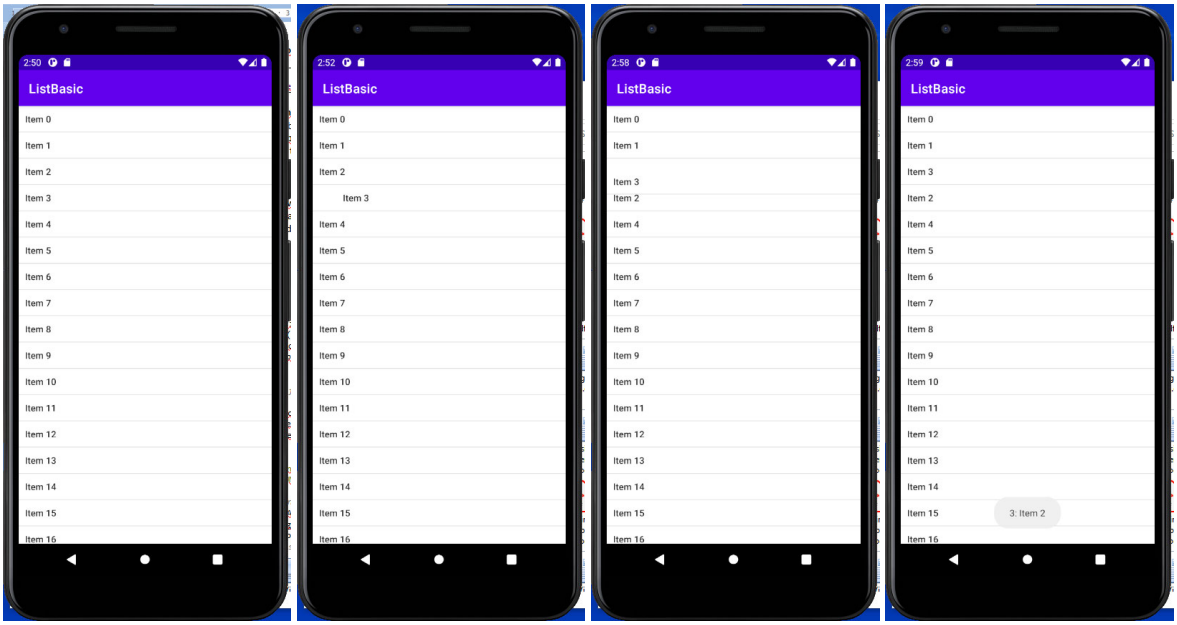
A Basic List using RecyclerView

Reference: https://developer.android.com/guide/topics/ui/layout/recyclerview
https://developer.android.com/guide/topics/ui/layout/recyclerview-custom
https://www.javatpoint.com/android-recyclerview-list-example
https://github.com/android/views-widgets-samples/tree/main/RecyclerView
https://www.journaldev.com/23208/android-recyclerview-drag-and-drop
https://www.journaldev.com/23164/android-recyclerview-swipe-to-delete-undo

This document describes how to do the following:

- Create a basic list using the RecyclerView
- Data is stored in a static array structure
- Click to select row
- Default swipe to delete row
- Default drag to move row



List Swipe to delete Drag to move Click to select

- A basic list using RecyclerView
- Add divider lines between items
- Detecting item clicks
- Delete a row or move a row
- Add a new item to the list
- Adding a Fast Scroll Bar
- Adding the Section Indexer to the Fast Scroll Bar

When inserting code, type in as much as possible and select the suggested line from the popup. For keywords that have more than one word, you can type the first letter of each word for it to popup. Be

careful that letters are case sensitive. Press **Enter** to accept the suggestion from the popup. For example to enter **ImageView**, you can type **IV** then press **Enter**.

To resolve an error after typing in a statement correctly, put your cursor on the error highlighted in red and either press Alt+Enter, or click the red bulb that appears. Then select the correct suggestion from the popup. In most cases, this will correct the error by automatically inserting missing boilerplate code.

A basic list using RecyclerView

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

Edit the strings.xml file

2. In the **strings.xml** file (in the folder **res** | **values**) 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">ListBasic</string>
    <string-array name="fruits">
        <item>Item 0</item>
        <item>Item 1</item>
        <item>Item 2</item>
        <item>Item 3</item>
        <item>Item 4</item>
        <item>Item 5</item>
        <item>Item 6</item>
        <item>Item 7</item>
        <item>Item 8</item>
        <item>Item 9</item>
        <item>Item 10</item>
        <item>Item 11</item>
        <item>Item 12</item>
        <item>Item 13</item>
        <item>Item 14</item>
        <item>Item 15</item>
        <item>Item 16</item>
        <item>Item 17</item>
        <item>Item 18</item>
        <item>Item 19</item>
        <item>Item 20</item>
    </string-array>
</resources>
```

Edit the activity_main.xml file

3. Delete the TextView object
4. Add a **RecyclerView** object
    - On a new line, type **<RV**
    - Press Enter to select **androidx.recyclerview.widget.RecyclerView** 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 **recyclerView** for the id name

5. 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">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

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

Create the list_item.xml file

6. 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 app | res folder
    - Select **New** | **Layout Resouce** File
    - Type in **list_item** for the name
7. Replace file content with the following
8. Here's the complete **list_item.xml** file

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/relativeLayout"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="10dp">

    <TextView
        android:id="@+id/text1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:textAppearance="@style/TextAppearance.Compat.Notification.Title"/>

</RelativeLayout>
```

Edit the MainActivity.java file

9. In the onCreate method create a **RecyclerView** variable name **recyclerView** and initialize it by connecting it to the recyclerView object that you created in activity_main.xml.

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

10. Set the layout manager to the view

```
recyclerView.setLayoutManager(new LinearLayoutManager(this));
```

11. Create a myAdapter variable.

```
MyAdapter myAdapter = new MyAdapter(this);
```

12. Set the recyclerView adapter to the myAdapter variable.

```
recyclerView.setAdapter(myAdapter);
```

13. Resolve the red MyAdapter error
    - 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

14. Resolve the red new MyAdapter(this) error
    - Put your cursor on the red error
    - Click on the red bulb
    - Select **Create constructor**
    - Click OK on the popup window
    - A new MyAdapter constructor method is created in the MyAdapter.java file

15. Don't resolve the red setAdapter(myAdapter) error yet. It will be resolved after the next few steps.

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

```
package com.example.listbasic;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

        RecyclerView recyclerView = findViewById(R.id.recyclerView);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        MyAdapter myAdapter = new MyAdapter(this);
        recyclerView.setAdapter(myAdapter);

    }
}
```

17. Here's an alternate version of the **MainActivity** class

```
public class MainActivity extends AppCompatActivity {
    RecyclerView recyclerView;
    MyAdapter adapter;

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

        recyclerView = findViewById(R.id.recyclerView);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        adapter = new MyAdapter(this);
        recyclerView.setAdapter(adapter);
    }
}
```

Edit the MyAdapter.java file

18. Add **extends** RecyclerView.Adapter<MyAdapter.MyViewHolder> after the class MyAdapter

```
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.MyViewHolder> {
```

19. Resolve the red MyAdapter class error
    - Put your cursor on the red class MyAdapter error
    - Click on the red bulb
    - Select **Implement methods**
    - Click OK on the popup window
    - The three methods, onCreateViewHolder, onBindViewHolder, and getItemCount, are added

20. Resolve the red MyViewHolder error
    - Put your cursor on the red MyViewHolder error
    - Click on the red bulb
    - Select **Create class MyViewHolder**
    - Click OK on the popup window
    - The MyViewHolder class is added

21. Resolve the red MyAdapter.MyViewHolder class error
    - Put your cursor on the red MyAdapter.MyViewHolder error
    - Click on the red bulb
    - Select **Make MyViewHolder extend androidx.recyclerview.widget.RecyclerView.ViewHolder**

22. Resolve the red MyViewHolder class error
    - Put your cursor on the red public class MyViewHolder extends error
    - Click on the red bulb
    - Select **Create constructor matching super**

23. There should be no more errors with an empty template of the MyAdapter class. Of course you could have manually type all this in. Now type in the rest of the code for each of the methods as shown next. There will be several red errors. Just press Alt+Enter to resolve them.

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

```
package com.example.listbasic;

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

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

public class MyAdapter extends RecyclerView.Adapter<MyAdapter.MyViewHolder> {
    Context context;
    private String[] fruits;

    public MyAdapter(MainActivity mainActivity) {
        context = mainActivity;
        fruits = context.getResources().getStringArray(R.array.fruits);
    }

    @NonNull
    @Override
    public MyAdapter.MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.list_item, parent, false);
        return new MyViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull MyAdapter.MyViewHolder holder, int
position) {
        holder.fruit.setText(fruits[position]);
    }

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

    public class MyViewHolder extends RecyclerView.ViewHolder {
        RelativeLayout relativeLayout;
        TextView fruit;
        public MyViewHolder(@NonNull View itemView) {
            super(itemView);
            relativeLayout = itemView.findViewById(R.id.relativeLayout);
            fruit = itemView.findViewById(R.id.text1);
```

```
        }
    }
}
```

Run it

25. That's it. Run the app on an actual device.

Add divider lines between items

Edit the MainActivity.java file

26. Add the following two lines in the **onCreate** method.

```
// add divider lines between items
DividerItemDecoration dividerItemDecoration = new
DividerItemDecoration(getApplicationContext(), DividerItemDecoration.VERTICAL);
recyclerView.addItemDecoration(dividerItemDecoration);
```

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

    RecyclerView recyclerView = findViewById(R.id.recyclerView);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));
    MyAdapter myAdapter = new MyAdapter(this);
    recyclerView.setAdapter(myAdapter);

    // add divider lines between items
    DividerItemDecoration dividerItemDecoration = new
DividerItemDecoration(getApplicationContext(), DividerItemDecoration.VERTICAL);
    recyclerView.addItemDecoration(dividerItemDecoration);
```

Detecting item clicks

Edit the MyAdapter.java file

Method 1:

This method is better/more efficient than Method 2

27. Implement the **OnClickListener** in the **MyViewHolder** class

```
public class MyViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener {
```

28. Resolve the red OnClickListener error
    - Put your cursor on the red OnClickListener error
    - Click on the red bulb
    - Select **Implement methods**
    - Click OK on the popup window
    - The **onClick** method is added

29. Call the setOnClickListener in the MyViewHolder constructor

```
public MyViewHolder(@NonNull View itemView) {
    super(itemView);
    relativeLayout = itemView.findViewById(R.id.relativeLayout);
    fruit = itemView.findViewById(R.id.text1);
    itemView.setOnClickListener(this);
}
```

30. In the onClick method, call the getLayoutPosition to get the position of the item clicked. Then you can do whatever you want with it.

```
    @Override
    public void onClick(View v) {
        int position = getLayoutPosition();
        Toast.makeText(context, position+": " +fruits[position],
Toast.LENGTH_SHORT).show();
    }
```

31. Here's the complete MyViewHolder class

```
public class MyViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener {
    RelativeLayout relativeLayout;
    TextView fruit;

    public MyViewHolder(@NonNull View itemView) {
        super(itemView);
        relativeLayout = itemView.findViewById(R.id.relativeLayout);
        fruit = itemView.findViewById(R.id.text1);
        itemView.setOnClickListener(this);
```

```
    }

    @Override
    public void onClick(View v) {
        int position = getLayoutPosition();
        Toast.makeText(v.getContext(), position+": " +fruits[position],
Toast.LENGTH_SHORT).show();

    }
}
```

Method 2:

32. Implement the **setOnClickListener** in the **onBindViewHolder** method. Note that the `relativeLayout` variable is declared in MyViewHolder.

```
    @Override
    public void onBindViewHolder(@NonNull MyAdapter.MyViewHolder holder, int
position) {
        holder.fruit.setText(fruits[position]);
        holder.relativeLayout.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(v.getContext(), position+": "
+fruits[position], Toast.LENGTH_SHORT).show();
            }
        });
    }
```

Delete a row or move a row

Edit the MyAdapter.java file

33. Create the **moveItem** and **deleteItem** methods at the end of the MyAdapter class. These two methods contain the code to actually perform the respective operations.

```
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.MyViewHolder> {
    Context context;

...

    void moveItem(int position1, int position2) {
        Log.d("myTag", "dragItem from "+position1+" to "+position2);
        Collections.swap(Arrays.asList(fruits), position1, position2);
        notifyItemMoved(position1, position2); // notify the recyclerView
    }

    void deleteItem(int position) {
        Log.d("myTag", "deleteItem Position "+position);
        ArrayList<String> tmp = new ArrayList<String>(Arrays.asList(fruits));
        tmp.remove(position);
        fruits = new String[tmp.size()];
        tmp.toArray(fruits);
        notifyItemRemoved(position); // notify the recyclerView
    }
}
```

Optimization. It is better to just use an ArrayList<String> variable instead of the String[] variable to store the fruits. Notice that in the deleteItem method, a temporary ArrayList<String> data structure is created and initialized with the contents of the String array fruits. The item is deleted from the ArrayList and then it is used to recreate and reinitialize the String array.

Edit the MainActivity.java file

34. In the **MainActivity** class, declare the **itemTouchHelper** variable.

```
public class MainActivity extends AppCompatActivity {
    ItemTouchHelper itemTouchHelper;
```

35. In the **onCreate** method implement the **itemTouchHelper** to listen to touches on the recyclerView.
    - Initialize the **itemTouchHelper** variable by assigning a new instance. The directions specified in the **ItemTouchHelper.SimpleCallback** determines what gesture directions are detected for what operation. The first parameter directions is for the move and the second parameter directions is for the delete.
    - Resolve the itemTouchHelper error by clicking on the red bulb and selecting implement methods.
    - The two methods, **onMove** and **onSwiped** are added. The onMove method is called when the up and down gesture directions are detected. The onSwiped method is called when the left and right gesture directions are detected.
    - Type in the rest of the code in the onMove and onSwiped methods.

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

    RecyclerView recyclerView = findViewById(R.id.recyclerView);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));
    MyAdapter myAdapter = new MyAdapter(this);
    recyclerView.setAdapter(myAdapter);

    // handle the touches
    // in the SimpleCallback the first parameter are the directions for the
move and the second parameter are the directions for the delete
    itemTouchHelper = new ItemTouchHelper(new
ItemTouchHelper.SimpleCallback(ItemTouchHelper.UP | ItemTouchHelper.DOWN,
ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT) {

        @Override
        public boolean onMove(@NonNull RecyclerView recyclerView, @NonNull
RecyclerView.ViewHolder source, @NonNull RecyclerView.ViewHolder destination)
{
            // code to swap elements here
            int from = source.getAdapterPosition();
            int to = destination.getAdapterPosition();
            myAdapter.moveItem(from, to);
            return true;    // must return true to allow drag and drop
reordering
        }

        @Override
        public void onSwiped(@NonNull RecyclerView.ViewHolder viewHolder, int
direction) {
```

```
            // code to delete item from list here
            int position = viewHolder.getAdapterPosition();
            myAdapter.deleteItem(position);
        }
    });
    // attach the itemTouchHelper to the recyclerView
    itemTouchHelper.attachToRecyclerView(recyclerView);
}
```

Run it

36. That's it. Run the app on an actual device.

    Swip left or right on the item to delete the item.

    Long press on an item and then drag up or down to move the item.

Add a new item to the list

Edit the activity_main.xml file

37. Add an EditText object to the layout. This allows the user to enter a fruit name.
38. Add a Button object to the layout. Clicking this button will add the fruit name in the EditText to the list.
39. 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:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/fruitName"
        android:layout_width="276dp"
        android:layout_height="49dp"
        android:layout_marginStart="50dp"
        android:layout_marginLeft="32dp"
        android:layout_marginEnd="32dp"
        android:layout_marginRight="32dp"
        android:hint="item"
        app:layout_constraintEnd_toStartOf="@+id/addButton"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/addButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="1dp"
        android:layout_marginEnd="4dp"
        android:layout_marginRight="4dp"
        android:text="add"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="0dp"
        android:layout_height="683dp"
        android:layout_marginTop="8dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/fruitName" />

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

Edit the MainActivity.java file

40. Write the code to handle the Add button click by adding the following code in the onCreate method.

```
// Handle the add button click
Button button = findViewById(R.id.addButton);
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Log.d("myTag", "Add button clicked");
        EditText newFruit = findViewById(R.id.fruitName);
        String fruit = newFruit.getText().toString();
        newFruit.setText("");
        myAdapter.addItem(fruit);
    }
});
```

Edit the MyAdapter.java file

41. Add the **addItem** method in the MyAdapter class with the code to actually add the new item to the list.

```
void addItem(String fruit) {
    Log.d("myTag", "addItem "+fruit);
    ArrayList<String> tmp = new ArrayList<String>(Arrays.asList(fruits));
    tmp.add(fruit);
    fruits = new String[tmp.size()];
    tmp.toArray(fruits);
    notifyDataSetChanged();
    // notifyItemInserted(tmp.size());
}
```

Run it

42. That's it. Run the app on an actual device.
    Type a fruit in the EditText box and press the Add button to add the item in the list.

Adding a Fast Scroll Bar

Reference: https://heartbeat.fritz.ai/adding-fast-scroll-to-recyclerview-in-android-5a963e2b509d
https://stackoverflow.com/questions/45370246/how-to-use-fastscrollenabled-in-recyclerview
https://github.com/quiph/RecyclerView-FastScroller

Edit the activity_main.xml file

43. Set the **fastScrollEnabled** flag to true in the RecyclerView.

```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

android:scrollbarSize="18dp"
app:fastScrollHorizontalThumbDrawable="@drawable/thumb_drawable"
app:fastScrollHorizontalTrackDrawable="@drawable/line_drawable"
app:fastScrollVerticalThumbDrawable="@drawable/thumb_drawable"
app:fastScrollVerticalTrackDrawable="@drawable/line_drawable"
app:fastScrollEnabled="true"/>
```

44. Add the drawable file **thumb.xml** with the following content

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

    <corners
        android:radius="50dp" />

    <padding
        android:paddingLeft="22dp"
        android:paddingRight="22dp" />

    <solid android:color="@color/teal_700" />

</shape>
```

45. Add the drawable file **thumb_drawable.xml** with the following content

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:state_pressed="true"
        android:drawable="@drawable/thumb"/>

    <item
        android:drawable="@drawable/thumb"/>
</selector>
```

46. Add the drawable file **line.xml** with the following content

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

    <solid android:color="@android:color/darker_gray" />

    <padding
        android:top="10dp"
        android:left="10dp"
        android:right="10dp"
        android:bottom="10dp"/>
</shape>
```

47. Add the drawable file **line_drawable.xml** with the following content

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:state_pressed="true"
        android:drawable="@drawable/line"/>

    <item
        android:drawable="@drawable/line"/>
</selector>
```

Adding the Section Indexer to the Fast Scroll Bar

Reference: https://developer.android.com/reference/android/widget/AlphabetIndexer
https://developer.android.com/reference/android/widget/SectionIndexer

https://stackoverflow.com/questions/38507825/android-alphabetical-fast-scrollview-in-recyclerview-with-collapsing-toolbar
https://blog.stylingandroid.com/scrolling-recyclerview-part-1/
https://blog.stylingandroid.com/recyclerview-fastscroll-part-1/
https://blog.stylingandroid.com/recyclerview-fastscroll-part-2/

https://www.programcreek.com/java-api-examples/?class=android.widget.SectionIndexer&method=getSections

Edit the java file

48. Create a new class name **SectionIndexerAdapter** that extends **ArrayAdapter<String>** and implements the **SectionIndexer**

```
private class SectionIndexerAdapter extends ArrayAdapter<String> implements
    SectionIndexer {
```

49. Inside this **SectionIndexerAdapter** class declare a String array containing the section names that you want

```
    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"};
```

50. Create the constructor and override three methods that are needed inside the **SectionIndexerAdapter** class

```
public SectionIndexerAdapter(Context context, int list_item, String[] items) {
    super(context, list_item, items);
}


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

@Override
public int getPositionForSection(int sectionIndex) {
}

@Override
```

```
public int getSectionForPosition(int position) {
}
```

51. The **getSections()** method simply returns the String array containing the section names

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

52. The **getPositionForSection(int sectionIndex)** method returns the starting position in the data list for the given section index. The data list is the array containing the list of items. The starting position is the index of this array where you want the given section to start.

```
@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 54
        return 54;
    else if (sectionIndex == 2) // section 2 (C) starts at index 79
        return 79;
    else if (sectionIndex == 3) // section 3 (D) starts at index 122
        return 122;
    else if (sectionIndex == 4) // section 4 (E) starts at index 140
        return 140;
    else if (sectionIndex == 5) // section 5 (F) starts at index
        ...
    }
}
```

53. The **getSectionForPosition(int position)** method returns the section number for the given position in the data list

```
@Override
public int getSectionForPosition(int position) {
    if(position < 54)         // positions 0 to 53 are in section 0 (A)
        return 0;
    else if(position < 79)    // positions 54 to 78 are in section 1 (B)
        return 1;
    else if(position < 122)    // positions 79 to 121 are in section 2 (C)
        return 2;
    else if(position < 140)    // positions 122 to 139 are in section 3 (D)
        return 3;
    else if(position < 148)    // positions 140 to 147 are in section 4 (E)
        return 4;
    else if(position < 176)    // positions 148 to 175 are in section 5 (F)
        ...
     }
```

```
}
```

54. Finally, create a new instance of the **SectionIndexerAdapter** in your code passing to it the context, layout, and the data array of your list

```
listView = findViewById(R.id.listView);
listView.setFastScrollEnabled(true);
listView.setAdapter(adapter);
SectionIndexerAdapter adapter = new SectionIndexerAdapter(this,
R.layout.list_item, items);
```