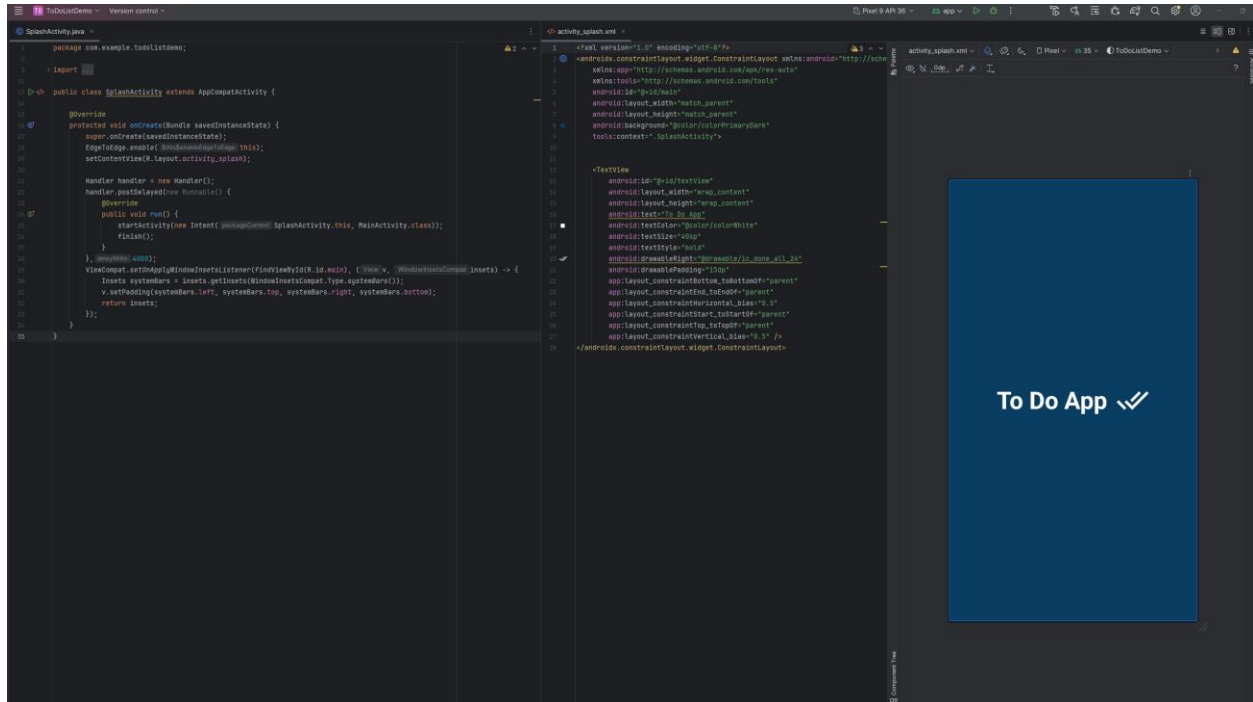


Raina Narayan

Screenshots of code:

Splash Activity



Add New Task Activity

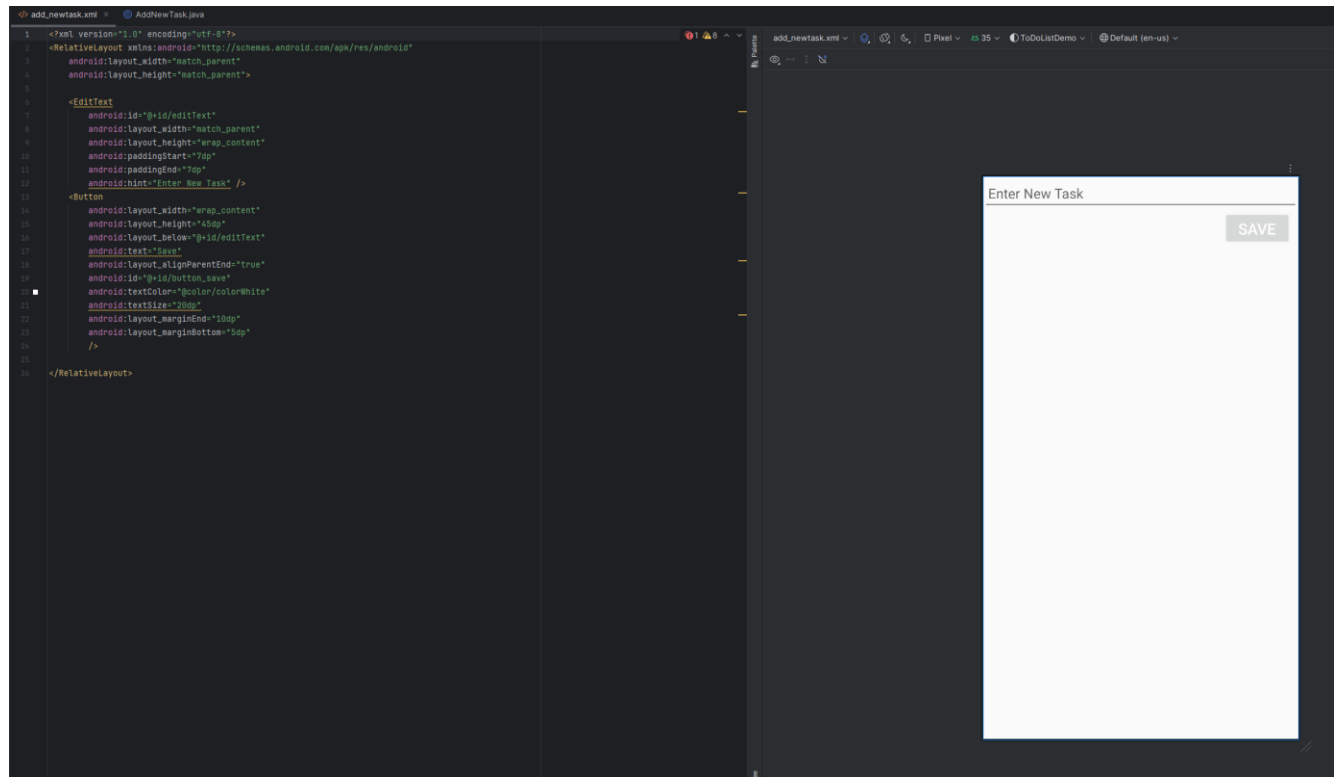
```
</> add_newtask.xml  AddNewTask.java  x
1  package com.example.todolistdemo;
2
3  > import ...
21
22 </> public class AddNewTask extends BottomSheetDialogFragment {
23
24     1 usage
25     public static final String TAG = "AddNewTask";
26
27     //widgets
28     4 usages
29     private EditText mEditText;
30     7 usages
31     private Button mSaveButton;
32     3 usages
33     private DataBaseHelper myDB;
34
35     1 usage
36     public static AddNewTask newInstance() { return new AddNewTask(); }
37
38     @Nullable
39     @Override
40     public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
41         View v = inflater.inflate(R.layout.add_newtask, container, attachToRoot: false);
42         return v;
43     }
44
45     @Override
46     public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
47         super.onViewCreated(view, savedInstanceState);
48
49         mEditText = view.findViewById(R.id.editText);
50         mSaveButton = view.findViewById(R.id.button_save);
51
52         myDB = new DataBaseHelper(getActivity());
53
54         boolean isUpdate = false;
55         Bundle bundle = getArguments();
56         if(bundle != null){
57             isUpdate = true;
58             String task = bundle.getString(key: "task");
59             mEditText.setText(task);
60
61             if(task.length() > 0 ){
62                 mSaveButton.setEnabled(false);
63             }
64         }
65     }
66 }
```

</> add_newtask.xml

AddNewTask.java

```
22 public class AddNewTask extends BottomSheetDialogFragment {
43     public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
61     }
62
63     mEditText.addTextChangedListener(new TextWatcher() {
64         @Override
65         public void beforeTextChanged(CharSequence s, int start, int count, int after) {
66
67         }
68
69         @Override
70         public void onTextChanged(CharSequence s, int start, int before, int count) {
71             if(s.toString().equals("")){
72                 mSaveButton.setEnabled(false);
73                 mSaveButton.setBackgroundColor(Color.GRAY);
74             } else{
75                 mSaveButton.setEnabled(true);
76                 mSaveButton.setBackgroundColor(getResources().getColor(R.color.colorPrimary));
77             }
78         }
79
80         @Override
81         public void afterTextChanged(Editable s) {
82
83         }
84     });
85
86     boolean finalIsUpdate = isUpdate;
87     mSaveButton.setOnClickListener(new View.OnClickListener() {
88         @Override
89         public void onClick(View v) {
90             String text = mEditText.getText().toString();
91             if(finalIsUpdate){
92                 myDB.updateTask(bundle.getInt(key: "id"),text);
93             } else {
94                 ToDoModel item = new ToDoModel();
95                 item.setTask(text);
96                 item.setStatus(0);
97                 myDB.insertTask(item);
98             }
99             dismiss();
100         }
101     });
102 }
103
104 @Override
105 public void onDismiss(@NonNull DialogInterface dialog) {
106     super.onDismiss(dialog);
107     Activity activity = getActivity();
108     if(activity instanceof OnDialogCloseListener){
109         ((OnDialogCloseListener) activity).onDialogClose(dialog);
110     }
111 }
112 }
113
```

Add New Task Layout



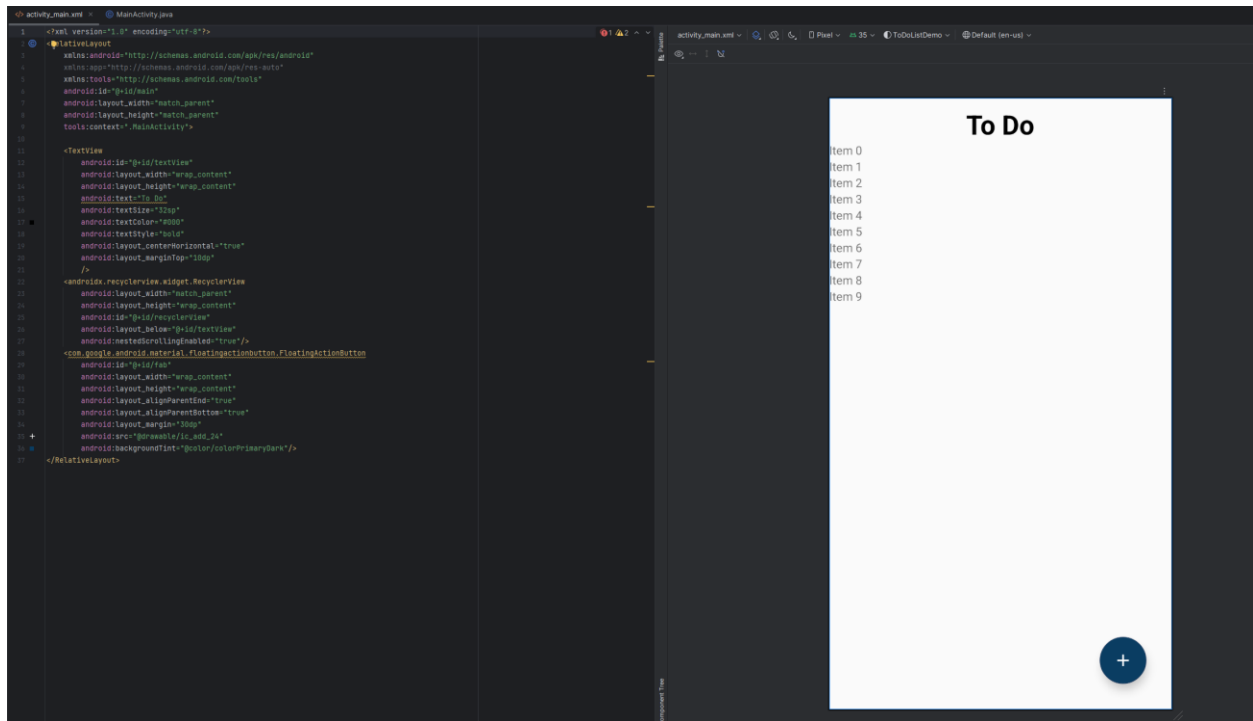
Main Activity

```

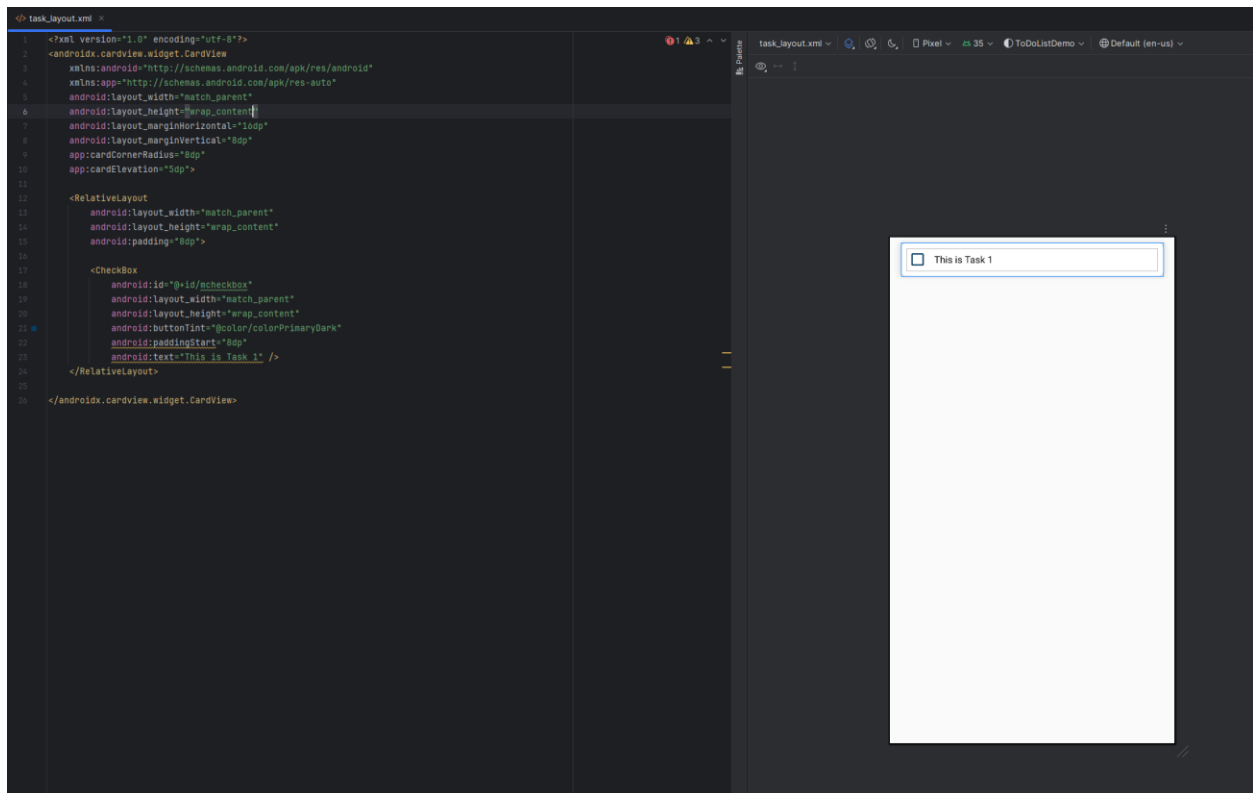
25
26 </> public class MainActivity extends AppCompatActivity implements OnDialogCloseListener{
27
28     5 usages
29     private RecyclerView mRecyclerView;
30     2 usages
31     private FloatingActionButton fab;
32     4 usages
33     private DataBaseHelper myDB;
34     7 usages
35     private List<ToDoModel> mList;
36     6 usages
37     private ToDoAdapter adapter;
38
39     @Override
40     protected void onCreate(Bundle savedInstanceState) {
41         super.onCreate(savedInstanceState);
42         EdgeToEdge.enable( $this$enableEdgeToEdge: this);
43         setContentView(R.layout.activity_main);
44
45         mRecyclerView = findViewById(R.id.recyclerView);
46         fab = findViewById(R.id.fab);
47         myDB = new DataBaseHelper( context: MainActivity.this);
48         mList = new ArrayList<>();
49         adapter = new ToDoAdapter(myDB, activity: MainActivity.this);
50
51         mRecyclerView.setHasFixedSize(true);
52         mRecyclerView.setLayoutManager(new LinearLayoutManager( context: this));
53         mRecyclerView.setAdapter(adapter);
54
55         mList = myDB.getAllTasks();
56         Collections.reverse(mList);
57         adapter.setTasks(mList);
58
59         fab.setOnClickListener(new View.OnClickListener() {
60             @Override
61             public void onClick(View v) {
62                 AddNewTask.newInstance().show(getSupportFragmentManager(), AddNewTask.TAG);
63             }
64         });
65
66         ItemTouchHelper itemTouchHelper = new ItemTouchHelper(new RecyclerViewTouchHelper(adapter));
67         itemTouchHelper.attachToRecyclerView(mRecyclerView);
68
69         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), ( View v, WindowInsetsCompat insets) -> {
70             Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
71             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
72             return insets;
73         });
74
75     1 usage
76     @Override
77     public void onDialogClose(DialogInterface dialogInterface) {
78         mList = myDB.getAllTasks();
79         Collections.reverse(mList);
80         adapter.setTasks(mList);
81         adapter.notifyDataSetChanged();
82     }

```

Main Layout



Task Layout



Recycler View Touch Helper

```
1 package com.example.todolistdemo;
2 import androidx.recyclerview.widget.ItemTouchHelper;
3
4 public class RecyclerViewTouchHelper extends ItemTouchHelper.SimpleCallback {
5     private ToDoAdapter adapter;
6
7     public RecyclerViewTouchHelper(TodoAdapter adapter) {
8         super(0, swipeDirs: ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT);
9         this.adapter = adapter;
10    }
11
12    @Override
13    public boolean onMove(@NonNull RecyclerView recyclerView, @NonNull RecyclerView.ViewHolder viewHolder, @NonNull RecyclerView.ViewHolder target) {
14        return false;
15    }
16
17    @Override
18    public void onSwiped(@NonNull RecyclerView.ViewHolder viewHolder, int direction) {
19        final int position = viewHolder.getAdapterPosition();
20        if(direction == ItemTouchHelper.RIGHT){
21            // adapter.notifyItemChanged(position);
22            AlertDialog.Builder builder = new AlertDialog.Builder(adapter.getContext());
23            builder.setTitle("Delete Task");
24            builder.setMessage("Are you sure?");
25            builder.setPositiveButton(R.string.yes, new DialogInterface.OnClickListener() {
26                @Override
27                public void onClick(DialogInterface dialogInterface, int i) {
28                    adapter.notifyItemChanged(position);
29                    adapter.deleteTask(position);
30                    // Toast.makeText(adapter.getContext(), "deleting task", Toast.LENGTH_SHORT).show();
31                }
32            });
33            builder.setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {
34                @Override
35                public void onClick(DialogInterface dialogInterface, int i) {
36                    adapter.notifyItemChanged(position);
37                }
38            });
39            AlertDialog dialog = builder.create();
40            dialog.show();
41        } else {
42            adapter.editItem(position);
43        }
44    }
45
46    @Override
47    public void onChildDraw(@NonNull Canvas c, @NonNull RecyclerView recyclerView, @NonNull RecyclerView.ViewHolder viewHolder, float dx, float dy, int actionState, boolean isCurrentlyActive) {
48        new RecyclerViewSwipeDecorator.Builder(c, recyclerView, viewHolder, dx, dy, actionState, isCurrentlyActive)
49            .addSwipeLeftBackgroundColor(ContextCompat.getColor(adapter.getContext(), R.color.colorPrimaryDark))
50            .addSwipeLeftActionIcon(R.drawable.ic_edit_24)
51            .addSwipeRightBackgroundColor(Color.RED)
52            .addSwipeRightActionIcon(R.drawable.ic_delete_24)
53            .create()
54            .decorate();
55
56        super.onChildDraw(c, recyclerView, viewHolder, dx, dy, actionState, isCurrentlyActive);
57    }
58 }
```

On Dialog Close Listener

```
1 package com.example.todolistdemo;
2 import androidx.recyclerview.widget.ItemTouchHelper;
3
4 public interface OnDialogCloseListener {
5     void onDialogClose(DialogInterface dialogInterface);
6 }
```

Database Helper

```
@ DataBaseHelper.java
1 package com.example.todolistdemo.Utils;
2 import androidx
3     @ usages
14 public class DataBaseHelper extends SQLiteOpenHelper {
15     private SQLiteDatabase db;
16     private static final String DATABASE_NAME = "TODO_DATABASE";
17     private static final String TABLE_NAME = "TODO_TABLE";
18     private static final String COL_1 = "ID";
19     private static final String COL_2 = "TASK";
20     private static final String COL_3 = "STATUS";
21     public DataBaseHelper(@Nullable Context context) {
22         super(context, DATABASE_NAME, factory: null, version: 1); //error says this is null
23     }
24     @Override
25     public void onCreate(SQLiteDatabase db) {
26         db.execSQL("CREATE TABLE IF NOT EXISTS " + TABLE_NAME + "(ID INTEGER PRIMARY KEY AUTOINCREMENT, TASK TEXT, STATUS INTEGER)");
27     }
28     @Override
29     public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
30         //INSERT METHOD
31     }
32     public void insertTask(ToDoModel model){
33         db = this.getWritableDatabase();
34         ContentValues values = new ContentValues();
35         values.put(COL_2, model.getTask());
36         values.put(COL_3, 0);
37         db.insert(TABLE_NAME, nullColumnHack: null, values);
38     }
39     //UPDATE METHOD
40     public void updateTask(int id, String task) {
41         db = this.getWritableDatabase();
42         ContentValues values = new ContentValues();
43         values.put(COL_2, task);
44         db.update(TABLE_NAME, values, whereClause: "ID=?", new String[]{String.valueOf(id)});
45     }
46     //UPDATE STATUS METHOD
47     public void updateStatus(int id, int status){
48         db = this.getWritableDatabase();
49         ContentValues values = new ContentValues();
50         values.put(COL_3, status);
51         db.update(TABLE_NAME, values, whereClause: "ID=?", new String[]{String.valueOf(id)});
52     }
53     //Delete TASK METHOD
54     public void deleteTask(int id){
55         db = this.getWritableDatabase();
56         db.delete(TABLE_NAME, whereClause: "ID=?", new String[]{String.valueOf(id)});
57     }
58 }
59 }
```

```
//GET ALL TASK METHOD
2 usages
public List<ToDoModel> getAllTasks(){
    db = this.getWritableDatabase();
    Cursor cursor = null;
    List<ToDoModel> modelList = new ArrayList<>();

    db.beginTransaction();
    try{
        cursor = db.query(TABLE_NAME, columns: null, selection: null, selectionArgs: null, groupBy: null, having: null, orderBy: null);
        if (cursor !=null){
            if (cursor.moveToFirst()){
                do {
                    ToDoModel task = new ToDoModel();
                    task.setId(cursor.getInt(cursor.getColumnIndexOrThrow(COL_1)));
                    task.setTask(cursor.getString(cursor.getColumnIndexOrThrow(COL_2)));
                    task.setStatus(cursor.getInt(cursor.getColumnIndexOrThrow(COL_3)));
                    modelList.add(task);
                } while (cursor.moveToNext());
            }
        }
    } finally {
        db.endTransaction();
        cursor.close();
    }
    return modelList;
}
}
```


To Do Model

```
ToDoModel.java x
1 package com.example.todolistdemo.Model;
2
3 17 usages
4 public class ToDoModel {
5     2 usages
6     private String task;
7     2 usages
8     private int id, status;
9
10    public int getId() {
11        return id;
12    }
13
14    public void setId(int id) {
15        this.id = id;
16    }
17
18    1 usage
19    public int getStatus() {
20        return status;
21    }
22
23    2 usages
24    public void setStatus(int status) {
25        this.status = status;
26    }
27
28    3 usages
29    public String getTask() {
30        return task;
31    }
32
33    2 usages
34    public void setTask(String task) {
35        this.task = task;
36    }
37 }
```

To Do Adapter

```
© ToDoAdapter.java x
1 package com.example.todolistdemo.Adapter;
2 > import ...
7 usages
20 public class ToDoAdapter extends RecyclerView.Adapter<ToDoAdapter.MyViewHolder> {
6 usages
21     private List<ToDoModel> mList;
3 usages
22     private MainActivity activity;
4 usages
23     private DataBaseHelper myDB;
1 usage
24     public ToDoAdapter (DataBaseHelper myDB, MainActivity activity){
25         this.activity = activity;
26         this.myDB = myDB;
27     }
28     @NonNull
29     @Override
30     public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
31         View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.task_layout, parent, attachToRoot: false);
32         return new MyViewHolder(v);
33     }
34     @Override
35     public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
36         final ToDoModel item = mList.get(position);
37         holder.mCheckBox.setText(item.getTask());
38         holder.mCheckBox.setChecked(toBoolean(item.getStatus()));
39         holder.mCheckBox.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
40             @Override
41             public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
42                 if(isChecked){
43                     myDB.updateStatus(item.getId(), status: 1);
44                 } else{
45                     myDB.updateStatus(item.getId(), status: 0);
46                 }
47             }
48         });
49     }
1 usage
50     public boolean toBoolean(int num){
51         return num !=0;
52     }
53     //METHOD TO RETURN CONTEXT
2 usages
54     public Context getContext(){ return activity; }
57
2 usages
58     public void setTasks(List<ToDoModel>mList){
59         this.mList = mList;
60         notifyDataSetChanged();
61     }
1 usage
62     public void deleteTask(int position){
63         ToDoModel item = mList.get(position);
64         myDB.deleteTask(item.getId());
65         mList.remove(position);
66         notifyItemRemoved(position);
67     }
```

1 usage

```
public void editItem(int position){
    ToDoModel item = mList.get(position);

    Bundle bundle = new Bundle();
    bundle.putInt("id",item.getId());
    bundle.putString("task", item.getTask());

    AddNewTask task = new AddNewTask();
    task.setArguments(bundle);
    task.show(activity.getSupportFragmentManager(),task.getTag());
}

@Override
public int getItemCount() {
    return mList.size();
}
```

4 usages

```
public static class MyViewHolder extends RecyclerView.ViewHolder{
    4 usages
    CheckBox mCheckBox;
    1 usage
    public MyViewHolder(@NonNull View itemView) {
        super(itemView);
        mCheckBox = itemView.findViewById(R.id.mcheckbox);
    }
}

}
```

Colors

```
</> colors.xml ×
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <color name="black">#FF000000</color>
4
5
6      <color name="colorWhite">#FFFFFF</color>
7      <color name="colorPrimary">#192A56</color>
8      <color name="colorPrimaryDark">#0A3D62</color>
9      <color name="colorAccent">#192A56</color>
10
11 </resources>
```

Build Gradle (Model: app)

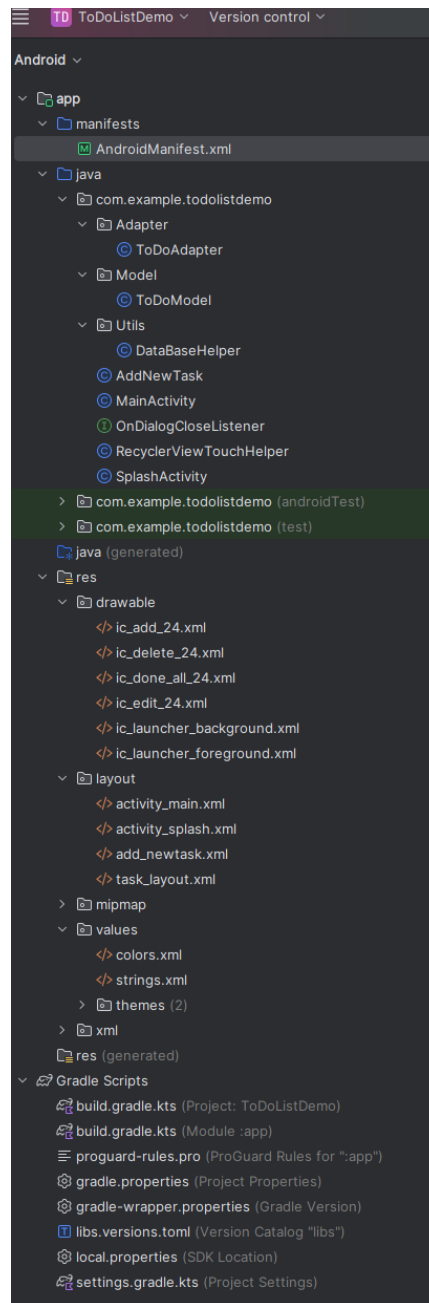
```
build.gradle.kts (:app) ×
You can use the Project Structure dialog to view and edit your project configuration

1  plugins {
2      alias(libs.plugins.android.application)
3  }
4
5  android {
6      namespace = "com.example.todolistdemo"
7      compileSdk = 35
8
9      defaultConfig {
10         applicationId = "com.example.todolistdemo"
11         minSdk = 26
12         targetSdk = 35
13         versionCode = 1
14         versionName = "1.0"
15
16         testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
17     }
18
19     buildTypes {
20         release {
21             isMinifyEnabled = false
22             proguardFiles(
23                 getDefaultProguardFile("proguard-android-optimize.txt"),
24                 "proguard-rules.pro"
25             )
26         }
27     }
28     compileOptions {
29         sourceCompatibility = JavaVersion.VERSION_11
30         targetCompatibility = JavaVersion.VERSION_11
31     }
32 }
33
34 dependencies {
35     implementation(libs.appcompat)
36     implementation(libs.material)
37     implementation(libs.activity)
38     implementation(libs.constraintlayout)
39     testImplementation(libs.junit)
40     androidTestImplementation(libs.ext.junit)
41     androidTestImplementation(libs.espresso.core)
42     //implementation("it.xabaras.android:recyclerview-swipedecorator:1.4")
43     implementation(libs.recyclerview.swipedecorator)
44
45 }
46
```

Android Manifest

```
build.gradle.kts (:app) × AndroidManifest.xml ×
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools">
4
5     <application
6         android:allowBackup="true"
7         android:dataExtractionRules="@xml/data_extraction_rules"
8         android:fullBackupContent="@xml/backup_rules"
9         android:icon="@mipmap/ic_launcher"
10        android:label="@string/app_name"
11        android:roundIcon="@mipmap/ic_launcher_round"
12        android:supportsRtl="true"
13        android:theme="@style/Theme.ToDoListDemo"
14        tools:targetApi="31">
15        <activity
16            android:name=".SplashActivity"
17            android:exported="true">
18            <intent-filter>
19                <action android:name="android.intent.action.MAIN" />
20
21                <category android:name="android.intent.category.LAUNCHER" />
22            </intent-filter>
23        </activity>
24
25        <activity
26            android:name=".MainActivity"
27            android:exported="true">
28
29        </activity>
30    </application>
31
32 </manifest>
```

Project Structure



Screenshots of output:

