1. Create empty project, Inside activity\_main layout insert RecyclerView widget. (to use RecyclerView)

    <androidx.recyclerview.widget.RecyclerView

        android:id="@+id/recyclerView"

        android:layout\_width="match\_parent"

        android:layout\_height="match\_parent" />

**activity\_main.xml contents:**

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="<http://schemas.android.com/apk/res/android>"

    xmlns:tools="<http://schemas.android.com/tools>"

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent"

    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView

        android:id="@+id/recyclerView"

        android:layout\_width="match\_parent"

        android:layout\_height="match\_parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

1. Create a new layout file item.xml (it will show for each item).

**Item.xml contents:**

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="<http://schemas.android.com/apk/res/android>"

    android:layout\_width="match\_parent"

    android:layout\_height="wrap\_content"

    android:layout\_margin="8dp"

    android:background="#8EAF9E">

        <TextView

            android:id="@+id/itemTextView"

            android:layout\_width="match\_parent"

            android:layout\_height="50dp"

            android:layout\_alignParentStart="true"

            android:textColor="#000000"

            android:gravity="center"

            android:textSize="20sp" />

</LinearLayout>

1. Create your own adapter (CustomAdapter in another file CustomAdapter.kt) which extends RecyclerView.Adapter functionality. To create own adapter, we need to override 3 methods which already available in RecyclerView.Adapter.

a) onCreateViewHolder()

b) onBindViewHolder()

c) getItemCount()

1. oncCreateViewHolder() creates MyViewHolder widget for each item with the item.xml layout file
2. onBindViewHolder() binds each item with MyViewHolder and position of the item as arguments. Using this position argument, we can get the item from the item list. And we can use this positioned item to display in MyViewHoledr widget(which was created by the help of item.xml)
3. getItemCount() sets the number of elements to be displayed in the RecyclerView. Can be got by `list.size` command

**CustomAdapter.kt contents:**

package com.example.myapplication //our packagename

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import android.widget.TextView

import androidx.annotation.NonNull

import androidx.recyclerview.widget.RecyclerView

//internal class only visible to current module com.example.myapplication

//inner class means just a nested class inside our current CustomAdapter class

//private var itemList means it is only available in current file CustomAdapter.kt

//Creating CustomAdapter class which takes itemList as parameter, where itemList is a list of string. Note that, CustomAdapter will be a extended from RecyclerView.Adapter class, where that RecyclerView.Adapter is made from our CustomAdapter class’s MyViewHolder class. It ends with () since it’s (RecyclerView.Adapter() ) a class

//Now create our MyViewHolder class, since it is a view, take view: View as parameter for this class, note that MyViewholder is extended from RecyclerView.ViewHolder(view) class.

//Since it’s a view, so populate with variable itemTextview: TextView = view.findViewById(R.id.itemTextView)

We can populate more variable if we need. Like ImageView or multiple TextViews.

internal class CustomAdapter(private var itemsList: List<String>) :

   RecyclerView.Adapter<CustomAdapter.MyViewHolder>() {

   internal inner class MyViewHolder(view: View) : RecyclerView.ViewHolder(view) {

      var itemTextView: TextView = view.findViewById(R.id.itemTextView)

   }

//So our viewholder is created, now we need to create a function when our viewholder triggers. Since its already created, we need to override it.

onCreateViewHolder takes to parameter, parent: ViewGroup, viewType: Int. Note that it belongs to MyViewHolder class.

Now we need to inflate or populate the view with our item xml, right?

In onCreateViewHolder function, create a val itemView = LayoutInflater.from(parent,context).inflate(R.layout.item, parent, false)

LayoutInflater class inflate new layout in current view, it takes from and inflate function to do this. From function gets the contex/parent/viewgroup. And current context, and returns a LayoutInflater class,

Then we can use inflate function upon that from returned value,

Inflate function takes a layout file to which it should inflate, then it sets the viewgroup/parent where to show it, then it uses root or not. Here non root, ie. False

Lastly our function should return MyViewHolder which we already told it takes a view as parameter, here we must provide the view we got in this function as val itemview.

   @NonNull

 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {

      val itemView = LayoutInflater.from(parent.context)

      .inflate(R.layout.item, parent, false)

      return MyViewHolder(itemView)

   }

// we created viewholder now, so we know which items will bind to that viewholder. From and to which layout. Now we need to bind those data with the viewholder

Create a onBindViewHolder function. Which takes holder variable which is of Myviewholder class type, and a position which is Int type. We already told onBindViewHolder takes two argument, which viewholder to use, and in which position to use, Here viewholder is MyViewHolder, and position is itemlist[position]

So val item = itemList[position], and set item to holder.itemTextView.text. where holder was a MyViewHolder.

Means, take the item positon and bind to our any holder: MyViewholder

   override fun onBindViewHolder(holder: MyViewHolder, position: Int) {

      val item = itemsList[position]

      holder.itemTextView.text = item

   }

//last and easy part, get totol list count by itemLsit.size.

   override fun getItemCount(): Int {

      return itemsList.size

   }

}

1. now in MainActivity.kt

get reference of the RecyclerView by findViewById.

Initialize our created CustomAdapter with items list.

Set layout manager with LinearLayoutManager for RecyclerView

Set adapter with CustomAdapter for the RecyclerView

Prepare item and update RecylcerView via CustomAdapter using notifyDataSetChanged()

**MainActivity.kt contents:**

package com.example.myapplication

import androidx.appcompat.app.AppCompatActivity

import android.os.Bundle

import androidx.recyclerview.widget.DefaultItemAnimator

import androidx.recyclerview.widget.LinearLayoutManager

import androidx.recyclerview.widget.RecyclerView

class MainActivity : AppCompatActivity() {

    private val itemsList = ArrayList<String>() //create a arraylist of string, which we can add items whenever needed

    private lateinit var customAdapter: CustomAdapter //create a var customAdapter which is of CustomAdapter class

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity\_main)

        title = "RecyclerView - [www.tutorialkart.com](http://www.tutorialkart.com)" //setting the title of the current page

        val recyclerView: RecyclerView = findViewById(R.id.recyclerView)

        customAdapter = CustomAdapter(itemsList) //Remember, CustomAdapter class used to took a parameter of List of strings.

        val layoutManager = LinearLayoutManager(applicationContext) //a layoutManager variable which is a object of LinearLayoutManager which takes applicationContext as argument.

        recyclerView.layoutManager = layoutManager //set our recylerView variable’s layoutmanager to created LinearLayoutmanager called layoutManager we created.

        recyclerView.adapter = customAdapter //set recyclerView.adapter to our customAdapter.

        prepareItems() //just a function which adds all items, in a list.

    }

    private fun prepareItems() {

        itemsList.add("Item 1") //add items in our previously created itemlist of ArrayList of string

        itemsList.add("Item 2")

        itemsList.add("Item 3")

        itemsList.add("Item 4")

        itemsList.add("Item 5")

        itemsList.add("Item 6")

        itemsList.add("Item 7")

        itemsList.add("Item 8")

        itemsList.add("Item 9")

        itemsList.add("Item 10")

        itemsList.add("Item 11")

        itemsList.add("Item 12")

        itemsList.add("Item 13")

        customAdapter.notifyDataSetChanged() //our customAdapter has all properties of RecyclerViewAdapter, so use it’s notifyDataSetChanged() function to notify we finished adding data. All this done inside the normal oncreate mainactivity function obviously, ie. In previous section. Its just a simple function to do things simpler.

    }

}