# CS 473 – MDP
# Mobile Device Programming



Maharishi International University

# CS 473 – MDP
# Mobile Device Programming

MS.CS Program
Department of Computer Science
Renuka Mohanraj, Ph.D.,

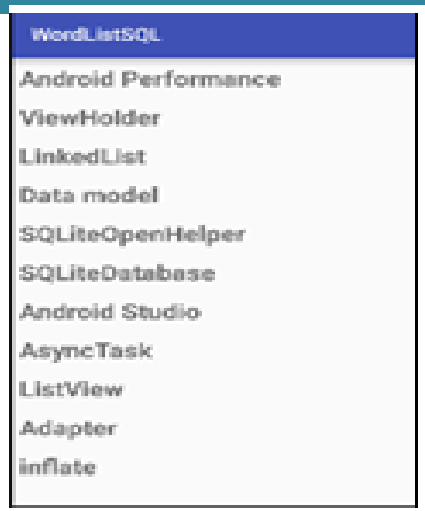
# CS 473 – MDP
# Mobile Device Programming

Lesson-6
User Input Controls – Day 2
Advanced UI – RecyclerView and CardView

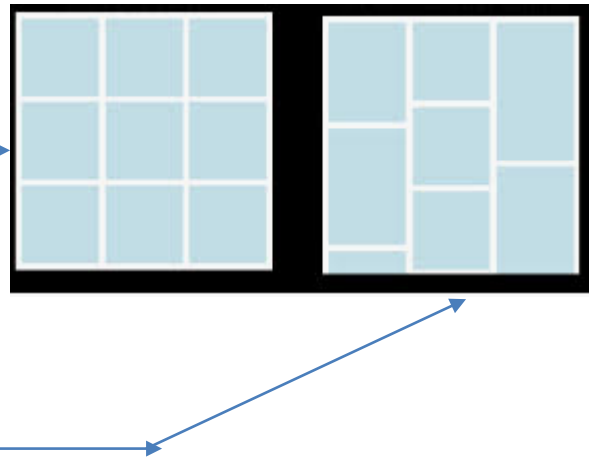
# Contents

- What is a Recycler View?
- RecyclerView Components Overview
- Implementing RecyclerView Steps
- Hands on Example – 1 using RecyclerView to show the Book List
- CardView
- Hands on Example – 2 using RecyclerView, CardView and Listener Implementation

# What is a Recycler View?

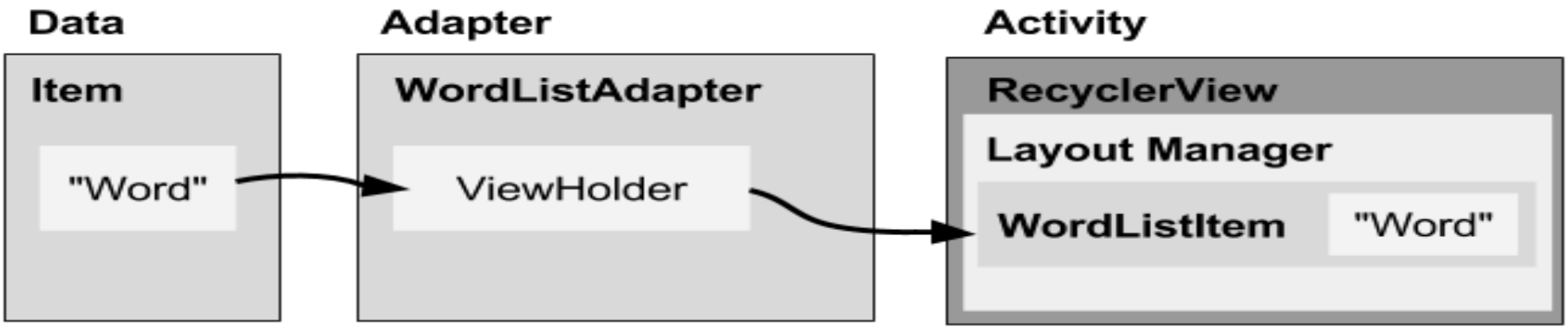



- Scrollable container for large data sets
- Efficient
  - uses and reuses limited number of views
  - Updates changing data fast
- RecyclerView also provides a choice of three built-in layout managers to control the way in which the list items are presented to the user
  - LinearLayoutManager
  - GridLayoutManager
  - StaggeredGridLayoutManager



# RecyclerView Components Overview

- **Data**
- **RecyclerView** scrolling list —RecyclerView
- **Layout** for one item of data—XML file
- **Layout manage**r handles the organization of UI components in a view—Recyclerview.LayoutManager
- **Adapter** connects data to the RecyclerView—RecyclerView.Adapter
- **View holder** has view information for displaying one item—RecyclerView.ViewHolder



# Implementing RecyclerView Steps

1. Add the RecyclerView dependency to app/build.gradle file ( once you drag the RecyclerCiew in the Layout, dependency added automatically)
2. Add RecyclerView to layout
3. Create XML layout for item
4. Extend RecyclerView.Adapter
5. Extend RecyclerView.ViewHolder
6. In onCreate of activity, create a RecyclerView with adapter and layout manager

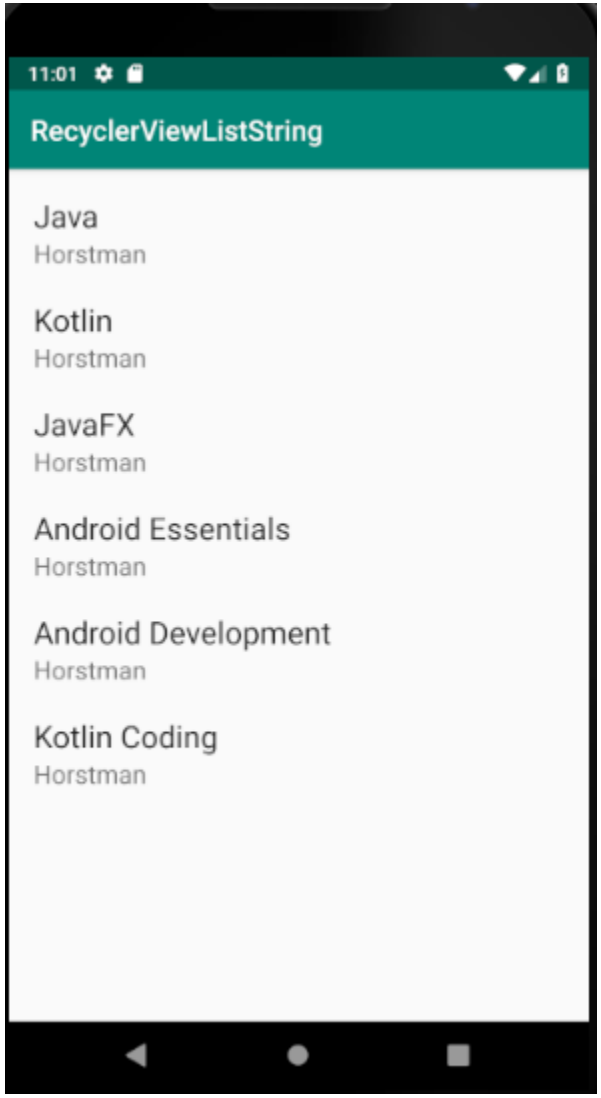
Refer step by step implementation and Demo codes from Lesson6 Day2

# Hands on Example - 1

Problem : Show the RecyclerView with the list of Books.

Book is data class with the attributes book name and author.

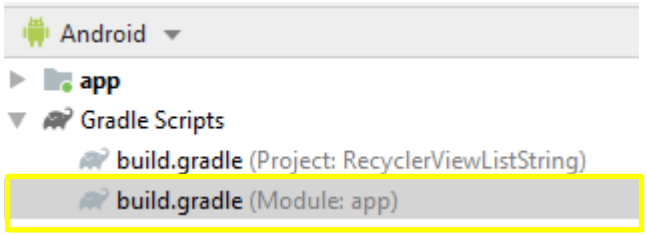
Refer : Lesson6/ RecyclerViewListString



# Implementing RecyclerView

**Step 1 : Add dependency to app/build.gradle**

**Click Gradle Scripts from Android Explorer**




```
dependencies {
  ...
 implementation 'androidx.recyclerview:recyclerview:1.0.0’
  ...
  }
```

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation"org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.core:core-ktx:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
    implementation 'androidx.recyclerview:recyclerview:1.0.0'
}
```

# Adding RecyclerView in the layout

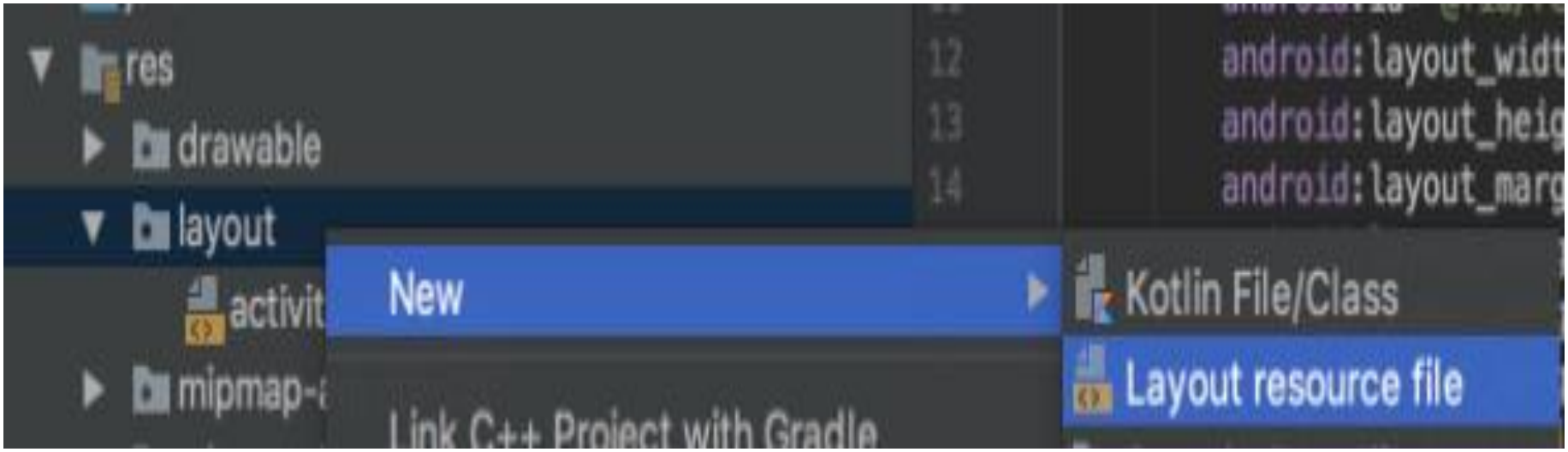
Step:2 In your activity_main.xml layout add the RecyclerView as below

**Add RecyclerView to XML Layout**

```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

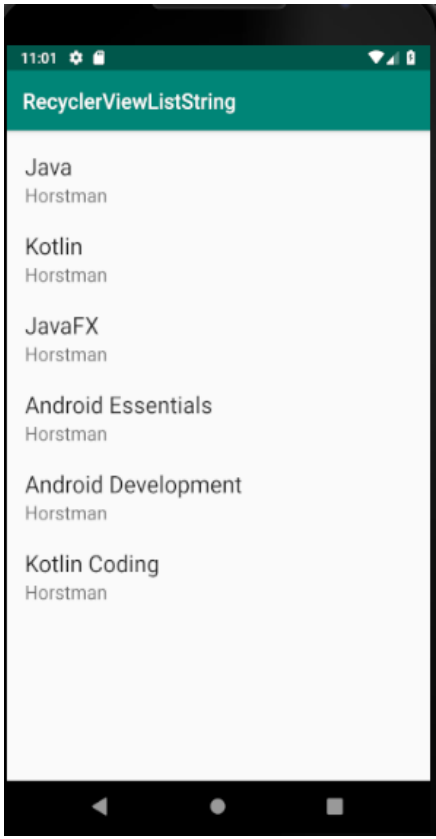
# Creating new layout to display the list item

Step 3. Create layout to show the Book item as per the screenshot and give a name as item_layout.xml



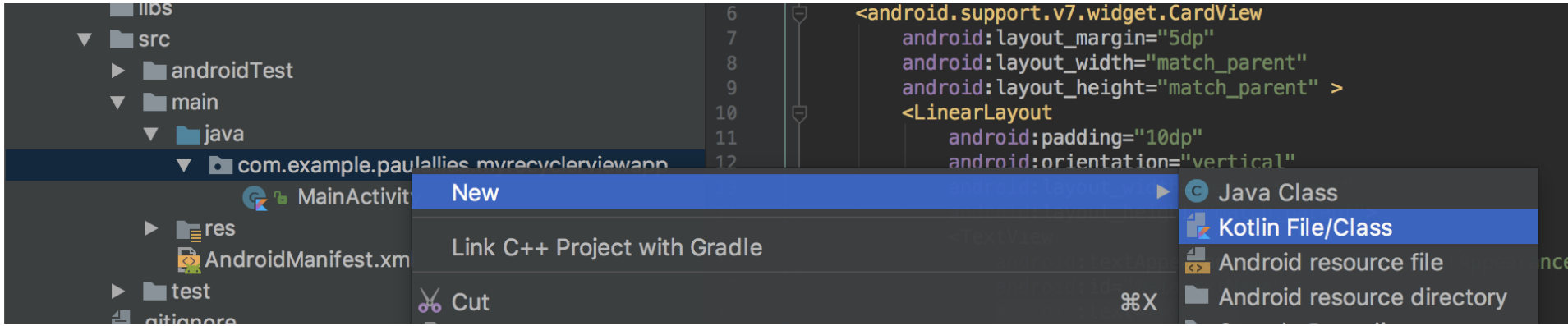
# item_layout.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="10dp">
    <TextView
        android:textAppearance="@style/Base.TextAppearance.AppCompat.Large"
        android:id="@+id/name"
        android:text="Book Name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <TextView
        android:textAppearance="@style/Base.TextAppearance.AppCompat.Medium"
        android:text="Author"
        android:id="@+id/author"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```



# Create a Data Class for Book

- Create Book.kt file as per the screenshot.




Book.kt file code

data class Book(val name: String, val author: String)

# Implementing RecyclerView

## 4. Implement the adapter by creating CustomAdapter.kt

Adapter has 3 required methods need to implement

- onCreateViewHolder()
  - This method creates and returns a ViewHolder object initialized with the view that is to be used to display the data. This view is typically created by inflating the XML layout file.
- onBindViewHolder()
  - This method is passed the ViewHolder object created by the *onCreateViewHolder()* method together with an integer value indicating the list item that is about to be displayed. Contained within the ViewHolder object is the layout assigned by the *onCreateViewHolder()* method. It is the responsibility of the *onBindViewHolder()* method to populate the views in the layout with the text and graphics corresponding to the specified item and to return the object to the RecyclerView where it will be presented to the user.
- getItemCount()
  - This method must return a count of the number of items that are to be displayed in the list.

# 4. Implement the adapter in CustomAdapter.kt

```
class CustomAdapter(val booklist: ArrayList<Book>):
RecyclerView.Adapter<CustomAdapter.MyViewHolder>() {

    override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
      holder?.bName?.text = booklist[position].name
      holder?.bAuthor.text = booklist[position].author
   }

   override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
MyViewHolder {
      val v =
LayoutInflater.from(parent?.context).inflate(R.layout.item_layout,
                parent, false)
      return MyViewHolder(v);
   }

   override fun getItemCount(): Int {
      return booklist.size
   }
```

# MyViewHolder.kt
# inner class

/*RecyclerView.Adapter accepts the generic type of your Adapter inner class ViewHolder type.

In this example Adapter class name is CustomAdapter and the MyViewHolder is the inner class */

```
    class MyViewHolder(itemView: View):
            RecyclerView.ViewHolder(itemView){
        val bName = itemView.findViewById<TextView>(R.id.name)
        val bAuthor = itemView.findViewById<TextView>(R.id.author)
    }
}
```

# Implementing RecyclerView in MainActivity.kt

## 5. Integrate adapter with Recyclerview

```
 import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
```

# Implementing RecyclerView in MainActivity.kt

```
//  get recycler view
      val rv = findViewById<RecyclerView>(R.id.recyclerView1)
      // Give the recycler view with Linear layout manager.
      rv?.layoutManager =  LinearLayoutManager(this)
      // Populate Book data list
      val books = ArrayList<Book>()
      books.add(Book("Java","Horstman"))
      books.add(Book("Kotlin","Horstman"))
      books.add(Book("JavaFX","Horstman"))
      books.add(Book("Android Essentials","Horstman"))
      books.add(Book("Android Development","Horstman"))
      books.add(Book("Kotlin Coding","Horstman"))
      // Create an adapter and supply the data to be displayed.
      var adapter = CustomAdapter(books)
      // Connect the adapter with the recycler view.
      rv.adapter = adapter
   }
}
```

# CardView

- Android 7.0 introduces a new widget called CardView.
- CardView wraps a layout and will often be the container used in a layout for each item within a ListView or RecyclerView.
- Appeared with shadow effects and rounded corners.
- CardView API is an easy way for you to show information inside cards that have a consistent look across the platform.

# Hands on Example - 2

- This example uses
  - Recyclerview displays the Fruit list
  - Each item displayed using CardView. Each row displayed with Fruit image, name of the fruit and small description about the fruit.
  - To display the image in circle shape needs third party library xml code as mentioned below.

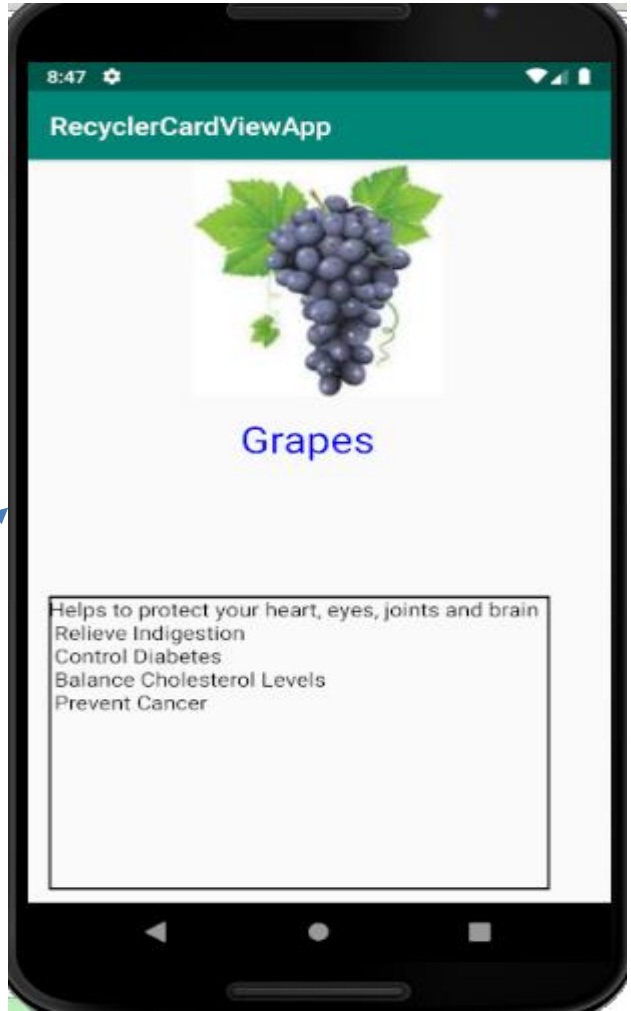
```
<com.mikhaellopez.circularimageview.CircularImageView
        android:id="@+id/imageView"
        android:layout_width="90dp"
        android:layout_height="90dp"
        android:src="@mipmap/ic_launcher" />
```

- Include the below line in build.gradle dependencies

**implementation 'de.hdodenhof:circleimageview:3.0.1'**

# Hands on Example - 2

- Once clicking each item, opens another activity gives detail description about the fruit. Here is the screen after clicking Grapes.
- Refer : Lesson6\RecyclerCardViewApp

# Adding CardView dependency

Inside build.gradle  dependencies add this line

```
dependencies {
implementation 'androidx.cardview:cardview:1.0.0’
}
```

# Adding CardView Layout

Add into below dependency in the Gradle Module.app to use cardview

implementation 'androidx.cardview:cardview:1.0.0’

XML Code for the CardView

```
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="20dp"
    android:id="@+id/card_view"
    app:cardBackgroundColor="#e4b9f1"
    app:cardCornerRadius="12dp"
    android:layout_marginBottom="4dp">
```

# ListView vs RecyclerView

- ListView
  - ListView provides only one type of view i.e Vertical ListView, and there is no way to implement the others such as Horizontal ListView, GridView etc.

- RecyclerView :
  - RecyclerView, with faster performance, is the advanced version of ListView which can be used to implement Vertical List, Horizontal List and Gridview with the help of Layout Managers.
  - RecyclerView is a container for **displaying large data sets** that can be scrolled very efficiently by maintaining a limited number of views.

# Main Point 4

- **RecyclerView** is flexible and efficient version of ListView. It is a container for rendering larger data set of views that can be recycled and scrolled very efficiently. **RecyclerView** is like traditional ListView widget, but with more flexibility to customizes and optimized to work with larger datasets.
- ***Science of Consciousness:*** *pure awareness is an abstraction of individual awareness; each individual provides a specific, concrete realization of unbounded and unmoving pure awareness. Pure awareness is more efficient and powerful.*