevant view's. By using these references our code can avoid time consuming findViewById() method to update the widgets with new data.

**3. RecyclerView.Adapter:**

RecyclerView includes a new kind of Adapter. It's a similar approach to the ones we already used but with some peculiarities such as a required ViewHolder for handling Views. We will have to override two main methods first one to inflate the view and its viewholder and another one to bind the data to the view. The main good thing in this is that the first method is called only when we really need to create a new view.

**4. ItemAnimator:**

RecyclerView.ItemAnimator will animate ViewGroup modification such as delete, select, add that notify the Adapter. DefaultItemAnimator can be used for providing default Animation and it works quite well.

***Features of Recycler View:***

- RecyclerView widget is a more advanced and flexible version of ListView. So, we can use RecylerView to display large database.
- RecyclerView contains integrated animations for adding, updating, and removing items.
- RecycleView inforces the recycling of views by using the ViewHolder pattern
- RecyclerView supports both grids and lists.
- RecyclerView supports vertical and horizontal scrolling.

***Implementation of RecycleView in an Application:***

Below is the example of RecyclerView in which we display the list of Person Names with the help of RecyclerView. In this example we are using LinearLayoutManager with vertical orientation 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 an ArrayList for Person Names and set a LayoutManager and finally we set the Adapter to show the items in RecyclerView. Whenever a user clicks on an item the name of the Person is displayed on the screen with the help of Toast.

Example:

**activity_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">


        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/recyclerView"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_margin="16dp"
            android:clipToPadding="false"
            android:padding="4dp"
            android:scrollbars="vertical" />

</LinearLayout>
```

**grid_item.xml**

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

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:padding="8dp" />
</LinearLayout>
```

**MyRecyclerViewAdapter.java**

```
package com.example.myapplication;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import java.util.List;

public class MyRecyclerViewAdapter extends
RecyclerView.Adapter<MyRecyclerViewAdapter.ViewHolder> {

    private List<String> mData;
    private LayoutInflater mInflater;
    private Context context;

    // data is passed into the constructor
    MyRecyclerViewAdapter(Context context, List<String> data) {
        this.mInflater = LayoutInflater.from(context);
        this.mData = data;
        this.context = context;
    }

    // inflates the row layout from xml when needed
    @Override
    @NonNull
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View view = mInflater.inflate(R.layout.grid_item, parent, false);
```

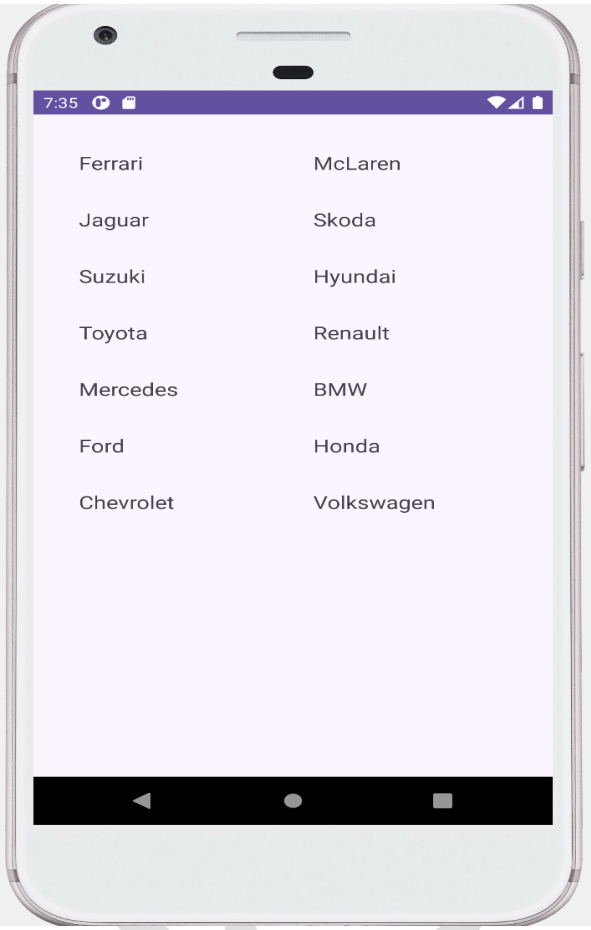
```
        return new ViewHolder(view);
    }

    // binds the data to the TextView in each row
    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        String car = mData.get(position);
        holder.myTextView.setText(car);

        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String value = car + " is my favorite car";
                Toast.makeText(context, value, Toast.LENGTH_SHORT).show();
            }
        });
    }

    // total number of rows
    @Override
    public int getItemCount() {
        return mData.size();
    }

    // stores and recycles views as they are scrolled off screen
    public class ViewHolder extends RecyclerView.ViewHolder {
        TextView myTextView;

        ViewHolder(View itemView) {
            super(itemView);
            myTextView = itemView.findViewById(R.id.textView);
        }
    }
}
```

**MainActivity.java**

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

```
import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import java.util.Arrays;

public class MainActivity extends AppCompatActivity {

   RecyclerView recyclerView;
   String[] myArray = {
         "Ferrari", "McLaren", "Jaguar", "Skoda", "Suzuki", "Hyundai", "Toyota",
         "Renault", "Mercedes", "BMW", "Ford", "Honda", "Chevrolet",
"Volkswagen",
   };

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

      recyclerView = findViewById(R.id.recyclerView);
      recyclerView.setLayoutManager(new GridLayoutManager(this, 2));
      MyRecyclerViewAdapter adapter = new MyRecyclerViewAdapter(this,
Arrays.asList(myArray));
      recyclerView.setAdapter(adapter);
   }
}
```

**OUTPUT:**




**End of Unit-6**