**MODUL**

**PRAKTIKUM PEMROGRAMAN MOBILE**

**(Program Studi Teknik Komputer)**

****

**Disusun oleh :**

**Yosep Murya Kusuma Ardhana**

**Pius Dian Widi Anggoro**

**LN Harnaningrum**

**SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER**

**AKAKOM**

**YOGYAKARTA**

**2020**

# KATA PENGANTAR

Syukur kami haturkan kepada Tuhan yang Mahakuasa, kami dapat menyelesaikan modul ini. Modul ini diperuntukkan bagi mahasiswa yang sedang mempelajari materi Pemrograman Mobile. Materi dimulai dari pengenalan aplikasi, kemudian membuat aplikasi sederhana dan dilanjutkan dengan penggunaan komponen-komponen untuk melengkapi pembuatan aplikasi. Modul ini menggunakan Bahasa pemrograman Kotlin beserta kelengkapan lainnya dan menggunakan project yang dibangun dengan menggunakan Android Studio.

Modul ini dibuat berdasarkan rencana pembelajaran yang sudah dibuat dan digunakan sebagai acuan kurikulum 2019. Kemungkinan masih banyak kekurangan yang ada pada setiap modul. Untuk itu, masukan dan saran untuk perbaikan modul ini sangat membantu kami dalam menyempurnakan modul ini.

Dan akhirnya, semoga modul ini membantu dosen pengampu dalam mentransfer ilmu ke pada mahasiswa dan membantu mahasiswa dalam menyerap ilmu yang diberikan oleh dosen. Semua masukan adalah berharga bagi perkembangan kami dan anak didik.

Yogyakarta, Desember 2019.

# DAFTAR ISI

[KATA PENGANTAR iii](#_Toc30062539)

[DAFTAR ISI iv](#_Toc30062540)

[MODUL 1 Pengenalan Aplikasi Mobile 5](#_Toc30062541)

[MODUL 2 Membuat Aplikasi dan Menjalankannya 27](#_Toc30062542)

[MODUL 3 Layout dengan Linear Layout dan Constrain Layout 39](#_Toc30062543)

[MODUL 4 Komponen Widget View 51](#_Toc30062544)

[MODUL 5 RecycleView 58](#_Toc30062545)

# 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. 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.

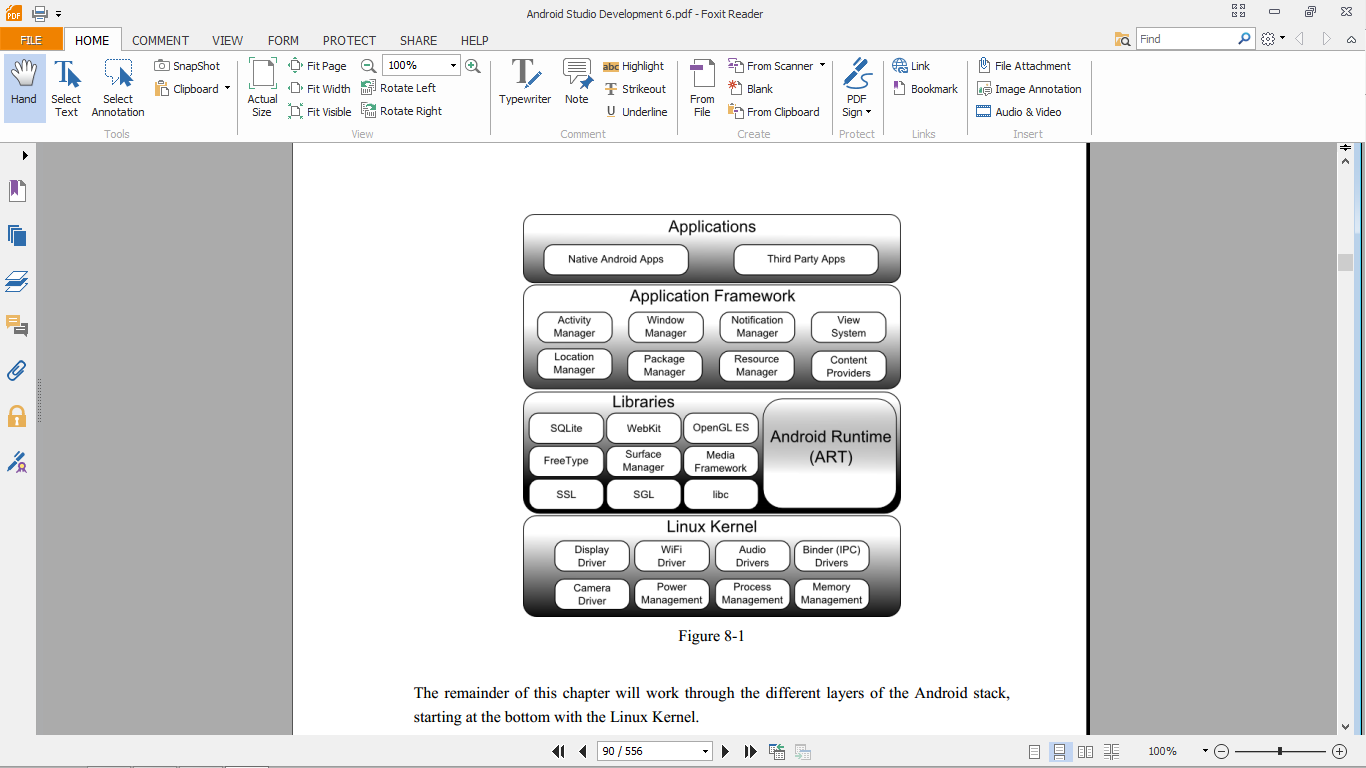
Kita akan menggunakan sumber referensi utama pada link andriod developer dan codelab berikut:

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 koding 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. Android dan Pemrograman Mobile.

Android adalah sistem operasi mobile yang open source. Tahun 2005, Google meng-akusisi perusahaan **Android Inc.** untuk memulai mengembangkan platform Android. Tahun 2007, sekelompok pemimpin industri datang bersama membentuk **Open Handset Alliance** ([http://www.openhandsetalliance.com](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.



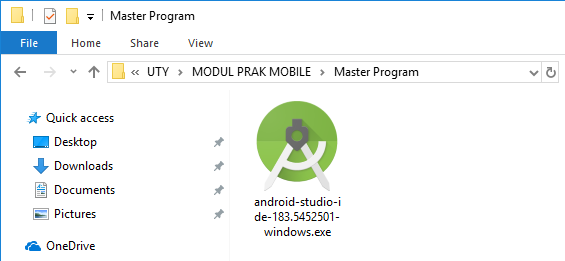
Gambar 1-1. Arsitektur Aplikasi Android

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.

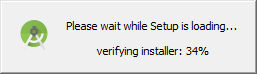
AVD pada dasarnya adalah emulator yang memungkinkan aplikasi Android untuk diuji tanpa perlu menginstal aplikasi pada perangkat berbasis Android fisik. Sebuah AVD dapat dikonfigurasi untuk meniru berbagai fitur perangkat keras termasuk pilihan ukuran layar, kapasitas memori dan fitur seperti kamera, 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 |

1. **Install Android Studio**
2. 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**



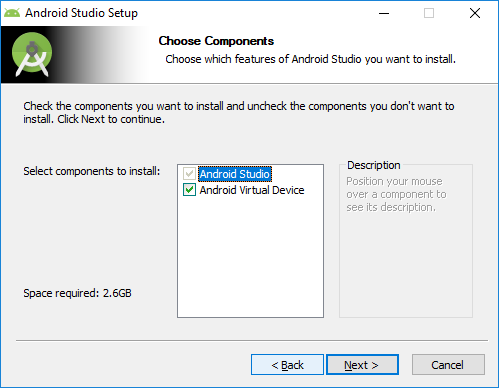
1. Tunggu beberapa saat sampai proses loading verifying installer selesai.



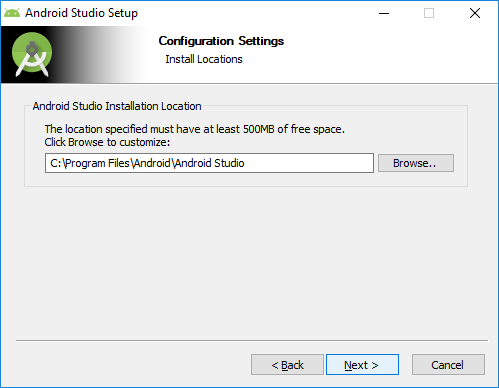
1. Setelah proses loading verifying installer selesai, akan terlihat jendela Android Studio Setup, **klik** button **Next >** untuk melanjutkan proses penginstallan.



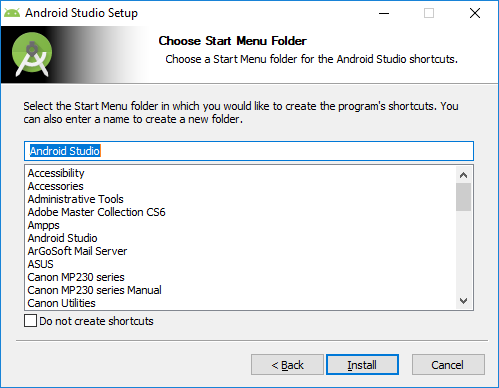
1. Beri tanda checklist atau centang bagian **Android Virutal Device**. Fungsi dari komponen Android Virtual Device yaitu untuk menampilkan interface android dalam bentuk virtual. Langkah selanjutnya **klik** pada button **Next >**



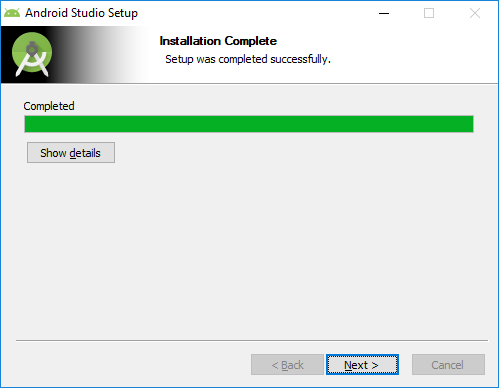
1. Pada bagian Configuration Settings akan diminta untuk memilih lokasi penginstallan, secara default lokasi penginstallan pada direktori **C:\Pragram Files\Android\Android Studio.** Jika ingin menggunakan lokasi defult maka dapat langsung menuju proses selanjutnya dengan meng-**klik** button **Next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



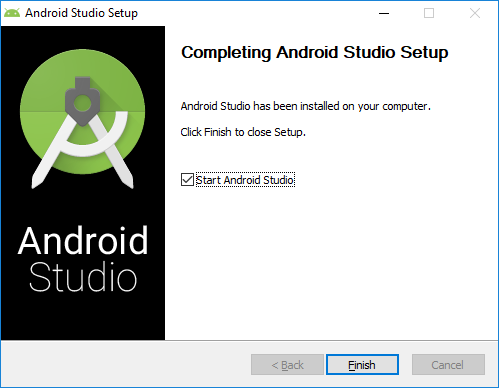
1. Pada bagian Choose Start Menu Folder, **klik** button **Install**



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



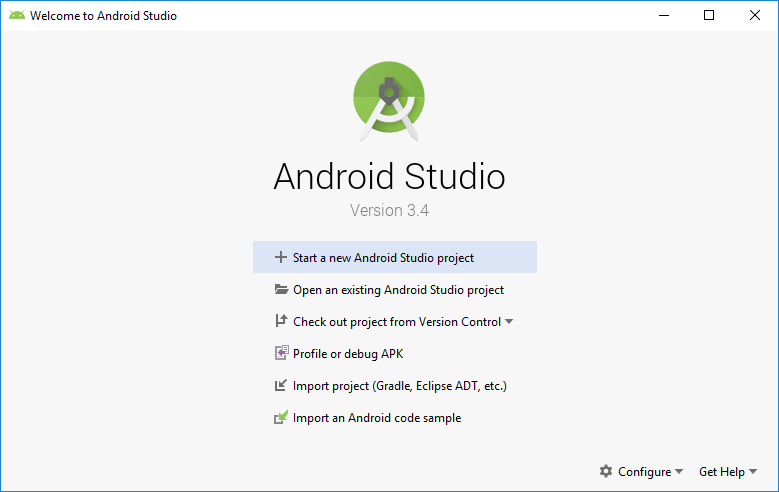
1. Proses penginstallan Android Studio telah selesai, **klik** button **Finish** untuk menutup jendela Android Studio Setup. Jika ingin menjalankan langsung program Android Studio maka beri tanda **checklist** atau centang pada bagian **Start Android Studio** sebelum menekan button Finish.



1. **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:

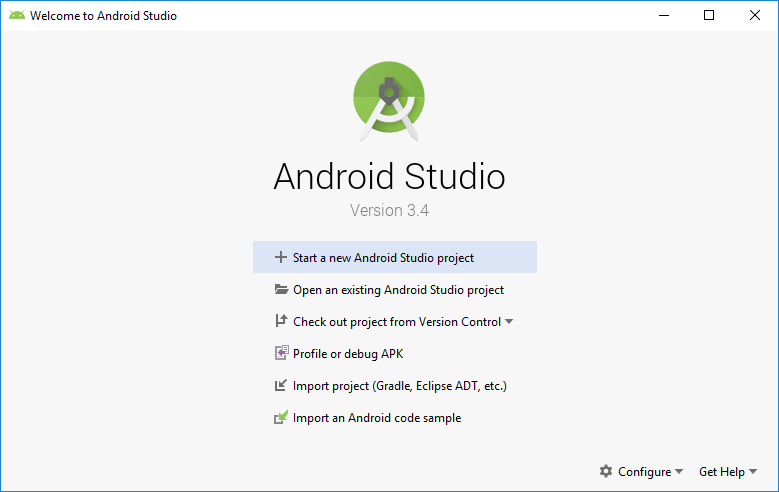
1. Buka Android Studio dari komputer yang dipakai.
2. Maka akan muncul jendela Welcome to Android Studio.



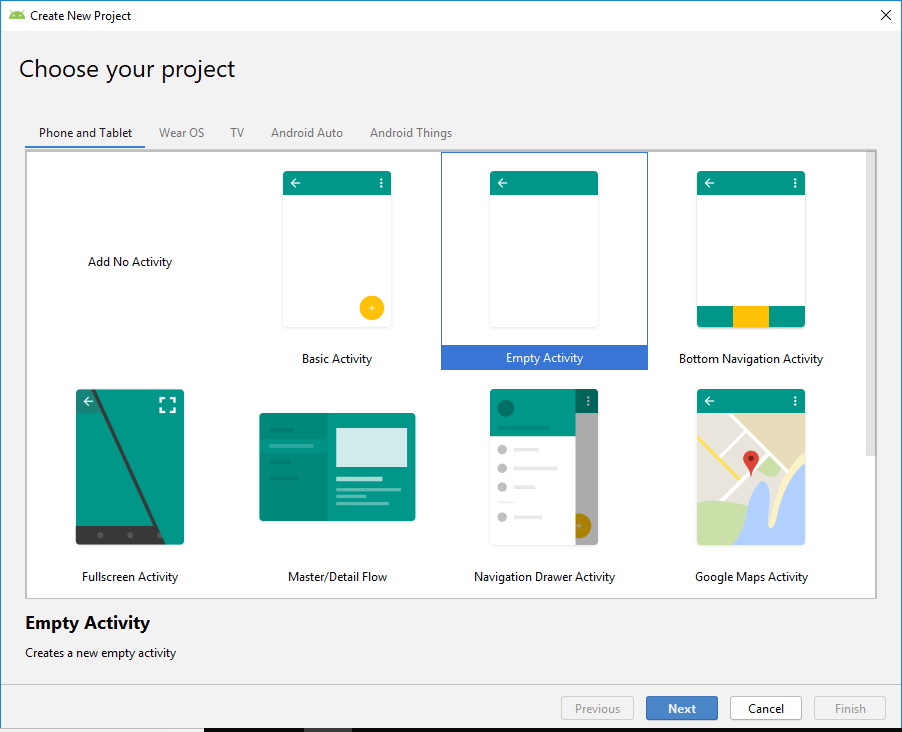
1. **Membuat Project Dengan Android Studio**

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

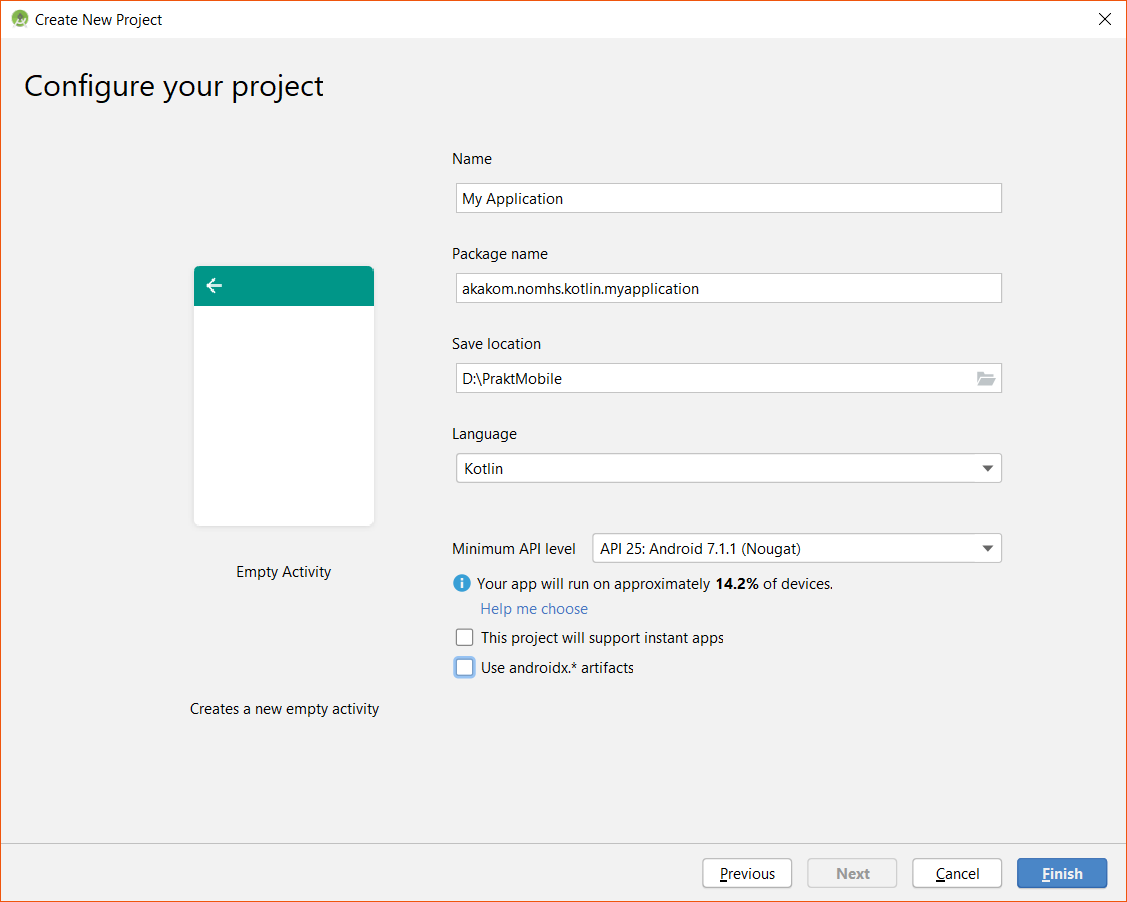
1. Dalam membuat project awal dengan Android Studio dengan cara **klik** pada **+ Start a new Android Studio project**



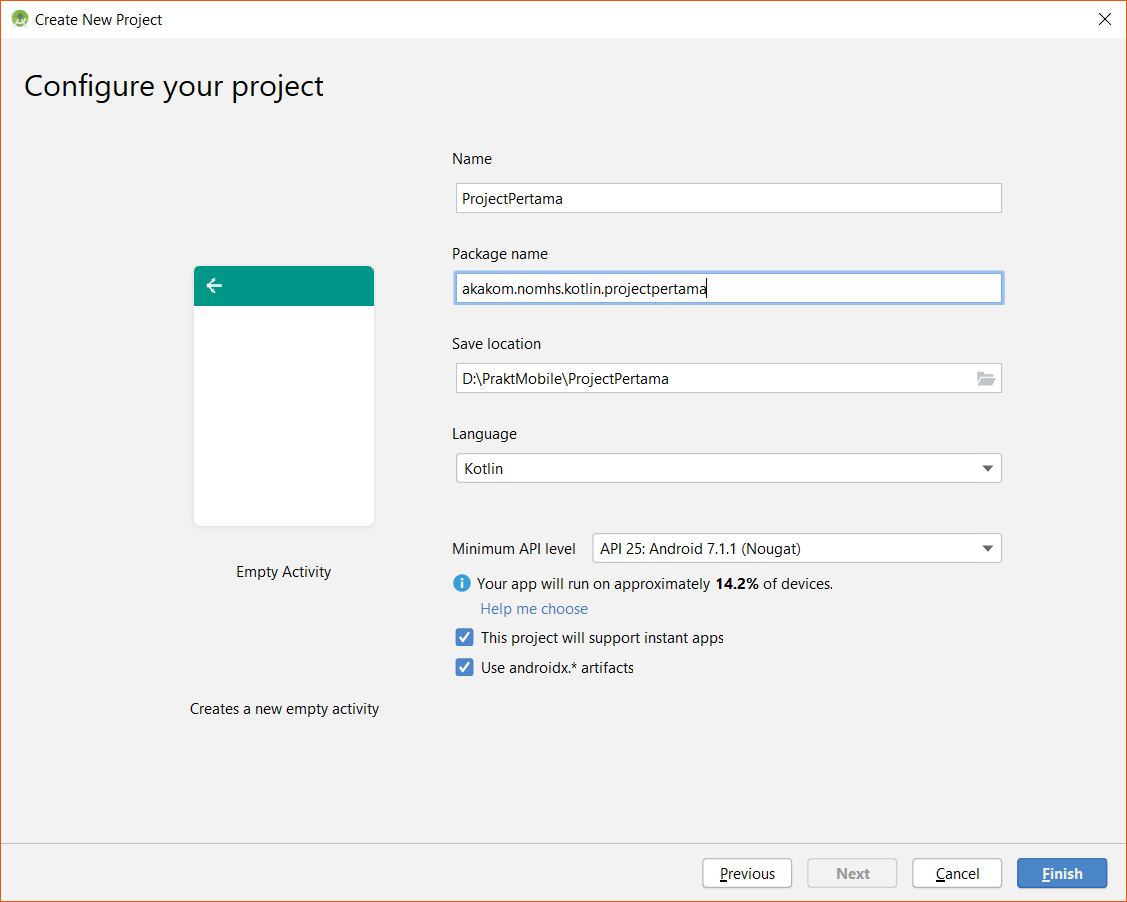
1. Kemudian akan terlihat jendela Create New Project, pada bagian ini programmer dapat memilih bentuk dari project yang akan dibuat (hal ini menyesuaikan dengan project yang akan dibuat). Sebagai latihan awal maka **pilih** pada bagian **Empty Activity**, kemudian **klik** button **Next.**



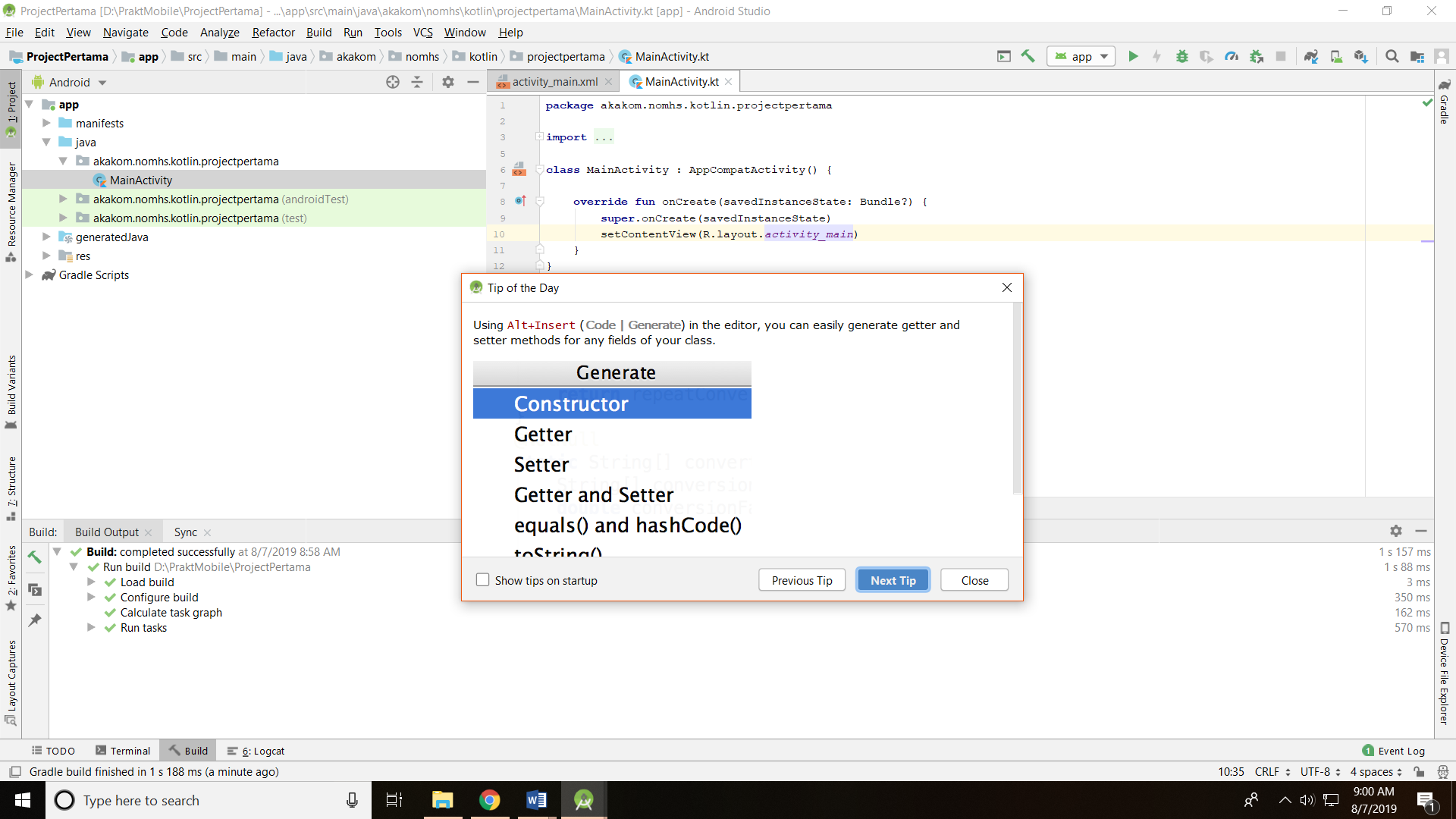
1. Pada jendela Create New Project, maka secara default Name project yaitu My Application, penamaan ini dapat dirubah sesuai dengan nama project yang dibuat.



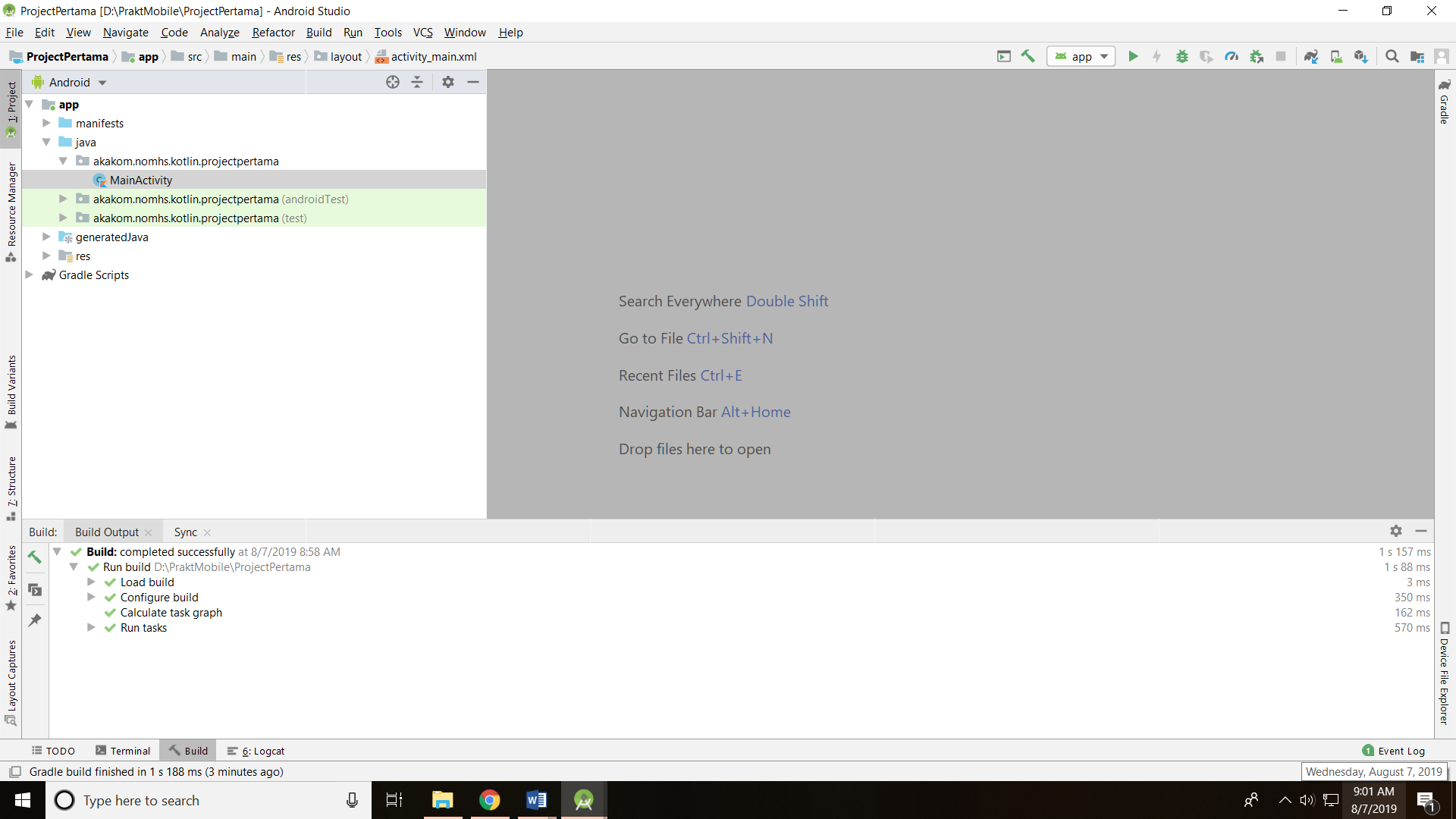
1. Sebagai bahan latihan, beri nama pada bagian Name : **ProjectPertama** kemudian pada bagian Save location : **.....** *(gunakan folder kerja anda),* kemudian **klik** button **Finish**.



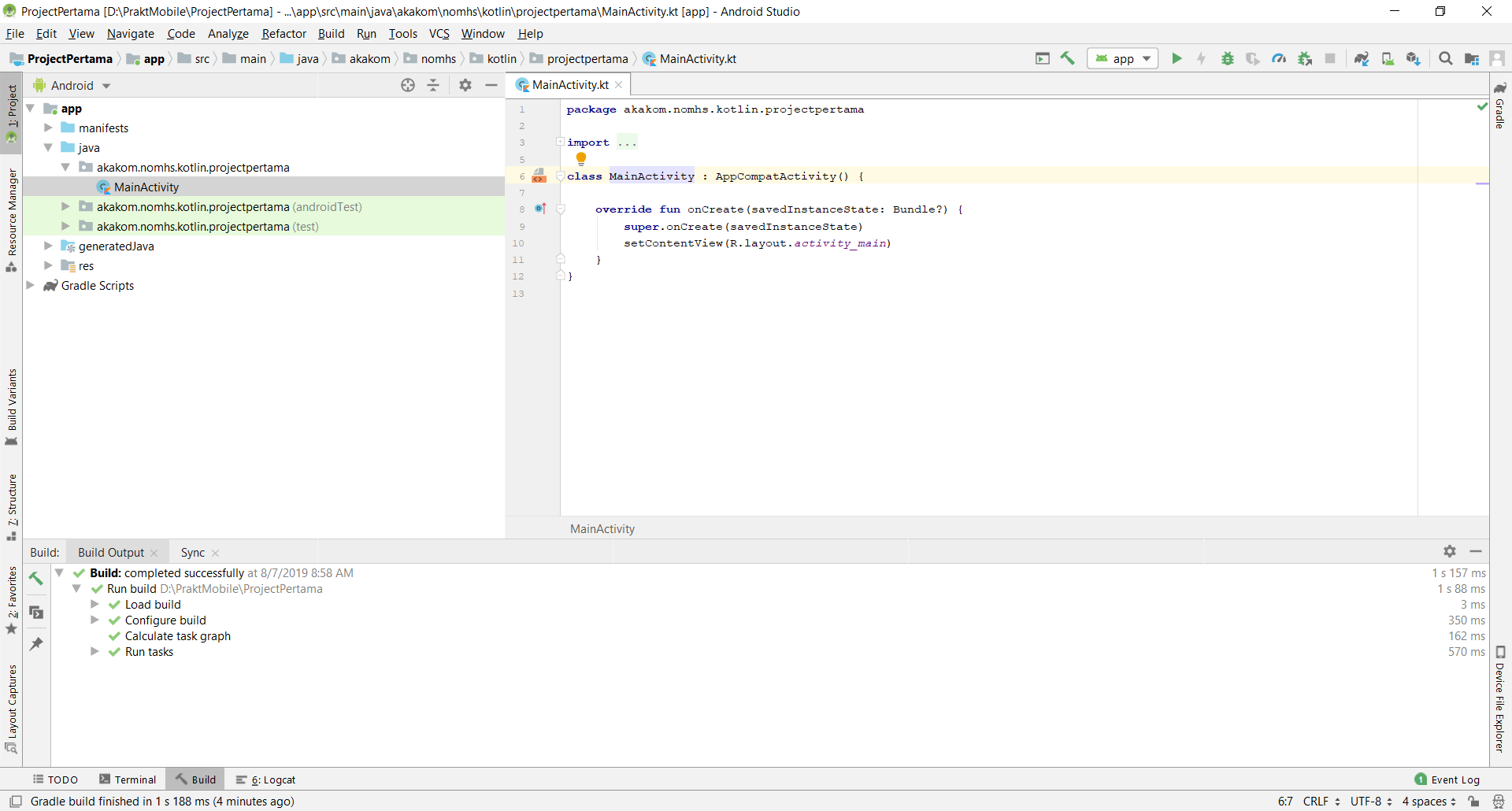
1. Pada saat awal atau pertamakali membuat project maka akan muncul sebuah jendela dengan nama Tips of The Day yaitu informasi tentang pemrograman yang dapat dijadikan acuan para programmer, untuk menutup jendela Tips of The Day dapat dengan cara **klik** button **Close**.



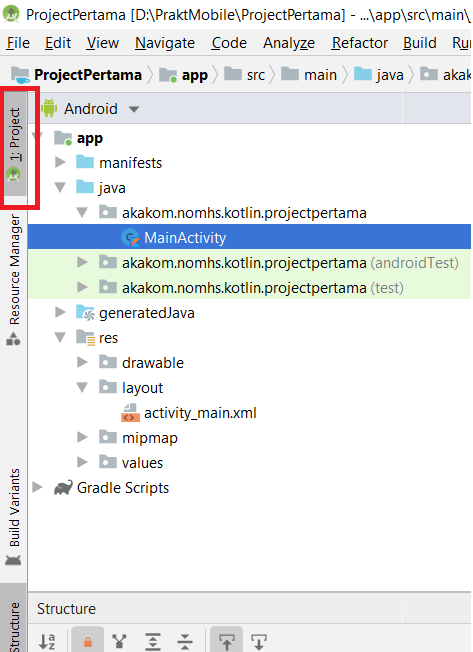
1. Jendela ProjectPertama akan muncul seperti pada gambar dibawah ini, dan tentunya belum secara utuh. Tunggulah beberapa saat sampai proses loading fitur-fitur atau library dan tools muncul.



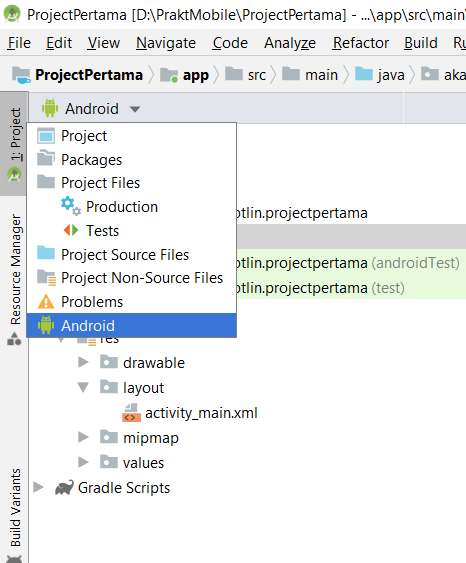
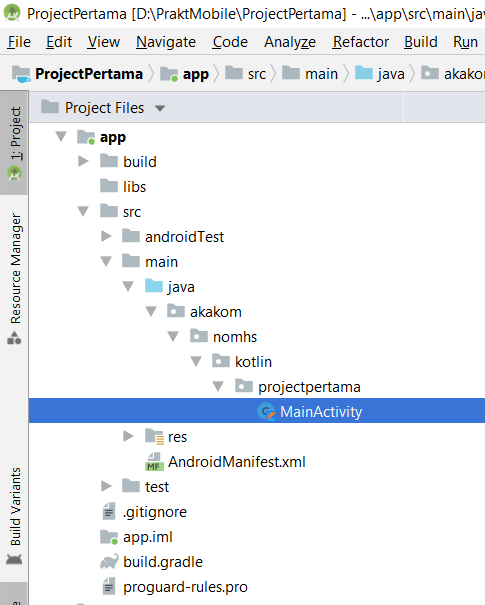
1. Jika proses loading fitur-fitur atau library selesai dan tools telah muncul, maka akan terlihat pada jendela IDE seperti pada gambar dibawah ini. Terdapat 3 bagian jendela pada Android Studio yaitu jendela Project, jendela Sourcecode dan jendela Built Output.



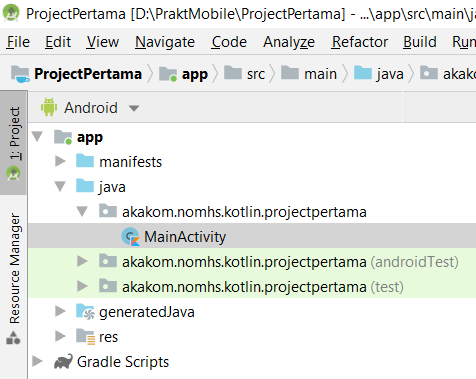
1. Pada jendela kiri atas, jika tab Project belum dipilih, klik tab Project.



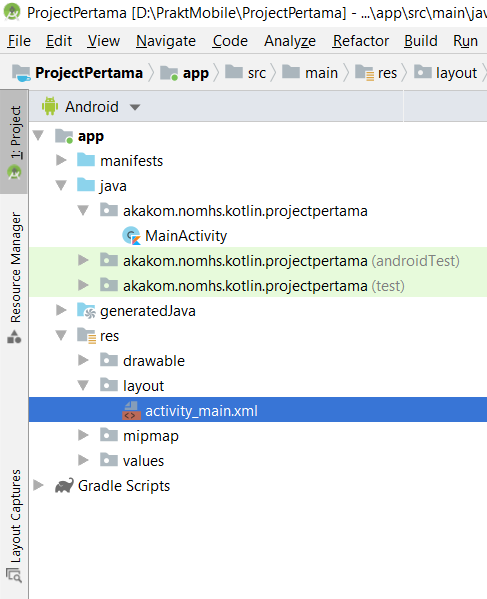
1. Hirarki Project dapat dipilih dengan menggunakan drop down menu. Hirarki standar adalah Android.

 dan 

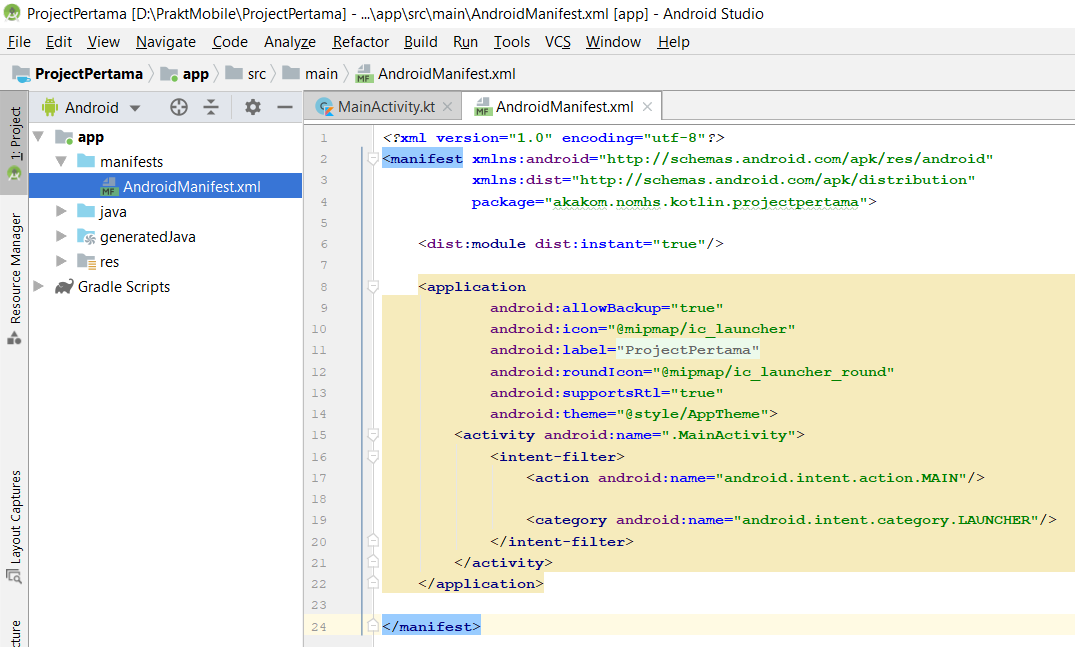
1. Untuk selanjutnya kita akan menggunakan hirarki Android.
2. Sekarang akan kita lihat panel Project. Yang pertama kita mengeksplorasi folder app.
3. Di panel Project> Android, eksplore folder app. Di dalam folder app ada empat subfolder: manifest, java, generateJava, dan res.
4. Buka folder java, dan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



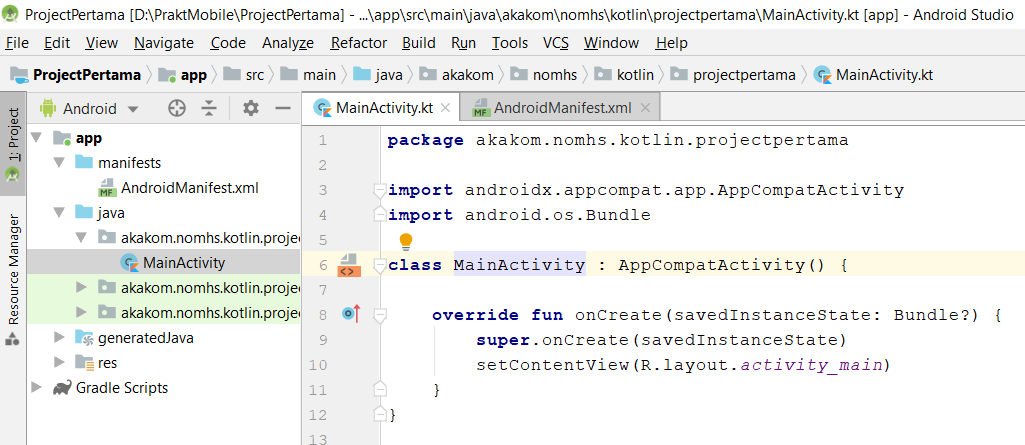
1. Folder java berisi semua kode Kotlin utama untuk aplikasi Android. Ada alasan historis mengapa kode Kotlin Anda muncul di folder java. Konvensi itu memungkinkan Kotlin untuk beroperasi tanpa hambatan dengan kode yang ditulis dalam bahasa pemrograman Java, bahkan dalam proyek dan aplikasi yang sama.
2. 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.
3. Dalam sistem file, file Kotlin memiliki ekstensi .kt dan ikon K. Pada tampilan Proyek, Android Studio menunjukkan nama kelas (MainActivity) tanpa ekstensi.
4. 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.
5. Kemudian kita akan melihat folder berikutnya, yaitu res. Di panel Project> Android, ekspand folder res.
6. 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.
7. 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.
8. Di dalam folder res, ekspand folder layout untuk melihat file activity\_main.xml.



1. 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.
2. Sekarang kita akan menjelajahi folder manifest dan AndroidManifest.xml
3. Folder manifes berisi file yang memberikan informasi penting tentang app kita ke sistem Android.
4. 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.



1. Perhatikan bahwa MainActivity direferensikan di elemen <activity>. Aktivitas apa pun di app kita harus dinyatakan dalam manifest. Contoh manifest untuk MainActivity.
2. Catat elemen <intent-filter> ……. <activity>. Elemen <action> dan <category> dalam filter maksud ini memberi tahu Android tempat memulai app ketika pengguna mengklik ikon run.
3. 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.
4. Terakhir, kita akan menjelajahi folder Script Gradle.
5. 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.)
6. 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 ProjectPertama.

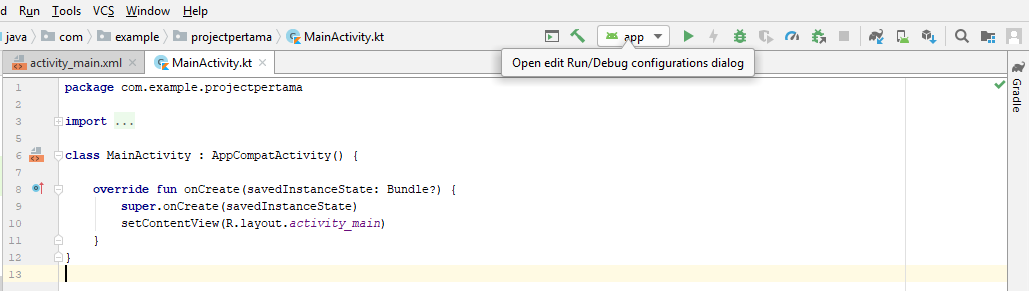


1. **Pengenalan Aplikasi Android dan IDE Android Studio**

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 :

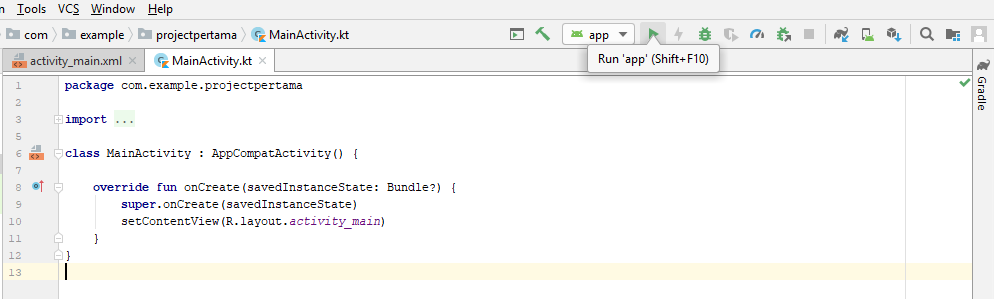
1. Open edit Run/Debug configurations dialog

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



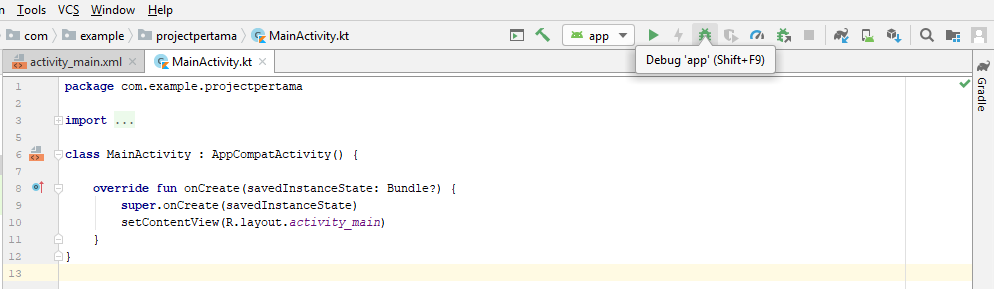
1. Run App

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



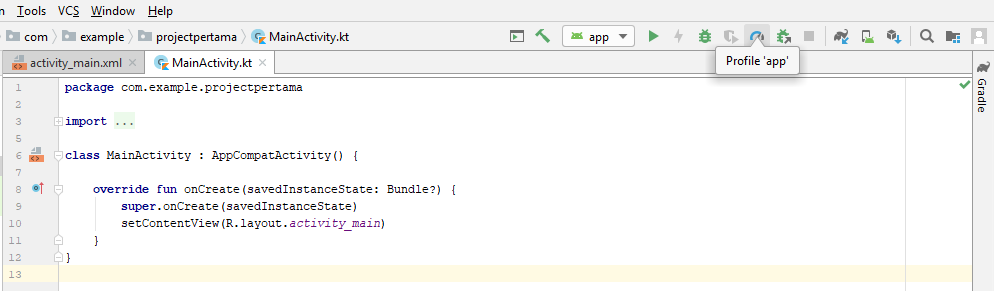
1. Debug App

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).



1. Profile App

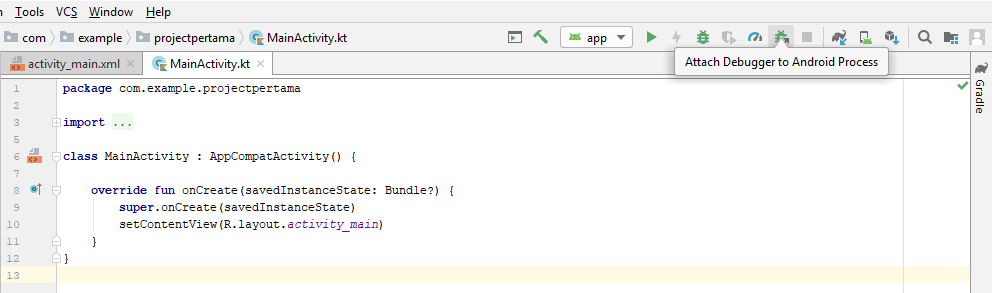
Profile App merupakan alat pembuatan profil baru yang menyediakan data realtime untuk CPU, memori, dan aktivitas jaringan aplikasi Anda. Programmer dapat melakukan pelacakan metode berbasis sampel untuk mengukur waktu eksekusi script, merekam heap-dump, menampilkan alokasi memori, dan memeriksa detail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



Dengan debugger Android Studio, maka programmer dapat :

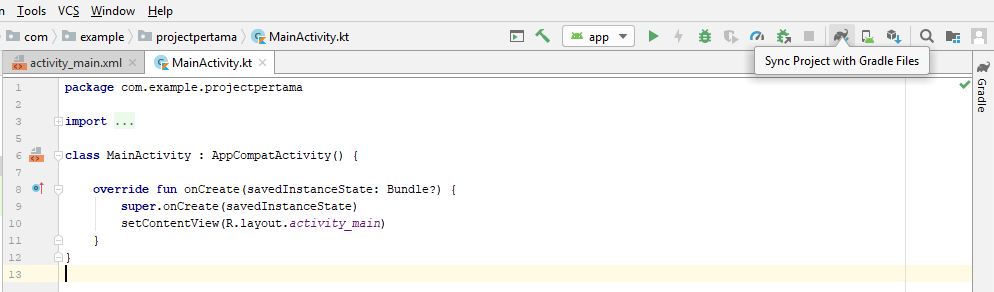
1. Memilih perangkat untuk men-debug pada aplikasi yang dibuat.
2. Menyetel breakpoint dalam kode Java dan C/C++ pada aplikasi yang dibuat.
3. Memeriksa variabel dan mengevaluasi ekspresi pada saat waktu proses.
4. Mengambil tangkapan layar dan video dari aplikasi yang dibuat.
5. Attach Debugger to Android Process

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



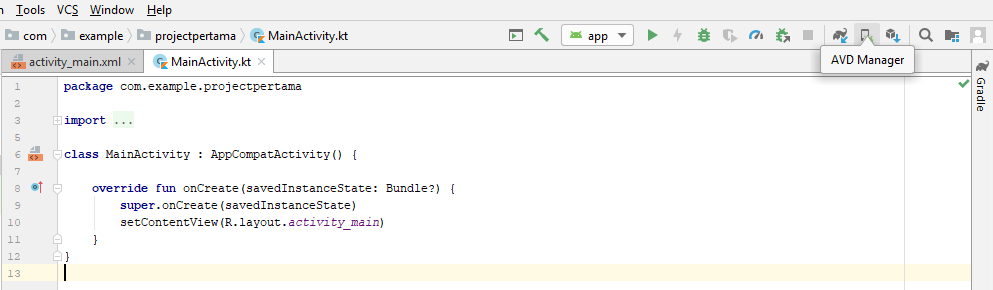
1. Sync Project with Gradle Files

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.



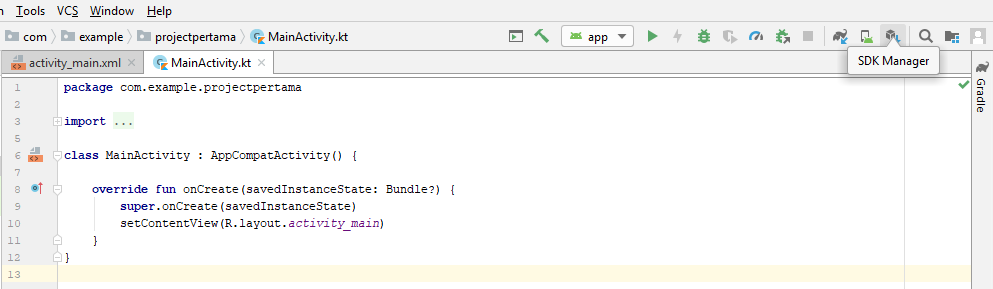
1. AVD Manager

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.



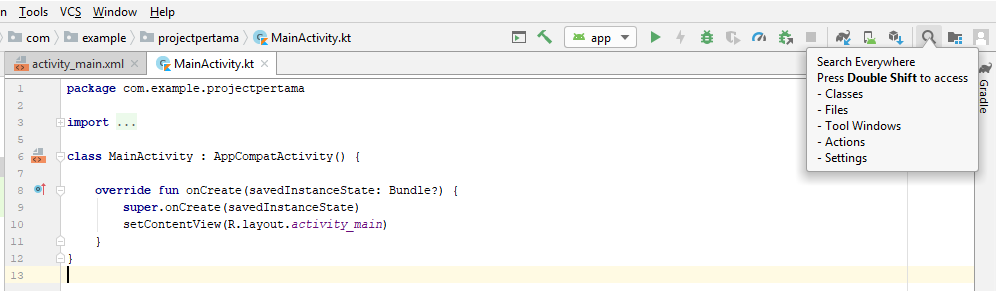
1. SDK Manager

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



1. Search Everywhere

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



1. **Materi Pengayaan**

a. Penggunaan Kotlin untuk Pengembangan Android.

Kotlin/Native memungkinkan developer untuk menggunakannya 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](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](https://kotlinlang.org/docs/reference/packages.html).

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](https://kotlinlang.org/docs/reference/functions.html).

**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](https://kotlinlang.org/docs/reference/properties.html).

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](https://kotlinlang.org/docs/reference/kotlin-doc.html) 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](https://kotlinlang.org/docs/reference/basic-types.html#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](https://kotlinlang.org/docs/reference/control-flow.html#if-expression).

**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](https://kotlinlang.org/docs/reference/null-safety.html).

**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](https://kotlinlang.org/docs/reference/classes.html) and [Type casts](https://kotlinlang.org/docs/reference/typecasts.html).

**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](https://kotlinlang.org/docs/reference/control-flow.html#for-loops).

**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](https://kotlinlang.org/docs/reference/control-flow.html#while-loops).

**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](https://kotlinlang.org/docs/reference/control-flow.html#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](https://kotlinlang.org/docs/reference/ranges.html).

**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](https://kotlinlang.org/docs/reference/lambdas.html).

**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](https://kotlinlang.org/docs/reference/classes.html) and [objects and instances](https://kotlinlang.org/docs/reference/object-declarations.html).

|  |  |
| --- | --- |
|  | **LATIHAN** |

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

|  |  |
| --- | --- |
| **tugas.png** | **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 permisson 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 menggambar 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)

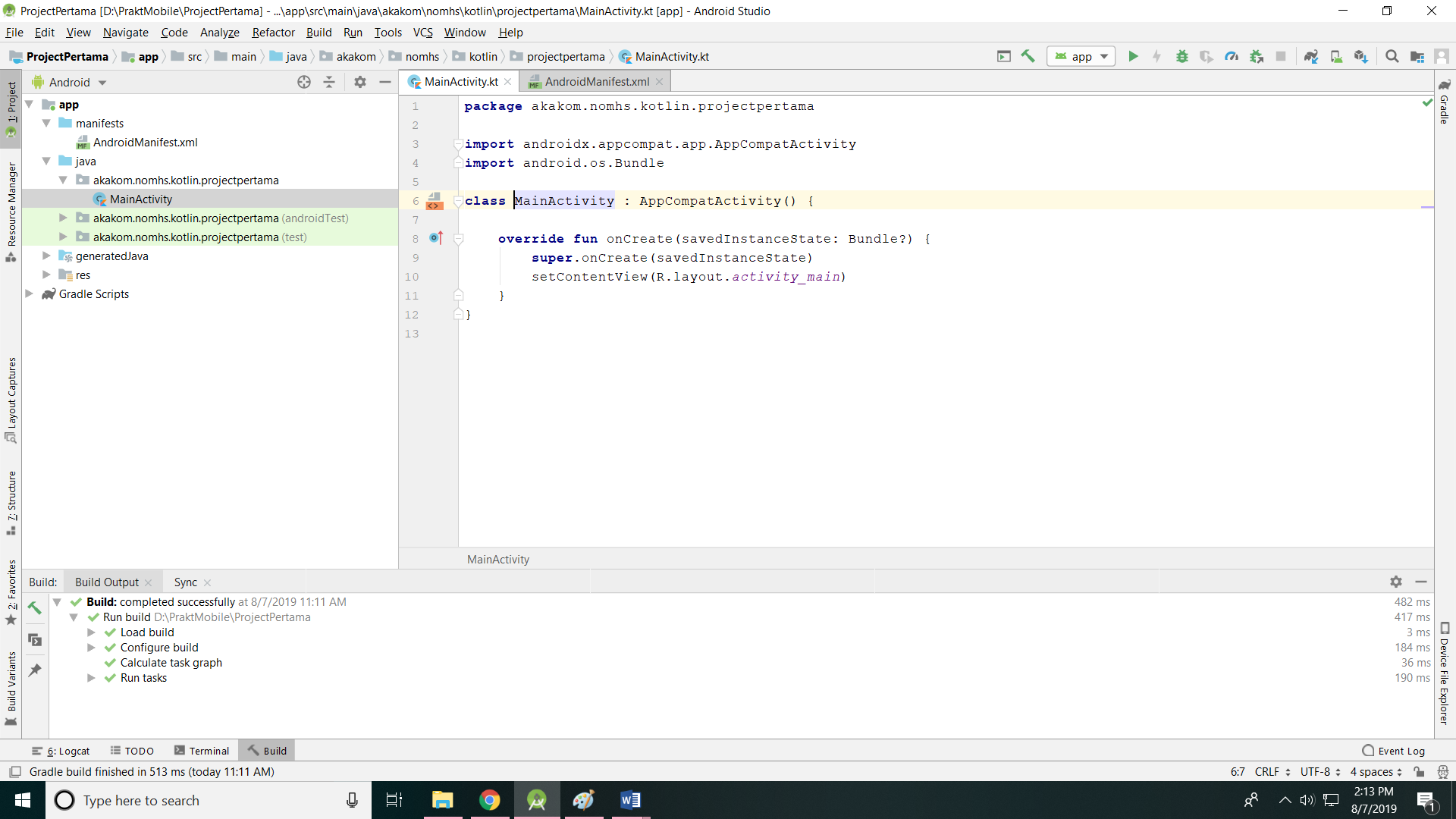
Setelah memenuhi persyaratan di atas, langkah selanjutnya adalah menginstal HAXM. HAXM adalah *hardware-assisted virtualization engine*yang menggunakan teknologi VT dari Intel untuk mempercepat aplikasi Android yang diemulasi di mesin host. HAXM diperlukan untuk menjalankan emulator di Android Studio.

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



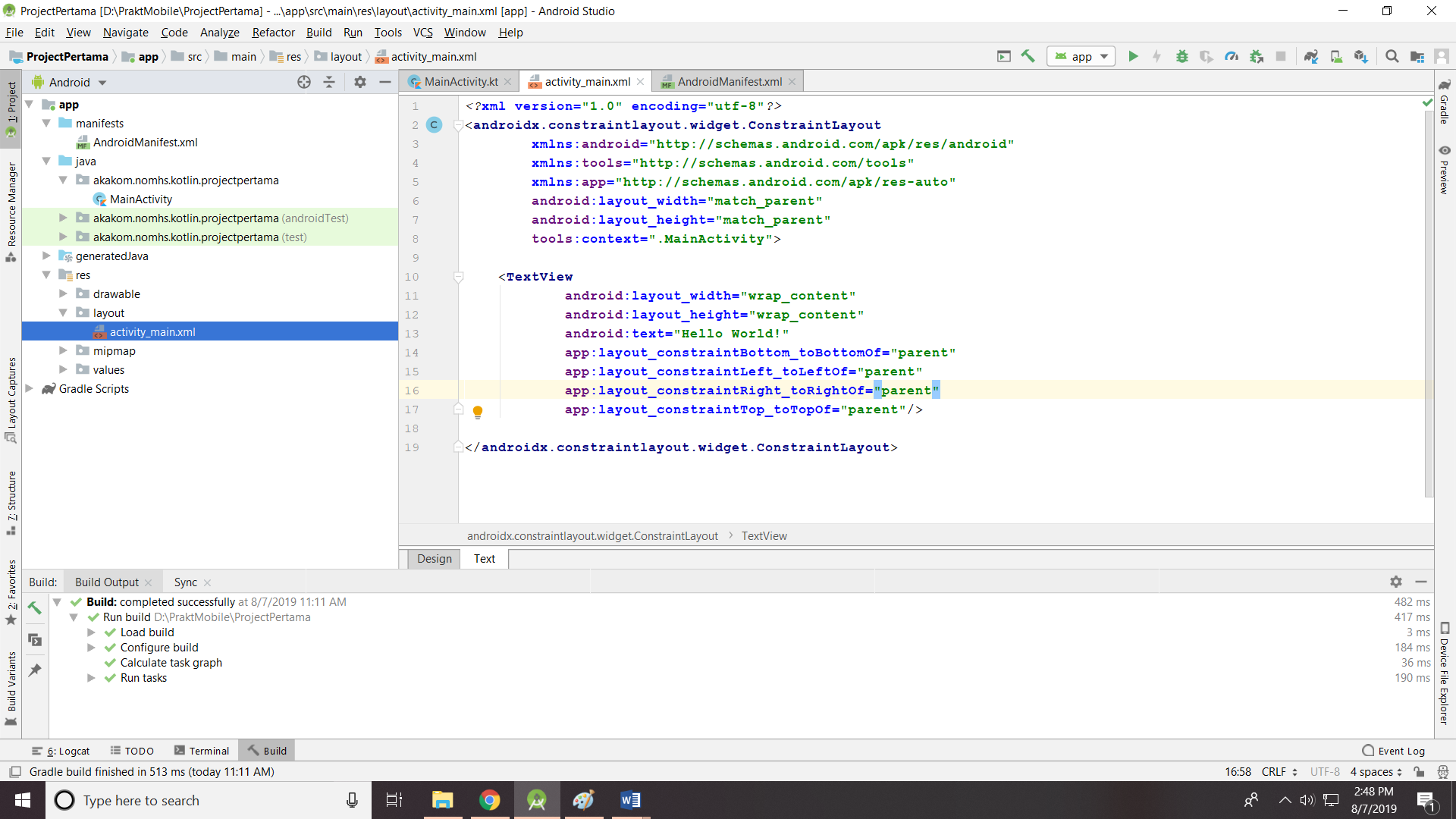
1. Pada MainActivity, terdapat beberapa koding yang dapat dijelaskan sebagai berikut.
2. Kelas MainActivity extends AppCompatActivity.

**class** MainActivity : AppCompatActivity() { …

1. 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.
2. 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.
3. Di onCreate (), kita menentukan layout mana yang dikaitkan dengan aktivitas, dan Anda mengembang tata letak. Metode setContentView () melakukan kedua hal itu.

**override fun** onCreate(savedInstanceState: Bundle?) {  
 **super**.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
}

1. 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 aset aplikasi, termasuk konten direktori res.
2. 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.
3. 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.
4. Tampilan adalah hal-hal seperti teks, gambar, dan tombol yang memperluas kelas tampilan. Ada banyak jenis tampilan, termasuk TextView, Button, ImageView, dan CheckBox.
5. Tampilan file layout adalah sebagai berikut.

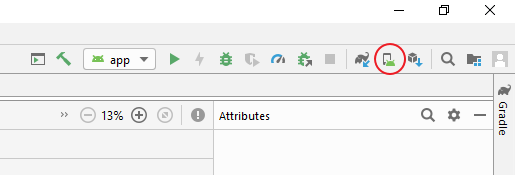


1. Untuk menjalankan aplikasi, dapat dilakukan dengan dua cara, pertama pada emulator, kedua pada perangkat mobile.
2. **Menjalankan pada Virtual Device.**

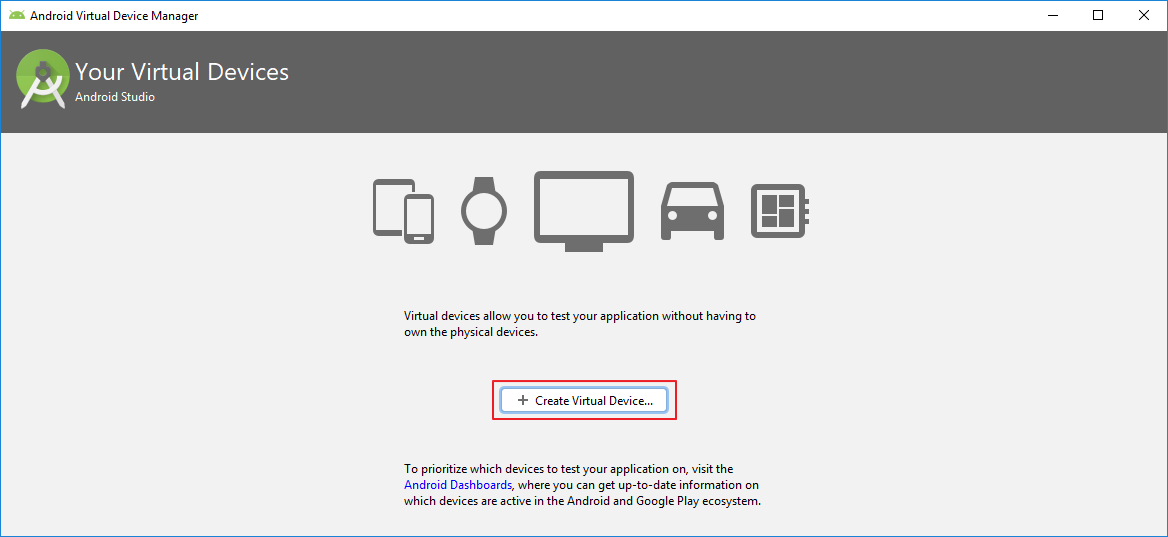
Telah dijelaskan sebelumnya tentang Virtual Device yaitu konfigurasi yang mendefinisikan karakteristik ponsel Android, tablet, Wear OS, atau perangkat Android TV yang ingin disimulasikan di Android Emulator.

Dalam membuat Virtual Device dapat melalui langkah-langkah berikut :

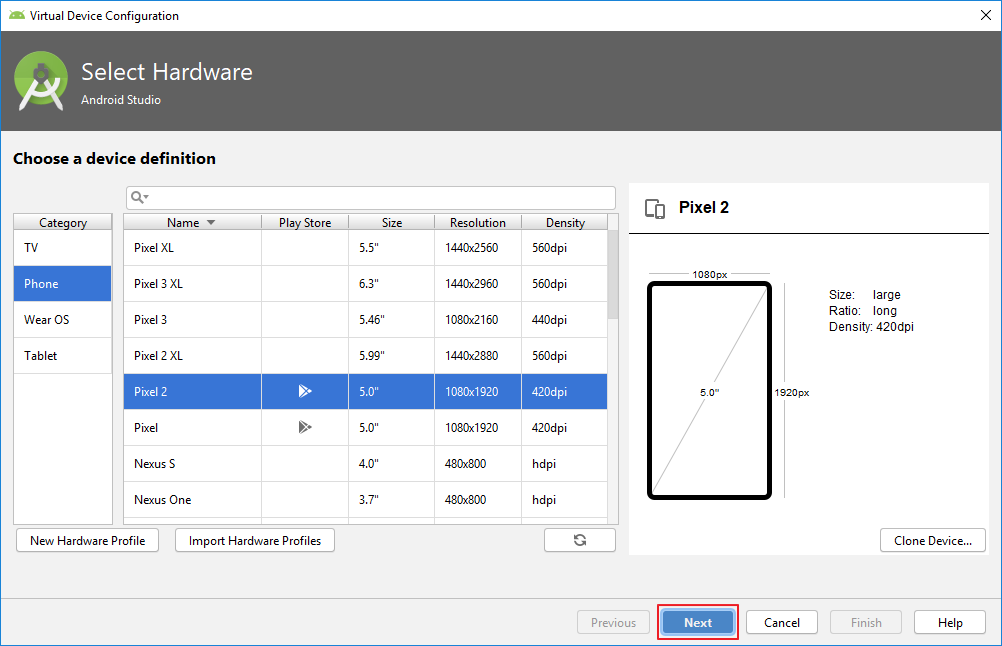
1. **Klik** pada **AVD Manger** yang terdapat pada toolbar Android Studio.



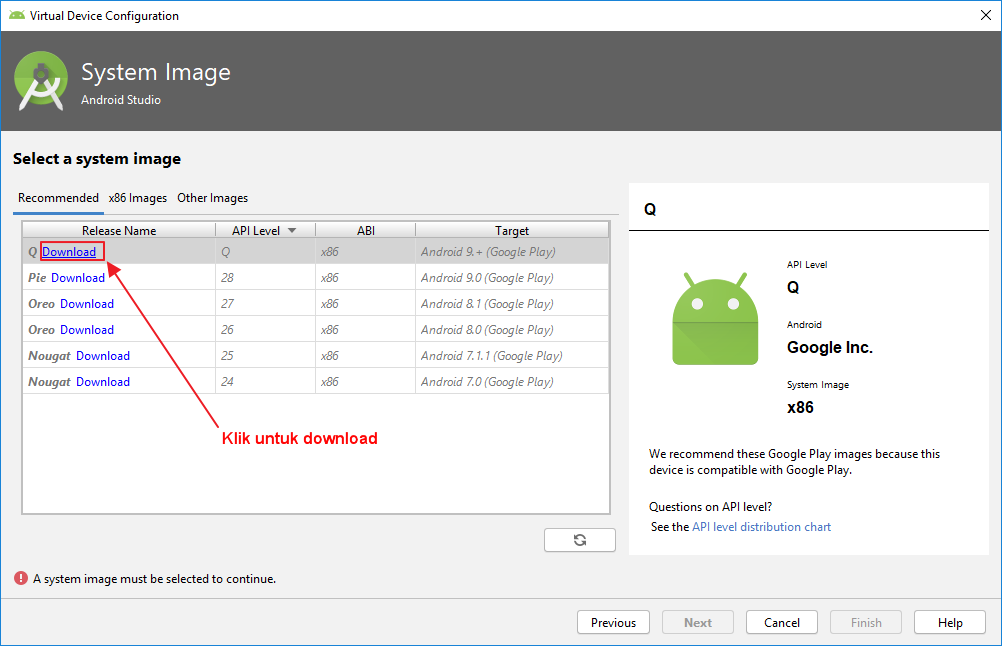
1. Pada jendela Android Virtual Device Manger, **klik** button **+ Create Virtual Device**.



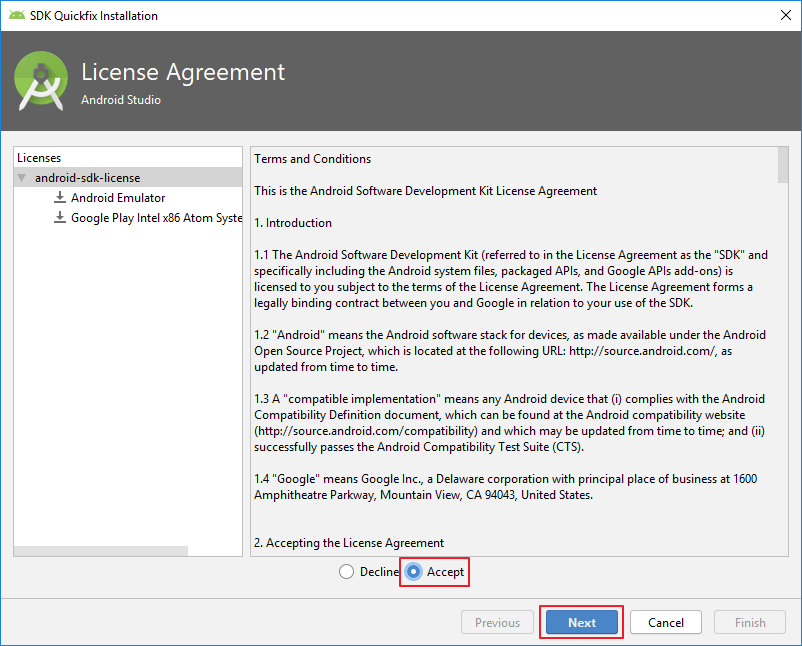
1. Pada jendela Virtual Device Configuration, pilih device yang akan digunakan untuk menampilkan hasil running project Android yang dibuat. Pilih pada **Pixel 2** kemudian **klik** button **Next**



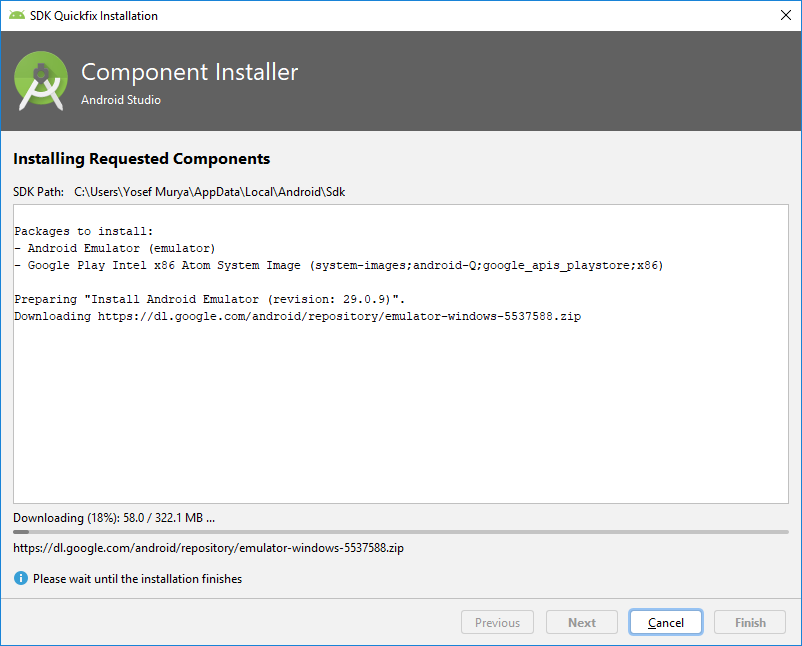
1. Jika pertamakali membuat emulator maka akan diminta untuk mendownload system images atau OS Android yang akan digunakan pada emulator. Silahkan **klik** pada bagian Q **download**



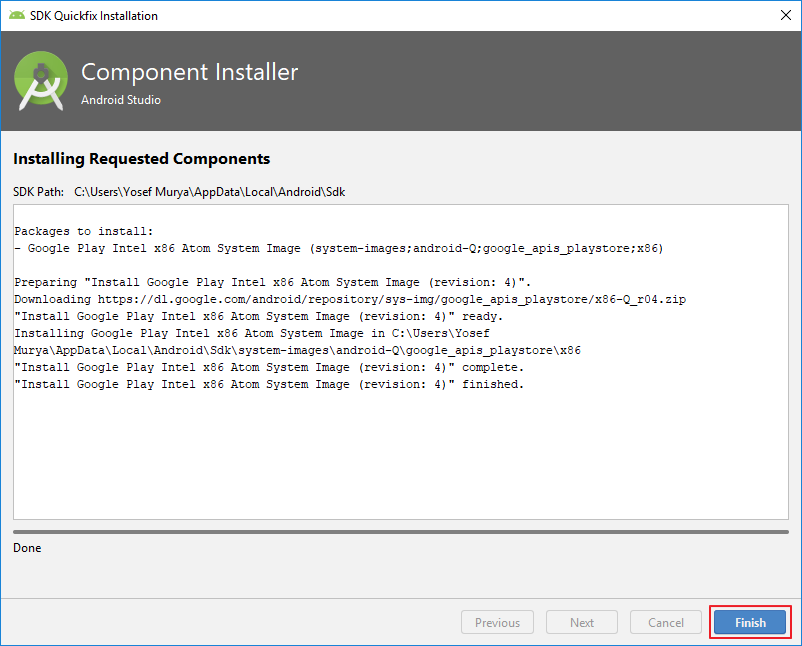
1. Pilih radio button **Accept** pada License Agreement, kemudian **klik** pada button **Next**



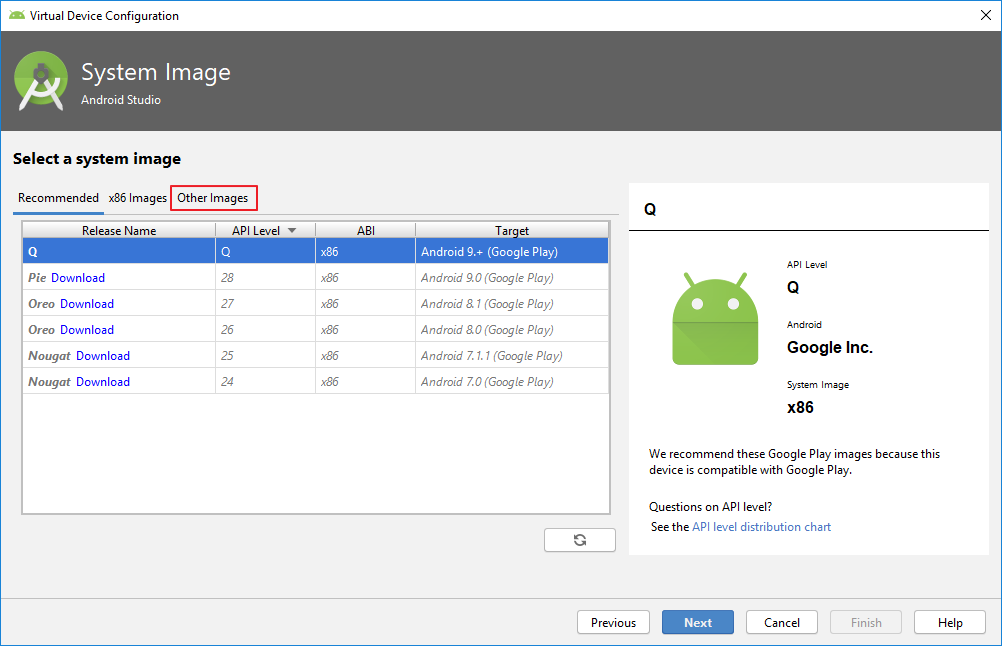
1. Tunggu beberapa saat sampai proses Download dan unzip selesai.



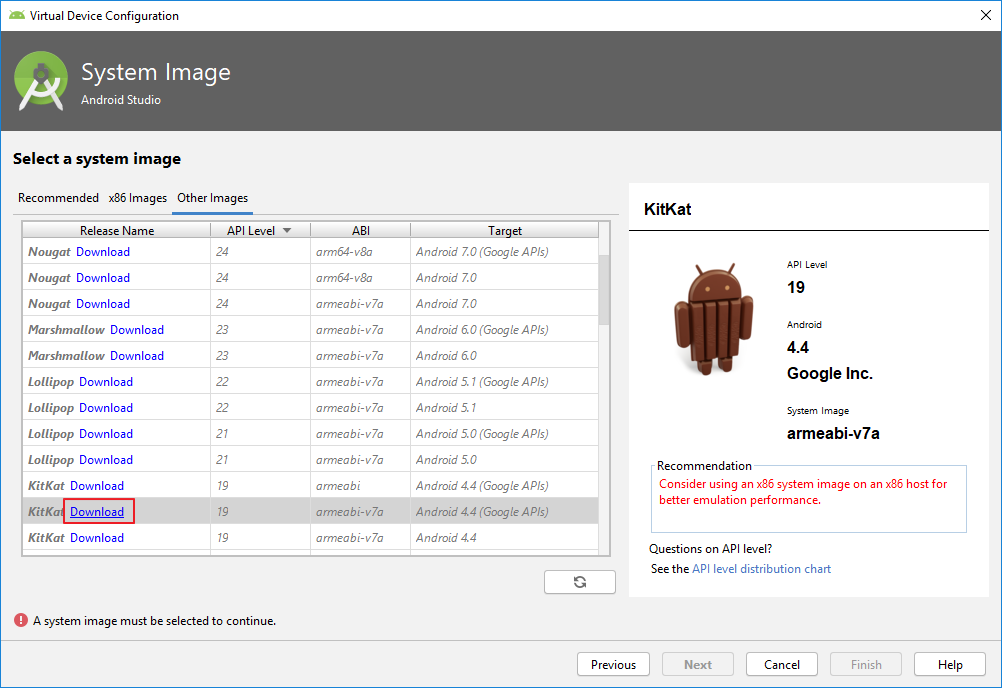
1. Setelah proses download dan unzipped selesai, **klik** button **Finish**



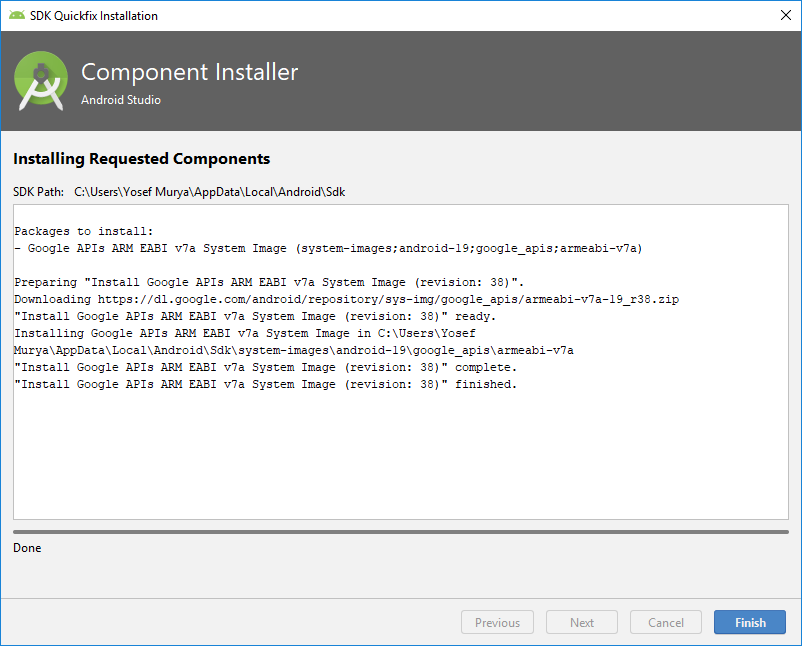
1. Langkah selanjutnya, **pilih** pada system image **Q** yang telah diinstall kemudian **klik** menu tab **other images**.



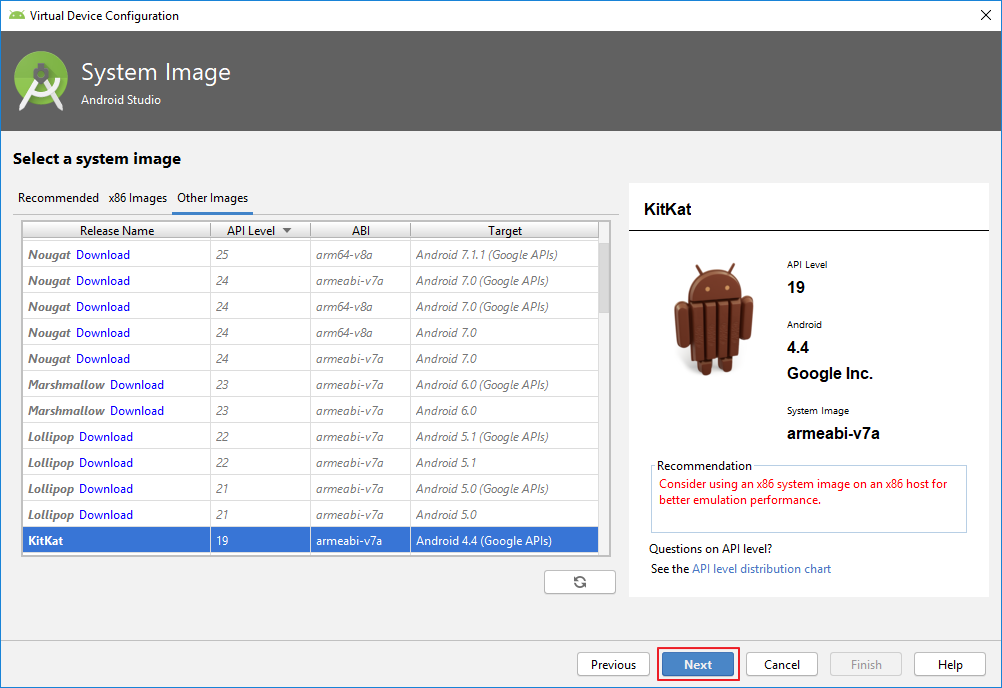
1. Kemudian sesuaikan dengan rekomendasi (Recommendation), agar emulator dapat dijalankan dengan mudah. Dikarenakan hardware yang digunakan untuk menjalankan emulator Android yaitu Intel Dual Code dengan RAM 4GB maka penulis memilih system image Kitkat dengan API 19, ABI armeabi-v7a, selanjutnya **klik**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



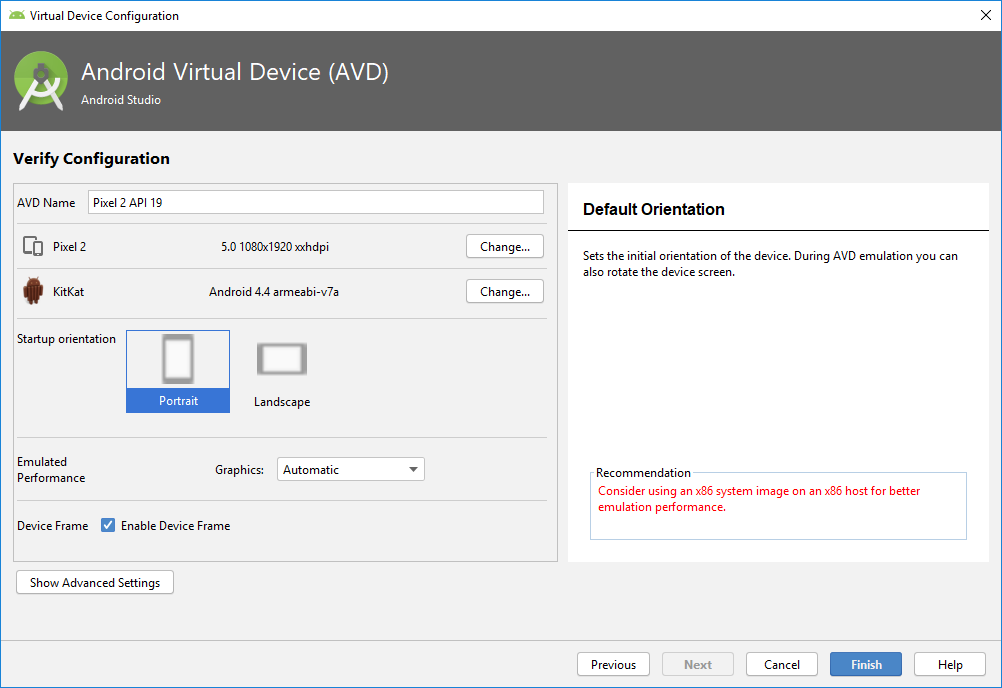
1. Proses download memerlukan waktu (pastikan anda terkoneksi internet), tunggu sampai proses download dan unzipped selesai, kemudian **klik** button **Finish**.



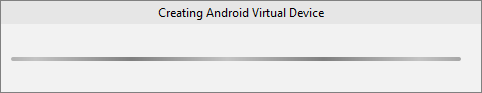
1. Maka akan terlihat system image KitKat telah berhasil didownload, **klik** button **Next** untuk melanjutkan ke proses selanjutnya.



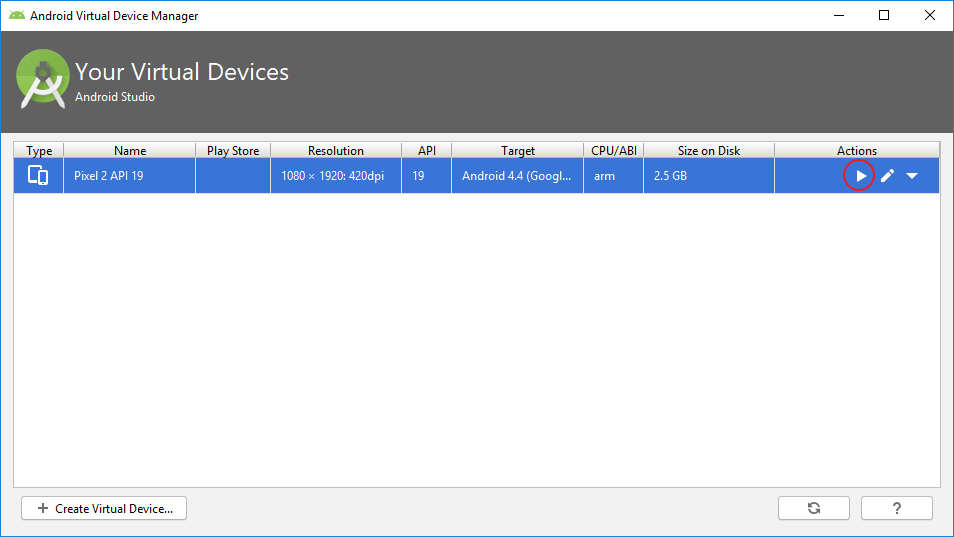
1. Pada bagian AVD Name akan terlihat secara default dengan nama Pixel 2 API 19 (dapat diganti dengan nama lain sesuai dengan kebutuhan), lalu pada bagian Startup orientation terdapat pilihan antara Potrait dan Landscape, secara default akan terpilih Potrait. **Klik** button **Finish**.



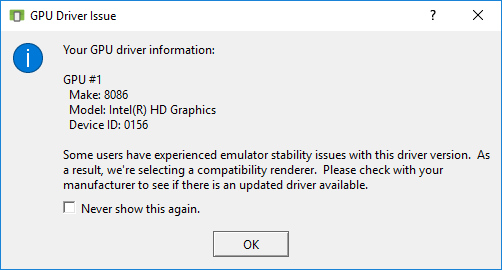
1. Proses pembuatan Android Virtual Device membutuhkan waktu beberapa detik, silahkan tunggu sampai proses pembuatan Android Virtual Device selesai.



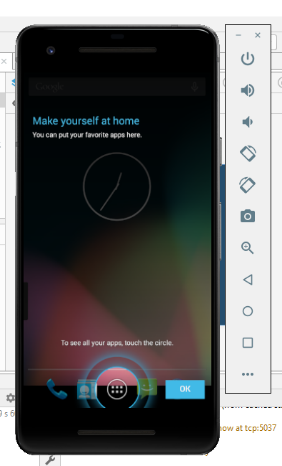
1. Jika pembuatan Android Virtual Device telah selesai dilakukan maka akan muncul Android Virtual Device yang telah dibuat seperti terlihat pada gambar dibawah ini. Untuk menjalankan Virutal Device, **klik** pada **icon play**.



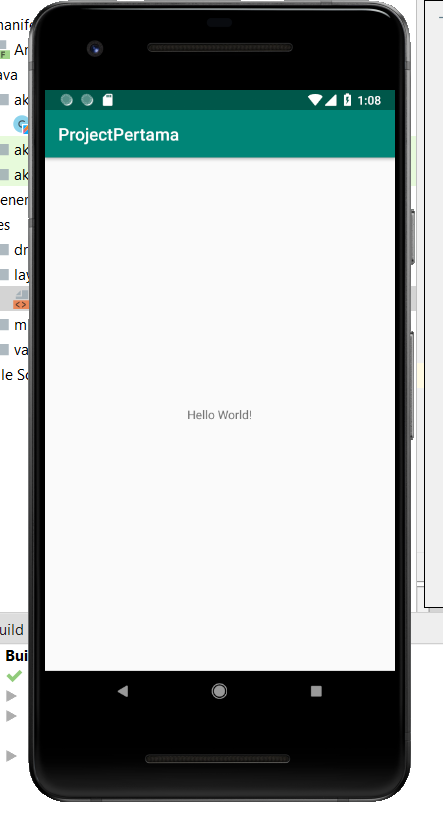
1. Pada bagian GPU Driver Issue akan terlihat informasi tentang Graphic Processing Unit pada laptop atau komputer anda, **klik** button **OK**.



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



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



1. **Physical Device.**

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

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.

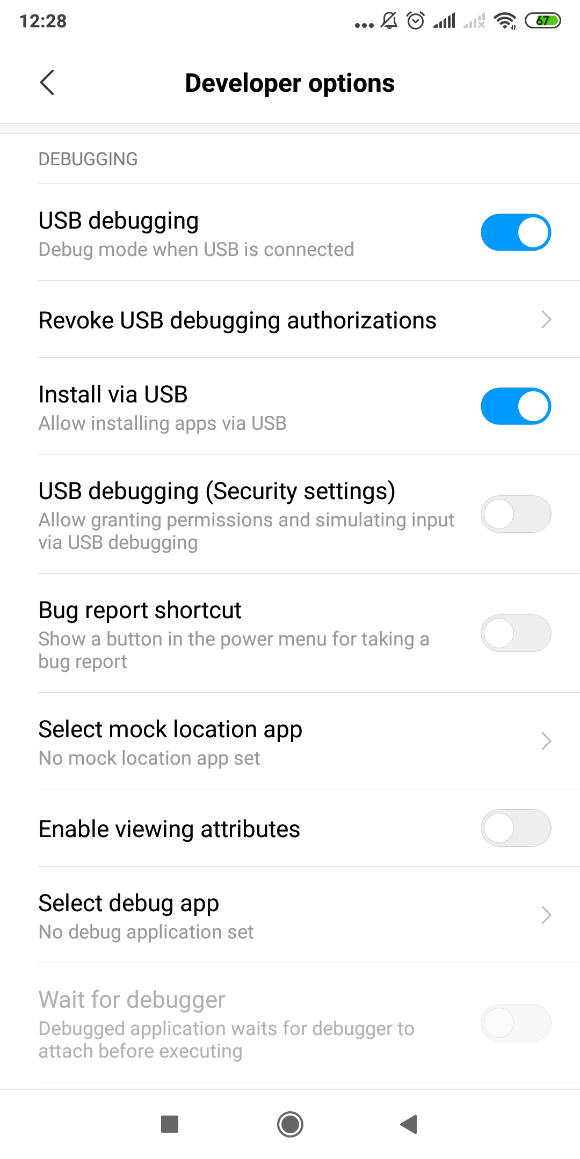
1. Pastikan peranti yang akan dipakai sesuai dengan target SDK atau paling tidak mendukung versi SDK terendah yang digunakan aplikasi.
2. Buka *setting*dan masuk ke dalam menu **About**. Pada halaman menu ini, Anda perlu menemukan informasi tentang **Build number***.*

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).



1. 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.
2. Kemudian aktifkan USB Debugging sampai pada bagian USB debugging terlihat aktif.



1. Jalankan aplikasi Anda pada perangkat handphone.

|  |  |
| --- | --- |
|  | **LATIHAN** |

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

|  |  |
| --- | --- |
| **tugas.png** | **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>

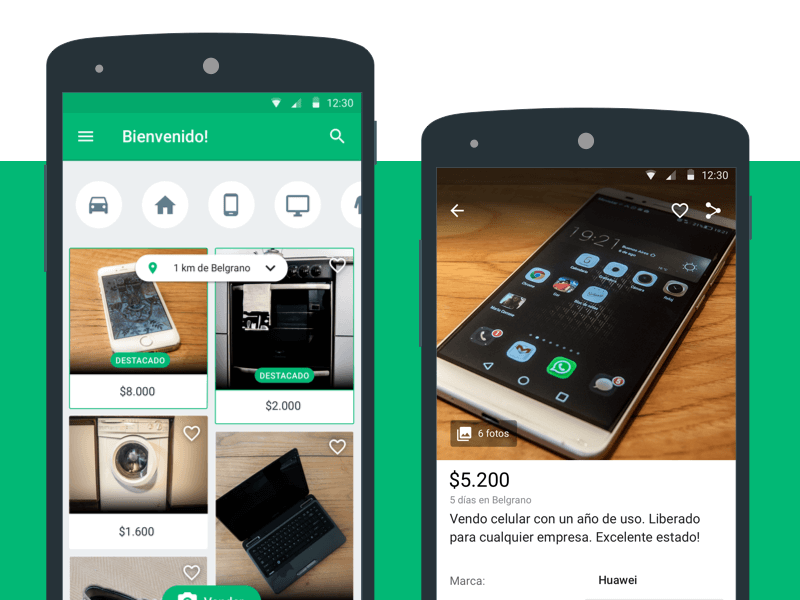
# MODUL 3 Layout dengan Linear Layout dan Constrain Layout

|  |  |
| --- | --- |
|  | 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 menggambar komponen tampilan ke layar yang mana pengguna dapat melihat dan berinteraksi langsung.

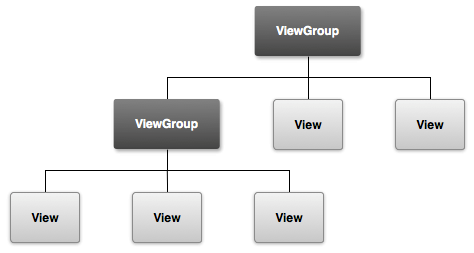
Contoh komponen turunan dari view seperti :

* **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 komponenviewgroup 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 :

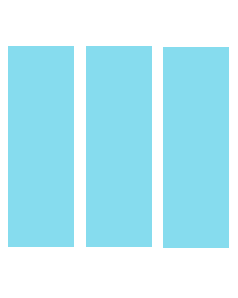
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3. android:layout\_width="match\_parent"
4. android:layout\_height="match\_parent"
5. android:orientation="vertical" >
6. <TextView android:id="@+id/text"
7. android:layout\_width="wrap\_content"
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 viewgroupLinearLayout menjadi kontainer untuk obyek turunan view, button, dan textview. Beberapa komponen viewgroup seperti linearlayout, relativelayout, framelayout,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.

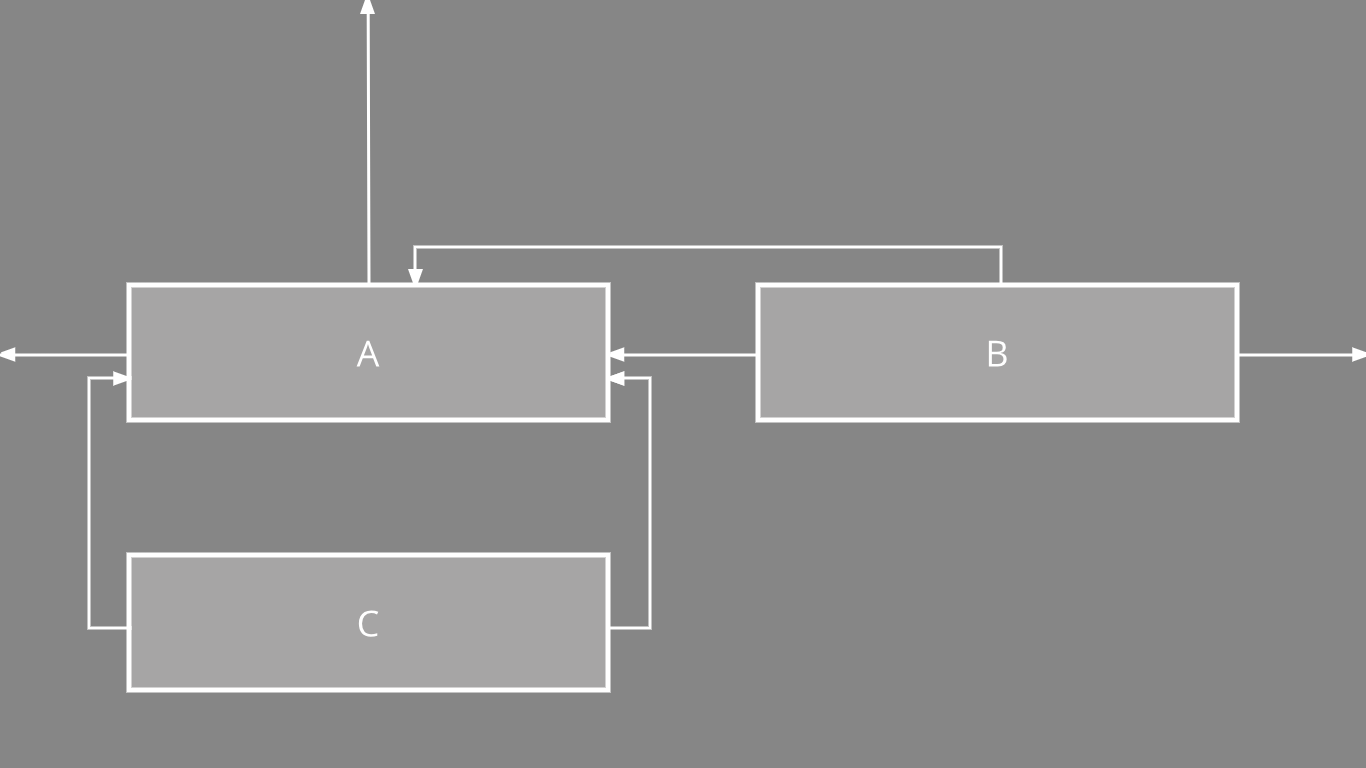
**ConstraintLayout** memiliki kesamaan dengan RelativeLayout. Dalam penggunaan semua view yang berada di dalamnya disusun berhubungan antara parent dan view lainnya. Tapi **ConstraintLayout** lebih fleksibel dari **RelativeLayout** dan mudah digunakan dengan dukungan Layout Editor pada Android Studio.

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.

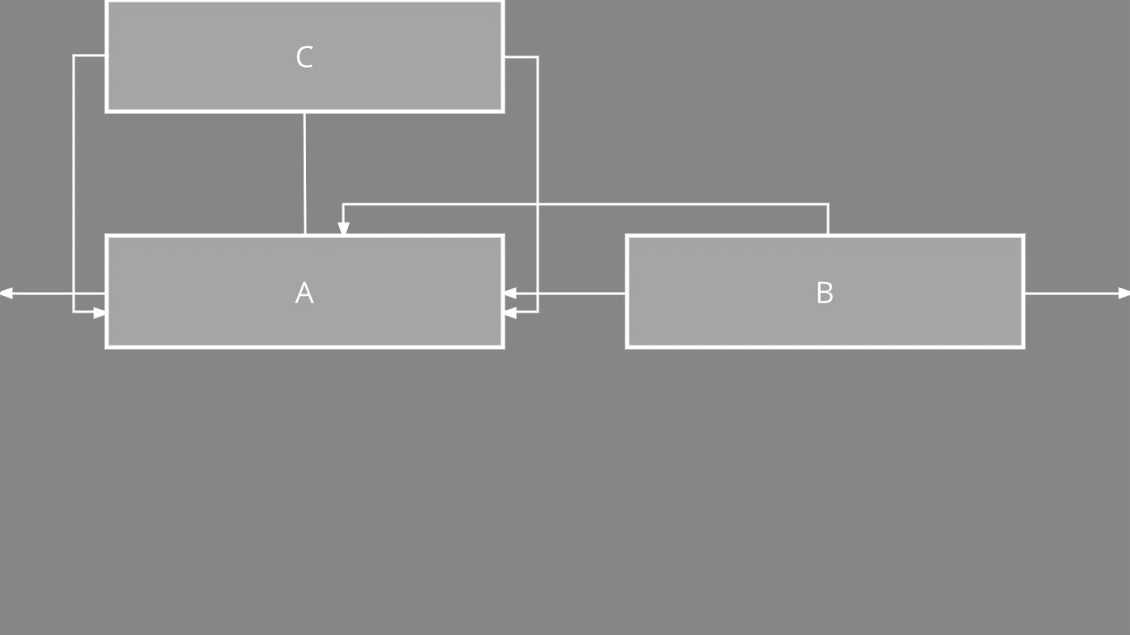
Kenapa gerangan? Karena setelah ditambahkan, view tersebut tidak memiliki ***constraint*** yang menghubungkannya dengan parent layout atau view lainnya. Sehingga ketika dijalankan, posisi dari view tersebut akan berada di bagian atas sebelah kiri.

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.

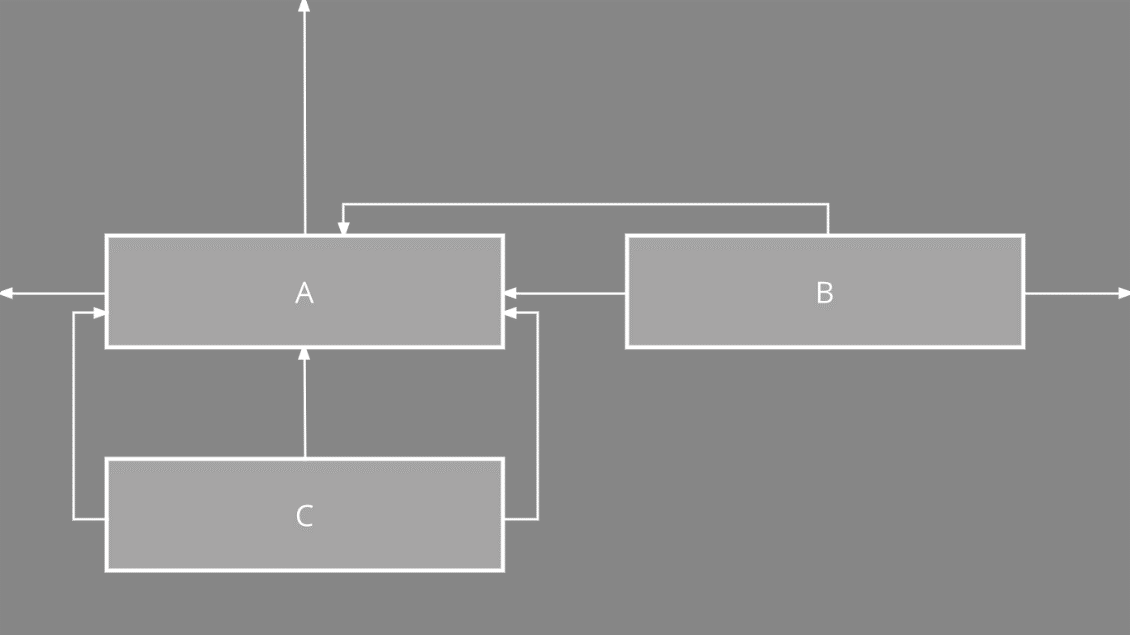
Setiap view setidaknya memiliki satu vertikal dan horizontal *constraint*. Misal kita memiliki sebuah layout dengan tampilan pada Layout Editor seperti berikut:



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:



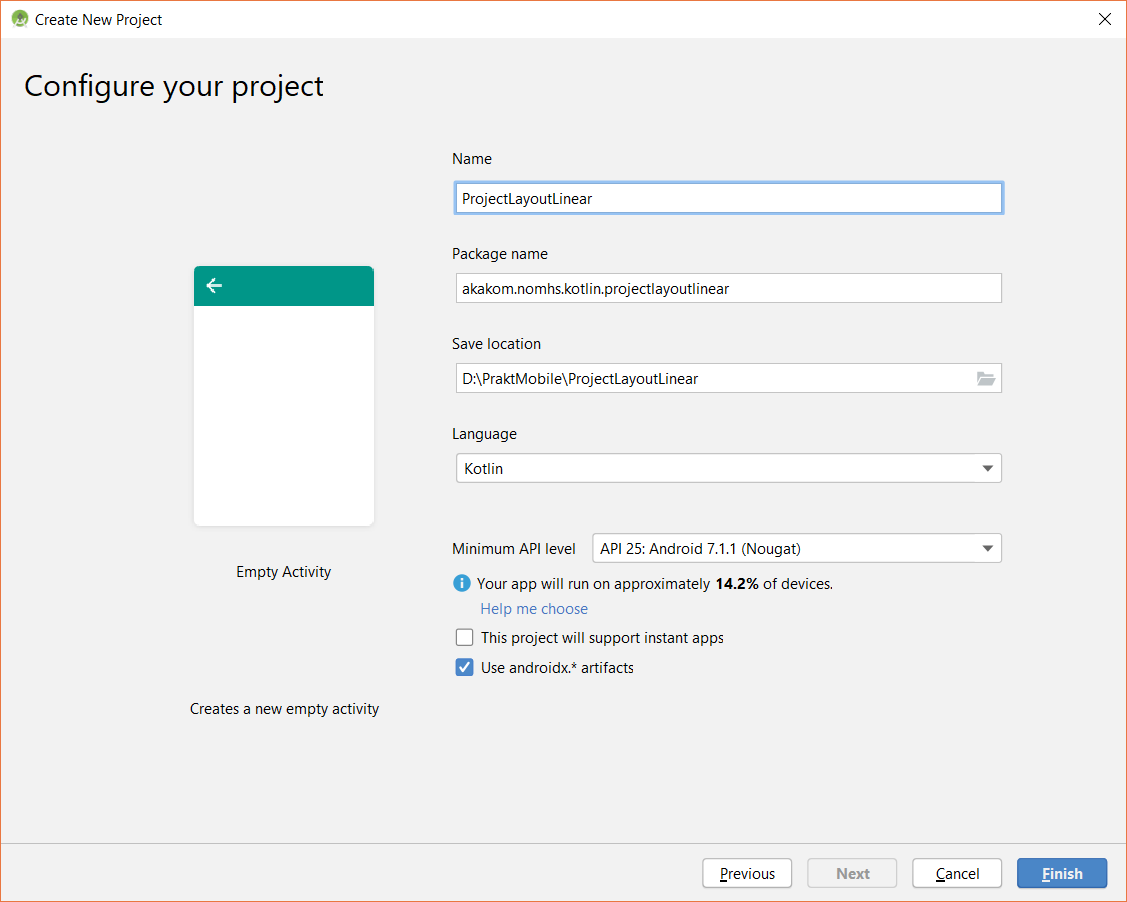
Berbeda jika kita menambahkan vertikal constraint pada **view C** yang diatur terikat dengan**view A** seperti berikut:



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**.



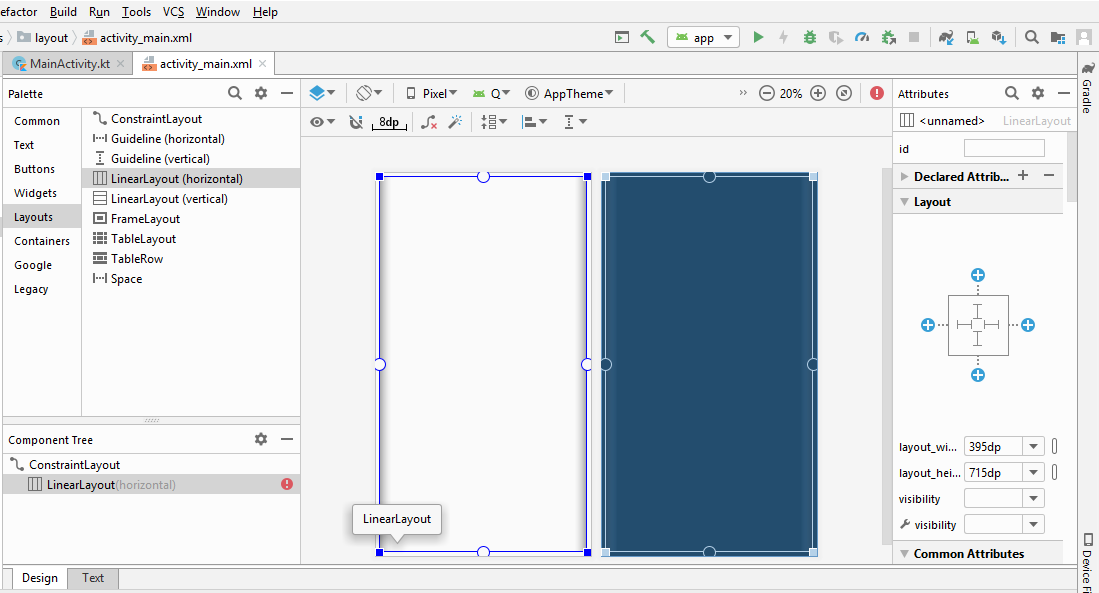
1. Langkah selanjutnya, buka file activity\_main.xml yang terdapat pada app 🡪 res 🡪 layout.
2. Secara default pada saat membuat project baru dengan nama ProjectLayoutLinear maka akan muncul teks Hello World.
3. Buka file activity\_main.xml kemudian pilih tab text, akan terlihat koding sebagai berikut.

*<?***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**>

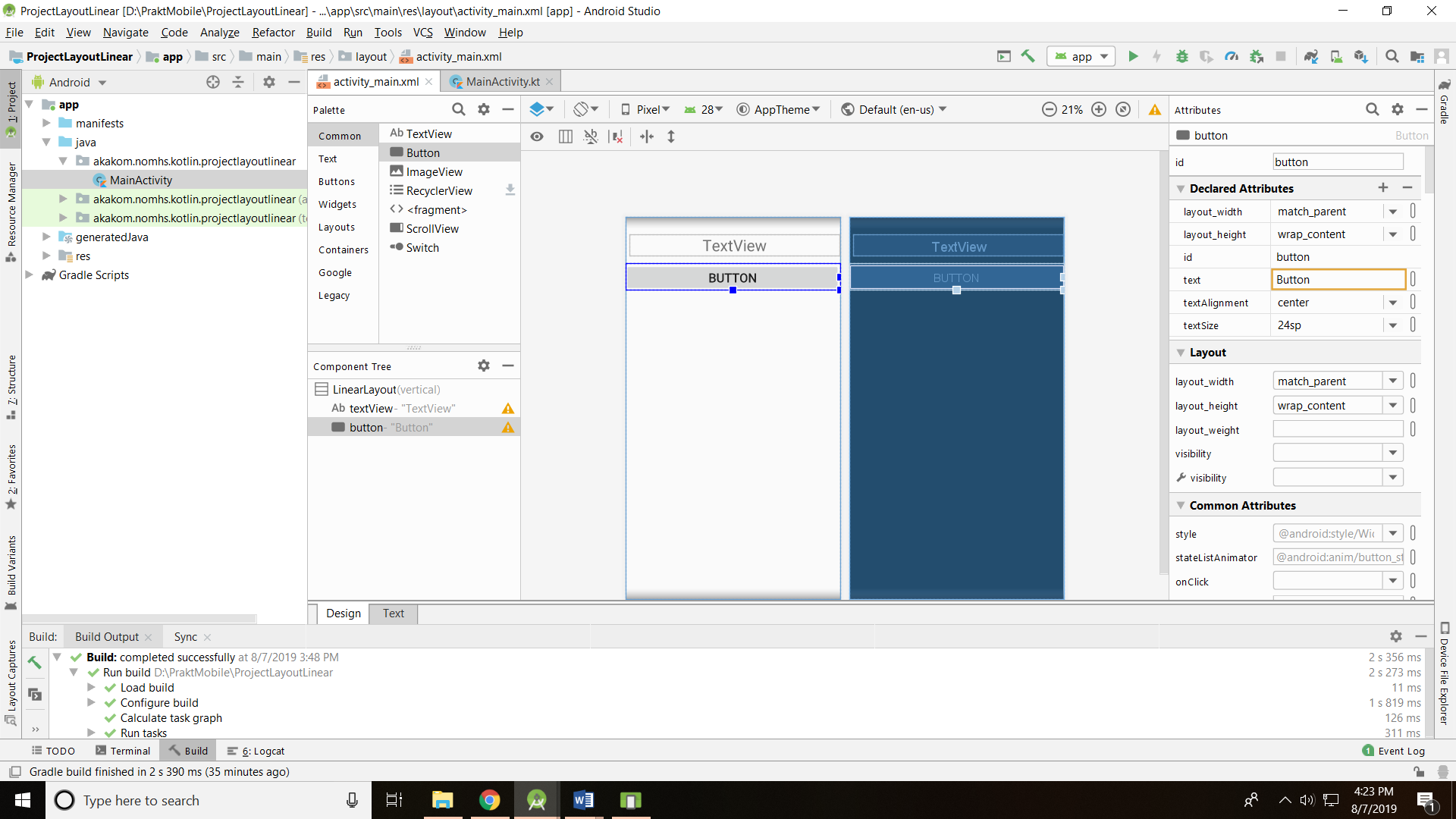
1. Standar layout pertama adalah Constrain Layout. Ubahlah menjadi Linear Layout dengan koding berikut. Perhatikan atribut **orientation**.

*<?***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**>

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



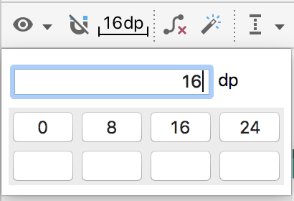
1. Tambahkan komponen TextView dan Button, sehingga menjadi sebagai berikut.



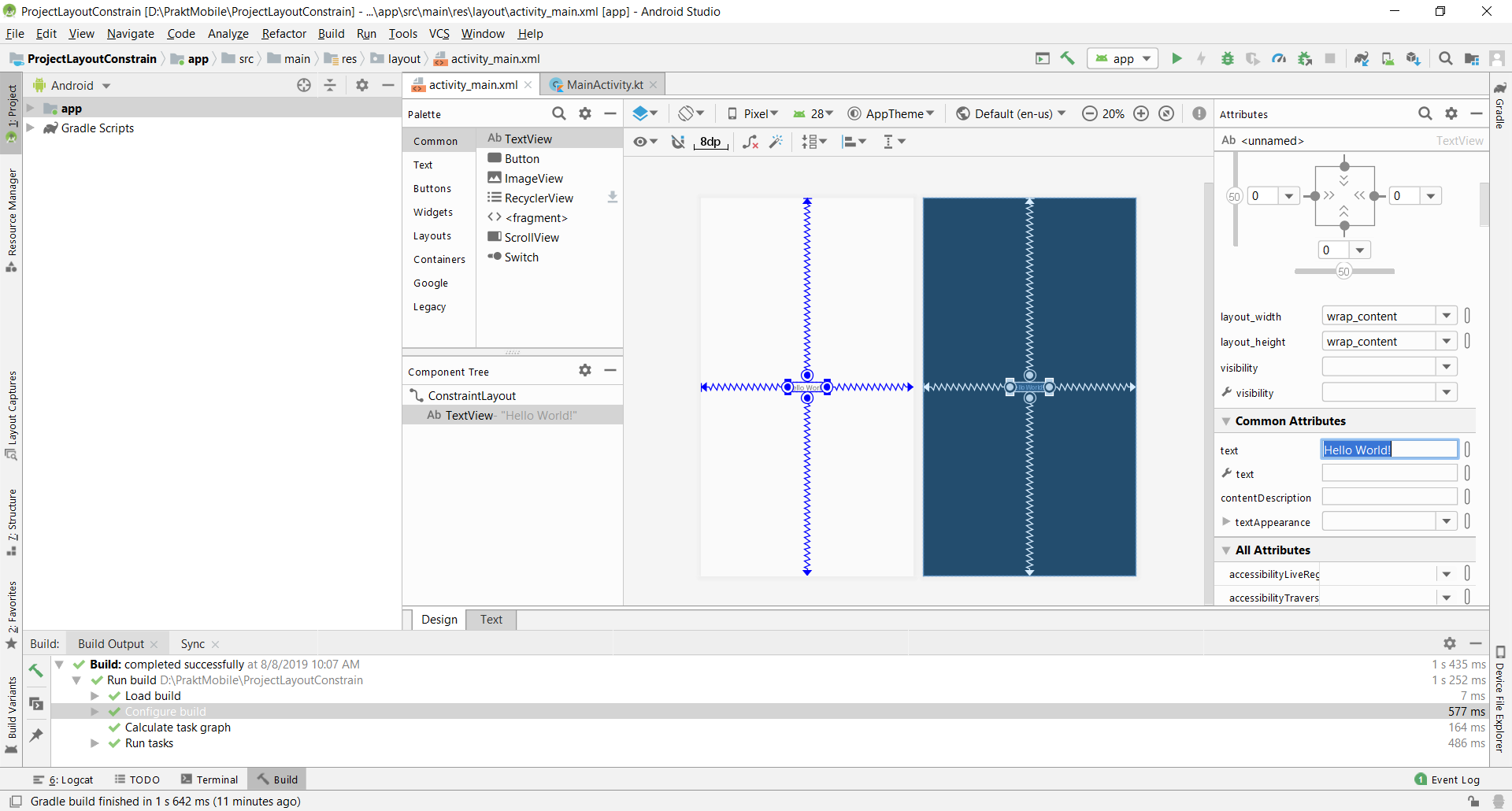
1. Jalankan dan amati hasilnya.
2. Ubah atribut-atribut yang ada dan jalankan lagi, amati perubahannya.
3. Kita akan mencoba menggunakan Constrain Layout. Buat project baru. Beri nama ProjectLayoutConstrain.
4. Buat Project baru dengan nama ProjectLayoutConstrain.
5. Buka file activity\_main.xml dan klik tab Desain.
6. 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.

|  |  |
| --- | --- |
| https://codelabs.developers.google.com/codelabs/kotlin-android-training-constraint-layout/img/56d39c18cc3fd857.png | Autoconnect is on. |
| https://codelabs.developers.google.com/codelabs/kotlin-android-training-constraint-layout/img/6308c97e93758821.png | Autoconnect is off |

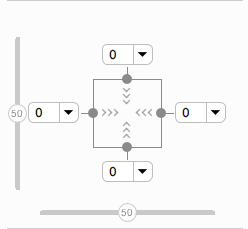
1. Gunakan toolbar untuk mengatur default margins ke 16dp. (Defaultnya adalah 8dp.)



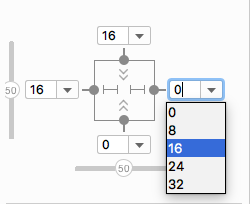
1. Ketika Anda mengatur margin default ke 16dp, constrain baru dibuat dengan margin ini, jadi Anda tidak perlu menambahkan margin setiap kali Anda menambahkan constrain.
2. Perbesar menggunakan ikon + https://codelabs.developers.google.com/codelabs/kotlin-android-training-constraint-layout/img/f422665b6659ecec.png di sebelah kanan toolbar, hingga teks Hello World terlihat di dalam tampilan teksnya.
3. Klik dua kali pada tampilan teks Hello World untuk membuka panel Attributes.



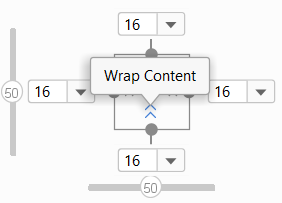
1. View Inspector, yang ditunjukkan pada gambar di bawah, adalah bagian dari panel Atribut. View Inspector mencakup kontrol untuk atribut layout seperti constrain, constraint types, constraint bias, and view margins.



1. 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.
2. 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 inspektur tampilan. Misalnya, dalam tangkapan layar berikut ini Anda menambahkan layout\_marginEnd (layout\_marginRight).

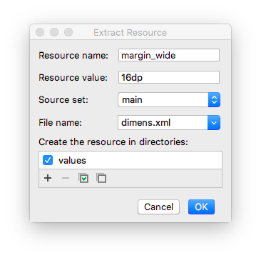


1. Sesuaikan batasan dan margin untuk TextView. Di view inspector, panah https://codelabs.developers.google.com/codelabs/kotlin-android-training-constraint-layout/img/630e06bb70ffdf9.png di dalam kotak mewakili tipe constrain:
   1. https://codelabs.developers.google.com/codelabs/kotlin-android-training-constraint-layout/img/630e06bb70ffdf9.png wrap content: Tampilan hanya selebar kontennya.



* 1. https://codelabs.developers.google.com/codelabs/kotlin-android-training-constraint-layout/img/c7555e729427ab7d.png fixed: Anda dapat menentukan dimensi sebagai margin tampilan di kotak teks di sebelah panah constrain tetap.
  2. https://codelabs.developers.google.com/codelabs/kotlin-android-training-constraint-layout/img/99fb246a63cb4c55.png 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.

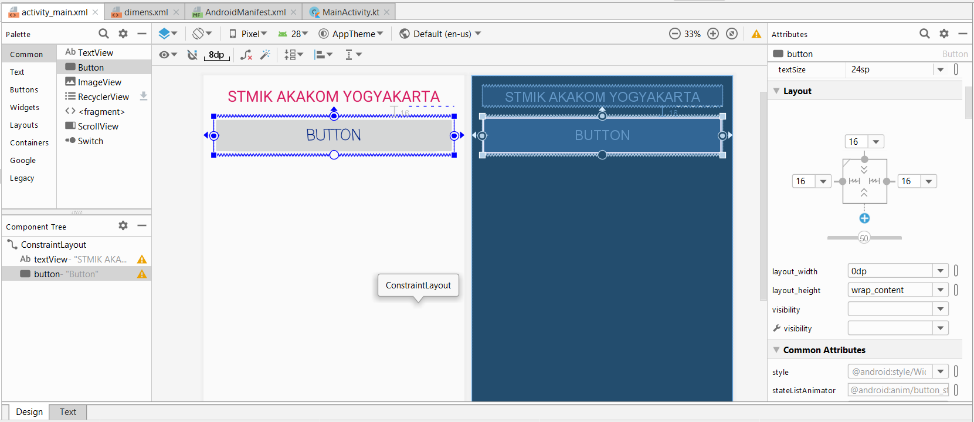
1. Di view inspector, ubah constrain kiri dan kanan ke Match Constraints. (Klik simbol panah untuk beralih di antara jenis constrain.)
2. Pindah ke tab Text, Ekstrak resource dimensi untuk layout\_marginStart, dan atur nama Resource ke margin\_wide. (blok pada bagian isian dari layout\_marginStart, pilih extract dimensions resource).



1. Kerjakan untuk layout\_marginEnd dan layout\_marginTop. Hasilnya dalah sebagai berikut

**android:layout\_marginTop="@dimen/margin\_wide"  
android:layout\_marginStart="@dimen/margin\_wide"  
android:layout\_marginEnd="@dimen/margin\_wide"**

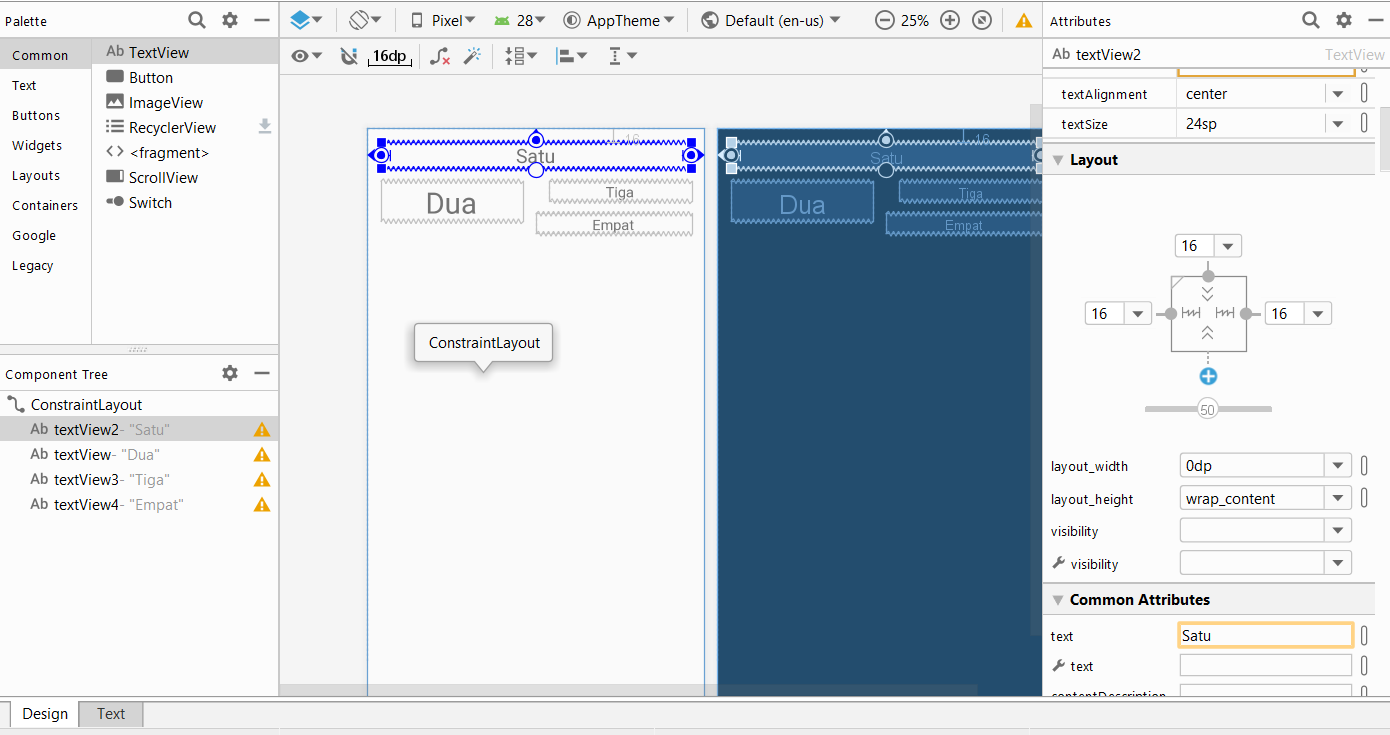
1. Kemudian, tambahkan satu button, atur constrainnya. Hasilnya sebagai berikut.



1. Jalankan dan amati hasilnya.
2. Ubah atribut-atribut yang ada dan jalankan lagi, amati perubahannya.

|  |  |
| --- | --- |
|  | **LATIHAN** |

1. Buat project baru dengan desain sebagai berikut.



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

|  |  |
| --- | --- |
| **tugas.png** | **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**

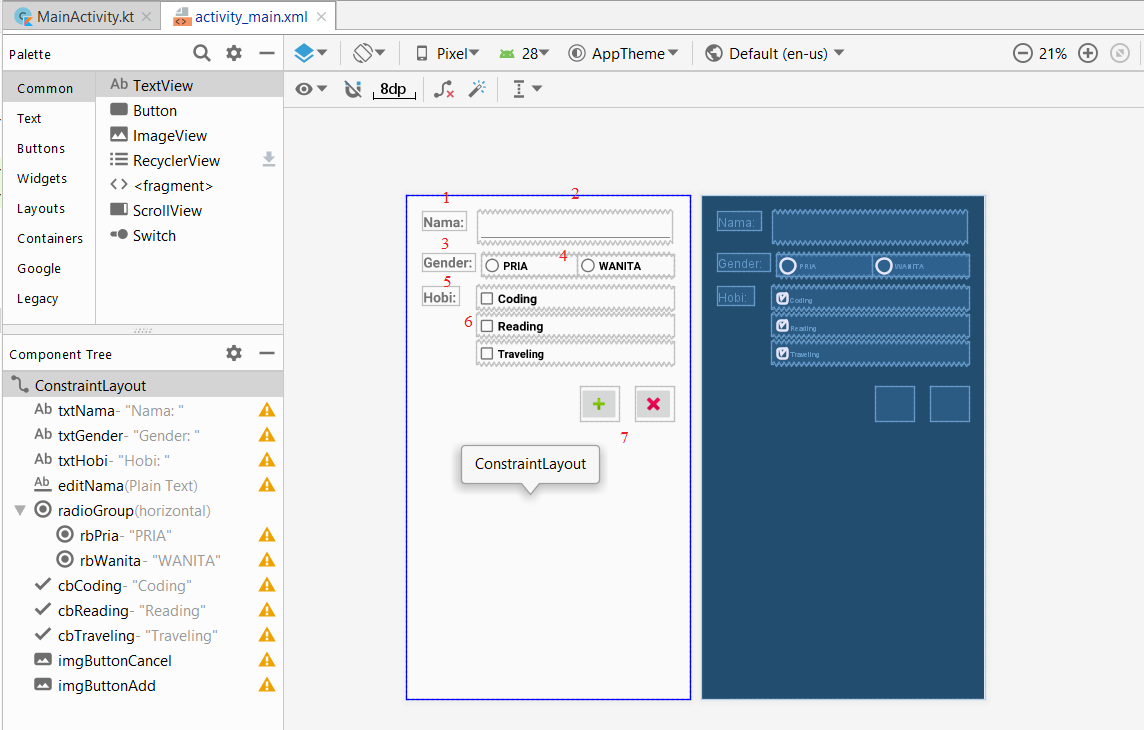
Event listener merupakan antarmuka di kelas [View](https://developer.android.com/reference/android/view/View.html) yang berisi method callback tunggal. Method ini akan dipanggil oleh framework Android jika View yang telah didaftarkan dengan listener dipicu oleh interaksi pengguna dengan item dalam UI.

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

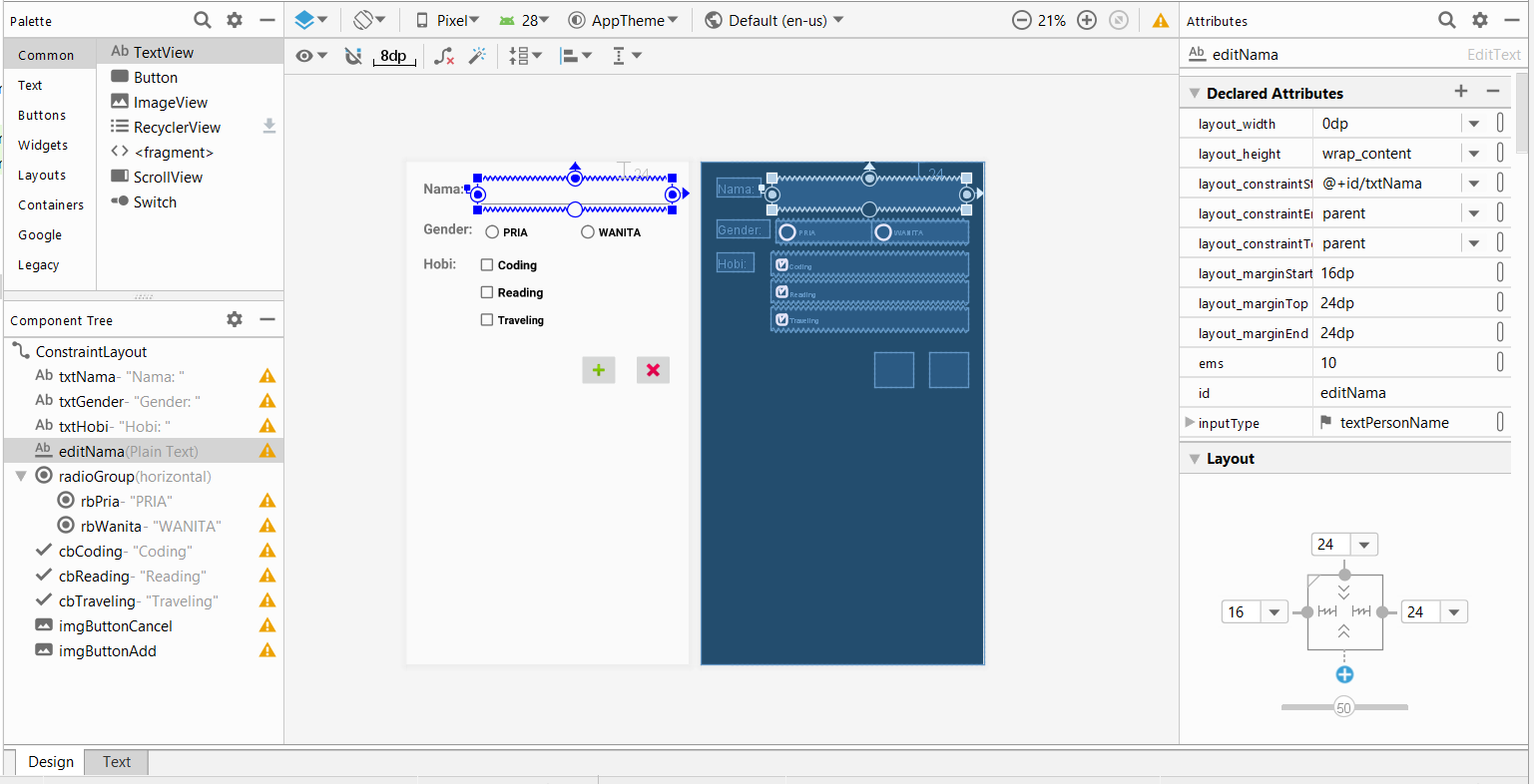
1. Method onClick() dari [View.OnClickListener](https://developer.android.com/reference/android/view/View.OnClickListener.html). Ini dipanggil baik saat pengguna menyentuh item (jika dalam mode sentuh), maupun memfokuskan pada item dengan tombol navigasi atau trackball dan menekan tombol "enter" yang sesuai atau menekan trackball.
2. Method onLongClick() dari [View.OnLongClickListener](https://developer.android.com/reference/android/view/View.OnLongClickListener.html). 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](https://developer.android.com/reference/android/view/View.OnFocusChangeListener.html). Ini dipanggil saat pengguna menyusuri ke atau dari item, dengan menggunakan tombol navigasi atau trackball.
4. Method onKey() dari [View.OnKeyListener](https://developer.android.com/reference/android/view/View.OnKeyListener.html). Ini dipanggil saat pengguna memfokuskan pada item dan menekan atau melepas tombol perangkat keras pada perangkat.
5. Method onTouch() dari [View.OnTouchListener](https://developer.android.com/reference/android/view/View.OnTouchListener.html). 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](https://developer.android.com/reference/android/view/View.OnCreateContextMenuListener.html). 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.



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



1. Untuk keseluruhan komponen widget, perhatikan nilai atribut di setiap komponen di bawah.
2. 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"**/>

1. 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"**/>

1. 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**>

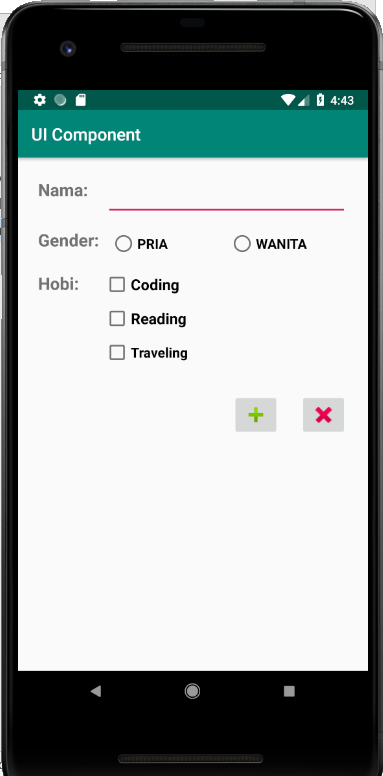
1. Nomor 6 ada 3 komponen CheckBox.

<**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"**/>

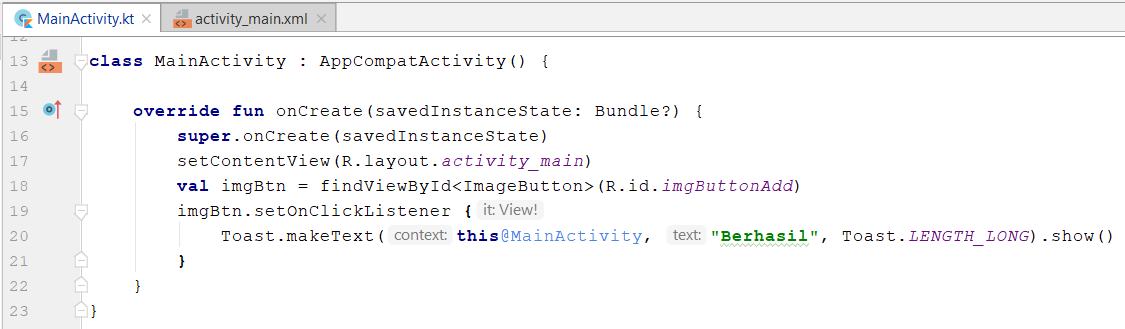
1. Nomor 7 adalah 2 komponen ImageButton. Perhatikan bahwa untuk ImageButton diperlukan file gambar, dalam contoh dibawah file gambarnya adalah ic\_delete dan ic\_input\_add.

<**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"**/>

1. Setelah selesai desain, jalankan. Hasilnya dalah sebagai berikut.



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



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

|  |  |
| --- | --- |
|  | **LATIHAN** |

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

|  |  |
| --- | --- |
| **tugas.png** | **TUGAS** |

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>

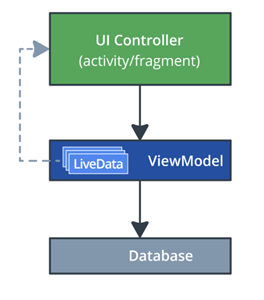
# MODUL 5 RecycleView

|  |  |
| --- | --- |
|  | CAPAIAN PEMBELAJARAN |
| Mahasiswa dapat merepresentasikan data dengan menggunakan komponen recylerview | |

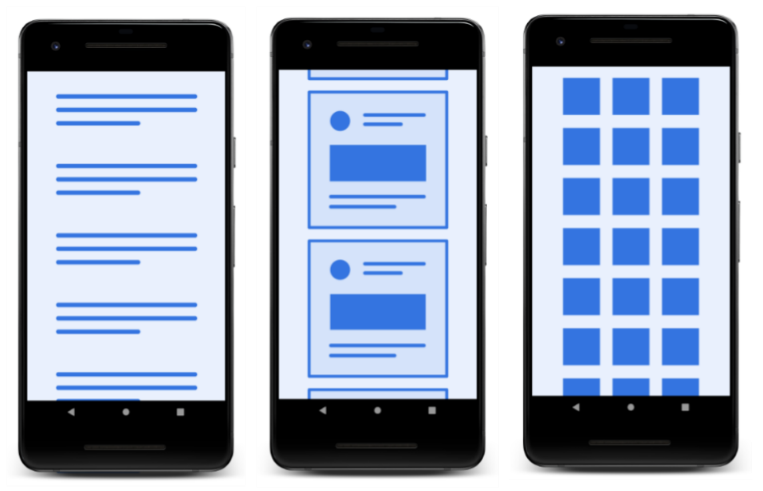
|  |  |
| --- | --- |
|  | KEBUTUHAN ALAT/BAHAN/SOFTWARE |
| 1. Android Studio 3.4. 2. Handphone Android versi 7.0 (Nougat) 3. Kabel data USB. 4. Driver ADB. | |

|  |  |
| --- | --- |
|  | DASAR TEORI |

RecycleView 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.



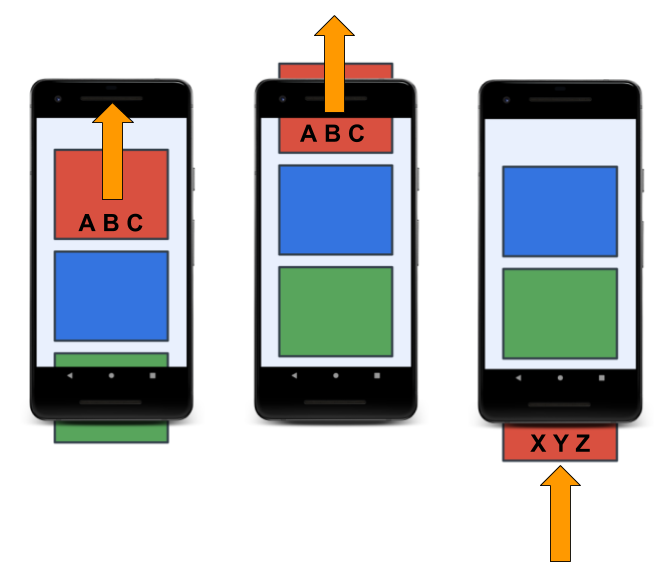
Manfaat terbesar dari RecyclerView adalah sangat efisien untuk daftar besar:

• 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!

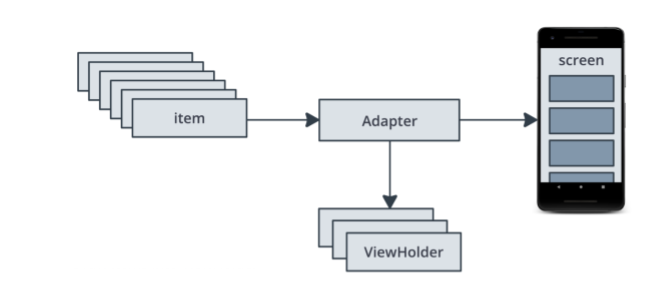
Dalam urutan yang ditunjukkan di bawah ini, kita dapat melihat bahwa satu tampilan telah diisi dengan data, ABC. Setelah itu tampilan bergulir dari layar, RecyclerView menggunakan kembali tampilan untuk data baru, XYZ.



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.

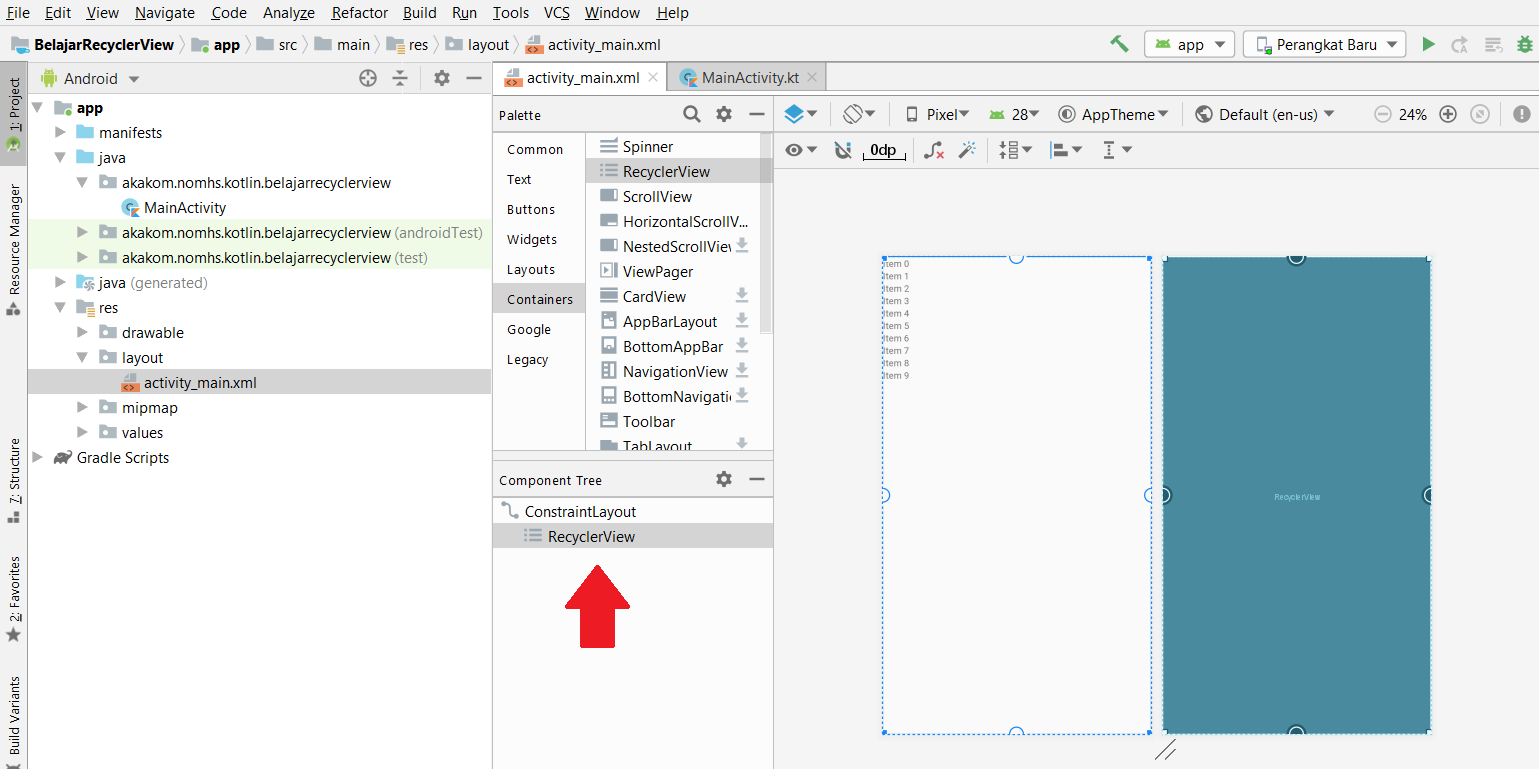
  view holder extends kelas ViewHolder. Ini berisi informasi tampilan untuk menampilkan satu item dari layout item. Penampil tampilan juga menambahkan informasi yang digunakan RecyclerView untuk memindahkan tampilan di layar secara efisien.

• Adaptor.

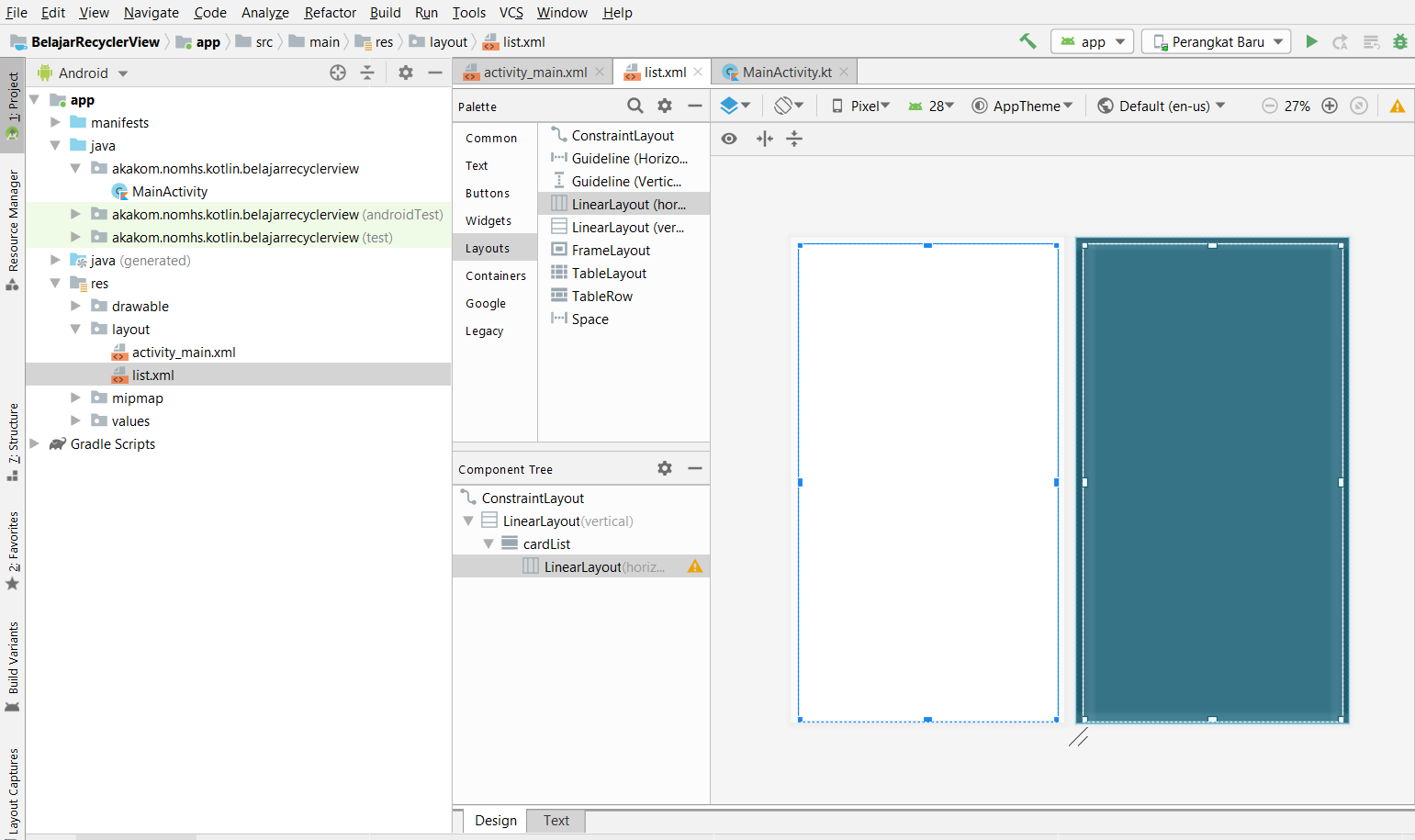
Adaptor menghubungkan data kita ke RecyclerView. Ini menyesuaikan data sehingga dapat ditampilkan di ViewHolder. RecyclerView menggunakan adaptor untuk mengetahui cara menampilkan data di layar.

|  |  |
| --- | --- |
|  | PRAKTIK |

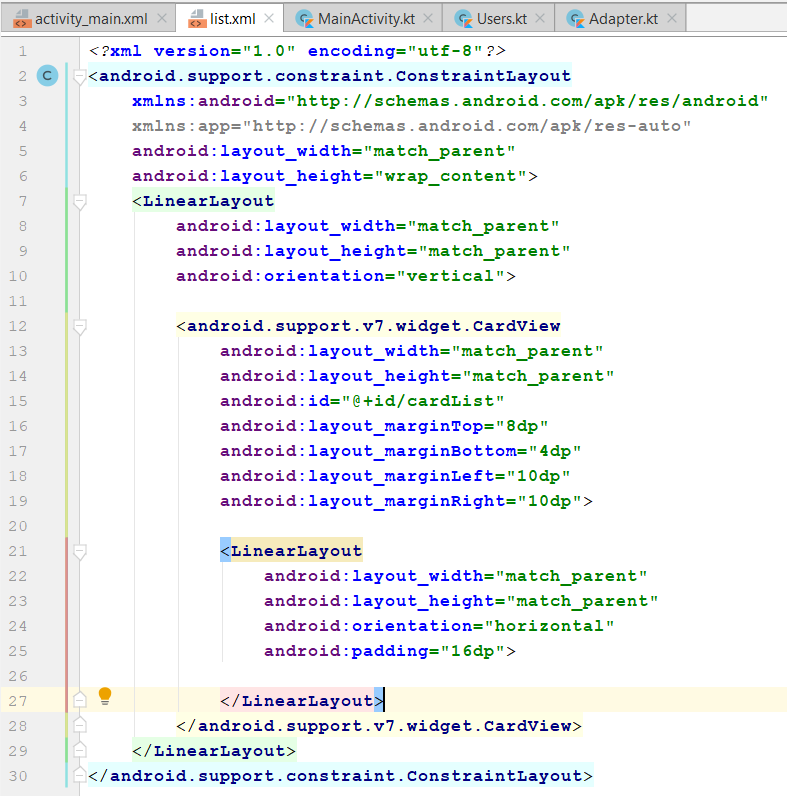
1. Buat Project baru. (dalam contoh ini diberi nama BelajarRecycleView)
2. Pada layout\_main.xml hapus komponen yang sudah ada (biasanya TextView).
3. Dari tab Palette, pilih Container, kemudian pilih RecyclerView
4. Klik dan drag ke Componen Tree, masukkan dibawah ConstrainLayout. Hasilnya seperti dibawah.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



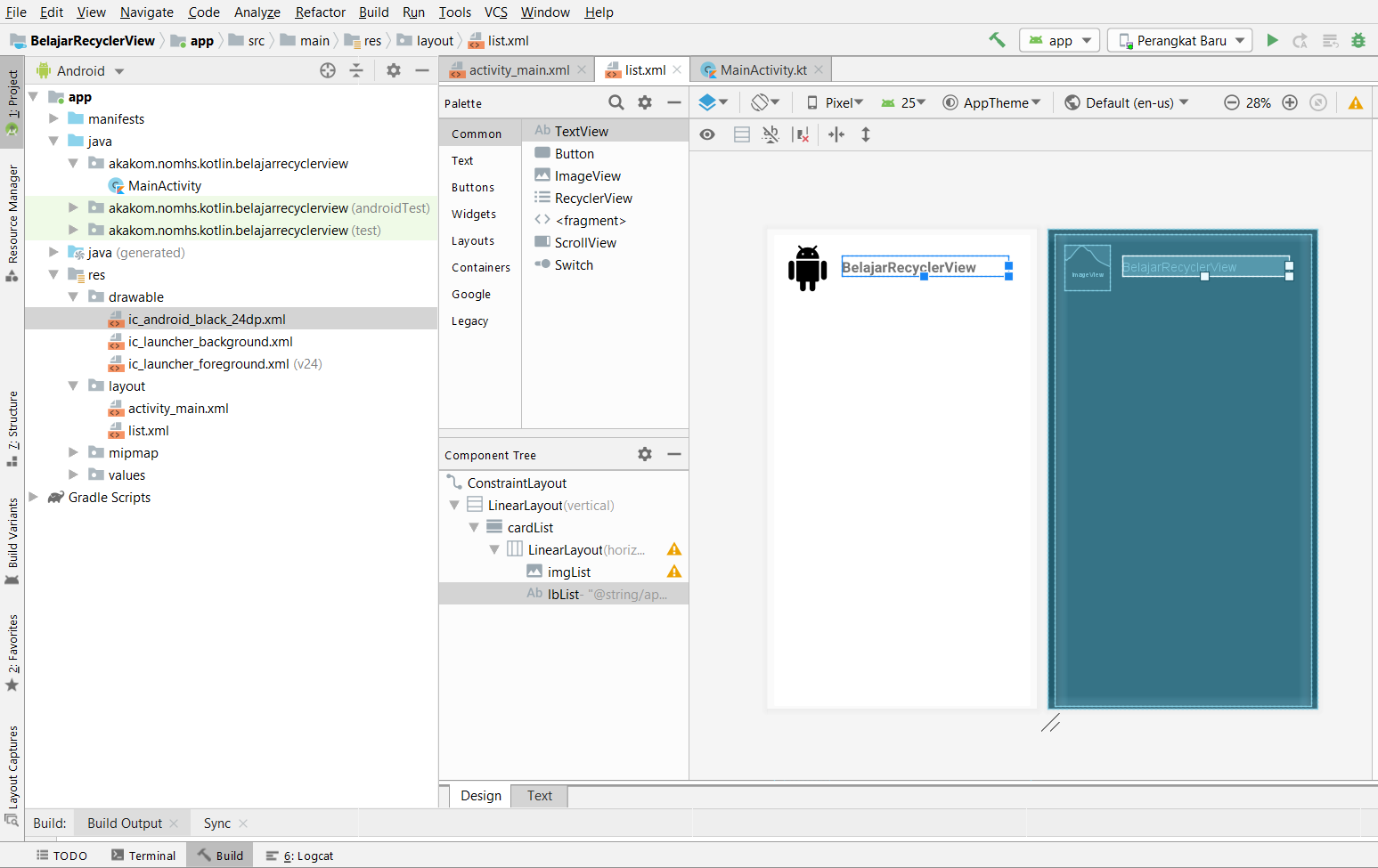
1. Buat sebuah file dibawah layout, beri nama list.xml
2. Tambahkan LinearLayout (vertical) didalam Container ConstrainLayout. Kemudian tambahkan cardaView dari Palette Container. Kemudian tambahkan LinearLayout(horisontal). Sampai sejauh ini, hasilnya dapat dilihat seperti gambar dibawah.

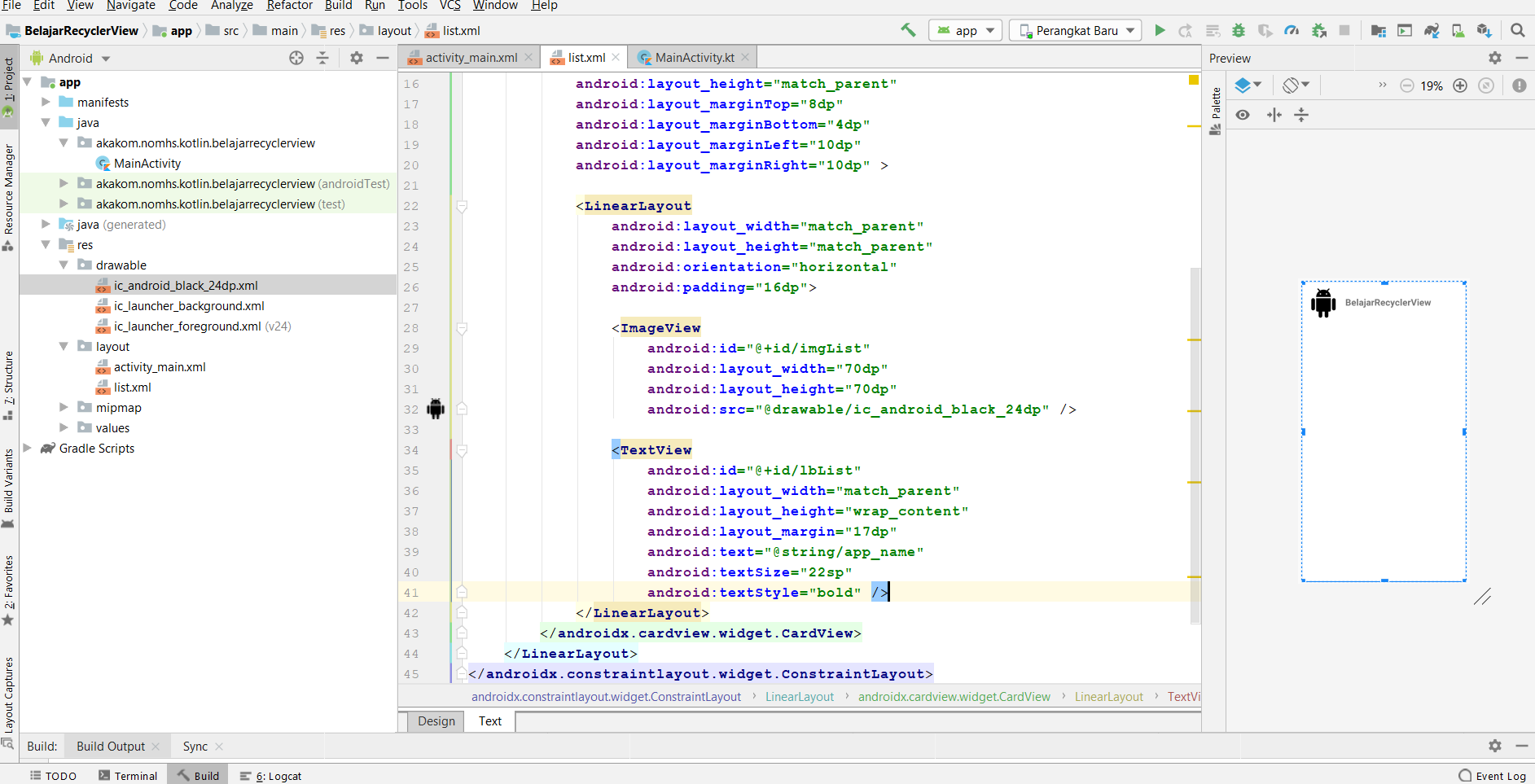


1. Kemudian, buka tab Text, ubah sehingga menjadi seperti dibawah ini.

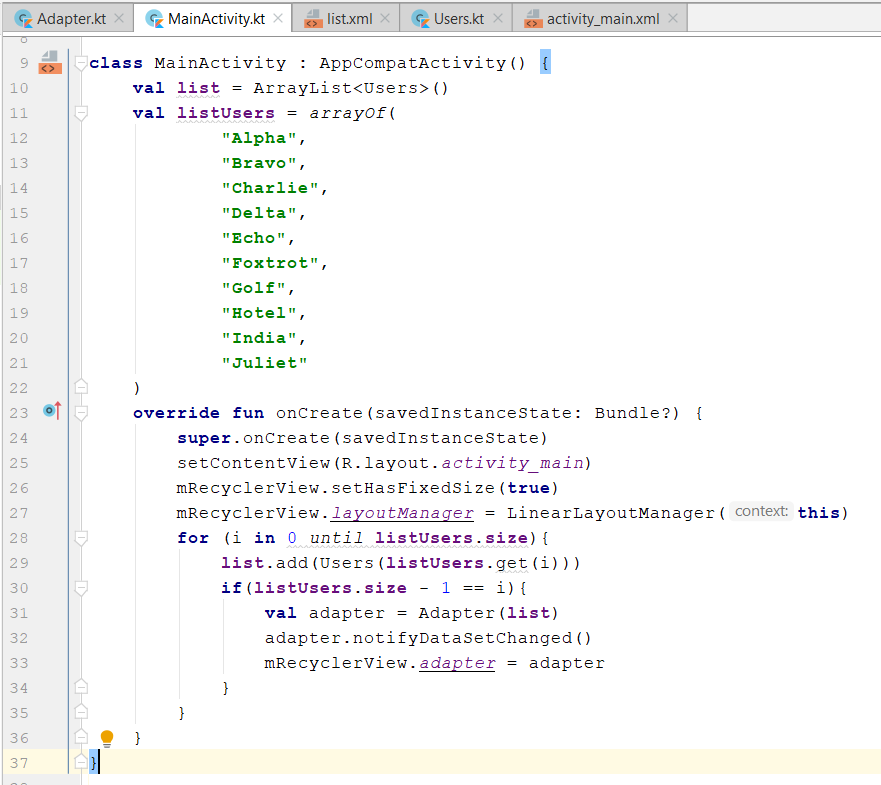


1. Kemudian tambahkan Image Button dan TextView dari tab Design dan kemudian modifikasi beberapa atribut dari tab Text. Perhatikan tampilan kedua tab dibawah.



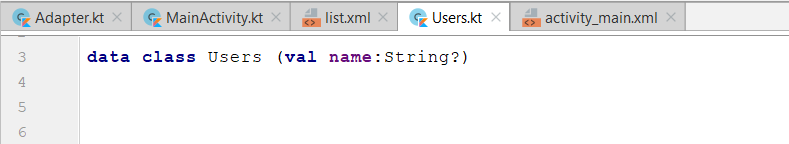


1. Buka file MainActivity.kt, modifikasilah menjadi seperti berikut ini. List digunakan untuk membuat daftar String yang akan ditampilkan pada RecycleView. Kemudian daftar tersebut dimasukkan ke dalam Adapter dan kemudian Adapter dipasang pada RecycleView.



1. 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 RecycleView. Dan class Users.kt digunakan untuk menampung daftar.





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



|  |  |
| --- | --- |
|  | **LATIHAN** |

Modifikasilah aplikasi dengan menambahkan Toast jika salah satu list dipilih.

|  |  |
| --- | --- |
| **tugas.png** | **TUGAS** |

Buat aplikasi baru dengan menerapkan RecycleView

|  |  |
| --- | --- |
|  | **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 Kamera

|  |  |
| --- | --- |
|  | CAPAIAN PEMBELAJARAN |
| 1. Mahasiswa dapat menggunakan kamera untuk mengambil photo 2. Mahasiswa dapat menggunakan cameraX | |

|  |  |
| --- | --- |
|  | 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 |

CameraX

CameraX adalah library dukungan Jetpack, yang dibuat untuk memudahkan Anda mengembangkan aplikasi kamera. CameraX memiliki antarmuka API yang konsisten dan mudah digunakan yang dapat digunakan di sebagian besar perangkat Android, dengan kompatibilitas hingga Android 5.0 (API level 21).

Meskipun memanfaatkan kapabilitas camera2, CameraX menggunakan pendekatan berbasis kasus penggunaan yang lebih sederhana, yang memperhatikan siklus proses. CameraX juga menyelesaikan masalah kompatibilitas perangkat secara otomatis sehingga kita tidak perlu menyertakan kode khusus perangkat dalam codebase. Fitur-fitur ini mengurangi jumlah kode yang perlu kita tulis saat menambahkan kapabilitas kamera ke aplikasi.

Terakhir, CameraX memungkinkan developer memanfaatkan pengalaman dan fitur kamera yang sama dengan yang disediakan oleh aplikasi kamera bawaan, hanya dengan menambahkan dua baris kode. Ekstensi CameraX adalah add-on opsional yang memungkinkan kita menambahkan efek seperti Potret, HDR, Mode Malam, dan Mode Beauty ke aplikasi yang kita buat pada perangkat yang didukung.

Silakan lanjut membaca tentang camerax pada link ini

<https://developer.android.com/training/camerax>

|  |  |
| --- | --- |
|  | PRAKTIK |

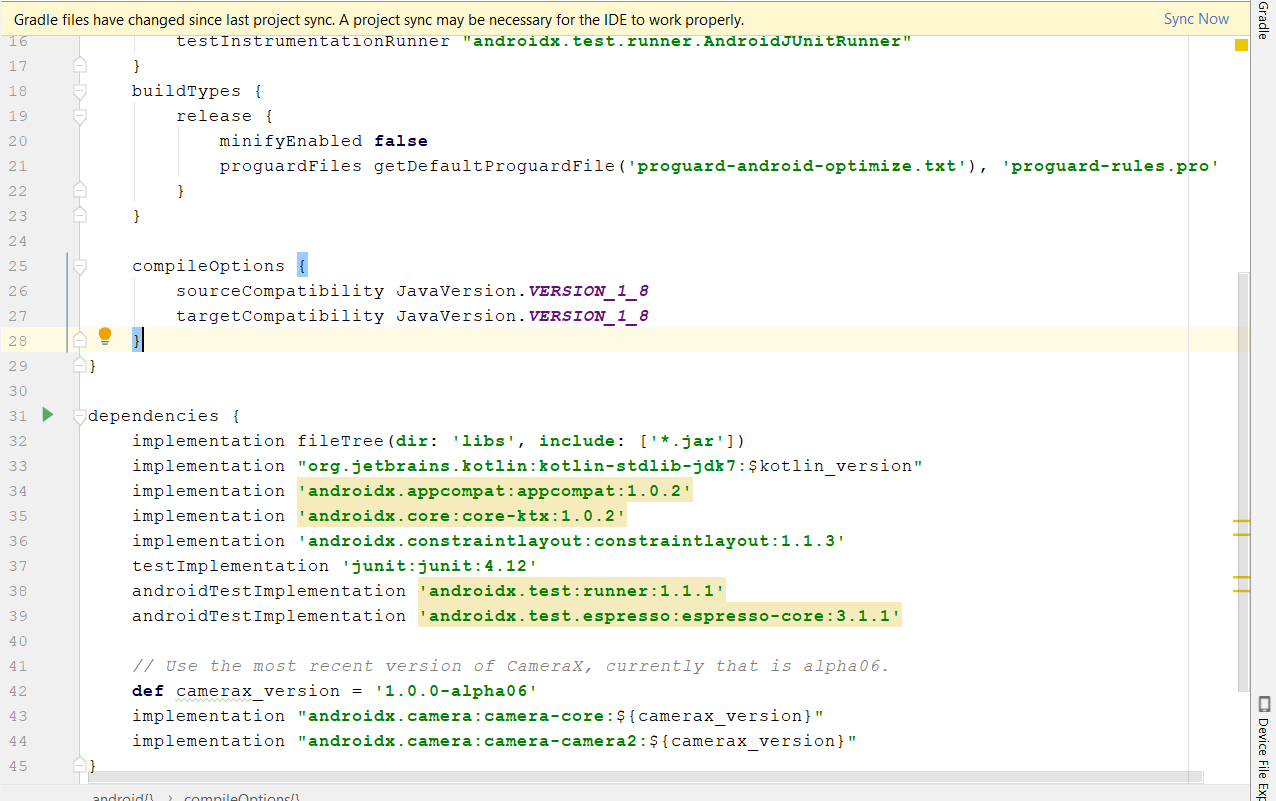
1. Buat project baru
2. Buka build.gradle(Modul:app). Tambahkan seperti berikut ini.

(Tambahkan di dalam blok dependencies)

*// Use the most recent version of CameraX, currently that is alpha06.***def** camerax\_version = **'1.0.0-alpha06'**implementation **"androidx.camera:camera-core:**${camerax\_version}**"**implementation **"androidx.camera:camera-camera2:**${camerax\_version}**"**

(Tambahkan di dalam blok android, sesudah blok buildTypes)

compileOptions {  
 sourceCompatibility JavaVersion.***VERSION\_1\_8*** targetCompatibility JavaVersion.***VERSION\_1\_8***}



1. Pilih Sync Now untuk sinkronisasi. Sekarang kita siap untuk menggunakan CameraX.
2. Membuat layout viewfinder. Kita akan menggunakan sebuah SurfaceTexture untuk menampilkan camera viewfinder. Kita akan menampilkan viewfinder dalam format persegi dengan ukuran tetap. Tambahkan pada activity\_main.xml sehingga menjadi 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"**>  
  
 <**TextureView  
 android:id="@+id/view\_finder"  
 android:layout\_width="640px"  
 android:layout\_height="640px"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"** />  
  
</**androidx.constraintlayout.widget.ConstraintLayout**>

1. Selanjutnya kita akan menambahkan permission untuk kamera. Buka AndroidManifest.xml dan tambahkan sebagai berikut.

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

1. Kemudian, buka MainActivity.kt, tambahkan import dan konstanta, di bagian atas setelah deklarasi package, sebelum deklarasi Kelas.

*// Your IDE likely can auto-import these classes, but there are several  
// different implementations so we list them here to disambiguate.***import** android.Manifest  
**import** android.content.pm.PackageManager  
**import** android.util.Size  
**import** android.graphics.Matrix  
**import** android.view.TextureView  
**import** android.widget.Toast  
**import** androidx.core.app.ActivityCompat  
**import** androidx.core.content.ContextCompat  
**import** java.util.concurrent.TimeUnit  
  
*// This is an arbitrary number we are using to keep track of the permission  
// request. Where an app has multiple context for requesting permission,  
// this can help differentiate the different contexts.***private const val** *REQUEST\_CODE\_PERMISSIONS* = 10  
  
*// This is an array of all the permission specified in the manifest.***private val** *REQUIRED\_PERMISSIONS* = *arrayOf*(Manifest.permission.*CAMERA*)

1. Kemudian pada blok kelas MainActivity, tambahkan sebagai berikut. Pertama, tambahkan pada deklarasi kelas, sebagai berikut.

**class** MainActivity : AppCompatActivity(), LifecycleOwner {  
  
 **override fun** onCreate(savedInstanceState: Bundle?) {  
 **super**.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
 }

*//tambahkan koding berikutnya disini.*

}

1. Di dalam blok MainActivity dan dibawah onCreate, tambahkan sebagai berikut.

**private val executor** = Executors.newSingleThreadExecutor()  
**private lateinit var viewFinder**: TextureView  
  
**private fun** startCamera() {  
 *//* ***TODO: Implement CameraX operations***}  
  
**private fun** updateTransform() {  
 *//* ***TODO: Implement camera viewfinder transformations***}  
  
*/\*\*  
 \* Process result from permission request dialog box, has the request  
 \* been granted? If yes, start Camera. Otherwise display a toast  
 \*/***override fun** onRequestPermissionsResult(  
 requestCode: Int, permissions: Array<String>, grantResults: IntArray) {  
 **if** (requestCode == *REQUEST\_CODE\_PERMISSIONS*) {  
 **if** (allPermissionsGranted()) {  
 **viewFinder**.post **{** startCamera() **}** } **else** {  
 Toast.makeText(**this**,  
 **"Permissions not granted by the user."**,  
 Toast.*LENGTH\_SHORT*).show()  
 finish()  
 }  
 }  
}  
  
*/\*\*  
 \* Check if all permission specified in the manifest have been granted  
 \*/***private fun** allPermissionsGranted() = *REQUIRED\_PERMISSIONS*.*all* **{** ContextCompat.checkSelfPermission(  
 *baseContext*, **it**) == PackageManager.*PERMISSION\_GRANTED***}**

1. Kemudian, tambahkan di dalam blok onCreate koding sebagai berikut.

*// Add this at the end of onCreate function***viewFinder** = findViewById(R.id.*view\_finder*)  
  
*// Request camera permissions***if** (allPermissionsGranted()) {  
 **viewFinder**.post **{** startCamera() **}**} **else** {  
 ActivityCompat.requestPermissions(  
 **this**, *REQUIRED\_PERMISSIONS*, *REQUEST\_CODE\_PERMISSIONS*)  
}  
  
*// Every time the provided texture view changes, recompute layout***viewFinder**.addOnLayoutChangeListener **{** \_, \_, \_, \_, \_, \_, \_, \_, \_ **->** updateTransform()  
**}**

1. Isikan koding berikut pada method startCamera

**private fun** startCamera() {  
 *//* ***TODO: Implement CameraX operations*** *// Create configuration object for the viewfinder use case* **val** previewConfig = PreviewConfig.Builder().*apply* **{** setTargetResolution(Size(640, 480))  
 **}**.build()  
   
 *// Build the viewfinder use case* **val** preview = Preview(previewConfig)  
  
 *// Every time the viewfinder is updated, recompute layout* preview.setOnPreviewOutputUpdateListener **{** *// To update the SurfaceTexture, we have to remove it and re-add it* **val** parent = **viewFinder**.*parent* **as** ViewGroup  
 parent.removeView(**viewFinder**)  
 parent.addView(**viewFinder**, 0)  
  
 **viewFinder**.*surfaceTexture* = **it**.*surfaceTexture* updateTransform()  
 **}** *// Bind use cases to lifecycle  
 // If Android Studio complains about "this" being not a LifecycleOwner  
 // try rebuilding the project or updating the appcompat dependency to  
 // version 1.1.0 or higher.* CameraX.bindToLifecycle(**this**, preview)  
}

1. Juga dalam method updateTransform

**private fun** updateTransform() {  
 **val** matrix = Matrix()  
  
 *// Compute the center of the view finder* **val** centerX = **viewFinder**.*width* / 2f  
 **val** centerY = **viewFinder**.*height* / 2f  
  
 *// Correct preview output to account for display rotation* **val** rotationDegrees = **when**(**viewFinder**.*display*.*rotation*) {  
 Surface.*ROTATION\_0* -> 0  
 Surface.*ROTATION\_90* -> 90  
 Surface.*ROTATION\_180* -> 180  
 Surface.*ROTATION\_270* -> 270  
 **else** -> **return** }  
 matrix.postRotate(-rotationDegrees.toFloat(), centerX, centerY)  
  
 *// Finally, apply transformations to our TextureView* **viewFinder**.setTransform(matrix)  
}

1. Jalankan dan amati hasilnya.
2. Sekarang kita akan menambahkan *image capture use case*. Tambahkan komponen pada activity\_main.xml sebagai berikut

<**ImageButton  
 android:id="@+id/capture\_button"  
 android:layout\_width="72dp"  
 android:layout\_height="72dp"  
 android:layout\_margin="24dp"  
 app:srcCompat="@android:drawable/ic\_menu\_camera"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"** />

1. Kita akan mendefinisikan objek konfigurasi yang digunakan untuk instantiate objek use case yang sebenarnya. Untuk mengambil foto, ketika tombol ambil ditekan, kita perlu memperbarui method `startCamera ()` dan menambahkan beberapa baris kode di akhir. Tambahkan koding berikut ini, sebelum panggilan ke CameraX.bindToLifecycle.

*// Create configuration object for the image capture use case***val** imageCaptureConfig = ImageCaptureConfig.Builder()  
 .*apply* **{** *// We don't set a resolution for image capture; instead, we  
 // select a capture mode which will infer the appropriate  
 // resolution based on aspect ration and requested mode* setCaptureMode(ImageCapture.CaptureMode.**MIN\_LATENCY**)  
 **}**.build()  
  
*// Build the image capture use case and attach button click listener***val** imageCapture = ImageCapture(imageCaptureConfig)  
findViewById<ImageButton>(R.id.*capture\_button*).setOnClickListener **{  
 val** file = File(*externalMediaDirs*.*first*(),  
 **"${**System.currentTimeMillis()**}.jpg"**)  
  
 imageCapture.takePicture(file, **executor**,  
 **object** : ImageCapture.OnImageSavedListener {  
 **override fun** onError(  
 imageCaptureError: ImageCapture.ImageCaptureError,  
 message: String,  
 exc: Throwable?  
 ) {  
 **val** msg = **"Photo capture failed: $**message**"** Log.e(**"CameraXApp"**, msg, exc)  
 **viewFinder**.post **{** Toast.makeText(*baseContext*, msg, Toast.*LENGTH\_SHORT*).show()  
 **}** }  
  
 **override fun** onImageSaved(file: File) {  
 **val** msg = **"Photo capture succeeded: ${**file.*absolutePath***}"** Log.d(**"CameraXApp"**, msg)  
 **viewFinder**.post **{** Toast.makeText(*baseContext*, msg, Toast.*LENGTH\_SHORT*).show()  
 **}** }  
 })  
**}**

1. Update pemanggilan ke CameraX.bindLifeCycle dengan menambahkan parameter use case yang baru saja dibuat.

CameraX.bindToLifecycle(**this**, preview,imageCapture)

1. Jalankan dan amati hasilnya.

|  |  |
| --- | --- |
|  | **LATIHAN** |

1. Kembangkan aplikasi dengan menambahkan image analisis use case.

|  |  |
| --- | --- |
| **tugas.png** | **TUGAS** |

1. Kembangkan aplikasi sehingga layar untuk kamera menjadi satu layar

|  |  |
| --- | --- |
|  | **REFERENSI** |

1. <https://codelabs.developers.google.com/codelabs/camerax-getting-started/index.html?index=..%2F..index#0>
2. <https://developer.android.com/training/camerax>

# MODUL 7 Akses GPS, Kompas, Proximity

|  |  |
| --- | --- |
|  | CAPAIAN PEMBELAJARAN |
| 1. Mahasiswa dapat menggunakan GPS (Current Location) 2. Mahasiswa dapat menggunakan kompas (Kompas) 3. Mahasiswa dapat menggunakan proximity (ProximitySensor) | |

|  |  |
| --- | --- |
|  | 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. Lokasi

Salah satu fitur unik dari aplikasi seluler adalah kesadaran lokasi. Pengguna seluler membawa perangkat mereka ke mana saja, dan menambahkan kesadaran lokasi ke aplikasi kita menawarkan pengalaman yang lebih kontekstual kepada pengguna. API lokasi yang tersedia di layanan Google Play memfasilitasi penambahan kesadaran lokasi ke aplikasi dengan pelacakan lokasi otomatis, geofencing, dan pengenalan aktivitas.

1. Optimalkan **lokasi untuk baterai**

Batas Lokasi Latar Belakang yang diperkenalkan di Android 8.0 (API level 26) telah membawa fokus baru ke subjek tentang cara penggunaan layanan lokasi mempengaruhi pengurasan baterai. Halaman ini membahas beberapa praktik terbaik layanan lokasi dan yang dapat kita lakukan sekarang untuk membuat aplikasi kita lebih hemat baterai. Menerapkan praktik terbaik ini menguntungkan aplikasi kita, terlepas dari versi platform yang digunakan.

Batas lokasi latar belakang di Android 8.0 memperkenalkan perubahan berikut:

* Pengumpulan lokasi latar belakang diperlambat dan lokasi dihitung, dan hanya dikirim beberapa kali dalam satu jam.
* Pemindaian Wi-Fi lebih konservatif, dan update lokasi tidak dihitung saat perangkat tetap terhubung dengan titik akses statis yang sama.
* Daya respons pembatasan wilayah berubah dari puluhan detik hingga sekitar dua menit. Perubahan ini secara kentara meningkatkan kinerja baterai—hingga 10 kali lebih baik pada beberapa perangkat.

Halaman ini mengasumsikan bahwa kita menggunakan Google Location Services API, yang menawarkan akurasi yang lebih tinggi dan mengenakan beban baterai yang lebih ringan dari API lokasi framework. Secara khusus, halaman ini mengasumsikan pemahaman tentang API penyedia lokasi gabungan, yang menggabungkan sinyal dari GPS, Wi-Fi, dan jaringan seluler, serta akselerometer, giroskop, magnetometer, dan sensor lainnya. Kita juga akan familier dengan API pembatasan wilayah, yang dibuat di atas API penyedia lokasi gabungan, dan dioptimalkan untuk kinerja baterai.

1. Mendapatkan **lokasi terakhir yang diketahui**

Dengan location API layanan Google Play, aplikasi Kita dapat meminta lokasi terakhir yang diketahui dari perangkat pengguna. Dalam sebagian besar kasus, Kita ingin mengetahui lokasi pengguna saat ini, yang biasanya setara dengan lokasi perangkat yang terakhir diketahui.

Secara khusus, gunakan penyedia lokasi fusi untuk mengambil lokasi terakhir perangkat yang diketahui. Penyedia lokasi fusi adalah salah satu location API di layanan Google Play. API tersebut mengelola teknologi lokasi yang mendasarinya dan memberikan API sederhana sehingga Kita dapat menentukan persyaratan tingkat tinggi, seperti akurasi tinggi atau penggunaan daya rendah. API tersebut juga mengoptimalkan penggunaan daya baterai perangkat.

1. Mengubah **setelan lokasi**

Jika aplikasi Kita perlu meminta lokasi atau menerima update izin, perangkat harus mengaktifkan setelan sistem yang sesuai, seperti pemindaian GPS atau Wi-Fi. Aplikasi Kita menentukan tingkat akurasi/konsumsi daya yang diperlukan serta interval update yang diinginkan, dan perangkat akan otomatis membuat perubahan yang sesuai pada setelan sistem, bukan mengaktifkan layanan secara langsung seperti GPS perangkat. Setelan ini ditentukan oleh objek data LocationRequest.

1. Menerima **update lokasi berkala**

Jika aplikasi Kita dapat terus melacak lokasi, hal itu dapat memberikan informasi yang lebih relevan kepada pengguna. Misalnya, jika aplikasi Kita membantu pengguna menemukan jalannya saat berjalan atau mengemudi, atau jika aplikasi Kita melacak lokasi aset, aplikasi perlu mendapatkan lokasi perangkat secara berkala. Selain lokasi geografis (lintang dan bujur), Kita mungkin ingin memberikan informasi lebih lanjut kepada pengguna seperti arah (arah pergerakan horizontal), ketinggian, atau kecepatan perangkat. Informasi ini dan lainnya tersedia di objek Location yang dapat diambil aplikasi Kita dari penyedia lokasi fusi.

Selagi Kita bisa mendapatkan lokasi perangkat dengan getLastLocation(), seperti yang diilustrasikan dalam tutorial mengenai Mendapatkan Lokasi Terakhir yang Diketahui, pendekatan langsung lainnya adalah meminta update berkala dari penyedia lokasi fusi. Sebagai respons atas hal tersebut, API mengupdate aplikasi Kita secara berkala dengan lokasi terbaik yang tersedia, berdasarkan penyedia lokasi yang saat ini tersedia seperti Wi-Fi dan GPS (Global Positioning System). Keakuratan lokasi ditentukan oleh penyedia, izin lokasi yang Kita minta, dan opsi yang Kita tetapkan dalam permintaan lokasi.

1. Menampilkan **alamat lokasi**

Mendapatkan lokasi terakhir yang diketahui dan Menerima update lokasi menjelaskan cara mendapatkan lokasi pengguna dalam bentuk objek Location yang berisi koordinat lintang dan bujur. Meskipun lintang dan bujur berguna untuk menghitung jarak atau menampilkan posisi peta, alamat lokasi sering kali lebih berguna. Misalnya, jika Kita ingin memberi tahu pengguna tempat mereka berada atau apa yang dekat, alamat jalan lebih bermakna daripada koordinat geografis (lintang/bujur) lokasi.

Dengan class Geocoder di API lokasi framework Android, Kita bisa mengonversi alamat ke koordinat geografis yang sesuai. Proses ini disebut geocoding. Selain itu, Kita bisa mengonversi lokasi geografis ke suatu alamat. Fitur pencarian alamat juga disebut sebagai geocoding balik.

**Sensor posisi**

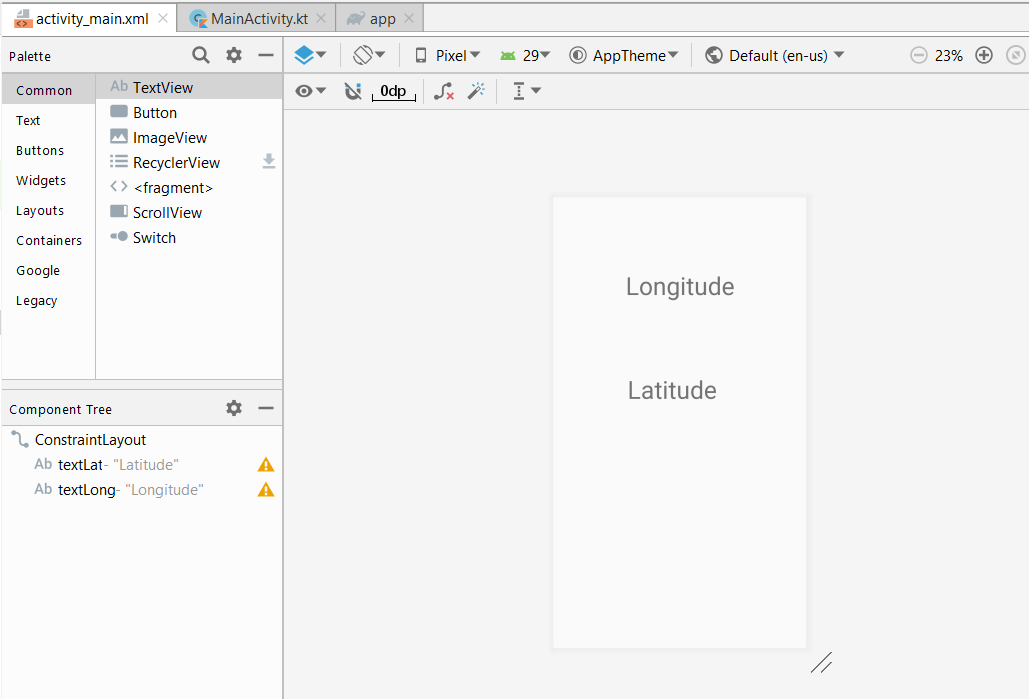
Platform Android menyediakan dua sensor yang memungkinkan Anda menentukan posisi perangkat: sensor medan geomagnetik dan akselerometer. Platform Android juga menyediakan sensor yang memungkinkan Anda menentukan seberapa dekat permukaan perangkat dengan suatu objek (dikenal sebagai *sensor kedekatan*). Sensor medan geomagnetik dan sensor kedekatan berbasis hardware. Sebagian besar produsen handset dan tablet menyertakan sensor medan geomagnetik. Demikian juga, produsen handset biasanya menyertakan sensor kedekatan untuk menentukan kapan handset dipegang dekat dengan wajah pengguna (misalnya, selama panggilan telepon). Untuk menentukan orientasi perangkat, Anda dapat menggunakan bacaan dari akselerometer perangkat dan sensor medan geomagnetik.

Sensor posisi berguna untuk menentukan posisi fisik perangkat dalam bingkai referensi dunia. Misalnya, Anda dapat menggunakan sensor medan geomagnetik yang dikombinasikan dengan akselerometer untuk menentukan posisi perangkat yang relatif dengan kutub utara magnetik. Anda juga dapat menggunakan sensor ini untuk menentukan orientasi perangkat di dalam bingkai referensi aplikasi Anda. Sensor posisi biasanya tidak digunakan untuk memantau pergerakan atau gerakan perangkat, seperti guncangan, gerakan miring, atau dorongan (untuk informasi selengkapnya, lihat [Sensor gerak](https://developer.android.com/guide/topics/sensors/sensors_motion)).

Sensor medan geomagnetik dan akselerometer menampilkan array multi-dimensi dari nilai sensor untuk masing-masing [SensorEvent](https://developer.android.com/reference/android/hardware/SensorEvent). Misalnya, sensor medan geomagnetik memberikan nilai kekuatan medan geomagnetik untuk masing-masing dari ketiga sumbu koordinat selama kejadian sensor tunggal. Demikian juga, sensor akselerometer mengukur akselerasi yang diterapkan ke perangkat selama kejadian sensor. Untuk informasi selengkapnya tentang sistem koordinat yang digunakan oleh sensor, lihat [Sistem koordinat sensor](https://developer.android.com/guide/topics/sensors/sensors_overview#sensors-coords). Sensor kedekatan memberikan nilai tunggal untuk setiap kejadian sensor.

|  |  |
| --- | --- |
|  | PRAKTIK |

1. Project pertama akan dibuat untuk mendeteksi lokasi kita saat ini.
2. Buat project baru (Dalam contoh ini : LokasiSekarang)
3. Buka layout activity\_main.xml tambahkan dua buah TextView.



1. Tambahkan dependencies berikut pada build.gradle

implementation **'com.google.android.gms:play-services-location:17.0.0'**

1. Tambahkan permission pada AndroidManifest sebagai berikut.

<**uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"**/>  
<**uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"**/>

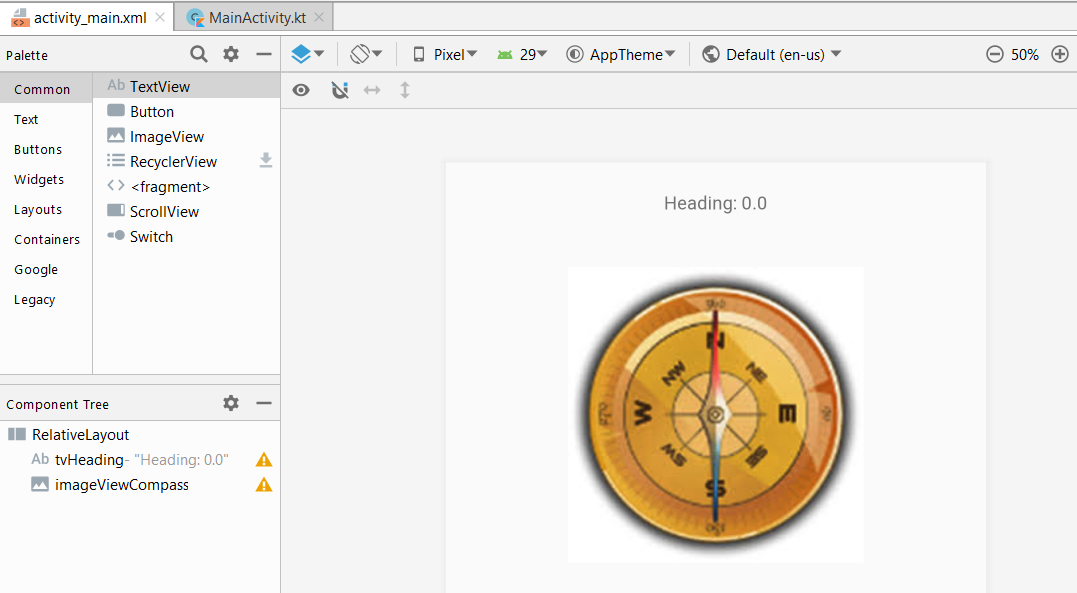
1. Buka MainActivity.kt dan tambahkan sehingga menjadi sebagai berikut.

**import** android.Manifest  
**import** android.content.Context  
**import** android.content.Intent  
**import** android.content.pm.PackageManager  
**import** android.location.LocationManager  
**import** androidx.appcompat.app.AppCompatActivity  
**import** android.os.Bundle  
**import** android.os.Looper  
**import** android.provider.Settings  
**import** android.widget.TextView  
**import** android.widget.Toast  
**import** androidx.core.app.ActivityCompat  
**import** com.google.android.gms.location.\*  
  
**class** MainActivity : AppCompatActivity() {  
  
 **internal var PERMISSION\_ID** = 44  
 **internal lateinit var mFusedLocationClient**: FusedLocationProviderClient  
 **internal lateinit var latTextView**: TextView  
 **internal lateinit var lonTextView**: TextView  
  
 **private val mLocationCallback** = **object** : LocationCallback() {  
 **override fun** onLocationResult(locationResult: LocationResult) {  
 **val** mLastLocation = locationResult.*lastLocation* **latTextView**.*text* = mLastLocation.*latitude*.toString() + **""  
 lonTextView**.*text* = mLastLocation.*longitude*.toString() + **""** }  
 }  
  
 **private val isLocationEnabled**: Boolean  
 **get**() {  
 **val** locationManager = getSystemService(Context.*LOCATION\_SERVICE*) **as** LocationManager  
 **return** locationManager.isProviderEnabled(LocationManager.*GPS\_PROVIDER*) || locationManager.isProviderEnabled(  
 LocationManager.*NETWORK\_PROVIDER* )  
 }  
  
 **override fun** onCreate(savedInstanceState: Bundle?) {  
 **super**.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
  
 **latTextView** = findViewById(R.id.*textLat*)  
 **lonTextView** = findViewById(R.id.*textLong*)  
 **mFusedLocationClient** = LocationServices.getFusedLocationProviderClient(**this**)  
  
 getLastLocation()  
 }  
  
 **private fun** getLastLocation() {  
 **if** (checkPermissions()) {  
 **if** (**isLocationEnabled**) {  
 **mFusedLocationClient**.*lastLocation*.addOnCompleteListener **{** task **->  
 val** location = task.*result* **if** (location == **null**) {  
 requestNewLocationData()  
 } **else** {  
 **latTextView**.*text* = **"Latitude = "**+location.*latitude*.toString() + **""  
 lonTextView**.*text* = **"Longitude = "**+location.*longitude*.toString() + **""** }  
 **}** } **else** {  
 Toast.makeText(**this**, **"Turn on location"**, Toast.*LENGTH\_LONG*).show()  
 **val** intent = Intent(Settings.*ACTION\_LOCATION\_SOURCE\_SETTINGS*)  
 startActivity(intent)  
 }  
 } **else** {  
 requestPermissions()  
 }  
 }  
  
  
 **private fun** requestNewLocationData() {  
  
 **val** mLocationRequest = LocationRequest()  
 mLocationRequest.*priority* = LocationRequest.*PRIORITY\_HIGH\_ACCURACY* mLocationRequest.*interval* = 0  
 mLocationRequest.*fastestInterval* = 0  
 mLocationRequest.*numUpdates* = 1  
  
 **mFusedLocationClient** = LocationServices.getFusedLocationProviderClient(**this**)  
 **mFusedLocationClient**.requestLocationUpdates(  
 mLocationRequest, **mLocationCallback**,  
 Looper.myLooper()  
 )  
 }  
  
 **private fun** checkPermissions(): Boolean {  
 **return if** (ActivityCompat.checkSelfPermission(**this**, Manifest.permission.*ACCESS\_COARSE\_LOCATION*) == PackageManager.*PERMISSION\_GRANTED* && ActivityCompat.checkSelfPermission(**this**, Manifest.permission.*ACCESS\_FINE\_LOCATION*) == PackageManager.*PERMISSION\_GRANTED*) {  
 **true** } **else false** }  
  
 **private fun** requestPermissions() {  
 ActivityCompat.requestPermissions(  
 **this**,  
 *arrayOf*(Manifest.permission.*ACCESS\_COARSE\_LOCATION*, Manifest.permission.*ACCESS\_FINE\_LOCATION*),  
 **PERMISSION\_ID** )  
 }  
  
 **override fun** onRequestPermissionsResult(requestCode: Int, permissions: Array<String>, grantResults: IntArray) {  
 **super**.onRequestPermissionsResult(requestCode, permissions, grantResults)  
 **if** (requestCode == **PERMISSION\_ID**) {  
 **if** (grantResults.**size** > 0 && grantResults[0] == PackageManager.*PERMISSION\_GRANTED*) {  
 getLastLocation()  
 }  
 }  
 }  
  
 **public override fun** onResume() {  
 **super**.onResume()  
 **if** (checkPermissions()) {  
 getLastLocation()  
 }  
  
 }  
}

1. Tambahkan dua buah TextView di activity\_maim.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity"**>  
  
 <**TextView  
 android:id="@+id/textLat"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Latitude"  
 android:textSize="30dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.452"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.419"** />  
  
 <**TextView  
 android:id="@+id/textLong"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Longitude"  
 android:textSize="30dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/textLat"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"** />  
  
</**androidx.constraintlayout.widget.ConstraintLayout**>

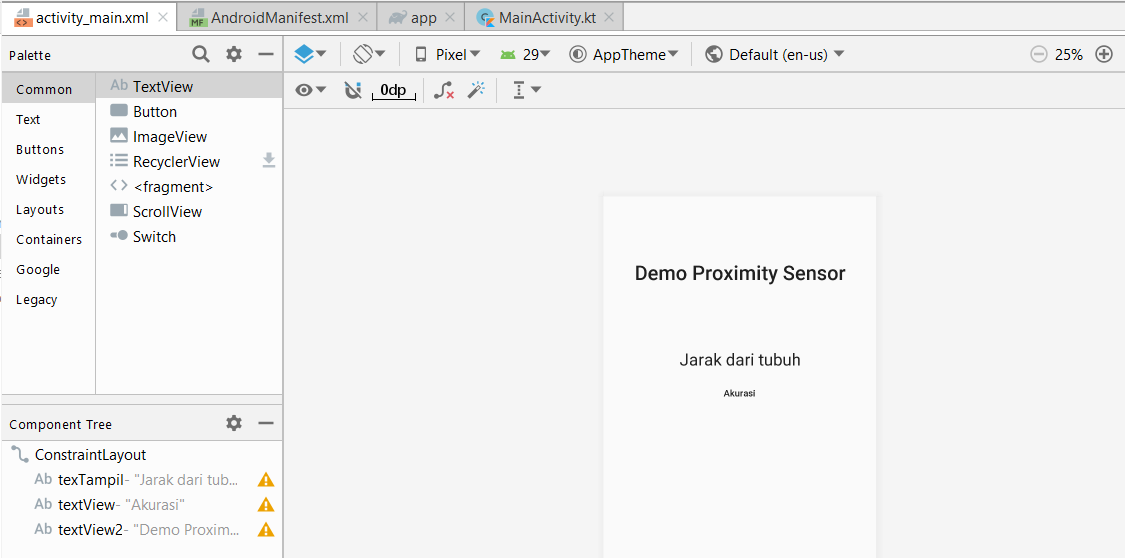
1. Jalankan dan amati hasilnya.
2. Kita akan membuat project kedua untuk menampilkan kompas.
3. Buat project baru (dalam contoh ini : Kompas)
4. Tambahkan TextView dan ImageView pada activity\_main.xml anda. Gambar kompas dapat anda cari di halaman web.



1. Juga pada MainActivity.kt tambahkan koding sehingga menjadi sebagai berikut.

**import** android.content.Context  
**import** android.hardware.Sensor  
**import** android.hardware.SensorEvent  
**import** android.hardware.SensorEventListener  
**import** android.hardware.SensorManager  
**import** androidx.appcompat.app.AppCompatActivity  
**import** android.os.Bundle  
**import** android.view.animation.Animation  
**import** android.view.animation.RotateAnimation  
**import** android.widget.ImageView  
**import** android.widget.TextView  
  
**class** MainActivity : AppCompatActivity(), SensorEventListener {  
 **private var image**: ImageView? = **null  
 private var currentDegree** = 90f  
 **private var mSensorManager**: SensorManager? = **null  
 internal lateinit var tvHeading**: TextView  
  
 **override fun** onCreate(savedInstanceState: Bundle?) {  
 **super**.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
  
  
 **image** = findViewById(R.id.*imageViewCompass*) **as** ImageView  
 **tvHeading** = findViewById(R.id.*tvHeading*) **as** TextView  
 **mSensorManager** = getSystemService(Context.*SENSOR\_SERVICE*) **as** SensorManager  
 }  
  
 **override fun** onResume() {  
 **super**.onResume()  
 **mSensorManager**!!.registerListener(**this**,  
 **mSensorManager**!!.getDefaultSensor(Sensor.*TYPE\_ORIENTATION*),  
 SensorManager.*SENSOR\_DELAY\_GAME*)  
 }  
  
 **override fun** onPause() {  
 **super**.onPause()  
 **mSensorManager**!!.unregisterListener(**this**)  
 }  
  
 **override fun** onSensorChanged(event: SensorEvent) {  
 **val** degree = Math.round(event.**values**[0]).toFloat()  
  
 **tvHeading**.*text* = **"Heading: "** + java.lang.Float.toString(degree) + **" degrees"  
 val** ra = RotateAnimation(  
 **currentDegree**, -degree,  
 Animation.*RELATIVE\_TO\_SELF*, 0.5f,  
 Animation.*RELATIVE\_TO\_SELF*,  
 0.5f)  
  
 ra.*duration* = 210  
 ra.*fillAfter* = **true  
  
 image**!!.startAnimation(ra)  
 **currentDegree** = -degree  
 }  
  
 **override fun** onAccuracyChanged(sensor: Sensor, accuracy: Int) {}  
  
}

1. Jalankan dan amati hasilnya.
2. Project ketiga menggunakan proximity sensor.
3. Buat project baru (dalam contoh ini : ProximitySensor)



1. Buka MainActivity.kt dan tambahkan koding sehingga menjadi sebagai berikut.

**import** android.content.Context  
**import** android.hardware.Sensor  
**import** android.hardware.SensorEvent  
**import** android.hardware.SensorEventListener  
**import** android.hardware.SensorManager  
**import** androidx.appcompat.app.AppCompatActivity  
**import** android.os.Bundle  
**import** kotlinx.android.synthetic.main.activity\_main.\*  
  
**class** MainActivity : AppCompatActivity(), SensorEventListener {  
  
 **private lateinit var sensorManager**: SensorManager  
 **private var mProximity**: Sensor? = **null  
  
 public override fun** onCreate(savedInstanceState: Bundle?) {  
 **super**.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
  
 *// Get an instance of the sensor service, and use that to get an instance of  
 // a particular sensor.* **sensorManager** = getSystemService(Context.*SENSOR\_SERVICE*) **as** SensorManager  
 **mProximity** = **sensorManager**.getDefaultSensor(Sensor.*TYPE\_PROXIMITY*)  
 }  
  
 **override fun** onAccuracyChanged(sensor: Sensor, accuracy: Int) {  
 *// Do something here if sensor accuracy changes.* **if** (accuracy==SensorManager.*SENSOR\_STATUS\_ACCURACY\_HIGH*) *//value =3* textView.setText(**"Akurasi tinggi"**)  
 **else if** (accuracy==SensorManager.*SENSOR\_STATUS\_ACCURACY\_MEDIUM*) *//value=2* textView.setText(**"Akurasi sedang"**)  
 **else** textView.setText(**"Akurasi rendah"**) *//value=1* }  
  
 **override fun** onSensorChanged(event: SensorEvent) {  
 **val** distance = event.**values**[0]  
 *// Do something with this sensor data.* **if** (distance<=5){  
 texTampil.setText(**"Jarak dekat"**)  
 } **else** texTampil.setText(**"Jarak jauh"**)  
 }  
  
 **override fun** onResume() {  
 *// Register a listener for the sensor.* **super**.onResume()  
  
 **mProximity**?.*also* **{** proximity **->  
 sensorManager**.registerListener(**this**, proximity, SensorManager.*SENSOR\_DELAY\_NORMAL*)  
 **}** }  
  
 **override fun** onPause() {  
 *// Be sure to unregister the sensor when the activity pauses.* **super**.onPause()  
 **sensorManager**.unregisterListener(**this**)  
 }  
 }

1. Jalankan dan amati hasilnya.

|  |  |
| --- | --- |
|  | **LATIHAN** |

1. Kembangkan aplikasi dengan menambahkan komponen sehingga menjadi aplikasi yang fungsional pada ketiga project tadi.

|  |  |
| --- | --- |
| **tugas.png** | **TUGAS** |

1. Kembangkan aplikasi dengan menambahkan komponen-komponen yang pernah dibuat. Jadikan aplikasi sederhana yang siap digunakan.

|  |  |
| --- | --- |
|  | **REFERENSI** |

<https://developer.android.com/training/location>

<https://developer.android.com/guide/topics/sensors/sensors_overview>

<https://developer.android.com/guide/topics/sensors/sensors_position>