Modul Hari Kedua

Basic Android Application Development

Hanya untuk penggunaan pribadi dan proses pembelajaran di kelas Basic Android Application Development Codepolitan.com

1. Intent

Kemarin kita telah belajar bagaimana membuat layout sebuah aplikasi Android dan menambahkan interaktifitas di dalamnya. Hari ini kita akan belajar menambahkan Activity baru dan berpindah dari satu Activity ke Activity lainnya.

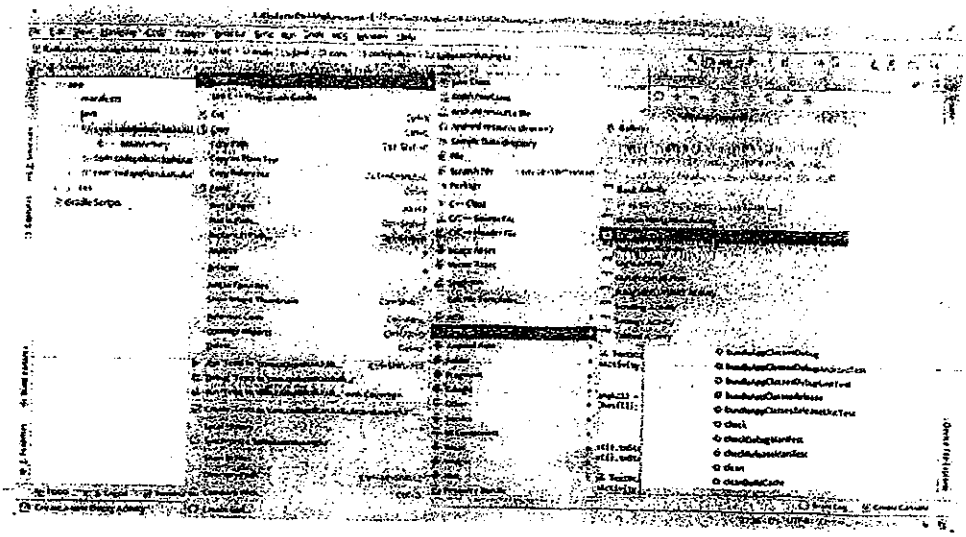
1.1. Explicit Intent

Explicit Intent adalah sebuah istilah saat kita ingi berpindah dari satu Activity ke Activity lain yang masih ada di dalam package sendiri (misalnya dari `MainActivity.java` ke `SecondActivity.java`). Kita akan melanjutkan project Kalkulator Dua Angka untuk menampilkan hasilnya di Activity baru.

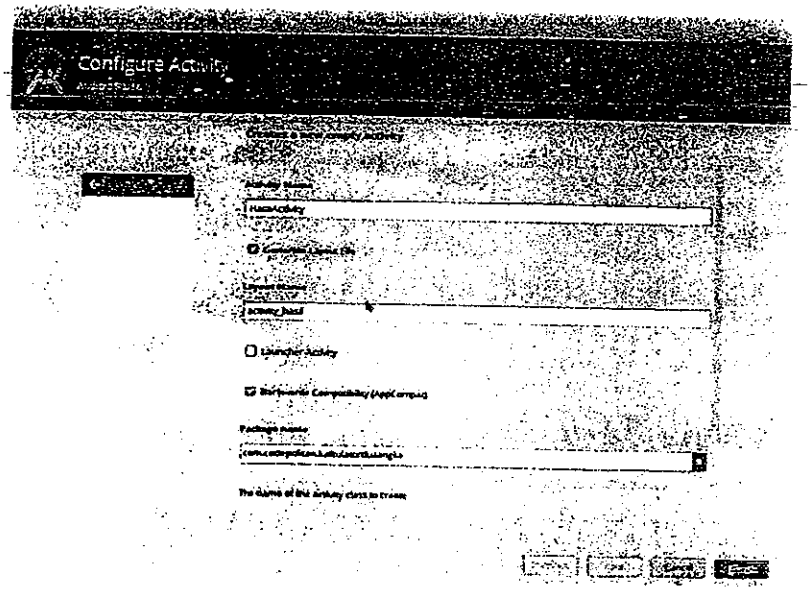
Pertama buat dulu salinan project kemarin kemudian **Open project** di Android Studio.

Membuat Activity Baru

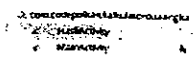
Selanjutnya kita perlu membuat Activity baru untuk menamapilkan hasilnya. Untuk membuat Activity baru caranya klik kanan di folder dengan package name yang kita buat pilih **New > Activity > Empty Activity**:



Pada jendela yang muncul, kita tuliskan `HasilActivity` sebagai nama Activity yang baru. Kemudian klik **Finish**.



Lalu pastikan bahwa Activity yang dibuat berada satu folder dengan MainActivity.



Pindah Activity

Setelah membuat Activity baru, sekarang mari kita coba berpindah dari MainActivity.java ke HasilActivity.java. Ubah isi method tambah menjadi:

```
public void tambah(View view) {
```

```
    String angka1 = etAngka1.getText().toString();
    String angka2 = etAngka2.getText().toString();

    if (TextUtils.isEmpty(angka1) && TextUtils.isEmpty(angka2)) {
        Toast.makeText(MainActivity.this, "Masukan kedua angka terlebih
 dahulu", Toast.LENGTH_SHORT).show();
    }

    Double hasil = Double.valueOf(angka1) + Double.valueOf(angka2);
    tvHasil.setText(String.valueOf(hasil));

    Intent pindahIntent = new Intent(MainActivity.this,
 HasilActivity.class);
    startActivity(pindahIntent);
}
```

Pada contoh kode di atas, kita hanya melakukan dua penambahan untuk melakukan perpindahan Activity:

```
Intent pindahIntent = new Intent(MainActivity.this,
 HasilActivity.class);
startActivity(pindahIntent);
```

Dibaris pertama kita membuat sebuah objek dengan nama `pindahIntent` dari kelas `Intent`. Parameter pertama yang kita kirimkan untuk inisiasi objek bertipe `Intent` adalah sebuah `Context` dan parameter keduanya adalah Activity tujuan.



Saat kita diminta mengirimkan sebuah objek `Context` cukup kirimkan *instance* Activity saat ini yang bisa didapatkan dengan menuliskan `NamaActivity.this`.

Mari kita jalankan aplikasinya untuk menguji apakah sudah bisa berpindah ke HasilActivity.



Yeay, kita sudah berhasil pindah Activity. Namun masih ada beberapa kekurangan.

- Tidak ada tombol back di Toolbar untuk kembali ke halaman sebelumnya
- Belum menampilkan hasil

Mari kita selesaikan satu persatu.

Menampilkan Back

Pada umumnya saat kita membuka Activity baru, di Activity baru tersebut terdapat sebuah tombol untuk kembali ke halaman utama. Kita bisa melakukannya dengan menambahkan sebuah properti di AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.codepolitan.kalkulatorduaangka">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
```

```
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />


    <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".HasilActivity"></activity>
    </application>

</manifest>
```

Perhatikan kode di atas! Setiap kali kita membuat Activity baru, maka AndroidManifest.xml akan menambah sebuah tag <activity></activity>. Agar sebuah Activity memiliki tombol back untuk kembali ke halaman sebelumnya, tambahkan properti android:parentActivityName=".MainActivity" sehingga isi AndroidManifest.xml menjadi:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.codepolitan.kalkulatorduaangka">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />


  <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".HasilActivity"
            android:parentActivityName=".MainActivity"
            ></activity>
    </application>

</manifest>
```

Mari kita coba jalankan lagi aplikasi yang sudah diperbarui. Perhatikan tombol panah yang muncul.



Karena sekarang kita sudah bisa kembali ke MainActivity dari HasilActivity, maka selanjutnya kita akan menampilkan hasil yang didapatkan dari MainActivity ke HasilActivity.

Ubah isi method `tambah()` menjadi:

```
public void tambah(View view) {
    String angka1 = etAngka1.getText().toString();
    String angka2 = etAngka2.getText().toString();

    if (TextUtils.isEmpty(angka1) && TextUtils.isEmpty(angka2)) {
        Toast.makeText(MainActivity.this, "Masukan kedua angka terlebih
 dahulu", Toast.LENGTH_SHORT).show();
    }

    Double hasil = Double.valueOf(angka1) + Double.valueOf(angka2);
    // tvHasil.setText(String.valueOf(hasil));

    Intent pindahIntent = new Intent(MainActivity.this,
 HasilActivity.class);
    pindahIntent.putExtra("HASIL", hasil);
    startActivity(pindahIntent);
}
```

Karena kita sudah tidak menampilkan hasil di MainActivity, maka kode `tvHasil.setText()` kita beri komentar (dapat pula di hapus).

Untuk mengirimkan data ke Activity baru, kita memanfaatkan objek Intent. Objek kita disini bernama pindahIntent sehingga perintah yang kita tulis adalah `pindahIntent.putExtra()`. Parameter pertama yang kita berikan adalah sebuah *key/kata kunci* yang nanti akan digunakan untuk mengambilnya. Parameter kedua adalah variabel atau data yang ingin di kirim, dalam kasus ini adalah variabel hasil yang sudah menampung penjumlahan.

Menampilkan Data di HasilActivity

Sekarang mari kita baca data yang dikirim dari MainActivity. Buka HasilActivity.java dan ubah isi method `onCreate` menjadi:

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

    Double hasilDariMainActivity =
 getIntent().getDoubleExtra("HASIL", 0);
    Log.d("HasilActivity", "Nilai: " + hasilDariMainActivity);
}
```

Kita mengambil data yang dikirim dari MainActivity dengan pemanggilan `getIntent()`. Selanjutnya kita tentukan data apa yang dikirim. Karena di MainActity data yang dikirim bertipe Double maka kita panggil `getDoubleExtra()`.

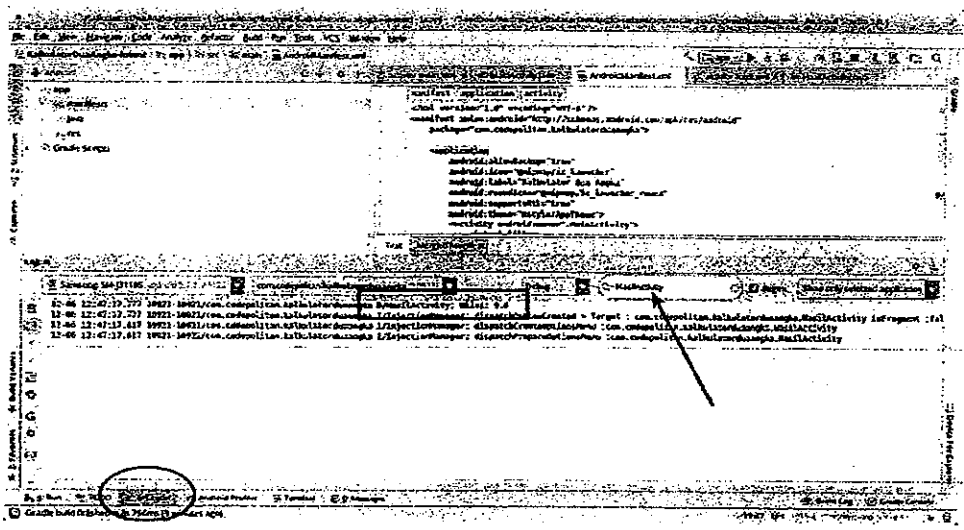


Selalu sesuaikan method untuk membaca data dengan tipe data aslinya. Misalnya, saat memanggil `putExtra()` kita mengirimkan data Double, maka membacanya dengan method `getDoubleExtra()`. Jika mengirimkan data String maka memanggilnya dengan `getStringExtra()`.



Perintah `Log.d()`, tidak mempengaruhi bagian manapun dari aplikasi, tapi berguna untuk melakukan *debugging*.

Mari kita jalankan lagi aplikasi ini sampai membuka Activity baru lalu perhatikan pesan Log-nya.



Di bagian Logcat (kotak merah) kita bisa melihat pesan-pesan Log yang diproduksi semua aplikasi di perangkat Android/emulator kita. Kita bisa mem-filter pesan ini untuk menampilkan hanya TAG tertentu. Pada saat menuliskan perintah `Log.d()`, parameter pertama yang kita tulis adalah TAG.

Perhatikan kotak merah. Terlihat bahwa data yang berasal dari MainActivity sudah diterima oleh HasilActivity. Sekarang saatnya kita tampilkan ke sebuah TextView. Karena kita belum memilikinya, mari kita buat terlebih dahulu.

Buka activity_hasil.xml lalu ubah isinya menjadi:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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"
    tools:context="com.codepolitan.kalkulatorduaangka.HasilActivity">

    <TextView
        android:textSize="100sp"
        android:layout_centerInParent="true"
        android:id="@+id/tvHasil"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</RelativeLayout>
```

Disini kita hanya mengubah ViewGroup dan menambah satu TextView. Jika diperhatikan kita memberikan id yang sama dengan TextView yang ada di MainActivity. Hal ini tidak masalah karena masing-masing Activity hanya membaca layoutnya sendiri-sendiri (dalam hal ini MainActivity membaca activity_main.xml dan HasilActivity membaca activity_hasil.xml).

Lalu di Hasil.Activity kita tambahkan kode untuk mengupdate isi TextView.

```
public class HasilActivity extends AppCompatActivity {

    TextView textViewHasil;

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

        Double hasilDariMainActivity =
 getIntent().getDoubleExtra("HASIL", 0);
        Log.d("HasilActivity", "Nilai: " + hasilDariMainActivity);

        textViewHasil = findViewById(R.id.tvHasil);
        textViewHasil.setText(String.valueOf(hasilDariMainActivity));
    }
}
```



Setiap kali ingin membaca atau mengubah suatu komponen TextView, Button, EditText, dll. (seperti melakukan setText() dan getText()) selalu lakukan findViewById() terlebih dahulu.

Sekarang HasilActivity sudah dapat menampilkan hasil perhitungan yang kita lakukan.

Apakah kamu bisa meneruskan operasi method yang lain?

```
public class MainActivity extends AppCompatActivity {

    EditText etAngka1,etAngka2;
    TextView tvHasil;

    @Override
```

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

        etAngka1 = findViewById(R.id.etAngka1);
        etAngka2 = findViewById(R.id.etAngka2);
        tvHasil = findViewById(R.id.tvHasil);
    }


    public void tambah(View view) {
        String angka1 = etAngka1.getText().toString();
        String angka2 = etAngka2.getText().toString();

        if (TextUtils.isEmpty(angka1) && TextUtils.isEmpty(angka2)) {
            Toast.makeText(MainActivity.this, "Masukan kedua angka
terlebih dahulu", Toast.LENGTH_SHORT).show();
        }

        Double hasil = Double.valueOf(angka1) + Double.valueOf(angka2);
        Intent pindahIntent = new Intent(MainActivity.this,
HasilActivity.class);
        pindahIntent.putExtra("HASIL", hasil);
        startActivity(pindahIntent);
    }

    public void kurang(View view) {
        String angka1 = etAngka1.getText().toString();
        String angka2 = etAngka2.getText().toString();

        if (TextUtils.isEmpty(angka1) && TextUtils.isEmpty(angka2)) {
            Toast.makeText(MainActivity.this, "Masukan kedua angka
terlebih dahulu", Toast.LENGTH_SHORT).show();
        }

        Double hasil = Double.valueOf(angka1) + Double.valueOf(angka2);

        Intent pindahIntent = new Intent(MainActivity.this,
HasilActivity.class);
        pindahIntent.putExtra("HASIL", hasil);
        startActivity(pindahIntent);
    }

    public void kali(View view) {
        String angka1 = etAngka1.getText().toString();
```

```
        String angka2 = etAngka2.getText().toString();

        if (TextUtils.isEmpty(angka1) && TextUtils.isEmpty(angka2)) {
            Toast.makeText(MainActivity.this, "Masukan kedua angka
terlebih dahulu", Toast.LENGTH_SHORT).show();
        }

        Double hasil = Double.valueOf(angka1) + Double.valueOf(angka2);
        Intent pindahIntent = new Intent(MainActivity.this,
HasilActivity.class);
        pindahIntent.putExtra("HASIL", hasil);
        startActivity(pindahIntent);    }

    public void bagi(View view) {
        String angka1 = etAngka1.getText().toString();
        String angka2 = etAngka2.getText().toString();

        if (TextUtils.isEmpty(angka1) && TextUtils.isEmpty(angka2)) {
            Toast.makeText(MainActivity.this, "Masukan kedua angka
terlebih dahulu", Toast.LENGTH_SHORT).show();
        }

        Double hasil = Double.valueOf(angka1) + Double.valueOf(angka2);
        Intent pindahIntent = new Intent(MainActivity.this,
HasilActivity.class);
        pindahIntent.putExtra("HASIL", hasil);
        startActivity(pindahIntent);    }
}
```

1.2. Implicit Intent

Sebelumnya kita telah belajar bagaimana berpindah Activity di project sendiri menggunakan explicit Intent. Selain pindah Activity di project sendiri, kita juga dapat pindah Activity ke aplikasi lain. Misalnya dari aplikasi yang kita buat ingin mengambil data dari Map sehingga kita perlu membuka aplikasi Google Maps. Untuk pindah Activity ke aplikasi lain kita bisa memanfaatkan implicit intent.

Untuk mempelajari implicit intent, silahkan baca panduan 2.3 Aktivitas dan Maksud Intent[1].

[1] https://google-developer-training.gitbooks.io/android-developer-fundamentals-course-practicals/content/idn/Unit%201/23_p_activities_and_implicit_intents.html

2. Mengenal Penggunaan Fragment

Mari kita buat project baru dengan ketentuan dengan nama Fragment, Minimum SDK Level 16, dan sebuah Empty Activity.

Buka file activity_main.xml lalu tambahkan kode berikut:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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"
    tools:context="com.codepolitan.fragment.MainActivity">

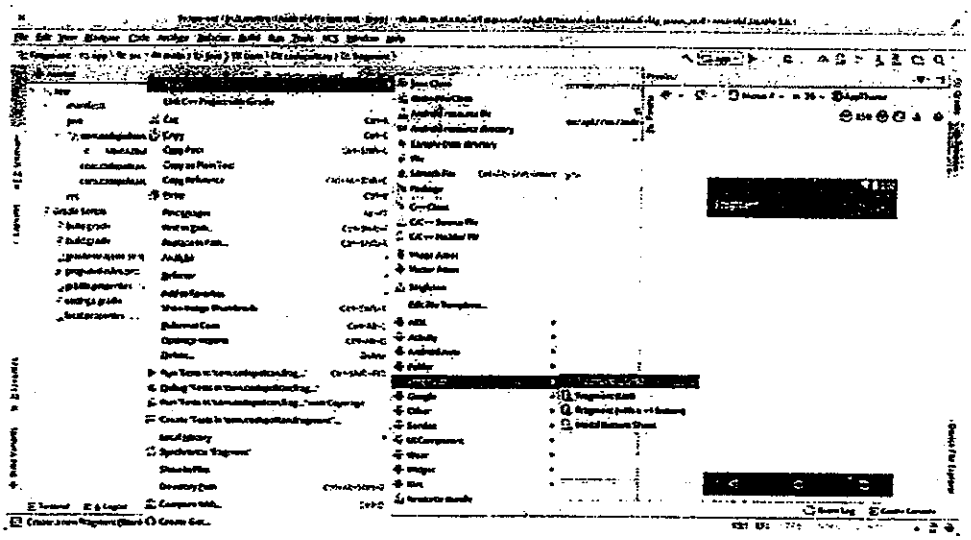
    <FrameLayout
        android:id="@+id/container"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</RelativeLayout>
```

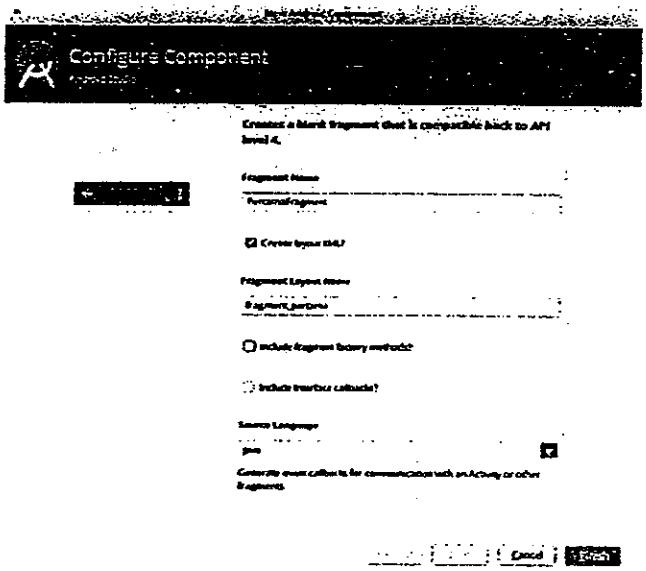
Disini kita menambahkan sebuah komponen baru bernama `FrameLayout` dengan id `container`. FrameLayout merupakan salah satu jenis ViewGroup seperti halnya RelativeLayout. Namun perbedaannya FrameLayout hanya dapat menyusun komponen didalamnya saling menumpuk sesuai urutan penulisan. Oleh karena itu FrameLayout sering digunakan sebagai tempat menempelkan Fragment.

2.1. Membuat Fragment

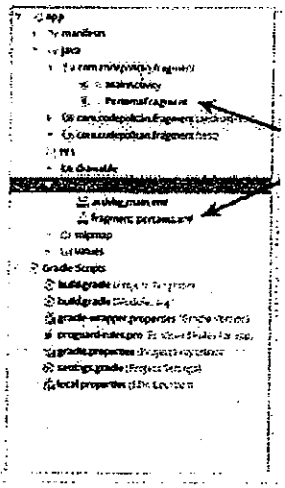
Untuk membuat Fragment baru caranya hampir sama dengan membuat Activity baru. Klik kanan pada folder dengan *package name* kita lalu pilih New > Fragment > Blank Fragment.



Pada jendela yang muncul berikan nama **PertamaFragment**, pastikan *Create layout XML* tercentang sehingga kita akan dibuatkan sebuah layout baru. Lalu pastikan pula kita menghilangkan tanda centang *Include fragment factory methods* dan *Include interface callback*. Klik Finish.



Pastikan kita sudah memiliki dua buah file baru. Satu bernama PertamaFragment.java dan satu lagi bernama fragment_pertama.xml.



Fragment merupakan salah satu komponen Android yang mirip dengan Activity dimana ia juga memiliki layout sendiri (fragment_pertama.xml). Hanya saja sebuah Fragment tidak dapat berdiri sendiri dan membutuhkan sebuah Activity sebagai inangnya. Oleh karena itu, di activity_main.xml kita menambahkan sebuah *container* sebagai tempat menempel PertamaFragment.

Agar nanti terlihat, mari kita ubah terlebih dahulu isi file fragment_pertama.xml menjadi:

```
<RelativeLayout
    android:background="@color/colorAccent"
    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="com.codepolitan.fragment.PertamaFragment">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Ini Fragment Pertama" />

</RelativeLayout>
```

Selanjutnya mari kita tempelkan PertamaFragment ke FrameLayout ber-id container di activity_main.xml. Buka file MainActivity.java, lalu ubah isi method onCreate().

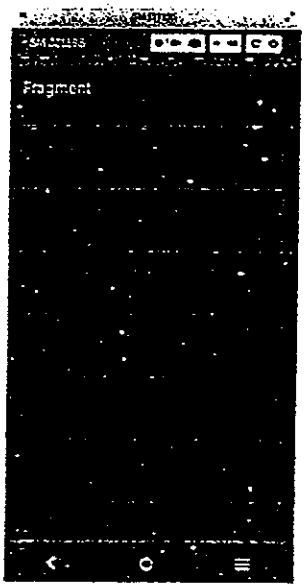
```
public class MainActivity extends AppCompatActivity {
```

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


getSupportFragmentManager().beginTransaction().add(R.id.container, new
PertamaFragment()).commit();
    }

}
```

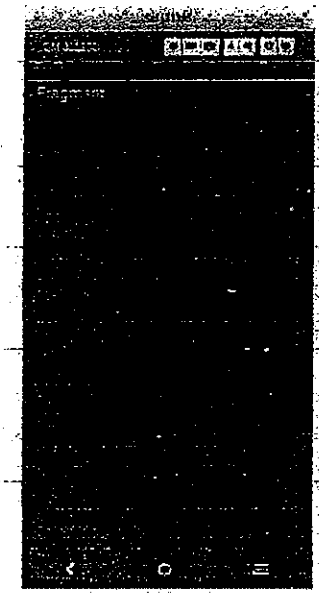
Ayo kita run aplikasi ini.





Method add akan menambahkan Fragment di atas Fragment yang lain. Sehingga di sebuah container kita bisa memiliki beberapa Fragment. Jika hanya ingin memiliki satu Fragment di satu container, ganti add dengan replace.

Pada contoh ini, MainActivity hanya bertindak sebagai tempat menempel Fragment. Layout yang kita lihat (background berwarna pink) merupakan layout milik Fragment.



2.2. Pindah Fragment

Masih di project yang sama, tambahkan tiga tombol di activity_main.xml. Ketiga tombol ini akan berfungsi sebagai navigasi untuk berpindah antar fragment.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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"
    tools:context="com.codepolitan.fragment.MainActivity">

    <FrameLayout
        android:layout_above="@id/navigasi"
        android:id="@+id/container"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <LinearLayout
        android:id="@+id/navigasi"
        android:layout_alignParentBottom="true"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <Button
            android:onClick="tampilFragmentPertama"
```

```
            android:layout_weight="1"
            android:text="Pertama"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <Button
            android:onClick="tampilFragmentKedua"
            android:layout_weight="1"
            android:text="Kedua"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <Button
            android:onClick="tampilFragmentKetiga"
            android:layout_weight="1"
            android:text="Ketiga"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

    </LinearLayout>

</RelativeLayout>
```

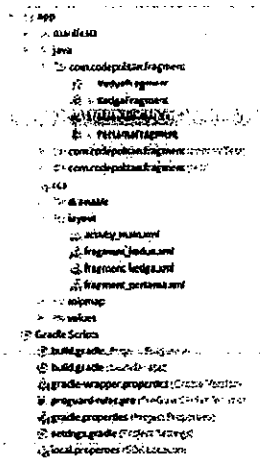
Karena kita juga menambahkan properti onClick di masing-masing tombol, maka kita juga perlu menambah method berikut di MainActivity.java:

```
public void tampilFragmentPertama(View view) {
}

public void tampilFragmentKedua(View view) {
}

public void tampilFragmentKetiga(View view) {
}
```

Karena kita baru memiliki satu Fragment, maka tambahkan dua Fragment lagi seperti tadi. Jangan lupa untuk mengubah teks dan warna background masing-masing layout Fragment yang dibuat agar terlihat saat ditempelkan



Lalu, kita perbarui ketiga method yang baru kita buat.

```
public void tampilFragmentPertama(View view) {

 getSupportFragmentManager().beginTransaction().add(R.id.container, new
 PertamaFragment()).commit();
}

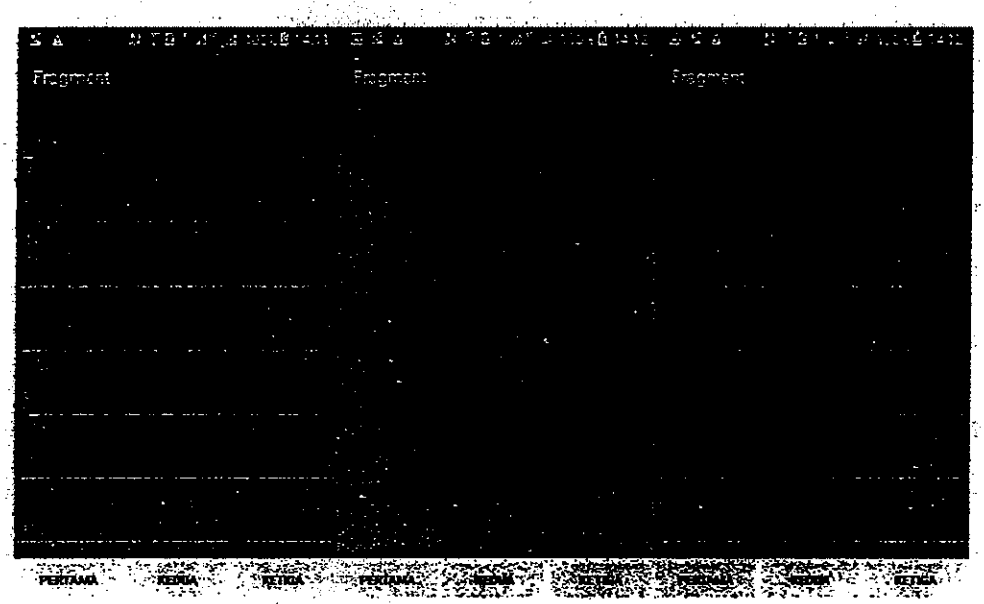
public void tampilFragmentKedua(View view) {

 getSupportFragmentManager().beginTransaction().add(R.id.container, new
 KeduaFragment()).commit();
}

public void tampilFragmentKetiga(View view) {

 getSupportFragmentManager().beginTransaction().add(R.id.container, new
 KetigaFragment()).commit();
}
```

Jalankan lagi aplikasi kita dan coba klik tiap tombol.



2.3. Bekerja dengan Fragment

Kita sudah belajar bagaimana membuat Fragment dan melakukan perpindahan antar Fragment di dalam sebuah Activity. Namun, Fragment yang kita buat masih kosong. Sekarang kita akan menambahkan konten didalamnya.

Konten yang akan kita buat mirip dengan materi pada hari pertama dimana sekarang kita akan membuat aplikasi penghitung luas persegi panjang, segitiga, dan lingkaran. PertamaFragment akan menampilkan program penghitung luas persegi panjang, FragmentKedua untuk segitiga, dan FragmentKetiga untuk lingkaran.

Pertama update isi fragment_pertama.xml (layout milik PertamaFragment.java):

```
<RelativeLayout
    android:layout_margin="16dp"
    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="com.codepolitan.fragment.PertamaFragment">

    <TextView
        android:id="@+id/judulPersegiPanjang"
        android:textSize="20sp"
        android:layout_centerHorizontal="true"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="Luas Persegi Panjang" />

    <EditText
        android:layout_below="@+id/judulPersegiPanjang"
        android:hint="Masukkan Panjang"
        android:id="@+id/panjang"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <EditText
        android:layout_below="@+id/panjang"
        android:hint="Masukkan Lebar"
        android:id="@+id/lebar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/btnHitung"
        android:layout_centerHorizontal="true"
        android:text="Hitung"
        android:layout_below="@+id/lebar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <TextView
        android:textSize="100sp"
        android:gravity="center"
        android:text="0"
        android:layout_below="@+id/btnHitung"
        android:id="@+id/tvHasil"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</RelativeLayout>
```

Buka PertamaFragment.java lalu tambahkan deklarasi berikut:

```
public class PertamaFragment extends Fragment {

    TextView textViewHasil;
    Button btnHitung;
    EditText etPanjang, etLebar;

    public PertamaFragment() {
```

```
    // Required empty public constructor
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_pertama, container,
false);
    }

}
```

Setelah membuat deklarasi objek, seperti halnya di Activity kita harus mengarahkan objek-objek ini ke komponen dengan id tertentu menggunakan findViewById.

Apabila di Activity kita menulisnya di method `onCreate()` maka di Fragment kita akan menulisnya di method `onCreateView()`.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View v = inflater.inflate(R.layout.fragment_pertama, container,
false);

    textViewHasil = v.findViewById(R.id.tvHasil);
    btnHitung = v.findViewById(R.id.btnHitung);
    etPanjang = v.findViewById(R.id.etPanjang);
    etLebar = v.findViewById(R.id.etLebar);

    return v;
}
```

Karena di Fragment kita tidak bisa memanggil method `findViewById()` secara langsung, maka kita akan memanfaatkan objek yang dikembalikan oleh `inflater.inflate()` dan menyimpannya ke dalam sebuah objek bertipe View, disini kita namakan v tapi bisa apa saja.

Dengan objek v yang kita buat, kita bisa memanggil `findViewById()`. Terkahir, kita **return** objek v ini.

Untuk mendeteksi klik disebuah tombol, kita tidak bisa melakukan hal yang sama dengan Activity (melalui properti `onClick()` di XML). Oleh karena itu, dibawah baris `findviewById()` dan sebelum **return**, tambahkan kode berikut:

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {

    View v = inflater.inflate(R.layout.fragment_pertama, container,
 false);

    textViewHasil = v.findViewById(R.id.tvHasil);
    btnHitung = v.findViewById(R.id.btnHitung);
    etPanjang = v.findViewById(R.id.etPanjang);
    etLebar = v.findViewById(R.id.etLebar);

    btnHitung.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Double panjang =
 Double.valueOf(etPanjang.getText().toString());
            Double lebar = Double.valueOf(etLebar.getText().toString());

            Double hasil = panjang * lebar;

            textViewHasil.setText(String.valueOf(hasil));
        }
    });

    return v;
}
```

Atau dapat pula kita tulis menjadi:

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View v = inflater.inflate(R.layout.fragment_pertama, container,
 false);

    textViewHasil = v.findViewById(R.id.tvHasil);
    btnHitung = v.findViewById(R.id.btnHitung);
```

```
        etPanjang = v.findViewById(R.id.etPanjang);
        etLebar = v.findViewById(R.id.etLebar);

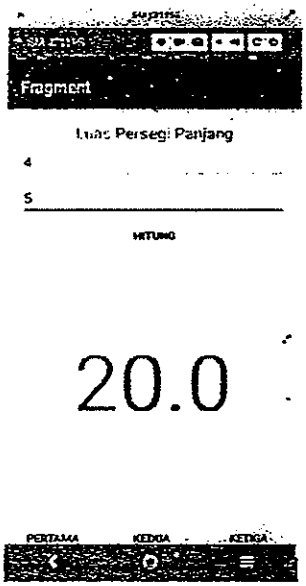
        btnHitung.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                hitungLuas();
            }
        });

        return v;
    }

    private void hitungLuas() {
        Double panjang = Double.valueOf(etPanjang.getText().toString());
        Double lebar = Double.valueOf(etLebar.getText().toString());

        Double hasil = panjang * lebar;

        textViewHasil.setText(String.valueOf(hasil));
    }
```



Bisakah kamu lengkapi kedua Fragment yang lain?