ListView

Reference: https://developer.android.com/reference/android/widget/ListView

Good introduction: https://www.vogella.com/tutorials/AndroidListView/article.html
https://guides.codepath.com/android/Using-an-ArrayAdapter-with-ListView

This document describes how to implement first a simple ListView with clickable items, and then a more advance ListView.

This document describes how to do the following:

A basic list using ListView

1. Create a new **Empty Activity** project and name it **ListView**.

Edit the strings.xml file

2. In the **strings.xml** file create a string array containing the data items that you want in your list. The name of this array is **listItems**

```
<resources>
    <string name="app_name">ListView</string>
    <string-array name="listItems">
        <item>Orange</item>
        <item>Apple</item>
        <item>Banana</item>
        <item>Grape</item>
        <item>Pear</item>
        <item>Cherry</item>
    </string-array>

</resources>
```

Edit the activity_main.xml file

3. Delete the TextView object
4. Add a **ListView** object
    - On a new line, type **<LV**
    - Press Enter to select **ListView** from the popup
    - Press Enter to select **match_parent** for the layout_width

- Press Enter to select **match_parent** for the layout_height
- Type **/** to close the tag with />
- Add an id attribute by typing **id**, select **id**, select **@+id/**, type **listView** for the id name
- Here are some optional attributes:
  divider
  dividerHeight
  listSelector

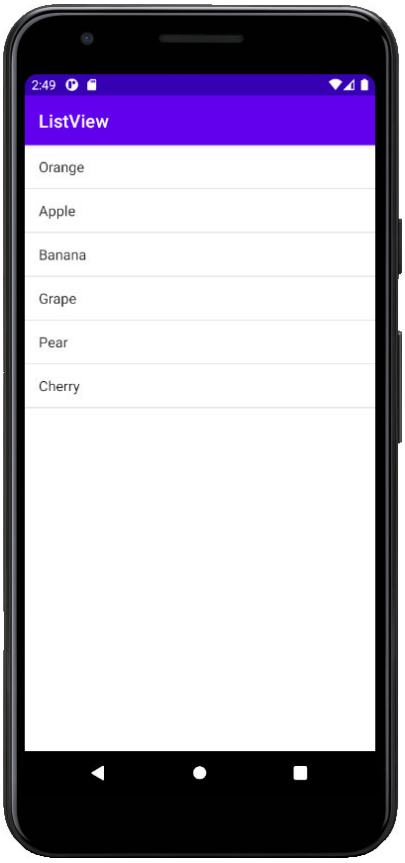
5. In the ListView object that you just defined, add an `android:entries="listItems"` attribute to connect the ListView with the data items

6. Here's the complete **activity_main.xml** file

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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=".MainActivity">

    <ListView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:entries="@array/listItems" />

        <!-- optional attributes -->
        android:divider="#f00" <!-- divider color -->
        android:dividerHeight="1dp" <!-- divider height -->
        android:listSelector="#0f0" <!-- item selected color -->

</androidx.constraintlayout.widget.ConstraintLayout>
```

Run it

7. Run the program to see the list.



ListView using ArrayAdapter

The previous method described does not give you full control to customize the list item attributes. The following method to create a list will give you full control over the list item attributes.

8. Delete the **android:entries="@array/listItems"** attribute for the ListView object in the activity_main.xml file if you have followed the previous instructions to create a list.

Edit the MainActivity.java file

9. In the onCreate method, create a String array variable name **fruits** and initialize it with data items from the string-array **listItems** declared in the strings.xml file

```
String[] fruits = getResources().getStringArray(R.array.listItems);
```

10. In the onCreate method create an **ArrayAdapter** variable name **adapter** and initialize it by connecting the built-in **android.R.layout.simple_list_item_1** layout with the **fruits** data array. This simple_list_item_1 layout contains only one TextView

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, fruits);
```

11. Create a ListView variable name **listView** and use findViewById to connect it to the **listView** layout

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

12. Set the adapter to the listView variable

```
listView.setAdapter(adapter);
```

13. To respond to clicks on the items in the list, call **setOnItemClickListener** on the listView variable in the onCreate method. Inside the **onItemClick** method you can do whatever you want given the position of the item that is passed in

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
        Toast.makeText(parent.getContext(), "selected "+fruits[position]+" at
index "+position, Toast.LENGTH_SHORT).show();
    }
});
```

14. Another way to set the click listener

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        listView.setOnItemClickListener(myItemClickListener);
```

```
    }

    private AdapterView.OnItemClickListener myItemClickListener = new
AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
            Toast.makeText(parent.getContext(), "selected "+fruits[position]+"
at index "+position, Toast.LENGTH_SHORT).show();

        }
    };
```

15. To respond to long clicks on the items in the list, call **setOnItemLongClickListener** on the listView variable in the onCreate method. Inside the **onItemLongClick** method you can do whatever you want given the position of the item that is passed in

```
listView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener()
{
    @Override
    public boolean onItemLongClick(AdapterView<?> parent, View view, int
position, long id) {
        Toast.makeText(getApplicationContext(), "Long click on
"+items[position]+" at index "+position, Toast.LENGTH_SHORT).show();
        return true;
    }
});
```

16. Here's the complete MainActivity.java code

```
package com.example.listview;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

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

        String[] fruits = getResources().getStringArray(R.array.listItems);

        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1, fruits);
        ListView listView = findViewById(R.id.listView);
        listView.setAdapter(adapter);
```

```
        listView.setOn

ItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
                Toast.makeText(parent.getContext(), "selected
"+fruits[position]+" at index "+position, Toast.LENGTH_SHORT).show();
            }
        });

        listView.setOnItemLongClickListener(new
AdapterView.OnItemLongClickListener() {
            @Override
            public boolean onItemLongClick(AdapterView<?> parent, View view, int
position, long id) {
                Toast.makeText(parent.getContext(), "Long click on
"+items[position]+" at index "+position, Toast.LENGTH_SHORT).show();
                return true;
            }
        });

    }
}
```

Run it

Using a custom adapter and custom layout file with three objects

Instead of having just a TextView for each item, it will have an ImageView and two TextViews. To do this we need a custom layout and a custom adapter instead of the ArrayAdapter.

17. Instead of using the built-in `android.R.layout.simple_list_item_1` layout we define our own custom layout file **list_item.xml** as shown next

Create and edit the list_item.xml layout file

18. Create a new layout resource file name **list_item**. This defines the layout for a list item in the list.
    - Right-click on the **layout** folder that is in the app/res folder
    - Select **New** | **Layout Resouce** File
    - Type in **list_item** for the name

19. Here's the complete **list_item.xml** file

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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="wrap_content"
    android:padding="5dp">

    <ImageView
        android:id="@+id/icon"
        android:layout_width="50dp"
        android:layout_height="50dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/firstLine"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="left"
        android:textStyle="bold"
        android:textSize="20sp"
        android:paddingLeft="60dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/icon"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/secondLine"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="left"
```

```
        android:textSize="14sp"
        android:paddingLeft="65dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/icon"
        app:layout_constraintTop_toBottomOf="@+id/firstLine" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Edit the strings.xml file

20. In the **strings.xml** file create three string arrays containing the data items that you want in your list.

```
<resources>
    <string name="app_name">ListView</string>

    <string-array name="listItems">
        <item>Orange</item>
        <item>Apple</item>
        <item>Banana</item>
        <item>Grape</item>
        <item>Pear</item>
        <item>Cherry</item>
    </string-array>

    <string-array name="prices">
        <item>$1.00</item>
        <item>$1.10</item>
        <item>$1.20</item>
        <item>$1.30</item>
        <item>$1.40</item>
        <item>$1.50</item>
    </string-array>

    <string-array name="pictures">
        <item>orange</item>
        <item>apple</item>
        <item>banana</item>
        <item>grape</item>
        <item>pear</item>
        <item>cherry</item>
    </string-array>
</resources>
```

Picture resources

21. Copy and paste six pictures into the **drawable** folder. Use the same names for these pictures as shown in step 20 in the pictures string array.

Edit the MainActivity.java file

22. In the onCreate method create three String arrays for the fruit names, prices and icons

```
String[] fruits = getResources().getStringArray(R.array.listItems);
String[] prices = getResources().getStringArray(R.array.prices);
// int[] icons  = {R.drawable.orange, R.drawable.apple, R.drawable.banana,
R.drawable.grape, R.drawable.pear};
String[] icons = getResources().getStringArray(R.array.pictures);
```

23. In the onCreate method use a custom adapter class type name **MyAdapter** to create a variable name **adapter** and initialize it

```
MyAdapter adapter = new MyAdapter(MainActivity.this, fruits, prices, icons);
```

24. Create a ListView variable **listView** and connect it to the **listView** layout

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

25. Set the adapter to the listView variable

```
listView.setAdapter(adapter);
```

26. Resolve the red error for the MyAdapter
    - Put your cursor on the red error
    - Click on the red bulb
    - Select **Create class MyAdapter**
    - Click OK on the popup window
    - A new MyAdapter.java file with the MyAdapter class is created

27. Resolve the red error for the MainActivity.this
    - Put your cursor on the red error
    - Click on the red bulb
    - Select **Create constructor**
    - Click OK on the popup window

28. Here's the complete **MainActivity.java** file

```
package com.example.listview;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

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

```
        String[] fruits = getResources().getStringArray(R.array.listItems);
        String[] prices = getResources().getStringArray(R.array.prices);
        // int[] icons  = {R.drawable.orange, R.drawable.apple,
R.drawable.banana, R.drawable.grape, R.drawable.pear};
        String[] icons = getResources().getStringArray(R.array.pictures);

        MyAdapter adapter = new MyAdapter(this, fruits, prices, icons);
        ListView listView = findViewById(R.id.listView);
        listView.setAdapter(adapter);

        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
                Toast.makeText(parent.getContext(), "selected
"+fruits[position]+" at index "+position, Toast.LENGTH_SHORT).show();
            }
        });
    }

}
```

Edit the MyAdapter.java file

29. Extend the MyAdapter class with **BaseAdapter**

```
public class MyAdapter extends BaseAdapter {
```

30. Resolve the red error
    - Put your cursor on the red error
    - Click on the red bulb
    - Select **Implement Methods**
    - Click OK on the popup window
    - Four new methods, getCount, getItem, getItemId and getView, are added.

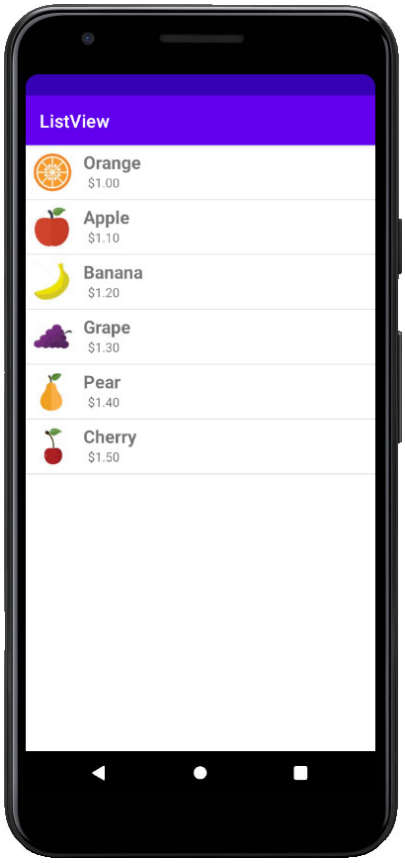
31. Here's the complete **MyAdapter.java** file

```
package com.example.listview;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.TextView;

public class MyAdapter extends BaseAdapter {
    Context context;
    LayoutInflater inflater;
    private String[] fruits;
    private String[] prices;
```

```
    private String[] icons;

    //    public MyAdapter(MainActivity mainActivity, String[] fruits, String[]
prices, int[] icons) {
    public MyAdapter(MainActivity mainActivity, String[] fruits, String[] prices,
String[] icons) {
        context = mainActivity;
        inflater = LayoutInflater.from(context);
        this.fruits = fruits;
        this.prices = prices;
        this.icons = icons;
    }

    @Override
    public int getCount() {
        return fruits.length;
    }

    @Override
    public Object getItem(int position) {
        return null;
    }

    @Override
    public long getItemId(int position) {
        return 0;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        if (convertView == null) {
            convertView = inflater.inflate(R.layout.list_item, null);
        }
        // ((ImageView)
convertView.findViewById(R.id.icon)).setImageResource(pictures[position]);
        ((ImageView)
convertView.findViewById(R.id.icon)).setImageResource(context.getResources().getI
dentifier(icons[position], "drawable", context.getPackageName()));
        ((TextView)
convertView.findViewById(R.id.firstLine)).setText(fruits[position]);
        ((TextView)
convertView.findViewById(R.id.secondLine)).setText(prices[position]);
        return convertView;
    }
}
```

Run it

32. That's it.


ListView
Orange
$1.00
Apple
$1.10
Banana
$1.20
Grape
$1.30
Pear
$1.40
Cherry
$1.50

Different methods to define the list item data array

33. Another way to initialize the pictures array

```
private String[] pics;
private int[] pictures;

public MyAdapter(MainActivity mainActivity) {
    pics = mainActivity.getResources().getStringArray(R.array.pictures);
    pictures = new int[fruits.length];
    for (int i=0; i<fruits.length; i++)
        pictures[i] = context.getResources().getIdentifier(pics[i],
"drawable", context.getPackageName());
}

public View getView(int position, View convertView, ViewGroup parent) {
    ((ImageView)
convertView.findViewById(R.id.image1)).setImageResource(pictures[position]);
}
```

34. Instead of defining the string-array list data items in the strings.xml file, you can do that in MyAdapter.java

```
public class MyAdapter extends BaseAdapter {
    Context context;
    LayoutInflater inflater;

//    private String[] fruits;
//    private String[] prices;
//    private String[] pictures;
    private String[] fruits = {"Apple", "Banana", "Orange", "Grape", "Pear"};
    private String[] prices = {"$1.00", "$1.10", "$1.20", "$1.30", "$1.40"};
    private int[] pictures  = {R.drawable.apple, R.drawable.banana,
R.drawable.orange, R.drawable.grape, R.drawable.pear};

    public MyAdapter(MainActivity mainActivity) {
        context = mainActivity;
        inflater = LayoutInflater.from(mainActivity);
//        fruits = mainActivity.getResources().getStringArray(R.array.fruits);
//        prices = mainActivity.getResources().getStringArray(R.array.prices);
//        pictures =
mainActivity.getResources().getStringArray(R.array.pictures);
    }

    @Override
    public int getCount() {
        return fruits.length;
    }

    @Override
    public Object getItem(int position) {
        return null;
    }
```

```
    @Override
    public long getItemId(int position) {
        return 0;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        if (convertView == null) {
            convertView = inflater.inflate(R.layout.list_item, null);
        }
        ((ImageView)
convertView.findViewById(R.id.image1)).setImageResource(pictures[position]);
//        ((ImageView)
convertView.findViewById(R.id.image1)).setImageResource(context.getResources()
.getIdentifier(pictures[position], "drawable", context.getPackageName()));
        ((TextView)
convertView.findViewById(R.id.text1)).setText(fruits[position]);
        ((TextView)
convertView.findViewById(R.id.text2)).setText(prices[position]);
        return convertView;
    }
}
```

Adding a Fast Scroll Bar

To enable the fast scroll bar for the list, either add the **android:fastScrollEnabled="true"** attribute to the listView in the XML file or **listView.setFastScrollEnabled(true)** in the java file.

Note that the scroll bar will be shown only if the listview height is 4x more than listview's visible height. Or you can set the `android:fastScrollAlwaysVisible="true"`

Method 1: Edit the activity_main.xml file

```
<ListView
    android:id="@+id/listView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fastScrollEnabled="true"
    android:fastScrollAlwaysVisible="true"
    android:listSelector="@color/lightgrey"/>
```

Method 2: Edit the MainActivity.java file

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

listView.setFastScrollEnabled(true);
listView.setFastScrollAlwaysVisible(true);
```

35. To test this out and to keep things simple we will use a layout with just one TextView and add many more items to the listItems string array as shown next

Edit the list_item.xml file

```
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/listItem"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="10dp"
    android:textStyle="bold"
    android:textSize="16dp"
    android:textAppearance="@style/TextAppearance.Compat.Notification.Title"
    />
```

Edit the strings.xml file

36. Use the following list of data items in the strings.xml file for this example

```
<resources>
    <string name="app_name">List</string>
    <string-array name="listItems">
        <item>A1</item>
        <item>A2</item>
```

```
        <item>A3</item>
        <item>A4</item>
        <item>A5</item>
        <item>B1</item>
        <item>B2</item>
        <item>B3</item>
        <item>B4</item>
        <item>B5</item>
        <item>C1</item>
        <item>C2</item>
        <item>C3</item>
        <item>C4</item>
        <item>C5</item>
        <item>D1</item>
        <item>D2</item>
        <item>D3</item>
        <item>D4</item>
        <item>D5</item>
        <item>E1</item>
        <item>E2</item>
        <item>E3</item>
        <item>E4</item>
        <item>E5</item>
        <item>F1</item>
        <item>F2</item>
        <item>F3</item>
        <item>F4</item>
        <item>F5</item>
        <item>G1</item>
        <item>G2</item>
        <item>G3</item>
        <item>G4</item>
        <item>G5</item>
        <item>H1</item>
        <item>H2</item>
        <item>H3</item>
        <item>H4</item>
        <item>H5</item>
        <item>I1</item>
        <item>I2</item>
        <item>I3</item>
        <item>I4</item>
        <item>I5</item>
        <item>J1</item>
        <item>J2</item>
        <item>J3</item>
        <item>J4</item>
        <item>J5</item>
        <item>K1</item>
        <item>K2</item>
        <item>K3</item>
        <item>K4</item>
        <item>K5</item>
        <item>L1</item>
        <item>L2</item>
```

```
        <item>L3</item>
        <item>L4</item>
        <item>L5</item>
        <item>M1</item>
        <item>M2</item>
        <item>M3</item>
        <item>M4</item>
        <item>M5</item>
        <item>N1</item>
        <item>N2</item>
        <item>N3</item>
        <item>N4</item>
        <item>N5</item>
        <item>O1</item>
        <item>O2</item>
        <item>O3</item>
        <item>O4</item>
        <item>O5</item>
        <item>P1</item>
        <item>P2</item>
        <item>P3</item>
        <item>P4</item>
        <item>P5</item>
        <item>Q1</item>
        <item>Q2</item>
        <item>Q3</item>
        <item>Q4</item>
        <item>Q5</item>
        <item>R1</item>
        <item>R2</item>
        <item>R3</item>
        <item>R4</item>
        <item>R5</item>
        <item>S1</item>
        <item>S2</item>
        <item>S3</item>
        <item>S4</item>
        <item>S5</item>
        <item>T1</item>
        <item>T2</item>
        <item>T3</item>
        <item>T4</item>
        <item>T5</item>
        <item>U1</item>
        <item>U2</item>
        <item>U3</item>
        <item>U4</item>
        <item>U5</item>
        <item>V1</item>
        <item>V2</item>
        <item>V3</item>
        <item>V4</item>
        <item>V5</item>
        <item>W1</item>
        <item>W2</item>
```

```
        <item>W3</item>
        <item>W4</item>
        <item>W5</item>
        <item>X1</item>
        <item>X2</item>
        <item>X3</item>
        <item>X4</item>
        <item>X5</item>
        <item>Y1</item>
        <item>Y2</item>
        <item>Y3</item>
        <item>Y4</item>
        <item>Y5</item>
        <item>Z1</item>
        <item>Z2</item>
        <item>Z3</item>
        <item>Z4</item>
        <item>Z5</item>
    </string-array>

</resources>
```

37. We need to make the appropriate corresponding changes in both MainActivity.java and MyAdapter.java files

Edit the MainActivity.java file

```
package com.example.listview;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

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

        String[] items = getResources().getStringArray(R.array.listItems);
        // MyAdapter adapter = new MyAdapter(this, R.layout.list_item, items);
        MyAdapter adapter = new MyAdapter(this, items);
        ListView listView = findViewById(R.id.listView);
        listView.setAdapter(adapter);
        listView.setFastScrollEnabled(true);

        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
```

```
            public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
                Toast.makeText(parent.getContext(), "selected "+items[position]+" at
index "+position, Toast.LENGTH_SHORT).show();
            }
        });
    }

}
```

Edit the MyAdapter.java file

```
package com.example.listview;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;

public class MyAdapter extends BaseAdapter {
    Context context;
    LayoutInflater inflater;
    private String[] list;

    // public MyAdapter(MainActivity mainActivity, int layout, String[] list) {
    public MyAdapter(MainActivity mainActivity, String[] list) {
        context = mainActivity;
        inflater = LayoutInflater.from(context);
        this.list = list;
    }

    @Override
    public int getCount() {
        return list.length;
    }

    @Override
    public Object getItem(int position) {
        return null;
    }

    @Override
    public long getItemId(int position) {
        return 0;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        if (convertView == null) {
            convertView = inflater.inflate(R.layout.list_item, null);
        }
        ((TextView) convertView.findViewById(R.id.listItem)).setText(list[position]);
```

```
        return convertView;
    }
}
```

Fast scrollbar color

38. To change the fast scrollbar color edit the file **res | values | themes | themes.xml** and change the color for **colorSecondary**

    ```
    <item name="colorSecondary">@color/purple_500</item>
    ```

Run it

Adding the Section Indexer to the Fast Scroll Bar

https://androidopentutorials.com/android-listview-fastscroll/

Edit the MyAdapter.java file

39. To add the section indexer to the fast scrollbar we need to implement the **SectionIndexer** to our adapter.

```
public class MyAdapter extends BaseAdapter implements SectionIndexer {
```

40. Resolve the red error for the MyAdapter
    - Put your cursor on the red error
    - Click on the red bulb
    - Select **Implement Methods**
    - Click OK on the popup window
    - Three new methods, **getSections**, **getPositionForSection** and **getSectionForPosition**, will be added

41. Inside the **MyAdapter** class declare a String array containing the section names that you want to use

```
    String[] sections = new String[] {
"A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S"
,"T","U","V","W","X","Y","Z"};
```

42. Add in the code for the three new methods.

43. Here's the complete code for the MyAdapter.java file

```
package com.example.listview;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.SectionIndexer;
import android.widget.TextView;

public class MyAdapter extends BaseAdapter implements SectionIndexer {
    Context context;
    LayoutInflater inflater;
    private String[] list;

    String[] sections = new String[] {
"A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","
V","W","X","Y","Z"};

    public MyAdapter(MainActivity mainActivity, String[] list) {
        context = mainActivity;
```

```
        inflater = LayoutInflater.from(context);
        this.list = list;
    }

    @Override
    public int getCount() {
        return list.length;
    }

    @Override
    public Object getItem(int position) {
        return list[position];
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        if (convertView == null) {
            convertView = inflater.inflate(R.layout.list_item, null);
        }
        ((TextView) convertView.findViewById(R.id.listItem)).setText(list[position]);
        return convertView;
    }

    @Override
    public Object[] getSections() {
        return sections;
    }

    @Override
    public int getPositionForSection(int sectionIndex) {
        if (sectionIndex == 0)      // section 0 (A) starts at index 0
            return 0;
        else if (sectionIndex == 1) // section 1 (B) starts at index 5
            return 5;
        else if (sectionIndex == 2) // section 2 (C) starts at index 10
            return 10;
        else if (sectionIndex == 3) // section 3 (D) starts at index 15
            return 15;
        else if (sectionIndex == 4) // section 4 (E) starts at index 20
            return 20;
        else if (sectionIndex == 5) // section 5 (F) starts at index 25
            return 25;
        else if (sectionIndex == 6) // section 6 (G) starts at index 30
            return 30;
        else if (sectionIndex == 7) // section 7 (H) starts at index 35
            return 35;
        else if (sectionIndex == 8) // section 8 (I) starts at index 40
            return 40;
        else if (sectionIndex == 9) // section 9 (J) starts at index 45
            return 45;
```

```
        else if (sectionIndex == 10) // section 10 (K) starts at index 50
            return 50;
        else if (sectionIndex == 11) // section 11 (L) starts at index 55
            return 55;
        else if (sectionIndex == 12) // section 12 (M) starts at index 60
            return 60;
        else if (sectionIndex == 13) // section 13 (N) starts at index 65
            return 65;
        else if (sectionIndex == 14) // section 14 (O) starts at index 70
            return 70;
        else if (sectionIndex == 15) // section 15 (P) starts at index 75
            return 75;
        else if (sectionIndex == 16) // section 16 (Q) starts at index 80
            return 80;
        else if (sectionIndex == 17) // section 17 (R) starts at index 85
            return 85;
        else if (sectionIndex == 18) // section 18 (S) starts at index 90
            return 90;
        else if (sectionIndex == 19) // section 19 (T) starts at index 95
            return 95;
        else if (sectionIndex == 20) // section 20 (U) starts at index 100
            return 100;
        else if (sectionIndex == 21) // section 21 (V) starts at index 105
            return 105;
        else if (sectionIndex == 22) // section 22 (W) starts at index 110
            return 110;
        else if (sectionIndex == 23) // section 23 (X) starts at index 115
            return 115;
        else if (sectionIndex == 24) // section 24 (Y) starts at index 120
            return 120;
        else
            return 125;             // section 25 (Z) starts at index 125
    }

    @Override
    public int getSectionForPosition(int position) {
        if(position < 5)         // positions 0 to 4 are in section 0 (A)
            return 0;
        else if(position < 10)    // positions 5 to 9 are in section 1 (B)
            return 1;
        else if(position < 15)    // positions 10 to 14 are in section 2 (C)
            return 2;
        else if(position < 20)    // positions 15 to 19 are in section 3 (D)
            return 3;
        else if(position < 25)    // positions 20 to 24 are in section 4 (E)
            return 4;
        else if(position < 30)    // positions 25 to 29 are in section 5 (F)
            return 5;
        else if(position < 35)    // positions 30 to 34 are in section 6 (G)
            return 6;
        else if(position < 40)    // positions 35 to 39 are in section 7 (H)
            return 7;
        else if(position < 45)    // positions 40 to 44 are in section 8 (I)
            return 8;
        else if(position < 50)    // positions 45 to 49 are in section 9 (J)
```

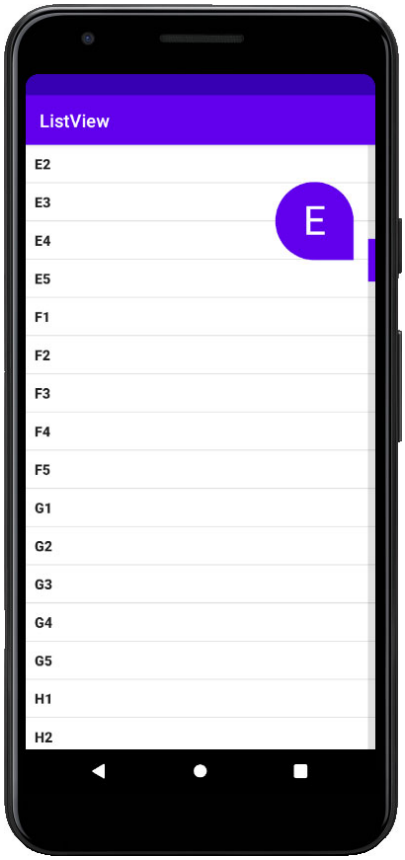
```
            return 9;
        else if(position < 55)    // positions 50 to 54 are in section 10 (K)
            return 10;
        else if(position < 60)    // positions 55 to 59 are in section 11 (L)
            return 11;
        else if(position < 65)    // positions 60 to 64 are in section 12 (M)
            return 12;
        else if(position < 70)    // positions 65 to 69 are in section 13 (N)
            return 13;
        else if(position < 75)    // positions 70 to 74 are in section 14 (O)
            return 14;
        else if(position < 80)    // positions 75 to 79 are in section 15 (P)
            return 15;
        else if(position < 85)    // positions 80 to 84 are in section 16 (Q)
            return 16;
        else if(position < 90)    // positions 85 to 89 are in section 17 (R)
            return 17;
        else if(position < 95)    // positions 90 to 94 are in section 18 (S)
            return 18;
        else if(position < 100)    // positions 95 to 99 are in section 19 (T)
            return 19;
        else if(position < 105)    // positions 100 to 104 are in section 20 (U)
            return 20;
        else if(position < 110)    // positions 105 to 109 are in section 21 (V)
            return 21;
        else if(position < 115)    // positions 110 to 114 are in section 22 (W)
            return 22;
        else if(position < 120)    // positions 115 to 119 are in section 23 (X)
            return 23;
        else if(position < 125)    // positions 120 to 124 are in section 24 (Y)
            return 24;
        else
            return 25;             // positions 125 to 129 are in section 25 (Z)
    }
}
```

44. No changes are needed in MainActivity.java after adding the fast scrollbar

```
package com.example.listview;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

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

        String[] items = getResources().getStringArray(R.array.listItems);
        MyAdapter adapter = new MyAdapter(this, items);
        ListView listView = findViewById(R.id.listView);
        listView.setAdapter(adapter);
        listView.setFastScrollEnabled(true);

        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
                Toast.makeText(parent.getContext(), "selected "+items[position]+" at
index "+position, Toast.LENGTH_SHORT).show();
            }
        });
    }

}
```

Run it



Automatic calculation of the section labels and positions

45. The following sample code is to automatically find the section labels and positions. Make the changes in the ListAdapter.java file.

```
String[] sectionLabels;
int[] positionForSection;

// return the section labels given the list of items
private void getSectionLabels(String[] items) {
    ArrayList<String> tempSections = new ArrayList<>();
    ArrayList<Integer> tempPositionForSection = new ArrayList<>();
    String currentSection = "";
    for (int position=0; position<items.length; position++) {
        if (!items[position].substring(0, 1).equals(currentSection)) {
            currentSection = items[position].substring(0, 1);
            tempSections.add(currentSection);       // save new section label
            tempPositionForSection.add(position);   // save new section position
        }
    }
    // init the sectionLabels and positionForSection arrays
    sectionLabels = new String[tempSections.size()];
    positionForSection = new int[tempPositionForSection.size()];
    for (int i=0; i<tempPositionForSection.size(); i++) {
        sectionLabels[i] = tempSections.get(i);
        positionForSection[i] = tempPositionForSection.get(i);
    }
}


// automatic calculation of the section positions
public int getPositionForSection(int sectionIndex) {
    return positionForSection[sectionIndex];
}

// automatic calculation of the section positions
public int getSectionForPosition(int position) {
    int i;
    for (i=0; i<positionForSection.length-1; i++) {
        if (position < positionForSection[i+1]) return i;
    }
    return i;
}
```

Improving performance with the ViewHolder pattern

Reference: https://guides.codepath.com/android/Using-an-ArrayAdapter-with-ListView

To improve performance, we should modify the custom adapter by applying the ViewHolder pattern which speeds up the population of the ListView considerably by caching view lookups for smoother, faster item loading

```
public class MyAdapter extends ArrayAdapter<User> {
   // View lookup cache
   private static class ViewHolder {
      TextView name;
      TextView home;
   }

   public MyAdapter(Context context, ArrayList<User> users) {
     super(context, R.layout.item_user, users);
   }

   @Override
   public View getView(int position, View convertView, ViewGroup parent) {
     // Get the data item for this position
     User user = getItem(position);
     // Check if an existing view is being reused, otherwise inflate the view
     ViewHolder viewHolder; // view lookup cache stored in tag
     if (convertView == null) {
       // If there's no view to re-use, inflate a brand new view for row
       viewHolder = new ViewHolder();
       LayoutInflater inflater = LayoutInflater.from(getContext());
       convertView = inflater.inflate(R.layout.item_user, parent, false);
       viewHolder.name = (TextView) convertView.findViewById(R.id.tvName);
       viewHolder.home = (TextView) convertView.findViewById(R.id.tvHome);
       // Cache the viewHolder object inside the fresh view
       convertView.setTag(viewHolder);
     } else {
        // View is being recycled, retrieve the viewHolder object from tag
        viewHolder = (ViewHolder) convertView.getTag();
     }
     // Populate the data from the data object via the viewHolder object
     // into the template view.
     viewHolder.name.setText(user.name);
     viewHolder.home.setText(user.hometown);
     // Return the completed view to render on screen
     return convertView;
   }
}
```

~~Basic row delete and row move~~

~~Reference: https://www.journaldev.com/23208/android-recyclerview-drag-and-drop~~
~~https://www.journaldev.com/23164/android-recyclerview-swipe-to-delete-undo~~

~~*Edit the MainActivity.java file*~~

46. ~~To delete or move a row we need to implement a custom **StartDragListener** to the MainActivity. We will also add a Snackbar to undo the row delete.~~

~~public class MainActivity extends AppCompatActivity implements StartDragListener {~~

~~47. Resolve the red StartDragListener error and select Create Interface StartDragListener~~

~~48. Here's the complete **StartDragListener.java** file~~

```
package com.example.listview;

import androidx.recyclerview.widget.RecyclerView;

public interface StartDragListener {
    void requestDrag(RecyclerView.ViewHolder viewHolder);
}
```

Create the ic_drag.xml file

This is the drag icon

49. Create the **ic_drag.xml** file in the **res | drawable** folder.

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24"
    android:viewportHeight="24"
    android:tint="?attr/colorControlNormal">
  <path
      android:fillColor="@android:color/white"
      android:pathData="M20,9H4v2h16V9zM4,15h16v-2H4V15z"/>
</vector>
```

Create the ic_delete.xml file

This is the delete icon

50. Create the **ic_delete.xml** file in the **res | drawable** folder.

```
<vector android:height="24dp" android:tint="#FFFFFF"
    android:viewportHeight="24.0" android:viewportWidth="24.0"
    android:width="24dp" xmlns:android="http://schemas.android.com/apk/res/android">
    <path android:fillColor="#FF000000" android:pathData="M6,19c0,1.1 0.9,2
2,2h8c1.1,0 2,-0.9 2,-2V7H6v12zM19,4h-3.5l-1,-1h-5l-1,1H5v2h14V4z"/>
</vector>
```