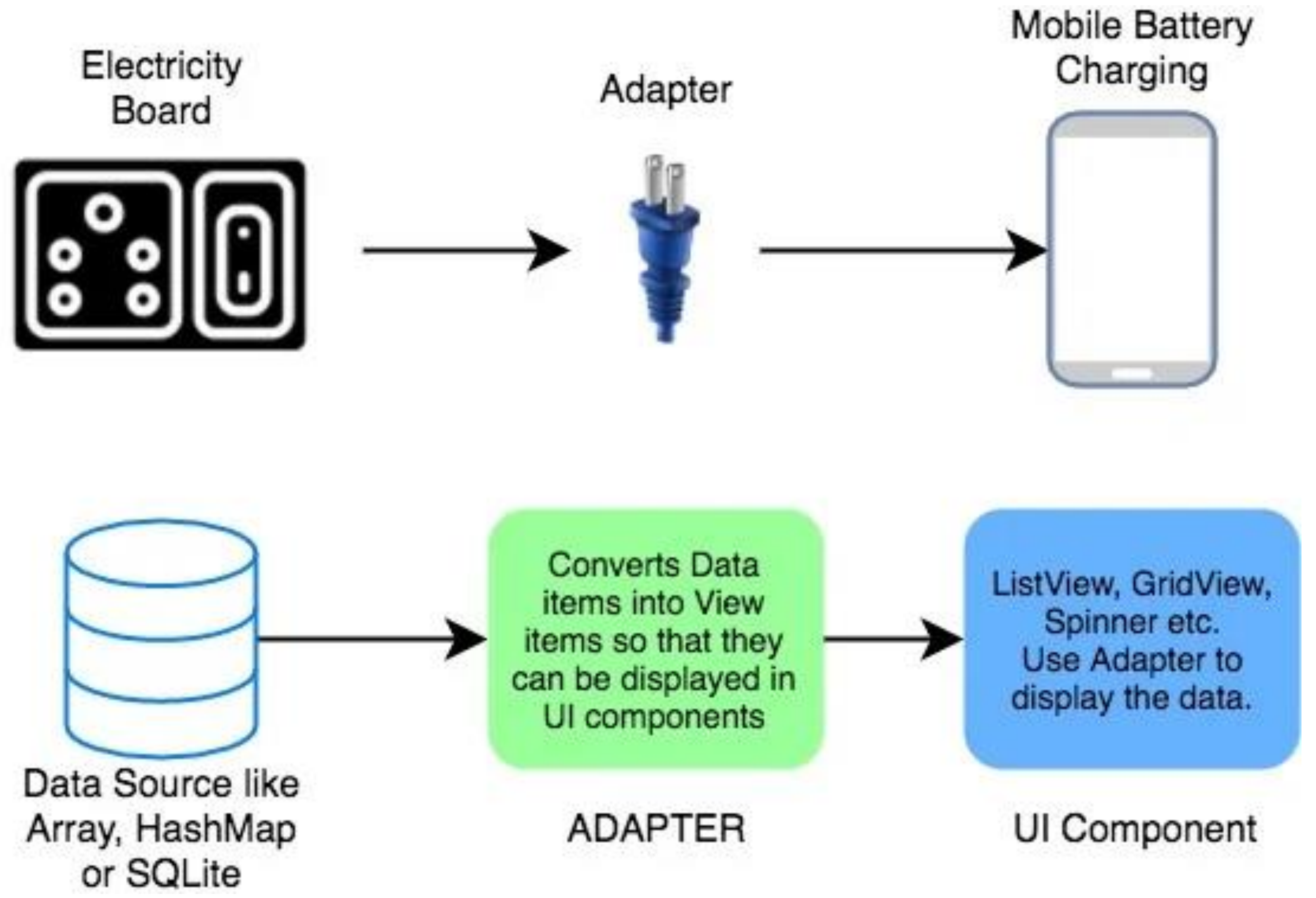
Android Lists & Adapters

- The display of elements in a lists is a very common pattern in mobile applications. The user sees a list of items and can scroll through them.
- **Adapter** is basically bridge between UI components and the data source that fill data into UI Component. Adapter is used to supply the data to like spinner, list view, grid view etc.
- Adapters are used to provide the data to the ListView object. The adapter also defines how each row in the ListView is displayed.
- The adapter is assigned to the ListView via the setAdapter() method on the ListView object.

Adapters


Electricity Board
Adapter
Mobile Battery Charging
Converts Data items into View items so that they can be displayed in UI components
ListView, GridView, Spinner etc. Use Adapter to display the data.
Data Source like Array, HashMap or SQLite
ADAPTER
UI Component

Adapters

- There are the some commonly used Adapter in Android used to fill the data in the UI components.

a) **BaseAdapter –** It is parent adapter for all other adapters

b) **ArrayAdapter –** It is used whenever we have a list of single items which is backed by an array

c) **Custom ArrayAdapter –** It displays the custom list of an Array.

d) **SimpleAdapter –** It is an easy adapter to map static data to views defined in your XML file

e) **Custom SimpleAdapter –** It is used whenever we need to display a customized list and needed to access the child items of the list or grid

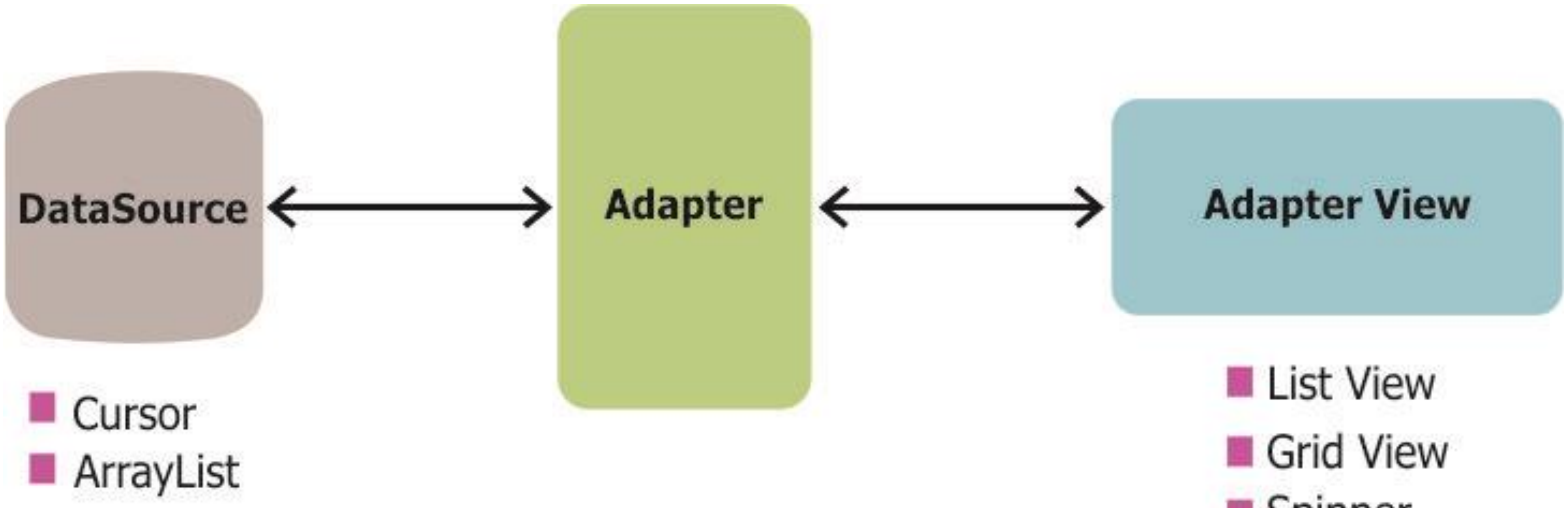
f) **CursorAdapter –** This adapter makes it easy and more controlled to access the binding of data values.

Adapters

- You can create your own Adapter class by extending the BaseAdapter class, which is the parent class for all other adapter class. Android SDK also provides some ready-to-use adapter classes, such as ArrayAdapter, SimpleAdapter etc.
- Make a ViewGroup to interact with data
- Some methods of adapters are:
  - isEmpty()
  - getItem(int position)
  - getCount()
  - getView()

AdapterView

- An Adapter View can be used to display large sets of data efficiently in form of List or Grid etc. provided to it by an Adapter. An Adapter View is capable of displaying millions of items on the User Interface, while keeping the memory and CPU usage very low.
- Adapter is only responsible for taking the data from a data source and converting it into View and then passing it to the AdapterView. Thus, it is used to manage the data. AdapterView is responsible for displaying the data.
- Some subclasses of adapter views are : ListView, GridView, Spinner, Gallery



- Cursor
- ArrayList

- List View
- Grid View
- Spinner

ListView

- ListView is used when you have to show items in a vertically scrolling list. E.g. Phone’s Contact List. With ListView, user can easily browse the information, while scrolling up and down. You can set divider between every item and set its height and color as per your UI design.
- Inside a ListView, we can show list of Text items by using TextView or pictures using ImageView, or any other view or a combination of views.
- As ListView is generally used to display a large set of data, hence it is not feasible to manually create list items for the complete data, hence Android provides us with special Adapter classes which can be used to supply data from datasets to ListView.

ListView example

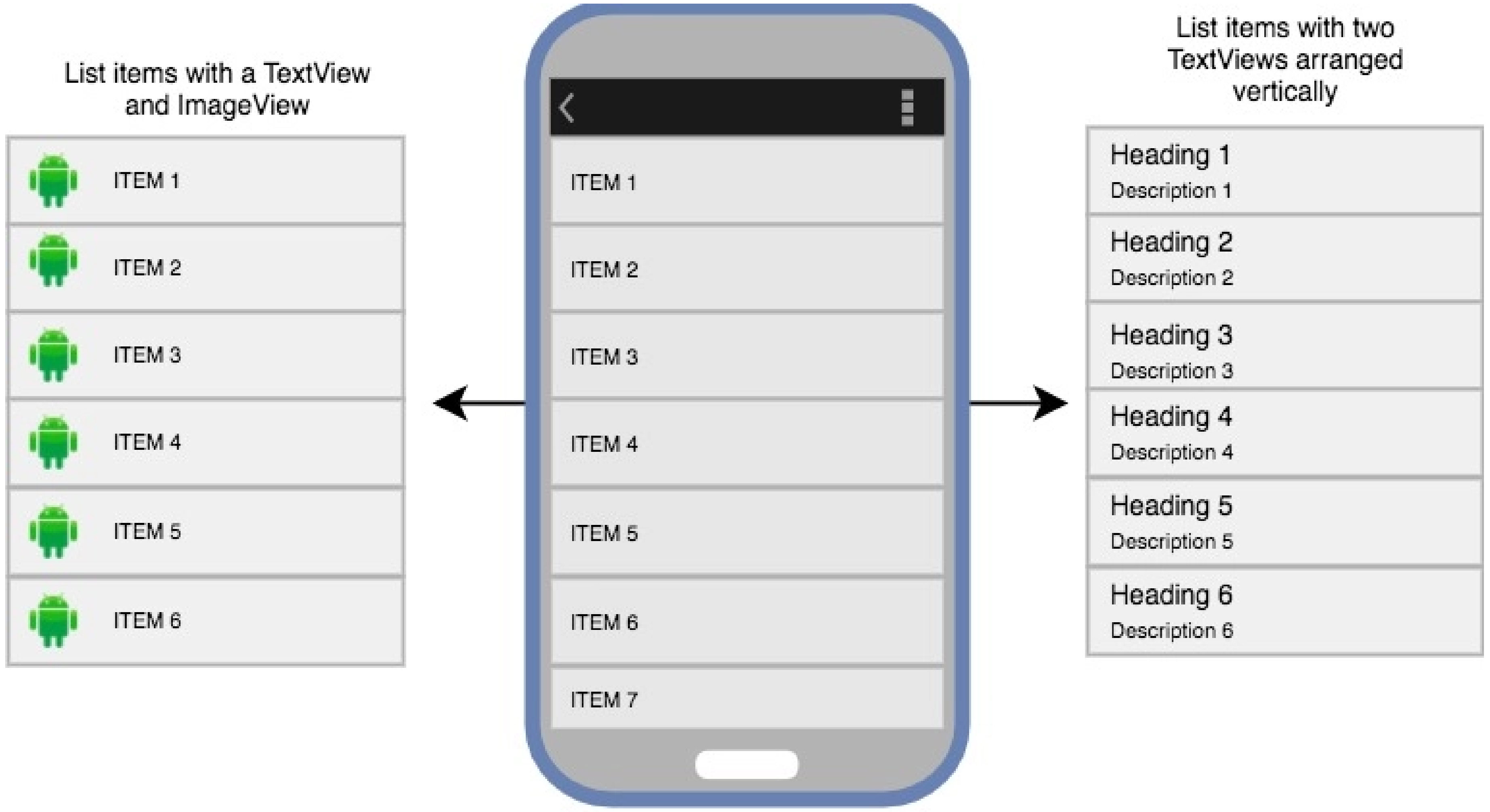
```
public class HelloAndroidActivity extends AppCompatActivity {


    @Override
    public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.list);

       String[] data = {"First", "Second", "Third"};
       ListView lv = (ListView)findViewById(R.id.list);
      lv.setAdapter(new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, data));
    }
}

<ListView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
   android:layout_height="match_parent"
    android:orientation="vertical"
    android:id="@+id/list" />
```

ListView



ListView in Android

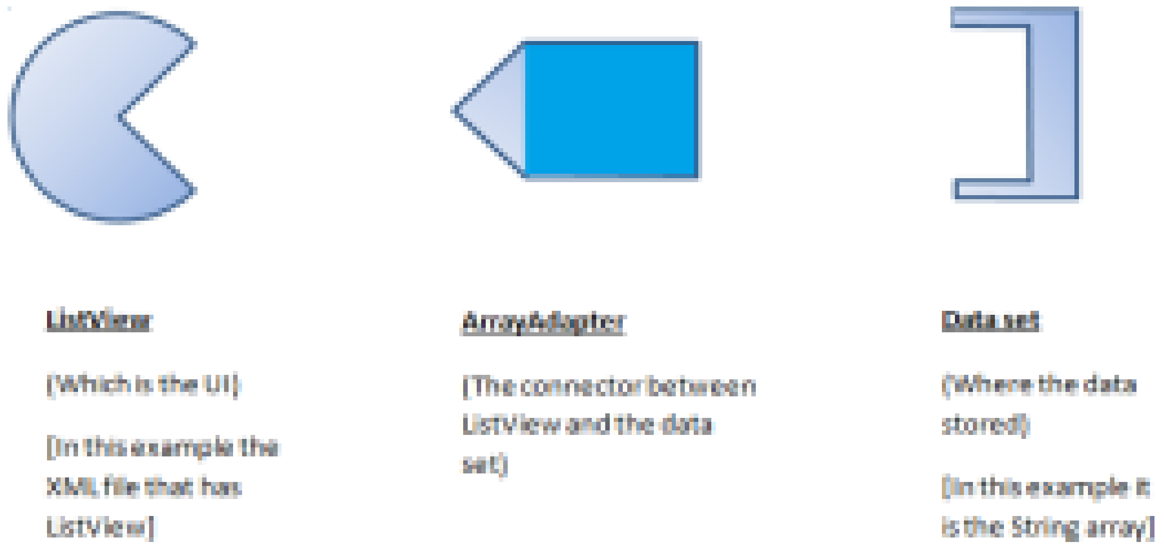
The List items can be anything, simple text, to a combination of an image and text etc.

Other views/adapters

- Spinner : Type of view that hold items in form of a dropdown menu available for user selection.
- GridView : GridView just works like ListView, the only difference is that GridView is used to display grid of View objects. The view objects can be a Text view, an Image view or a view group which has both an image and some text. Best example for this view is phone gallery which shows images in a grid. We can set number of columns to specific number and auto-fit the images into the columns.
- ExpandableListView : List with hidden values which can expand.
- TabWidget : Tabbed layouts.

ListView using Adapters

- An adapter extend the BaseAdapter class. Android provides some standard adapters; the most important are ArrayAdapter and CursorAdapter.
- ArrayAdapter can handle data based on Arrays or java.util.List.


ListView
(Which is the UI)
[In this example the XML file that has ListView]
ArrayAdapter
[The connector between ListView and the data set)
Data set
(Where the data stored)
[In this example it is the String array]

Working of ListView

- **Find out what data you want to display in list:**
  - For our example taking a static array of strings. But for complex scenarios it can be a response from server, or data fetched from database.
- **Declare the list view object in your xml layout:**
  - ListView is the user interface element we are using to represent data to user. So, the layout contains a list view. Make sure you provide an appropriate id.
- **Now finally, feed the list view with the data sets:**
  - For this we use adapters. We can customize our adapter. This can be done using Array adapter class available in android.

Android Lists

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_listviewexampleactivity);
       ListView listview = (ListView) findViewById(R.id.listview);
      String[] values = new String[] { "Android", "iPhone",WindowsMobile",
        "Blackberry“};
```

```
     ArrayList<String> list = new ArrayList<String>();
     for (int i = 0; i < values.length; ++i) {
       list.add(values[i]);
     }

      ArrayAdapter adp = new ArrayAdapter(this,
                                  android.R.layout.simple_list_item_1, list);
      listview.setAdapter(adapter);
```

Recycler View

- A RecyclerView is a container used to display a list or grid of data, like text or photos. When a list scrolls, only a handful of views are actually displayed on the screen. When a view scrolls off screen, RecyclerView reuses it and fills it with new data. This makes your app more efficient both in time and space, because it recycles existing structures instead of constantly creating new ones.
- Using a RecyclerView has the following key steps:

a) Define a model class to use as the data source

b) Add a RecyclerView to your activity to display the items

c) Create a custom row layout XML file to visualize the item

d) Create a RecyclerView.Adapter and ViewHolder to render the item

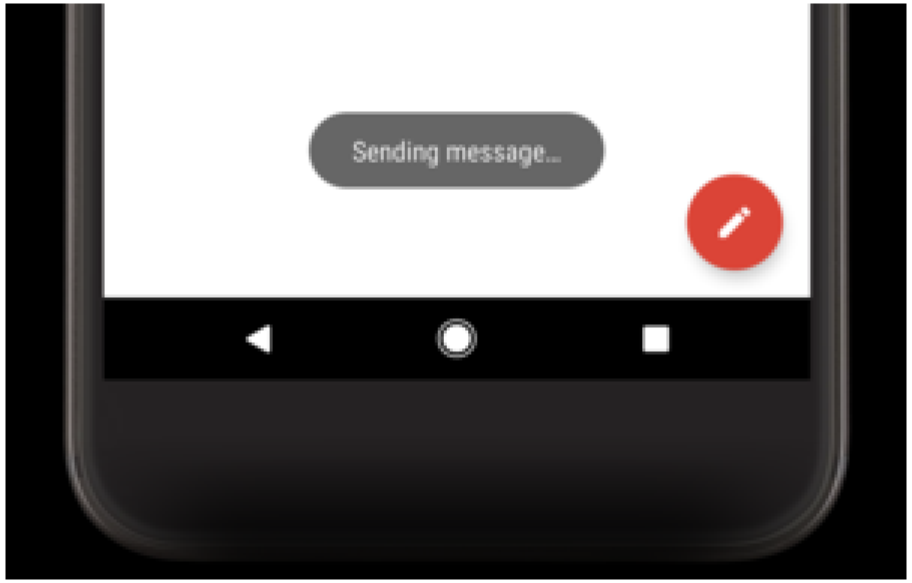
e) Bind the adapter to the data source to populate the RecyclerView

Recycler View

- RecyclerView uses the ViewHolder pattern which improves performance by allowing access to item views without frequent calls to findViewById().
- RecyclerView, uses LayoutManagers which support lists that can scroll vertically and horizontally, staggered lists, and grids. Custom LayoutManagers are also possible to create.
   a) LinearLayoutManager : shows items in a vertical or horizontal scrolling list.
   b) GridLayoutManager : shows items in a grid.
   c) StaggeredGridLayoutManager : shows items in a staggered grid.
- RecyclerView provides default item animations and a way to customize them.

Toast Notifications

- A toast notification is a message that pops up on the surface of the active view filling the amount of space required for the message.
- The current activity remains visible and interactive.
- The notification automatically fades in and out, and does not accept interaction events.
- A toast notification can be created and displayed from an Activity or Service.
- If you create a toast notification from a Service, it appears in front of the Activity currently in focus.



Toast Notifications

- First, instantiate a Toast object with one of the makeText() methods.
- This method takes three parameters: the application Context, the text message and the duration for the toast.
- It returns a properly initialized Toast object and using the show() method you can display it.

```
Context context = getApplicationContext();
String text = "Hello toast!";
int duration = Toast.LENGTH_SHORT;
Toast toast = Toast.makeText(context, text, duration);
toast.show();
```

- The same result can be obtained in a short and easy way using a single line command.

```
Toast.makeText(context, "Hello toast!", Toast.LENGTH_SHORT).show();
```

Toast Notifications

- A standard toast notification appears near the bottom of the screen, centered horizontally.
- You can change this position with the setGravity(int, int, int) method.
- This accepts three parameters: a Gravity constant, an x-position offset, and a y-position offset.

```
Toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

- For example, with the previous code you decide that the toast will appear in the top left corner.