Modul 4 – Threads

Kita buat sebuah proyek aplikasi bernama MyAsyncTask dengan **Empty Activity** (konfigurasi dasar) seperti pada sebelumnya.

1. Jika sudah, kondisikan agar activity_main.xml menjadi seperti ini:

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/an
   droid"
3.     xmlns:tools="http://schemas.android.com/tools"
4.     android:layout_width="match_parent"
5.     android:layout_height="match_parent"
6.     android:paddingBottom="@dimen/activity_vertical_margin"
7.     android:paddingLeft="@dimen/activity_horizontal_margin"
8.     android:paddingRight="@dimen/activity_horizontal_margin"
9.     android:paddingTop="@dimen/activity_vertical_margin"
10.    tools:context="com.dicoding.myserviceapp.MainActivity">
11.    <TextView
12.        android:id="@+id/tv_status"
13.        android:layout_width="wrap_content"
14.        android:layout_height="wrap_content"
15.        android:text="Status Aplikasi"/>
16. </RelativeLayout>
```

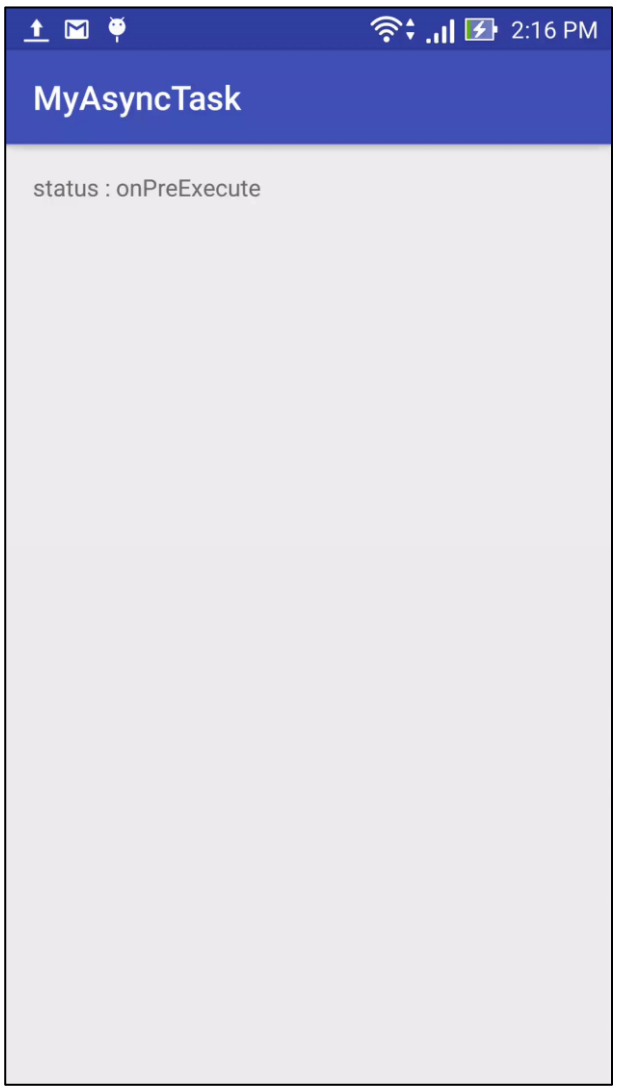
Kita memiliki sebuah obyek TextView berID **tv_status** yang akan menampilkan proses yang sedang dijalankan oleh obyek AsyncTask.

2. Selanjutnya lengkapi kode pada MainActivity sehingga menjadi seperti ini:

```
1. public class MainActivity extends AppCompatActivity {
2.     public static final String DEMO_ASYNC = "DemoAsync";
3.     private TextView tvStatus;
4.     @Override
5.     protected void onCreate(Bundle savedInstanceState) {
6.         super.onCreate(savedInstanceState);
7.         setContentView(R.layout.activity_main);
8.         tvStatus = (TextView)findViewById(R.id.tv_status);
9.         DemoAsync demoAsync = new DemoAsync();
10.        demoAsync.execute("Halo Ini Demo AsyncTask");
11.    }
12.    private class DemoAsync extends AsyncTask<String, Void, String>{
13.        @Override
14.        protected void onPreExecute() {
15.            super.onPreExecute();
16.            tvStatus.setText("status : onPreExecute");
17.        }
18.        @Override
19.        protected String doInBackground(String... params) {
20.            Log.d(DEMO_ASYNC, "status : doInBackground");
21.            try{
22.                Thread.sleep(5000);
23.            }catch (Exception e){
```

```
24.                 Log.d(DEMO_ASYNC, e.getMessage());
25.             }
26.             return params[0];
27.         }
28.         @Override
29.         protected void onPostExecute(String s) {
30.             super.onPostExecute(s);
31.             tvStatus.setText("status : onPostExecute : "+s);
32.         }
33.     }
34. }
```

3. Sekarang, silakan jalankan aplikasi MyAsyncTask dan perhatikan obyek TextView akan melakukan perubahan status terhadap obyek DemoAsyncTask yang dijalankan.



Pada metode doInBackground(), kita hanya menulis *log* prosesnya saja. Bukan sebuah

pendekatan yang baik untuk mengakses komponen antarmuka yang berada pada *thread* utama (*main thread*) ke dalam *background thread*.

Bedah Kode :

Proyek yang kita kerjakan adalah contoh sederhana dari penggunaan AsyncTask untuk membuat proses berjalan secara *asynchronous* dan tetap bisa berkomunikasi (synchronize) dengan *ui thread*. Setiap eksekusi dari metode dari AsyncTask yang kita buat kita pantau dengan menggunakan sebuah textview tvStatus.

```
1.  DemoAsync demoAsync = new DemoAsync();
2.  demoAsync.execute("Halo Ini Demo AsyncTask");
```

Baris diatas akan membuat obyek DemoAsync dan menjanlakan AsyncTask dengan inputan berupa obyek string.

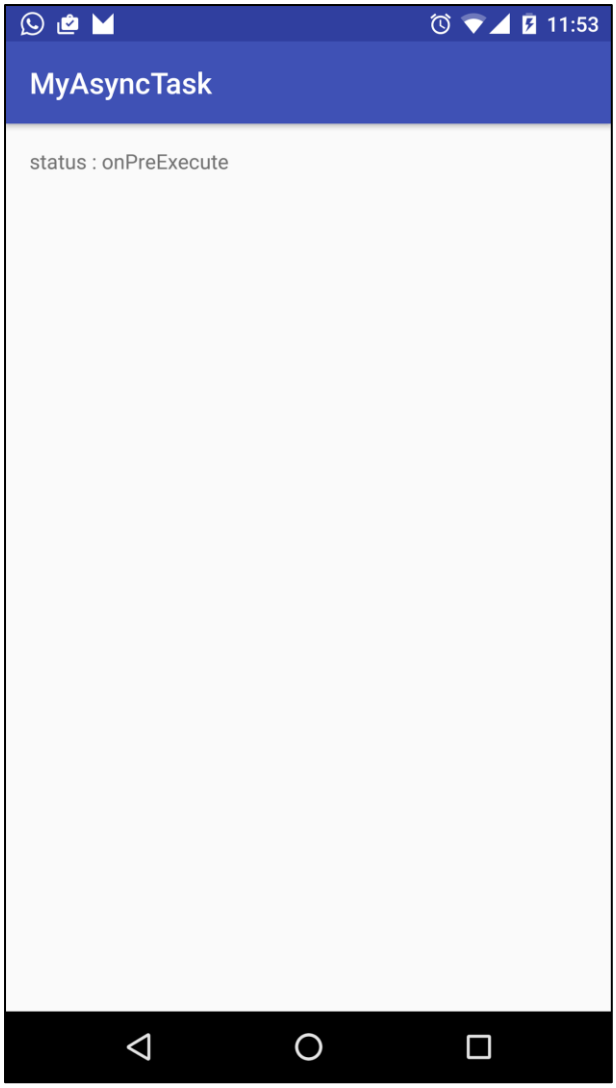
Proses eksekusi kode AsyncTask diatas diawali dengan private class DemoAsync extends AsyncTask{ ....

AsyncTask ini memiliki 3 tipe generik.

- **Param** : Menerima inputan obyek string.
- **Progress** : Void ⇒ Merupakan obyek dari void dengan kata lain, tidak ada *progress* yang akan ditampilkan.
- **Result** : Hasil dari proses berupa obyek string.

```
1. @Override
2. protected void onPreExecute() {
3.         super.onPreExecute();
4.         tvStatus.setText("status : onPreExecute");
5. }
```

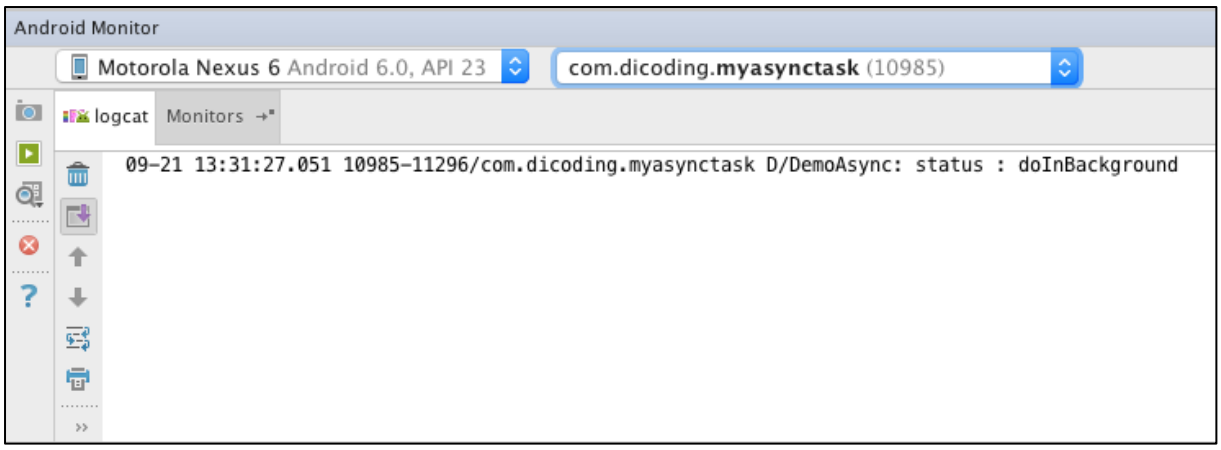
Pada metode onPreExecute(), kita hanya menuliskan status onPreExecute kedalam obyek TextView untuk menandakan bahwa metode onPreExecute telah dijalankan.




```
1. @Override
2. protected String doInBackground(String... params) {
3.     Log.d(DEMO_ASYNC, "status : doInBackground");
4.     try{
5.         Thread.sleep(5000);
6.     }catch (Exception e){
7.         Log.d(DEMO_ASYNC, e.getMessage());
8.     }
9.     return params[0];
10. }
```

Kita melakukan proses *sleep/idle* selama **5 detik (5000 miliseconds)** dan mengembalikan nilai `param[0]` (param indeks ke 0) yang bernilai 'Halo ini demo AsyncTask'. Jika diperhatikan, ada tanda `...` dalam param tersebut, ini merupakan bentuk lain dari array dan menunjukkan bahwa inputan dari sebuah `AsyncTask` bisa lebih dari satu.

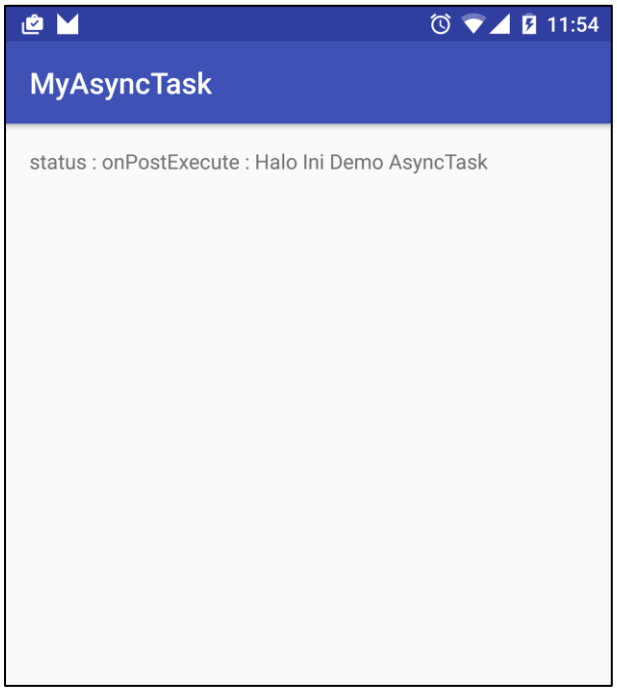
Aturan kedua menjelaskan, jangan melakukan manipulasi terhadap *ui toolkit* pada *worker thread* yang berjalan secara *asynchronous*. Oleh karena itu, kita hanya bisa memantau proses yang terjadi di dalam *log* di *tab* android monitor pada Android Studio seperti berikut:



Terakhir, metode onPostExecute() akan menampilkan hasil proses yang dilakukan di **doInBackground()** sebagai berikut :

```
1. @Override
2. protected void onPostExecute(String s) {
3.     super.onPostExecute(s);
4.     tvStatus.setText("status : onPostExecute : "+s);
5. }
```

Nilai dari string s adalah 'Halo ini Demo AsyncTask' dan hasilnya akan menjadi seperti berikut :



Sederhana bukan? Dengan menggunakan *asynchronous task* untuk menyelesaikan proses yang memakan waktu dan membutuhkan komputasi intensif, maka aplikasi akan tetap responsif.

Selamat! Kamu sudah belajar cukup jauh tentang fundamental aplikasi Android.

Lembar Kerja Praktikum

NPM:	Nama Asisten:
Nama:	Nilai:
Kelas:	

Anda diminta untuk membuat Apps seperti pada gambar di bawah ini. Aplikasi berikut terdiri dari TextView dan Button. Ketika aplikasi dijalankan akan muncul tulisan **Hello world!** Pada TextView, kemudian jika Button **Press Me** di klik aksi yang terjadi adalah akan merubah tulisan pada TextView menjadi **Button Pressed**. Dengan catatan perubahan terjadi setelah 20 detik Button di klik.

Berikut adalah code ketika Button di click:

```
public void buttonClick(View view) {
        long endTime = System.currentTimeMillis() + 20*1000;

        while (System.currentTimeMillis() < endTime) {
            synchronized (this) {
                try {
                    wait(endTime - System.currentTimeMillis());
                } catch (Exception e) {
                }
            }
        }
        TextView myTextView =
                (TextView)findViewById(R.id.myTextView);
        myTextView.setText("Button Pressed");
    }
```

Aplikasi yang Anda buat akan menyebabkan **Error.** Hal ini dikarenakan respon dari aplikasi tersebut melebihi 5 detik (idle selama 20 detik).

Tugas Anda adalah **Perbaiki Aplikasi tersebut agar dapat berjalan dengan baik dengan menggunakan Thred dan Handler maupun AsyncTask** (Ketika Button di klik 20 detik kemudian TextView akan berubah).