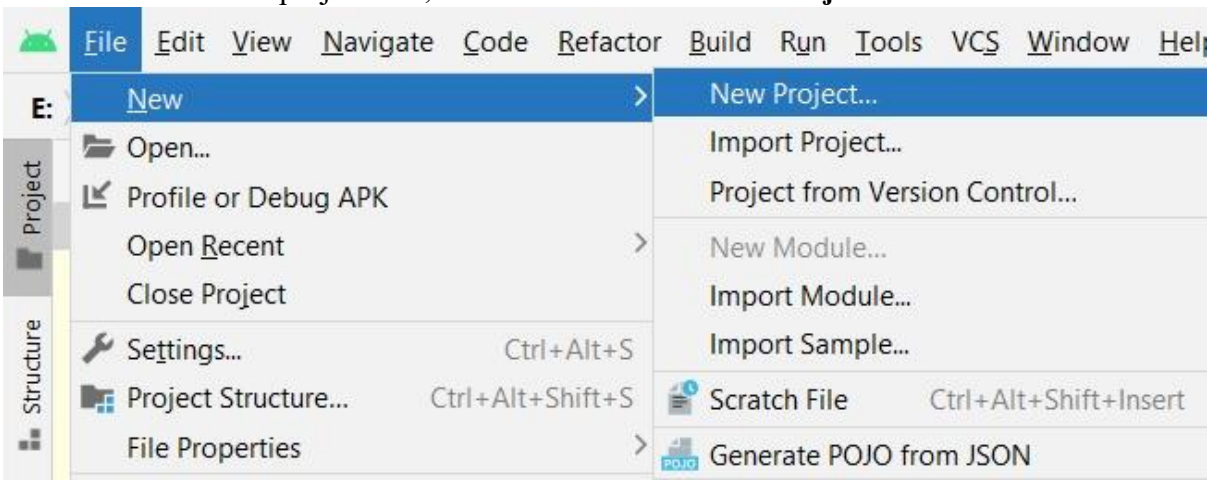
Dasar Pengembangan Aplikasi Android

Ringkasan Pengenalan Android Studio

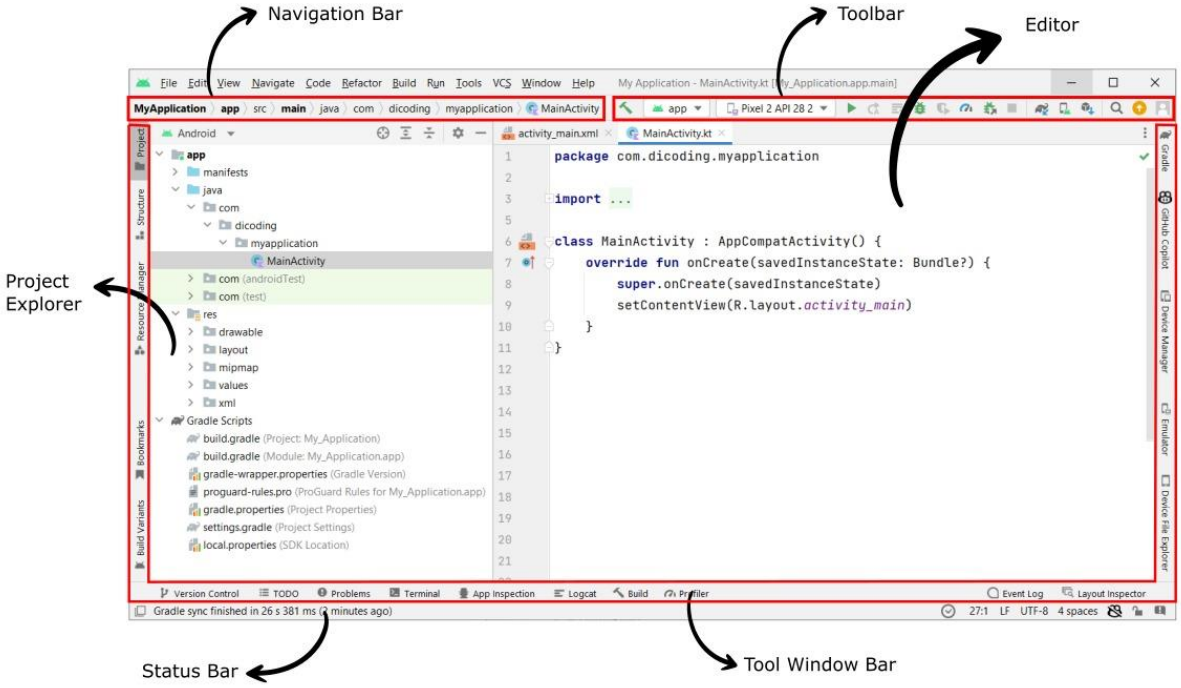
- **Android Studio** adalah IDE (*Integrated Development Environment*) berbasis Intellij IDEA yang digunakan khusus untuk membangun aplikasi Android.
- Bahasa pemrograman yang bisa digunakan untuk mengembangkan aplikasi Android adalah Kotlin dan Java. Namun, yang direkomendasikan dari Google adalah Kotlin.
- Android Studio memiliki fitur yang lengkap untuk mengembangkan aplikasi seperti berikut.
    - **Template**: template memulai project maupun Activity tanpa harus membuatnya dari nol.
    - **Intelligent code editor**: code completion yang memudahkan untuk menulis kode dengan cepat tanpa harus menuliskan secara lengkap. Selain itu, juga ada warning apabila terdapat kesalahan penulisan kode.
    - **Design tool**: digunakan untuk mendesain aplikasi beserta melihat preview secara langsung sebelum dijalankan.
    - **Flexible build system**: Android Studio menggunakan Gradle yang fleksibel untuk menciptakan build variant yang berbeda untuk berbagai device. Anda juga dapat menganalisa prosesnya secara mendetail.
    - **Emulator**: menjalankan aplikasi tanpa harus menggunakan device Android. Dilengkapi dengan Instant Run untuk melihat perubahan tanpa harus build project dari awal.
    - **Debugging**: memudahkan untuk mencari tahu masalah.
    - **Testing**: menjalankan pengujian untuk memastikan semua kode aman sebelum rilis.
    - **Publish**: membuat berkas AAB/APK dan menganalisanya guna dibagikan dan di-*publish* ke PlayStore.
    - **Integrasi**: Terhubung dengan berbagai layanan yang memudahkan untuk mengembangkan aplikasi, seperti Github, Firebase, dan Google Cloud.

- **JRE** (Java Runtime Environment) adalah Virtual Machine untuk menjalankan program Java, sedangkan **JDK** (Java SE Development Kit) merupakan compiler dan tools untuk membuat program.
- Saat ini ketika menginstal Android Studio sudah terdapat JDK yang bawaan (OpenJDK) yang bisa digunakan. Jadi, Anda tidak perlu menginstal JDK secara terpisah.
- Untuk melihat proses **instalasi Android Studio** secara lebih detail di masing-masing OS, Anda dapat melihatnya pada tautan berikut.
    - https://developer.android.com/studio/install
- **Android SDK** adalah sekumpulan tool yang digunakan untuk mengembangkan aplikasi Android.
- Berikut adalah beberapa cara untuk membuat project baru pada Android Studio.
    - Ketika pertama kali membuka Android Studio dan muncul window **"Welcome to Android Studio"**, klik **"Start a new Android Studio project."**
    - Jika sudah membuka project lain, klik **File → New → New Project**.



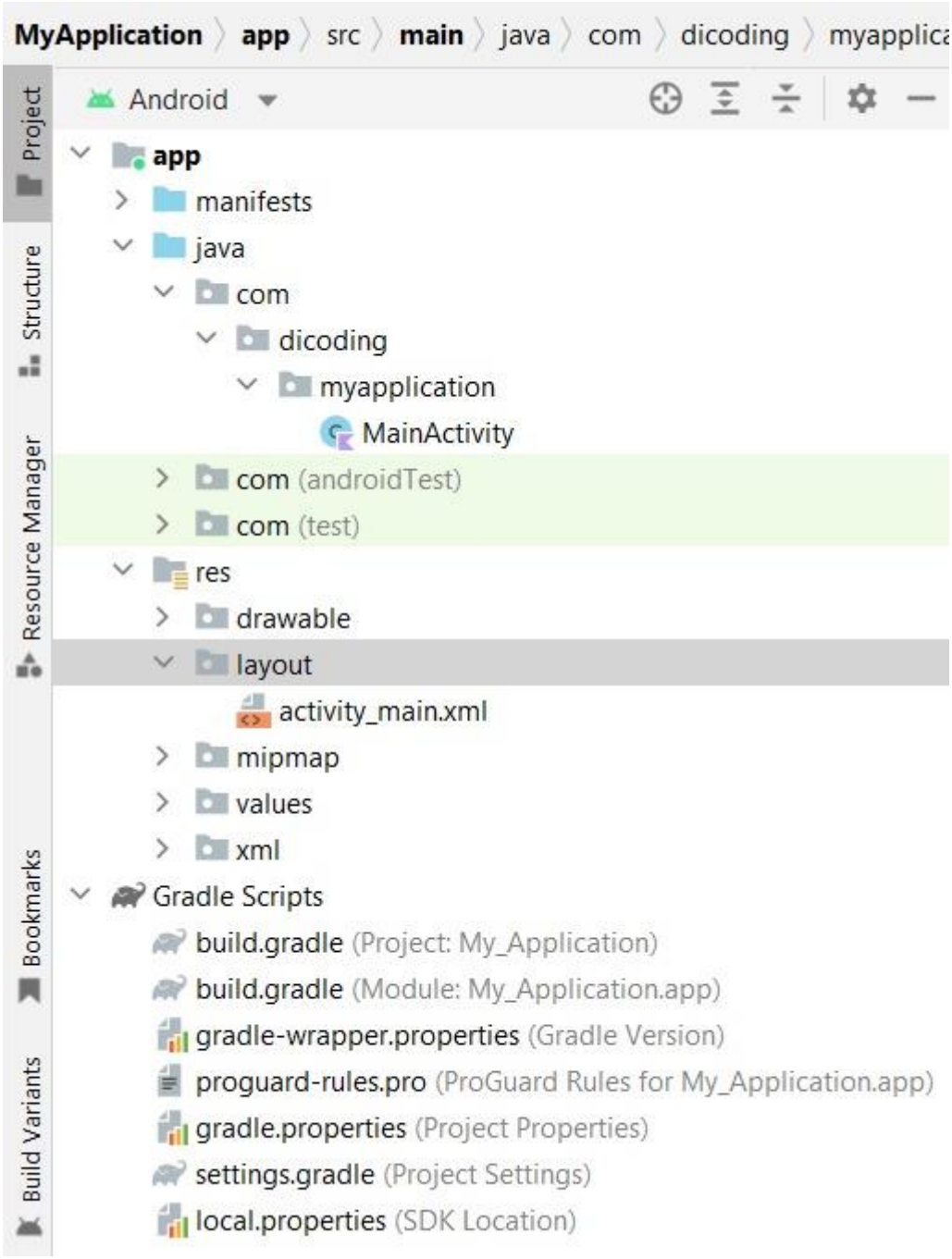
- **Package name**: digunakan sebagai identifikasi unik dari aplikasi ketika di-publish di PlayStore. Antara satu aplikasi dengan aplikasi lainnya harus berbeda. Biasanya penamaan package menggunakan format domain, tetapi urutannya dibalik.
- **Minimum SDK**: digunakan untuk menentukan versi minimum Android yang dapat menjalankan project ini. Anda dapat membuka **Help me choose** pada saat membuat project untuk melihat distribusi pemakai Android pada setiap versi sehingga bisa menentukan minimum SDK yang cocok.
- Saat versi baru Android dirilis, developer hendaknya terus mengikuti best-practice terbaru untuk memastikan aplikasi yang dibuat tetap memberikan pengalaman yang terbaik di sebanyak mungkin device. Untuk mencari tahu tentang versi terbaru, Anda dapat melihatnya pada tautan berikut.
    - What is API Level?

- Berikut adalah bagian-bagian dari **antarmuka Android Studio**.



  - **Toolbar**: merupakan tools yang sering digunakan dalam development, mulai dari copy/paste, build, menjalankan aplikasi, hingga menjalankan emulator.
  - **Navigation bar**: membantu untuk melihat struktur dari kedalaman (*depth*) dan posisi proyek yang saat ini sedang dibuka.
  - **Project Explorer dan Editor**: bagian utama dari IDE untuk melihat struktur project dan menulis kode
  - **Tool Window Bar**: menu yang mengelilingi editor ini merupakan button yang dapat di-expand atau menampilkan tools secara detail dan individual.
  - **Status Bar**: terletak di bagian paling bawah dalam Android Studio, ia berfungsi untuk menampilkan status proyek dan pesan peringatan (warning message) bila ada.

- Berikut adalah struktur project di dalam Android Studio.



- **AndroidManifest** merupakan file XML yang memberikan beragam informasi penting kepada sistem Android sebelum aplikasi dijalankan.
    - Tag **application** berfungsi untuk mendeskripsikan komponen dasar aplikasi Android, mulai dari activity, services, broadcast receiver, dan content provider.
    - Tag **uses-permission** berfungsi untuk menentukan izin yang harus dimiliki oleh aplikasi untuk mengakses sistem, seperti internet, external storage, contact, lokasi, dan lain sebagainya.

- **Folder java** dengan folder sesuai nama package digunakan untuk meletakkan berkas source code yang ditulis dalam bahasa Kotlin/Java, termasuk juga source set test untuk Unit Test dan androidTest untuk Instrumentation Test.
- **Folder res** digunakan untuk mengatur *resource* di dalam aplikasi seperti berikut.
    - **drawable**: untuk menyimpan berkas gambar maupun ikon.
    - **layout**: untuk menyimpan berkas desain aplikasi yang berupa XML.
    - **mipmap**: untuk menyimpan logo dalam berbagai dimensi.
    - **values**: berisi berbagai macam sumber data, seperti **colors.xml** untuk warna, **strings.xml** untuk teks, **dimens.xml** untuk ukuran, dan **themes.xml** untuk membuat theme dan style.
    - Selain itu, juga ada jenis folder resource yang bisa Anda tambahkan seperti yang ada pada tautan berikut.
        - App resources overview
- **Gradle** merupakan open source build automation system. Automation system berguna untuk mengotomatisasi proses pembuatan dari software build dan proses-proses terkait lainnya termasuk compile source code menjadi binary code, packaging binary code, dan menjalankan automated test.
- Ada dua jenis gradle script yang biasa Anda ubah ketika membangun aplikasi.
    - **build.gradle Level Project**: merupakan software build tingkat teratas yang meliputi keseluruhan dari proyek dari sebuah aplikasi. Di dalamnya berisi konfigurasi library Android dan Kotlin untuk semua module.
    - **build.gradle Level Module**: merupakan software build yang ada pada setiap module. Beberapa konfigurasi yang diedit di antaranya adalah android settings, defaultConfig dan productFlavors, buildTypes, dan dependencies.
- Berikut adalah beberapa detail konfigurasi yang dapat diubah pada gradle script level modul.
    - **namespace**: package name unik yang digunakan untuk Play Store, sama seperti yang kita atur pada saat membuat project.
    - **compileSdk**: merupakan versi SDK yang digunakan untuk meng-*compile* project.
    - **applicationId**: biasanya sama dengan namespace, nilainya bisa berubah untuk kebutuhan build variant.
    - **minSdk**: merupakan versi SDK minimal yang didukung oleh project ini. Android dengan versi di bawahnya tidak dapat menjalankan aplikasi ini.
    - **targetSdk**: target versi SDK yang menandakan aplikasi ini sudah dites pada versi SDK tertentu. *Best practice*-nya adalah memilih SDK yang terbaru. Jika tidak didefinisikan, nilainya sama dengan minSdk.

    - **versionCode**: nilai integer yang menandakan versi dari aplikasi. Apabila aplikasi sudah di-publish di PlayStore dengan versionCode 1, kita perlu mengubah versionCode menjadi 2 (incremental satu tingkat) ketika ingin meng-update aplikasinya lagi. Jika tidak diubah, PlayStore akan menolak APK yang di-upload.
    - **versionName**: versi aplikasi berupa String yang biasa ditunjukkan ke user, misalnya 1.0.1.
    - **buildTypes**: di dalamnya terdapat properties dari debuggable, ProGuard enabling, debug signing, version name suffix dan test information.
    - **dependencies**: di dalamnya terdapat informasi tentang library yang digunakan oleh aplikasi.
- Berikut adalah beberapa shortcut yang sering digunakan pada Android Studio.
    - **Shift+Shift**: pencarian semua jenis berkas yang masih dalam 1 proyek.
    - **Ctrl+Shift+F**: pencarian teks di seluruh berkas proyek.
    - **Ctrl+D**: menggandakan bagian yang dipilih.
    - **Ctrl+Q**: melihat dokumentasi dengan tampilan minimal.
    - **Alt+Enter**: melihat solusi pada kode yang error.
    - **Ctrl+Alt+L**: memformat ulang atau merapikan kode.
    - **Shift+F10**: menjalankan aplikasi ke emulator atau devices.

Shortcut lainnya dapat dilihat pada tautan berikut.

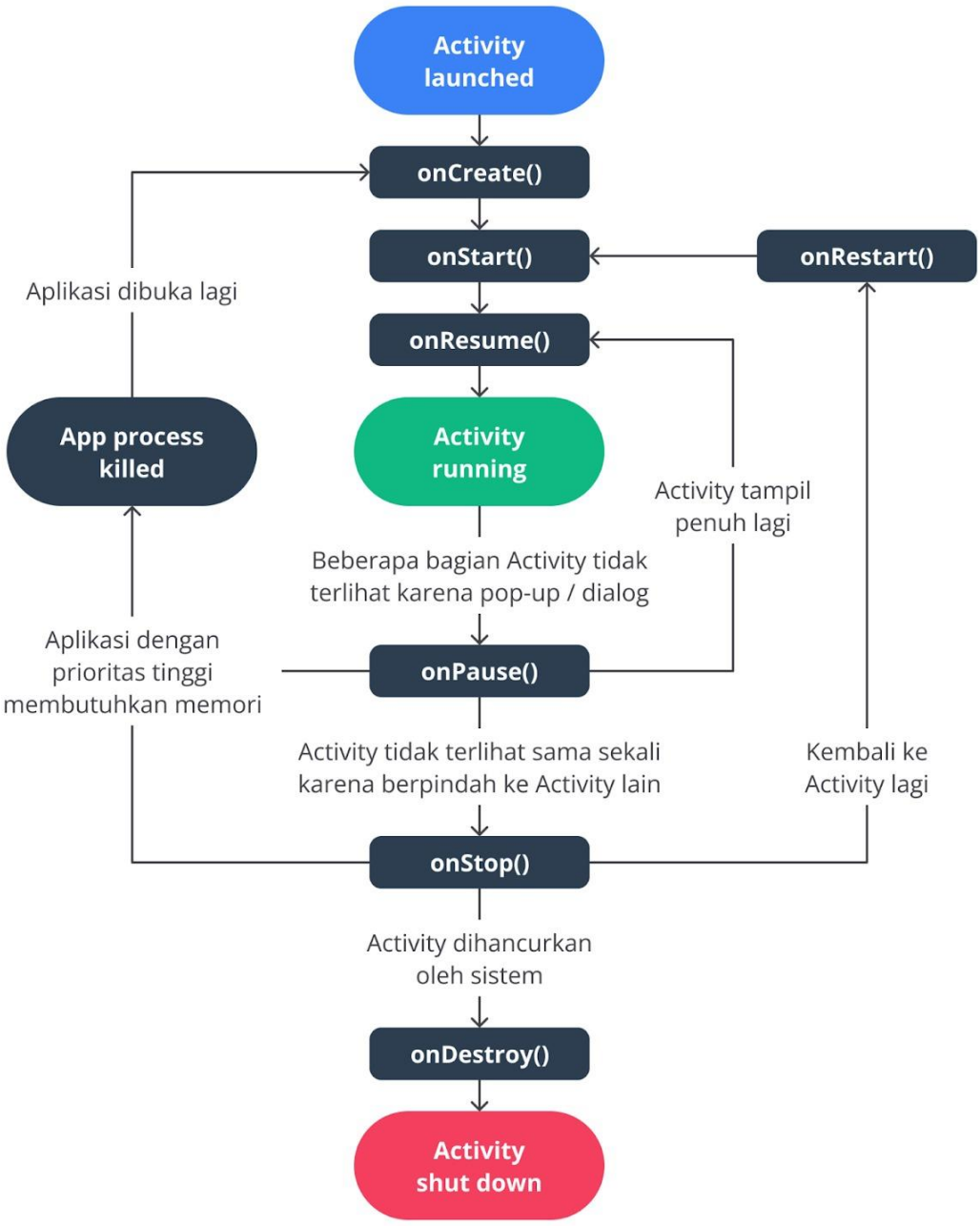
- Untuk meminimalisir salah ketik (typo) dalam pemanggilan class, method hingga variabel sebaiknya memanfaatkan Code Completion di Android Studio. Terdapat dua jenis code completion yang sering digunakan di Android Studio.
    - **Basic Code Completion (Ctrl+Space)**: pemanggilan code completion standar untuk membantu kita melengkapi kode.
    - **Statement Completion (Ctrl+Shift+Enter)**: perintah ini sangat membantu karena kita bisa menyelesaikan kode tanpa harus mengetik lengkap dan tanpa tanda kurung, kurung siku, kurung kurawal, serta banyak macam pemformatan lainnya.
- **Emulator** adalah tool yang menyimulasikan OS Android di dalam komputer.
- Berikut adalah beberapa alternatif untuk menjalankan project Android.
    - Android Virtual Device (AVD), emulator bawaan dari Android Studio.
    - Emulator ketiga (Memu, Nox, Genymotion, Bluestack, dll).
    - Physical device (perlu mengaktifkan Developer Mode).
- Berikut adalah langkah-langkah untuk mengaktifkan **Developer Mode** pada device.

    - Settings → About Phone → Tekan Build Number 7 kali.
    - Settings → Developer Options → Aktifkan USB debugging.
    - Hubungkan device dengan komputer via USB.
- Ada dua jenis **format** yang perlu Anda ketahui untuk memublikasikan aplikasi.
    - **APK** (Android Package): merupakan berkas executable yang bisa langsung dijalankan di dalam OS Android.
    - **AAB** (Android App Bundle): merupakan berkas yang didistribusikan oleh Google Play ke pengguna. Dengan format ini, ukuran aplikasi bisa menjadi jauh lebih kecil karena ia hanya mengunduh bagian (seperti bahasa, arsitektur, dan density) yang diperlukan saja.
- Dari sisi kredibilitas, ada 2 macam berkas APK/AAB yang dapat Anda buat.
    - **Unsigned APK/AAB**: merupakan berkas yang digunakan untuk pengujian saja.
    - **Signed APK/AAB**: merupakan berkas yang digunakan untuk diupload ke PlayStore. Bedanya yaitu memerlukan keystore.
- Berikut adalah cara untuk membuat APK/AAB versi rilis.
    - Klik Build → Generate Signed Bundle / APK....
    - Selanjutnya, pilih Android App Bundle dan klik Next.
    - Pilih atau buat keystore.
    - Isi data dan klik Next.
    - Selanjutnya pilih release dan klik Finish.
    - Tunggus proses selesai. Ketika berhasil, notifikasi akan tampil.
- **Keystore** adalah sebuah berkas biner yang berisi informasi tentang satu atau lebih private key untuk mencegah pemalsuan aplikasi. Berkas penting ini harus dijaga dan disimpan karena dibutuhkan setiap kali membuat Signed AAB untuk meng-update aplikasi.

Activity

- **Activity** merupakan komponen penting Android yang berfungsi untuk menampilkan user interface ke layar pengguna dan mengelola interaksi yang ada di dalamnya.
- Setiap Activity harus terdaftar di **AndroidManifest.xml** supaya tidak terjadi error.

- Lifecycle merupakan urutan state yang menunjukkan posisi proses aplikasi mulai dari Activity diciptakan sampai dihancurkan. Berikut adalah lifecycle dari sebuah Activity.



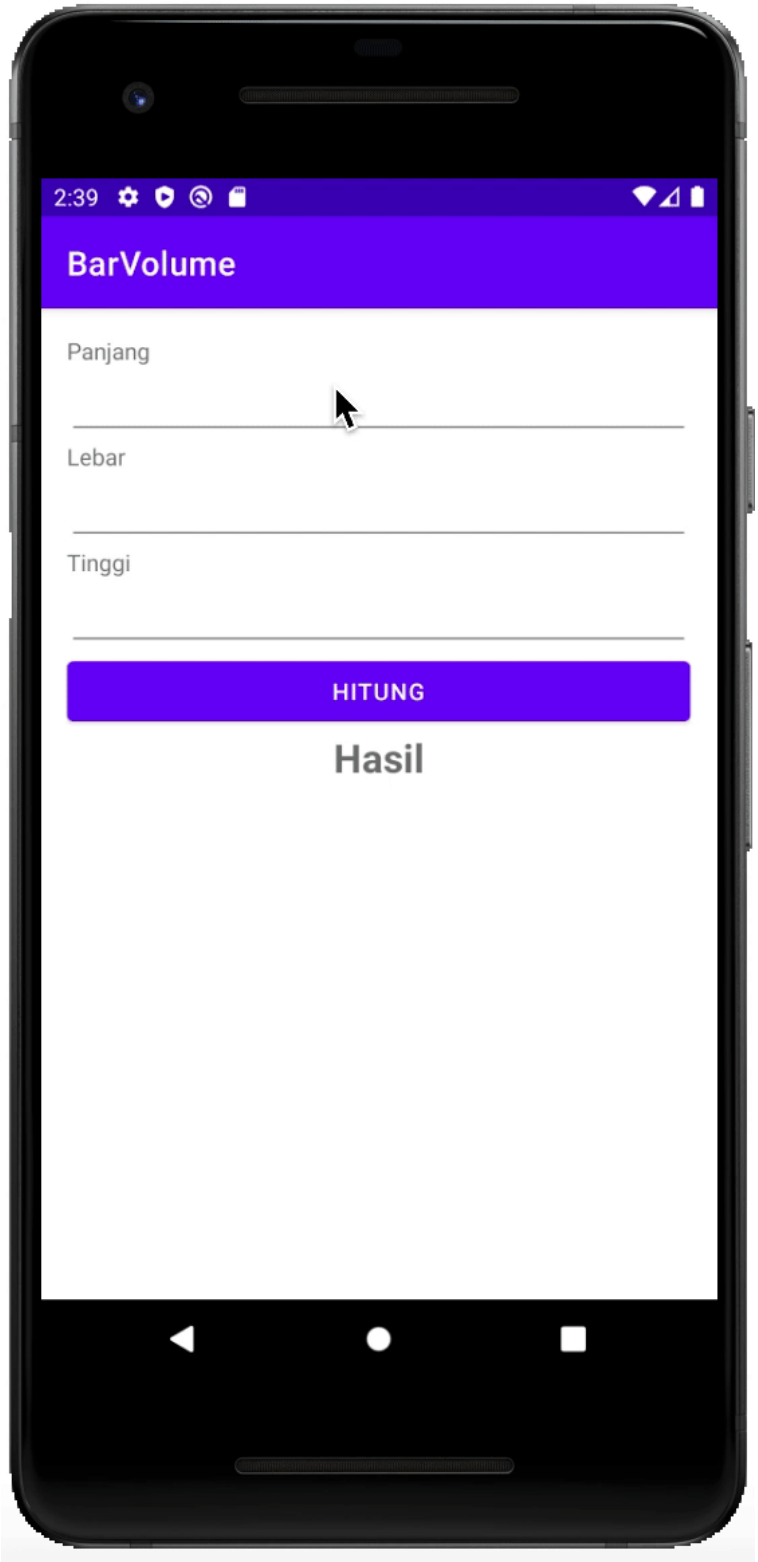
- Anda dapat menambahkan kode pada setiap setiap state lifecycle dengan override fungsi yang ada pada Activity.
- Berikut adalah contoh kode untuk mengatur logika berupa perhitungan volume balok di dalam Activity.

```
1. private lateinit var edtWidth: EditText
2. private lateinit var edtHeight: EditText
```

```
3. private lateinit var edtLength: EditText
4. private lateinit var btnCalculate: Button
5. private lateinit var tvResult: TextView
6.
7. override fun onCreate(savedInstanceState: Bundle?) {
8.     super.onCreate(savedInstanceState)
9.     setContentView(R.layout.activity_main)
10.
11.         edtWidth = findViewById(R.id.edt_width)
12.         edtHeight = findViewById(R.id.edt_height)
13.         edtLength = findViewById(R.id.edt_length)
14.         btnCalculate = findViewById(R.id.btn_calculate)
15.         tvResult = findViewById(R.id.tv_result)
16.
17.         btnCalculate.setOnClickListener {
18.             val inputLength =
   edtLength.text.toString().trim()
19.             val inputWidth = edtWidth.text.toString().trim()
20.             val inputHeight =
   edtHeight.text.toString().trim()
21.
22.             val volume = inputLength.toDouble() *
   inputWidth.toDouble() * inputHeight.toDouble()
23.             tvResult.text = volume.toString()
24.         }
25.     }
```

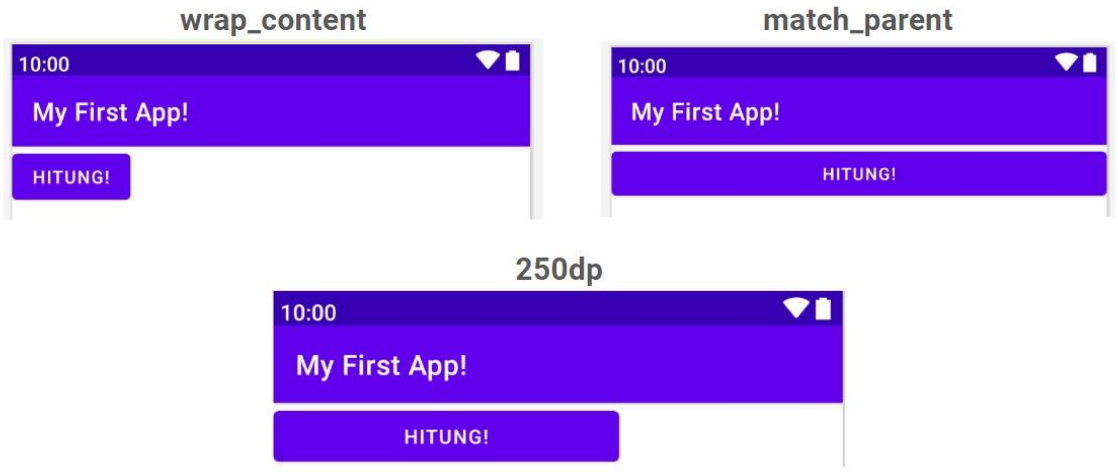
- **onCreate()** merupakan fungsi yang pertama kali dipanggil ketika membuka sebuah Activity.
- **setContentView** digunakan untuk menampilkan XML pada sebuah Activity.
- **findViewById** digunakan untuk menginisialisasi variabel dengan komponen View yang ada di XML sesuai dengan id-nya.
- **setOnClickListener** digunakan untuk melakukan aksi tertentu ketika suatu komponen ditekan.
- Untuk mengambil suatu teks dari EditText, Anda bisa menggunakan fungsi **getText().toString()** (text.toString pada Kotlin).
- Untuk mengatur teks suatu TextView, Anda bisa menggunakan fungsi **setText** (.text pada Kotlin).

- Berikut adalah hasil jadi dari contoh penerapan logika di atas.



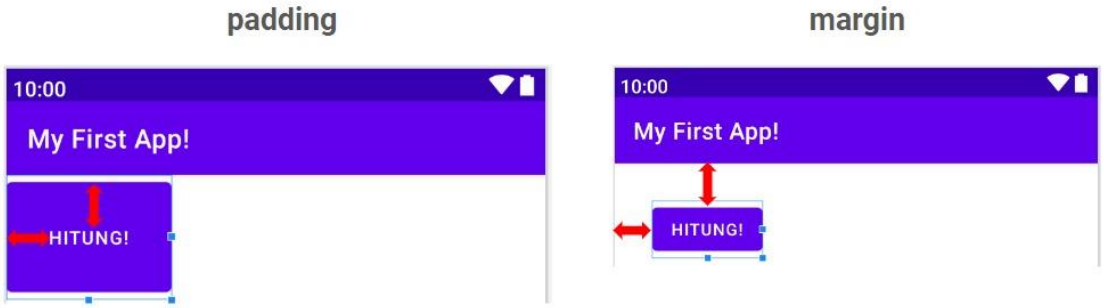
- **onSaveInstanceState** digunakan untuk mempertahankan data ketika terjadi configuration change.
- **Configuration change** terjadi karena perubahan yang signifikan pada aplikasi sehingga akan membuat Activity dihancurkan dan dibuat lagi. Contohnya seperti perubahan orientasi, perubahan bahasa, dan ukuran layar.

- Dokumentasi terkait Activity dapat dilihat pada tautan berikut.
    - Activity: Documentation
- Perhatikan bahwa dalam penulisan kode XML, ada dua cara dalam penulisan tag seperti berikut.
    - **Self-closing tag**: tag diawali dengan < dan diakhiri dengan />. Biasanya digunakan untuk View tanpa isi.
    - **Opening tag** dan **closing tag**: opening tag diawali dengan < dan diakhiri dengan > saja. Sebagai gantinya, ada closing tag dengan format **</NamaView>**. Biasanya digunakan untuk layout yang menampung View lain di dalamnya.
- DI dalam sebuah tag View, Anda bisa mengubah beberapa attribute dengan beberapa format berikut.
    - **android:<property_name>="@+id/view_id"** untuk penulisan id.
    - **android:<property_name>="<property_value>"** untuk attribute biasa.
    - **android:<property_name>="@<resource_type>/resource_id"** untuk attribute yang memanggil value dari folder res, seperti string, color, dan dimens.
- Penulisan teks secara langsung (*hardcoded*) merupakan praktik yang kurang baik karena seharusnya kita menuliskan semua teks pada berkas **res/values/strings.xml** dan setelah itu baru memanggilnya.
- Dalam menentukan tinggi dan lebar suatu View, terdapat beberapa alternatif value yang bisa digunakan seperti berikut.

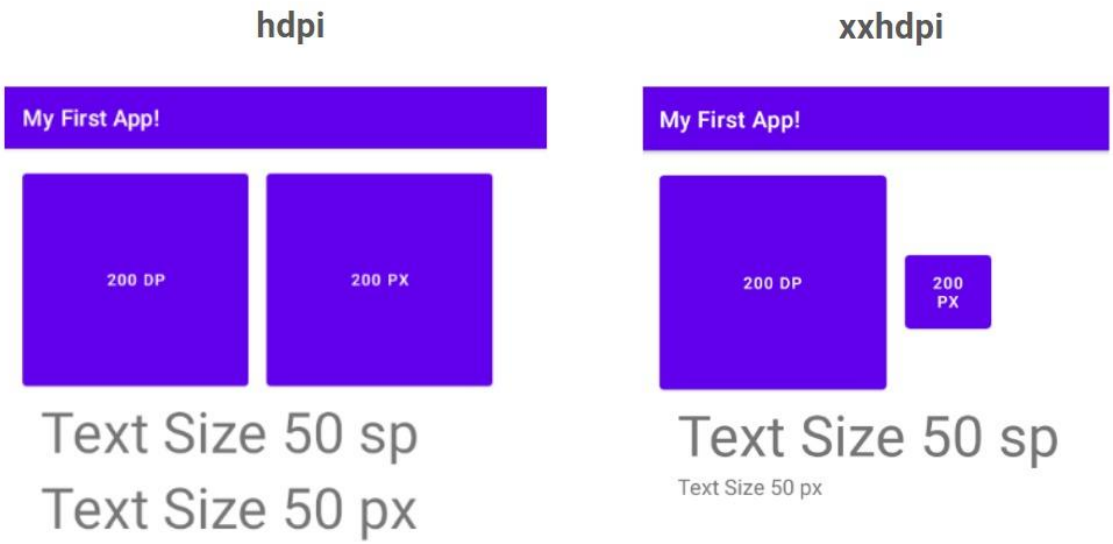


    - **wrap_content**: ukuran menyesuaikan dengan ukuran konten di dalamnya.
    - **match_parent**: ukuran menyesuaikan dengan ukuran parent (View induknya). Apabila di paling luar, berarti mengikuti ukuran layar device-nya.
    - **fixed size**: Anda bisa menentukan ukuran dengan nilai tetap dengan satuan dp.

- Berikut beberapa alternatif untuk memberikan jarak antar View.



- **Padding**: jarak dari isi ke tepi komponen (dalam).
- **Margin**: jarak dari tepi komponen ke komponen lain (luar).
- Berikut beberapa satuan ukuran yang bisa Anda gunakan dalam mendesain menggunakan XML.

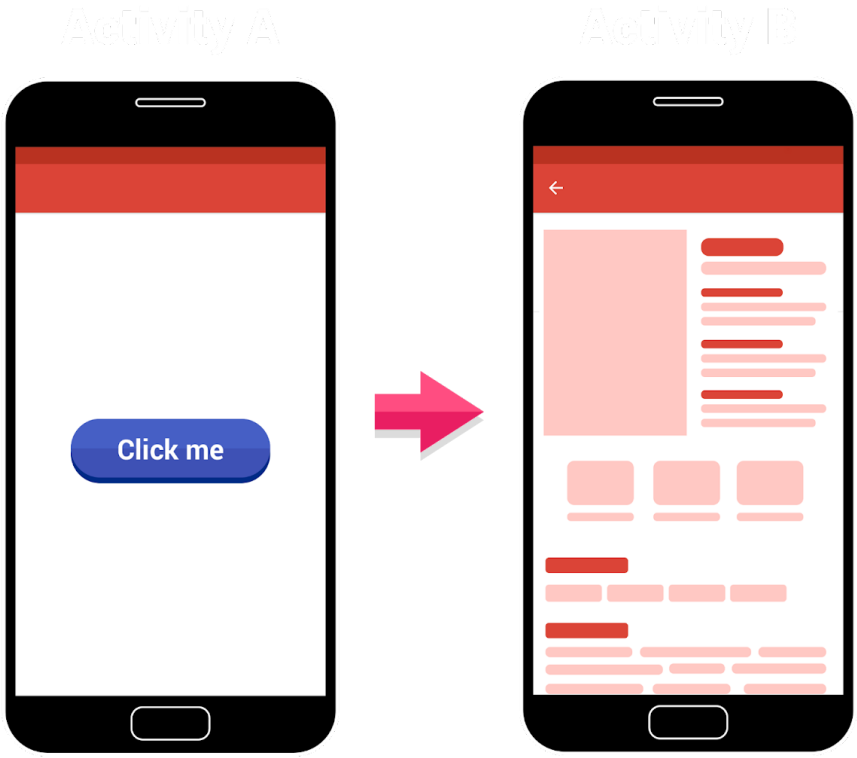


  - **sp** (scale independent pixel): digunakan untuk ukuran teks, tetapi diskalakan berdasarkan preferensi ukuran font pengguna.
  - **dp** (density independent pixel): digunakan untuk semuanya selain ukuran teks.
  - **px**: ukuran piksel sebenarnya di layar. Penggunaan px tidak disarankan karena dapat menghasilkan ukuran yang berbeda.

Intent

- **Intent** adalah mekanisme untuk melakukan sebuah action dan komunikasi antar komponen aplikasi misal Activity, Service, dan Broadcast Receiver.
- Intent memiliki dua bentuk yaitu:

- **Explicit Intent**: digunakan untuk menjalankan komponen lain dengan tujuan yang sudah jelas atau eksplisit. Biasanya digunakan untuk berpindah ke Activity lain pada satu aplikasi.



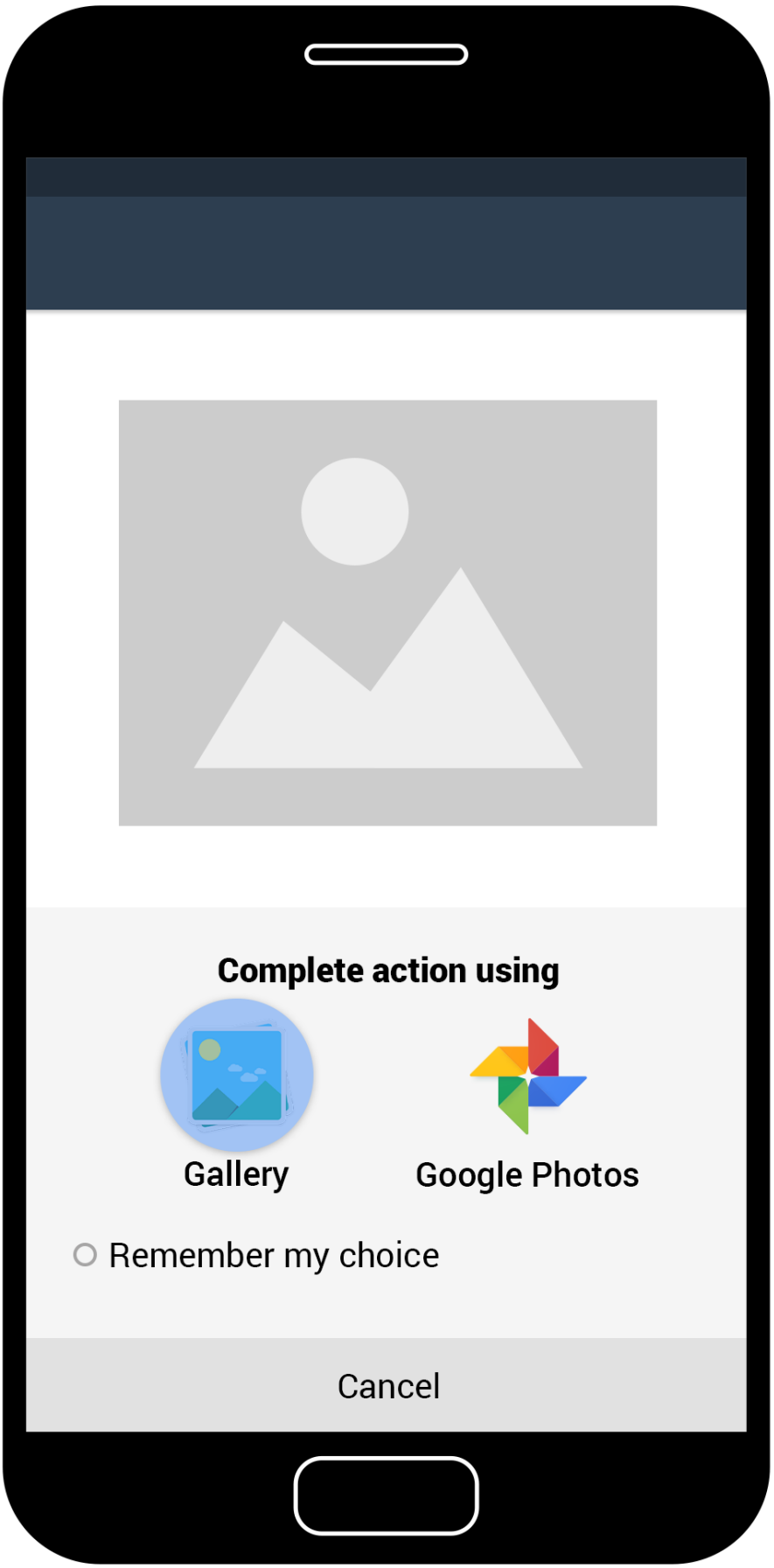
Berikut contoh kodenya.

```
val moveIntent = Intent(this@MainActivity,
DetailActivity::class.java)
startActivity(moveIntent)
```

Context adalah sebuah kelas yang digunakan untuk mengakses resource dari activity tersebut. Pada Activity, Anda bisa menggunakan this.

Implicit Intent: digunakan untuk menjalankan komponen lain dengan tujuan yang belum jelas alias implisit. Biasanya digunakan untuk melakukan aksi tertentu dengan berpindah ke aplikasi lain, misalnya membuka maps, camera, atau foto. Jika ada lebih dari satu aplikasi yang bisa menangani aksi tersebut, opsi aplikasi akan muncul. Contohnya ketika menambahkan gambar pada aplikasi tertentu, pilihannya bisa menggunakan Gallery atau

Google Photos.


Complete action using
Gallery
Google Photos
Remember my choice
Cancel

Berikut contoh kodenya.

val maps = Intent(Intent.ACTION_VIEW, Uri.parse("geo:-7.053948,110.4318891,z=15"))

startActivity(maps)

Berikut ini adalah contoh kode untuk mengirim dan menerima data

```
1. //mengirim data
2. val moveIntent = Intent(this@MainActivity,
   DetailActivity::class.java)
3. moveIntent.putExtra("extra_name", "DicodingAcademy Boy")
4. moveIntent.putExtra("extra_age", 5)
5. startActivity(moveIntent)
6.
7. //menerima data
8. val name = intent.getStringExtra("extra_name")
9. val age = intent.getIntExtra(EXTRA_AGE, 0) //untuk Int ada
   default value
```

- Fungsi **putExtra(key, value)** digunakan untuk menambahkan data di dalam Intent dengan bentuk *key-value*.
- Fungsi **intent.getFooExtra(key)** digunakan untuk mengambil data berdasarkan key. Dengan “Foo” adalah tipe data dari dari data yang dikirim.
- Pembuatan **konstanta** diperlukan supaya tidak salah dalam menuliskan key.
- **Parcelable** adalah suatu interface yang memungkinkan kita melakukan pengiriman satu objek sekaligus di dalam Intent. Berikut contohnya dalam bahasa Java.

```
• public class Person implements Parcelable {
• private String name;
• private int age;
• public Person(){
• }
• public String getName() {
• return name;
• }
• public void setName(String name) {
• this.name = name;
• }
• public int getAge() {
• return age;
• }
• public void setAge(int age) {
• this.age = age;
• }
• protected Person(Parcel in) {
• name = in.readString();
• age = in.readInt();
• }
```

```
@Override
public void writeToParcel(Parcel dest, int flags) {
dest.writeString(name);
dest.writeInt(age);
}
@Override
public int describeContents() {
return 0;
}
public static final Creator<Person> CREATOR = new
Creator<Person>() {
@Override
public Person createFromParcel(Parcel in) {
return new Person(in);
}
@Override
public Person[] newArray(int size) {
return new Person[size];
}
};
```

Parcelize adalah plugin khusus untuk bahasa Kotlin yang dapat digunakan untuk membuat implementasi Parcelable secara otomatis. Berikut contoh kodenya.

DataClass

```
…
import kotlinx.parcelize.Parcelize
@Parcelize
data class Person(
    val name: String?,
    val age: Int?,
) : Parcelable
```

build.gradle.kts (module

```
plugins {
    alias(libs.plugins.androidApplication)
    alias(libs.plugins.jetbrainsKotlinAndroid)
    id("kotlin-parcelize")
}
```

6.

7. ...

Selain mengirimkan data, Anda pun bisa mendapatkan data kembalian dari Activity yang dibuka. Berikut contoh kodenya.

```
1. // Activity A
2. private val resultLauncher: ActivityResultLauncher<Intent> =
   registerForActivityResult(
3.     ActivityResultContracts.StartActivityForResult()
4. ) { result ->
5.     if (result.resultCode == MoveForResultActivity.RESULT_CODE && result.data
   != null) {
6.         val selectedValue =
7.             result.data?.getIntExtra(MoveForResultActivity.EXTRA_SELECTED_VALU
   E, 0)
8.         tvResult.text = "Hasil : $selectedValue"
9.     }
10. }
11. …
12. val moveForResultIntent = Intent(this@MainActivity,
    MoveForResultActivity::class.java)
13. resultLauncher.launch(moveForResultIntent)
14.
15. // Activity B
16. val resultIntent = Intent()
17. resultIntent.putExtra(EXTRA_SELECTED_VALUE, value)
18. setResult(RESULT_CODE, resultIntent)
19. finish()
```

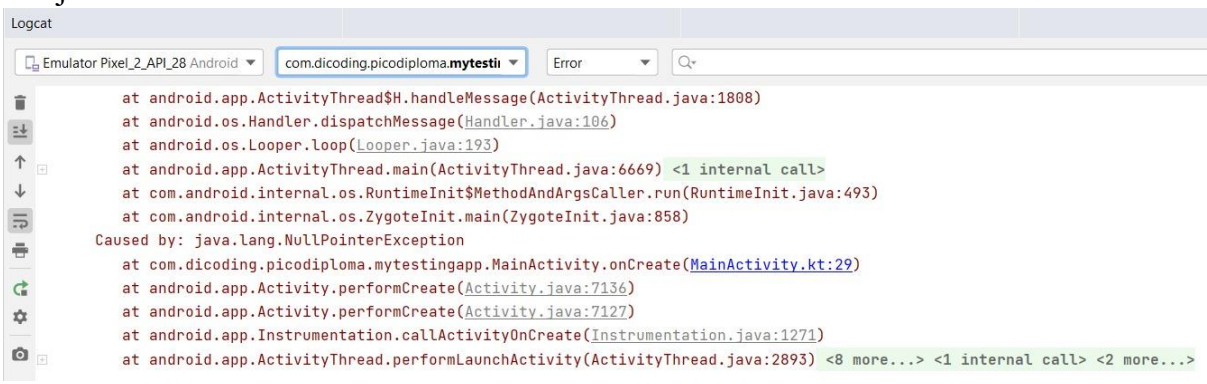
- Fungsi **setResult** digunakan untuk mengirimkan nilai balik ke Activity yang memanggilnya.
- Fungsi **registerForActivityResult** digunakan untuk menerima data yang dikirimkan oleh setResult dari Activity lain. Fungsi ini mengembalikan objek **ActivityResultLauncher** yang digunakan untuk menjalankan Activity lain.
- Fungsi **finish()** digunakan untuk menutup/menghancurkan Activity dan kembali ke Activity sebelumnya.

- Dokumentasi terkait Intent dapat dilihat pada tautan berikut.
    - Intent
    - Intent Filter

Debugging

Debugging

- **Debugging** adalah proses untuk mengidentifikasi bug atau masalah yang muncul dan mencari solusinya.
- Apabila terjadi error atau force close, berikut hal yang perlu Anda lakukan.
    - Pastikan peranti masih terhubung dengan Android Studio.
    - Lihat bagian Logcat yang ada di bagian bawah Android Studio. Scroll ke atas sampai bertemu teks berwarna merah. Anda juga bisa mengubah jenis log menjadi **Error** untuk memfilter data.



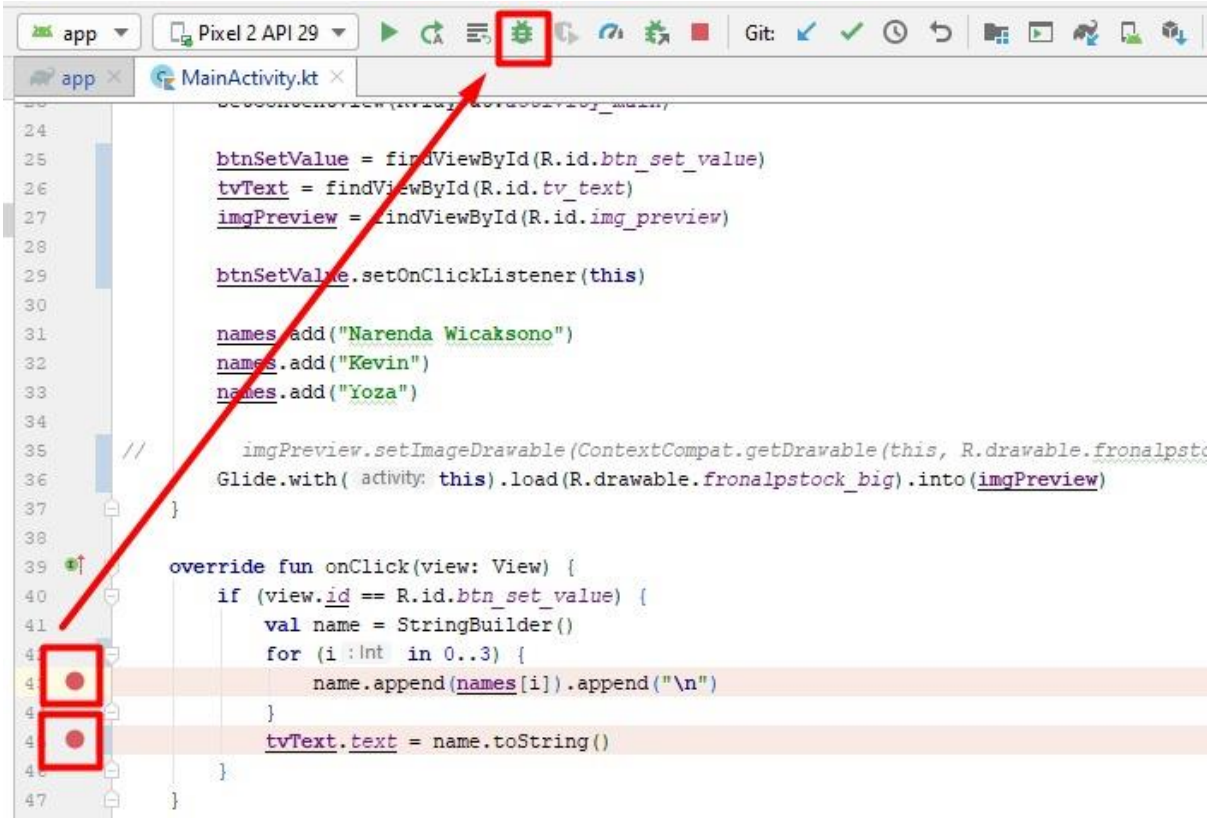
    - Pahami pesan error yang dimaksud pada awal baris dan penyebabnya pada kata "**Caused by: …**". Pada contoh di atas, penyebabnya adalah NullPointerException.
    - Seringnya Logcat akan memberi tahu baris kode yang error dengan warna biru yang bisa Anda tekan untuk menuju ke kode yang bermasalah. Pada contoh di atas, error terjadi pada MainActivity baris 29.
    - Apabila Anda tidak paham dengan error yang dimaksud, gunakan mesin pencarian (googling) untuk mencari artinya.
    - Apabila masih tidak paham juga, silakan tanyakan ke forum diskusi, teman, atau komunitas.
    - Selengkapnya bisa dilihat di blog berikut.
        - Android Studio: Aplikasi Error! Apa yang harus dilakukan? - Dicoding Blog
- Terkadang aplikasi langsung force close tanpa menampilkan eror. Hal ini karena dia langsung restart app ketika terjadi force close, jadi tidak terlihat errornya. Apabila ini terjadi, buka kembali aplikasi dan buat aplikasi eror sampai berulang kali. Ketika pop up muncul, barulah nanti pesan eror akan muncul. Hal lain yang perlu diperhatikan

adalah pastikan Anda memilih emulator dan aplikasi yang tepat pada Logcat. Jangan sampai memilih aplikasi yang sudah *dead* padahal bisa dibuka.

- **Logging** adalah sebuah cara untuk menampilkan data di dalam Logcat. Berguna untuk mendeteksi nilai suatu object atau memverifikasi suatu fungsi. Berikut contohnya.

1. Log.d("MainActivity", "Data yang ingin ditampilkan")

- Berikut beberapa variasi log yang bisa Anda pilih.
    - **Log.e()** untuk log error.
    - **Log.w()** untuk log warning.
    - **Log.i()** untuk log information.
    - **Log.d()** untuk log debug.
    - **Log.v()** untuk log verbose.
- **Debug breakpoint** adalah fitur di dalam Android Studio untuk mendeteksi suatu nilai pada baris kode tertentu ketika aplikasi dijalankan. Caranya yaitu dengan memilih baris kode yang ingin diperiksa dan tekan tombol **Debug 'app' (Shift+F9)** pada toolbar.

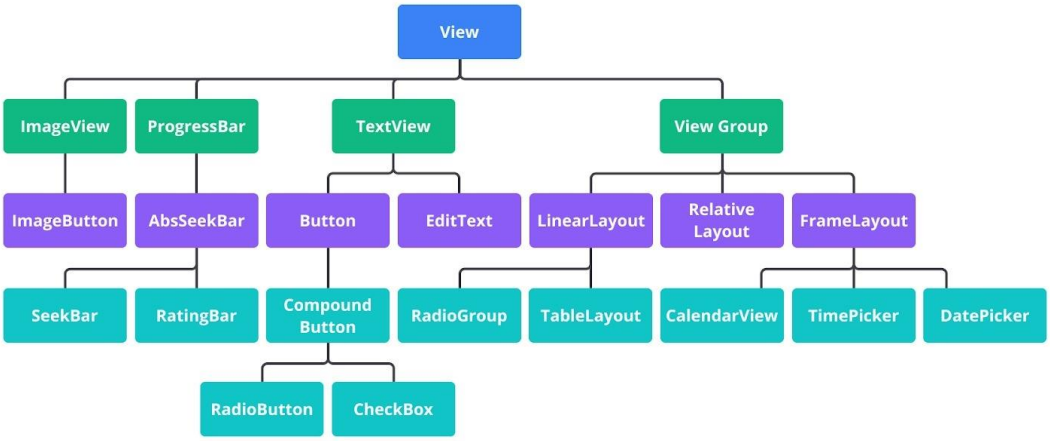


- Error **NullPointerException** muncul dikarenakan memanggil variabel yang masih bernilai null.
- Berikut dokumentasi terkait proses logging dan debugging.
    - Write and view logs

- Debug your app

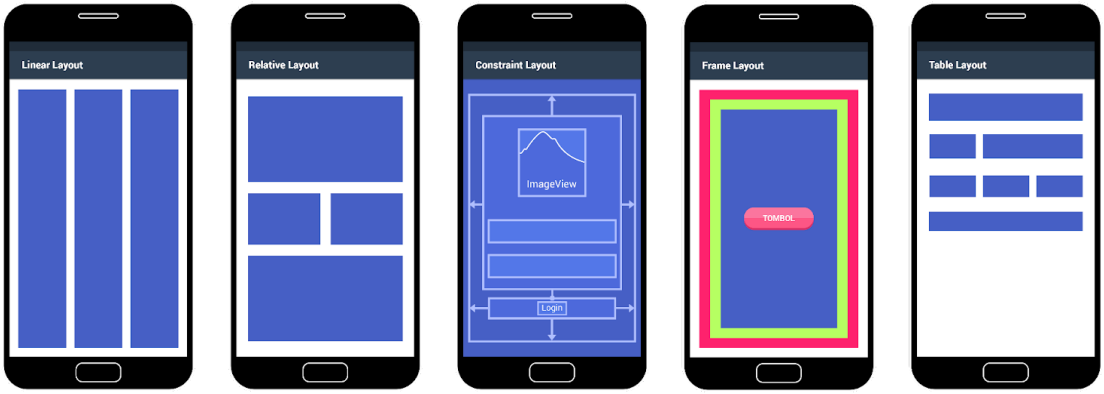
View dan ViewGroup

- Pada dasarnya, semua elemen UI di aplikasi Android dibangun menggunakan dua buah komponen inti, yaitu View dan ViewGroup.



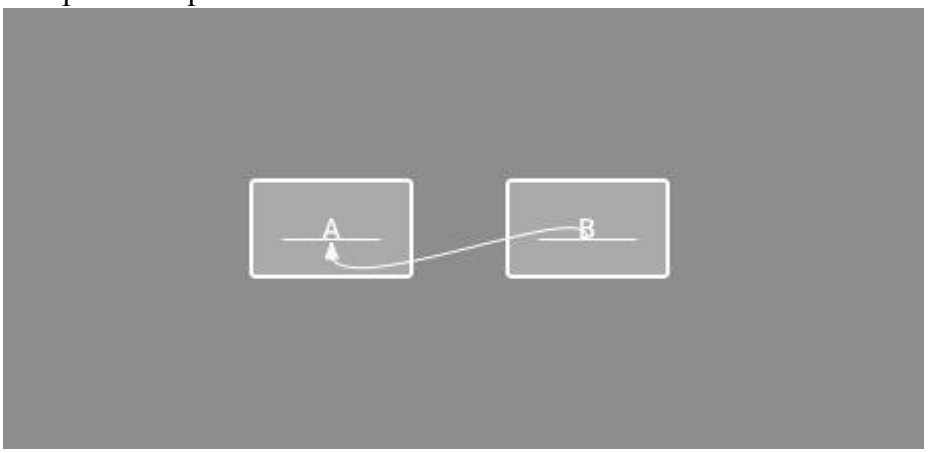
- **View** merupakan komponen dasar yang tampil di layar dan dapat digunakan untuk berinteraksi dengan pengguna. Contoh komponen turunan dari View adalah TextView, Button, ImageView, RadioButton, Checkbox, dll.

- **ViewGroup** adalah sebuah View spesial yang mewadahi objek-objek View lainnya dan berguna untuk mengatur posisinya.

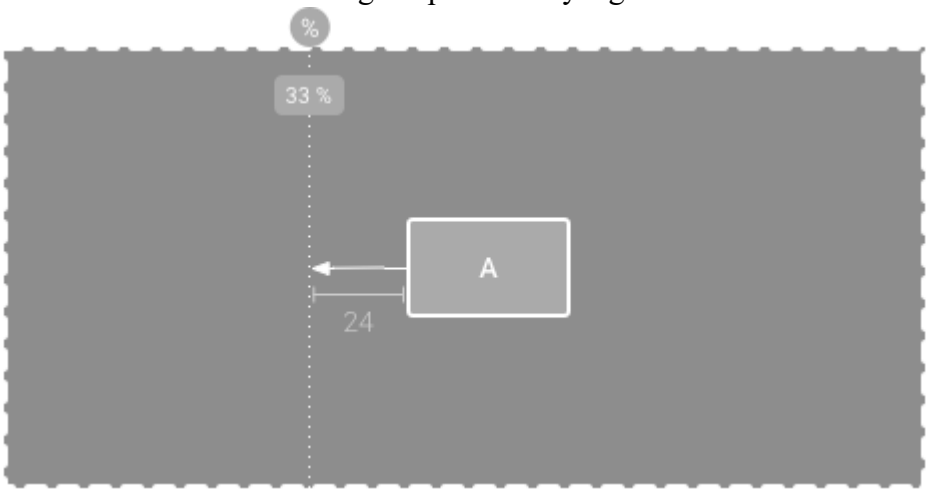


  - **LinearLayout** digunakan untuk menempatkan komponen-komponen di dalamnya secara horizontal atau vertikal.
  - **RelativeLayout** digunakan untuk menempatkan masing-masing komponen secara fleksibel dan relatif terhadap komponen yang lainnya. Misalnya komponen A di sebelah kanan komponen B atau komponen A rata tengah dengan parent-nya.
  - **FrameLayout** digunakan untuk membuat komponen menjadi menumpuk atau saling menutupi satu dengan yang lainnya.
  - **TableLayout** digunakan untuk menyusun komponen dalam baris dan kolom.
  - **ConstraintLayout** Merupakan layout default dan direkomendasikan di dalam XML. Dengan ConstraintLayout, Anda dapat menyusun tampilan yang kompleks cukup dengan satu lapis hierarki saja. Hal ini akan memberikan performa dan proses rendering yang lebih baik daripada penggunaan nested layout (layout di dalam layout).
- Berikut beberapa fitur pada ConstraintLayout.
  - **Relative Positioning**: memposisikan komponen secara relatif terhadap komponen yang lain.
  - **Center Positioning & Bias**: untuk menentukan alignment dengan menggunakan persentase, default-nya 50% atau tengah.

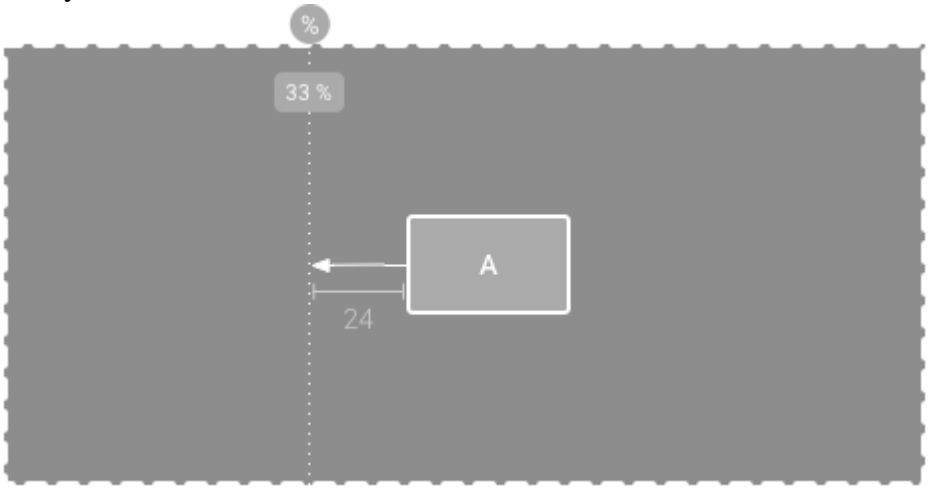
- **Baseline alignment**: untuk membuat text pada suatu komponen sejajar dengan teks pada komponen lain.



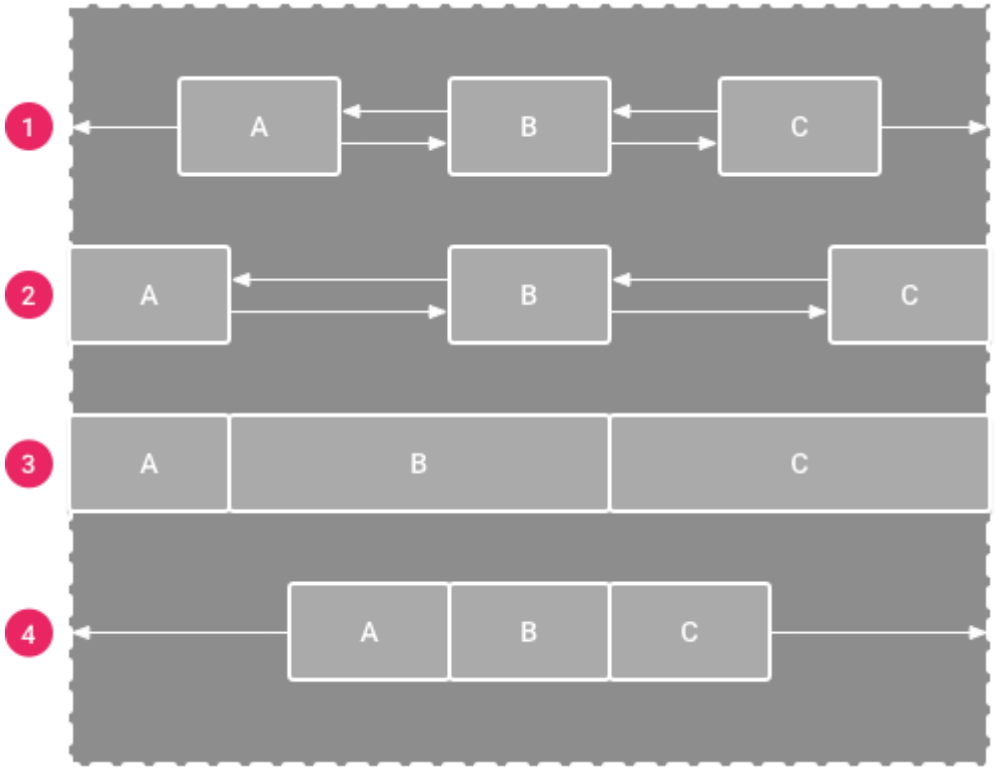
- **Guideline**: untuk membuat garis pembantu yang tidak terlihat oleh user.



- **Barrier**: sama seperti Guideline, tetapi posisinya dapat mengikuti komponen lainnya.

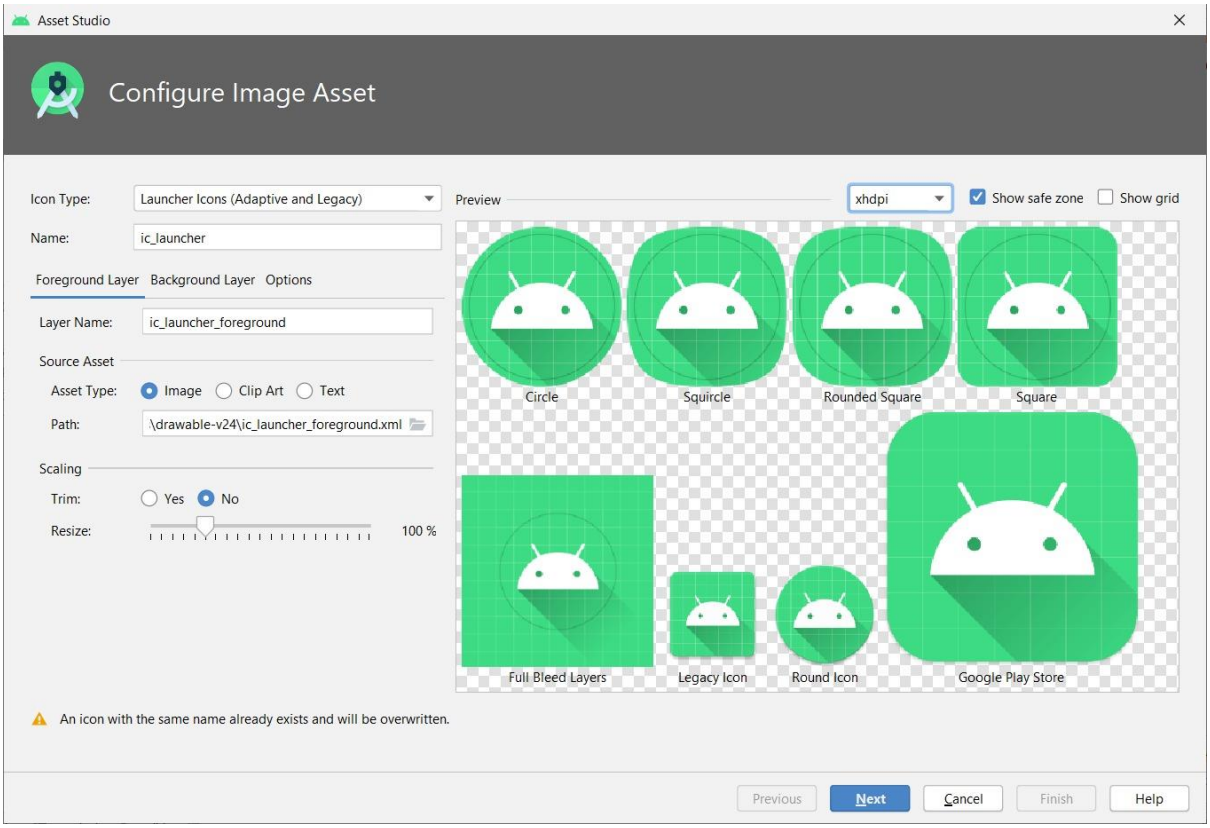


- **Chain**: mengatur sekumpulan komponen secara linear.



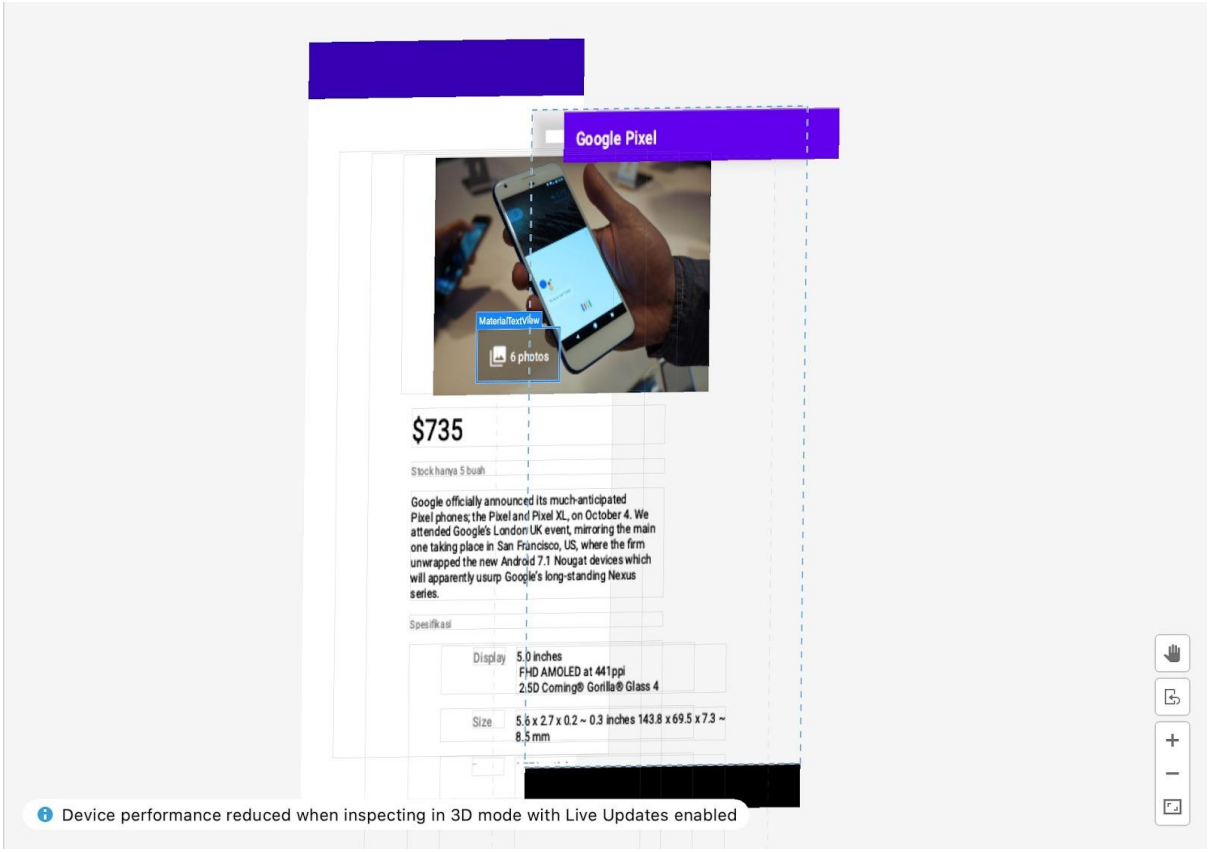
  - **Spread**: setiap elemen akan menyebar (default style).
  - **Spread inside**: tampilan pertama dan terakhir akan ditempelkan pada batasan di setiap ujung rantai, sedangkan sisanya akan didistribusikan secara merata.
  - **Weighted**: jika beberapa widget disetel ke "match constraint", mereka akan membagi ruang yang tersedia.
  - **Packed**: elemen akan menyatu dan dikemas bersama.

- **ScrollView** digunakan untuk membuat komponen di dalamnya dapat digeser (scroll) secara vertikal maupun horizontal. Di dalamnya hanya boleh ada satu layout ViewGroup, tidak boleh lebih.
- Untuk menambahkan gambar ke **drawable**, copy paste gambar dari explorer ke folder drawable. Pilih folder drawable (bukan drawable-v24) untuk mendukung semua versi Android.
- Ingat bahwa nama file yang ada di dalam resource drawable harus menggunakan **huruf kecil** dan **underscore** saja.
- Untuk mengubah **icon aplikasi**, klik kanan pada folder res dan pilih **New → Image Asset**.



Pilih Launcher Icons (Adaptive and Legacy) pada Icon Type, pilih gambar pada Path, lalu klik Next → Finish sehingga, icon akan tersimpan di folder mipmap dengan berbagai density (mdpi, hdpi, xhdpi, xxhdpi, dan xxxhdpi).

- **Layout Inspector** digunakan untuk menganalisis hierarki dan properties pada suatu tampilan.



- Penggunaan **Start** dan **End** pada margin daripada Right dan Left berfungsi supaya mendukung dua jenis bahasa, baik Left to Right (LTR) seperti bahasa Inggris maupun Right to Left (RTL) seperti bahasa Arab.

Style dan Theme

- **Style** merupakan sebuah kumpulan properti yang dibutuhkan untuk mendefinisikan bagaimana sebuah komponen View ataupun ViewGroup ditampilkan.
- Pemusatan style cocok digunakan untuk mengumpulkan atribute yang berulang-ulang digunakan di banyak komponen sehingga jika ada perubahan, cukup ubah di satu tempat saja.
- **Style** terdefinisi dalam file xml sendiri. Anda bisa menemukannya di **res → values → themes.xml**.
- Semua style yang dibuat harus berada dalam tag resources. Setiap style dibuat menggunakan tag style dan tag item untuk mendefinisikan property-nya.

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <resources>
3.     <style name="CodeFont"
   parent="@android:style/TextAppearance.Medium">
4.         <item name="android:layout_width">match_parent</item>
```

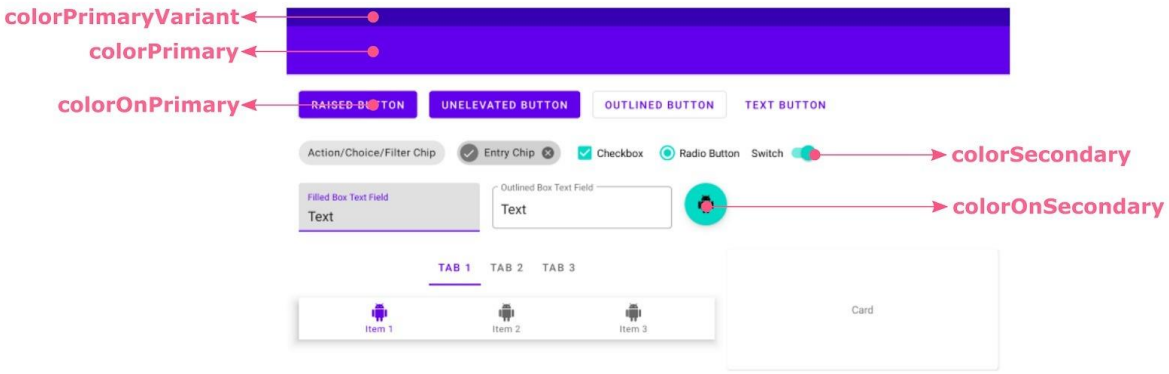
```
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textcolor">#00FF00</item>
        <item name="android:typeface">monospace</item>
    </style>
</resources>
```

- Untuk memanggil style pada XML, gunakan property style = “@style/NamaStyle”.

```
<TextView
    style="@style/CodeFont"
    android:text="@string/hello" />
```

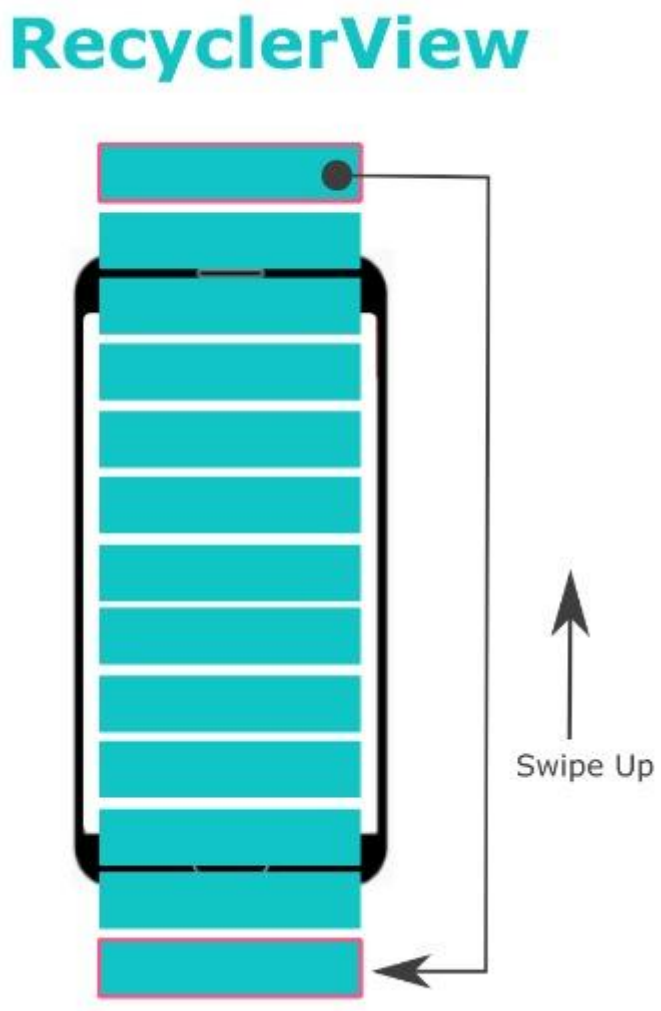
- **Theme** merupakan sebuah style yang diterapkan khusus untuk Activity dan Application pada berkas AndroidManifest.xml.
- **Dark Theme** dapat mengurangi konsumsi energi pada handphone dan membuat aplikasi lebih mudah dibaca walaupun cahaya sedikit.
- **Material Design** merupakan panduan desain yang dibuat oleh Google untuk membuat user interface dan user experience pada Android.
- Untuk membuat theme, gunakan tag style dengan parent berupa Theme, seperti “Theme.MaterialComponents.DayNight.DarkActionBar” yang merupakan tema dengan background putih dan action bar berwarna gelap.
- Beberapa atribut yang dapat dikustomisasi di dalamnya yaitu:
    - **colorPrimary**: warna utama aplikasi yang tampil pada Action Bar dan komponen Button.
    - **colorPrimaryVariant**: variasi dari warna utama yang biasanya digunakan pada Status Bar.
    - **colorOnPrimary**: warna yang digunakan ketika ada text/icon di atas warna primary.
    - **colorSecondary**: warna utama sekunder yang tampil pada Action Bar dan komponen EditText.
    - **colorSecondaryVariant**: variasi dari warna sekunder.
    - **colorOnSecondary**: warna yang digunakan ketika ada teks/icon di atas warna sekunder.

- **android:statusBarColor**: warna yang digunakan untuk status bar.



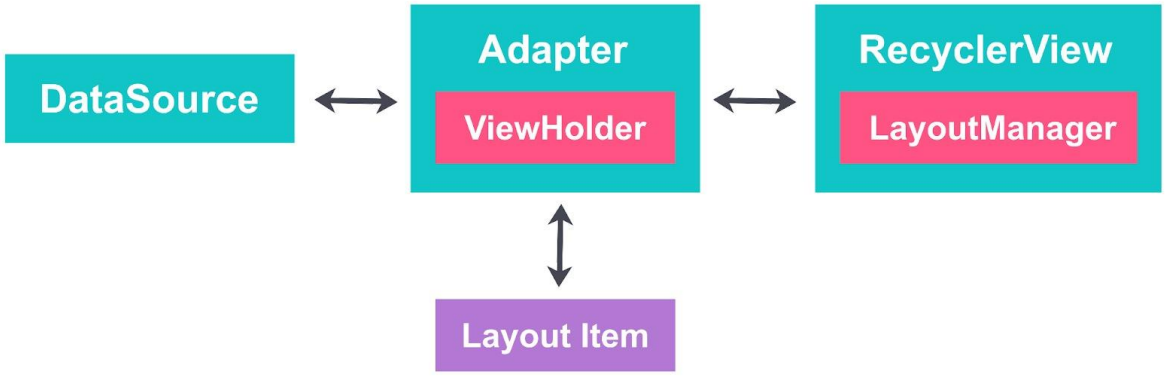
- Semua variabel warna sebaiknya dimasukkan di dalam berkas **colors.xml**.

RecyclerView

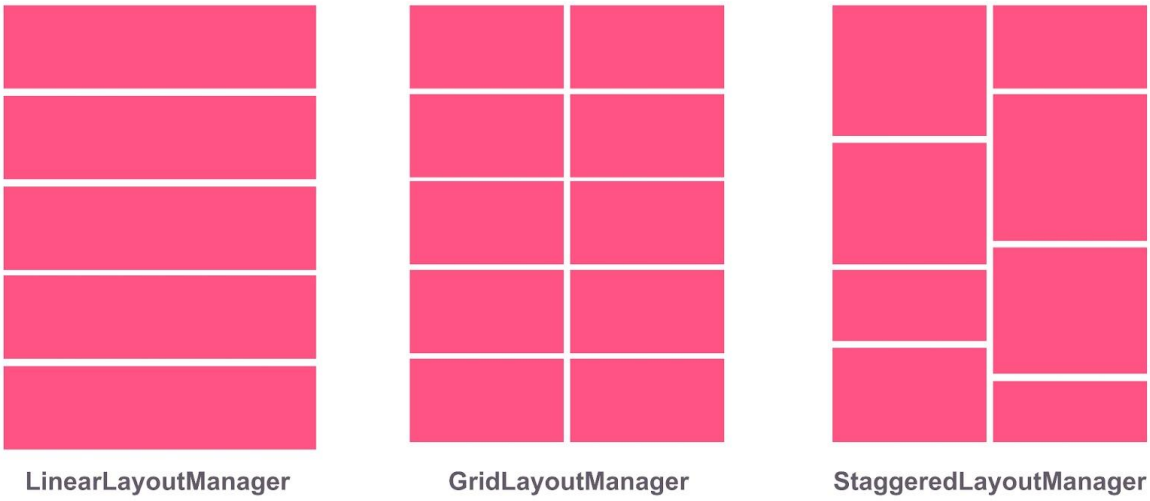


- **RecyclerView** merupakan komponen yang digunakan untuk menampilkan data dalam jumlah banyak secara dinamis.

- Dengan RecyclerView, objek yang dibuat hanya sebatas ukuran layar dan beberapa di atas dan di bawahnya saja. Selanjutnya, ia menggunakan kembali item yang sudah tidak terlihat sehingga penggunaan memori menjadi lebih efisien.
- Berikut adalah beberapa bagian yang perlu Anda ketahui untuk menampilkan data dengan RecyclerView.



  - **RecyclerView**: merupakan komponen ViewGroup yang perlu ditambahkan ke dalam berkas XML.
  - **LayoutManager**: digunakan untuk mengatur tata letak item di dalam RecyclerView. Berikut beberapa opsi yang bisa Anda pilih.



    - **LinearLayoutManager**: tampilan satu kolom secara linear.
    - **GridLayoutManager**: tampilan lebih dari satu kolom atau grid.
    - **StaggeredLayoutManager**: tampilan menyesuaikan tinggi setiap item.
  - **Data source**: sumber data yang akan ditampilkan ke dalam RecyclerView, biasanya berupa List/ArrayList.
  - **Layout item**: berkas tampilan XML untuk setiap baris item.

- **RecyclerView.Adapter**: class yang digunakan untuk menghubungkan data source dengan RecyclerView. Di dalamnya terdapat beberapa fungsi berikut.
  - **onCreateViewHolder()**: digunakan untuk membuat ViewHolder baru yang terhubung dengan layout item.
  - **onBindViewHolder()**: digunakan untuk menetapkan data source ke dalam ViewHolder sesuai dengan posisinya.
  - **getItemCount()**: digunakan untuk menetapkan ukuran dari jumlah data yang ingin ditampilkan.
- **RecyclerView.ViewHolder**: digunakan untuk menentukan bagaimana data ditampilkan ke dalam layout Item.

- Berikut ini adalah contoh kode untuk membuat RecyclerView.
  - **activity_main.xml**
  - item_row.xml
  - data class
  - Adapter
  - MainActivity

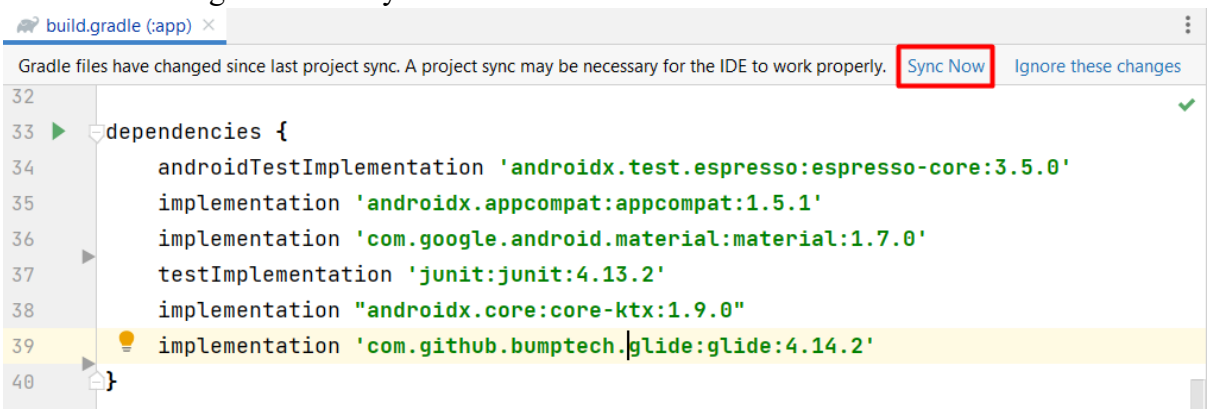
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.recyclerview.widget.RecyclerView
      android:id="@+id/rv_heroes"
      android:layout_width="0dp"
      android:layout_height="0dp"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

- Untuk mendapatkan **context** pada **Adapter**, gunakan itemView.getContext().
- Ada banyak cara untuk menambahkan aksi onClick pada item RecyclerView. Berikut di antaranya.
    - Langsung menambahkan setOnClickListener di dalam adapter.
    - Membuat callback menggunakan interface pada parameter.
    - Menambahkan lambda pada parameter.

Library & ViewBinding

- **Library** adalah tools atau source code yang sudah dibuat oleh orang lain (pihak ketiga) dan bisa dipakai di dalam aplikasi Anda.
- Setelah menambahkan dependency pada **build.gradle (module:app)**, klik **Sync Now** untuk mengunduh library.



- Untuk menampilkan gambar dari internet, Anda bisa menggunakan library **Glide**, **Picasso**, **Fresco**, dan **Coil**.
- **View Binding** adalah fitur untuk mengakses semua View yang ada di dalam XML secara otomatis tanpa perlu menggunakan findViewById().
- Untuk mengaktifkan ViewBinding, Anda perlu menambahkan kode berikut pada **build.gradle (module:app)**.

```
1. android {
2.     ...
3.     buildFeatures {
4.         viewBinding = true
5.     }
6. }
```

- View Binding akan men-generate class baru berdasarkan XML dengan format PascalCaseBinding, misal **activity_main.xml** menjadi **ActivityMainBinding**.
- Setiap id yang ada di dalam XML akan dibuat variabelnya secara otomatis dengan format lowerPascalCase, seperti **tv_welcome** menjadi **tvWelcome**.

- Untuk menggunakan View Binding Anda perlu membuat object View Binding yang meng-*inflate* Activity yang digunakan dan ganti parameter pada setContentView dengan root dari View Binding seperti berikut:

```
1. class MainActivity : AppCompatActivity() {
2.
3.     private lateinit var binding: ActivityMainBinding
4.
5.     override fun onCreate(savedInstanceState: Bundle?) {
6.         super.onCreate(savedInstanceState)
7.         binding = ActivityMainBinding.inflate(layoutInfla
   ter)
8.         setContentView(binding.root)
9.         binding.tvWelcome.text = "Hello Dicoding"
10.       }
11.   }
```

Berikut adalah cara menggunakan View Binding di dalam Adapter.

1. ...
2. override fun onCreateViewHolder(viewGroup: ViewGroup, i: Int): ListViewHolder {
3. **val binding = ItemRowHeroBinding.inflate(LayoutInflater.from(viewGroup.context), viewGroup, false)**
4. return ListViewHolder(**binding**)
5. }
6.
7. override fun onBindViewHolder(holder: ListViewHolder, position: Int) {
8. val (name, description, photo) = listHero[position]
9. holder.**binding.imgItemPhoto**.setImageResource(photo)

10. holder.**binding.tvItemName**.text = name
11. holder.**binding.tvItemDescription**.text = description
12. holder.itemView.setOnClickListener {
13. onItemClickCallback.onItemClicked(listHero[holder.adapterPosition])
14. }
15. }
16.
17. class ListViewHolder(**var binding: ItemRowHeroBinding**) : RecyclerView.ViewHolder(**binding.root**)

- Sebenarnya selain View Binding, ada juga beberapa alternatif lain yang bisa Anda gunakan untuk mem-binding View, tetapi View Binding merupakan solusi terbaik untuk saat ini. ViewBinding memiliki banyak kelebihan daripada alternatif lain, yaitu:
    - **Compile Time Safety** : Aman dari eror null yang biasanya disebabkan karena mengambil id dari layout yang berbeda.
    - **Elegance** : Kode lebih terlihat bersih dan tidak banyak boilerplate code (kode yang ditulis berulang-ulang).
    - **Correct Type of View** : Memberikan tipe komponen yang tepat, berbeda dengan findViewById yang bisa saja salah menentukan tipe komponen.
    - **Build Speed** : Kecepatan saat project dijalankan lebih cepat daripada DataBinding.
    - **Support Java & Kotlin** : Mendukung kedua bahasa. Berbeda dengan Kotlin Synthetic yang hanya mendukung bahasa Kotlin.
- Anda bisa membaca penjelasan lebih lanjut tentang View Binding pada tautan berikut:
    - View Binding Documentation