MODUL

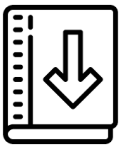
MODUL 1
Pengenalan Aplikasi Mobile

CAPAIAN PEMBELAJARAN

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

KEBUTUHAN ALAT/BAHAN/SOFTWARE

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



DASAR TEORI

1.1. Pendahuluan.

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

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

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

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

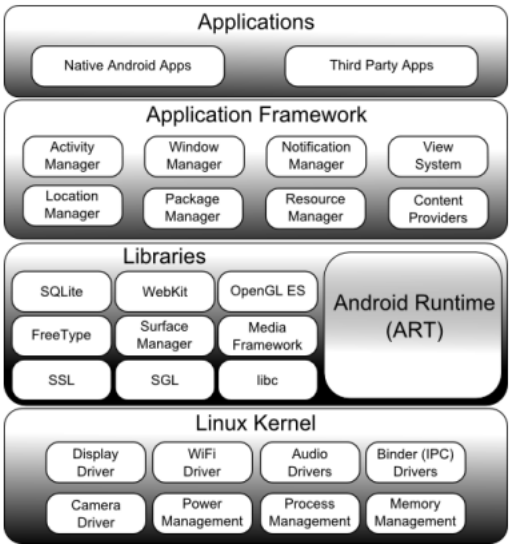
1. https://kotlinlang.org/docs/reference/ untuk belajar dasar pemrograman kotlin.
2. https://developer.android.com/kotlin untuk pemahaman kenapa android menggunakan kotlin.
3. https://developer.android.com/courses/kotlin-android-fundamentals/toc untuk codelab review untuk dasar pemrograman kotlin.
4. https://codelabs.developers.google.com/android-kotlin-fundamentals/ untuk belajar 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.

1.2. 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). November 2007, Android SDK dirilis pertama kali dengan "tampilan awal" (**early look**). September 2008, T-Mobile mengumumkan ketersediaan **HTC Dream G1**, smartphone pertama yang berbasiskan platform Android. •Beberapa hari berikutnya Google mengumumkan ketersediaan Android SDK Release Candidate 1.0. Oktober 2008, Google membuat kode program dari platform Android tersedia di bawah "**Apache's open source license**".

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

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



Gambar 1-1. Arsitektur Aplikasi Android

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

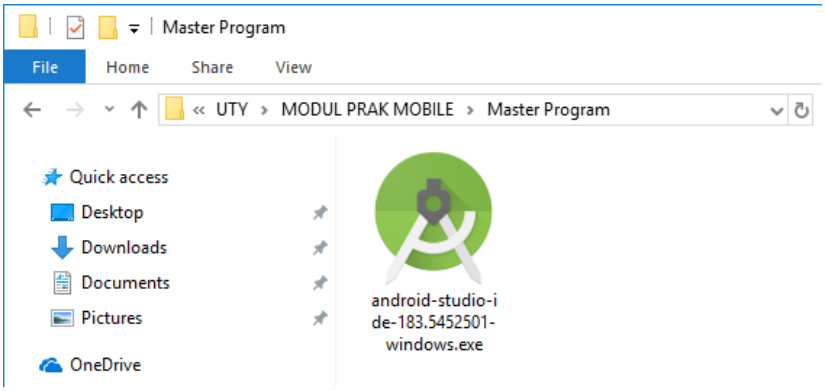
AVD pada dasarnya adalah emulator yang memungkinkan aplikasi Android untuk diuji tanpa perlu menginstal aplikasi pada perangkat berbasis Android fisik. Sebuah AVD dapat dikonfigurasi untuk meniru berbagai fitur perangkat keras termasuk pilihan ukuran layar, kapasitas memori dan fitur seperti kamera,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

PRAKTIK

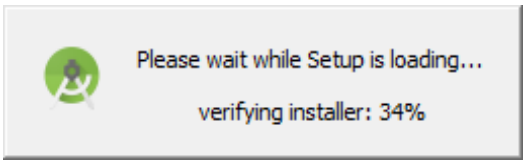
I. Install Android Studio

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

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

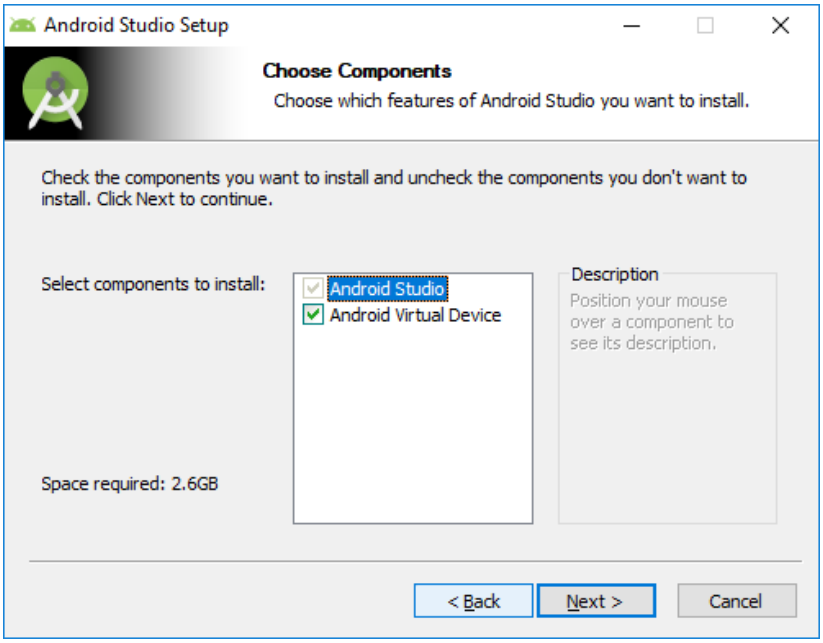


2. Tunggu beberapa saat sampai proses loading verifying installer selesai.

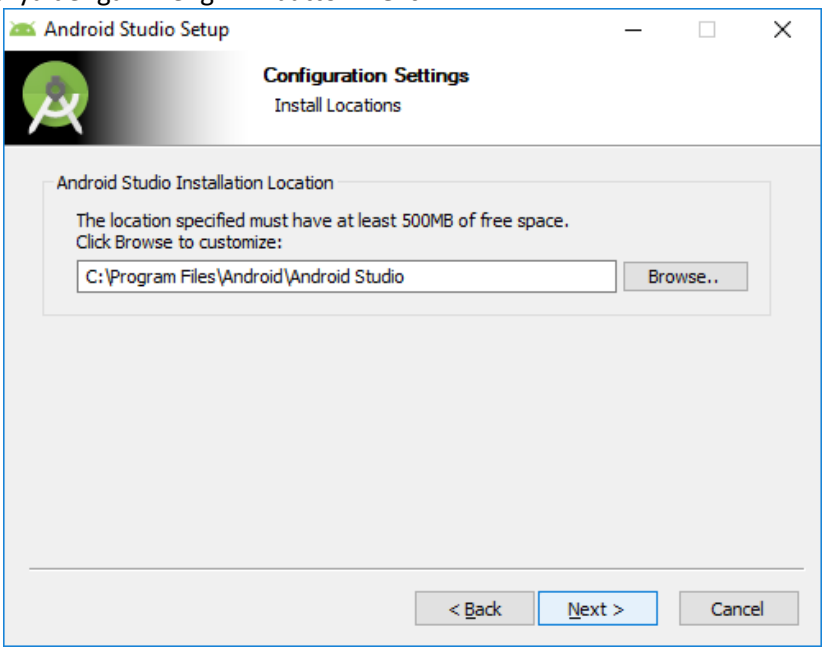


3. Setelah proses loading verifying installer selesai, akan terlihat jendela Android Studio Setup, **klik** button **Next >** untuk melanjutkan proses penginstallan.

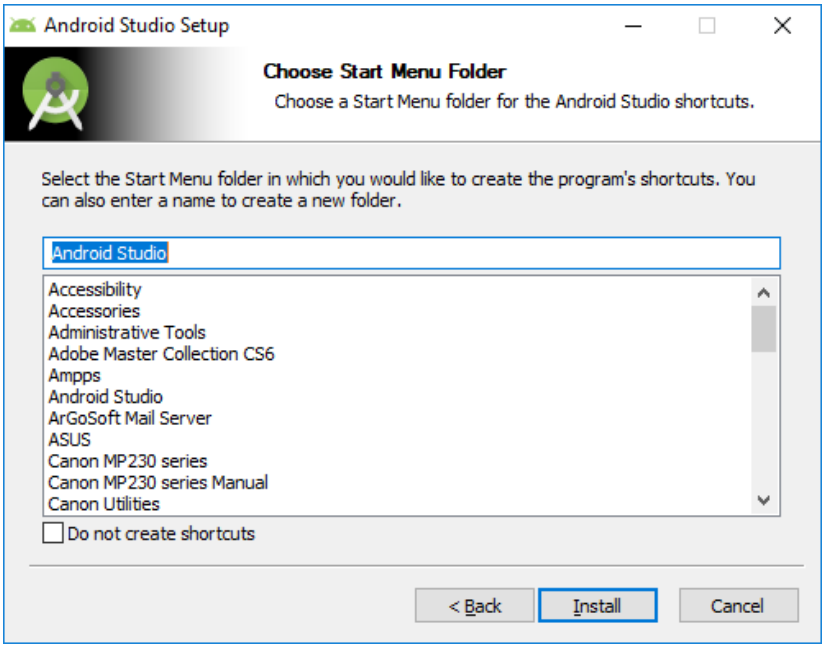
4. Beri tanda checklist atau centang bagian **Android Virutal Device**. Fungsi dari komponen Android Virtual Device yaitu untuk menampilkan interface android dalam bentuk virtual. Langkah selanjutnya **klik** pada button **Next >**



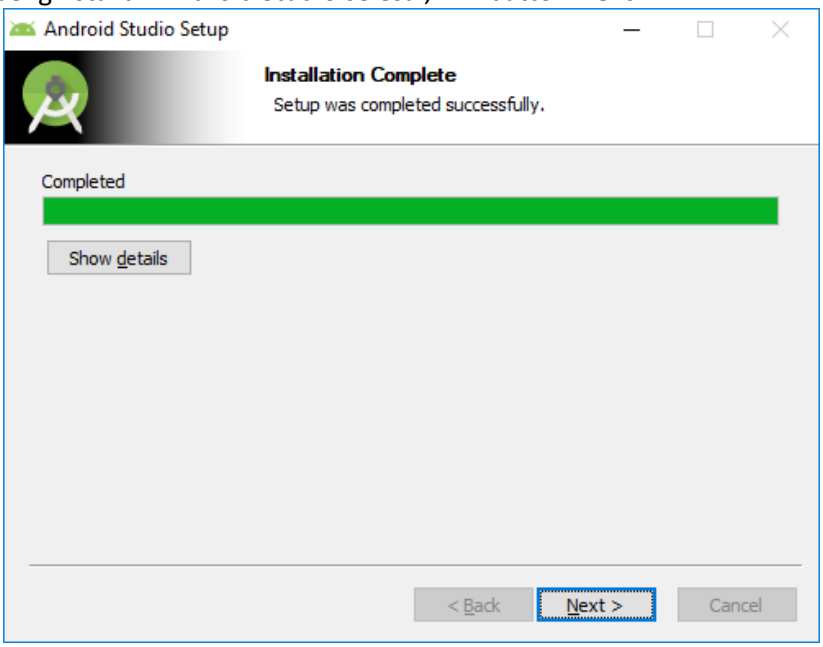
5. Pada bagian Configuration Settings akan diminta untuk memilih lokasi penginstallan, secara default lokasi penginstallan pada direktori **C:\Pragram Files\Android\Android Studio.** Jika ingin menggunakan lokasi defult maka dapat langsung menuju proses selanjutnya dengan meng-**klik** button **Next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



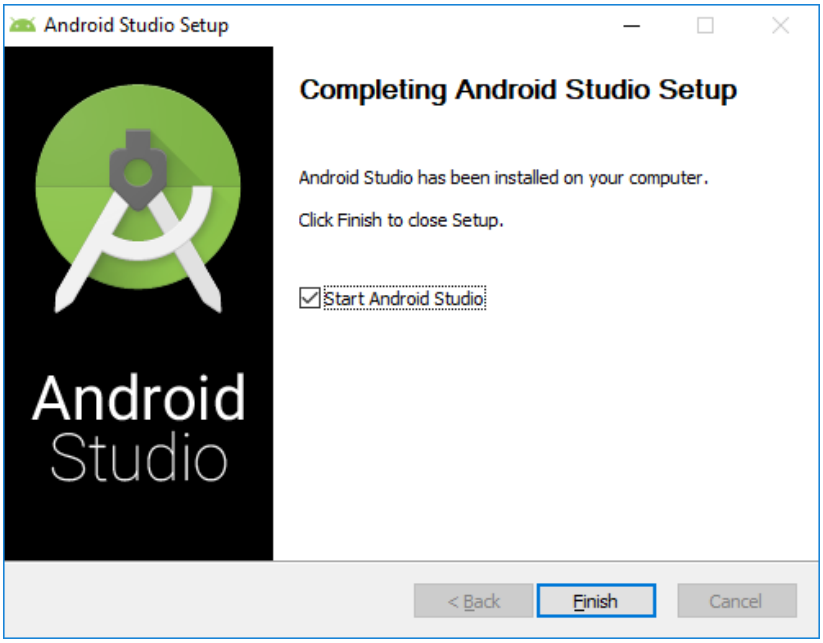
6. Pada bagian Choose Start Menu Folder, **klik** button **Install**



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



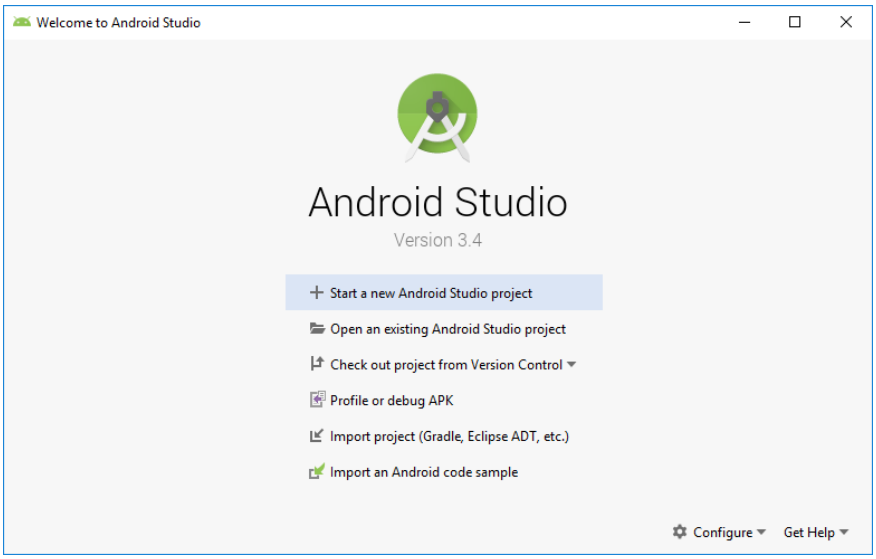
8. Proses penginstallan Android Studio telah selesai, **klik** button **Finish** untuk menutup jendela Android Studio Setup. Jika ingin menjalankan langsung program Android Studio maka beri tanda **checklist** atau centang pada bagian **Start Android Studio** sebelum menekan button Finish.



II. Running Program Android Studio

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

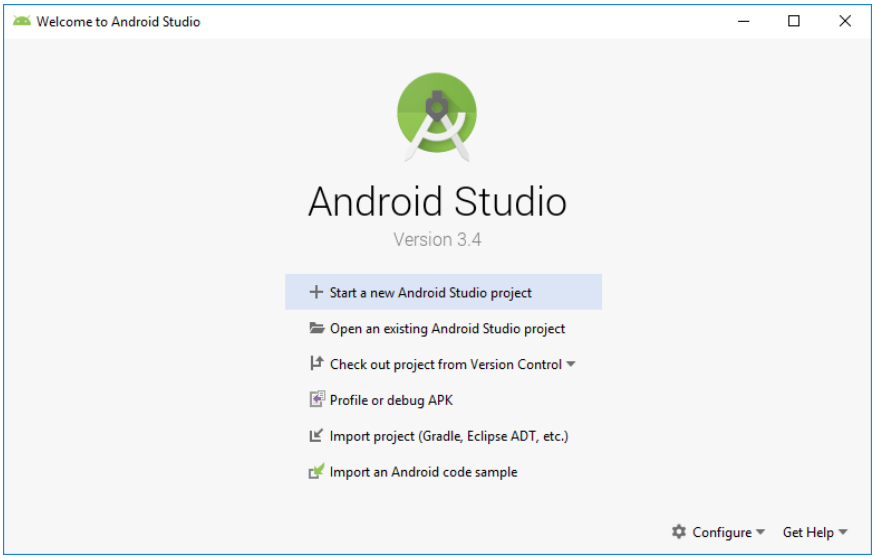
1. Buka Android Studio dari komputer yang dipakai.
2. Maka akan muncul jendela Welcome to Android Studio.



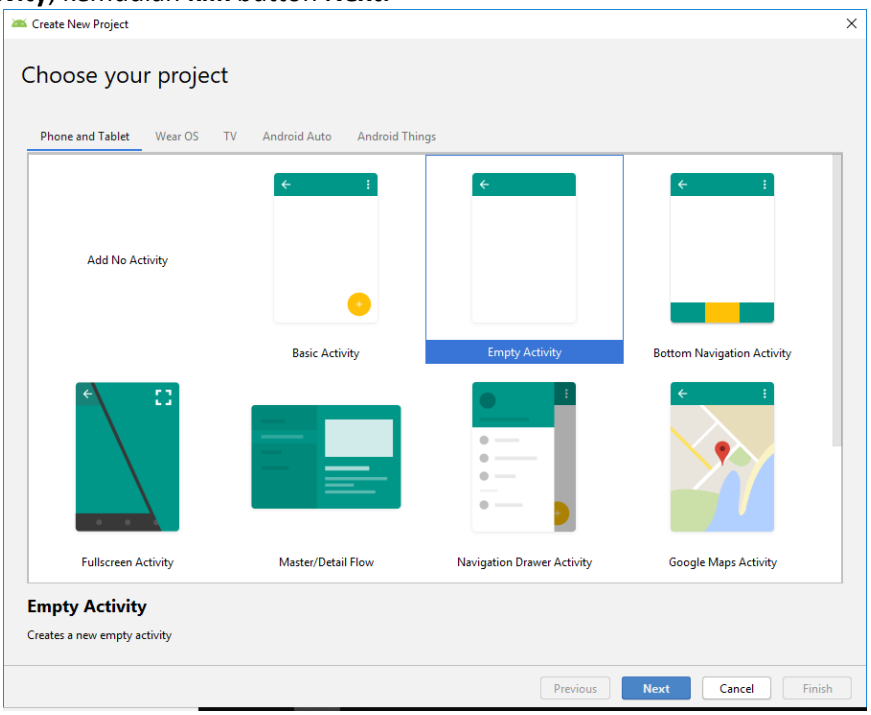
III. Membuat Project Dengan Android Studio

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

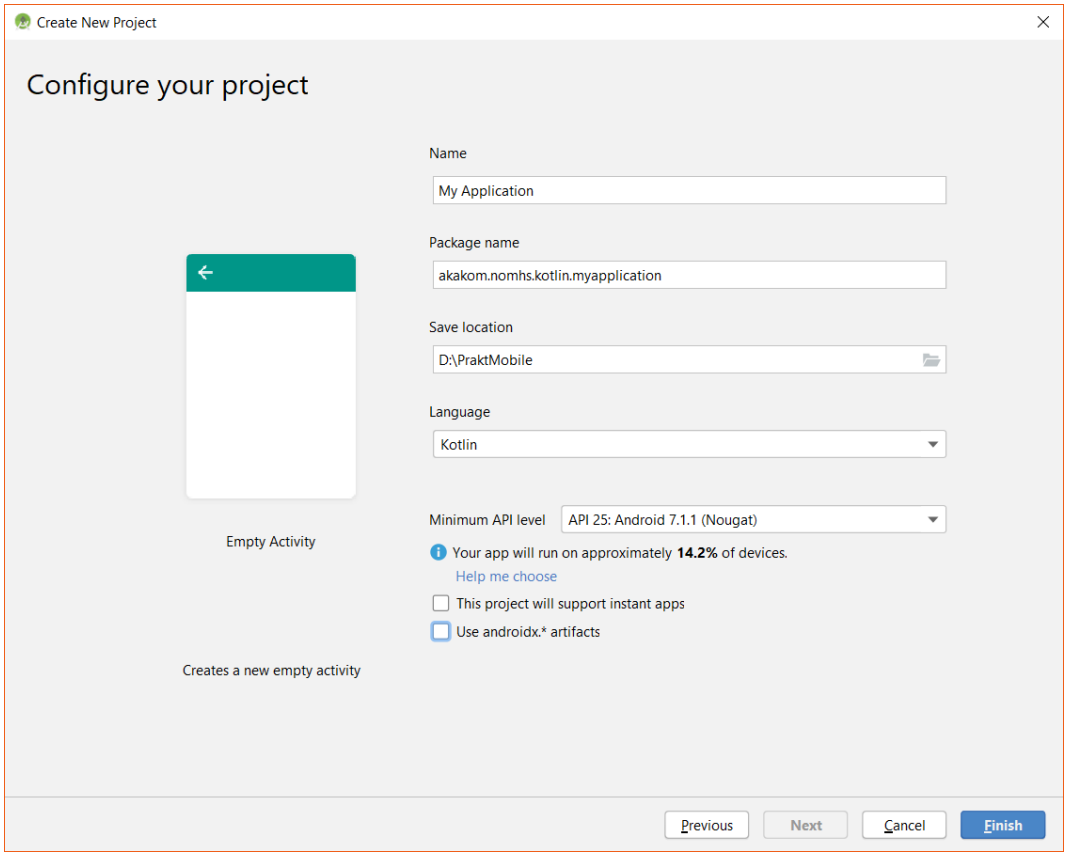
1. Dalam membuat project awal dengan Android Studio dengan cara **klik** pada **+ Start a new Android Studio project**



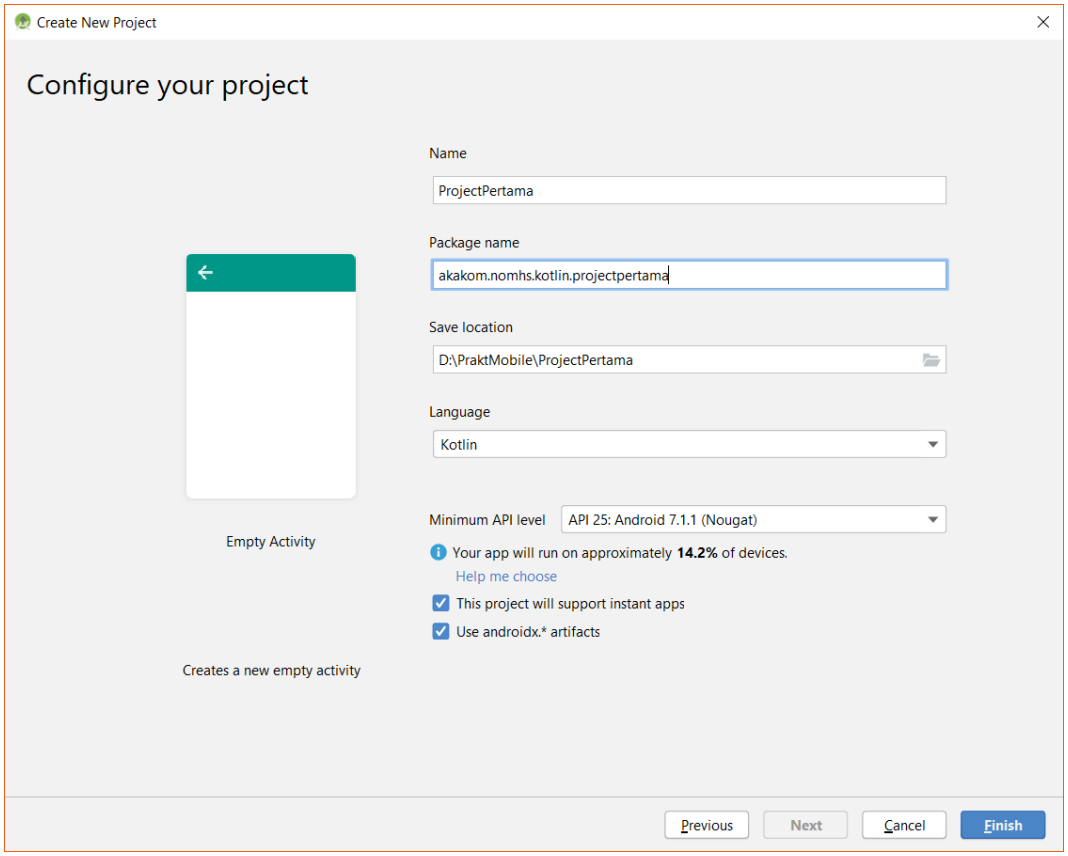
2. Kemudian akan terlihat jendela Create New Project, pada bagian ini programmer dapat memilih bentuk dari project yang akan dibuat (hal ini menyesuaikan dengan project yang akan dibuat). Sebagai latihan awal maka **pilih** pada bagian **Empty Activity**, kemudian **klik** button **Next.**



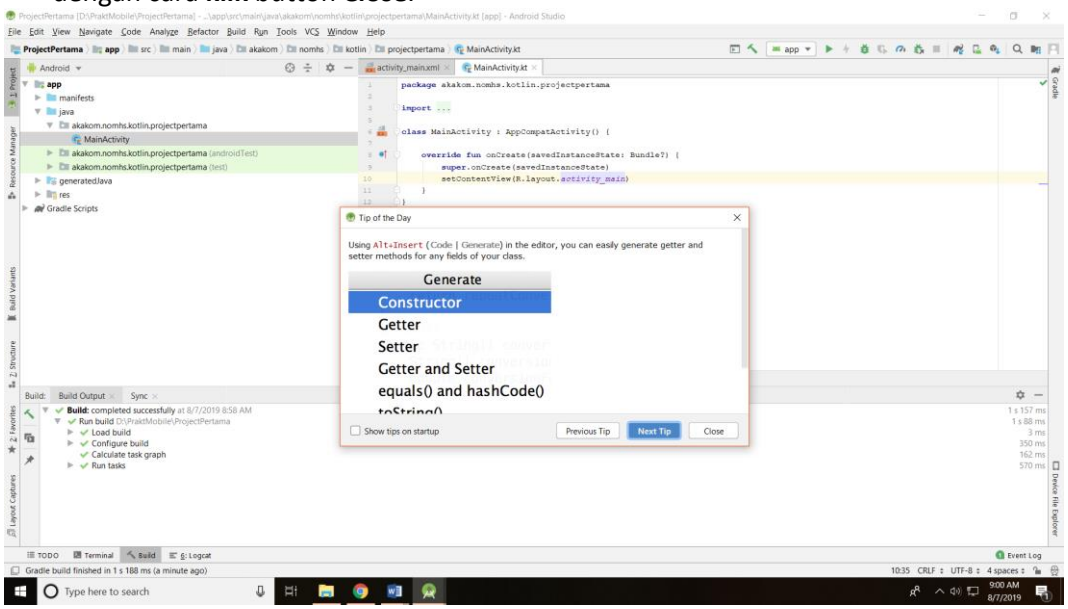
3. Pada jendela Create New Project, maka secara default Name project yaitu My Application, penamaan ini dapat dirubah sesuai dengan nama project yang dibuat.



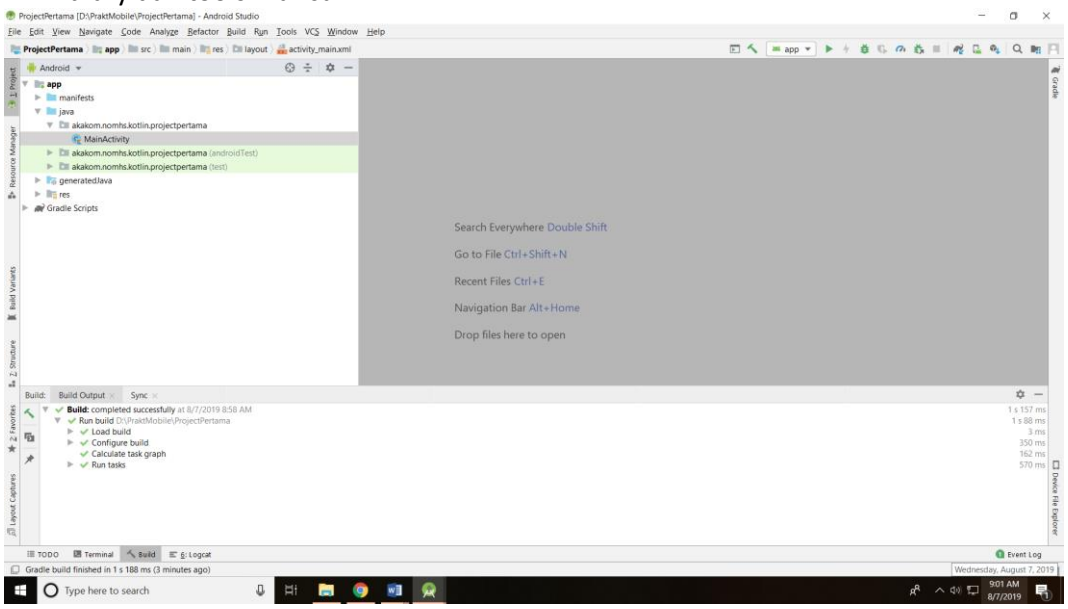
4. Sebagai bahan latihan, beri nama pada bagian Name : **ProjectPertama** kemudian pada bagian Save location : **.....** *(gunakan folder kerja anda),* kemudian **klik** button **Finish**.



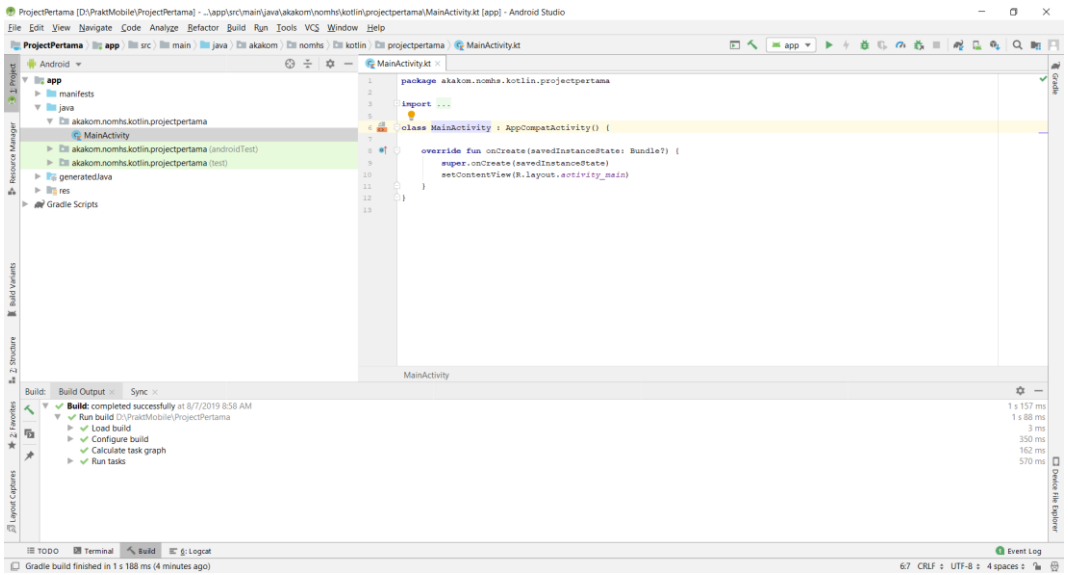
5. Pada saat awal atau pertamakali membuat project maka akan muncul sebuah jendela dengan nama Tips of The Day yaitu informasi tentang pemrograman yang dapat dijadikan acuan para programmer, untuk menutup jendela Tips of The Day dapat dengan cara **klik** button **Close**.



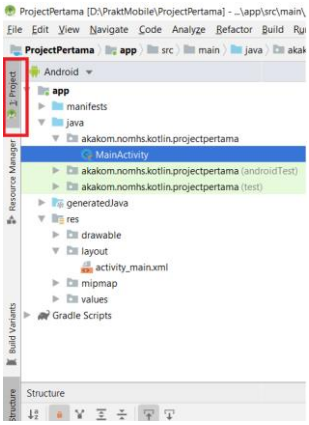
6. Jendela ProjectPertama akan muncul seperti pada gambar dibawah ini, dan tentunya belum secara utuh. Tunggulah beberapa saat sampai proses loading fitur-fitur atau library dan tools muncul.



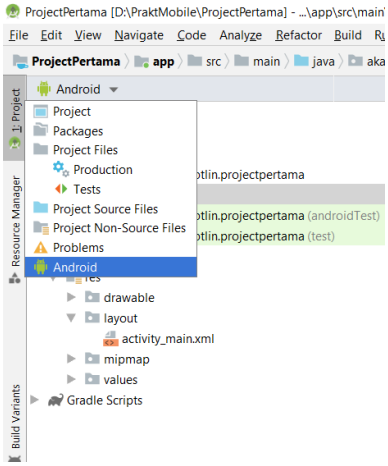
7. Jika proses loading fitur-fitur atau library selesai dan tools telah muncul, maka akan terlihat pada jendela IDE seperti pada gambar dibawah ini. Terdapat 3 bagian jendela pada Android Studio yaitu jendela Project, jendela Sourcecode dan jendela Built Output.



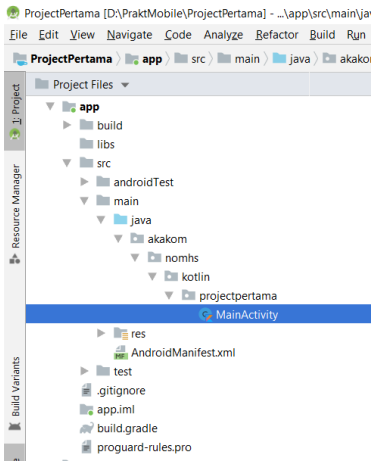
8. Pada jendela kiri atas, jika tab Project belum dipilih, klik tab Project.



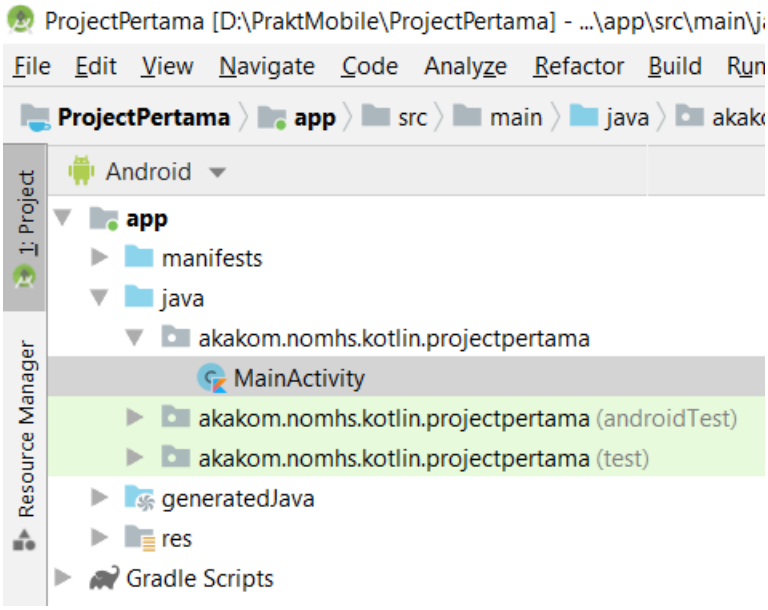
9. Hirarki Project dapat dipilih dengan menggunakan drop down menu. Hirarki standar adalah Android.



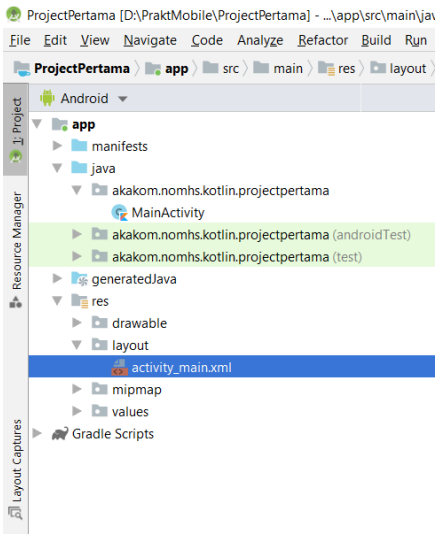
dan



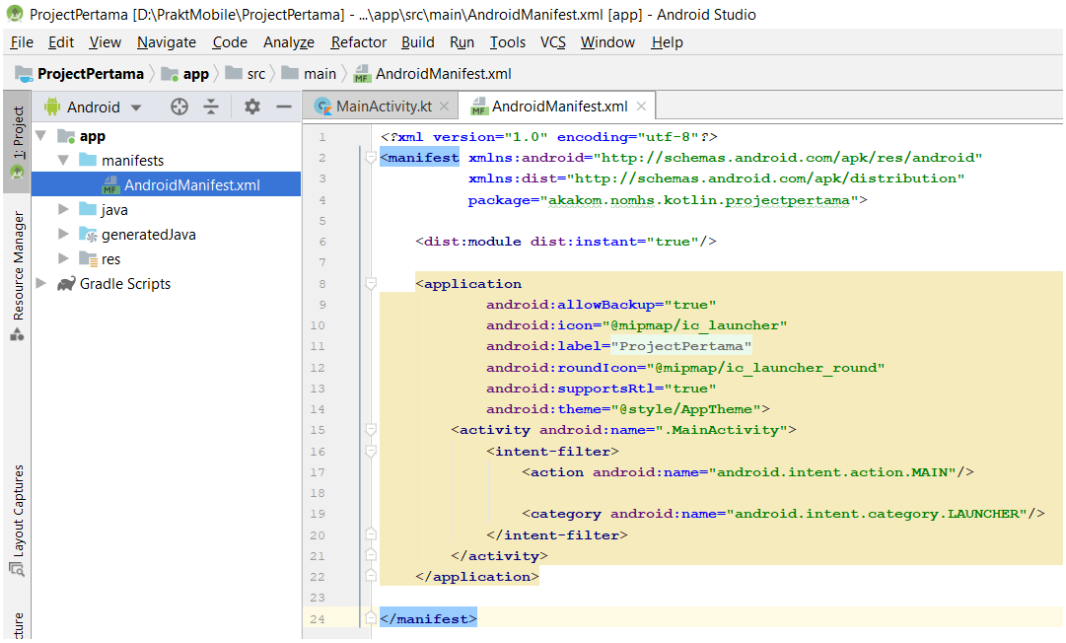
10. Untuk selanjutnya kita akan menggunakan hirarki Android.
11. Sekarang akan kita lihat panel Project. Yang pertama kita mengeksplorasi folder app.
12. Di panel Project> Android, eksplore folder app. Di dalam folder app ada empat subfolder: manifest, java, generateJava, dan res.
13. Buka folder java,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



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



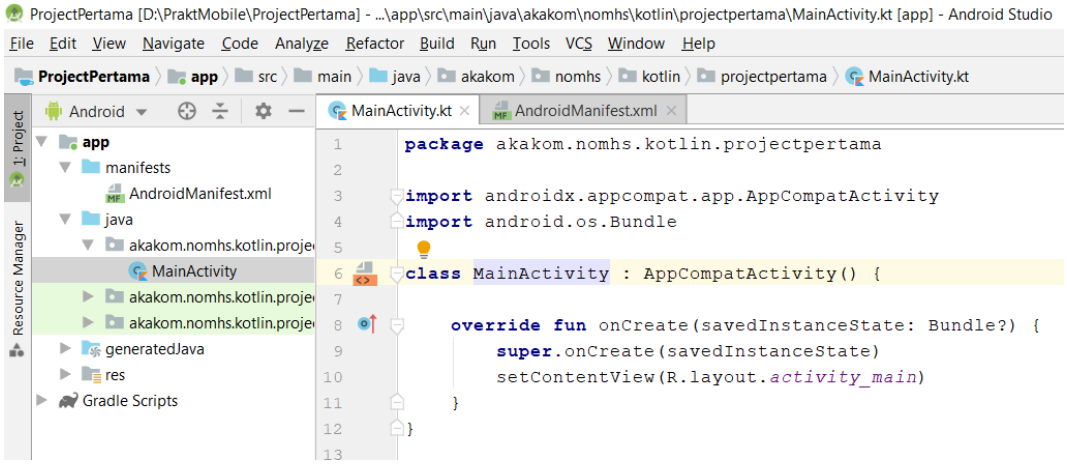
22. Activity kita biasanya dikaitkan dengan file tata letak UI, yang didefinisikan sebagai file XML di direktori res/layout. File tata letak itu biasanya dinamai berdasarkan aktivitasnya. Dalam hal ini, nama aktivitas adalah MainActivity, jadi layout yang terkait adalah activity_main.
23. Sekarang kita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



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

28. File AndroidManifest.xml juga merupakan tempat kita menentukan izin apa pun yang dibutuhkan aplikasi kita. Izin mencakup kemampuan aplikasi kita untuk membaca kontak telepon, mengirim data melalui internet, atau mengakses perangkat keras seperti kamera perangkat.
29. Terakhir, kita akan menjelajahi folder Script Gradle.
30. Gradle adalah sistem otomasi bangunan yang menggunakan bahasa khusus domain untuk menggambarkan struktur, konfigurasi, dan dependensi project app. Ketika kita mengkompilasi dan menjalankan aplikasi kita, kita melihat informasi tentang Gradle build running. kita juga melihat informasi tentang Android Package Kit (APK) yang diinstal. (APK adalah format file paket yang digunakan sistem operasi Android untuk mendistribusikan dan menginstal aplikasi seluler.)

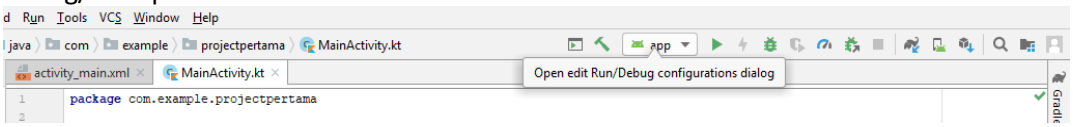
31. Kita akan fokus pada dua bagian pada bagian script program pada project Android yaitu .kt dan .xml. Untuk .kt dapat dibaca kotlin yaitu bahasa pemrograman pada Android Studio untuk membangun sistem. Sedangkan .xml dapat dibaca Extensible Markup Language yaitu bahasa pemrograman pada Android Studio untuk membangun user interface. **Klik** pada bagian **activity_main.xml** untuk melihat design atau atampilan project ProjectPertama.



IV. Pengenalan Aplikasi Android dan IDE Android Studio

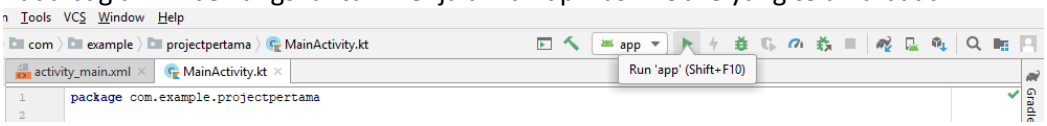
Untuk dapat membuat aplikasi mobile dengan Android Studio maka perlu mengenali lingkungan Aplikasi Android dan IDE Android Studio. Berikut beberapa IDE pada Android Studio yang nantinya akan digunakan selama proses pembuatan project :

1. Open edit Run/Debug configurations dialog
   Bagian ini berfungsi untuk memilih atau membuka, edit dan melakukan debug/run aplikasi.



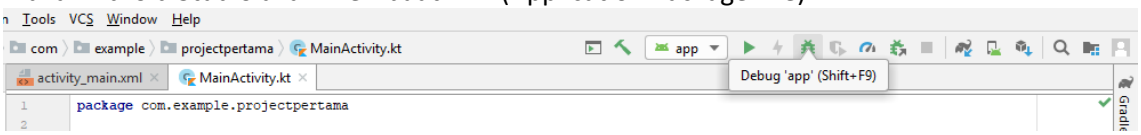
2. Run App

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



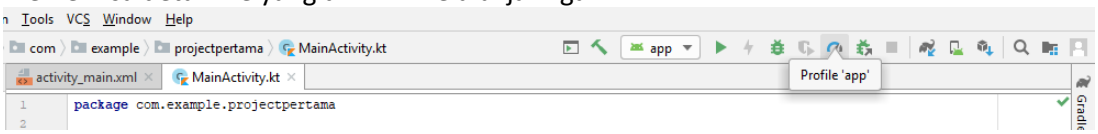
3. Debug App

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



4. Profile App

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

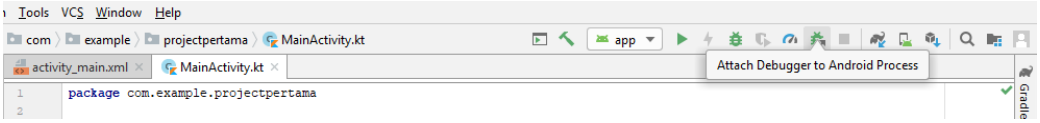


Dengan debugger Android Studio, maka programmer dapat :

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

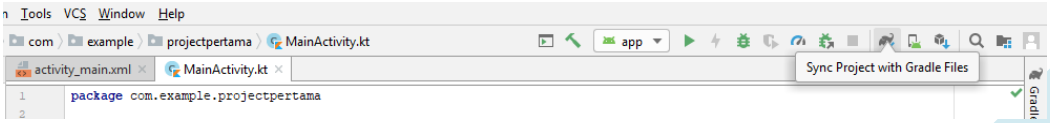
5. Attach Debugger to Android Process

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.

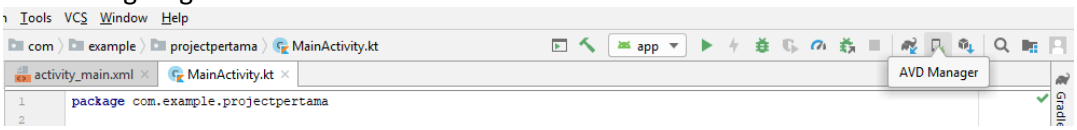


6. Sync Project with Gradle Files

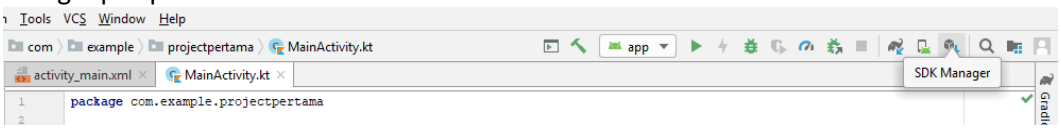
Jika programmer membuat perubahan pada file konfigurasi build dalam project Android yang dibuat, maka Android Studio mewajibkan programmer melakukan sinkronisasi file project sehingga sistem dapat mengimpor perubahan konfigurasi build dan menjalankan beberapa pemeriksaan untuk memastikan konfigurasi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



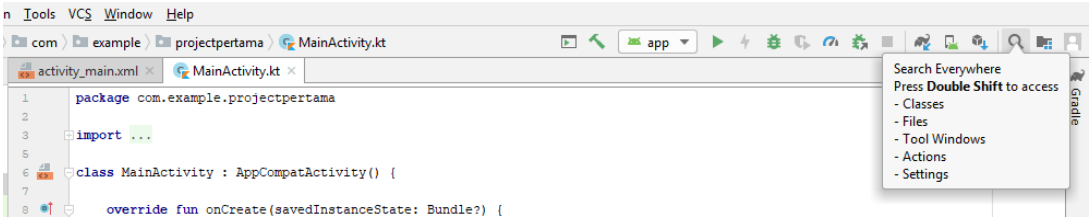
7. AVD Manager
   AVD atau Android Virtual Device adalah fitur pada Android Studio untuk membuat device dalam bentuk virtual sehingga ketika aplikasi atau project dirunning maka akan muncul di virtual device tersebut. Namun AVD juga dapat untuk membuat physical device yaitu melakukan running aplikasi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



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



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



V. Materi Pengayaan

a. Penggunaan Kotlin untuk Pengembangan Android.

Kotlin/Native memungkinkan developer untuk 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) untuk mencoba menjalankan program kotlin.

Dikutip dari https://kotlinlang.org/docs/reference/basic-syntax.html
Defining packages.
Package specification should be at the top of the source file:

```
package my.demo

import java.util.*

// ...
```

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

Defining **functions**

Function having two `Int` parameters with `Int` return type:

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

Target platform: JVMRunning on kotlin v. 1.3.41

Function with an expression body and inferred return type:

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

Target platform: JVMRunning on kotlin v. 1.3.41

Function returning no meaningful value:

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

Target platform: JVMRunning on kotlin v. 1.3.41

`Unit` return type can be omitted:

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

Target platform: JVMRunning on kotlin v. 1.3.41

See Functions.

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

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

```
val c: Int  // Type required when no initializer is provided
c = 3       // deferred assignment
```
Target platform: JVMRunning on kotlin v. 1.3.41

Variables that can be reassigned use the var keyword:

```
var x = 5 // `Int` type is inferred
x += 1
```
Target platform: JVMRunning on kotlin v. 1.3.41

Top-level variables:

```
val PI = 3.14
var x = 0

fun incrementX() {
    x += 1
}
```
Target platform: JVMRunning on kotlin v. 1.3.41

See also Properties And Fields.

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

```
// This is an end-of-line comment

/* This is a block comment
   on multiple lines. */
```
Unlike Java, block comments in Kotlin can be nested.
See Documenting Kotlin Code for information on the documentation comment syntax.

Using string templates

```
var a = 1
// simple name in template:
val s1 = "a is $a"

a = 2
// arbitrary expression in template:
val s2 = "${s1.replace("is", "was")}, but now is $a"
```
Target platform: JVMRunning on kotlin v. 1.3.41

See String templates.

Using conditional expressions

```
fun maxOf(a: Int, b: Int): Int {
    if (a > b) {
        return a
    } else {
        return b
    }
}
```
Target platform: JVMRunning on kotlin v. 1.3.41

Using if as an expression:

```
fun maxOf(a: Int, b: Int) = if (a > b) a else b
```
Target platform: JVMRunning on kotlin v. 1.3.41

See if-expressions.

Using nullable values and checking for null

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

Return null if str does not hold an integer:

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

Use a function returning nullable value:

```
fun printProduct(arg1: String, arg2: String) {
    val x = parseInt(arg1)
    val y = parseInt(arg2)

    // Using `x * y` yields error because they may hold nulls.
    if (x != null && y != null) {
        // x and y are automatically cast to non-nullable after null check
        println(x * y)
    }
    else {
        println("either '$arg1' or '$arg2' is not a number")
    }
}
```
Target platform: JVMRunning on kotlin v. 1.3.41

or

```
// ...
if (x == null) {
    println("Wrong number format in arg1: '$arg1'")
    return
}
if (y == null) {
    println("Wrong number format in arg2: '$arg2'")
    return
}

// x and y are automatically cast to non-nullable after null check
println(x * y)
```
Target platform: JVMRunning on kotlin v. 1.3.41

See Null-safety.

Using type checks and automatic casts

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

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

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

```
}
```
Target platform: JVMRunning on kotlin v. 1.3.41

or

```
fun getStringLength(obj: Any): Int? {
    if (obj !is String) return null

    // `obj` is automatically cast to `String` in this branch
    return obj.length
}
```
Target platform: JVMRunning on kotlin v. 1.3.41

or even

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

    return null
}
```
Target platform: JVMRunning on kotlin v. 1.3.41

See Classes and Type casts.

Using a for loop

```
val items = listOf("apple", "banana", "kiwifruit")
for (item in items) {
    println(item)
}
```
Target platform: JVMRunning on kotlin v. 1.3.41

or

```
val items = listOf("apple", "banana", "kiwifruit")
for (index in items.indices) {
    println("item at $index is ${items[index]}")
}
```
Target platform: JVMRunning on kotlin v. 1.3.41

See for loop.

Using a while loop

```
val items = listOf("apple", "banana", "kiwifruit")
var index = 0
while (index < items.size) {
    println("item at $index is ${items[index]}")
    index++
}
```
Target platform: JVMRunning on kotlin v. 1.3.41

See while loop.

Using when expression

```
fun describe(obj: Any): String =
    when (obj) {
        1          -> "One"
        "Hello"    -> "Greeting"
        is Long    -> "Long"
        !is String -> "Not a string"
        else       -> "Unknown"
    }
```
Target platform: JVMRunning on kotlin v. 1.3.41

See when expression.

Using ranges

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

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

Target platform: JVMRunning on kotlin v. 1.3.41

Check if a number is out of range:

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

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

Target platform: JVMRunning on kotlin v. 1.3.41

Iterating over a range:

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

Target platform: JVMRunning on kotlin v. 1.3.41

or over a progression:

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

Target platform: JVMRunning on kotlin v. 1.3.41

See Ranges.

Using collections

Iterating over a collection:

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

Target platform: JVMRunning on kotlin v. 1.3.41

Checking if a collection contains an object using in operator:

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

Target platform: JVMRunning on kotlin v. 1.3.41

Using lambda expressions to filter and map collections:

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

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

Target platform: JVMRunning on kotlin v. 1.3.41

See Higher-order functions and Lambdas.

Creating basic classes and their instances:

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

Target platform: JVMRunning on kotlin v. 1.3.41

See classes and objects and instances.

LATIHAN

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

TUGAS

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

REFERENSI

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