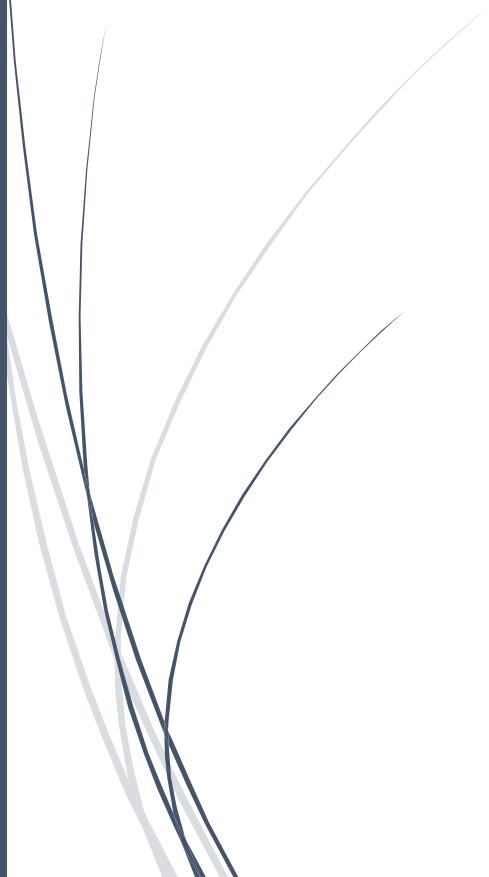




7/18/2020

MODUL PRAKTIKUM MOBILE PROGRAMMING



Sepyan Purnama Kristanto
POLITEKNIK NEGERI BANYUWANGI

MODUL

MODUL 1

Pengenalan Aplikasi Mobile



CAPAIAN PEMBELAJARAN

1. Mahasiswa diharapkan dapat memahami dan mengetahui tentang aplikasi mobile, cara penginstallan Android Studio, debug dengan virtual device dan debug dengan physical device.
2. Mahasiswa Mampu memahami tentang Pengembangan Aplikasi Android



KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. Android Studio 3.4.
2. Handphone Android versi 7.0 (Nougat)
3. Kabel data USB.
4. Driver ADB.



DASAR TEORI

1.1. Pendahuluan.

Selamat datang di Praktikum Pemrograman Mobile. Kita akan menggunakan Android Kotlin untuk praktikum ini. Praktikum ini menggunakan codelab yang menuntun kita membangun aplikasi Android menggunakan Kotlin. Prasyarat untuk dapat mengikuti praktikum ini dengan baik adalah memiliki pengetahuan dalam bahasa pemrograman

berorientasi objek penuh seperti Java, C ++. Untuk pengembangan, disarankan untuk menggunakan sumber referensi selain modul praktikum ini.

Kita juga harus terbiasa menavigasi GitHub dan terbiasa dengan konsep-konsep: Algoritma pemrograman, penanganan error, bagaimana kode dibangun, dikompilasi, dan dieksekusi, secara umum.

Kita akan menggunakan sumber referensi utama pada link andriod developer dan codelab berikut:

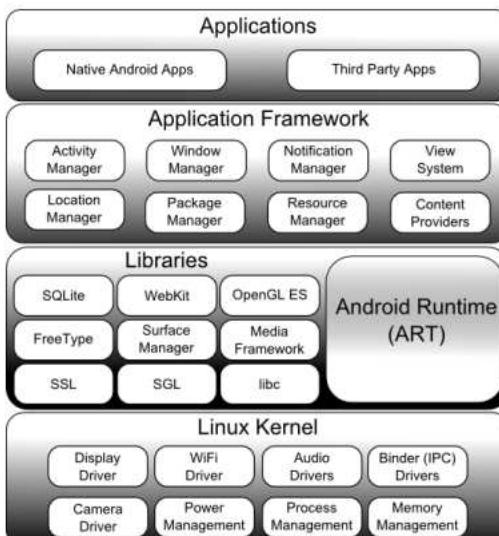
1. <https://kotlinlang.org/docs/reference/> untuk belajar dasar pemrograman kotlin.
2. <https://developer.android.com/kotlin> untuk pemahaman kenapa android menggunakan kotlin.
3. <https://developer.android.com/courses/kotlin-android-fundamentals/toc> untuk codelab review untuk dasar pemrograman kotlin.
4. <https://codelabs.developers.google.com/android-kotlin-fundamentals/> untuk belajar coding project kotlin.
5. <https://developer.android.com/kotlin/learn> untuk belajar tentang pemrograman kotlin untuk android.
6. <https://developer.android.com/kotlin/resources> untuk mengakses sumber-sumber yang digunakan.

1.2. Android dan Pemrograman Mobile.

Android adalah sistem operasi mobile yang open source. Tahun 2005, Google mengakuisisi perusahaan **Android Inc.** untuk memulai mengembangkan platform Android. Tahun 2007, sekelompok pemimpin industri datang bersama membentuk **Open Handset Alliance** (<http://www.openhandsetalliance.com>). November 2007, Android SDK dirilis pertama kali dengan “tampilan awal” (**early look**). September 2008, T-Mobile mengumumkan ketersediaan **HTC Dream G1**, smartphone pertama yang berbasiskan platform Android. •Beberapa hari berikutnya Google mengumumkan ketersediaan Android SDK Release Candidate 1.0. Oktober 2008, Google membuat kode program dari platform Android tersedia di bawah **“Apache’s open source license”**.

Android adalah platform mobile pertama yang lengkap, terbuka dan bebas. Sistem operasi yang mendasari android dilisensikan dibawah GNU, General Public Lisensi Versi 2 (GPL), yang sering dikenal dengan istilah “copyleft” lisensi di mana setiap perbaikan pihak ketiga harus terus jatuh di bawah terms. Android didistribusikan dibawah lisensi Apache Software (ASL/Apache2), yang memungkinkan untuk distribusi kedua dan seterusnya.

Android disusun dalam bentuk software stack yang terdiri dari aplikasi, sistem operasi, lingkungan run-time, middleware, layanan dan pustaka (library). Setiap lapisan dari tumpukan, dan unsur-unsur yang sesuai dalam setiap lapisan, saling terintegrasi untuk memberikan pengembangan aplikasi dan lingkungan eksekusi yang optimal untuk perangkat mobile. Arsitektur ini ditampilkan pada Gambar 1-1.



Gambar 1-1. Arsitektur Aplikasi Android

Dalam rangka mengembangkan aplikasi Android di Android Studio akan diperlukan untuk mengkompilasi dan menjalankan aplikasi beberapa kali. Aplikasi Android dapat diuji dengan menginstal dan menjalankannya baik pada perangkat fisik atau dalam *Virtual Device* (AVD) lingkungan emulator Android. Sebelum AVD dapat digunakan, terlebih dahulu harus dibuat dan dikonfigurasi untuk mencocokkan spesifikasi model perangkat tertentu. Tujuan dari bab ini, oleh karena itu, adalah untuk bekerja melalui langkah-langkah yang terlibat dalam menciptakan suatu perangkat virtual menggunakan Nexus 7 tablet sebagai contoh referensi.

AVD pada dasarnya adalah emulator yang memungkinkan aplikasi Android untuk diuji tanpa perlu menginstal aplikasi pada perangkat berbasis Android fisik. Sebuah AVD dapat dikonfigurasi untuk meniru berbagai fitur perangkat keras termasuk pilihan ukuran layar, kapasitas memori dan fitur seperti kamera, dukungan navigasi GPS atau accelerometer. Sebagai bagian dari instalasi Android Studio standar, memungkinkan emulator dipasang dan dikonfigurasi untuk berbagai perangkat yang berbeda. AVD baru dibuat dan dikelola dengan menggunakan Android Virtual Device Manager, yang dapat digunakan dalam mode baris perintah atau dengan lebih antarmuka grafis yang user-friendly.

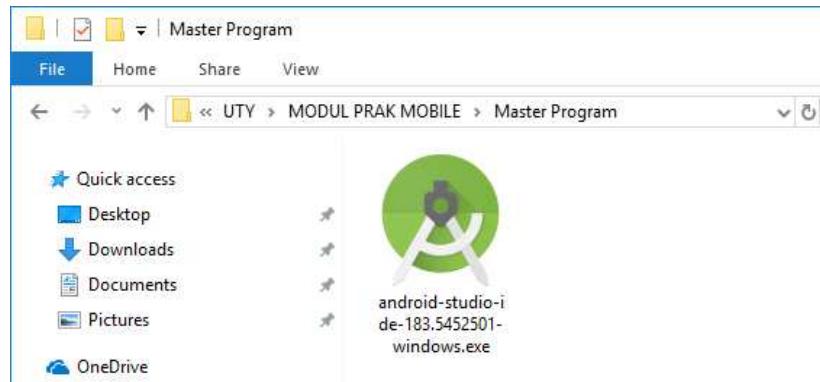


PRAKTIK

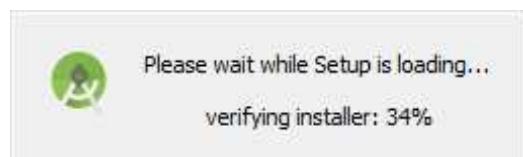
I. Install Android Studio

1. Android studio didapatkan melalui website resmi <https://developer.android.com/studio>, jika Android Studio telah didownload maka

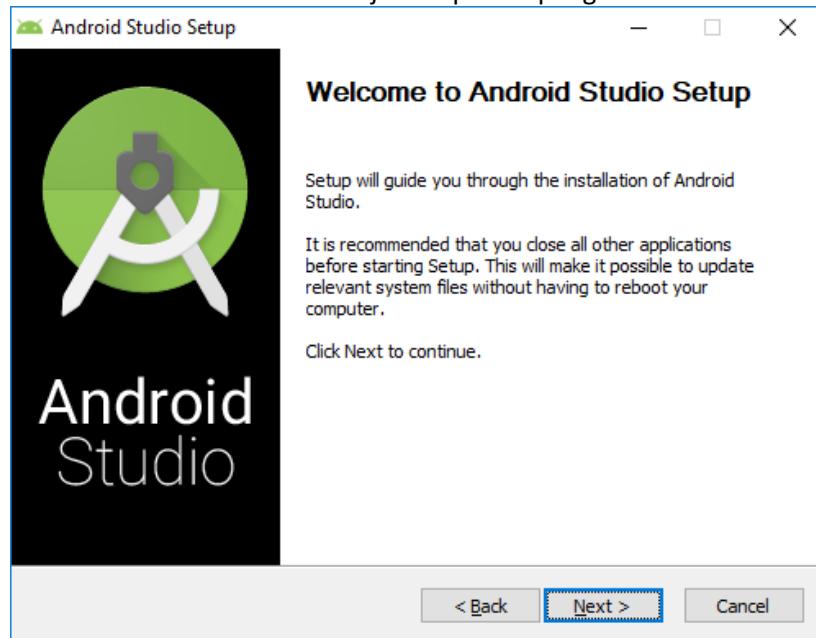
anda dapat langsung menginstall dengan cara, klik 2x pada **android-studio-ide-183.5452501-windows.exe**



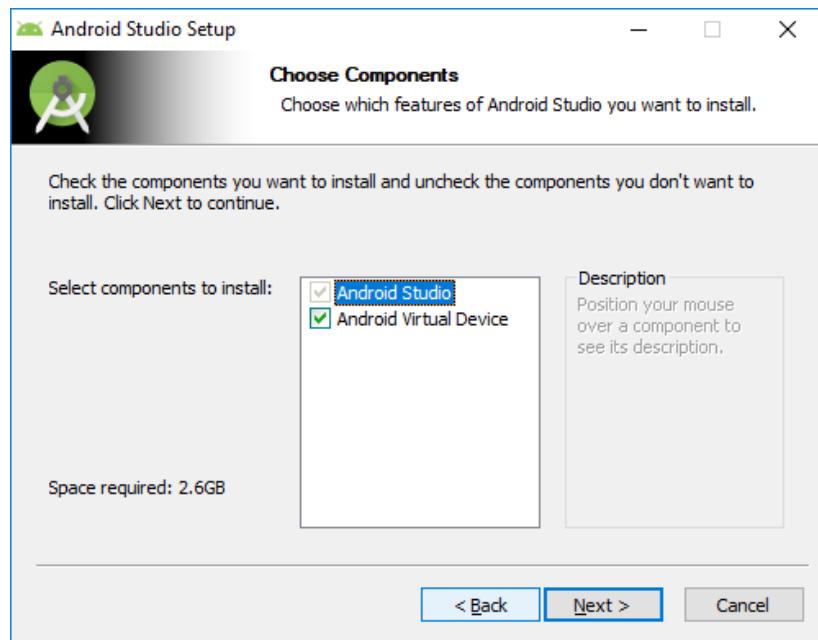
2. Tunggu beberapa saat sampai proses loading verifying installer selesai.



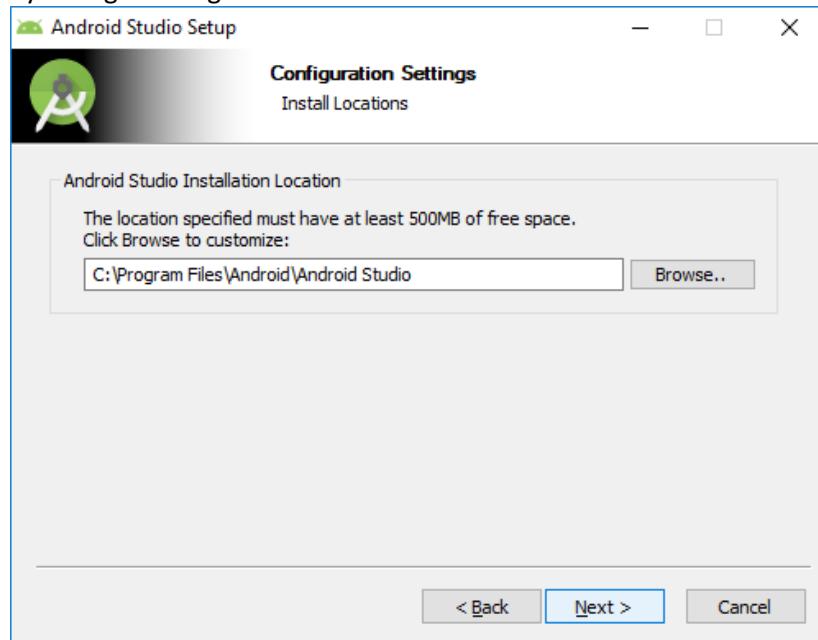
3. Setelah proses loading verifying installer selesai, akan terlihat jendela Android Studio Setup, **klik button Next >** untuk melanjutkan proses penginstallan.



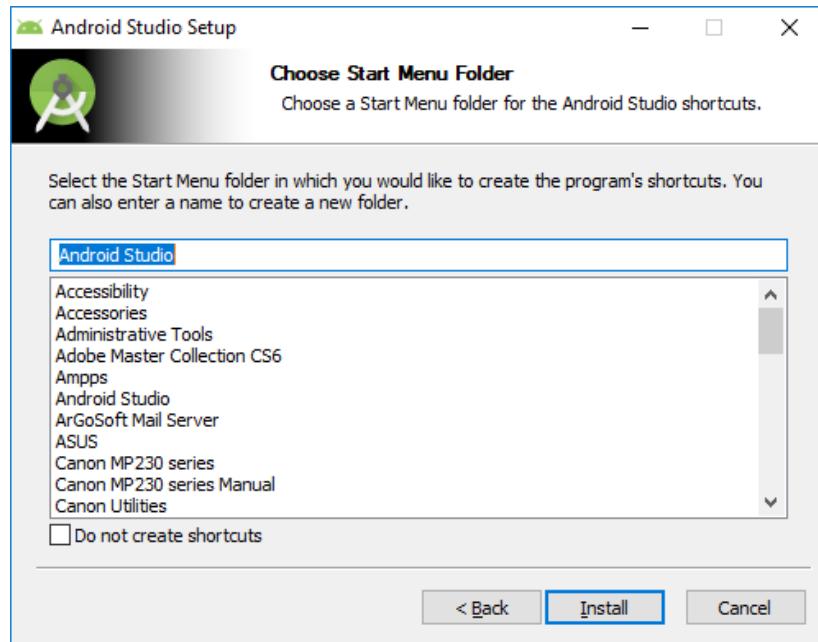
4. Beri tanda checklist atau centang bagian **Android Virutal Device**. Fungsi dari komponen Android Virtual Device yaitu untuk menampilkan interface android dalam bentuk virtual. Langkah selanjutnya **klik** pada button **Next >**



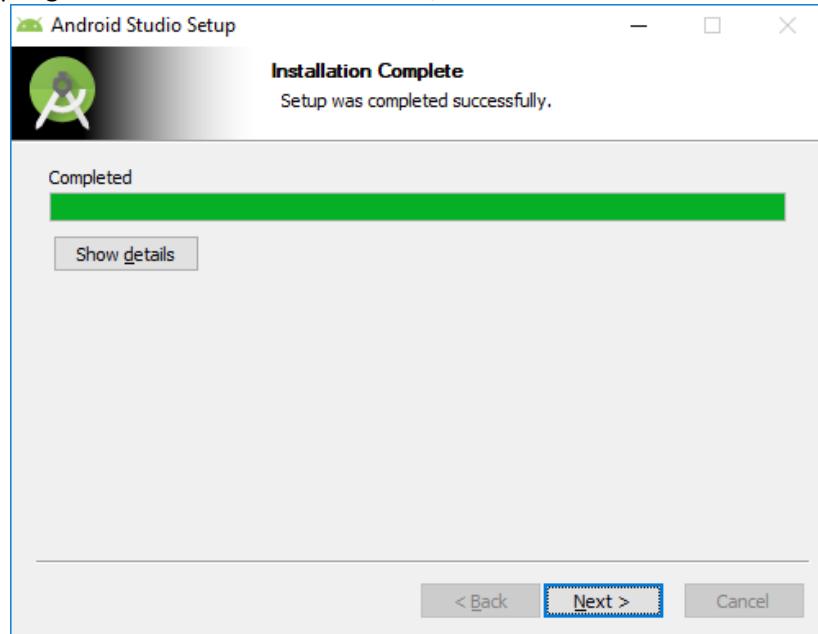
5. Pada bagian Configuration Settings akan diminta untuk memilih lokasi penginstallan, secara default lokasi penginstallan pada direktori **C:\Program Files\Android\Android Studio**. Jika ingin menggunakan lokasi defult maka dapat langsung menuju proses selanjutnya dengan meng-klik button **Next >**



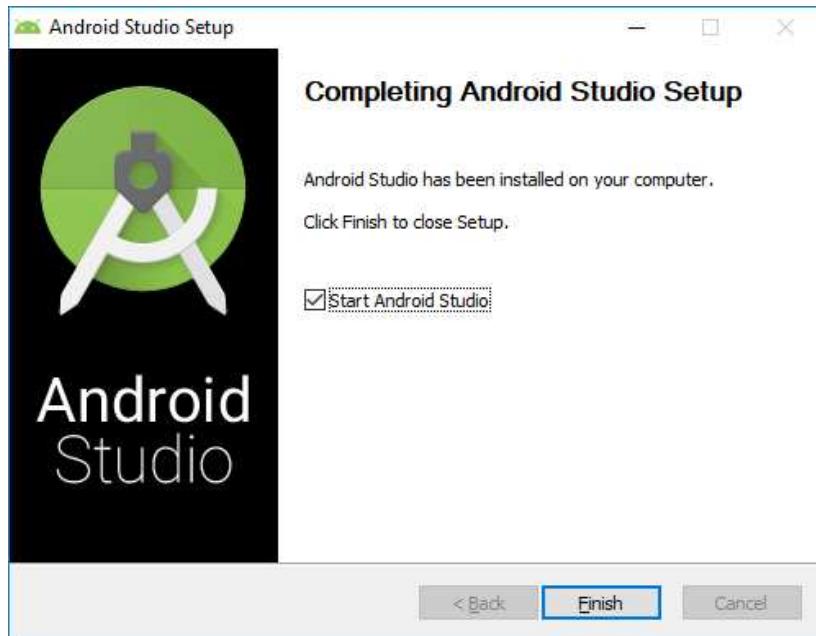
6. Pada bagian Choose Start Menu Folder, klik button **Install**



7. Tunggu beberapa saat sampai proses penginstallan Android Studio selesai. Setelah proses penginstallan Android Studio selesai, **klik** button **Next >**



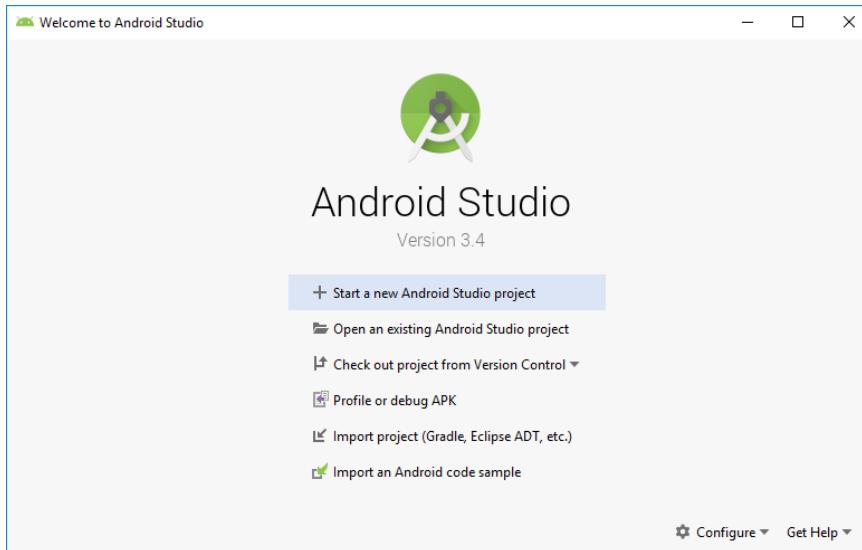
8. Proses penginstallan Android Studio telah selesai, **klik** button **Finish** untuk menutup jendela Android Studio Setup. Jika ingin menjalankan langsung program Android Studio maka beri tanda **checkbox** atau centang pada bagian **Start Android Studio** sebelum menekan button Finish.



II. Running Program Android Studio

Dalam menjalankan program program Android Studio sama halnya dalam menjalankan program-program lainnya, agar mudah dalam memahami cara menjalankan Android Studio dapat dengan mengikuti langkah-langkah berikut:

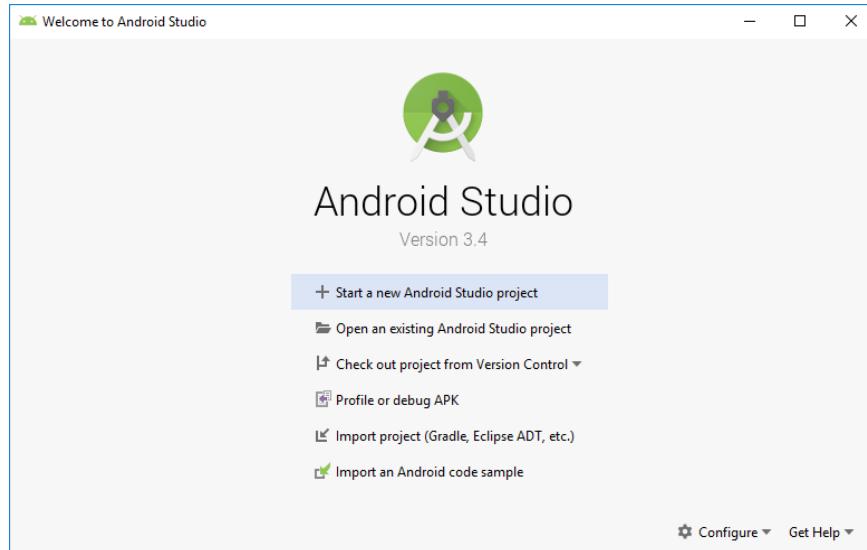
1. Buka Android Studio dari komputer yang dipakai.
2. Maka akan muncul jendela Welcome to Android Studio.



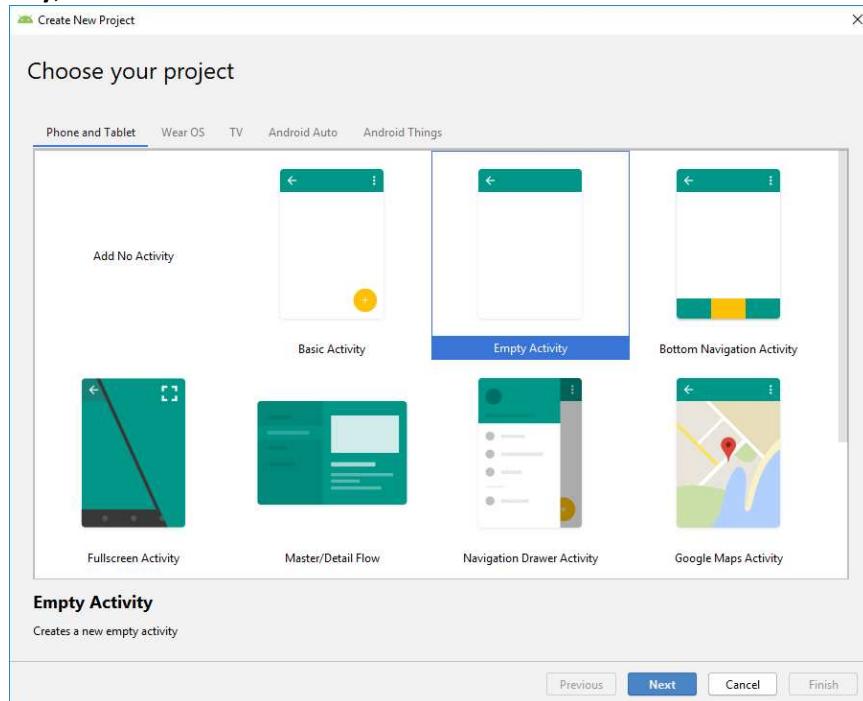
III. Membuat Project Dengan Android Studio

Pembuatan project dengan Android Studio dapat dilakukan dengan langkah-langkah sebagai berikut :

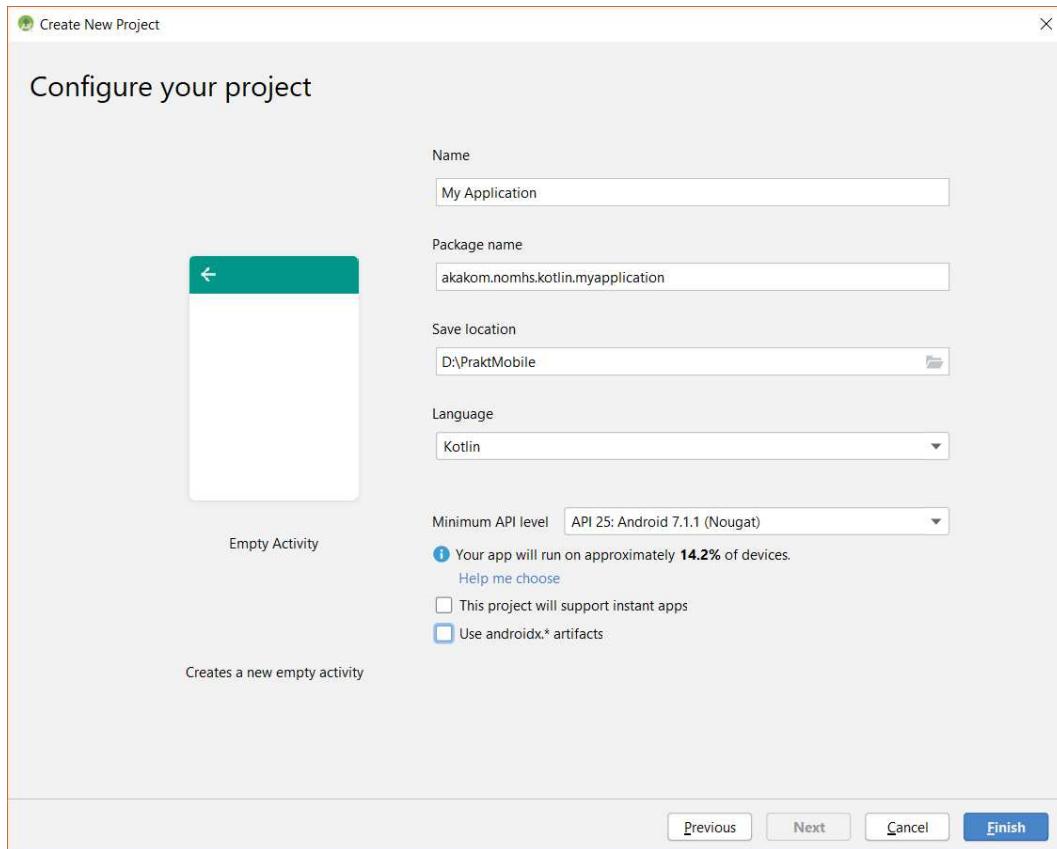
1. Dalam membuat project awal dengan Android Studio dengan cara **klik pada + Start a new Android Studio project**



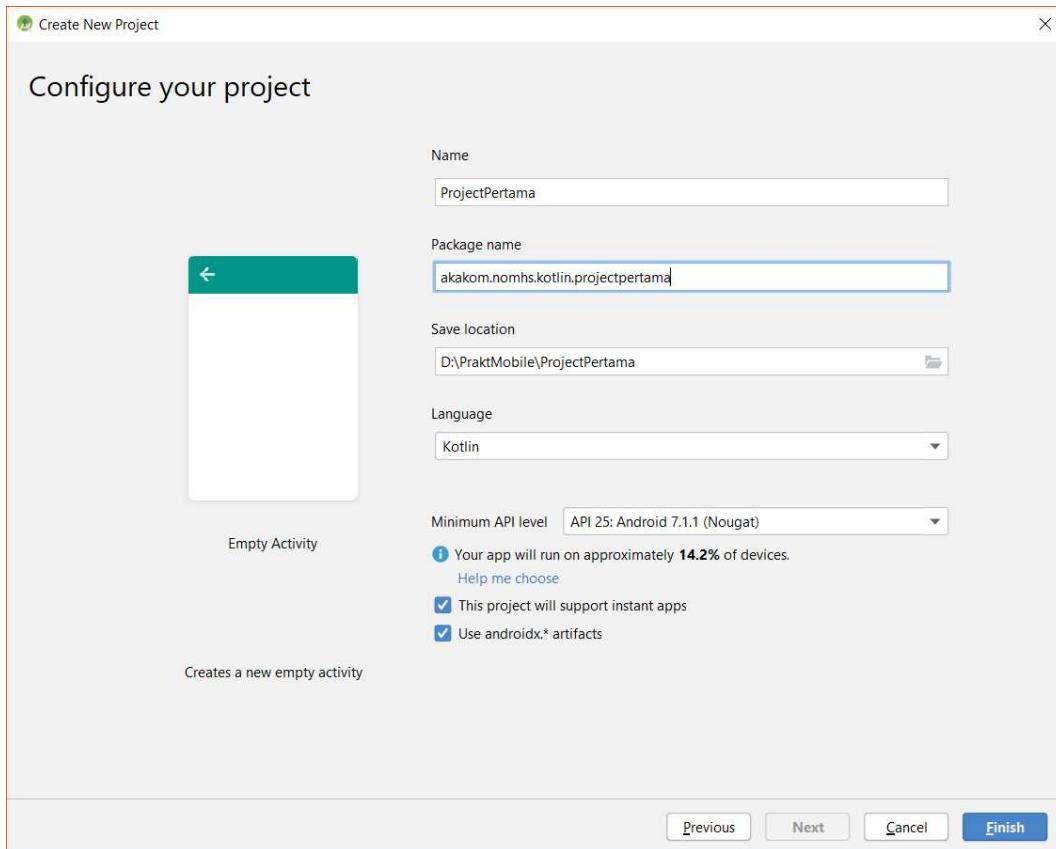
2. Kemudian akan terlihat jendela Create New Project, pada bagian ini programmer dapat memilih bentuk dari project yang akan dibuat (hal ini menyesuaikan dengan project yang akan dibuat). Sebagai latihan awal maka **pilih** pada bagian **Empty Activity**, kemudian **klik** button **Next**.



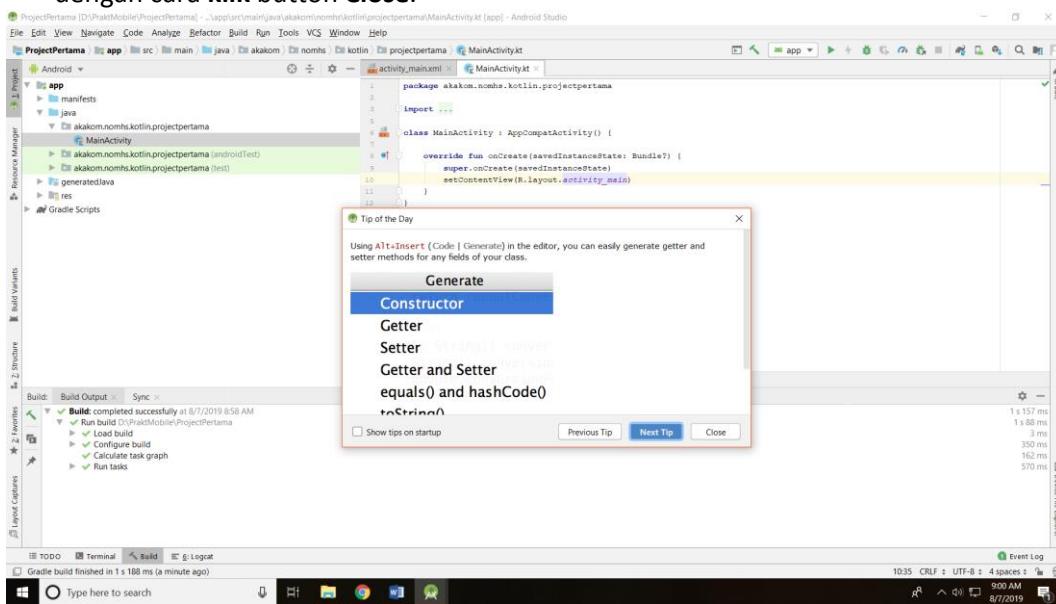
3. Pada jendela Create New Project, maka secara default Name project yaitu My Application, penamaan ini dapat dirubah sesuai dengan nama project yang dibuat.



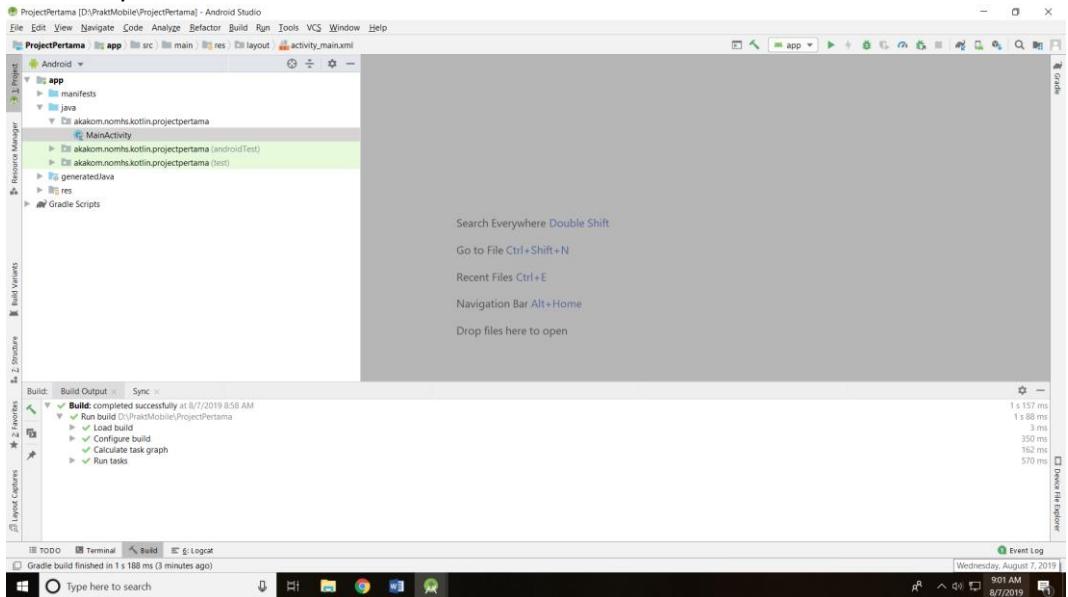
4. Sebagai bahan latihan, beri nama pada bagian Name : **ProjectPertama** kemudian pada bagian Save location : (*gunakan folder kerja anda*), kemudian klik button **Finish**.



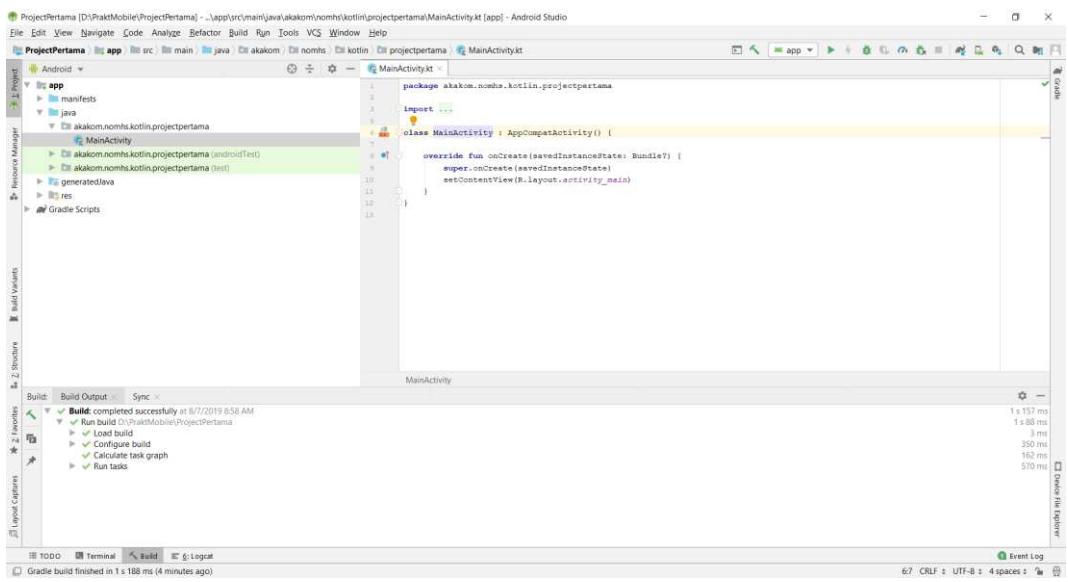
5. Pada saat awal atau pertamakali membuat project maka akan muncul sebuah jendela dengan nama Tips of The Day yaitu informasi tentang pemrograman yang dapat dijadikan acuan para programmer, untuk menutup jendela Tips of The Day dapat dengan cara klik button Close.



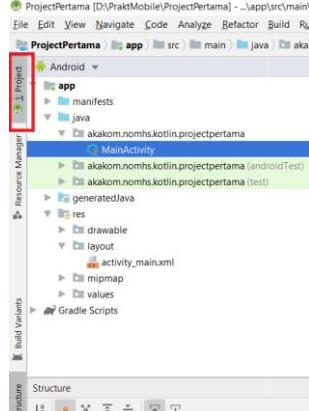
6. Jendela ProjectPertama akan muncul seperti pada gambar dibawah ini, dan tentunya belum secara utuh. Tungguah beberapa saat sampai proses loading fitur-fitur atau library dan tools muncul.



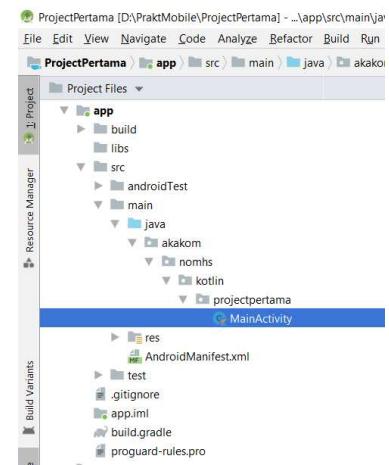
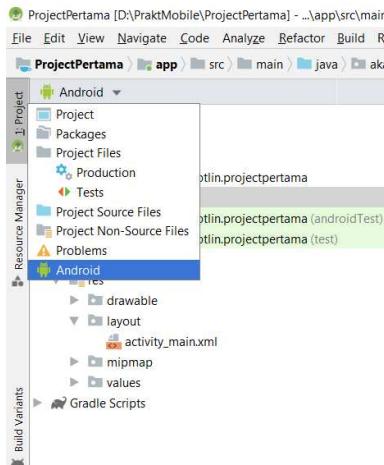
7. Jika proses loading fitur-fitur atau library selesai dan tools telah muncul, maka akan terlihat pada jendela IDE seperti pada gambar dibawah ini. Terdapat 3 bagian jendela pada Android Studio yaitu jendela Project, jendela Sourcecode dan jendela Built Output.



8. Pada jendela kiri atas, jika tab Project belum dipilih, klik tab Project.

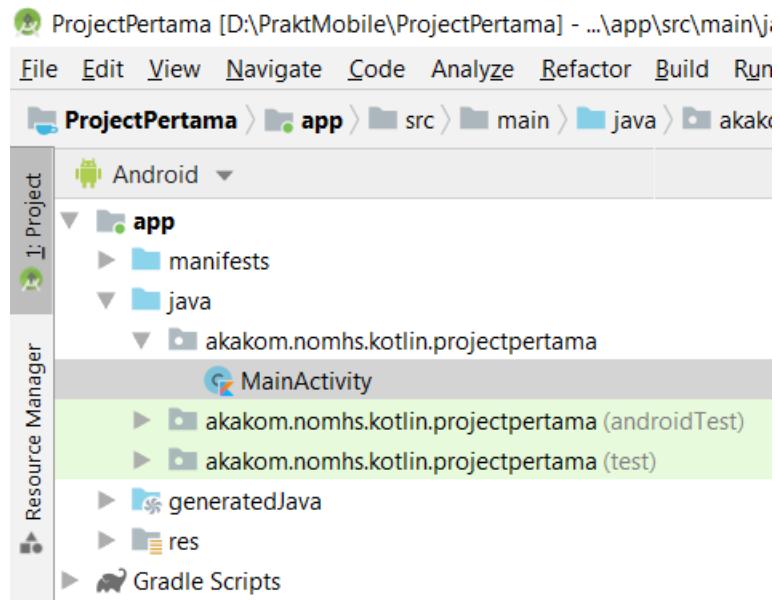


9. Hirarki Project dapat dipilih dengan menggunakan drop down menu. Hirarki standar adalah Android.

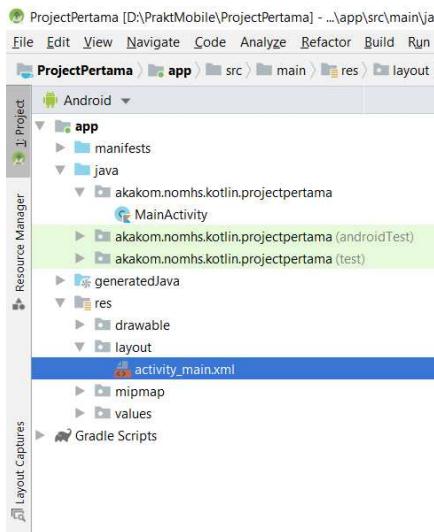


dan

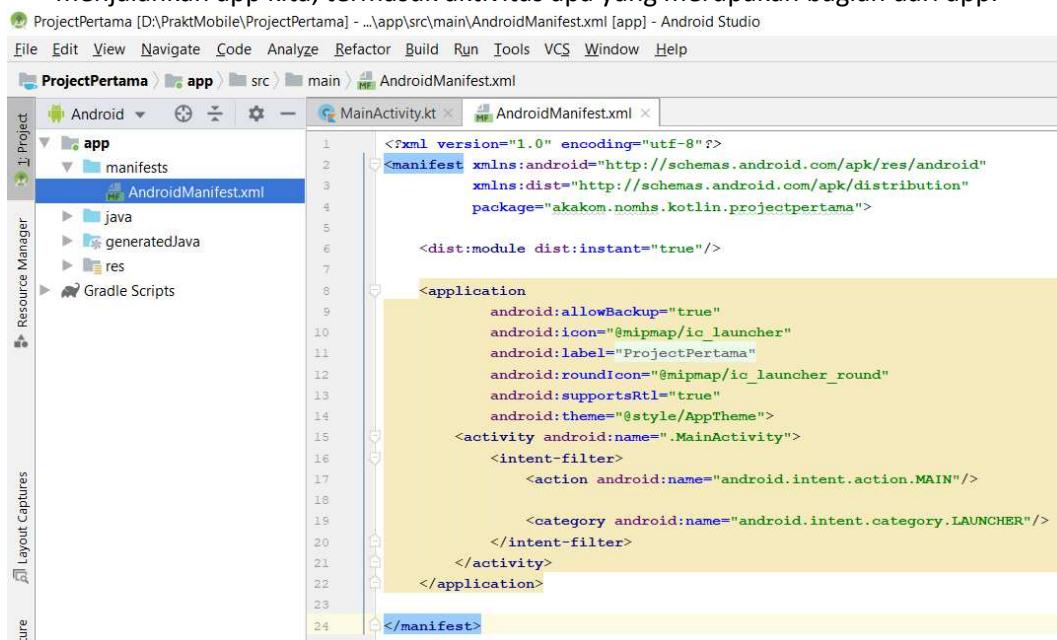
10. Untuk selanjutnya kita akan menggunakan hirarki Android.
 11. Sekarang akan kita lihat panel Project. Yang pertama kita mengeksplorasi folder app.
 12. Di panel Project> Android, eksplor folder app. Di dalam folder app ada empat subfolder: manifest, java, generateJava, dan res.
 13. Buka folder java, dan kemudian ekspan folder **akakom.nomhs.android.ProjectPertama** untuk melihat file MainActivity Kotlin.



14. Folder java berisi semua kode Kotlin utama untuk aplikasi Android. Ada alasan historis mengapa kode Kotlin Anda muncul di folder java. Konvensi itu memungkinkan Kotlin untuk beroperasi tanpa hambatan dengan kode yang ditulis dalam bahasa pemrograman Java, bahkan dalam proyek dan aplikasi yang sama.
15. File kelas aplikasi kita terkandung dalam tiga subfolder, seperti yang ditunjukkan pada gambar di atas. Folder **akakom.nomhs.android.ProjectPertama** berisi semua file untuk paket app. Secara khusus, kelas MainActivity adalah titik masuk utama untuk app kita. Dua folder lain di folder java digunakan untuk kode yang terkait dengan pengujian, seperti tes unit.
16. Dalam sistem file, file Kotlin memiliki ekstensi .kt dan ikon K. Pada tampilan Proyek, Android Studio menunjukkan nama kelas (MainActivity) tanpa ekstensi.
17. Catat folder **generatedJava**. Folder ini berisi file yang dihasilkan Android Studio saat membangun aplikasi. Jangan edit apa pun di folder ini, karena perubahan yang dilakukan, mungkin ditimpa ketika kita membangun kembali app. Tetapi penting untuk mengetahui tentang folder ini ketika kita perlu melihat file-file ini selama debugging.
18. Kemudian kita akan melihat folder berikutnya, yaitu res. Di panel Project> Android, ekspand folder res.
19. Folder res menyimpan sumber daya. Sumber daya di Android adalah konten statis yang digunakan dalam aplikasi kita. Sumber daya termasuk gambar, string teks, tata letak layar, gaya, dan nilai-nilai seperti warna heksadesimal atau dimensi standar.
20. Aplikasi Android memisahkan kode dan sumber daya Kotlin sebanyak mungkin. Itu membuatnya lebih mudah untuk menemukan semua string atau ikon yang digunakan di UI app. Juga, ketika kita mengubah salah satu file sumber daya ini, perubahan itu berlaku di mana-mana file tersebut digunakan dalam aplikasi.
21. Di dalam folder res, ekspand folder layout untuk melihat file activity_main.xml.



22. Activity kita biasanya dikaitkan dengan file tata letak UI, yang didefinisikan sebagai file XML di direktori res/layout. File tata letak itu biasanya dinamai berdasarkan aktivitasnya. Dalam hal ini, nama aktivitas adalah MainActivity, jadi layout yang terkait adalah activity_main.
23. Sekarang kita akan menjelajahi folder manifest dan AndroidManifest.xml
24. Folder manifes berisi file yang memberikan informasi penting tentang app kita ke sistem Android.
25. Buka folder manifest dan klik dua kali AndroidManifest.xml untuk membukanya. File AndroidManifest.xml mencakup detail yang dibutuhkan sistem Android untuk menjalankan app kita, termasuk aktivitas apa yang merupakan bagian dari app.



26. Perhatikan bahwa MainActivity direferensikan di elemen <activity>. Aktivitas apa pun di app kita harus dinyatakan dalam manifest. Contoh manifest untuk MainActivity.
27. Catat elemen <intent-filter> <activity>. Elemen <action> dan <category> dalam filter maksud ini memberi tahu Android tempat memulai app ketika pengguna mengklik ikon run.

28. File AndroidManifest.xml juga merupakan tempat kita menentukan izin apa pun yang dibutuhkan aplikasi kita. Izin mencakup kemampuan aplikasi kita untuk membaca kontak telepon, mengirim data melalui internet, atau mengakses perangkat keras seperti kamera perangkat.
29. Terakhir, kita akan menjelajahi folder Script Gradle.
30. Gradle adalah sistem otomasi bangunan yang menggunakan bahasa khusus domain untuk menggambarkan struktur, konfigurasi, dan dependensi project app. Ketika kita mengkompilasi dan menjalankan aplikasi kita, kita melihat informasi tentang Gradle build running. kita juga melihat informasi tentang Android Package Kit (APK) yang diinstal. (APK adalah format file paket yang digunakan sistem operasi Android untuk mendistribusikan dan menginstal aplikasi seluler.)
31. Kita akan fokus pada dua bagian pada bagian script program pada project Android yaitu .kt dan .xml. Untuk .kt dapat dibaca kotlin yaitu bahasa pemrograman pada Android Studio untuk membangun sistem. Sedangkan .xml dapat dibaca Extensible Markup Language yaitu bahasa pemrograman pada Android Studio untuk membangun user interface. **Klik pada bagian activity_main.xml untuk melihat design atau atampilan project Pertama.**

```

1 package akakom.nomhs.kotlin.projectpertama
2
3 import androidx.appcompat.app.AppCompatActivity
4 import android.os.Bundle
5
6 class MainActivity : AppCompatActivity() {
7
8     override fun onCreate(savedInstanceState: Bundle?) {
9         super.onCreate(savedInstanceState)
10        setContentView(R.layout.activity_main)
11    }
12 }
13

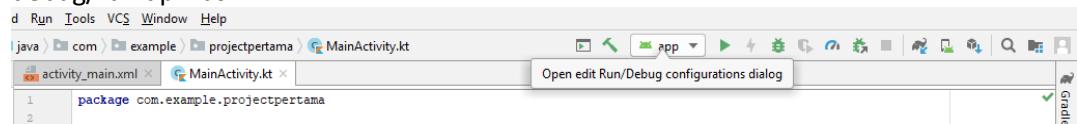
```

IV. Pengenalan Aplikasi Android dan IDE Android Studio

Untuk dapat membuat aplikasi mobile dengan Android Studio maka perlu mengenali lingkungan Aplikasi Android dan IDE Android Studio. Berikut beberapa IDE pada Android Studio yang nantinya akan digunakan selama proses pembuatan project :

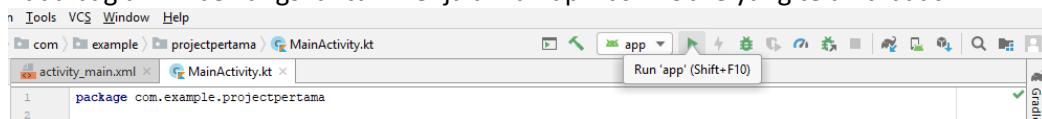
1. Open edit Run/Debug configurations dialog

Bagian ini berfungsi untuk memilih atau membuka, edit dan melakukan debug/run aplikasi.



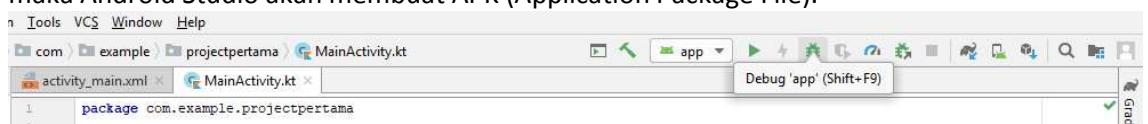
2. Run App

Pada bagian ini berfungsi untuk menjalankan aplikasi mobile yang telah dibuat.



3. Debug App

Debug App berfungsi untuk melakukan compile script dan melakukan pengecekan apakah terdapat script yang error atau tidak, jika script tidak mengalami error maka Android Studio akan membuat APK (Application Package File).



4. Profile App

Profile App merupakan alat pembuatan profil baru yang menyediakan data realtime untuk CPU, memori, dan aktivitas jaringan aplikasi Anda. Programmer dapat melakukan pelacakan metode berbasis sampel untuk mengukur waktu eksekusi script, merekam heap-dump, menampilkan alokasi memori, dan memeriksa detail file yang dikirim melalui jaringan.

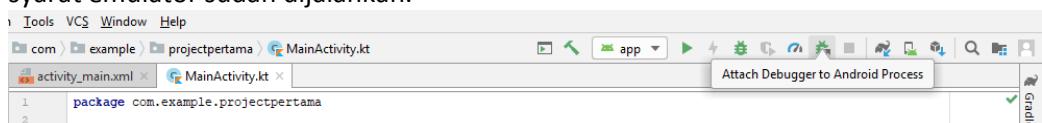


Dengan debugger Android Studio, maka programmer dapat :

- Memilih perangkat untuk men-debug pada aplikasi yang dibuat.
- Menyetel breakpoint dalam kode Java dan C/C++ pada aplikasi yang dibuat.
- Memeriksa variabel dan mengevaluasi ekspresi pada saat waktu proses.
- Mengambil tangkapan layar dan video dari aplikasi yang dibuat.

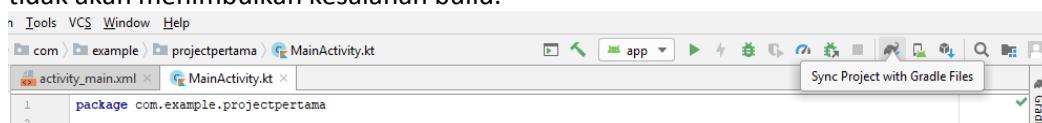
5. Attach Debugger to Android Process

Fungsi dari Attach Debugger to Android Proces yaitu melakukan debugging dan running proses dari script yang telah diubah atau update programmer dengan syarat emulator sudah dijalankan.



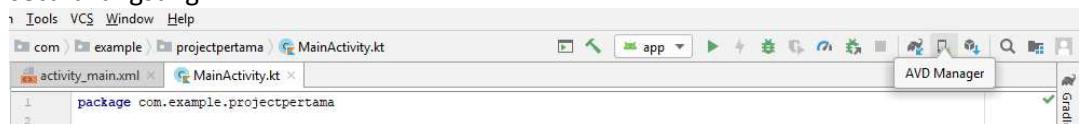
6. Sync Project with Gradle Files

Jika programmer membuat perubahan pada file konfigurasi build dalam project Android yang dibuat, maka Android Studio mewajibkan programmer melakukan sinkronisasi file project sehingga sistem dapat mengimpor perubahan konfigurasi build dan menjalankan beberapa pemeriksaan untuk memastikan konfigurasi tidak akan menimbulkan kesalahan build.



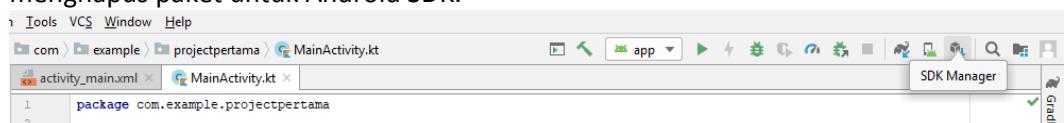
7. AVD Manager

AVD atau Android Virtual Device adalah fitur pada Android Studio untuk membuat device dalam bentuk virtual sehingga ketika aplikasi atau project dirunning maka akan muncul di virtual device tersebut. Namun AVD juga dapat untuk membuat physical device yaitu melakukan running aplikasi atau project melalui smartphone secara langsung.



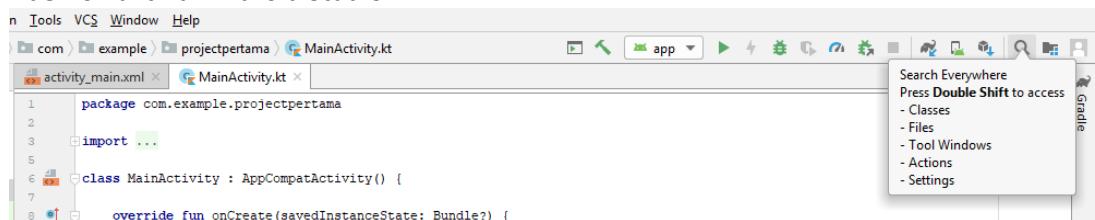
8. SDK Manager

SDK Manager adalah fitur untuk melihat, menginstal, memperbarui, dan menghapus paket untuk Android SDK.



9. Search Everywhere

Search Everywhere adalah sebuah fitur untuk menampilkan daftar Gradle Daemon aktif di Android Studio.



V. Materi Pengayaan

a. Penggunaan Kotlin untuk Pengembangan Android.

Kotlin/Native memungkinkan developer untuk menggunakan sebagai bahasa pemrograman dalam pengembangan aplikasi di platform lain seperti *embedded system*, desktop, macOS, dan iOS. Bahkan tak menutup kemungkinan Kotlin juga bisa digunakan untuk *data science* dan *machine learning*. Kotlin sangat cocok untuk mengembangkan aplikasi Android, membawa semua keunggulan bahasa modern ke platform Android tanpa memperkenalkan batasan baru:

1. **Compatibility.** Kotlin sepenuhnya kompatibel dengan JDK 6. Ini memastikan bahwa aplikasi yang dibangun dengan Kotlin dapat berjalan pada perangkat Android yang lebih lama tanpa ada masalah. Android Studio pun mendukung penuh pengembangan dengan bahasa Kotlin.
2. **Performance.** Dengan struktur *bytecode* yang sama dengan Java, aplikasi yang dibangun dengan Kotlin dapat berjalan setara dengan aplikasi yang dibangun dengan Java. Terdapat juga fitur seperti **inline function** pada Kotlin yang membuat kode yang dituliskan dengan **lambda** bisa berjalan lebih cepat dibandingkan kode yang sama dan dituliskan dengan Java.

3. **Interoperability.** Semua *library* Android yang tersedia, dapat digunakan pada Kotlin.
4. **Compilation Time.** Kotlin mendukung kompilasi inkremental yang efisien. Oleh karena itu, proses *build* biasanya sama atau lebih cepat dibandingkan dengan Java.

b. Memulai kotlin

Kita akan membuat program kotlin dengan dibandingkan dengan java. Gunakan laman web (<https://try.kotlinlang.org>) untuk mencoba menjalankan program kotlin.

Dikutip dari <https://kotlinlang.org/docs/reference/basic-syntax.html>

Defining packages.

Package specification should be at the top of the source file:

```
package my.demo

import java.util.*

// ...
```

It is not required to match directories and packages: source files can be placed arbitrarily in the file system.

See [Packages](#).

Defining functions

Function having two Int parameters with Int return type:

```
fun sum(a: Int, b: Int): Int {
    return a + b
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

Function with an expression body and inferred return type:

```
fun sum(a: Int, b: Int) = a + b
```

Target platform: JVMRunning on kotlin v. 1.3.41

Function returning no meaningful value:

```
fun printSum(a: Int, b: Int): Unit {
    println("sum of $a and $b is ${a + b}")
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

Unit return type can be omitted:

```
fun printSum(a: Int, b: Int) {
    println("sum of $a and $b is ${a + b}")
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [Functions](#).

Defining variables

Read-only local variables are defined using the keyword **val**. They can be assigned a value only once.

```
val a: Int = 1 // immediate assignment
val b = 2 // `Int` type is inferred
```

```
val c: Int // Type required when no initializer is provided  
c = 3 // deferred assignment
```

Target platform: JVMRunning on kotlin v. 1.3.41

Variables that can be reassigned use the var keyword:

```
var x = 5 // `Int` type is inferred  
x += 1
```

Target platform: JVMRunning on kotlin v. 1.3.41

Top-level variables:

```
val PI = 3.14  
var x = 0  
  
fun incrementX() {  
    x += 1  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

See also Properties And Fields.

Comments

Just like Java and JavaScript, Kotlin supports end-of-line and block comments.

```
// This is an end-of-line comment  
  
/* This is a block comment  
on multiple lines. */
```

Unlike Java, block comments in Kotlin can be nested.

See Documenting Kotlin Code for information on the documentation comment syntax.

Using string templates

```
var a = 1  
// simple name in template:  
val s1 = "a is $a"  
  
a = 2  
// arbitrary expression in template:  
val s2 = "${s1.replace("is", "was")}, but now is $a"
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [String templates](#).

Using conditional expressions

```
fun maxOf(a: Int, b: Int): Int {  
    if (a > b) {  
        return a  
    } else {  
        return b  
    }  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

Using **if** as an expression:

```
fun maxOf(a: Int, b: Int) = if (a > b) a else b
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [if-expressions](#).

Using nullable values and checking for null

A reference must be explicitly marked as nullable when **null** value is possible.

Return **null** if str does not hold an integer:

```
fun parseInt(str: String): Int? {  
    // ...  
}
```

Use a function returning nullable value:

```
fun printProduct(arg1: String, arg2: String) {  
    val x = parseInt(arg1)  
    val y = parseInt(arg2)  
  
    // Using `x * y` yields error because they may hold nulls.  
    if (x != null && y != null) {  
        // x and y are automatically cast to non-nullable after null check  
        println(x * y)  
    }  
    else {  
        println("either '$arg1' or '$arg2' is not a number")  
    }  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

or

```
// ...  
if (x == null) {  
    println("Wrong number format in arg1: '$arg1'")  
    return  
}  
if (y == null) {  
    println("Wrong number format in arg2: '$arg2'")  
    return  
}  
  
// x and y are automatically cast to non-nullable after null check  
println(x * y)
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [Null-safety](#).

Using type checks and automatic casts

The **is** operator checks if an expression is an instance of a type. If an immutable local variable or property is checked for a specific type, there's no need to cast it explicitly:

```
fun getStringLength(obj: Any): Int? {  
    if (obj is String) {  
        // `obj` is automatically cast to `String` in this branch  
        return obj.length  
    }  
}
```

// `obj` is still of type `Any` outside of the type-checked branch
return null

```
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

or

```
fun getStringLength(obj: Any): Int? {  
    if (obj !is String) return null  
  
    // `obj` is automatically cast to `String` in this branch  
    return obj.length  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

or even

```
fun getStringLength(obj: Any): Int? {  
    // `obj` is automatically cast to `String` on the right-hand side of `&&`  
    if (obj is String && obj.length > 0) {  
        return obj.length  
    }  
  
    return null  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [Classes](#) and [Type casts](#).

Using a for loop

```
val items = listOf("apple", "banana", "kiwifruit")  
for (item in items) {  
    println(item)  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

or

```
val items = listOf("apple", "banana", "kiwifruit")  
for (index in items.indices) {  
    println("item at $index is ${items[index]}")  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [for loop](#).

Using a while loop

```
val items = listOf("apple", "banana", "kiwifruit")  
var index = 0  
while (index < items.size) {  
    println("item at $index is ${items[index]}")  
    index++  
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [while loop](#).

Using when expression

```
fun describe(obj: Any): String =  
    when (obj) {  
        1      -> "One"  
        "Hello" -> "Greeting"  
        is Long   -> "Long"  
        !is String -> "Not a string"  
        else      -> "Unknown"  
    }
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [when expression](#).

Using ranges

Check if a number is within a range using `in` operator:

```
val x = 10
val y = 9
if (x in 1..y+1) {
    println("fits in range")
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

Check if a number is out of range:

```
val list = listOf("a", "b", "c")

if (-1 !in 0..list.lastIndex) {
    println("-1 is out of range")
}
if (list.size !in list.indices) {
    println("list size is out of valid list indices range, too")
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

Iterating over a range:

```
for (x in 1..5) {
    print(x)
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

or over a progression:

```
for (x in 1..10 step 2) {
    print(x)
}
println()
for (x in 9 downTo 0 step 3) {
    print(x)
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [Ranges](#).

Using collections

Iterating over a collection:

```
for (item in items) {
    println(item)
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

Checking if a collection contains an object using `in` operator:

```
when {
    "orange" in items -> println("juicy")
    "apple" in items -> println("apple is fine too")
}
```

Target platform: JVMRunning on kotlin v. 1.3.41

Using lambda expressions to filter and map collections:

```
val fruits = listOf("banana", "avocado", "apple", "kiwifruit")
fruits
    .filter { it.startsWith("a") }
    .sortedBy { it }
```

```
.map { it.toUpperCase() }  
.forEach { println(it) }
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [Higher-order functions and Lambdas.](#)

Creating basic classes and their instances:

```
val rectangle = Rectangle(5.0, 2.0) //no 'new' keyword required  
val triangle = Triangle(3.0, 4.0, 5.0)
```

Target platform: JVMRunning on kotlin v. 1.3.41

See [classes](#) and [objects and instances](#).



LATIHAN

1. Pelajari dan cobalah bahasa pemrograman kotlin lebih lanjut dari laman web kotlinlang.org



TUGAS

1. Install Android Studio pada perangkat komputer anda masing-masing di rumah
2. Silakan mempelajari bahasa pemrograman kotlin lebih lanjut dari laman web kotlinlang.org



REFERENSI

1. <https://kotlinlang.org/docs/reference/>
2. <https://developer.android.com/kotlin>
3. <https://developer.android.com/courses/kotlin-android-fundamentals/toc>
4. <https://codelabs.developers.google.com/android-kotlin-fundamentals/>
5. <https://developer.android.com/kotlin/learn>
6. <https://developer.android.com/kotlin/resources>

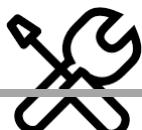
MODUL 2

MEMBUAT APLIKASI DAN MENJALANKANNYA



CAPAIAN PEMBELAJARAN

1. Mahasiswa mampu membuat aplikasi sederhana dengan desain standard yang disediakan dan menjalankan aplikasi di emulator maupun di perangkat mobile (keluaran)



KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. Android Studio 3.4.
2. Handphone Android versi 7.0 (Nougat)
3. Kabel data USB.
4. Driver ADB.



DASAR TEORI

Programmer yang menggunakan bahasa pemrograman berorientasi objek seperti Java akan terbiasa karena aplikasi Android ditulis di Kotlin, ini masih sangat banyak terjadi. Android, bagaimanapun juga, mengambil konsep yang dapat digunakan kembali komponen ke tingkat yang lebih tinggi.

Aplikasi Android diciptakan dengan menggunakan satu atau lebih komponen bersama, yang dikenal sebagai Activity. Sebuah Activity adalah satu, modul mandiri dari aplikasi yang biasanya berkorelasi langsung ke layar antarmuka pengguna. Activity dimaksudkan sebagai komponen, yang dapat digunakan kembali dan dapat dipertukarkan, dan bisa dibagi di antara aplikasi yang berbeda. Sebuah aplikasi email yang ada, misalnya, mungkin berisi Activity khusus untuk membuat dan mengirim pesan email. Seorang pengembang mungkin menulis sebuah aplikasi yang juga memiliki persyaratan untuk mengirim pesan email. Daripada mengembangkan Activity komposisi email khusus untuk aplikasi baru, pengembang hanya dapat menggunakan Activity dari aplikasi email yang ada.

Activity diciptakan sebagai subclass dari kelas Activity Android dan harus dieksekusi sehingga menjadi terpisah sepenuhnya dari Activity lain dalam aplikasi. Dengan kata lain, Activity bersama tidak bisa dipanggil langsung dalam program (karena aplikasi lain dapat menggunakan Aktivitas) dan satu Activity tidak bisa langsung memanggil metode atau mengakses data Activity lain. Sebagai gantinya, untuk mencapai tujuan ini, dengan menggunakan Intents dan Content Providers. Secara default, suatu Activity tidak dapat memberikan hasil dengan aktivitas yang ia dipanggil. Jika fungsi ini diperlukan, Activity harus secara khusus dimulai sebagai sub-aktivitas.

Aplikasi Manifest

File yang mengatur berbagai elemen dalam aplikasi adalah file Manifest. Berkas Manifest berbasis XML ini, menguraikan Activity, Service, Content Provider dan permissio yang membentuk suatu aplikasi secara lengkap. Selain file Manifest dan file Dex yang berisi kode-kode byte, paket aplikasi Android biasanya berisi kumpulan berkas Resources (sumber daya). Berkas ini mengandung sumber daya seperti string, gambar, huruf dan warna yang muncul dalam antarmuka pengguna secara bersama-sama, dengan representasi XML layout antarmuka pengguna. Secara default, berkas ini disimpan dalam /res, sub-direktori dalam hirarki proyek aplikasi.

Bila aplikasi dikompilasi, kelas bernama R dibuat, yang berisi referensi ke sumber daya aplikasi. File manifest dan sumber daya ini digabungkan untuk membuat apa yang dikenal sebagai Konteks Aplikasi. Konteks ini, diwakili oleh kelas Context Android, dapat digunakan dalam kode aplikasi untuk mendapatkan akses ke sumber daya aplikasi pada saat runtime. Selain itu, berbagai metode dapat dipanggil pada konteks aplikasi untuk mengumpulkan informasi dan membuat perubahan pada lingkungan aplikasi pada saat runtime.

Jelajahi file activity dan layout.

Kita akan fokus pada dua file paling penting yang membentuk aplikasi kita: File MainActivity Kotlin, dan file layout activity_main.xml.

Langkah 1: Periksa MainActivity

MainActivity adalah contoh Activity. Suatu Activity adalah kelas inti Android yang menggambarkan antarmuka pengguna aplikasi Android (UI) dan menerima acara masukan. Saat aplikasi kita diluncurkan, aplikasi meluncurkan aktivitas yang ditentukan dalam file AndroidManifest.xml. Banyak bahasa pemrograman menentukan metode utama yang memulai program. Aplikasi Android tidak memiliki metode utama. Sebaliknya, file AndroidManifest.xml menunjukkan bahwa MainActivity harus diluncurkan ketika pengguna mengetuk ikon peluncur aplikasi. Untuk meluncurkan suatu kegiatan, OS Android menggunakan informasi dalam manifes untuk mengatur lingkungan aplikasi dan membangun MainActivity. Kemudian MainActivity melakukan beberapa pengaturan secara bergantian. Setiap aktivitas memiliki file layout terkait. Activity dan layout dihubungkan oleh proses yang dikenal sebagai layout inflasi. Saat Activity dimulai, tampilan yang didefinisikan dalam file tata letak XML diubah menjadi (atau "digelembungkan" menjadi) objek tampilan Kotlin di memori. Setelah ini terjadi, Activity dapat menarik objek-objek ini ke layar dan juga secara dinamis memodifikasinya.

Run Emulator

Uji coba aplikasi wajib dilakukan seorang *developer*. Proses *running* atau *debugging* bisa dilakukan dengan dua cara, yaitu *running* dengan emulator atau peranti (*device*). Baik emulator maupun peranti memiliki kelebihan dan kekurangan masing-masing. Kita sebagai *developer* tinggal pilih mana yang sesuai keperluan.

Persiapan Running Menggunakan Emulator

Sebelum menggunakan emulator, pastikan beberapa hal berikut ini:

Virtualization

Untuk menjalankan emulator di dalam Android Studio, pastikan aspek virtualization. Sistem kita harus memenuhi persyaratannya, yakni ketentuan prosesor dan sistem operasi dari laptop / PC yang kita gunakan.

Processor

- Prosesor Intel: Jika laptop/pc kita menggunakan prosesor Intel, maka pastikan ia mendukung Intel VT-x, Intel EM64T (Intel 64), dan Execute Disable (XD) Bit functionality.
- Prosesor AMD: Jika laptop/pc kita menggunakan AMD, maka pastikan bahwa ia support dengan AMD Virtualization (AMD-V) dan Supplemental Streaming SIMD Extensions 3 (SSSE3).

Sistem Operasi

- Intel : Jika menggunakan processor Intel maka kita dapat menjalankannya di sistem operasi Windows, Linux, maupun Mac.
- AMD : Untuk prosesor AMD maka hanya bisa menjalankannya di sistem operasi Linux.

Menginstal Hardware Accelerated Execution Manager (HAXM)

Setelah memenuhi persyaratan di atas, langkah selanjutnya adalah menginstal HAXM. HAXM adalah *hardware-assisted virtualization engine* yang menggunakan teknologi VT dari Intel untuk mempercepat aplikasi Android yang diemulasi di mesin host. HAXM diperlukan untuk menjalankan emulator di Android Studio.

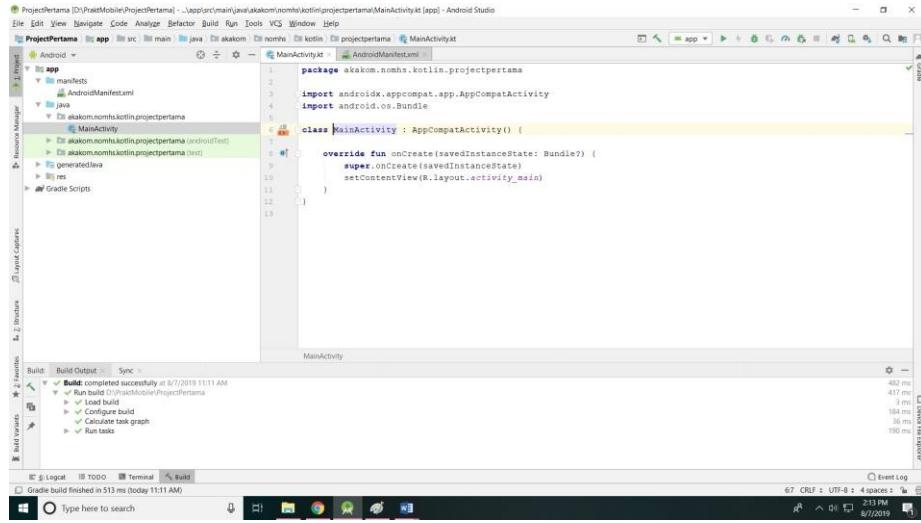
HAXM diperlukan jika sistem operasi yang kita gunakan adalah Windows atau Mac. Untuk menginstalnya, ikuti petunjuk berikut ini.

1. Buka SDK Manager.
2. Pilih SDK Update Sites, kemudian hidupkan Intel HAXM.
3. Tekan OK.
4. Cari berkas installer-nya di directory folder sdk komputer Anda, ~sdk\extras\intel\Hardware_Accelerated_Execution_Manager\intelhaxm-android.exe.
5. Jalankan installer dan ikuti petunjuknya sampai selesai.



PRAKTIK

1. Memulai membuat project baru, gunakan langkah2 yang sudah dikerjakan pada pertemuan pertama.
2. Buat menjadi tampilan seperti berikut



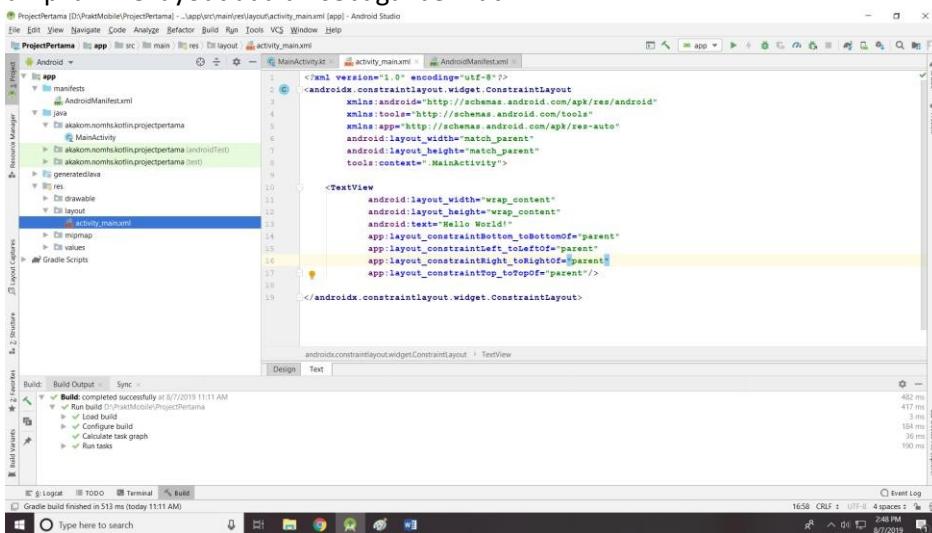
3. Pada MainActivity, terdapat beberapa coding yang dapat dijelaskan sebagai berikut.
 4. Kelas MainActivity extends AppCompatActivity.
- ```
class MainActivity : AppCompatActivity() { ... }
```
5. AppCompatActivity adalah subkelas Kegiatan yang mendukung semua fitur Android modern sambil memberikan kompatibilitas dengan versi Android yang lebih lama. Untuk membuat aplikasi kita tersedia untuk sejumlah besar perangkat dan pengguna mungkin, selalu gunakan AppCompatActivity.
  6. Perhatikan metode onCreate(). Activity tidak menggunakan konstruktor untuk menginisialisasi objek. Sebagai gantinya, serangkaian metode yang telah ditentukan (disebut "metode siklus hidup") disebut sebagai bagian dari pengaturan aktivitas. Salah satu metode siklus hidup tersebut adalah onCreate (), yang selalu ditimpa di aplikasi kita sendiri.
  7. Di onCreate (), kita menentukan layout mana yang dikaitkan dengan aktivitas, dan Anda mengembangkan tata letak. Metode setContentView () melakukan kedua hal itu.

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

8. Metode setContentView() mereferensikan layout menggunakan R.layout.activity\_main, yang sebenarnya merupakan referensi integer. Kelas R dihasilkan ketika kita membangun aplikasi kita. Kelas R mencakup semua asset aplikasi, termasuk konten direktori res.
9. Dalam kasus ini, R.layout.activity\_main merujuk ke kelas R yang dihasilkan, folder layout, dan file layout activity\_main.xml. (Sumber daya tidak termasuk ekstensi file.) Kita akan merujuk ke banyak sumber daya aplikasi (termasuk gambar, string, dan elemen dalam file layout) menggunakan referensi serupa di kelas R.
10. Periksa dan jelajahi file layout aplikasi. Semua activity di aplikasi kita memiliki file layout terkait di direktori res/layout aplikasi. File layout adalah file XML yang

mengungkapkan seperti apa sebenarnya aktivitas itu. File layout melakukan ini dengan menentukan tampilan dan menentukan di mana tampilan muncul di layar.

11. Tampilan adalah hal-hal seperti teks, gambar, dan tombol yang memperluas kelas tampilan. Ada banyak jenis tampilan, termasuk TextView, Button, ImageView, dan CheckBox.
12. Tampilan file layout adalah sebagai berikut.



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Hello World!"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintLeft_toLeftOf="parent"
 app:layout_constraintRight_toRightOf="parent"
 app:layout_constraintTop_toTopOf="parent"/>

```

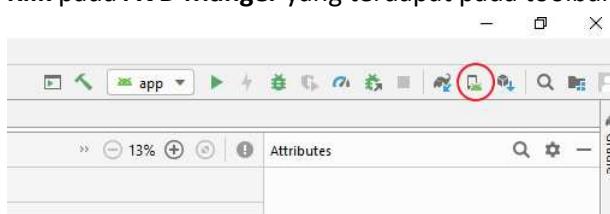
13. Untuk menjalankan aplikasi, dapat dilakukan dengan dua cara, pertama pada emulator, kedua pada perangkat mobile.

#### 14. Menjalankan pada Virtual Device.

Telah dijelaskan sebelumnya tentang Virtual Device yaitu konfigurasi yang mendefinisikan karakteristik ponsel Android, tablet, Wear OS, atau perangkat Android TV yang ingin disimulasikan di Android Emulator.

Dalam membuat Virtual Device dapat melalui langkah-langkah berikut :

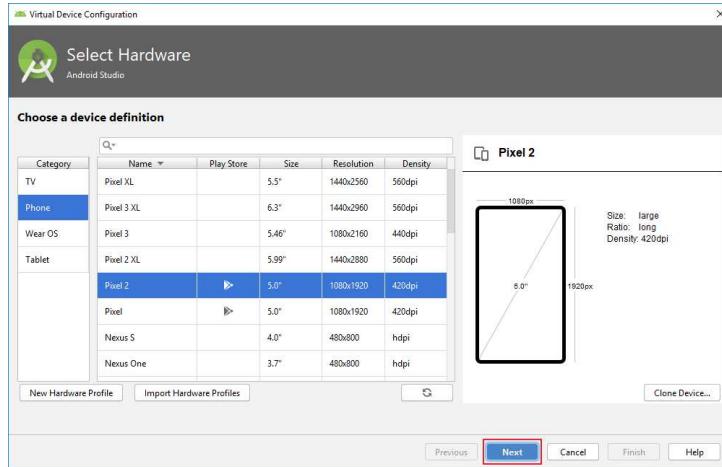
1. Klik pada AVD Manger yang terdapat pada toolbar Android Studio.



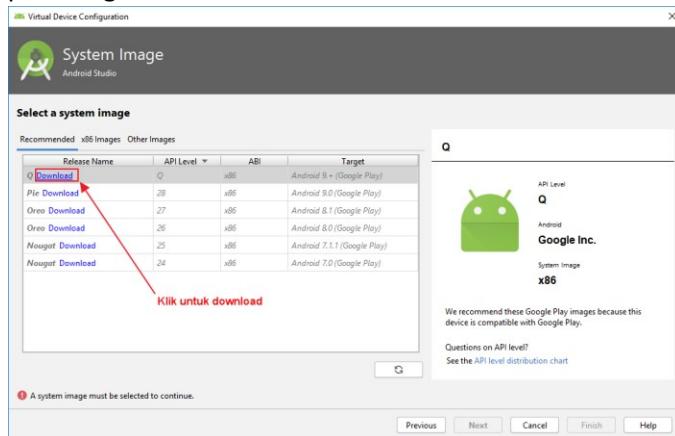
2. Pada jendela Android Virtual Device Manger, klik button + Create VirtualDevice.



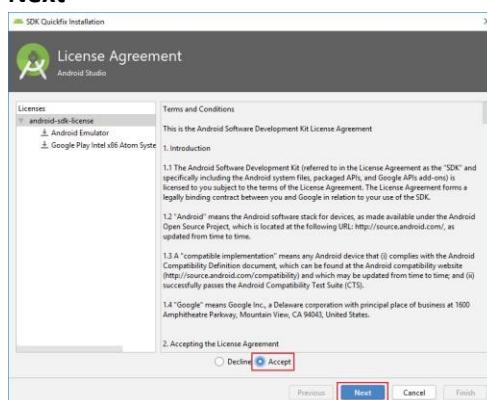
3. Pada jendela Virtual Device Configuration, pilih device yang akan digunakan untuk menampilkan hasil running project Android yang dibuat. Pilih pada **Pixel 2** kemudian klik button **Next**



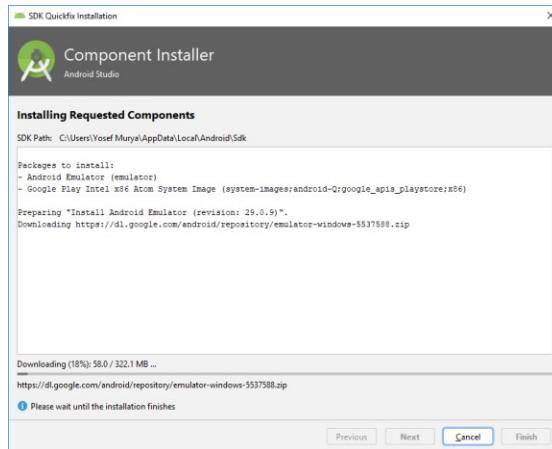
4. Jika pertamakali membuat emulator maka akan diminta untuk mendownload system images atau OS Android yang akan digunakan pada emulator. Silahkan klik pada bagian **Q download**



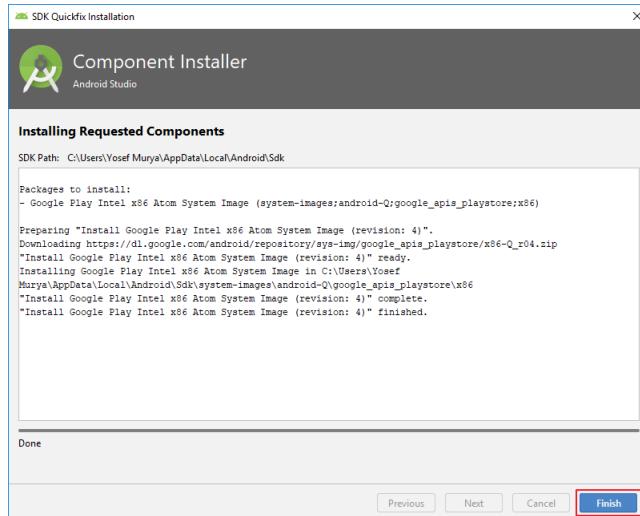
5. Pilih radio button **Accept** pada License Agreement, kemudian klik pada button **Next**



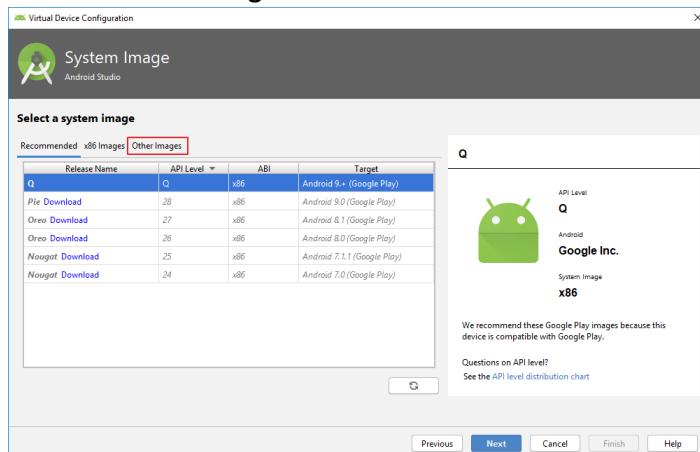
6. Tunggu beberapa saat sampai proses Download dan unzip selesai.



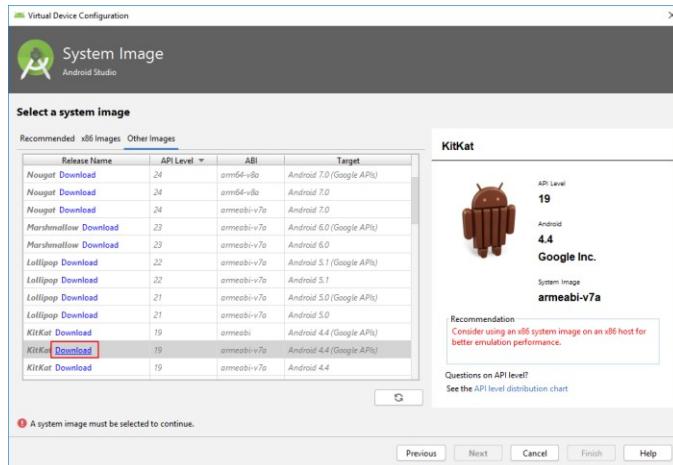
7. Setelah proses download dan unzipped selesai, klik button **Finish**



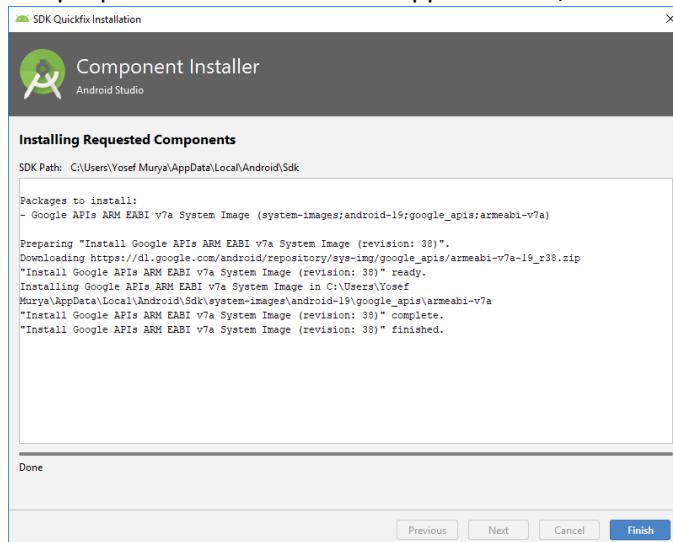
8. Langkah selanjutnya, pilih pada system image **Q** yang telah diinstall kemudian klik menu tab **other images**.



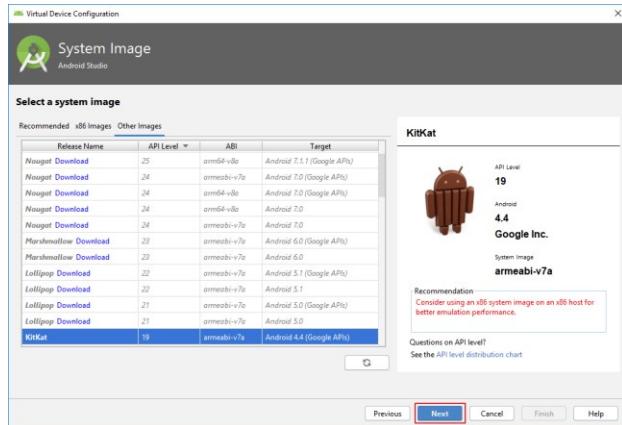
- Kemudian sesuaikan dengan rekomendasi (Recommendation), agar emulator dapat dijalankan dengan mudah. Dikarenakan hardware yang digunakan untuk menjalankan emulator Android yaitu Intel Dual Code dengan RAM 4GB maka penulis memilih system image Kitkat dengan API 19, ABI armeabi-v7a, selanjutnya klik Download.



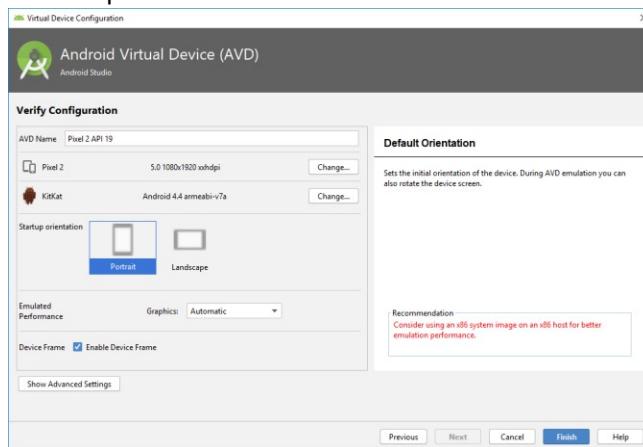
- Proses download memerlukan waktu (pastikan anda terkoneksi internet), tunggu sampai proses download dan unzipped selesai, kemudian klik button **Finish**.



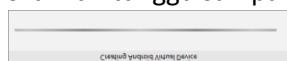
- Maka akan terlihat system image KitKat telah berhasil didownload, klik button **Next** untuk melanjutkan ke proses selanjutnya.



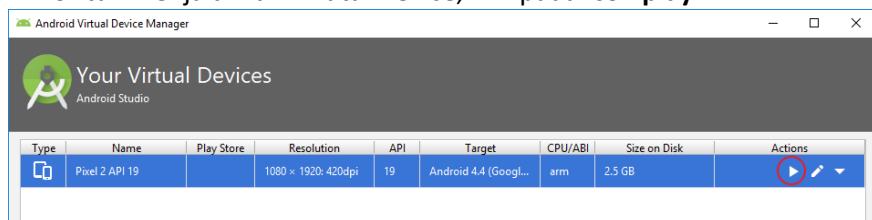
12. Pada bagian AVD Name akan terlihat secara default dengan nama Pixel 2 API 19 (dapat diganti dengan nama lain sesuai dengan kebutuhan), lalu pada bagian Startup orientation terdapat pilihan antara Potrait dan Landscape, secara default akan terpilih Potrait. **Klik button Finish.**



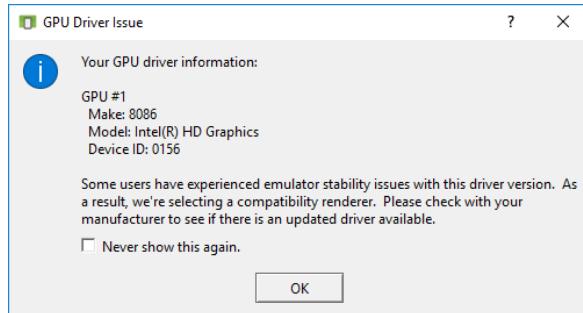
13. Proses pembuatan Android Virtual Device membutuhkan waktu beberapa detik, silahkan tunggu sampai proses pembuatan Android Virtual Device selesai.



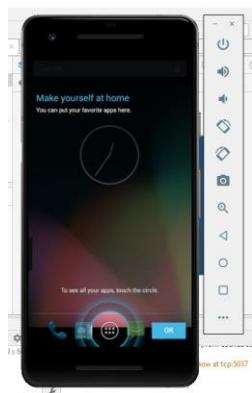
14. Jika pembuatan Android Virtual Device telah selesai dilakukan maka akan muncul Android Virtual Device yang telah dibuat seperti terlihat pada gambar dibawah ini. Untuk menjalankan Virutal Device, **klik pada icon play.**



15. Pada bagian GPU Driver Issue akan terlihat informasi tentang Graphic Processing Unit pada laptop atau komputer anda, **klik button OK.**



16. Maka akan muncul virtual device yang berfungsi untuk menampilkan hasil coding Android nantinya.



17. Jalankan aplikasi pada virtual device yang baru saja dibuat.



### 15. Physical Device.

Selain menggunakan virtual device dalam melakukan debug, anda dapat menggunakan physical device.

Bila Anda hendak melakukan *run* atau *debugging*, lebih baik Anda menjalankannya pada peranti *smartphone* asli. *Running* dengan menggunakan peranti memiliki beberapa kelebihan jika dibandingkan dengan emulator yaitu :

- Lebih cepat;
- Fitur seperti geo-location, push notif bisa digunakan;
- Bisa mengetahui daya serap baterai terhadap aplikasi;
- Lebih mudah.

Dengan menggunakan peranti smartphone asli, kita dapat memastikan bahwa aplikasi kita berjalan dengan wajar ketika sudah sampai di tangan pengguna. Kendala dari pendekatan ini adalah beragamnya model peranti yang ada di pasaran. Namun, pembahasan mengenai hal tersebut tidak tercakup dalam kelas ini.

Mari ikuti langkah-langkah untuk menjalankan proses *run* atau *debugging*. Tampilan dari langkah berikut bisa dipastikan akan berbeda dengan peranti yang Anda pakai. Akan tetapi secara garis besar langkahnya akan sama.

1. Pastikan peranti yang akan dipakai sesuai dengan target SDK atau paling tidak mendukung versi SDK terendah yang digunakan aplikasi.
2. Buka *setting* dan masuk ke dalam menu **About**. Pada halaman menu ini, Anda perlu menemukan informasi tentang **Build number**.

Berikut persiapan yang harus dilakukan :

1. Siapkan smartphone dan kabel data. Smartphone dengan OS Mobile Android, sedangkan kabel data adalah kabel yang dapat membaca data dari smartphone ke laptop (karena beberapa kasus terdapat kabel data yang hanya dapat digunakan untuk mengisi power atau batrei).



2. Penggunaan physical device dalam melakukan debug memerlukan beberapa pengaturan pada smartphone yaitu mengaktifkan developer option pada smartphone. Silakan mencari referensi untuk mengaktifkan developer option dari handphone yang anda pakai.
3. Kemudian aktifkan USB Debugging sampai pada bagian USB debugging terlihat aktif.

12:28 ... ☰ ⚡



## Developer options

### DEBUGGING

#### USB debugging

Debug mode when USB is connected



#### Revoke USB debugging authorizations



#### Install via USB

Allow installing apps via USB



#### USB debugging (Security settings)

Allow granting permissions and simulating input via USB debugging



#### Bug report shortcut

Show a button in the power menu for taking a bug report



4. Jalankan aplikasi Anda pada perangkat handphone.



## LATIHAN

1. Ganti tulisan Hello wold dengan tulisan “Selamat datang di Politeknik Banyuwangi”
2. Jalankan.
3. Tambahkan tulisan lain dengan memodifikasi koding pada layout.



## TUGAS

1. Buat aplikasi android dengan perangkat komputer anda di rumah dan jalankan pada perangkat anda.
2. Jelaskan tentang koding yang ada di file activity\_main.xml



## REFERENSI

1. <https://kotlinlang.org/docs/reference/>
2. <https://developer.android.com/kotlin>

3. <https://developer.android.com/courses/kotlin-android-fundamentals/toc>
4. <https://codelabs.developers.google.com/android-kotlin-fundamentals/>
5. <https://developer.android.com/kotlin/learn>
6. <https://developer.android.com/kotlin/resources>

## **MODUL 3**

### **Layout dengan Linear Layout dan Constraint Layout**



#### **CAPAIAN PEMBELAJARAN**

---

1. Mahasiswa mampu membuat Layout dengan Linear Layout dan Constraint Layout



#### **KEBUTUHAN ALAT/BAHAN/SOFTWARE**

---

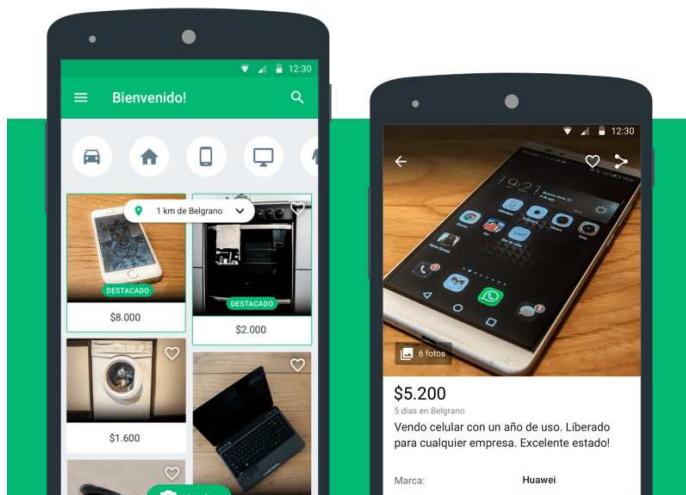
1. Android Studio 3.4.
2. Handphone Android versi 7.0 (Nougat)
3. Kabel data USB.
4. Driver ADB.



#### **DASAR TEORI**

---

Pada modul ini, kita akan mempelajari komponen View dan ViewGroup. Kedua komponen ini dapat berkolaborasi sehingga membentuk antar muka dengan contoh seperti pada gambar di bawah ini:



Pada dasarnya semua elemen antar pengguna di aplikasi Android dibangun menggunakan dua buah komponen inti, yaitu view dan viewgroup.

Sebuah view adalah obyek yang menggambarkan komponen tampilan ke layar yang mana pengguna dapat melihat dan berinteraksi langsung.

Contoh komponen turunan dari view seperti :

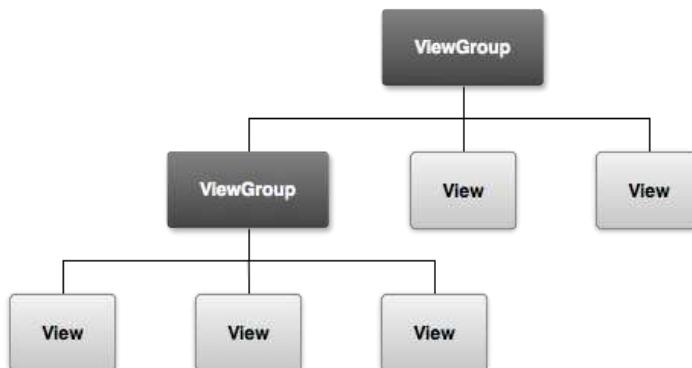
- **TextView**, komponen yang berguna untuk menampilkan teks ke layar.
- **Button**, komponen yang membuat pengguna dapat berinteraksi dengan cara ditekan untuk melakukan sesuatu.
- **ImageView**, Komponen untuk menampilkan gambar.
- **ListView**, komponen untuk menampilkan informasi dalam bentuk list.
- **GridView**, komponen untuk menampilkan informasi dalam bentuk grid.
- **RadioButton**, komponen yang memungkinkan pengguna dapat memilih satu pilihan dari berbagai pilihan yang disediakan.
- **Checkbox**, komponen yang memungkinkan pengguna dapat memilih lebih dari satu dari pilihan yang ada.

Sedangkan viewgroup adalah sebuah obyek yang mewadahi obyek-obyek view dan viewgroup itu sendiri sehingga membentuk satu kesatuan tampilan aplikasi yang utuh.

Contoh komponen viewgroup adalah:

- **LinearLayout**
- **FrameLayout**
- **RelativeLayout**
- **TableLayout**

Hierarki komponen view dan viewgroup dapat digambarkan dengan diagram berikut:



Jika diterjemahkan di dalam sebuah viewgroup akan ditampung dua buah komponen view dan satu komponen viewgroup yang terdiri dari 3 buah komponen view. Salah satu contoh dari tampilan dalam file layout xml untuk merepresentasikan kolaborasi view dan viewgroup seperti ini :

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3. android:layout_width="match_parent"
4. android:layout_height="match_parent"
5. android:orientation="vertical" >
6. <TextView android:id="@+id/text"
7. android:layout_width="wrap_content"
```

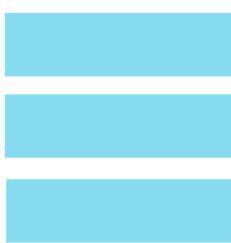
```
8. android:layout_height="wrap_content"
9. android:text="I am a TextView" />
10. <Button android:id="@+id/button"
11. android:layout_width="wrap_content"
12. android:layout_height="wrap_content"
13. android:text="I am a Button" />
14. </LinearLayout>
```

Obyek turunan viewgroup **LinearLayout** menjadi kontainer untuk obyek turunan view, button, dan textview. Beberapa komponen viewgroup seperti linearlayout, relativelayout, framelayou, dan tablelayout merupakan komponen yang paling banyak digunakan untuk menjadi *parent/root* dari komponen-komponen view.

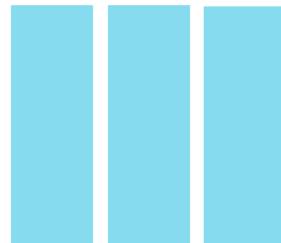
Berikut adalah definisi singkat dan inti dari komponen-komponen di atas terhadap penempatan komponen view (*child*) di dalamnya. Kita akan membahas Linear Layout dan Constrain Layout.

### LinearLayout

Layout ini akan menempatkan komponen-komponen di dalamnya secara horizontal atau vertikal. LinearLayout memiliki atribut *weight* untuk masing-masing *child* view yang berguna untuk menentukan porsi ukuran view dalam sebuah ruang (*space*) yang tersedia.



android:orientation="vertical"



android:orientation="horizontal"

**Constrain Layout. Apa itu ConstraintLayout?** (<https://blog.dicoding.com/kenal-lebih-dekat-dengan-constraintlayout/>)

**ConstraintLayout** merupakan salah satu komponen **ViewGroup** yang dapat kita gunakan untuk menyusun tampilan aplikasi yang kompleks tanpa adanya nested layout. ConstraintLayout tersedia dengan dukungan kompatibilitas mulai dari Android 2.3 (API Level 9) sampai dengan yang terbaru.

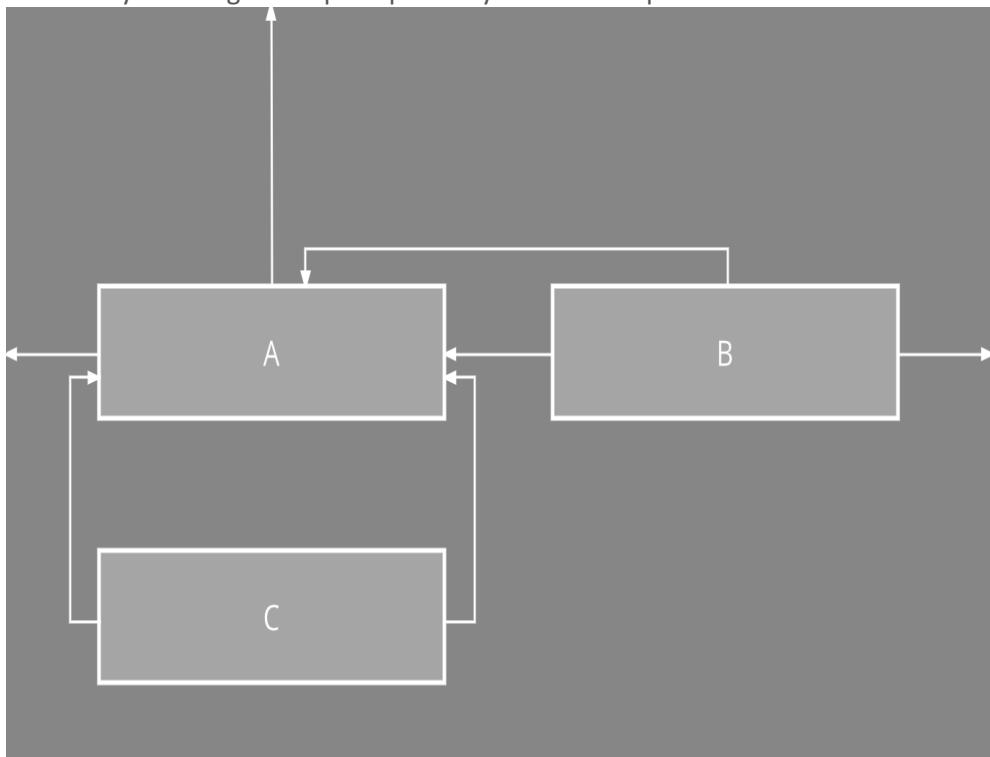
**ConstraintLayout** memiliki kesamaan dengan **RelativeLayout**. Dalam penggunaan semua view yang berada di dalamnya disusun berhubungan antara parent dan view lainnya. Tapi **ConstraintLayout** lebih fleksibel dari **RelativeLayout** dan mudah digunakan dengan dukungan Layout Editor pada Android Studio.

*Let's say* kita menambah view baru ke dalam **ConstraintLayout**. Kita gunakan *drag and drop* di Layout Editor yang berada pada tab **Design** atau dengan menambahnya secara manual melalui tab **Text**. Kita perlu menentukan posisi dari view atau bagaimana agar view tersebut terhubung dengan parent layout atau view lainnya.

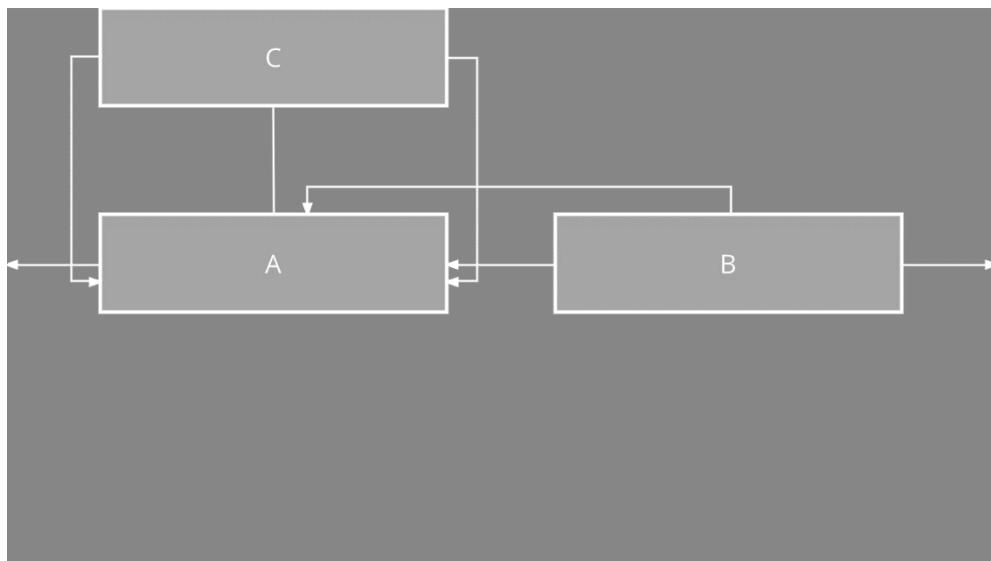
Kenapa gerangan? Karena setelah ditambahkan, view tersebut tidak memiliki **constraint** yang menghubungkannya dengan parent layout atau view lainnya. Sehingga ketika dijalankan, posisi dari view tersebut akan berada di bagian atas sebelah kiri.

Berbeda ceritanya dengan RelativeLayout. Saat kita ingin menentukan posisi atau menghubungkan dua buah view, kita bisa menggunakan attribute seperti **layout\_below** atau **layout\_above**. Nah untuk **ConstraintLayout** kita akan menggunakan **constraint** sebagai dasar dalam menentukan posisi agar sebuah view dapat terhubung dengan view lainnya sesuai harapan kita.

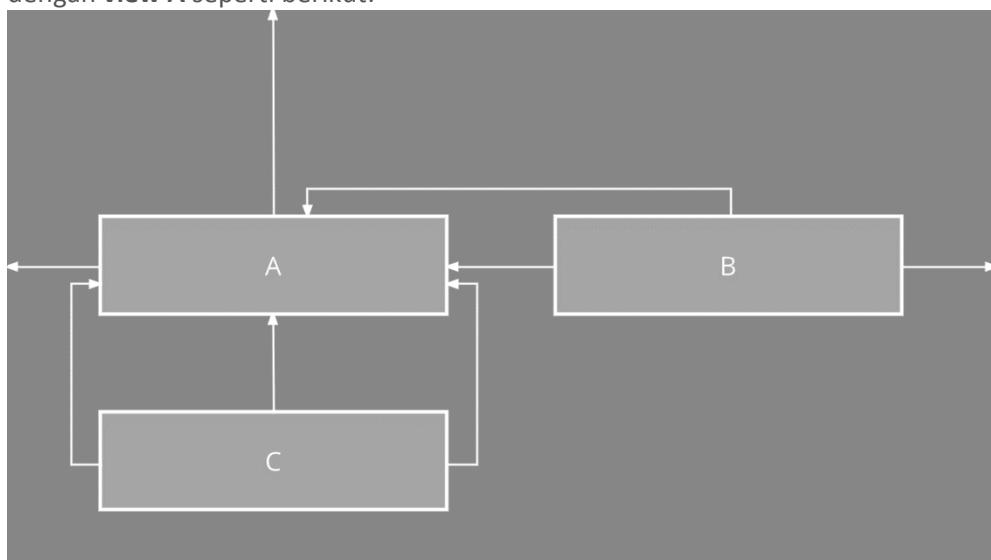
Setiap view setidaknya memiliki satu *vertical* dan *horizontal constraint*. Misal kita memiliki sebuah layout dengan tampilan pada Layout Editor seperti berikut:



Susunan tampilan di atas akan terlihat normal. Tidak ada yang salah di Layout Editor. Tapi jika kita perhatikan seksama, **view C** diatas hanya memiliki *horizontal constraint* yang diatur sejajar dengan **view A**. Sehingga ketika jika kita coba menjalankannya, sama seperti yang disebutkan diatas, maka posisi dari **view C** akan berada di posisi atas seperti berikut:



Berbeda jika kita menambahkan vertikal constraint pada **view C** yang diatur terikat dengan **view A** seperti berikut:



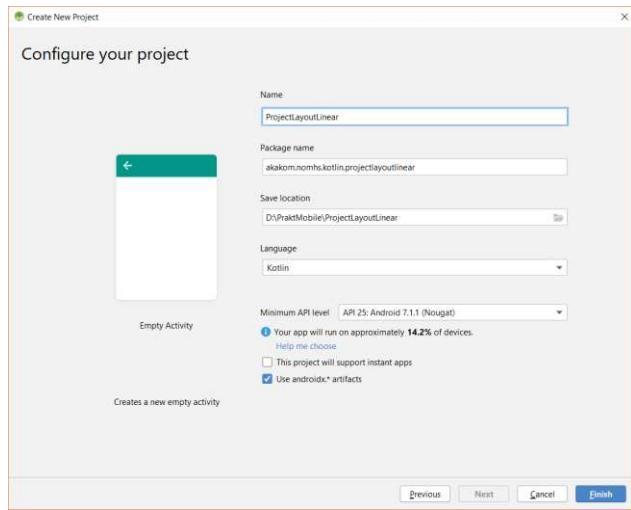
Ketika dijalankan, apa yang terjadi? Yang tampil akan sesuai dengan apa yang terlihat di Layout Editor.



## PRAKTIK

---

1. Buatlah project dengan nama ProjectLayoutLinear dengan cara **klik** menu **File → New → New Project ...**
2. Kemudian pilih **Empty Activity**, lalu **klik** button **Next**
3. Beri project anda yang baru dengan nama ProjectLayoutLinear, kemudian **klik** button **Finish**.



4. Langkah selanjutnya, buka file activity\_main.xml yang terdapat pada app → res → layout.
5. Secara default pada saat membuat project beru dengan nama ProjectLayoutLinear maka akan muncul teks Hello World.
6. Buka file activity\_main.xml kemudian pilih tab text, akan terlihat koding sebagai berikut.

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

 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Hello World!"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintLeft_toLeftOf="parent"
 app:layout_constraintRight_toRightOf="parent"
 app:layout_constraintTop_toTopOf="parent"/>

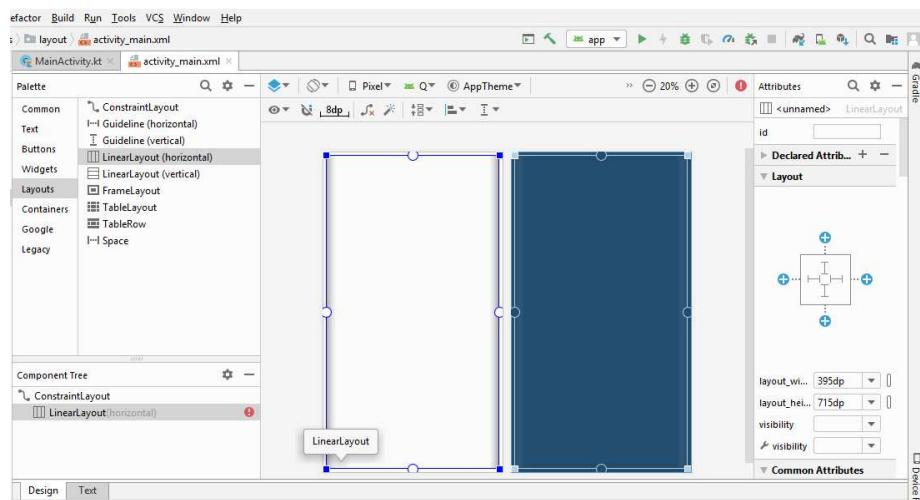
</androidx.constraintlayout.widget.ConstraintLayout>
```

7. Standar layout pertama adalah Constrain Layout. Ubahlah menjadi Linear Layout dengan koding berikut. Perhatikan atribut **orientation**.

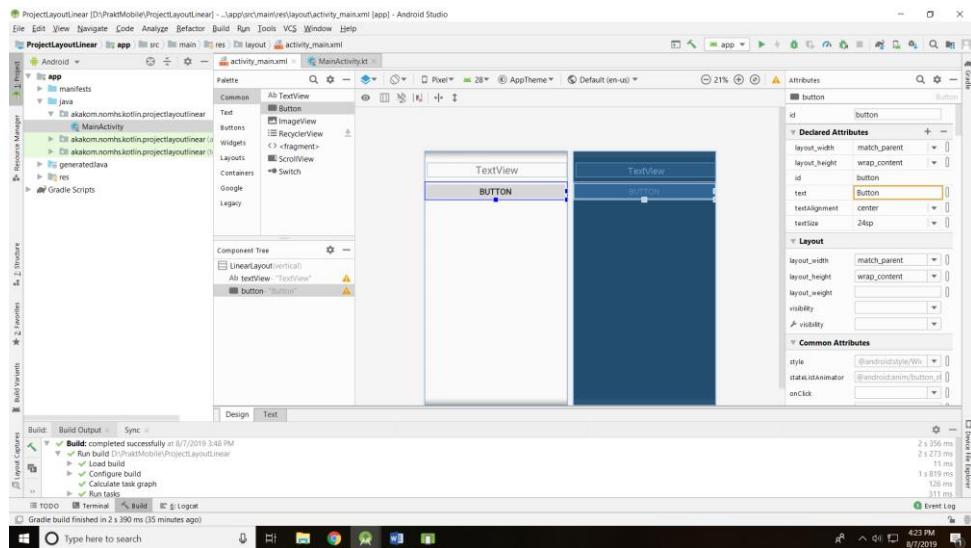
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"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

</LinearLayout>
```

- Maka hasilnya akan terlihat sebuah linear layout berhasil ditempatkan di bagian area atau canvas.



- Tambahkan komponen TextView dan Button, sehingga menjadi sebagai berikut.



- Jalankan dan amati hasilnya.
- Ubah atribut-atribut yang ada dan jalankan lagi, amati perubahannya.
- Kita akan mencoba menggunakan Constrain Layout. Buat project baru. Beri nama ProjectLayoutConstrain.
- Buat Project baru dengan nama ProjectLayoutConstrain.
- Buka file activity\_main.xml dan klik tab Desain.
- Anda akan menambahkan constrain secara manual, maka koneksi otomatis kita matikan. Di toolbar, temukan **Turn Off/On Autoconnect** toggle button, yang ditunjukkan di bawah ini. (Jika Anda tidak dapat melihat toolbar, klik di dalam area editor desain dari Layout Editor.) Pastikan autoconnect tidak aktif.



Autoconnect is on.

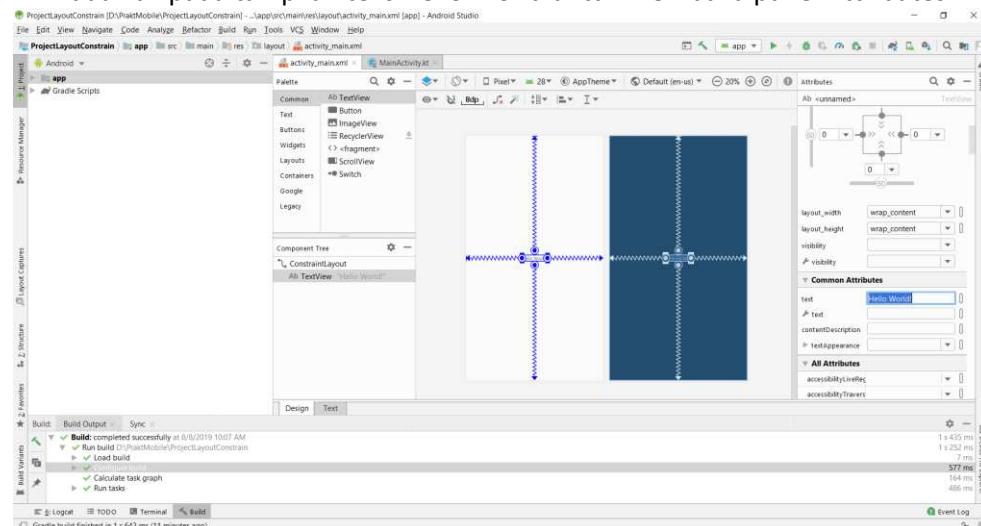


Autoconnect is off

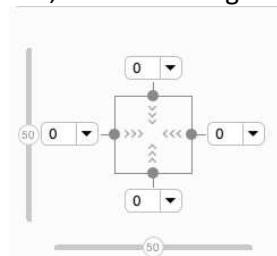
16. Gunakan toolbar untuk mengatur default margins ke 16dp. (Defaultnya adalah 8dp.)



17. Ketika Anda mengatur margin default ke 16dp, constrain baru dibuat dengan margin ini, jadi Anda tidak perlu menambahkan margin setiap kali Anda menambahkan constrain.
18. Perbesar menggunakan ikon  $\ominus 33\% + \oplus$  di sebelah kanan toolbar, hingga teks Hello World terlihat di dalam tampilan teksnya.
19. Klik dua kali pada tampilan teks Hello World untuk membuka panel Attributes.

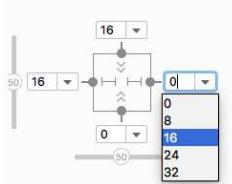


20. View Inspector, yang ditunjukkan pada gambar di bawah, adalah bagian dari panel Atribut. View Inspector mencakup kontrol untuk atribut layout seperti constrain, constraint types, constraint bias, and view margins.



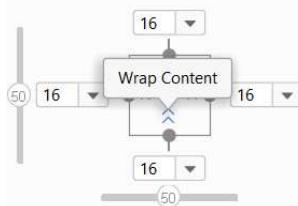
21. Constraint bias menempatkan elemen tampilan di sepanjang sumbu horizontal dan vertikal. Secara default, tampilan dipusatkan di antara dua constrain dengan bias 50%. Untuk menyesuaikan bias, Anda dapat menarik slider bias di view inspector. Menarik slider bias mengubah posisi tampilan sepanjang sumbu.

22. Tambahkan margin untuk TextView Hello World. Perhatikan bahwa dalam view inspector, margin kiri, kanan, atas, dan bawah untuk tampilan teks adalah 0. Margin default tidak ditambahkan secara otomatis, karena tampilan ini dibuat sebelum Anda mengubah margin default. Untuk margin kiri, kanan, dan atas, pilih 16dp dari menu drop-down di inspektor tampilan. Misalnya, dalam tangkapan layar berikut ini Anda menambahkan layout\_marginEnd (layout\_marginRight).



23. Sesuaikan batasan dan margin untuk TextView. Di view inspector, panah >>> di dalam kotak mewakili tipe constrain:

- a. >>> wrap content: Tampilan hanya selebar kontennya.



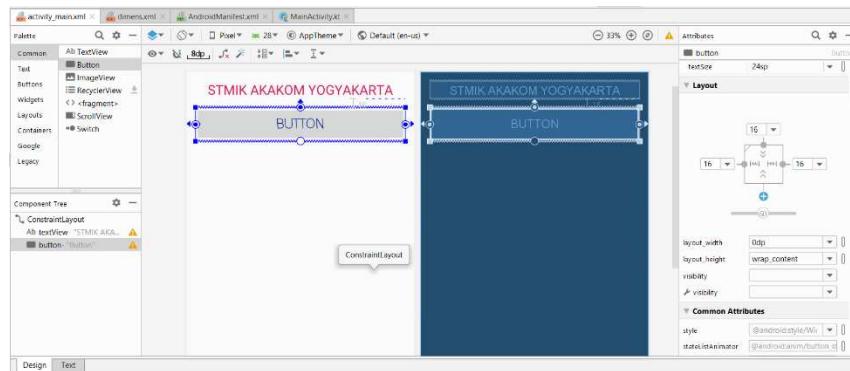
- b. ┌─┐ fixed: Anda dapat menentukan dimensi sebagai margin tampilan di kotak teks di sebelah panah constrain tetap.
- c. ┌─┐ match constrain: Tampilan melebar sebanyak mungkin untuk memenuhi constrain di setiap sisi, setelah memperhitungkan margin tampilan sendiri. Constrain ini sangat fleksibel, karena memungkinkan layout untuk beradaptasi dengan berbagai ukuran dan orientasi layar. Dengan membiarkan tampilan sesuai dengan constrain, Anda membutuhkan layout yang lebih sedikit untuk aplikasi yang Anda buat.
24. Di view inspector, ubah constrain kiri dan kanan ke Match Constraints. (Klik simbol panah untuk beralih di antara jenis constrain.)
25. Pindah ke tab Text, Ekstrak resource dimensi untuk layout\_marginStart, dan atur nama Resource ke margin\_wide. (blok pada bagian isian dari layout\_marginStart, pilih extract dimensions resource).



26. Kerjakan untuk layout\_marginEnd dan layout\_marginTop. Hasilnya adalah sebagai berikut

```
android:layout_marginTop="@dimen/margin_wide"
android:layout_marginStart="@dimen/margin_wide"
android:layout_marginEnd="@dimen/margin_wide"
```

27. Kemudian, tambahkan satu button, atur constrainnya. Hasilnya sebagai berikut.

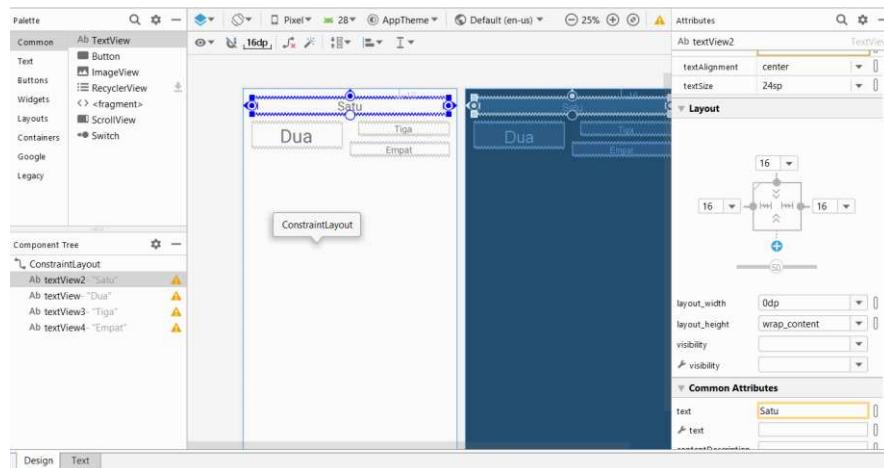


28. Jalankan dan amati hasilnya.
29. Ubah atribut-atribut yang ada dan jalankan lagi, amati perubahannya.



## LATIHAN

1. Buat project baru dengan desain sebagai berikut.



2. Buat project baru dengan menggunakan Linear Layout dengan minimal 3 komponen (TextView/Button) yang ditambahkan dan eksplorasilah atribut-atribut yang ada.
3. Buat project baru dengan menggunakan Constrain Layout dengan minimal 3 komponen (TextView/Button) yang ditambahkan dan eksplorasilah atribut-atribut yang ada.



## TUGAS

1. Buat Project pada perangkat komputer anda untuk mengimplementasikan layout dengan berbagai bentuk tampilan.



## REFERENSI

---

1. <https://kotlinlang.org/docs/reference/>
2. <https://developer.android.com/kotlin>
3. <https://developer.android.com/courses/kotlin-android-fundamentals/toc>
4. <https://codelabs.developers.google.com/android-kotlin-fundamentals/>
5. <https://developer.android.com/kotlin/learn>
6. <https://developer.android.com/kotlin/resources>

## **MODUL 4**

### **Komponen Widget View**



#### **CAPAIAN PEMBELAJARAN**

---

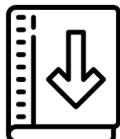
Mahasiswa mampu menggunakan Widget View (masukan) untuk membuat aplikasi sederhana



#### **KEBUTUHAN ALAT/BAHAN/SOFTWARE**

---

1. Android Studio 3.4.
2. Handphone Android versi 7.0 (Nougat)
3. Kabel data USB.
4. Driver ADB.



#### **DASAR TEORI**

---

Paket widget pada dasarnya merupakan visualisasi dari elemen user interface (UI) yang digunakan pada layar aplikasi Android di mana kita dapat merancang sendiri sesuai kebutuhan.

Widget di dalam Android ditampilkan dengan konsep *View*. Di mana aplikasi Android pada umumnya menggunakan widget sebagai Layout XML. Untuk mengimplementasikan widget, selain file kotlin kita juga membutuhkan tambahan dua file. Berikut ini adalah file-file yang umumnya kita butuhkan apabila kita membuat widget:

1. File Kotlin. Berupa file yang mengimplementasikan aksi dari widget. Jika kita mendefinisikan suatu widget beserta posisinya di layar yang didefinisikan dari file XML, kita harus melakukan coding di file kotlin yang dapat mengambil semua nilai atribut dari file layout XML yang didefinisikan.
2. File XML. Sebuah file yang mendefinisikan komponen elemen-elemen XML yang digunakan untuk inisialisasi widget serta atribut yang mendukungnya.
3. Layout XML. File XML menggambarkan atau penambahan keterangan pada layout widget kita.

Komponen widget TextView dan Button sudah kita bahas pada modul sebelumnya. Beberapa komponen widget akan kita bahas saat ini. Widget EditText untuk menuliskan teks ke aplikasi dan akan ditangkap oleh aplikasi untuk diolah. Widget Image Button untuk membuat button yang diberi gambar. Widget Image View untuk membuat tampilan gambar. Sedangkan widget RadioButton/ RadioGroup biasanya digunakan bersama-sama.

Di dalam satu RadioGroup terdapat beberapa RadioButton. Dan di dalam satu RadioGroup user hanya dapat melakukan satu check/pemilihan RadioButton. Dan yang terakhir widget akan kita bahas CheckBox, pilihan yang dapat dipilih lebih dari satu item.

#### **Event Handling.**

Android dapat menangani **event** dari interaksi dengan pengguna. Saat mempertimbangkan event dalam user interface, pendekatannya adalah menangkap event dari objek **View** tertentu yang digunakan pengguna untuk berinteraksi. Kelas View menyediakan sarana untuk melakukannya.

Dalam berbagai kelas View yang akan digunakan untuk menyusun layout, mungkin dapat dilihat beberapa method callback publik yang tampak berguna untuk kejadian UI. Method ini dipanggil oleh framework Android ketika masing-masing tindakan terjadi pada objek itu. Misalnya, jika View (seperti Button) disentuh, method onTouchEvent() akan dipanggil pada objek itu. Kelas View salah satunya berisi sekumpulan interface bertumpuk dengan callback yang mudah didefinisikan. Antarmuka ini, yang disebut event listener, digunakan untuk melakukan interaksi pengguna dengan UI.

#### **Event listener**

Event listener merupakan antarmuka di kelas View yang berisi method callback tunggal. Method ini akan dipanggil oleh framework Android jika View yang telah didaftarkan dengan listener dipicu oleh interaksi pengguna dengan item dalam UI.

Yang juga disertakan dalam antarmuka event listener adalah method callback berikut ini:

1. Method onClick() dari View.OnClickListener. Ini dipanggil baik saat pengguna menyentuh item (jika dalam mode sentuh), maupun memfokuskan pada item dengan tombol navigasi atau trackball dan menekan tombol "enter" yang sesuai atau menekan trackball.
2. Method onLongClick() dari View.OnLongClickListener. Ini dipanggil baik saat pengguna menyentuh dan menahan item (jika dalam mode sentuh), maupun memfokuskan pada item dengan tombol navigasi atau trackball dan menekan serta menahan tombol "enter" yang sesuai atau menekan dan menahan trackball (selama satu detik).
3. Method onFocusChange() dari View.OnFocusChangeListener. Ini dipanggil saat pengguna menyusuri ke atau dari item, dengan menggunakan tombol navigasi atau trackball.
4. Method onKey() dari View.OnKeyListener. Ini dipanggil saat pengguna memfokuskan pada item dan menekan atau melepas tombol perangkat keras pada perangkat.
5. Method onTouch() dari View.OnTouchListener. Ini dipanggil saat pengguna melakukan tindakan yang digolongkan sebagai peristiwa sentuh, termasuk penekanan, pelepasan, atau isyarat perpindahan pada layar (dalam batasan item itu).
6. Method onCreateContextMenu() dari View.OnCreateContextMenuListener. Ini dipanggil saat Menu Konteks sedang dibuat (akibat "klik lama" terus-menerus).

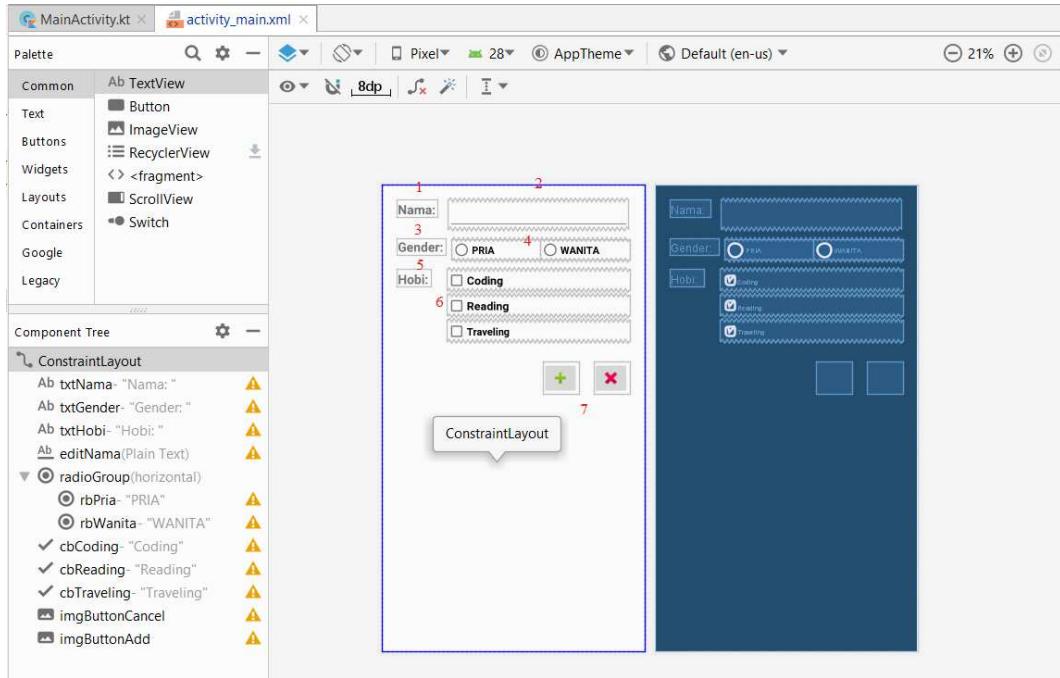


## **PRAKTIK**

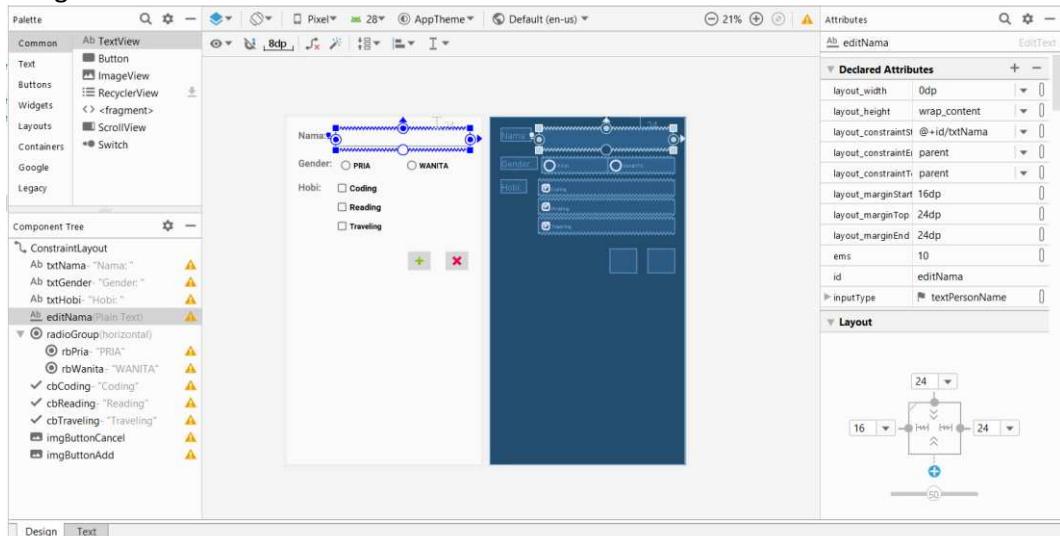
---

1. Buat Project Baru dengan nama **UIComponent**.

2. Kemudian, buat desain layout seperti pada gambar dibawah. Gunakan tab design dan klik - drag komponen yang diperlukan ke layar desain.



3. Perhatikan, atur atribut komponen widget dengan menggunakan tab attributes. Contoh, untuk komponen widget EditText, pengaturannya adalah seperti pada gambar di bawah.



4. Untuk keseluruhan komponen widget, perhatikan nilai atribut di setiap komponen di bawah.  
 5. Nomor 1, 3 dan 5 adalah komponen TextView.

```
<TextView
 android:text="Nama: "
 android:layout_width="wrap_content"
```

```

 android:layout_height="wrap_content"
 android:id="@+id/txtNama"
 android:layout_marginTop="24dp"
 app:layout_constraintTop_toTopOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 android:layout_marginStart="24dp"
 android:textSize="20sp"
 android:textStyle="bold"/>/>
<TextView
 android:text="Gender: "
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:id="@+id/txtGender"
 app:layout_constraintTop_toBottomOf="@+id/txtNama"
 app:layout_constraintStart_toStartOf="parent"
 android:layout_marginStart="24dp"
 android:layout_marginTop="32dp"
 android:textSize="20sp"
 android:textStyle="bold"/>/>
<TextView
 android:text="Hobi: "
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:id="@+id/txtHobi"
 app:layout_constraintTop_toBottomOf="@+id/txtGender"
 app:layout_constraintStart_toStartOf="parent"
 android:layout_marginStart="24dp"
 android:layout_marginTop="24dp"
 android:textSize="20sp"
 android:textStyle="bold"/>/>

```

6. Nomor 2 adalah komponen EditText.

```

<EditText
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:inputType="textPersonName"
 android:ems="10"
 android:id="@+id/editNama"
 android:layout_marginTop="24dp"
 app:layout_constraintTop_toTopOf="parent"
 app:layout_constraintStart_toEndOf="@+id/txtNama"
 android:layout_marginStart="16dp"
 android:layout_marginEnd="24dp"
 app:layout_constraintEnd_toEndOf="parent"/>/>

```

7. Nomor 4 ada dua macam komponen, satu buah RadioGroup dan dua buah RadioButton.

```

<RadioGroup
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 app:layout_constraintTop_toBottomOf="@+id/editNama"
 app:layout_constraintStart_toEndOf="@+id/txtGender"
 app:layout_constraintEnd_toEndOf="parent"
 android:layout_marginEnd="24dp"
 android:layout_marginTop="16dp"
 android:layout_marginStart="8dp"
 android:orientation="horizontal"
 android:id="@+id/radioGroup">
 <RadioButton
 android:text="PRIA"
 android:layout_width="match_parent"/>

```

```

 android:layout_height="wrap_content"
 android:id="@+id/rbPria"
 android:layout_weight="1"
 android:textSize="16sp"
 android:textStyle="bold"/>
 <RadioButton
 android:text="WANITA"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:id="@+id/rbWanita"
 android:layout_weight="1"
 android:textSize="16sp"
 android:textStyle="bold"/>
</RadioGroup>

```

8. Nomor 6 ada 3 komponen CheckBox.

```

<CheckBox
 android:text="Coding"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:id="@+id/cbCoding"
 app:layout_constraintStart_toEndOf="@+id/txtHobi"
 android:layout_marginStart="24dp"
 android:layout_marginTop="16dp"
 app:layout_constraintTop_toBottomOf="@+id/radioGroup"
 app:layout_constraintEnd_toEndOf="parent"
 android:layout_marginEnd="24dp"
 android:textSize="18sp"
 android:textStyle="bold"/>
<CheckBox
 android:text="Reading"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:id="@+id/cbReading"
 android:layout_marginTop="8dp"
 app:layout_constraintTop_toBottomOf="@+id/cbCoding"
 app:layout_constraintEnd_toEndOf="parent"
 android:layout_marginEnd="24dp"
 app:layout_constraintStart_toEndOf="@+id/txtHobi"
 android:layout_marginStart="24dp"
 android:textSize="18sp"
 android:textStyle="bold"/>
<CheckBox
 android:text="Traveling"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:id="@+id/cbTraveling"
 android:layout_marginEnd="24dp"
 app:layout_constraintTop_toBottomOf="@+id/cbReading"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toEndOf="@+id/txtHobi"
 android:layout_marginStart="24dp"
 android:layout_marginTop="8dp"
 android:textSize="16sp"
 android:textStyle="bold"/>

```

9. Nomor 7 adalah 2 komponen ImageButton. Perhatikan bahwa untuk ImageButton diperlukan file gambar, dalam contoh dibawah file gambarnya adalah ic\_delete dan ic\_input\_add.

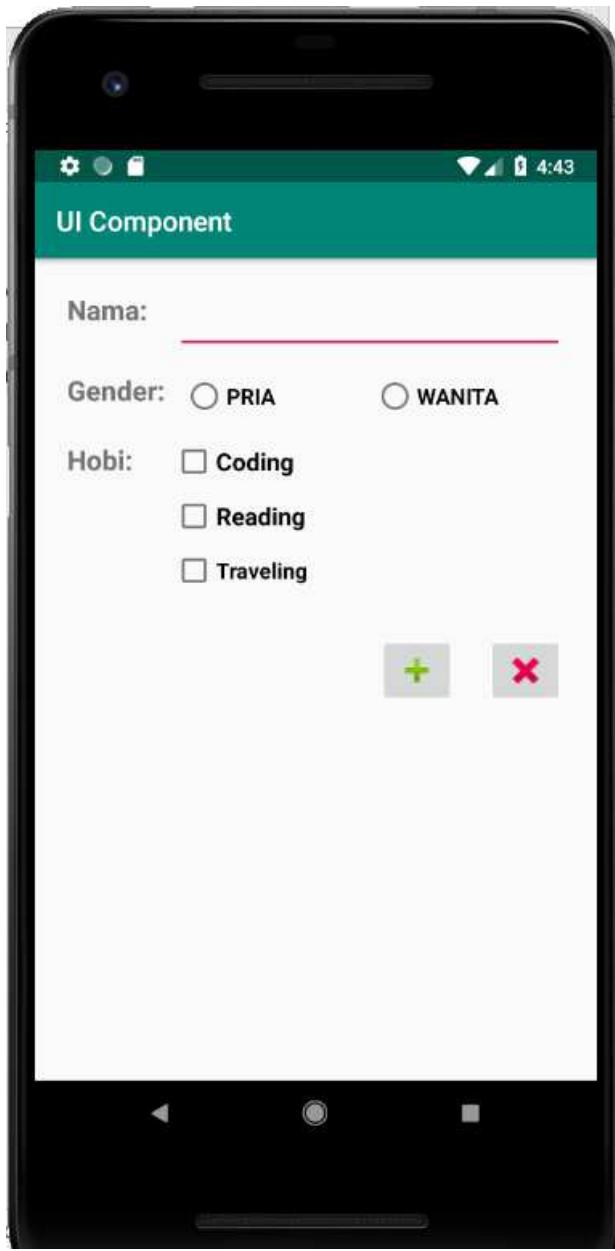
```

<ImageButton
 android:layout_width="wrap_content"

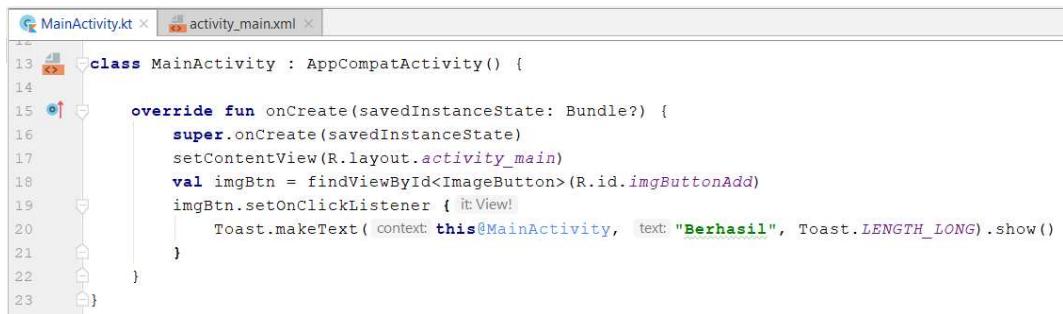
```

```
 android:layout_height="wrap_content"
 app:srcCompat="@android:drawable/ic_delete"
 android:id="@+id/imgButtonCancel"
 android:layout_marginTop="32dp"
 app:layout_constraintTop_toBottomOf="@+id/cbTraveling"
 app:layout_constraintEnd_toEndOf="parent"
 android:layout_marginEnd="24dp"/>"
<ImageButton
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 app:srcCompat="@android:drawable/ic_input_add"
 android:id="@+id/imgButtonAdd"
 app:layout_constraintEnd_toStartOf="@+id/imgButtonCancel"
 android:layout_marginEnd="24dp"
 android:layout_marginTop="32dp"
 app:layout_constraintTop_toBottomOf="@+id/cbTraveling"/>"
```

10. Setelah selesai desain, jalankan. Hasilnya adalah sebagai berikut.



11. Kita akan menambahkan event handling untuk ImageButton. Tambahkan koding pada MainActivity.kt, sehingga menjadi sebagai berikut.



```
13 class MainActivity : AppCompatActivity() {
14
15 override fun onCreate(savedInstanceState: Bundle?) {
16 super.onCreate(savedInstanceState)
17 setContentView(R.layout.activity_main)
18 val imgBtn = findViewById<ImageButton>(R.id.imgButtonAdd)
19 imgBtn.setOnClickListener { it: View!
20 Toast.makeText(context: this@MainActivity, text: "Berhasil", Toast.LENGTH_LONG).show()
21 }
22 }
23 }
```

12. Jalankan, dan beri event klik pada ImageButton plus.



## LATIHAN

---

1. Buat project baru, buat antar muka berbeda yang melibatkan komponen-komponen diatas.



## TUGAS

---

1. Analisislah atribut komponen untuk constrain layout.



## REFERENSI

---

1. <https://kotlinlang.org/docs/reference/>
2. <https://developer.android.com/kotlin>
3. <https://developer.android.com/courses/kotlin-android-fundamentals/toc>
4. <https://codelabs.developers.google.com/android-kotlin-fundamentals/>
5. <https://developer.android.com/kotlin/learn>
6. <https://developer.android.com/kotlin/resources>

## **MODUL 5**

### **RecyclerView**



#### **CAPAIAN PEMBELAJARAN**

---

1. Mahasiswa dapat merepresentasikan data dengan menggunakan komponen recyclerview



#### **KEBUTUHAN ALAT/BAHAN/SOFTWARE**

---

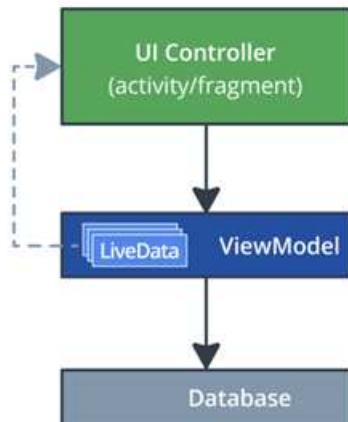
1. Android Studio 3.4.
2. Handphone Android versi 7.0 (Nougat)
3. Kabel data USB.
4. Driver ADB.



#### **DASAR TEORI**

---

RecyclerView adalah tampilan yang menggunakan arsitektur yang disederhanakan dengan UI controller, ViewModel, dan LiveData.



Menampilkan list atau grid data adalah salah satu tugas UI paling umum di Android. Daftar bervariasi dari yang sederhana hingga yang sangat kompleks. Daftar tampilan teks mungkin menampilkan data sederhana, seperti daftar belanja. Daftar yang kompleks, seperti daftar tujuan

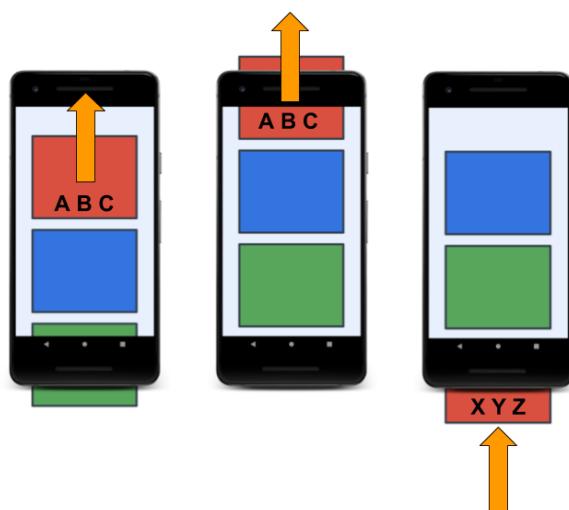
liburan yang beranotasi, dapat menunjukkan kepada pengguna banyak detail di dalam scrolling grid dengan header. Untuk mendukung semua kasus penggunaan ini, Android menyediakan widget RecyclerView.



Manfaat terbesar dari RecyclerView adalah sangat efisien untuk daftar besar:

- Secara default, RecyclerView hanya berfungsi untuk memproses atau menggambar item yang saat ini terlihat di layar. Misalnya, jika list memiliki seribu elemen tetapi hanya 10 elemen yang terlihat, RecyclerView hanya berfungsi untuk menggambar 10 item di layar. Ketika pengguna melakukan scroll, RecyclerView mengetahui item baru apa yang seharusnya ada di layar dan tidak cukup berfungsi untuk menampilkan item itu.
- Ketika suatu item scroll dari layar, tampilan item tersebut didaur ulang. Itu berarti item diisi dengan konten baru yang scroll ke layar. Perilaku RecyclerView ini menghemat banyak waktu pemrosesan dan membantu scroll list dengan lancar.
- Ketika suatu item berubah, alih-alih menggambar ulang seluruh daftar, RecyclerView dapat memperbarui satu item itu. Ini adalah keuntungan efisiensi yang sangat besar ketika menampilkan daftar item kompleks!

Dalam urutan yang ditunjukkan di bawah ini, kita dapat melihat bahwa satu tampilan telah diisi dengan data, ABC. Setelah itu tampilan bergulir dari layar, RecyclerView menggunakan kembali tampilan untuk data baru, XYZ.

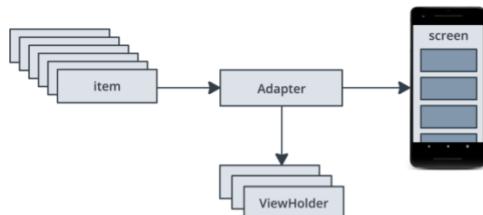


#### Adapter pattern

Jika kita pernah bepergian antar negara yang menggunakan soket listrik yang berbeda, kita mungkin tahu bagaimana kita bisa mencolokkan perangkat kita ke outlet dengan menggunakan adaptor. Adaptor memungkinkan kita mengonversi satu jenis steker ke yang lain, yang benar-benar mengubah satu antarmuka menjadi yang lain. Pola adaptor dalam rekayasa perangkat lunak

membantu objek bekerja dengan API lain. RecyclerView menggunakan adaptor untuk mengubah data aplikasi menjadi sesuatu yang dapat ditampilkan RecyclerView, tanpa mengubah cara aplikasi menyimpan dan memproses data. Untuk aplikasi pelacak tidur, kita membuat adaptor yang mengadaptasi data menjadi sesuatu yang RecyclerView tahu cara menampilkannya, tanpa mengubah ViewModel.

Mengimplementasikan sebuah RecyclerView



Untuk menampilkan data dalam RecyclerView, memerlukan bagian-bagian berikut:

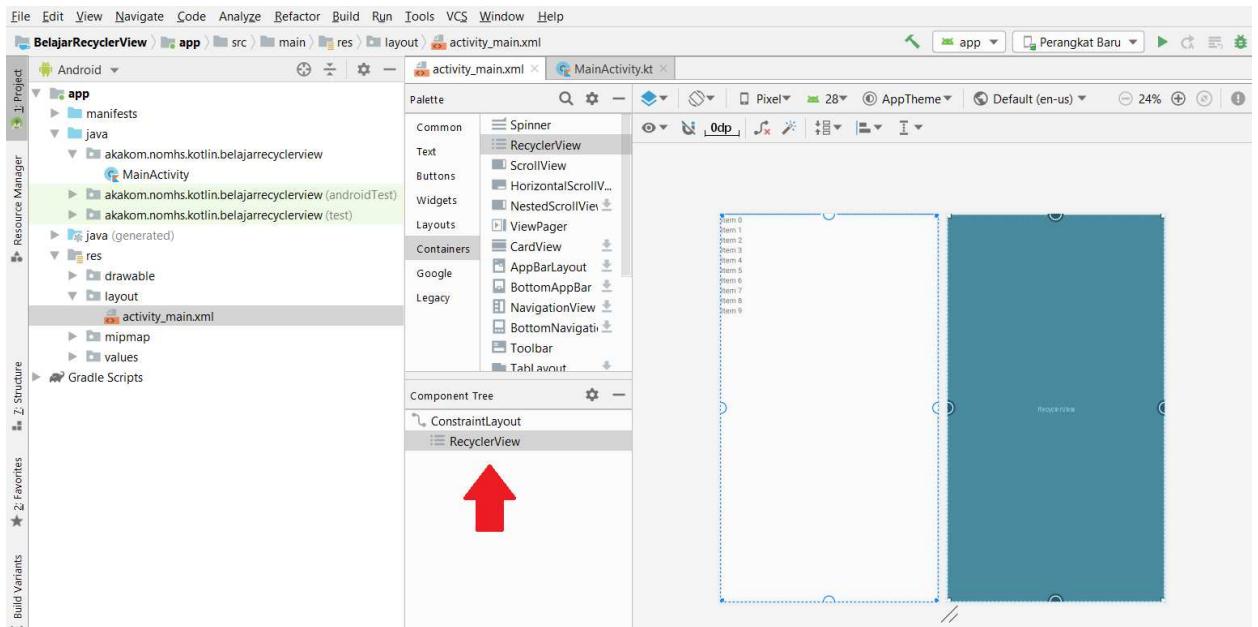
- Data untuk ditampilkan.
- Mesin virtual RecyclerView didefinisikan dalam file layout, untuk bertindak sebagai wadah untuk tampilan.
- Layout untuk satu item data.  
Jika semua item list terlihat sama, kita dapat menggunakan layout yang sama untuk semuanya, tetapi itu tidak wajib. Layout item harus dibuat secara terpisah dari layout fragmen, sehingga tampilan satu item pada satu waktu dapat dibuat dan diisi dengan data.
- Layout Manager.  
Layout Manager menangani organisasi (layout) komponen UI dalam tampilan.
- View holder.  
view holder extends kelas ViewHolder. Ini berisi informasi tampilan untuk menampilkan satu item dari layout item. Penampil tampilan juga menambahkan informasi yang digunakan RecyclerView untuk memindahkan tampilan di layar secara efisien.
- Adaptor.  
Adaptor menghubungkan data kita ke RecyclerView. Ini menyesuaikan data sehingga dapat ditampilkan di ViewHolder. RecyclerView menggunakan adaptor untuk mengetahui cara menampilkan data di layar.



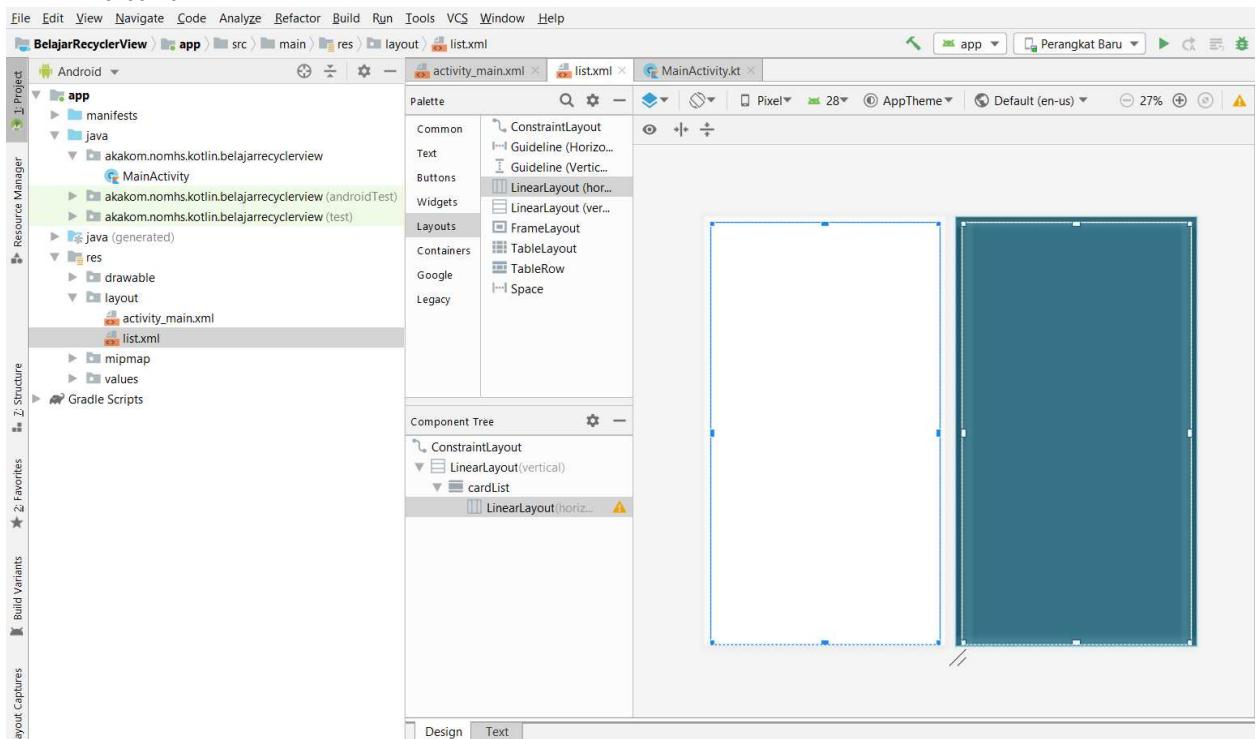
## PRAKTIK

---

1. Buat Project baru. (dalam contoh ini diberi nama BelajarRecyclerView)
2. Pada layout\_main.xml hapus komponen yang sudah ada (biasanya TextView).
3. Dari tab Palette, pilih Container, kemudian pilih RecyclerView
4. Klik dan drag ke Componen Tree, masukkan dibawah ConstraintLayout. Hasilnya seperti dibawah. (Perhatikan panah merah)



5. Buat sebuah file dibawah layout, beri nama list.xml
6. Tambahkan LinearLayout (vertical) didalam Container ConstraintLayout. Kemudian tambahkan cardaView dari Palette Container. Kemudian tambahkan LinearLayout(horisontal). Sampai sejauh ini, hasilnya dapat dilihat seperti gambar dibawah.

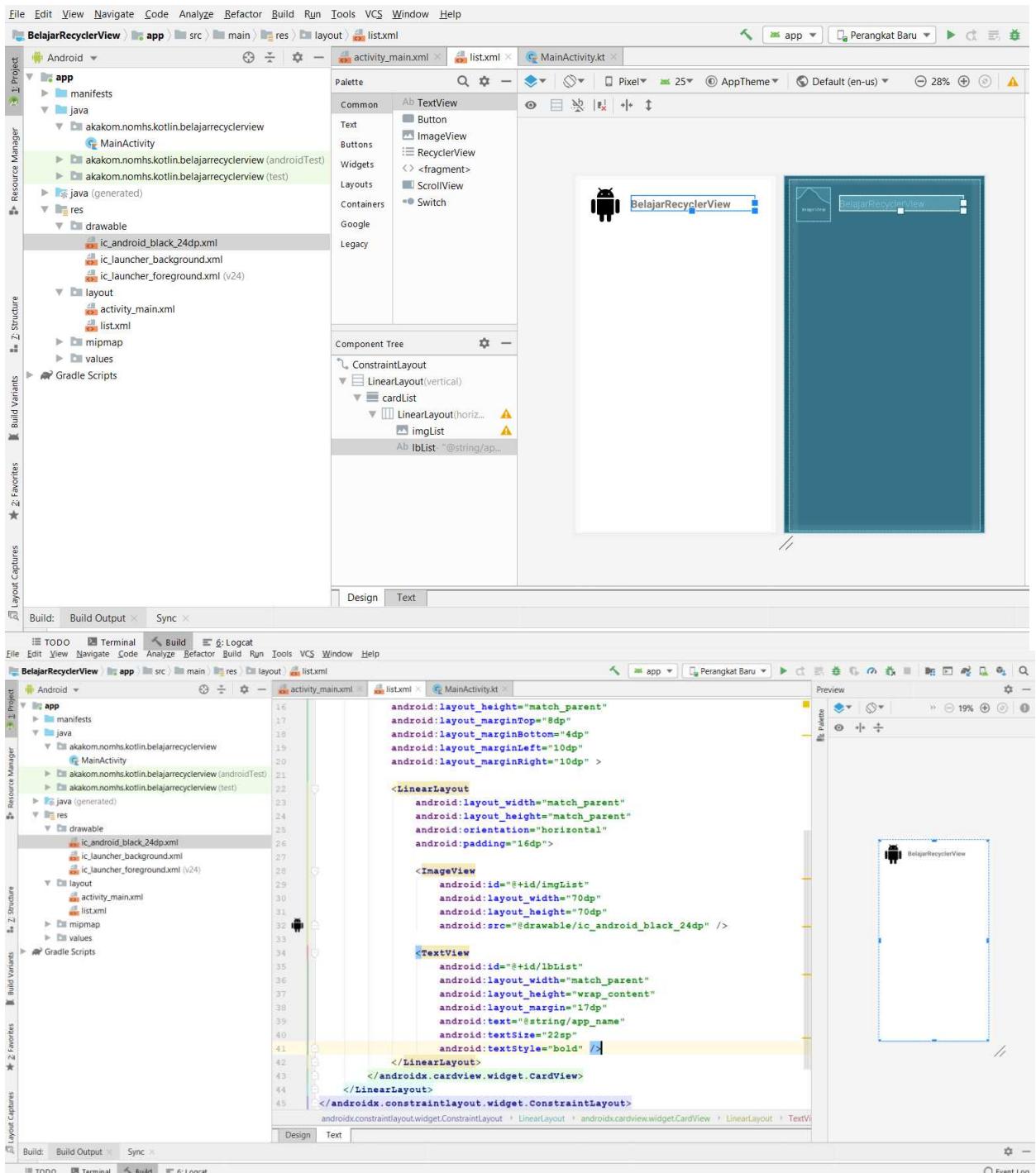


7. Kemudian, buka tab Text, ubah sehingga menjadi seperti dibawah ini.

The screenshot shows the Android Studio code editor with the tab bar at the top containing activity\_main.xml, list.xml, MainActivity.kt, Users.kt, and Adapter.kt. The code in activity\_main.xml is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
 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">
<LinearLayout
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical">
<android.support.v7.widget.CardView
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:id="@+id/cardList"
 android:layout_marginTop="8dp"
 android:layout_marginBottom="4dp"
 android:layout_marginLeft="10dp"
 android:layout_marginRight="10dp">
<LinearLayout
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="horizontal"
 android:padding="16dp">
</LinearLayout>
</android.support.v7.widget.CardView>
</LinearLayout>
</android.support.constraint.ConstraintLayout>
```

8. Kemudian tambahkan Image Button dan TextView dari tab Design dan kemudian modifikasi beberapa atribut dari tab Text. Perhatikan tampilan kedua tab dibawah.



- Buka file `MainActivity.kt`, modifikasiyah menjadi seperti berikut ini. List digunakan untuk membuat daftar String yang akan ditampilkan pada RecycleView. Kemudian daftar tersebut dimasukkan ke dalam Adapter dan kemudian Adapter dipasang pada RecycleView.

```
Adapter.kt x MainActivity.kt x list.xml x Users.kt x activity_main.xml x
9 class MainActivity : AppCompatActivity() {
10 val list = ArrayList<Users>()
11 val listUsers = arrayOf(
12 "Alpha",
13 "Bravo",
14 "Charlie",
15 "Delta",
16 "Echo",
17 "Foxtrot",
18 "Golf",
19 "Hotel",
20 "India",
21 "Juliet"
22)
23 override fun onCreate(savedInstanceState: Bundle?) {
24 super.onCreate(savedInstanceState)
25 setContentView(R.layout.activity_main)
26 mRecyclerview.setHasFixedSize(true)
27 mRecyclerview.layoutManager = LinearLayoutManager(context: this)
28 for (i in 0 until listUsers.size) {
29 list.add(Users(listUsers.get(i)))
30 if(listUsers.size - 1 == i){
31 val adapter = Adapter(list)
32 adapter.notifyDataSetChanged()
33 mRecyclerview.adapter = adapter
34 }
35 }
36 }
37 }
```

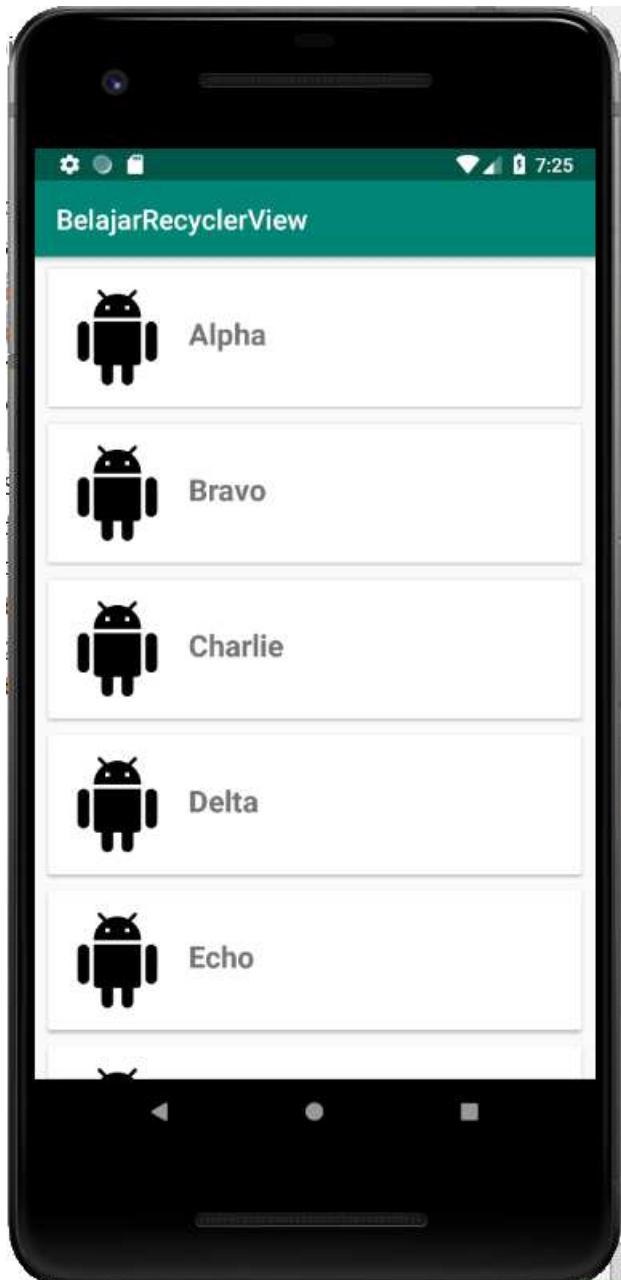
10. Setelah itu buat file kotlin dibawah package yang sama dengan file MainActivity.kt dengan nama Users.kt dan Adapter.kt. Program Adapter.kt ini digunakan untuk membuat Adapter dari RecyclerView. Dan class Users.kt digunakan untuk menampung daftar.

```
Adapter.kt x MainActivity.kt x list.xml x Users.kt x activity_main.xml x
9 class Adapter(private val list:ArrayList<Users>) : RecyclerView.Adapter<Adapter.Holder>() {
10
11 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): Holder {
12 return Holder(LayoutInflater.from(parent.context).inflate(R.layout.list,parent, attachToRoot: false))
13 }
14
15 override fun getItemCount(): Int = list.size
16
17 override fun onBindViewHolder(holder: Holder, position: Int) {
18 holder.view.lbList.text = list.get(position).name
19 }
20
21 class Holder(val view: View) : RecyclerView.ViewHolder(view)
22 }
```

```
Adapter.kt | MainActivity.kt | list.xml | Users.kt | activity_main.xml |
```

```
3 data class Users (val name:String?)
4
5
6
```

11. Jalankan. Dan anda akan mendapatkan hasil seperti berikut ini. Scroll ke bawah untuk mendapatkan list berikutnya.





## LATIHAN

---

1. Modifikasilah aplikasi dengan menambahkan Toast jika salah satu list dipilih.



## TUGAS

---

1. Buat aplikasi baru dengan menerapkan RecyclerView



## REFERENSI

---

1. <https://kotlinlang.org/docs/reference/>
2. <https://developer.android.com/kotlin>
3. <https://developer.android.com/courses/kotlin-android-fundamentals/toc>
4. <https://codelabs.developers.google.com/android-kotlin-fundamentals/>
5. <https://developer.android.com/kotlin/learn>
6. <https://developer.android.com/kotlin/resources>

## **MODUL 6**

### **Project Terintegrasi**



#### **CAPAIAN PEMBELAJARAN**

---

1. Mahasiswa dapat membuat sebuah Projek Aplikasi Android sederhana yang melibatkan komponen-komponen yang sudah dipelajari.



#### **KEBUTUHAN ALAT/BAHAN/SOFTWARE**

---

1. Android Studio 3.4.
2. Handphone Android versi 7.0 (Nougat)
3. Kabel data USB.
4. Driver ADB.

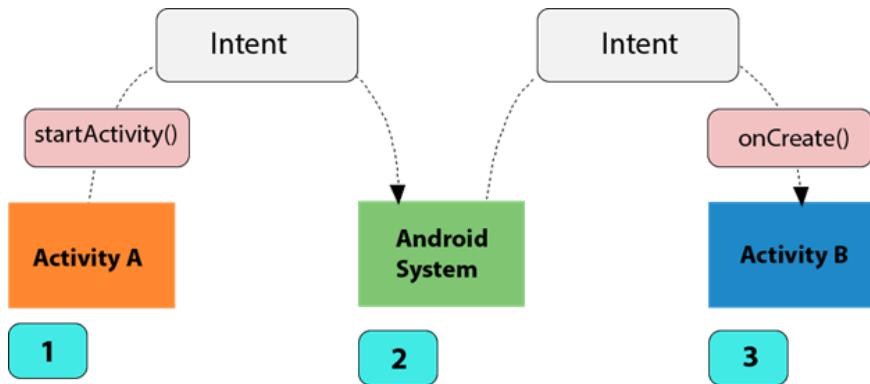


#### **DASAR TEORI**

---

Sebuah aplikasi tidak mungkin hanya melibatkan satu komponen dalam pembuatannya. Untuk itu, perlu untuk mempelajari bagaimana membuat aplikasi yang mengintegrasikan beberapa komponen. Project ini akan digunakan untuk mengintegrasikan komponen-komponen membentuk aplikasi yang fungsional. Selain itu, digunakan juga intent untuk membuat aplikasi yang berhubungan dengan halaman lain.

Apa itu intents? Android menggunakan *intents* untuk melakukan pekerjaan tertentu di dalam aplikasinya. Begitu kita menguasai penggunaan intents, maka semua pengembangan aplikasi baru yang ada akan terbuka. Pada pertemuan ke 6 ini akan membahas tentang apa intents itu dan bagaimana dia digunakan. Intents adalah sebuah metode android untuk me-relay informasi tertentu dari satu aktivitas ke aktivitas lainnya. Secara lebih sederhana, Intents mengekspresikan kepada android untuk melakukan sesuatu.



Kita bisa menganggap intent sebagai sebuah pesan yang dilewatkan diantara banyak aktivitas. Misalnya, mempunyai aktivitas yang mengharuskan untuk membuka web browser dan menampilkan sebuah halaman di perangkat android. Aktivitas kita akan mengirimkan “keinginan (intent) untuk membuka halaman x di web browser” yang dikenal dengan Intent WEB-SEARCH\_ACTION, ke Android IntentResolver. IntentResolver mengurai melalui sebuah daftar Aktivitas dan memilih salah satu yang paling cocok dengan Intent kita; dalam hal ini adalah Web Browser Activity. Lalu Intent Resolver mengirimkan halaman kita ke web browser dan memulai Web Browser Activity.

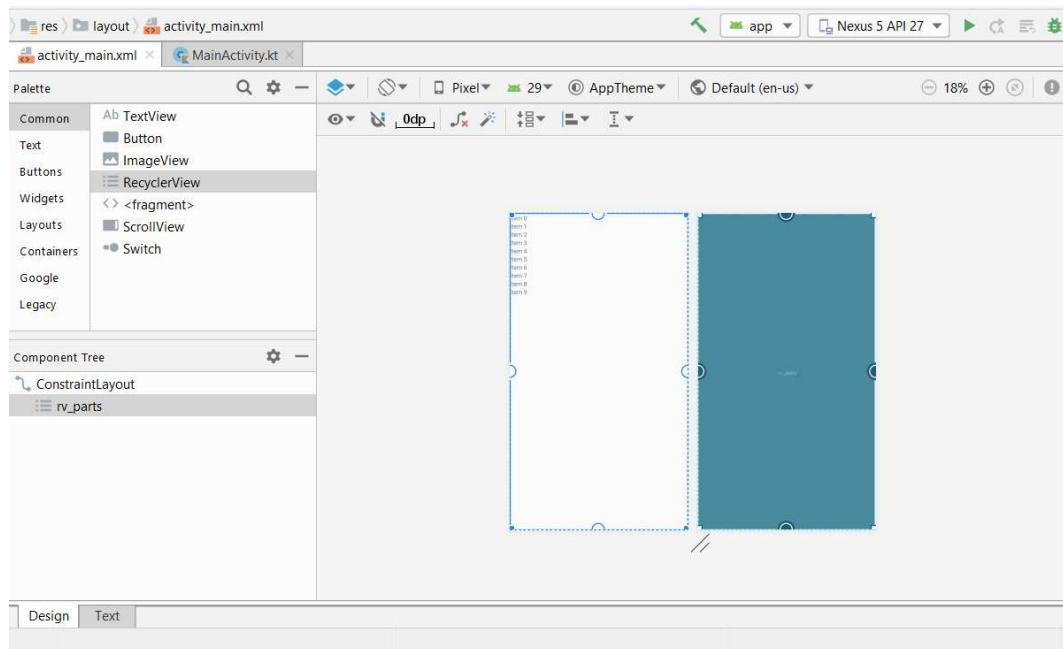
Intent dipecah ke dalam dua kategori utama:

- **Activity Action Intents:** Intent yang digunakan untuk memanggil Activity di luar aplikasi. Hanya satu Activity yang bisa ditangani oleh Intent. Misalnya saja untuk sebuah web browser, kita harus membuka Web Browser Activity untuk menampilkan halaman.
- **Broadcast Intents:** Intents yang dikirimkan untuk menangani lebih dari satu Activity. Contohnya, Broadcast intent yang akan menjadi sebuah pesan yang dikirimkan oleh Android mengenai tingkat baterei saat itu. Banyak aktivitas bisa memproses Intent ini dan melakukan reaksi yang sesuai – misalnya, membatalkan sebuah Activity bila tingkat baterei berada di bawah titik tertentu.

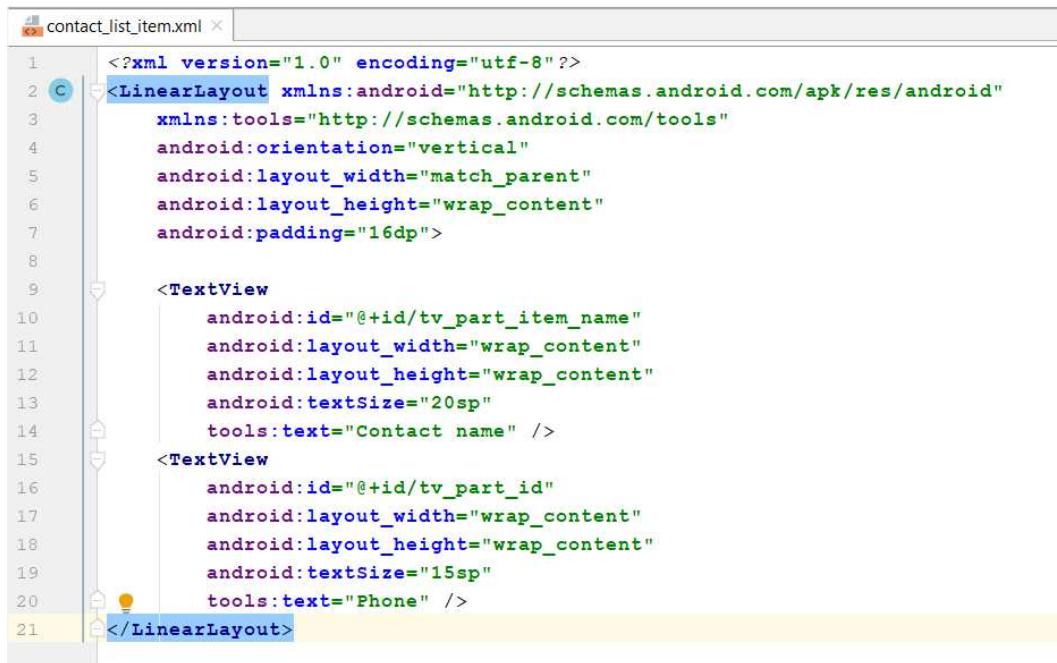


## PRAKTIK

1. Kita akan membuat sebuah project dengan menggunakan komponen view dan dengan menggunakan intent untuk menjalankan activity dari activity yang lain. Kita akan membuat sebuah daftar nomor telepon. Daftar akan ditampilkan dengan menggunakan komponen RecyclerView, isi daftar akan dibuat dalam sebuah View yang dipanggil oleh RecyclerView.
2. Buat Project baru. (dalam contoh ini diberi nama ProjectEnam)
3. Pada layout\_main.xml hapus komponen yang sudah ada (biasanya TextView).
4. Tambahkan komponen RecyclerView. Beri nama rv\_part.



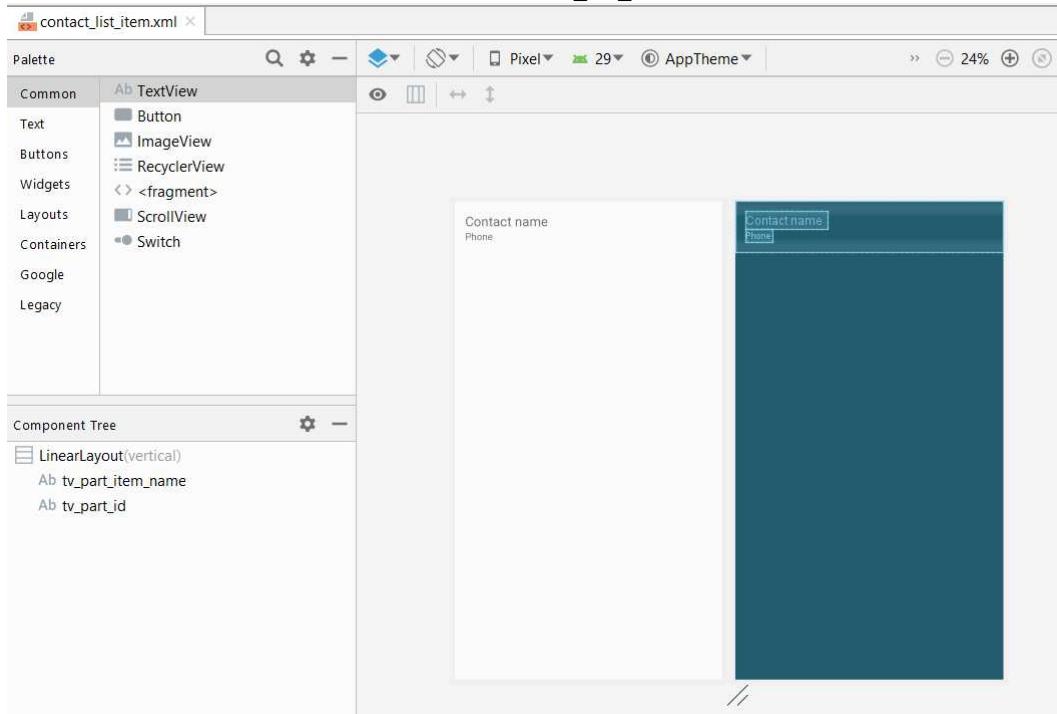
5. View ini nanti yang akan menjadi tampilan utama dari aplikasi ini.
6. Buat file view dengan nama contact\_list\_item.xml yang akan digunakan untuk mengisi daftar dalam RecyclerView.
7. Gunakan LinearLayout untuk layoutnya, dan tambahkan dua buah TextView dengan atribut seperti dibawah ini.



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:orientation="vertical"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:padding="16dp">

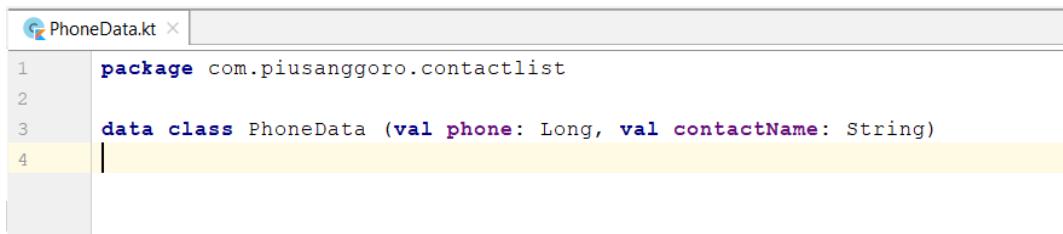
 <TextView
 android:id="@+id/tv_part_item_name"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:textSize="20sp"
 tools:text="Contact name" />
 <TextView
 android:id="@+id/tv_part_id"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:textSize="15sp"
 tools:text="Phone" />
</LinearLayout>
```

Tab Text dari contact\_list\_item.xml



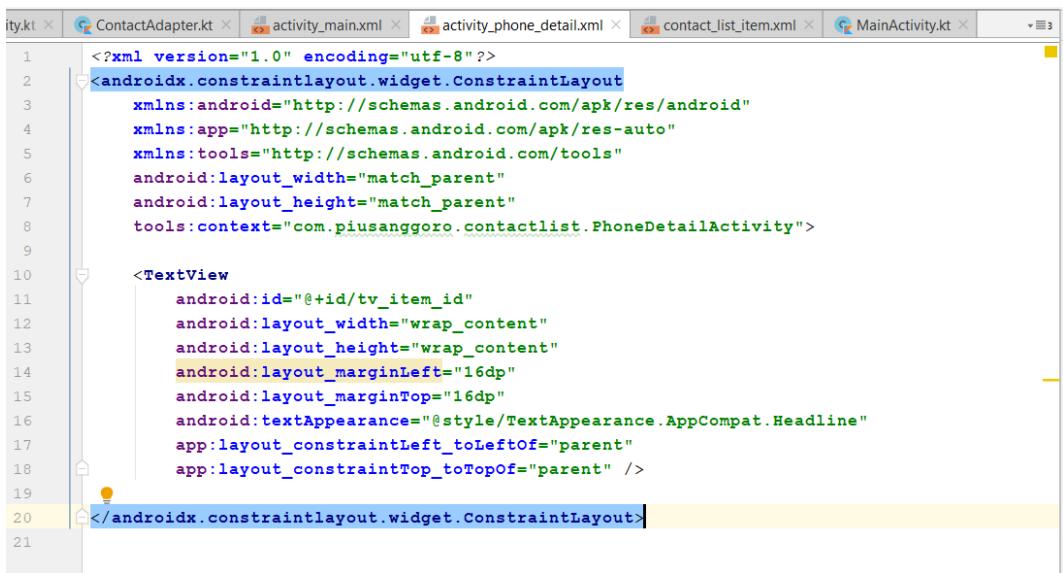
Tab Design dari contact\_list\_item.xml

8. Buat sebuah kelas data yang digunakan untuk menyimpan data telepon, dengan nama PhoneData.kt



```
1 package com.piussanggoro.contactlist
2
3 data class PhoneData (val phone: Long, val contactName: String)
4
```

1. Selanjutnya buat kelas activity yang berhubungan dengan view tersebut dengan nama PhoneDetailActivity.kt (layout: activity\_phone\_detail.xml)
2. Activity (dan View) ini akan dijalankan dengan menjalankan Intent dari kelas utama (MainActivity)



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3 xmlns:android="http://schemas.android.com/apk/res/android"
4 xmlns:app="http://schemas.android.com/apk/res-auto"
5 xmlns:tools="http://schemas.android.com/tools"
6 android:layout_width="match_parent"
7 android:layout_height="match_parent"
8 tools:context="com.piussanggoro.contactlist.PhoneDetailActivity">
9
10 <TextView
11 android:id="@+id/tv_item_id"
12 android:layout_width="wrap_content"
13 android:layout_height="wrap_content"
14 android:layout_marginLeft="16dp"
15 android:layout_marginTop="16dp"
16 android:textAppearance="@style/TextAppearance.AppCompat.Headline"
17 app:layout_constraintLeft_toLeftOf="parent"
18 app:layout_constraintTop_toTopOf="parent" />
19
20 </androidx.constraintlayout.widget.ConstraintLayout>
21
```

3. Baris 14-19 digunakan untuk menjalankan Intent dan menangkap serta menampilkan data yang dikirimkan melalui Intent. Data yang dikirimkan disimpan pada argumen dari method getStringExtra. (Dalam hal ini Intent.EXTRA\_TEXT)
4. Baris 18 digunakan untuk menampilkan data ke view pada komponen TextView (tv\_item\_id)

```
1 package com.piussanggoro.contactlist
2
3 import android.content.Intent
4 import androidx.appcompat.app.AppCompatActivity
5 import android.os.Bundle
6 import kotlinx.android.synthetic.main.activity_phone_detail.*
7
8 class PhoneDetailActivity : AppCompatActivity() {
9
10 override fun onCreate(savedInstanceState: Bundle?) {
11 super.onCreate(savedInstanceState)
12 setContentView(R.layout.activity_phone_detail)
13
14 var intentThatStartedThisActivity = getIntent()
15
16 if (intentThatStartedThisActivity.getStringExtra(Intent.EXTRA_TEXT) != null) {
17 var partId = intentThatStartedThisActivity.getStringExtra(Intent.EXTRA_TEXT)
18 tv_item_id.text = partId
19 }
20 }
21
22}
```

5. Buat sebuah Adapter dengan nama ContactAdapter.kt
6. Adapter ini nanti akan dipasang pada RecyclerView. Adapter diisi dengan daftar telepon yang disiapkan pada MainActivity.

```
1 import androidx.recyclerview.widget.RecyclerView
2 import android.view.LayoutInflater
3 import android.view.View
4 import android.view.ViewGroup
5 import kotlinx.android.synthetic.main.contact_list_item.view.*
6
7 class ContactAdapter (val phoneItemList: List<PhoneData>, val clickListener: (PhoneData) -> Unit) : RecyclerView.Adapter<RecyclerView.ViewHolder>() {
8
9 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): RecyclerView.ViewHolder {
10 val inflater = LayoutInflater.from(parent.context)
11 val view = inflater.inflate(R.layout.contact_list_item, parent, false)
12 return PartViewHolder(view)
13 }
14
15 override fun onBindViewHolder(holder: RecyclerView.ViewHolder, position: Int) {
16 (holder as PartViewHolder).bind(phoneItemList[position], clickListener)
17 }
18
19 override fun getItemCount() = phoneItemList.size
20
21 class PartViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
22 fun bind(phone: PhoneData, clickListener: (PhoneData) -> Unit) {
23 itemView.tv_part_item_name.text = phone.contactName
24 itemView.tv_part_id.text = phone.phone.toString()
25 itemView.setOnClickListener { clickListener(phone) }
26 }
27 }
28
29}
30
31}
```

7. Tambahkan pada MainActivity, sehingga menjadi sebagai berikut

```
1 package com.piussanggoro.contactlist
2
3 import android.content.Intent
4 import android.os.Bundle
5 import androidx.appcompat.app.AppCompatActivity
6 import androidx.recyclerview.widget.LinearLayoutManager
7 import kotlinx.android.synthetic.main.activity_main.*
8
9 class MainActivity : AppCompatActivity() {
10 override fun onCreate(savedInstanceState: Bundle?) {
11 super.onCreate(savedInstanceState)
12 setContentView(R.layout.activity_main)
13 val testData = createPhoneData()
14 rv_parts.layoutManager = LinearLayoutManager(context)
15 rv_parts.setHasFixedSize(true)
16 rv_parts.adapter = ContactAdapter(testData, { phoneItem : PhoneData -> phoneItemClicked(phoneItem) })
17 }
18 private fun phoneItemClicked(phoneItem : PhoneData) {
19 val showDetailActivityIntent = Intent(packageContext: this, PhoneDetailActivity::class.java)
20 showDetailActivityIntent.putExtra(Intent.EXTRA_TEXT, phoneItem.phone.toString())
21 startActivity(showDetailActivityIntent)
22 }
23 private fun createPhoneData() : List<PhoneData> {
24 val partList = ArrayList<PhoneData>()
25 partList.add(PhoneData(phone: 9864934, contactName: "Alpha"))
26 partList.add(PhoneData(phone: 1341933, contactName: "Bravo"))
27 partList.add(PhoneData(phone: 1401624, contactName: "Charlie"))
28 return partList
29 }
30 }
```

#### 8. Terakhir, daftarkan activity PhoneDetailActivity pada AndroidManifest

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3 xmlns:tools="http://schemas.android.com/tools"
4 package="com.piussanggoro.contactlist">
5
6 <application
7 android:allowBackup="true"
8 android:icon="@mipmap/ic_launcher"
9 android:label="ContactList"
10 android:roundIcon="@mipmap/ic_launcher_round"
11 android:supportsRtl="true"
12 android:theme="@style/AppTheme"
13 tools:ignore="AllowBackup,GoogleAppIndexingWarning">
14 <activity android:name="com.piussanggoro.contactlist.MainActivity">
15 <intent-filter>
16 <action android:name="android.intent.action.MAIN" />
17
18 <category android:name="android.intent.category.LAUNCHER" />
19 </intent-filter>
20 </activity>
21 <activity
22 android:name="com.piussanggoro.contactlist.PhoneDetailActivity"
23 android:parentActivityName="com.piussanggoro.contactlist.MainActivity">
24 </activity>
25 </application>
26
27 </manifest>
```

#### 9. Jalankan dan amati hasilnya.



jika diklik salah satu daftar, akan muncul

10. Tambahkan menampilkan nama pada tampilan kedua.



## LATIHAN

---

1. Modifikasilah aplikasi dengan menambahkan detil data pemilih nomor telepon.



## TUGAS

---

1. Buat aplikasi baru dengan mengembangkan project di atas



## REFERENSI

---

1. <https://kotlinlang.org/docs/reference/>
2. <https://developer.android.com/kotlin>
3. <https://developer.android.com/courses/kotlin-android-fundamentals/toc>
4. <https://codelabs.developers.google.com/android-kotlin-fundamentals/>
5. <https://developer.android.com/kotlin/learn>
6. <https://developer.android.com/kotlin/resources>



## **MODUL 7**

### **Menu dan Dialog**



#### **CAPAIAN PEMBELAJARAN**

---

1. Mahasiswa dapat menggunakan menu dan Dialog.



#### **KEBUTUHAN ALAT/BAHAN/SOFTWARE**

---

1. Android Studio 3.4.
2. Handphone Android versi 7.0 (Nougat)
3. Kabel data USB.
4. Driver ADB.



#### **DASAR TEORI**

---

#### **Komponen Navigasi.**

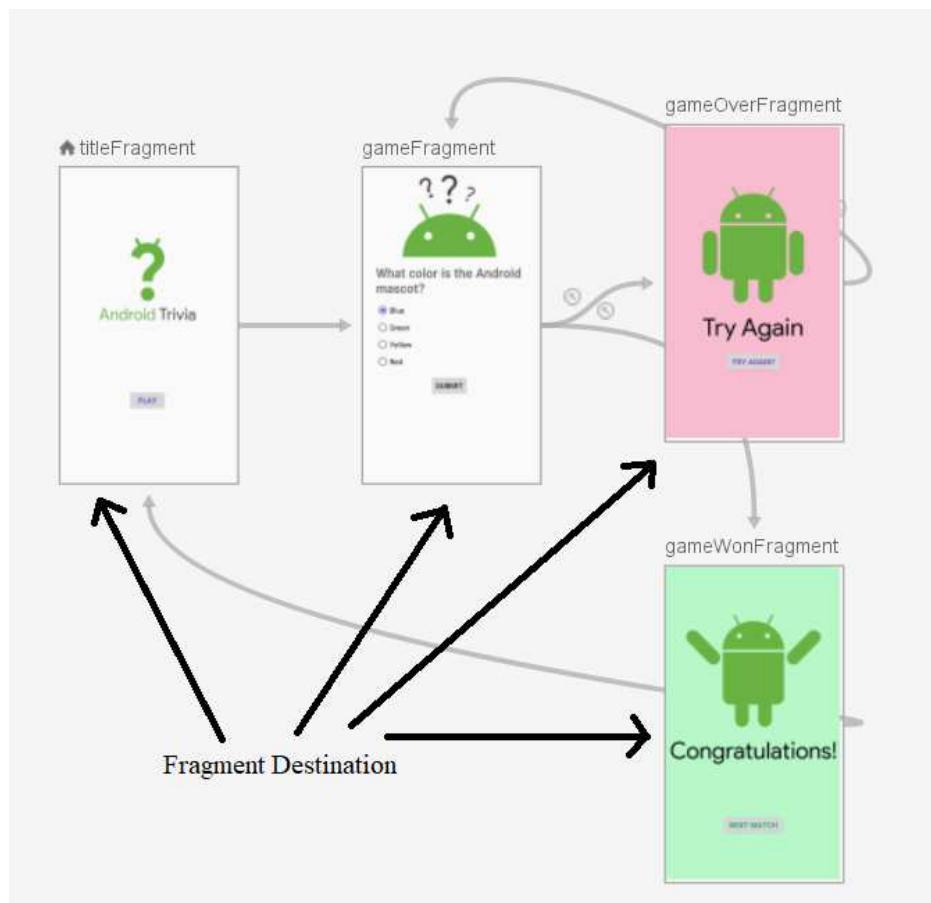
**Navigation Graph** (resource XML baru) adalah sumber daya yang berisi semua informasi terkait navigasi di satu lokasi terpusat, termasuk semua tempat di aplikasi, yang dikenal sebagai tujuan, dan kemungkinan jalur yang dapat dilalui pengguna melalui aplikasi.

**NavHostFragment** (Layout XML view) adalah widget khusus yang ditambahkan ke layout, yang menampilkan berbagai tujuan dari Navigation Graph. **NavController** (objek Kotlin / Java) adalah objek yang melacak posisi saat ini dalam Navigation Graph, yang mengatur pertukaran konten tujuan di NavHostFragment saat kita bergerak melalui Navigation Graph. Saat kita menavigasi, kita akan menggunakan objek NavController, memberi tahu ke mana kita ingin pergi atau jalur apa yang ingin kita ambil dalam Navigation Graph. NavController kemudian akan menunjukkan tujuan yang sesuai di NavHostFragment.

### Destinasi Navigation Graph.

Komponen Navigasi memperkenalkan konsep destination. Destination adalah tempat apa pun yang dapat kita navigasi di aplikasi, biasanya sebuah fragment atau activity.

Navigation Graph adalah jenis sumber daya baru yang mendefinisikan semua jalur yang mungkin seorang pengguna dapat ambil melalui aplikasi. Ini menunjukkan secara visual semua tujuan yang dapat dicapai dari tujuan tertentu. Android Studio menampilkan grafik di Editor Navigasinya. Contoh Navigation Graph adalah sebagai berikut.



### Mengeksplorasi Navigation Editor

1. Buka res/navigation/mobile\_navigation.xml
2. Klik **Design** untuk menuju mode Design.

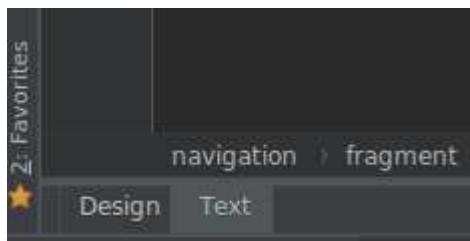
Navigation graph menunjukkan destination yang ada.



Anatomi dari file navigation XML.

Semua perubahan yang dibuat di Editor Navigasi grafis mengubah file XML yang menyertainya, mirip dengan cara Editor Layout memodifikasi layout XML.

Klik tab **Text** :



Akan muncul file xml berikut:

```
<navigation
 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:id="@+id/navigation"
 app:startDestination="@+id/titleFragment">

 <!-- ...tags for fragments and activities here -->

</navigation>
```

catatan:

- <navigation> adalah root node dari setiap navigation graph.
- <navigation> terdiri satu atau lebih destination, direpresentasikan dengan elemen <activity> atau <fragment> .
- app:startDestination adalah atribut yang menentukan destination yang dijalankan secara default ketika pengguna membuka app pertama kali.

Untuk sebuah fragment destination:

```
<fragment
 android:id="@+id/titleFragment"
 android:name=
 "com.example.android.navigation.TitleFragment"
 android:label="Intro"
 tools:layout="@layout/fragment_title">
 <action
 android:id="@+id/action_titleFragment_to_gameFragment"
 app:destination="@+id/gameFragment" />
</fragment>
```

Catatan:

android:id mendefinisikan ID untuk fragmen yang dapat digunakan untuk referensi destination di tempat lain dalam XML ini dan kode kita.

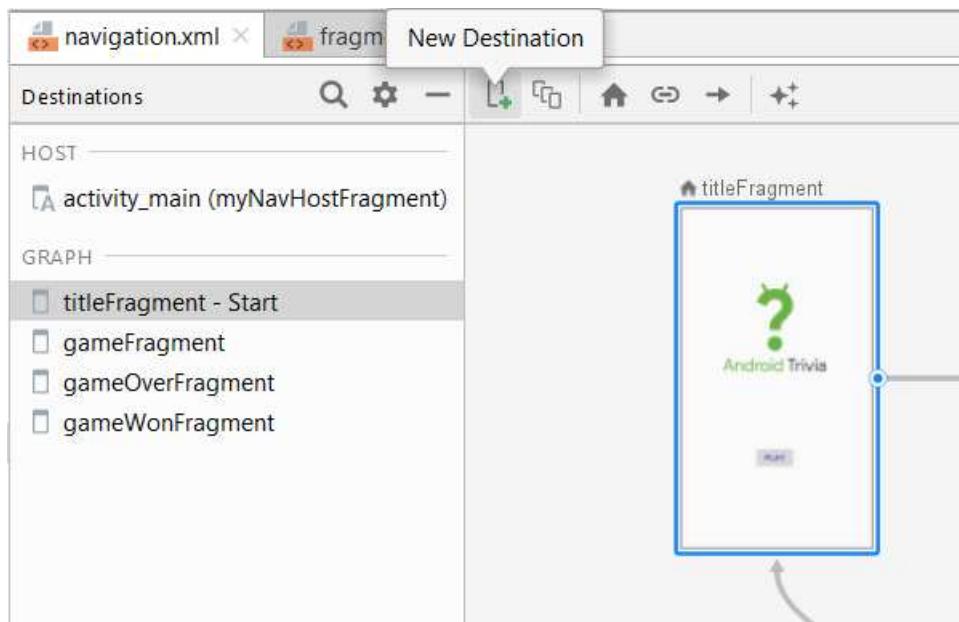
android:name mendeklarasikan nama kelas yang memenuhi syarat dari fragment untuk dipakai saat menavigasi ke destinasi tersebut.

tools:layout menentukan layout apa yang harus ditampilkan dalam editor grafis.

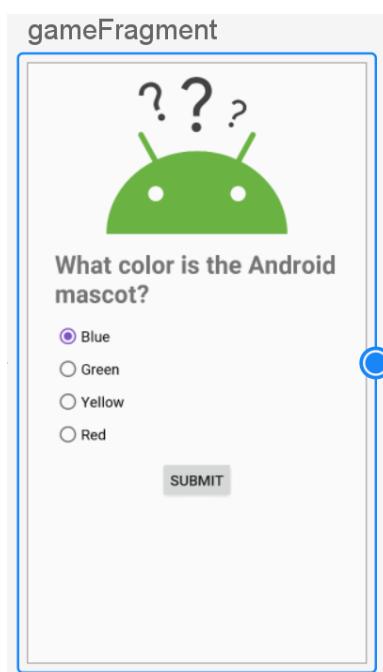
Menambahkan Destination ke Navigation Graph.

Aplikasi sampel dimulai dengan beberapa destination dalam grafik. Pada langkah ini, kita akan menambahkan destination baru! Kita harus menambahkan destination ke Navigation Graph sebelum dapat menavigasi ke sana.

1. Buka res/navigation/navigation.xml, dan klik tab **Design** .
2. Klik ikon **New Destination** , dan pilih "gameFragment"



Kita akan peroleh sebagai berikut.



Dengan file xml seperti berikut.

```
<fragment
 android:id="@+id/gameFragment"
 android:name="com.example.android.navigation.GameFragment"
 android:label="Game"
 tools:layout="@layout/fragment_game">
</fragment>
```

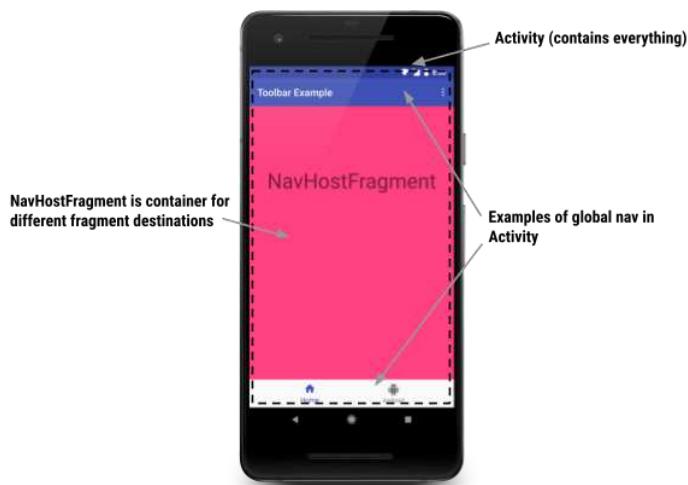
## Menggunakan Navigation Graph untuk menavigasi.

Saat ini kita memiliki navigation graph yang luar biasa ini, tetapi kita sebenarnya tidak menggunakannya untuk menavigasi.

### Activities dan Navigation

Komponen Navigasi mengikuti panduan yang dijabarkan dalam Prinsip Navigasi. Prinsip Navigasi merekomendasikan kita menggunakan aktivitas sebagai titik masuk untuk aplikasi kita. Aktifitas juga akan berisi navigasi global, seperti bottom nav. Sebagai perbandingan, fragment akan menjadi layout spesifik tujuan yang sebenarnya.

Agar semua ini berfungsi, kita perlu memodifikasi layout aktivitas kita untuk memuat widget khusus yang disebut NavHostFragment. NavHostFragment menukar destination fragment yang berbeda masuk dan keluar saat kita menavigasi navigation graph.



Layout sederhana yang mendukung navigasi mirip dengan gambar di atas terlihat seperti ini:

```
<LinearLayout
 ...>
 <androidx.appcompat.widget.Toolbar
 ...>
 <fragment
 android:id="@+id/myNavHostFragment"
 android:name=
 "androidx.navigation.fragment.NavHostFragment"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 app:defaultNavHost="true"
 app:navGraph="@navigation/navigation" />
 <com.google.android.material.bottomnavigation.BottomNavigationView
 ...>
```

```
</LinearLayout>
```

Catatan:

- Ini adalah layout for an activity, berisi navigation global, termasuk sebuah bottom nav dan a toolbar
- `android:name="androidx.navigation.fragment.NavHostFragment"` and `app:defaultNavHost="true"` menghubungkan back button sistem ke `NavHostFragment`
- `app:navGraph="@navigation/navigation"` mengaitkan `NavHostFragment` dengan navigation graph. Navigation graph ini menentukan semua destination pengguna dapat menavigasi ke, di fragment `NavHostFragment` ini.

### NavController

Akhirnya, ketika pengguna melakukan sesuatu seperti mengklik tombol, kita perlu memicu perintah navigasi. Kelas khusus yang disebut NavController adalah apa yang memicu swap fragmen di NavHostFragment.

```
// Command untuk navigasi ke gameFragment_to_gameWonFragment
findNavController()
 .navigate(R.id.action_gameFragment_to_gameWonFragment)
```

Perhatikan bahwa kita memberikan destination ID atau action untuk menavigasi. Ini adalah ID yang ditentukan dalam navigation graph XML. Ini adalah contoh memberikan destination ID. NavController sangat powerfull karena ketika kita memanggil metode seperti `navigate()` atau `popBackStack()`, NavController menerjemahkan perintah-perintah ini ke dalam operasi framework yang sesuai berdasarkan jenis destination yang kita navigasikan ke atau dari. Misalnya, ketika kita memanggil `navigate()` dengan sebuah activity destination, NavController memanggil `startActivity()` atas nama kita. Ada beberapa cara untuk mendapatkan objek NavController yang terkait dengan NavHostFragment. Di Kotlin, disarankan untuk menggunakan salah satu fungsi ekstensi berikut, tergantung pada apakah akan memanggil perintah navigasi dari dalam sebuah fragment, activity atau view:

- [Fragment.findNavController\(\)](#)
- [View.findNavController\(\)](#)
- [Activity.findNavController\(viewId: Int\)](#)

NavController dikaitkan dengan NavHostFragment. Jadi, metode apa pun yang digunakan, kita harus yakin bahwa fragment, view, atau view ID adalah NavHostFragment itu sendiri, atau memiliki NavHostFragment sebagai induk. Kalau tidak, kita akan mendapatkan IllegalStateException.

### Navigasi ke sebuah Destination dengan NavController

Kita akan bernaligasi menggunakan NavController dengan menghubungkan tombol Navigate Ke Destination untuk menavigasi ke destination `action_gameFragment_to_gameWonFragment` (yang merupakan tujuan adalah GameWonFragment):

1. Buka `TitleFragment.kt`
2. Sambungkan `navigate_destination_button` dalam `onViewCreated()`

#### `TitleFragment.kt`

```
playButton.setOnClickListener { view : View ->
 view.findNavController().
 navigate(R.id.action_titleFragment_to_gameFragment)
}
```

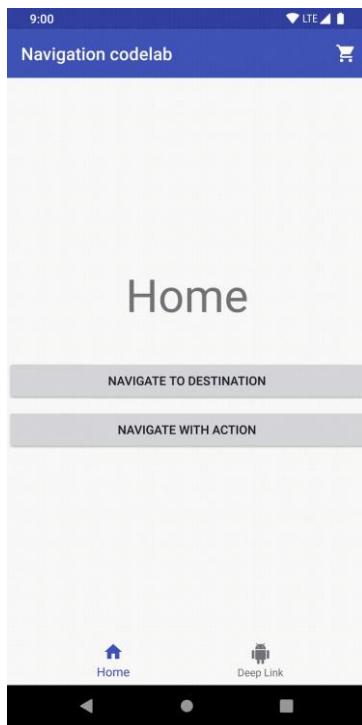
2. Jalankan app dan **Navigate To Destination** button. Button menavigasi ke **game\_fragment** destination.

Kita juga dapat menggunakan metode `Navigation.createNavigateOnClickListener(@IdRes destId: int, bundle: Bundle)`. Metode ini akan membangun `OnClickListener` untuk menavigasi ke tujuan yang diberikan, dengan sekumpulan argumen untuk diteruskan ke tujuan. Kodenya sebagai berikut.

```
val button = view.findViewById<Button>(R.id.navigate_destination_button)
button?.setOnClickListener(
 Navigation.createNavigateOnClickListener(R.id.game_fragment, null)
)
```

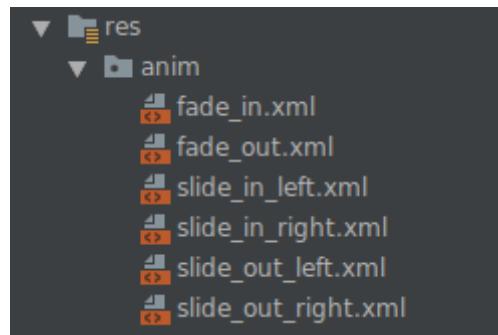
### Mengubah transisi Navigation.

Setiap panggilan `navigate()` memiliki transisi standar yang tidak terlalu menarik yang terkait dengannya, seperti terlihat di bawah:



Transisi default, serta atribut lain yang terkait dengan panggilan, dapat diganti dengan menyertakan satu set NavOptions. NavOptions menggunakan pola Builder yang memungkinkan kita mengganti dan menetapkan hanya opsi yang dibutuhkan. Ada juga ktx DSL untuk NavOptions, yang akan digunakan.

Untuk transisi animasi, kita dapat menentukan resource animation XML di folder resource anim dan kemudian menggunakan animation tersebut untuk transisi. Beberapa contoh termasuk dalam kode app:

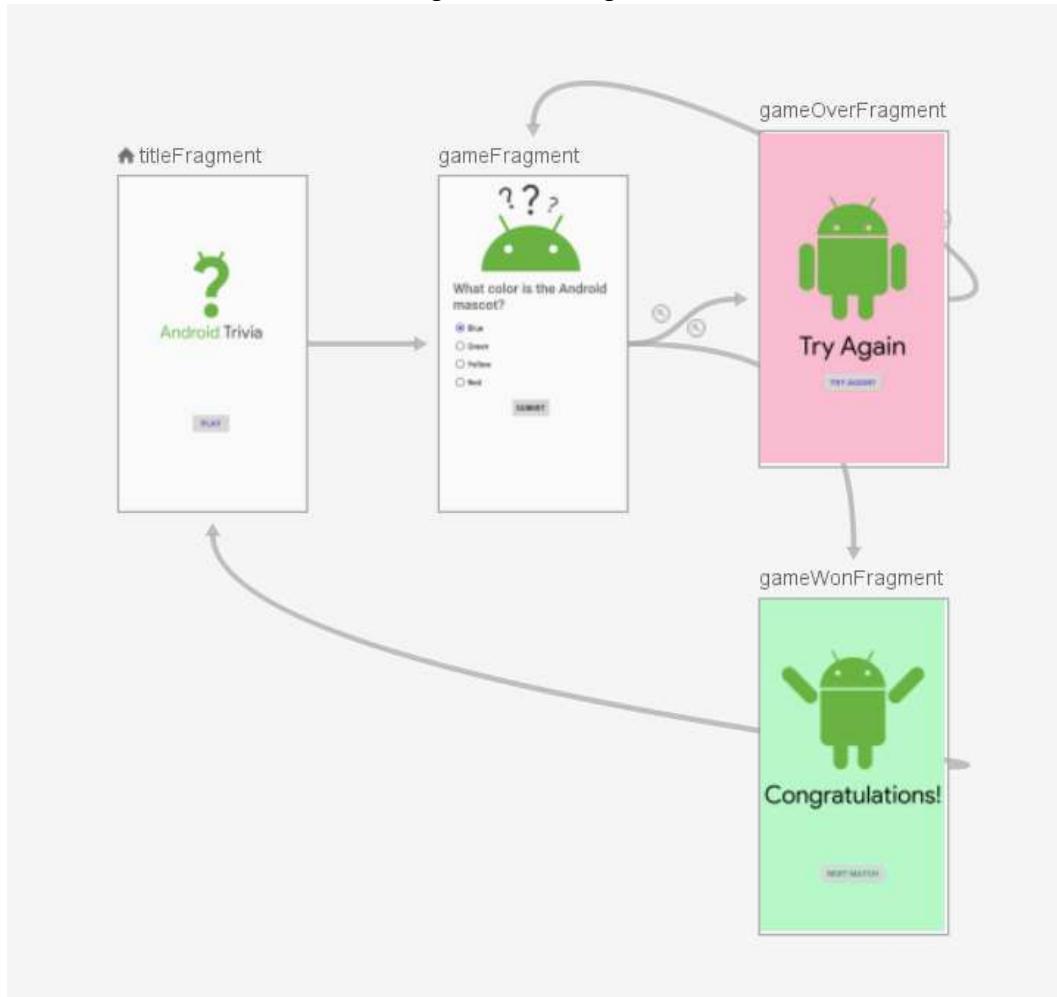




## PRAKTIK

---

1. Kita akan membuat menu dengan desain sebagai berikut.



2. Buat project baru.
3. Tambahkan data binding enabled true pada build.gradle

```

10 applicationId "akakom.nomhs.kotlin.belajarnavigasi2"
11 minSdkVersion 29
12 targetSdkVersion 29
13 versionCode 1
14 versionName "1.0"
15 testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
16
17 buildTypes {
18 release {
19 minifyEnabled false
20 proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
21 }
22 }
23 dataBinding enabled = true
24
25 dependencies {
26 implementation fileTree(dir: 'libs', include: ['*.jar'])
27 android { dataBinding true }
28 }

```

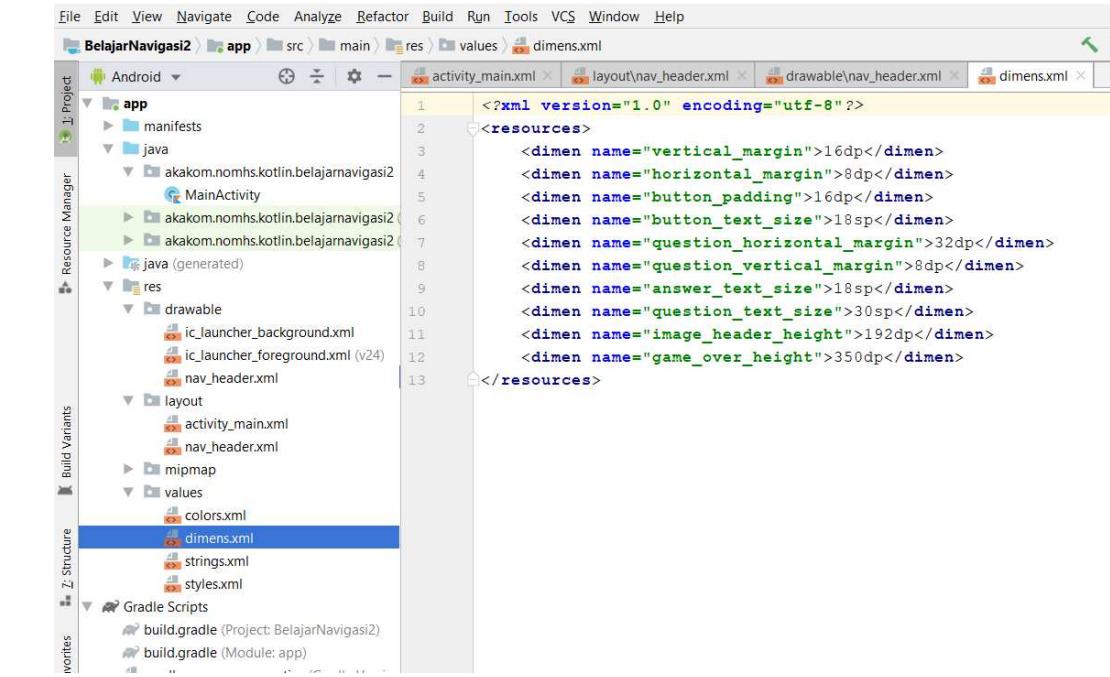
4. Tambahkan sebuah file xml pada resource layout, dengan nama **navheader.xml** dan kemudian tambahkan pada kode sehingga menjadi sebagai berikut

```

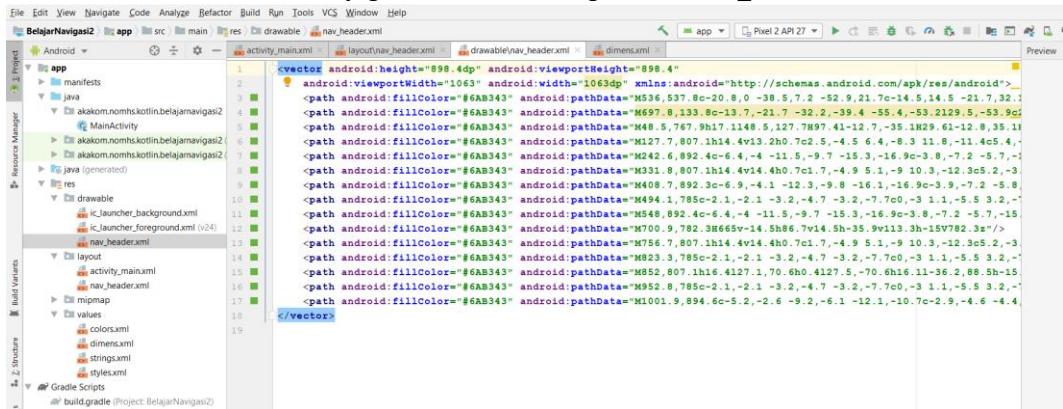
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3 xmlns:android="http://schemas.android.com/apk/res/android"
4 xmlns:app="http://schemas.android.com/apk/res-auto"
5 android:id="@+id/navHeader"
6 android:layout_width="match_parent"
7 android:layout_height="192dp"
8 android:background="?attr/colorPrimaryDark"
9 android:padding="16dp"
10 android:theme="@style/ThemeOverlay.AppCompat.Dark">
11
12 <ImageView
13 android:id="@+id/navHeaderImage"
14 android:layout_width="0dp"
15 android:layout_height="0dp"
16 android:layout_marginStart="@dimen/horizontal_margin"
17 android:layout_marginTop="8dp"
18 android:layout_marginEnd="@dimen/horizontal_margin"
19 android:layout_marginBottom="24dp"
20 android:scaleType="fitCenter"
21 app:layout_constraintBottom_toBottomOf="parent"
22 app:layout_constraintEnd_toEndOf="parent"
23 app:layout_constraintStart_toStartOf="parent"
24 app:layout_constraintTop_toTopOf="parent"
25 app:srcCompat="@drawable/nav_header" />
26
27 </androidx.constraintlayout.widget.ConstraintLayout>

```

5. Perhatikan atribut yang masih berwarna merah.
6. Pertama anda akan tambahkan untuk resource dimen. Buat folder dimens pada resource: res/value. Kemudian tuliskan sebagai berikut



## 7. Kemudian tambahkan juga file drawable dengan nama nav\_header.xml.



Lengkapnya sebagai berikut

```

<vector android:height="898.4dp"
 android:viewportHeight="898.4"
 android:viewportWidth="1063" android:width="1063dp"
 xmlns:android="http://schemas.android.com/apk/res/android">
 <path android:fillColor="#6AB343" android:pathData="M536,537.8c-20.8,0 -38.5,7.2 -52.9,21.7c-14.5,14.5 -21.7,32.1
 <path android:fillColor="#6AB343" android:pathData="M697.8,133.8c-13.7,-21.7 -32.2,-39.4 -55.4,-53.2c3.4,14.5,14.8 32.1,22.1 52.9,22.1c20.8,0 38.6,-7.4 53.4,-22.1
 <path android:fillColor="#6AB343" android:pathData="M408.7,892.3c-6.9,-4.1 -12.3,-9.8 -16.1,-16.9c-3.9,-7.2 5.8
 <path android:fillColor="#6AB343" android:pathData="M494.1,785c-2.1,-2.1 -3.2,-4.7 -3.2,-7.7c0,-1.1,-5.5 3.2,-15.3,-16.9c-3.8,-7.2 5.7,-15.3
 <path android:fillColor="#6AB343" android:pathData="M700.9,894.6c-5.2,-2.6 -9.2,-6.1 -12.1,-10.7c-2.9,-4.6 -4.6
 <path android:fillColor="#6AB343" android:pathData="M756.7,807.1h14.4v14.4h0.7c1.7,-4.9 5.1,-9 10.3,-12.3c5.2,-3
 <path android:fillColor="#6AB343" android:pathData="M823.3,785c-2.1,-2.1 -3.2,-4.7 -3.2,-7.7c0,-3 1.1,-5.5 3.2,-15.3
 <path android:fillColor="#6AB343" android:pathData="M852.807.1h16.4l27.1,70.6h4127.5,-70.6h16.1l-36.2,88.5h-15.3
 <path android:fillColor="#6AB343" android:pathData="M952.8,785c-2.1,-2.1 -3.2,-4.7 -3.2,-7.7c0,-3 1.1,-5.5 3.2,-15.3
 <path android:fillColor="#6AB343" android:pathData="M1001.9,894.6c-5.2,-2.6 -9.2,-6.1 -12.1,-10.7c-2.9,-4.6 -4.6
 </vector>

```

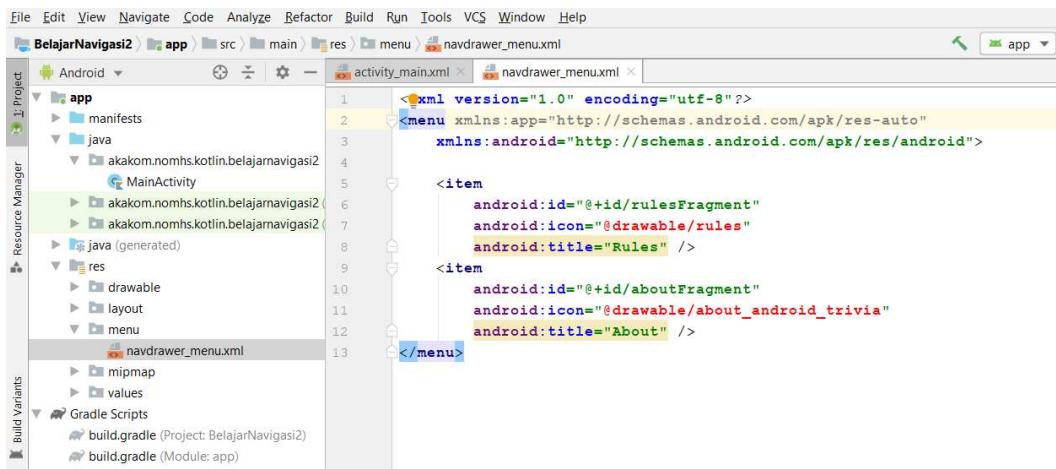
```

1.3 -9.5,-0.8 -13.7,1.5c-4.3,2.3 -7.4,6.2 -8.7,10.8c-1.3,4.6 -0.8,9.5
1.6,13.7130,54.7c-5.3,3.5 -10.5,7.4 -15.4,11.5c-31.8,26.9 -53.2,58.9
-64.2,95.9199.8,41.7c5.8,-20.2 16.2,-37.2 31.2,-50.8c15,-13.6 34.1,-
20.4 57.3,-20.4c22,0 39.6,6.2 52.9,18.7c13.3,12.5 20,27.6
20,45.6c0,15.1 -3.9,28.4 -11.7,39.9c-7.8,11.6 -20.4,25.2 -37.8,40.8c-
30.1,26 -50.9,49.2 -62.5,69.4c-11.6,20.3 -17.4,44.3 -
17.4,72v28.6h113.7v-13.9c0,-20.2 3.6,-37 10.8,-50.3c7.2,-13.3 20.7,-
29.2 40.4,-47.7c25.5,-24.3 44.5,-47 57.3,-68.1c12.7,-21.1 19.1,-46.7
19.1,-76.8C721.7,186.9 713.7,159 697.8,133.8z" />
<path android:fillColor="#6AB343"
 android:pathData="M48.5,767.9h17.1148.5,127.7H97.41-12.7,-35.1H29.61-
 12.8,35.1H0L48.5,767.9zM79.4,846.21-16.9,-461-5,-13.7h-0.71-
 5.2,13.71-16.8,46H79.4z"/>
<path android:fillColor="#6AB343"
 android:pathData="M127.7,807.1h14.4v13.2h0.7c2.5,-4.5 6.4,-8.3 11.8,-
 11.4c5.4,-3.1 11.1,-4.6 17.1,-4.6c10.8,0 19.1,3.2 24.7,9.5c5.6,6.3
 8.5,14.9 8.5,25.9v56h-15v-53.7c0,-15.9 -7.4,-23.9 -22.3,-23.9c-4.8,0
 -9.1.4 -12.8,4.1c-3.8,2.7 -6.8,6.3 -8.8,10.7c-2.1,4.4 -3.1,9 -
 3.1,13.9v48.9h-15.2V807.1z"/>
<path android:fillColor="#6AB343" android:pathData="M242.6,892.4c-6.4,-
 4 -11.5,-9.7 -15.3,-16.9c-3.8,-7.2 -5.7,-15.3 -5.7,-24.2c0,-8.9 1.9,-
 17 5.7,-24.2c3.8,-7.2 8.9,-12.8 15.3,-16.9c6.4,-4 13.4,-6.1 21.1,-
 6.1c6.9,0 13.1,1.5 18.6,4.5c5.5,3 9.6,6.8 12.3,11.3h0.71-0.7,-12.1v-
 40.1h15v127.7h-14.3v-13.2h-0.7c-2.7,4.6 -6.8,8.5 -12.3,11.5c-5.5,3 -
 11.7,4.5 -18.6,4.5C256,898.4 249,896.4 242.6,892.4zM280.5,880.7c4.5,-
 2.7 8,-6.6 10.7,-11.7c2.7,-5.1 4,-11 4,-17.7c0,-6.8 -1.3,-12.7 -4,-
 17.8c-2.7,-5.1 -6.2,-8.9 -10.7,-11.6s-9.3,-4 -14.5,-4c-5.2,0 -
 10.1,1.4 -14.5,4.1c-4.5,2.7 -8,6.6 -10.7,11.7c-2.7,5.1 -4,10.9 -
 4,17.6c0,6.7 1.3,12.5 4,17.6c2.7,5.1 6.2,9 10.7,11.7c4.5,2.7 9.3,4.1
 14.5,4.1C271.2,884.7 276.1,883.4 280.5,880.7z"/>
<path android:fillColor="#6AB343"
 android:pathData="M331.8,807.1h14.4v14.4h0.7c1.7,-4.9 5.1,-9 10.3,-
 12.3c5.2,-3.3 10.3,-5 15.4,-5c3.7,0 7.1,0.6 10.3,1.8v16.2c-3.2,-1.8 -
 7.4,-2.7 -12.5,-2.7c-4.2,0 -8.1,1.2 -11.7,3.7c-3.6,2.4 -6.5,5.7 -
 8.7,9.7s-3.2,8.5 -3.2,13.2v49.4h-15.2V807.1z"/>
<path android:fillColor="#6AB343" android:pathData="M408.7,892.3c-6.9,-
 4.1 -12.3,-9.8 -16.1,-16.9c-3.9,-7.2 -5.8,-15.2 -5.8,-24c0,-8.8 1.9,-
 16.8 5.8,-24c3.9,-7.2 9.2,-12.8 16.1,-16.9c6.9,-4.1 14.7,-6.2 23.4,-
 6.2c8.7,0 16.5,2.1 23.5,6.2c7,4.1 12.4,9.8 16.2,16.9c3.9,7.2 5.8,15.2
 5.8,24c0,8.8 -1.9,16.8 -5.8,24c-3.9,7.2 -9.3,12.8 -16.2,16.9c-7,4.1 -
 14.8,6.2 -23.5,6.2C423.4,898.4 415.6,896.4
 408.7,892.3zM447,880.6c4.6,-2.7 8.4,-6.6 11.1,-11.7c2.8,-5.1 4.2,-
 10.9 4.2,-17.6c0,-6.7 -1.4,-12.5 -4.2,-17.6c-2.8,-5.1 -6.5,-8.9 -
 11.1,-11.7c-4.6,-2.7 -9.6,-4.1 -15,-4.1c-5.2,0 -10.2,1.4 -14.8,4.1c-
 4.6,2.7 -8.4,6.6 -11.1,11.7c-2.8,5.1 -4.2,10.9 -4.2,17.6c0,6.7
 1.4,12.5 4.2,17.6c2.8,5.1 6.5,9 11.1,11.7c4.6,2.7 9.6,4.1
 14.8,4.1C437.4,884.7 442.4,883.3 447,880.6z"/>
<path android:fillColor="#6AB343" android:pathData="M494.1,785c-2.1,-
 2.1 -3.2,-4.7 -3.2,-7.7c0,-3 1.1,-5.5 3.2,-7.7c2.1,-2.1 4.7,-3.2
 7.7,-3.2c3,0 5.5,1.1 7.7,3.2s3.2,4.7 3.2,7.7c0,3 -1,5.5 -3.1,7.7c-
 2.1,2.1 -4.7,3.2 -7.8,3.2C498.8,788.2 496.3,787.1
 494.1,785zM494.3,807.1h15v88.5h-15V807.1z"/>
<path android:fillColor="#6AB343" android:pathData="M548,892.4c-6.4,-4
 -11.5,-9.7 -15.3,-16.9c-3.8,-7.2 -5.7,-15.3 -5.7,-24.2c0,-8.9 1.9,-17
 5.7,-24.2c3.8,-7.2 8.9,-12.8 15.3,-16.9c6.4,-4 13.4,-6.1 21.1,-
 6.1c6.9,0 13.1,1.5 18.6,4.5c5.5,3 9.6,6.8 12.3,11.3h0.71-0.7,-12.1v-
 40.1h15v127.7h-14.3v-13.2h-0.7c-2.7,4.6 -6.8,8.5 -12.3,11.5c-5.5,3 -
 11.7,4.5 -18.6,4.5C256,898.4 249,896.4 242.6,892.4zM280.5,880.7c4.5,-
 2.7 8,-6.6 10.7,-11.7c2.7,-5.1 4,-11 4,-17.7c0,-6.8 -1.3,-12.7 -4,-
 17.8c-2.7,-5.1 -6.2,-8.9 -10.7,-11.6s-9.3,-4 -14.5,-4c-5.2,0 -
 10.1,1.4 -14.5,4.1c-4.5,2.7 -8,6.6 -10.7,11.7c-2.7,5.1 -4,10.9 -
 4,17.6c0,6.7 1.3,12.5 4,17.6c2.7,5.1 6.2,9 10.7,11.7c4.5,2.7 9.3,4.1
 14.5,4.1C271.2,884.7 276.1,883.4 280.5,880.7z"/>
```

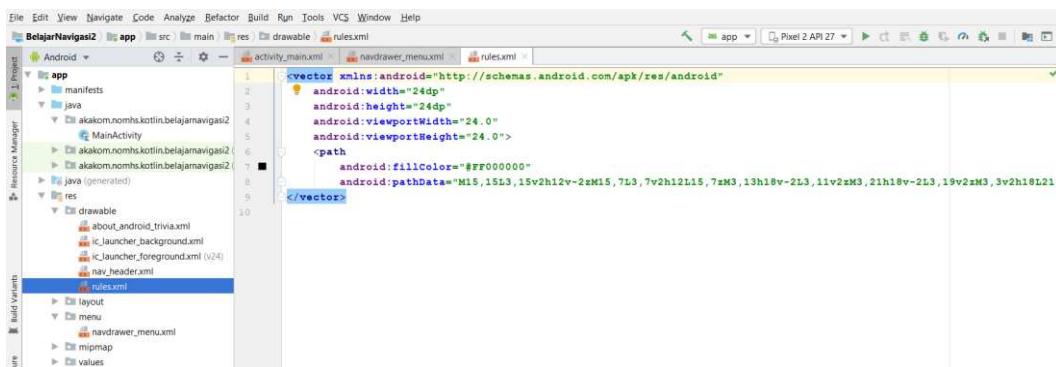
```

5.7,-24.2c3.8,-7.2 8.9,-12.8 15.3,-16.9c6.4,-4 13.4,-6.1 21.1,-
6.1c6.9,0 13.1,1.5 18.6,4.5c5.5,3 9.6,6.8 12.3,11.3h0.71-0.7,-12.1v-
40.1h15v127.7h-14.3v-13.2h-0.7c-2.7,4.6 -6.8,8.5 -12.3,11.5c-5.5,3 -
11.7,4.5 -18.6,4.5C561.5,898.4 554.4,896.4
548,892.4zM585.9,880.7c4.5,-2.7 8,-6.6 10.7,-11.7c2.7,-5.1 4,-11 4,-
17.7c0,-6.8 -1.3,-12.7 -4,-17.8c-2.7,-5.1 -6.2,-8.9 -10.7,-11.6c-
4.5,-2.7 -9.3,-4 -14.5,-4c-5.2,0 -10.1,1.4 -14.5,4.1c-4.5,2.7 -8,6.6
-10.7,11.7c-2.7,5.1 -4,10.9 -4,17.6c0,6.7 1.3,12.5 4,17.6c2.7,5.1
6.2,9 10.7,11.7c4.5,2.7 9.3,4.1 14.5,4.1C576.6,884.7 581.5,883.4
585.9,880.7z"/>
<path android:fillColor="#6AB343" android:pathData="M700.9,782.3H665v-
14.5h86.7v14.5h-35.9v113.3h-15V782.3z"/>
<path android:fillColor="#6AB343"
 android:pathData="M756.7,807.1h14.4v14.4h0.7c1.7,-4.9 5.1,-9 10.3,-
12.3c5.2,-3.3 10.3,-5 15.4,-5c3.7,0 7.1,0.6 10.3,1.8v16.2c-3.2,-1.8 -
7.4,-2.7 -12.5,-2.7c-4.2,0 -8.1,1.2 -11.7,3.7c-3.6,2.4 -6.5,5.7 -
8.7,9.7s-3.2,8.5 -3.2,13.2v49.4h-15.2V807.1z"/>
<path android:fillColor="#6AB343" android:pathData="M823.3,785c-2.1,-
2.1 -3.2,-4.7 -3.2,-7.7c0,-3 1.1,-5.5 3.2,-7.7c2.1,-2.1 4.7,-3.2
7.7,-3.2c3,0 5.5,1.1 7.7,3.2s3.2,4.7 3.2,7.7c0,3 -1,5.5 -3.1,7.7c-
2.1,2.1 -4.7,3.2 -7.8,3.2C828,788.2 825.4,787.1
823.3,785zM823.5,807.1h15v88.5h-15V807.1z"/>
<path android:fillColor="#6AB343"
 android:pathData="M852,807.1h16.4l27.1,70.6h0.4l27.5,-70.6h16.11-
36.2,88.5h-15.3L852,807.1z"/>
<path android:fillColor="#6AB343" android:pathData="M952.8,785c-2.1,-
2.1 -3.2,-4.7 -3.2,-7.7c0,-3 1.1,-5.5 3.2,-7.7c2.1,-2.1 4.7,-3.2
7.7,-3.2c3,0 5.5,1.1 7.7,3.2s3.2,4.7 3.2,7.7c0,3 -1,5.5 -3.1,7.7c-
2.1,2.1 -4.7,3.2 -7.8,3.2C957.5,788.2 954.9,787.1
952.8,785zM953,807.1h15v88.5h-15V807.1z"/>
<path android:fillColor="#6AB343" android:pathData="M1001.9,894.6c-
5.2,-2.6 -9.2,-6.1 -12.1,-10.7c-2.9,-4.6 -4.4,-9.7 -4.4,-15.4c0,-9.4
3.5,-16.7 10.6,-21.9c7.1,-5.2 16.1,-7.8 27.2,-7.8c5.5,0 10.4,0.6
14.8,1.8c4.4,1.2 7.9,2.5 10.5,3.9v-4.6c0,-4.3 -1.1,-8.1 -3.3,-11.5c-
2.2,-3.4 -5.1,-6 -8.8,-7.8c-3.7,-1.8 -7.7,-2.8 -12,-2.8c-10.1,0 -
17.8,4.1 -23.2,12.31-12.7,-8.4c3.7,-5.5 8.6,-9.7 14.6,-12.8c6.1,-3
12.9,-4.5 20.5,-4.5c12.1,0 21.7,3.3 28.7,9.8c7,6.5 10.5,15.3
10.5,26.4v55.1h-14.4v-13h-0.7c-2.9,4.5 -6.7,8.3 -11.6,11.3c-4.9,3 -
10.5,4.5 -16.9,4.5C1012.9,898.4 1007.1,897.2
1001.9,894.6zM1034.1,881.3c4.3,-2.5 7.8,-5.9 10.4,-10.2c2.7,-4.3 4,-
8.9 4,-13.9c-6.7,-4.3 -14.3,-6.4 -23,-6.4c-7.5,0 -13.5,1.7 -
17.9,5.1c-4.5,3.4 -6.7,7.7 -6.7,12.9c0,4.8 2,8.7 5.9,11.7c3.9,3
8.5,4.5 13.7,4.5C1025.3,885.1 1029.9,883.8 1034.1,881.3z"/>
</vector>
```

8. Buat sebuah file **navdrawer\_menu.xml** pada folder resource **menu** (jika menu belum ada, dibuat terlebih dulu). Anda masih membutuhkan file **rules.xml** dan **about\_android\_trivia.xml** pada resource **drawable**.



## 9. Berikut kedua file tersebut



### rules.xml

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
 android:width="24dp"
 android:height="24dp"
 android:viewportWidth="24.0"
 android:viewportHeight="24.0">
 <path
 android:fillColor="#FF000000"
 android:pathData="M15,15L3,15v2h12v-2zM15,7L3,7v2h12L15,7zM3,13h18v-2L3,11v2zM3,21h18v-2L3,19v2zM3,3v2h18L21,
 </vector>
```

### about\_android\_trivia.xml

```
<vector android:height="535dp" android:viewportHeight="535"
 android:viewportWidth="507" android:width="507dp"
 xmlns:android="http://schemas.android.com/apk/res/android">
<path android:fillColor="#6AB343" android:pathData="M255.6,256.5c-9.9,0
-18.4,3.5 -25.3,10.3c-6.9,6.9 -10.3,15.3 -10.3,25.3s3.4,18.4
10.3,25.5c6.9,7 15.3,10.6 25.3,10.6c9.9,0 18.4,-3.5 25.5,-10.6c7,-7
 </path>
```

```

10.6,-15.5 10.6,-25.5s-3.5,-18.4 -10.6,-25.3C274.1,260 265.6,256.5
255.6,256.5z"/>
<path android:fillColor="#6AB343" android:pathData="M332.8,63.8c-6.6,-
10.4 -15.4,-18.8 -26.4,-25.4l14.1,-25.7c1.1,-2 1.4,-4.3 0.7,-6.5c-
0.6,-2.2 -2.1,-4.1 -4.1,-5.2l317,1c-4.2,-2.2 -9.4,-0.7 -11.6,3.5l-
14.6,26.6c-11.1,-4 -23.2,-6 -36.4,-6c-12.8,0 -24.5,2 -
34.9,5.9l205,4.5c-1.1,-2 -2.9,-3.5 -5.1,-4.1c-2.2,-0.6 -4.5,-0.4 -
6.5,0.7c-2.1,1.1 -3.5,2.9 -4.2,5.2c-0.6,2.2 -0.4,4.5
0.7,6.5l14.3,26.1c-2.5,1.7 -5,3.5 -7.3,5.5c-15.2,12.8 -25.4,28.1 -
30.6,45.7l47.6,19.9c2.8,-9.7 7.7,-17.7 14.9,-24.2C235.9,79.3 245,76
256,76c10.5,0 18.9,3 25.3,8.9c6.3,5.9 9.5,13.2 9.5,21.7c0,7.2 -
1.9,13.5 -5.6,19c-3.7,5.5 -9.7,12 -18,19.5c-14.4,12.4 -24.3,23.5 -
29.8,33.1c-5.5,9.7 -8.3,21.1 -8.3,34.4v13.7h54.2v-6.6c0,-9.7 1.7,-
17.7 5.2,-24c3.4,-6.3 9.9,-13.9 19.2,-22.8c12.1,-11.6 21.2,-22.4
27.3,-32.5c6.1,-10.1 9.1,-22.3 9.1,-36.6C344.2,89.2 340.4,75.8
332.8,63.8z"/>
 <path android:fillColor="#6AB343"
android:pathData="M23.1,366.2h8.2l23.1,60.9h-81-6,-
16.8H14.1L8,427.2H0L23.1,366.2zM37.9,403.6l-8.1,-221-2.4,-6.6h-0.3l-
2.5,6.6l-8.22H37.9z"/>
 <path android:fillColor="#6AB343"
android:pathData="M60.9,385h6.9v6.3h0.3c1.2,-2.2 3.1,-4 5.6,-
5.4c2.6,-1.5 5.3,-2.2 8.2,-2.2c5.2,0 9.1,1.5 11.8,4.5c2.7,3 4,7.1
4,12.3v26.7h-7.1v-25.6c0,-7.6 -3.5,-11.4 -10.6,-11.4c-2.3,0 -4.3,0.7
-6.1,2c-1.8,1.3 -3.2,3 -4.2,5.1c-1,2.1 -1.5,4.3 -1.5,6.6v23.3h-
7.2v385z"/>
 <path android:fillColor="#6AB343"
android:pathData="M115.7,425.6c-3.1,-1.9 -5.5,-4.6 -7.3,-8c-1.8,-3.4
-2.7,-7.3 -2.7,-11.5c0,-4.3 0.9,-8.1 2.7,-11.5c1.8,-3.4 4.3,-6.1
7.3,-8c3.1,-1.9 6.4,-2.9 10,-2.9c3.3,0 6.2,0.7 8.8,2.2c2.6,1.4
4.6,3.2 5.9,5.4h0.3l-0.3,-5.8v-19.1h7.1v60.9h-6.8v-6.3h-0.3c-1.3,2.2
-3.3,4 -5.9,5.5c-2.6,1.4 -5.6,2.2 -8.8,2.2C122.1,428.5 118.8,427.6
115.7,425.6zM133.8,420.1c2.1,-1.3 3.8,-3.1 5.1,-5.6c1.3,-2.4 1.9,-
5.2 1.9,-8.4c0,-3.2 -0.6,-6.1 -1.9,-8.5c-1.3,-2.4 -3,-4.3 -5.1,-
5.5c-2.1,-1.3 -4.4,-1.9 -6.9,-1.9c-2.5,0 -4.8,0.7 -6.9,2c-2.1,1.3 -
3.8,3.2 -5.1,5.6c-1.3,2.4 -1.9,5.2 -1.9,8.4s0.6,6 1.9,8.4c1.3,2.4
3,4.3 5.1,5.6c2.1,1.3 4.4,2 6.9,2C129.4,422 131.7,421.3
133.8,420.1z"/>
 <path android:fillColor="#6AB343"
android:pathData="M158.3,385h6.9v6.9h0.3c0.8,-2.3 2.4,-4.3 4.9,-
5.9c2.5,-1.6 4.9,-2.4 7.4,-2.4c1.8,0 3.4,0.3 4.9,0.9v7.7c-1.5,-0.9 -
3.5,-1.3 -6,-1.3c-2,0 -3.8,0.6 -5.6,1.7c-1.7,1.2 -3.1,2.7 -4.1,4.6c-
1,1.9 -1.5,4 -1.5,6.3v23.5h-7.2v385z"/>
 <path android:fillColor="#6AB343"
android:pathData="M194.9,425.6c-3.3,-2 -5.9,-4.7 -7.7,-8.1c-1.8,-3.4
-2.8,-7.2 -2.8,-11.4c0,-4.2 0.9,-8 2.8,-11.4c1.8,-3.4 4.4,-6.1 7.7,-
8.1c3.3,-2 7,-2.9 11.1,-2.9c4.1,0 7.9,1 11.2,2.9c3.3,2 5.9,4.7
7.7,8.1c1.8,3.4 2.8,7.2 2.8,11.4c0,4.2 -0.9,8 -2.8,11.4c-1.8,3.4 -
4.4,6.1 -7.7,8.1c-3.3,2 -7,2.9 -11.2,2.9C201.9,428.5 198.2,427.5
194.9,425.6zM213.2,420c2.2,-1.3 4,-3.2 5.3,-5.6c1.3,-2.4 2,-5.2 2,-
8.4s-0.7,-6 -2,-8.4c-1.3,-2.4 -3.1,-4.3 -5.3,-5.6c-2.2,-1.3 -4.6,-2
-7.1,-2c-2.5,0 -4.9,0.7 -7.1,2c-2.2,1.3 -4,3.2 -5.3,5.6c-1.3,2.4 -
2,5.2 -2,8.4s0.7,6 2,8.4c1.3,2.4 3.1,4.3 5.3,5.6c2.2,1.3 4.6,2
7.1,2C208.6,422 211,421.3 213.2,420z"/>

```

```

 <path android:fillColor="#6AB343"
 android:pathData="M235.7,374.4c-1,-1 -1.5,-2.2 -1.5,-3.7c0,-1.4
0.5,-2.6 1.5,-3.7c1,-1 2.2,-1.5 3.7,-1.5c1.4,0 2.6,0.5 3.7,1.5c1,1
1.5,2.2 1.5,3.7c0,1.4 -0.5,2.6 -1.5,3.7c-1,1 -2.2,1.5 -
3.7,1.5c237.9,375.9 236.7,375.4 235.7,374.4zM235.8,385h7.1v42.2h-
7.1V385z"/>
 <path android:fillColor="#6AB343"
 android:pathData="M261.4,425.6c-3.1,-1.9 -5.5,-4.6 -7.3,-8c-1.8,-3.4
-2.7,-7.3 -2.7,-11.5c0,-4.3 0.9,-8.1 2.7,-11.5c1.8,-3.4 4.3,-6.1
7.3,-8c3.1,-1.9 6.4,-2.9 10,-2.9c3.3,0 6.2,0.7 8.8,2.2c2.6,1.4
4.6,3.2 5.9,5.4h0.31-0.3,-5.8v-19.1h7.1v60.9h-6.8v-6.3h-0.3c-1.3,2.2
-3.3,4 -5.9,5.5c-2.6,1.4 -5.6,2.2 -8.8,2.2c267.8,428.5 264.5,427.6
261.4,425.6zM279.5,420.1c2.1,-1.3 3.8,-3.1 5.1,-5.6c1.3,-2.4 1.9,-
5.2 1.9,-8.4c0,-3.2 -0.6,-6.1 -1.9,-8.5c-1.3,-2.4 -3,-4.3 -5.1,-
5.5s-4.4,-1.9 -6.9,-1.9s-4.8,0.7 -6.9,2c-2.1,1.3 -3.8,3.2 -5.1,5.6c-
1.3,2.4 -1.9,5.2 -1.9,8.4s0.6,6 1.9,8.4c1.3,2.4 3,4.3
5.1,5.6c2.1,1.3 4.4,2 6.9,2S277.3,421.3 279.5,420.1z"/>
 <path android:fillColor="#424242"
 android:pathData="M334.3,373.1h-17.1v-6.9h41.4v6.9h-17.1v54h-
7.1V373.1z"/>
 <path android:fillColor="#424242"
 android:pathData="M360.9,385h6.9v6.9h0.3c0.8,-2.3 2.4,-4.3 4.9,-
5.9c2.5,-1.6 4.9,-2.4 7.4,-2.4c1.8,0 3.4,0.3 4.9,0.9v7.7c-1.5,-0.9 -
3.5,-1.3 -6,-1.3c-2,0 -3.8,0.6 -5.6,1.7c-1.7,1.2 -3.1,2.7 -4.1,4.6c-
1,1.9 -1.5,4 -1.5,6.3v23.5h-7.2V385z"/>
 <path android:fillColor="#424242"
 android:pathData="M392.7,374.4c-1,-1 -1.5,-2.2 -1.5,-3.7c0,-1.4
0.5,-2.6 1.5,-3.7c1,-1 2.2,-1.5 3.7,-1.5c1.4,0 2.6,0.5 3.7,1.5c1,1
1.5,2.2 1.5,3.7c0,1.4 -0.5,2.6 -1.5,3.7c-1,1 -2.2,1.5 -
3.7,1.5c394.9,375.9 393.7,375.4 392.7,374.4zM392.8,385h7.1v42.2h-
7.1V385z"/>
 <path android:fillColor="#424242"
 android:pathData="M406.4,385h7.8112.9,33.7h0.2113.1,-33.7h7.71-
17.3,42.2h-7.3L406.4,385z"/>
 <path android:fillColor="#424242"
 android:pathData="M454.5,374.4c-1,-1 -1.5,-2.2 -1.5,-3.7c0,-1.4
0.5,-2.6 1.5,-3.7c1,-1 2.2,-1.5 3.7,-1.5c1.4,0 2.6,0.5 3.7,1.5c1,1
1.5,2.2 1.5,3.7c0,1.4 -0.5,2.6 -1.5,3.7c-1,1 -2.2,1.5 -
3.7,1.5c456.7,375.9 455.5,375.4 454.5,374.4zM454.5,385h7.1v42.2h-
7.1V385z"/>
 <path android:fillColor="#424242"
 android:pathData="M477.9,426.7c-2.5,-1.2 -4.4,-2.9 -5.8,-5.1c-1.4,-
2.2 -2.1,-4.6 -2.1,-7.4c0,-4.5 1.7,-8 5.1,-10.5c3.4,-2.5 7.7,-3.7
13,-3.7c2.6,0 5,0.3 7.1,0.9c2.1,0.6 3.8,1.2 5,1.9v-2.2c0,-2 -0.5,-
3.9 -1.6,-5.5c-1.1,-1.6 -2.5,-2.9 -4.2,-3.7c-1.8,-0.9 -3.7,-1.3 -
5.7,-1.3c-4.8,0 -8.5,2 -11.1,5.91-6,-4c1.8,-2.6 4.1,-4.6 7,-
6.1c2.9,-1.4 6.2,-2.2 9.8,-2.2c5.8,0 10.4,1.6 13.7,4.7c3.3,3.1 5,7.3
5,12.6v26.3h-6.9V421h-0.3c-1.4,2.2 -3.2,4 -5.5,5.4c-2.3,1.4 -5,2.2 -
8.1,2.2S480.4,427.9 477.9,426.7zM493.3,420.4c2,-1.2 3.7,-2.8 5,-
4.9s1.9,-4.3 1.9,-6.6c-3.2,-2 -6.8,-3.1 -11,-3.1c-3.6,0 -6.4,0.8 -
8.6,2.4c-2.1,1.6 -3.2,3.7 -3.2,6.2c0,2.3 0.9,4.1 2.8,5.6c1.9,1.4
4.1,2.2 6.6,2.2c489.1,422.1 491.2,421.6 493.3,420.4z"/>
 <path android:fillColor="#424242"
 android:pathData="M162.7,472.7h8.2123.1,60.9h-81-6,-16.8h-26.31-

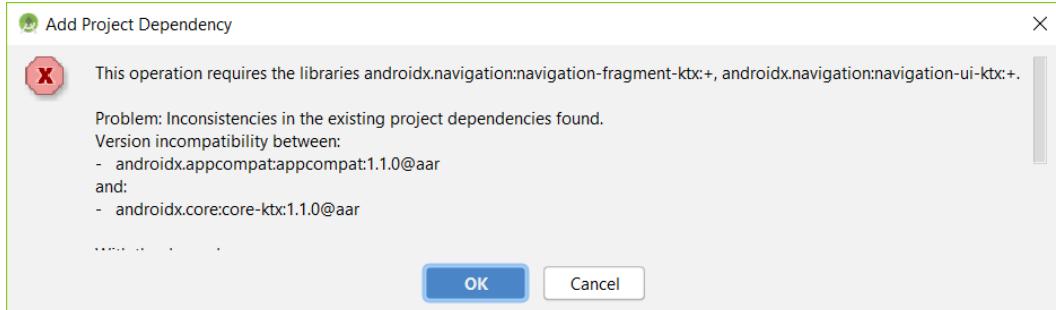
```

```

6.1,16.8h-8L162.7,472.7zM177.4,510.11-8.1,-221-2.4,-6.6h-0.31-
2.5,6.61-8,22H177.4z"/>
 <path android:fillColor="#424242"
 android:pathData="M213.5,532.9c-2.6,-1.4 -4.5,-3.3 -5.8,-5.5h-
0.3v6.3h-6.9v-60.9h7.2v19.11-0.3,5.8h0.3c1.4,-2.2 3.3,-4 5.9,-
5.4c2.6,-1.4 5.5,-2.2 8.8,-2.2c3.6,0 7,1 10,2.9c3.1,2 5.5,4.6
7.3,8c1.8,3.4 2.7,7.2 2.7,11.5c0,4.3 -0.9,8.1 -2.7,11.5c-1.8,3.4 -
4.3,6.1 -7.3,8c-3.1,2 -6.4,2.9 -10,2.9c219.1,535 216.1,534.3
213.5,532.9zM228.2,526.5c2.1,-1.3 3.8,-3.2 5.1,-5.6c1.3,-2.4 1.9,-
5.2 1.9,-8.4c0,-3.2 -0.6,-6 -1.9,-8.4c-1.3,-2.4 -3,-4.3 -5.1,-5.6c-
2.1,-1.3 -4.4,-2 -6.9,-2c-2.5,0 -4.8,0.6 -6.9,1.9c-2.1,1.3 -3.8,3.1
-5.1,5.5c-1.3,2.4 -1.9,5.2 -1.9,8.5c0,3.2 0.6,6 1.9,8.4c1.3,2.4
3,4.3 5.1,5.6c2.1,1.3 4.4,1.9 6.9,1.9c223.8,528.5 226.1,527.8
228.2,526.5z"/>
 <path android:fillColor="#424242"
 android:pathData="M258.8,532.1c-3.3,-2 -5.9,-4.7 -7.7,-8.1c-1.8,-3.4
-2.8,-7.2 -2.8,-11.4c0,-4.2 0.9,-8 2.8,-11.4c1.8,-3.4 4.4,-6.1 7.7,-
8.1c3.3,-2 7,-2.9 11.1,-2.9c4.1,0 7.9,1 11.2,2.9s5.9,4.7
7.7,8.1c1.8,3.4 2.8,7.2 2.8,11.4c0,4.2 -0.9,8 -2.8,11.4c-1.8,3.4 -
4.4,6.1 -7.7,8.1s-7,2.9 -11.2,2.9c265.8,535 262,534
258.8,532.1zM277,526.5c2.2,-1.3 4,-3.2 5.3,-5.6c1.3,-2.4 2,-5.2 2,-
8.4c0,-3.2 -0.7,-6 -2,-8.4c-1.3,-2.4 -3.1,-4.3 -5.3,-5.6c-2.2,-1.3
-4.6,-2 -7.1,-2c-2.5,0 -4.9,0.7 -7.1,2c-2.2,1.3 -4,3.2 -5.3,5.6c-
1.3,2.4 -2,5.2 -2,8.4c0,3.2 0.7,6 2,8.4c1.3,2.4 3.1,4.3
5.3,5.6c2.2,1.3 4.6,2 7.1,2c272.5,528.5 274.8,527.8 277,526.5z"/>
 <path android:fillColor="#424242" android:pathData="M303,530.5c-
2.7,-3 -4,-7.2 -4,-12.5v-26.5h7.2v25.4c0,7.8 3.5,11.7
10.5,11.7c2.3,0 4.4,-0.7 6.3,-2c1.8,-1.3 3.3,-3 4.3,-5.1c1,-2.1
1.5,-4.3 1.5,-6.6v-23.3h7.1v42.2H329v-6.3h-0.3c-1.2,2.2 -3.1,4 -
5.7,5.4c-2.5,1.5 -5.2,2.2 -8.1,2.2c309.7,535 305.7,533.5
303,530.5z"/>
 <path android:fillColor="#424242"
 android:pathData="M358.4,534.2c-1.4,-0.6 -2.7,-1.4 -3.7,-2.3c-1.1,-
1.1 -2,-2.4 -2.5,-3.9c-0.5,-1.5 -0.8,-3.2 -0.8,-5.2v498h-7.5v-
6.6h7.5v-12.6h7.1v12.6h10.4v6.6h-10.4v23c0,2.4 0.5,4.2 1.4,5.4c1,1.3
2.4,1.9 4.4,1.9c1.8,0 3.3,-0.4 4.6,-1.2v7c-0.9,0.3 -1.8,0.6 -
2.6,0.7c-0.8,0.1 -1.9,0.2 -3.3,0.2c361.5,535 359.9,534.7
358.4,534.2z"/>
</vector>

```

10. Tambahkan direktori **navigation** pada resource file. Kemudian buat sebuah file **navigation.xml** pada direktori baru tersebut. Jika ada pesan berikut, klik OK



11. Tulislah kode pada file **navigation.xml** menjadi seperti berikut.

```
<?xml version="1.0" encoding="utf-8"?>
<navigation
 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:id="@+id/navigation"
 app:startDestination="@+id/titleFragment">

 <fragment
 android:id="@+id/titleFragment"
 android:name="com.latihan.darmanto.modul7.TitleFragment"
 android:label="Intro"
 tools:layout="@layout/fragment_title">
 <action

 android:id="@+id/action_titleFragment_to_gameFragment"
 app:destination="@+id/gameFragment" />
 </fragment>
 <fragment
 android:id="@+id/gameFragment"
 android:name="com.latihan.darmanto.modul7.GameFragment"
 android:label="Game"
 tools:layout="@layout/fragment_game">
 <action

 android:id="@+id/action_gameFragment_to_gameOverFragment"
 app:destination="@+id/gameOverFragment"
 app:popUpTo="@+id/gameFragment"
 app:popUpToInclusive="true" />
 <action

 android:id="@+id/action_gameFragment_to_gameWonFragment"
 app:destination="@+id/gameWonFragment"
 app:popUpTo="@+id/gameFragment"
 app:popUpToInclusive="true" />
 </action>
 </fragment>
 <fragment
 android:id="@+id/gameOverFragment"

 android:name="com.latihan.darmanto.modul7.GameOverFragment"
 android:label="Game Failed"
 tools:layout="@layout/fragment_game_over">
 <action

 android:id="@+id/action_gameOverFragment_to_gameFragment"
 app:destination="@+id/gameFragment"
```

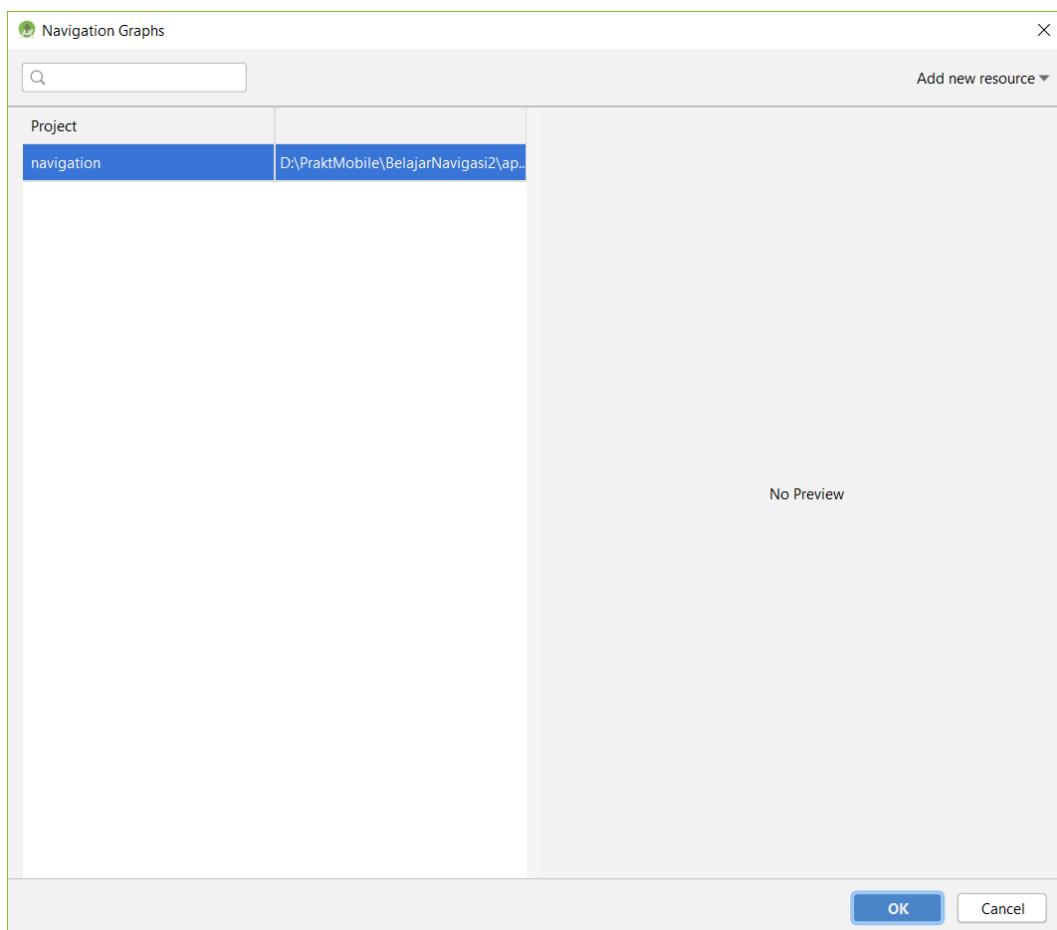
```

 app:popUpTo="@+id/titleFragment"
 app:popUpToInclusive="false" />
 </fragment>
 <fragment
 android:id="@+id/gameWonFragment"

 android:name="com.latihan.darmanto.modul7.GameWonFragment"
 android:label="Game Won"
 tools:layout="@layout/fragment_game_won">
 <action

 android:id="@+id/action_gameWonFragment_to_titleFragment"
 app:destination="@+id/titleFragment"
 app:popUpTo="@+id/titleFragment"
 app:popUpToInclusive="false" />
 </fragment>
</navigation>
```

12. Kemudian, kembali ke **activity\_main.xml**. Tambahkan **NavHostFragment**. Akan muncul layar berikut, klik OK



13. Tambahkan juga **NavigationView**.

14. Pindahlah ke Text mode dan ubahlah kodenya sehingga menjadi sebagai berikut

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto">

 <androidx.drawerlayout.widget.DrawerLayout
 android:id="@+id/drawerLayout"
 android:layout_width="match_parent"
 android:layout_height="match_parent">

 <LinearLayout
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical">

 <fragment
 android:id="@+id/myNavHostFragment"
 android:name=
 "androidx.navigation.fragment.NavHostFragment"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 app:defaultNavHost="true"
 app:navGraph="@navigation/navigation" />
 </LinearLayout>

 <com.google.android.material.navigation.NavigationView
 android:id="@+id/navView"
 android:layout_width="wrap_content"
 android:layout_height="match_parent"
 android:layout_gravity="start"
 app:headerLayout="@layout/nav_header"
 app:menu="@menu/navdrawer_menu" />
 </androidx.drawerlayout.widget.DrawerLayout>
</layout>
```

15. Ubahlah resource **string** menjadi sebagai berikut

```
<resources>
 <string name="app_name">BelajarNavigasi2</string>
 <string name="game_over">Game Over</string>
 <string name="try_again">Try Again?</string>
 <string name="android_trivia">Android Trivia</string>
 <string name="submit_button">Submit</string>
 <string name="congratulations">Congratulations!</string>
</resources>
```

16. Kemudian buatlah empat buah fragment sebagai berikut

fragment\_tittle.xml

```

<?xml version="1.0" encoding="utf-8"?>

<layout 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"
 tools:context="com.example.android.navigation.TitleFragment">

 <androidx.constraintlayout.widget.ConstraintLayout
 android:id="@+id/titleConstraint"
 android:layout_width="match_parent"
 android:layout_height="match_parent">

 <Button
 android:id="@+id/playButton"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_marginStart="@dimen/horizontal_margin"
 android:layout_marginEnd="@dimen/horizontal_margin"
 android:paddingStart="@dimen/button_padding"
 android:paddingEnd="@dimen/button_padding"
 android:text="Play"
 android:textColor="@color/colorAccent"
 android:textSize="@dimen/button_text_size"
 android:textStyle="bold"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/titleImage" />

 <ImageView
 android:id="@+id/titleImage"
 android:layout_width="0dp"
 android:layout_height="@dimen/image_header_height"
 android:layout_marginStart="@dimen/horizontal_margin"
 android:layout_marginEnd="@dimen/horizontal_margin"
 android:scaleType="fitCenter"
 app:layout_constraintBottom_toTopOf="@+id/playButton"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintHorizontal_bias="1.0"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent"
 app:srcCompat="@drawable/android_trivia" />

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

```

fragment\_game.xml

```

<?xml version="1.0" encoding="utf-8"?>
<layout 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"

```

```
tools:context="com.example.android.navigation.InGame">

<ScrollView
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:fillViewport="true">

 <androidx.constraintlayout.widget.ConstraintLayout
 android:id="@+id/frameLayout"
 android:layout_width="match_parent"
 android:layout_height="wrap_content">

 <Button
 android:id="@+id/submitButton"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_marginStart=
 "@dimen/question_horizontal_margin"
 android:layout_marginTop="@dimen/vertical_margin"
 android:layout_marginEnd=
 "@dimen/question_horizontal_margin"
 android:text="@string/submit_button"
 android:textSize="@dimen/button_text_size"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf=
 "@+id/questionRadioGroup" />

 <ImageView
 android:id="@+id/questionImage"
 android:layout_width="0dp"
 android:layout_height="@dimen/image_header_height"
 android:layout_marginStart="@dimen/horizontal_margin"
 android:layout_marginTop="@dimen/vertical_margin"
 android:layout_marginEnd="@dimen/horizontal_margin"
 android:layout_marginBottom="@dimen/vertical_margin"
 android:scaleType="fitCenter"
 app:layout_constraintBottom_toTopOf="@+id/questionText"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent"
 app:layout_constraintVertical_chainStyle="packed"
 app:srcCompat="@drawable/android_category_simple" />

 <RadioGroup
 android:id="@+id/questionRadioGroup"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_marginStart=
 "@dimen/question_horizontal_margin"
 android:layout_marginTop="@dimen/vertical_margin"
 android:layout_marginEnd=
 "@dimen/question_horizontal_margin"
 android:animateLayoutChanges="true"
```

```
 android:orientation="vertical"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf=
 "@+id/questionText">>

 <RadioButton
 android:id="@+id/firstAnswerRadioButton"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_marginBottom=
 "@dimen/question_vertical_margin"
 android:checked="true"
 android:text="Blue"
 android:textSize="@dimen/answer_text_size" />

 <RadioButton
 android:id="@+id/secondAnswerRadioButton"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_marginBottom=
 "@dimen/question_vertical_margin"
 android:text="Green"
 android:textSize="@dimen/answer_text_size" />

 <RadioButton
 android:id="@+id/thirdAnswerRadioButton"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_marginBottom=
 "@dimen/question_vertical_margin"
 android:text="Yellow"
 android:textSize="@dimen/answer_text_size" />

 <RadioButton
 android:id="@+id/fourthAnswerRadioButton"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Red"
 android:textSize="@dimen/answer_text_size" />
 </RadioGroup>

 <TextView
 android:id="@+id/questionText"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_marginStart=
 "@dimen/question_horizontal_margin"
 android:layout_marginTop="@dimen/vertical_margin"
 android:layout_marginEnd=
 "@dimen/question_horizontal_margin"
 android:layout_marginBottom="@dimen/vertical_margin"
 android:fontFamily="sans-serif"
 android:text="What color is the Android mascot?"
```

```

 android:textSize="@dimen/question_text_size"
 android:textStyle="bold"
 android:typeface="normal"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf=
 "@+id/questionImage" />
 </androidx.constraintlayout.widget.ConstraintLayout>
</ScrollView>
</layout>

```

#### fragment\_game\_over.xml

```

<?xml version="1.0" encoding="utf-8"?>

<layout 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"
 tools:context="com.example.android.navigation.GameOverFragment">

 <androidx.constraintlayout.widget.ConstraintLayout
 android:id="@+id/gameOverConstraintLayout"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:background="@color/gameOverBackground">

 <ImageView
 android:id="@+id/gameOverFragment"
 android:layout_width="wrap_content"
 android:layout_height="362dp"
 android:layout_marginStart="@dimen/horizontal_margin"
 android:layout_marginEnd="@dimen/horizontal_margin"
 android:layout_marginBottom="8dp"
 android:scaleType="fitCenter"
 app:layout_constraintBottom_toTopOf="@+id/tryAgainButton"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent"
 app:layout_constraintVertical_chainStyle="packed"
 app:srcCompat="@drawable/try_again" />

 <Button
 android:id="@+id/tryAgainButton"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_marginStart="@dimen/vertical_margin"
 android:layout_marginTop="@dimen/vertical_margin"
 android:layout_marginEnd="@dimen/vertical_margin"
 android:layout_marginBottom="8dp"
 android:paddingStart="@dimen/button_padding"
 android:paddingEnd="@dimen/button_padding"
 android:text="@string/try_again"
 android:textColor="?colorAccent" />

```

```

 android:textSize="@dimen/button_text_size"
 android:textStyle="bold"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/gameOverFragment"
 />
 </androidx.constraintlayout.widget.ConstraintLayout>
</layout>

```

### fragment\_game\_won.xml

```

<?xml version="1.0" encoding="utf-8"?>
<layout 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"
 tools:context="com.example.android.navigation.GameWonFragment">

 <androidx.constraintlayout.widget.ConstraintLayout
 android:id="@+id/gameWonConstraintLayout"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:background="@color/youWinBackground">

 <Button
 android:id="@+id/nextMatchButton"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:paddingStart="@dimen/button_padding"
 android:paddingEnd="@dimen/button_padding"
 android:text="Next Match"
 android:textColor="@color/youWinDark"
 android:textSize="@dimen/button_text_size"
 android:textStyle="bold"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/youWinImage" />

 <ImageView
 android:id="@+id/youWinImage"
 android:layout_width="0dp"
 android:layout_height="@dimen/game_over_height"
 android:layout_marginStart="@dimen/horizontal_margin"
 android:layout_marginTop="8dp"
 android:layout_marginEnd="@dimen/horizontal_margin"
 android:layout_marginBottom="@dimen/vertical_margin"
 android:scaleType="fitCenter"
 app:layout_constraintBottom_toTopOf="@+id/nextMatchButton"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent"

```

```

 app:srcCompat="@drawable/you_win" />

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

```

17. Ubahlah, menjadi

**Color.xml**

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
 <color name="colorPrimary">#008577</color>
 <color name="colorPrimaryDark">#00574B</color>
 <color name="colorAccent">#7e53c5</color>
 <color name="youWinDark">#35A571</color>
 <color name="youWinBackground">#B7F8C9</color>
 <color name="gameOverBackground">#f8bbd0</color>
</resources>

```

**String.xml**

```

<resources>
 <string name="app_name">BelajarNavigasi2</string>
 <string name="game_over">Game Over</string>
 <string name="try_again">Try Again?</string>
 <string name="android_trivia">Android Trivia</string>
 <string name="submit_button">Submit</string>
 <string name="congratulations">Congratulations!</string>
</resources>

```

18. Tambahkan pada drawable

**android\_category\_simple.xml**

```

<vector android:height="411.3dp" android:viewportHeight="411.3"
 android:viewportWidth="473.8" android:width="473.8dp"
 xmlns:android="http://schemas.android.com/apk/res/android">
 <path android:fillColor="#6AB343"
 android:pathData="M353.1,221.9136.7,-67c2,-3.5 0.7,-8 -2.9,-9.9c-3.6,-
 1.9 -8,-0.6 -9.9,2.91-37.1,67.7c-31.2,-13.9 -66.1,-21.7 -103.1,-21.6c-
 36.8,0 -71.7,7.7 -102.8,21.51-37.1,-67.6c-1.9,-3.6 -6.4,-4.9 -9.9,-
 2.9c-3.6,1.9 -4.9,6.4 -2.9,9.9136.7,67C48.7,259.1 0,330
 0,411.31473.8,0C473.8,330 425.1,259.2 353.1,221.9M129.1,325.5c-10.9,0 -
 19.8,-8.9 -19.8,-19.8c0,-10.9 8.9,-19.9 19.8,-19.9c11,0 19.9,9
 19.9,19.9c149,316.6 140.1,325.5 129.1,325.5M344.8,325.5c-11,0 -19.9,-
 8.9 -19.9,-19.9c0,-10.9 8.9,-19.9 19.9,-19.9c10.9,0 19.8,9
 19.8,19.9c364.7,316.6 355.8,325.5 344.8,325.5"/>
 <path android:fillColor="#424242" android:pathData="M226.8,93.1c0,-
 6.6 1.2,-12 3.6,-16.4c2.4,-4.4 6.9,-9.5 13.5,-15.5c5.1,-4.7 8.8,-8.8
 11.1,-12.1c2.3,-3.3 3.4,-7.4 3.4,-12.1c0,-6.3 -2,-11.4 -6.1,-15.3c-

```

```

 4.1,-3.9 -9.7,-5.8 -16.9,-5.8c-6.6,0 -11.7,1.9 -15.5,5.6c-3.8,3.7 -
 6.7,8.4 -8.7,13.91-15.5,-6.5c2.6,-7.6 7.3,-14.3 14,-20.2C216.4,2.9
 225,0 235.5,0c7.9,0 14.9,1.6 21,4.8c6.1,3.2 10.8,7.6 14.2,13.3c3.3,5.6
 5,11.9 5,18.9c0,5 -1.1,9.6 -3.1,13.9c-2.1,4.3 -4.4,7.7 -6.8,10.4c-
 2.4,2.7 -5.5,5.8 -9.3,9.3c-4.3,3.9 -7.5,7.7 -9.5,11.2c-2,3.5 -3.1,7.7 -
 3.1,12.4v8.3h-16.9V93.1zM226.6,140.4c-2.4,-2.4 -3.6,-5.4 -3.6,-9c0,-3.4
 1.2,-6.3 3.6,-8.8c2.4,-2.4 5.3,-3.6 8.8,-3.6c3.5,0 6.5,1.2
 9,3.6c2.4,2.4 3.6,5.3 3.6,8.8c0,3.5 -1.2,6.5 -3.6,9c-2.4,2.4 -5.4,3.6 -
 9,3.6C231.9,144.1 229,142.9 226.6,140.4z"/>
 <path android:fillColor="#424242" android:pathData="M126.3,82.2c-
 2.1,-4.4 -3,-8.5 -2.7,-12.2c0.3,-3.7 1.7,-8.6 4.2,-14.6c2,-4.8 3.2,-8.7
 3.7,-11.6c0.5,-3 0,-6.1 -1.5,-9.2c-2,-4.2 -4.9,-7 -8.9,-8.4c-4,-1.3 -
 8.4,-0.9 -13.2,1.4c-4.4,2.1 -7.3,4.9 -8.7,8.7c-1.4,3.7 -1.9,7.7 -
 1.5,12.11-12.5,0.5c-0.6,-6 0.4,-11.9 3.1,-18c2.7,-6 7.6,-10.7 14.6,-
 14c5.3,-2.5 10.5,-3.6 15.6,-3.3c5.1,0.3 9.7,1.8 13.7,4.5c4,2.8 7.1,6.5
 9.3,11.2c1.6,3.4 2.3,6.8 2.2,10.3c-0.1,3.5 -0.5,6.6 -1.3,9.2c-0.8,2.6 -
 1.9,5.6 -3.4,9.2c-1.7,4 -2.7,7.5 -2.9,10.6c-0.3,3 0.3,6.1
 1.8,9.312.6,5.61-11.4,5.3L126.3,82.2zM140.9,114.2c-2.4,-0.9 -4.1,-2.5 -
 5.3,-4.9c-1.1,-2.3 -1.2,-4.6 -0.3,-7c0.9,-2.4 2.5,-4.1 4.8,-5.2c2.4,-
 1.1 4.8,-1.2 7.2,-0.4c2.4,0.9 4.1,2.5 5.2,4.8c1.1,2.4 1.2,4.8 0.4,7.2c-
 0.9,2.4 -2.5,4.1 -4.9,5.3C145.6,114.9 143.3,115 140.9,114.2z"/>
 <path android:fillColor="#424242" android:pathData="M320.7,127.6c-
 0.7,-1.7 -0.7,-3.4 0,-5.1c0.7,-1.7 1.9,-2.8 3.5,-3.5c1.7,-0.7 3.3,-0.7
 5,0c1.7,0.7 2.9,1.9 3.6,3.6c0.7,1.7 0.7,3.3 0,5c-0.7,1.7 -1.9,2.9 -
 3.6,3.6c-1.7,0.7 -3.4,0.7 -5.1,0C322.5,130.5 321.3,129.3
 320.7,127.6zM330.2,104.7c1.3,-3.2 3,-5.6 5,-7.2c2.1,-1.6 5.3,-3.3 9.6,-
 4.8c3.4,-1.3 6,-2.5 7.8,-3.7c1.8,-1.2 3.1,-2.9 4.1,-5.2c1.3,-3.1 1.3,-
 5.9 0.1,-8.6c-1.2,-2.7 -3.6,-4.8 -7.1,-6.2c-3.2,-1.3 -6.1,-1.4 -8.7,-
 0.4c-2.6,1.1 -4.9,2.7 -7.51-6.2,-6.3c2.8,-3.2 6.4,-5.5 10.8,-7c4.4,-1.5
 9.2,-1.2 14.3,0.9c3.8,1.6 6.9,3.7 9.2,6.5c2.3,2.8 3.7,5.9
 4.2,9.3c0.5,3.4 0.1,6.8 -1.3,10.2c-1,2.4 -2.4,4.5 -4.3,6.1c-1.9,1.7 -
 3.7,2.9 -5.4,3.7c-1.7,0.8 -3.8,1.7 -6.4,2.7c-2.9,1 -5.2,2.2 -6.9,3.5c-
 1.7,1.3 -3,3.1 -4,5.41-1.6,41-8.2,-3.4L330.2,104.7z"/>
</vector>

```

### android\_trivia.xml

```

<vector android:height="428.5dp" android:viewportHeight="428.5"
 android:viewportWidth="507" android:width="507dp"
 xmlns:android="http://schemas.android.com/apk/res/android">
 <path android:fillColor="#6AB343" android:pathData="M255.6,256.5c-
 9.9,0 -18.4,3.5 -25.3,10.3c-6.9,6.9 -10.3,15.3 -10.3,25.3s3.4,18.4
 10.3,25.5c6.9,7 15.3,10.6 25.3,10.6c9.9,0 18.4,-3.5 25.5,-10.6c7,-7
 10.6,-15.5 10.6,-25.5s-3.5,-18.4 -10.6,-25.3C274.1,260 265.6,256.5
 255.6,256.5z"/>
 <path android:fillColor="#6AB343" android:pathData="M332.8,63.8c-
 6.6,-10.4 -15.4,-18.8 -26.4,-25.4l14.1,-25.7c1.1,-2 1.4,-4.3 0.7,-6.5c-
 0.6,-2.2 -2.1,-4.1 -4.1,-5.2L317,1c-4.2,-2.2 -9.4,-0.7 -11.6,3.51-
 14.6,26.6c-11.1,-4 -23.2,-6 -36.4,-6c-12.8,0 -24.5,2 -
 34.9,5.9L205,4.5c-1.1,-2 -2.9,-3.5 -5.1,-4.1c-2.2,-0.6 -4.5,-0.4 -
 6.5,0.7c-2,1.1 -3.5,2.9 -4.2,5.2c-0.6,2.2 -0.4,4.5 0.7,6.5l14.3,26.1c-
 2.5,1.7 -5.3,5.5c-15.2,12.8 -25.4,28.1 -
 30.6,45.7147.6,19.9c2.8,-9.7 7.7,-17.7 14.9,-24.2C235.9,79.3 245,76

```

```

256,76c10.5,0 18.9,3 25.3,8.9c6.3,5.9 9.5,13.2 9.5,21.7c0,7.2 -1.9,13.5
-5.6,19c-3.7,5.5 -9.7,12 -18,19.5c-14.4,12.4 -24.3,23.5 -29.8,33.1c-
5.5,9.7 -8.3,21.1 -8.3,34.4v13.7h54.2v-6.6c0,-9.7 1.7,-17.7 5.2,-
24c3.4,-6.3 9.9,-13.9 19.2,-22.8c12.1,-11.6 21.2,-22.4 27.3,-32.5c6.1,-
10.1 9.1,-22.3 9.1,-36.6C344.2,89.2 340.4,75.8 332.8,63.8z"/>
 <path android:fillColor="#6AB343"
android:pathData="M23.1,366.2h8.2123.1,60.9h-81-6,-
16.8H14.1L8,427.2H0L23.1,366.2zM37.9,403.6l-8.1,-221-2.4,-6.6h-0.3l-
2.5,6.6l-8,22H37.9z"/>
 <path android:fillColor="#6AB343"
android:pathData="M60.9,385h6.9v6.3h0.3c1.2,-2.2 3.1,-4 5.6,-5.4c2.6,-
1.5 5.3,-2.2 8.2,-2.2c5.2,0 9.1,1.5 11.8,4.5c2.7,3 4,7.1 4,12.3v26.7h-
7.1v-25.6c0,-7.6 -3.5,-11.4 -10.6,-11.4c-2.3,0 -4.3,0.7 -6.1,2c-1.8,1.3
-3.2,3 -4.2,5.1c-1,2.1 -1.5,4.3 -1.5,6.6v23.3h-7.2V385z"/>
 <path android:fillColor="#6AB343" android:pathData="M115.7,425.6c-
3.1,-1.9 -5.5,-4.6 -7.3,-8c-1.8,-3.4 -2.7,-7.3 -2.7,-11.5c0,-4.3 0.9,-
8.1 2.7,-11.5c1.8,-3.4 4.3,-6.1 7.3,-8c3.1,-1.9 6.4,-2.9 10,-2.9c3.3,0
6.2,0.7 8.8,2.2c2.6,1.4 4.6,3.2 5.9,5.4h0.3l-0.3,-5.8v-19.1h7.1v60.9h-
6.8v-6.3h-0.3c-1.3,2.2 -3.3,4 -5.9,5.5c-2.6,1.4 -5.6,2.2 -
8.8,2.2C122.1,428.5 118.8,427.6 115.7,425.6zM133.8,420.1c2.1,-1.3 3.8,-
3.1 5.1,-5.6c1.3,-2.4 1.9,-5.2 1.9,-8.4c0,-3.2 -0.6,-6.1 -1.9,-8.5c-
1.3,-2.4 -3,-4.3 -5.1,-5.5c-2.1,-1.3 -4.4,-1.9 -6.9,-1.9c-2.5,0 -
4.8,0.7 -6.9,2c-2.1,1.3 -3.8,3.2 -5.1,5.6c-1.3,2.4 -1.9,5.2 -
1.9,8.4s0.6,6 1.9,8.4c1.3,2.4 3,4.3 5.1,5.6c2.1,1.3 4.4,2
6.9,2C129.4,422 131.7,421.3 133.8,420.1z"/>
 <path android:fillColor="#6AB343"
android:pathData="M158.3,385h6.9v6.9h0.3c0.8,-2.3 2.4,-4.3 4.9,-
5.9c2.5,-1.6 4.9,-2.4 7.4,-2.4c1.8,0 3.4,0.3 4.9,0.9v7.7c-1.5,-0.9 -
3.5,-1.3 -6,-1.3c-2,0 -3.8,0.6 -5.6,1.7c-1.7,1.2 -3.1,2.7 -4.1,4.6c-
1,1.9 -1.5,4 -1.5,6.3v23.5h-7.2V385z"/>
 <path android:fillColor="#6AB343" android:pathData="M194.9,425.6c-
3.3,-2 -5.9,-4.7 -7.7,-8.1c-1.8,-3.4 -2.8,-7.2 -2.8,-11.4c0,-4.2 0.9,-8
2.8,-11.4c1.8,-3.4 4.4,-6.1 7.7,-8.1c3.3,-2 7,-2.9 11.1,-2.9c4.1,0
7.9,1 11.2,2.9c3.3,2 5.9,4.7 7.7,8.1c1.8,3.4 2.8,7.2 2.8,11.4c0,4.2 -
0.9,8 -2.8,11.4c-1.8,3.4 -4.4,6.1 -7.7,8.1c-3.3,2 -7,2.9 -
11.2,2.2C201.9,428.5 198.2,427.6 194.9,425.6zM213.2,420c2.2,-1.3 4,-3.2
5.3,-5.6c1.3,-2.4 2,-5.2 2,-8.4s-0.7,-6 -2,-8.4c-1.3,-2.4 -3.1,-4.3 -
5.3,-5.6c-2.2,-1.3 -4.6,-2 -7.1,-2c-2.5,0 -4.9,0.7 -7.1,2c-2.2,1.3 -
4,3.2 -5.3,5.6c-1.3,2.4 -2,5.2 -2,8.4s0.7,6 2,8.4c1.3,2.4 3.1,4.3
5.3,5.6c2.2,1.3 4.6,2 7.1,2C208.6,422 211,421.3 213.2,420z"/>
 <path android:fillColor="#6AB343" android:pathData="M235.7,374.4c-
1,-1 -1.5,-2.2 -1.5,-3.7c0,-1.4 0.5,-2.6 1.5,-3.7c1,-1 2.2,-1.5 3.7,-
1.5c1.4,0 2.6,0.5 3.7,1.5c1,1 1.5,2.2 1.5,3.7c0,1.4 -0.5,2.6 -1.5,3.7c-
1,1 -2.2,1.5 -3.7,1.5C237.9,375.9 236.7,375.4
235.7,374.4zM235.8,385h7.1v42.2h-7.1V385z"/>
 <path android:fillColor="#6AB343" android:pathData="M261.4,425.6c-
3.1,-1.9 -5.5,-4.6 -7.3,-8c-1.8,-3.4 -2.7,-7.3 -2.7,-11.5c0,-4.3 0.9,-
8.1 2.7,-11.5c1.8,-3.4 4.3,-6.1 7.3,-8c3.1,-1.9 6.4,-2.9 10,-2.9c3.3,0
6.2,0.7 8.8,2.2c2.6,1.4 4.6,3.2 5.9,5.4h0.3l-0.3,-5.8v-19.1h7.1v60.9h-
6.8v-6.3h-0.3c-1.3,2.2 -3.3,4 -5.9,5.5c-2.6,1.4 -5.6,2.2 -
8.8,2.2C267.8,428.5 264.5,427.6 261.4,425.6zM279.5,420.1c2.1,-1.3 3.8,-
3.1 5.1,-5.6c1.3,-2.4 1.9,-5.2 1.9,-8.4c0,-3.2 -0.6,-6.1 -1.9,-8.5c-
1.3,-2.4 -3,-4.3 -5.1,-5.5s-4.4,-1.9 -6.9,-1.9c-2.5,0 -4.8,0.7 -6.9,2c-
2.1,1.3 -3.8,3.2 -5.1,5.6c-1.3,2.4 -1.9,5.2 -1.9,8.4s0.6,6

```

```

1.9,8.4c1.3,2.4 3,4.3 5.1,5.6c2.1,1.3 4.4,2 6.9,2C275,422 277.3,421.3
279.5,420.1z"/>
 <path android:fillColor="#424242" android:pathData="M334.3,373.1h-
17.1v-6.9h41.4v6.9h-17.1v54h-7.1V373.1z"/>
 <path android:fillColor="#424242"
android:pathData="M360.9,385h6.9v6.9h0.3c0.8,-2.3 2.4,-4.3 4.9,-
5.9c2.5,-1.6 4.9,-2.4 7.4,-2.4c1.8,0 3.4,0.3 4.9,0.9v7.7c-1.5,-0.9 -
3.5,-1.3 -6,-1.3c-2,0 -3.8,0.6 -5.6,1.7c-1.7,1.2 -3.1,2.7 -4.1,4.6c-
1,1.9 -1.5,4 -1.5,6.3v23.5h-7.2V385z"/>
 <path android:fillColor="#424242" android:pathData="M392.7,374.4c-
1,-1 -1.5,-2.2 -1.5,-3.7c0,-1.4 0.5,-2.6 1.5,-3.7c1,-1 2.2,-1.5 3.7,-
1.5c1.4,0 2.6,0.5 3.7,1.5c1,1 1.5,2.2 1.5,3.7c0,1.4 -0.5,2.6 -1.5,3.7c-
1,1 -2.2,1.5 -3.7,1.5C394.9,375.9 393.7,375.4
392.7,374.4zM392.8,385h7.1v42.2h-7.1V385z"/>
 <path android:fillColor="#424242"
android:pathData="M406.4,385h7.8112.9,33.7h0.2113.1,-33.7h7.71-
17.3,42.2h-7.3L406.4,385z"/>
 <path android:fillColor="#424242" android:pathData="M454.5,374.4c-
1,-1 -1.5,-2.2 -1.5,-3.7c0,-1.4 0.5,-2.6 1.5,-3.7c1,-1 2.2,-1.5 3.7,-
1.5c1.4,0 2.6,0.5 3.7,1.5c1,1 1.5,2.2 1.5,3.7c0,1.4 -0.5,2.6 -1.5,3.7c-
1,1 -2.2,1.5 -3.7,1.5C456.7,375.9 455.5,375.4
454.5,374.4zM454.5,385h7.1v42.2h-7.1V385z"/>
 <path android:fillColor="#424242" android:pathData="M477.9,426.7c-
2.5,-1.2 -4.4,-2.9 -5.8,-5.1c-1.4,-2.2 -2.1,-4.6 -2.1,-7.4c0,-4.5 1.7,-
8 5.1,-10.5c3.4,-2.5 7.7,-3.7 13,-3.7c2.6,0 5,0.3 7.1,0.9c2.1,0.6
3.8,1.2 5,1.9v-2.2c0,-2 -0.5,-3.9 -1.6,-5.5c-1.1,-1.6 -2.5,-2.9 -4.2,-
3.7c-1.8,-0.9 -3.7,-1.3 -5.7,-1.3c-4.8,0 -8.5,2 -11.1,5.91-6,-4c1.8,-
2.6 4.1,-4.6 7,-6.1c2.9,-1.4 6.2,-2.2 9.8,-2.2c5.8,0 10.4,1.6
13.7,4.7c3.3,3.1 5,7.3 5,12.6v26.3h-6.9V421h-0.3c-1.4,2.2 -3.2,4 -
5.5,5.4c-2.3,1.4 -5,2.2 -8.1,2.2s480.4,427.9
477.9,426.7zM493.3,420.4c2,-1.2 3.7,-2.8 5,-4.9s1.9,-4.3 1.9,-6.6c-
3.2,-2 -6.8,-3.1 -11,-3.1c-3.6,0 -6.4,0.8 -8.6,2.4c-2.1,1.6 -3.2,3.7 -
3.2,6.2c0,2.3 0.9,4.1 2.8,5.6c1.9,1.4 4.1,2.2 6.6,2.2C489.1,422.1
491.2,421.6 493.3,420.4z"/>
</vector>

```

try\_again.xml

```

<vector android:height="806.5dp" android:viewportHeight="806.5"
 android:viewportWidth="534.4" android:width="534.4dp"
 xmlns:android="http://schemas.android.com/apk/res/android">
 <path android:fillColor="#6AB343" android:pathData="M39,230.3c-
21.5,0 -39,16.8 -39,37.4l0,156.8c0,445.3 17.5,462 39,462c21.5,0 39,-
16.7 39,-37.5l0,-156.8c78,247.1 60.5,230.3 39,230.3M352.7,55.4127.2,-
47.7c1.5,-2.5 0.5,-5.7 -2.1,-7.1c-2.7,-1.3 -5.9,-0.4 -7.3,2.11-
27.5,48.2c-23.1,-9.9 -49,-15.4 -76.4,-15.4c-27.3,0 -53.2,5.5 -
76.2,15.3L162.9,2.8c-1.4,-2.5 -4.7,-3.5 -7.3,-2.1c-2.6,1.4 -3.6,4.5 -
2.1,7127.2,47.7c-53.5,26.5 -89.6,76.9 -89.6,134.91351,0C442.1,132.3
406.1,82 352.7,55.4M186.7,166.5c-8.1,0 -14.7,-6.3 -14.7,-14.1c0,-7.8
6.6,-14.2 14.7,-14.2c8.1,0 14.7,6.4 14.7,14.2C201.5,160.2 194.9,166.5
186.7,166.5M346.6,166.5c-8.1,0 -14.7,-6.3 -14.7,-14.1c0,-7.7 6.6,-14.1
14.7,-14.2c8.1,0 14.7,6.4 14.7,14.2C361.3,160.1 354.7,166.5
346.6,166.5M93.1,202.8l0.1,243c0,22.1 18.6,39.9

```

```

41.6,40128.3,010,83c0,20.7 17.5,37.4 38.9,37.4c21.5,0 39,-16.8 39,-
37.510,-83152.6,010,83c0,20.6 17.5,37.5 39,37.4c21.5,0 39,-16.8 39,-
37.510,-82.9128.4,0c22.9,0 41.6,-17.9 41.6,-4010,-
243L93.1,202.8zM534.4,267.7c0,-20.7 -17.4,-37.4 -39,-37.4c-21.5,0 -
39,16.8 -39,37.510,156.8c0,20.7 17.4,37.4 39,37.4c21.5,0 39,-16.7 39,-
37.5L534.4,267.7z"/>
 <path android:fillColor="#FF000000"
 android:pathData="M28.8,698.2H2.7v-
10.1h63.2v10.1H39.8v79.3H28.8V698.2z"/>
 <path android:fillColor="#FF000000"
 android:pathData="M69.5,715.5h10.5v10.1h0.5c1.2,-3.4 3.7,-6.3 7.5,-
8.6c3.8,-2.3 7.5,-3.5 11.2,-3.5c2.7,0 5.2,0.4 7.5,1.2v11.4c-2.3,-1.2 -
5.4,-1.9 -9.1,-1.9c-3,0 -5.9,0.9 -8.5,2.6c-2.6,1.7 -4.7,4 -6.3,6.8c-
1.6,2.8 -2.3,5.9 -2.3,9.3v34.6H69.5V715.5z"/>
 <path android:fillColor="#FF000000"
 android:pathData="M139.6,773.5l-26.8,-58h12120,45.6h0.3119.4,-45.6h121-
40.3,88.9h-11.3L139.6,773.5z"/>
 <path android:fillColor="#FF000000"
 android:pathData="M240.7,688.1h12.5l35.4,89.4h-12.21-9.2,-24.6H2271-
9.4,24.6h-12.2L240.7,688.1zM263.2,742.91-12.3,-32.21-3.6,-9.6h-0.5l-
3.8,9.6l-12.2,32.2H263.2z"/>
 <path android:fillColor="#FF000000"
 android:pathData="M306.4,801.1c-4.9,-3.6 -8.3,-7.7 -10,-12.5l10.3,-
4.1c1.3,3.7 3.6,6.6 7.8.9c3.3,2.3 7.3,3.4 11.9,3.4c6.8,0 12.1,-1.9
15.8,-5.7c3.7,-3.8 5.5,-9.2 5.5,-16v-6.6h-0.5c-2.1,3.2 -5.1,5.8 -
9,7.9c-3.9,2.1 -8.3,3.1 -13.3,3.1c-5.5,0 -10.5,-1.4 -15.2,-4.3c-4.7,-
2.8 -8.4,-6.8 -11.2,-11.8c-2.8,-5 -4.2,-10.7 -4.2,-16.9s1.4,-11.9 4.2,-
16.9c2.8,-5 6.5,-8.9 11.2,-11.8c4.7,-2.8 9.8,-4.3 15.2,-4.3c4.9,0 9.4,1
13.3,3.1c3.9,2.1 6.9,4.7 9,7.9h0.5v-9h10.5V775c0,6.8 -1.4,12.6 -
4.2,17.4c-2.8,4.7 -6.6,8.3 -11.4,10.6c-4.8,2.3 -10.2,3.5 -
16.3,3.5C317.7,806.5 311.3,804.7 306.4,801.1zM336.3,767.1c3.2,-1.9
5.8,-4.6 7.7,-8.1c2,-3.5 2.9,-7.7 2.9,-12.4s-1,-8.9 -2.9,-12.5c-2,-3.6
-4.5,-6.3 -7.7,-8.1c-3.2,-1.8 -6.7,-2.8 -10.5,-2.8c-3.8,0 -7.3,0.9 -
10.5,2.8c-3.2,1.9 -5.8,4.6 -7.7,8.1c-1.9,3.5 -2.9,7.7 -2.9,12.4s1,8.9
2.9,12.4c2,3.5 4.5,6.2 7.7,8.1c3.2,1.9 6.7,2.8 10.5,2.8C329.6,769.9
333.1,768.9 336.3,767.1z"/>
 <path android:fillColor="#FF000000"
 android:pathData="M381.2,776.8c-3.8,-1.8 -6.7,-4.3 -8.8,-7.5c-2.1,-3.2
-3.2,-6.8 -3.2,-10.8c0,-6.6 2.6,-11.7 7.7,-15.4c5.2,-3.7 11.8,-5.5
19.8,-5.5c4,0 7.6,0.4 10.8,1.2c3.2,0.8 5.8,1.7 7.7,2.7v-3.2c0,-3 -0.8,-
5.7 -2.4,-8.1c-1.6,-2.4 -3.8,-4.2 -6.4,-5.5c-2.7,-1.3 -5.6,-1.9 -8.7,-
1.9c-7.4,0 -13.2,2.9 -16.9,8.6l-9.2,-5.9c2.7,-3.8 6.2,-6.8 10.7,-
8.9c4.4,-2.1 9.4,-3.2 15,-3.2c8.8,0 15.8,2.3 20.9,6.9c5.1,4.6 7.7,10.7
7.7,18.5v38.6h-10.5v-9.1h-0.5c-2.1,3.2 -4.9,5.8 -8.5,7.9c-3.6,2.1 -
7.7,3.2 -12.3,3.2C389.2,779.5 385,778.6 381.2,776.8zM404.7,767.5c3.1,-
1.7 5.7,-4.1 7.6,-7.1s2.9,-6.2 2.9,-9.7c-4.9,-3 -10.4,-4.5 -16.8,-4.5c-
5.5,0 -9.8,1.2 -13.1,3.6c-3.2,2.4 -4.9,5.4 -4.9,9.1c0,3.3 1.4,6.1
4.3,8.2c2.9,2.1 6.2,3.2 10,3.2C398.2,770.1 401.5,769.3 404.7,767.5z"/>
 <path android:fillColor="#FF000000" android:pathData="M441.7,700c-
1.6,-1.5 -2.3,-3.3 -2.3,-5.4c0,-2.1 0.8,-3.9 2.3,-5.4c1.6,-1.5 3.4,-2.2
5.6,-2.2c2.2,0 4.0,7 5.6,2.2c1.6,1.5 2.3,3.3 2.3,5.4c0,2.1 -0.8,3.9 -
2.3,5.4c-1.5,1.5 -3.4,2.2 -5.7,2.2C445.1,702.3 443.3,701.5
441.7,700zM441.8,715.5h10.9v62h-10.9V715.5z"/>
 <path android:fillColor="#FF000000"

```

```

 android:pathData="M469.9,715.5h10.5v9.2h0.5c1.8,-3.2 4.7,-5.8 8.6,-
8c3.9,-2.2 8.1,-3.2 12.5,-3.2c7.9,0 13.9,2.2 18,6.6c4.1,4.4 6.2,10.5
6.2,18.1v39.2h-10.9v-37.6c0,-11.2 -5.4,-16.7 -16.2,-16.7c-3.5,0 -6.6,1
-9.4,2.9c-2.8,1.9 -4.9,4.4 -6.4,7.5c-1.5,3.1 -2.3,6.3 -2.3,9.7v34.2h-
11.1V715.5z" />
</vector>

```

you\_win.xml

```

<vector android:height="839.3dp" android:viewportHeight="839.3"
 android:viewportWidth="936.8" android:width="936.8dp"
 xmlns:android="http://schemas.android.com/apk/res/android">
 <path android:fillColor="#6AB343"
 android:pathData="M269.6,269.7c15.2,-15.2 15.2,-39.9 0,-
55.1L154.2,99.2C139,84 114.3,84 99.1,99.2c-15.2,15.2 -15.2,39.9
0,55.11115.4,115.4C229.7,284.9 254.4,284.9
269.6,269.7M554.5,57.7L581.7,8c1.5,-2.6 0.5,-5.9 -2.1,-7.4c-2.7,-1.4 -
5.9,-0.5 -7.3,2.2L544.8,53c-23.1,-10.3 -49,-16.1 -76.4,-16c-27.3,0 -
53.2,5.7 -76.2,16L364.7,2.9c-1.4,-2.6 -4.7,-3.6 -7.3,-2.2c-2.6,1.4 -
3.6,4.7 -2.1,7.3127.2,49.6C329,85.2 292.9,137.7
292.9,1981351,0C644,137.7 607.9,85.3 554.5,57.7M388.6,134.4c-8.1,0 -
14.7,-6.6 -14.7,-14.7c0,-8.1 6.6,-14.7 14.7,-14.7c8.1,0 14.7,6.6
14.7,14.7C403.3,127.8 396.7,134.4 388.6,134.4M548.4,134.4c-8.1,0 -
14.7,-6.6 -14.7,-14.7c0,-8.1 6.6,-14.7 14.7,-14.7c8.1,0 14.7,6.7
14.7,14.8C563.1,127.8 556.5,134.4 548.4,134.4M294.9,21110.1,252.9c0,23
18.6,41.6 41.6,41.6128.3,010,86.3c0,21.5 17.5,39 38.9,39c21.5,0 39,-
17.5 39,-3910,-86.3152.6,010,86.3c0,21.5 17.5,39 39,39c21.5,0 39,-17.5
39,-3910,-86.3128.4,0c22.9,0 41.6,-18.6 41.6,-41.610,-
252.9L294.9,211zM669.7,269.7c15.2,15.2 39.9,15.2 55.1,01115.4,-
115.4c15.3,-15.2 15.2,-39.9 0,-55.1C825,84 800.4,84
785.1,99.2L669.7,214.6C654.5,229.8 654.5,254.5 669.7,269.7"/>
 <path android:fillColor="#FF000000" android:pathData="M23.7,804.8c-
7.4,-4.2 -13.2,-10.1 -17.4,-17.5C2.1,779.8 0,771.6 0,762.6c0,-9 2.1,-
17.2 6.3,-24.7c4.2,-7.5 10,-13.3 17.4,-17.5c7.4,-4.2 15.5,-6.4 24.4,-
6.4c7,0 13.4,1.3 19.2,4c5.8,2.6 10.7,6.4 14.9,11.21-7.8,7.7c-3.5,-4.2 -
7.3,-7.4 -11.6,-9.4c-4.3,-2 -9.2,-3 -14.6,-3c-6.7,0 -12.8,1.6 -
18.5,4.7c-5.6,3.2 -10.1,7.6 -13.5,13.4c-3.3,5.8 -5,12.4 -5,20s1.7,14.2
5,20c3.3,5.8 7.8,10.2 13.5,13.4c5.6,3.2 11.8,4.7 18.5,4.7c11.4,0 21,-
4.6 28.9,-13.917.9,7.8c-4.2,5.1 -9.5,9.2 -15.9,12.2c-6.3,3 -13.3,4.5 -
20.9,4.5C39.2,811.2 31,809.1 23.7,804.8z"/>
 <path android:fillColor="#FF000000" android:pathData="M108,806.7c-
5,-3 -9,-7.1 -11.8,-12.3c-2.8,-5.2 -4.2,-11.1 -4.2,-17.5c0,-6.4 1.4,-
12.2 4.2,-17.5c2.8,-5.2 6.7,-9.4 11.8,-12.4c5,-3 10.7,-4.5 17,-
4.5c6.3,0 12,1.5 17.1,4.5c5.1,3 9,7.1 11.8,12.4c2.8,5.2 4.2,11.1
4.2,17.5c0,6.4 -1.4,12.2 -4.2,17.5c-2.8,5.2 -6.8,9.4 -11.8,12.3c-5.1,3
-10.8,4.5 -17.1,4.5C118.7,811.2 113.1,809.7 108,806.7zM136,798.2c3.4,-2
6.1,-4.8 8.1,-8.5c2,-3.7 3.1,-8 3.1,-12.8c0,-4.9 -1,-9.1 -3.1,-12.8c-
2,-3.7 -4.7,-6.5 -8.1,-8.5c-3.4,-2 -7,-3 -10.9,-3c-3.8,0 -7.4,1 -
10.8,3c-3.4,2 -6.1,4.8 -8.1,8.5c-2,3.7 -3.1,8 -3.1,12.8c0,4.9 1,9.1
3.1,12.8c2,3.7 4.7,6.5 8.1,8.5c3.4,2 7,3 10.8,3C129,801.2 132.6,800.2
136,798.2z"/>
 <path android:fillColor="#FF000000"
 android:pathData="M169.4,744.6h10.5v9.6h0.5c1.8,-3.3 4.7,-6.1 8.6,-

```

```

8.3c3.9,-2.3 8.1,-3.4 12.5,-3.4c7.9,0 13.9,2.3 18,6.9c4.1,4.6 6.2,10.9
6.2,18.8v40.8h-10.9V770c0,-11.6 -5.4,-17.4 -16.2,-17.4c-3.5,0 -6.6,1 -
9.4,3c-2.8,2 -4.9,4.6 -6.4,7.8c-1.5,3.2 -2.3,6.6 -2.3,10.1v35.6h-
11V744.6z"/>
 <path android:fillColor="#FF000000" android:pathData="M250,833.7c-
4.9,-3.7 -8.3,-8.1 -10,-13110.3,-4.3c1.3,3.8 3.6,6.9 7,9.3c3.3,2.4
7.3,3.6 11.9,3.6c6.8,0 12.1,-2 15.8,-6c3.7,-4 5.5,-9.5 5.5,-16.7v-6.9h-
0.5c-2.1,3.3 -5.1,6 -9,8.2c-3.9,2.2 -8.3,3.2 -13.3,3.2c-5.5,0 -10.5,-
1.5 -15.2,-4.4c-4.7,-3 -8.4,-7 -11.2,-12.3c-2.8,-5.2 -4.2,-11.1 -4.2,-
17.6s1.4,-12.4 4.2,-17.6c2.8,-5.2 6.5,-9.3 11.2,-12.3c4.7,-3 9.8,-4.4
15.2,-4.4c4.9,0 9.4,1.1 13.3,3.2c3.9,2.2 6.9,4.9 9,8.2h0.5v-
9.4H301v61.9c0,7.1 -1.4,13.1 -4.2,18.1c-2.8,4.9 -6.6,8.6 -11.4,11c-
4.8,2.4 -10.2,3.6 -16.3,3.6c261.3,839.3 254.9,837.4
250,833.7zM279.9,798.2c3.2,-2 5.8,-4.8 7.7,-8.4c2,-3.7 2.9,-8 2.9,-
12.9c0,-5 -1,-9.3 -2.9,-13c-1.9,-3.7 -4.5,-6.5 -7.7,-8.4c-3.2,-1.9 -
6.7,-2.9 -10.5,-2.9c-3.8,0 -7.3,1 -10.5,2.9c-3.2,2 -5.8,4.8 -7.7,8.4c-
2,3.7 -2.9,8 -2.9,12.9c0,5 1,9.3 2.9,12.9c1.9,3.7 4.5,6.5 7.7,8.4c3.2,2
6.7,2.9 10.5,2.9c273.2,801.2 276.7,800.2 279.9,798.2z"/>
 <path android:fillColor="#FF000000"
android:pathData="M317.2,744.6h10.5v10.5h0.5c1.2,-3.6 3.7,-6.5 7.5,-
9c3.8,-2.4 7.5,-3.6 11.2,-3.6c2.7,0 5.2,0.4 7.5,1.3v11.8c-2.3,-1.3 -
5.4,-2 -9.1,-2c-3,0 -5.9,0.9 -8.5,2.7c-2.6,1.8 -4.7,4.1 -6.3,7.1c-1.6,3
-2.3,6.2 -2.3,9.6v36h-11.1v744.6z"/>
 <path android:fillColor="#FF000000"
android:pathData="M370.4,808.4c-3.8,-1.9 -6.7,-4.5 -8.8,-7.8c-2.1,-3.3
-3.2,-7.1 -3.2,-11.2c0,-6.8 2.6,-12.2 7.7,-16c5.2,-3.8 11.8,-5.7 19.8,-
5.7c4,0 7.6,0.4 10.8,1.3c3.2,0.9 5.8,1.8 7.7,2.9v-3.4c0,-3.1 -0.8,-5.9
-2.4,-8.4c-1.6,-2.5 -3.8,-4.4 -6.4,-5.7c-2.7,-1.3 -5.6,-2 -8.7,-2c-
7.4,0 -13.3 -16.9,91-9.2,-6.1c2.7,-4 6.2,-7.1 10.7,-9.3s9.4,-3.3 15,-
3.3c8.8,0 15.8,2.4 20.9,7.1c5.1,4.8 7.7,11.2 7.7,19.2v40.2h-10.5v-9.5h-
0.5c-2.1,3.3 -4.9,6 -8.4,8.3c-3.6,2.2 -7.7,3.3 -12.3,3.3c378.4,811.2
374.2,810.3 370.4,808.4zM393.9,798.7c3.1,-1.8 5.7,-4.3 7.6,-7.4c1.9,-
3.1 2.9,-6.5 2.9,-10.1c-4.9,-3.1 -10.4,-4.7 -16.8,-4.7c-5.5,0 -9.8,1.2
-13.1,3.7s-4.9,5.6 -4.9,9.4c0,3.5 1.4,6.3 4.3,8.5s6.2,3.3
10,3.3c387.5,801.4 390.8,800.5 393.9,798.7z"/>
 <path android:fillColor="#FF000000"
android:pathData="M446.4,809.9c-2.2,-0.9 -4.1,-2.1 -5.7,-3.6c-1.7,-1.7
-3,-3.6 -3.8,-5.9c-0.8,-2.3 -1.2,-4.9 -1.2,-8v-37.7h-11.4v-10h11.4v-
19.2h10.9v19.2h15.9v10h-15.9v35.1c0,3.7 0.7,6.5 2.2,8.2c1.5,2 3.7,2.9
6.8,2.9c2.7,0 5,-0.6 7,-1.8v10.7c-1.4,0.5 -2.7,0.9 -4,1.1c-1.3,0.2 -
2.9,0.3 -5,0.3c451,811.2 448.6,810.7 446.4,809.9z"/>
 <path android:fillColor="#FF000000"
android:pathData="M480.9,804.3c-4.1,-4.6 -6.2,-11 -6.2,-19.1v-
40.6h11v38.7c0,11.9 5.3,17.8 16,17.8c3.6,0 6.7,-1 9.6,-3c2.8,-2 5,-4.6
6.5,-7.8c1.5,-3.2 2.3,-6.6 2.3,-10.1v-35.6h531v64.5h-10.4v-9.6h-0.5c-
1.9,3.3 -4.8,6.1 -8.6,8.3c-3.9,2.3 -8,3.4 -12.4,3.4c491.1,811.2
485,808.9 480.9,804.3z"/>
 <path android:fillColor="#FF000000"
android:pathData="M547.3,716h11.1v93.1h-11.1v716z"/>
 <path android:fillColor="#FF000000"
android:pathData="M582.1,808.4c-3.8,-1.9 -6.7,-4.5 -8.8,-7.8c-2.1,-3.3
-3.2,-7.1 -3.2,-11.2c0,-6.8 2.6,-12.2 7.7,-16c5.2,-3.8 11.8,-5.7 19.8,-
5.7c4,0 7.6,0.4 10.8,1.3c3.2,0.9 5.8,1.8 7.7,2.9v-3.4c0,-3.1 -0.8,-5.9
-2.4,-8.4c-1.6,-2.5 -3.8,-4.4 -6.4,-5.7c-2.7,-1.3 -5.6,-2 -8.7,-2c-

```

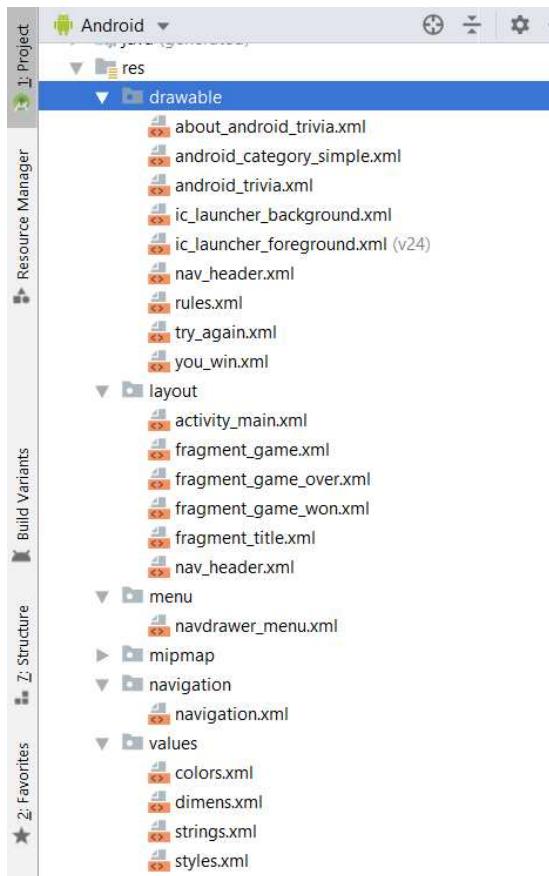
```

 7.4,0 -13,3 -16.9,91-9.2,-6.1c2.7,-4 6.2,-7.1 10.7,-9.3s9.4,-3.3 15,-
 3.3c8.8,0 15.8,2.4 20.9,7.1c5.1,4.8 7.7,11.2 7.7,19.2v40.2h-10.5v-9.5h-
 0.5c-2.1,3.3 -4.9,6 -8.5,8.3c-3.6,2.2 -7.7,3.3 -12.3,3.3C590.1,811.2
 585.8,810.3 582.1,808.4zM605.5,798.7c3.1,-1.8 5.7,-4.3 7.6,-7.4c2,-3.1
 2.9,-6.5 2.9,-10.1c-4.9,-3.1 -10.4,-4.7 -16.8,-4.7c-5.5,0 -9.8,1.2 -
 13.1,3.7s-4.9,5.6 -4.9,9.4c0,3.5 1.4,6.3 4.3,8.5c2.9,2.2 6.2,3.3
 10,3.3C599.1,801.4 602.4,800.5 605.5,798.7z"/>
 <path android:fillColor="#FF000000" android:pathData="M658,809.9c-
 2.2,-0.9 -4.1,-2.1 -5.7,-3.6c-1.7,-1.7 -3,-3.6 -3.8,-5.9c-0.8,-2.3 -
 1.2,-4.9 -1.2,-8v-37.7h-11.4v-10h11.4v-19.2h10.9v19.2H674v10h-
 15.9v35.1c0,3.7 0.7,6.5 2.2,8.2c1.5,2 3.7,2.9 6.8,2.9c2.7,0 5,-0.6 7,-
 1.8v10.7c-1.4,0.5 -2.7,0.9 -4,1.1c-1.3,0.2 -2.9,0.3 -5,0.3C662.6,811.2
 660.2,810.7 658,809.9z"/>
 <path android:fillColor="#FF000000"
 android:pathData="M688.5,728.5c-1.6,-1.6 -2.3,-3.4 -2.3,-5.6c0,-2.2
 0.8,-4 2.3,-5.6c1.6,-1.6 3.4,-2.3 5.6,-2.3c2.2,0 4,0.8 5.6,2.3c1.6,1.6
 2.3,3.4 2.3,5.6c0,2.2 -0.8,4 -2.3,5.6c-1.5,1.6 -3.4,2.3 -
 5.7,2.3C691.9,730.8 690,730.1 688.5,728.5zM688.6,744.6h10.9v64.5h-
 10.9V744.6z"/>
 <path android:fillColor="#FF000000"
 android:pathData="M728.4,806.7c-5,-3 -9,-7.1 -11.8,-12.3c-2.8,-5.2 -
 4.2,-11.1 -4.2,-17.5c0,-6.4 1.4,-12.2 4.2,-17.5c2.8,-5.2 6.7,-9.4
 11.8,-12.4c5,-3 10.7,-4.5 17,-4.5c6.3,0 12,1.5 17.1,4.5s9,7.1
 11.8,12.4c2.8,5.2 4.2,11.1 4.2,17.5c0,6.4 -1.4,12.2 -4.2,17.5c-2.8,5.2
 -6.8,9.4 -11.8,12.3c-5.1,3 -10.8,4.5 -17.1,4.5C739.1,811.2 733.4,809.7
 728.4,806.7zM756.3,798.2c3.4,-2 6.1,-4.8 8.1,-8.5c2,-3.7 3.1,-8 3.1,-
 12.8c0,-4.9 -1,-9.1 -3.1,-12.8c-2,-3.7 -4.7,-6.5 -8.1,-8.5c-3.4,-2 -7,-
 3 -10.9,-3c-3.8,0 -7.4,1 -10.8,3c-3.4,2 -6.1,4.8 -8.1,8.5c-2,3.7 -3.1,8
 -3.1,12.8c0,4.9 1,9.1 3.1,12.8c2,3.7 4.7,6.5 8.1,8.5c3.4,2 7,3
 10.8,3C749.3,801.2 753,800.2 756.3,798.2z"/>
 <path android:fillColor="#FF000000"
 android:pathData="M789.7,744.6h10.5v9.6h0.5c1.8,-3.3 4.7,-6.1 8.6,-
 8.3c3.9,-2.3 8.1,-3.4 12.5,-3.4c7.9,0 13.9,2.3 18,6.9c4.1,4.6 6.2,10.9
 6.2,18.8v40.8h-10.9V770c0,-11.6 -5.4,-17.4 -16.2,-17.4c-3.5,0 -6.6,1 -
 9.4,3c-2.8,2 -4.9,4.6 -6.4,7.8c-1.5,3.2 -2.3,6.6 -2.3,10.1v35.6h-
 11.1V744.6z"/>
 <path android:fillColor="#FF000000" android:pathData="M866.4,806c-
 4.8,-3.5 -8.1,-7.8 -10,-12.919.8,-4.3c1.7,4.1 4.2,7.2 7.3,9.4c3.2,2.2
 6.6,3.2 10.3,3.2c4.2,0 7.6,-0.9 10.3,-2.7c2.7,-1.8 4.1,-4 4.1,-6.6c0,-
 2.5 -1,-4.6 -3.1,-6.2c-2,-1.6 -5.5,-3.1 -10.5,-4.31-8.8,-2.2c-4.9,-1.1
 -9,-3.2 -12.4,-6.1c-3.4,-2.9 -5.1,-6.8 -5.1,-11.7c0,-4 1.1,-7.4 3.4,-
 10.3c2.3,-2.9 5.3,-5 9.2,-6.5c3.8,-1.5 8,-2.2 12.5,-2.2c5.5,0 10.6,1.3
 15.1,3.8c4.5,2.6 7.8,6.2 9.7,10.91-9.8,4.2c-1.4,-3.1 -3.5,-5.4 -6.3,-
 6.9c-2.8,-1.5 -6,-2.2 -9.4,-2.2c-3.4,0 -6.4,0.8 -9.2,2.5c-2.7,1.6 -
 4.1,3.8 -4.1,6.4c0,2.3 0.9,4 2.7,5.3c1.8,1.3 4.6,2.5
 8.5,3.519.6,2.3c6.4,1.6 11.2,4.1 14.4,7.3c3.2,3.2 4.7,7.1
 4.7,11.6c0,3.7 -1.1,7.1 -3.2,10.1c-2.2,3 -5.2,5.4 -9.1,7.2c-3.9,1.7 -
 8.3,2.6 -13.3,2.6C877,811.2 871.2,809.4 866.4,806z"/>
 <path android:fillColor="#FF000000"
 android:pathData="M922.7,806.7c-1.6,-1.6 -2.4,-3.6 -2.4,-5.9c0,-2.3
 0.8,-4.2 2.4,-5.8c1.6,-1.6 3.6,-2.4 5.9,-2.4c2.3,0 4.2,0.8
 5.8,2.4c1.6,1.6 2.4,3.5 2.4,5.8c0,2.3 -0.8,4.3 -2.4,5.9c-1.6,1.6 -
 3.5,2.4 -5.8,2.4C926.2,809.1 924.3,808.3

```

```
922.7,806.7zM923.1,747.6v716H934v31.6l-1.3,34.1h-8.5L923.1,747.6z" />
</vector>
```

19. Sekarang kita punya struktur file sebagai berikut



20. Kemudian, buat file kotlin untuk fragment.

#### GameFragment.kt

```
package akakom.nomhs.kotlin.belajarnavigasi2

import akakom.nomhs.kotlin.belajarnavigasi2.databinding.FragmentGameBinding
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.databinding.DataBindingUtil
import androidx.fragment.app.Fragment
import androidx.navigation.findNavController

class GameFragment : Fragment() {
 override fun onCreateView(inflater:
 LayoutInflater, container: ViewGroup?,
 savedInstanceState: Bundle?): View? {
```

```

 val binding = DataBindingUtil.inflate<FragmentGameBinding>(
 inflater, R.layout.fragment_game, container, false)
 binding.submitButton.setOnClickListener { view: View ->
 val checkedId =
 binding.questionRadioGroup.checkedRadioButtonId
 if (-1 != checkedId) {
 var answerIndex = 0
 when (checkedId) {
 R.id.firstAnswerRadioButton -> answerIndex = 0
 R.id.secondAnswerRadioButton -> answerIndex = 1
 R.id.thirdAnswerRadioButton -> answerIndex = 2
 R.id.fourthAnswerRadioButton -> answerIndex = 3
 }
 if (answerIndex == 1) {
 view.findNavController()
 .navigate(R.id.action_gameFragment_to_gameWonFragment)
 } else {
 view.findNavController()
 .navigate(R.id.action_gameFragment_to_gameOverFragment)
 }
 }
 return binding.root
 }
}

```

### GameOverFragment.kt

```

package akakom.nomhs.kotlin.belajarnavigasi2

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.databinding.DataBindingUtil
import androidx.fragment.app.Fragment
import androidx.navigation.findNavController
import akakom.nomhs.kotlin.belajarnavigasi2
 .databinding.FragmentGameOverBinding

class GameOverFragment : Fragment() {
 override fun onCreateView(inflater: LayoutInflater,
 container: ViewGroup?,
 savedInstanceState: Bundle?): View? {
 val binding: FragmentGameOverBinding = DataBindingUtil.inflate(
 inflater, R.layout.fragment_game_over, container, false)
 binding.tryAgainButton.setOnClickListener { view: View ->
 view.findNavController()
 .navigate(R.id.action_gameOverFragment_to_gameFragment)
 }
 return binding.root
 }
}

```

```
 }
}
```

### GameWonFragment.kt

```
package akakom.nomhs.kotlin.belajarnavigasi2

import akakom.nomhs.kotlin.belajarnavigasi2
 .databinding.FragmentGameWonBinding
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.databinding.DataBindingUtil
import androidx.fragment.app.Fragment
import androidx.navigation.findNavController

class GameWonFragment : Fragment() {
 override fun onCreateView(inflater: LayoutInflater,
 container: ViewGroup?,
 savedInstanceState: Bundle?): View? {
 val binding: FragmentGameWonBinding = DataBindingUtil.inflate(
 inflater, R.layout.fragment_game_won, container, false)
 binding.nextMatchButton.setOnClickListener { view: View ->
 view.findNavController().
 navigate(R.id.action_gameWonFragment_to_titleFragment)
 }
 return binding.root
 }
}
```

### TitleFragment.kt

```
package akakom.nomhs.kotlin.belajarnavigasi2

import android.os.Bundle
import android.view.*
import androidx.fragment.app.Fragment
import androidx.databinding.DataBindingUtil
import androidx.navigation.findNavController
import akakom.nomhs.kotlin.belajarnavigasi2
 .databinding.FragmentTitleBinding

class TitleFragment : Fragment() {
 override fun onCreateView(inflater: LayoutInflater,
 container: ViewGroup?,
 savedInstanceState: Bundle?): View? {
 val binding = DataBindingUtil.
 inflate<FragmentTitleBinding>(inflater,
 R.layout.fragment_title, container, false)
 binding.playButton.setOnClickListener { view : View ->
 view.findNavController()
```

```

 .navigate(R.id.action_titleFragment_to_gameFragment)
 }
 return binding.root
}
}

```

21. Kemudian, buka file **navigation**, buat sehingga tampilan menjadi seperti langkah 1. Cara memasukkan fragment dan menghubungkan action, lihat pada dasar teori. Koding akhirnya adalah sebagai berikut.

```

<?xml version="1.0" encoding="utf-8"?>
<navigation 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:id="@+id/navigation"
 app:startDestination="@+id/titleFragment">

 <fragment
 android:id="@+id/titleFragment"
 android:name=
 "akakom.nomhs.kotlin.belajarnavigasi2.TitleFragment"
 android:label="Intro"
 tools:layout="@layout/fragment_title">
 <action
 android:id="@+id/action_titleFragment_to_gameFragment"
 app:destination="@+id/gameFragment" />
 </fragment>
 <fragment
 android:id="@+id/gameFragment"
 android:name=
 "akakom.nomhs.kotlin.belajarnavigasi2.GameFragment"
 android:label="Game"
 tools:layout="@layout/fragment_game">
 <action
 android:id="@+id/action_gameFragment_to_gameOverFragment"
 app:destination="@+id/gameOverFragment"
 app:popUpTo="@+id/gameFragment"
 app:popUpToInclusive="true" />
 <action
 android:id="@+id/action_gameFragment_to_gameWonFragment"
 app:destination="@+id/gameWonFragment"
 app:popUpTo="@+id/gameFragment"
 app:popUpToInclusive="true" />
 </fragment>
 <fragment
 android:id="@+id/gameOverFragment"
 android:name=
 "akakom.nomhs.kotlin.belajarnavigasi2.GameOverFragment"
 android:label="Game Failed"
 tools:layout="@layout/fragment_game_over">
 <action
 android:id="@+id/action_gameOverFragment_to_gameFragment"
 app:destination="@+id/gameFragment"
 app:popUpTo="@+id/titleFragment"

```

```

 app:popUpToInclusive="false" />
 </fragment>
<fragment
 android:id="@+id/gameWonFragment"
 android:name=
 "akakom.nomhs.kotlin.belajarnavigasi2.GameWonFragment"
 android:label="Game Won"
 tools:layout="@layout/fragment_game_won">
 <action
 android:id="@+id/action_gameWonFragment_to_titleFragment"
 app:destination="@+id/titleFragment"
 app:popUpTo="@+id/titleFragment"
 app:popUpToInclusive="false" />
 </fragment>
</navigation>
```

22. Ubahlah **MainActivity.kt** menjadi sebagai berikut.

```

package akakom.nomhs.kotlin.belajarnavigasi2

import akakom.nomhs.kotlin.belajarnavigasi2.databinding.ActivityMainBinding
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.databinding.DataBindingUtil
import androidx.drawerlayout.widget.DrawerLayout
import androidx.navigation.findNavController
import androidx.navigation.ui.NavigationUI

class MainActivity : AppCompatActivity() {
 private lateinit var drawerLayout: DrawerLayout
 override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)

 val binding = DataBindingUtil
 .setContentView<ActivityMainBinding>(this,
 R.layout.activity_main)
 drawerLayout = binding.drawerLayout

 val navController =
 this.findNavController(R.id.myNavHostFragment)
 NavigationUI.setupActionBarWithNavController
 (this, navController, drawerLayout)
 NavigationUI.setupWithNavController(binding
 .navView, navController)
 }

 override fun onSupportNavigateUp(): Boolean {
 val navController = this.findNavController
 (R.id.myNavHostFragment)
 return NavigationUI.navigateUp(navController, drawerLayout)
 }
}
```

```
}
```

23. Jalankan dan amati hasilnya.



### LATIHAN

---

1. Modifikasilah aplikasi dengan menambahkan detil data pemilih nomor telepon.



### TUGAS

---

1. Buat aplikasi baru dengan mengembangkan project diatas



### REFERENSI

---

1. <https://kotlinlang.org/docs/reference/>
2. <https://developer.android.com/kotlin>
3. <https://developer.android.com/courses/kotlin-android-fundamentals/toc>
4. <https://codelabs.developers.google.com/android-kotlin-fundamentals/>
5. <https://developer.android.com/kotlin/learn>
6. <https://developer.android.com/kotlin/resources>

## **MODUL 8**

### **Fragment**



#### **CAPAIAN PEMBELAJARAN**

---

1. Mahasiswa dapat membuat aplikasi dengan menggunakan fragment.



#### **KEBUTUHAN ALAT/BAHAN/SOFTWARE**

---

1. Android Studio 3.4.
2. Handphone Android versi 7.0 (Nougat)
3. Kabel data USB.
4. Driver ADB.



#### **DASAR TEORI**

---

#### **FRAGMENT**

Fragment mewakili perilaku atau bagian dari antarmuka pengguna dalam `FragmentActivity`. Kita bisa mengombinasikan beberapa fragmen dalam satu aktivitas untuk membangun UI multipanel dan menggunakan kembali sebuah fragmen dalam beberapa aktivitas. Kita bisa menganggap fragmen sebagai bagian modular dari aktivitas, yang memiliki daur hidup sendiri, menerima kejadian masukan sendiri, dan yang bisa kita tambahkan atau hapus saat aktivitas berjalan (semacam "subaktivitas" yang bisa digunakan kembali dalam aktivitas berbeda).

Fragmen harus selalu tersemat dalam aktivitas dan daur hidup fragmen secara langsung dipengaruhi oleh daur hidup aktivitas host-nya. Misalnya, saat aktivitas dihentikan sementara, semua fragmen di dalamnya juga dihentikan sementara, dan bila aktivitas dimusnahkan, semua fragmen juga demikian. Akan tetapi, saat aktivitas berjalan (dalam status daur hidup *dilanjutkan*), Kita bisa memanipulasi setiap fragmen secara terpisah, seperti menambah atau membuangnya. Saat melakukan transaksi fragmen, Kita juga bisa menambahkannya ke back-stack yang dikelola oleh aktivitas—setiap entri back-stack merupakan catatan transaksi fragmen yang terjadi. Dengan back-stack pengguna dapat membalikkan transaksi fragmen (mengarah mundur), dengan menekan tombol *Kembali*.

Bila kita menambahkan fragmen sebagai bagian dari layout aktivitas, fragmen tersebut berada di `ViewGroup` di dalam hierarki tampilan aktivitas dan fragmen menentukan layout tampilannya sendiri. Kita bisa menyisipkan fragmen ke dalam layout aktivitas dengan mendeklarasikan fragmen dalam file layout aktivitas, sebagai elemen `<fragment>`, atau dari kode aplikasi kita dengan menambahkannya ke  `ViewGroup` yang ada.

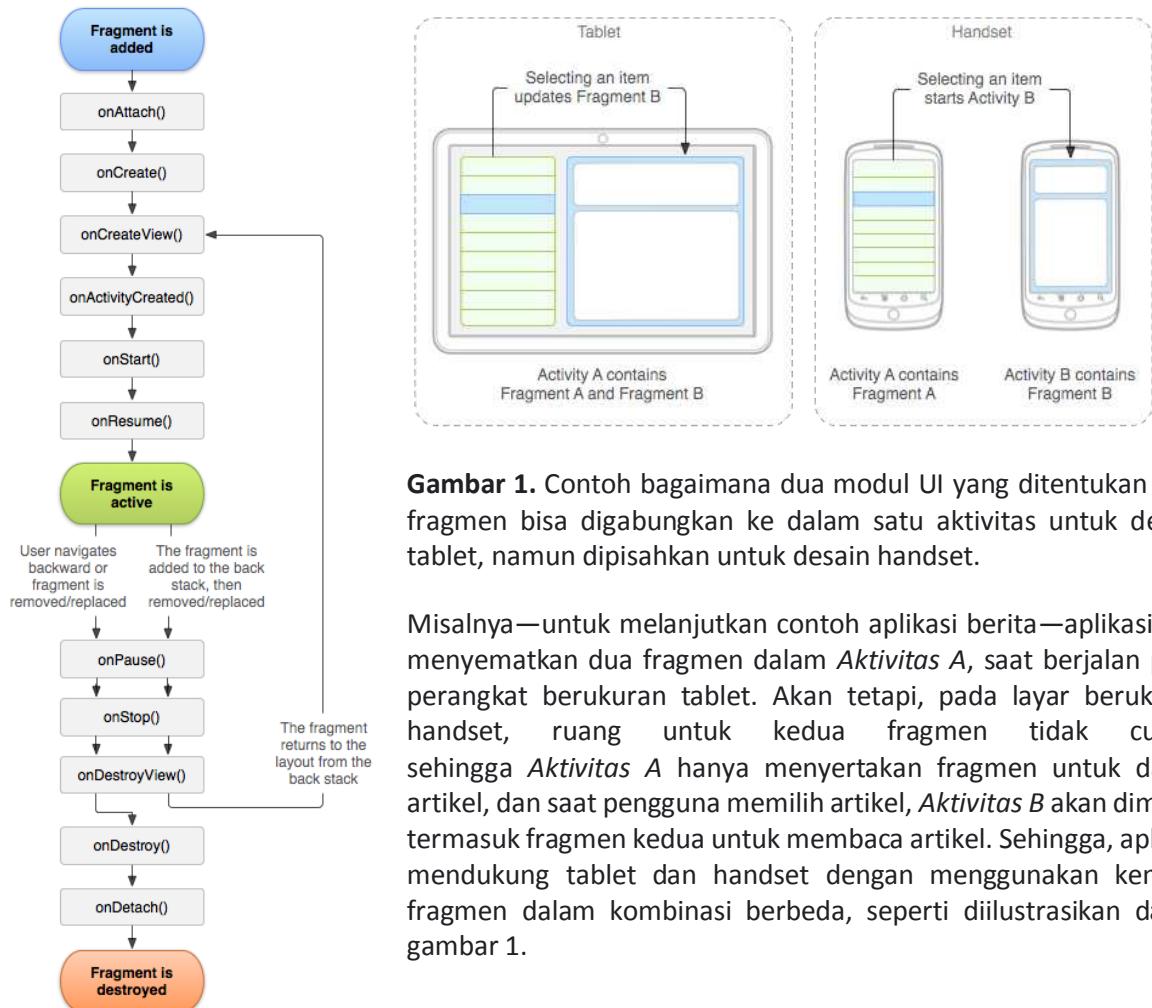
Kita akan membahas cara membuat aplikasi menggunakan fragmen, termasuk cara fragmen mempertahankan statusnya bila ditambahkan ke back-stack aktivitas, berbagi kejadian dengan aktivitas, dan fragmen lain dalam aktivitas, berkontribusi pada bilah aksi aktivitas, dan lainnya.

## Filosofi Desain

Android memperkenalkan fragmen di Android 3.0 (API level 11), terutama untuk mendukung desain UI yang lebih dinamis dan fleksibel pada layar besar, seperti tablet. Karena layar tablet jauh lebih besar daripada layar handset, maka lebih banyak ruang untuk mengombinasikan dan bertukar komponen UI. Fragmen memungkinkan desain seperti itu tanpa perlu mengelola perubahan kompleks pada hierarki tampilan. Dengan membagi layout aktivitas menjadi beberapa fragmen, kita bisa mengubah penampilan aktivitas saat waktu proses dan mempertahankan perubahan itu di back-stack yang dikelola oleh aktivitas. Mode-mode tersebut kini tersedia secara luas melalui library dukungan fragmen.

Misalnya, aplikasi berita bisa menggunakan satu fragmen untuk menampilkan daftar artikel di sebelah kiri dan fragmen lainnya untuk menampilkan artikel di sebelah kanan—kedua fragmen ini muncul di satu aktivitas, berdampingan, dan masing-masing fragmen memiliki serangkaian metode callback daur hidup dan menangani kejadian masukan pengguna sendiri. Sehingga, sebagai ganti menggunakan satu aktivitas untuk memilih artikel dan aktivitas lainnya untuk membaca artikel, pengguna bisa memilih artikel dan membaca semuanya dalam aktivitas yang sama, sebagaimana diilustrasikan dalam layout tablet pada gambar 1.

Kita harus mendesain masing-masing fragmen sebagai komponen aktivitas modular dan bisa digunakan kembali. Yakni, karena setiap fragmen mendefinisikan layoutnya dan perilakunya dengan callback daur hidupnya sendiri, kita bisa memasukkan satu fragmen dalam banyak aktivitas, sehingga kita harus mendesainnya untuk digunakan kembali dan mencegah memanipulasi satu fragmen dari fragmen lain secara langsung. Ini terutama penting karena dengan fragmen modular kita bisa mengubah kombinasi fragmen untuk ukuran layar yang berbeda. Saat mendesain aplikasi untuk mendukung tablet maupun handset, kita bisa menggunakan kembali fragmen dalam konfigurasi layout yang berbeda untuk mengoptimalkan pengalaman pengguna berdasarkan ruang layar yang tersedia. Misalnya, pada handset, fragmen mungkin perlu dipisahkan untuk menyediakan UI panel tunggal bila lebih dari satu yang tidak cocok dalam aktivitas yang sama.



**Gambar 1.** Contoh bagaimana dua modul UI yang ditentukan oleh fragmen bisa digabungkan ke dalam satu aktivitas untuk desain tablet, namun dipisahkan untuk desain handset.

Misalnya—untuk melanjutkan contoh aplikasi berita—aplikasi bisa menyematkan dua fragmen dalam *Aktivitas A*, saat berjalan pada perangkat berukuran tablet. Akan tetapi, pada layar berukuran handset, ruang untuk kedua fragmen tidak cukup, sehingga *Aktivitas A* hanya menyertakan fragmen untuk daftar artikel, dan saat pengguna memilih artikel, *Aktivitas B* akan dimulai, termasuk fragmen kedua untuk membaca artikel. Sehingga, aplikasi mendukung tablet dan handset dengan menggunakan kembali fragmen dalam kombinasi berbeda, seperti diilustrasikan dalam gambar 1.

**Gambar 2.** Daur hidup fragmen (saat aktivitasnya berjalan).

## Membuat Fragmen

Untuk membuat fragmen, kita harus membuat subclass `Fragment` (atau subclass-nya yang ada). Class `Fragment` memiliki kode yang mirip seperti `Activity`. Class ini memiliki metode callback yang serupa dengan aktivitas, seperti `onCreate()`, `onStart()`, `onPause()`, dan `onStop()`. Sebenarnya, jika kita mengonversi aplikasi Android saat ini untuk menggunakan fragmen, kita mungkin cukup memindahkan kode dari metode callback aktivitas ke masing-masing metode callback fragmen.

Biasanya, kita harus mengimplementasikan setidaknya metode daur hidup berikut ini:

`onCreate()` : Sistem akan memanggilnya saat membuat fragmen. Dalam implementasi, kita harus melakukan inisialisasi komponen penting dari fragmen yang ingin dipertahankan saat fragmen dihentikan sementara atau dihentikan, kemudian dilanjutkan.

`onCreateView()` : Sistem akan memanggilnya saat fragmen menggambar antarmuka penggunanya untuk yang pertama kali. Untuk menggambar UI fragmen, kita harus mengembalikan View dari metode ini yang menjadi root layout fragmen. Hasil yang dikembalikan bisa berupa null jika fragmen tidak menyediakan UI.

`onPause()` : Sistem akan memanggil metode ini sebagai indikasi pertama bahwa pengguna sedang meninggalkan fragmen kita (walau itu tidak selalu berarti fragmen sedang dimusnahkan). Di sinilah biasanya kita harus mengikat perubahan yang harus dipertahankan di luar sesi pengguna saat ini (karena pengguna mungkin tidak akan kembali).

Kebanyakan aplikasi harus mengimplementasikan setidaknya tiga metode ini untuk setiap fragmen, tetapi ada beberapa metode callback lain yang juga harus kita gunakan untuk menangani berbagai tahap daur hidup fragmen. Perhatikan bahwa kode yang mengimplementasikan aksi daur hidup dari komponen dependen harus ditempatkan di komponen itu sendiri, bukan dalam implementasi callback fragmen.

## Menambahkan antarmuka pengguna

Fragmen biasanya digunakan sebagai bagian dari antarmuka pengguna aktivitas dan menyumbangkan layoutnya sendiri ke aktivitas. Untuk menyediakan layout fragmen, kita harus mengimplementasikan metode callback `onCreateView()`, yang dipanggil sistem Android bila tiba saatnya fragmen menggambar layoutnya. Implementasi kita atas metode ini harus mengembalikan View yang menjadi root layout fragmen. Untuk mengembalikan layout dari `onCreateView()`, Kita bisa memekarkannya dari resource layout yang ditentukan di XML. Untuk membantu melakukannya, `onCreateView()` menyediakan objek `LayoutInflater`.

Misalnya, terdapat subclass `Fragment` yang memuat layout dari file `example_fragment.xml`:

```
class ExampleFragment : Fragment() {

 override fun onCreateView(
 inflater: LayoutInflater,
 container: ViewGroup?,
 savedInstanceState: Bundle?
): View {
 // Inflate the layout for this fragment
 return inflater.inflate(R.layout.example_fragment, container, false)
 }
}
```

Parameter `container` yang diteruskan ke `onCreateView()` adalah induk `ViewGroup` (dari layout aktivitas) tempat layout fragmen akan disisipkan. Parameter `savedInstanceState` adalah `Bundle` yang menyediakan data tentang instance fragmen sebelumnya, jika fragmen dilanjutkan.

Metode `inflate()` mengambil tiga argumen:

1. ID sumber daya layout yang ingin dimekarkan.
2. ViewGroup akan menjadi induk dari layout yang dimekarkan. container perlu diteruskan agar sistem menerapkan parameter layout ke tampilan akar layout yang dimekarkan, yang ditetapkan dalam tampilan induk yang akan dituju.
3. Boolean yang menunjukkan apakah layout yang dimekarkan harus dilampirkan ke ViewGroup (parameter kedua) selama pemekaran. (Dalam hal ini, ini salah karena sistem sudah memasukkan layout yang dimekarkan ke dalam container—meneruskan true akan membuat tampilan grup berlebih dalam layout akhir.) Kita kini telah melihat cara membuat fragmen yang menyediakan layout. Berikutnya, kita perlu menambahkan fragmen ke aktivitas.

### Menambahkan fragmen ke aktivitas

Biasanya, fragmen berkontribusi pada sebagian UI ke aktivitas host, yang disematkan sebagai bagian dari hierarki tampilan keseluruhan aktivitas. Ada dua cara untuk menambahkan fragmen ke layout aktivitas:

#### Deklarasikan fragmen dalam file layout aktivitas.

Dalam hal ini, Kita bisa menetapkan properti layout fragmen seakan-akan sebuah tampilan. Misalnya, berikut ini adalah file layout untuk aktivitas dengan dua fragmen:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:orientation="horizontal"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 <fragment android:name="com.example.news.ArticleListFragment"
 android:id="@+id/list"
 android:layout_weight="1"
 android:layout_width="0dp"
 android:layout_height="match_parent" />
 <fragment android:name="com.example.news.ArticleReaderFragment"
 android:id="@+id/viewer"
 android:layout_weight="2"
 android:layout_width="0dp"
 android:layout_height="match_parent" />
</LinearLayout>
```

Atribut android:name dalam <fragment> menetapkan class Fragment untuk dibuat instance-nya dalam layout.

Saat sistem membuat layout aktivitas, sistem membuat instance setiap fragmen sebagaimana yang ditetapkan dalam layout dan memanggil metode onCreateView() masing-masing, untuk mengambil setiap fragmen. Sistem akan menyisipkan View yang dikembalikan oleh fragmen secara langsung, menggantikan elemen <fragment>. Atau, secara programatis tambahkan fragmen ke ViewGroup yang ada. Kapan saja saat aktivitas berjalan, Kita bisa menambahkan fragmen ke layout aktivitas. Kita cukup menetapkan ViewGroup di tempat memasukkan fragmen.

Untuk membuat transaksi fragmen dalam aktivitas (seperti menambah, membuang, atau mengganti fragmen), kita harus menggunakan API dari FragmentTransaction. Kita bisa mengambil instance FragmentTransaction dari FragmentActivity seperti ini:

```
val fragmentManager = supportFragmentManager
val fragmentTransaction = fragmentManager.beginTransaction()
```

Selanjutnya kita bisa menambahkan fragmen menggunakan metode add(), dengan menetapkan fragmen yang akan ditambahkan dan tampilan tempat menyisipkannya. Sebagai contoh:

```
val fragment = ExampleFragment()
fragmentTransaction.add(R.id.fragment_container, fragment)
fragmentTransaction.commit()
```

Argumen pertama yang diteruskan ke add() adalah ViewGroup tempat fragmen harus dimasukkan, yang ditetapkan oleh ID resource, dan parameter kedua merupakan fragmen yang akan ditambahkan. Setelah membuat perubahan dengan FragmentTransaction, Kita harus memanggil commit() untuk menerapkan perubahan.

## Mengelola Fragmen

Untuk mengelola fragmen dalam aktivitas, kita perlu menggunakan FragmentManager. Untuk mendapatkannya, panggil getSupportFragmentManager() dari aktivitas kita.

Beberapa hal yang dapat kita lakukan dengan FragmentManager antara lain:

1. Dapatkan fragmen yang ada di aktivitas dengan findFragmentById() (untuk fragmen yang menyediakan UI dalam layout aktivitas) atau findFragmentByTag() (untuk fragmen yang menyediakan atau tidak menyediakan UI).
2. Tarik fragmen dari back-stack, dengan popBackStack() (menyimulasikan perintah *Kembali* oleh pengguna).
3. Daftarkan listener untuk perubahan pada back-stack, dengan addOnBackStackChangedListener().

## Melakukan Transaksi Fragmen

Fitur menarik terkait penggunaan fragmen di aktivitas adalah kemampuan menambah, membuang, mengganti, dan melakukan tindakan lain dengannya, sebagai respons atas interaksi pengguna. Setiap set perubahan yang kita lakukan untuk aktivitas disebut transaksi dan kita bisa melakukan transaksi menggunakan API di FragmentTransaction. Kita juga bisa menyimpan setiap transaksi ke back-stack yang dikelola aktivitas, sehingga pengguna bisa mengarah mundur melalui perubahan fragmen (mirip mengarah mundur melalui aktivitas).

Kita bisa memperoleh instance FragmentTransaction dari FragmentManager seperti ini:

```
val fragmentManager = supportFragmentManager
val fragmentTransaction = fragmentManager.beginTransaction()
```

Setiap transaksi merupakan serangkaian perubahan yang ingin dilakukan pada waktu yang sama. Kita bisa menyiapkan semua perubahan yang ingin dilakukan untuk transaksi mana saja menggunakan metode seperti `add()`, `remove()`, dan `replace()`. Kemudian, untuk menerapkan transaksi pada aktivitas, kita harus memanggil `commit()`. Akan tetapi, sebelum memanggil `commit()`, kita mungkin perlu memanggil `addToBackStack()`, untuk menambahkan transaksi ke back-stack transaksi fragmen. Back-stack ini dikelola oleh aktivitas dan memungkinkan pengguna kembali ke status fragmen sebelumnya, dengan menekan tombol *Kembali*. Misalnya, dengan cara ini kita bisa mengganti satu fragmen dengan yang fragmen lain, dan mempertahankan status sebelumnya di back-stack:

```
val newFragment = ExampleFragment()
val transaction = supportFragmentManager.beginTransaction()
transaction.replace(R.id.fragment_container, newFragment)
transaction.addToBackStack(null)
transaction.commit()
```

Dalam contoh ini, `newFragment` menggantikan fragmen apa saja (jika ada) yang saat ini berada dalam kontainer layout yang diidentifikasi melalui ID `R.id.fragment_container`. Dengan memanggil `addToBackStack()`, transaksi yang diganti disimpan ke back-stack sehingga pengguna bisa membalikkan transaksi dan mengembalikan fragmen sebelumnya dengan menekan tombol *Kembali*. `FragmentActivity` lalu secara otomatis mengambil fragmen dari back-stack melalui `onBackPressed()`. Jika kita menambahkan beberapa perubahan pada transaksi—seperti `add()` atau `remove()`—dan memanggil `addToBackStack()`, maka semua perubahan yang diterapkan sebelum Kita memanggil `commit()` akan ditambahkan ke back-stack sebagai transaksi tunggal dan tombol *Kembali* akan membalikkannya bersama-sama.

Urutan menambahkan perubahan pada `FragmentTransaction` tidak berpengaruh, kecuali:

1. Kita harus memanggil `commit()` paling akhir.
2. Jika Kita menambahkan beberapa fragmen ke container yang sama, maka urutan penambahannya akan menentukan urutan munculnya dalam hierarki tampilan.

Jika kita tidak memanggil `addToBackStack()` saat melakukan transaksi yang membuang fragmen, maka fragmen itu akan dimusnahkan bila transaksi diikat dan pengguna tidak bisa mengarah kembali ke sana. Sedangkan, jika Kita memanggil `addToBackStack()` saat menghapus fragmen, maka fragmen itu akan *dihentikan* dan nanti dilanjutkan jika pengguna mengarah kembali.

Memanggil `commit()` tidak langsung menjalankan transaksi. Namun sebuah jadwal akan dibuat untuk dijalankan pada thread UI aktivitas (thread "utama") begitu thread bisa melakukannya. Akan tetapi, jika perlu Kita bisa memanggil `executePendingTransactions()` dari thread UI untuk segera mengeksekusi transaksi yang diserahkan oleh `commit()`. Hal itu biasanya tidak perlu kecuali jika transaksi merupakan dependensi bagi tugas dalam thread lain.

## Berkomunikasi dengan Aktivitas

Meskipun Fragment diimplementasikan sebagai objek yang tidak bergantung pada FragmentActivity dan bisa digunakan dalam banyak aktivitas, instance tertentu dari fragmen secara langsung terkait dengan aktivitas yang menjadi hostnya. Khususnya, fragmen bisa mengakses instance FragmentActivity dengan getActivity() dan dengan mudah melakukan tugas-tugas seperti mencari tampilan dalam layout aktivitas:

KOTLINJAVA

```
val listView: View? = activity?.findViewById(R.id.list)
```

Demikian pula, aktivitas Kita bisa memanggil metode di fragmen dengan mendapatkan referensi ke Fragment dari FragmentManager, menggunakan findFragmentById() atau findFragmentByTag(). Sebagai contoh:

```
val fragment =
supportFragmentManager.findFragmentById(R.id.example_fragment) as
ExampleFragment
```

## Membuat callback kejadian pada aktivitas

Dalam beberapa kasus, kita mungkin perlu fragmen untuk membagikan kejadian atau data dengan aktivitas dan/atau fragmen lain yang di-host oleh aktivitas. Untuk membagikan data, buat ViewModel bersama, seperti diuraikan dalam Membagikan data antar bagian fragmen di panduan ViewModel. Jika harus menyebarkan kejadian yang tidak dapat ditangani dengan ViewModel, Kita dapat mendefinisikan antarmuka callback di dalam fragment dan mengharuskan kejadian host menerapkannya. Saat aktivitas menerima callback melalui antarmuka, aktivitas akan bisa berbagi informasi itu dengan fragmen lain dalam layout jika perlu. Misalnya, jika sebuah aplikasi berita memiliki dua fragmen dalam aktivitas—satu untuk menampilkan daftar artikel (fragmen A) dan satu lagi untuk menampilkan artikel (fragmen B)—maka fragmen A harus memberi tahu aktivitas bila item daftar dipilih sehingga aktivitas bisa memberi tahu fragmen B untuk menampilkan artikel. Dalam hal ini, antarmuka OnArticleSelectedListener dideklarasikan di dalam fragmen A:

```
public class FragmentA : ListFragment() {
 ...
 // Container Activity must implement this interface
 interface OnArticleSelectedListener {
 fun onArticleSelected(articleUri: Uri)
 }
 ...
}
```

Selanjutnya aktivitas yang menjadi host fragmen akan mengimplementasikan antarmuka OnArticleSelectedListener dan menggantikan onArticleSelected() untuk memberi tahu fragmen B mengenai kejadian dari fragmen A. Untuk memastikan bahwa aktivitas host mengimplementasikan antarmuka ini, metode callback fragmen A onAttach() (yang dipanggil sistem saat menambahkan fragmen ke aktivitas) membuat instance OnArticleSelectedListener dengan membuat Activity yang diteruskan ke onAttach():

```
public class FragmentA : ListFragment() {

 var listener: OnArticleSelectedListener? = null
 ...
 override fun onAttach(context: Context) {
 super.onAttach(context)
 listener = context as? OnArticleSelectedListener
 if (listener == null) {
 throw ClassCastException("$context must implement
 OnArticleSelectedListener")
 }
 }
 ...
}
```

Jika aktivitas belum mengimplementasikan antarmuka, maka fragmen akan melontarkan ClassCastException. Jika berhasil, anggota mListener yang menyimpan referensi ke implementasi aktivitas OnArticleSelectedListener, sehingga fragmen A bisa berbagi kejadian dengan aktivitas, dengan memanggil metode yang didefinisikan oleh antarmuka OnArticleSelectedListener. Misalnya, jika fragmen A adalah ekstensi dari ListFragment, maka setiap kali pengguna mengklik item daftar, sistem akan memanggil onListItemClick() di fragmen, yang selanjutnya memanggil onArticleSelected() untuk berbagi kejadian dengan aktivitas:

```
public class FragmentA : ListFragment() {

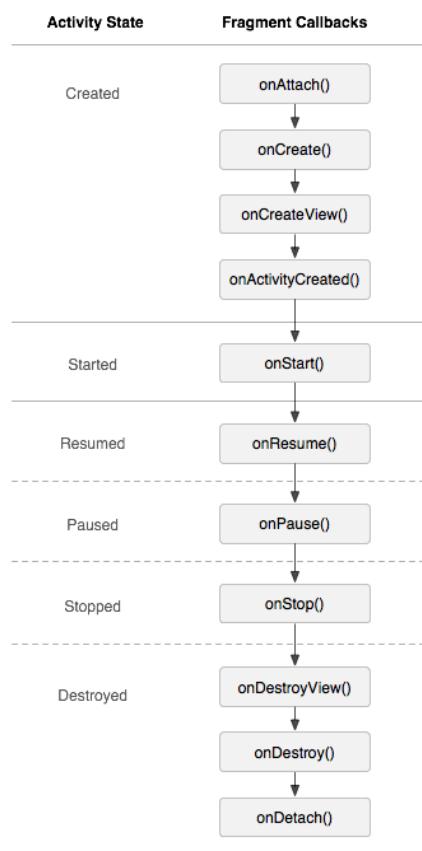
 var listener: OnArticleSelectedListener? = null
 ...
 override fun onListItemClick(l: ListView, v: View, position: Int,
 id: Long) {
 // Append the clicked item's row ID with the content provider Uri
 val noteUri: Uri =
 ContentUris.withAppendedId(ArticleColumns.CONTENT_URI, id)
 // Send the event and Uri to the host activity
 listener?.onArticleSelected(noteUri)
 }
 ...
}
```

Parameter id yang diteruskan ke onListItemClick() merupakan ID baris dari item yang diklik, yang digunakan aktivitas (atau fragmen lain) untuk mengambil artikel dari ContentProvider aplikasi.

### Menambahkan item ke Bilah Aplikasi

Fragmen kita bisa menyumbangkan item menu ke Menu Opsi aktivitas (dan, konsekuensinya, bilah aplikasi) dengan mengimplementasikan `onCreateOptionsMenu()`. Agar metode ini bisa menerima panggilan, Kita harus memanggil `setHasOptionsMenu()` selama `onCreate()`, untuk menunjukkan bahwa fragmen ingin menambahkan item ke Menu Opsi. Jika tidak, fragmen tidak menerima panggilan ke `onCreateOptionsMenu()`. Setiap item yang selanjutnya Kita tambahkan ke Menu Opsi dari fragmen akan ditambahkan ke item menu yang ada. Fragmen juga menerima callback ke `onOptionsItemSelected()` bila item menu dipilih. Kita juga bisa mendaftarkan tampilan dalam layout fragmen untuk menyediakan menu konteks dengan memanggil `registerForContextMenu()`. Bila pengguna membuka menu konteks, fragmen akan menerima panggilan ke `onCreateContextMenu()`. Bila pengguna memilih item, fragmen akan menerima panggilan ke `onContextItemSelected()`.

### Menangani Daur Hidup Fragmen



**Gambar 3.** Efek daur hidup aktivitas pada daur hidup fragmen.

Mengelola daur hidup fragmen mirip sekali dengan mengelola daur hidup aktivitas. Seperti aktivitas, fragmen bisa berada dalam tiga status:

**Dilanjutkan**

Fragmen terlihat dalam aktivitas yang berjalan.

**Dihentikan sementara**

Aktivitas lain berada di latar depan dan memiliki fokus, namun aktivitas tempat fragmen berada masih terlihat (aktivitas latar depan sebagian terlihat atau tidak menutupi seluruh layar).

#### Dihentikan

Fragment tidak terlihat. Aktivitas host telah dihentikan atau fragmen telah dihapus dari aktivitas namun ditambahkan ke back-stack. Fragmen yang dihentikan masih hidup (semua status dan informasi anggota masih disimpan oleh sistem). Akan tetapi, fragmen tidak terlihat lagi oleh pengguna dan akan dimatikan jika aktivitas dimatikan.

Seperti halnya aktivitas, kita dapat mempertahankan status UI fragment di seluruh perubahan konfigurasi dan habisnya proses menggunakan kombinasi `onSaveInstanceState(Bundle)`, `ViewModel`, serta penyimpanan lokal persisten. Perbedaan paling signifikan dalam daur hidup antara aktivitas dan fragmen ada pada cara penyimpanannya dalam back-stack masing-masing. Aktivitas ditempatkan ke dalam back-stack aktivitas yang dikelola oleh sistem saat dihentikan, secara default (sehingga pengguna bisa mengarah kembali ke aktivitas dengan tombol *Kembali*). Namun, fragmen ditempatkan ke dalam back-stack yang dikelola oleh aktivitas host hanya jika kita secara eksplisit meminta instance tersebut disimpan dengan memanggil `addToBackStack()` selama transaksi yang menghapus segmen tersebut. Jika tidak, pengelolaan daur hidup fragmen mirip sekali dengan mengelola daur hidup aktivitas; berlaku praktik yang sama.

#### Mongoordinasi dengan daur hidup aktivitas

Daur hidup aktivitas tempat fragmen berada akan memengaruhi secara langsung siklus hidup fragmen sedemikian rupa sehingga setiap callback daur hidup aktivitas menghasilkan callback yang sama untuk masing-masing fragmen. Misalnya, bila aktivitas menerima `onPause()`, maka masing-masing fragmen dalam aktivitas akan menerima `onPause()`. Namun fragmen memiliki beberapa callback daur hidup ekstra, yang menangani interaksi unik dengan aktivitas untuk melakukan tindakan seperti membangun dan memusnahkan UI fragmen. Metode callback tambahan ini adalah:

`onAttach()` Dipanggil bila fragmen telah dikaitkan dengan aktivitas (Activity diteruskan di sini).  
`onCreateView()` Dipanggil untuk membuat hierarki tampilan yang dikaitkan dengan fragmen.  
`onActivityCreated()` Dipanggil bila metode `onCreate()` aktivitas telah dikembalikan.  
`onDestroyView()` Dipanggil bila hierarki tampilan yang terkait dengan fragmen dihapus.  
`onDetach()` Dipanggil bila fragmen diputuskan dari aktivitas.

Alur daur hidup fragmen, karena dipengaruhi oleh aktivitas host-nya, diilustrasikan oleh gambar 3. Dalam gambar tersebut, kita bisa melihat bagaimana setiap status aktivitas yang berurutan menentukan metode callback mana yang mungkin diterima fragmen. Misalnya, saat aktivitas menerima callback `onCreate()`, fragmen dalam aktivitas akan menerima tidak lebih dari callback `onActivityCreated()`. Setelah status aktivitas diteruskan kembali, kita bisa bebas menambah dan membuang fragmen untuk aktivitas tersebut. Sehingga, hanya saat aktivitas berada dalam status dilanjutkan, daur hidup fragmen bisa berubah secara independen. Akan tetapi, saat aktivitas meninggalkan status dilanjutkan, fragmen akan kembali didorong melalui daur hidupnya oleh aktivitas.



## PRAKTIK

---

<https://androidwave.com/fragment-communication-using-viewmodel/>

1. Kita akan membuat aplikasi yang menggunakan fragment dan komunikasi data dengan konsep ViewModel.
2. Buat project baru.
3. Langkah pertama, cek pada dependencies

```
dependencies {

 implementation 'androidx.appcompat:appcompat:1.0.2'
 implementation 'com.google.android.material:material:1.0.0'

 implementation 'androidx.lifecycle:lifecycle-extensions:2.0.0'
 implementation 'androidx.legacy:legacy-support-v4:1.0.0'
 implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
}
```

4. Kemudian, buka pada activity\_mail.xml. Ubah kodennya sehingga menjadi seperti berikut

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
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">

 <com.google.android.material.appbar.AppBarLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:theme="@style/AppTheme">

 <com.google.android.material.tabs.TabLayout
 android:id="@+id/tabs"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:background="?attr/colorPrimary"
 app:tabTextColor="@android:color/background_light" />
 </com.google.android.material.appbar.AppBarLayout>

 <androidx.viewpager.widget.ViewPager
 android:id="@+id/view_pager"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 app:layout_behavior="@string/appbar_scrolling_view_behavior" />
</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

5. Buat sebuah class untuk view model

```

import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel

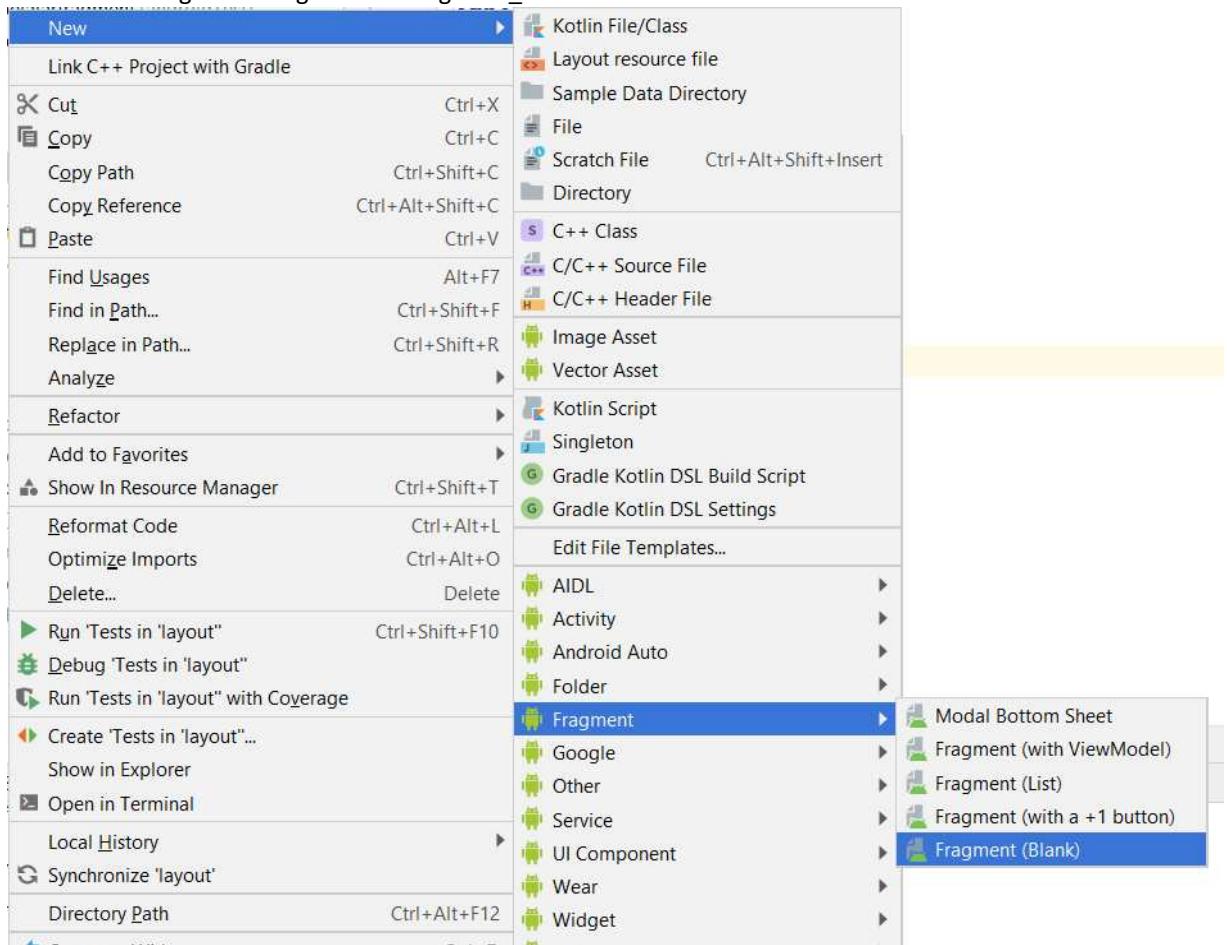
class CommunicationViewModel : ViewModel() {
 private val mName = MutableLiveData<String>()

 val name: LiveData<String>
 get() = mName

 fun setName(name: String) {
 mName.value = name
 }
}

```

6. Buat sebuah fragment dengan nama fragment\_first.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"
 android:layout_width="match_parent"
 android:layout_height="match_parent">

```

```

<ImageView
 android:id="@+id/imageView"
 android:layout_width="72dp"
 android:layout_height="72dp"
 android:layout_marginStart="8dp"
 android:layout_marginLeft="8dp"
 android:layout_marginTop="24dp"
 android:layout_marginEnd="8dp"
 android:layout_marginRight="8dp"
 android:src="@drawable/user_avatar"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent" />

<com.google.android.material.textfield.TextInputLayout
 android:id="@+id/textInputLayout"
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_marginStart="16dp"
 android:layout_marginLeft="16dp"
 android:layout_marginTop="32dp"
 android:layout_marginEnd="16dp"
 android:layout_marginRight="16dp"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/imageView">

 <com.google.android.material.textfield.TextInputEditText
 android:id="@+id/textInputTextName"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:hint="Masukkan Nama" />
</com.google.android.material.textfield.TextInputLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

- Buat juga kode program untuk FirstFragment.kt

```

import android.os.Bundle
import android.text.Editable
import android.text.TextWatcher
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.lifecycle.ViewModelProviders

import com.google.android.material.textfield.TextInputEditText

class FirstFrament : Fragment() {
 private var communicationViewModel: CommunicationViewModel? = null
```

```

override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)
 communicationViewModel =
 ViewModelProviders.of(requireActivity()).get(CommunicationViewModel::class.java)
}

override fun onCreateView(
 inflater: LayoutInflater, container: ViewGroup?,
 savedInstanceState: Bundle?
): View? {
 return inflater.inflate(R.layout.fragment_first,
 container, false)
}

override fun onViewCreated(view: View,
 savedInstanceState: Bundle?) {
 super.onViewCreated(view, savedInstanceState)
 val nameEditText = view.findViewById<TextInputEditText>
 (R.id.textInputTextName)
 nameEditText.addTextChangedListener(
 object : TextWatcher {
 override fun beforeTextChanged(
 charSequence: CharSequence, i: Int, i1: Int, i2: Int) {
 }

 override fun onTextChanged(charSequence: CharSequence,
 i: Int, i1: Int, i2: Int) {
 communicationViewModel!!.setName(charSequence.toString())
 }

 override fun afterTextChanged(editable: Editable) {
 }
 })
}

companion object {
 fun newInstance(): FirstFrament {
 return FirstFrament()
 }
}
}

```

8. Buat fragment kedua, beri nama fragment\_second.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=".SecondFragment">

```

```

<ImageView
 android:id="@+id/imageView2"
 android:layout_width="72dp"
 android:layout_height="72dp"
 android:layout_marginStart="8dp"
 android:layout_marginLeft="8dp"
 android:layout_marginTop="24dp"
 android:layout_marginEnd="8dp"
 android:layout_marginRight="8dp"
 android:src="@drawable/user_avatar"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent" />

<TextView
 android:id="@+id/textViewName"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_marginStart="8dp"
 android:layout_marginTop="8dp"
 android:layout_marginEnd="8dp"
 android:gravity="center"
 android:hint="User Display Name"
 android:textColor="@color/colorPrimaryDark"
 android:textSize="22sp"
 android:textStyle="bold"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintHorizontal_bias="0.0"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/textView"
 tools:text="Haloo Dunia!" />

<TextView
 android:id="@+id/textView"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_marginTop="24dp"
 android:text="Selamat datang"
 android:textAlignment="center"
 android:textSize="22sp"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/imageView2" />

</androidx.constraintlayout.widget.ConstraintLayout>

```

9. Programnya SecondFragment.kt

```

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup

```

```

import android.widget.TextView
import androidx.fragment.app.Fragment
import androidx.lifecycle.Observer
import androidx.lifecycle.ViewModelProviders

class SecondFragment : Fragment() {
 private var communicationViewModel: CommunicationViewModel? = null
 private var txtName: TextView? = null

 override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)
 communicationViewModel = ViewModelProviders.
 of(requireActivity()).
 get(CommunicationViewModel::class.java)
 }

 override fun onCreateView(
 inflater: LayoutInflater, container: ViewGroup?,
 savedInstanceState: Bundle?
): View? {
 return inflater.inflate(R.layout.fragment_second,
 container, false)
 }

 override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
 super.onViewCreated(view, savedInstanceState)
 txtName = view.findViewById(R.id.textViewName)
 communicationViewModel!!.name.observe(requireActivity(),
 Observer { s -> txtName!!.text = s })
 }

 companion object {
 fun newInstance(): SecondFragment {
 return SecondFragment()
 }
 }
}

```

10. Buat sebuah adapter dengan nama ViewPagerAdapter.kt

```

import android.content.Context
import androidx.annotation.StringRes
import androidx.fragment.app.Fragment
import androidx.fragment.app.FragmentManager
import androidx.fragment.app.FragmentPagerAdapter

class ViewPagerAdapter(private val mContext: Context, fm:
FragmentManager) :
 FragmentPagerAdapter(fm) {

 override fun getItem(position: Int): Fragment {
 return if (position == 0) {

```

```

 FirstFrament.newInstance()
 } else {
 SecondFragment.newInstance()
 }
}

override fun getPageTitle(position: Int): CharSequence? {
 return mContext.resources.getString(TAB_TITLES[position])
}

override fun getCount(): Int {
 return 2
}

companion object {
 @StringRes
 private val TAB_TITLES = intArrayOf(R.string.tab_text_1,
 R.string.tab_text_2)
}
}

```

11. Kemudian, buka MainActivity.kt dan tuliskan kode program, sehingga menjadi seperti berikut

```

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {

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

 view_pager.adapter = ViewPagerAdapter(
 this, supportFragmentManager)
 tabs.setupWithViewPager(view_pager)
 }
}

```

12. Jalankan dan amati hasilnya



## LATIHAN

---

- Modifikasilah aplikasi dengan menambahkan satu fragment lagi.



## TUGAS

---

1. Buat aplikasi baru dengan mengembangkan project diatas



## REFERENSI

---

1. <https://kotlinlang.org/docs/reference/>
2. <https://developer.android.com/kotlin>
3. <https://developer.android.com/courses/kotlin-android-fundamentals/toc>
4. <https://codelabs.developers.google.com/android-kotlin-fundamentals/>
5. <https://developer.android.com/kotlin/learn>
6. <https://developer.android.com/kotlin/resources>

## MODUL 9

# Room Database



### CAPAIAN PEMBELAJARAN

1. Mahasiswa dapat membuat Aplikasi dengan database.



### KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. Android Studio 3.4.
2. Handphone Android versi 7.0 (Nougat)
3. Kabel data USB.
4. Driver ADB.

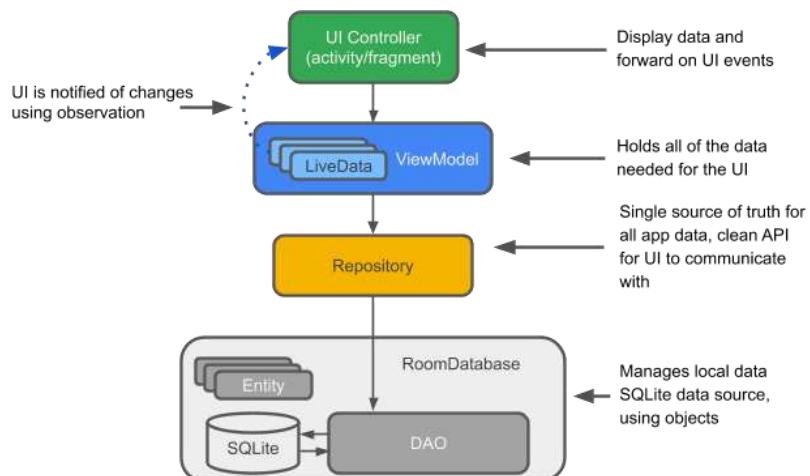


### DASAR TEORI

<https://codelabs.developers.google.com/codelabs/android-room-with-a-view-kotlin/#0>

Komponen arsitektur membantu kita menyusun aplikasi dengan cara yang kuat, dapat diuji, dan dapat dipelihara dengan kode yang lebih sederhana. Saat ini kita akan berfokus pada subset komponen, yaitu LiveData, ViewModel, dan Room.

Berikut diagram yang menunjukkan bentuk dasar arsitektur:



**Entity:** Kelas beranotasi yang menjelaskan tabel database saat bekerja dengan Room.

**SQLite database:** Di penyimpanan perangkat. Room persistence library membuat dan mengelola database ini untuk kita.

**DAO:** Data access object. Pemetaan query SQL ke fungsi. Ketika kita menggunakan DAO, kita memanggil metode, dan Room mengurus sisanya.

Apa itu DAO? Di DAO (objek akses data), kita menentukan kueri SQL dan mengaitkannya dengan panggilan metode. Kompiler memeriksa SQL dan menghasilkan kueri dari anotasi kenyamanan untuk kueri umum, seperti @Insert. Room menggunakan DAO untuk membuat API bersih untuk kode kita.

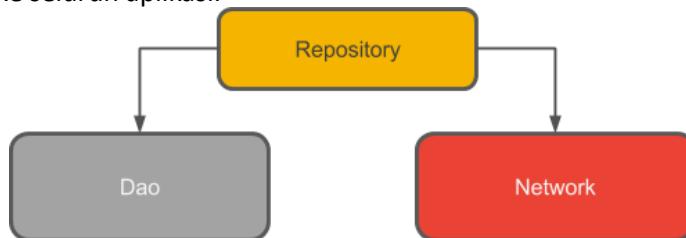
DAO harus berupa antarmuka atau kelas abstrak. Secara default, semua kueri harus dijalankan pada thread terpisah. Room memiliki dukungan coroutines, memungkinkan pertanyaan kita dijelaskan dengan pengubah penangguhan dan kemudian dipanggil dari coroutine atau dari fungsi suspensi lain.

**Room database** : Menyederhanakan pekerjaan basis data dan berfungsi sebagai titik akses ke basis data SQLite yang mendasarinya (menyembunyikan SQLiteOpenHelper). Database Room menggunakan DAO untuk mengeluarkan pertanyaan ke database SQLite.

Apa itu database Room? Room adalah lapisan basis data di atas basis data SQLite. Room menangani tugas-tugas biasa yang kita gunakan untuk menangani dengan SQLiteOpenHelper. Room menggunakan DAO untuk mengeluarkan pertanyaan ke basis datanya. Secara default, untuk menghindari kinerja UI yang buruk, Room tidak memungkinkan kita untuk mengeluarkan pertanyaan pada thread utama. Ketika kueri Room mengembalikan LiveData, kueri secara otomatis dijalankan secara tidak sinkron pada background thread. Room menyediakan pemeriksaan waktu kompilasi terhadap pernyataan SQLite.

**Repository**: Kelas yang kita buat yang terutama digunakan untuk mengelola beberapa sumber data.

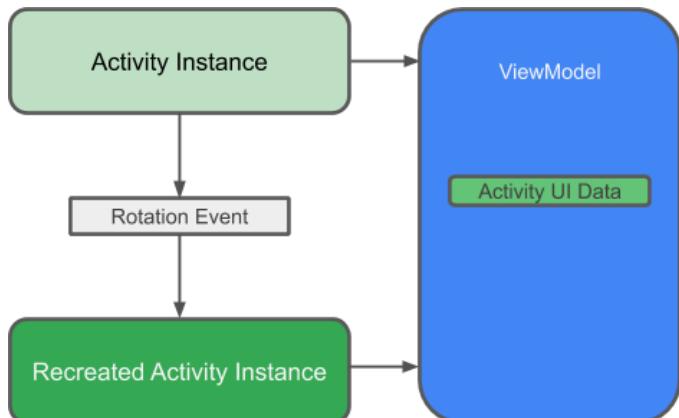
Apa itu Repository? Kelas repository mengabstraksi akses ke banyak sumber data. Repository bukan bagian dari library Komponen Arsitektur, tetapi merupakan praktik terbaik yang disarankan untuk pemisahan kode dan arsitektur. Kelas Repository menyediakan API bersih untuk akses data ke seluruh aplikasi.



Mengapa menggunakan Repository? Repository mengelola kueri dan memungkinkan kita untuk menggunakan beberapa backend. Dalam contoh paling umum, Repository mengimplementasikan logika untuk memutuskan apakah akan mengambil data dari jaringan atau menggunakan hasil yang di-cache dalam database lokal.

**ViewModel**: Bertindak sebagai pusat komunikasi antara Repository (data) dan UI. UI tidak perlu lagi khawatir tentang asal-usul data. Contoh ViewModel adalah Activity/Fragments.

Apa itu ViewModel? Peran ViewModel adalah untuk menyediakan data ke UI dan survive dari perubahan konfigurasi. ViewModel bertindak sebagai pusat komunikasi antara Repository dan UI. Kita juga dapat menggunakan ViewModel untuk berbagi data antar fragmen. ViewModel adalah bagian dari lifecycle library.



Mengapa menggunakan ViewModel? ViewModel menyimpan data UI aplikasi kita dengan cara yang sadar siklus yang selamat dari perubahan konfigurasi. Memisahkan data UI aplikasi kita dari kelas Activity dan Fragmen, memungkinkan kita mengikuti prinsip tanggung jawab tunggal dengan lebih baik: Activity dan fragmen bertanggung jawab untuk menggambar data ke layar, sementara ViewModel dapat menangani memegang dan memproses semua data yang diperlukan untuk UI. Di ViewModel, gunakan LiveData untuk data yang dapat diubah yang akan digunakan atau ditampilkan oleh UI. Menggunakan LiveData memiliki beberapa manfaat:

- Kita dapat menempatkan pengamat pada data (bukan polling untuk perubahan) dan hanya memperbarui UI ketika data benar-benar berubah.
- Repozitori dan UI sepenuhnya dipisahkan oleh ViewModel.
- Tidak ada panggilan database dari ViewModel (ini semua ditangani di Repozitori), membuat kode lebih dapat diuji.

**viewModelScope.** Di Kotlin, semua coroutine dijalankan di dalam CoroutineScope. Lingkup mengontrol masa pakai coroutine melalui pekerjaannya. Ketika kita membatalkan pekerjaan lingkup, itu membatalkan semua coroutine dimulai dalam lingkup itu. Pustaka siklus hidup AndroidX-viewmodel-ktx menambahkan viewModelScope sebagai fungsi ekstensi dari kelas ViewModel, memungkinkan kita untuk bekerja dengan scope.

**LiveData:** Kelas pemegang data yang dapat diamati. Selalu memegang / menyimpan versi data terbaru, dan memberi tahu pengamatnya ketika data telah berubah. LiveData sadar akan siklus hidup. Komponen UI hanya mengamati data yang relevan dan jangan berhenti atau melanjutkan pengamatan. LiveData secara otomatis mengelola semua ini karena menyadari perubahan status siklus hidup yang relevan saat mengamati.

Saat data berubah, kita biasanya ingin mengambil tindakan, seperti menampilkan data yang diperbarui di UI. Ini berarti kita harus mengamati data sehingga ketika itu berubah, kita dapat bereaksi. Tergantung pada bagaimana data disimpan, ini bisa rumit. Mengamati perubahan pada data di berbagai komponen aplikasi kita dapat membuat jalur ketergantungan yang eksplisit dan kaku antar komponen. Ini membuat pengujian dan debugging menjadi sulit, antara lain. LiveData, lifecycle library class untuk observasi data, memecahkan masalah ini. Gunakan nilai kembalian tipe LiveData dalam deskripsi metode kita, dan Room menghasilkan semua kode yang diperlukan untuk memperbarui LiveData ketika database diperbarui.

**Beberapa penjelasan akan disampaikan di langkah praktik.**

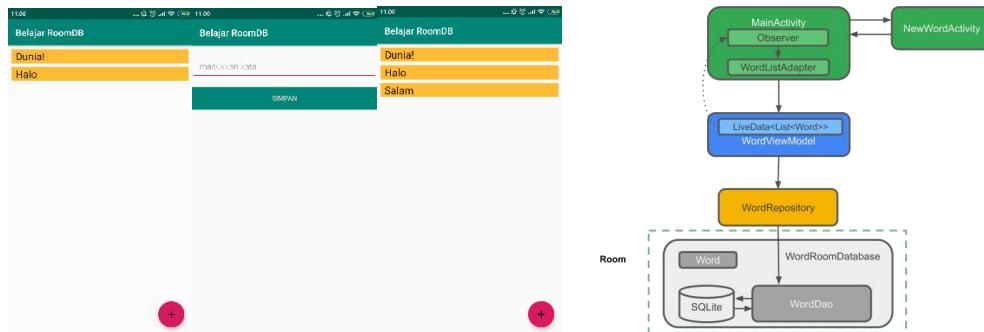


## PRAKTIK

---

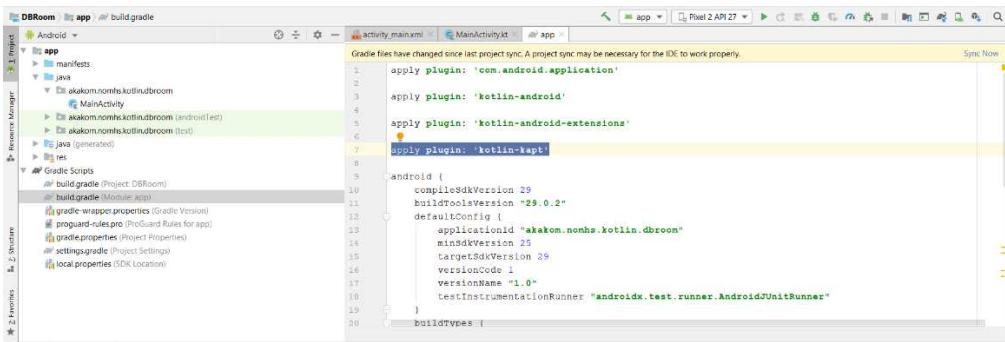
1. Kita akan membuat aplikasi sebagai berikut.

- Bekerja dengan database untuk mendapatkan dan menyimpan data, dan prapopulasikan database dengan beberapa kata.
- Menampilkan semua kata dalam RecyclerView di MainActivity.
- Membuka aktivitas kedua saat pengguna menekan tombol +. Saat pengguna memasukkan kata, tambahkan kata ke database dan daftar.



- Buat sebuah project baru. Buat juga sebuah Empty Activity
- Perbarui Gradle, dengan langkah berikut.
- Di build.gradle (Module: app), tambahkan berikut

```
apply plugin: 'kotlin-kapt'
```

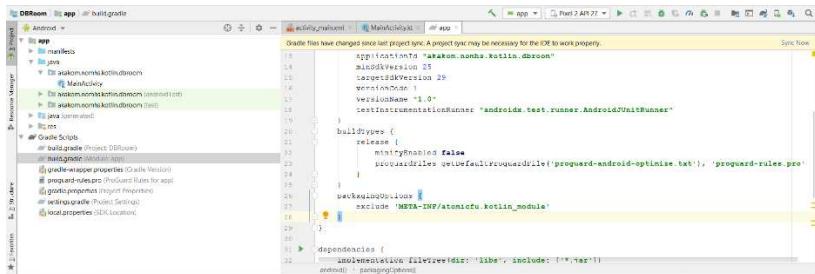


- Masih di build.gradle(Module:app), tambahkan blok packagingOptions di dalam blok android untuk mengecualikan *atomic functions module* dari package dan mencegah warnings.

```

packagingOptions {
 exclude 'META-INF/atomicfu.kotlin_module'
}

```



- Tambahkan blok kode berikut

```

// Room
implementation "androidx.room:room-runtime:$rootProject.roomVersion"
implementation "androidx.room:room-ktx:$rootProject.roomVersion"
kapt "androidx.room:room-compiler:$rootProject.roomVersion"
androidTestImplementation "androidx.room:room-testing:$rootProject.roomVersion"

// Lifecycle
implementation "androidx.lifecycle:lifecycle-

```

```

extensions:$rootProject.archLifecycleVersion"
kapt "androidx.lifecycle:lifecycle-compiler:$rootProject.archLifecycleVersion"
androidTestImplementation "androidx.arch.core:core-
testing:$rootProject.androidxArchVersion"

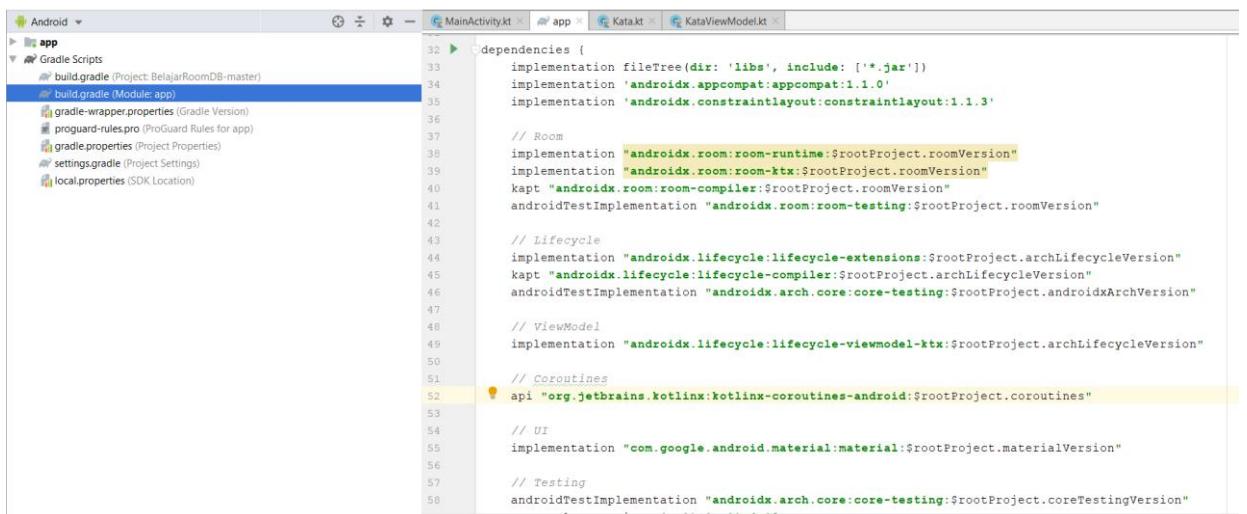
// ViewModel
implementation "androidx.lifecycle:lifecycle-viewmodel-
ktx:$rootProject.archLifecycleVersion"

// Coroutines
api "org.jetbrains.kotlinx:kotlinx-coroutines-android:$rootProject.coroutines"

// UI
implementation
"com.google.android.material:material:$rootProject.materialVersion"

// Testing
androidTestImplementation "androidx.arch.core:core-
testing:$rootProject.coreTestingVersion"

```

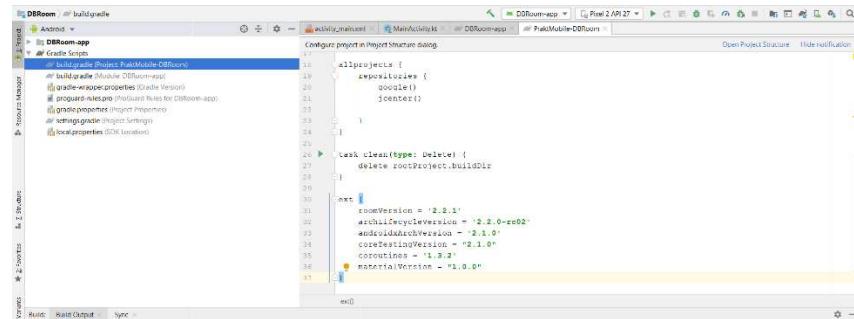


4. Di file build.gradle (Project: DBRoom) tambahkan nomor versi ke akhir file, seperti yang diberikan dalam kode di bawah ini.

```

ext {
 roomVersion = '2.2.1'
 archLifecycleVersion = '2.2.0-rc02'
 androidxArchVersion = '2.1.0'
 coreTestingVersion = "2.1.0"
 coroutines = '1.3.2'
 materialVersion = "1.0.0"
}

```

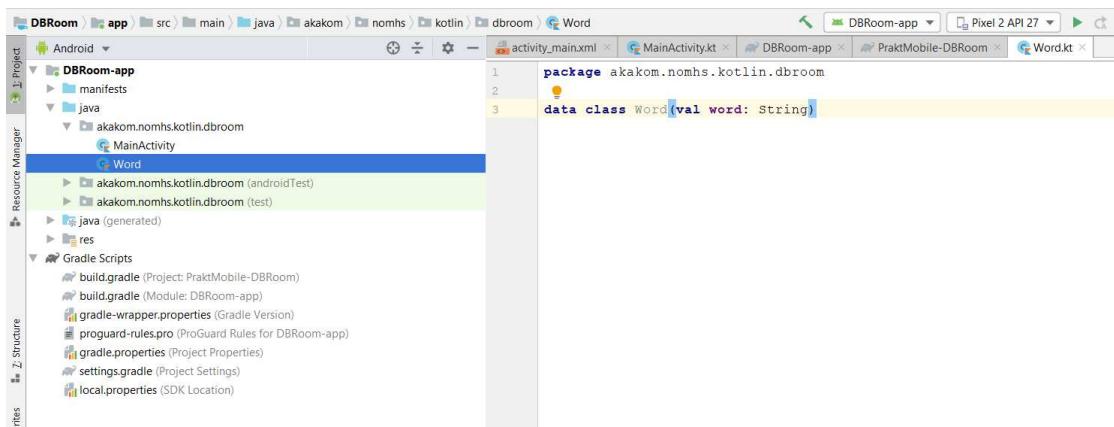


5. Kita akan membuat sebuah Entity sebagai berikut. Room memungkinkan kita membuat tabel melalui Entity.

word_table table	
word	(Primary Key, String)
	"Hello"
	"World"

6. Buat file kelas Kotlin baru bernama Word yang berisi kelas data Word. Kelas ini akan menjelaskan Entitas (yang mewakili tabel SQLite) untuk kata-kata yang dimasukkan. Setiap properti publik di kelas mewakili kolom dalam tabel. Room pada akhirnya akan menggunakan properti ini untuk membuat tabel dan instantiate objek dari baris dalam database.

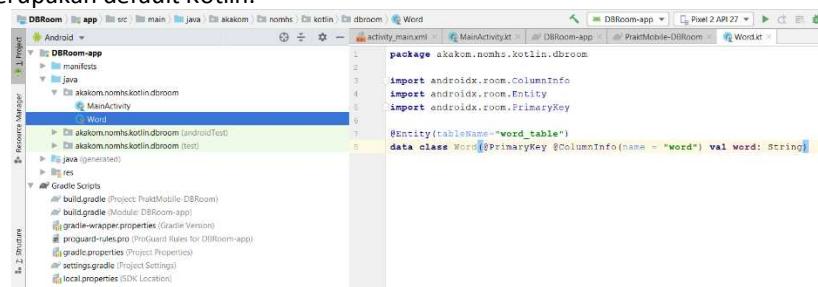
```
data class Word(val word: String)
```



7. Untuk membuat kelas Word bermakna bagi basis data Room, kita perlu memberi anotasi. Anotasi mengidentifikasi bagaimana setiap bagian dari kelas ini berhubungan dengan entri dalam database. Room menggunakan informasi ini untuk menghasilkan kode.

```
@Entity(tableName="word_table")
data class Word(@PrimaryKey @ColumnInfo(name = "word") val word: String)
```

- a. `@Entity (tableName = "word_table")`. Setiap kelas `@Entity` mewakili tabel SQLite. Buat anotasi deklarasi kelas Anda untuk menunjukkan bahwa itu adalah entitas. Kita bisa menentukan nama tabel jika kita ingin itu berbeda dari nama kelas. Tabel ini dinamai tabel "word\_table".
- b. `@PrimaryKey`. Setiap entitas membutuhkan kunci utama. Agar semuanya sederhana, setiap kata bertindak sebagai kunci utama sendiri.
- c. `@ColumnInfo (name = "word")`. Tentukan nama kolom dalam tabel jika kita ingin itu berbeda dari nama variabel anggota. Kolom ini dinamai "word".
- d. Setiap properti yang disimpan dalam database harus memiliki visibilitas publik, yang merupakan default Kotlin.



8. Kita akan membuat file DAO untuk : Memesan semua kata sesuai abjad, Memasukkan sebuah kata, Menghapus semua kata. Buat file kelas Kotlin baru bernama WordDao. Salin

dan tempel kode berikut ke WordDao dan perbaiki impor yang diperlukan untuk membuatnya dikompilasi.

```
@Dao
interface WordDao {

 @Query("SELECT * from word_table ORDER BY word ASC")
 fun getAlphabetizedWords(): LiveData<List<Word>>

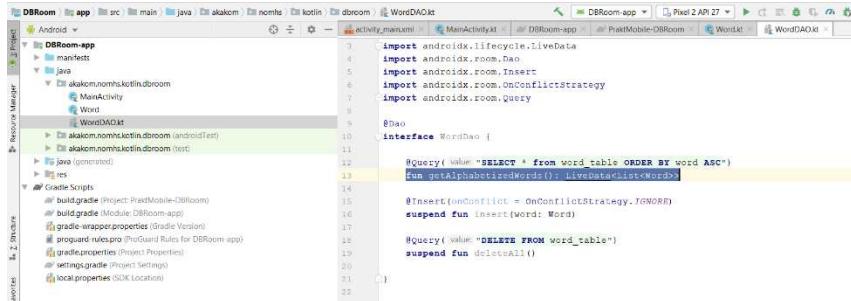
 @Insert(onConflict = OnConflictStrategy.IGNORE)
 suspend fun insert(word: Word)

 @Query("DELETE FROM word_table")
 suspend fun deleteAll()
}
```

- a. WordDao adalah interface; DAO harus berupa interface atau kelas abstrak.
  - b. Anotasi @Dao mengidentifikasinya sebagai kelas DAO untuk Room.
  - c. suspend fun insert(word: Word): Menyatakan fungsi tunda untuk memasukkan satu kata.
  - d. Anotasi @Insert adalah anotasi metode DAO khusus di mana kita tidak harus menyediakan SQL apa pun!
  - e. onConflict = OnConflictStrategy.IGNORE: Strategi konflik yang dipilih mengabaikan kata baru jika itu persis sama dengan yang sudah ada dalam list.
  - f. suspend fun deleteAll (): Menyatakan fungsi tunda untuk menghapus semua kata.
  - g. Tidak ada penjelasan kenyamanan untuk menghapus banyak entitas, jadi ini dijelaskan dengan @Query umum.
  - h. @Query ("DELETE FROM word\_table"): @Query mengharuskan untuk memberikan kueri SQL sebagai parameter string ke anotasi, memungkinkan kueri baca kompleks dan operasi lainnya.
  - i. fun getAlphabetizedWords (): List <Word>: Metode untuk mendapatkan semua kata dan mengembalikannya ke List dari Word.
  - j. @Query ("SELECT \* from word\_table ORDER BY word ASC"): Permintaan yang mengembalikan daftar kata yang diurutkan dalam urutan menaik.
9. Saat data berubah, kita biasanya ingin mengambil tindakan, seperti menampilkan data yang diperbarui di UI. Ini berarti kita harus mengamati data sehingga ketika itu berubah, kita dapat bereaksi. Tergantung pada bagaimana data disimpan, ini bisa rumit. Mengamati perubahan pada data di berbagai komponen aplikasi akan dapat membuat jalur ketergantungan yang eksplisit dan kaku antar komponen. Ini antara lain yang membuat pengujian dan debugging menjadi sulit. LiveData, lifecycle library class untuk observasi data, memecahkan masalah ini. Gunakan nilai kembalian tipe LiveData dalam deskripsi metode Anda, dan Room menghasilkan semua kode yang diperlukan untuk memperbarui LiveData ketika database diperbarui. Di WordDao, ubah deklarasi metode getAlphabetizedWords () sehingga List <Word> yang dikembalikan dibungkus dengan LiveData.

```
@Query("SELECT * from word_table ORDER BY word ASC")
fun getAlphabetizedWords(): LiveData<List<Word>>
```

Kemudian, kita melacak perubahan data melalui Observer di MainActivity.



10. Kelas database Room Anda harus abstrak dan extends RoomDatabase. Biasanya, Anda hanya perlu satu instance dari database Room untuk seluruh aplikasi. Buat file kelas Kotlin bernama **WordRoomDatabase** dan tambahkan kode ini ke dalamnya:

```

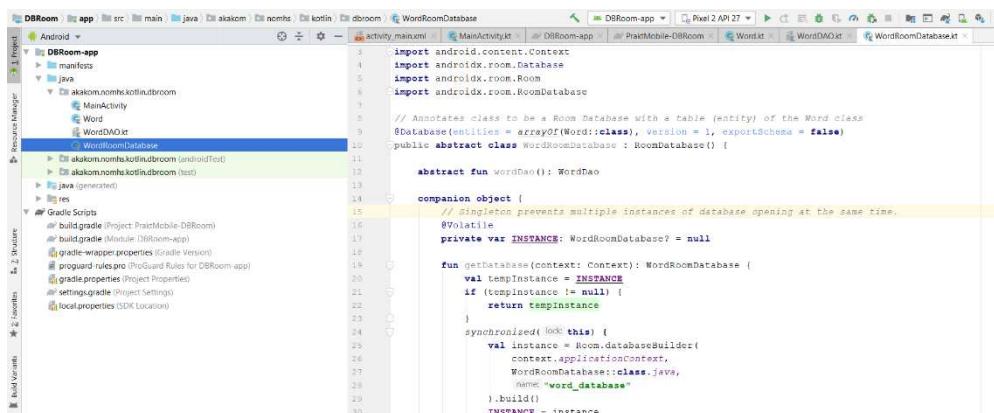
// Annotates class to be a Room Database with a table (entity) of the Word class
@Database(entities = arrayOf(Word::class), version = 1, exportSchema = false)
public abstract class WordRoomDatabase : RoomDatabase() {

 abstract fun wordDao(): WordDao

 companion object {// Singleton prevents multiple instances of
 // database opening at the same time.
 @Volatile
 private var INSTANCE: WordRoomDatabase? = null

 fun getDatabase(context: Context): WordRoomDatabase {
 val tempInstance = INSTANCE
 if (tempInstance != null) {
 return tempInstance
 }
 synchronized(this) {
 val instance = Room.databaseBuilder(
 context.applicationContext,
 WordRoomDatabase::class.java,
 "word_database"
).build()
 INSTANCE = instance
 return instance
 }
 }
 }
}

```



- Kelas database untuk Room harus abstrak dan extends RoomDatabase
- Kita memberikan keterangan kelas menjadi database Room dengan @ Database dan menggunakan parameter penjelasan untuk menyatakan entitas yang termasuk dalam database dan mengatur nomor versi. Setiap entitas terkait dengan tabel yang akan dibuat dalam database. Dalam aplikasi nyata, kita harus mempertimbangkan

- pengaturan direktori untuk Room untuk digunakan untuk mengekspor skema sehingga kita dapat memeriksa skema saat ini ke sistem kontrol versi kita.
- c. Kita membuat database menyediakan DAO dengan membuat method abstrak "getter" untuk setiap @Dao.
  - d. Kita telah mendefinisikan sebuah singleton, WordRoomDatabase, untuk mencegah agar beberapa instance database tidak dibuka secara bersamaan.
  - e. getDatabase mengembalikan singleton. Ini akan membuat database saat pertama kali diakses, menggunakan pembuat basis data Room untuk membuat objek RoomDatabase dalam konteks aplikasi dari kelas WordRoomDatabase dan menamakannya "word\_database".
11. Selanjutnya, buat file kelas Kotlin bernama WordRepository dan tuliskan kode berikut ke dalamnya:

```
class WordRepository(private val wordDao: WordDao) {
 val allWords: LiveData<List<Word>>=wordDao.getAlphabetizedWords()
 suspend fun insert(word: Word) {
 wordDao.insert(word)
 }
}
```

DAO dilewatkan ke konstruktor repositori sebagai lawan dari keseluruhan database. Hal ini karena ini hanya membutuhkan akses ke DAO, karena DAO berisi semua method read/write untuk database. Tidak perlu mengekspos seluruh database ke repositori.

Daftar kata-kata adalah milik umum, yang diinisialisasi dengan mendapatkan daftar kata-kata LiveData dari Room; kita bisa melakukan ini karena cara kita mendefinisikan method getAlphabetizedWords untuk mengembalikan LiveData di langkah "LiveData class". Room mengeksekusi semua pertanyaan pada thread terpisah. Kemudian diamati LiveData akan memberi tahu pengamat di thread utama ketika data telah berubah. Pengubah suspend memberi tahu kompiler bahwa ini perlu dipanggil dari coroutine atau fungsi penangguhan lain.

12. Buat sebuah file Kotlin class untuk WordViewModel dan tuliskan kode berikut:

```
// Class extends AndroidViewModel and requires application as a parameter.
class WordViewModel(application: Application) : AndroidViewModel(application) {
 private val repository: WordRepository
 val allWords: LiveData<List<Word>>

 init {
 val wordsDao = WordRoomDatabase.getDatabase(application,
 viewModelScope).wordDao()
 repository = WordRepository(wordsDao)
 allWords = repository.allWords
 }

 fun insert(word: Word) = viewModelScope.launch {
 repository.insert(word)
 }
}
```

```

import androidx.lifecycle.AndroidViewModel
import androidx.lifecycle.LiveData
import androidx.lifecycle.viewModelScope
import kotlinx.coroutines.launch

// Class extends AndroidViewModel and requires application as a parameter.
class WordViewModel(application: Application) : AndroidViewModel(application) {

 // The ViewModel maintains a reference to the repository to get data.
 private val repository: WordRepository
 // LiveData gives us updated words when they change.
 val allWords: LiveData<List<Word>>

 init {
 // Gets reference to WordDao from WordRoomDatabase to construct
 // the correct WordRepository.
 val wordsDao = WordRoomDatabase.getDatabase(application).wordDao()
 repository = WordRepository(wordsDao)
 allWords = repository.allWords
 }

 /**
 * The implementation of insert() in the database is completely hidden from the UI.
 * Room ensures that you're not doing any long running operations on
 * the main thread, blocking the UI, so we don't need to handle changing Dispatchers.
 * ViewModels have a coroutine scope based on their lifecycle called
 * viewModelScope which we can use here.
 */
 fun insert(word: Word) = viewModelScope.launch { repository.insert(word) }
}

```

- Membuat kelas yang disebut WordViewModel yang mendapatkan Application sebagai parameter dan extends AndroidViewModel.
- Menambahkan variabel private member untuk menyimpan referensi ke repositori.
- Menambahkan variabel anggota LiveData public ke cache daftar kata.
- Membuat blok init yang mendapatkan referensi ke WordDao dari WordRoomDatabase.
- Di blok init, dibangun WordRepository berdasarkan WordRoomDatabase.
- Di blok init, menginisialisasi LiveData allWords menggunakan repositori.
- Membuat metode insert() yang memanggil metode insert () Repotori. Dengan cara ini, implementasi insert () dienkapsulasi dari UI. Kita tidak ingin memasukkan untuk memblokir thread utama, jadi kami meluncurkan coroutine baru dan memanggil sisipan repositori, yang merupakan fungsi suspend. Seperti yang disebutkan, ViewModels memiliki ruang lingkup coroutine berdasarkan siklus hidup mereka yang disebut viewModelScope, yang kita gunakan di sini.

13. Tambahkan style untuk daftar item dalam values/styles.xml:

```

<!-- Base application theme. -->
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
 <!-- Customize your theme here. -->
 <item name="colorPrimary">@color/colorPrimary</item>
 <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
 <item name="colorAccent">@color/colorAccent</item>
</style>

```

```

<resources>
 <!-- Base application theme. -->
 <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
 <!-- Customize your theme here. -->
 <item name="colorPrimary">@color/colorPrimary</item>
 <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
 <item name="colorAccent">@color/colorAccent</item>
 <item name="android:layout_width">match_parent</item>
 <item name="android:layout_marginBottom">8dp</item>
 <item name="android:paddingLeft">8dp</item>
 <item name="android:background">@android:color/holo_orange_light</item>
 <item name="android:textAppearance">@android:style/TextAppearance.Large</item>
 </style>
</resources>

```

14. Tambahkan layout layout/recyclerview\_item.xml :

```

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

 <TextView
 android:id="@+id/textView"

```

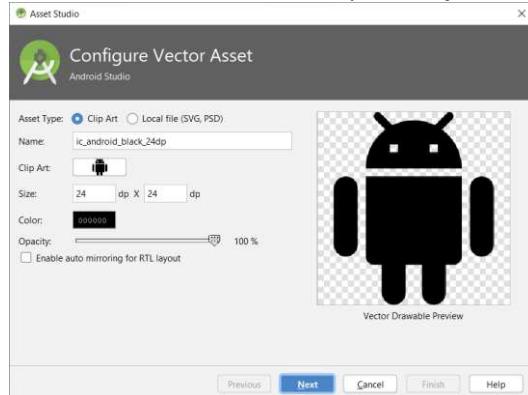
```

 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:background="@android:color/holo_orange_light" />
 </LinearLayout>

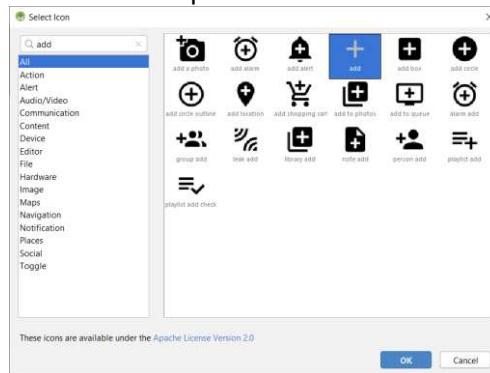
```

15. Tampilan FAB harus sesuai dengan tindakan yang tersedia, jadi kami ingin mengganti ikon dengan simbol '+'. Kita perlu menambahkan Vector Aset baru.

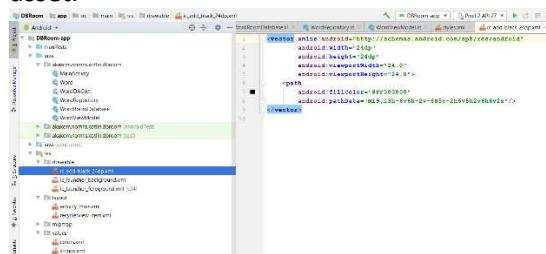
- Pilih **File > New > Vector Asset**.
- Klik ikon Android robot icon pada **Clip Art**: field.



- Cari "add" dan pilih '+' asset. Klik **OK**.



- Konfirmasi path ikon sebagai main > drawable dan klik **Finish** untuk menambahkan asset.



16. Di layout/activity\_main.xml, update FAB untuk menambahkan drawable baru.

```

<com.google.android.material.floatingactionbutton.FloatingActionButton
 android:id="@+id/fab"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintEnd_toEndOf="parent"
 android:src="@drawable/ic_add_black_24dp"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_margin="16dp" />

```

17. Kita akan menampilkan data dalam RecyclerView, yang sedikit lebih bagus daripada hanya membuang data dalam TextView. Codelab ini berasumsi bahwa kita tahu cara kerja RecyclerView, RecyclerView.Layout, RecyclerView.ViewHolder, dan RecyclerView.Adapter. Perhatikan bahwa variabel kata-kata dalam adaptor menyimpan data.

18. Buat file kelas Kotlin untuk WordListAdapter yang memperluas RecyclerView.Adapter. Berikut kodenya.

```

class WordListAdapter internal constructor(context: Context) : RecyclerView.Adapter<WordListAdapter.WordViewHolder>() {

 private val inflater: LayoutInflater =
 LayoutInflater.from(context)
 private var words = emptyList<Word>() // Cached copy of words

 inner class WordViewHolder(itemView: View) :
 RecyclerView.ViewHolder(itemView) {
 val wordItemView: TextView =
 itemView.findViewById(R.id.textView)
 }

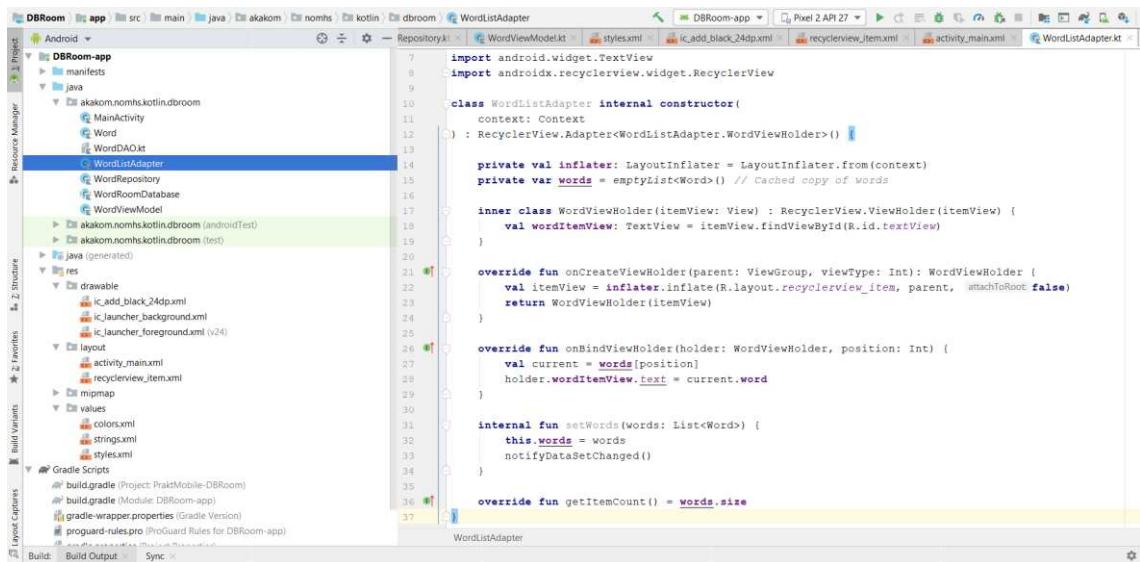
 override fun onCreateViewHolder(parent: ViewGroup,
 viewType: Int): WordViewHolder {
 val itemView = inflater.inflate(R.layout.recyclerview_item,
 parent, false)
 return WordViewHolder(itemView)
 }

 override fun onBindViewHolder(holder: WordViewHolder,
 position: Int) {
 val current = words[position]
 holder.wordItemView.text = current.word
 }

 internal fun setWords(words: List<Word>) {
 this.words = words
 notifyDataSetChanged()
 }

 override fun getItemCount() = words.size
}

```



19. Tambahkan RecyclerView dalam method onCreate() dari MainActivity. Dalam method onCreate() sesudah setContentView:

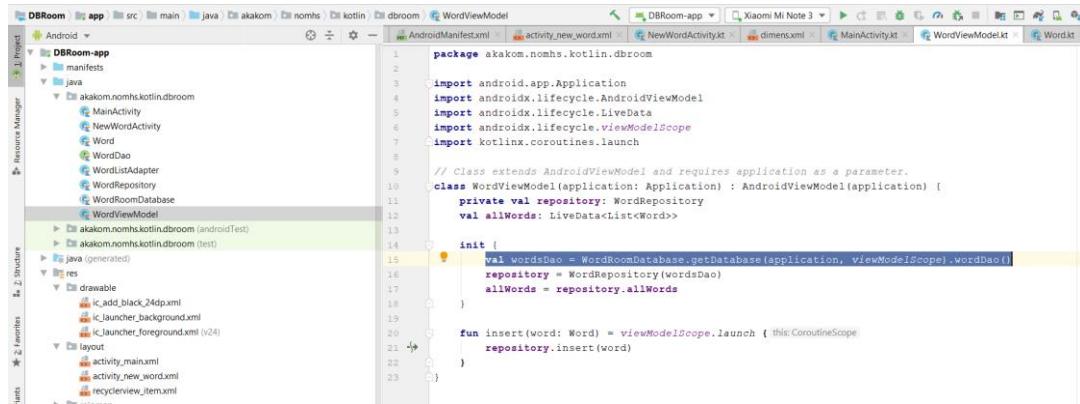
```

val recyclerView = findViewById<RecyclerView>(R.id.recyclerview)
val adapter = WordListAdapter(this)
recyclerView.adapter = adapter
recyclerView.layoutManager = LinearLayoutManager(this)

```

20. Jalankan aplikasi untuk memastikan semuanya berfungsi. Tidak ada item, karena kita belum menghubungkan data, jadi aplikasi harus menampilkan latar belakang abu-abu tanpa item daftar.
21. Untuk meluncurkan coroutine kita membutuhkan CoroutineScope. Perbarui metode getDatabase dari kelas WordRoomDatabase, untuk juga mendapatkan lingkup coroutine sebagai parameter. Perbarui penginisialisasi pengambilan basis data di blok init dari WordViewModel untuk juga melewati ruang lingkup.

```
val wordsDao = WordRoomDatabase.getDatabase(application, viewModelScope).wordDao()
```



22. Di WordRoomDatabase, kita membuat implementasi kustom dari RoomDatabase.Callback(), yang juga mendapatkan CoroutineScope sebagai parameter konstruktor. Kemudian, mengganti metode onOpen untuk mengisi basis data. Berikut adalah kode untuk membuat panggilan balik di dalam kelas WordRoomDatabase:

```

private class WordDatabaseCallback(
 private val scope: CoroutineScope
) : RoomDatabase.Callback() {

 override fun onOpen(db: SupportSQLiteDatabase) {
 super.onOpen(db)
 INSTANCE?.let { database ->
 scope.launch {
 populateDatabase(database.wordDao())
 }
 }
 }

 suspend fun populateDatabase(wordDao: WordDao) {
 // Delete all content here.
 wordDao.deleteAll()

 // Add sample words.
 var word = Word("Hello")
 wordDao.insert(word)
 word = Word("World!")
 wordDao.insert(word)

 // TODO: Add your own words!
 }
}
```

23. Terakhir, tambahkan callback ke urutan pembuatan basis data tepat sebelum memanggil .build() di Room.databaseBuilder().

```
.addCallback(WordDatabaseCallback(scope))
```

Kode lengkapnya menjadi sebagai berikut.

```
@Database(entities = arrayOf(Word::class), version = 1,
exportSchema = false)
public abstract class WordRoomDatabase : RoomDatabase() {

 abstract fun wordDao(): WordDao

 private class WordDatabaseCallback(
 private val scope: CoroutineScope
) : RoomDatabase.Callback() {

 override fun onOpen(db: SupportSQLiteDatabase) {
 super.onOpen(db)
 INSTANCE?.let { database ->
 scope.launch {
 populateDatabase(database.wordDao())
 }
 }
 }

 suspend fun populateDatabase(wordDao: WordDao) {
 // Delete all content here.
 wordDao.deleteAll()

 // Add sample words.
 var word = Word("Hello")
 wordDao.insert(word)
 word = Word("World!")
 wordDao.insert(word)

 // TODO: Add your own words!
 }
 }

 companion object {
 @Volatile
 private var INSTANCE: WordRoomDatabase? = null

 fun getDatabase(
 context: Context,
 scope: CoroutineScope
): WordRoomDatabase {
 // if the INSTANCE is not null, then return it,
 // if it is, then create the database
 return INSTANCE ?: synchronized(this) {
 val instance = Room.databaseBuilder(
 context.applicationContext,
 WordRoomDatabase::class.java,
 "word_database"
)
 .addCallback(WordDatabaseCallback(scope))
 .build()
 INSTANCE = instance
 // return instance
 instance
 }
 }
 }
}
```

24. Tambahkan berikut pada string resources values/strings.xml

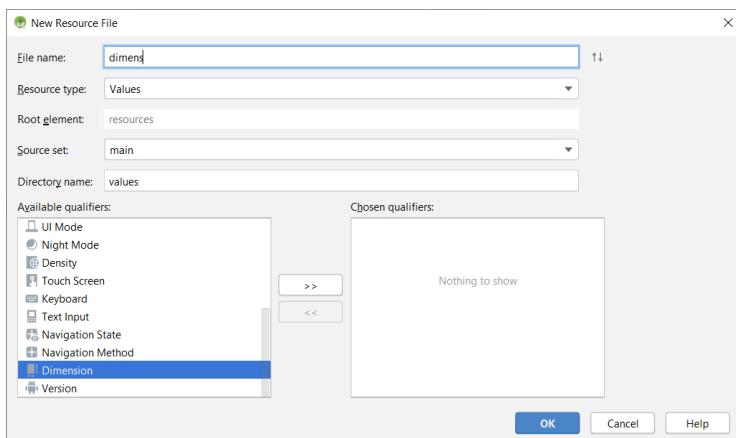
```
<string name="hint_word">Kata...</string>
<string name="button_save">Simpan</string>
<string name="empty_not_saved">Kata tidak dapat disimpan karena kosong.</string>
```

25. Tambahkan pada color resource value/colors.xml

```
<color name="buttonLabel">#d3d3d3</color>
```

26. Buat sebuah dimension resource file:

- Klik app module dalam **Project** window.
- Pilih **File > New > Android Resource File**
- Dari Available Qualifiers, pilih **Dimension**
- Beri nama file: dimens

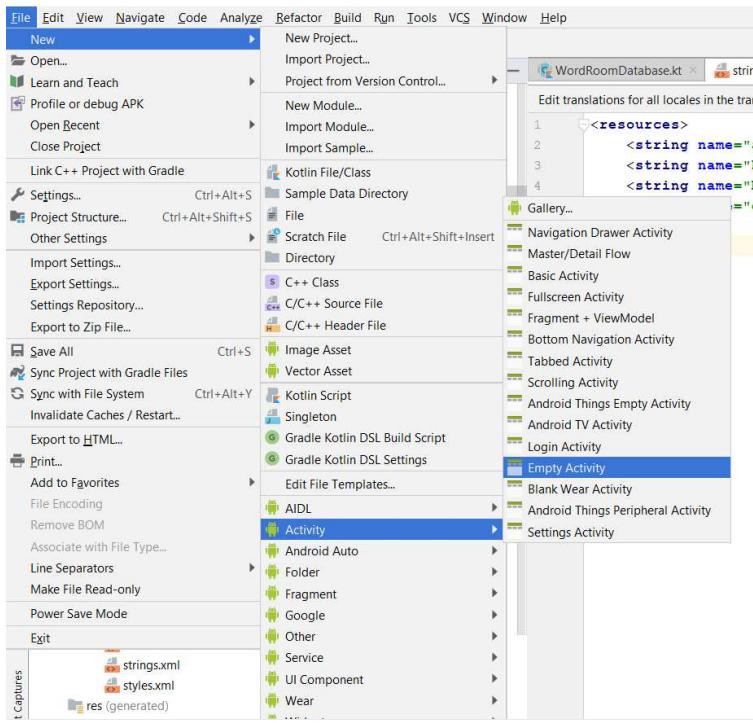


27. Tambahkan pada dimension resources dalam values/dimens.xml:

```
<dimen name="small_padding">6dp</dimen>
<dimen name="big_padding">16dp</dimen>
```

28. Buat sebuah empty Android Activity baru dengan template Empty Activity

- Pilih **File > New > Activity > Empty Activity**
- Tuliskan **NewWordActivity** untuk nama Activity.



29. Verifikasi bahwa activity baru telah ditambahkan ke Android Manifest!

```
<activity android:name=".NewWordActivity"></activity>
```

30. Update file `activity_new_word.xml` dalam folder layout folder dengan code berikut

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

 <EditText
 android:id="@+id/edit_word"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:fontFamily="sans-serif-light"
 android:hint="@string/hint_word"
 android:inputType="textAutoComplete"
 android:padding="@dimen/small_padding"
 android:layout_marginBottom="@dimen/big_padding"
 android:layout_marginTop="@dimen/big_padding"
 android:textSize="18sp" />

 <Button
 android:id="@+id/button_save"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:background="@color/colorPrimary"
 android:text="@string/button_save"
 android:textColor="@color/buttonLabel" />

</LinearLayout>
```

31. Update kode untuk activity NewWordActivity

```

class NewWordActivity : AppCompatActivity() {

 private lateinit var editWordView: EditText

 public override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)
 setContentView(R.layout.activity_new_word)
 editWordView = findViewById(R.id.edit_word)

 val button = findViewById<Button>(R.id.button_save)
 button.setOnClickListener {
 val replyIntent = Intent()
 if (TextUtils.isEmpty(editWordView.text)) {
 setResult(Activity.RESULT_CANCELED, replyIntent)
 } else {
 val word = editWordView.text.toString()
 replyIntent.putExtra(EXTRA_REPLY, word)
 setResult(Activity.RESULT_OK, replyIntent)
 }
 finish()
 }
 }

 companion object {
 const val EXTRA_REPLY =
 "com.example.android.wordlistsql.REPLY"
 }
}

```

32. Kemudian kita akan menyambungkan UI ke basis data dengan menyimpan kata-kata baru yang dimasukkan pengguna dan menampilkan konten saat ini dari basis data kata di RecyclerView. Untuk menampilkan konten saat ini dari database, tambahkan pengamat yang mengamati LiveData di ViewModel. Setiap kali data berubah, panggilan balik onChanged () dipanggil, yang memanggil metode setWord () adaptor untuk memperbarui data cache adaptor dan refresh daftar yang ditampilkan. Di MainActivity, buat variabel anggota untuk ViewModel:

```
private lateinit var wordViewModel: WordViewModel
```

33. Gunakan ViewModelProviders untuk mengaitkan ViewModel dengan Activity. Saat Activity pertama kali dimulai, ViewModelProviders akan membuat ViewModel. Ketika activity dihancurkan, misalnya melalui perubahan konfigurasi, ViewModel tetap ada. Ketika activity dibuat kembali, ViewModelProviders mengembalikan ViewModel yang ada. Di onCreate () di bawah blok kode RecyclerView, dapatkan ViewModel dari ViewModelProvider.

```
wordViewModel = ViewModelProvider(this).
 get(WordViewModel::class.java)
```

34. Juga di onCreate (), tambahkan observer untuk properti LiveData allWords dari WordViewModel. Metode onChanged () fresh ketika data yang diamati berubah dan aktivity berada di latar depan.

```
wordViewModel.allWords.observe(this, Observer { words ->
 // Update the cached copy of the words in the adapter.
}
```

```
 words?.let { adapter.setWords(it) }
 })
```

35. Kita ingin membuka NewWordActivity saat menekan FAB dan, setelah kembali ke MainActivity, untuk memasukkan kata baru ke dalam basis data atau menampilkan Toast, mulai dengan mendefinisikan kode permintaan:

```
private val newWordActivityRequestCode = 1
```

36. Di MainActivity, tambahkan kode onActivityResult () untuk NewWordActivity. Jika activity kembali dengan RESULT\_OK, masukkan kata yang dikembalikan ke database dengan memanggil metode insert () dari WordViewModel.

```
override fun onActivityResult(requestCode: Int, resultCode: Int,
data: Intent?) {
 super.onActivityResult(requestCode, resultCode, data)

 if (requestCode == newWordActivityRequestCode && resultCode
 == Activity.RESULT_OK) {
 data?.getStringExtra(NewWordActivity.EXTRA_REPLY)?.let {
 val word = Word(it)
 wordViewModel.insert(word)
 }
 } else {
 Toast.makeText(
 applicationContext,
 R.string.empty_not_saved,
 Toast.LENGTH_LONG).show()
 }
}
```

37. Di MainActivity, mulai NewWordActivity ketika pengguna menekan FAB. Di MainActivity onCreate, cari FAB dan tambahkan onClickListener dengan kode ini:

```
val fab = findViewById<FloatingActionButton>(R.id.fab)
fab.setOnClickListener {
 val intent = Intent(this@MainActivity,
 NewWordActivity::class.java)
 startActivityForResult(intent, newWordActivityRequestCode)
}
```

38. Jadi kode akhir dari MainActivity adalah sebagai berikut:

```
class MainActivity : AppCompatActivity() {
 private lateinit var wordViewModel: WordViewModel
 private val newWordActivityRequestCode = 1
 override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)
 setContentView(R.layout.activity_main)
 val recyclerView =
 findViewById<RecyclerView>(R.id.recyclerview)
 val adapter = WordListAdapter(this)
 recyclerView.adapter = adapter
 recyclerView.layoutManager = LinearLayoutManager(this)
 wordViewModel = ViewModelProvider(this).
 get(WordViewModel::class.java)
 wordViewModel.allWords.observe(this, Observer { words ->
 // Update the cached copy of the words in the adapter.
 words?.let { adapter.setWords(it) }
 })
 }
}
```

```

 }
 }

 override fun onActivityResult(requestCode: Int,
 resultCode: Int, data: Intent?) {
 super.onActivityResult(requestCode, resultCode, data)

 if (requestCode == newWordActivityRequestCode &&
 resultCode == Activity.RESULT_OK) {
 data?.getStringExtra(NewWordActivity.EXTRA_REPLY)?.let
 {
 val word = Word(it)
 wordViewModel.insert(word)
 }
 } else {
 Toast.makeText(
 applicationContext,
 R.string.empty_not_saved,
 Toast.LENGTH_LONG).show()
 }
 }
}

```

39. Jalankan aplikasi dan amati hasilnya



### LATIHAN

---

1. Modifikasilah aplikasi .



### TUGAS

---

1. Buat aplikasi baru dengan mengembangkan project diatas



### REFERENSI

---

1. <https://kotlinlang.org/docs/reference/>
2. <https://developer.android.com/kotlin>
3. <https://developer.android.com/courses/kotlin-android-fundamentals/toc>
4. <https://codelabs.developers.google.com/android-kotlin-fundamentals/>

5. <https://developer.android.com/kotlin/learn>
6. <https://developer.android.com/kotlin/resources>

## MODUL 10

### Coroutine and Room



#### CAPAIAN PEMBELAJARAN

1. Mahasiswa dapat menggunakan menu dan Dialog.



#### KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. Android Studio 3.4.
2. Handphone Android versi 7.0 (Nougat)
3. Kabel data USB.
4. Driver ADB.

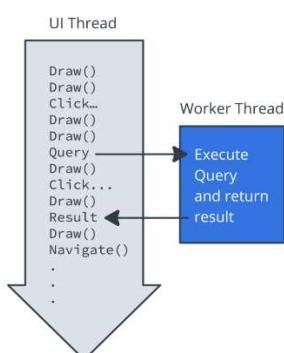


#### DASAR TEORI

<https://codelabs.developers.google.com/codelabs/kotlin-android-training-coroutines-and-room/#0>

#### Coroutines

Di Kotlin, coroutine adalah cara untuk menangani task yang sudah berjalan lama secara elegan dan efisien. Kotlin coroutine memungkinkan kita mengonversi kode berbasis panggilan balik ke kode sekuensial. Kode yang ditulis secara berurutan biasanya lebih mudah dibaca dan bahkan dapat menggunakan fitur bahasa seperti pengecualian. Pada akhirnya, coroutine dan callback melakukan hal yang sama: keduanya menunggu hingga hasilnya tersedia dari task yang sudah berjalan lama dan melanjutkan eksekusi.



Coroutine memiliki sifat-sifat berikut:

- Coroutine asynchronous and non-blocking.

- Coroutines menggunakan fungsi suspend untuk membuat urutan kode asinkron.

### **Coroutines adalah asynchronous**

Coroutine berjalan secara independen dari langkah-langkah eksekusi utama program. Eksekusi ini bisa paralel atau pada prosesor terpisah. Bisa juga bahwa sementara sisa aplikasi sedang menunggu input, kita memasukkan sebuah bit pemrosesan. Salah satu aspek penting dari async adalah kita tidak dapat mengharapkan bahwa hasilnya tersedia, sampai kita secara eksplisit menunggu untuk itu.

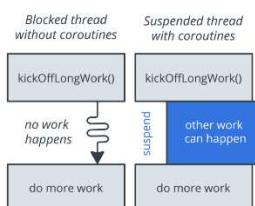
Misalnya, katakanlah kita memiliki pertanyaan yang memerlukan penelitian, dan kita meminta seorang kolega untuk menemukan jawabannya. Mereka pergi dan mengerjakannya, yang seperti mereka melakukan pekerjaan "secara tidak sinkron" dan "pada thread yang terpisah." kita dapat terus melakukan pekerjaan lain yang tidak bergantung pada jawabannya, sampai rekan kita kembali dan memberi tahu apa jawabannya.

### **Coroutines are non-blocking.**

Non-blocking berarti coroutine tidak memblokir thread utama atau UI. Jadi dengan coroutine, pengguna selalu memiliki pengalaman semulus mungkin, karena interaksi UI selalu mendapat prioritas.

### **Coroutines menggunakan fungsi suspend untuk membuat urutan kode asinkron.**

Penanganuhan kata kunci adalah cara Kotlin menandai suatu fungsi, atau jenis fungsi, yang tersedia untuk coroutine. Ketika coroutine memanggil fungsi yang ditandai dengan menangguhkan, alih-alih memblokir sampai fungsi kembali seperti panggilan fungsi normal, coroutine menunda eksekusi hingga hasilnya siap. Kemudian coroutine dilanjutkan di tempat yang ditinggalkannya, dengan hasilnya. Sementara coroutine ditangguhkan dan menunggu hasilnya, ia membuka blokir thread-nya. Dengan begitu, fungsi atau coroutine lain dapat berjalan. Kata kunci yang ditangguhkan tidak menentukan thread bahwa kode berjalan. Fungsi suspend dapat berjalan pada thread latar belakang, atau pada thread utama.



Untuk menggunakan coroutine di Kotlin, Anda memerlukan tiga hal:

- sebuah job
- sebuah dispatcher
- scope

Job: Pada dasarnya, job adalah apa saja yang dapat dibatalkan. Setiap coroutine memiliki job, dan kita dapat menggunakan job itu untuk membatalkan coroutine. Job dapat diatur ke dalam hierarki orang parent-child. Membatalkan job dari parent segera membatalkan semua child job itu, yang jauh lebih nyaman daripada membatalkan setiap coroutine secara manual.

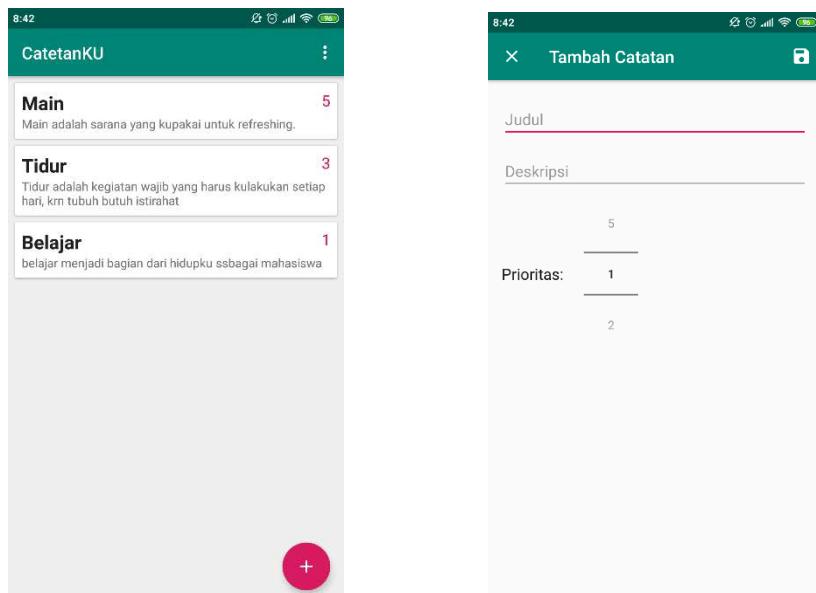
Dispatcher: Dispatcher mengirimkan coroutine untuk berjalan di berbagai thread. Misalnya, Dispatcher.Main menjalankan tugas di thread utama, dan Dispatcher.IO melepaskan muatan yang memblokir tugas I / O ke kumpulan thread yang dibagi.

Scope: Lingkup coroutine mendefinisikan konteks di mana coroutine berjalan. Lingkup menggabungkan informasi tentang pekerjaan dan operator koroutin. Cakupan melacak coroutine. Saat kita meluncurkan coroutine, itu "dalam scope," yang berarti bahwa kita telah menunjukkan scope yang akan melacak coroutine.



## PRAKTIK

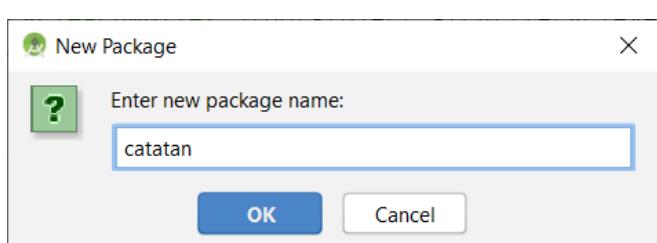
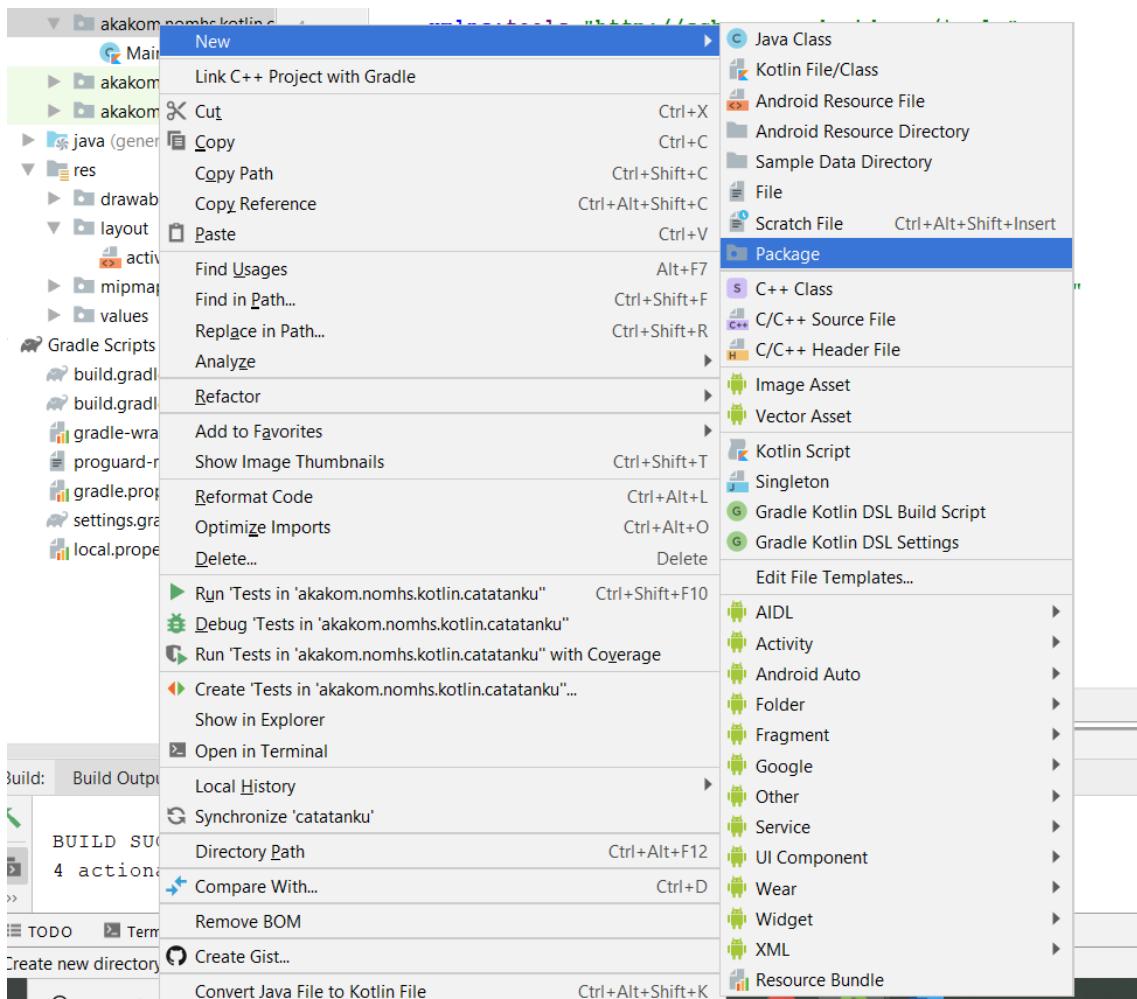
1. Kita akan membuat menu dengan desain sebagai berikut. Kita akan bekerja dengan satu table beberapa field.



2. Layar pertama, ditunjukkan di sebelah kiri, digunakan untuk menampilkan data rekaman catatan yang sudah pernah dimasukkan pengguna. Layar kedua, ditunjukkan di sebelah kanan, digunakan untuk menambah catatan.
3. Buat project baru
4. Perbaharui Gradle seperti modul 9 (langkah 3)
5. Tambahkan komponen RecyclerView pada activity\_main.xml

```
<androidx.recyclerview.widget.RecyclerView
 android:id="@+id/recycler_view"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:background="@color/lightGray"
 tools:itemCount="5"
 tools:listitem="@layout/note_item" />
```

6. Buat package baru dibawah paket yang sudah ada, dengan nama **catatan**.



7. Buat kelas kotlin baru untuk kelas data dengan nama Note, dibawah package catatan.

```
@Entity(tableName = "note_table")
data class Note(
 var title: String,
 var description: String,
 var priority: Int
) {
 @PrimaryKey(autoGenerate = true)
 var id: Int = 0
}
```

```

1 package akakom.nomhs.kotlin.catatanku.catatan
2
3 import androidx.room.Entity
4 import androidx.room.PrimaryKey
5
6 @Entity(tableName = "note_table")
7 data class Note(
8 var title: String,
9 var description: String,
10 var priority: Int
11) {
12 @PrimaryKey(autoGenerate = true)
13 var id: Int = 0
14 }

```

8. Masih di dalam paket catatan, buat file DAO

```

@Dao
interface NoteDao {

 @Insert
 fun insert(note: Note)

 @Update
 fun update(note: Note)

 @Delete
 fun delete(note: Note)

 @Query("DELETE FROM note_table")
 fun deleteAllNotes()

 @Query("SELECT * FROM note_table ORDER BY priority DESC")
 fun getAllNotes(): LiveData<List<Note>>

}

```

9. Dalam paket catatan juga, buat file NoteDatabase

```

@Database(entities = [Note::class], version = 1)
abstract class NoteDatabase : RoomDatabase() {

 abstract fun noteDao(): NoteDao

 companion object {
 private var instance: NoteDatabase? = null

 fun getInstance(context: Context): NoteDatabase? {
 if (instance == null) {
 synchronized(NoteDatabase::class) {
 instance = Room.databaseBuilder(
 context.applicationContext,
 NoteDatabase::class.java, "note_database"
)
 .fallbackToDestructiveMigration()
 .addCallback(roomCallback)
 .build()
 }
 }
 return instance
 }

 fun destroyInstance() {
 instance = null
 }
 }
}

```

```

 private val roomCallback = object : RoomDatabase.Callback() {
 override fun onCreate(db: SupportSQLiteDatabase) {
 super.onCreate(db)
 PopulateDbAsyncTask(instance)
 .execute()
 }
 }

 class PopulateDbAsyncTask(db: NoteDatabase?) : AsyncTask<Unit, Unit, Unit>() {
 private val noteDao = db?.noteDao()

 override fun doInBackground(vararg p0: Unit?) {
 noteDao?.insert(Note("Coba 1", "Deskripsi 1", 1))
 }
 }
}

```

10. Selanjutnya, masih dibawah package catatan, buatlah file kelas Kotlin bernama NoteRepository

```

class NoteRepository(application: Application) {

 private var noteDao: NoteDao

 private var allNotes: LiveData<List<Note>>

 init {
 val database: NoteDatabase = NoteDatabase.getInstance(
 application.applicationContext
)!!
 noteDao = database.noteDao()
 allNotes = noteDao.getAllNotes()
 }

 fun insert(note: Note) {
 val insertNoteAsyncTask = InsertNoteAsyncTask(noteDao).execute(note)
 }

 fun update(note: Note) {
 val updateNoteAsyncTask = UpdateNoteAsyncTask(noteDao).execute(note)
 }

 fun delete(note: Note) {
 val deleteNoteAsyncTask = DeleteNoteAsyncTask(noteDao).execute(note)
 }

 fun deleteAllNotes() {
 val deleteAllNotesAsyncTask = DeleteAllNotesAsyncTask(
 noteDao
).execute()
 }

 fun getAllNotes(): LiveData<List<Note>> {
 return allNotes
 }

 companion object {
 private class InsertNoteAsyncTask(noteDao: NoteDao) : AsyncTask<Note, Unit, Unit>() {
 val noteDao = noteDao

 override fun doInBackground(vararg p0: Note?) {
 noteDao.insert(p0[0]!!)
 }
 }

 private class UpdateNoteAsyncTask(noteDao: NoteDao) : AsyncTask<Note, Unit, Unit>() {
 val noteDao = noteDao

 override fun doInBackground(vararg p0: Note?) {
 noteDao.update(p0[0]!!)
 }
 }

 private class DeleteNoteAsyncTask(noteDao: NoteDao) : AsyncTask<Note, Unit, Unit>() {
 val noteDao = noteDao

 override fun doInBackground(vararg p0: Note?) {
 noteDao.delete(p0[0]!!)
 }
 }

 private class DeleteAllNotesAsyncTask(noteDao: NoteDao) : AsyncTask<Unit, Unit, Unit>() {
 val noteDao = noteDao

 override fun doInBackground(vararg p0: Unit?) {
 noteDao.deleteAllNotes()
 }
 }
 }
}

```

```
}
```

11. Buat sebuah file class Kotlin dengan nama NoteViewModel di package utama.

```
class NoteViewModel(application: Application) : AndroidViewModel(application)
{
 private var repository: NoteRepository =
 NoteRepository(application)
 private var allNotes: LiveData<List<Note>> = repository.getAllNotes()

 fun insert(note: Note) {
 repository.insert(note)
 }

 fun update(note: Note) {
 repository.update(note)
 }

 fun delete(note: Note) {
 repository.delete(note)
 }

 fun deleteAllNotes() {
 repository.deleteAllNotes()
 }

 fun getAllNotes(): LiveData<List<Note>> {
 return allNotes
 }
}
```

12. Tambahkan layout pada layout/note\_item.xml

```
<androidx.cardview.widget.CardView
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:orientation="vertical"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_marginEnd="8dp"
 android:layout_marginStart="8dp"
 android:layout_marginTop="8dp">

 <RelativeLayout
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:padding="8dp">

 <TextView
 android:id="@+id/text_view_priority"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_alignParentEnd="true"
 android:text="1"
 android:textAppearance="@style/TextAppearance.AppCompat.Large"
 android:textColor="@color/colorAccent"
 android:textSize="18sp" />

 <TextView
 android:id="@+id/text_view_title"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_alignParentStart="true"
 android:layout_toStartOf="@+id/text_view_priority"
 android:ellipsize="end"
 android:maxLines="1"
 android:text="Judul"
 android:textAppearance="@style/TextAppearance.AppCompat.Large"
 android:textStyle="bold" />

```

```

<TextView
 android:id="@+id/text_view_description"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_below="@+id/text_view_title"
 android:text="Deskripsi"/>

</RelativeLayout>

</androidx.cardview.widget.CardView>

```

13. Buat vector Asset untuk symbol + (tambah), x ( tutup) dan save (simpan)
14. Tambahkan sebuah FAB pada activity\_main.xml

```

<com.google.android.material.floatingactionbutton.FloatingActionButton
 android:id="@+id/buttonAddNote"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:src="@drawable/ic_save_black_24dp"
 android:layout_marginEnd="16dp"
 android:layout_marginBottom="16dp"
 android:layout_gravity="bottom|right"/>

```

15. Buat adapter dengan nama NoteAdapter.

```

class NoteAdapter : ListAdapter<Note, NoteAdapter.NoteHolder>(DIFF_CALLBACK) {

 companion object {
 private val DIFF_CALLBACK = object : DiffUtil.ItemCallback<Note>() {
 override fun areItemsTheSame(oldItem: Note, newItem: Note): Boolean {
 return oldItem.id == newItem.id
 }

 override fun areContentsTheSame(oldItem: Note, newItem: Note): Boolean {
 return oldItem.title == newItem.title && oldItem.description == newItem.description
 && oldItem.priority == newItem.priority
 }
 }
 }

 private var listener: OnItemClickListener? = null

 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): NoteHolder {
 val itemView: View = LayoutInflater.from(parent.context).inflate(R.layout.note_item, parent, false)
 return NoteHolder(itemView)
 }

 override fun onBindViewHolder(holder: NoteHolder, position: Int) {
 val currentNote: Note = getItem(position)

 holder.textViewTitle.text = currentNote.title
 holder.textViewPriority.text = currentNote.priority.toString()
 holder.textViewDescription.text = currentNote.description
 }

 fun getNoteAt(position: Int): Note {
 return getItem(position)
 }

 inner class NoteHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
 init {
 itemView.setOnClickListener {
 val position = adapterPosition
 if (position != RecyclerView.NO_POSITION) {
 listener?.onItemClick(getItem(position))
 }
 }
 }

 var textViewTitle: TextView = itemView.text_view_title
 var textViewPriority: TextView = itemView.text_view_priority
 var textViewDescription: TextView = itemView.text_view_description
 }

 interface OnItemClickListener {
 fun onItemClick(note: Note)
 }

 fun setOnItemClickListener(listener: OnItemClickListener) {
 this.listener = listener
 }
}

```

16. Buat sebuah menu untuk menu utama pada menu/main\_menu.xml

```

<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

```

```

<ns0:menu xmlns:app="http://schemas.android.com/apk/res-auto">
 <item
 android:id="@+id/delete_all_notes"
 android:title="Hapus Semua Catatan"
 app:showAsAction="never" />
</menu>

```

17. Buat sebuah menu untuk menu tambah pada menu/add\_note\_menu.xml

```

<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto">
 <item
 android:id="@+id/save_note"
 android:icon="@drawable/ic_save"
 android:title="Simpan"
 app:showAsAction="ifRoom">
 </item>
</menu>

```

18. Buat sebuah activity untuk menambah data, dengan nama activity\_add\_note.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:orientation="vertical"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:padding="16dp">

 <EditText
 android:id="@+id/edit_text_title"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_marginTop="8dp"
 android:hint="Judul"
 android:inputType="text"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent" />

 <EditText
 android:id="@+id/edit_text_description"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_marginTop="16dp"
 android:hint="Deskripsi"
 android:inputType="textMultiLine"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/edit_text_title" />

 <TextView
 android:id="@+id/textView"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Prioritas:"
 android:textAppearance="@android:style/TextAppearance.Medium"
 app:layout_constraintBottom_toBottomOf="@+id/number_picker_priority"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="@+id/number_picker_priority" />

 <NumberPicker
 android:id="@+id/number_picker_priority"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_marginStart="24dp"
 android:layout_marginTop="8dp"
 app:layout_constraintStart_toEndOf="@+id/textView"
 app:layout_constraintTop_toBottomOf="@+id/edit_text_description" />

```

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

## 19. Buat kelas Kotlinnya AddEditNoteActivity.kt

```
class AddEditNoteActivity : AppCompatActivity() {
 companion object {
 const val EXTRA_ID = "com.piussanggoro.notesapp.EXTRA_ID"
 const val EXTRA_JUDUL = "com.piussanggoro.notesapp.EXTRA_JUDUL"
 const val EXTRA_DESKRIPSI = "com.piussanggoro.notesapp.EXTRA_DESKRIPSI"
 const val EXTRA_PRIORITAS = "com.piussanggoro.notesapp.EXTRA_PRIORITAS"
 }

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

 number_picker_priority.minValue = 1
 number_picker_priority.maxValue = 5

 supportActionBar?.setHomeAsUpIndicator(R.drawable.ic_close_black_24dp)

 if (intent.hasExtra(EXTRA_ID)) {
 title = "Edit Catatan"
 edit_text_title.setText(intent.getStringExtra(EXTRA_JUDUL))
 edit_text_description.setText(intent.getStringExtra(EXTRA_DESKRIPSI))
 number_picker_priority.value = intent.getIntExtra(EXTRA_PRIORITAS, 1)
 } else {
 title = "Tambah Catatan"
 }
 }

 override fun onCreateOptionsMenu(menu: Menu?): Boolean {
 menuInflater.inflate(R.menu.add_note_menu, menu)
 return true
 }

 override fun onOptionsItemSelected(item: MenuItem?): Boolean {
 return when (item?.itemId) {
 R.id.save_note -> {
 saveNote()
 true
 }
 else -> super.onOptionsItemSelected(item)
 }
 }

 private fun saveNote() {
 if (edit_text_title.text.toString().trim().isBlank() || edit_text_description.text.toString().trim().isBlank()) {
 Toast.makeText(this, "Catatan kosong!", Toast.LENGTH_SHORT).show()
 return
 }

 val data = Intent().apply {
 putExtra(EXTRA_JUDUL, edit_text_title.text.toString())
 putExtra(EXTRA_DESKRIPSI, edit_text_description.text.toString())
 putExtra(EXTRA_PRIORITAS, number_picker_priority.value)
 if (intent.getIntExtra(EXTRA_ID, -1) != -1) {
 putExtra(EXTRA_ID, intent.getIntExtra(EXTRA_ID, -1))
 }
 }

 setResult(Activity.RESULT_OK, data)
 finish()
 }
}
```

## 20. Daftarkan activity tersebut di AndroidManifest.xml

```
<activity android:name=".AddEditNoteActivity"
 android:parentActivityName=".MainActivity">
</activity>
```

## 21. Pada tag MainActivity, tambahkan juga launchMode pad, sehingga menjadi

```
<activity android:name=".MainActivity"
 android:launchMode="singleTop">
<intent-filter>
 <action android:name="android.intent.action.MAIN"/>

 <category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
```

## 22. Ubahlah activity\_main.xml sehingga menjadi sebagai berikut. Tambahkan/ubah atribut yang belum ada (dikenali).

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:orientation="vertical"
```

```

 android:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context=".MainActivity">

 <androidx.recyclerview.widget.RecyclerView
 android:id="@+id/recycler_view"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:background="@color/lightGray"
 tools:itemCount="5"
 tools:listitem="@layout/note_item" />

 <com.google.android.material.floatingactionbutton.FloatingActionButton
 android:id="@+id/buttonAddNote"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:src="@drawable/ic_add_black_24dp"
 android:layout_marginEnd="16dp"
 android:layout_marginBottom="16dp"
 android:layout_gravity="bottom|right"/>

</androidx.coordinatorlayout.widget.CoordinatorLayout>

```

23. Ubah MainActivity sehingga menjadi sebagai berikut

```

class MainActivity : AppCompatActivity() {

 companion object {
 const val ADD_NOTE_REQUEST = 1
 const val EDIT_NOTE_REQUEST = 2
 }

 private lateinit var noteViewModel: NoteViewModel

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

 buttonAddNote.setOnClickListener {
 startActivityForResult(
 Intent(this, AddEditNoteActivity::class.java),
 ADD_NOTE_REQUEST
)
 }
 }

 recycler_view.layoutManager = LinearLayoutManager(this)
 recycler_view.setHasFixedSize(true)

 val adapter = NoteAdapter()
 recycler_view.adapter = adapter

 noteViewModel = ViewModelProviders.of(this).get(NoteViewModel::class.java)
 noteViewModel.getAllNotes().observe(this, Observer<List<Note>> {
 adapter.submitList(it)
 })

 ItemTouchHelper(object :
 ItemTouchHelper.SimpleCallback(0, ItemTouchHelper.LEFT.or(ItemTouchHelper.RIGHT)) {
 override fun onMove(
 recyclerView: RecyclerView,
 viewHolder: RecyclerView.ViewHolder,
 target: RecyclerView.ViewHolder
): Boolean {
 return false
 }

 override fun onSwiped(viewHolder: RecyclerView.ViewHolder, direction: Int) {
 noteViewModel.delete(adapter.getNoteAt(viewHolder.adapterPosition))
 Toast.makeText(baseContext, "Catatan dihapus!", Toast.LENGTH_SHORT).show()
 }
 }).attachToRecyclerView(recycler_view)

 adapter.setOnItemClickListener(object : NoteAdapter.OnItemClickListener {
 override fun onClick(note: Note) {
 val intent = Intent(baseContext, AddEditNoteActivity::class.java)
 intent.putExtra(AddEditNoteActivity.EXTRA_ID, note.id)
 intent.putExtra(AddEditNoteActivity.EXTRA_JUDUL, note.title)
 intent.putExtra(AddEditNoteActivity.EXTRA_DESKRIPSI, note.description)
 }
 })
}

```

```

 intent.putExtra(AddEditNoteActivity.EXTRA_PRIORITY, note.priority)
 startActivityForResult(intent, EDIT_NOTE_REQUEST)
 }
}

override fun onCreateOptionsMenu(menu: Menu?): Boolean {
 menuInflater.inflate(R.menu.main_menu, menu)
 return true
}

override fun onOptionsItemSelected(item: MenuItem?): Boolean {
 return when (item?.itemId) {
 R.id.delete_all_notes -> {
 noteViewModel.deleteAllNotes()
 Toast.makeText(this, "Semua sudah dihapus!", Toast.LENGTH_SHORT).show()
 true
 }
 else -> {
 super.onOptionsItemSelected(item)
 }
 }
}

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
 super.onActivityResult(requestCode, resultCode, data)

 if (requestCode == ADD_NOTE_REQUEST && resultCode == Activity.RESULT_OK) {
 val newNote = Note(
 data!!.getStringExtra(AddEditNoteActivity.EXTRA_JUDUL),
 data.getStringExtra(AddEditNoteActivity.EXTRA_DESKRIPSI),
 data.getIntExtra(AddEditNoteActivity.EXTRA_PRIORITY, 1)
)
 noteViewModel.insert(newNote)
 Toast.makeText(this, "Catatan disimpan!", Toast.LENGTH_SHORT).show()
 } else if (requestCode == EDIT_NOTE_REQUEST && resultCode == Activity.RESULT_OK) {
 val id = data?.getIntExtra(AddEditNoteActivity.EXTRA_ID, -1)

 if (id == -1) {
 Toast.makeText(this, "Pembaharuan gagal!", Toast.LENGTH_SHORT).show()
 }

 val updateNote = Note(
 data!!.getStringExtra(AddEditNoteActivity.EXTRA_JUDUL),
 data.getStringExtra(AddEditNoteActivity.EXTRA_DESKRIPSI),
 data.getIntExtra(AddEditNoteActivity.EXTRA_PRIORITY, 1)
)
 updateNote.id = data.getIntExtra(AddEditNoteActivity.EXTRA_ID, -1)
 noteViewModel.update(updateNote)
 } else {
 Toast.makeText(this, "Catatan tidak disimpan!", Toast.LENGTH_SHORT).show()
 }
}
}

```

24. Jalankan dan amati hasilnya



### LATIHAN

---

- Modifikasilah aplikasi dengan menambahkan detil data pemilih nomor telepon.



### TUGAS

---

- Buat aplikasi baru dengan mengembangkan project diatas



## REFERENSI

---

1. <https://kotlinlang.org/docs/reference/>
2. <https://developer.android.com/kotlin>
3. <https://developer.android.com/courses/kotlin-android-fundamentals/toc>
4. <https://codelabs.developers.google.com/android-kotlin-fundamentals/>
5. <https://developer.android.com/kotlin/learn>
6. <https://developer.android.com/kotlin/resources>

# **MODUL 11**

## **Mengambil Data dan menampilkan gambar dari Internet**



### **CAPAIAN PEMBELAJARAN**

---

1. Mahasiswa dapat membuat aplikasi Koneksi dengan layanan backend server.



### **KEBUTUHAN ALAT/BAHAN/SOFTWARE**

---

1. Android Studio 3.4.
2. Handphone Android versi 7.0 (Nougat)
3. Kabel data USB.
4. Driver ADB.



### **DASAR TEORI**

---

#### **1. Retrofit (digunakan untuk koneksi ke API web service)**

##### **Apa itu Retrofit?**

<https://code.tutsplus.com/id/tutorials/sending-data-with-retrofit-2-http-client-for-android--cms-27845>

Retrofit adalah client HTTP type-safe untuk Android dan Java. Retrofit memudahkan untuk terhubung ke layanan web REST dengan menerjemahkan API ke dalam antarmuka Java. Kita akan melihat salah satu library HTTP yang paling populer dan sering direkomendasikan untuk Android. Library yang kuat ini mempermudah penggunaan data JSON atau XML, yang kemudian diurai menjadi Plain Old Java Objects (POJOs).

Permintaan GET, POST, PUT, PATCH, dan DELETE semua bisa dieksekusi. Seperti kebanyakan perangkat lunak open-source, Retrofit dibangun di atas beberapa library dan alat bantu lainnya. Di balik layar, Retrofit memanfaatkan OkHttp (dari pengembang yang sama) untuk menangani permintaan jaringan. Selain itu, Retrofit tidak memiliki konverter JSON built-in untuk mengurai dari objek JSON ke Java. Sebagai gantinya, ini mendukung library konverter JSON untuk menangani hal itu:

Gson: com.squareup.retrofit:converter-gson  
Jackson: com.squareup.retrofit:converter-jackson  
Moshi: com.squareup.retrofit:converter-moshi

Untuk Protocol buffers, Retrofit mendukung:

Protobuf: com.squareup.retrofit2:converter-protobuf  
Wire: com.squareup.retrofit2:converter-wire

dan untuk XML Retrofit, mendukung:

Simple Framework: com.squareup.retrofit2:converter-simpleframework

## Jadi Mengapa Menggunakan Retrofit?

Mengembangkan library HTTP type-safe sendiri untuk berinteraksi dengan API REST bisa menjadi kesulitan yang nyata: kita harus menangani banyak aspek, seperti membuat koneksi, caching, mencoba kembali permintaan yang gagal, threading, parsing respons, penanganan kesalahan, dan banyak lagi. Retrofit, di sisi lain, adalah library yang terencana, terdokumentasi dan teruji yang akan menghemat banyak waktu dan tenaga.

Kita akan pelajari cara menggunakan Retrofit 2 untuk menangani permintaan jaringan dengan Retrofit 2 untuk menangani permintaan jaringan dengan membangun aplikasi sederhana yang akan menjalankan permintaan POST, permintaan PUT (untuk memperbarui entitas), dan permintaan DELETE. Kita juga akan pelajari cara mengintegrasikannya dengan RxJava dan cara membatalkan permintaan. Kita akan menggunakan API yang disediakan oleh JSONPlaceholder - ini adalah API REST online palsu untuk pengujian dan pembuatan prototipe.

## Glide (untuk download gambar (caching) )

### Apa itu Glide?

<https://code.tutsplus.com/id/tutorials/code-an-image-gallery-android-app-with-glide--cms-28207>

Glide adalah sumber terbuka library Android yang populer untuk memuat gambar, video, dan GIF animasi. Dengan Glide, anda dapat memuat dan menampilkan media dari berbagai sumber, seperti server jarak jauh atau sistem file lokal. Secara default, Glide menggunakan penerapan khusus **HttpURLConnection** untuk memuat gambar melalui internet. Namun, Glide juga menyediakan plugin ke pustaka jaringan populer lainnya seperti **Volley** atau **OkHttp**.

### Jadi mengapa menggunakan glide?

Mengembangkan fungsi pemuat dan tampilan media kita sendiri di Java bisa menjadi keribetan yang nyata: kita harus mengurus cache, decoding, mengelola koneksi jaringan,

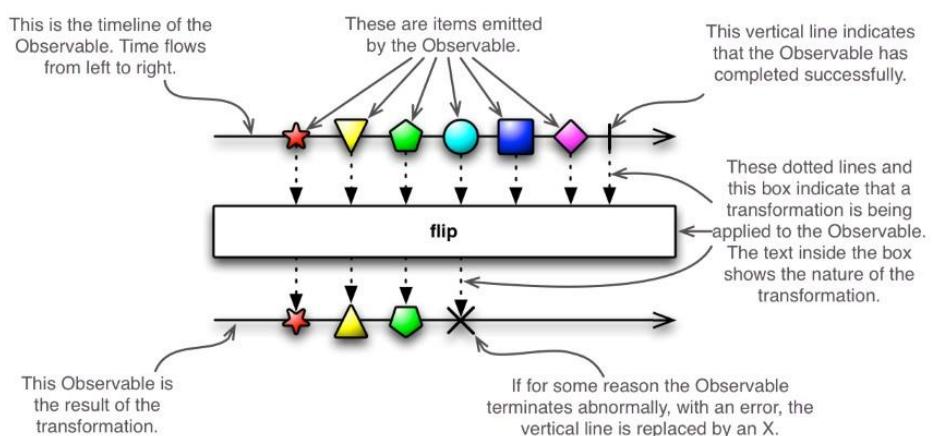
threading, exception, dan banyak lagi. Glide adalah library yang mudah digunakan, terencana dengan baik, terdokumentasi dengan baik, dan benar-benar teruji yang dapat menghemat banyak waktu berharga.

## Reactivex (untuk mendukung mvvm)

### Apa itu RxJava?

<https://code.tutsplus.com/id/tutorials/getting-started-with-rxjava-20-for-android--cms-28345>

RxJava adalah library yang memungkinkan kita membuat aplikasi dalam gaya pemrograman reaktif. Pada intinya, pemrograman reaktif menyediakan cara pemrosesan dan reaksi yang bersih dan efisien terhadap arus data real-time, termasuk data dengan nilai dinamis. Aliran data ini belum tentu harus mengambil bentuk tipe data tradisional, karena RxJava cukup banyak memperlakukan semuanya sebagai arus data - mulai dari variabel hingga properti, cache, dan bahkan events masukan pengguna seperti klik dan gesekan. Data yang dipancarkan oleh setiap aliran dapat berupa nilai, kesalahan, atau sinyal "komplit", walaupun kita tidak perlu menerapkan dua hal terakhir. Setelah membuat aliran pemancar data, kita menggabungkannya dengan objek reaktif yang mengkonsumsi dan kemudian bertindak berdasarkan data ini, melakukan tindakan yang berbeda bergantung pada apa yang telah dipancarkan oleh arus. RxJava meliputi sejumlah operator yang berguna untuk bekerja dengan aliran, sehingga mudah untuk melakukan hal-hal seperti filtering, pemetaan, menunda, penghitungan, dan banyak lagi.



Untuk membuat alur data stream dan objek yang bereaksi terhadapnya, RxJava memperluas pola perancangan perangkat lunak Observer. Intinya, di RxJava, kita miliki objek Observable yang memancarkan aliran data dan kemudian berhenti, dan objek Observer yang berlangganan menjadi Observable. Observer menerima pemberitahuan setiap kali mereka menugaskan Observable memancarkan nilai, kesalahan, atau sinyal yang telah komplit.

Jadi pada tingkat yang sangat tinggi, RxJava adalah tentang:

- Membuat Observable.

- Memberikan beberapa data Observable untuk dipancarkan.
- Membuat Observer.
- Menetapkan Observer menjadi Observable.
- Memberi tugas Observer untuk melakukan setiap kali menerima emisi dari Observable yang ditugaskan.

## Mengapa RxJava?

Belajar teknologi baru membutuhkan waktu dan usaha, dan karena library berorientasi data, RxJava tidak selalu menjadi API yang paling mudah untuk diatasi. Untuk membantu kita memutuskan apakah belajar RxJava layak untuk investasi awal, mari jelajahi beberapa manfaat utama untuk menambahkan library RxJava ke proyek Android kita.

**Lebih ringkas, kode yang dapat dibaca.** Kode yang rumit, verbose dan sulit dibaca adalah *selalu* kabar buruk. Kode Messy lebih rentan terhadap bug dan inefisiensi lainnya, dan jika ada kesalahan yang terjadi maka kita akan memiliki waktu yang jauh lebih sulit untuk melacak sumber kesalahan ini jika kode kita berantakan. Bahkan jika proyek kita tidak dibangun tanpa ada kesalahan, kode yang rumit masih bisa kembali menghantui kita biasanya saat kita memutuskan untuk merilis pembaruan. RxJava menyederhanakan kode yang diperlukan untuk menangani data dan events dengan memungkinkan kita mendeskripsikan apa yang ingin kita capai, daripada menulis daftar instruksi untuk aplikasi kita. RxJava juga menyediakan alur kerja sekitar yang dapat kita gunakan untuk menangani semua data dan events di seluruh aplikasi kita-buatlah Observable, buat Observer, menetapkan observable ke observer, rinsen dan ulangi. Pendekatan formulatik ini membuat kode yang mudah dibaca dan mudah dibaca manusia.

**Multithreading Made Easy.** Aplikasi android modern harus bisa *multitask*. Paling tidak, pengguna kita akan berharap untuk dapat terus berinteraksi dengan UI saat aplikasi kita melakukan beberapa pekerjaan di latar belakang, seperti mengelola koneksi jaringan, mendownload file, atau bermain musik. Masalahnya adalah bahwa Android adalah single-threaded secara default, jadi jika aplikasi kita pernah berhasil multi-task maka kita perlu membuat beberapa thread tambahan. Android memang menyediakan sejumlah cara untuk membuat thread tambahan, seperti layanan dan IntentServices, namun tidak satupun dari solusi ini sangat mudah diterapkan, dan mereka dapat dengan cepat menghasilkan kompleks, kode verbose yang rentan terhadap kesalahan. RxJava bertujuan untuk menghilangkan kesulitan dari pembuatan aplikasi Android multi-threaded, dengan menyediakan penjadwal dan operator khusus. RxJava memberi kita cara mudah untuk menentukan thread dimana pekerjaan harus dilakukan dan thread di mana hasil dari pekerjaan ini harus diposting. RxJava 2.0 mencakup sejumlah penjadwal secara default, termasuk Schedulers.newThread, yang *terutama* berguna karena menciptakan thread baru. Untuk mengubah thread tempat di mana pekerjaan dilakukan, kita hanya perlu mengubah tempat observer berlangganan ke observable,

dengan menggunakan operator subscribeOn. Sebagai contoh, di sini kita membuat thread baru dan menentukan bahwa pekerjaan harus dilakukan pada thread baru ini:

```
observable.subscribeOn(Schedulers.newThread())
```

Masalah lama lainnya dengan multithreading di Android adalah kita hanya dapat memperbarui UI aplikasi kita dari untaian utama. Biasanya, kapan pun kita perlu mengeposkan hasil beberapa karya latar ke UI aplikasi kita, kita harus membuat Handler yang berdedikasi. Sekali lagi, RxJava memiliki solusi yang jauh lebih mudah. Kita bisa menggunakan operator observeOn untuk menentukan bahwa Observable harus mengirimkan notifikasinya menggunakan scheduler yang berbeda, yang pada dasarnya memungkinkan kita mengirim data Observable kita ke thread pilihan kita, termasuk thread UI utama. Ini berarti bahwa dengan hanya dua baris kode, kita dapat membuat thread baru dan mengirim hasil kerja yang dilakukan di thread ini ke thread utama Android:

```
.subscribeOn(Schedulers.newThread())
.observeOn(AndroidSchedulers.mainThread())
```

**Peningkatan Fleksibilitas.** Observables memancarkan data mereka dengan cara yang sepenuhnya menyembunyikan cara data dibuat. Karena observers kita bahkan tidak dapat melihat bagaimana data dibuat, kita bebas untuk menerapkan Observables kita dengan cara apapun yang kita inginkan. Setelah kita menerapkan Observables kita, RxJava menyediakan operator dalam jumlah besar yang dapat kita gunakan untuk menyaring, menggabungkan dan mengubah data yang sedang dipancarkan oleh Observables. Kita bahkan bisa menjerumuskan lebih dan lebih banyak operator bersamaan sampai kita menciptakan aliran data yang tepat dibutuhkan aplikasi kita. Misalnya, kita dapat menggabungkan data dari beberapa aliran, memfilter arus yang baru digabungkan, lalu menggunakan data yang dihasilkan sebagai masukan untuk arus data berikutnya. Dan ingat bahwa di RxJava, hampir semua hal diperlakukan sebagai aliran data, jadi kita bahkan dapat menerapkan operator ini ke "data" non-tradisional, seperti events klik.

**Membuat Aplikasi yang Lebih Responsif.** Gone adalah hari-hari ketika sebuah aplikasi bisa lolos dengan memuat halaman konten dan kemudian menunggu pengguna untuk mengetuk tombol **Berikutnya**. Saat ini, aplikasi seluler khas kita harus dapat bereaksi terhadap berbagai peristiwa dan data yang terus berkembang, idealnya secara real time. Misalnya, aplikasi jejaring sosial khas kita harus terus mendengarkan keinginan masuk, komentar, dan permintaan pertemanan, sambil mengelola koneksi jaringan di latar belakang dan merespon segera kapan pun pengguna mengetuk atau menggesek layar. Library RxJava dirancang untuk dapat mengelola berbagai data dan events secara bersamaan dan secara real time, menjadikannya alat yang ampuh untuk menciptakan jenis aplikasi yang sangat responsif yang diharapkan oleh pengguna ponsel modern.



## PRAKTIK

1. Kita akan membuat aplikasi untuk mengambil data dari Internet dan menampilkan gambar hasil ke view.
2. Atur gradle Module.app. Tambahkan

```
apply plugin: 'kotlin-kapt'
```

```
def lifeCycleExtensionVersion = '1.1.1'
def supportVersion = '28.0.0'
def retrofitVersion = '2.3.0'
def glideVersion = '4.8.0'
def rxJavaVersion = '2.0.1'

dependencies {
 implementation fileTree(dir: 'libs', include: ['*.jar'])
 implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
 implementation 'com.android.support:appcompat-v7:28.0.0'
 implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-core:1.2.1'

 //retrofit
 implementation "com.squareup.retrofit2:retrofit:$retrofitVersion"
 implementation "com.squareup.retrofit2:converter-gson:$retrofitVersion"
 implementation "com.squareup.retrofit2:adapter-rxjava2:$retrofitVersion"

 //rxJava
 implementation "io.reactivex.rxjava2:rxjava:$rxJavaVersion"
 implementation "io.reactivex.rxjava2:rxandroid:$rxJavaVersion"

 //Glide
 implementation "com.github.bumptech.glide:glide:$glideVersion"

 //ui
 implementation "com.android.support:appcompat-v7:$supportVersion"
 implementation "android.arch.lifecycle:extensions:$lifeCycleExtensionVersion"
 implementation 'com.android.support:cardview-v7:28.0.0'
 implementation "com.android.support:recyclerview-v7:$supportVersion"
 implementation 'com.android.support.constraint:constraint-layout:1.1.3'
}
```

3. Buat sebuah project, kemudian buat 4 buah package : **api**, **model**, **view**, **viewmodel**
4. Pindahkan MainActivity.kt ke dalam package **view**.
5. Buat interface data **PhotosApi** di dalam package **api**

```
interface PhotosApi {
 @GET("photos")
 fun getPhotos(): Single<List<Photo>>
}
```

6. Buat kelas **PhotosService** untuk membuat instance dari retrofit, masih di package **api**

```

class PhotosService {
 private val BASE_URL = "https://jsonplaceholder.typicode.com/"
 private val api: PhotosApi

 init {
 api = Retrofit.Builder()
 .baseUrl(BASE_URL)
 .addConverterFactory(GsonConverterFactory.create())
 .addCallAdapterFactory(RxJava2CallAdapterFactory.create())
 .build()
 .create(PhotosApi::class.java)
 }

 fun getPhotos(): Single<List<Photo>> {
 return api.getPhotos()
 }
}

```

7. Buat kelas data **Photo** untuk menghubungkan model ke android, di package model

```

data class Photo(
 @SerializedName("id")
 val id: Int?,
 @SerializedName("title")
 val title: String?,
 @SerializedName("thumbnailUrl")
 val thumbnail: String?
)

```

8. Buat kelas PhotoListAdapter di package view

```

class PhotoListAdapter(var photos: ArrayList<Photo>) :
 RecyclerView.Adapter<PhotoListAdapter.ViewHolder>() {

 fun updatePhotos(newPhotos: List<Photo>) {
 photos.clear()
 photos.addAll(newPhotos)
 notifyDataSetChanged()
 }

 override fun onCreateViewHolder(parent: ViewGroup, p1: Int) = ViewHolder(
 LayoutInflater.from(parent.context).inflate(R.layout.item_list,
 parent, false)
)

 override fun getItemCount() = photos.size

 override fun onBindViewHolder(holder: ViewHolder, position: Int) {
 holder.bind(photos[position])
 }

 class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {
 fun bind(photos: Photo) {
 itemView.tvTitle.text = photos.title
 itemView.setOnClickListener { view ->
 Toast.makeText(itemView.context, "Hello", Toast.LENGTH_LONG).show()
 }
 Glide.with(itemView.context).load(photos.thumbnail).into(itemView.imageView)
 }
 }
}

```

```
 }
```

## 9. Buat kelas ListViewMode di package viewmodel

```
class ListViewModel : ViewModel() {
 private val photosService = PhotosService()
 private val disposable = CompositeDisposable()
 val photos = MutableLiveData<List<Photo>>()

 fun fetchData() {
 disposable.add(
 photosService.getPhotos()
 .subscribeOn(Schedulers.newThread())
 .observeOn(AndroidSchedulers.mainThread())
 .subscribeWith(object :
 DisposableSingleObserver<List<Photo>>() {
 override fun onSuccess(value: List<Photo>?) {
 photos.value = value
 }
 override fun onError(e: Throwable?) {
 Log.e("ERRORFETCHDATA", "error$e")
 }
 })
)
 }

 override fun onCleared() {
 super.onCleared()
 disposable.clear()
 }
}
```

## 10. Tambahkan komponen **RecyclerView** pada layout **activity\_main.xml**

```
<Android.support.v7.widget.RecyclerView
 android:id="@+id/rv_list"
 android:layout_width="0dp"
 android:layout_height="0dp"
 app:layout_constraintEnd_toEndOf="parent"
 android:layout_marginEnd="8dp"
 app:layout_constraintStart_toStartOf="parent"
 android:layout_marginStart="8dp"
 android:layout_marginTop="8dp"
 app:layout_constraintTop_toTopOf="parent"
 android:layout_marginBottom="8dp"
 app:layout_constraintBottom_toBottomOf="parent"/>
```

## 11. Buat file layout dengan nama **item\_list.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:orientation="horizontal"
 android:layout_marginBottom="8dp"
 android:layout_height="wrap_content">

 <Android.support.v7.widget.CardView
 android:layout_width="match_parent"
```

```

 android:layout_height="wrap_content"
 tools:layout_editor_absoluteX="8dp"
 app:layout_constraintTop_toTopOf="parent">
 <android.support.constraint.ConstraintLayout
 android:layout_width="match_parent"
 android:padding="8dp"
 android:layout_height="wrap_content">
 <TextView
 android:text="@string/imagename"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:id="@+id/tvTitle"
 app:layout_constraintBottom_toBottomOf="@+id/imageView"

 android:textAppearance="@style/TextAppearance.AppCompat.Small"
 app:layout_constraintTop_toTopOf="@+id/imageView"
 app:layout_constraintStart_toEndOf="@+id/imageView"
 android:layout_marginStart="8dp"
 />
 <ImageView
 android:layout_width="80dp"
 android:scaleType="fitXY"
 android:layout_height="80dp"
 tools:srcCompat="@tools:sample/avatars"
 android:id="@+id/imageView"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent"
 app:layout_constraintBottom_toBottomOf="parent"/>

 </android.support.constraint.ConstraintLayout>
 </android.support.v7.widget.CardView>
</LinearLayout>

```

## 12. Tambahkan resource string berikut

```
<string name="imagename">imagename</string>
```

## 13. Ubah MainActivity sehingga menjadi sebagai berikut

```

class MainActivity : AppCompatActivity() {

 lateinit var viewModel: ListViewModel
 private val photoListAdapter = PhotoListAdapter(arrayListOf())

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

 viewModel = ViewModelProviders.of(this).get(ListViewModel::class.java)
 viewModel.fetchData()

 rv_list.apply {
 layoutManager = LinearLayoutManager(context)
 adapter = photoListAdapter
 }
 observeViewModel()
 }

 fun observeViewModel() {
 viewModel.photos.observe(this, Observer { photos ->
 photos?.let {
 photoListAdapter.updatePhotos(it)
 }
 })
 }
}

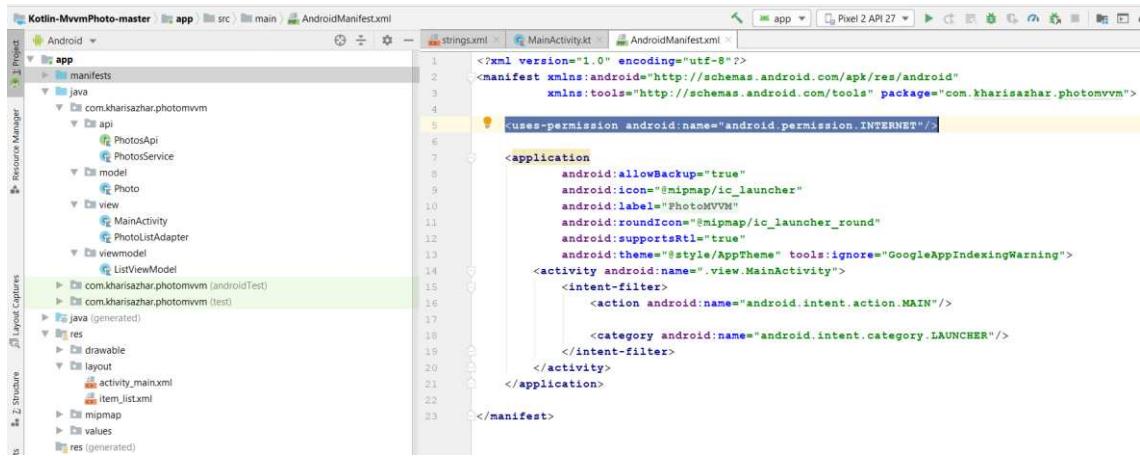
```

```
 })
 }
}
```

14. Berikan permission untuk internet access pada **AndroidManifest**

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

15. Susunan akhir file menjadi sebagai berikut. Perhatikan di **AndroidManifest**. Untuk activity dari **MainActivity** berada pada package **view**.



The screenshot shows the Android Studio interface with the project 'Kotlin-MvvmPhoto-master' open. The left sidebar shows the project structure under 'app': manifest, java (com.kharisazhar.photomvvm), res, and generated files. The 'manifests' folder contains 'AndroidManifest.xml'. The right pane shows the XML code for 'AndroidManifest.xml'. A yellow highlight is placed over the line containing the `<uses-permission android:name="android.permission.INTERNET"/>` tag. The code is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools" package="com.kharisazhar.photomvvm">
 <uses-permission android:name="android.permission.INTERNET"/>

 <application
 android:allowBackup="true"
 android:icon="@mipmap/ic_launcher"
 android:label="PhotoVM"
 android:roundIcon="@mipmap/ic_launcher_round"
 android:supportsRtl="true"
 android:theme="@style/AppTheme" tools:ignore="GoogleAppIndexingWarning">
 <activity android:name=".view.MainActivity">
 <intent-filter>
 <action android:name="android.intent.action.MAIN"/>
 <category android:name="android.intent.category.LAUNCHER"/>
 </intent-filter>
 </activity>
 </application>
</manifest>
```

16. Jalankan dan amati hasilnya.

17. Analisislah hasil tampilan tersebut alurnya bagaimana.



## LATIHAN

1. Modifikasilah aplikasi dengan menambahkan detil data yang lain. Akses api dari alamat web nya



## TUGAS

1. Buat aplikasi baru dengan mengembangkan project di atas



## REFERENSI

---

1. <https://kotlinlang.org/docs/reference/>
2. <https://developer.android.com/kotlin>
3. <https://developer.android.com/courses/kotlin-android-fundamentals/toc>
4. <https://codelabs.developers.google.com/android-kotlin-fundamentals/>
5. <https://developer.android.com/kotlin/learn>
6. <https://developer.android.com/kotlin/resources>

## **MODUL 12**

### **Styles dan themes, material design, dimens dan colors**



#### **CAPAIAN PEMBELAJARAN**

---

Mahasiswa dapat membuat desain aplikasi sederhana secara mandiri.



#### **KEBUTUHAN ALAT/BAHAN/SOFTWARE**

---

1. Android Studio 3.4.
2. Handphone Android versi 7.0 (Nougat)
3. Kabel data USB.
4. Driver ADB.



#### **DASAR TEORI**

---

Apa itu Material Design dan Material Component untuk Android?

Material Design adalah sistem untuk membangun produk digital yang berani dan indah. Dengan menyatukan gaya, branding, interaksi, dan gerak di bawah seperangkat prinsip dan komponen yang konsisten, tim produk dapat mewujudkan potensi desain terbesar mereka.

Untuk aplikasi Android, **Material Components for Android (MDC Android)** menyatukan desain dan rekayasa dengan library komponen untuk menciptakan konsistensi di seluruh aplikasi. Ketika sistem Desain Bahan berkembang, komponen-komponen ini diperbarui untuk memastikan implementasi pixel-konsisten yang konsisten dan kepatuhan terhadap standar pengembangan front-end Google. MDC juga tersedia untuk web, iOS, dan Flutter.

#### **Material Components for Android (MDC Android).**

MDC TextField. Fitur MDC Text Field meliputi:

- Menampilkan built-in error feedback
- Mendukung toggle untuk visibilitas password menggunakan app: passwordToggleEnabled
- Menawarkan built-in helper text functionality menggunakan app: helperText
- Menampilkan jumlah karakter total dan maks menggunakan app: counterEnabled dan app: counterMaxLength

MDC Button. Fitur MDC Button meliputi:

- Built-in touch feedback (disebut MDC Ripple) secara default
- Default elevation
- Customizable corner radius and stroke

AppBarLayout.

AppBarLayout adalah LinearLayout vertikal yang mengimplementasikan banyak fitur konsep desain material app bar, yaitu scrolling gestures. Children harus menyediakan perilaku pengguliran yang diinginkan melalui setScrollFlags (int) dan atribut layout xml terkait: app: layout\_scrollFlags. Pandangan ini sangat bergantung pada apa yang digunakan sebagai anak langsung dalam CoordinatorLayout. Jika kita menggunakan AppBarLayout di dalam ViewGroup yang berbeda, sebagian besar fungsinya tidak akan berfungsi. AppBarLayout juga membutuhkan saudara scrolling yang terpisah untuk mengetahui kapan harus scrolling. Pengikatan dilakukan melalui kelas perilaku AppBarLayout.ScrollingViewBehavior, yang berarti bahwa kita harus mengatur perilaku tampilan scrolling menjadi turunan dari AppBarLayout.ScrollingViewBehavior. Resource string yang berisi nama kelas lengkap tersedia.

MaterialCardView.

Kelas ini memasok gaya Material untuk card dalam konstruktor. Widget akan menampilkan style Material default yang benar tanpa menggunakan style flag. Lebar Stroke dapat diatur menggunakan atribut strokeWidth. Atur warna stroke menggunakan atribut strokeColor. Tanpa strokeColor, kartu tidak akan memberikan batas garis, terlepas dari nilai strokeWidth. Card mengimplementasikan Checkable, cara default untuk beralih ke android: checked\_state tidak disediakan. Klien harus memanggil isChecked (boolean). Ini menunjukkan aplikasi: checkedIcon dan mengubah warna overlay. Card juga memiliki status khusus yang dimaksudkan untuk digunakan ketika card adalah aplikasi yang dapat diseret: dragged\_state. Ini digunakan dengan memanggil setDragged (boolean). Ini mengubah warna overlay dan mengangkat kartu untuk menyampaikan gerakan. Catatan: Hindari pengaturan setClipToOutline (boolean) menjadi true. Ada tampilan perantara untuk klip konten, pengaturan ini akan memiliki konsekuensi kinerja negatif. Hirarki tampilan aktual yang ada di bawah MaterialCardView TIDAK dijamin untuk cocok dengan hierarki tampilan yang ditulis dalam XML. Akibatnya, panggilan ke getParent () pada anak-anak dari MaterialCardView, tidak akan mengembalikan MaterialCardView itu sendiri, melainkan tampilan perantara. Jika kita perlu mengakses MaterialCardView secara langsung, atur android: id dan gunakan findViewById (int).



## PRAKTIK

---

<https://codelabs.developers.google.com/codelabs/mdc-101-kotlin/#0>

1. Kita akan membuat aplikasi yang memanfaatkan material design.
2. Untuk Project awal, gunakan project yang sudah ada, kemudian jalankan. Anda akan menemukan tampilan sebagai berikut.



3. Selanjutnya kita akan menambahkan textField. Kita akan menggunakan komponen textField MDC, yang mencakup fungsi bawaan yang menampilkan floating label and error message.
4. Buka file shr\_login\_fragment.xml dari resource layout, kemudian tambahkan dua elemen TextInputLayout dengan child TextInputEditText di dalam <LinearLayout>, di bawah label "SHRINE" <TextView>

```

<com.google.android.material.textfield.TextInputLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_margin="4dp"
 android:hint="@string/shr_hint_username">

 <com.google.android.material.textfield.TextInputEditText
 android:layout_width="match_parent"
 android:layout_height="wrap_content" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
 android:id="@+id/password_text_input"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_margin="4dp"
 android:hint="@string/shr_hint_password">

 <com.google.android.material.textfield.TextInputEditText
 android:id="@+id/password_edit_text"
 android:layout_width="match_parent"
 android:layout_height="wrap_content" />
</com.google.android.material.textfield.TextInputLayout>

```

5. Tambahkan validasi input. Komponen TextInputLayout menyediakan fungsionalitas umpan balik kesalahan bawaan. Untuk menampilkan umpan balik kesalahan, lakukan perubahan berikut ke shr\_login\_fragment.xml:

Setel atribut app:errorEnabled menjadi true pada elemen Password di TextInputLayout. Ini akan menambahkan extra padding untuk pesan kesalahan di bawah text field.

Setel atribut android:inputType ke "textPassword" pada elemen Password TextInputEditText. Ini akan menyembunyikan teks input di password field.

```

<com.google.android.material.textfield.TextInputLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_margin="4dp"
 android:hint="@string/shr_hint_username">

 <com.google.android.material.textfield.TextInputEditText
 android:layout_width="match_parent"
 android:layout_height="wrap_content" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
 android:id="@+id/password_text_input"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_margin="4dp"
 android:hint="@string/shr_hint_password"
 app:errorEnabled="true">

 <com.google.android.material.textfield.TextInputEditText
 android:id="@+id/password_edit_text"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:inputType="textPassword" />
</com.google.android.material.textfield.TextInputLayout>

```

6. Jalankan dan amati hasilnya. Perhatikan animasi floating label.

- Kemudian kita akan menambahkan dua buah button untuk halaman login : "Cancel" dan "Next". Kita akan menggunakan komponen Button MDC, yang dilengkapi dengan iconic Material Design ink ripple effect built-in.
- Buka file shr\_login\_fragment.xml, tambahkan <RelativeLayout> di dalam tag <LinearLayout>, dibawa elemen TextInputLayout. Kemudian tambahkan dua buah elemen <MaterialButton> di dalam tag <RelativeLayout>.

```
<RelativeLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content">

 <com.google.android.material.button.MaterialButton
 android:id="@+id/next_button"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_alignParentEnd="true"
 android:layout_alignParentRight="true"
 android:text="@string/shr_button_next" />

 <com.google.android.material.button.MaterialButton
 android:id="@+id/cancel_button"
 style="@style/Widget.MaterialComponents.Button.TextButton"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_marginEnd="12dp"
 android:layout_marginRight="12dp"
 android:layout_toStartOf="@id/next_button"
 android:layout_toLeftOf="@id/next_button"
 android:text="@string/shr_button_cancel" />

</RelativeLayout>
```

- Jalankan dan amati hasilnya.
- Kita akan menambahkan navigasi ke Fragment berikutnya. Kita akan menambahkan beberapa kode Kotlin ke LoginFragment.kt untuk menghubungkan tombol "NEXT" kita untuk transisi ke fragmen lain. Tambahkan metode private boolean isPasswordValid di LoginFragment.kt di bawah onCreateView (), dengan logika untuk menentukan apakah kata sandi itu valid atau tidak.

```
private fun isPasswordValid(text: Editable?): Boolean {
 return text != null && text.length >= 8
}
```

- Selanjutnya kita akan menambahkan listener pada Button Next, yang menetapkan dan menghapus kesalahan berdasarkan method isPasswordValid () yang baru saja kita buat. Di onCreateView (), klik listener ini harus ditempatkan di antara inflater line dan return view line. Sekarang mari kita tambahkan key listener kunci ke password TextInputEditText untuk listen ke key events yang akan menghapus kesalahan. Listener ini juga harus menggunakan isPasswordValid() untuk memeriksa apakah password itu valid atau tidak. Tambahkan langsung di bawah click listener di onCreateView(). Method onCreateView menjadi sebagai berikut.

```
override fun onCreateView(
 inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View?
{
 // Inflate the layout for this fragment.
 val view = inflater.inflate(R.layout.shr_login_fragment, container, false)

 // Set an error if the password is less than 8 characters.
 view.next_button.setOnClickListener({
 if (!isPasswordValid(password_edit_text.text!!)) {
 password_text_input.error = getString(R.string.shr_error_password)
 } else {
 // Clear the error.
 password_text_input.error = null
 }
 })
}
```

```

 // Navigate to the next Fragment.
 (activity as NavigationHost).navigateTo(ProductGridFragment(), false)
 }

 // Clear the error once more than 8 characters are typed.
 view.password_edit_text.setOnKeyListener({ _, _, _ ->
 if (isPasswordValid(password_edit_text.text!!)) {
 // Clear the error.
 password_text_input.error = null
 }
 false
 })
 return view
}

```

12. Jalankan dan amati hasilnya.
13. Aplikasi akan kita kembangkan dengan menambahkan komponen lainnya. Menambahkan top app bar. Kita akan menambahkan navigasi untuk membantu user mengarahkan aplikasi ke lokasi yang diharapkan. Navigasi mengacu pada komponen, interaksi, isyarat visual, dan arsitektur informasi yang memungkinkan pengguna untuk bergerak seputar aplikasi. Ini membantu membuat konten dan fitur dapat ditemukan, sehingga tugas mudah diselesaikan. Material Design menawarkan pola navigasi yang memastikan tingkat kegunaan yang tinggi. Salah satu komponen yang paling terlihat adalah top app bar. Untuk menyediakan navigasi dan memberi pengguna akses cepat ke tindakan lain, kita akan tambahkan top app bar.
14. Menambahkan widget AppBar. Buka file shr\_product\_grid\_fragment.xml, hapus blok <LinearLayout> yang berisi "You did it!" TextView dan ganti dengan yang berikut ini:

```

<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
 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=".ProductGridFragment">

 <com.google.android.material.appbar.AppBarLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content">

 <androidx.appcompat.widget.Toolbar
 android:id="@+id/app_bar"
 style="@style/Widget.Shrine.Toolbar"
 android:layout_width="match_parent"
 android:layout_height="?attr actionBarSize"
 app:title="@string/shr_app_name"
 app:navigationIcon="@drawable/shr_menu"/>
 </com.google.android.material.appbar.AppBarLayout>
</FrameLayout>

```

15. Menambahkan action buttons dan style di top app bar. Di fungsi onCreateView dari kelas ProductGridFragment.kt, atur activity Toolbar yang akan digunakan sebagai ActionBar menggunakan setSupportActionBar. Kita dapat melakukan ini setelah tampilan dibuat dengan inflator.

```

override fun onCreateView(
 inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View?
{
 // Inflate the layout for this fragment with the ProductGrid theme
 val view = inflater.inflate(R.layout.shr_product_grid_fragment, container, false)

 // Set up the toolbar.
 (activity as AppCompatActivity).setSupportActionBar(view.app_bar)

 return view;
}

```

16. Selanjutnya, langsung di bawah metode yang baru saja kita ubah untuk mengatur toolbar, mari kita ganti onCreateOptionsMenu untuk mengembang konten shr\_toolbar\_menu.xml ke dalam toolbar:

```
override fun onCreateOptionsMenu(menu: Menu, menuInflater: MenuInflater) {
 menuInflater.inflate(R.menu.shr_toolbar_menu, menu)
 super.onCreateOptionsMenu(menu, menuInflater)
}
```

17. Akhirnya, ganti onCreate () di ProductGridFragment.kt, dan setelah memanggil super (), panggil setHasOptionsMenu dengan true:

```
override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)
 setHasOptionsMenu(true)
}
```

18. Cuplikan kode di atas mengatur app bar dari layout XML menjadi Action Bar untuk activity ini. Panggilan balik padaCreateOptionsMenu memberi tahu aktivitas apa yang harus digunakan sebagai menu. Dalam hal ini, panggilan itu akan menempatkan item menu dari R.menu.shr\_toolbar\_menu ke app bar. File menu berisi dua item: "Search" dan "Filter".
19. Jalankan dan amati hasilnya. Sekarang toolbar memiliki ikon navigasi, judul, dan dua ikon tindakan di sisi kanan. toolbar juga menampilkan ketinggian menggunakan bayangan halus yang menunjukkan layer itu pada layer yang berbeda dari konten.
20. Menambahkan card. Kita akan menambahkan satu card di bawah top app bar. Card harus memiliki wilayah untuk gambar, judul, dan label untuk teks sekunder. Tambahkan yang berikut ini di shr\_product\_grid\_fragment.xml di bawah AppBarLayout.

```
<com.google.android.material.card.MaterialCardView
 android:layout_width="160dp"
 android:layout_height="180dp"
 android:layout_marginBottom="16dp"
 android:layout_marginLeft="16dp"
 android:layout_marginRight="16dp"
 android:layout_marginTop="70dp"
 app:cardBackgroundColor="?attr/colorPrimaryDark"
 app:cardCornerRadius="4dp">

 <LinearLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_gravity="bottom"
 android:background="#FFFFFF"
 android:orientation="vertical"
 android:padding="8dp">

 <TextView
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:padding="2dp"
 android:text="@string/shr_product_title"
 android:textAppearance="?attr/textAppearanceHeadline6" />

 <TextView
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:padding="2dp"
 android:text="@string/shr_product_description"
 android:textAppearance="?attr/textAppearanceBody2" />
 </LinearLayout>
</com.google.android.material.card.MaterialCardView>
```

21. Jalankan dan amati hasilnya. Kita dapat melihat Card sisipan dari tepi kiri, dan sudut-sudutnya membulat dan bayangan (yang menyatakan ketinggian Card). Seluruh elemen disebut " container." Selain dari container, semua elemen di dalamnya adalah opsional. Kita dapat menambahkan elemen-elemen berikut ke dalam container: teks header, thumbnail atau avatar, teks subjudul, pembagi, dan bahkan tombol dan ikon. Card yang baru saja kita buat, misalnya, berisi dua TextViews (satu untuk judul, dan satu untuk teks sekunder) di LinearLayout, selaras dengan bagian bawah Card. Card biasanya ditampilkan dalam koleksi dengan Card lain. Pada lanjutan project ini, kita akan meletakkannya sebagai koleksi di grid.
22. Buat grid card. Ketika beberapa card hadir di layar, card tersebut dikelompokkan bersama menjadi satu atau lebih koleksi. Card dalam grid adalah coplanar, artinya mereka berbagi resting elevation yang sama satu sama lain (kecuali diambil atau diseret).
23. Buat grid card. Ketika beberapa card hadir di layar, card tersebut dikelompokkan bersama menjadi satu atau lebih koleksi. Card dalam grid adalah coplanar, artinya mereka berbagi resting elevation yang sama satu sama lain (kecuali diambil atau diseret).
24. Siapkan grid card. Lihat dan pelajari file shr\_product\_card.xml yang sudah ada. Layout card ini berisi card dengan gambar (dalam hal ini, NetworkImageView, yang memungkinkan kita memuat dan menampilkan gambar dari URL), dan dua TextView. Selanjutnya, lihat dan pelajari juga file ProductCardRecyclerViewAdapter yang sudah ada. Ada dalam paket yang sama dengan ProductGridFragment. Kelas adaptor di atas mengelola konten dari grid yang ada. Untuk menentukan apa yang harus dilakukan oleh setiap tampilan dengan kontennya, kita akan segera menulis kode untuk onBindViewHolder(). Dalam paket yang sama, kita juga dapat melihat ProductCardViewHolder. Kelas ini menyimpan tampilan yang memengaruhi layout card, sehingga kita dapat memodifikasinya nanti. Untuk menyiapkan grid, pertama-tama kita menghapus placeCard MaterialCardView dari shr\_product\_grid\_fragment.xml. Selanjutnya, kita harus menambahkan komponen yang mewakili grid card kita. Dalam hal ini, kita akan menggunakan RecyclerView. Tambahkan komponen RecyclerView ke shr\_product\_grid\_fragment.xml kita di bawah komponen XML AppBarLayout.

```
<androidx.core.widget.NestedScrollView
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:layout_marginTop="56dp"
 android:background="@color/productGridBackgroundColor"
 android:paddingStart="@dimen/shr_product_grid_spacing"
 android:paddingEnd="@dimen/shr_product_grid_spacing"

 app:layout_behavior="@string/appbar_scrolling_view_behavior">

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

</androidx.core.widget.NestedScrollView>
```

25. Terakhir, di onCreateView(), tambahkan kode inisialisasi RecyclerView ke ProductGridFragment.kt setelah kita memanggil setUpToolbar (tampilan) dan sebelum pernyataan return. Cuplikan kode di atas berisi langkah-langkah inisialisasi yang diperlukan untuk menyiapkan RecyclerView. Ini termasuk pengaturan layout manager RecyclerView, ditambah inisialisasi dan pengaturan adapter RecyclerView.

```
override fun onCreateView(
 inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View?
{
 // Inflate the layout for this fragment with the ProductGrid theme
 val view = inflater.inflate(R.layout.shr_product_grid_fragment, container, false)

 // Set up the toolbar.
 (activity as AppCompatActivity).setSupportActionBar(view.app_bar)

 // Set up the RecyclerView
 view.recycler_view.setHasFixedSize(true)
 view.recycler_view.layoutManager =
```

```

 GridLayoutManager(context, 2, RecyclerView.VERTICAL, false)
 val adapter = ProductCardRecyclerViewAdapter(
 ProductEntry.initProductEntryList(resources))
 view.recycler_view.adapter = adapter
 val largePadding = resources.getDimensionPixelSize(R.dimen.shr_product_grid_spacing)
 val smallPadding =
 resources.getDimensionPixelSize(R.dimen.shr_product_grid_spacing_small)
 view.recycler_view.addItemDecoration(ProductGridItemDecoration(largePadding,
 smallPadding))

 return view;
}

```

26. Jalankan dan amati hasilnya.
27. Menambahkan gambar dan teks. Untuk setiap card, tambahkan gambar, nama produk, dan harga. Abstraksi ViewHolder kita menampung view untuk setiap card. Di ViewHolder kita, tambahkan tiga tampilan sebagai berikut.

```

class ProductCardViewHolder(itemView: View) //TODO: Find and store views from
itemView
 : RecyclerView.ViewHolder(itemView) {
 var productImage: NetworkImageView =
 itemView.findViewById(R.id.product_image)
 var productTitle: TextView = itemView.findViewById(R.id.product_title)
 var productPrice: TextView = itemView.findViewById(R.id.product_price)
}

```

28. Perbarui method onBindViewHolder () di ProductCardRecyclerViewAdapter untuk menetapkan judul, harga, dan gambar produk untuk setiap tampilan produk seperti yang ditunjukkan di bawah ini:

```

override fun onBindViewHolder(holder: ProductCardViewHolder, position:
Int) {
 // TODO: Put ViewHolder binding code here in MDC-102
 if (position < productList.size) {
 val product = productList[position]
 holder.productTitle.text = product.title
 holder.productPrice.text = product.price
 ImageRequester.setImageFromUrl(holder.productImage,
product.url)
 }
}

```

29. Kode di atas memberi tahu adapter RecyclerView apa yang harus dilakukan dengan setiap card, menggunakan ViewHolder. Di sini, adapter menetapkan data teks pada masing-masing TextViews ViewHolder, dan memanggil ImageRequester untuk mendapatkan gambar dari URL. ImageRequester adalah kelas yang kita sediakan untuk kenyamanan kita, dan kelas ini menggunakan library Volley.
30. Jalankan dan amati hasilnya.



## LATIHAN

---

1. Modifikasilah aplikasi dengan menambahkan/mengubah warna, typography, elevation dan layout.
2. Gunakan referensi berikut.

<https://codelabs.developers.google.com/codelabs/mdc-103-kotlin/#0>



## TUGAS

---

1. Modifikasi aplikasi dengan mengembangkan project diatas.
2. Gunakan referensi berikut.

<https://codelabs.developers.google.com/codelabs/mdc-104-kotlin/#0>



## REFERENSI

---

1. <https://kotlinlang.org/docs/reference/>
2. <https://developer.android.com/kotlin>
3. <https://developer.android.com/courses/kotlin-android-fundamentals/toc>
4. <https://codelabs.developers.google.com/android-kotlin-fundamentals/>
5. <https://developer.android.com/kotlin/learn>
6. <https://developer.android.com/kotlin/resources>
7. <https://codelabs.developers.google.com/codelabs/mdc-101-kotlin/#0>
8. <https://codelabs.developers.google.com/codelabs/mdc-102-kotlin/#0>
9. <https://codelabs.developers.google.com/codelabs/mdc-103-kotlin/#0>
10. <https://codelabs.developers.google.com/codelabs/mdc-104-kotlin/#0>