

# **LAPORAN PRAKTIKUM**

## **PEMROGRAMAN BERBASIS MOBILE**

### **PERTEMUAN KE-8**



**Disusun Oleh :**

**NAMA** : Raden Isnawan Argi Aryasatya  
**NIM** : 195410257  
**JURUSAN** : Informatika  
**JENJANG** : S1  
**KELAS** : 5

**Laboratorium Terpadu**  
**Sekolah Tinggi Manajemen Informatika Komputer**  
**AKAKOM**  
**YOGYAKARTA**

**2021**

## **PERTEMUAN KE-8** **(VIEWMODEL DAN VIEWMODELPROVIDER)**

### **TUJUAN**

Mahasiswa mampu menggunakan ViewModel dan ViewModelPrivider dalam aplikasi untuk menyimpan dan mengelola data terkait UI

### **DASAR TEORI**

#### **1. Selamat datang**

Anda menggunakan kelas ViewModel untuk menyimpan dan mengelola data terkait UI dengan cara yang sadar siklus. Kelas ViewModel memungkinkan data bertahan dari perubahan konfigurasi perangkat seperti rotasi layar dan perubahan pada ketersediaan keyboard. Anda menggunakan kelas ViewModelFactory untuk membuat instance dan mengembalikan objek ViewModel yang bertahan dari perubahan konfigurasi.

#### **Apa yang akan Anda pelajari**

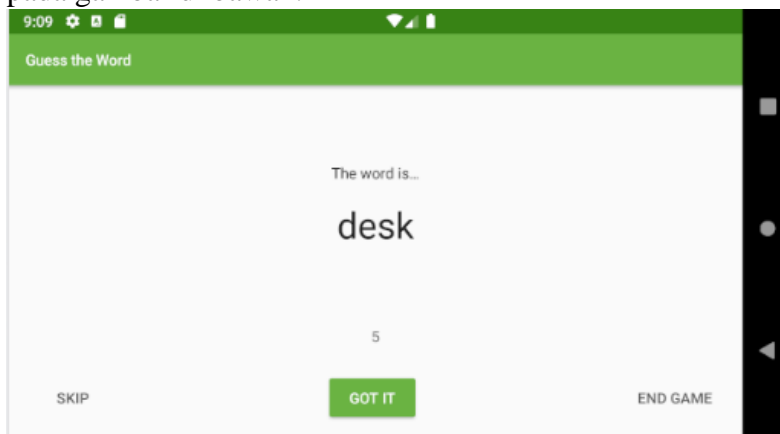
- Cara menggunakan arsitektur aplikasi Android yang direkomendasikan.
- Cara menggunakan kelas Lifecycle, ViewModel, dan ViewModelFactory di aplikasi Anda.
- Cara mempertahankan data UI melalui perubahan konfigurasi perangkat.
- Apa pola desain metode pabrik dan bagaimana menggunakannya.
- Cara membuat objek ViewModel menggunakan antarmuka ViewModelProvider.Factory.

#### **Apa yang akan Anda lakukan**

- Tambahkan ViewModel ke aplikasi, untuk menyimpan data aplikasi sehingga data bertahan dari perubahan konfigurasi.
- Gunakan ViewModelFactory dan pola desain metode pabrik untuk membuat instance objek ViewModel dengan parameter konstruktor.

#### **2. Ikhtisar Aplikasi**

Dalam codelab Pelajaran 5 ini, Anda mengembangkan aplikasi GuessTheWord, dimulai dengan kode permulaan. GuessTheWord adalah permainan gaya sandiwara dua pemain, di mana para pemain berkolaborasi untuk mencapai skor setinggi mungkin. Pemain pertama melihat kata-kata di aplikasi dan memerankannya secara bergantian, pastikan untuk tidak menampilkan kata tersebut ke pemain kedua. Pemain kedua mencoba menebak kata tersebut. Untuk memainkan game, pemain pertama membuka aplikasi di perangkat dan melihat sebuah kata, misalnya "desk", seperti yang ditunjukkan pada gambar di bawah.



Pemain pertama memerankan kata, berhati-hatilah agar tidak benar-benar mengucapkan kata itu sendiri.

- Saat pemain kedua menebak kata dengan benar, pemain pertama menekan tombol Got It, yang menambah hitungan satu dan menampilkan kata berikutnya.
- Jika pemain kedua tidak dapat menebak kata, pemain pertama menekan tombol Skip, yang akan mengurangi hitungan satu kali dan melompat ke kata berikutnya.
- Untuk mengakhiri permainan, tekan tombol End Game. (Fungsi ini tidak ada dalam kode awal untuk codelab pertama dalam seri ini.)

---

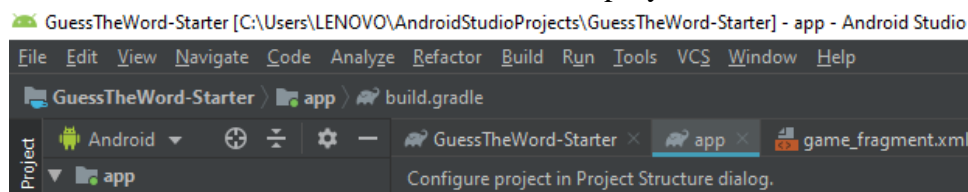
## PRAKTIK

### 3. Tugas: Jelajahi kode starter

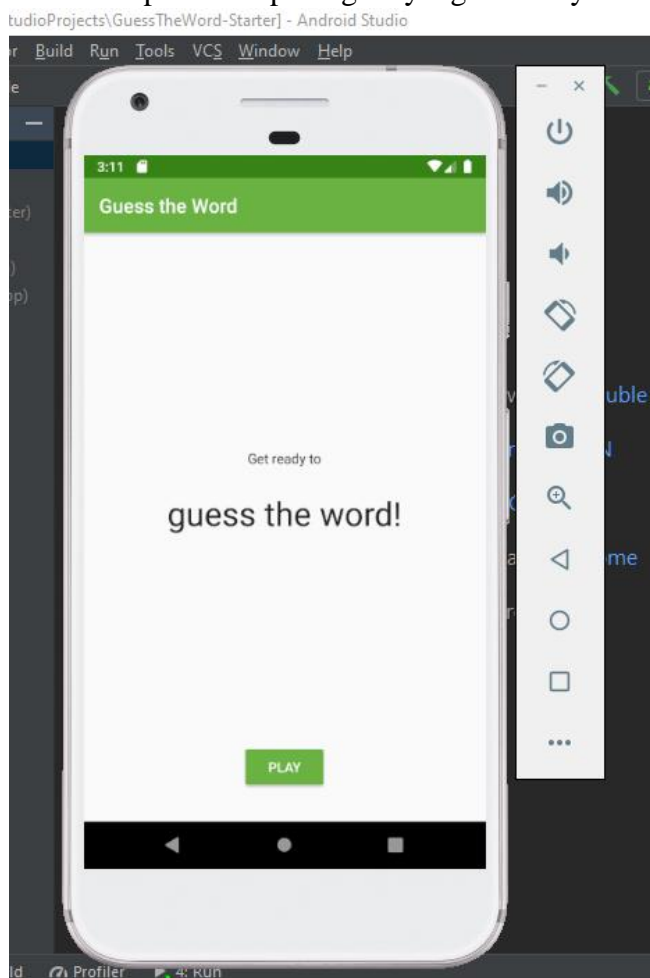
Dalam tugas ini, Anda mendownload dan menjalankan aplikasi starter dan memeriksa kodenya.

#### Langkah 1: Memulai

1. Unduh kode GuessTheWord-Starter dan buka proyek di Android Studio.

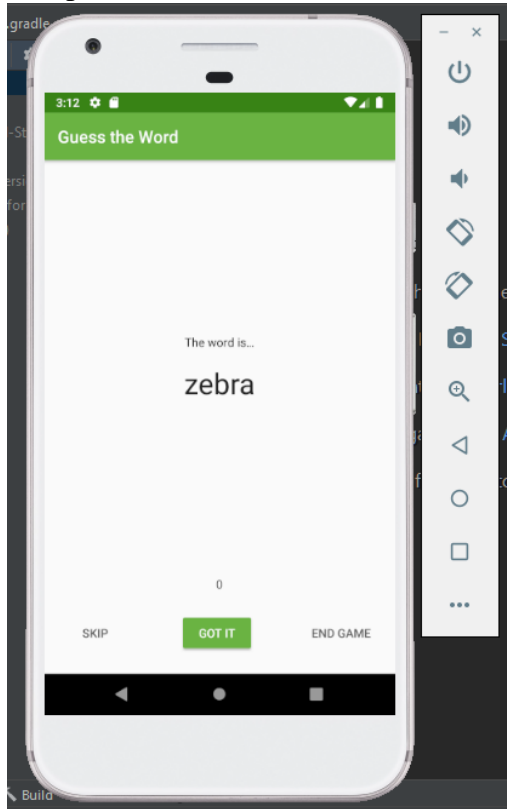


2. Jalankan aplikasi di perangkat yang diberdayakan Android, atau di emulator.

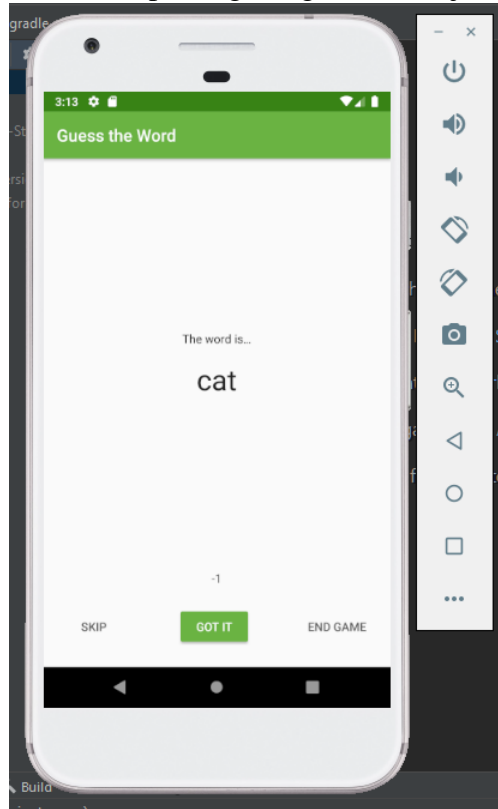


3. Ketuk tombolnya. Perhatikan bahwa tombol Skip menampilkan kata berikutnya dan mengurangi skor satu per satu, dan tombol Got It menampilkan kata berikutnya dan meningkatkan skor satu per satu. Tombol End Game tidak diterapkan, jadi tidak ada yang terjadi saat Anda mengetuknya.

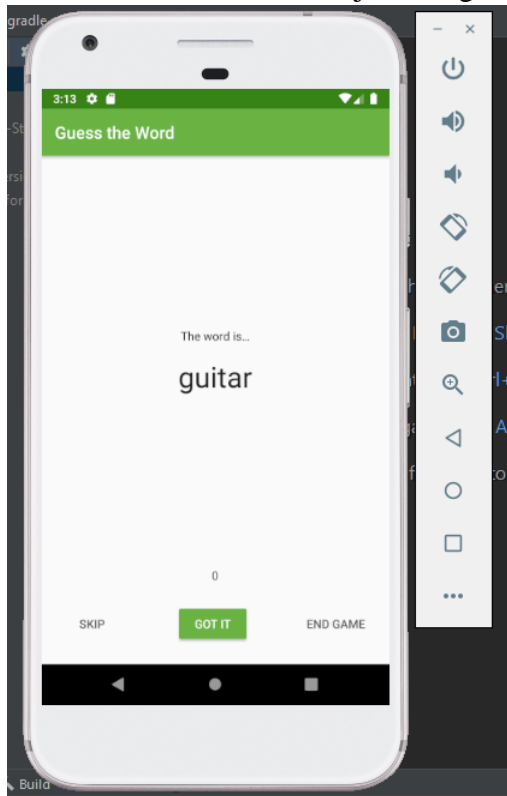
Tampilan awal, skor 0



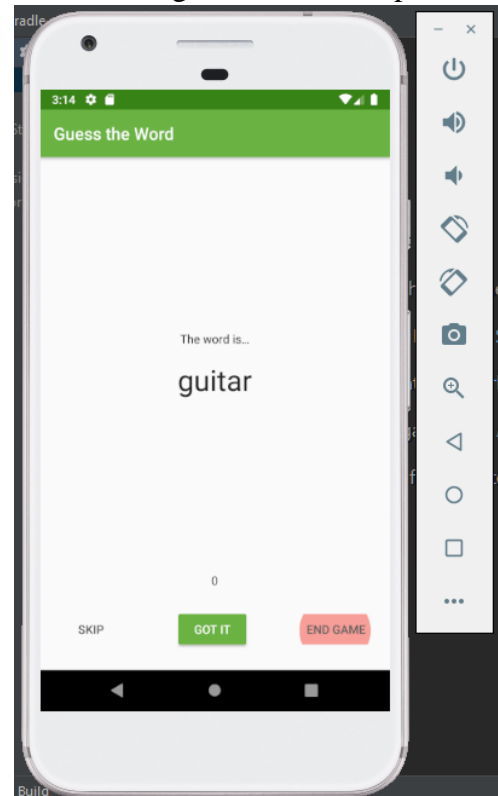
tombol skip mengurangi skor menjadi -1



Got it menambah skor menjadi 0 lagi



tombol end game tidak diterapkan

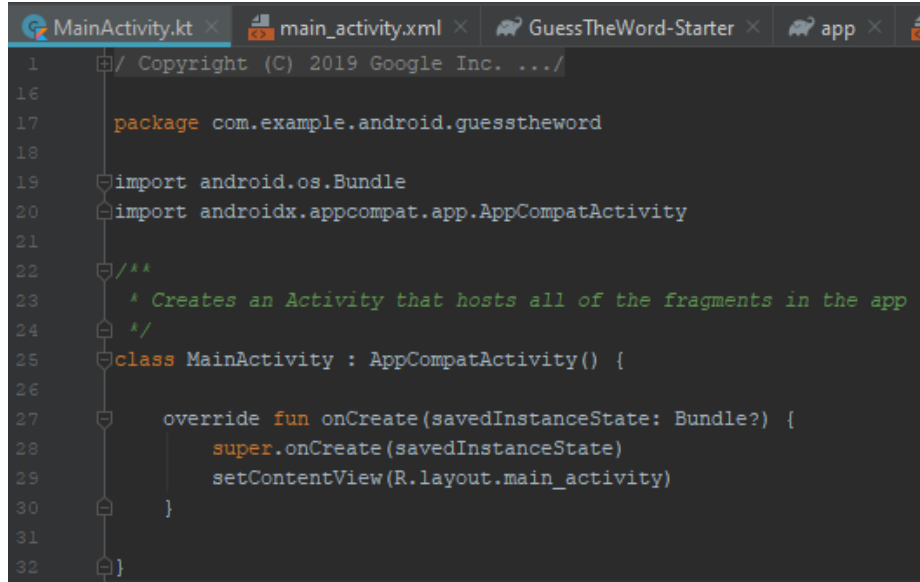


## Langkah 2: Lakukan panduan kode

1. Di Android Studio, jelajahi kode untuk merasakan cara kerja aplikasi.
2. Pastikan untuk melihat file yang dijelaskan di bawah ini, yang sangat penting.

### MainActivity.kt

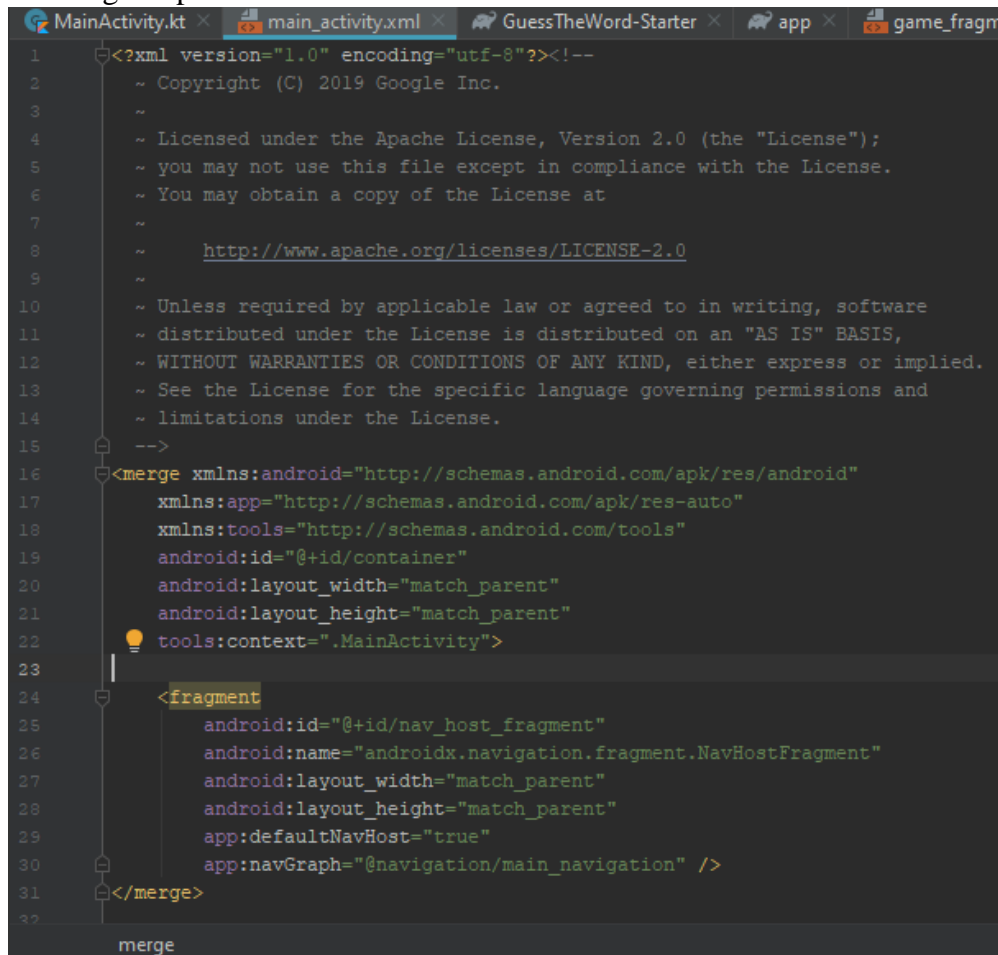
File ini hanya berisi kode default yang dihasilkan template.



```
1  // Copyright (C) 2019 Google Inc. .../
16
17  package com.example.android.guesstheword
18
19  import android.os.Bundle
20  import androidx.appcompat.app.AppCompatActivity
21
22  /**
23   * Creates an Activity that hosts all of the fragments in the app
24   */
25  class MainActivity : AppCompatActivity() {
26
27      override fun onCreate(savedInstanceState: Bundle?) {
28          super.onCreate(savedInstanceState)
29          setContentView(R.layout.main_activity)
30      }
31
32  }
```

### res / layout / main\_activity.xml

File ini berisi tata letak utama aplikasi. NavHostFragment menghosting fragmen lain saat pengguna menavigasi aplikasi.



```
1  <?xml version="1.0" encoding="utf-8"?><!--
2      ~ Copyright (C) 2019 Google Inc.
3      ~
4      ~ Licensed under the Apache License, Version 2.0 (the "License");
5      ~ you may not use this file except in compliance with the License.
6      ~ You may obtain a copy of the License at
7      ~
8      ~     http://www.apache.org/licenses/LICENSE-2.0
9      ~
10     ~ Unless required by applicable law or agreed to in writing, software
11     ~ distributed under the License is distributed on an "AS IS" BASIS,
12     ~ WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13     ~ See the License for the specific language governing permissions and
14     ~ limitations under the License.
15     -->
16  <merge xmlns:android="http://schemas.android.com/apk/res/android"
17        xmlns:app="http://schemas.android.com/apk/res-auto"
18        xmlns:tools="http://schemas.android.com/tools"
19        android:id="@+id/container"
20        android:layout_width="match_parent"
21        android:layout_height="match_parent"
22        tools:context=".MainActivity">
23
24      <fragment
25          android:id="@+id/nav_host_fragment"
26          android:name="androidx.navigation.fragment.NavHostFragment"
27          android:layout_width="match_parent"
28          android:layout_height="match_parent"
29          app:defaultNavHost="true"
30          app:navGraph="@navigation/main_navigation" />
31  </merge>
32
merge
```

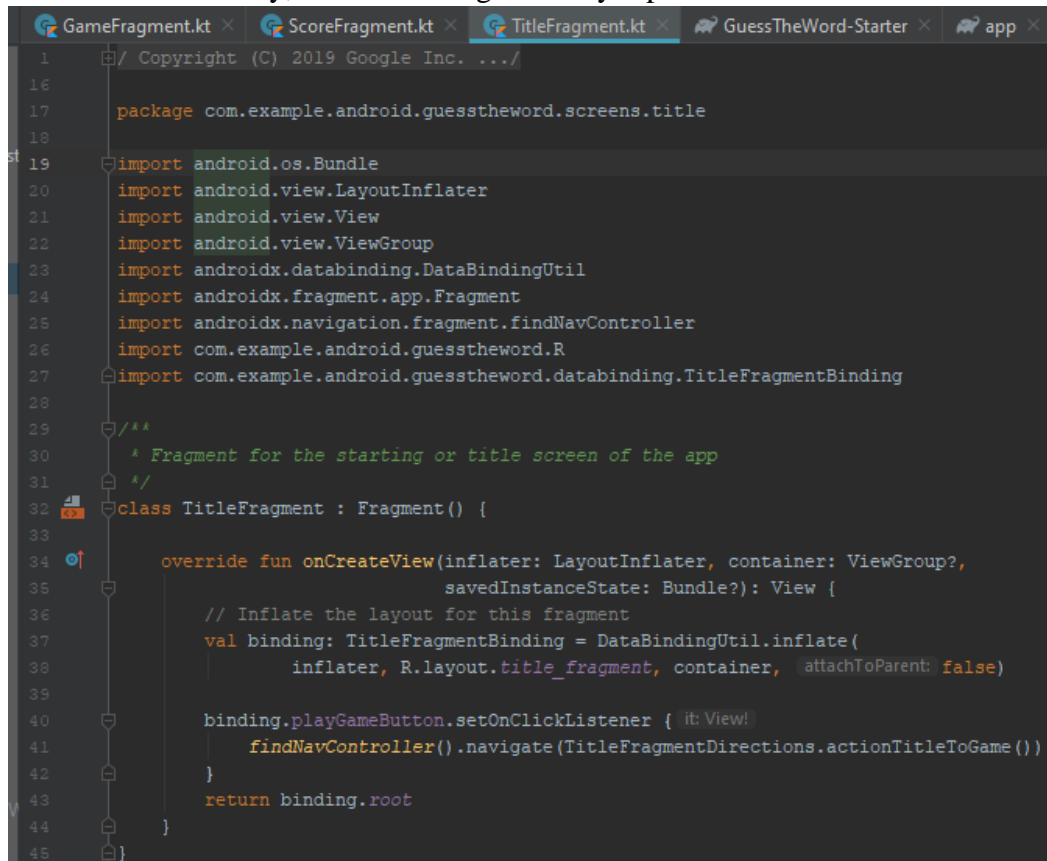
## Fragmen UI

Kode awal memiliki tiga fragmen dalam tiga paket berbeda di bawah paket `com.example.android.guesstheword.screens`:

- **title/TitleFragment** untuk layar judul
- **game/GameFragment** untuk layar judul
- **score/ScoreFragment** untuk layar judul

### screens/title/TitleFragment.kt

Fragmen judul adalah layar pertama yang ditampilkan saat aplikasi diluncurkan. Penangan klik diatur ke tombol Play, untuk menavigasi ke layar permainan.

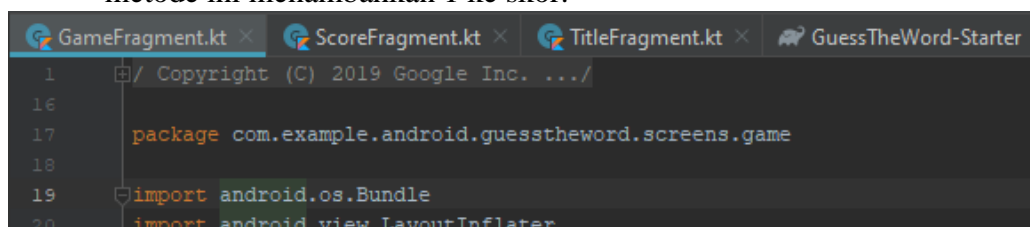


```
1  // Copyright (C) 2019 Google Inc. .../
16
17  package com.example.android.guesstheword.screens.title
18
19  import android.os.Bundle
20  import android.view.LayoutInflater
21  import android.view.View
22  import android.view.ViewGroup
23  import androidx.databinding.DataBindingUtil
24  import androidx.fragment.app.Fragment
25  import androidx.navigation.fragment.findNavController
26  import com.example.android.guesstheword.R
27  import com.example.android.guesstheword.databinding.TitleFragmentBinding
28
29  /**
30   * Fragment for the starting or title screen of the app
31   */
32  class TitleFragment : Fragment() {
33
34      override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
35                              savedInstanceState: Bundle?): View {
36          // Inflate the layout for this fragment
37          val binding: TitleFragmentBinding = DataBindingUtil.inflate(
38              inflater, R.layout.title_fragment, container, attachToParent: false)
39
40          binding.playGameButton.setOnClickListener { it: View?
41              findNavController().navigate(TitleFragmentDirections.actionTitleToGame())
42          }
43          return binding.root
44      }
45  }
```

### screens/game/GameFragment.kt

Ini adalah fragmen utama, di mana sebagian besar aksi permainan berlangsung:

- Variabel ditentukan untuk kata saat ini dan skor saat ini.
- WordList yang didefinisikan di dalam metode `resetList()` adalah contoh daftar kata yang akan digunakan dalam permainan.
- Metode `onSkip()` adalah pengendali klik untuk tombol Skip. Ini mengurangi skor sebesar 1, kemudian menampilkan kata berikutnya menggunakan metode `nextWord()`.
- Metode `onCorrect()` adalah pengendali klik untuk tombol Got It. Metode ini diimplementasikan mirip dengan metode `onSkip()`. Satu-satunya perbedaan adalah bahwa metode ini menambahkan 1 ke skor.



```
1  // Copyright (C) 2019 Google Inc. .../
16
17  package com.example.android.guesstheword.screens.game
18
19  import android.os.Bundle
20  import android.view.LayoutInflater
```

```

21 import android.view.View
22 import android.view.ViewGroup
23 import androidx.databinding.DataBindingUtil
24 import androidx.fragment.app.Fragment
25 import com.example.android.guesstheword.R
26 import com.example.android.guesstheword.databinding.GameFragmentBinding
27
28 /**
29  * Fragment where the game is played
30  */
31 class GameFragment : Fragment() {
32
33     // The current word
34     private var word = ""
35
36     // The current score
37     private var score = 0
38
39     // The list of words - the front of the list is the next word to guess
40     private lateinit var wordList: MutableList<String>
41
42     private lateinit var binding: GameFragmentBinding
43
44     override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
45                               savedInstanceState: Bundle?): View? {
46
47         // Inflate view and obtain an instance of the binding class
48         binding = DataBindingUtil.inflate(
49             inflater,
50             R.layout.game_fragment,
51             container,
52             attachToParent: false
53         )
54
55         resetList()
56         nextWord()
57
58         binding.correctButton.setOnClickListener { onCorrect() }
59         binding.skipButton.setOnClickListener { onSkip() }
60         updateScoreText()
61         updateWordText()
62         return binding.root
63     }
64
65     /**
66     * Resets the list of words and randomizes the order
67     */
68     private fun resetList() {
69         wordList = mutableListOf(
70             "queen",
71             "hospital",
72             "basketball",
73             "cat",
74             "change",
75             "snail",
76             "soup",
77             "calendar",
78             "sad",
79             "desk",
80             "guitar",
81             "home",
82             "railway",
83             "zebra",
84             "jelly",

```

```

86         "car",
87         "crow",
88         "trade",
89         "bag",
90         "roll",
91         "bubble"
92     )
93     wordList.shuffle()
94 }
95 /** Methods for buttons presses */
96 private fun onSkip() {
97     score--
98     nextWord()
99 }
100
101 private fun onCorrect() {
102     score++
103     nextWord()
104 }
105
106 /**
107  * Moves to the next word in the list
108  */
109 private fun nextWord() {
110     if (!wordList.isEmpty()) {
111         //Select and remove a word from the list
112         word = wordList.removeAt(index: 0)
113     }
114     updateWordText()
115     updateScoreText()
116 }
117
118 /** Methods for updating the UI */
119 private fun updateWordText() {
120     binding.wordText.text = word
121 }
122
123 private fun updateScoreText() {
124     binding.scoreText.text = score.toString()
125 }
126 }

```

### screens/score/ScoreFragment.kt

ScoreFragment adalah layar terakhir dalam permainan, dan ini menampilkan skor akhir pemain. Dalam codelab ini, Anda menambahkan implementasi untuk menampilkan layar ini dan menunjukkan skor akhir.

```

GameFragment.kt x ScoreFragment.kt x TitleFragment.kt x GuessTheWord-Starter
1  // Copyright (C) 2019 Google Inc. .../
16
17  package com.example.android.guesstheword.screens.score
18
19  import android.os.Bundle
20  import android.view.LayoutInflater
21  import android.view.View
22  import android.view.ViewGroup
23  import androidx.databinding.DataBindingUtil
24  import androidx.fragment.app.Fragment
25  import androidx.navigation.fragment.navArgs
26  import com.example.android.guesstheword.R
27  import com.example.android.guesstheword.databinding.ScoreFragmentBinding
28  /**
29   * Fragment where the final score is shown, after the game is over

```



```

30  */
31  class ScoreFragment : Fragment() {
32      override fun onCreateView(
33          inflater: LayoutInflater,
34          container: ViewGroup?,
35          savedInstanceState: Bundle?
36      ): View? {
37          // Inflate view and obtain an instance of the binding class.
38          val binding: ScoreFragmentBinding = DataBindingUtil.inflate(
39              inflater,
40              R.layout.score_fragment,
41              container,
42              attachToParent: false
43          )
44          return binding.root
45      }
46  }

```

### res/navigation/main\_navigation.xml

Grafik navigasi menunjukkan bagaimana fragmen terhubung melalui navigasi:

- Dari fragmen judul, pengguna dapat membuka fragmen game.
- Dari fragmen game, pengguna dapat membuka fragmen skor.
- Dari fragmen skor, pengguna dapat menavigasi kembali ke fragmen game.

```

main_navigation.xml x GameViewModel.kt x GameFragment.kt x ScoreFragment.kt x Score
1  <?xml version="1.0" encoding="utf-8"?><!--
2      ~ Copyright (C) 2019 Google Inc.
3      ~
4      ~ Licensed under the Apache License, Version 2.0 (the "License");
5      ~ you may not use this file except in compliance with the License.
6      ~ You may obtain a copy of the License at
7      ~
8      ~ http://www.apache.org/licenses/LICENSE-2.0
9      ~
10     ~ Unless required by applicable law or agreed to in writing, software
11     ~ distributed under the License is distributed on an "AS IS" BASIS,
12     ~ WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13     ~ See the License for the specific language governing permissions and
14     ~ limitations under the License.
15     -->
16
17     <navigation xmlns:android="http://schemas.android.com/apk/res/android"
18         xmlns:app="http://schemas.android.com/apk/res-auto"
19         xmlns:tools="http://schemas.android.com/tools"
20         app:startDestination="@id/title_destination">
21
22         <fragment
23             android:id="@+id/title_destination"
24             android:name="com.example.android.guessstheword.screens.title.TitleFragment"
25             android:label="title_fragment"
26             tools:layout="@layout/title_fragment">
27             <action
28                 android:id="@+id/action_title_to_game"
29                 app:destination="@id/game_destination"
30                 app:enterAnim="@anim/slide_in_right"
31                 app:exitAnim="@anim/slide_out_left"
32                 app:launchSingleTop="true"
33                 app:popEnterAnim="@anim/slide_in_right"
34                 app:popExitAnim="@anim/slide_out_left" />
35         </fragment>
36         <fragment
37             android:id="@+id/game_destination"
38             android:name="com.example.android.guessstheword.screens.game.GameFragment"
39             android:label="game_fragment"
40             tools:layout="@layout/game_fragment">
41             <action
42                 android:id="@+id/action_game_to_score"

```

```

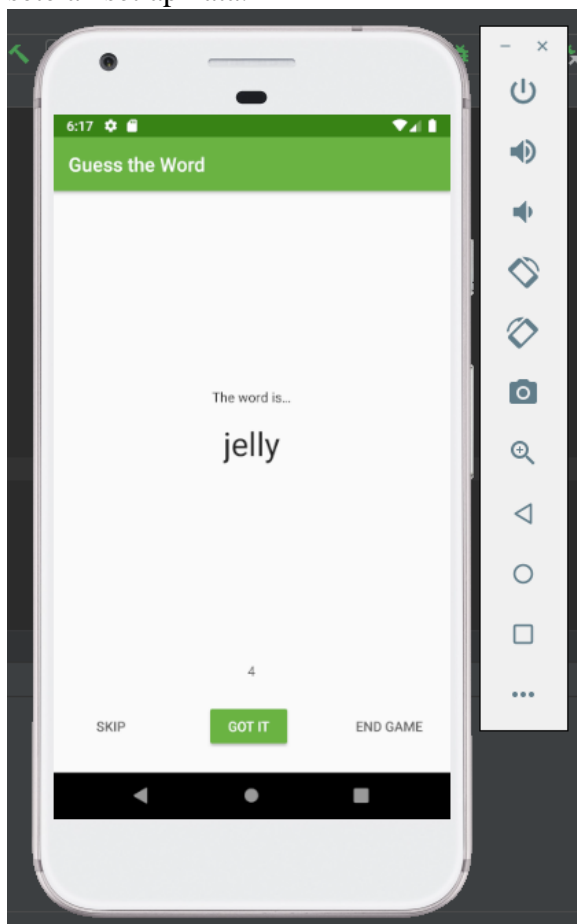
43         app:destination="@id/score_destination"
44         app:enterAnim="@anim/slide_in_right"
45         app:exitAnim="@anim/slide_out_left"
46         app:launchSingleTop="true"
47         app:popEnterAnim="@anim/slide_in_right"
48         app:popExitAnim="@anim/slide_out_left"
49         app:popUpTo="@id/title_destination" />
50     </fragment>
51     <fragment
52         android:id="@+id/score_destination"
53         android:name="com.example.android.guesstheword.screens.score.ScoreFragment"
54         android:label="score_fragment"
55         tools:layout="@layout/score_fragment">
56         <action
57             android:id="@+id/action_restart"
58             app:destination="@+id/game_destination"
59             app:enterAnim="@anim/slide_in_right"
60             app:exitAnim="@anim/slide_out_left"
61             app:popEnterAnim="@anim/slide_in_right"
62             app:popExitAnim="@anim/slide_out_left"
63             app:popUpTo="@id/title_destination" />
64         <argument
65             android:name="score"
66             android:defaultValue="0"
67             app:argType="integer" />
68     </fragment>
69 </navigation>

```

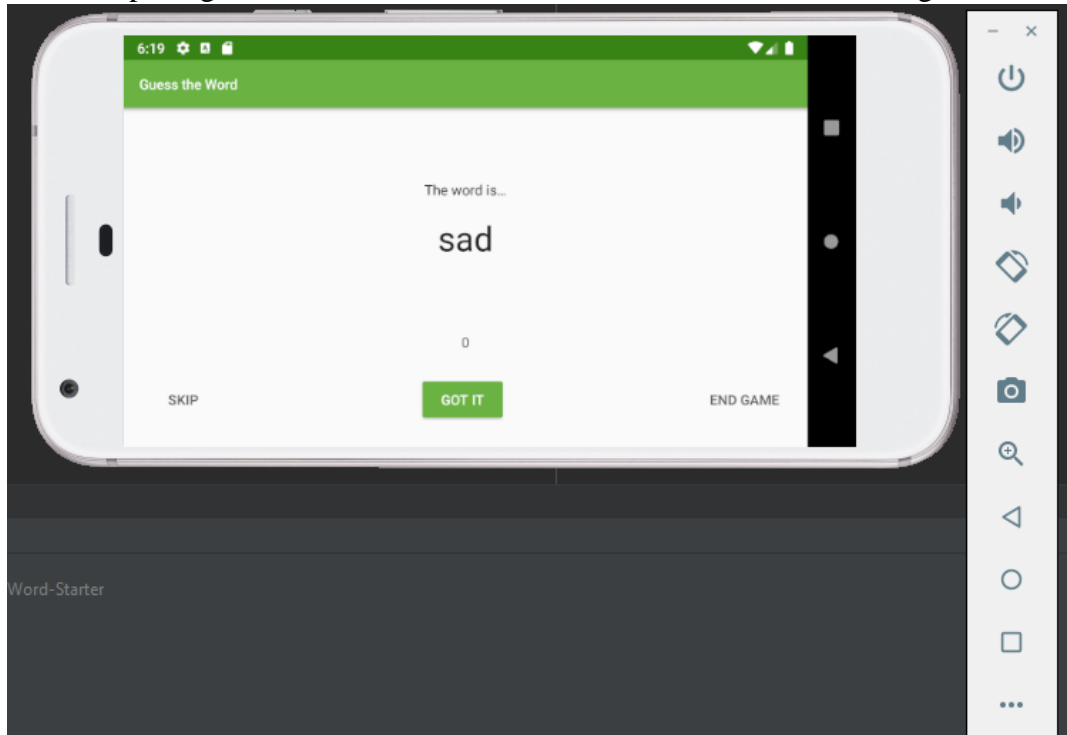
#### 4. Tugas: Temukan masalah di aplikasi starter

Dalam tugas ini, Anda menemukan masalah dengan aplikasi starter GuessTheWord.

1. Jalankan kode awal dan mainkan permainan melalui beberapa kata, ketuk Skip atau Got It setelah setiap kata.

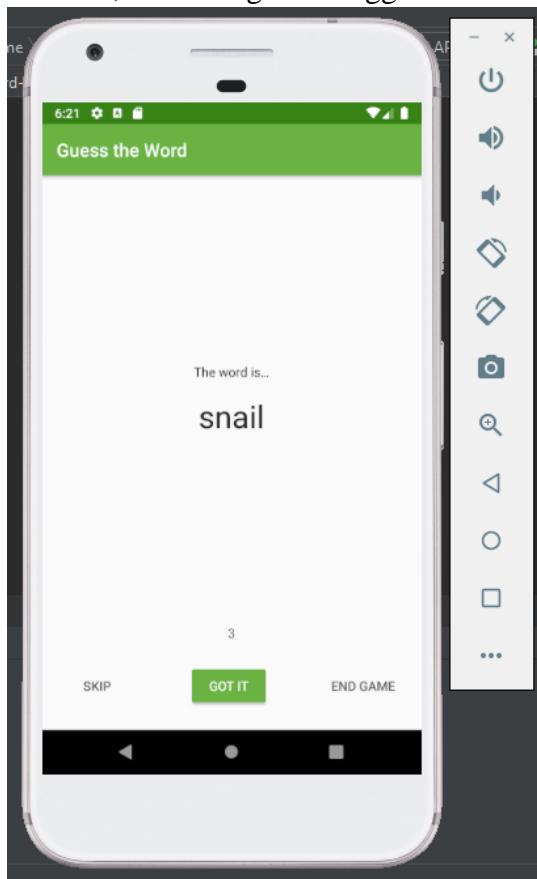


2. Layar permainan sekarang menunjukkan kata dan skor saat ini. Ubah orientasi layar dengan memutar perangkat atau emulator. Perhatikan bahwa skor saat ini hilang.

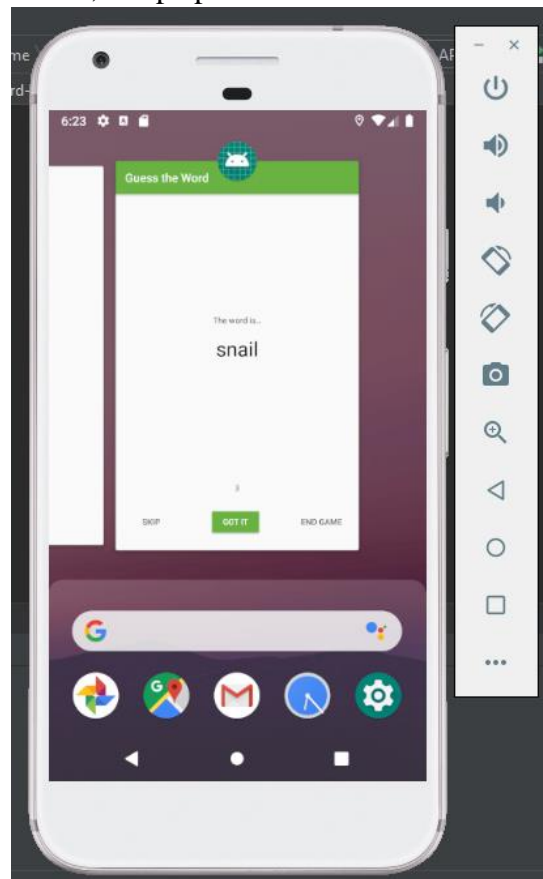


3. Jalankan permainan melalui beberapa kata lagi. Saat layar game ditampilkan dengan beberapa skor, tutup dan buka kembali aplikasi. Perhatikan bahwa permainan dimulai ulang dari awal, karena status aplikasi tidak disimpan.

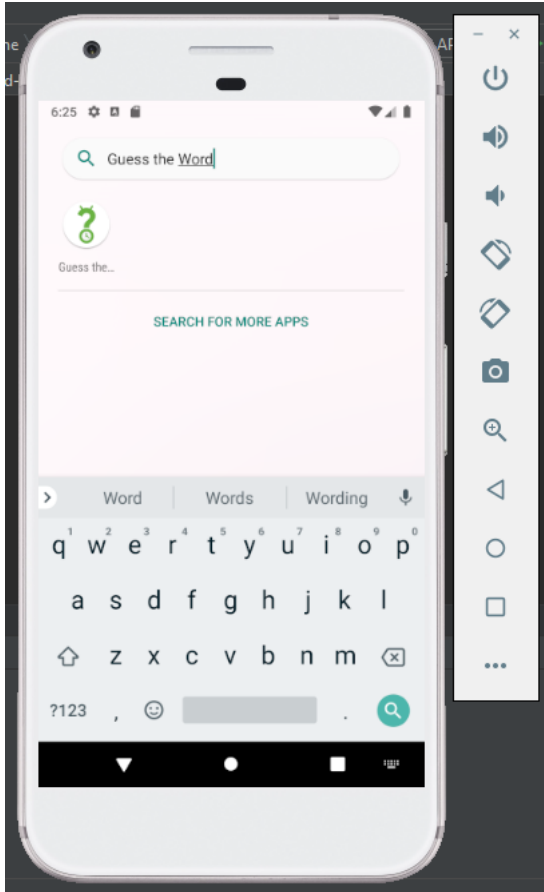
Pertama, mainkan game hingga skor 3



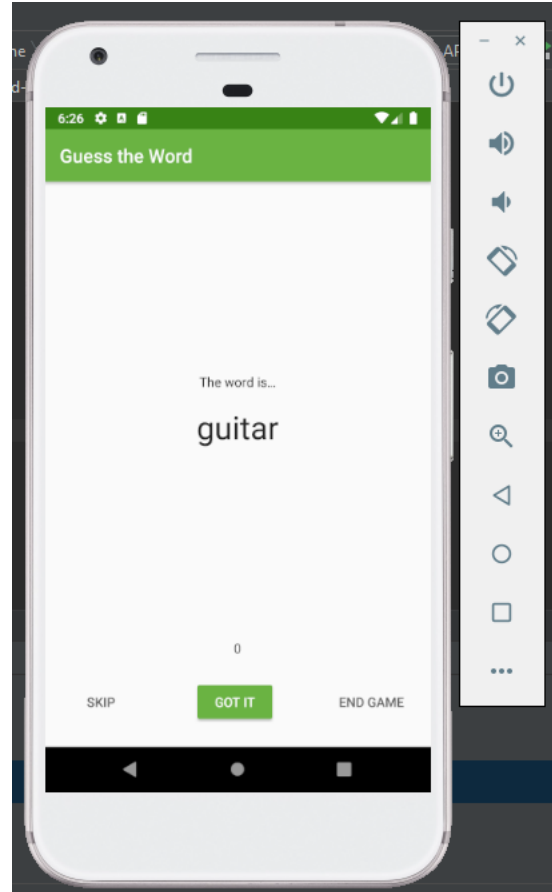
kedua, tutup aplikasi



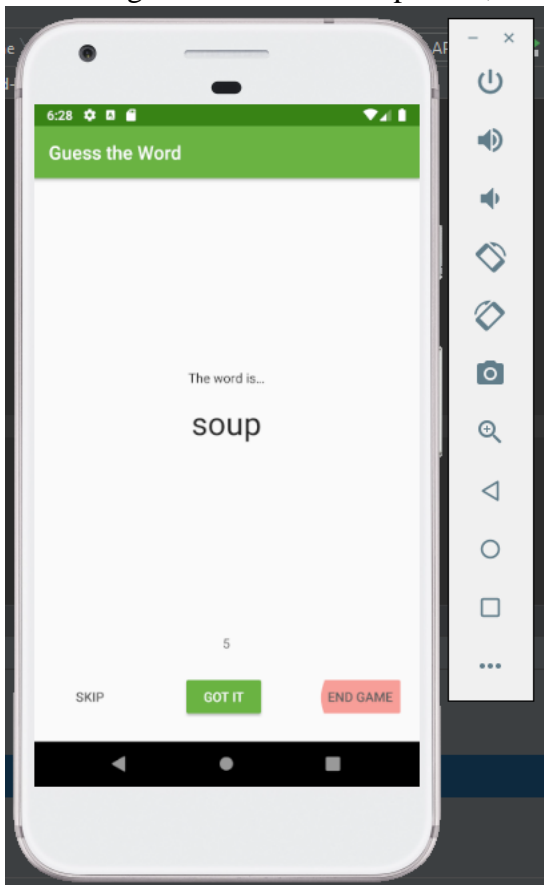
Ketiga, buka kembali aplikasi



keempat, game dimulai dari awal (not saved)



4. Mainkan game melalui beberapa kata, lalu ketuk tombol End Game. Tidak ada yang terjadi.



## Masalah di aplikasi

- Aplikasi pemula tidak menyimpan dan memulihkan status aplikasi selama perubahan konfigurasi, seperti saat orientasi perangkat berubah, atau saat aplikasi dimatikan dan dimulai ulang. Anda bisa mengatasi masalah ini menggunakan callback `onSaveInstanceState()`. Namun, menggunakan metode `onSaveInstanceState()` mengharuskan Anda menulis kode tambahan untuk menyimpan status dalam bundel, dan menerapkan logika untuk mengambil status tersebut. Selain itu, jumlah data yang dapat disimpan minimal.
- Layar game tidak mengarah ke layar skor saat pengguna mengetuk tombol End Game.

## Arsitektur aplikasi

Arsitektur aplikasi adalah cara mendesain kelas aplikasi Anda, dan hubungan di antara mereka, sehingga kode diatur, bekerja dengan baik dalam skenario tertentu, dan mudah digunakan. Dalam kumpulan empat codelab ini, penyempurnaan yang Anda lakukan pada aplikasi `GuessTheWord` mengikuti pedoman arsitektur aplikasi Android, dan Anda menggunakan Komponen Arsitektur Android. Arsitektur aplikasi Android mirip dengan pola arsitektur MVVM (model-view-viewmodel). Aplikasi `GuessTheWord` mengikuti prinsip desain pemisahan masalah dan dibagi menjadi beberapa kelas, dengan setiap kelas menangani masalah yang terpisah. Dalam codelab pertama pelajaran ini, kelas yang Anda gunakan adalah UI controller, `ViewModel`, dan `ViewModelFactory`.

### UI Controller

Pengontrol UI (UI controller) adalah kelas berbasis UI seperti `Aktivitas` atau `Fragmen`. Pengontrol UI hanya boleh berisi logika yang menangani UI dan interaksi sistem operasi seperti menampilkan tampilan dan menangkap input pengguna. Jangan letakkan logika pengambilan keputusan, seperti logika yang menentukan teks yang akan ditampilkan, ke dalam pengontrol UI. Dalam kode awal `GuessTheWord`, pengontrol UI terdiri dari tiga fragmen: `GameFragment`, `ScoreFragment`, dan `TitleFragment`. Mengikuti prinsip desain "pemisahan perhatian", `GameFragment` hanya bertanggung jawab untuk menggambar elemen game ke layar dan mengetahui kapan pengguna mengetuk tombol, dan tidak lebih. Saat pengguna mengetuk tombol, informasi ini diteruskan ke `GameViewModel`.

### ViewModel

`ViewModel` menyimpan data untuk ditampilkan dalam fragmen atau aktivitas yang terkait dengan `ViewModel`. `ViewModel` dapat melakukan penghitungan dan transformasi sederhana pada data untuk menyiapkan data untuk ditampilkan oleh pengontrol UI. Dalam arsitektur ini, `ViewModel` melakukan pengambilan keputusan. `GameViewModel` menyimpan data seperti nilai skor, daftar kata, dan kata saat ini, karena ini adalah data yang akan ditampilkan di layar. `GameViewModel` juga berisi logika bisnis untuk melakukan penghitungan sederhana guna memutuskan bagaimana status data saat ini.

### ViewModelFactory

Sebuah `ViewModelFactory` membuat instance objek `ViewModel`, dengan atau tanpa parameter konstruktor.

---

## 5. Tugas: Membuat GameViewModel

Kelas `ViewModel` dirancang untuk menyimpan dan mengelola data terkait UI. Dalam aplikasi ini, setiap `ViewModel` dikaitkan dengan satu fragmen. Dalam tugas ini, Anda menambahkan `ViewModel` pertama ke aplikasi Anda, `GameViewModel` untuk `GameFragment`. Anda juga mempelajari apa artinya `ViewModel` sadar-siklus.

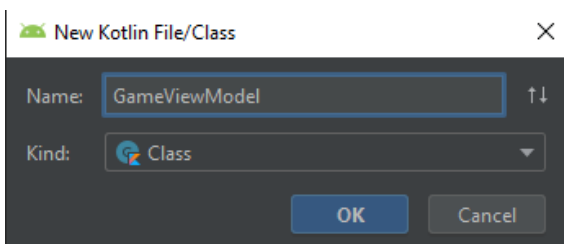
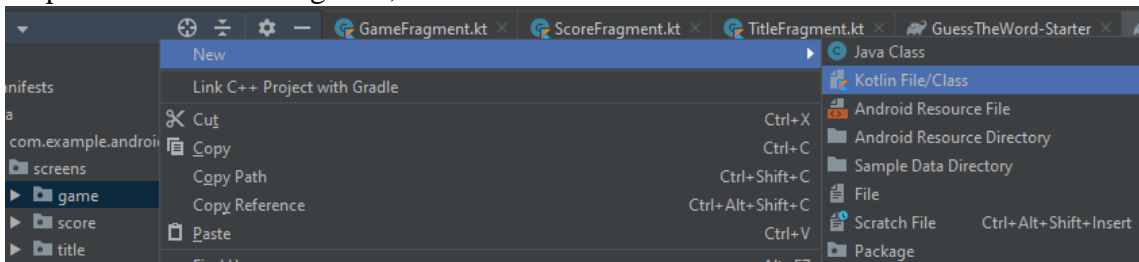
## Langkah 1: Tambahkan kelas GameViewModel

1. Buka file build.gradle (module: app). Di dalam blok dependencies, tambahkan dependensi Gradle untuk ViewModel.

Jika Anda menggunakan versi terbaru pustaka, aplikasi solusi harus dikompilasi seperti yang diharapkan. Jika tidak, coba selesaikan masalah, atau kembalikan ke versi yang ditunjukkan di bawah.

```
60 //ViewModel
61 implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.2.0'
62 }
```

2. Di paket folder screens/game/, buat kelas Kotlin baru bernama GameViewModel.



3. Buat kelas GameViewModel memperluas kelas abstrak ViewModel.

```
6 class GameViewModel : ViewModel() {
```

4. Untuk membantu Anda lebih memahami bagaimana ViewModel sadar siklus hidup, tambahkan blok init dengan pernyataan log.

```
GameViewModel.kt x GameFragment.kt x ScoreFragment.kt x TitleFrame
1 package com.example.android.guessstheword.screens.game
2
3 import android.util.Log
4 import androidx.lifecycle.ViewModel
5
6 class GameViewModel : ViewModel() {
7     init {
8         Log.i( tag: "GameViewModel", msg: "GameViewModel created!")
9     }
10 }
```

## Langkah 2: Ganti onCleared () dan tambahkan logging

ViewModel dimusnahkan saat fragmen terkait dilepas, atau saat aktivitas selesai. Tepat sebelum ViewModel dihancurkan, callback onCleared() dipanggil untuk membersihkan sumber daya.

1. Di kelas GameViewModel, override metode onCleared ().

```
10 override fun onCleared() {
```

2. Tambahkan pernyataan log di dalam onCleared() untuk melacak siklus hidup GameViewModel.

```
10 override fun onCleared() {
11     super.onCleared()
12     Log.i( tag: "GameViewModel", msg: "GameViewModel destroyed!")
13 }
```

### Langkah 3: Kaitkan GameViewModel dengan fragmen game

ViewModel harus dikaitkan dengan pengontrol UI. Untuk mengaitkan keduanya, Anda membuat referensi ke ViewModel di dalam pengontrol UI. Pada langkah ini, Anda membuat referensi GameViewModel di dalam pengontrol UI yang sesuai, yaitu GameFragment.

1. Di kelas GameFragment, tambahkan filed tipe GameViewModel di bagian atas sebagai variabel kelas.

```
44 private lateinit var viewModel: GameViewModel
```

### Langkah 4: Inisialisasi ViewModel

Selama perubahan konfigurasi seperti rotasi layar, pengontrol UI seperti fragmen dibuat ulang. Namun, instance ViewModel tetap bertahan. Jika Anda membuat instance ViewModel menggunakan kelas ViewModel, objek baru dibuat setiap kali fragmen dibuat ulang. Sebagai gantinya, buat instance ViewModel menggunakan ViewModelProvider.

Cara kerja ViewModelProvider:

- ViewModelProvider mengembalikan ViewModel yang ada jika ada, atau membuat yang baru jika belum ada.
- ViewModelProvider membuat instance ViewModel terkait dengan cakupan yang diberikan (aktivitas atau fragmen).
- ViewModel yang dibuat dipertahankan selama cakupannya hidup. Misalnya, jika cakupannya adalah fragmen, ViewModel akan dipertahankan hingga fragmen dilepaskan.

Inisialisasi ViewModel, menggunakan metode ViewModelProvider.get() untuk membuat ViewModelProvider:

1. Di kelas GameFragment, inisialisasi variabel viewModel. Letakkan kode ini di dalam onCreateView(), setelah definisi variabel binding. Gunakan metode ViewModelProvider.get(), dan teruskan dalam konteks GameFragment terkait dan kelas GameViewModel.

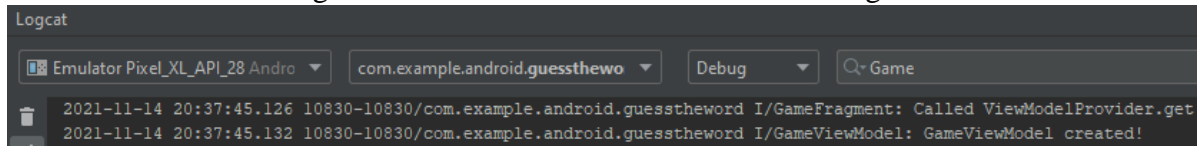
```
viewModel = ViewModelProvider( owner: this).get(GameViewModel::class.java)
```

2. Di atas inisialisasi objek ViewModel, tambahkan pernyataan log ke log panggilan metode ViewModelProvider.get().

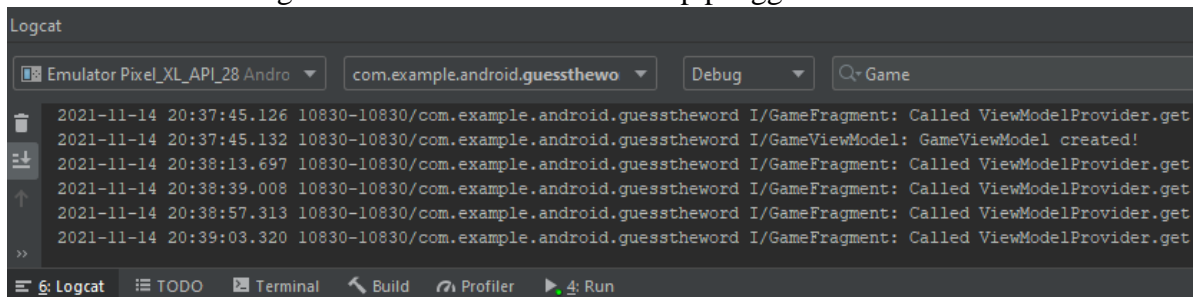
```
Log.i( tag: "GameFragment", msg: "Called ViewModelProvider.get")
viewModel = ViewModelProvider( owner: this).get(GameViewModel::class.java)
```



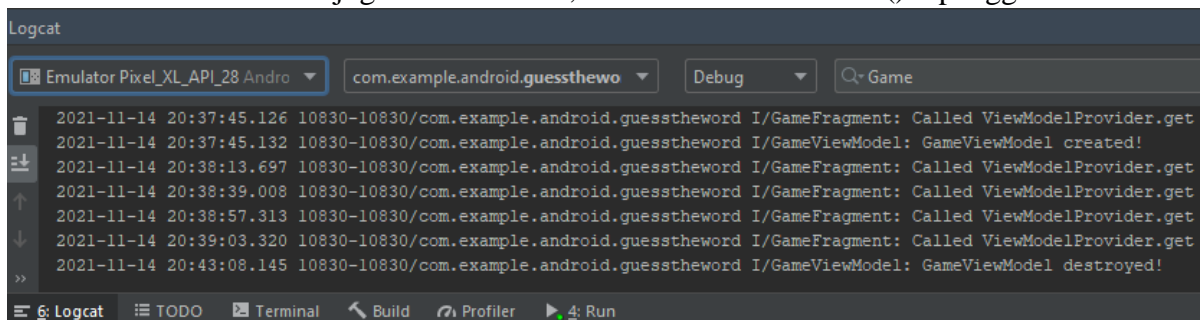
3. Jalankan aplikasinya. Di Android Studio, buka panel Logcat dan filter di Game. Ketuk tombol Play di perangkat atau emulator Anda. Layar permainan terbuka. Seperti yang ditunjukkan di Logcat, metode `onCreateView()` dari `GameFragment` memanggil metode `ViewModelProvider.get()` untuk membuat `GameViewModel`. Pernyataan logging yang Anda tambahkan ke `GameFragment` dan `GameViewModel` muncul di Logcat.



4. Aktifkan pengaturan putar otomatis pada perangkat atau emulator Anda dan ubah orientasi layar beberapa kali. `GameFragment` dimusnahkan dan dibuat ulang setiap kali, jadi `ViewModelProvider.get()` dipanggil setiap kali. Tapi `GameViewModel` dibuat hanya sekali, dan tidak dibuat ulang atau dimusnahkan untuk setiap panggilan.



5. Keluar dari game atau keluar dari fragmen game. `GameFragment` dihancurkan. `GameViewModel` terkait juga dimusnahkan, dan callback `onCleared()` dipanggil.



## 6. Tugas: Mengisi GameViewModel

`ViewModel` bertahan dari perubahan konfigurasi, jadi ini adalah tempat yang baik untuk data yang perlu bertahan dari perubahan konfigurasi:

- Letakkan data yang akan ditampilkan di layar, dan kode untuk memproses data tersebut, di `ViewModel`.
- `ViewModel` tidak boleh berisi referensi ke fragmen, aktivitas, atau tampilan, karena aktivitas, fragmen, dan tampilan tidak bertahan dari perubahan konfigurasi.

Sebagai perbandingan, berikut ini cara penanganan data UI `GameFragment` di aplikasi pemula sebelum Anda menambahkan `ViewModel`, dan setelah Anda menambahkan `ViewModel`:

- Sebelum Anda menambahkan `ViewModel`: Saat aplikasi mengalami perubahan konfigurasi seperti rotasi layar, fragmen game dihancurkan dan dibuat ulang. Datanya hilang.
- Setelah Anda menambahkan `ViewModel` dan memindahkan data UI fragmen game ke dalam `ViewModel`: Semua data yang dibutuhkan fragmen untuk ditampilkan sekarang menjadi



ViewModel. Saat aplikasi mengalami perubahan konfigurasi, ViewModel bertahan, dan datanya dipertahankan.

Dalam tugas ini, Anda memindahkan data UI aplikasi ke kelas GameViewModel, bersama dengan metode untuk memproses data. Anda melakukan ini agar data disimpan selama perubahan konfigurasi.

### Langkah 1: Pindahkan field data dan pemrosesan data ke ViewModel

Pindahkan field data dan metode berikut dari GameFragment ke GameViewModel:

1. Pindahkan field data word, score, dan wordList. Pastikan word dan score tidak bersifat private. Jangan pindahkan variabel binding, GameFragmentBinding, karena berisi referensi ke tampilan. Variabel ini digunakan untuk memekarkan tata letak, menyiapkan pendengar klik, dan menampilkan data di layar — tanggung jawab fragmen.
2. Pindahkan metode resetList() dan nextWord(). Metode ini memutuskan kata apa yang akan ditampilkan di layar.
3. Dari dalam metode onCreateView(), pindahkan panggilan metode ke resetList() dan nextWord() ke blok init GameViewModel.

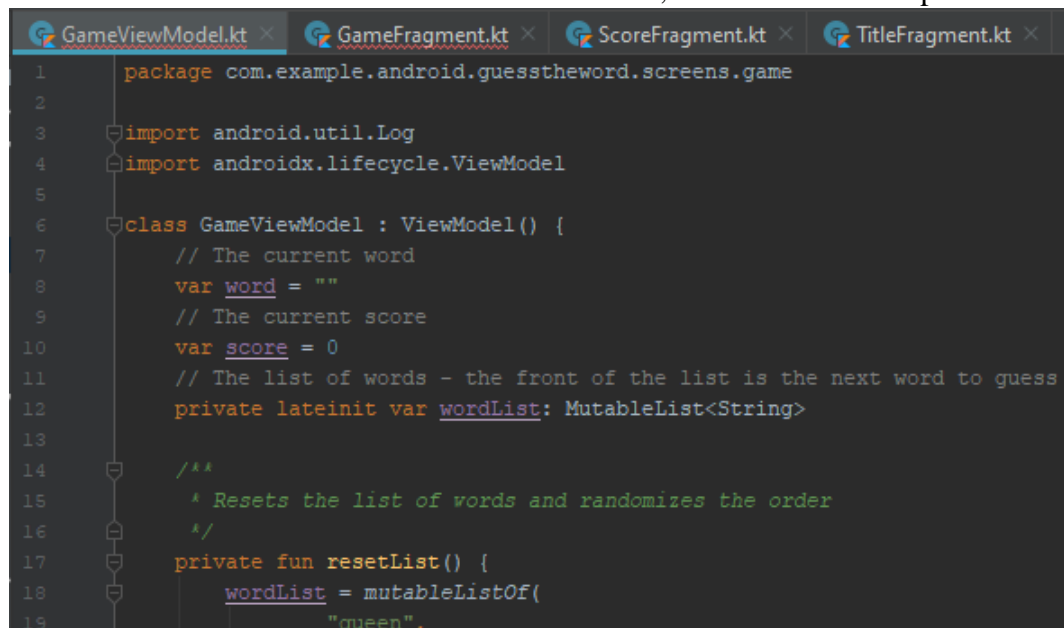
Metode ini harus ada di blok init, karena Anda harus menyetel ulang daftar kata saat ViewModel dibuat, bukan setiap kali fragmen dibuat. Anda dapat menghapus pernyataan log di blok init di GameViewModel.

Penangan klik onSkip() dan onCorrect() di GameFragment berisi kode untuk memproses data dan memperbarui UI. Kode untuk memperbarui UI harus tetap dalam fragmen, tetapi kode untuk memproses data perlu dipindahkan ke ViewModel.

Untuk saat ini, letakkan metode yang identik di kedua tempat:

1. Salin metode onSkip() dan onCorrect() dari GameFragment ke GameViewModel.
2. Di GameViewModel, pastikan metode onSkip() dan onCorrect() tidak bersifat private, karena Anda akan mereferensikan metode ini dari fragmen.

Berikut adalah kode untuk kelas GameViewModel, setelah melakukan pemfaktoran ulang:



```
1 package com.example.android.guesstheword.screens.game
2
3 import android.util.Log
4 import androidx.lifecycle.ViewModel
5
6 class GameViewModel : ViewModel() {
7     // The current word
8     var word = ""
9     // The current score
10    var score = 0
11    // The list of words - the front of the list is the next word to guess
12    private lateinit var wordList: MutableList<String>
13
14    /**
15     * Resets the list of words and randomizes the order
16     */
17    private fun resetList() {
18        wordList = mutableListOf(
19            "queen",
```

```

20         "hospital",
21         "basketball",
22         "cat",
23         "change",
24         "snail",
25         "soup",
26         "calendar",
27         "sad",
28         "desk",
29         "guitar",
30         "home",
31         "railway",
32         "zebra",
33         "jelly",
34         "car",
35         "crow",
36         "trade",
37         "bag",
38
39         "roll",
40         "bubble"
41     )
42     wordList.shuffle()
43 }
44 init {
45     resetList()
46     nextWord()
47     Log.i( tag: "GameViewModel", msg: "GameViewModel created!")
48 }
49 /**
50  * Moves to the next word in the list
51  */
52 private fun nextWord() {
53     if (!wordList.isEmpty()) {
54         //Select and remove a word from the list
55         word = wordList.removeAt( index 0)
56     }
57     updateWordText()
58     updateScoreText()
59 }
60 /** Methods for buttons presses */
61 fun onSkip() {
62     score--
63     nextWord()
64 }
65
66 fun onCorrect() {
67     score++
68     nextWord()
69 }
70
71 override fun onCleared() {
72     super.onCleared()
73     Log.i( tag: "GameViewModel", msg: "GameViewModel destroyed!")
74 }
75 }

```

Berikut adalah kode untuk kelas GameFragment, setelah pemfaktoran ulang:  
(screenshot di halaman selanjutnya)

```

GameViewModel.kt × GameFragment.kt × ScoreFragment.kt × TitleFragment.kt × Gu
1  // Copyright (C) 2019 Google Inc. .../
16
17  package com.example.android.guesstheword.screens.game
18
19  import android.os.Bundle
20  import android.util.Log
21  import android.view.LayoutInflater
22  import android.view.View
23  import android.view.ViewGroup
24  import androidx.databinding.DataBindingUtil
25  import androidx.fragment.app.Fragment
26  import androidx.lifecycle.ViewModelProvider
27  import com.example.android.guesstheword.R
28  import com.example.android.guesstheword.databinding.GameFragmentBinding
29
30  /**
31   * Fragment where the game is played
32   */
33  class GameFragment : Fragment() {
34
35      private lateinit var binding: GameFragmentBinding
36
37      private lateinit var viewModel: GameViewModel
38
39      override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
40                               savedInstanceState: Bundle?): View? {
41
42          // Inflate view and obtain an instance of the binding class
43          binding = DataBindingUtil.inflate(
44              inflater,
45              R.layout.game_fragment,
46              container,
47              attachToParent: false
48          )
49          Log.i( tag: "GameFragment", msg: "Called ViewModelProvider.get")
50          viewModel = ViewModelProvider( owner: this ).get( GameViewModel::class.java )
51
52          binding.correctButton.setOnClickListener { onCorrect() }
53          binding.skipButton.setOnClickListener { onSkip() }
54          updateScoreText()
55          updateWordText()
56          return binding.root
57      }
58      /** Methods for button click handlers */
59      private fun onSkip() {
60          score--
61          nextWord()
62      }
63
64      private fun onCorrect() {
65          score++
66          nextWord()
67      }
68      /** Methods for updating the UI */
69      private fun updateWordText() {
70          binding.wordText.text = word
71      }
72
73      private fun updateScoreText() {
74          binding.scoreText.text = score.toString()
75      }
76  }

```

## Langkah 2: Perbarui referensi ke penanganan klik dan field data di GameFragment

1. Di GameFragment, perbarui metode onSkip() dan onCorrect(). Hapus kode untuk memperbarui skor dan sebagai gantinya panggil metode onSkip() dan onCorrect() yang sesuai di viewModel.
2. Karena Anda memindahkan metode nextWord() ke ViewModel, fragmen game tidak bisa lagi mengaksesnya.

Di GameFragment, dalam metode onSkip() dan onCorrect(), ganti panggilan ke nextWord() dengan updateScoreText() dan updateWordText(). Metode ini menampilkan data di layar.

```
59 private fun onSkip() {  
60     viewModel.onSkip()  
61     updateWordText()  
62     updateScoreText()  
63 }  
64 private fun onCorrect() {  
65     viewModel.onCorrect()  
66     updateScoreText()  
67     updateWordText()  
68 }
```

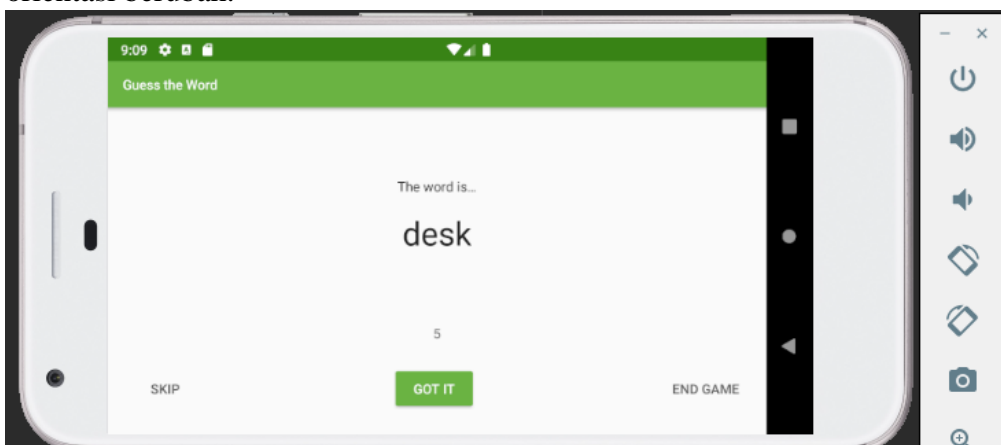
3. Di GameFragment, perbarui variabel score dan word untuk menggunakan variabel GameViewModel, karena variabel ini sekarang ada di GameViewModel.

```
70 private fun updateWordText() {  
71     binding.wordText.text = viewModel.word  
72 }  
73  
74 private fun updateScoreText() {  
75     binding.scoreText.text = viewModel.score.toString()  
76 }
```

4. Dalam GameViewModel, di dalam metode nextWord(), hapus panggilan ke metode updateWordText() dan updateScoreText(). Metode ini sekarang dipanggil dari GameFragment.

```
updateWordText()  
updateScoreText()
```

5. Bangun aplikasi dan pastikan tidak ada kesalahan. Jika Anda mengalami kesalahan, bersihkan dan bangun kembali proyek tersebut.
6. Jalankan aplikasi dan mainkan game melalui beberapa kata. Saat Anda berada di layar game, putar perangkat. Perhatikan bahwa skor saat ini dan kata saat ini dipertahankan setelah orientasi berubah.



## 7. Tugas: Menerapkan click listener untuk tombol End Game

Dalam tugas ini, Anda menerapkan pemroses klik untuk tombol End Game.

1. Di GameFragment, tambahkan metode yang disebut onEndGame(). Metode onEndGame() akan dipanggil saat pengguna mengetuk tombol End Game.

```
private fun onEndGame() {  
}
```

2. Di GameFragment, di dalam metode onCreateView(), temukan kode yang menyetel listener klik untuk tombol Got It dan Skip. Tepat di bawah dua baris ini, setel pendengar klik untuk tombol End Game. Gunakan variabel binding. Di dalam pemroses klik, panggil metode onEndGame().

```
binding.endGameButton.setOnClickListener { onEndGame() }
```

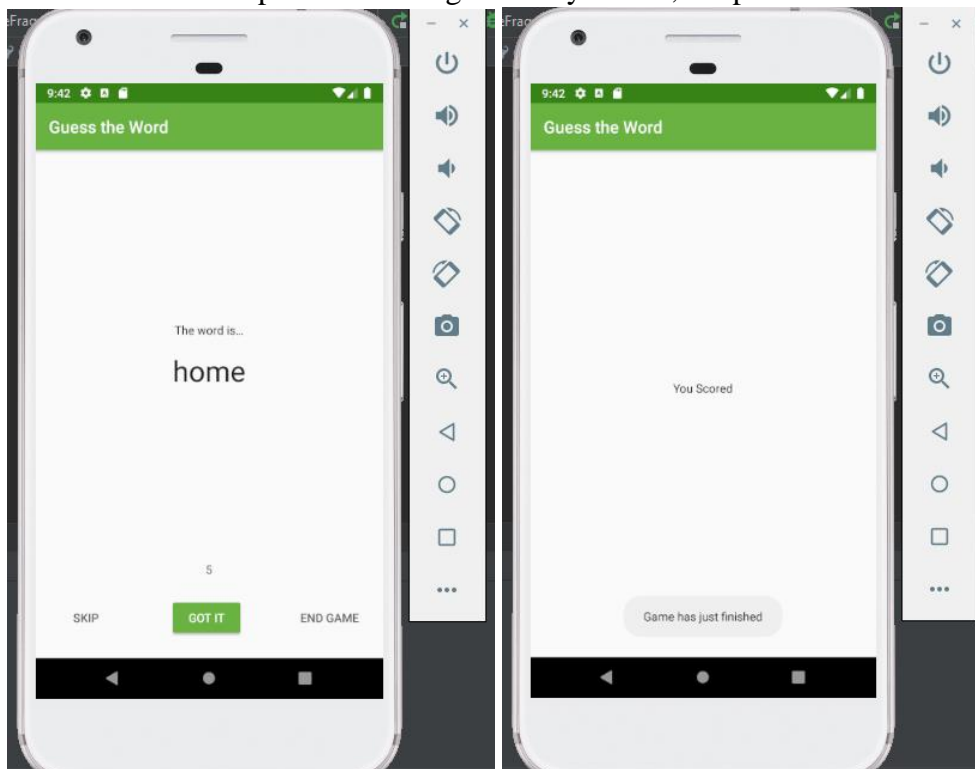
3. Di GameFragment, tambahkan metode yang disebut gameFinished() untuk menavigasi aplikasi ke layar skor. Berikan skor sebagai argumen, menggunakan Safe Args.

```
private fun gameFinished() {  
    Toast.makeText(activity, text: "Game has just finished", Toast.LENGTH_SHORT).show()  
    val action = GameFragmentDirections.actionGameToScore()  
    action.score = viewModel.score  
    NavHostFragment.findNavController( fragment: this).navigate(action)  
}
```

4. Dalam metode onEndGame(), panggil metode gameFinished ().

```
private fun onEndGame() {  
    gameFinished()  
}
```

5. Jalankan aplikasi, mainkan game, dan putar beberapa kata. Ketuk tombol End Game. Perhatikan bahwa aplikasi menavigasi ke layar skor, tetapi skor akhir tidak ditampilkan.

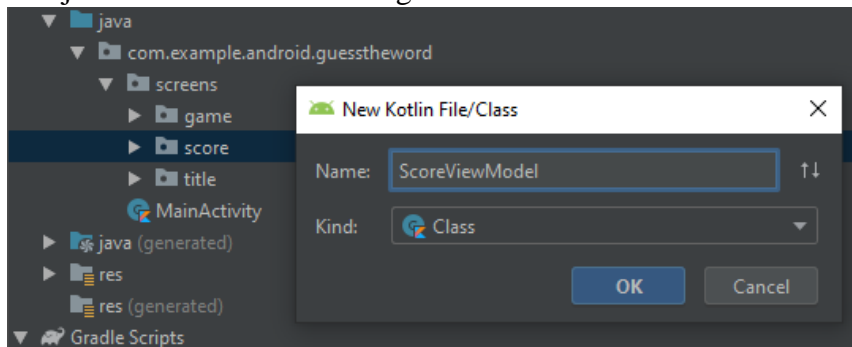


## 8. Tugas: Menggunakan ViewModelFactory

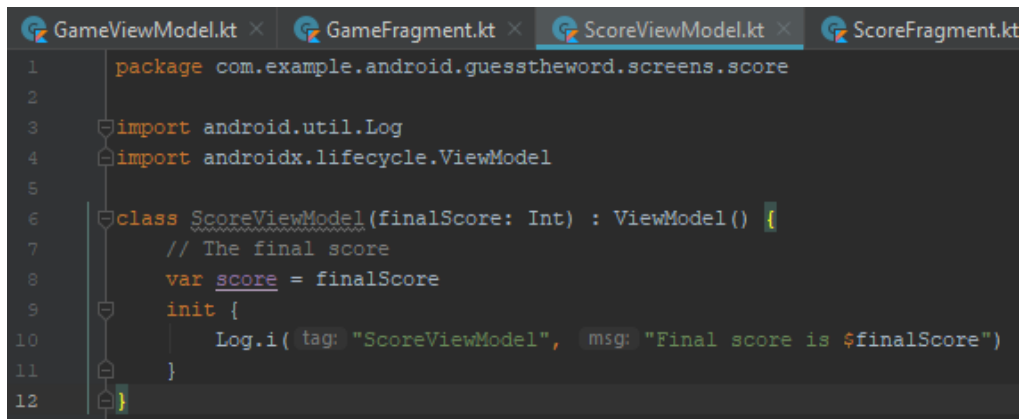
Saat pengguna mengakhiri permainan, ScoreFragment tidak menampilkan skor. Anda ingin ViewModel yang memegang skor akan ditampilkan oleh ScoreFragment. Anda akan meneruskan nilai skor selama inisialisasi ViewModel menggunakan pola metode pabrik (factory method pattern). Pola metode pabrik adalah pola desain kreasi yang menggunakan metode pabrik untuk membuat objek. Metode pabrik adalah metode yang mengembalikan instance dari kelas yang sama.

Dalam tugas ini, Anda membuat ViewModel dengan konstruktor berparameter untuk fragmen skor dan metode pabrik untuk membuat instance ViewModel.

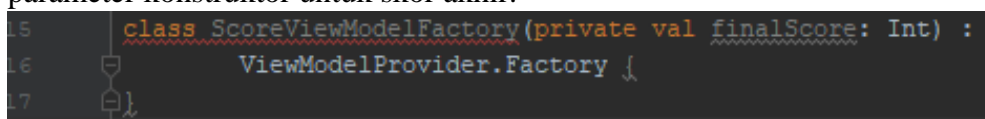
1. Di bawah paket score, buat kelas Kotlin baru yang disebut ScoreViewModel. Kelas ini akan menjadi ViewModel untuk fragmen score.



2. Perluas kelas ScoreViewModel dari ViewModel. Tambahkan parameter konstruktor untuk skor akhir. Tambahkan blok init dengan pernyataan log.
3. Di kelas ScoreViewModel, tambahkan variabel yang disebut score untuk menyimpan skor akhir.



4. Di bawah paket score, buat kelas Kotlin yang dinamai ScoreViewModelFactory. Kelas ini akan bertanggungjawab untuk instantiasi objek ScoreViewModel.
5. Perluas kelas ScoreViewModelFactory dari ViewModelProvider.Factory. Tambahkan parameter konstruktor untuk skor akhir.



6. Dalam ScoreViewModelFactory, Android Studio menunjukkan error tentang unimplemented abstract member. Untuk menyelesaikan error, override metode create(). Dalam metode create(), kembalikan objek ScoreViewModel yang baru diciptakan.

```
class ScoreViewModelFactory(private val finalScore: Int) :  
    ViewModelProvider.Factory {  
    override fun <T : ViewModel?> create(modelClass: Class<T>): T {  
        if (modelClass.isAssignableFrom(ScoreViewModel::class.java)) {  
            return ScoreViewModel(finalScore) as T  
        }  
        throw IllegalArgumentException("Unknown ViewModel class")  
    }  
}
```

7. Dalam ScoreFragment, ciptakan variabel kelas untuk ScoreViewModel dan ScoreViewModelFactory.

```
private lateinit var viewModel: ScoreViewModel  
private lateinit var viewModelFactory: ScoreViewModelFactory
```

8. Dalam ScoreFragment, didalam onCreateView(), setelah inisialisasi variabel binding, inisialisasi viewModelFactory. Gunakan ScoreViewModelFactory. Lewatkan dalam skor akhir dari bundle argument, sebagaimana parameter konstruktor ke ScoreViewModelFactory().

```
viewModelFactory = ScoreViewModelFactory(ScoreFragmentArgs.fromBundle(requireArguments()).score)
```

9. Di onCreateView(), setelah menginisialisasi viewModelFactory, inisialisasi objek viewModel. Panggil metode ViewModelProvider.get(), teruskan konteks fragmen skor terkait dan viewModelFactory. Ini akan membuat objek ScoreViewModel menggunakan metode pabrik yang ditentukan di kelas viewModelFactory.

```
viewModel = ViewModelProvider( owner: this, viewModelFactory)  
    .get(ScoreViewModel::class.java)
```

10. Dalam metode onCreateView(), setelah menginisialisasi viewModel, setel teks tampilan scoreText ke skor akhir yang ditentukan dalam ScoreViewModel.

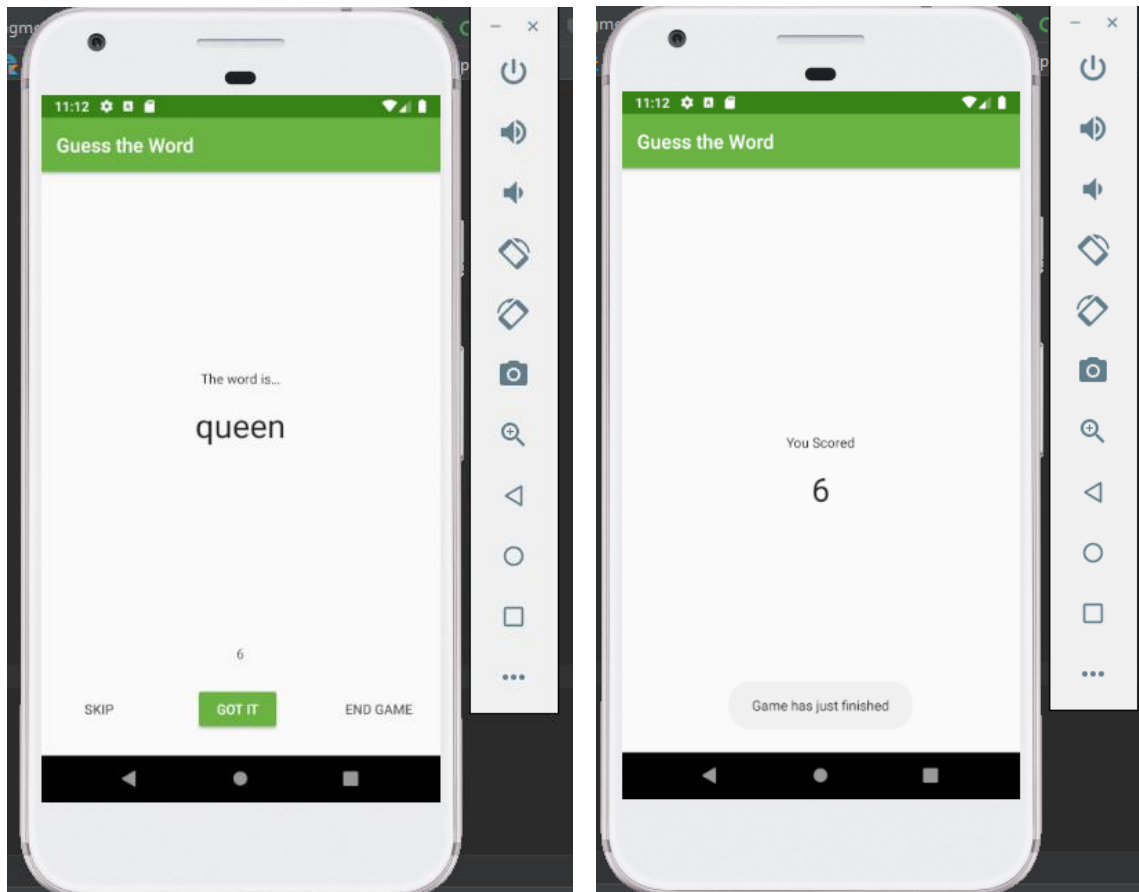
```
binding.scoreText.text = viewModel.score.toString()
```

11. Jalankan aplikasi Anda dan mainkan gamenya. Gilir beberapa atau semua kata dan ketuk End Game. Perhatikan bahwa fragmen skor sekarang menampilkan skor akhir.

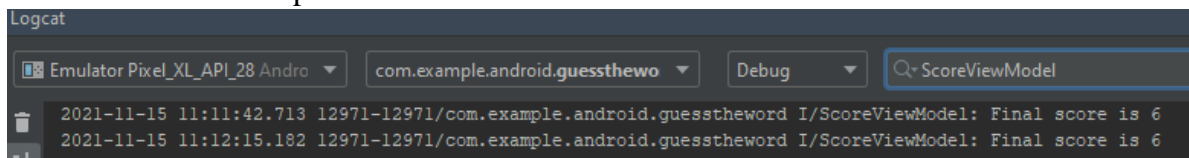
(screenshot emulator di halaman selanjutnya)

(screenshot emulator di halaman selanjutnya)

(screenshot emulator di halaman selanjutnya)



12. Opsional: Periksa log ScoreViewModel di Logcat dengan memfilter di ScoreViewModel. Nilai skor harus ditampilkan.

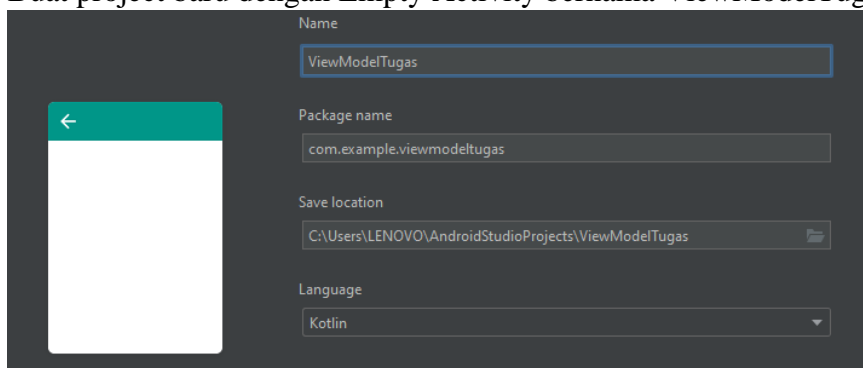


## TUGAS

Buat aplikasi baru dengan menerapkan ViewModel dan ViewModelProvider dengan click listener

Di tugas ini, saya membuat aplikasi untuk menampilkan angka dan nilai boolean. Jika angka ganjil, maka akan muncul nilai false, dan jika angka genap maka akan muncul nilai true. Berikut langkah-langkahnya:

1. Buat project baru dengan Empty Activity bernama ViewModelTugas





2. Buka build.gradle (Module: app), lalu tambahkan dependencies untuk lifecycle, livedata, dan viewmodel

```
36 def lifecycle_version = "2.2.0"
37
38 // ViewModel
39 implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:$lifecycle_version"
40 // LiveData
41 implementation "androidx.lifecycle:lifecycle-livedata-ktx:$lifecycle_version"
```

Penjelasan:

Kode di atas merupakan kode untuk menambahkan opsi viewmodel dan livedata ke dalam project android studio yang kita buat. Kita gunakan versi lifecycle 2.2.0 sebagai bridge untuk menambahkan viewmodel dan livedata.

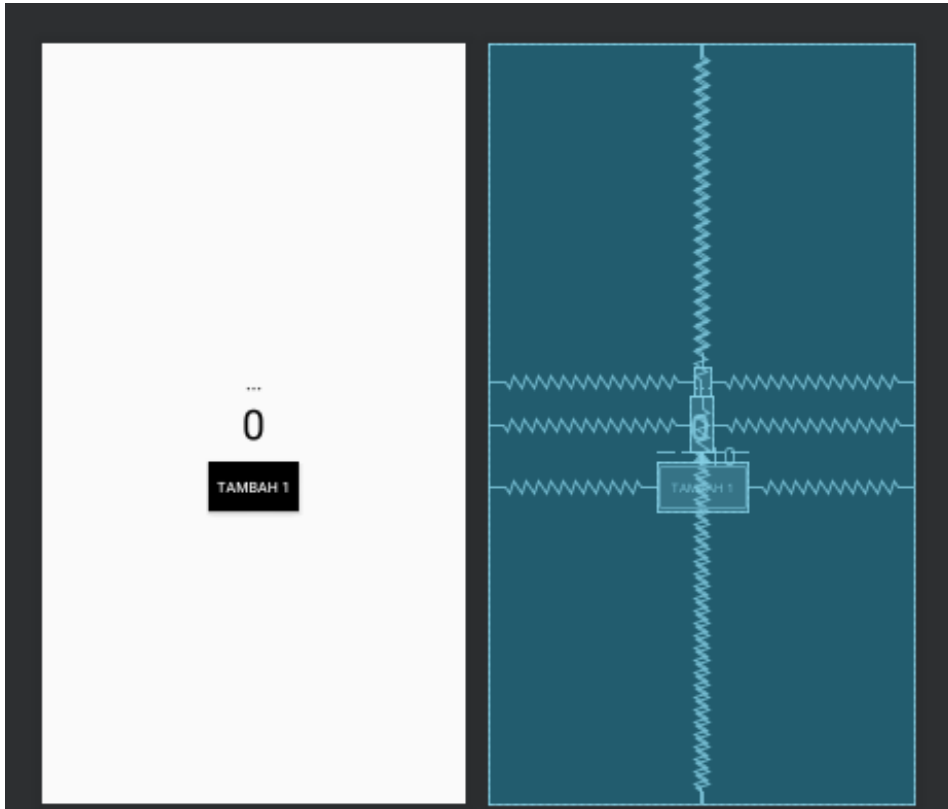
3. Buka activity\_main.xml, dan tulis kode berikut untuk membuat tampilan pada layout.

```
activity_main.xml x MainActivity.kt x app x
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context=".MainActivity" >
9
10    <TextView
11        android:id="@+id/tv_textView"
12        android:layout_width="wrap_content"
13        android:layout_height="wrap_content"
14        android:layout_centerInParent="true"
15        android:text="0"
16        android:textColor="@android:color/black"
17        android:textSize="40sp" />
18
19    <TextView
20        android:id="@+id/tv_booleanText"
21        android:layout_above="@id/tv_textView"
22        android:layout_width="wrap_content"
23        android:layout_height="wrap_content"
24        android:layout_centerInParent="true"
25        android:text="..."
26        android:textColor="@android:color/black"
27        android:textSize="20sp" />
28
29    <Button
30        android:id="@+id/btn_button"
31        android:layout_width="wrap_content"
32        android:layout_height="wrap_content"
33        android:layout_below="@id/tv_textView"
34        android:layout_centerHorizontal="true"
35        android:layout_margin="10dp"
36        android:background="@android:color/black"
37        android:text="tambah 1"
38        android:textColor="@android:color/white" />
39
40 </RelativeLayout>
```

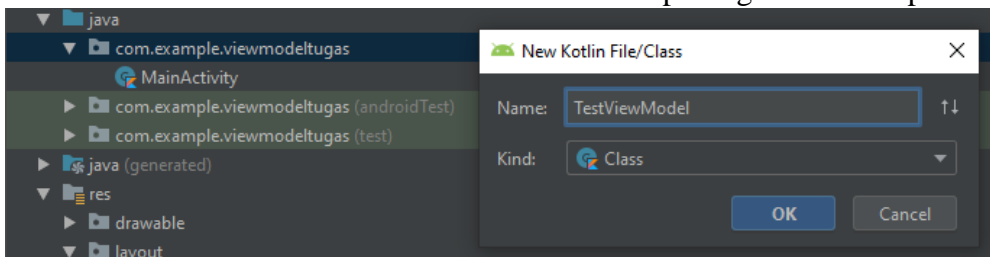
Penjelasan:

Di file xml ini, kita menggunakan RelativeLayout. RelativeLayout adalah layout yang menempatkan suatu item relative terhadap parent atau item lain pada UI. Di dalamnya, kita tambahkan widget berupa button dan dua textview untuk menampilkan angka dan boolean.

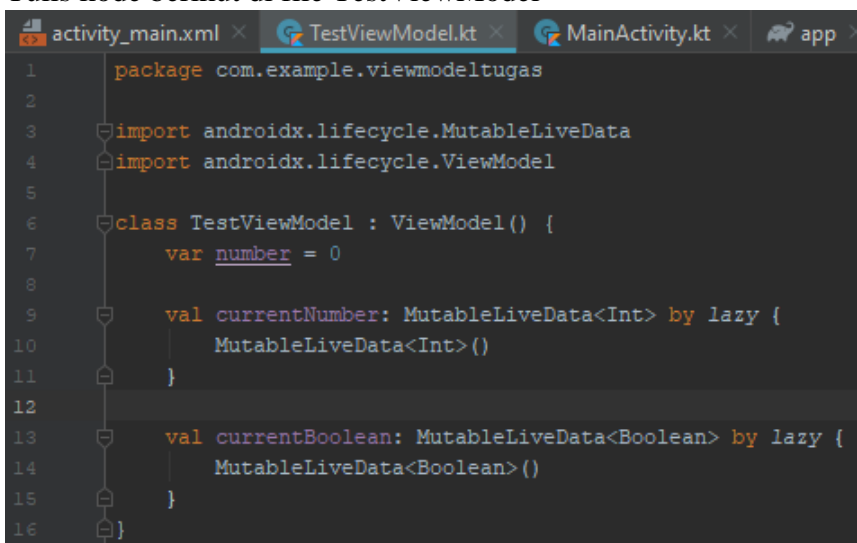
4. Sehingga tampilan pada tab design menjadi seperti berikut



5. Buat class baru bernama TestViewModel di dalam package com.example.viewmodeltugas



6. Tulis kode berikut di file TestViewModel



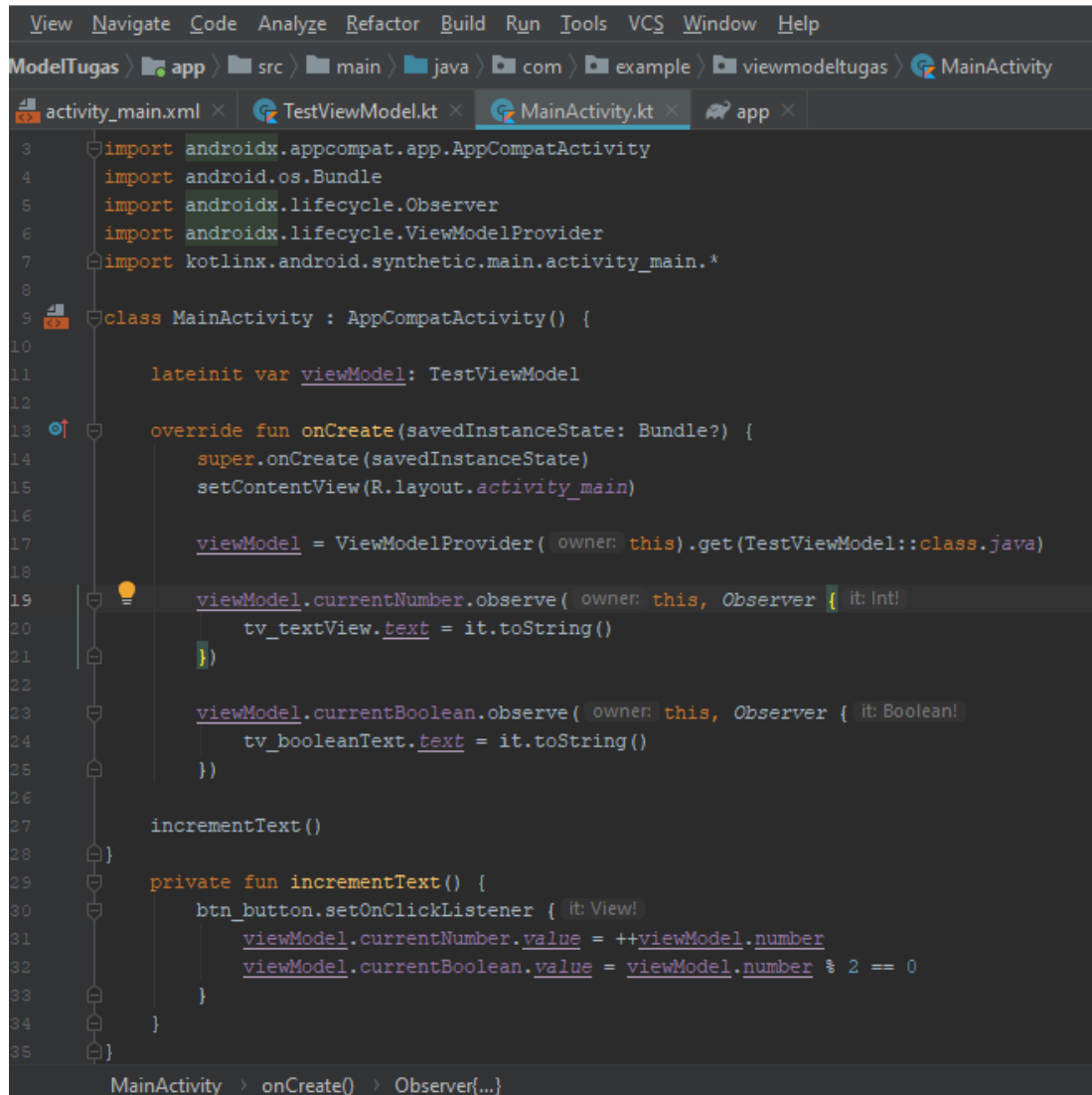
Penjelasan:

Di kode berikut, kita gunakan MutableLiveData untuk menampung dan mengupdate angka maupun boolean. Untuk variabel currentNumber, kita menggunakan tipe data Integer untuk

angka dan pada variabel `currentBoolean` menggunakan tipe data boolean untuk menampung dan menampilkan true and false. Lazy initialization berguna untuk supaya variabel tidak akan diinisialisasi kecuali saat kita menggunakan variabel itu di dalam code.

## 7. Tulis kode berikut pada MainActivity.kt

ModelTugas [C:\Users\LENOVO\AndroidStudioProjects\ViewModelTugas] - ...app\src\main\java\com\example\viewmodeltu



```
1  import androidx.appcompat.app.AppCompatActivity
2  import android.os.Bundle
3  import androidx.lifecycle.Observer
4  import androidx.lifecycle.ViewModelProvider
5  import kotlinx.android.synthetic.main.activity_main.*
6
7  class MainActivity : AppCompatActivity() {
8
9      lateinit var viewModel: TestViewModel
10
11      override fun onCreate(savedInstanceState: Bundle?) {
12          super.onCreate(savedInstanceState)
13          setContentView(R.layout.activity_main)
14
15          viewModel = ViewModelProvider( owner: this ).get( TestViewModel::class.java )
16
17          viewModel.currentNumber.observe( owner: this, Observer { it: Int!
18              tv_textView.text = it.toString()
19          })
20
21          viewModel.currentBoolean.observe( owner: this, Observer { it: Boolean!
22              tv_booleanText.text = it.toString()
23          })
24
25          incrementText()
26      }
27
28      private fun incrementText() {
29          btn_button.setOnClickListener { it: View!
30              viewModel.currentNumber.value = ++viewModel.number
31              viewModel.currentBoolean.value = viewModel.number % 2 == 0
32          }
33      }
34  }
35
```

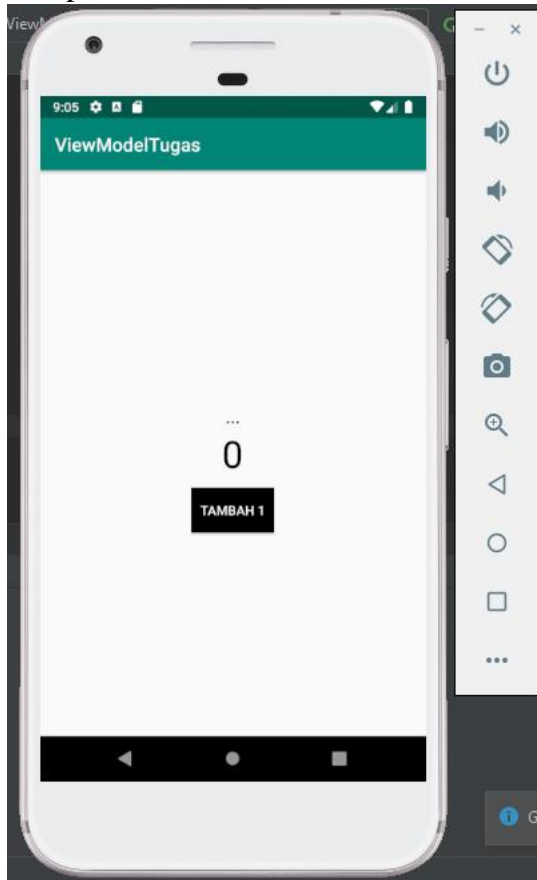
Penjelasan:

Hal pertama yang harus kita lakukan adalah membuat variabel `lateinit` bertipe `viewModel` yang akan menampung data dari class `TestViewModel` yang sudah kita buat tadi. Kedua, kita lakukan instansiasi (pembuatan objek) dengan menuliskan `viewModel = viewModelProvider`. Kemudian inisialisasikan dengan keyword `this`. Lalu, kita gunakan method `get` untuk mendapatkan data dari `viewModel` yang akan kita gunakan yaitu dari class `TestViewModel` dengan.

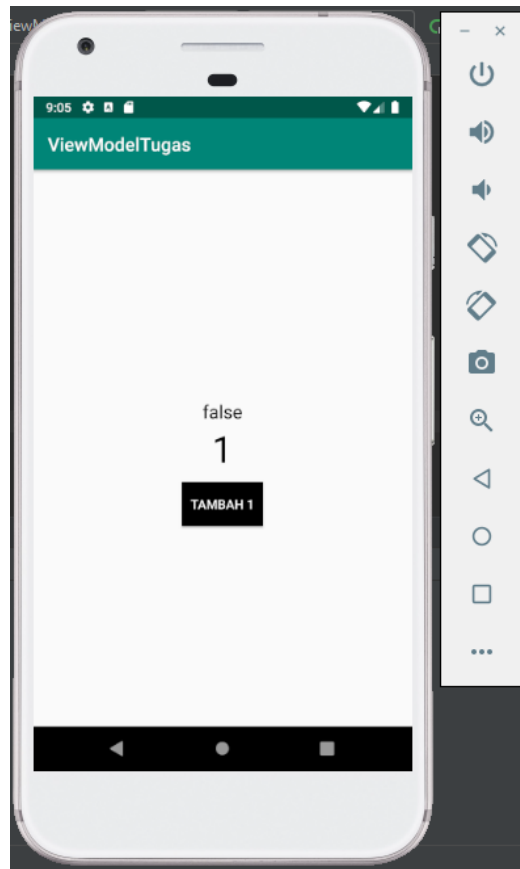
Selanjutnya, kita buat sebuah method atau fungsi untuk mengendalikan perilaku Button dan melakukan iterasi integer dan iterasi boolean yaitu `private fun incrementText()`. Pertama kita gunakan `setOnClickListener` untuk memberikan fungsi atau efek ketika button `btn_button` diklik. Iterasi pertama yang kita gunakan adalah `++viewModel.number` yang melakukan iterasi angka yang terus bertambah satu setiap button diklik. Iterasi kedua adalah untuk boolean yaitu dengan `viewModel.number % 2 == 0` yang artinya jika genap maka true dan jika ganjil maka false.

8. Jalankan aplikasi dan klik button beberapa kali

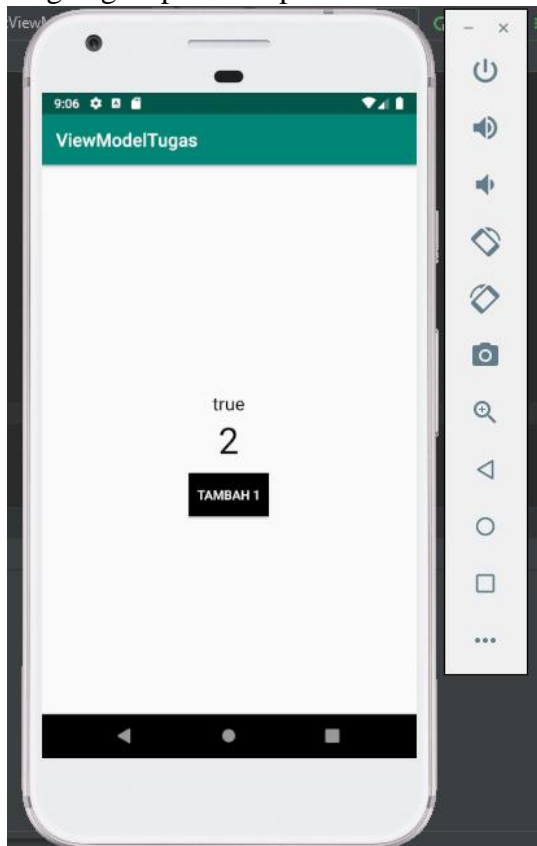
Tampilan awal



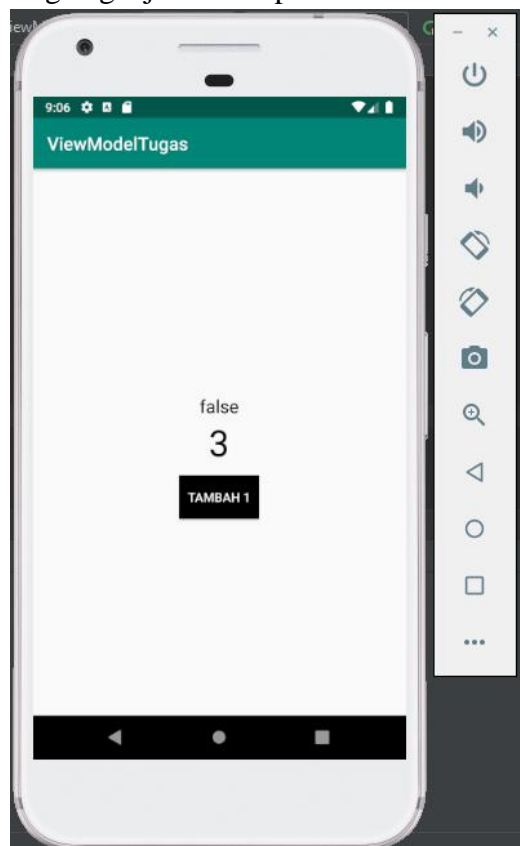
angka ganjil menampilkan false



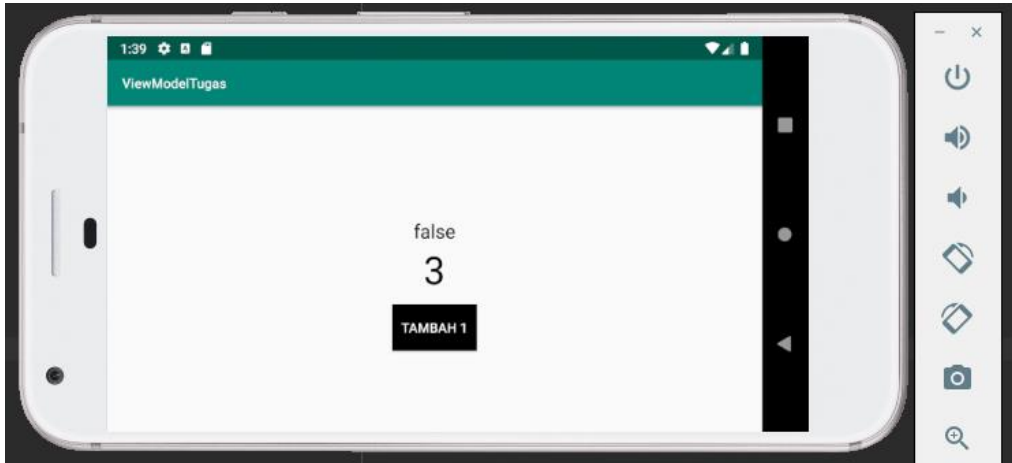
Angka genap menampilkan true



Angka ganjil menampilkan false



Saat dirotate, nilai pada aplikasi masih tersimpan



---

## KESIMPULAN

Pada pertemuan 8 ini, saya berhasil memenuhi tujuan dibuatnya laporan ini yaitu mampu menggunakan ViewModel dan ViewModelProvider dalam aplikasi untuk menyimpan dan mengelola data terkait UI. Tidak hanya ViewModel dan ViewModelProvider, saya juga mempelajari tentang bagaimana mengaplikasikan viewModel terhadap button yang diberi fungsi clickOnListener. Saya mempelajari banyak hal diantaranya tentang apa itu viewModel, bagaimana bentuk arsitektur viewModel, cara menggunakan kelas Lifecycle dan ViewModelFactory, cara mempertahankan data UI melalui perubahan konfigurasi perangkat, dan cara membuat objek ViewModel menggunakan UI ViewModelProvider.Factory.

Terima Kasih