# BAB IV ANDROID STUDIO #2

## Tujuan

1. Praktikan mampu mengetahui lingkungan pengembangan Sistem Operasi *Mobile* Android.
2. Praktikan mampu memahami model pengembangan MVP (*Model – View –Presenter)*
3. Praktikan memahami konsep penggunaan API *(Application Program Interface)*
4. Praktikan memahami konsep penggunaan retrofit.

## Alat dan Bahan

1. PC / Notebook (minimal RAM 4GB)
2. Android Studio 3.2, JDK, Android SDK
3. Device Android dan kabel USB (wajib)

## Dasar Teori

### API

API adalah singkatan dari *Application Program Interface*, dan memungkinkan *developer* untuk mengintegrasikan dua bagian dari aplikasi atau dengan aplikasi yang berbeda secara bersamaan. API terdiri dari berbagai elemen seperti function, protocols, dan tools lainnya yang memungkinkan *developer* untuk membuat aplikasi. Tujuan penggunaan API adalah untuk mempercepat proses *development* dengan menyediakan *function* secara terpisah sehingga developer tidak perlu membuat fitur yang serupa. Penerapan API akan sangaat terasa jika fitur yang diinginkan sudah sangat kompleks, tentu membutuhkan waktu untuk membuat yang serupa dengannya. Misalnya: integrasi dengan *payment gateway*. Terdapat berbagai jenis sistem API yang dapat digunakan, termasuk sistem operasi, *library, dan web*.

(Sumber: codepolitan.com)

### Retrofit

Retrofit adalah library Android yang dibuat oleh Squre yang digunakan sebagai REST Client pada Android, yang pasti akan memudahkan kita dalam programing. Karena kita tidak perlu lagi untuk membuat method-method sendiri untuk menggunakan REST Client API dari backend.

(sumber:medium.com)

### MVP

MVP (Model-View-Presenter) adalah pola arsitektur antarmuka pengguna dirancang untuk memfasilitasi unit testing otomatis dan meningkatkan pemisahan keprihatinan di logika presentasi:

1. Model adalah antarmuka yang mendefinisikan data yang akan ditampilkan atau ditindaklanjuti dalam antarmuka pengguna.
2. View adalah antarmuka pasif yang menampilkan data (model) dan mengarahkan perintah pengguna (kejadian) ke presenter untuk bertindak atas data tersebut.
3. Presenter bertindak berdasarkan model dan pandangan. Ini mengambil data dari repositori (model), dan memformatnya untuk ditampilkan dalam tampilan.

Biasanya, implementasi tampilan instantiates objek presenter beton, menyediakan referensi untuk dirinya sendiri.

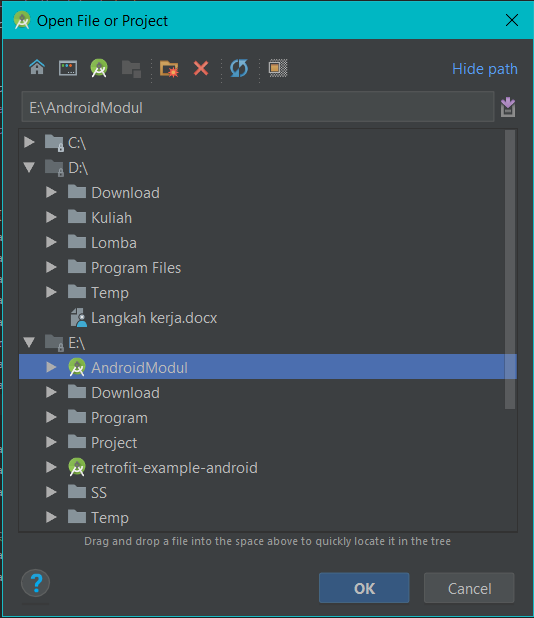
(sumber:medium.com)

## Langkah Percobaan

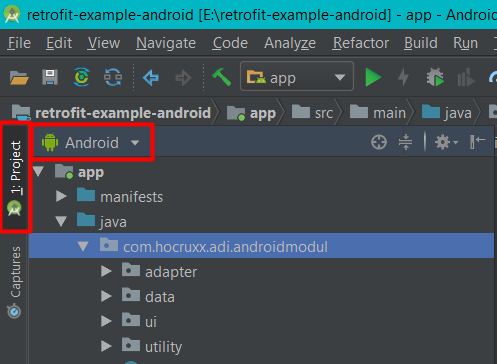
### Persiapan Package

### Buka Android Studio

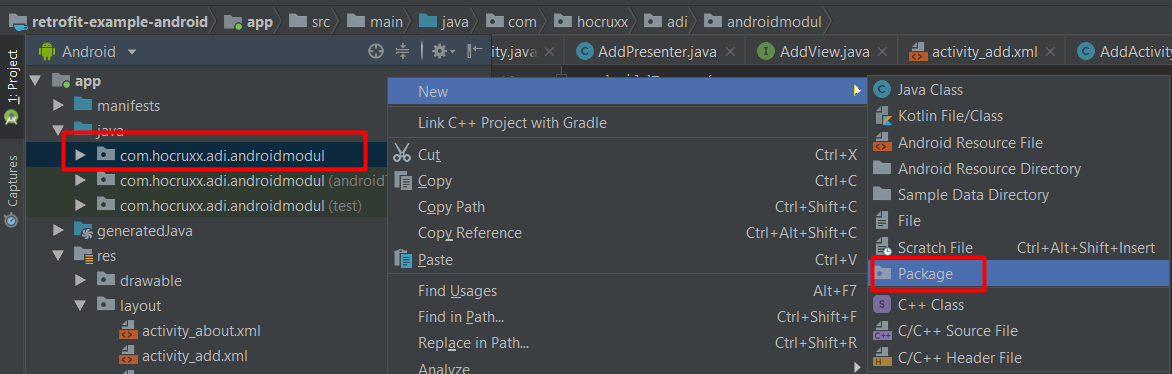
### Buka backup file project Modul 2, dengan cara klik File – Open – Arahkan ke File Project Android Modul 2



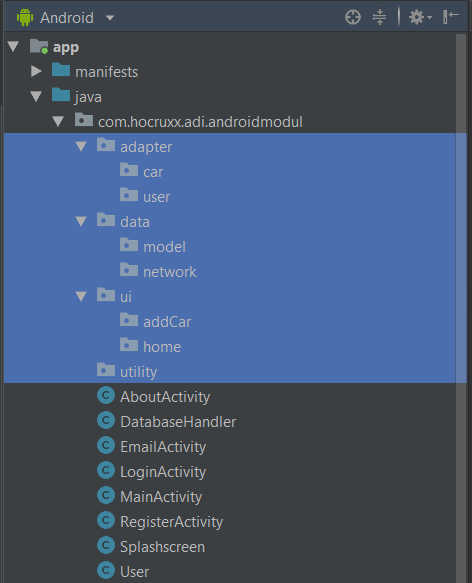
1. Tunggu hingga **build.gradle** dan **indexing** selesai seluruhnya
2. Pada tab **project**, pilih mode **Android**.



1. Lalu buat package baru dengan cara klik kanan pada **main package java – new – package**



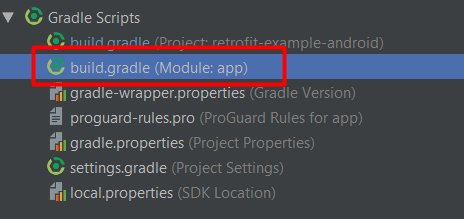
1. Tambahkan beberapa package baru sehingga menjadi seperti berikut



Package – package ini akan digunakan untuk menampung class – class dalam metode MVP sesuai dengan kegunaanya, dengan ini dokumentasi projek akan lebih mudah dan terstruktur.

### Persiapan Retrofit

1. Selanjutnya, tambahkan beberapa dependencies ke build.gradle. Dobel klik **build.gradle (Module:app)**.



1. Lalu tambahkan, code berikut

//Design

implementation 'com.android.support:recyclerview-v7:28.0.0'

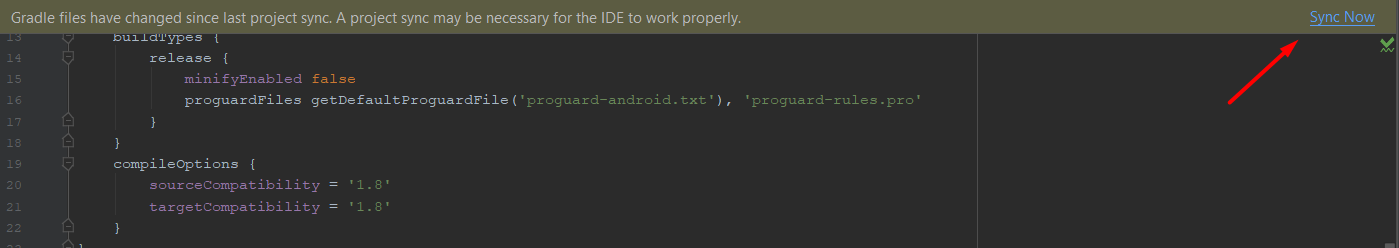
//Squareup - Retrofit

implementation 'com.squareup.retrofit2:retrofit:2.4.0'

implementation 'com.squareup.retrofit2:converter-gson:2.4.0'

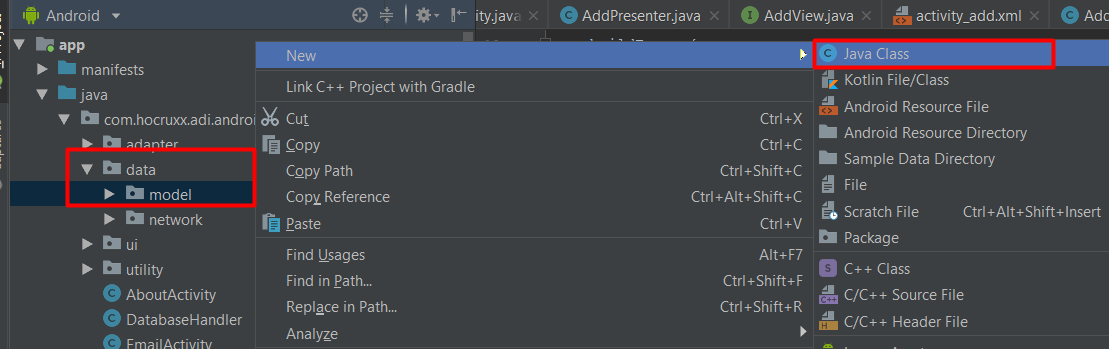
Retrofit merupakan library yang akan digunakan untuk mengubah API yang diperoleh dari Refstull Server (HTTP) ke dalam bentuk data yang dapat diolah oleh Java

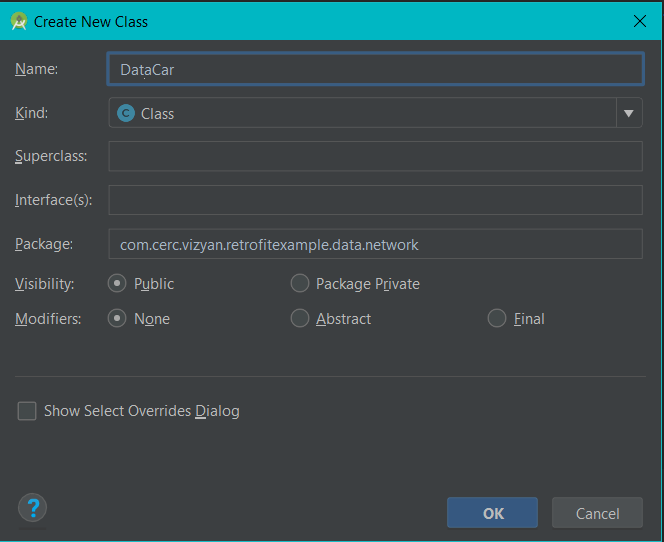
1. Setelahnya, klik tombol **Sync Now** dibagian kanan atas



Tunggu hingga **download** dan **build.gradle** selesai

1. Setelah package dan dependencies, selanjutnya mempersiapkan Retrofit client. Buat class model pada package **data - model** dengan nama **DataCar**



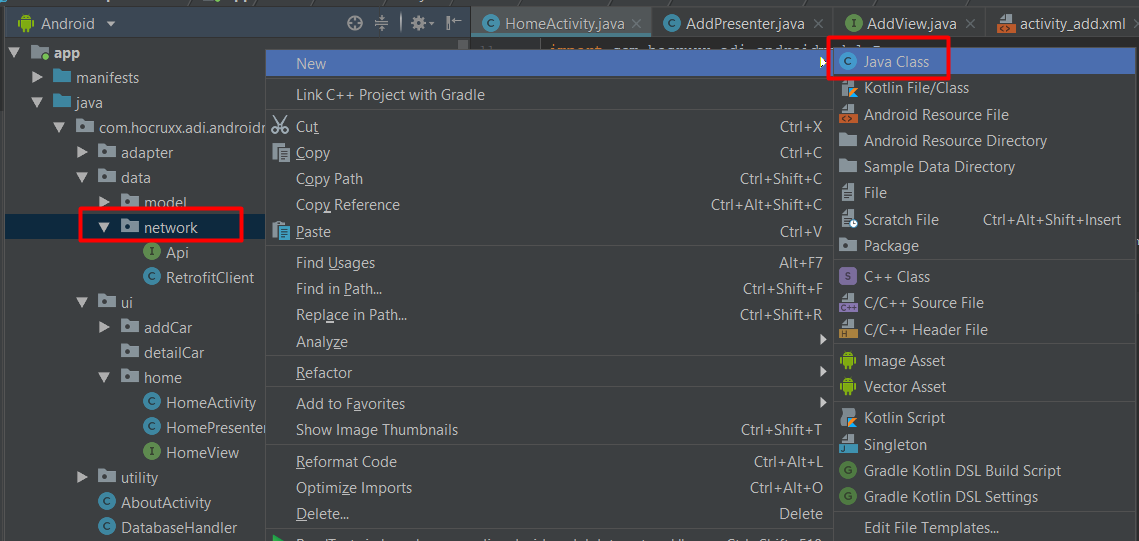


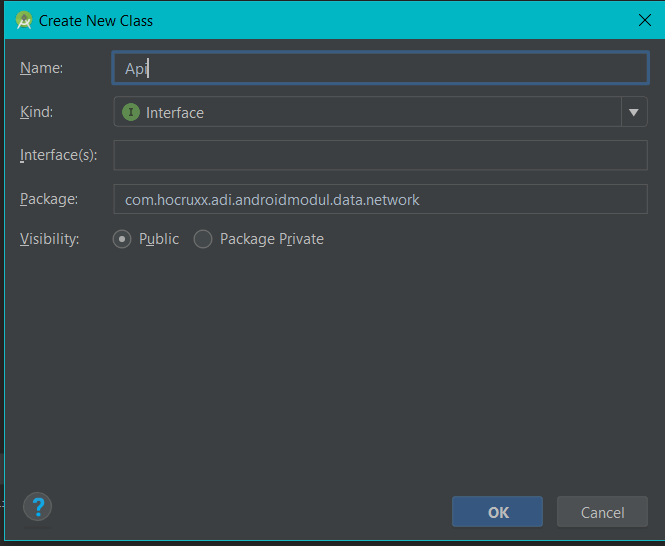
Pada model ini, akan menggunakan POJO (Plain Old Java Object) yaitu class yang berdiri sendiri dan tidak ada ketergantungan dengan class lain. Pada class DataCar tambahkan implements Parcalable untuk membungkus data sehingga dapat dikirimkan/dipindahkan antar activity.

Code ada pada DataCar.txt

**Replace** seluruh code yang ada di dalam class tersebut kecuali **package com.xxxxx.xxxxxx.xxxx** yang ada pada baris teratas dengan code yang ada pada **DataCar.txt**

1. Buat **interface** pada package **data - network** dengan nama **Api** sama seperti langkah sebelumnya, namun pada window box, pilih **interface** lalu beri nama **Api**





Interface ini digunakan untuk menginisialisasi key pada saat POST, GET, DELETE dan lain-lain ke Endpoint Restfull API. Pada praktikum ini akan menggunakan POST dan GET (all).

Code ada pada Api.txt

**Replace** seluruh code yang ada di dalam class tersebut kecuali code **package** **com.xxxxx.xxxxxx.xxxx** yang terletak dibaris teratas dengan code yang ada pada **Api.txt**

1. Buat class baru pada package **data – network** dengan nama **RetrofitClient**, yang berfungsi untuk inisialisasi library retrofit.

Code ada pada RetrofitClient.txt

**Replace** code yang ada di dalam class tersebut kecuali code **package** **com.xxxxx.xxxxxx.xxxx** yang terletak dibaris teratas dengan code yang ada pada **RetrofitClient.txt**

1. Buat class pada package **utility** dengan nama **Constant**, yang akan berisikan variable - variable konstan atau tetap. **Tambahkan** (bukan replace) code berikut kedalam class **Constant**

public static final String BASE\_URL = "https://vizyan.000webhostapp.com/";

public final class Extra {

public static final String DATA = "EXTRA\_DATA";

}

* + 1. **Pembuatan Activity**

Selanjutnya, membuat activity – activity yang diperlukan pada aplikasi. Dengan konsep MVP, setiap activity nantinya terdiri dari 3 class. 1 class activity, 1 class presenter dan 1 class interface (view).

Activity berfungsi untuk menghubungkan System Aplikasi dengan User

Presenter berfungsi untuk menghubungkan Model dengan Retrofil Client

Interface (View) berfungsi untuk menghubungkan Activity dengan Presenter

1. Membuat home
2. Pertama, buatlah presenter pada package **ui -** **home** dengan cara yang sama seperti membuat class biasa. Beri nama **HomePresenter.**

Code ada pada HomePresenter.txt

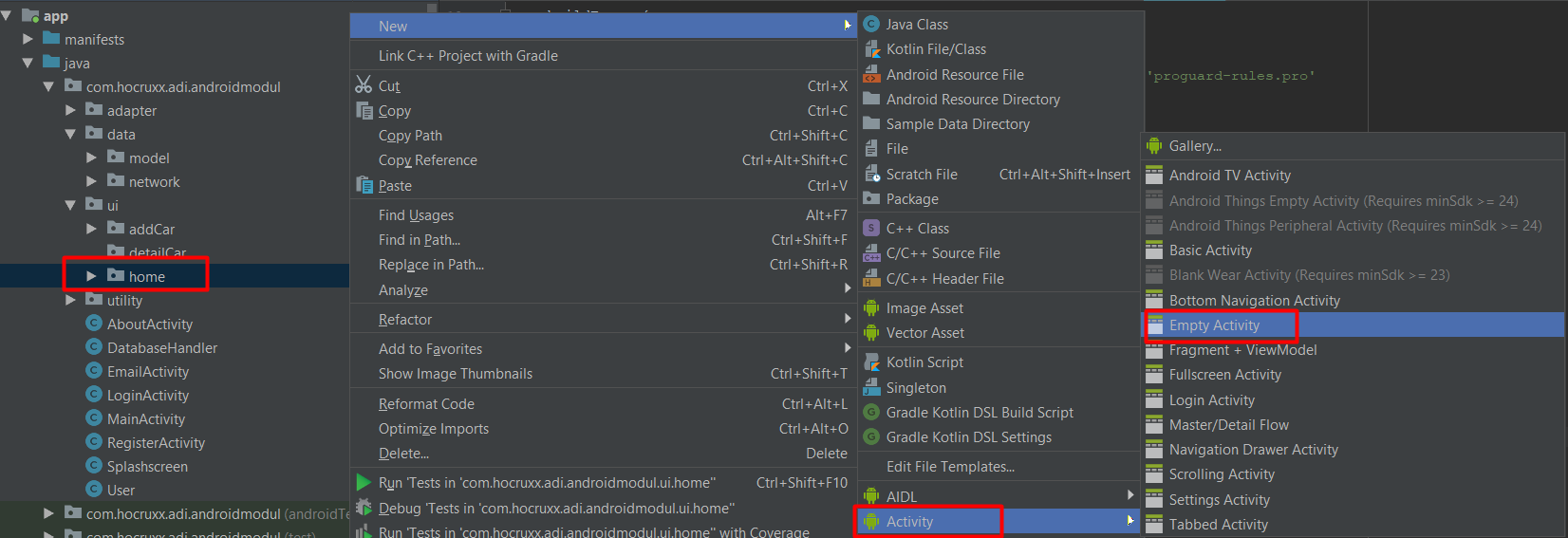
Sama seperti sebelumnya, **replace** seluruh code didalamnya kecuali **pakcage com.xxxxx.xxx.xxxx** yang ada pada baris pertama di class tersebut

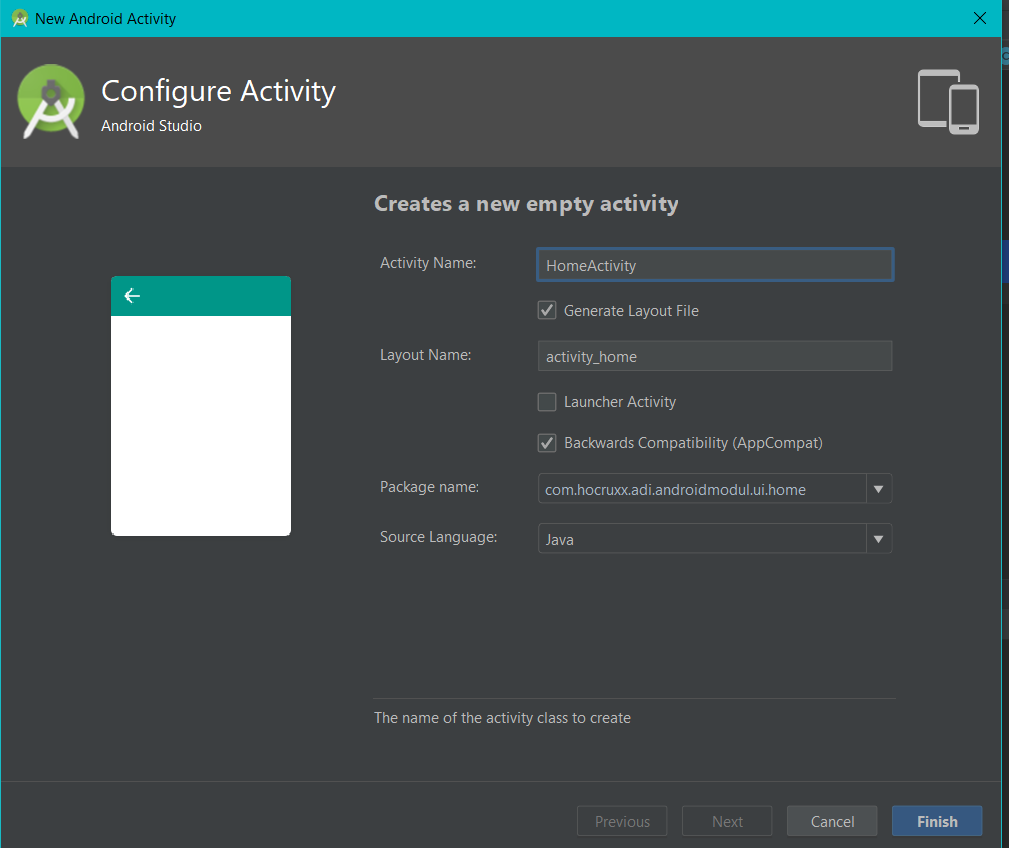
1. Setelah itu buat **interface** pada package **ui -** **home** dengan nama **HomeView**. Lalu **tambahkan** (bukan replace) code berikut didalam classnya

void successShowCar(List<DataCar> dataCars);

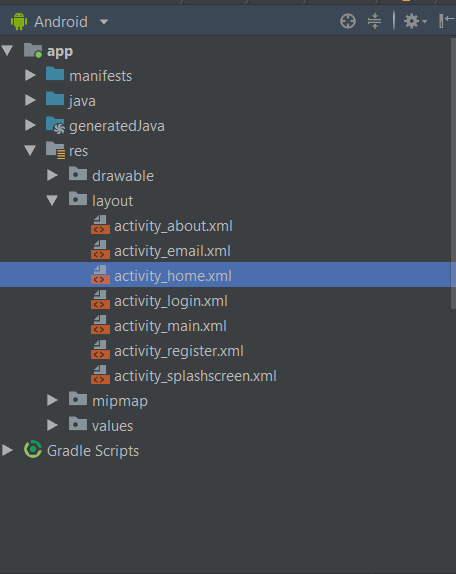
void failedShowCar(String message);

1. Selanjutnya, buat activity dengan cara klik kanan package **home** **– New – Activity – Empty Activity**. Beri nama **HomeActivity**. Lalu klik finish

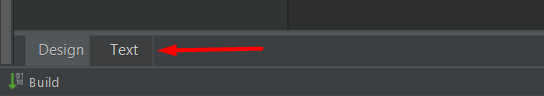




1. Sebelum ke code activity, design terlebih dahulu tampilan pada layout activity nya. Klik package **res – layout – activity\_home.xml**



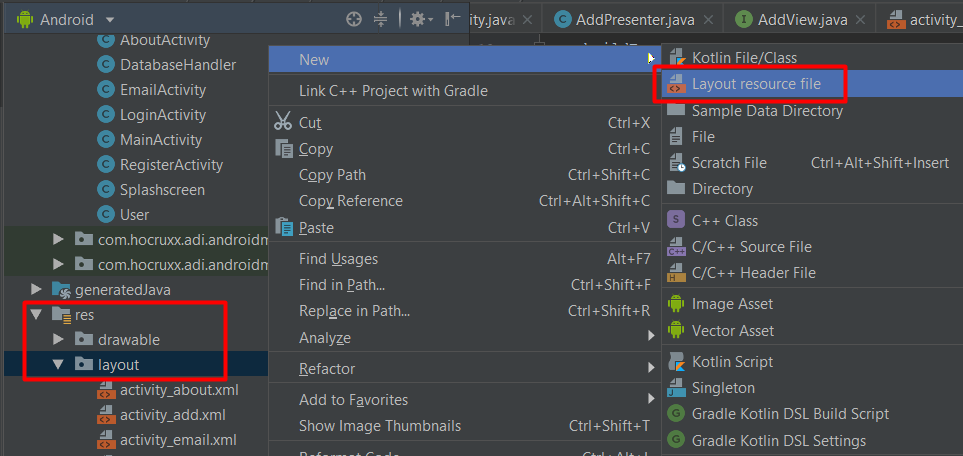
1. Pilih mode text, lalu tambahkan code berikut kedalamnya



Code ada pada activity\_home.txt

**Replace** seluruh code yang ada pada activity\_home.xml dengan code di file activity\_home.txt

1. Buat layout item, dengan cara klik kanan pada **res – layout – Layout Resource File**. Beri nama **item\_car**



Code ada pada item\_car.txt

**Replace** seluruh code yang ada pada item\_car.xml dengan code di file item\_car.txt

1. Selanjutnya tambahkan code pada HomeActivity. Dengan cara dobel click pada class HomeActivity

Code ada pada HomeActivity.txt

**Replace** code yang ada di dalam class tersebut kecuali code **package** **com.xxxxx.xxxxxx.xxxx** yang terletak dibaris teratas dengan code yang ada pada **HomeActivity.txt**

1. Setelahnya, buat adapter untuk menampung list/daftar data API yang didapatkan, dalam praktikum ini yaitu daftar API Car. Buat class baru pada package **adapter – car** dengan nama **CarAdapter.**

Code ada pada CarAdapter.txt

**Replace** code yang ada di dalam class tersebut kecuali code **package** **com.xxxxx.xxxxxx.xxxx** yang terletak dibaris teratas dengan code yang ada pada **CarAdapter.txt**

1. Buat **interface** di package yang sama dengan nama CarListener. **Tambahkan** (bukan replace) code berikut didalam classnya

void onCarClick(DataCar dataCar);

1. Buat class di package yang sama dengan nama CarViewHolder

Code ada pada CarViewHolder.txt

**Replace** code yang ada di dalam class tersebut kecuali code **package** **com.xxxxx.xxxxxx.xxxx** yang terletak dibaris teratas dengan code yang ada pada **CarViewHolder.txt**

1. Membuat addCar

Sama seperti pada home, tambahkan presenter, interface (view) dan activity dengan nama masing – masing. AddPresenter, AddView dan AddActivity. Tambahkan didalam **package** **addCar** lalu masing – masing replace code yang ada didalamnya dengan code yang ada pada file.txt dibawah

1. AddPresenter

Code ada pada AddPresenter.txt

1. AddView

Code ada pada AddView.txt

1. activity\_add.xml

Code ada pada activity\_add.txt

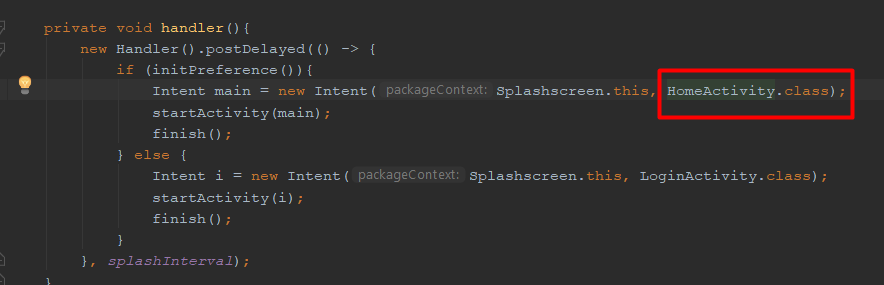
1. AddActivity

Code ada pada AddActivity.txt

1. Pada LoginActivity, ganti bagian intent yang mengarah ke MainActivity menjadi HomeActivity. Seperti pada gambar

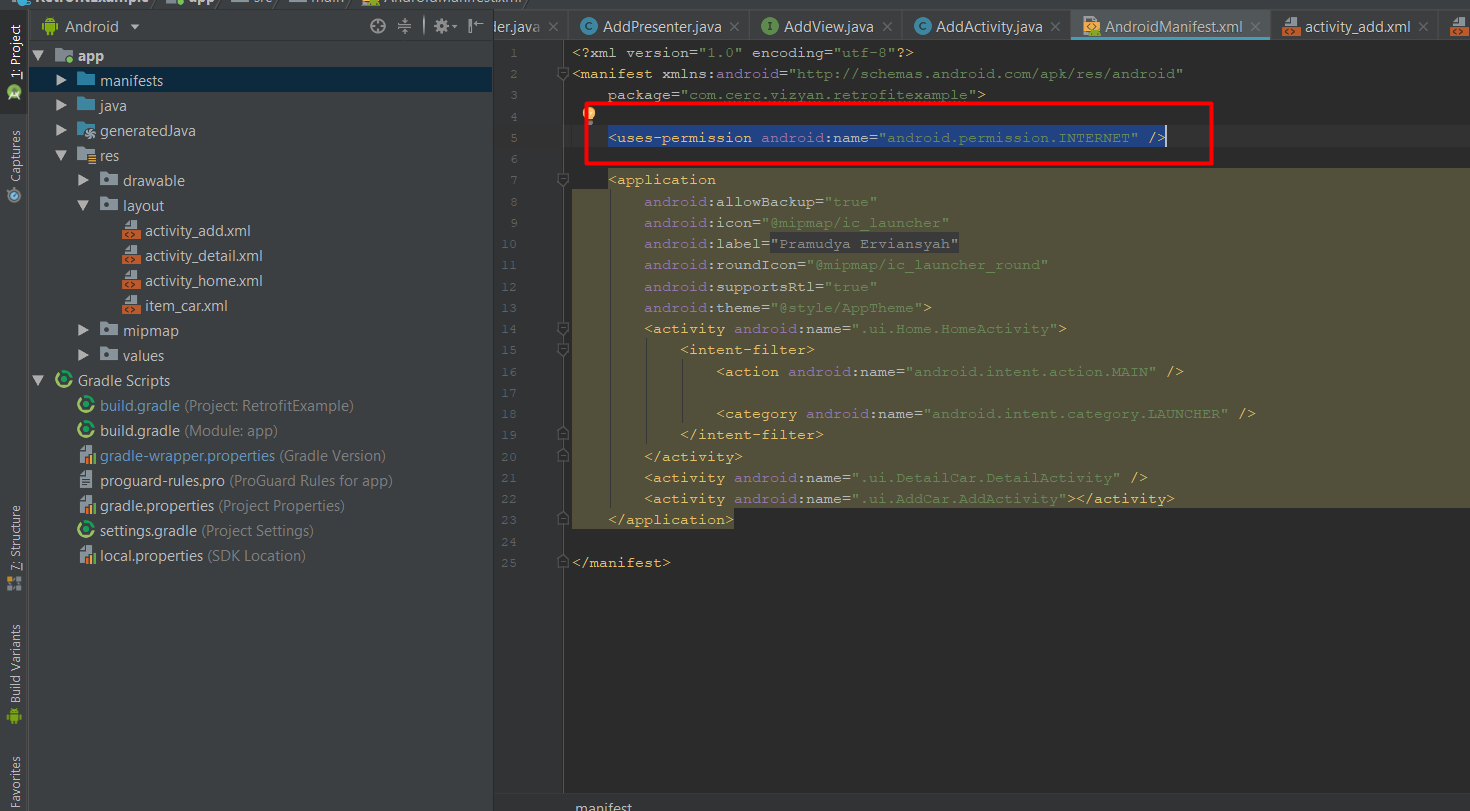


1. Ganti pula code pada activity Splashscreen, seperti gambar dibawah

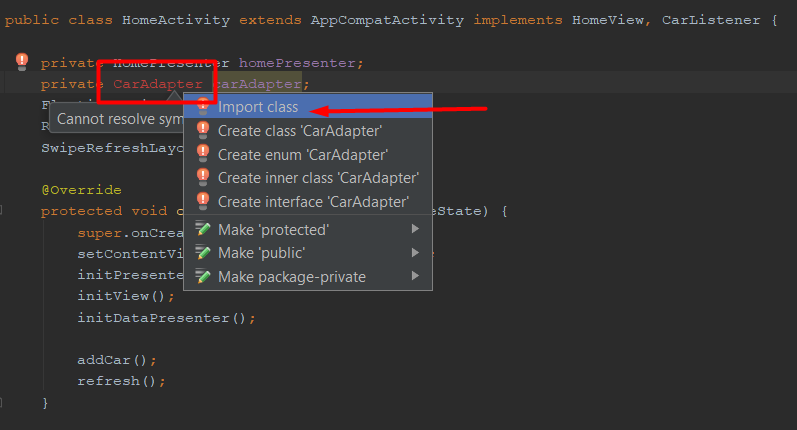


1. Terakhir tambahkan permission pada manifest untuk akses internet dengan cara klik file **Manifest** seperti pada gambar berikut, lalu tambahkan

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



1. Cek semua class yang ada, apabila terdapat code berwarna merah. Arahkan kursor ke code tersebut lalu tekan tombol kombinasi pada keyboard **alt + enter**. Lalu pilih **import class**



1. Deploy aplikasi ke emulator android atau ke smartphone. Jangan lupa, aktifkan terlebih dahulu **debugging mode** pada smartphone masing – masing melalui developer mode
2. Teliti lagi langkah kerja bila masih gagal, ingat **case-sensitive**