Adapters and ListView

Himadri Parikh

Contents

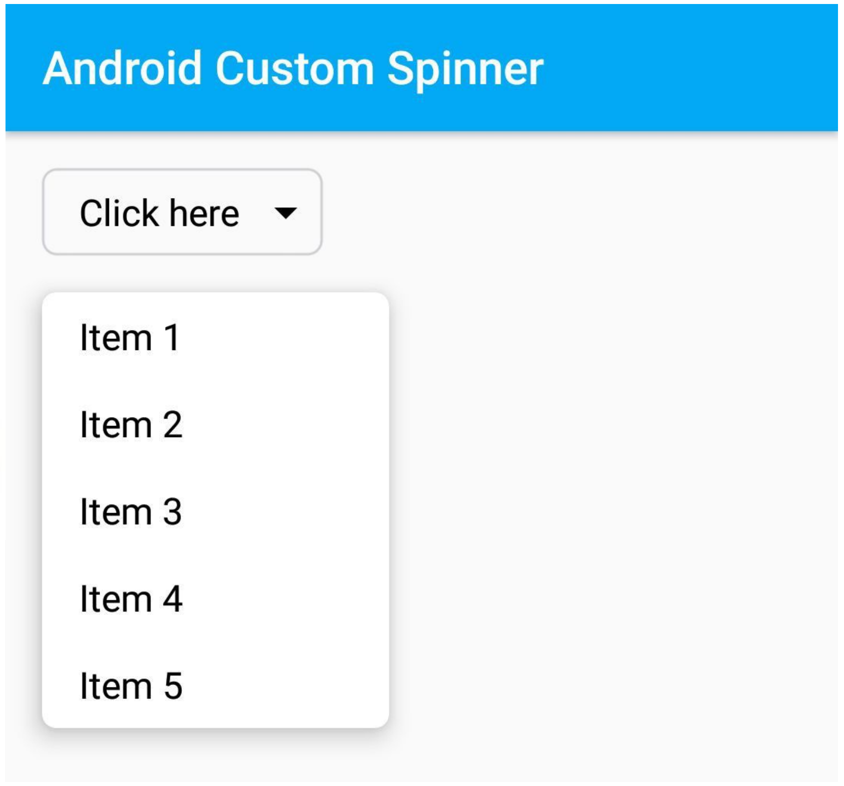
- Types of adapters
- Array adapter
- Base adapter
- List view
- Customized List view
- Gallery using Grid view

Adapters

Adapters

- An Adapter object acts as a bridge between an AdapterView and the underlying data for that view.
- The Adapter provides access to the data items.
- The Adapter is also responsible for making a View for each item in the data set.
- There are two types of adapters:
  - Array adapter
  - Base adapter

String ITEMS[] = new String[]{“Item 1”, “Item 2”, “Item 3”, “Item 4”, “Item 5”};



Array - adapter

- **ArrayAdapter** is a class which can work with array of data. You need to override only getview()method.

ArrayAdapter<String> adapter = new ArrayAdapter <> (getApplicationContext(), android.R.layout.*simple_list_item_1*, LIST);

- Here,
  - ArrayAdpater – is a generic class which takes String type of data
  - Adapter – object of ArrayAdapter
  - getApplicationContext() – the current context
  - android.R.layout.simple_list_item_1 – Built-in layout used to display single item of listview.
  - LIST – array from which listview will be populated.

BaseAdapter

- **BaseAdapter** as the name suggests, is a base class for all the adapters.
- When you are extending the Base adapter class you need to implement all the methods like getCount(), getId() etc.

```
class CustomAdapter extends BaseAdapter {
   @Override
   public int getCount() { return IMAGES.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) {
      return convertView;
   }
}
```

ListView

ListView

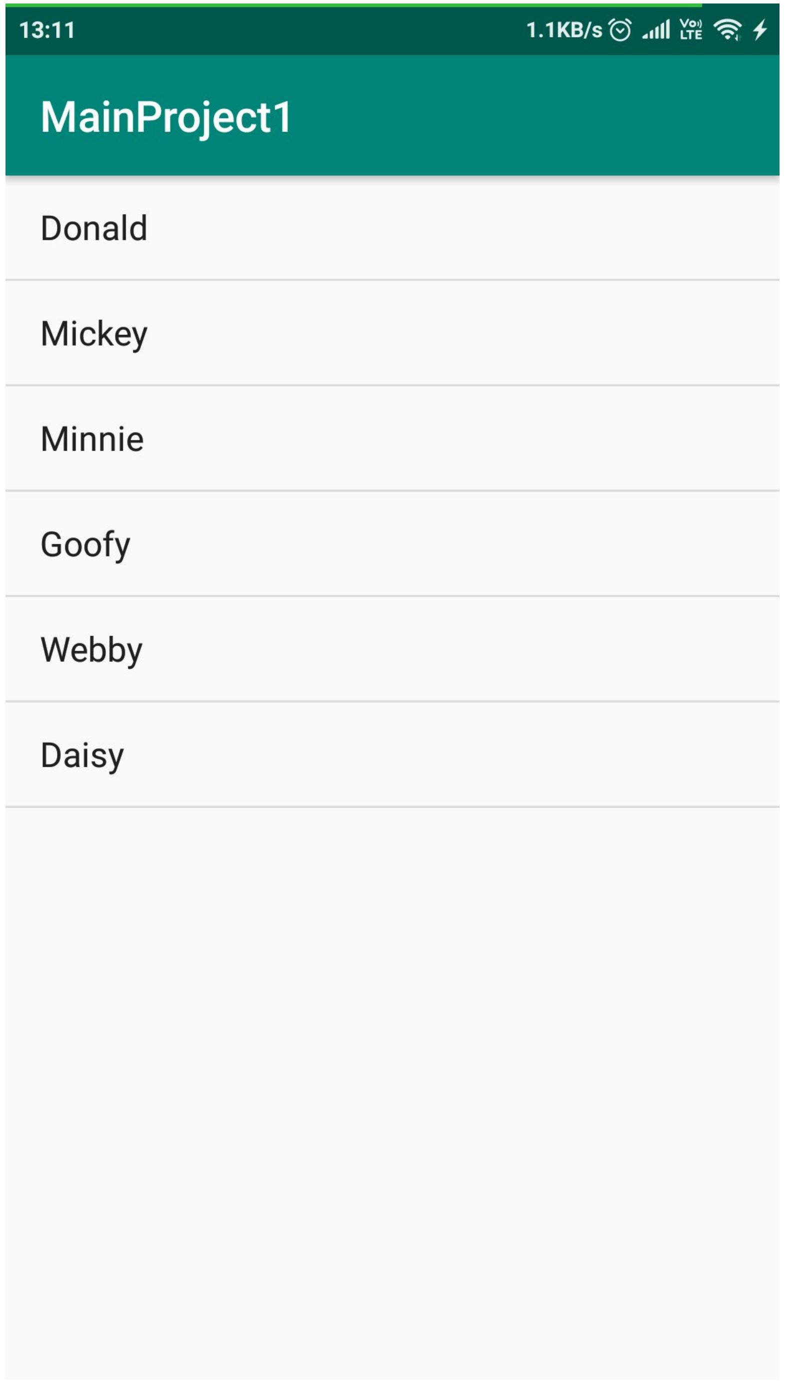
- ListView is a view group that displays a list of scrollable items.
- The list items are automatically inserted to the list
  - using an Adapter that pulls content from a source such as an array or database query and
  - converts each item result into a view that's placed into the list.
- Examples
- Types of ListView:
  - Simple ListView
  - Customised ListView

Coding for Simple ListView

- A ListView requires,
  - an Array - that contains all the items
  - ListView - in layout file
  - Array – adapter - for simple ListView
  - OnItemCLickListener - to perform action on item click

- Array:
  - String LIST[]={"Superman", "Spiderman", "Ironman", "Batman"};
- Bind view from XML file:
  - android.widget.ListView listView = findViewById(R.id.*mylist*);
- Define Array-adapter:
  - ArrayAdapter<String> adapter = new ArrayAdapter <> (getApplicationContext(), android.R.layout.*simple_list_item_1*, LIST);
- Link ListView with Array-adapter:
  - listView.setAdapter(adapter);
- Set OnItemClickListener:

  -
    ```
    listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
      Toast.makeText(getApplicationContext(),LIST[position], Toast.LENGTH_SHORT)
    .show()    }
    });
    ```



MainProject1

Donald

Mickey

Minnie

Goofy

Webby

Daisy

Custom ListView

- Custom ListView means modifying the display of listView as per your requirement.
- Examples

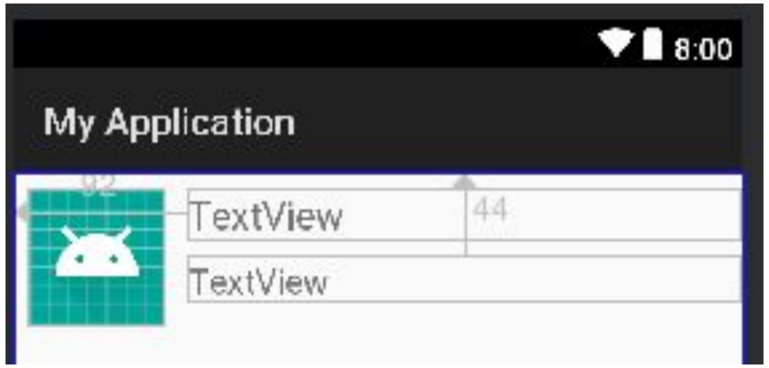
Coding for Custom ListView

- A Cuxtom ListView requires,
  - Arrays - that contains all the items(two or more arrays)
  - ListView - in layout file
  - Add images to the drawable folder
  - Single Item View - new layout for single item
  - Baseadapter - for Custom ListView
  - getView() - to populate the listView
  - OnItemCLickListener - to perform action on item click

- Arrays:
  - int IMAGES[] = {R.drawable.*donald*, R.drawable.*daisy*, R.drawable.*mickey*, R.drawable.*minney*, R.drawable.*goofy*};
  - String NAMES[] = {"Donald", "Daisy", "Mickey", "Minney", "Goofy"};
  - String DESCRIPTION[] = {"Duck", "Duck", "Mouse", "Mouse", "Dog"};
- Bind View from XML file:
  - ListView listView = findViewById(R.id.*mylistview*);
- Create a Java Class(External or Inner) that extends BaseAdapter and implements all method of it.:
  - class CustomAdapter extends BaseAdapter
- In getCount() method, you need to pass the length of array:

  -
    ```
    @Override
    public int getCount() {
        return IMAGES.length;
    }
    ```

- Design a new Layout file with layout(Relative/constraint) that contains the following :
  - An ImageView
  - TextView – for name
  - TextView – for Description



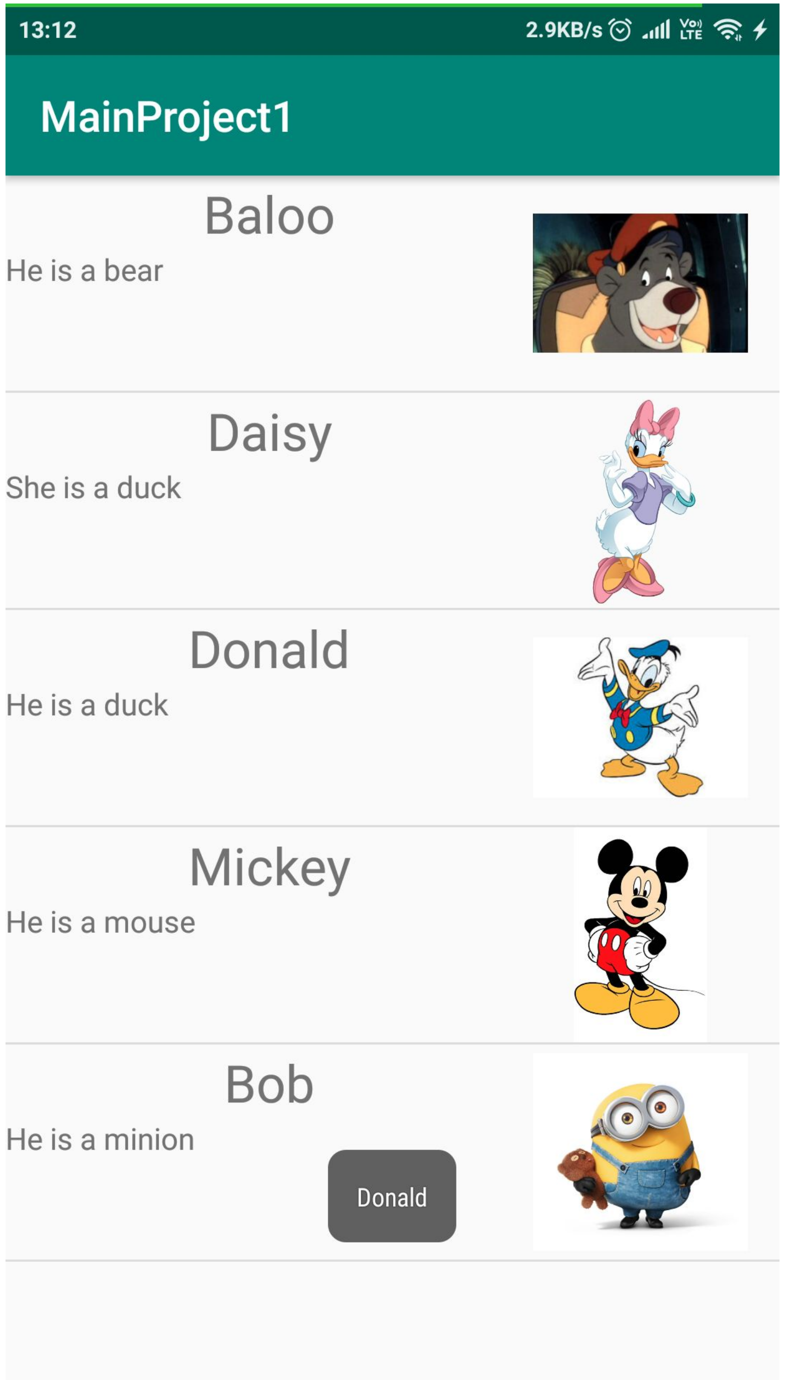
- getView() method is responsible to build the single item in ListView:

  -
    ```
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        convertView = getLayoutInflater().inflate(R.layout.single_view, null);

        ImageView imageView = convertView.findViewById(R.id.myimageView);
        TextView textView_name =
    convertView.findViewById(R.id.textview_Name);
        TextView textView_desc = convertView.findViewById(R.id.textView_Desc);

        imageView.setImageResource(IMAGES[position]);
        textView_name.setText(NAMES[position]);
        textView_desc.setText(DESCRIPTION[position]);

        return convertView;
    }
    ```

- Bind the adapter with ListView:

  -
    ```
    CustomAdapter customAdapter = new CustomAdapter();
    listView.setAdapter(customAdapter);
    ```
- Set OnItemClickListener:

  -
    ```
    listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position,
    long id) {
            Toast.makeText(getApplicationContext(), NAMES[position],
    Toast.LENGTH_SHORT).s
        }
    });
    ```


13:12 2.9KB/s

MainProject1

Baloo

He is a bear

Daisy

She is a duck

Donald

He is a duck

Mickey

He is a mouse

Bob

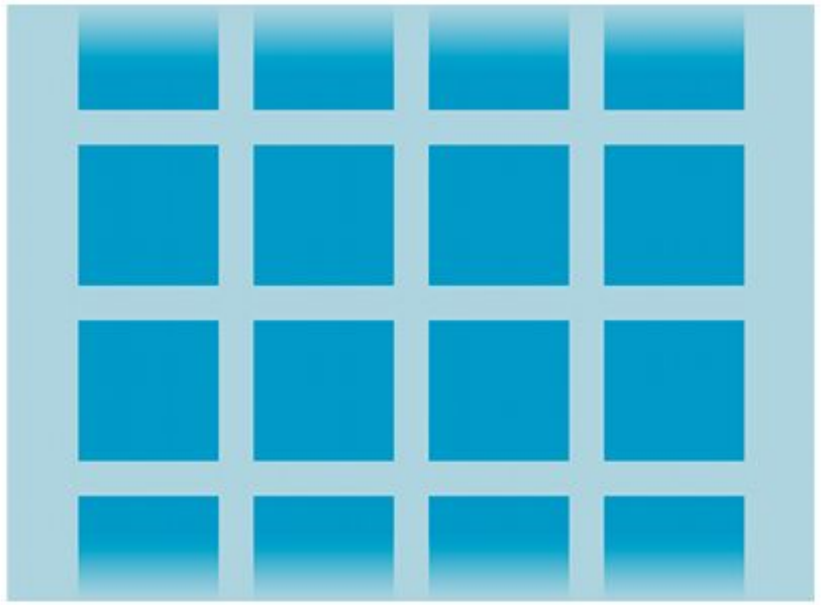
He is a minion

Donald

Gallery using GridView

GridView

- GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid. The grid items are automatically inserted to the layout using a ListAdapter.



Creating a Gallery using GridView

- To create a Gallery using GridView we require:
  - One Activity Class - that loads the GridView
  - Another Activity Class - that zooms the Image, user clicked on GridView
  - One Java Class that extends BaseAdapter and populates the Item in GridView

Coding for GridView

- A GridView requires,
  - Arrays - that contains all the images(in ImageAdapter Class)
  - GridView - in layout file
  - Add images to the drawable folder
  - Another Activity - that shows zoomed image
  - BaseAdapter - for Gallery/GridView
  - getView() - to populate the GridView
  - OnItemCLickListener - to zoom the image on item click

- Take one gridView in XML file and set:
  - android:stretchMode="columnWidth”
- Bind View from XML file:
  - GridView gridView = findViewById(R.id.*mygridview*);
- Create a java class that extends BaseAdapter:
  - public class ImageAdapter extends BaseAdapter
- Create an array in this class:
  - public int IMAGES[] = {R.drawable.*goofy*, R.drawable.*minney*, R.drawable.*mickey*, R.drawable.*daisy*, R.drawable.*donald*};
- Implement the constructor:
  - private Context context;
  - public ImageAdapter(Context context){   this.context = context; }

- Implement method of BaseAdapter class:
  - @Override
    public int getCount() { return IMAGES.length;   }
  - @Override
    public Object getItem(int position) {   return IMAGES[position]; }
  - @Override
    public long getItemId(int position) {   return 0;   }
  - @Override
    public View getView(int position, View convertView, ViewGroup parent) {

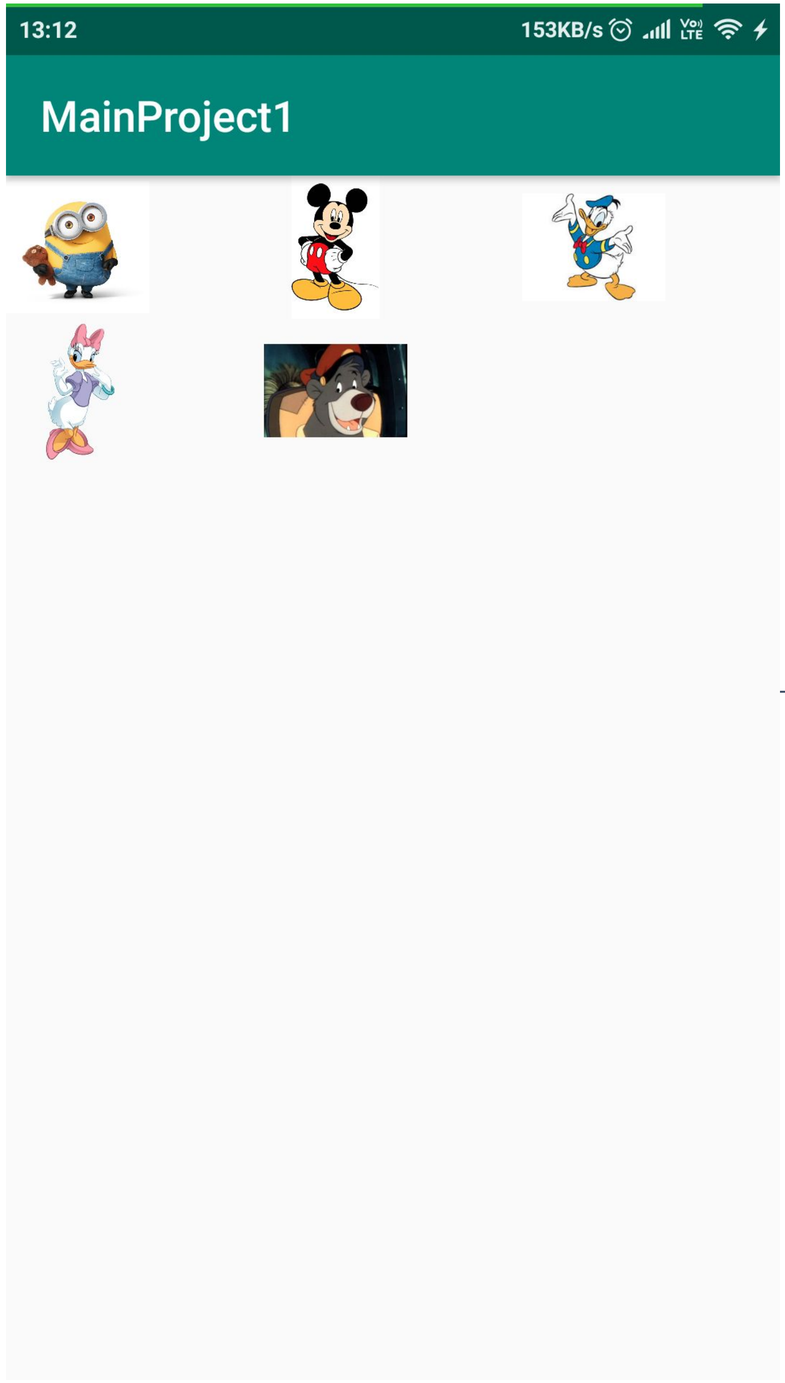
```
    ImageView imageView = new ImageView(context);
    imageView.setImageResource(IMAGES[position]);
    imageView.setScaleType(ImageView.ScaleType.CENTER_INSIDE);
    imageView.setLayoutParams( new GridView.LayoutParams(240, 240));

    return imageView ;
}
```

- Coming to GridViewActivity set Adapter:
  - gridView.setAdapter(new ImageAdapter(this));
- Set OnItemClickListener:

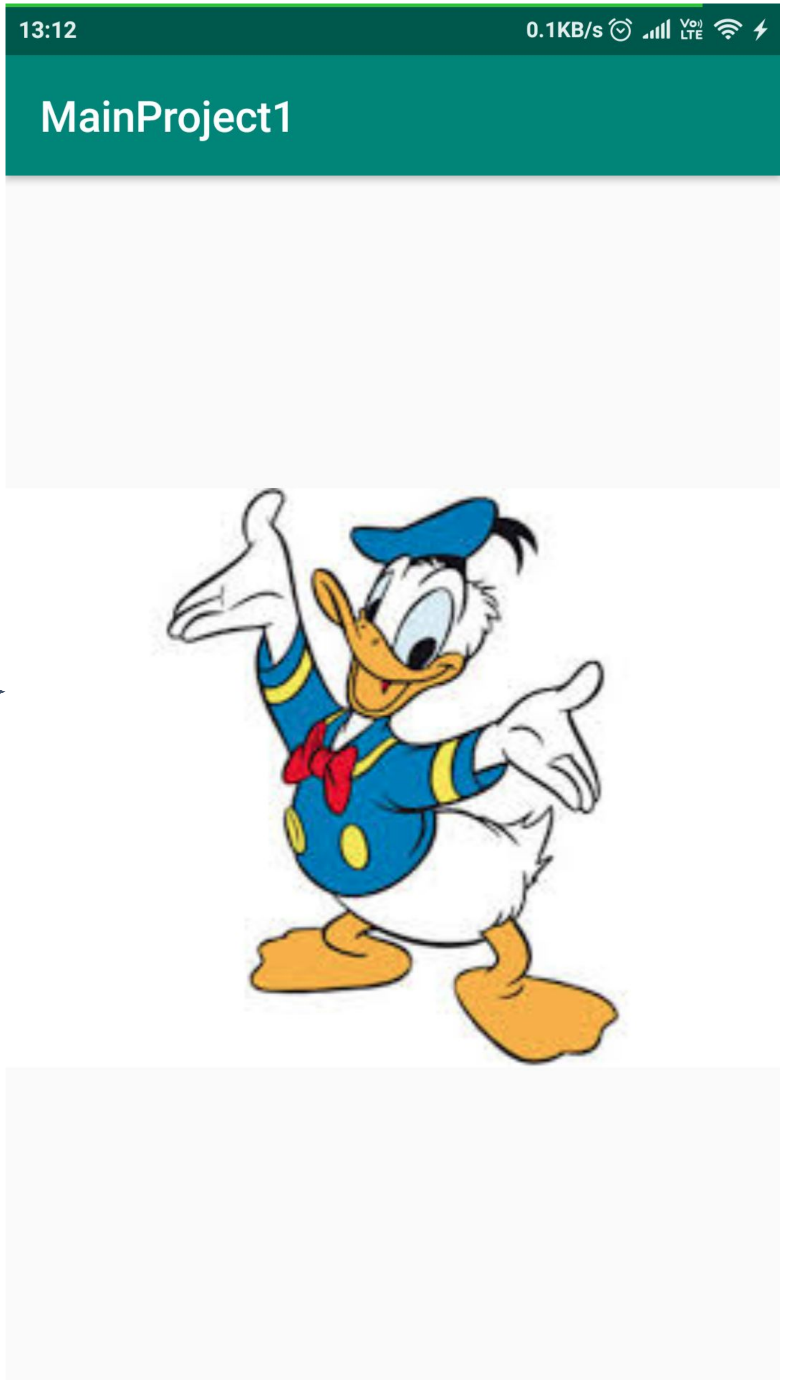
  -
    ```
    gridView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position,
    long id) {
            Intent intent = new Intent(getApplicationContext(), FullImage.class);
            intent.putExtra("id", position);

            startActivity(intent);
        }
    });
    ```

- Set ImageView in XML layout file of FullImage Activity in such a way that it covers the full layout:
- Coming to FullImage Class:

  -
    ```
    Intent intent = getIntent();
    int position  = intent.getExtras().getInt("id");

    ImageAdapter adapter = new ImageAdapter(this);

    ImageView imageView = findViewById(R.id.fullImage);
    imageView.setImageResource(adapter.IMAGES[position]);
    ```



$$\xrightarrow{\text{Intent}}$$



THANK YOU FOR LISTENING
ANY QUESTIONS??
makeameme.org