

UNIT 6

Listview, Gridview and Recyclerview

(6 Hours) — Mobile Programming (CAC351)

6.1 Adapter Views – Basic Idea

Lists and grids of items are core to mobile apps (contacts, messages, products). Instead of adding every item to the layout manually, Android uses an **Adapter** that pulls data from a data source and converts each item into a view. Only visible items are created, saving memory.

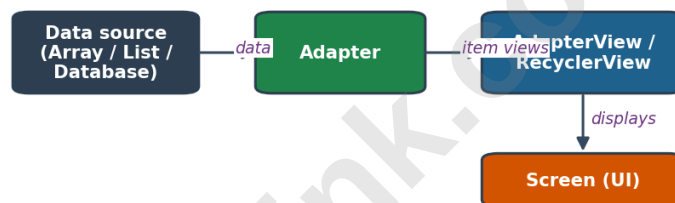


Figure 6.1: Data flows from the data source through an adapter to the list/grid

6.2 ListView

Introduction and Features

ListView displays a vertically scrolling list of items, one per row. It is an *AdapterView* and is the oldest list widget in Android; it has been superseded by RecyclerView for new work but is still part of the syllabus.

- Vertical scrolling list of selectable rows.
- Data supplied by an Adapter (ArrayAdapter, SimpleAdapter, CursorAdapter, BaseAdapter).
- Supports item click (*OnItemClickListener*), long click, dividers and header/footer views.
- Reuses row views with *convertView* (recycling) in custom adapters.
- Choice modes: none, single, multiple.

Implementing a ListView

```
<!-- activity_main.xml -->
<ListView android:id="@+id/listView"
    android:layout_width="match_parent" android:layout_height="match_parent" />
```

Listing 6.1: ListView in XML

```
ListView listView = findViewById(R.id.listView);
String[] cities = {"Kathmandu", "Pokhara", "Lalitpur", "Bhaktapur", "Biratnagar"};

ArrayAdapter<String> adapter = new ArrayAdapter<>(
    this, android.R.layout.simple_list_item_1, cities);
listView.setAdapter(adapter);

listView.setOnItemClickListener((parent, view, position, id) ->
    Toast.makeText(this, cities[position], Toast.LENGTH_SHORT).show());
```

Listing 6.2: ListView with ArrayAdapter

For rows with an image and multiple text fields, create a custom row layout and extend *BaseAdapter* (or *ArrayAdapter*) and override *getView()*.

```
public class StudentAdapter extends ArrayAdapter<Student> {
    public StudentAdapter(Context c, List<Student> list) { super(c, 0, list); }

    @NonNull @Override
    public View getView(int pos, View convertView, @NonNull ViewGroup parent) {
        if (convertView == null) // reuse if possible
            convertView = LayoutInflater.from(getContext())
                .inflate(R.layout.row_student, parent, false);
        Student s = getItem(pos);
        ((TextView) convertView.findViewById(R.id.tvName)).setText(s.getName());
        ((TextView) convertView.findViewById(R.id.tvRoll)).setText("Roll: " + s.getRoll());
        return convertView;
    }
}
```

Listing 6.3: Custom adapter with getView()

6.3 GridView

Introduction and Features

GridView shows items in a two-dimensional, scrollable grid of rows and columns, e.g., a photo gallery or app launcher.

- Number of columns set with *android:numColumns* (or *auto_fit*).
- Spacing controlled with *horizontalSpacing*, *verticalSpacing* and *columnWidth*.
- Same adapter model as *ListView* (*ArrayAdapter*, *BaseAdapter*).
- Supports *setOnItemClickListener()*; *stretchMode* defines how extra space is used.

```
<GridView android:id="@+id/gridView"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:numColumns="3" android:horizontalSpacing="8dp" android:verticalSpacing="8dp" />
```

Listing 6.4: GridView in XML

```

GridView grid = findViewById(R.id.gridView);
int[] images = { R.drawable.a, R.drawable.b, R.drawable.c, R.drawable.d };

grid.setAdapter(new BaseAdapter() {
    @Override public int getCount() { return images.length; }
    @Override public Object getItem(int p) { return images[p]; }
    @Override public long getItemId(int p) { return p; }
    @Override public View getView(int p, View v, ViewGroup parent) {
        ImageView iv = (v == null) ? new ImageView(MainActivity.this) : (ImageView) v;
        iv.setLayoutParams(new AbsListView.LayoutParams(300, 300));
        iv.setScaleType(ImageView.ScaleType.CENTER_CROP);
        iv.setImageResource(images[p]);
        return iv;
    }
});
grid.setOnItemClickListener((p, v, pos, id) ->
    Toast.makeText(this, "Item " + (pos + 1), Toast.LENGTH_SHORT).show());

```

Listing 6.5: GridView with a custom BaseAdapter

6.4 RecyclerView

Introduction and Features

RecyclerView (AndroidX) is the modern, flexible and efficient widget for large lists and grids. It **recycles** off-screen item views and rebinds them with new data, and it enforces the **ViewHolder** pattern.

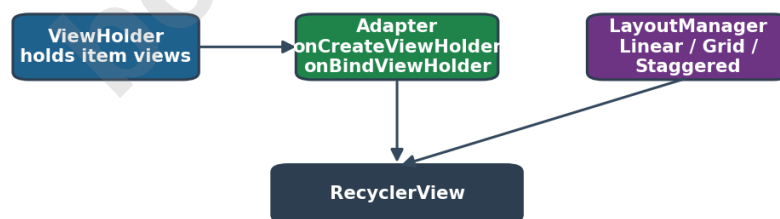


Figure 6.2: Main parts of a RecyclerView

Feature	Description
ViewHolder pattern (mandatory)	Caches references to item views; avoids repeated findViewById()
LayoutManager	LinearLayoutManager (list), GridLayoutManager (grid), StaggeredGridLayoutManager
Item animations	Built-in add/remove/move animations via ItemAnimator
Item decorations	Dividers and spacing with ItemDecoration

Feature	Description
Efficient updates	notifyItemInserted/Removed/Changed; DiffUtil / ListAdapter for smart diffs
Orientation	Vertical or horizontal scrolling

ListView vs GridView vs RecyclerView

Aspect	ListView	GridView	RecyclerView
Layout	Vertical list only	Grid	List, grid, staggered, custom
ViewHolder	Optional (manual)	Optional (manual)	Mandatory
Animations	No built-in	No built-in	Built-in ItemAnimator
Horizontal scroll	No	No	Yes
Performance	Fair	Fair	Best
Status	Legacy	Legacy	Recommended

Implementing a RecyclerView

Step 1 – Add dependency (build.gradle): *implementation 'androidx.recyclerview:recyclerview:1.3.2'*

Step 2 – Layouts: a RecyclerView in the activity and a row layout *item_student.xml*.

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

Listing 6.6: RecyclerView in activity_main.xml

Step 3 – Model class and adapter with a ViewHolder:

```

public class StudentAdapter extends RecyclerView.Adapter<StudentAdapter.VH> {
    private final List<Student> list;
    public StudentAdapter(List<Student> list) { this.list = list; }

    static class VH extends RecyclerView.ViewHolder {
        TextView tvName, tvRoll;
        VH(View v) {
            super(v);
            tvName = v.findViewById(R.id.tvName);
            tvRoll = v.findViewById(R.id.tvRoll);
        }
    }
    @NonNull @Override
    public VH onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View v = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.item_student, parent, false);
        return new VH(v); // called only when a new holder is needed
    }
    @Override
    public void onBindViewHolder(@NonNull VH h, int position) {
        Student s = list.get(position); // called to fill data
        h.tvName.setText(s.getName());
        h.tvRoll.setText("Roll: " + s.getRoll());
        h.itemView.setOnClickListener(v -> { /* handle click */ });
    }
    @Override public int getItemCount() { return list.size(); }
}

```

Listing 6.7: RecyclerView.Adapter and ViewHolder

Step 4 – Attach to RecyclerView in the activity:

```

RecyclerView rv = findViewById(R.id.recyclerView);
rv.setLayoutManager(new LinearLayoutManager(this)); // list
// rv.setLayoutManager(new GridLayoutManager(this, 2)); // grid with 2 columns
List<Student> students = new ArrayList<>();
students.add(new Student("Asha", 1));
students.add(new Student("Bikash", 2));
StudentAdapter adapter = new StudentAdapter(students);
rv.setAdapter(adapter);

// later, when data changes:
students.add(new Student("Rita", 3));
adapter.notifyItemInserted(students.size() - 1);

```

Listing 6.8: Setting up the RecyclerView

Example 1 — Adapter callbacks

A list has 1000 items but only 10 fit on screen. How many times is `onCreateViewHolder()` called?

Solution

Only about 10–15 times (visible rows plus a few cached). As the user scrolls, off-screen holders are *recycled* and `onBindViewHolder()` is called again with new data, so 1000 views are never created.

Note: Common mistakes: forgetting `setLayoutManager()` (nothing is displayed), inflating with `attachToRoot = true`, and calling `notifyDataSetChanged()` when a more specific notify method could be

used.

Key Takeaways

- ✓ Adapters connect data to list/grid views.
- ✓ ListView = vertical list, GridView = grid; both use ArrayAdapter/BaseAdapter and are legacy widgets.
- ✓ RecyclerView = recycles views, mandatory ViewHolder, pluggable LayoutManager, animations – the recommended choice.
- ✓ Adapter methods: onCreateViewHolder(), onBindViewHolder(), getItemCount().
- ✓ Update data with notifyItemInserted / Removed / Changed.

— *End of Unit 6* —

bcalink.com