



BINAR
ACADEMY

Activity & Activity Lifecycle

Silver - Chapter 3 - Topic 2

**Selamat datang di Chapter 3 Topic 2 online
course Android Developer dari Binar Academy!**





Hi Teman-Teman 🙋

Pada topik sebelumnya, kita sudah belajar konsep pemrograman dengan fitur **Android Permission**.

Nah, di **Topik 2** ini kita bakal mendalami fitur lain yang penting dalam pembuatan aplikasi, yaitu **Activity dan Activity Lifecycle**.

Yuk, lanjut!



Detailnya, kita bakal bahas hal-hal berikut ini:

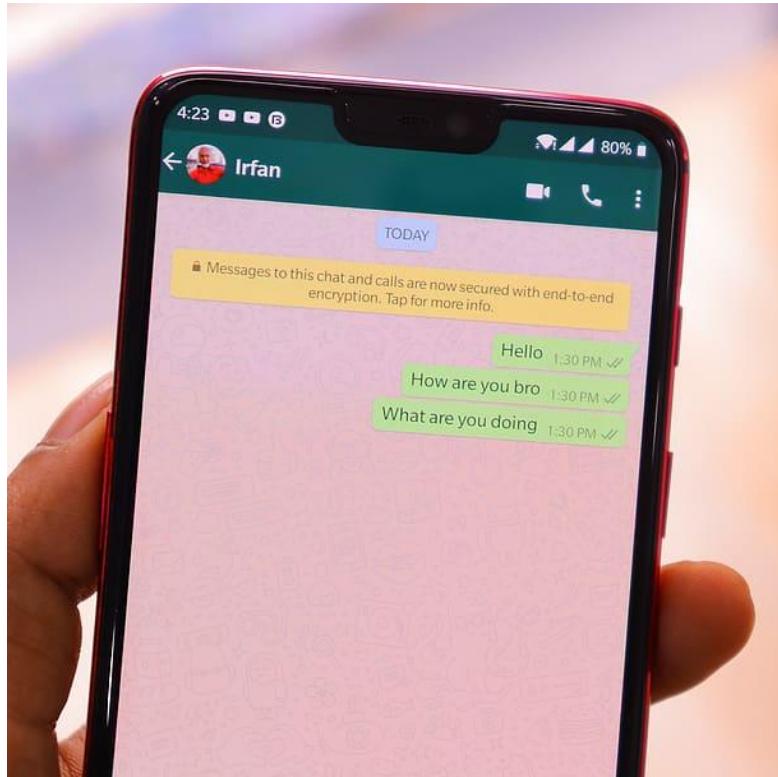
- Pemahaman Konsep Activity
- Activity dengan File XML
- Activity dengan File Class
- Activity Lifecycle





Sesuai dengan detail pembahasan tadi,
kita coba pahamin dulu nih Konsep
Activity pada Android.

Apa sih **Activity** pada Android itu?



Apa itu Activity?

Bisa dibilang Activity adalah **layar tampilan pada aplikasi kita**.

Dengan adanya si Activity ini, kita jadi bisa berinteraksi sekaligus menampilkan berbagai layar di aplikasi kita. Misalnya kayak layar login, layar chat, layar register, dan lain-lain.

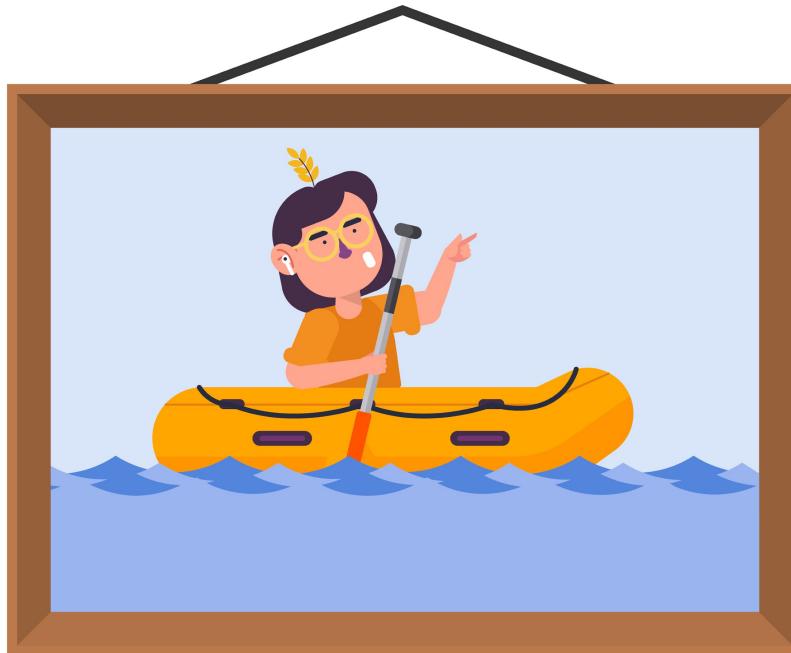


Biar paham, coba pakai analogi, ya!

Activity tuh mirip kayak sebuah foto yang dipajang di dinding rumah kita.

Foto yang dipajang biasanya memanfaatkan bingkai supaya bisa digantung dan dilihat oleh orang lain.

Nah, dengan bingkai ini foto juga bisa diubah-ubah atau diganti sesuai keinginan pemiliknya. Tapi bingkainya? tetep sama..



Dari cerita itu, coba kondisinya kamu ganti dengan panduan seperti ini:

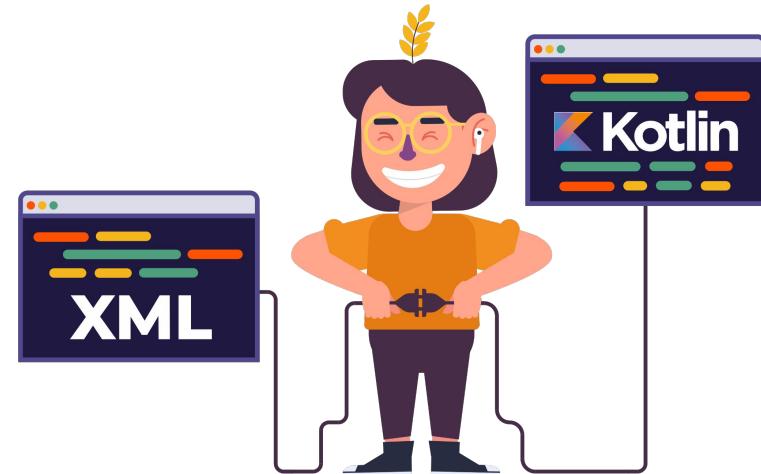
Bingkai: Activity yang menampilkan layar aplikasi kita.

Foto: Desain tampilan aplikasi yang bisa kita sesuaikan.

Ngomongin tampilan aplikasi, secara default, Activity terdiri dari 2 file yang saling terhubung, yaitu :

- **File Layout XML**, dan
- **File Class (Kotlin atau Java)**.

Biar makin paham, kita ulas dulu satu-satu yaa~





File XML



Activity dengan File XML?

File Layout XML bertugas untuk menyediakan hal-hal yang akan ditampilkan pada Activity.

Nggak cuma itu, di file Layout XML kita juga bisa tambahin View berupa Widget dan View Group untuk mengatur tata letak.





Yepp, ngatur tata letaknya, kita bisa lho mendesain tampilan dalam layout XML ini dengan mode design yang caranya cukup **drag and drop atau metode html kayak bikin website.**

Selain itu, kita juga bisa liat tampilan live preview. Sehingga, jadi bisa intip deh tampilan aplikasinya di HP kita nanti kayak gimana.





Tadaa.. begini deh contoh Layout XML yang tadi kita omongin

The screenshot shows the Android Studio interface with the XML code on the left and the corresponding UI design on the right.

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingStart="24dp"
    android:paddingEnd="24dp"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Nama Saya"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/editText2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="Email Saya"
        app:layout_constraintTop_toBottomOf="@+id/editText" />

    <androidx.appcompat.widget.AppCompatButton
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:text="Button"
        app:layout_constraintTop_toBottomOf="@+id/editText2" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

UI Design (ConstraintLayout):

- EditText 1 (Nama Saya):** A text input field with placeholder text "Nama Saya". It has constraints: end-to-end of parent, start-to-start of parent, and top-to-top of parent.
- EditText 2 (Email Saya):** A text input field with placeholder text "Email Saya". It has constraints: top-to-bottom of EditText 1, and top-to-bottom of parent.
- AppCompatButton:** A button with text "Button". It has constraints: top-to-bottom of EditText 2, and top-to-bottom of parent. It also has a margin of 32dp from the top of its parent.



File XML aja gak cukup looh~

Hmm kenapa gitu? Karena Layout XML itu kan fungsinya mengurus Tampilannya aja yaa... biar tampilannya interaktif, kita membutuhkan **Kotlin Class** sebagai **Class Activity**.

Wow faktanya, dengan Class Activity ini, kita bisa melakukan berbagai aksi logic.





File Class



File Class

Lanjut ke sini, File Class itu bisa berupa Kotlin Class atau Java Class.

Class dari suatu Activity bertugas **menangani proses logika untuk setiap interaksi pengguna dengan aplikasi kita.**

Proses logika yang kayak gimana? di file Class kita dapat menangani event click, scroll, double-click, dan lain sebagainya terhadap tampilan Activity.





Untuk menggunakan tampilan XML yang udah dibuat dengan Activity, bisa **menggunakan method `setContentView()` dalam method `onCreate()`** dengan parameter yang mengarah ke file XML yang diinginkan.

Kayak apa praktiknya? Kita coba geser slide setelah ini.





Pada contoh di sebelumnya, terdapat baris code :

```
setContentView(R.layout.activity_note_detail)
```

Baris code tersebut, akan menerapkan file XML yang bernama **activity_note_detail.xml** ke Activity.

Bisa kita amati bahwa file Layout XML dapat kita akses dengan perantara suatu Class yang bernama **R.java**.

Apa sih Class R.java ini?



```
class NoteDetailActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_note_detail)
    }
}
```



Apa itu Class R.java?

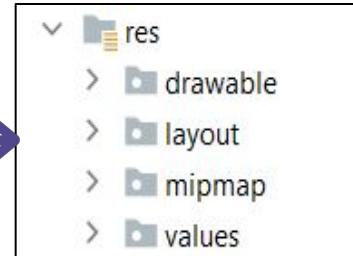
Sebenarnya Class R.java ini nggak hanya menghubungkan Activity dengan Layout XML aja, tapi juga dengan semua sumber daya / resource yang ada di dalam folder **res**.

Class R ini akan dibuat secara otomatis ketika proses **gradle build**.



R.
 anim (binar.academy.myapplication.R)
 animator (binar.academy.myapplication.R)
 attr (binar.academy.myapplication.R)
 bool (binar.academy.myapplication.R)
 color (binar.academy.myapplication.R)
 dimen (binar.academy.myapplication.R)
 drawable (binar.academy.myapplication.R)
 id (binar.academy.myapplication.R)
 integer (binar.academy.myapplication.R)
 interpolator (binar.academy.myapplication.R)
 layout (binar.academy.myapplication.R)
 menu (binar.academy.myapplication.R)

Press Enter to insert, Tab to replace



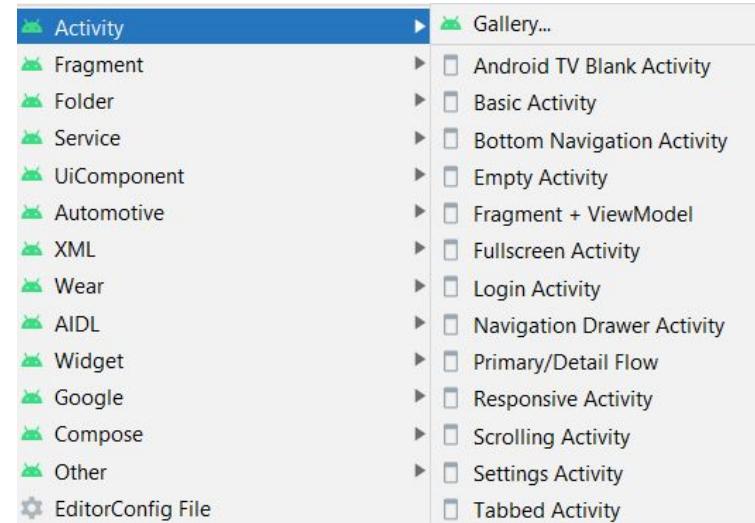


Trus, Gimana cara membuat Activity?

Bisa sambil merem! Kalo kita udah buat project Android-nya, bisa praktekin hal dibawah ini :

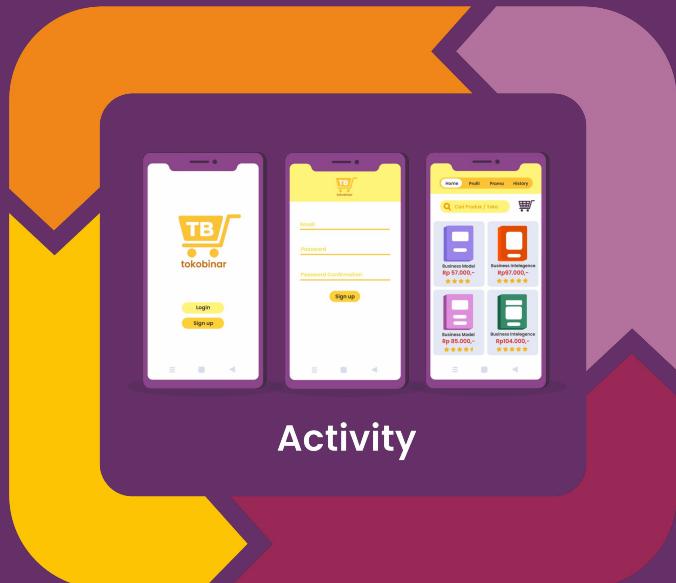
- Klik kanan pada package tempat Activity baru akan dibuat
- Arahkan ke New
- Arahkan ke Activity

Voilà! maka setelah itu bakal tampil berbagai macam jenis template Activity yang udah disediakan oleh Android Studio.





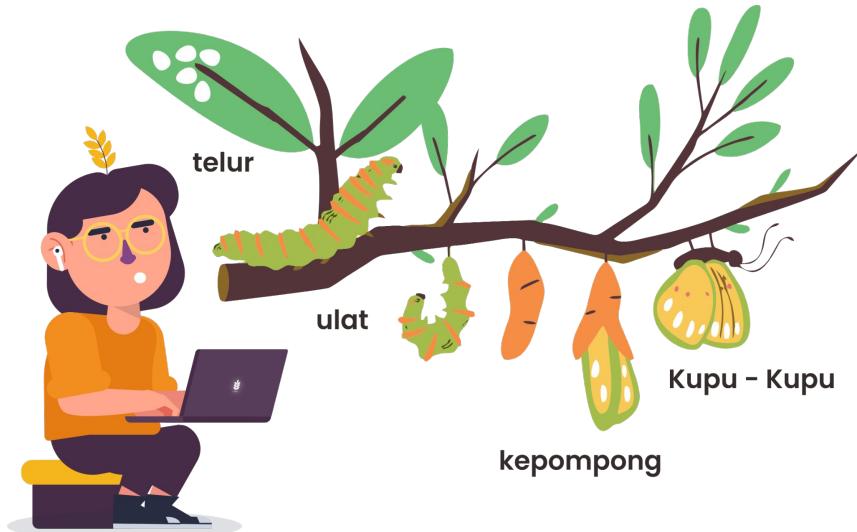
LIFECYCLE



Gengs ini nih yang seru.. Activity itu melalui sebuah metamorfosis.

Yepp, ada fase lahir dan akhir yang mirip kayak kupu-kupu.

Tapi.. kayak apa sih Lifecycle yang dimiliki oleh **Activity**?

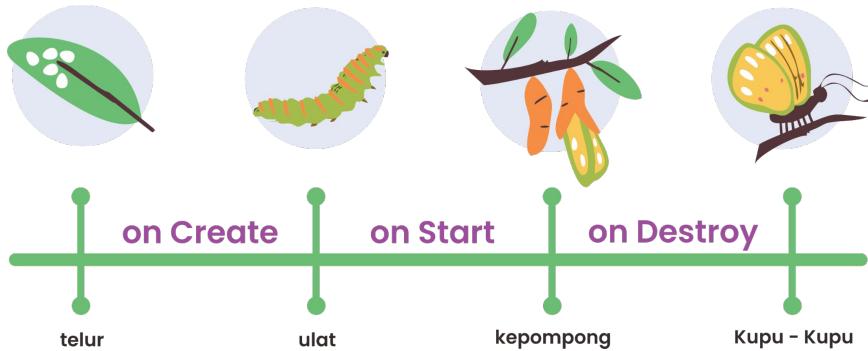


Biar gampang, kita pakai analogi, ya!

Activity Lifecycle **memiliki berbagai fase**. Dimulai ketika Activity itu “dilahirkan” atau dibuat.

Activity itu “Hidup” dengan berbagai perlakuan dari pengguna, hingga Activity itu “diakhiri” oleh Garbage Collector.

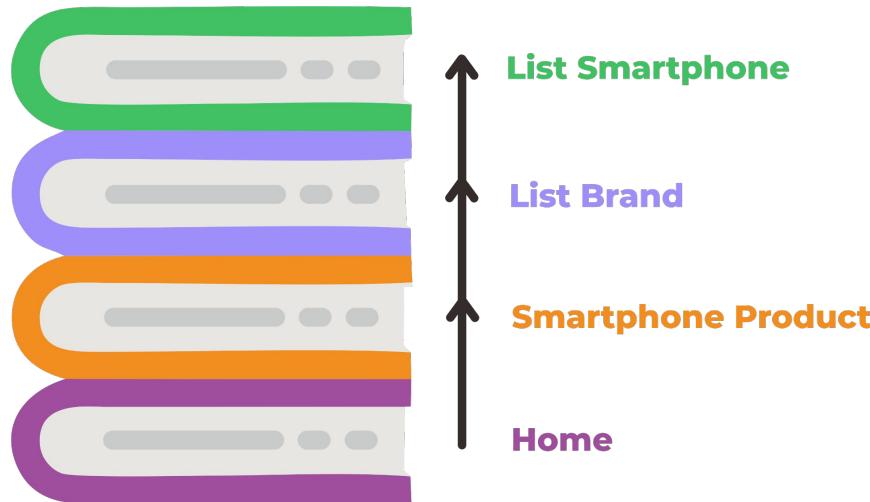
Sama seperti siklus hidup pada kupu-kupu yang dimulai dari telur dan ulat.



Dari cerita itu, coba kondisinya kamu ganti dengan panduan seperti ini :

Setiap fase makhluk hidup ini (telur, larva, dll) diibaratkan seperti method yang ada di Activity, yaitu :

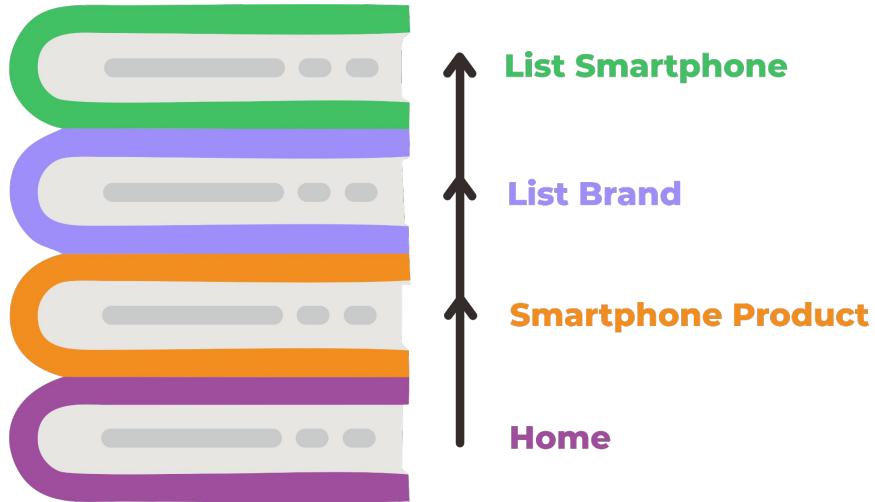
- `onCreate()`,
- `onStart()`, hingga
- `onDestroy()`.



Jadi, Apa itu Activity Lifecycle?

Gampangnya, Activity Life cycle adalah **Activity dalam sistem yang dikelola ke dalam tumpukan Activity**.

Ketika sebuah Activity baru dimulai, biasanya ditempatkan pada tumpukan paling atas dan menjadi Activity yang berjalan.



Nah, Activity sebelumnya selalu berada di bawah pada tumpukan.

Dan nggak akan berjalan di bagian *foreground* (bagian depan) lagi sampai Activity baru keluar.

Kurang lebih alurnya kayak gambar disamping nih.



Suatu activity pada dasarnya memiliki 4 kondisi:

Karena Activity adalah tampilan di layar HP kita, ada beberapa kondisi yang perlu kita tahu ketika suatu tampilan itu muncul.

Dijamin 100% kamu pasti udah familiar banget nih sama tampilan ini..

Tinggal belajar kondisinya aja~





Kondisi 1

Activity berada pada *foreground* (pada posisi teratas dari tumpukan)

Proses yang Terjadi

- Activity akan aktif atau sedang berjalan.
- Biasanya Activity yang berada pada foreground merupakan Activity yang sedang berinteraksi dengan pengguna.



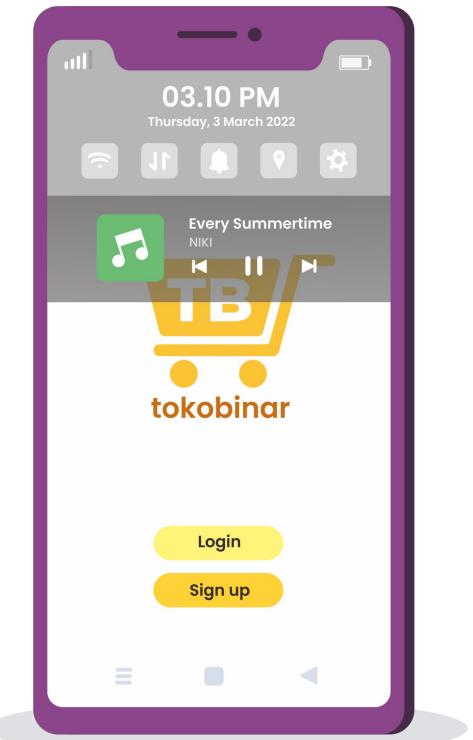


Kondisi 2

Activity telah kehilangan fokus tetapi masih tampil pada pengguna.

Proses yang Terjadi

- Biasanya Activity tersebut sebelumnya tidak tampil secara utuh atau Activity tersebut memiliki fokus di atas Activity kita.
- Activity lain memiliki posisi lebih tinggi dalam mode multi-window atau Activity itu sendiri tidak fokus dalam mode window saat itu.
- Activity tersebut benar-benar berjalan dan mempertahankan semua informasi dan tetap melekat pada window manager.



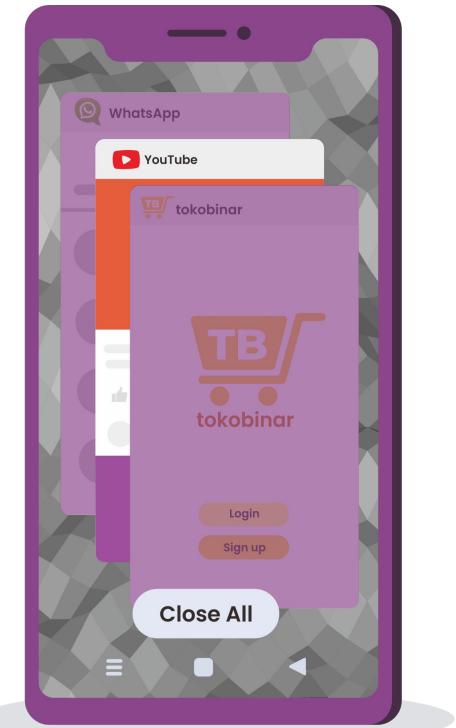


Kondisi 3

Activity sepenuhnya ditimpak oleh Activity lain,

Proses yang Terjadi

- Activity akan dihentikan atau disembunyikan.
- Activity tersebut masih menyimpan semua informasi kondisi dan anggotanya.
- Namun, tampilannya tidak lagi terlihat oleh pengguna dan akan **dihancurkan otomatis** oleh sistem jika memori diperlukan di tempat lain.



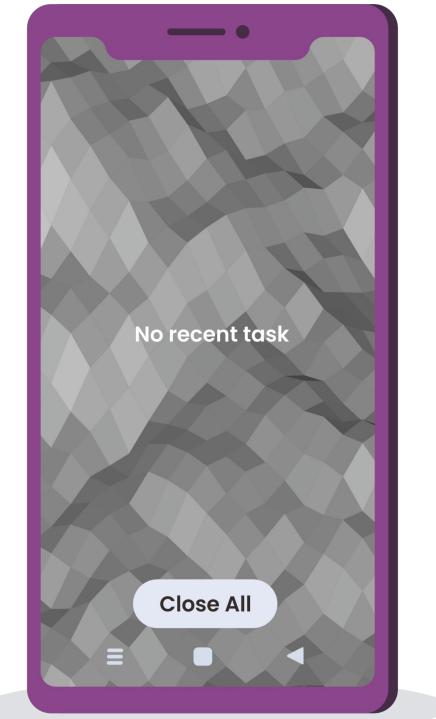


Kondisi 4

Activity dihancurkan atau dimatikan prosesnya oleh sistem

Proses yang Terjadi

- Sistem dapat menghilangkan Activity dari memori dengan meminta untuk menghancurnya, atau hanya mematikan prosesnya.
- Ketika itu ditampilkan lagi kepada pengguna, itu harus sepenuhnya dimulai kembali dan dikembalikan ke kondisi sebelumnya.



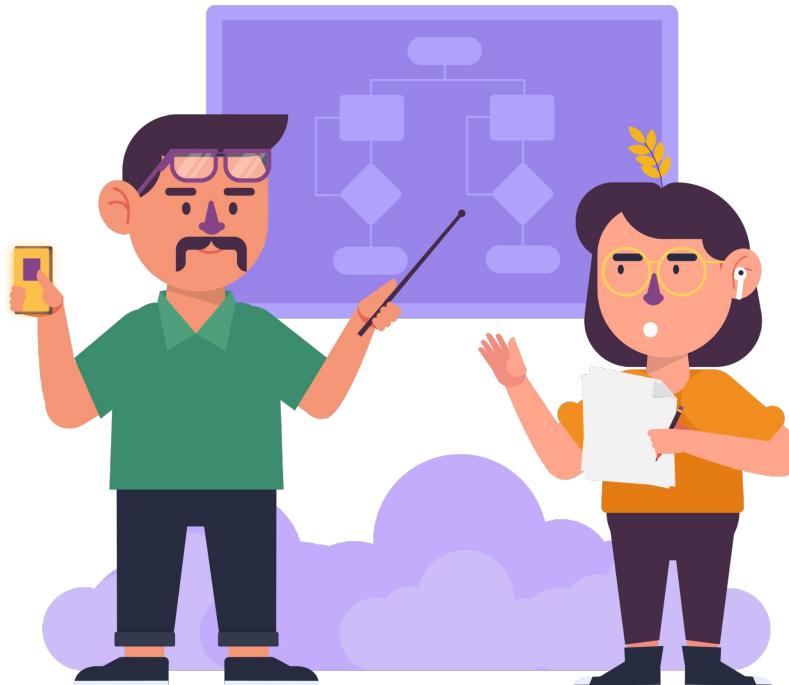


Diagram Activity Lifecycle

Untuk menyempurnakan pemahaman kita, pada beberapa slide ke depan, kita bakal nemuin diagram alur suatu Activity

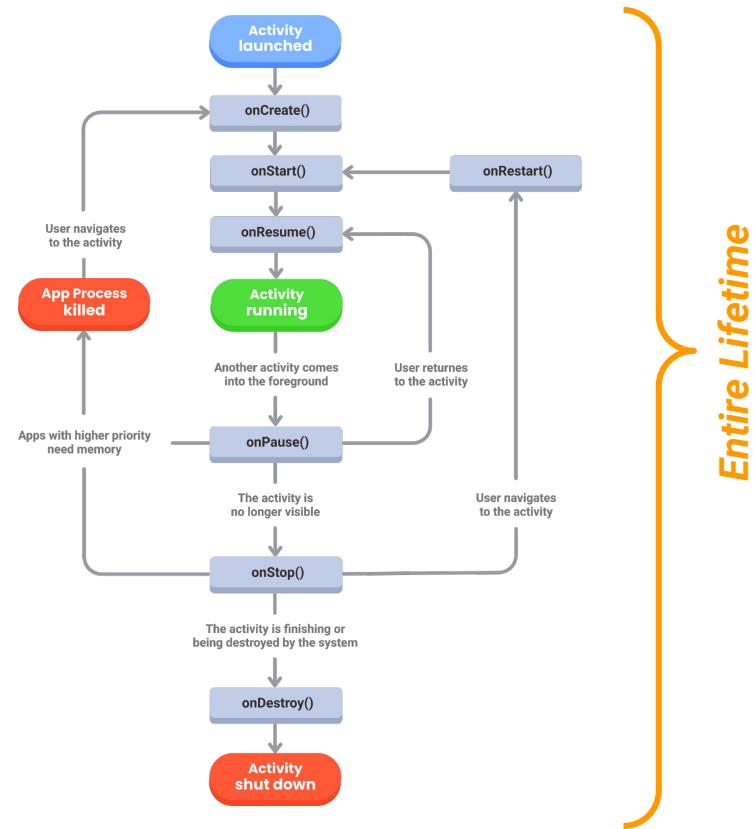
Kalo kamu liat bentuk kotak persegi pada diagram, itu mewakili metode *callback* yang bisa kita terapin untuk melakukan operasi ketika Activity bergerak antar kondisi.

Kalo kamu nemu bentuk oval, itu adalah kondisi utama dimana Activity dapat berada ketika dijalankan.



Ngomongin Diagram Activity Lifecycle tadi, ada tiga **Nested Loops** pada Android Lifecycle yang terkait dengan Activity:

1. **Entire Lifetime**
2. **Visible Lifetime**
3. **Foreground Lifetime**



1. Entire Lifetime

Merupakan durasi waktu antara panggilan ke **onCreate()** dan **onDestroy()**.

Activity ini membuat kita harus menyiapkan semua sources seperti User Interfaces di **onCreate()** dan melepas kannya di **onDestroy()**.



2. Visible Lifetime

Merupakan durasi waktu antara panggilan yang terjadi `onStart()` hingga ke `onStop()`.

Pada waktu ini, pengguna dapat melihat Activity yang tampil pada layar dan bisa berinteraksi dengan pengguna tersebut.

Kalo Activity-nya dimulai, maka `onStop()` bakal dipanggil dalam Lifecycle dan membuat Activity yang saat ini terlihat jadi nggak terlihat lagi.



Selama activity di-Visible Lifetime, `onStart()` dan `onStop()` biasanya akan disebut beberapa kali dulu sehingga Activity berubah antara terlihat dan disembunyikan dari pengguna.

Biasanya kalo gitu, status yang terlihat hanya terkait dengan faktor bentuk yang lebih besar. Dimana suatu Activity terlihat, tetapi tidak ditampilkan di *foreground*.



3. Foreground lifetime

Merupakan durasi Activity yang terjadi antara panggilan ke `onResume()` hingga ke `onPause()`.

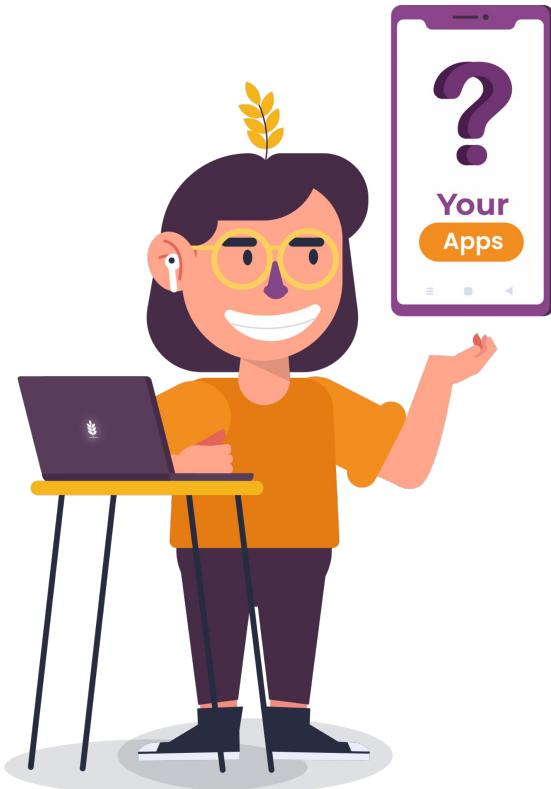
Pada waktu ini, Activity akan terlihat aktif, dan sering berinteraksi dengan pengguna.

Funfact-nya, suatu Activity bisa sering melakukan transisi masuk dan keluar dari *foreground*.



Misalnya gini, `onPause()` dipanggil saat perangkat tidur atau ketika dialog muncul.

Karena kondisi ini sering terjadi transisi, code dalam kedua method ini harus cukup ringan untuk menghindari transisi lambat yang membuat pengguna menunggu.



Biar pahamnya komplit, yuk kita ikut challenge ini~

Buatlah suatu Aplikasi hanya dengan **Satu Activity**, lalu jalankan aplikasi tersebut.

Setelah aplikasi terbuka, aktifkan rotasi otomatis handphone, lalu **ubah rotasi aplikasi** yang tadinya **portrait** menjadi **landscape**.

Amati dan tuliskan secara lengkap dan berurutan **method dalam Activity Lifecycle** apa saja **yang telah dijalankan** oleh Activity tersebut!

Prosedur pengumpulan tugas diserahkan kepada mentor.

Kita akan membahas bersama-sama di pertemuan berikutnya.

Saatnya kita Quiz!





1. Method apa yang berfungsi untuk menerapkan suatu Layout XML pada suatu Activity?

- A. setLayoutXML()
- B. setContentView()
- C. setXMLContent()



1. Method apa yang berfungsi untuk menerapkan suatu Layout XML pada suatu Activity?

- A. setLayoutXML()
- B. setContentView()
- C. setXMLContent()

Method `setContentView()` dipanggil dalam method `onCreate()`. Method ini digunakan untuk *binding* Layout XML dengan Activity.

Method `setContentView()` harus dipanggil sebelum semua hal yang berhubungan dengan View dilakukan. Termasuk *binding* View dan penanganan event dari View.



2. Apa perantara agar Activity Class (Java/Kotlin) dapat mengakses Layout XML?

- A. R.java
- B. Build.gradle
- C. Resource.java



2. Apa perantara agar Activity Class (Java/Kotlin) dapat mengakses Layout XML?

- A. R.java
- B. Build.gradle
- C. Resource.java

Sudah jelas bahwa Class R.java bisa mengakses Layout XML untuk *di-binding* dengan Activity. Tidak hanya Layout XML, Class R.java juga bisa digunakan untuk mengakses semua *resource* yang ada pada folder res di project Android kita. Seperti animasi, font, string, mipmap, dan masih banyak lagi.



3. Pada Activity Lifecycle, method pertama dan terakhir yang dijalankan adalah?

- A. onCreate() dan onDestroy()
- B. onStart() dan onStop()
- C. onStart() dan onDestroy()



3. Pada Activity Lifecycle, method pertama dan terakhir yang dijalankan adalah?

- A. onCreate() dan onDestroy()
- B. onStart() dan onStop()
- C. onStart() dan onDestroy()

Seperti yang dapat kita ketahui pada diagram Activity Lifecycle, bahwa onCreate() akan dijalankan pertama kali sebelum method-method lainnya.

Lalu method terakhir yang akan dijalankan adalah method onDestroy() yang akan menghancurkan Activity agar *memory* yang digunakan oleh Activity bisa digunakan oleh proses lainnya. Hal ini dilakukan oleh Garbage Collector.



4. Di bawah ini, method mana saja yang termasuk *foreground lifetime*?

- A. Semua method yang ada pada Activity Lifecycle
- B. Hanya onResume() dan onPause()
- C. onStart(), onResume(), onPause(), dan onStop()



4. Di bawah ini, method mana saja yang termasuk *foreground lifetime*?

- A. Semua method yang ada pada Activity Lifecycle
- B. Hanya onResume() dan onPause()
- C. onStart(), onResume(), onPause(), dan onStop()

Hanya method onResume() dan onPause() yang termasuk dalam *foreground Lifetime*.

Sedangkan opsi A, semua method yang ada pada Activity Lifecycle termasuk dari Entire Lifetime dan opsi C, onStart(), onResume(), onPause(), dan onStop() termasuk dalam Visible Lifetime.



5. Manakah urutan yang benar pada *activity lifecycle*?

- A. onStart - onCreate - onResume - onStop - onPause - onDestroy
- B. onCreate - onStart - onResume - onPause - onStop - onDestroy
- C. onCreate - onStart - onResume - onStop - onPause - onDestroy



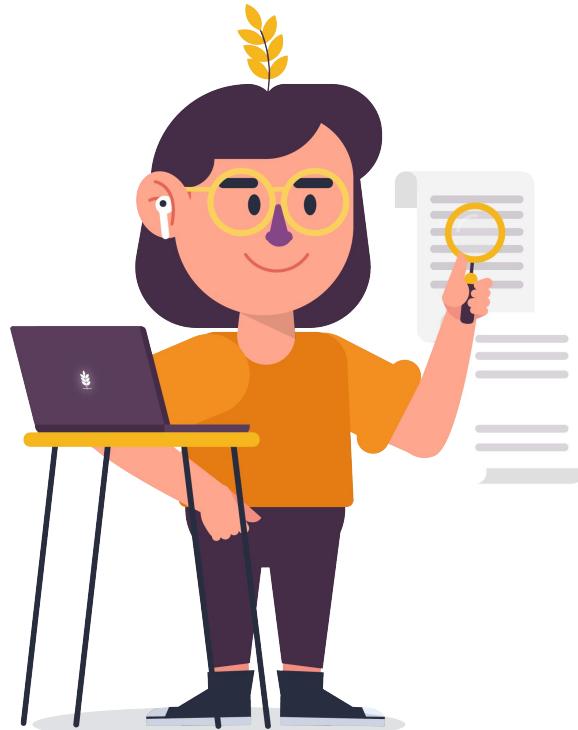
5. Manakah urutan yang benar pada *activity lifecycle*?

- A. onStart - onCreate - onResume - onStop - onPause - onDestroy
- B. onCreate - onStart - onResume - onPause - onStop - onDestroy
- C. onCreate - onStart - onResume - onStop - onPause - onDestroy

Betul banget! Urutan yang benar pada *Activity Lifecycle* adalah onCreate - onStart - onResume - onPause - onStop - onDestroy.

Referensi dan bacaan lebih lanjut~

1. [Activity | Android Developers](#)
2. [The Activity Lifecycle | Android Developers](#)





Nah, selesai sudah pembahasan kita di Chapter 3 Topic 2 ini.

Selanjutnya, kita bakal bahas saudaranya Activity, yaitu **Fragment & Fragment Lifecycle**.

Penasaran kayak gimana? Cus langsung ke topik selanjutnya~



Terima Kasih!



Next Topic

loading...