Summary

Mobile Programming Android & Kotlin

Code yang digunakan dalam course ini bisa kamu dapatkan melalui link:
https://github.com/skillacademyid/programming-with-android

MOBILE PROGRAMMING

- Pengenalan mobile programming dan android
    - Apa itu mobile programming?
      Pemrograman aplikasi yang ditujukan untuk smartphone, baik itu Android maupun iOS. Menggunakan bahasa pemrograman dan **software development kit (SDK)** yang memang digunakan untuk mengembangkan aplikasi mobile. Misalnya, dengan menggunakan Kotlin atau Swift. Mobile programming itu sendiri dapat dibagi kedalam 2 kategori pemrograman. **Native mobile programming** dengan menggunakan software development kit (SDK) yang disiapkan oleh pemilik SDK tersebut **(Google/Apple)**, maupun **Hybrid mobile programming** menggunakan bahasa pemrograman yang sama dengan ketika kita akan membuat sebuah website, contohnya adalah **JavaScript** atau bahasa baru seperti **Dart** dengan menggunakan **framework Flutter**.

    - Apa itu android?
      Sistem operasi berbasis **linux dan open source** (kode sumber terbuka) yang dikembangkan oleh google bersama dengan beberapa perusahaan teknologi lainnya, yang tergabung ke dalam satu konsorsium besar dengan nama **open handset alliance** (OHA). Android tidak hanya dibuat khusus untuk Smartphone, sistem operasi tersebut saat ini juga dapat digunakan untuk tablet, smart watch, smart tv dan bahkan smart car, untuk navigation system pada mobil kita. Saat ini versi android paling terbaru adalah **Android10**. Dimana, pada versi tersebut sudah disematkan banyak sekali fitur cerdas yang mampu menjadikan smartphone kita saat ini jauh lebih pintar lagi, contohnya seperti kemampuan untuk menambah intelegensi pada software **Digital Wellbeing** di smartphone kita, yang mampu mengatur secara otomatis kapan dan dimana smartphone kita harus beradaptasi terhadap perubahan waktu, dengan tujuan untuk membantu kita agar mengurangi ketergantungan terhadap smartphone yang berlebihan.

    - Apa itu kotlin?
      Bahasa pemrograman yang dibuat oleh **JetBrains (**https://www.jetbrains.com**)** yang mulai dikembangkan pada tahun **2010**. Kotlin merupakan bahasa pemrograman yang bersifat static typed, berorientasi objek dan pemrograman fungsional, yang ditujukan untuk **JVM, Android, Web/JavaScript dan juga Native platform**. Nama kotlin itu sendiri diambil dari sebuah pulau di Rusia dengan nama yang sama. Versi pertama dari Kotlin mulai dirilis pada **Februari 2016**. Pada pertengahan **2017**, **Google** telah mengumumkan bahwa Kotlin menjadi bahasa utama yang akan digunakan untuk pengembangan aplikasi android selanjutnya.

Saat ini, versi Kotlin sudah sampai pada versi **1.3.60** yang baru saja dirilis pada **November 2019**. Apa saja kelebihan dari bahasa Kotlin tersebut ?

- **Concise**, yang artinya penulisan syntax pada pemrograman dengan bahasa Kotlin sangat ringkas, sehingga mampu mengurangi jumlah baris kode yang terlalu panjang, seperti pemrograman dengan bahasa Java, yang dikenal dengan istilah **Minimize Boilerplate Code**.
- **Safe**, Kotlin mampu menghindari beberapa kesalahan fatal atau error ketika menjalankan aplikasi sehingga aplikasi berhenti bekerja seketika, seperti mencegah terjadinya **NullPointerException**.
- **Interoperable**, Secara bahasa Kotlin mampu menggunakan ***dependency*** atau ***library*** yang sudah disediakan oleh Java secara **100%** dan juga sebaliknya. Sehingga, kita bisa saja menulis dengan Kotlin dan memanggilnya pada ***class*** di Java dan sebaliknya.
- **Tool-Friendly**, Dengan JetBrains dan Google sebagai pendukung utama dalam pengembangan bahasa Kotlin. Maka dapat dipastikan sudah tersedia berbagai macam bantuan ataupun ***utility*** yang dibutuhkan oleh developer, ketika menulis kode menggunakan bahasa pemrograman Kotlin.

JAVA DEVELOPMENT KIT (JDK), ANDROID STUDIO & KOTLIN

- Download Java Development Kit
  - Windows (32bit) : https://download.oracle.com/otn/java/jdk/8u231-b11/5b13a193868b4bf28bcb45c792fce896/jdk-8u231-windows-i586.exe
  - Windows (64bit) : https://download.oracle.com/otn/java/jdk/8u231-b11/5b13a193868b4bf28bcb45c792fce896/jdk-8u231-windows-x64.exe
  - Mac OS X **(alternatif selain menggunakan brew)** : https://download.oracle.com/otn/java/jdk/8u231-b11/5b13a193868b4bf28bcb45c792fce896/jdk-8u231-macosx-x64.dmg

- Download Android Studio : https://developer.android.com/studio

- Download Kotlin Compiler **(command-line tools) :** https://github.com/JetBrains/kotlin/releases/download/v1.3.60/kotlin-compiler-1.3.60.zip

MOBILE PROGRAMMING ANDROID USING KOTLIN

- Dasar Pemrograman Android dengan Kotlin
  - Dasar pemrograman kotlin, mulai dari fitur dan fungsi dasar
    Beberapa cara untuk latihan dasar pemrograman kotlin bisa dilakukan melalui :
    - **Kotlin playground website**, dengan mengunjungi https://play.kotlinlang.org
    - **Kotlin command line tools** (compiler), setelah rekan-rekan berhasil meng-install **Kotlin Compiler**, silahkan membuka (Command Prompt pada Windows) atau (Terminal pada Mac OS X), dan ketik perintah **kotlinc**

- Membuat file **Kotlin** sederhana dengan menggunakan **Android Studio**

Beberapa fitur dasar yang sering digunakan ketika kita menulis kode Kotlin :

- **Static** atau **final modifier** dengan **val**
- **Modified variable** dengan **var**
- **Optional variable/data type** dengan tanda "**?**"
- **Explicit variable/data type**, contohnya -> **val name: String = "Kotlin"**
- **Implicit variable/data type**, contohnya -> **val name = "Kotlin"**
- **String templates** sebagai ***utility*** yang dapat digunakan untuk menambahkan value dari suatu variabel dengan tipe data apapun ke dalam tipe data **String**
- **Kotlin class** sebagai dasar dalam membuat sebuah class di Kotlin
- **Kotlin data class** sebagai dasar dalam membuat tipe data/objek baru, yang dapat digunakan di Kotlin class
- **Object** sebagai dasar dalam membuat pola/konsep **Singleton**

Beberapa fungsi dasar yang sering digunakan ketika kita menulis kode Kotlin;

- **Basic function**, menggunakan modifier **fun**
- **Basic function** dengan menambahkan **return value**, contohnya -> **fun getName(): String = "Kotlin" Basic function** dengan **parameter optional**, contohnya -> **fun setName(name: String = "Kotlin") {}**

- Cara membuat contoh aplikasi sederhana di Android, serta fungsi event listener sederhana
  - Membuat project android baru, dengan Android Studio
  - Membuat **action/event listener** sederhana di Android dengan fungsi **setOnClickListener**

- Arsitektur Aplikasi Android
  - Standar arsitektur siklus hidup **(life cycle)** dari masing-masing komponen yang ada pada Android

    Siklus hidup **(life cycle)** dari suatu **Activity**, meliputi;
    - **onCreate**, dipanggil ketika suatu **Activity** dijalankan pertama kali
    - **onPause**, dipanggil ketika suatu **Activity** dihentikan oleh user, dengan menekan button **Home** pada smartphone
    - **onResume**, dipanggil ketika suatu **Activity** dibuka kembali, setelah user menekan button **Home** pada smartphone
    - **onDestroy**, dipanggil ketika suatu **Activity** dihentikan oleh user, dengan menekan button **Back** pada smartphone atau pada navigasi **Back** yang ada di layar smartphone

    Siklus hidup **(life cycle)** dari suatu **Fragment**, meliputi;
    - **onCreateView**, dipanggil ketika suatu **Fragment** di dalam **Activity** dijalankan pertama kali
    - **onViewCreated**, dipanggil setelah layout dari **Fragment** berhasil ditampilkan pada layar smartphone user

    - **onDestroyView**, dipanggil ketika suatu **Fragment** yang ada di dalam **Activity** diganti dengan **Fragment** lainnya atau ketika user menekan button **Back** pada smartphone atau pada navigasi **Back** yang ada di layar smartphone

  - Komponen UI yang ada di Android
    - **ConstraintLayout**, digunakan untuk membuat layout dinamis pada Android
    - **TextView**, digunakan untuk menampilkan label atau text dasar pada Android
    - **Button**, digunakan untuk menampilkan button dasar pada Android
    - **InputText**, digunakan untuk menampilkan inputan, baik berupa text, angka, password dan inputan lainnya pada Android
    - **RecyclerView**, digunakan untuk membuat tampilan berupa **list** pada Android
    - **CustomView** dengan bantuan **FrameLayout**, digunakan sebagai alternatif dalam membuat tampilan tanpa bantuan **Fragment**

  - Cara membuat komunikasi **client-server** sederhana di Android, menggunakan networking library
    - Belajar memahami konsep **REST API** dengan bantuan aplikasi, seperti **Postman (https://getpostman.com)**
    - Belajar menggunakan library networking, seperti **OkHttp Client** dan **Retrofit**
    - Belajar menambahkan **Android Internet Permission** pada **Android Manifest**
    - Belajar implementasi networking sederhana dengan contoh aplikasi **MovieDB (https://developers.themoviedb.org/3/getting-started/introduction)**
    - Menggunakan **Google GSON** untuk manipulasi **JSON** object
    - Referensi OKHttp Client -> https://square.github.io/okhttp
    - Referensi Retrofit -> https://square.github.io/retrofit
    - Referensi Google GSON -> https://github.com/google/gson

- Android Basic Architecture and Design Pattern
  - Basic View Binding dengan **Kotlin Android Extensions**
    - Belajar memahami konsep dasar dari **findViewById**
    - Belajar memahami konsep dasar dari **Kotlin Android Extensions**, dengan menambahkan plugin baru pada file **gradle** seperti berikut -> **apply plugin: 'kotlin-android-extensions'**
    - Komparasi antara **findViewById** dengan **Kotlin Android Extensions**
  - Dependency Injection
    - Introduction to Dagger2 by Google **(https://dagger.dev)**
    - Belajar dasar dari **Dependency Injection (DI)** menggunakan **Dagger2**
    - Belajar memahami konsep dasar dari **Kotlin Annotation Processing**, dengan menambahkan plugin baru pada file **gradle** seperti berikut -> **apply plugin: 'kotlin-kapt'**
    - Belajar membuat dependency module
    - Belajar membuat dependency scope per module
    - Belajar integrasi dependency component antar module
    - Pemahaman tentang dependency graph pada dagger2
  - Model-View-Presenter

- Pengenalan tentang design pattern **Model-View-Presenter (MVP)**
- Belajar membedakan antara design pattern **MVP** dengan **Model-View-Controller (MVC)**
- Belajar implementasi sederhana, bagaimana menggunakan **MVP**

○ Reactive Programming

- Introduction to ReactiveX **(http://reactivex.io)**
- Belajar memahami konsep **Observable** menggunakan **Reactive Programming**, referensi bisa dilihat pada -> http://reactivex.io/intro.html
- Menambahkan dependency **RxJava (https://github.com/ReactiveX/RxJava)**
- Menambahkan dependency **RxKotlin (https://github.com/ReactiveX/RxKotlin)**
- Menambahkan dependency **RxAndroid (https://github.com/ReactiveX/RxAndroid)**
- Menambahkan support reactive adapter dependency untuk **Retrofit (https://github.com/square/retrofit/tree/master/retrofit-adapters/rxjava2)**
- Belajar integrasi **Retrofit Adapter** menggunakan Reactive Programming
- Pemahaman tentang **Schedulers** yang terdapat pada Reactive Programming, seperti;
  - **io** : Tipe scheduler yang digunakan untuk melakukan binding pada proses input/output dari suatu stream secara **asynchronous**
  - **immediate** : Tipe scheduler yang digunakan untuk melakukan proses stream secara langsung pada thread yang sama
  - **trampoline** : Tipe scheduler yang digunakan untuk melakukan proses stream secara berkala **(queuing schedule)** pada thread yang sama
  - **newThread** : Tipe scheduler yang digunakan untuk melakukan proses stream dengan cara selalu membuat thread baru
  - **computation** : Tipe scheduler yang digunakan untuk melakukan proses stream sebagai proses komputasi (perhitungan) terhadap data yang sedang dieksekusi di dalam stream, fungsi ini juga disebut sebagai **event-loop processing**
- Pemahaman tentang berbagai macam metode stream yang terdapat pada Reactive Programming, seperti;
  - **Observable** : Stream yang digunakan untuk melakukan proses ***threading*** kurang lebih sebanyak 1000 element dalam 1 proses
  - **Flowable** : Stream yang berfungsi hampir serupa dengan **Observable**, namun memiliki urutan atau *flow* dari 1 proses ke proses lainnya dalam stream yang sama
  - **Single** : Stream yang digunakan untuk melakukan proses ***threading***, yang dimana output dari proses tersebut hanya terdapat 1 thread saja sehingga bisa dikatakan lebih ringan dan tidak membebani pemakaian memori yang cukup besar untuk melakukan proses thread
  - **Maybe** : Stream yang digunakan untuk melakukan proses ***threading*** dengan pendekatan optional, dimana hasil dari thread tersebut akan mengeluarkan dua buah proses, antara benar/salah ataupun sukses/gagal
  - **Completable** : Stream yang digunakan untuk melakukan proses ***threading*** dengan hanya melihat hasil akhir, tanpa melakukan pengecekan terhadap proses di dalam stream tersebut, apakah prosesnya akan sukses ataupun gagal