**Section 41 – Advanced RecyclerView (CardView, Selections, Multiple View Types, Swipe)**

**1. Key Concepts Taught**

* Using **CardView** with RecyclerView for richer UI presentation.
* Creating **custom adapters** with RecyclerView.Adapter and custom ViewHolder.
* Structuring RecyclerView packages: **Model Class**, **Adapter Class**, **Activity**, and **Item Layout**.
* Adding **elevation, padding, background colors** for better card appearance.
* Implementing **swipe actions** with a 3rd-party library (SwipeRevealLayout + ViewBinderHelper).
* RecyclerView **item click handling** inside ViewHolder.
* Setting up **dummy/mock data** for demonstration.
* Differences between adapter creation **step order** (why instructor prefers creating ViewHolder first).
* **Single item selection** (missing in transcript → will add).
* **Multiple item selection** (missing in transcript → will add).
* **Multiple view types** in RecyclerView (missing in transcript → will add).

**2. Step-by-Step Implementation with Code + Detailed Comments**

**A. CardView + RecyclerView**

1. **Add CardView dependency** (if not already in build.gradle):
2. implementation "androidx.cardview:cardview:1.0.0"
3. **Create package & Activity**
   * Package: cardview
   * Activity: CardViewActivity
4. **Create Item Layout (item\_card.xml)**
5. <androidx.cardview.widget.CardView
6. xmlns:android="http://schemas.android.com/apk/res/android"
7. android:layout\_width="match\_parent"
8. android:layout\_height="wrap\_content"
9. android:layout\_margin="8dp"
10. android:elevation="4dp"
11. android:background="@color/black">
12. <LinearLayout
13. android:layout\_width="match\_parent"
14. android:layout\_height="wrap\_content"
15. android:orientation="vertical"
16. android:padding="8dp">
17. <!-- Four TextViews for planet details -->
18. <TextView android:id="@+id/textNameCard" .../>
19. <TextView android:id="@+id/textGravityCard" .../>
20. <TextView android:id="@+id/textDiameterCard" .../>
21. <TextView android:id="@+id/textOtherInfoCard" .../>
22. </LinearLayout>
23. </androidx.cardview.widget.CardView>
24. **Create Model Class (PlanetCard.java)**
25. public class PlanetCard {
26. private String name;
27. private String gravity;
28. private String diameter;
29. private String otherInfo;
30. public PlanetCard(String name, String gravity, String diameter, String otherInfo) {
31. this.name = name;
32. this.gravity = gravity;
33. this.diameter = diameter;
34. this.otherInfo = otherInfo;
35. }
36. // Getters
37. public String getName() { return name; }
38. public String getGravity() { return gravity; }
39. public String getDiameter() { return diameter; }
40. public String getOtherInfo() { return otherInfo; }
41. }
42. **Create Adapter & ViewHolder (CardAdapter.java)**
43. public class CardAdapter extends RecyclerView.Adapter<CardAdapter.CardViewHolder> {
44. private Context context;
45. private List<PlanetCard> planetList;
46. public CardAdapter(Context context, List<PlanetCard> planetList) {
47. this.context = context;
48. this.planetList = planetList;
49. }
50. // ViewHolder for CardView
51. static class CardViewHolder extends RecyclerView.ViewHolder {
52. TextView name, gravity, diameter, otherInfo;
53. public CardViewHolder(@NonNull View itemView) {
54. super(itemView);
55. name = itemView.findViewById(R.id.textNameCard);
56. gravity = itemView.findViewById(R.id.textGravityCard);
57. diameter = itemView.findViewById(R.id.textDiameterCard);
58. otherInfo = itemView.findViewById(R.id.textOtherInfoCard);
59. }
60. void bind(PlanetCard planet) {
61. name.setText(planet.getName());
62. gravity.setText(planet.getGravity());
63. diameter.setText(planet.getDiameter());
64. otherInfo.setText(planet.getOtherInfo());
65. }
66. }
67. @NonNull
68. @Override
69. public CardViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
70. View view = LayoutInflater.from(context).inflate(R.layout.item\_card, parent, false);
71. return new CardViewHolder(view);
72. }
73. @Override
74. public void onBindViewHolder(@NonNull CardViewHolder holder, int position) {
75. holder.bind(planetList.get(position));
76. }
77. @Override
78. public int getItemCount() {
79. return planetList.size();
80. }
81. }
82. **Set up RecyclerView in Activity**
83. public class CardViewActivity extends AppCompatActivity {
84. RecyclerView recyclerView;
85. CardAdapter adapter;
86. List<PlanetCard> planetList;
87. @Override
88. protected void onCreate(Bundle savedInstanceState) {
89. super.onCreate(savedInstanceState);
90. setContentView(R.layout.activity\_cardview);
91. recyclerView = findViewById(R.id.recyclerViewCards);
92. recyclerView.setLayoutManager(new LinearLayoutManager(this));
93. planetList = new ArrayList<>();
94. planetList.add(new PlanetCard("Earth", "9.8 m/s²", "12742 km", "Has life"));
95. planetList.add(new PlanetCard("Mars", "3.7 m/s²", "6779 km", "Red Planet"));
96. adapter = new CardAdapter(this, planetList);
97. recyclerView.setAdapter(adapter);
98. }
99. }

**B. Swipe Actions with SwipeRevealLayout**

1. **Add dependency in build.gradle**
2. implementation 'com.chauthai.swipereveallayout:swipe-reveal-layout:1.4.1'
3. **Item Layout with SwipeReveal**
4. <com.chauthai.swipereveallayout.SwipeRevealLayout
5. android:id="@+id/swipe\_layout"
6. android:layout\_width="match\_parent"
7. android:layout\_height="wrap\_content"
8. app:dragEdge="right">
9. <!-- Hidden Layout -->
10. <LinearLayout android:orientation="horizontal">
11. <TextView android:id="@+id/textEdit" android:text="Edit" .../>
12. <TextView android:id="@+id/textDelete" android:text="Delete" .../>
13. </LinearLayout>
14. <!-- Main Layout -->
15. <TextView android:id="@+id/textName" .../>
16. </com.chauthai.swipereveallayout.SwipeRevealLayout>
17. **Adapter with ViewBinderHelper**
18. public class SwipeAdapter extends RecyclerView.Adapter<SwipeAdapter.SwipeViewHolder> {
19. private List<Employee> employees;
20. private Context context;
21. private ViewBinderHelper binderHelper = new ViewBinderHelper();
22. public SwipeAdapter(Context context, List<Employee> employees) {
23. this.context = context;
24. this.employees = employees;
25. binderHelper.setOpenOnlyOne(true);
26. }
27. class SwipeViewHolder extends RecyclerView.ViewHolder {
28. SwipeRevealLayout swipeLayout;
29. TextView textName, textEdit, textDelete;
30. public SwipeViewHolder(@NonNull View itemView) {
31. super(itemView);
32. swipeLayout = itemView.findViewById(R.id.swipe\_layout);
33. textName = itemView.findViewById(R.id.textName);
34. textEdit = itemView.findViewById(R.id.textEdit);
35. textDelete = itemView.findViewById(R.id.textDelete);
36. textEdit.setOnClickListener(v ->
37. Toast.makeText(context, "Edit clicked", Toast.LENGTH\_SHORT).show()
38. );
39. textDelete.setOnClickListener(v ->
40. Toast.makeText(context, "Delete clicked", Toast.LENGTH\_SHORT).show()
41. );
42. }
43. }
44. @NonNull
45. @Override
46. public SwipeViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
47. View view = LayoutInflater.from(context).inflate(R.layout.item\_employee\_swipe, parent, false);
48. return new SwipeViewHolder(view);
49. }
50. @Override
51. public void onBindViewHolder(@NonNull SwipeViewHolder holder, int position) {
52. Employee emp = employees.get(position);
53. binderHelper.bind(holder.swipeLayout, String.valueOf(emp.getName()));
54. holder.textName.setText(emp.getName());
55. }
56. @Override
57. public int getItemCount() {
58. return employees.size();
59. }
60. }

**C. (Added) Single Item Selection**

* Maintain a selectedPosition in adapter.
* Highlight selected item in onBindViewHolder.
* On item click → update selectedPosition & call notifyDataSetChanged().

**D. (Added) Multiple Selection**

* Maintain a Set<Integer> of selected positions.
* On click → toggle position in the set.
* Use different background for selected items.

**E. (Added) Multiple View Types**

* Override getItemViewType(position) to return type constants.
* Inflate different layouts in onCreateViewHolder.

**3. Tools, Libraries, APIs Used**

* RecyclerView (androidx.recyclerview.widget.RecyclerView)
* CardView (androidx.cardview.widget.CardView)
* SwipeRevealLayout (3rd-party library)
* ViewBinderHelper (comes with SwipeRevealLayout)
* LinearLayoutManager for list layout

**4. Best Practices & Modern Alternatives**

* Use **ViewBinding** or **DataBinding** instead of findViewById.
* Use **ListAdapter + DiffUtil** instead of manual notifyDataSetChanged() for performance.
* For swipe, prefer **ItemTouchHelper** (built-in) if you don’t need advanced hidden layouts.
* Use **sealed classes** in Kotlin for multiple view types.

**Part B – Extra Important Things Not Covered**

* **DiffUtil**: Efficient RecyclerView updates.
* **Paging 3** for large datasets.
* **ItemAnimator** customization for better transitions.
* **Stable IDs** in RecyclerView to preserve state on updates.
* Accessibility improvements for swipe actions.
* Using **MaterialCardView** from Material Components for richer UI features.