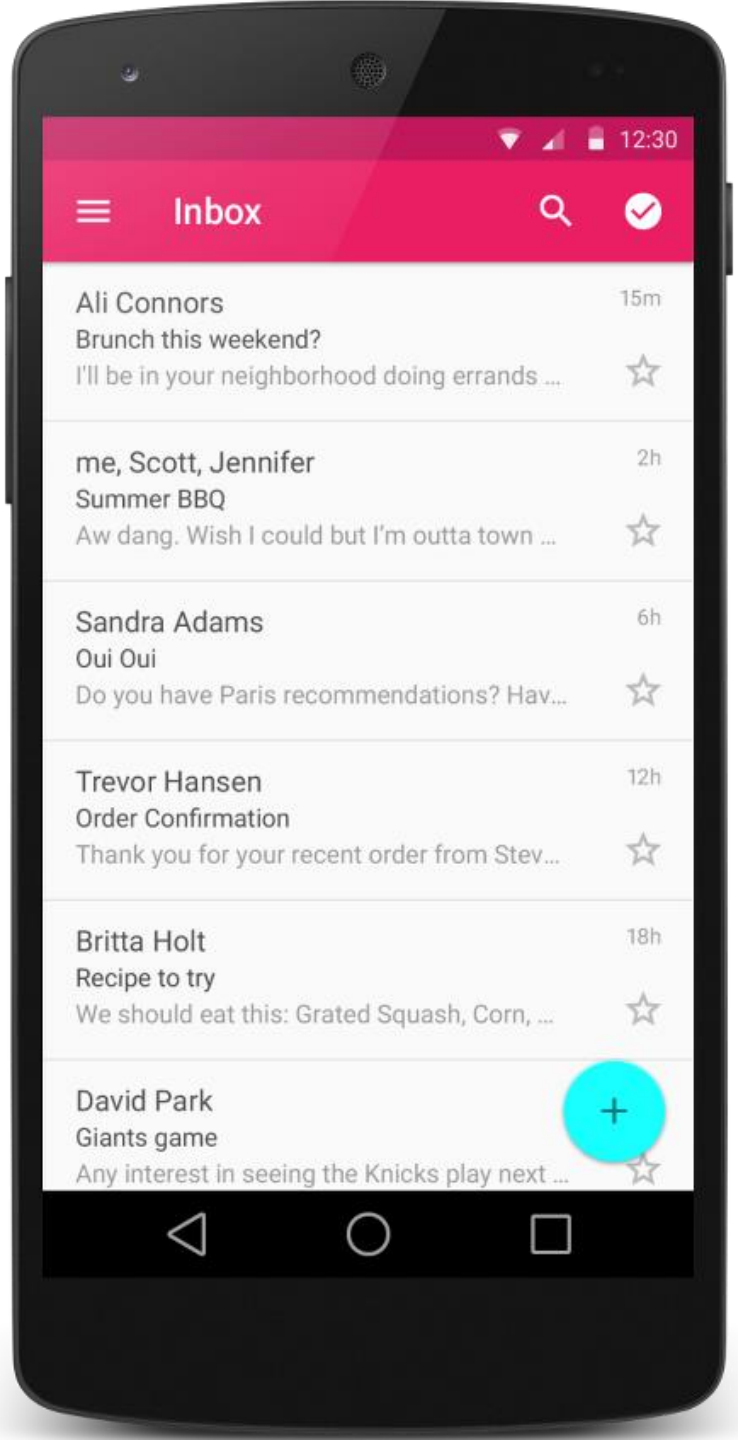
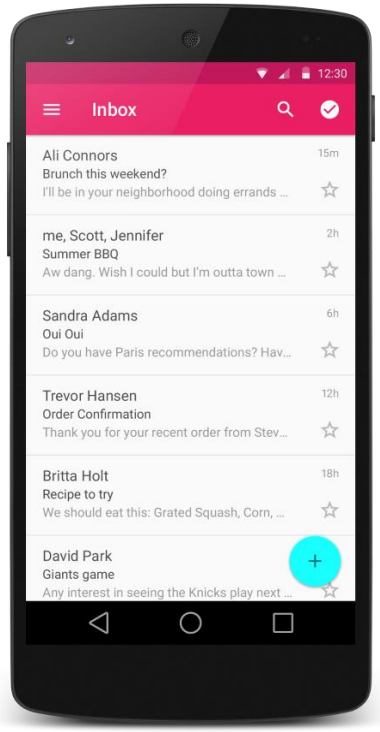


ListView and Adapters

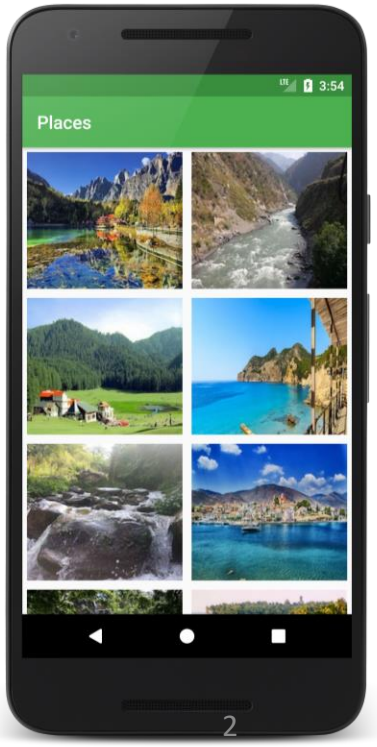
ListView and GridView

- When the **content** for your layout is dynamic or not predetermined, you can use a layout that subclasses AdapterView to populate the layout with views at runtime.



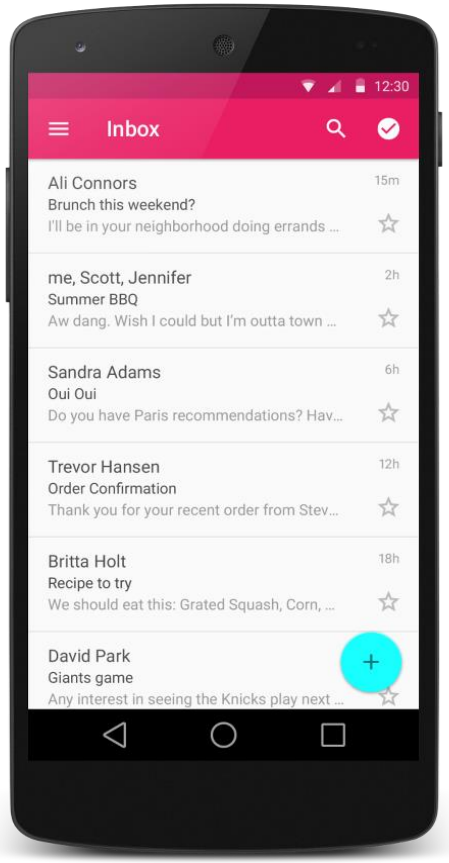
ListView provide a simple way to present scrolling lists of rows

GridView displays a scrolling grid of columns and rows

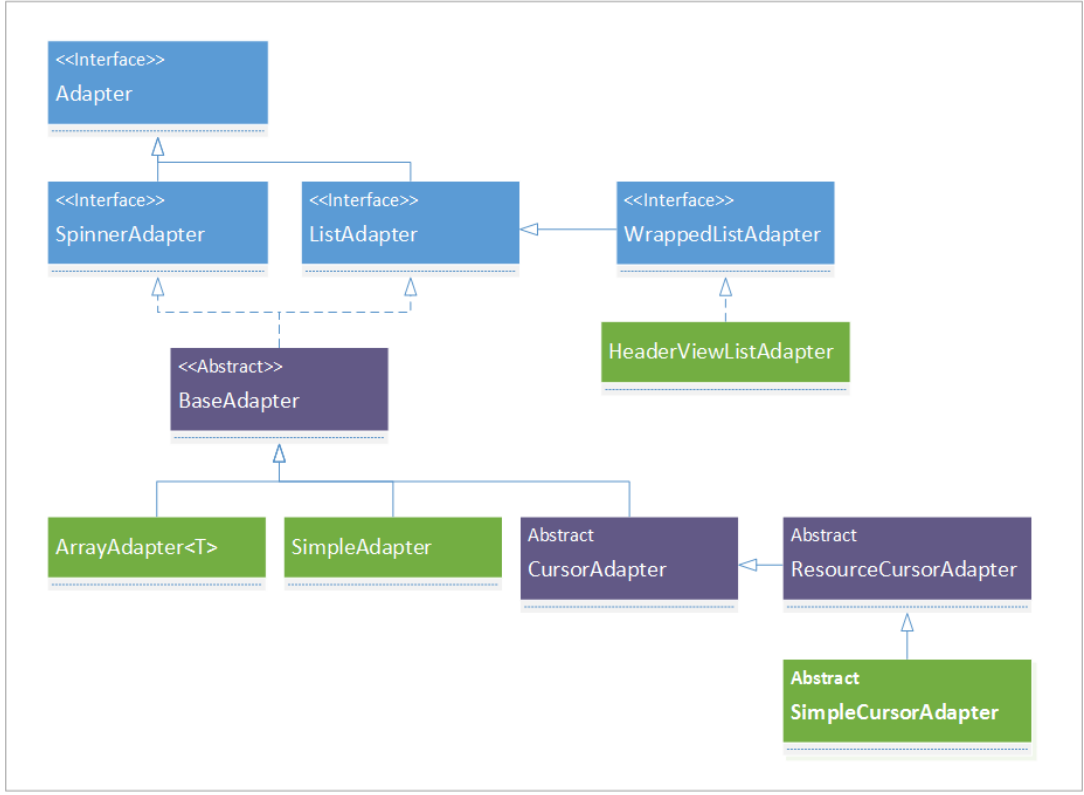


ListView and GridView: components

- Rows: the visible representation of the data in the list or grid.
- Adapter: a non-visual class that binds the data source to the list view.
- **Different Types of Adapter**
  - **ArrayAdapter:** It always accepts an Array or List as input.
  - **CursorAdapter:** It always accepts an instance of cursor as an input means
  - **SimpleAdapter:** It mainly accepts a static data defined in the resources like array or database.
  - **BaseAdapter:** It is a generic implementation for all three adapter types and it can be used in the views according to our requirements.

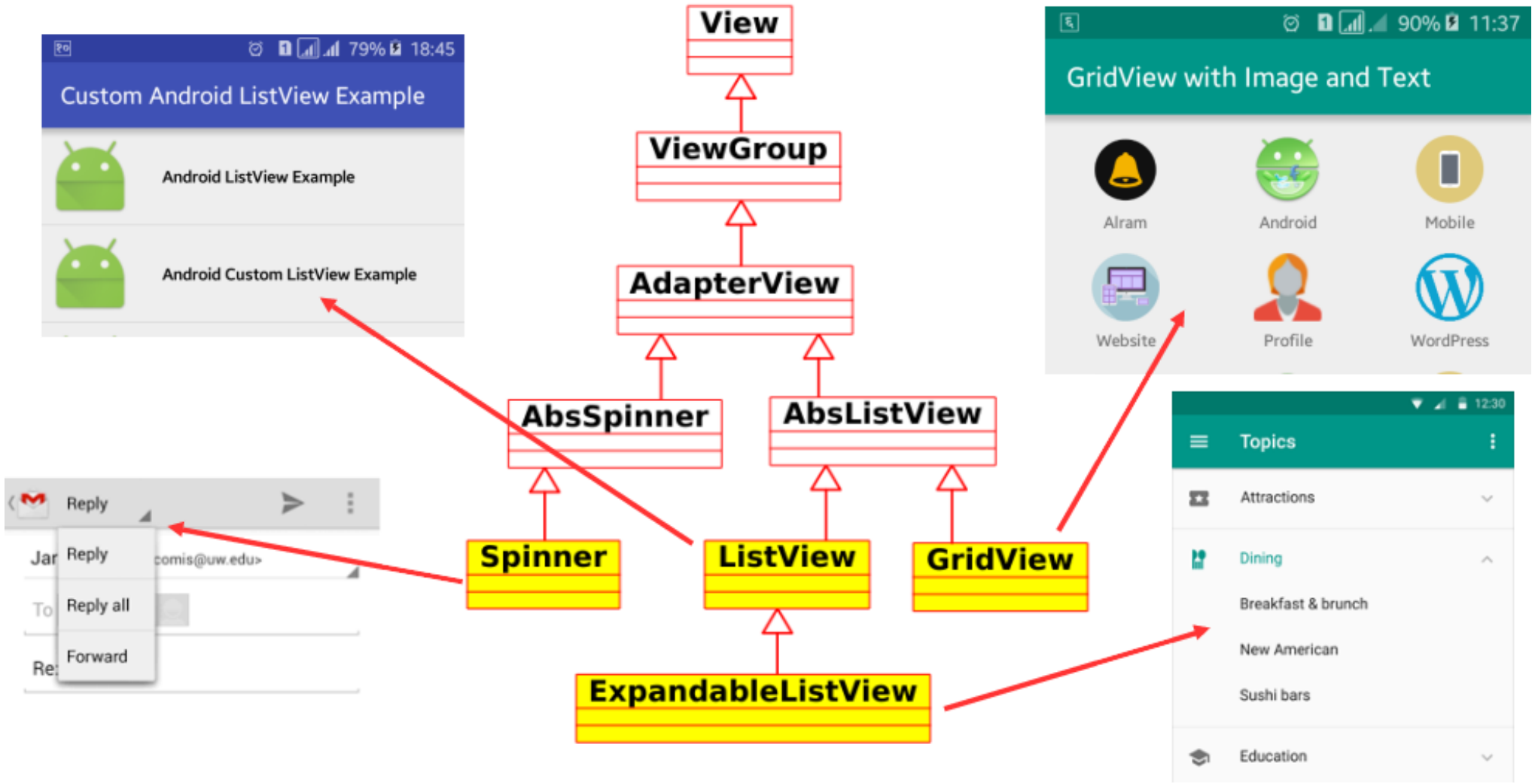


ListView and Adapter



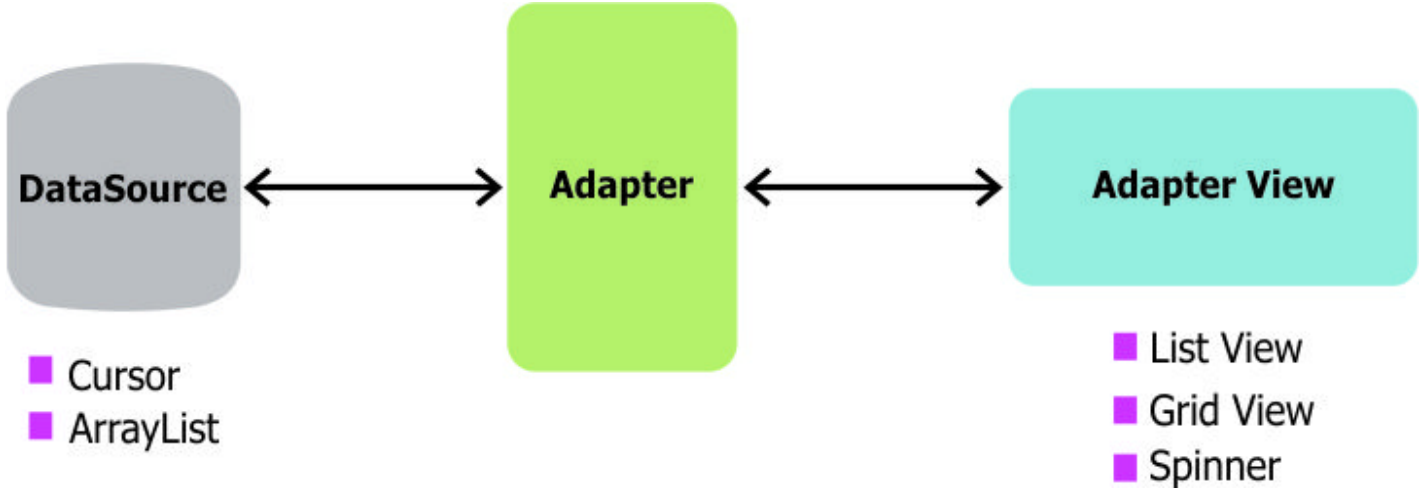
- The **Adapter** is the component which populates the **ListView** with actual data.
- It provides a bridge between the ListView and the actual data structure containing data to visualize:
  - every adapter is an extension of the class ;
  - it uses a layout to format the row content;
  - it manages filtering and sorting of data;
  - it provides method to notify that data has changed.
- Android provides <u>several default adapters</u>; you can create your <u>customized adapter</u> providing an extension of BaseAdapter.

AdapterView class diagram

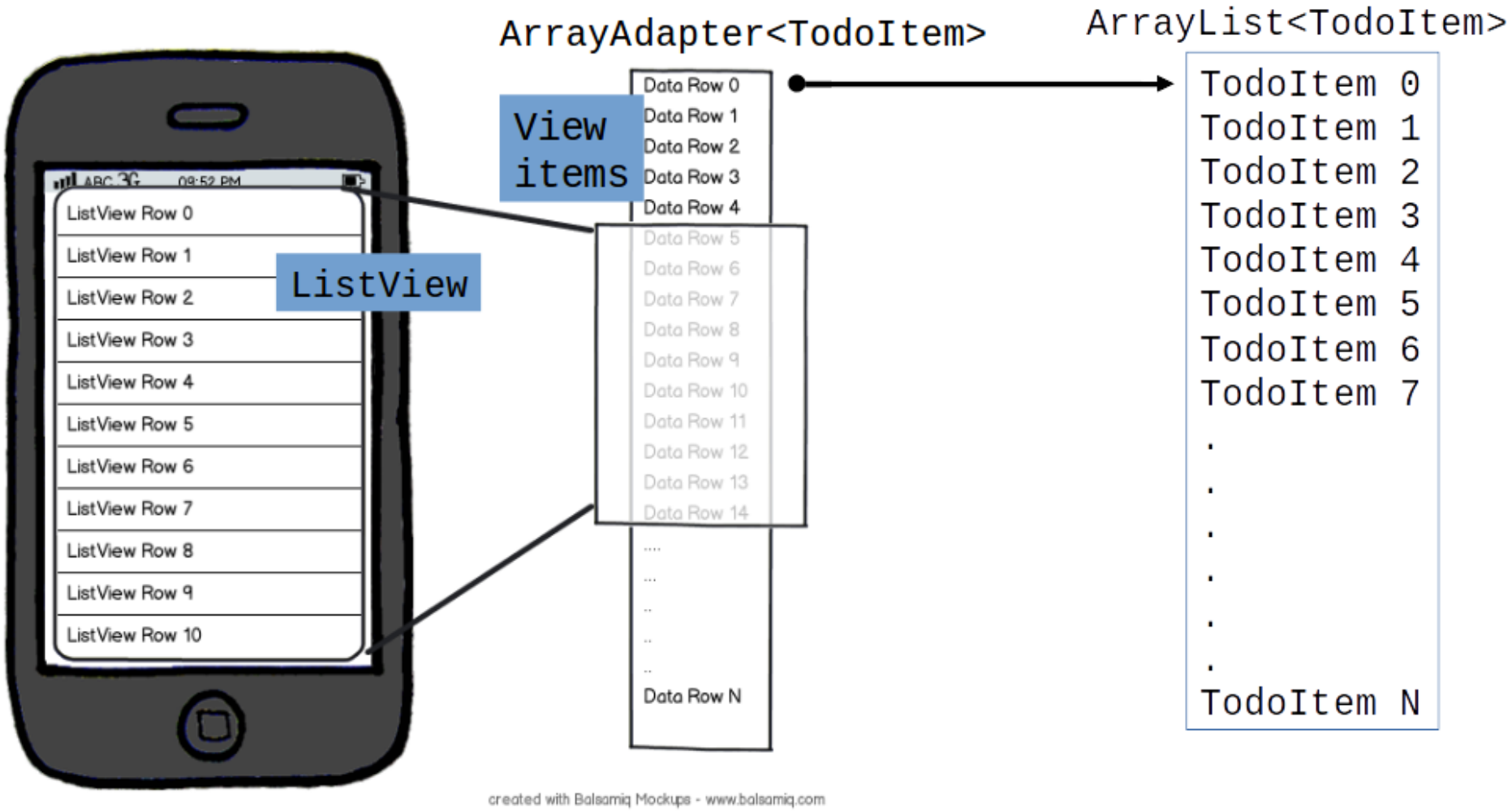


Populating the listview using ArrayAdapter

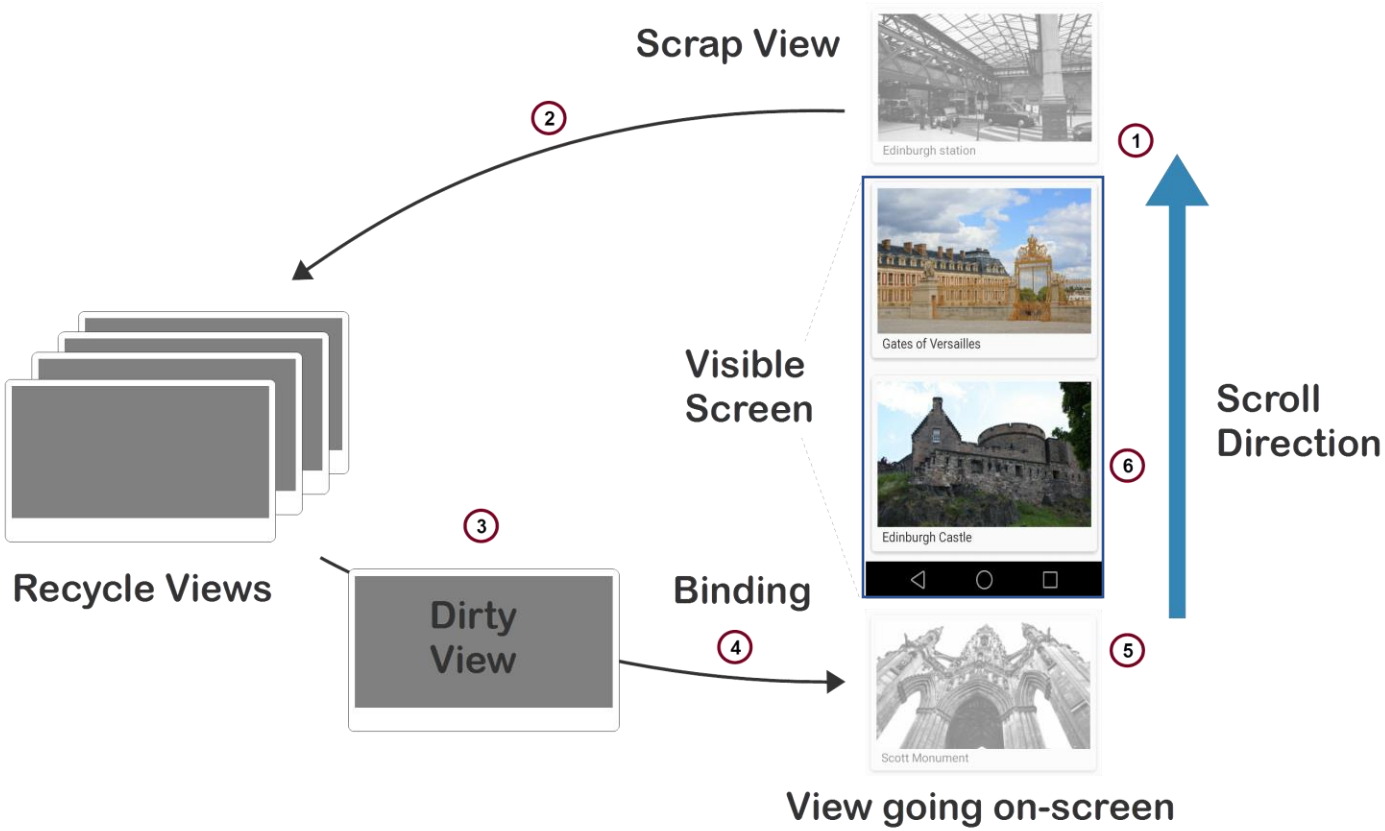
- Any time we want to show a vertical list of scrollable items we will use a ListView which has data populated using an Adapter.
- The simplest adapter to use is called an ArrayAdapter because the adapter converts an ArrayList of objects into View items loaded into the ListView container.



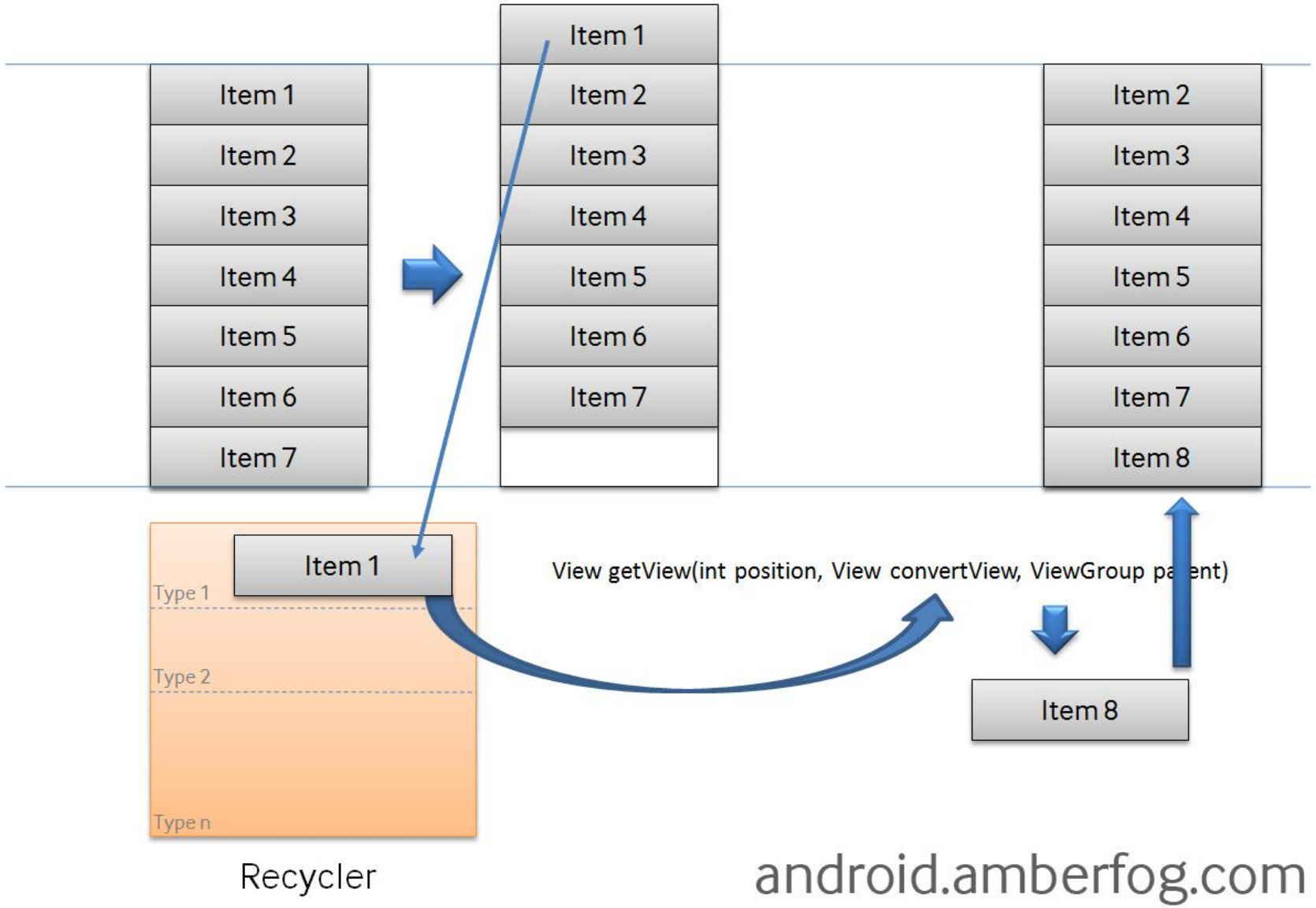
Populating the listview using ArrayAdapter



View recycling



- When using an adapter and a ListView, we need to make sure to understand how view recycling works.
- When your ListView is connected to an adapter, the adapter will instantiate rows until the ListView has been fully populated with enough items to fill the full height of the list.
- At that point, no additional row items are created in memory.


Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7
Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7
Item 8
Item 1
Type 1
Type 2
Type n
View getView(int position, View convertView, ViewGroup parent)
Item 8
Recycler
android.amberfog.com

A couple of reasons to recycle views

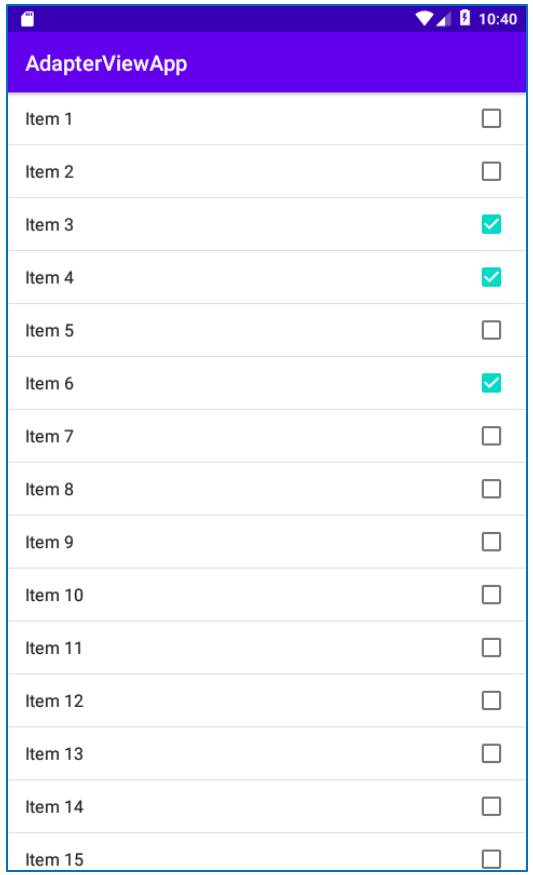
- Object creation is relatively expensive.
- Every additional object that is created needs to be dealt with by the garbage collection system.
- This is more important for more complex views, but inflating and laying out the view objects can be expensive.
- Most often, you are only making minor changes to the view in getView that won't affect the layout (e.g, setting text) so you might be able to avoid the layout overhead.
- Remember that Android is designed to be run in a resource constrained environment

ArrayAdapter

ListView using an ArrayAdapter

```
listView.adapter = ArrayAdapter( context: this,
    android.R.layout.simple_list_item_1, list)
```

```
val list = arrayListOf<String>()
for (item in 1..100){
    list.add("Item $item")
}
```



the second parameter for ArrayAdapter constructor can have below values.

- **simple_list_item_1** : Each list item is a TextView object.
- **simple_list_item_2** : Two TextView objects.
- **simple_list_item_checked** : Each list item is a checked checkbox.
- **simple_list_item_single_choice** : Display a radio button in the right of each list item. Even it is a radio button, it can be multiple checked if ListView's **android:choiceMode value is multipleChoice.**
- **simple_list_item_multiple_choice** : Display a checkbox in the right of each list item. Even it is a checkbox, it can be single checked if ListView's **android:choiceMode value is singleChoice.**

ListViewInKotlin using an ArrayAdapter

- Modify activity_main.xml


```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  tools:context=".MainActivity" >

  <ListView
    android:id="@+id/list_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
</LinearLayout>
```

ListViewInKotlin using an ArrayAdapter

- Modify MainActivity.kt

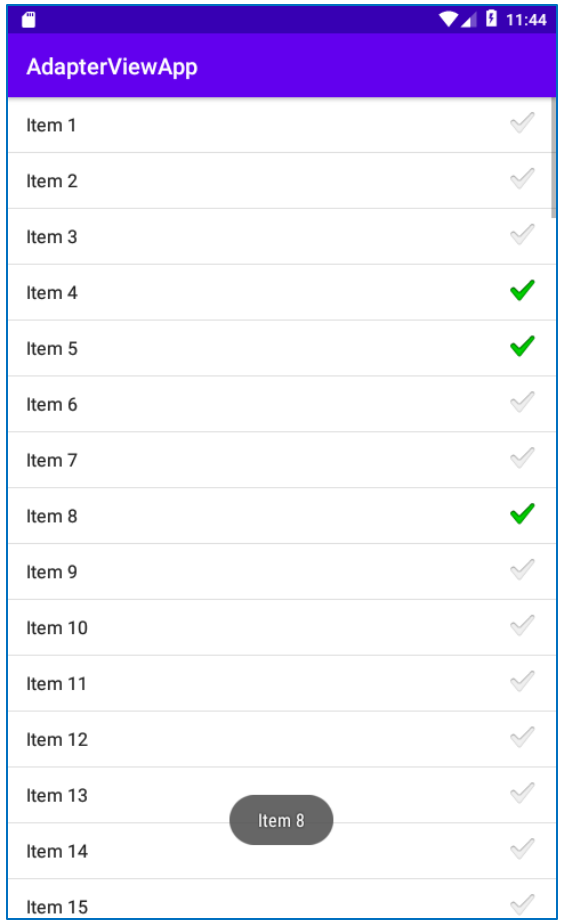

```
class MainActivity : AppCompatActivity() {
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    // Create data
    val users = arrayOf(
      "Virat Kohli", "Rohit Sharma", "Steve Smith",
      "Kane Williamson", "Ross Taylor", "Mario Rossi",
      "Giuseppe verdi", "Pippo Baudo",
      "Virat Kohli", "Rohit Sharma", "Steve Smith",
      "Kane Williamson", "Ross Taylor", "Mario Rossi",
      "Giuseppe verdi", "Pippo Baudo"
    )
    // pass data to the Adapter
    list_view.adapter = ArrayAdapter(this, android.R.layout.simple_list_item_1, users)
  }
}
```

Context, resource, Array<String!>

GridViewInKotlin using an ArrayAdapter


```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  tools:context=".MainActivity" >

  <GridView
    android:id="@+id/list_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:numColumns="auto_fit" />
</LinearLayout>
```

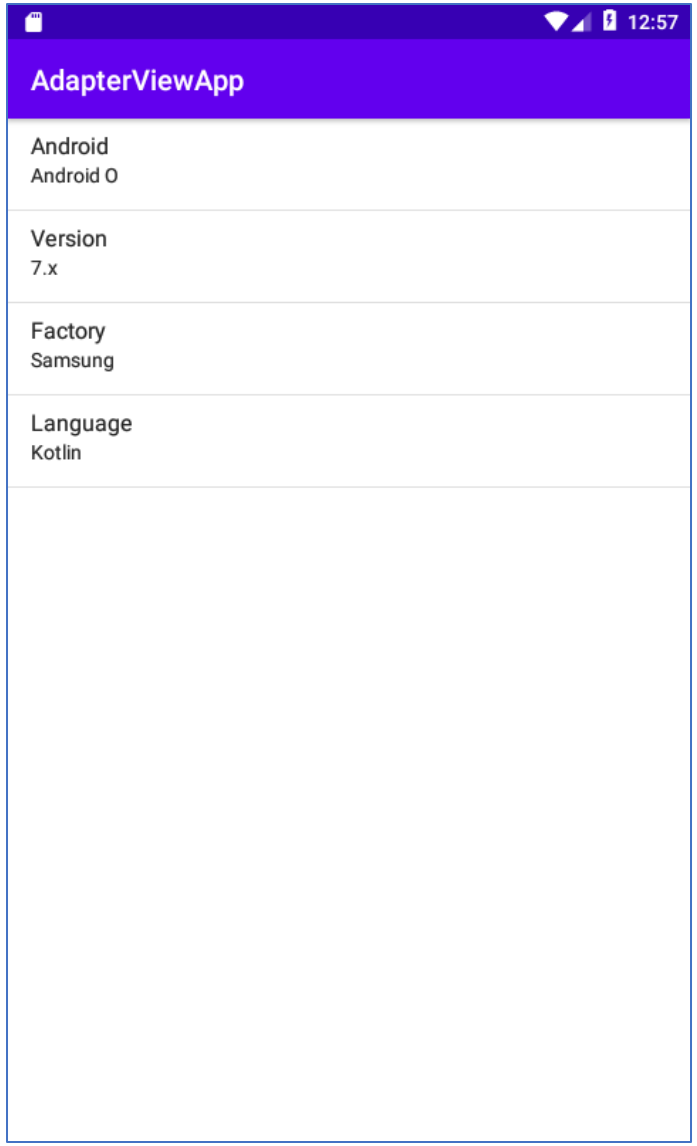
ListView OnItemClickListener



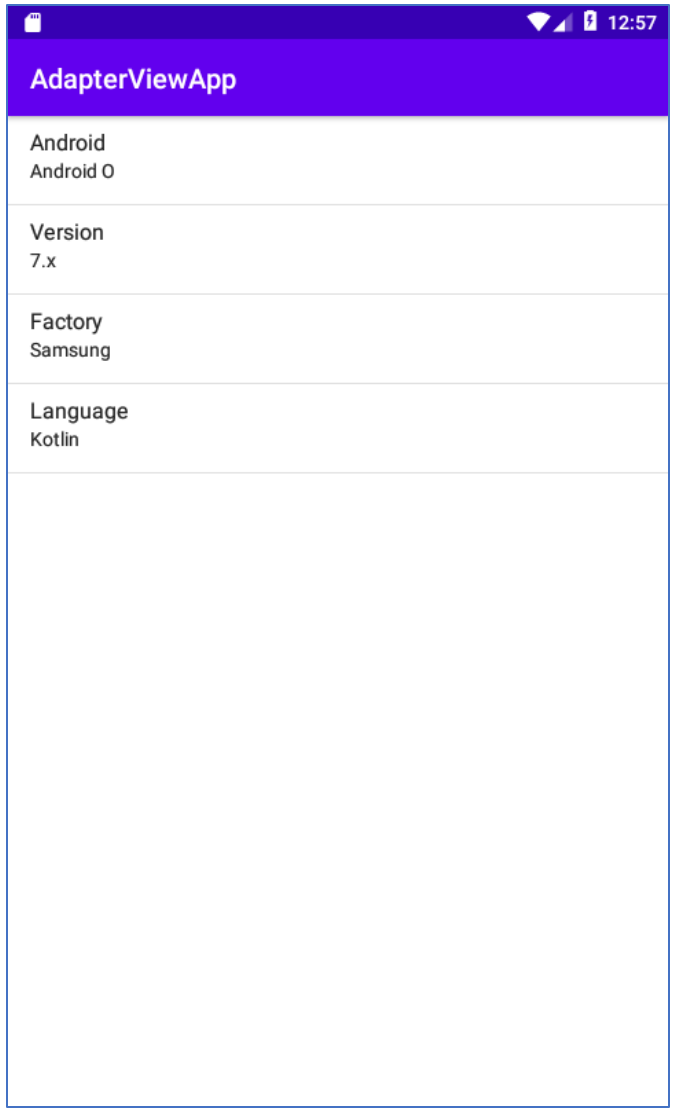

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    val list = arrayListOf<String>()
    for (item in 1..100){
        list.add("Item $item")
    }
    listView.adapter = ArrayAdapter( context: this,
        android.R.layout.simple_list_item_checked, list)

    listView.setOnItemClickListener {parent, view, position, id ->
        Toast.makeText( context: this, list[position], Toast.LENGTH_SHORT).show()
    }
}
```

```
<ListView
    android:id="@+id/listView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:choiceMode="multipleChoice"/>
```

SimpleAdapter

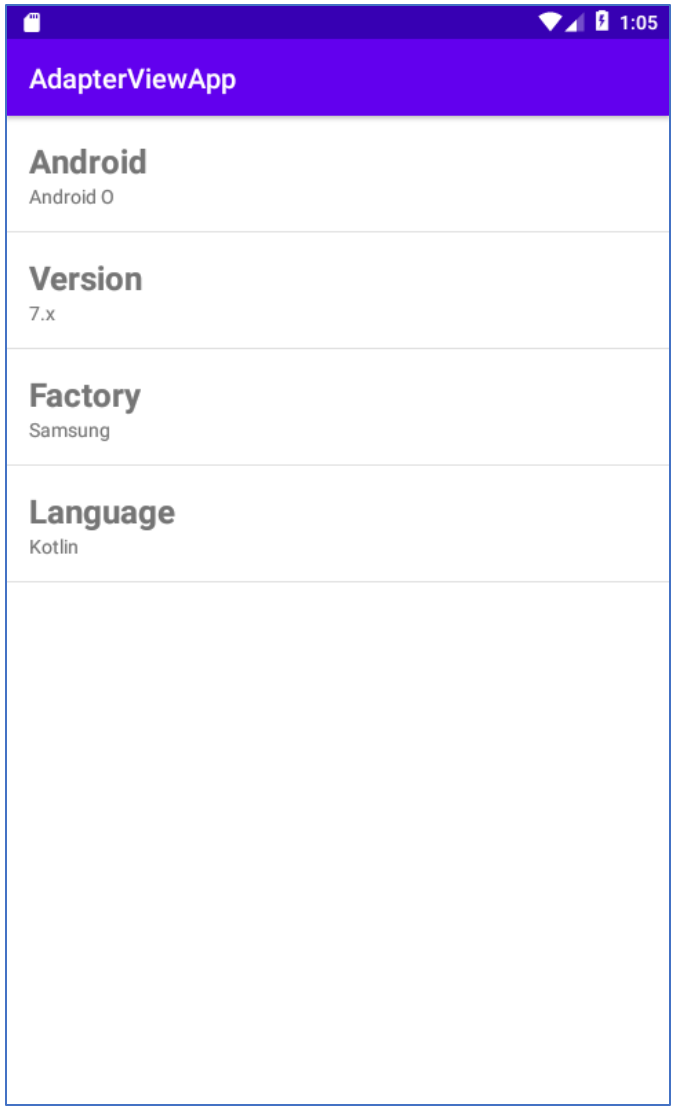


SimpleAdapter using android's layout




```kotlin
private val listTitle = arrayOf("Android", "Version", "Factory", "Language")
private val listDetails = arrayOf("Android O", "7.x", "Samsung", "Kotlin")
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    val data = ArrayList<HashMap<String, Any>>()
    for (i in listTitle.indices){
        val item = HashMap<String, Any>()
        item["title"] = listTitle[i]
        item["details"] = listDetails[i]
        data.add(item)
    }
    listView.adapter = SimpleAdapter( context: this, data,
        android.R.layout.simple_list_item_2,
        arrayOf("title", "details"),
        intArrayOf(android.R.id.text1, android.R.id.text2)
    )
```

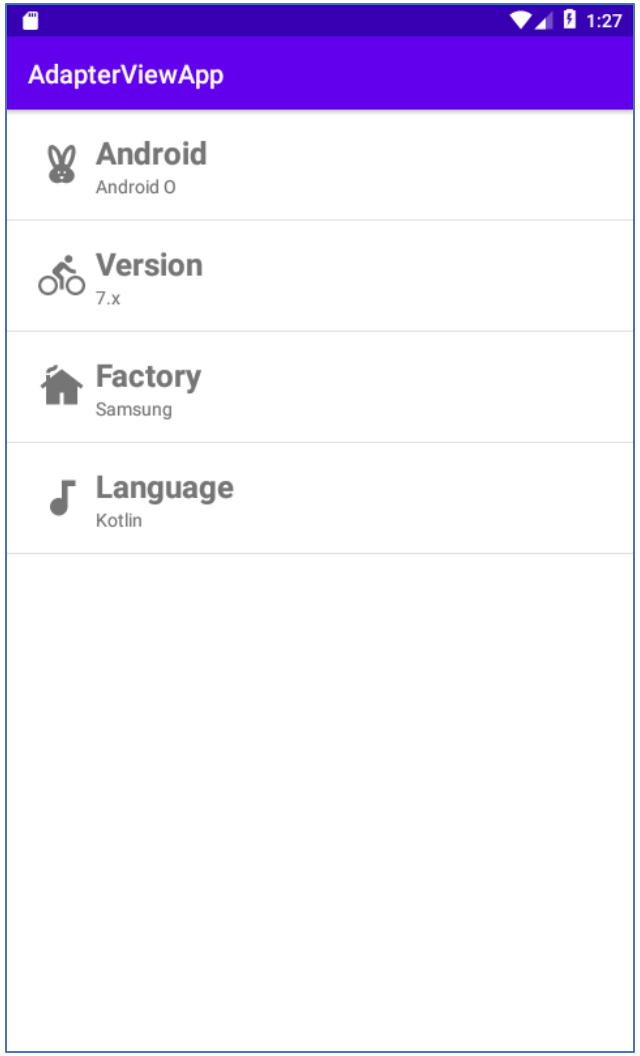
SimpleAdapter using our layout



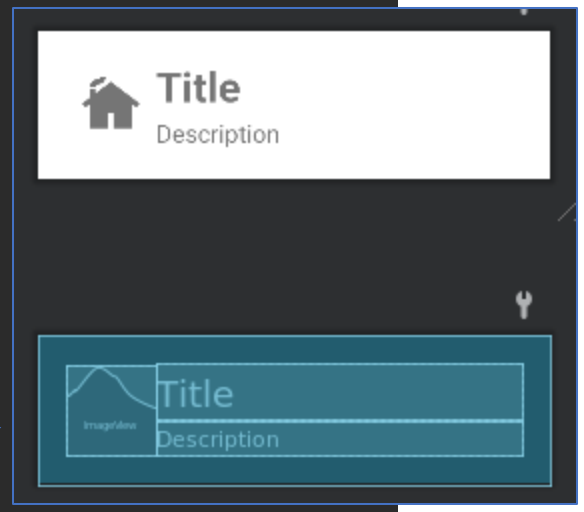

```kotlin
private val listTitle = arrayOf("Android", "Version", "Factory", "Language")
private val listDetails = arrayOf("Android O", "7.x", "Samsung", "Kotlin")
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    val data = ArrayList<HashMap<String, Any>>()
    for (i in listTitle.indices){
        val item = HashMap<String, Any>()
        item["title"] = listTitle[i]
        item["details"] = listDetails[i]
        data.add(item)
    }
    listView.adapter = SimpleAdapter( context: this, data,
        R.layout.list_row_items,
        arrayOf("title", "details"),
        intArrayOf(R.id.tvTitle, R.id.tvDesc)
    )
```

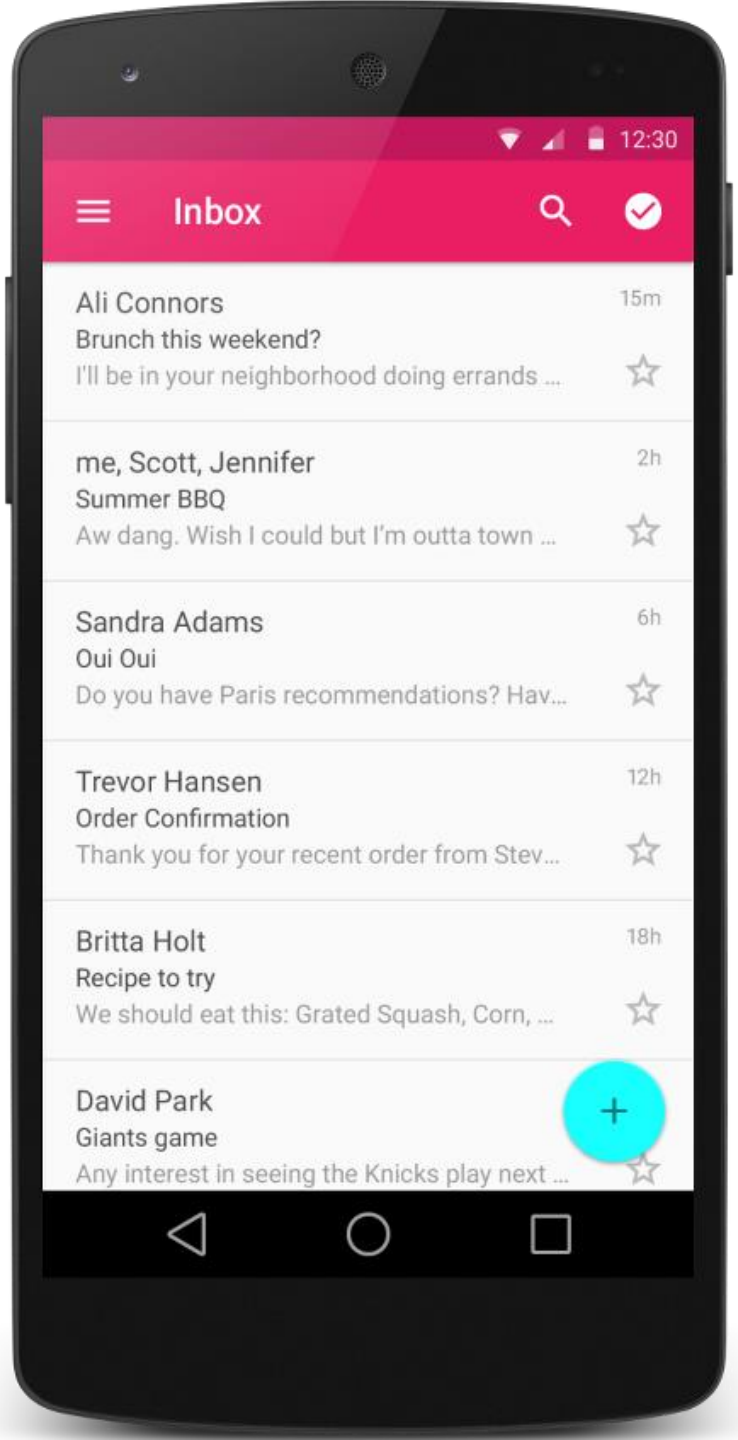
```xml
<TextView
    android:id="@+id/tvTitle"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textStyle="bold"
    android:textSize="24sp"
    android:text="Title" />
<TextView
    android:id="@+id/tvDesc"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Description" />
```

SimpleAdapter using our layout and images




```kotlin
private val listImgIds = arrayOf(R.drawable.ic_rabbit, R.drawable.ic_bycicle,
    R.drawable.ic_home, R.drawable.ic_music)
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    val data = ArrayList<HashMap<String, Any>>()
    for (i in listTitle.indices){
        val item = HashMap<String, Any>()
        item["title"] = listTitle[i]
        item["details"] = listDetails[i]
        item["image"] = listImgIds[i]
        data.add(item)
    }
    listView.adapter = SimpleAdapter( context: this, data,
        R.layout.list_row_items_3,
        arrayOf("image", "title", "details"),
        intArrayOf(R.id.imageView, R.id.tvTitle, R.id.tvDesc)
    )
```



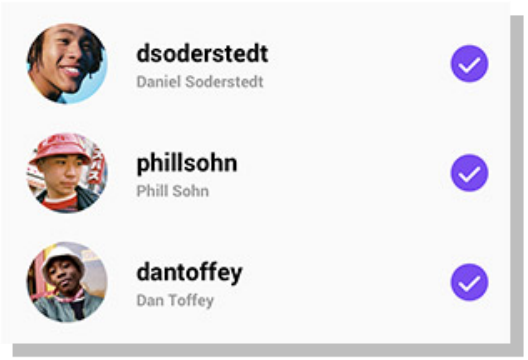


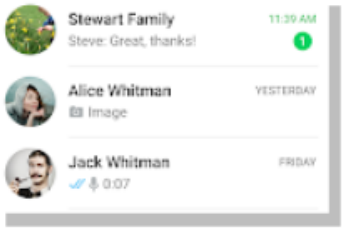
ListView and Custom Adapters

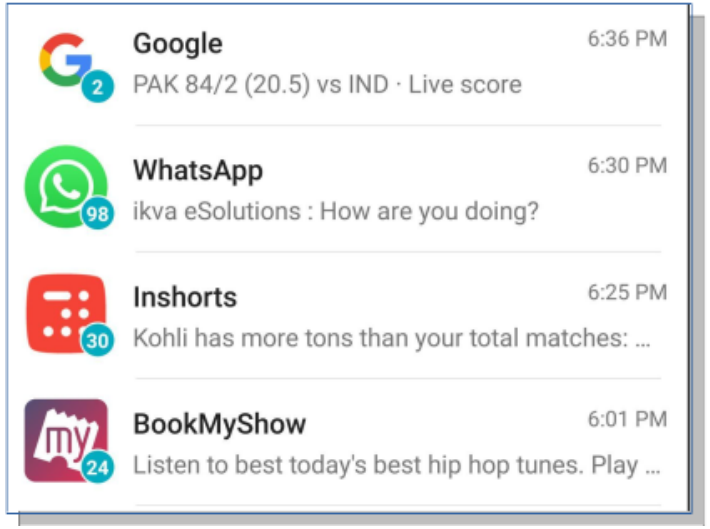
by extending BaseAdapter

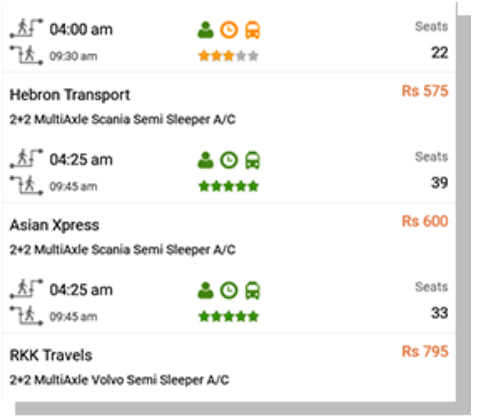
Customized ListView and GridView

- ListView provide a simple way to present scrolling lists of rows.
- Rows can be:
  - formatted with a built-in style
  - extensively customized.



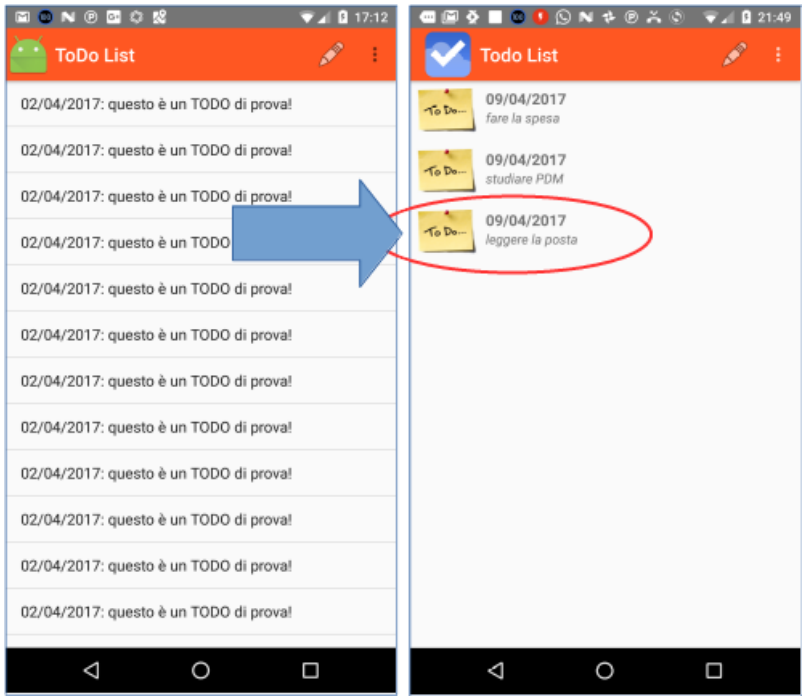






Customized adapter

- By default, the ArrayAdapter uses the toString() method of the items to populate a TextView within the layout you specify.
- If you need something different (e.g, a row consisting of different fields) you must customize the ArrayAdapter to populate the layout used for each View to represent the underlying array data.
- To do so, extend **BaseAdapter** with a type-specific variation, overriding the getView() method to assign object properties to layout Views.



Override: android.widget.Adapter.getView()

View getView(int position, View convertView, ViewGroup parent)

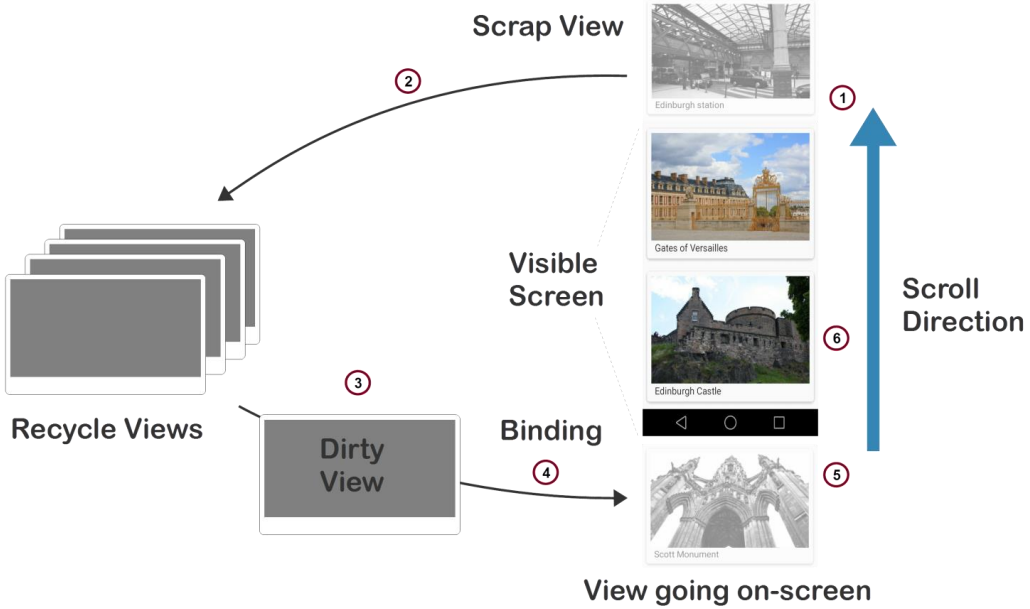
Get a View that displays the data at the specified position in the data set.

position
The position of the item within the adapter's data set of the item whose view we want.

convertView
The old view to reuse, if possible.

parent
The parent that this view will eventually be attached to.

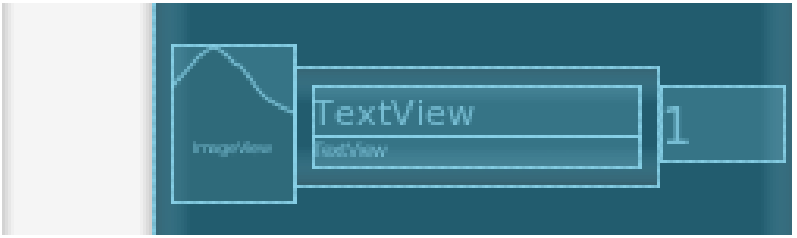
The role of **convertView** parameter

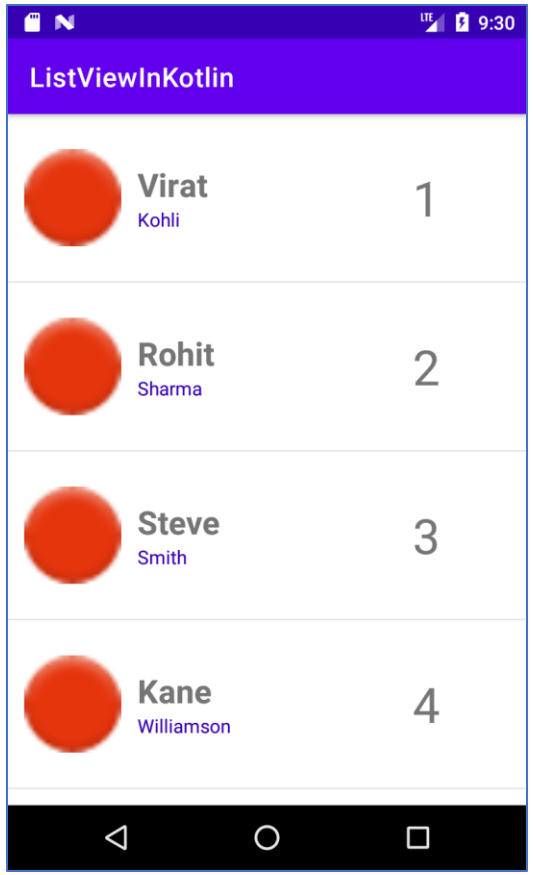


- Android calculates <u>how many item</u> of the list view <u>can be displayed</u> to fit the device screen at a given time.
- Suppose that it has room for 2 elements:
  - Accommodating the first 2 elements Android calls getView with convertView = null
    - You have to inflate a new view to crete the View to display.
  - If 2 elements are already displayed and you require to display a new one (e.g., scrolling the list), Android <u>calls getView providing as convertView the reference to the View that will be out of the screen</u>.
  - You should reuse this View so to minimize the inflate calls which are relatively slow.

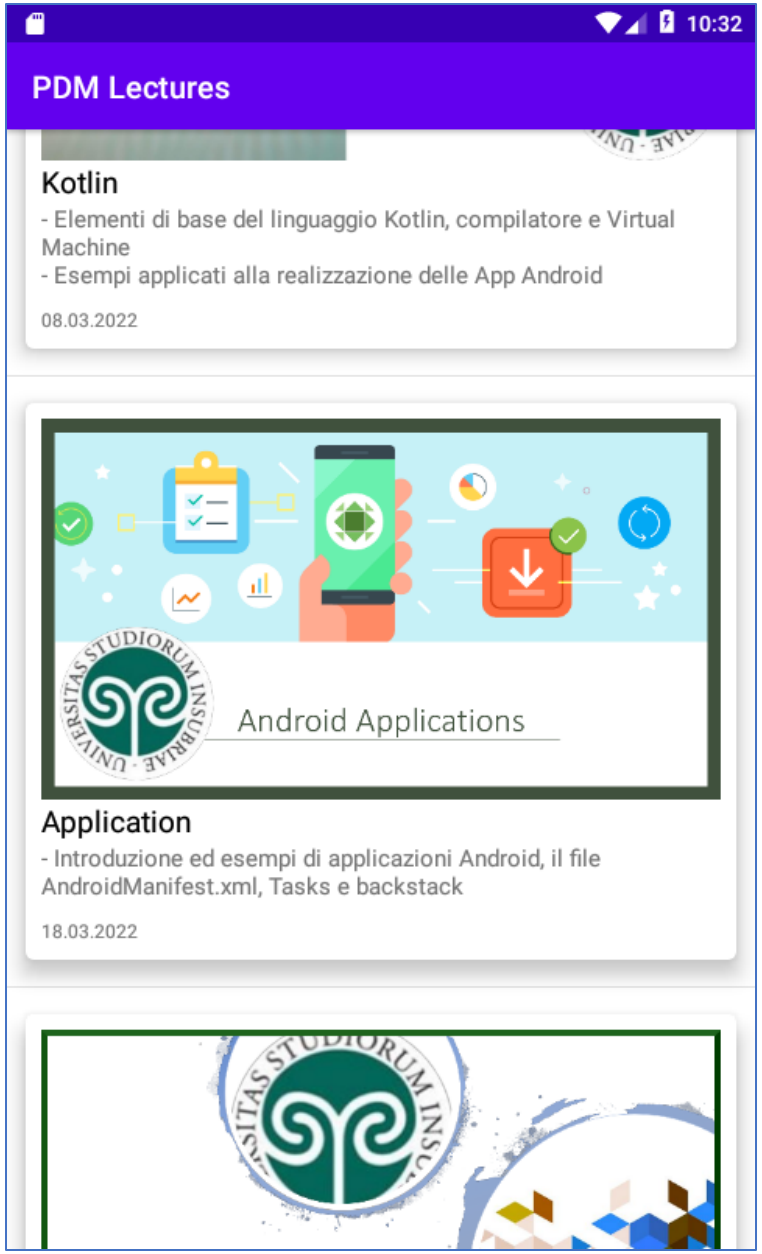
ListViewInKotlin with a Custom Adapter

- define the layout of a row



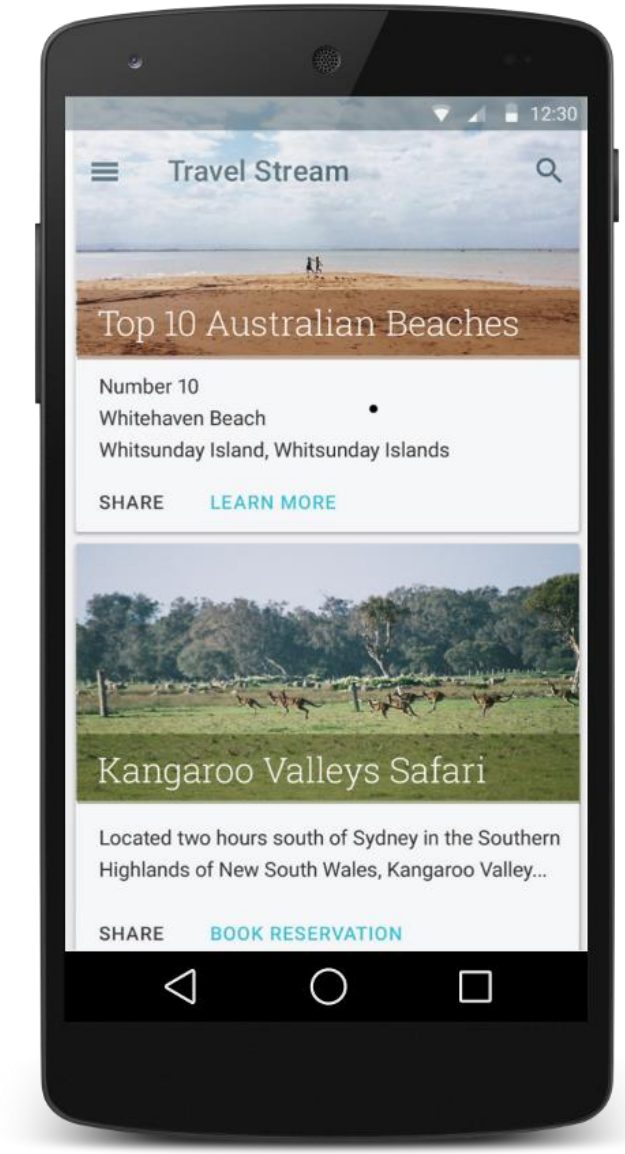


Custom Adapter Example



CardView

- an easy way for you to show information inside cards that have a consistent look across the platform.
- These cards have a default elevation above their containing view group, so the system draws shadows below them.
- Cards provide an easy way to contain a group of views while providing a consistent style for the container.



```
class DataSource{
  companion object{
    fun createDataSet(): ArrayList<Lecture>{
      val list = ArrayList<Lecture>()
```

```
data class Lecture (
  var title: String,
  var topics: String,
  var image: String,
  var date: String
) {
  override fun toString(): String {
    return "Lecture(title='$title', image='$image', date='$date')"
  }
}
```

```
list.add(Lecture(
"Presentazione del corso",
"- Presentazione del corso\n" +
"- Introduzione al sistema Ster (non fa parte del materiale didattico)\n" +
"- Questionario \"COSA MI ASPETTO DA QUESTO CORSO\" e analisi delle risposte.",
"https://raw.githubusercontent.com/ignaziogallo/PDM/master/images/K00-Presentazione-corso.png",
"22.02.2022"
))
list.add(Lecture(
"Introduzione",
" - Panoramica introduttiva agli argomenti principali trattati nel corso\n" +
"- Introduzione ad AndroidStudio e creazione di una prima App.",
"https://raw.githubusercontent.com/ignaziogallo/PDM/master/images/K01-Introduzione.png",
"25.02.2022"
))
```

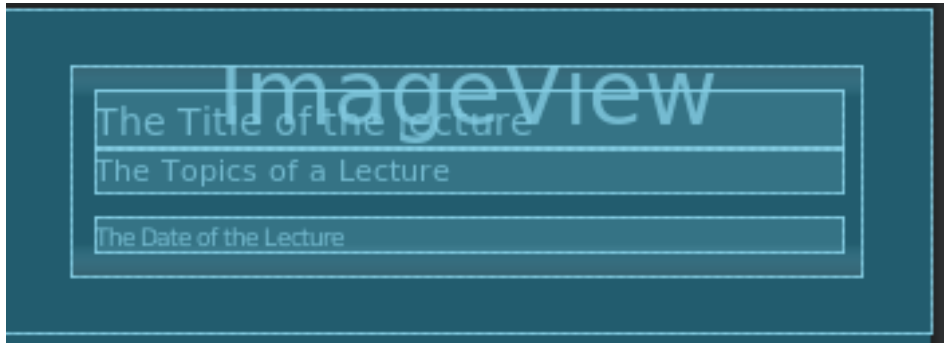
```
class DataSource{
companion object{
fun createDataSet(): ArrayList<Lecture>{
val list = ArrayList<Lecture>()

list.add(
Lecture(
"Hardware and Android Operative System",
"- Caratteristiche tipiche dell'hardware tipico dei disporitivi
mobili.\n" +
"- Tecnologia RISC e CISC\n" +
"- Il Sistema Operativo Android. ",
"https://raw.githubusercontent.com/ignaziogallo/PDM/maste
r/images/K02-hw_and_SO_Architecture.png",
"01.03.2022"
)
)
list.add(
Lecture(
"AndroidStudio",
"AndroidStudio.",
"https://raw.githubusercontent.com/ignaziogallo/PDM/maste
r/images/K03-AndroidStudio.png",
"01.03.2022"
)
)
```

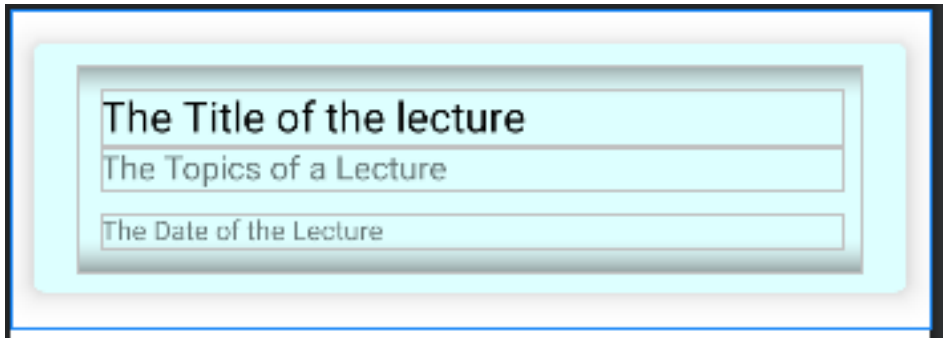
```
class DataSource{
companion object{
fun createDataSet(): ArrayList<Lecture>{
val list = ArrayList<Lecture>()

list.add(
Lecture(
"GIT",
"Git come sistema di versioning sia indipendente che integrato
in AndroidStudio",
"https://raw.githubusercontent.com/ignaziogallo/PDM/master
/images/K04-GIT.png",
"01.03.2022"
)
)
list.add(
Lecture(
"Gradle",
"- Ant, Maven e Gradle come tools di build automation.\n" +
"- Il linguaggio Groovy per la creazione di script di Gradle.\n" +
"- Esercizi su Gradle e Groovy",
"https://raw.githubusercontent.com/ignaziogallo/PDM/master
/images/K05-Gradle.png",
"04.03.2022"
)
)
```

```
class DataSource{
companion object{
fun createDataSet(): ArrayList<Lecture>{
val list = ArrayList<Lecture>()

list.add(
Lecture(
"Kotlin",
"- Elementi di base del linguaggio Kotlin, compilatore e Virtual
Machine\n" +
"- Esempi applicati alla realizzazione delle App Android",
"https://raw.githubusercontent.com/ignaziogallo/PDM/master
/images/K06-kotlin-1.png",
"08.03.2022"
)
)
list.add(
Lecture(
"Application",
"- Introduzione ed esempi di applicazioni Android, il file
AndroidManifest.xml, Tasks e backstack",
"https://raw.githubusercontent.com/ignaziogallo/PDM/master
/images/K07-Application.png",
"18.03.2022"
)
)
```

```
class DataSource{
companion object{
fun createDataSet(): ArrayList<Lecture>{
val list = ArrayList<Lecture>()

list.add(
Lecture(
"Activity",
"- Activity e Intent.\n" +
"- Intent espliciti ed impliciti\n" +
"- startActivityForResults e registerActivityForResults\n" +
"- Ciclo di vita di una Activity\n" +
"- Esempi",
"https://raw.githubusercontent.com/ignaziogallo/PDM/mast
er/images/K08-Activity.png",
"22.03.2022"
)
)
return list
}
}
}
```

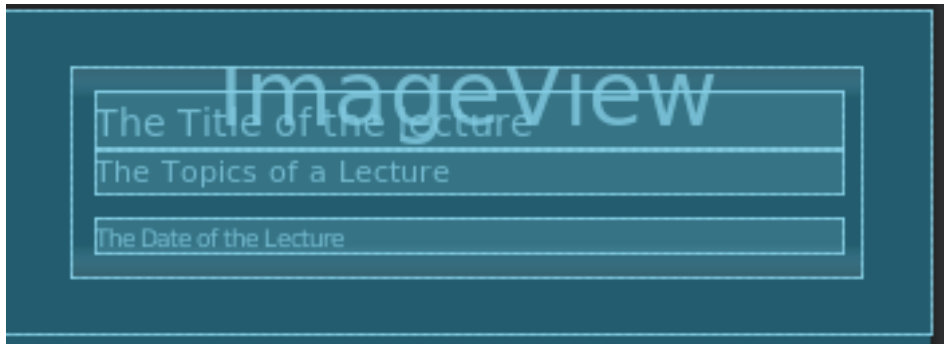
CardView

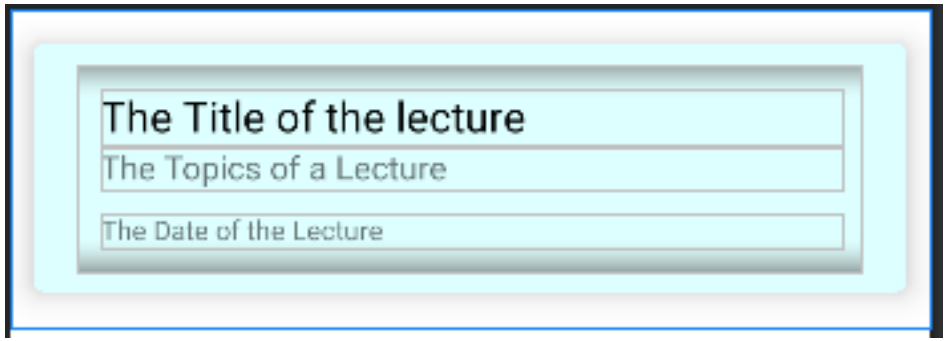





```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:cardElevation="10dp"
    app:cardCornerRadius="5dp"
    app:cardUseCompatPadding="true">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="10dp"
        android:id="@+id/container1"
        >
        <ImageView...>
        <TextView...>
        <TextView...>
        <TextView...>
    </LinearLayout>
</androidx.cardview.widget.CardView>
```

CardView






```
<ImageView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/lecture_image"
    android:layout_margin="0dp"
    android:adjustViewBounds="true"
    android:scaleType="fitXY"
    android:padding="0dp"
    />
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/lecture_title"
    android:text="The Title of the lecture"
    android:textColor="#000"
    android:textSize="19sp"
    />
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/lecture_topics"
    android:text="The Topics of a Lecture"
    android:textSize="15sp"
    />
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/lecture_date"
    android:text="The Date of the Lecture"
    android:textSize="12sp"
    android:layout_marginTop="10dp"
    />
```

Build.gradle

```
dependencies {
implementation 'androidx.core:core-ktx:1.7.0'
implementation 'androidx.appcompat:appcompat:1.4.1'
implementation 'com.google.android.material:material:1.5.0'
implementation 'androidx.constraintlayout:constraintlayout:2.1.3'
testImplementation 'junit:junit:4.13.2'
androidTestImplementation 'androidx.test.ext:junit:1.1.3'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'

implementation 'com.github.bumptech.glide:glide:4.13.0'
annotationProcessor 'com.github.bumptech.glide:compiler:4.13.0'
}
```

https://github.com/bumptech/glide

glide

Glide is a fast and efficient open-source media management and image loading **framework** for Android that wraps media decoding, memory and disk caching, and resource pooling into a simple and easy to use interface.

INTERNET permission

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="it.insubria.recyclerexample">

<uses-permission android:name="android.permission.INTERNET"/>

<application...
</application>

</manifest>
```

MainActivity Layout

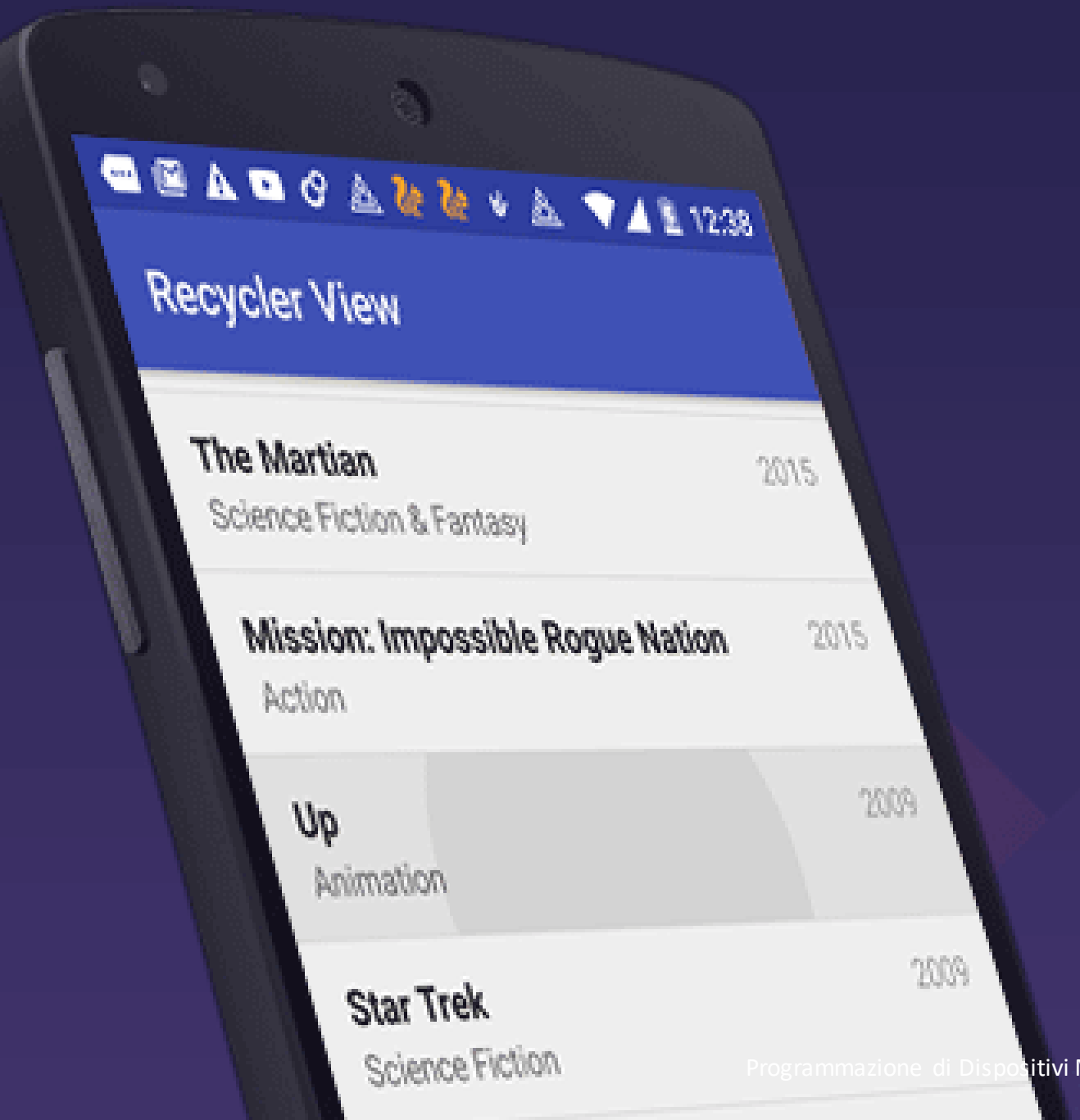
```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ListView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/list_view"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        />
</androidx.constraintlayout.widget.ConstraintLayout>
```

```kotlin
class MainActivity : AppCompatActivity() {
  private lateinit var lectureAdapter: LectureListAdapter

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main_list)
    lectureAdapter = LectureListAdapter(this, DataSource.createDataSet())
    list_view.adapter = lectureAdapter
  }
}
```

LectureListAdapter

```
class LectureListAdapter(private val context: Context, private var lectureList: List<Lecture>) : BaseAdapter() {
    override fun getCount(): Int {...}
    override fun getItem(position: Int): Any {...}
    override fun getItemId(position: Int): Long {...}
    override fun getView(position: Int, convertView: View?, parent: ViewGroup?): View? {
        var newView = convertView
        if (newView == null)
            newView = LayoutInflater.from(context).inflate(
                R.layout.layout_lecture_list_item, parent, attachToRoot: false)
        val lecture = lectureList[position]
        val requestOptions = RequestOptions()
            .placeholder(R.drawable.ic_launcher_background)
            .error(R.drawable.ic_launcher_background)
        Glide.with(context)
            .applyDefaultRequestOptions(requestOptions)
            .load(lecture.image)
            .into(newView!!.lecture_image)
        newView.lecture_title?.text = lecture.title
        newView.lecture_date?.text = lecture.date
        newView.lecture_topics?.text = lecture.topics
        return newView
    }
}
```

```
override fun getCount(): Int {
    return lectureList.count()
}
override fun getItem(position: Int): Any {
    return lectureList[position]
}
override fun getItemId(position: Int): Long {
    val lecture: Lecture = lectureList[position]
    return (lecture.date + lecture.title + lecture.topics).hashCode().toLong()
}
```



Android Working With

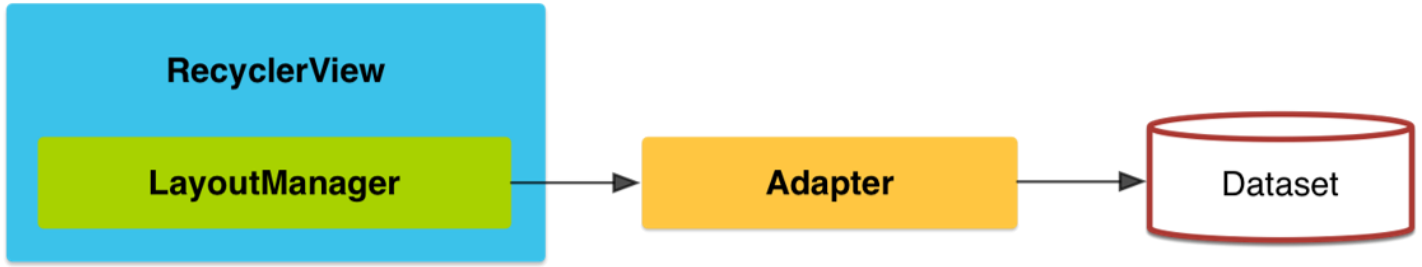
RecyclerView

Lists using RecyclerView

- The **RecyclerView** widget is a more advanced and flexible version of **ListView**.
- This widget is a container for displaying large data sets that can be scrolled very efficiently by maintaining a limited number of views.
- Use the RecyclerView widget when you have data collections whose elements change at runtime based on user action or network events.
- The RecyclerView class simplifies the display and handling of large data sets by providing:
  - Layout managers for positioning items
  - Default animations for common item operations, such as removal or addition of items

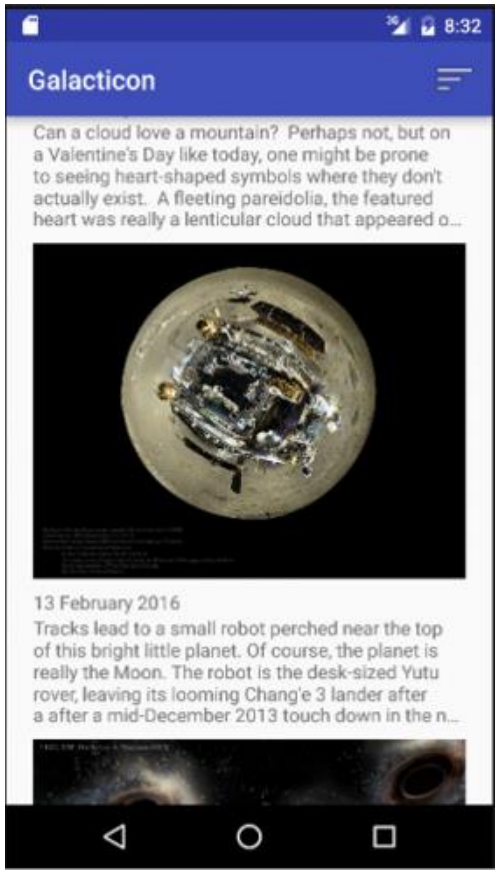
RecyclerView: layout managers

- You also have the flexibility to define custom layout managers and animations for RecyclerView widgets.
- RecyclerView provides these built-in layout managers:
  - **LinearLayoutManager** shows items in a vertical or horizontal scrolling list.
  - **GridLayoutManager** shows items in a grid.
  - **StaggeredGridLayoutManager** shows items in a staggered grid.



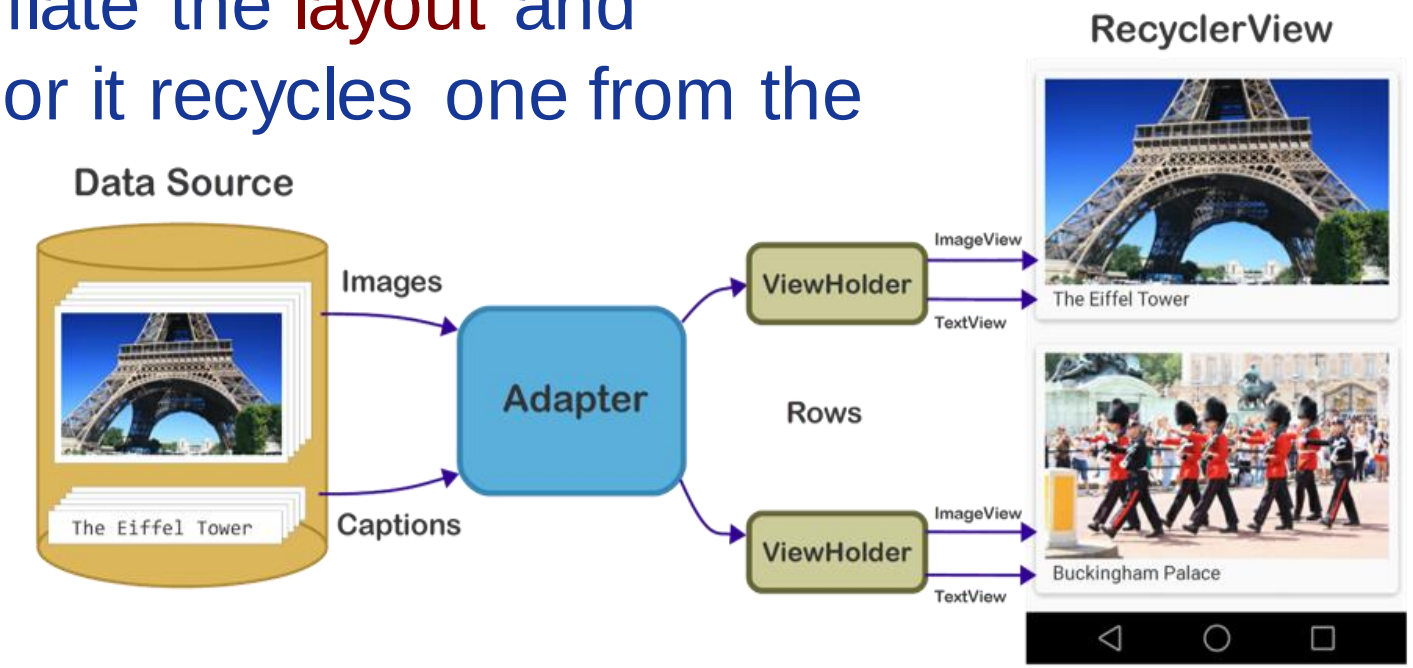
RecyclerView: layout managers and more..

- A RecyclerView can be thought of as a combination of a ListView and a GridView.
- You can also create your own LayoutManagers
- There are extra features that separate your code into maintainable components even as they enforce memory-efficient design patterns.



RecyclerView: ViewHolders

- RecyclerView uses an **Adapter** to act as a data source.
- You have to create **ViewHolders** to keep references in memory.
- When you need a new view, it either creates a new ViewHolder object to inflate the layout and hold those references, or it recycles one from the existing stack.
- Now you know why it's called a RecyclerView!



RecyclerView Example

- Create an App that shows PDM lectures
- RecyclerView to show the items (CardView)
- RecyclerView.ViewHolder
- RecyclerView.Adapter

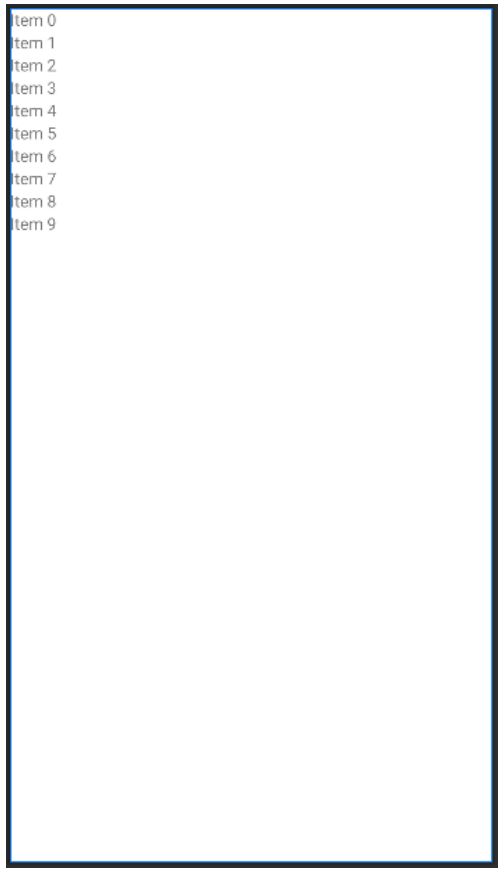
```kotlin
class MainActivity : AppCompatActivity() {
  private lateinit var lectureAdapter: LectureRecyclerAdapter

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

    initRecyclerView()
    addDataSet()
  }

  private fun initRecyclerView(){
    recycler_view.layoutManager =
                  LinearLayoutManager(this@MainActivity)
    lectureAdapter = LectureRecyclerAdapter()
    recycler_view.adapter = lectureAdapter
  }
  private fun addDataSet(){
    val data = DataSource.createDataSet()
    lectureAdapter.submitList(data)
  }
}
```

```kotlin
data class Lecture (
  var title: String,
  var topics: String,
  var image: String,
  var date: String
) {
  override fun toString(): String {
    return "Lecture(title='$title', image='$image', date='$date')"
  }
}
```

```kotlin
class LectureRecyclerAdapter : RecyclerView.Adapter<RecyclerView.ViewHolder>(){
    private var items: List<Lecture> = ArrayList()

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): RecyclerView.ViewHolder {
        return LectureViewHolder(
            LayoutInflater.from(parent.context).inflate(R.layout.layout_lecture_list_item, parent, attachToRoot: false)
        )
    }
    override fun onBindViewHolder(holder: RecyclerView.ViewHolder, position: Int) {
        when(holder) {
            is LectureViewHolder -> {
                holder.bind(items.get(position))
            }
        }
    }
    override fun getItemCount(): Int {
        return items.size
    }
    fun submitList(lectureList: List<Lecture>){
        items = lectureList
    }

    class LectureViewHolder(itemView: View): RecyclerView.ViewHolder(itemView){...}
}
```

```kotlin
class LectureRecyclerAdapter : RecyclerView.Adapter<RecyclerView.ViewHolder>(){
    private var items: List<Lecture> = ArrayList()

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): RecyclerView.ViewHolder {...}
    override fun onBindViewHolder(holder: RecyclerView.ViewHolder, position: Int) {...}
    override fun getItemCount(): Int {...}
    fun submitList(lectureList: List<Lecture>){...}

    class LectureViewHolder(itemView: View): RecyclerView.ViewHolder(itemView){
        val lecture_image = itemView.lecture_image
        val lecture_title = itemView.lecture_title
        val lecture_date = itemView.lecture_date
        val lecture_topics = itemView.lecture_topics

        fun bind(lecture: Lecture){
            val requestOptions = RequestOptions()
                .placeholder(R.drawable.ic_launcher_background)
                .error(R.drawable.ic_launcher_background)

            Glide.with(itemView.context)
                .applyDefaultRequestOptions(requestOptions)
                .load(lecture.image)
                .into(lecture_image)
            lecture_title.setText(lecture.title)
            lecture_date.setText(lecture.date)
            lecture_topics.setText(lecture.topics)
        }
    }
}
```

RecyclerView




```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    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:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/recycler_view"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        />

</androidx.constraintlayout.widget.ConstraintLayout>
```