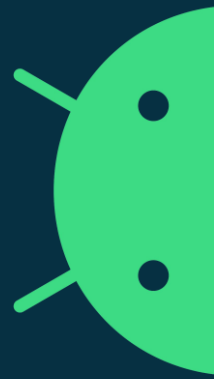
Lab.
Pemrograman Mobile



android

Pertemuan 6

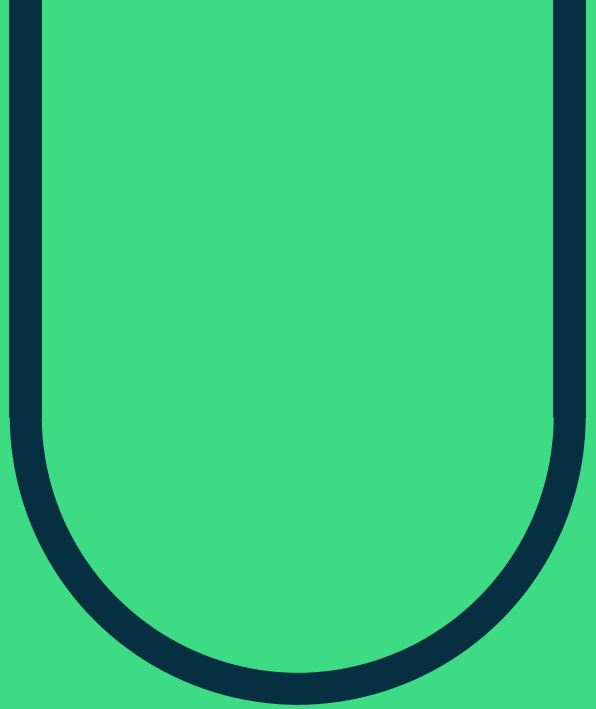
android

Learning Objectives

- RecyclerView

android

Pert. 6
RecycleView



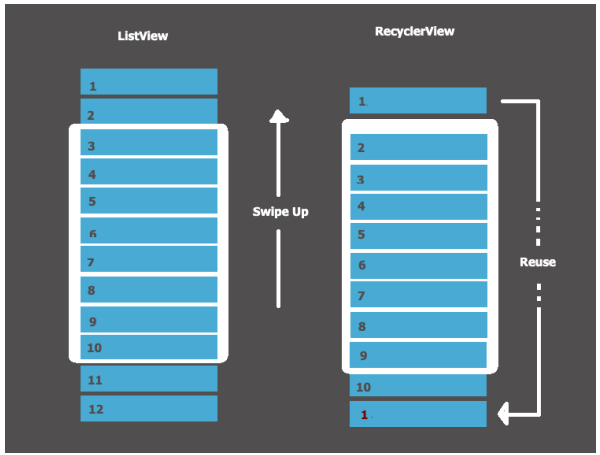
android

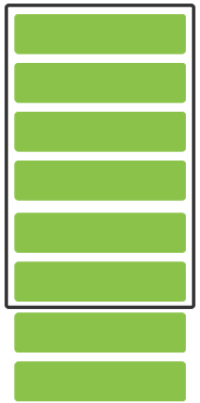
RecyclerView

A **RecyclerView** is an advanced version of **ListView** with improved performance. When you have a long list of items to show you can use **RecyclerView**. It has the ability to reuse its views. In **RecyclerView** when the **View** goes out of the screen or not visible to the user it won't destroy it, it will reuse these views. This feature helps in reducing power consumption and providing more responsiveness to the application.

android

The operation of **RecyclerView** can be schematically represented as follows. The visible list items are displayed on the screen. When you scroll the list, when the top item moves off the screen and becomes invisible, its contents are cleared. In this case, the “clean” element itself is placed at the bottom of the screen and filled with new data, in other words, it is reused, hence the name **Recycle**.





android

Using the **RecyclerView** requires configuring / implementing the following components:

- **RecyclerView**: manages everything. It's mostly pre-written by Android. We provide the components and configuration.

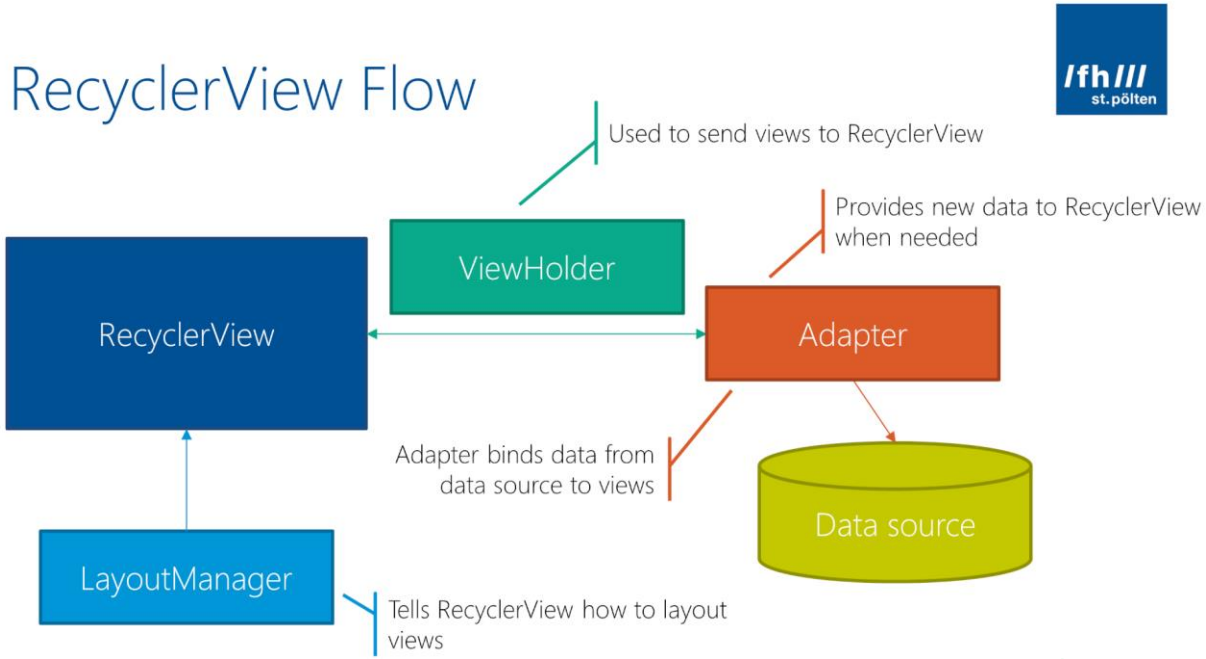
- **Adapter**: we will spend most of our time coding this class. It connects to the data source. When instructed by the **RecyclerView**, it creates or updates the individual items in the list.

- **ViewHolder**: a simple class that assigns / updates the data in the view items. When a view is re-used, the previous data is overwritten.

- **Data source**: anything you like – from a simple array up to a full-blown data source. Your Adapter interacts with it.

- **LayoutManager**: is responsible for placing all the individual view items on the screen and making sure they get the screen-space they need.

RecyclerView Flow



android

Setup RecyclerView

First, add the **RecyclerView** dependency in build.gradle (module) file by writing this line of code under depedencies block. Then re-sync the gradle. x.x.x is the version of **RecyclerView**.

```
dependencies {
...
    implementation("androidx.recyclerview:recyclerview:x.x.x")
...
}
```

Create a RecyclerView Layout

Then, create a layout as you want it to be. It is a single item layout that we will use in **RecyclerView**.

Create a new Data Class

Create a new data class that we will use data of custom generic to pass in the list that will be shown in RecyclerView.

```
@Parcelize
data class CatItem(
    val catImage : Int,
    val catName : String
) : Parcelable
```

Create a new Data Class

Create a new data class that we will use data of custom generic to pass in the list that will be shown in RecyclerView.

```
@Parcelize
data class CatItem(
    val catImage : Int,
    val catName : String
) : Parcelable
```

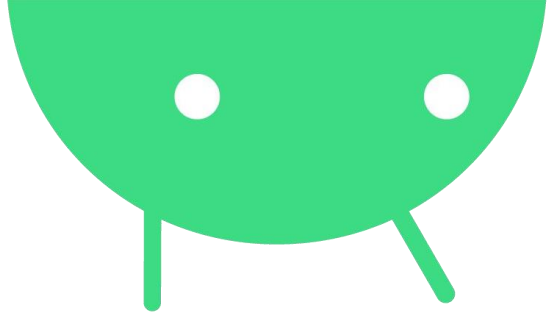
Create the Adapter Class

Create an Adapter class for the recycler view. Using View binding we use the generated class of the layout we've made before to add data and view in the recycler view of MainActivity.kt in our case.

```
class CatListAdapter(
    private val cats: ArrayList<CatItem>,
    val itemClickListener: (CatItem) -> Unit
): RecyclerView.Adapter<CatListAdapter.CatViewHolder>() {

    // write the code to implement RecyclerView Adapter

}
```

Initialize the Adapter in Main Class

```
class MainActivity : AppCompatActivity() {
...
   override fun onCreate(savedInstanceState: Bundle?) {
       super.onCreate(savedInstanceState)
       ...
       // initialize the adapter,
          // and pass the required argument
       val catAdapter = CatListAdapter(catList)

       // attach adapter to the recycler view
       binding.rvCats.adapter = catAdapter
       ...
   }
...
}
```



Any Question?