TEKNOLOGI MULTIMEDIA

Modul 2

TEORI ACTIVITY

Oleh
Dr. Ing. Melvi, S.T., M. T.
Aryanto, S.T., M.T>

MODUL TEORI ACTIVITY

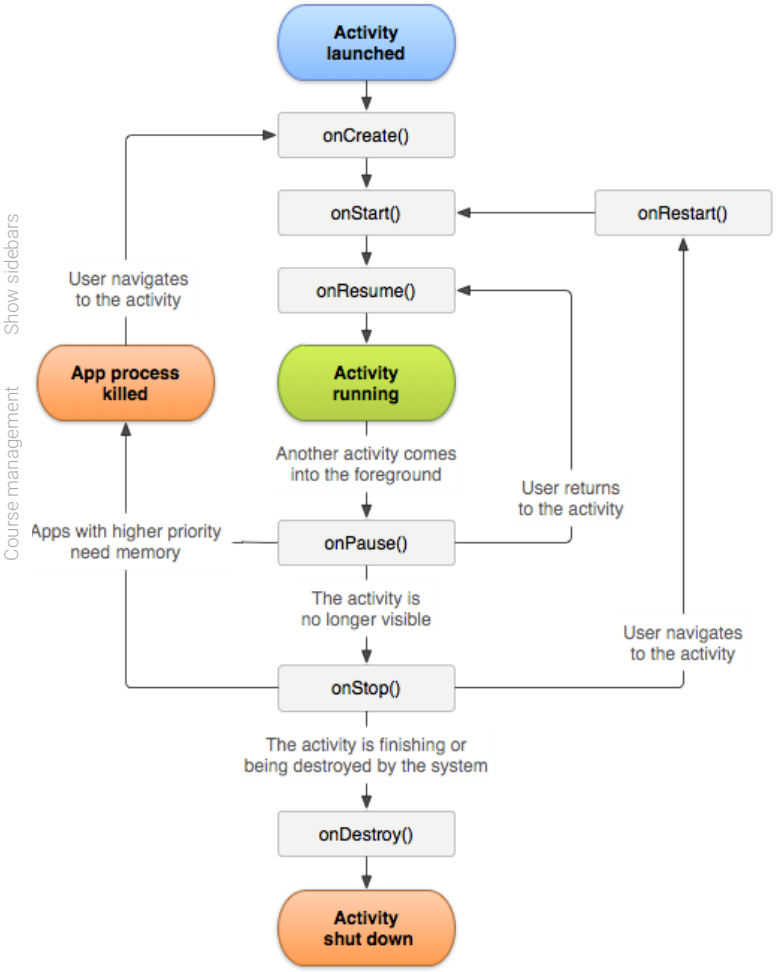
TEORI ACTIVITY

Activity merupakan salah satu komponen penting Android yang berfungsi untuk menampilkan *user interface* ke layar pengguna. Ini seperti pada saat Anda melihat daftar percakapan pada aplikasi *chat* atau daftar *email* pada aplikasi Gmail di ponsel Android Anda. Di dalamnya Anda dapat berinteraksi dengan aplikasi Anda, baik dengan menekan tombol atau menampilkan *list*.

Seperti ketika Anda membuat project baru di Android Studio, biasanya akan ada dua berkas yang sudah tercipta, yaitu **MainActivity** dan **activity_main.xml.** MainActivity ini disebut sebagai class Activity karena mewarisi (*extends*) *superclass* Activity. Tugasnya yaitu menampilkan layout activity_main.xml dan mengelola interaksi yang ada di dalamnya.

Umumnya dalam sebuah aplikasi terdapat lebih dari satu Activity yang saling terhubung dengan tugas yang berbeda-beda. Yang perlu diperhatikan yaitu setiap Activity harus terdaftar di **AndroidManifest.xml.** Secara *default*, ia akan didaftarkan jika Anda membuat Activity baru dengan cara otomatis. Caranya yaitu klik kanan pada nama package → New → Activity → pilih template Activity yang tersedia.

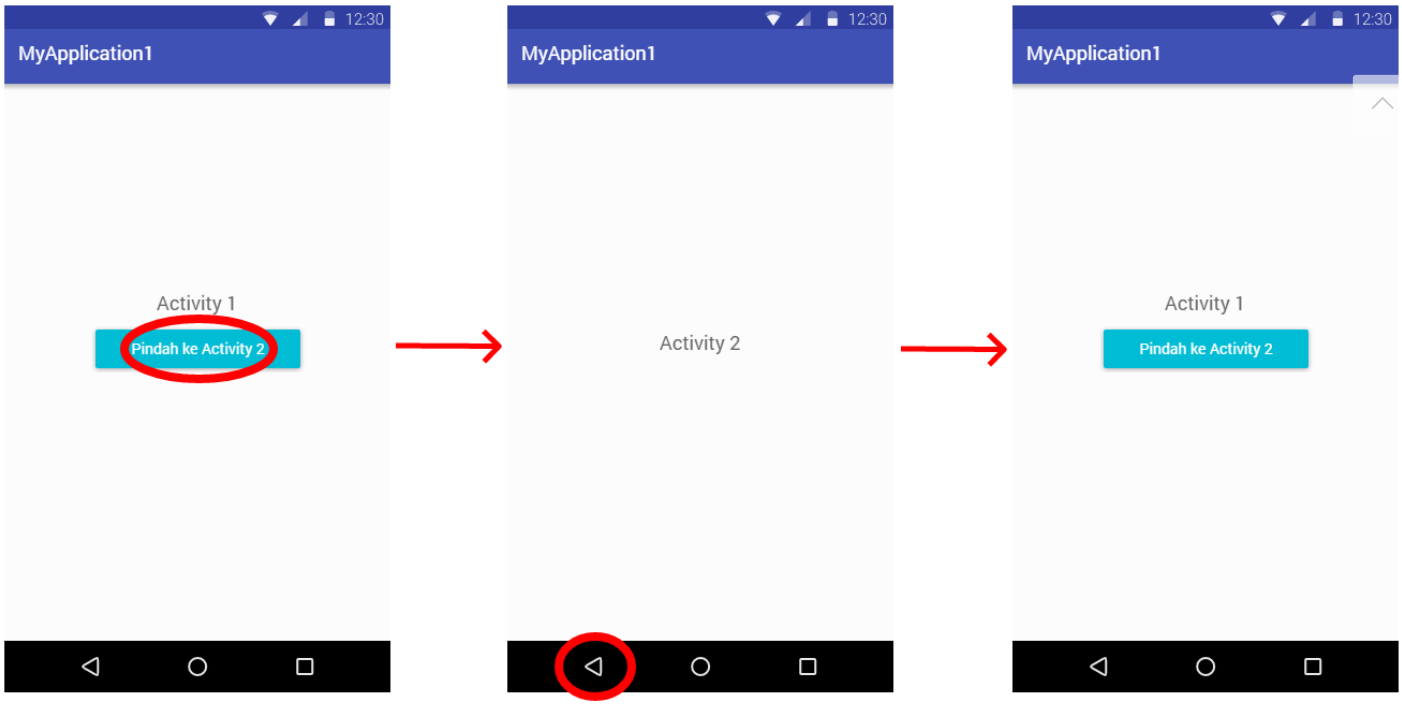
Activity Lifecycle



Developer yang baik harus mengetahui secara detail tentang *life cycle* sebuah Activity. Terutama untuk melakukan aksi yang tepat, saat terjadi perubahan *state* Activity. *Callback methods* yang ada dapat digunakan untuk melakukan beragam proses terkait state dari Activity. Misalnya melakukan semua inisialisasi komponen di onCreate(), melakukan *disconnect* terhadap koneksi ke *server* pada onStop() atau onDestroy() dan lain sebagainya.

Pemahaman yang baik tentang daur hidup Activity akan membuat implementasi rancangan aplikasi Anda menjadi lebih baik. Hal ini juga akan meminimalisir terjadinya *error/ bug/ force close* yang tidak diinginkan.

Last In, First Out (LIFO)



Gambar 1	Gambar 2	Gambar 3
Aktif: Activity 1 **onCreate() → onStart() → onResume()**	Aktif: Activity 2 Stack append: Activity 2 [**onResume()**]	Activity 1 **onStop() → onRestart() → onStart() → onResume()**
Aksi: Klik Button1 (Pindah)	Aksi: Klik Hardware Back Button	Aktif: Activity 1
Stack append: Activity 1 [**onStop()**]	Activity 2 [**finish()**] Stack pop: Activity 2 [**onDestroy()**]	

Gambar 1

Jika Anda memiliki sebuah aplikasi yang terdiri dari 2 Activity, maka Activity pertama akan dijalankan setelah pengguna meluncurkan aplikasi melalui ikon aplikasi di layar device. Activity yang ada saat ini berada pada posisi Activity *running* setelah melalui beberapa *state* onCreate (*created*) → onStart (*started*) → onResume (resumed) dan masuk ke dalam sebuah stack Activity.

Bila pada Activity pertama Anda menekan sebuah tombol untuk menjalankan activity kedua, maka posisi *state* dari Activity pertama berada pada posisi *stop*. Saat itu, *callback* onStop() pada Activity pertama akan dipanggil.
Ini terjadi karena Activity pertama sudah tidak berada pada layar *foreground* / tidak lagi ditampilkan. Semua informasi terakhir pada Activity pertama akan disimpan secara otomatis. Sementara itu, Activity kedua masuk ke dalam *stack* dan menjadi Activity terakhir yang masuk.

Gambar 2

Activity kedua sudah muncul di layar sekarang. Ketika Anda menekan tombol *back* pada *physical button* menu utama atau menjalankan metode finish(), maka Activity kedua Anda akan dikeluarkan dari *stack*.
Pada kondisi di atas, state Activity kedua akan berada pada *destroy*. Oleh karenanya, metode onDestroy() akan dipanggil. Kejadian keluar dan masuk *stack* pada proses di atas menandakan sebuah model *Last In, First Out*. Activity kedua menjadi yang terakhir masuk *stack* (*Last In*) dan yang paling pertama keluar dari *stack* (*First Out*).

Gambar 3

Activity pertama akan dimunculkan kembali di layar setelah melalui beberapa *state* dengan rangkaian *callback method* yang terpanggil, onStop → onRestart → onStart → onResume.

Saving Activity State

etika sebuah Activity mengalami *pause* kemudian *resume*, maka state dari sebuah Activity tersebut dapat terjaga. Sebabnya, obyek activity masih rsimpan di *memory* sehingga dapat dikembalikan *state*-nya.

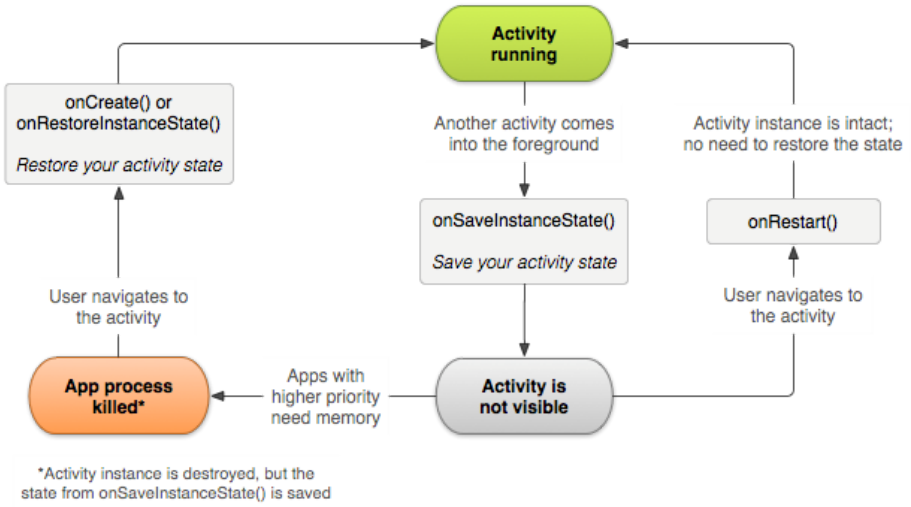
engan menjaga *state* dari Activity, maka ketika Activity tersebut ditampilkan, kondisinya akan tetap sama dengan kondisi sebelumnya.

kan tetapi ketika sistem menghancurkan Activity untuk keperluan memori misalnya karena memori habis, maka obyek Activity dihancurkan. hasil, ketika Activity ingin ditampilkan kembali diperlukan proses *recreate* Activity yang dihancurkan tadi.

ejadian di atas adalah hal yang lumrah terjadi. Oleh karena itu, perubahan yang terjadi pada Activity perlu disimpan terlebih dahulu sebelum ia hancurkan. Di sinilah metode onSaveInstanceState() digunakan.

alam onSaveInstanceState terdapat bundle yang dapat digunakan untuk menyimpan informasi. Informasi dapat disimpan dengan memanfaatkan fungsi seperti putString() dan putInt().

Ketika Activity di-*restart*, bundle akan diberikan kepada metode onCreate dan onRestoreInstanceState. Bundle tersebut akan dimanfaatkan untuk mengembalikan kembali perubahan yang telah terjadi sebelumnya.



Proses penghancuran Activity dapat juga terjadi ketika terdapat perubahan konfigurasi seperti perubahan orientasi layar (*portrait-landscape*), *keyboard availability*, dan perubahan bahasa. Penghancuran ini akan menjalankan *callback method* onDestroy dan kemudian menjalankan onCreate. Penghancuran ini dimaksudkan agar Activity dapat menyesuaikan diri dengan konfigurasi baru yang muncul pada kejadian-kejadian sebelumnya.

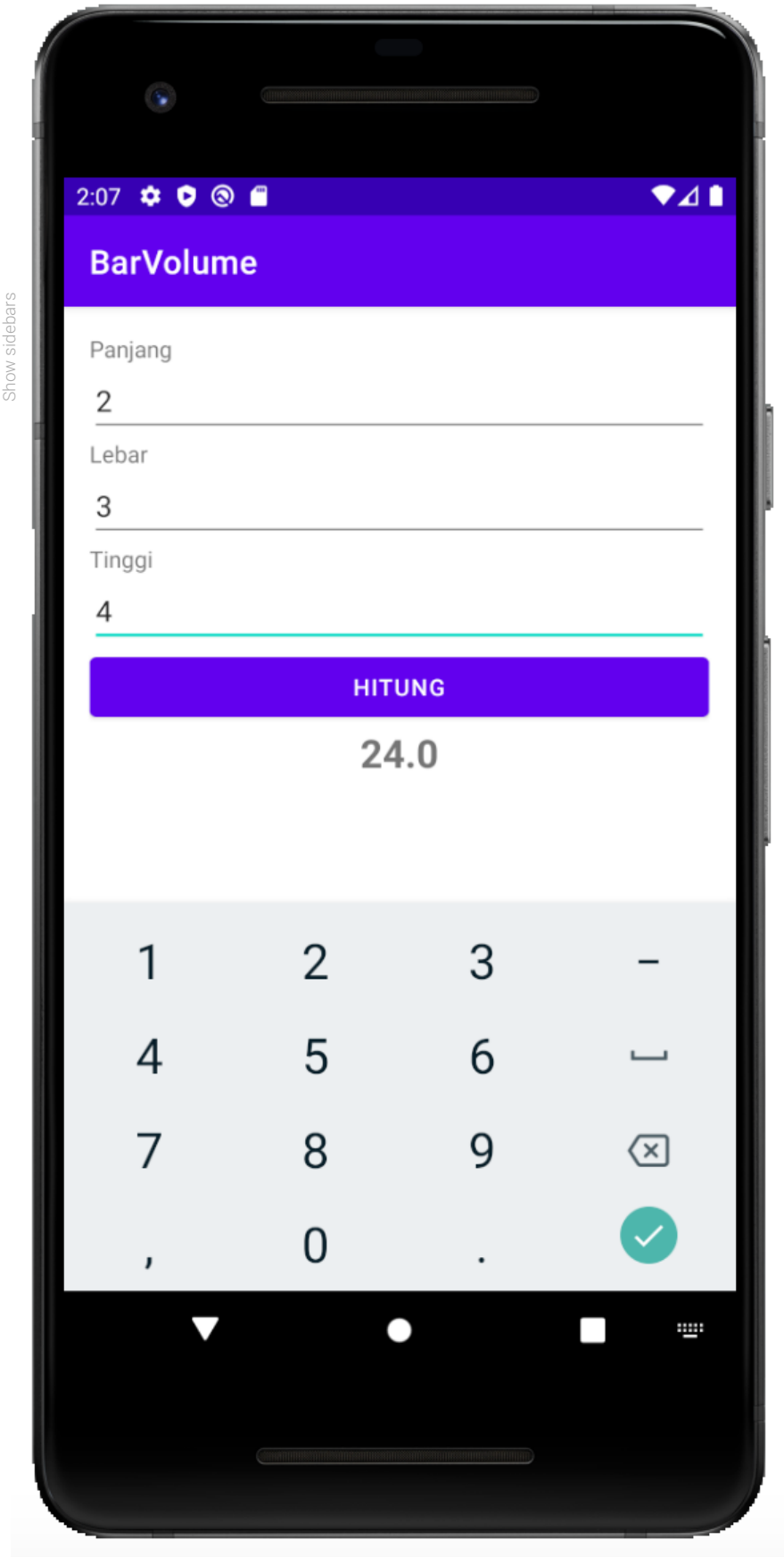
Hal yang perlu diingat ketika menggunakan onSaveInstanceState adalah untuk tidak menyimpan data yang besar pada bundle. Contohnya, hindari penyimpanan data *bitmap* pada bundle. Bila data pada bundle berukuran besar, proses serialisasi dan deserialisasi akan memakan banyak memori.

CODELAB MEMBUAT PROYEK BARUTEORI ACTIVITY

MEMBUAT PROYEK BARU TEORI ACTIVITY

Codelab ini bertujuan untuk mengimplementasikan komponen Activity pada aplikasi pertama yang Anda bangun. Harapannya aktifitas ini dapat memberi gambaran yang jelas tentang cara kerja activity.

Codelab pertama adalah dengan membuat aplikasi yang dapat menghitung volume balok. Seperti ini tampilannya.



Logika Dasar

Melakukan input ke dalam obyek EditText → melakukan validasi input → melakukan perhitungan volume balok ketika tombol hitung diklik.

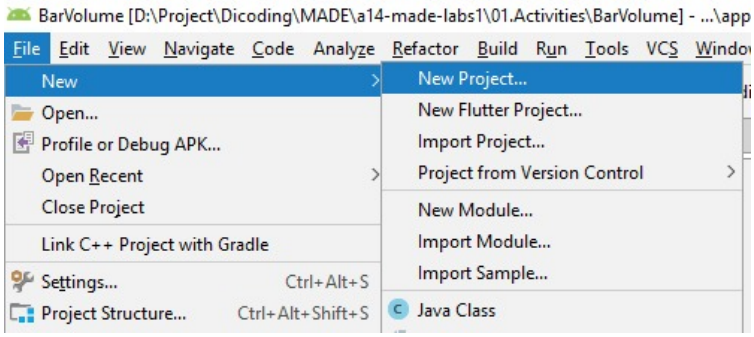
Codelab Membuat Proyek Baru

Hai! Pasti sudah tidak sabar ya untuk memulai membuat aplikasi pertama kalian. Sebelum mulai lebih lanjut, ada video menarik nih buat teman-teman supaya bisa mendapatkan gambaran terlebih dahulu bagaimana proses pembuatan aplikasi pertama. Silakan dicek ya!

Okay, jika sudah dapat gambarannya, yuk kita lanjut untuk bikin aplikasinya. *Cuss!*

1. Buat proyek baru dengan klik **File → New → New Project** pada Android Studio Anda atau Anda bisa memilih **Start a new Android Studio project** di bagian **dashboard**.

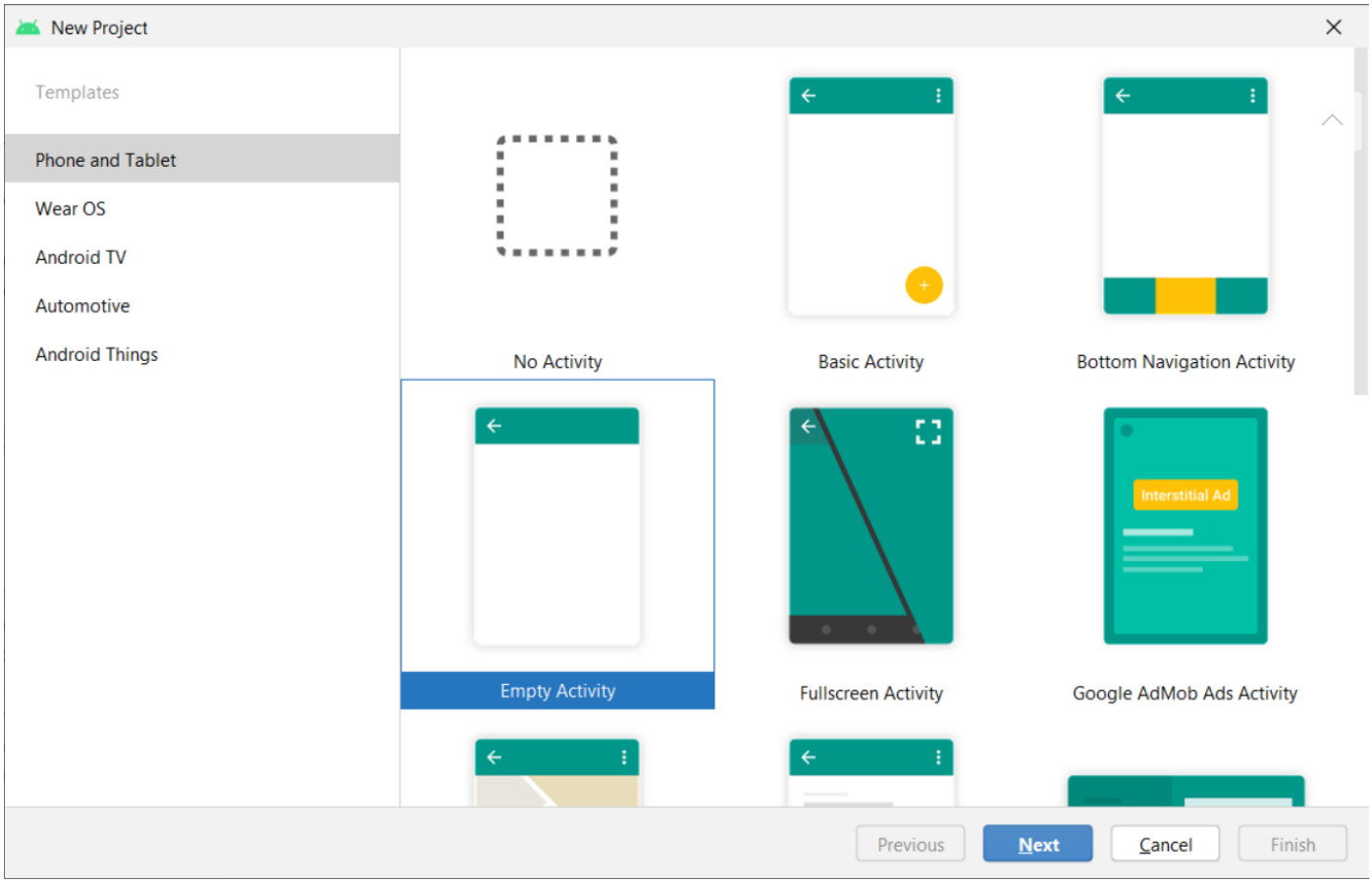
    - Windows
    - iOS



2. Pada bagian ini kita akan memilih tipe activity awal dari *template* yang telah disediakan. Saat ini Android Studio sudah menyediakan berbagai macam *template* activity dari yang paling sederhana hingga yang paling kompleks seperti:

Nama Template Activity	Fungsinya
No Activity	Project murni tanpa ada Activity yang ditambahkan
Basic Activity	Activity dengan *template* komponen *material design* seperti FloatingActionButton dan menu Toolbar
Bottom Navigation Activity	Activity dengan tampilan *bottom navigation menu* di bagian bawah
Empty Activity	Activity dalam bentuk yang paling sederhana (tanpa menu)
Fullscreen Activity	Activity *fullscreen* tanpa status bar
Google AdMob Ads Activity	Activity dengan konfigurasi *default* iklan Admob
Google Maps Activity	Activity dengan menyediakan konfigurasi dasar Google Maps
Login Activity	Activity untuk halaman *login*
Master/Detail Flow	Activity yang diperuntukan untuk alur aplikasi *master detail* pada peranti tablet
Navigation Drawer Activity	Activity dengan tampilan *side bar menu* yang bisa di-*swipe*
Settings Activity	Activity yang diperuntukan untuk konfigurasi aplikasi
Scrolling Activity	Activity dengan kemampuan *scroll* konten didalamnya secara vertikal
Tabbed Activity	Activity yang diperuntukan untuk menampilkan lebih dari satu tampilan dengan menggunakan Tab dan dapat digeser ke kanan dan ke kiri (*swipe*) dengan menggunakan komponen ViewPager
Fragment + ViewModel	Activity dengan menerapkan Fragment & ViewModel

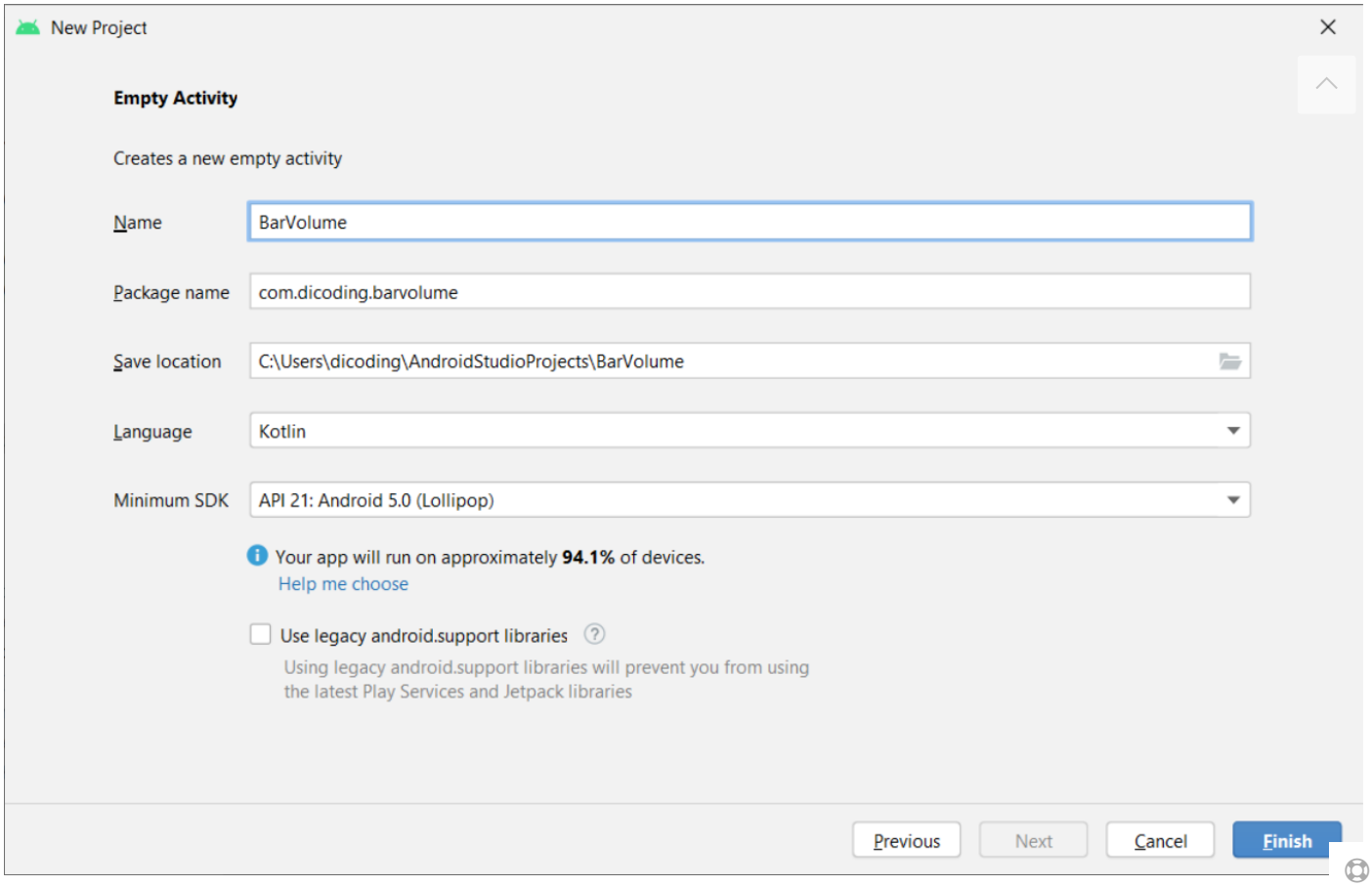
Selain itu, Anda juga bisa memilih *target device* mana yang akan dibuat seperti **Phone and Tablet**, **Wear OS**, **TV**, **Android Auto** atau **Android Things**.



Saat ini kita pilih tipe **Empty Activity** untuk belajar dari yang paling dasar, lalu klik **Next** untuk melanjutkan.

3. Selanjutnya masukkan **nama aplikasi** dan nama *package* aplikasi Anda. Sebaiknya jangan sama dengan apa yang ada di contoh, karena ini berfungsi sebagai id dari aplikasi yang Anda buat. Kemudian Anda bisa menentukan lokasi proyek yang akan Anda buat. Setelah itu pilih tipe gawai/peranti (*device*) untuk aplikasi beserta target minimum SDK yang akan digunakan. Pilihan target Android SDK akan mempengaruhi banyaknya peranti yang dapat menggunakan aplikasi. Di sini kita memilih nilai minimum SDK kita pasang ke **Level 21 (Lollipop)**. Klik

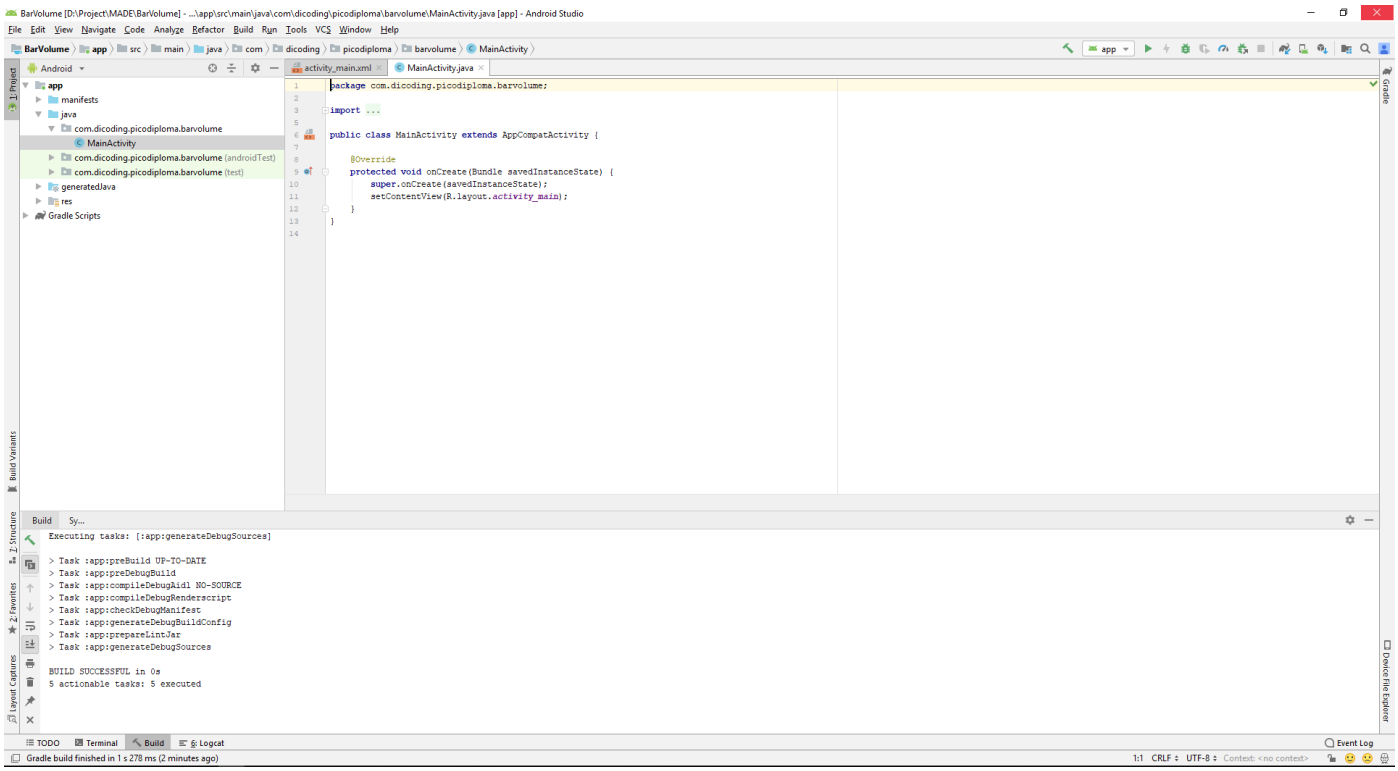
Finish untuk melanjutkan.



Catatan:

Pilih : Java

4. Tampilan layar Anda akan seperti contoh di bawah ini:



5. Di sebelah kanan Anda adalah *workspace* di mana Activity Anda berada dan bernama MainActivity dengan layout-nya activity_main.xml. Di sebelah kiri Anda terdapat struktur proyek, di mana nanti kita akan banyak menambahkan berbagai komponen baru, *asset* dan *library*. Untuk

lebih mengenal Android Studio lebih dalam silakan baca materi ini https://developer.android.com/studio/intro/index.html.

Selanjutnya kita akan mulai melakukan pengkodean aplikasi atau lebih enaknya disebut *ngoding*. Berikut flow umumnya:

1. **Ngoding Layout** untuk user interface aplikasi.
2. **Ngoding Activity** untuk menambahkan logika aplikasi.

Untuk mengoptimalkan proses pengetikan, Anda dapat memanfaatkan *code completion* dengan menekan **ctrl + space**. Android Studio juga akan otomatis mengimpor *package* dari komponen yang digunakan.

larang Keras untuk copy - paste! Ngoding pelan-pelan akan membuat Anda lebih jago di masa depan.

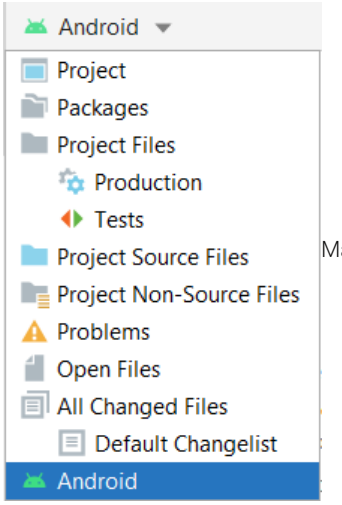
elamat ngoding!

Show sidebars

Codelab Layouting

Menambahkan Code Sederhana pada Layout Activity

1. Silakan pilih berkas **activity_main.xml** pada *workspace* Anda(**res/layout/activity_main.xml**).
   Pastikan **project window** pada pilihan **Android** seperti di bawah ini.



   Maka akan ada tampilan seperti ini, kemudian pilih *tab* ***Code*** di sebelah pojok kanan atas.

   Ubah layout dasar dari ConstraintLayout menjadi LinearLayout seperti berikut:

Selanjutnya tambahkan baris-baris yang dicetak **tebal** di bawah ini. Berikut adalah contoh cara menuliskannya.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

</LinearLayout>
```

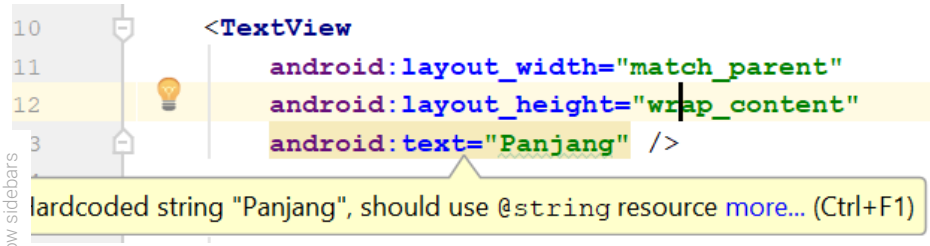
Show sidebars

-

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    android:orientation="vertical">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Panjang" />

    <EditText
        android:id="@+id/edt_length"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="numberDecimal"
        android:lines="1" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Lebar" />

    <EditText
        android:id="@+id/edt_width"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="numberDecimal"
        android:lines="1" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Tinggi" />

    <EditText
        android:id="@+id/edt_height"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="numberDecimal"
        android:lines="1" />

    <Button
        android:id="@+id/btn_calculate"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Hitung" />

    <TextView
        android:id="@+id/tv_result"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Hasil"
        android:textSize="24sp"
        android:textStyle="bold" />
</LinearLayout>
```

Perlu diperhatikan *root layout* (tag layout terluar) yang dipakai di sini adalah LinearLayout. Jika kita menggunakan Android Studio versi 3 ke atas, secara *default root* yang dipakai adalah ConstraintLayout. Agar sesuai dengan latihan ini, kita tinggal menggantinya menjadi LinearLayout. Untuk Layout akan dibahas nanti pada modul yang berbeda. Jadi ikuti saja dulu ya!

- Perhatikan bahwa ada *warning* berwarna kuning pada atribut android:text di layout tersebut.

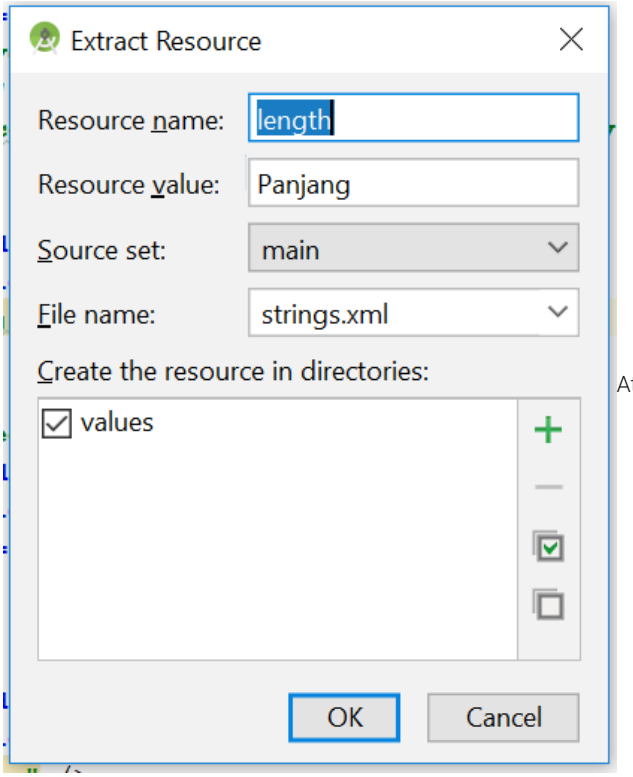


Ini karena kita melakukan *hardcoding* (menuliskan ks secara langsung) pada nilai *string*-nya. Ini merupakan praktik yang kurang baik karena seharusnya kita menuliskan semua teks pada berkas **es/values/strings.xml** dan setelah itu baru memanggilnya.

Okey, mari kita hilangkan *warning* tersebut dengan menekan **Alt+Enter** (option + return pada Mac) atau menekan lampu kuning yang muncul pada attribut android:text. Akan muncul dialog seperti ini, pilih **extract string resource**.

-

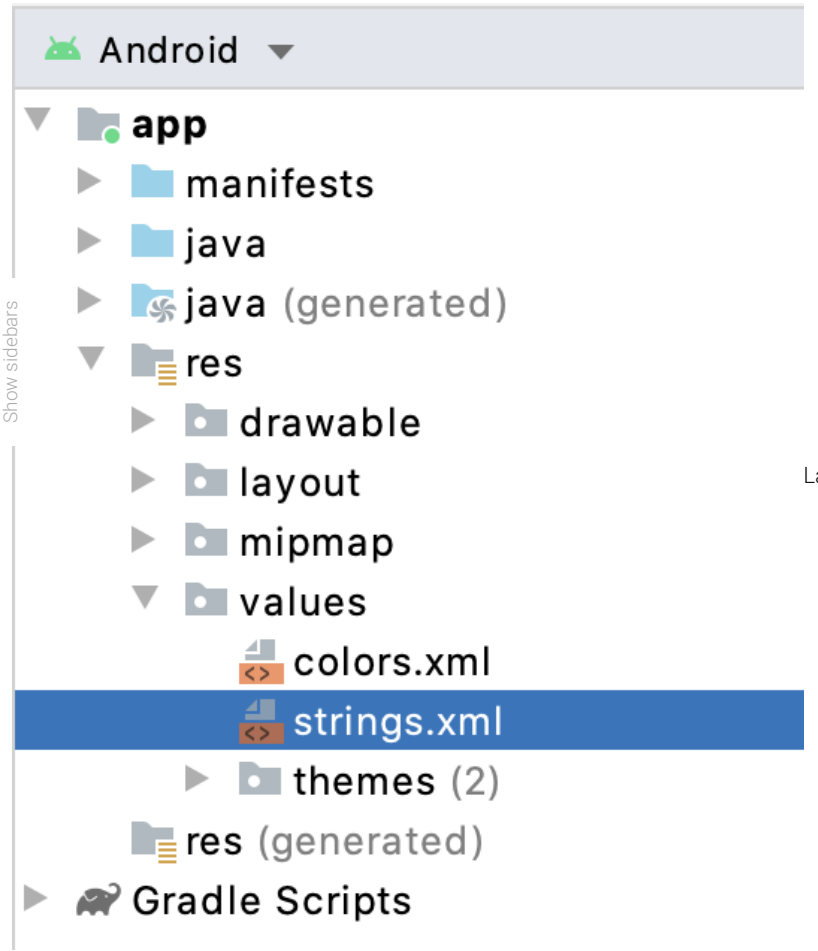
Kemudian akan muncul *dialog* seperti di bawah ini. Sesuaikan dengan nama yang ada.



Atau bisa juga melihat animasi berikut:

-

Fungsi **extract string resource** akan secara otomatis menambahkan nilai dari android:text ke dalam berkas **res/values/strings.xml**.



Lakukan hal yang sama pada setiap text lainnya. Jika kita buka berkas **strings.xml** yang ada di folder **res/value**, maka isinya akan menjadi seperti ini:

-
```
1. <resources>
2.     <string name="app_name">BarVolume</string>
3.     <string name="width">Lebar</string>
4.     <string name="height">Tinggi</string>
5.     <string name="calculate">Hitung</string>
6.     <string name="result">Hasil</string>
7.     <string name="length">Panjang</string>
8. </resources>
```

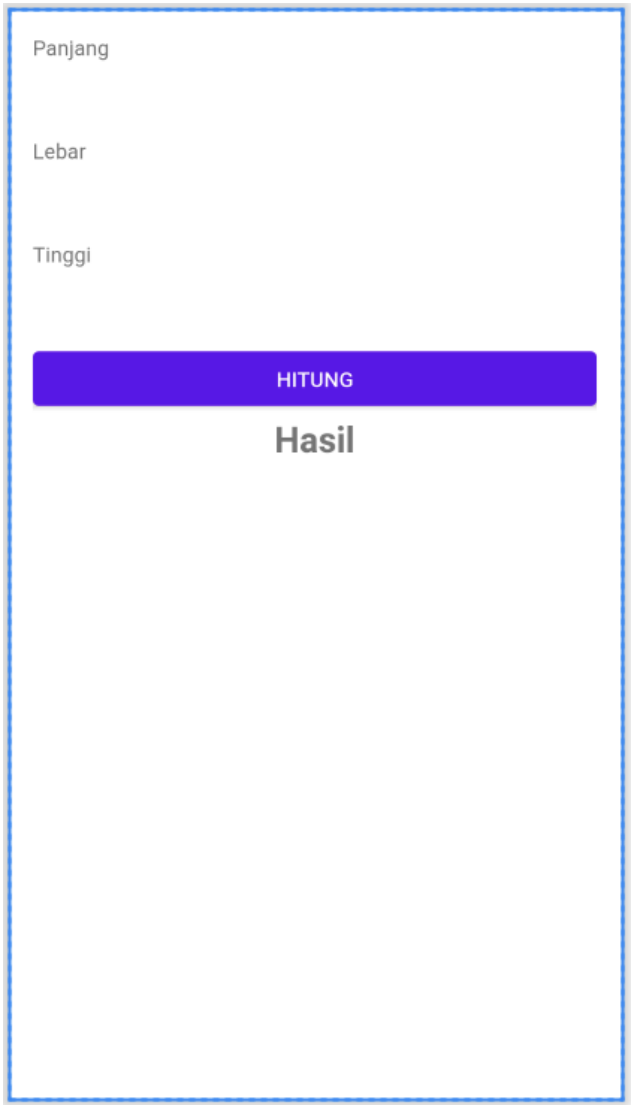
-

Maka kode di dalam activity_main.xml akan menjadi seperti ini:

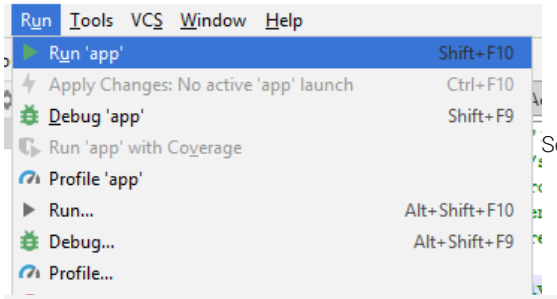
5.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    android:orientation="vertical">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/length" />

    <EditText
        android:id="@+id/edt_length"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="numberDecimal"
        android:lines="1" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/width" />

    <EditText
        android:id="@+id/edt_width"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="numberDecimal"
        android:lines="1" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/height" />

    <EditText
        android:id="@+id/edt_height"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="numberDecimal"
        android:lines="1" />

    <Button
        android:id="@+id/btn_calculate"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/calculate" />

    <TextView
        android:id="@+id/tv_result"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="@string/result"
        android:textSize="24sp"
        android:textStyle="bold" />
</LinearLayout>

```

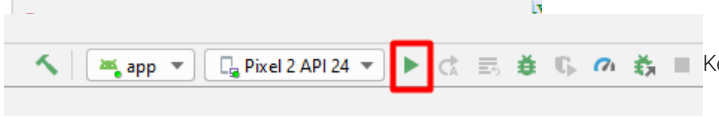
6. Jika Anda perhatikan, hasil layout sementara akan menjadi seperti ini:



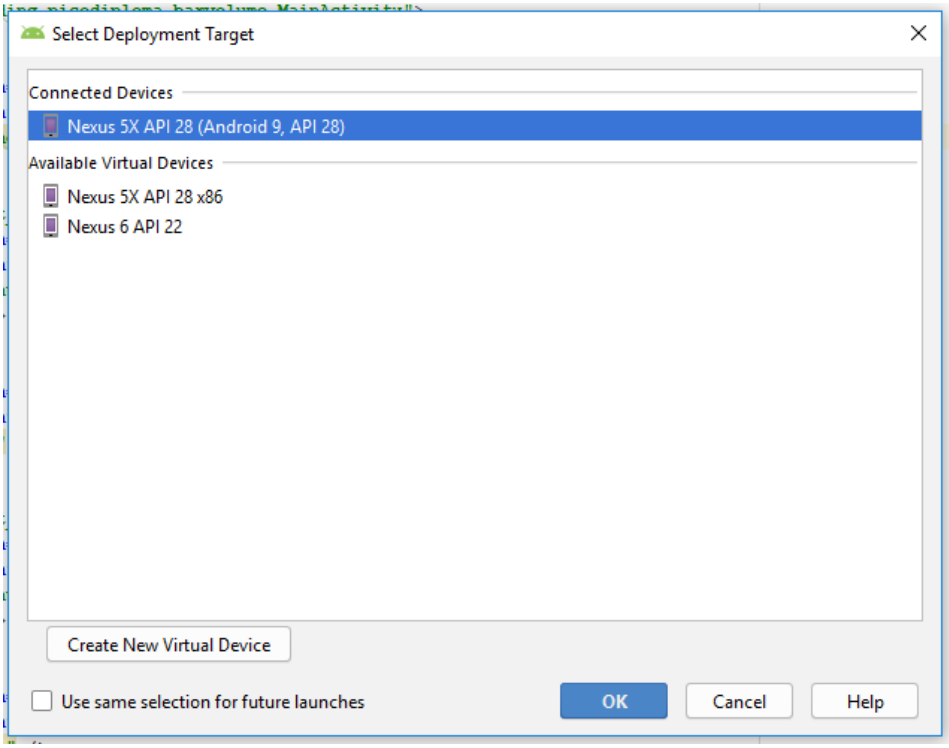
7. Setelah selesai mendesain aplikasi, silakan coba jalankan aplikasi dengan memilih menu **Run** → **Run 'app'** dari *menu bar*.



Selain cara di atas, Anda juga dapat menekan icon berikut di toolbar:



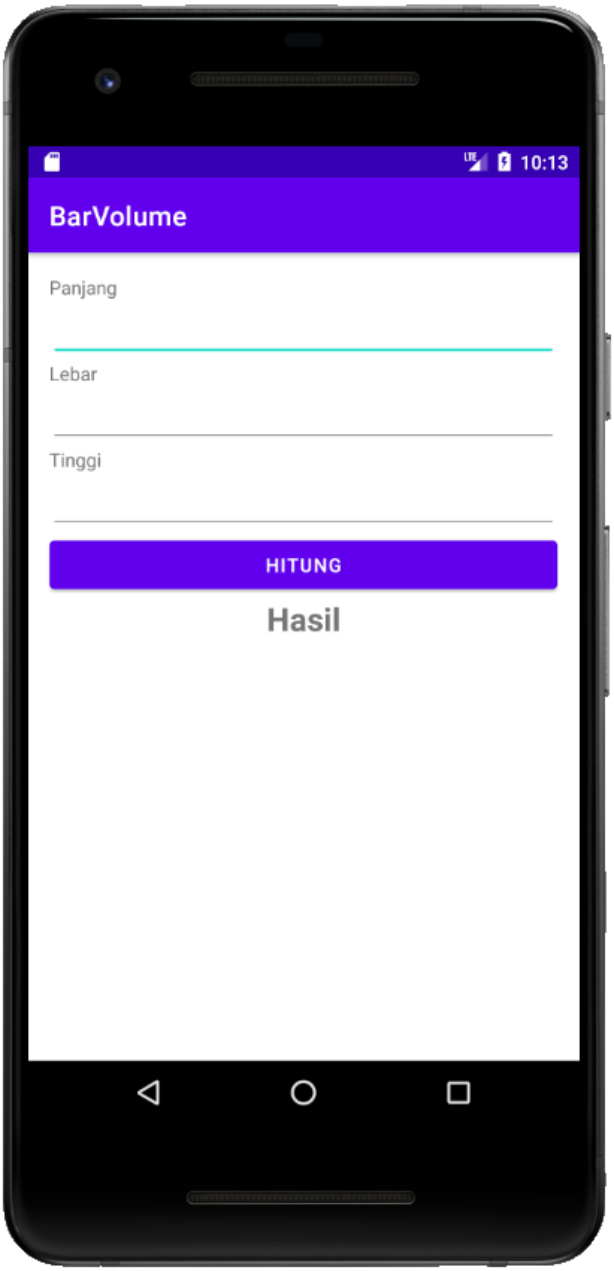
Kemudian akan muncul pilihan seperti ini:



Itu tandanya ADB (*Android Debugger*) pada peranti yang Anda punya telah terhubung dengan Android Studio. Jika Anda tidak memiliki peranti, maka Anda dapat menggunakan emulator. Ikuti materinya di modul sebelumnya atau lihat materi di sini.

Catatan:
Kami merekomendasikan Anda menggunakan peranti Android sewaktu mengembangkan aplikasi. Selain karena beban memori pada peranti Anda akan jadi lebih rendah, pendekatan ini juga akan memungkinkan Anda untuk merasakan bagaimana aplikasi berjalan di *device* sebenarnya.

Pilih **OK** untuk menjalankan dan tunggu hingga proses *building* dan instalasi APK selesai. Jika sudah, seharusnya hasilnya akan seperti ini:



Saat ini aplikasi telah tampil, namun ketika Anda coba untuk memasukkan angka dan klik tombol **Hitung**, aplikasi tidak akan merespons. Hal ini karena kita belum menambahkan logika kode pada **MainActivity**. Nah, yuk kita lanjut ke codelab selanjutnya supaya aplikasi pertama kita bisa berjalan dengan semestinya. Semangat!

7.

Codelab Kode Logika

Menambahkan Kode Logika Sederhana pada MainActivity.

1. Selanjutnya setelah selesai, lanjutkan dengan membuka berkas **MainActivity** dan lanjutkan *ngoding* baris-baris di bawah ini. Tambahkan beberapa variabel yang akan digunakan untuk menampung View.
    - Java

•

```
1. private EditText edtWidth;
2. private EditText edtHeight;
3. private EditText edtLength;
4. private Button btnCalculate;
5. private TextView tvResult;
```

Catatan:

Perhatikan bagaimana cara menuliskan kodenya. Biasakan menekan **Enter** ketika mengetik supaya komponen di-*import* secara otomatis, kecuali ama variabel yang Anda tentukan sendiri seperti edtWidth harus diketik satu per satu.

Show sidebars

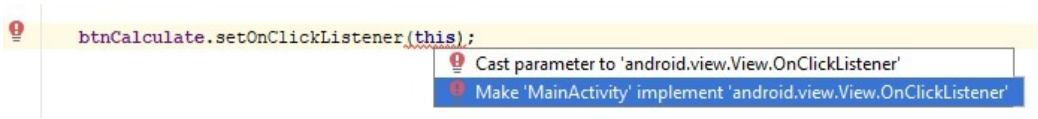
```
nport android.os.Bundle;
nport androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

    }

}
```

- Kemudian inisiasi variabel yang telah kita buat dengan menambahkan kode berikut di dalam metode onCreate.
  - Java

```
1. @Override
2. protected void onCreate(Bundle savedInstanceState) {
3.     super.onCreate(savedInstanceState);
4.     setContentView(R.layout.activity_main);
5.
6.     edtWidth = findViewById(R.id.edt_width);
7.     edtHeight = findViewById(R.id.edt_height);
8.     edtLength = findViewById(R.id.edt_length);
9.     btnCalculate = findViewById(R.id.btn_calculate);
10.    tvResult = findViewById(R.id.tv_result);
11.
12.    btnCalculate.setOnClickListener(this);
13.
14. }
```

- Akan muncul baris merah pada kata this. Hal ini karena ketika kita menggunakan setOnClickListener(this), maka kita perlu menambahkan interface View.OnClickListener di kelas **MainActivity**. Silakan **klik di atas baris merah** tersebut, kemudian **tekan tombol Alt + Enter (option + return pada Mac)** atau **menekan lampu merah** yang muncul lalu pilih aksi berikut untuk implement interface**.**

- Java



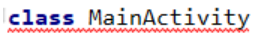
Maka secara otomatis akan ada penambahan kode pada kelas **MainActivity** seperti berikut ini:

- Java

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    ...

}
```

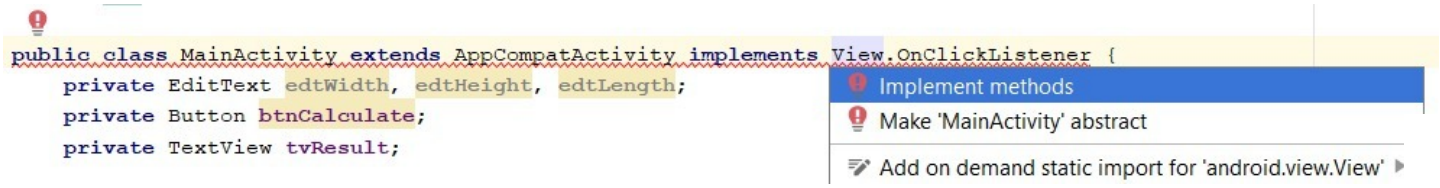
Show sidebars

Jika terdapat baris merah seperti ini:



Jangan khawatir! Silakan **klik di atas baris merah** tersebut, kemudian **tekan tombol Alt + Enter (option + return pada Mac)** atau **menekan lampu merah** yang muncul lalu pilih **implement methods (Java)** atau **implement members (Kotlin)**.

- Java



Maka secara otomatis akan ada penambahan metode onClick di kelas **MainActivity**. Setelah itu, tambahkan kode berikut ke dalam metode onCl

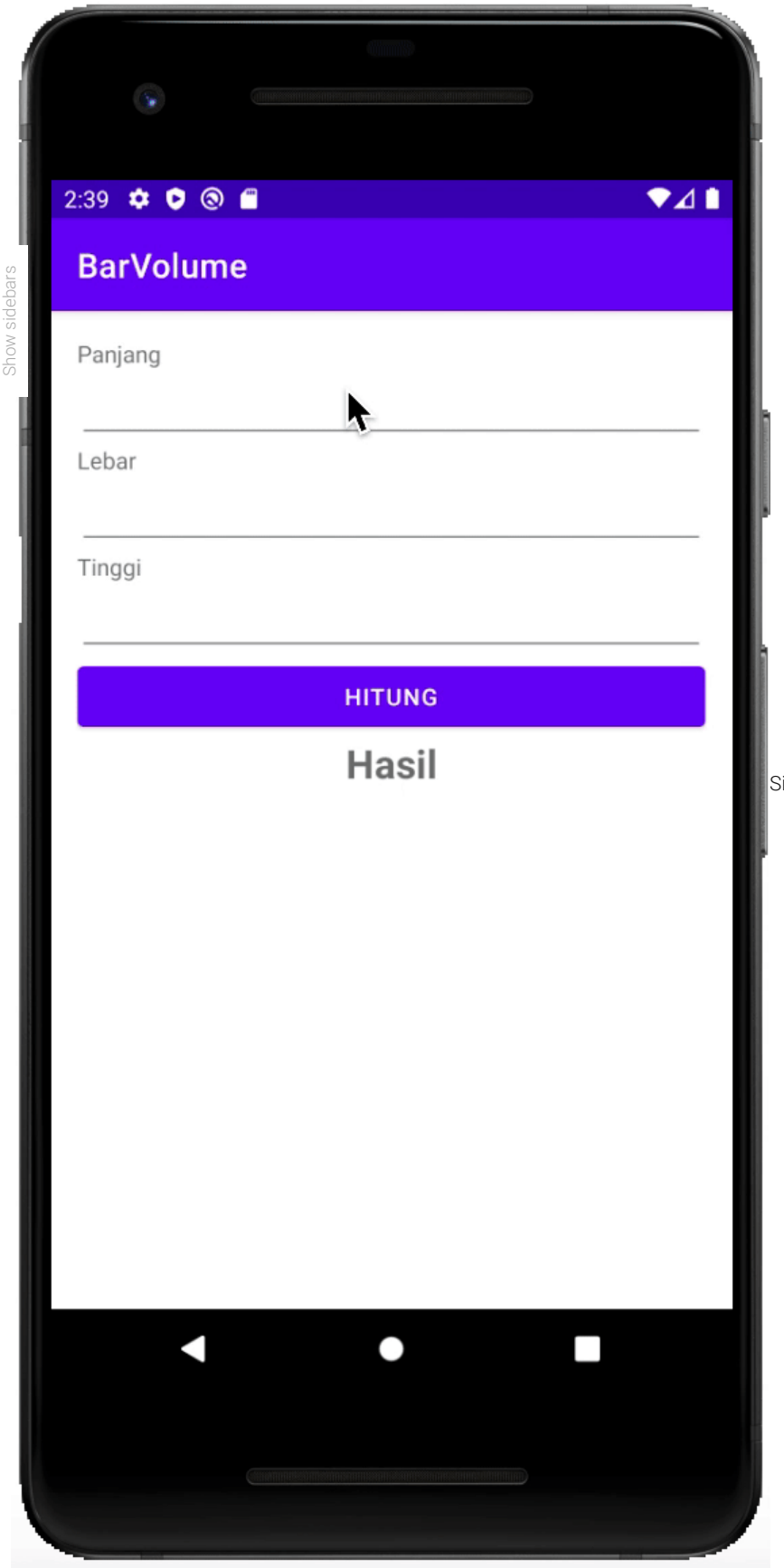
- Java

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    ...

}

@Override
public void onClick(View v) {
    if (v.getId() == R.id.btn_calculate) {
        String inputLength = edtLength.getText().toString().trim();
        String inputWidth = edtWidth.getText().toString().trim();
        String inputHeight = edtHeight.getText().toString().trim();

        double volume = Double.valueOf(inputLength) * Double.valueOf(inputWidth) * Double.valueOf(inputHeight);
        tvResult.setText(String.valueOf(volume));
    }
}
```

- Akhirnya kelas **MainActivity** akan memiliki kode seperti berikut ini:
- Java

```java
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    private EditText edtWidth, edtHeight, edtLength;
    private Button btnCalculate;
    private TextView tvResult;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        edtWidth = findViewById(R.id.edt_width);
        edtHeight = findViewById(R.id.edt_height);
        edtLength = findViewById(R.id.edt_length);
        btnCalculate = findViewById(R.id.btn_calculate);
        tvResult = findViewById(R.id.tv_result);

        btnCalculate.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        if (v.getId() == R.id.btn_calculate) {
            String inputLength = edtLength.getText().toString().trim();
            String inputWidth = edtWidth.getText().toString().trim();
            String inputHeight = edtHeight.getText().toString().trim();

            double volume = Double.valueOf(inputLength) * Double.valueOf(inputWidth) * Double.valueOf(inputHeight);
            tvResult.setText(String.valueOf(volume));
        }
    }
}
```

Setelah selesai, silakan jalankan aplikasi kembali. Jika sudah, seharusnya hasilnya akan seperti ini:



Silakan masukkan nilai panjang, lebar, dan tinggi kemudian tekan tombol **Hitung** dan hasilnya akan ditampilkan di objek textview tvResult. Namun masih ada sedikit masalah di sini, yaitu Anda tetap melakukan proses perhitungan walaupun salah satu nilainya kosong. Hal ini akan menyebabkan aplikasi *force close* karena perhitungan tidak dapat diproses. Maka untuk mengatasinya Anda akan menggunakan percabangan untuk mengecek apakah masing-masing EditText kosong atau tidak.

- Silakan buka kembali kelas **MainActivity**. Tambahkan kode berikut ke dalam metode onClicksebelum melakukan perhitungan.

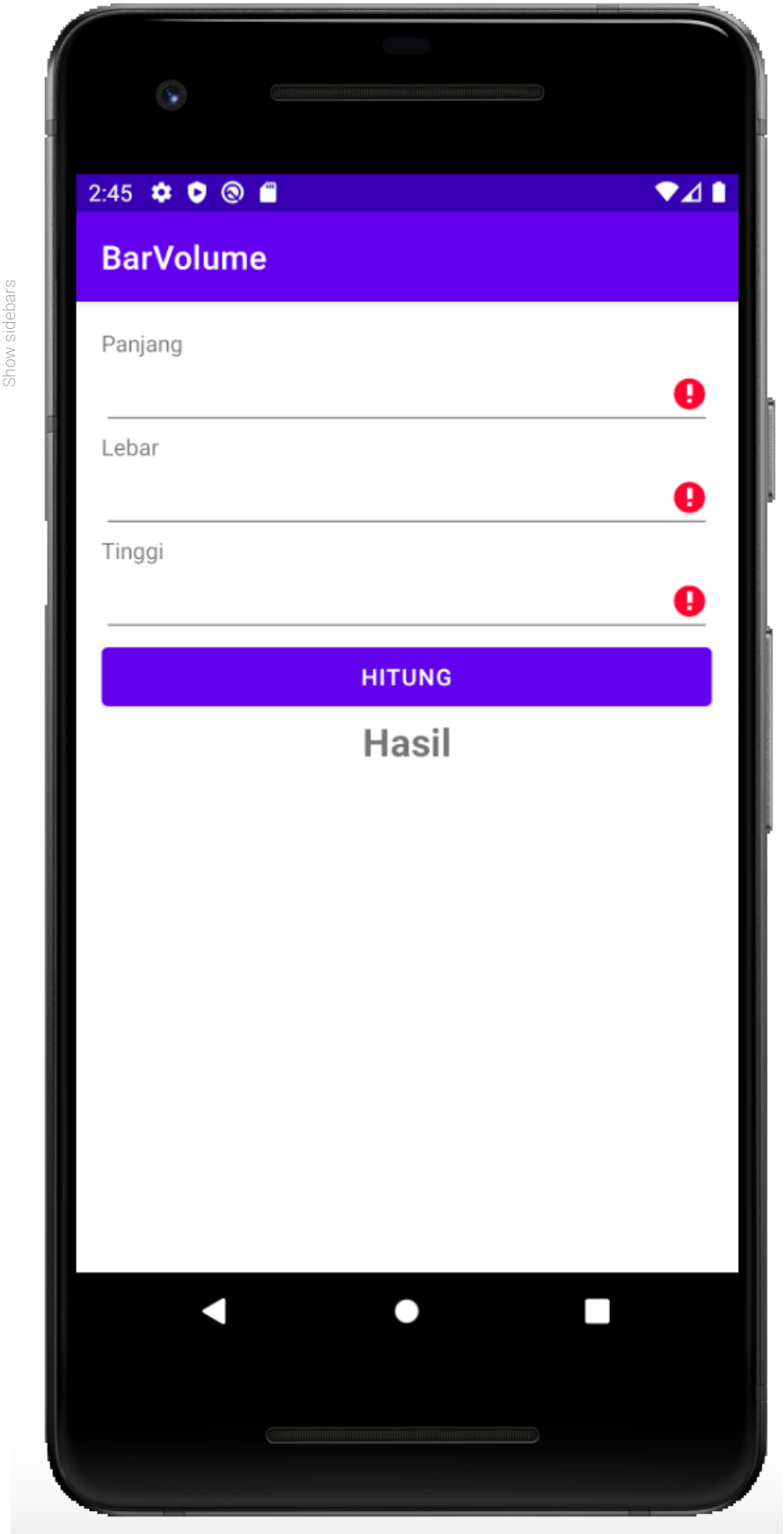
- Java

```
@Override
public void onClick(View v) {
    if (v.getId() == R.id.btn_calculate) {
        String inputLength = edtLength.getText().toString().trim();
        String inputWidth = edtWidth.getText().toString().trim();
        String inputHeight = edtHeight.getText().toString().trim();

        boolean isEmptyFields = false;

        if (TextUtils.isEmpty(inputLength)) {
            isEmptyFields = true;
            edtLength.setError("Field ini tidak boleh kosong");
        }

        if (TextUtils.isEmpty(inputWidth)) {
            isEmptyFields = true;
            edtWidth.setError("Field ini tidak boleh kosong");
        }

        if (TextUtils.isEmpty(inputHeight)) {
            isEmptyFields = true;
            edtHeight.setError("Field ini tidak boleh kosong");
        }

        if (!isEmptyFields) {
            double volume = Double.valueOf(inputLength) * Double.valueOf(inputWidth) * Double.valueOf(inputHeight);
            tvResult.setText(String.valueOf(volume));
        }
    }
}
```

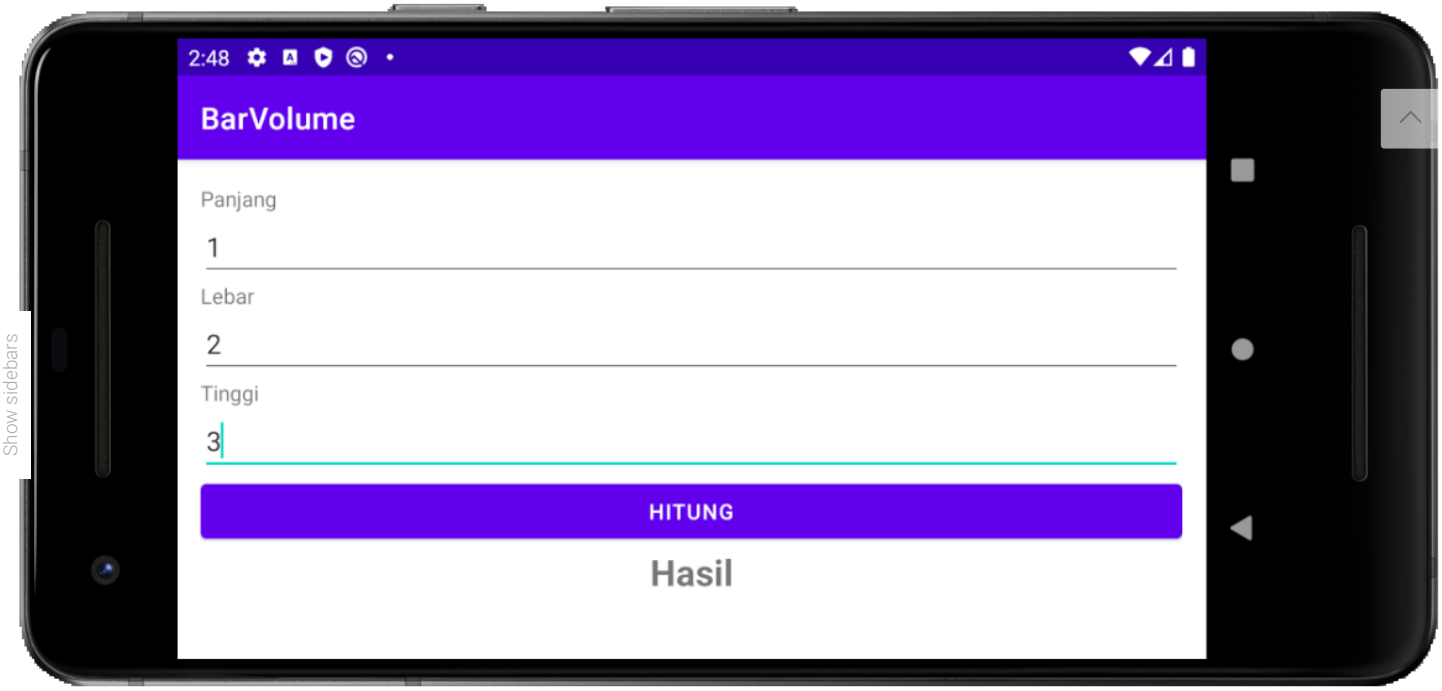
- Akhirnya kelas **MainActivity** akan memiliki kode seperti berikut ini:
  - Java

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    private EditText edtWidth, edtHeight, edtLength;
    private Button btnCalculate;
    private TextView tvResult;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        edtWidth = findViewById(R.id.edt_width);
        edtHeight = findViewById(R.id.edt_height);
        edtLength = findViewById(R.id.edt_length);
        btnCalculate = findViewById(R.id.btn_calculate);
        tvResult = findViewById(R.id.tv_result);

        btnCalculate.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        if (v.getId() == R.id.btn_calculate) {
            String inputLength = edtLength.getText().toString().trim();
            String inputWidth = edtWidth.getText().toString().trim();
            String inputHeight = edtHeight.getText().toString().trim();

            boolean isEmptyFields = false;

            if (TextUtils.isEmpty(inputLength)) {
                isEmptyFields = true;
                edtLength.setError("Field ini tidak boleh kosong");
            }

            if (TextUtils.isEmpty(inputWidth)) {
                isEmptyFields = true;
                edtWidth.setError("Field ini tidak boleh kosong");
            }

            if (TextUtils.isEmpty(inputHeight)) {
                isEmptyFields = true;
                edtHeight.setError("Field ini tidak boleh kosong");
            }

            if (!isEmptyFields) {
                double volume = Double.valueOf(inputLength) * Double.valueOf(inputWidth) * Double.valueOf(inputHeight);
                tvResult.setText(String.valueOf(volume));
            }
        }
    }
}
```

- Jalan kembali aplikasi Anda dengan memilih menu **Run → Run 'app'** atau shortcut **Shift + F10**. Cobalah langsung menekan tombol **HITUNG** tanpa mengisi EditText, maka aplikasi Anda tidak akan force close dan akan muncul peringatan bahwa **"Field ini tidak boleh kosong"**.



- Apakah kita sudah selesai? Belum! Masih ada yang kurang. Ketika nilai volume sudah dihitung dan kemudian terjadi pergantian orientasi (portrait-landscape) pada peranti, maka hasil perhitungan tadi akan hilang.



Untuk mengatasinya, tambahkan metode onSaveInstanceState() pada MainActivitydan sesuaikan seperti berikut:

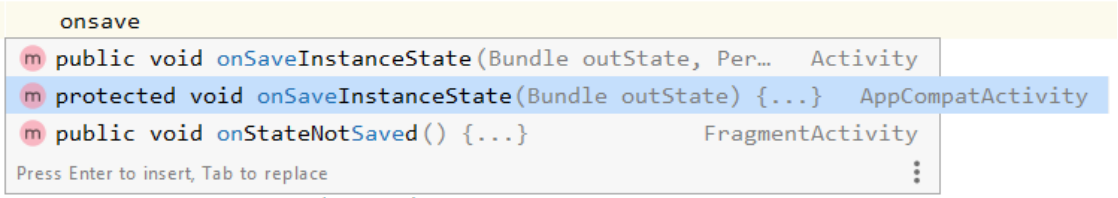
- Java

```
1. public class MainActivity extends AppCompatActivity implements View.OnClickListener{
2.     private EditText edtWidth;
3.     private EditText edtHeight;
4.     private EditText edtLength;
5.     private Button btnCalculate;
6.     private TextView tvResult;
7.
8.     private static final String STATE_RESULT = "state_result";
9.
10.    @Override
11.    protected void onCreate(Bundle savedInstanceState) {
12.        ...
13.    }
14.
15.    @Override
16.    protected void onSaveInstanceState(Bundle outState) {
17.        super.onSaveInstanceState(outState);
18.        outState.putString(STATE_RESULT, tvResult.getText().toString());
19.    }
20.
21.    ...
22. }
```

Catatan:

Karena onSaveInstanceState adalah class yang ada di superclass AppCompatActivity, Anda bisa membuat fungsi secara otomatis dengan hanya mengetikkan huruf **onSave...** dan pilih yang hanya memiliki satu parameter.

```
    }
```



Kemudian tambahkan juga beberapa baris berikut pada baris terakhir metode onCreate.

- Java

10.
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    ...

    if (savedInstanceState != null) {
        String result = savedInstanceState.getString(STATE_RESULT);
        tvResult.setText(result);
    }
}
```

Show sidebars

11. Silakan jalankan kembali aplikasinya. Ulangi proses perhitungan seperti sebelumnya. Kemudian ganti orientasi peranti Anda. Jika sudah benar maka hasil perhitungan tidak akan hilang.

Bedah Kode

Pembahasan tentang layout xml

Layout merupakan *user interface* dari suatu activity. Layout dituliskan dalam format xml (extensible markup language).

```
xml version="1.0" encoding="utf-8"?>
```

Baris ini mengidentifikasi bahwa berkas ini berformat xml.

```
xmlns:android="http://schemas.android.com/apk/res/android"
```

Kode di atas menandakan namespace yang digunakan dalam keseluruhan berkas xml ini.

Macam Views

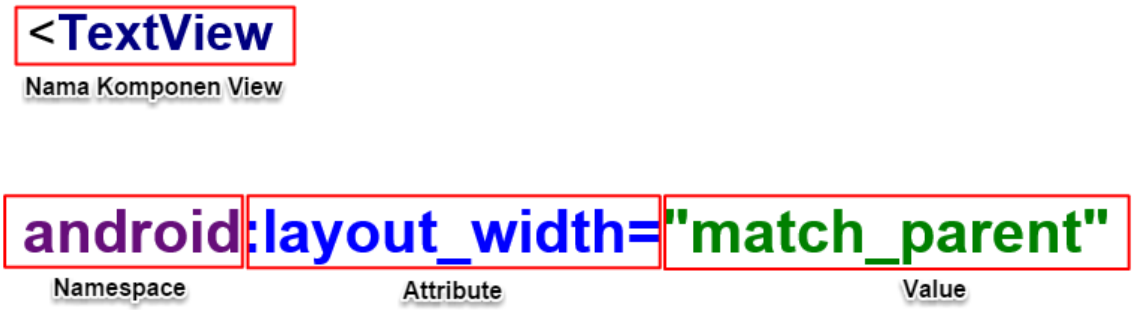
Di sini kita menggunakan beberapa komponen *user interface* yang disebut view. Di antaranya:

- **TextView** : Komponen view untuk menampilkan teks ke layar
- **EditText** : Komponen view untuk memberikan input teks
- **Button** : Komponen view untuk melakukan sebuah aksi klik
- **LinearLayout** : Komponen view bertipe viewgroup yang menjadi *parent* dari semua sub komponen view (sub view) di dalamnya. Komponen ini bersifat sebagai kontainer untuk komponen lain dengan orientasi secara vertikal atau horizontal.

Cara membaca :

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/calculate"
    android:layout_marginBottom="16dp"/>
```

Komponen di atas adalah sebuah TextView. Perhatikan gambar di bawah ini. Warna ungu menandakan *namespace* yang digunakan; warna biru adalah atribut dari komponen dan warna hijau adalah nilai dari atribut. Penjelasannya seperti di bawah ini:



Show sidebars

- **match_parent:** Ini berarti bahwa ukuran dimensi sebuah View disesuaikan dengan ukuran layar secara horizontal jika pada layout_width dan vertikal jika pada layout_height**.**
- **wrap_content :** Ini berarti bahwa ukuran dimensi sebuah View disesuaikan dengan ukuran konten di dalamnya baik secara horizontal pada layout_width dan vertikal jika pada layout_height**.**
- **@string/calculate:** value calculate berasal dari berkas strings.xml yang bisa Anda lihat dengan cara menekan dan tahan tombol **Ctrl** (atau command) + arahkan kursor ke atasnya dan kemudian klik sekali. .

```
<resources>
    <string name="app_name">BarVolume</string>
    <string name="width">Lebar</string>
    <string name="height">Tinggi</string>
    <string name="calculate">Hitung</string>
    <string name="result">Hasil</string>
    <string name="length">Panjang</string>
</resources>
```

Penggunaan *centralize resource value* akan memudahkan Anda sewaktu mengembangkan aplikasi Android. Cara tersebut digunakan agar Anda tidak menulis nilai yang sama berulang-ulang.

Apa itu '@+id/' ?

Salah satu contoh penerapan penggunaan **@+id/** sebagai berikut:

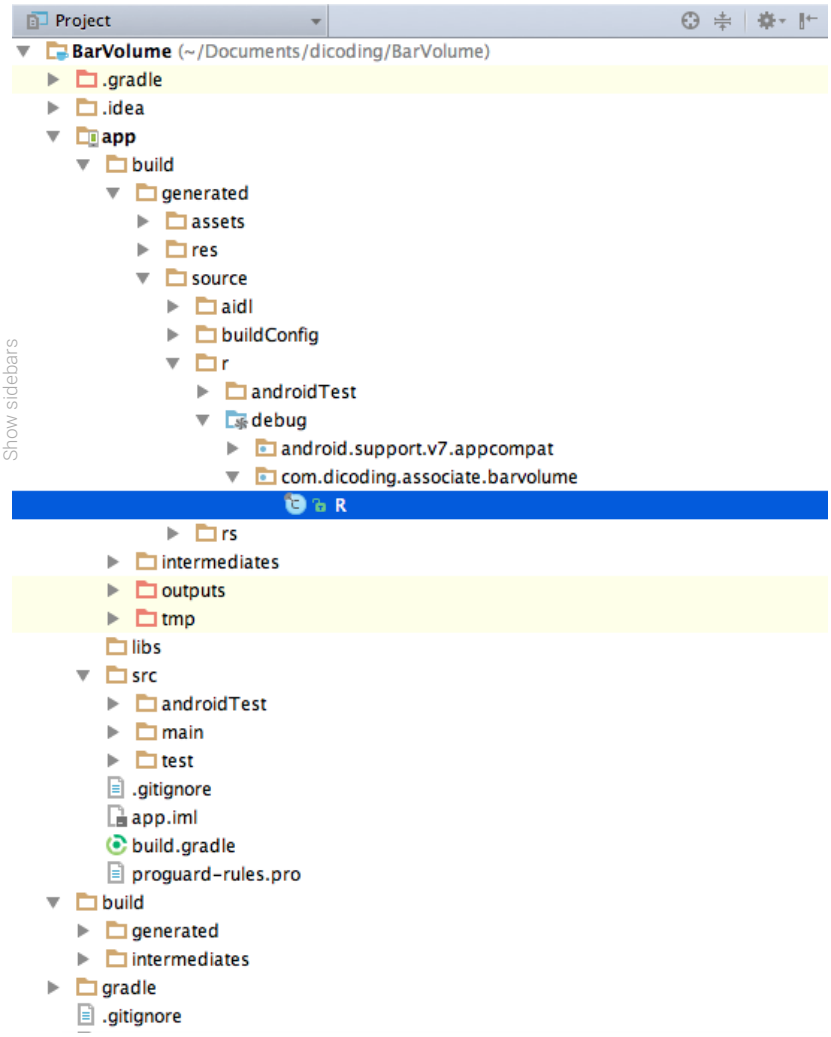
```
1. <Button
2.     android:id="@+id/btn_calculate"
3.     android:layout_width="match_parent"
4.     android:layout_height="wrap_content"
5.     android:text="@string/calculate"/>
```

Penjelasannya sebagai berikut:

```
1. android:id="@+id/btn_calculate"
```

Jika kita memberikan id pada sebuah view maka kita telah memberikan *identifier* untuk view tersebut. Pemberian id ini dimaksudkan agar kita bisa melakukan manipulasi/pengendalian pada *level logic* di komponen seperti activity atau fragment.

Id di atas akan diciptakan di berkas R dan disimpan dalam bentuk *hexa* bertipe data integer public static final int btn_calculate=0x7f0b0057.



Acuan untuk menyusun tampilan pada relativelayout akan dibahas pada modul selanjutnya.

Pembahasan tentang Logika Kode

Kode logika dituliskan ke dalam kelas Java atau Koltin. Di sinilah semua aktifitas dari suatu aplikasi berjalan.

Activity

- Java

```
1. public class MainActivity extends AppCompatActivity
```

Menandakan bahwa kelas Java / Kotlin di atas merupakan sebuah activity karena *inherit* ke *superclass* bernama AppCompatActivity**.**

OnClickListener

- Java

```
1. implements View.OnClickListener
```

Ini adalah *listener* yang kita implementasikan untuk memantau kejadian klik pada komponen tombol (button)

Views

- Java

```
1. private EditText edtWidth;
2. private EditText edtHeight;
3. private EditText edtLength;
4. private Button btnCalculate;
5. private TextView tvResult;
```

Kode di atas mendeklarasikan semua komponen view yang akan dimanipulasi. Kita deklarasikan secara global agar bisa dikenal di keseluruhan bagian kelas.

Show sidebars

nCreate

- Java

```
1. @Override
2. protected void onCreate(Bundle savedInstanceState) {
3.     super.onCreate(savedInstanceState);
4.     setContentView(R.layout.activity_main);
5.     edtWidth = findViewById(R.id.edt_width);
6.     edtHeight = findViewById(R.id.edt_height);
7.     edtLength = findViewById(R.id.edt_length);
8.     btnCalculate = findViewById(R.id.btn_calculate);
9.     tvResult = findViewById(R.id.tv_result);
10.
11.    btnCalculate.setOnClickListener(this);
12. }
```

Metode onCreate() merupakan metode utama pada activity. Di sinilah kita dapat mengatur layout xml. Semua proses inisialisasi komponen yang digunakan akan dijalankan di sini. Pada metode ini kita *casting* semua komponen view yang kita telah deklarasikan sebelumnya, agar dapat kita manipulasi.

SetContentView

- Java

```
1. setContentView(R.layout.activity_main);
```

Maksud baris di atas adalah kelas MainActivity akan menampilkan tampilan yang berasal dari layout activity_main.xml**.**

Casting View

- Java

```
1. edtWidth = findViewById(R.id.edt_width);
```

Maksud dari baris di atas adalah obyek EditTextedtWidth disesuaikan (cast) dengan komponen EditText ber-ID edt_width di layout activity_main.xml melalui metode findViewById().

SetOnClickListener

- Java

```
1. btnCalculate.setOnClickListener(this);
```

Kita memasang *event click listener* untuk obyek btnCalculate sehingga sebuah aksi dapat dijalankan ketika obyek tersebut diklik. Keyword this merujuk pada obyek Activity saat ini yang telah mengimplementasikan listener OnClickListener sebelumnya. Sehingga ketika btnCalculate diklik, maka fungsi onClick akan dipanggil dan melakukan proses yang ada di dalamnya.

Mengambil value dari EditText

- Java

```
String inputLength = edtLength.getText().toString().trim();
String inputWidth = edtWidth.getText().toString().trim();
String inputHeight = edtHeight.getText().toString().trim();
```


ntaks .text.toString() di atas berfungsi untuk mengambil isi dari sebuah EditText kemudian menyimpannya dalam sebuah variabel. ambahan .trim() berfungsi untuk menghiraukan spasi jika ada, sehingga nilai yang didapat hanya berupa angka.

Cek inputan yang kosong

- Java

```
boolean isEmptyFields = false;

if (TextUtils.isEmpty(inputLength)) {
    isEmptyFields = true;
    edtLength.setError("Field ini tidak boleh kosong");
}

if (TextUtils.isEmpty(inputWidth)) {
    isEmptyFields = true;
    edtWidth.setError("Field ini tidak boleh kosong");
}

if (TextUtils.isEmpty(inputHeight)) {
    isEmptyFields = true;
    edtHeight.setError("Field ini tidak boleh kosong");
}
```

Sintaks .isEmpty() berfungsi untuk mengecek apakah inputan dari Editext itu masih kosong. Jika iya, maka kita akan menampilkan pesan *error* dengan menggunakan .setError("Field ini tidak boleh kosong") dan mengganti variabel Boolean isEmptyField menjadi true supaya bisa lanjut ke proses selanjutnya.

Menampilkan data ke TextView

- Java

```
if (!isEmptyFields) {
    double volume = Double.valueOf(inputLength) * Double.valueOf(inputWidth) * Double.valueOf(inputHeight);
    tvResult.setText(String.valueOf(volume));
}
```

Sintaks !isEmptyFields memiliki arti "jika semua inputan tidak kosong". Jika kondisi tersebut terpenuhi, maka langkah selanjutnya yaitu melakukan proses perhitungan. Karena yang didapat dari EditText berupa String maka Anda perlu mengkonversinya terlebih dahulu dengan menggunakan Double.valueOf(). Langkah terakhir yaitu menampikan hasil perhitungan pada TextView tvResult dengan menggunakan .setText(). Di sini dapat kita lihat bahwa kita perlu merubah datanya yang sebelumnya Double menjadi String dengan menggunakan String.valueOf() karena untuk menampilkan data dengan setText() harus berupa String.

Pembahasan SaveInstanceState

- Java

```
1. @Override
2. protected void onSaveInstanceState(Bundle outState) {
3.     super.onSaveInstanceState(outState);
4.     outState.putString(STATE_RESULT, tvResult.getText().toString());
5. }
```

Perhatikan metode onSaveInstanceState**.** Di dalam metode tersebut, hasil perhitungan yang ditampilkan pada tvResult dimasukkan pada undle kemudian disimpan isinya. Untuk menyimpan data disini menggunakan konsep **Key-Value**, dengan STATE_RESULT sebagai *key* dan isi dari Result sebagai *value*. Fungsi onSaveInstanceState akan dipanggil secara otomatis sebelum sebuah Activity hancur. Di sini kita perlu enambahkan onSaveInstanceState karena ketika orientasi berubah, Activity tersebut akan di-*destroy* dan memanggil fungsi onCreate lagi, ehingga kita perlu menyimpan nilai hasil perhitungan tersebut supaya data tetap terjaga dan tidak hilang ketika orientasi berubah.

- Java

```
1. if (savedInstanceState != null){
2.     String hasil = savedInstanceState.getString(STATE_RESULT);
3.     tvResult.setText(hasil);
4. }
```

Pada onCreate inilah kita menggunakan nilai *bundle* yang telah kita simpan sebelumnya pada onSaveInstanceState. Nilai tersebut kita dapatkan dengan menggunakan *Key* yang sama dengan saat menyimpan, yaitu STATE_RESULT. Kemudian kita isikan kembali pada tvResult.