**Part 1: Creating RecyclerView Project with Dataset**

In this tutorial, an example project will be created that makes use of both the CardView and RecyclerView components to create a scrollable list of cards.

1. Start Android Studio. Create a new project with the name **CardDemo**, select the **Empty Views Activity** template, and generate the layout file.

1. Configure the project with the view binding features.
2. Open **MainActivity** and add the following member variable.

private lateinit var binding: ActivityMainBinding

1. Set the binding object in the MainActivity.

**Part 2: Create a Card Layout and RecyclerView**

1. Within the Project tool window right-click on the app -> res -> **layout** entry and select the **New -> Layout Resource File** menu option. In the New Resource Dialog enter **card\_layout** into the File name: field and **androidx.cardview.widget.CardView** into the root element field before clicking on the **OK** button.

1. Modify the layout header as follows:

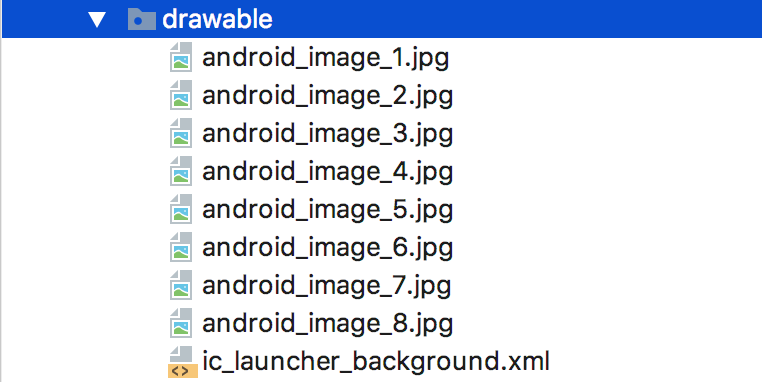


1. Add the following views into the CardView container:

<androidx.constraintlayout.widget.ConstraintLayout  
 android:id="@+id/relativeLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="16dp">  
  
 <ImageView  
 android:id="@+id/itemImage"  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <TextView  
 android:id="@+id/itemTitle"  
 android:layout\_width="236dp"  
 android:layout\_height="39dp"  
 android:layout\_marginStart="16dp"  
 android:textSize="30sp"  
 app:layout\_constraintLeft\_toRightOf="@+id/itemImage"  
 app:layout\_constraintStart\_toEndOf="@+id/itemImage"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <TextView  
 android:id="@+id/itemDetail"  
 android:layout\_width="236dp"  
 android:layout\_height="16dp"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="8dp"  
 app:layout\_constraintLeft\_toRightOf="@+id/itemImage"  
 app:layout\_constraintStart\_toEndOf="@+id/itemImage"  
 app:layout\_constraintTop\_toBottomOf="@+id/itemTitle" />  
</androidx.constraintlayout.widget.ConstraintLayout>

1. Select the **activity\_main.xml** file and drag a **RecyclerView** object from the Containers section of the palette onto the layout so that it is positioned in the center of the screen where it should automatically resize to fill the entire screen.

1. Change the ID of the RecyclerView instance to **recyclerView** and the layout\_width and layout\_height properties to **match\_constraint.**
2. Copy all the eight images to the drawable folder.

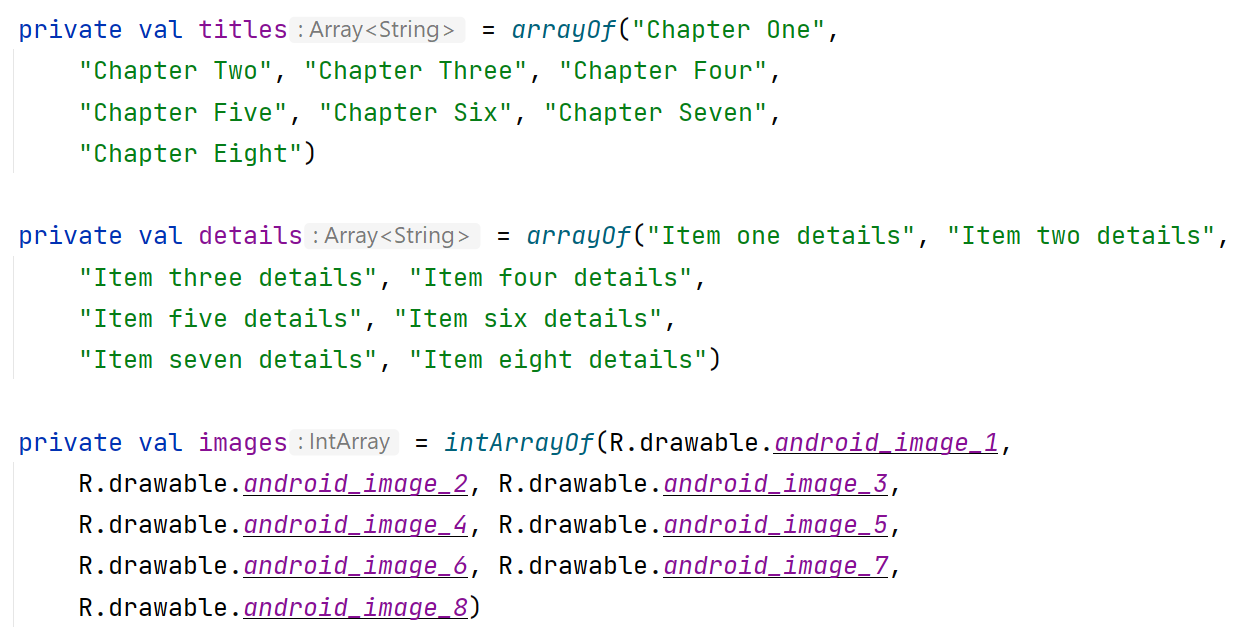


**Part 3: Create RecyclerView Adapter**

1. Right-click **the default package of source code** and select **New > Kotlin Class/File.**

1. Name the class **RecyclerAdapter.**

1. In the class, add in the following array items:



1. Extends the **RecyclerAdapter class with the RecyclerView.Adapter class.**

class RecyclerAdapter:**RecyclerView.Adapter<RecyclerAdapter.ViewHolder>() {}**

1. Click the class declaration (**RecyclerAdapter**), then click the red light bulb on the left side of the pane. Choose **Implement methods.** A dialog appears that asks you to choose which methods to implement. Choose all three methods and click **OK.** Android Studio creates empty placeholders for all the methods.

1. Inside the **RecyclerAdapter** class, add a new inner class  **ViewHolder**  with this signature:

inner class ViewHolder(itemView: View):RecyclerView.ViewHolder(itemView) {}

1. Add variables to the ViewHolder inner class for the adapter:

var itemImage: ImageView  
var itemTitle: TextView  
var itemDetail: TextView

1. The ViewHolder class contains an ImageView and two TextView variables together with a constructor method that initializes those variables with references to the three view items in the card\_layout.xml file.

1. Create an init method to initialise all the three view items.

init {  
 itemImage = itemView.findViewById(R.id.*itemImage*)

...

}

1. Implement the OnClickListener to the itemView and display which item was clicked with the Snackbar view.

itemView.setOnClickListener **{** v: View **->**

**...**

**}**

1. You can now fill in the getItemCount() method to return the size of array.

override fun getItemCount(): Int {  
 return titles!!.size  
}

1. Fill out the onCreateViewHolder() method with this code:

override fun onCreateViewHolder(viewGroup: ViewGroup, i: Int): ViewHolder {  
 val v = LayoutInflater.from(viewGroup.*context*)  
 .inflate(R.layout.*card\_layout*, viewGroup, false)  
 return ViewHolder(v)  
}

1. Fill out the onBindViewHolder() method with the code below:

override fun onBindViewHolder(viewHolder: ViewHolder, i: Int) {  
 viewHolder.itemTitle.*text* = titles[i]  
 viewHolder.itemDetail.*text* = details[i]  
 viewHolder.itemImage.setImageResource(images[i])  
}

1. Run your app to make sure that there are no errors.

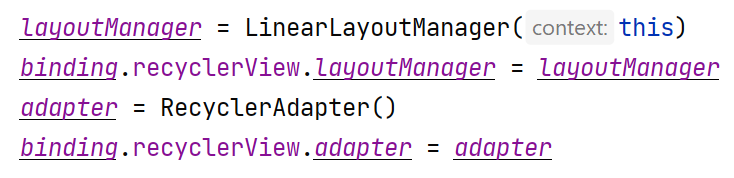
**Part 4: Create the RecyclerView in the Activity**

1. Now that you have an adapter with a ViewHolder, you can finally create a RecyclerView and connect all the pieces to display your data. Open **MainActivity.**

1. Add member variables in the **MainActivity** class:

private var *layoutManager*: RecyclerView.LayoutManager? = null  
private var *adapter*: RecyclerAdapter? = null

1. In the onCreate() method of MainActivity, add the following code that creates the RecyclerView and connects it with an adapter and the data. The comments explain each line. You must insert this code after the **setContentView** method:



1. Run your app.
2. Click the card item to view the Snackbar message.

