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

The RecyclerView app demonstrates how to use a [RecyclerView](https://developer.android.com/reference/android/support/v7/widget/RecyclerView.html) to display a long scrollable list of words. You create the dataset (the words), the RecyclerView itself, and the actions the user can take.

1. Start Android Studio. Create a new project with the name **RecyclerView**, 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 two private member variables.

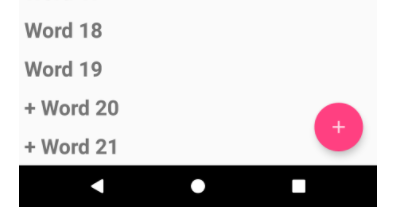
private lateinit var binding: ActivityMainBinding  
private var mWordList = LinkedList<String>()

1. Set the binding object in the MainActivity.

1. Add code within the onCreate() method that populates mWordList with words:

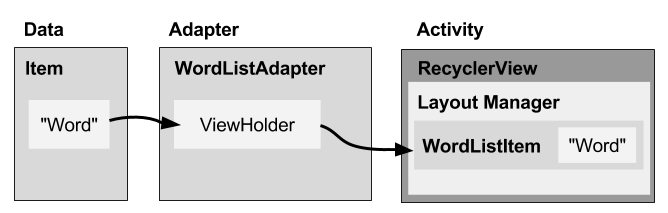
for (i in 0..19) {  
 mWordList.addLast("Word $i")  
}

1. Add the FAB icon.
2. Expand **res** in the **Project > Android** pane, and right-click (or Control-click) the **drawable** folder.
3. Choose **New > Image Asset**. The Configure Image Asset dialog appears.
4. Choose **Action Bar and Tab Items** in the drop-down menu at the top of the dialog.
5. Change **ic\_action\_name** in the **Name** field to **ic\_add\_for\_fab**.
6. Click the clip art image (the Android logo next to **Clipart:**) to select a clip art image as the icon. A page of icons appears. Click the icon you want to use for the FAB, such as the plus (**+**) sign.
7. Choose **HOLO\_DARK** from the **Theme** drop-down menu. This sets the icon to be white against a dark-colored (or black) background. Click **Next**.
8. Click **Finish** in the Confirm Icon Path dialog.



**Part 2: Create a RecyclerView**

1. The diagram below shows the relationship between the data, the adapter, the ViewHolder, and the layout manager.



1. To implement these pieces, you will need to:
2. Add a RecyclerView element to the MainActivity XML content layout (content\_main.xml) for the RecyclerView app.
3. Create an XML layout file (wordlist\_item.xml) for one list item, which is WordListItem.
4. Create an adapter (WordListAdapter) with a ViewHolder (WordViewHolder). Implement the method that takes the data, places it in the ViewHolder, and lets the layout manager know to display it.
5. In the onCreate() method of MainActivity, create a RecyclerView and initialize it with the adapter and a standard layout manager.

**Part 2.1: Add a RecyclerView element:**

1. Open **content\_main.xml** in your RecyclerView app. It shows a "Hello World" TextView at the center of a ConstraintLayout.

1. Click the **Text** tab to show the XML code. Replace the entire TextView element with the following:

<androidx.recyclerView.widget.RecyclerView

android:id="@+id/recyclerview"

android:layout\_width="match\_parent"

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

**Part 2.2: Create the layout for one list item**

1. Right-click the **app > res > layout** folder and choose **New > Layout resource file.**

1. Name the file **wordlist\_item** and click **OK.**
2. In the new layout file, click the **Text** tab to show the XML code .
3. Change the ConstraintLayout that was created with the file to a LinearLayout with the following attributes (extract resources as you go):

|  |  |
| --- | --- |
| **LinearLayout attribute** | **Value** |
| android:layout\_width | "match\_parent" |
| android:layout\_height | "wrap\_content" |
| android:orientation | "vertical" |
| android:padding | "6dp" |

1. Add a TextView for the word to the LinearLayout. Use word as the ID of the word:

|  |  |
| --- | --- |
| **Attribute** | **Value** |
| android:id | "@+id/word" |
| android:layout\_width | "match\_parent" |
| android:layout\_height | "wrap\_content" |
| android:textSize | "24sp" |
| android:textStyle | "bold" |

1. Open the themes.xml from values folder. Add the following style in this file.

<style name="word\_title">

<item name="android:layout\_width">match\_parent</item>

<item name="android:layout\_height">wrap\_content</item>

<item name="android:textSize">24sp</item>

<item name="android:textStyle">bold</item>

</style>

1. Refactor the TextView.

<TextView

android:id="@+id/word"

style="@style/word\_title" />

**Part 2.3: Create RecyclerView Adapter**

1. Right-click **java/com.android.example.recyclerview** and select **New > Kotlin Class/File.**

1. Name the class **WordListAdapter.**

1. Give WordListAdapter the following signature:

public class WordListAdapter(val context: Context, val wordList:LinkedList<String>):  
 RecyclerView.Adapter<WordListAdapter.WordViewHolder>() {

}

1. Click the class declaration (**WordListAdapter**), 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. Note how onCreateViewHolder and onBindViewHolder both reference the WordViewHolder, which hasn't been implemented yet

1. Inside the WordListAdapter class, add a new WordViewHolder inner class with this signature:

inner class WordViewHolder(itemView: View, adapter: WordListAdapter) : RecyclerView.ViewHolder(itemView) {  
}

1. Add variables to the WordViewHolder inner class for the TextView and the adapter:

val wordItemView = itemView.findViewById<TextView?>(R.id.*word*)  
 val mAdapter = adapter

1. Run your app to make sure that you have no errors. Your will still see only a blank view

1. You need to hold your data in the adapter, and WordListAdapter needs a constructor that initializes the word list from the data. To hold your data in the adapter, create a private linked list of strings in WordListAdapter and call it mWordList:

private lateinit var mWordList: LinkedList<String>

1. WordListAdapter needs a constructor that initializes the word list from the data. To create a View for a list item, the WordListAdapter needs to inflate the XML for a list item. You use a layout inflator for that job. Start by creating a member variable for the inflater in WordListAdapter:

private lateinit var mInflater: LayoutInflater

1. Implement the **init** method to initialise the **mWordList** and **mInflater** members.

init {  
 mInflater = LayoutInflater.from(context)  
 mWordList = wordList *//from the MainActivity (20 words)*}

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

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

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

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): WordViewHolder {  
 val mItemView = mInflater!!.inflate(R.layout.*wordlist\_item*,  
 parent, false  
 )  
  *//try to create each row in the recyclerview layout* return WordViewHolder(mItemView, this)  
}

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

override fun onBindViewHolder(holder: WordViewHolder, position: Int) {  
 val mCurrent = mWordList!![position]  
 holder.wordItemView.*text* = mCurrent  
}

The onBindViewHolder() method connects your data to the view holder.

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

**Part 2.4: Create the RecyclerView in the Activity**

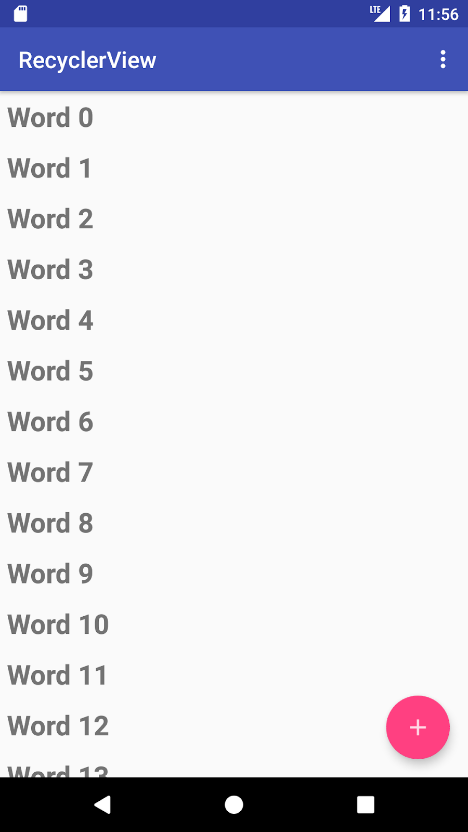
1. Now that you have an adapter with a ViewHolder, you can finally create a RecyclerView and connect all the pieces to display your data. Open **MainActivity.**

1. Add member variables for the adapter:

private lateinit var mAdapter: WordListAdapter

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 mWordList initialization:
2. mAdapter = WordListAdapter(this, mWordList)  
     
   *//set the layout for the recyclerview  
   //If this step was skipped, compilation error will appear*binding.recyclerview.setLayoutManager(  
    LinearLayoutManager(this)  
   )  
     
   *//set the adapter for the recyclerview*binding.recyclerview.setAdapter(mAdapter)

1. Run your app.



**Part 3: Make the list interactive**

1. Open **WordListAdapter.** Change the WordViewHolder class signature to implement [View.onClickListener](https://developer.android.com/reference/android/view/View.OnClickListener.html):

inner class WordViewHolder(itemView: View, adapter: WordListAdapter) : RecyclerView.ViewHolder(itemView), **View.OnClickListener** {

1. Click the class header and on the red light bulb to implement stubs for the required methods, which in this case is just the onClick() method.

1. Add the following code to the body of the onClick() method.

override fun onClick(v: View?) {  
 val position = *layoutPosition* val element = mWordList.get(position)  
  
 mWordList.set(position, "Clicked $element")  
  
 mAdapter.notifyDataSetChanged()  
}

1. Connect the onClickListener with the View. Add this code to the WordViewHolder constructor (below the this.mAdapter = adapter line)

init {  
 itemView.setOnClickListener(this)  
}

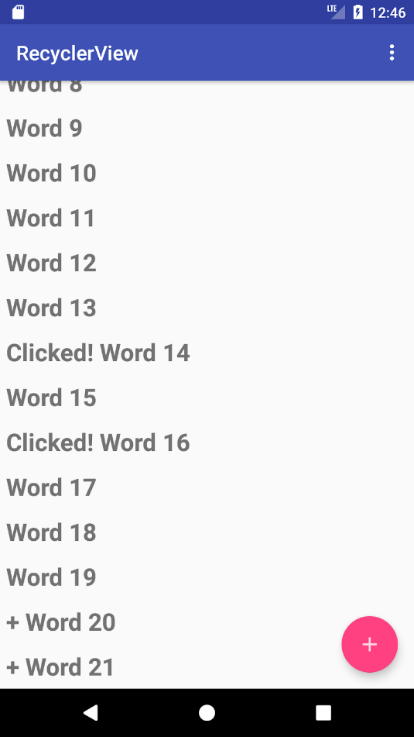
1. Run your app. Click items to see the text change.

**Part 4: Add Behaviour to the FAB**

1. Open **MainActivity**. The onCreate() method sets an OnClickListener() to the FloatingActionButton with an onClick() method for taking an action. Change the onClick() method to the following:

binding.fabList.setOnClickListener **{** val size = mWordList.size  
  
 mWordList.addLast("+Word $size")  
 binding.recyclerview.*adapter*?.notifyDataSetChanged()  
 binding.recyclerview.smoothScrollToPosition(size)  
**}**

1. Run the app. Scroll the list of words and click items.



1. Add items by clicking the FAB.