Exercise #3
Mobile Device Development
List View – Recycler View
Due: 92/225/2020@11:59pm

Part 1: ListView

One of the UI component we use often is ***ListView*** – i.e when we need to show items in a vertical scrolling list. One interesting aspect is this component can be deeply customized and can be adapted to our needs. In this exercise, we will analyze the basic concepts behind it and how to use this component.

In order to show items inside the list, we use an *adapter* to create views for list item and to populate the list. First, we need to create or use existing adapters. Android SDK has some standard adapters that are useful in many cases. If you have specific scenarios, you can create custom adapters as discussed in the lecture.

In this example, suppose we want to show a list of planets. To make things simple and focus our attention on the ***ListView*** component we will use, a very simple and basic adapter called ***SimpleAdapter***.

Let's create an android project with an activity called ***MainActivity*** and a layout called ***activity_main.xml***. This layout is very simple, it just contains the listView component.

```
01    <RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
02        xmlns:tools="http://schemas.android.com/tools"
03        android:layout_width="match_parent"
04        android:layout_height="match_parent" >
05
06        <ListView android:id="@+id/listView"
07                  android:layout_height="match_parent"
08                  android:layout_width="match_parent"/>
09
10    </RelativeLayout>
```

Our MainActivity class is:

```
01    public class MainActivity extends Activity {
02
03        @Override
04        public void onCreate(Bundle savedInstanceState) {
05            super.onCreate(savedInstanceState);
06            setContentView(R.layout.activity_main);
07            ....
08        }
09    ...
10    }
```

To populate our listView, we use SimpleAdapter, a standard adapter present in SDK. Let's suppose we want to show a list with the solar system planets. SimpleAdapter accepts a **java.util.List** with element's type of **java.util.Map**. In our case we have then:

```
01 // The data to show
02 List<Map<String, String>> planetsList = new
   ArrayList<Map<String,String>>();
03
04 .....
05
06    private void initList() {
07     // We populate the planets
08
09     planetsList.add(createPlanet("planet", "Mercury"));
10     planetsList.add(createPlanet("planet", "Venus"));
11     planetsList.add(createPlanet("planet", "Mars"));
12     planetsList.add(createPlanet("planet", "Jupiter"));
13     planetsList.add(createPlanet("planet", "Saturn"));
14     planetsList.add(createPlanet("planet", "Uranus"));
15     planetsList.add(createPlanet("planet", "Neptune"));
16
17 }
18
19 private HashMap<String, String> createPlanet(String key, String name) {
20     HashMap<String, String> planet = new HashMap<String, String>();
21     planet.put(key, name);
22
23     return planet;
24 }
```

and the SimpleAdapter can be instantiated as:

```
1     // This is a simple adapter that accepts as parameter
2     // Context
3     // Data list
4     // The row layout that is used during the row creation
5     // The keys used to retrieve the data
6     // The View id used to show the data. The key number and the view id
7     // must match
8     SimpleAdapter simpleAdpt = new SimpleAdapter(this, planetsList,
android.R.layout.simple_list_item_1, new String[] {"planet"}, new int[]
{android.R.id.text1});
```

where the first parameter is the context reference (our Activity), the second the data we want to show in the list. The 3rd is the layout we want to use for each row in the list. This is a very simple layout that contains just a ***TextView*** with id ***text1.*** The 4th parameter is an array of keys

used to retrieve the data from the Map. Each list element of java.util.List represents a row in the ListView and each element inside the row must have a unique key that is used to retrieve the element content. In our case to make things very simple we just used planet as key. The 5th element is an array of ***int*** representing the ids of the View inside the row layout. In our case is just text1 id. Please notice that the keys array length must match the ids array length.

We have almost done. Let's modify the onCreate method in our Activity like that:

```
01    @Override
02    public void onCreate(Bundle savedInstanceState) {
03        super.onCreate(savedInstanceState);
04        setContentView(R.layout.activity_main);
05
06        initList();
07
08        // We get the ListView component from the layout
09        ListView lv = (ListView) findViewById(R.id.listView);
10
11        // This is a simple adapter that accepts as parameter
12        // Context
13        // Data list
14        // The row layout that is used during the row creation
15        // The keys used to retrieve the data
16        // The View id used to show the data. The key number and the view
17        // id must match
18        SimpleAdpt simpleAdpt = new SimpleAdapter(this, planetsList,
android.R.layout.simple_list_item_1, new String[] {"planet"}, new int[]
{android.R.id.text1});
19
20        lv.setAdapter(simpleAdpt);
21    }
```

If we run our project we have:

User interaction

Once you created your list and populated it with the items, you want to interact with the users giving them a chance to click on an item. To do it we have to register the appropriate listener.

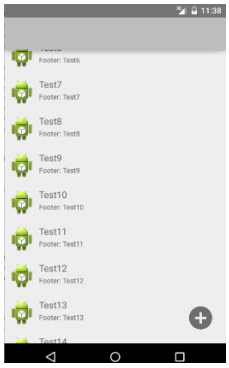
If we want to listen when the user clicks on an item we simply have to implement the ***AdapterView.OnItemClickListener().***

```
01    // React to user clicks on item
02    lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
03
04         public void onItemClick(AdapterView<?> parentAdapter, View view,
05             int position, long id) {
06
07             TextView clickedView = (TextView) view;
08
09             Toast.makeText(MainActivity.this, "Item with id ["+id+"] -
Position ["+position+"] - Planet ["+clickedView.getText()+"]",
Toast.LENGTH_SHORT).show();
10
11         }
12    });
```

When the user clicks on an item, it shows the position and id of the item clicked using a Toast.

Part 2: RecyclerView

In part 2, you will create a project which uses the `RecyclerView` class to display a list.



Create project and add the Gradle dependency

Create a new Android project using the edu.sjsu.android.recyclerview top level package name. Add the following dependency to your Gradle build file.

```
dependencies {
    ...
    implementation "androidx.recyclerview:recyclerview:1.1.0"
}
```

Create layout files

Create or update the layout file called ***activity_main.xml*** so that it contains the ***RecyclerView***.

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

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

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_marginBottom="12dp"
        android:layout_marginRight="12dp"
        android:elevation="2dp"
        android:src="@android:drawable/ic_menu_add" />
</RelativeLayout>
```

Note: The `ImageView` has the `android:elevation` attribute set to draw a shadow for the image.

Also create a layout called ***row_layout*** to use in each item.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="?android:attr/listPreferredItemHeight"
    android:padding="6dip" >

    <ImageView
        android:id="@+id/icon"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:layout_alignParentBottom="true"
        android:layout_alignParentTop="true"
        android:layout_marginRight="6dip"
        android:contentDescription="TODO"
        android:src="@mipmap/ic_launcher" />

    <TextView
        android:id="@+id/secondLine"
        android:layout_width="fill_parent"
        android:layout_height="26dip"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_toRightOf="@id/icon"
```

```
        android:ellipsize="marquee"
        android:singleLine="true"
        android:text="Description"
        android:textSize="12sp" />

    <TextView
        android:id="@+id/firstLine"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_above="@id/secondLine"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:layout_alignWithParentIfMissing="true"
        android:layout_toRightOf="@id/icon"
        android:gravity="center_vertical"
        android:text="Example application"
        android:textSize="16sp" />

</RelativeLayout>
```

Create Adapter Class

```
package edu.sjsu.android.recyclerview;

import java.util.ArrayList;
import java.util.List;
import androidx.recyclerview.widget.RecyclerView;import
android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.TextView;

public class MyAdapter extends RecyclerView.Adapter<MyAdapter.ViewHolder> {
    private List<String> values;

    // Provide a reference to the views for each data item
    // Complex data items may need more than one view per item, and
    // you provide access to all the views for a data item in a view holder
    public class ViewHolder extends RecyclerView.ViewHolder {
        // each data item is just a string in this case
        public TextView txtHeader;
        public TextView txtFooter;
        public View layout;

        public ViewHolder(View v) {
            super(v);
            layout = v;
            txtHeader = (TextView) v.findViewById(R.id.firstLine);
            txtFooter = (TextView) v.findViewById(R.id.secondLine);
        }
    }
    public void add(int position, String item) {
        values.add(position, item);
        notifyItemInserted(position);
    }
```

```
    public void remove(int position) {
        values.remove(position);
        notifyItemRemoved(position);
    }

    // Provide a suitable constructor (depends on the kind of dataset)
    public MyAdapter(List<String> myDataset) {
        values = myDataset;
    }

    // Create new views (invoked by the layout manager)
    @Override
    public MyAdapter.ViewHolder onCreateViewHolder(ViewGroup parent,
                                                   int viewType) {
        // create a new view
        LayoutInflater inflater = LayoutInflater.from(parent.getContext());
        View v = inflater.inflate(R.layout.row_layout, parent, false);
        // set the view's size, margins, paddings and layout parameters
        ViewHolder vh = new ViewHolder(v);
        return vh;
    }

    // Replace the contents of a view (invoked by the layout manager)
    @Override
    public void onBindViewHolder(ViewHolder holder, final int position) {
        // - get element from your dataset at this position
        // - replace the contents of the view with that element
        final String name = values.get(position);
        holder.txtHeader.setText(name);
        holder.txtHeader.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                remove(position);
            }
        });
        holder.txtFooter.setText("Footer: " + name);
    }

    // Return the size of your dataset (invoked by the layout manager)
    @Override
    public int getItemCount() {
        return values.size();
    }
}
```

Create Activity Class

Now you can configure the `RecyclerView` in your activity.

```
package edu.sjsu.android.recyclerview;

import android.app.Activity;
import android.os.Bundle;
import androidx.recyclerview.widget.ItemTouchHelper;
import androidx.recyclerview.widget.LinearLayoutManager;
```

```
import androidx.recyclerview.widget.RecyclerView;
import java.util.ArrayList;
import java.util.List;

public class MainActivity extends Activity {
    private RecyclerView recyclerView;
    private RecyclerView.Adapter mAdapter;
    private RecyclerView.LayoutManager layoutManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        recyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);
        // use this setting to
        // improve performance if you know that changes
        // in content do not change the layout size
        // of the RecyclerView
        recyclerView.setHasFixedSize(true);
        // use a linear layout manager
        layoutManager = new LinearLayoutManager(this);
        recyclerView.setLayoutManager(layoutManager);
        List<String> input = new ArrayList<>();
        for (int i = 0; i < 100; i++) {
            input.add("Test" + i);
        }// define an adapter
        mAdapter = new MyAdapter(input);
        recyclerView.setAdapter(mAdapter);
    }
}
```

Add swipe to dismiss support to your RecyclerView

We want to implement swipe to dismiss so that the user can swipe elements out.

```
// put this after your definition of your recyclerview
// input in your data mode in this example a java.util.List
// adapter is your adapter
 ItemTouchHelper.SimpleCallback simpleItemTouchCallback =
                new ItemTouchHelper.SimpleCallback(0, ItemTouchHelper.LEFT |
ItemTouchHelper.RIGHT) {
    @Override
    public boolean onMove(RecyclerView recyclerView,
       RecyclerView.ViewHolder viewHolder, RecyclerView.ViewHolder target) {
         return false;
    }
    @Override
    public void onSwiped(RecyclerView.ViewHolder viewHolder, int swipeDir) {
         input.remove(viewHolder.getAdapterPosition());
         adapter.notifyItemRemoved(viewHolder.getAdapterPosition());
    }
 };
 ItemTouchHelper itemTouch = new ItemTouchHelper(simpleItemTouchCallback);
 itemTouch.attachToRecyclerView(recyclerView);
```

Submission

1. Push your project directory along with the source to remote bitbucket repository by the due date.
2. Invite and share your project repository the Grader (yan.chen01@sjsu.edu) and Instructor (ramin.moazeni@sjsu.edu).
3. Submit a Readme.pdf to Canvas including your name, repository access link, instructions to run your program (if any), snapshot of your running program
4. Your project directory will be graded according to the state your project directory was in at due time when fetched.