



**BINAR**  
ACADEMY

# RecyclerView

**Gold** - Chapter 4 - Topic 3

---

**Selamat datang di Chapter 4 Topic 3 *online course*  
Android Developer dari Binar Academy!**





# Hai teman-teman ✨

Gimana nih di topik sebelumnya? Makin jago kan? 😊

Pada topik sebelumnya, udah belajar tentang Dialog di Android. Di topik kali ini, kita eksplor tentang tata cara membuat tampilan **RecyclerView**. Caranya mirip-mirip seperti pas kita belajar buat layout. Jadi, tampilan ini pasti nggak asing di mata kita.

Kayak gimana sih? Yuk, lanjut!





**Detailnya, kita bakal bahas hal-hal berikut ini:**

- Apa itu RecyclerView
- Penggunaan RecyclerView
- DiffUtil untuk RecyclerView

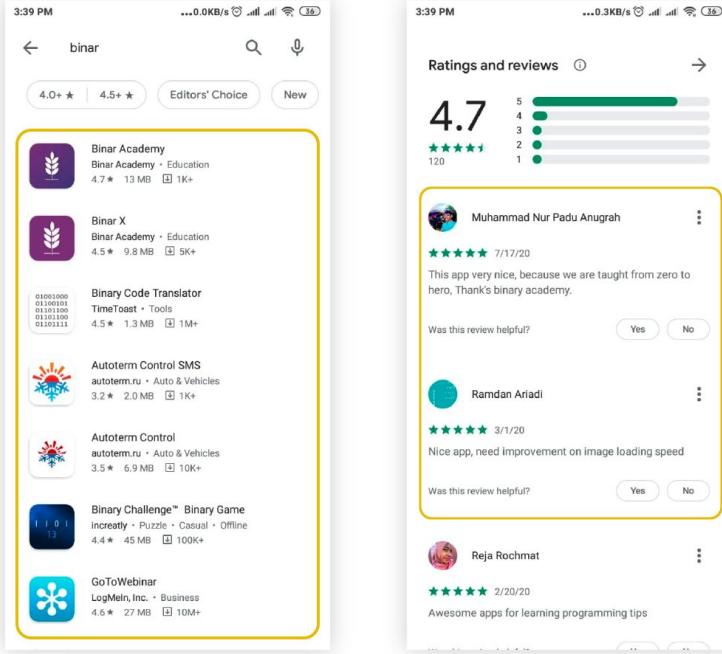




**Tak kenal maka tak sayang.**

**Sebelum pakai RecyclerView, ada baiknya kalau kita kenalan dulu sama si dia~**

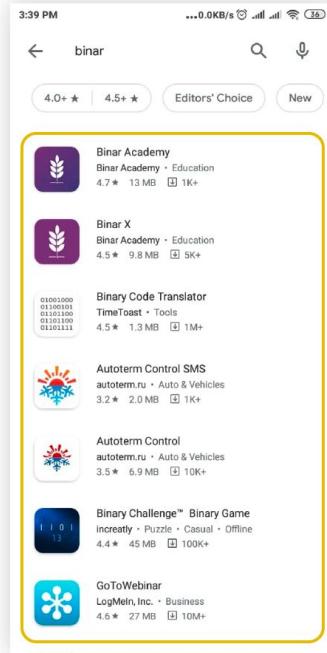
**Apa itu RecyclerView?**



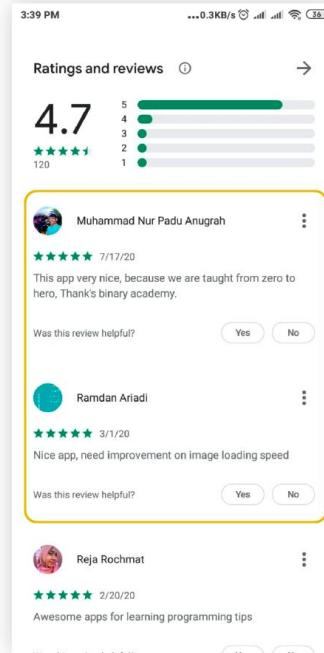
**Recycler View** adalah komponen View pada Android yang memungkinkan kita untuk **menampilkan daftar** yang kita inginkan pada aplikasi yang akan kita buat.



## Apa Itu RecyclerView?



Hasil Pencarian App



Daftar Ulasan App  
Google Play Store

Contohnya kayak gambar di samping ini nih!

Kalo diperhatiin, ternyata kita sering liat penggunaan **RecyclerView** di aplikasi yang kita pake sehari-hari, kan?

Hayoo.. siapa yang belum pernah liat dua gambar di samping?



# Tampilan daftar bisa itu bisa dibuat menggunakan tiga komponen utama dari RecyclerView, yakni :

RecyclerView

Berfungsi untuk menampilkan daftar beserta layout XML yang akan menjadi tampilan setiap baris daftar yang kita buat.

Adapter

Berfungsi untuk memproses data-data yang akan ditampilkan dalam daftar yang akan dibuat.

LayoutManager

Berfungsi untuk mengatur tata letak daftar seperti apa yang akan kita tampilkan.



**Ada RecyclerView buat menampilkan daftar, Adapter buat memproses data dan LayoutManager untuk tata letak.**

Nah, tiga sekawan udah kita resapi. Biar makin familiar, kita coba praktik aja yuk!

✨**Cara ngebuat RecylerView.**✨



### Yuk kita buat RecyclerView!

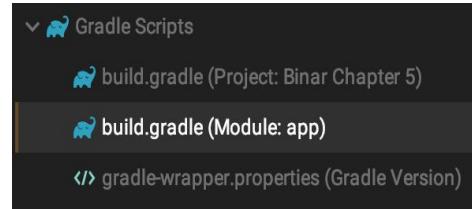
Gampang! Kita buat dengan 6 langkah :

**Pertama**, sebelum kita menggunakan RecyclerView, pastiin project kita udah **menambahkan dependency RecyclerView** di file Gradle level app.





**Kedua**, buka build.gradle (Module: app)

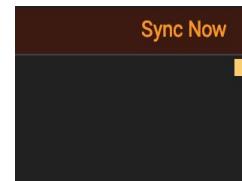


**Ketiga**, pada bagian *dependencies*, tambahin :

```
implementation 'androidx.recyclerview:recyclerview:version_code'
```

```
// RecyclerView  
implementation 'androidx.recyclerview:recyclerview:1.2.0'
```

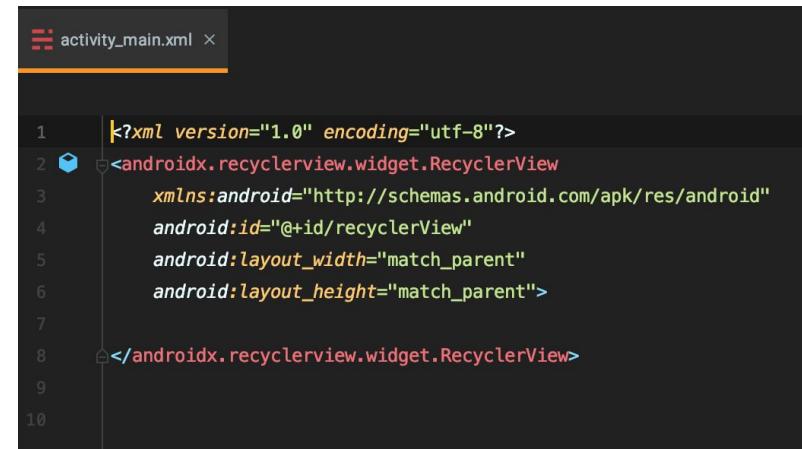
**Keempat**, klik “Sync Now” yang muncul pada bagian atas kanan jendela build.gradle. Lalu tunggu sampai proses sync selesai.





**Kelima**, setelah dependency ditambahin, kita bisa menggunakan View RecyclerView di Layout XML kita!

**Keenam**, dan terakhir, jangan lupa untuk memberikan ID pada RecyclerView yang akan dibuat ya!

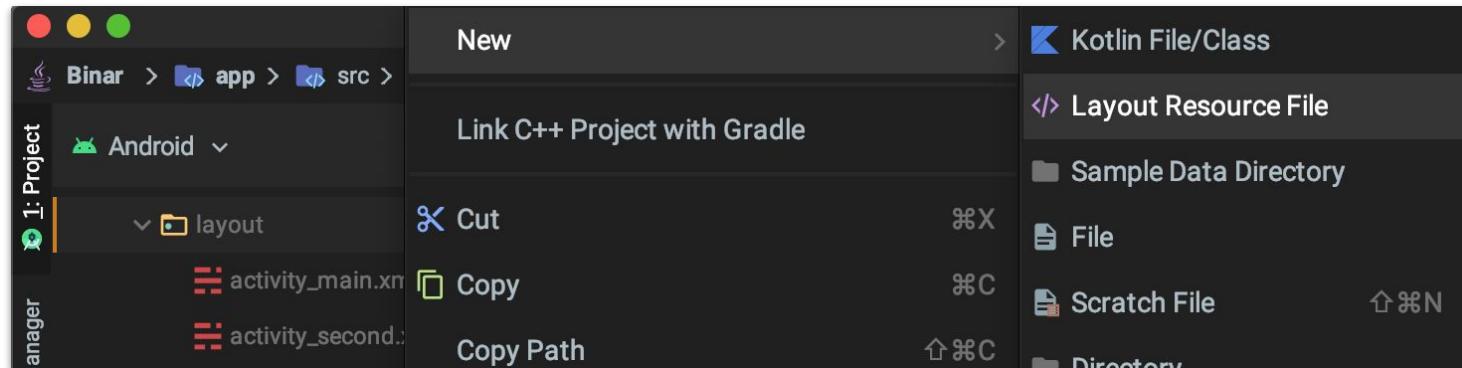


```
activity_main.xml x

1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.recyclerview.widget.RecyclerView
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:id="@+id/recyclerView"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8 </androidx.recyclerview.widget.RecyclerView>
9
10
```



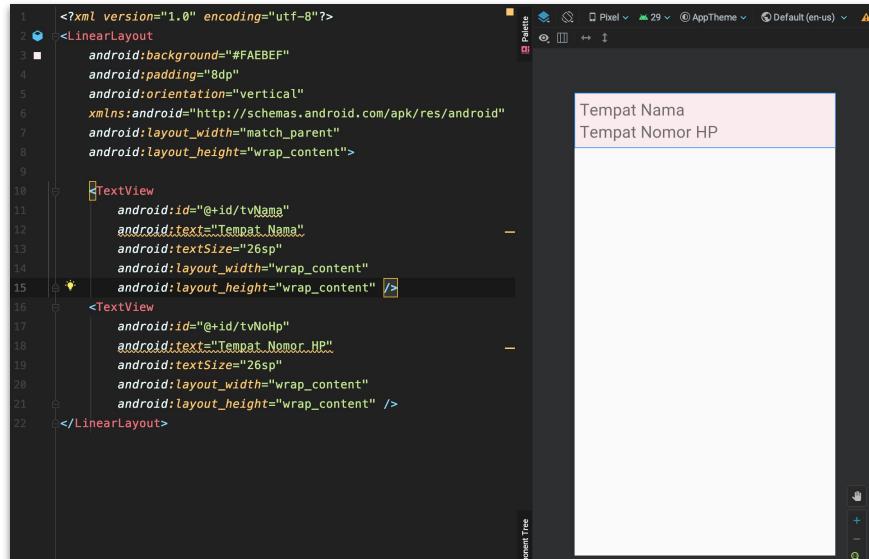
## RecyclerView - Item XML



Setelah buat kode RecyclerView, selanjutnya kita buat Item XML untuk layoutnya! Caranya kayak gini :

1. Klik kanan Folder Layout pada Project kita
2. Arahkan ke New
3. Klik Layout Resource File

Oya, Kamu bisa memberikan nama layout XML yang diinginkan yaa. Misalnya, kita buat namanya **my\_contact\_item.xml**



The screenshot shows the Android Studio interface with the XML code for a RecyclerView item on the left and its preview on the right.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:background="#F0EAEF"
    android:padding="8dp"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

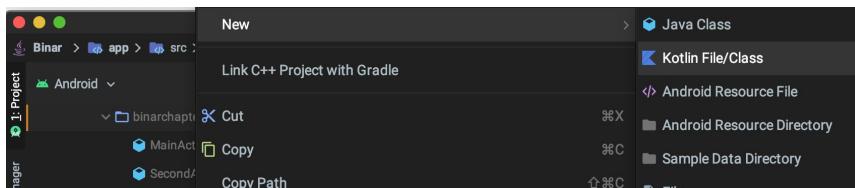
    <TextView
        android:id="@+id/tvNama"
        android:text="Tempat_Nama"
        android:textSize="26sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/tvNoHp"
        android:text="Tempat_Nomor_HP"
        android:textSize="26sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

The preview window shows a light purple rectangular card with two lines of text: "Tempat Nama" and "Tempat Nomor HP".

Setelah kita kasih nama, lalu kita bikin layoutnya kayak gambar disamping.

Kalo udah gitu, teks **Tempat Nama** dan **Tempat Nomor HP** bakal berubah-ubah sesuai daftar data yang kita sediakan.

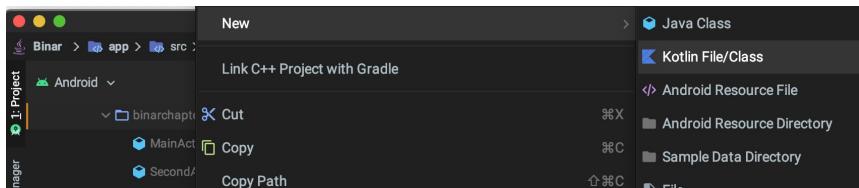


## Membuat POJO

Tadi siapa yang masih inget kalo bikin RecyclerView itu perlu melalui tiga komponen?

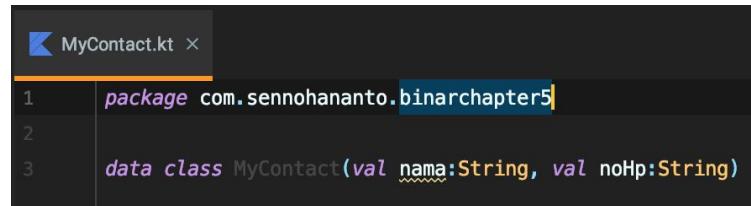
Nah kita kan udah selesai bikin RecyclerView-nya, lanjut nih ke tahap kedua yaitu bikin Adapter.

Tapi, sebelum kita membuat Adapter, kita siapkan dulu POJO alias Plain Old Java Object yang **menyimpan setiap data kontak kita**.



Caranya bikin POJO yaitu klik kanan package tempat file yang akan dibuat

1. Arahkan ke New
2. Klik Kotlin File/Class



```
MyContact.kt ×  
1 package com.sennohananto.binarchapter5  
2  
3 data class MyContact(val nama:String, val noHp:String)
```

Kita buat nama POJO nya dengan :  
**MyContact.kt**

Di Kotlin, kita bisa membuat POJO dengan menggunakan data class.



## Membuat Adapter

Ini dia yang kita tunggu-tunggu. Setelah POJO, kita langsung ke Adapter.

Caranya, kita buat code **ContactAdapter.kt** kayak kode disamping👉

```
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import kotlinx.android.synthetic.main.my_contact_item.view.*

class ContactAdapter(val listContact : ArrayList<MyContact>) : RecyclerView.Adapter<ContactAdapter.ViewHolder>() {

    //Class Holder
    class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView)

    //Membuat Holder
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.my_contact_item,parent, attachToRoot: false)
        return ViewHolder(view)
    }

    //Memberitahu banyaknya List yang akan ditampilkan
    override fun getItemCount(): Int {
        return listContact.size
    }

    //Melakukan penetapan data yang akan ditampilkan pada setiap item / baris
    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        holder.itemView.tvNama.text = listContact[position].nama
        holder.itemView.tvNoHp.text = listContact[position].noHp
    }

}
```



## Membuat LayoutManager

Pada komponen terakhir, kita siapin Layout Manager pada Activity sebelum semua selesai.

Karena kita ingin menampilkan daftarnya sebagai daftar biasa dengan arah Vertikal, kita bisa menggunakan LinearLayoutManager.

Oiya.. selain LinearLayoutManager, kamu juga bisa pake GridLayoutManager lho.

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

        //Membuat arrayList of MyContact sebagai Daftar Kontak kita
        val listContact = arrayListOf(
            MyContact( nama: "Sabrina", noHp: "08111122333"),
            MyContact( nama: "Pak De", noHp: "08555444333"),
            MyContact( nama: "Pak Le", noHp: "085633290384"),
            MyContact( nama: "Bu De", noHp: "087729385538"),
            MyContact( nama: "Bu Le", noHp: "08965889922"),
            MyContact( nama: "Kido", noHp: "085298337866"),
            MyContact( nama: "Sidiu", noHp: "081266729033"),
            MyContact( nama: "Binarian", noHp: "081178549922"),
            MyContact( nama: "My Hubby", noHp: "085198769323"),
            MyContact( nama: "Sayangku", noHp: "081234892341"),
            MyContact( nama: "Mantan", noHp: "085277907396"),
            MyContact( nama: "My Crush", noHp: "089936744332"),
            MyContact( nama: "My Ex", noHp: "082277395833")
        )

        //Membuat Adapter
        val adapter = ContactAdapter(listContact)

        //menyiapkan Layout Manager
        val layoutManager = LinearLayoutManager( context: this,LinearLayoutManager.VERTICAL, reverseLayout: false)

        //Set layout manager pada RecyclerView
        recyclerView.layoutManager = layoutManager

        //Set adapter pada recyclerView
        recyclerView.adapter = adapter
    }
}
```

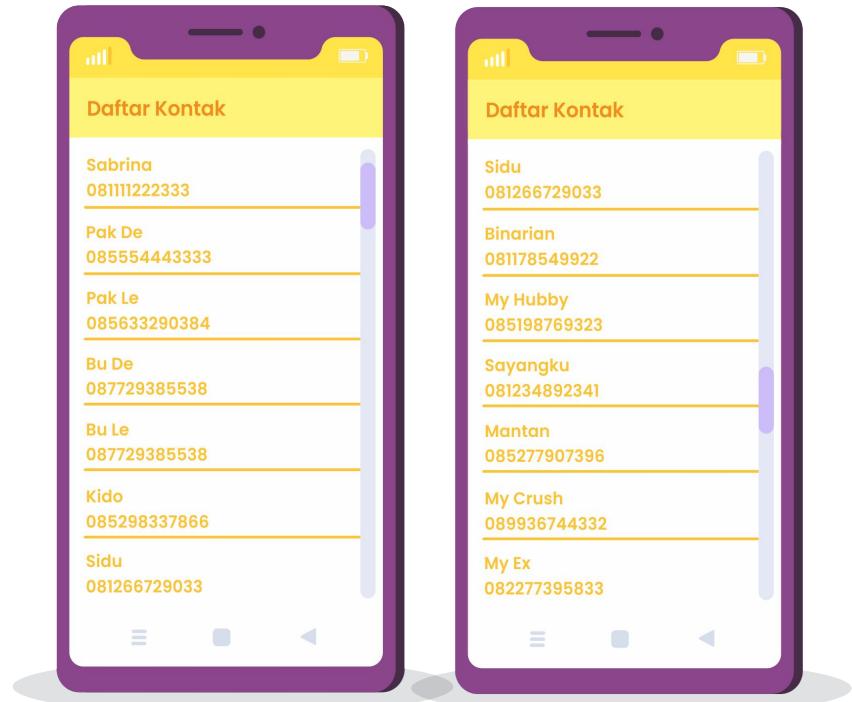


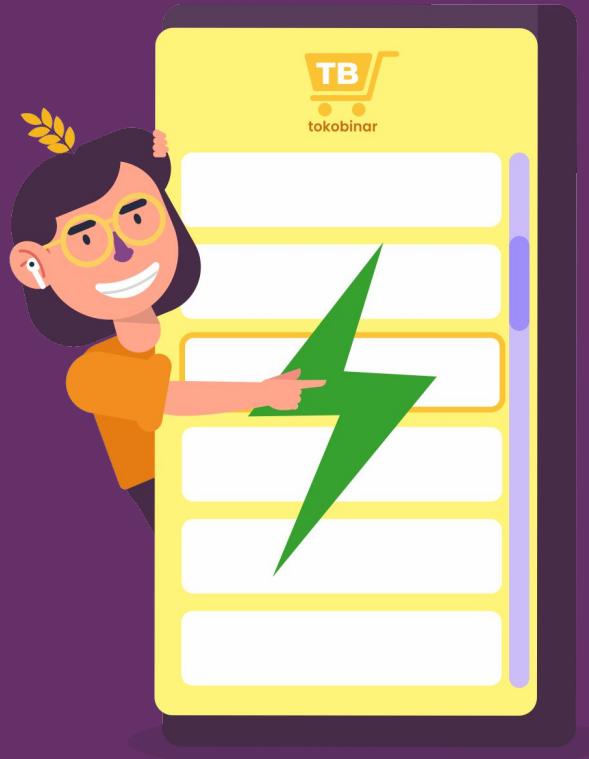
## Yeay Selesai!

Bukan sulap bukan sihir, tampilannya bakal muncul kayak gambar disamping!

Eh bentar... tampilannya bukan kamu banget? pingin coba diubah sesuai karakter zodiak kamu? boleeeeh~

Kita bisa berkreasi di bagian **my\_contact\_item.xml** untuk mempercantik tampilan daftar kita.





**Asiik, udah bisa bikin RecyclerView~**

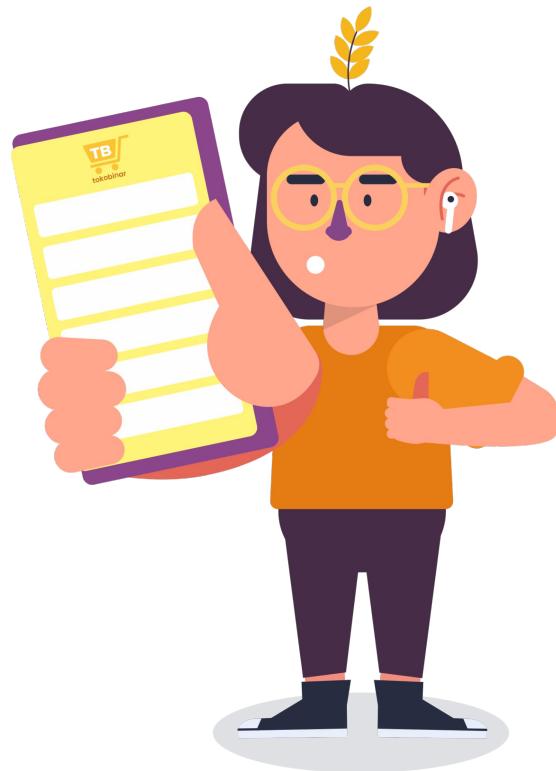
**Selain bisa dikreasikan, tampilan RecyclerView juga bisa dibuat lebih smooth dan nggak lemot.**

**Ada nih kita bakal belajar satu teknik, namanya DiffUtil.**



## Apa itu DiffUtil?

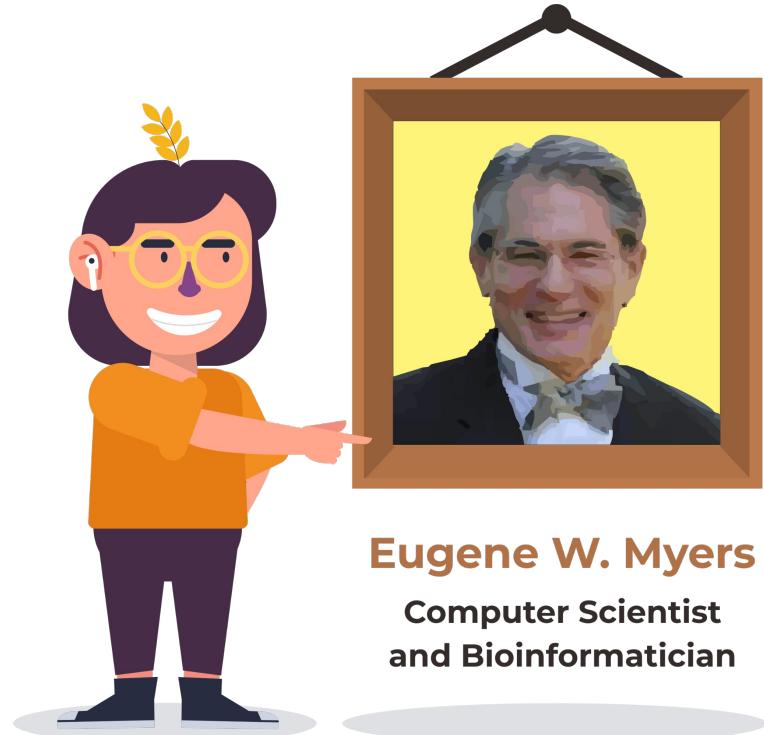
DiffUtil adalah sebuah utility class yang dapat **menghitung perbedaan antara dua list** dan **mengeluarkan operasi list dengan pembaruan data yang lebih terbaru**.





DiffUtil menggunakan algoritma perbedaan Eugene W. Myers untuk menghitung jumlah minimal pembaruan yang mengubah satu list menjadi list lainnya.

Algoritma ini tidak menangani item yang dipindahkan, sehingga DiffUtil menjalankan tahap kedua untuk mendeteksi item yang dipindahkan.

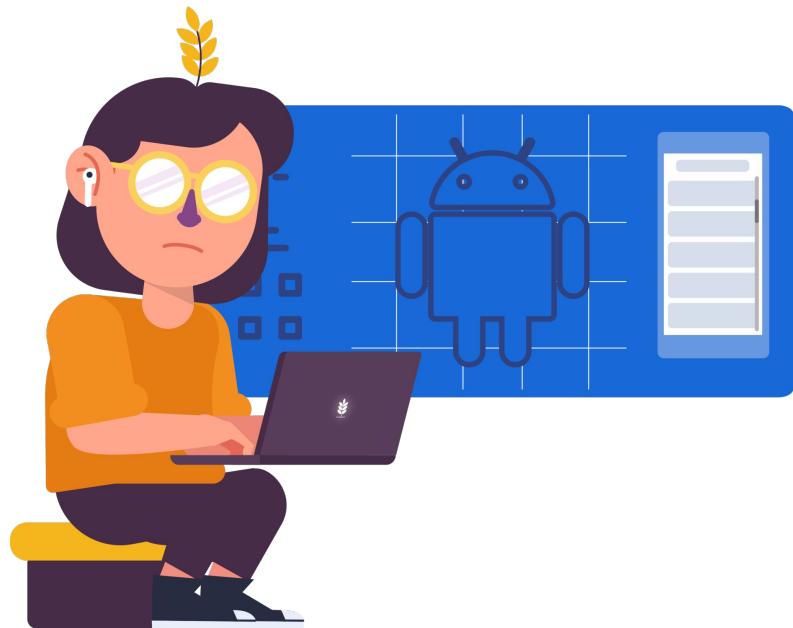


**Eugene W. Myers**  
**Computer Scientist  
and Bioinformatician**



Dikarenakan DiffUtil menjalankan tahap kedua untuk mendeteksi item yang dipindahkan, jadi fungsi dan kegunaan dari DiffUtil adalah untuk mendapat data yang selalu terbaru untuk ditampilkan di recyclerView.

Agak membagongkan? Oke kita lihat slide selanjutnya



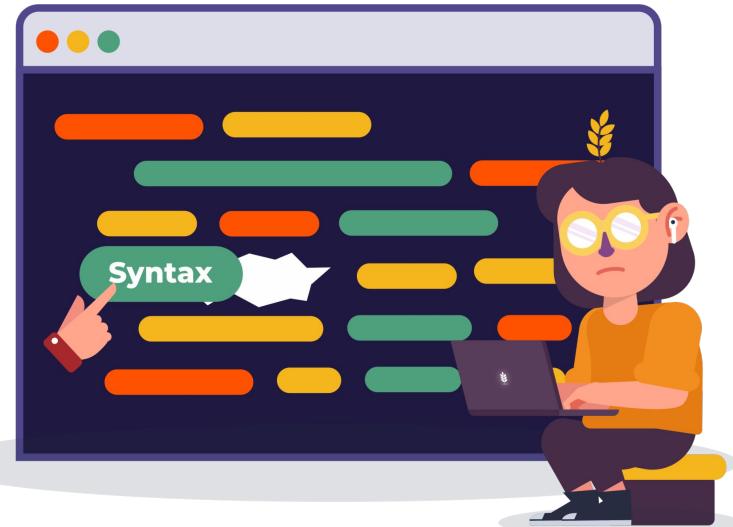


## Kenapa Pakai DiffUtil?

Kita flashback dulu, sebelumnya kita telah mempelajari cara membuat RecyclerView dengan mudah. Ternyata, cara pendeklarasian tersebut memiliki celah ketika ingin mendapatkan data terbaru.

Celahnya apa? Yakni kita perlu memanggil syntax berikut untuk mengupdate dataset.

```
adapter.notifyDataSetChanged()
```





```
adapter.notifyDataSetChanged()
```

Kalau memanggil syntax di atas, semua tampilan yang terlihat akan dibangun ulang untuk menampilkan datanya yang terbaru.

Nah proses ini lah yang menjadi operasi yang berat karena **harus membangun ulang recyclerView**.

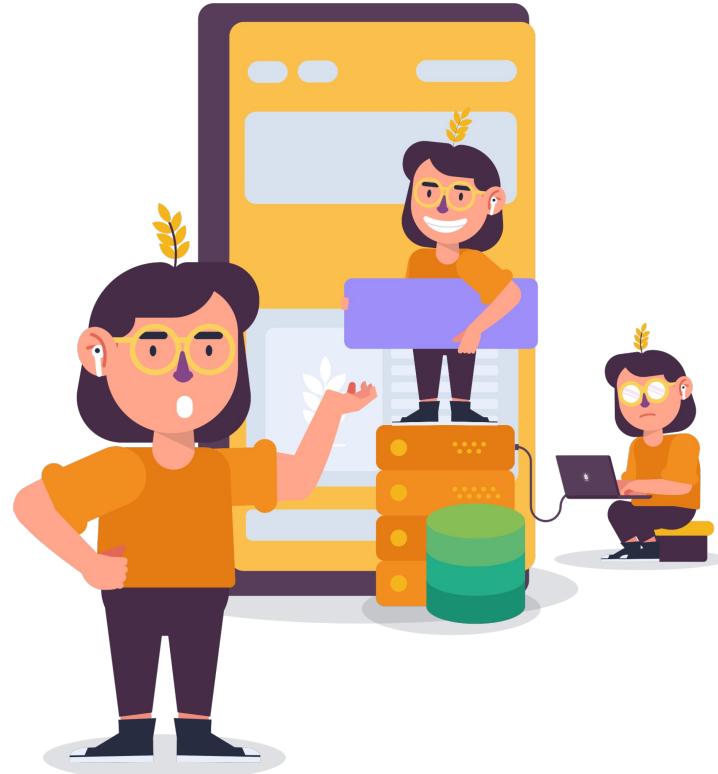




```
adapter.notifyDataSetChanged()
```

Padahal, bisa aja data yang ditambahin dan berubah cuman satu.

Bisa juga kita cuman menambahkan data baru ke dalam dataset kita, yang artinya nggak perlu ada perubahan pada data yang udah ada sebelumnya.





Selain itu, untuk membuat ulang tampilan pada RecyclerView kita perlu pake beberapa fungsi lain. Misalnya seperti:

- `notifyItemMoved`
- `notifyItemRangeChanged`
- `notifyItemRangeInserted`
- `notifyItemRangeRemoved`





## Terus, Peran DiffUtil apa?

DiffUtil hadir untuk memudahkan kita dalam mengupdate dataset pada RecyclerView, sesuai dengan dataset yang mau kita tampilkan.

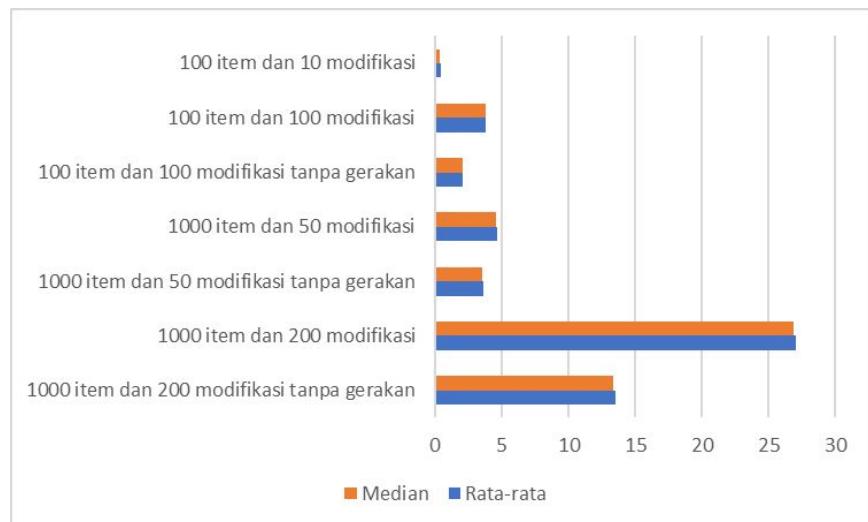
Kita juga nggak perlu harus menggunakan function yang akan membuat ulang tampilan RecyclerView.





## Keuntungan menggunakan DiffUtil?

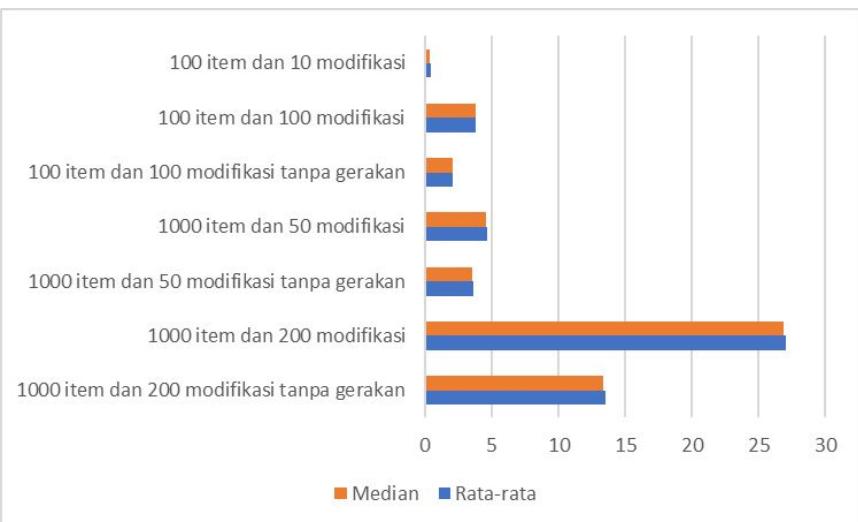
Nah ada nih hasil pengujian yang menggambarkan bahwa penerapan DiffUtil lebih baik untuk RecyclerView





Hasil ini di dasarkan pada Nexus 5x dengan OS Android Marshmallow. Perbandingannya bisa lihat di graph disamping.

Catatan: Ukuran maksimum daftar bisa  $2^{26}$  karena batasan yang disediakan.





Gimana, udah tau kan kenapa mending pake DiffUtil aja?

Terakhir nih, biar pemahaman kita makin sempurna, kita coba terapin [Cara Deklarasi DiffUtil pada RecyclerView](#).

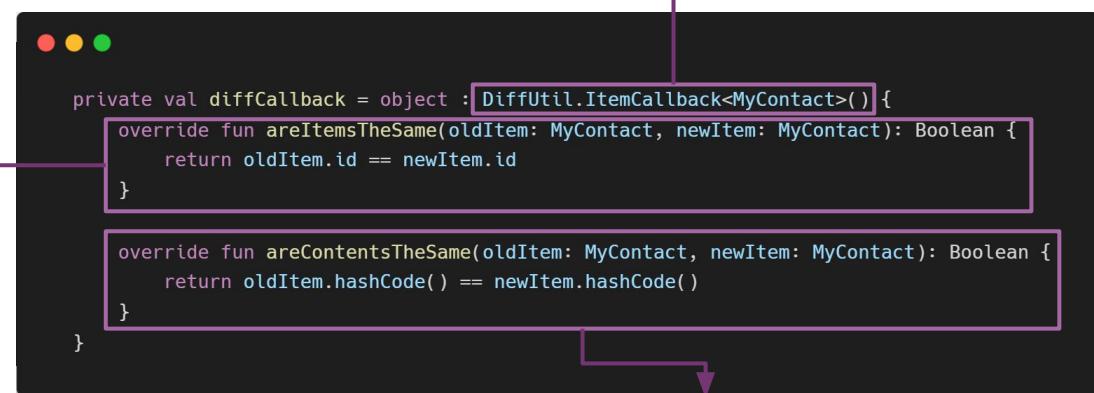
Langsung aja deh meluncur ke langkah-langkah deklarasinya~



## Langkah 1:

First thing first, untuk mendeklarasikan DiffUtil pada adapter kita bisa memanggil syntax berikut :

```
DiffUtil.ItemCallback<[Model]>()
```



```
private val diffCallback = object : DiffUtil.ItemCallback<MyContact>() {
    override fun areItemsTheSame(oldItem: MyContact, newItem: MyContact): Boolean {
        return oldItem.id == newItem.id
    }

    override fun areContentsTheSame(oldItem: MyContact, newItem: MyContact): Boolean {
        return oldItem.hashCode() == newItem.hashCode()
    }
}
```

Pada fungsi pertama yang kita override yaitu **areItemTheSame**.

Kita perlu menentukan apa yang menjadi **indikator pembeda antara item satu dan item yang lain**. Di sini kita bisa menggunakan id untuk menjadi indikator pembeda.

Kemudian, indikator terakhir adalah **areContentsTheSame**.

Fungsinya untuk menentukan indikator yang akan digunakan untuk menentukan **apakah item pada dataset kita berubah atau tidak**.



## Langkah 2 :

Lanjut~ Setelah itu kita deklarasikan **AsyncListDiffer** seperti kode disamping :

```
private val differ = AsyncListDiffer(this, diffCallback)
```

---

**AsyncListDiffer** dapat menerima nilai secara LiveData dan asynchronous serta mengupdatenya ke dataset.

```
fun submitData(value: ArrayList<MyContact>) = differ.submitList(value)
```

Setelahnya kita buat sebuah function supaya bisa memanggil adapter yang telah kita buat dan memasukkan data dari sebuah activity atau fragment yang menggunakan.



## Langkah 3:

Setelahnya pada **onBindViewHolder** dan **getItemCount** pada adapter yang sebelumnya, kita ubah menjadi seperti berikut:

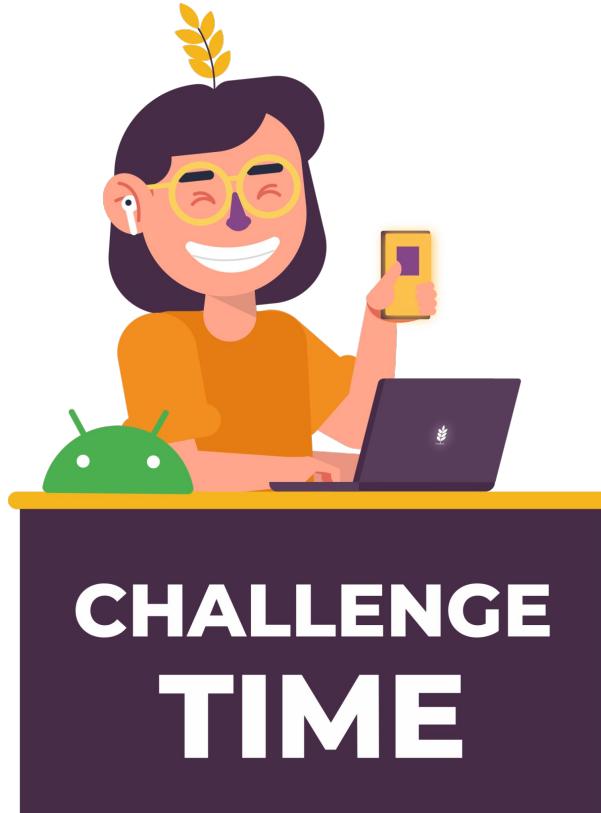
```
override fun onBindViewHolder(holder: ViewHolder, position: Int) {  
    val data = differ.currentList[position]  
  
    holder.itemView.tvNama.text = data.nama  
    holder.itemView.tvNoHp.text = data.noHp  
}  
  
override fun getItemCount(): Int = differ.currentList.size
```



## Gimana? Bisa mengikuti?

Untuk liat wujud full kode nya, bisa dilihat pada gambar di samping:

```
class ContactAdapter(val listContact: ArrayList<MyContact>) :  
    RecyclerView.Adapter<ContactAdapter.ViewHolder>() {  
  
    private val diffCallback = object : DiffUtil.ItemCallback<MyContact>() {  
        override fun areItemsTheSame(oldItem: MyContact, newItem: MyContact): Boolean {  
            return oldItem.id == newItem.id  
        }  
  
        override fun areContentsTheSame(oldItem: MyContact, newItem: MyContact): Boolean {  
            return oldItem.hashCode() == newItem.hashCode()  
        }  
    }  
  
    private val differ = AsyncListDiffer(this, diffCallback)  
  
    fun submitData(value: ArrayList<MyContact>) = differ.submitList(value)  
  
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {  
        val view = LayoutInflater.from(parent.context).inflate(R.layout.my_contact_item, parent, false)  
        return ViewHolder(view)  
    }  
  
    override fun onBindViewHolder(holder: ViewHolder, position: Int) {  
        val data = differ.currentList[position]  
  
        holder.itemView.tvNama.text = data.nama  
        holder.itemView.tvNoHp.text = data.noHp  
    }  
  
    override fun getItemCount(): Int = differ.currentList.size  
  
    class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView)  
}
```



## Biar makin paham, kita coba Challenge ini yuk~

Buatlah suatu Daftar apapun dengan:

- Menggunakan **LinearLayoutManager** atau **GridLayoutManager**;
- menggunakan **CardView**;
- menampilkan **Gambar** pada item-nya.

Buat sebagus mungkin tampilannya ya!

*Pengumpulan tugas diserahkan kepada mentor*

Kita akan membahas bersama-sama di pertemuan berikutnya.

# Saatnya kita Quiz!





## 1. Komponen Widget View pada XML sebagai tempat tampilnya sebuah daftar / list disebut ...

- A. RecyclerView
- B. Adapter
- C. LayoutManager



### 1. Komponen Widget View pada XML sebagai tempat tampilnya sebuah daftar / list disebut ...

- A. RecyclerView
- B. Adapter
- C. LayoutManager

Benar! RecyclerView adalah komponen Widget View pada XML sebagai tempat tampilnya List / Daftar



**2. Untuk menyiapkan kumpulan data untuk ditampilkan pada RecyclerView, dibutuhkan sebuah class khusus. Class tersebut adalah ....**

- A. RecyclerView
- B. Adapter
- C. LayoutManager



**2. Untuk menyiapkan Kumpulan data untuk ditampilkan pada RecyclerView, dibutuhkan sebuah class khusus. Class tersebut adalah ....**

- A. RecyclerView
- B. Adapter
- C. LayoutManager

**Betul! Class adapterlah yang bertugas menyiapkan tampilan setiap data dari list kita ke RecyclerView!**



**3. Class yang bertugas untuk menata tampilan dari daftar di RecyclerView?**

- A. RecyclerView
- B. Adapter
- C. LayoutManager



### 3. Class yang bertugas untuk menata tampilan dari daftar di RecyclerView?

- A. RecyclerView
- B. Adapter
- C. LayoutManager

**Benar!. Layout manager bertugas untuk menata tampilan daftar yang akan ada di RecyclerView.**

**Sebuah LayoutManager bisa kita tetapkan pada Recyclerview dengan cara :**  
`recyclerView.setLayoutManager(<LayoutManager>)`



#### 4. Berikut ini yang merupakan fungsi utama dari penggunaan class DiffUtil adalah ....

- A. Tidak perlu harus menggunakan function yang akan membuat ulang tampilan recyclerView
- B. Mendapat data yang selalu terbaru untuk ditampilkan di recyclerView
- C. Memudahkan user mendapatkan data dengan tampilan yang berbentuk GridView



### 4. Berikut ini yang merupakan fungsi utama dari penggunaan class DiffUtil adalah ....

- A. Tidak perlu harus menggunakan function yang akan membuat ulang tampilan recyclerView
- B. Mendapat data yang selalu terbaru untuk ditampilkan di recyclerView
- C. Memudahkan user mendapatkan data dengan tampilan yang berbentuk GridView

**Benar! DiffUtil hadir untuk memudahkan kita dalam mengupdate dataset pada recyclerView berdasarkan perubahan pada dataset yang hendak kita tampilkan.**

**Kita tidak perlu harus menggunakan function yang akan membuat ulang tampilan recyclerView.**



**5. Manakah dari method berikut yang akan tampil ketika kita menggunakan DiffUtil pada RecyclerView?**

- A. areItemsTheSame, areContentsTheSame
- B. areItemSame, areContentSame
- C. areSameTheItem, areSameTheContent



### 5. Manakah dari method berikut yang akan tampil ketika kita menggunakan DiffUtil pada RecyclerView?

- A. areItemsTheSame, areContentsTheSame
- B. areItemSame, areContentSame
- C. areSameTheItem, areSameTheContent

areItemsTheSame dan areContentsTheSame akan di override ketika kita menggunakan fitur DiffUtil pada recyclerView.

## Referensi dan bacaan lebih lanjut~

1. [Android RecyclerView Tutorial with Kotlin | raywenderlich.com](#)
2. [Create dynamic lists with RecyclerView | Android Developers](#)
3. [Speed up Your Android RecyclerView Using DiffUtil | Raywenderlich](#)



Nah, selesai sudah pembahasan kita di Chapter 4 Topic 3 ini.

Selanjutnya, kita bakal bahas metode penyimpanan data aplikasi **SharedPreference**

Penasaran kayak gimana? Cus langsung ke topik selanjutnya~



# Terima Kasih!



Next Topic

loading...