

LAPORAN PRAKTIKUM

PEMROGRAMAN BERBASIS MOBILE

PERTEMUAN KE-6



Disusun Oleh :

NAMA : Raden Isnawan Argi Aryasatya
NIM : 195410257
JURUSAN : Informatika
JENJANG : S1
KELAS : 5

Laboratorium Terpadu
Sekolah Tinggi Manajemen Informatika Komputer
AKAKOM
YOGYAKARTA

2021

PERTEMUAN KE-6 (FRAGMENT)

TUJUAN

Mahasiswa dapat membuat aplikasi dengan menggunakan fragment.

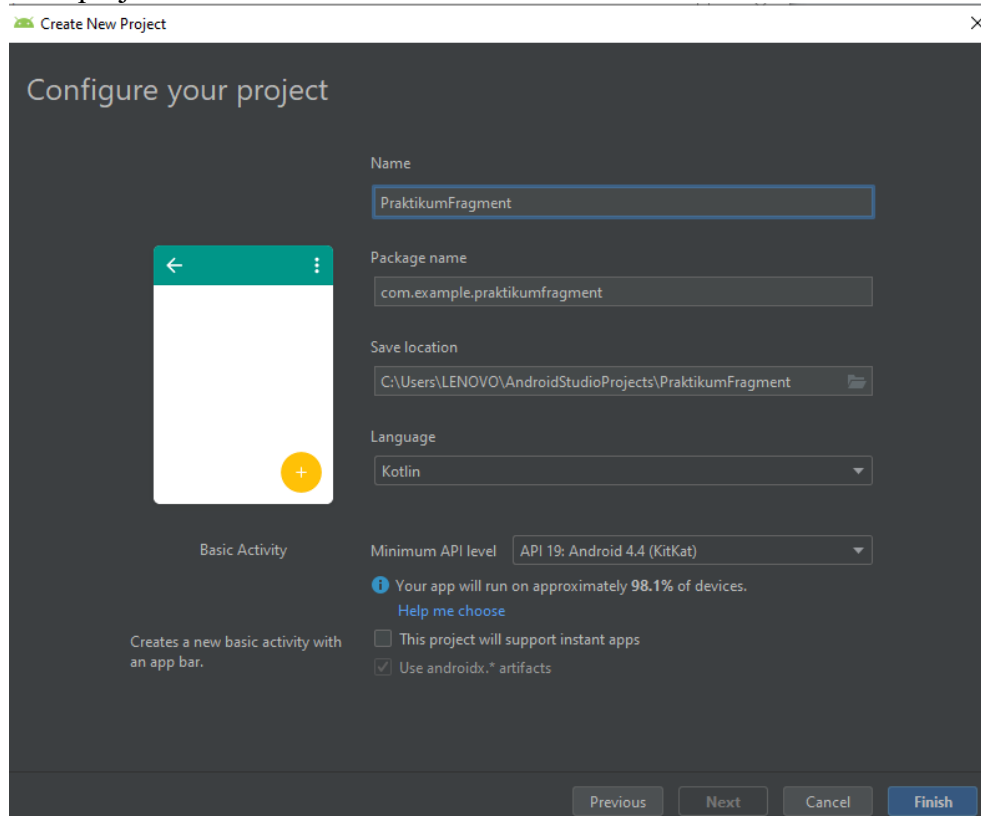
DASAR TEORI

Fragment mewakili perilaku atau bagian dari antarmuka pengguna dalam `FragmentActivity`. Kita bisa mengombinasikan beberapa fragmen dalam satu aktivitas untuk membangun UI multipanel dan menggunakan kembali sebuah fragmen dalam beberapa aktivitas. Kita bisa menganggap fragmen sebagai bagian modular dari aktivitas, yang memiliki daur hidup sendiri, menerima kejadian masukan sendiri, dan yang bisa kita tambahkan atau hapus saat aktivitas berjalan (semacam "subaktivitas" yang bisa digunakan kembali dalam aktivitas berbeda).

Fragmen harus selalu tersemat dalam aktivitas dan daur hidup fragmen secara langsung dipengaruhi oleh daur hidup aktivitas host-nya. Misalnya, saat aktivitas dihentikan sementara, semua fragmen di dalamnya juga dihentikan sementara, dan bila aktivitas dimusnahkan, semua fragmen juga demikian. Akan tetapi, saat aktivitas berjalan (dalam status daur hidup dilanjutkan), Kita bisa memanipulasi setiap fragmen secara terpisah, seperti menambah atau membuangnya. Saat melakukan transaksi fragmen, Kita juga bisa menambahkannya ke back-stack yang dikelola oleh aktivitas—setiap entri back-stack merupakan catatan transaksi fragmen yang terjadi. Dengan back-stack pengguna dapat membalikkan transaksi fragmen (mengarah mundur), dengan menekan tombol Kembali.

PRAKTIK

1. Kita akan membuat aplikasi yang menggunakan fragment dan komunikasi data dengan konsep `ViewModel`.
2. Buat project baru.



3. Langkah pertama, cek pada dependencies

```
26 dependencies {
27     implementation fileTree(dir: 'libs', include: ['*.jar'])
28     implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
29     implementation 'androidx.appcompat:appcompat:1.0.2'
30     implementation 'androidx.core:core-ktx:1.0.2'
31     implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
32     implementation 'com.google.android.material:material:1.0.0'
33     implementation 'androidx.lifecycle:lifecycle-extensions:2.0.0'
34     implementation 'androidx.legacy:legacy-support-v4:1.0.0'
35     testImplementation 'junit:junit:4.12'
36     androidTestImplementation 'androidx.test.ext:junit:1.1.0'
37     androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.1'
38 }
```

Penjelasan:

- **androidx.appcompat:appcompat** = base class untuk activity yang harus menggunakan beberapa fitur platform baru pada perangkat Android lama
- **com.google.android.material:material** = Komponen material untuk Android adalah static library yang dapat kita tambahkan ke aplikasi Android untuk menggunakan API yang menyediakan implementasi material design
- **androidx.constraintlayout:constraintlayout** = ConstraintLayout adalah pengelola tata letak untuk Android yang memungkinkan kita memposisikan dan mengukur widget dengan cara yang fleksibel
- **androidx.lifecycle:lifecycle-extensions** = Komponen berbasis Lifecycle melakukan tindakan sebagai respons terhadap perubahan status siklus proses komponen lain, seperti aktivitas dan fragmen. Komponen-komponen ini membantu kita menghasilkan kode yang lebih rapi dan sering kali lebih ringan, yang lebih mudah dipelihara.
- **androidx.legacy:legacy-support-v4** = static library yang dapat kita tambahkan ke aplikasi Android untuk menggunakan API yang tidak tersedia untuk versi platform lama atau API utilitas yang bukan merupakan bagian dari API framework.

4. Kemudian, buka pada activity mail.xml. Ubah kodenya sehingga menjadi seperti berikut

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.coordinatorlayout.widget.CoordinatorLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context=".MainActivity">
9
10     <com.google.android.material.appbar.AppBarLayout
11         android:layout_width="match_parent"
12         android:layout_height="wrap_content"
13         android:theme="@style/AppTheme">
14
15         <com.google.android.material.tabs.TabLayout
16             android:id="@+id/tabs"
17             android:layout_width="match_parent"
18             android:layout_height="wrap_content"
19             android:background="?attr/colorPrimary"
20             app:tabTextColor="@android:color/background_light"/>
21     </com.google.android.material.appbar.AppBarLayout>
22
23     <androidx.viewpager.widget.ViewPager
24         android:id="@+id/view_pager"
25         android:layout_width="match_parent"
26         android:layout_height="match_parent"
27         app:layout_behavior="@string/appbar_scrolling_view_behavior"
28     />
29 </androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Penjelasan:

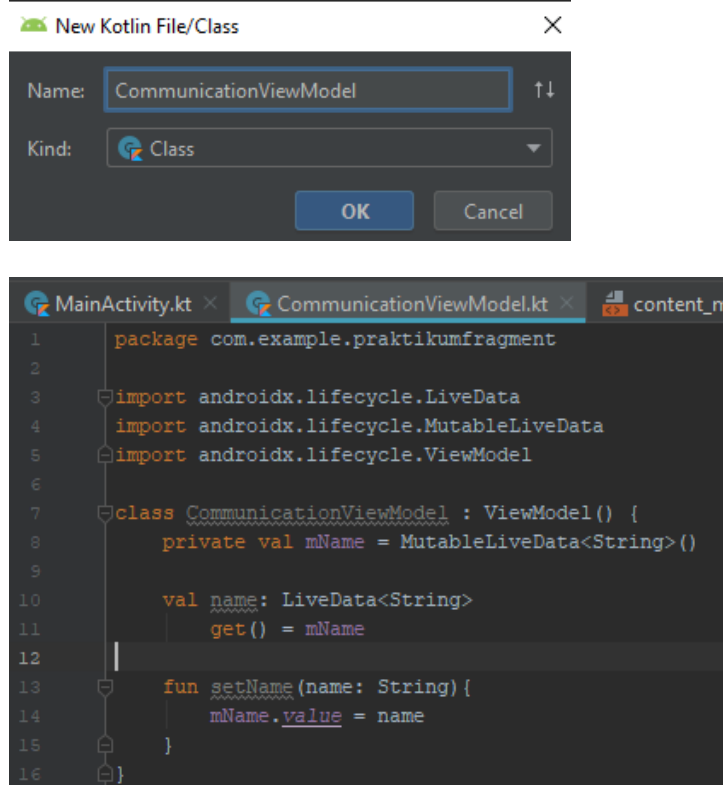
Pertama-tama, kita buat dulu beberapa UI untuk aplikasi demo ini. Kali ini kita menggunakan CoordinatorLayout sebagai metode pengatur tampilan layout. **CoordinatorLayout** adalah super-powered FrameLayout (FrameLayout dengan fitur dan jangkauan yang lebih banyak dan luas). CoordinatorLayout ditujukan untuk dua kasus penggunaan utama: Sebagai dekorasi top-level application atau layout chrome serta juga sebagai container untuk interaksi tertentu dengan satu atau beberapa child views.

AppBarLayout adalah LinearLayout vertikal yang mengimplementasikan banyak fitur dari konsep desain material bar aplikasi

TabLayout digunakan untuk mengimplementasikan tab horizontal. TabLayout dirilis oleh Android setelah diberhentikannya fitur ActionBar. TabLayout diperkenalkan di design support library yang berguna untuk mengimplementasikan tab. Tab dibuat menggunakan method `newTab()` dari kelas TabLayout.

ViewPager adalah widget yang memungkinkan pengguna untuk menggeser ke kiri atau kanan untuk melihat layar baru. Dengan kata lain, ini ViewPager adalah sebuah cara atau method untuk menampilkan banyak tab kepada pengguna. ViewPager juga memiliki kemampuan untuk secara dinamis menambah dan menghapus halaman (atau tab) kapan saja.

5. Buat sebuah class untuk view model



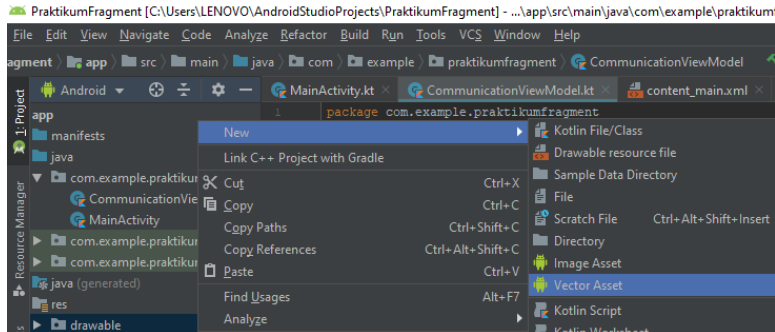
Penjelasan:

- **LiveData** adalah kelas data holder yang dapat diamati dalam suatu lifecycle. Ini berarti bahwa Observer dapat ditambahkan berpasangan dengan LifecycleOwner, dan Observer akan diberi tahu tentang modifikasi data yang di-wrap hanya jika LifecycleOwner dalam keadaan aktif.
- **MutableLiveData** adalah sebuah kelas yang memperluas class type LiveData. MutableLiveData biasanya digunakan untuk menyediakan metode `postValue()` , `setValue()` secara publik, yaitu sesuatu yang tidak disediakan oleh kelas LiveData.

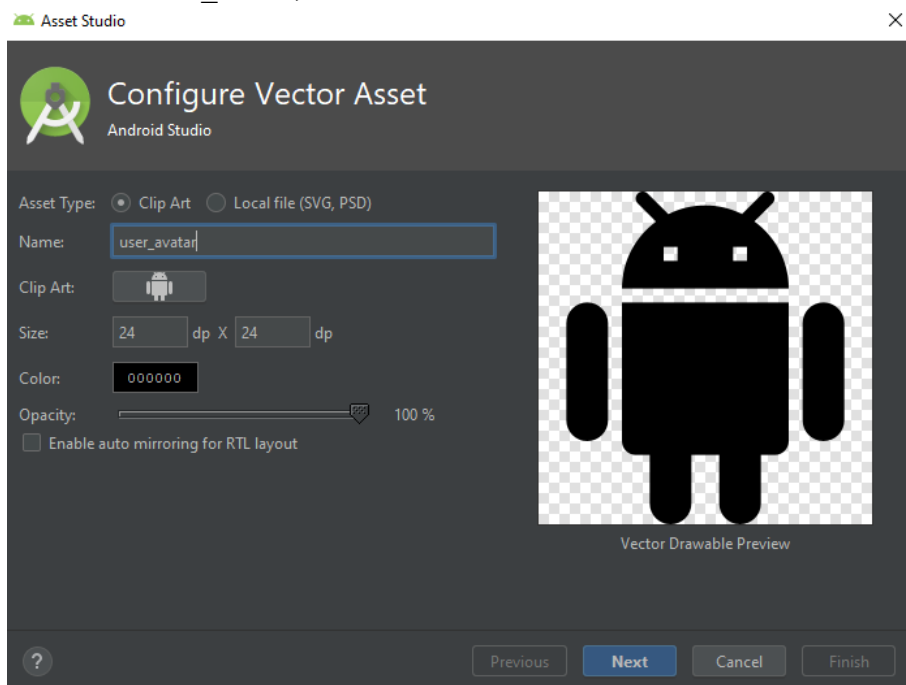
- **ViewModel** adalah kelas yang dirancang untuk menyimpan dan mengelola data terkait UI dengan pada lifecycle

6. Buat sebuah vektor dengan nama user_avatar.xml berisi sebagai berikut

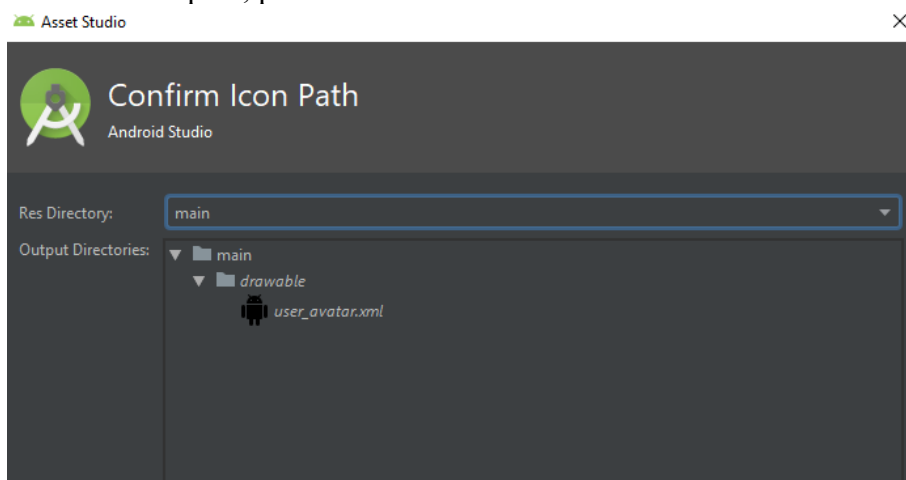
Klik drawable -> New -> Vector Asset

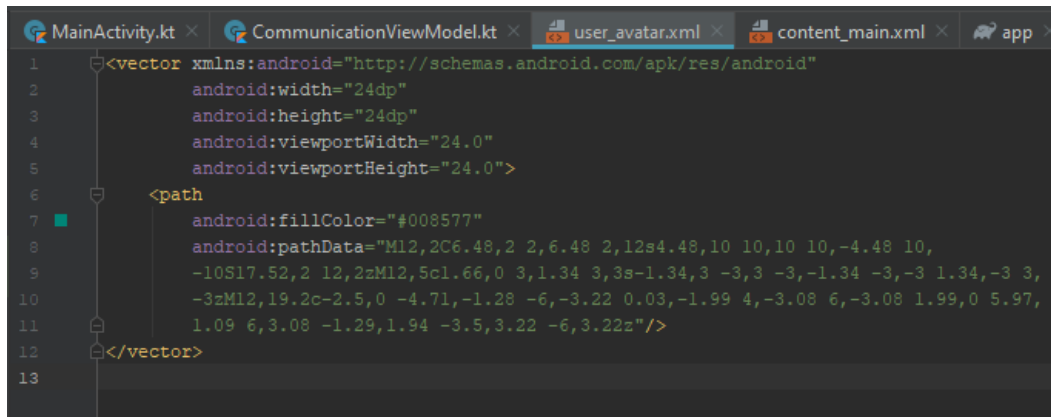


Beri nama user_avatar, lalu klik next



Confirm icon path, pilih main





Penjelasan:

- **android:width** dan **android:height** sebagai attribute yang berfungsi untuk menentukan width dan height dari ukuran VectorDrawable.
- **android:tint** sebagai attribute yang berfungsi untuk menentukan warna dari VectorDrawable.
- **android:viewportWidth** dan **android:viewportHeight** sebagai attribute yang berfungsi untuk menentukan width dan height dari ukuran canvas VectorDrawable. Canvas maksudnya adalah tempat kita melakukan proses penggambarannya.
- **<path>** berfungsi sebagai tag untuk melakukan penggambaran pada canvas.
- **android:fillColor** berfungsi sebagai attribute untuk memberikan isi warna dari path.
- **android:pathData** berfungsi sebagai attribute untuk melakukan gambar melalui command-command di canvas.

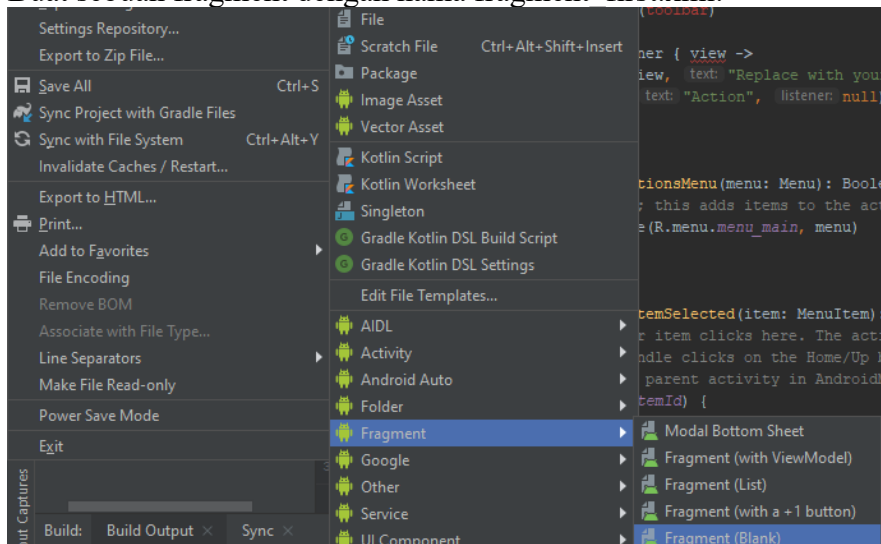
Keterangan pathData:

- M atau m (Move to) = memindahkan titik koordinat dari satu titik ke titik lainnya.
- L atau l (Line to) = menggambar garis lurus dari satu titik ke titik lainnya.
- H atau h (Horizontal to) = menggambar garis horizontal dari satu titik ke titik lainnya.
- V atau v (Vertical to) = menggambar garis vertical dari satu titik ke titik lainnya.
- Z atau z (Close path) = menutup path.

7. Buat string seperti berikut

```
<string name="tab_text_1">Input</string>
<string name="tab_text_2">Hasil</string>
```

8. Buat sebuah fragment dengan nama fragment first.xml.



```

CommunicationViewModel.kt x user_avatar.xml x fragment_first.xml x FirstFragment.kt
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8     <ImageView
9         android:id="@+id/imageView"
10        android:layout_width="72dp"
11        android:layout_height="72dp"
12        android:layout_marginEnd="8dp"
13        android:layout_marginLeft="8dp"
14        android:layout_marginRight="8dp"
15        android:layout_marginStart="8dp"
16        android:layout_marginTop="24dp"
17        android:src="@drawable/user_avatar"
18        app:layout_constraintEnd_toEndOf="parent"
19        app:layout_constraintStart_toStartOf="parent"
20        app:layout_constraintTop_toTopOf="parent"
21    />
22
23    <com.google.android.material.textfield.TextInputLayout
24        android:id="@+id/textInputLayout"
25        android:layout_width="0dp"
26        android:layout_height="wrap_content"
27        android:layout_marginEnd="16dp"
28        android:layout_marginLeft="16dp"
29        android:layout_marginRight="16dp"
30        android:layout_marginStart="16dp"
31        android:layout_marginTop="32dp"
32        app:layout_constraintEnd_toEndOf="parent"
33        app:layout_constraintStart_toStartOf="parent"
34        app:layout_constraintTop_toBottomOf="@+id/imageView"
35        style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox">
36
37        <com.google.android.material.textfield.TextInputEditText
38            android:id="@+id/textInputTextName"
39            android:layout_width="match_parent"
40            android:layout_height="wrap_content"
41            android:hint="Masukkan Nama"
42        />
43    </com.google.android.material.textfield.TextInputLayout>
44 </androidx.constraintlayout.widget.ConstraintLayout>

```

Penjelasan:

Fragment_first ini adalah fragment untuk menampilkan ImageView yang akan menampilkan vector yang tadi sudah kita buat. Kemudian di bawah vector tersebut, ada tempat atau field untuk memasukkan Nama dengan pesan "Masukkan Nama".

9. Buat juga kode program untuk FirstFragment.kt

```

FirstFragment.kt x fragment_first.xml x CommunicationViewModel.kt x user_avatar.xml x
1 package com.example.praktikumfragment
2
3
4 import android.os.Bundle
5 import android.text.Editable
6 import android.text.TextWatcher
7 import android.view.LayoutInflater
8 import android.view.View
9 import android.view.ViewGroup
10 import androidx.fragment.app.Fragment
11 import androidx.lifecycle.ViewModelProviders
12
13 import com.google.android.material.textfield.TextInputEditText
14 /**
15  * A simple [Fragment] subclass.

```

```

16  */
17  class FirstFragment : Fragment() {
18
19      private var communicationViewModel: CommunicationViewModel? = null
20
21      override fun onCreate(savedInstanceState: Bundle?) {
22          super.onCreate(savedInstanceState)
23          communicationViewModel =
24              ViewModelProviders.of(requireActivity()).
25                  get(CommunicationViewModel::class.java)
26      }
27
28      override fun onCreateView(
29          inflater: LayoutInflater, container: ViewGroup?,
30          savedInstanceState: Bundle?
31      ): View? {
32          return inflater.inflate(R.layout.fragment_first,
33              container, attachToRoot: false)
34      }
35
36      override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
37          super.onViewCreated(view, savedInstanceState)
38          val nameEditText = view.findViewById<TextInputEditText>(R.id.textInputTextName)
39          nameEditText.addTextChangedListener(
40              object : TextWatcher {
41                  override fun beforeTextChanged(
42                      charSequence: CharSequence, i: Int, il: Int, i2: Int
43                  ) {
44                  }
45
46                  override fun onTextChanged(
47                      charSequence: CharSequence,
48                      i: Int, il: Int, i2: Int
49                  ) {
50                      communicationViewModel!!.setName(charSequence.toString())
51                  }
52
53                  override fun afterTextChanged(editable: Editable) {
54                  }
55              })
56      }
57
58      companion object {
59          fun newInstance(): FirstFragment {
60              return FirstFragment()
61          }
62      }
63  }

```

Penjelasan:

File kotlin ini adalah tempat dimana kita memproses setiap widget fragmen_first.xml.

Berikut komponen yang ada di dalam class tersebut

- **Editable**: kelas untuk teks yang konten dan markup nya dapat diubah. Editable adalah antarmuka untuk teks yang konten dan markupnya dapat diubah
- **TextWatch**: Digunakan untuk melihat input text field dan kita juga dapat langsung memperbarui data pada view lain. **TextWatch** dapat berguna untuk menghitung jumlah karakter yang dimasukkan dalam bidang teks secara instan dan mengukur kekuatan password.
- **LayoutInflater**: kelas yang digunakan untuk membuat instance file layout XML ke dalam object view yang sesuai
- **View**: View adalah basic building block UI (User Interface) di android. View adalah kotak persegi kecil yang merespons input pengguna. Contoh nya EditText, Button, CheckBox
- **ViewGroup**: adalah special view yang dapat menampung view-view lain
- **ViewModelProviders**: adalah lifecycle yang berisi utility method untuk kelas ViewModelStore & mengembalikan ViewModelProvider saat kita menggunakan metode of()

10. Buat fragment kedua, beri nama fragment `second.xml`

```
FirstFragment.kt x SecondFragment.kt x fragment_second.xml x ViewF
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context=".SecondFragment">
9
10    <ImageView
11        android:id="@+id/imageView2"
12        android:layout_width="72dp"
13        android:layout_height="72dp"
14        android:layout_marginEnd="8dp"
15        android:layout_marginLeft="8dp"
16        android:layout_marginRight="8dp"
17        android:layout_marginStart="8dp"
18        android:layout_marginTop="24dp"
19        android:src="@drawable/user_avatar"
20        app:layout_constraintEnd_toEndOf="parent"
21        app:layout_constraintStart_toStartOf="parent"
22        app:layout_constraintTop_toTopOf="parent"
23    />
24
25    <TextView
26        android:id="@+id/textViewName"
27        android:layout_width="0dp"
28        android:layout_height="wrap_content"
29        android:layout_marginEnd="8dp"
30        android:layout_marginStart="8dp"
31        android:layout_marginTop="8dp"
32        android:gravity="center"
33        android:hint="User Display Name"
34        android:textColor="@color/colorPrimaryDark"
35        android:textSize="22sp"
36        android:textStyle="bold"
37        app:layout_constraintEnd_toEndOf="parent"
38        app:layout_constraintHorizontal_bias="0.0"
39        app:layout_constraintStart_toStartOf="parent"
40        app:layout_constraintTop_toBottomOf="@+id/textView"
41        tools:text="Haloo Dunia!" />
42
43    <TextView
44        android:id="@+id/textView"
45        android:layout_width="0dp"
46        android:layout_height="wrap_content"
47        android:layout_marginTop="24dp"
48        android:text="Selamat Datang"
49        android:textAlignment="center"
50        android:textSize="22sp"
51        app:layout_constraintEnd_toEndOf="parent"
52        app:layout_constraintStart_toStartOf="parent"
53        app:layout_constraintTop_toBottomOf="@+id/imageView2"/>
54
55 </androidx.constraintlayout.widget.ConstraintLayout>
```

Penjelasan:

Fragment `second.xml` ini adalah fragment untuk menampilkan `ImageView` yang akan menampilkan vector yang tadi sudah kita buat. Kemudian di bawah vector tersebut, ada tempat untuk menampilkan nama yang sudah kita input di `fragment_first` tadi.

11. Programnya SecondFragment.kt

```
FirstFragment.kt x SecondFragment.kt x fragment_second.xml x ViewPagerAdap
1 package com.example.praktikumfragment
2
3
4 import android.os.Bundle
5 import android.view.LayoutInflater
6 import android.view.View
7 import android.view.ViewGroup
8 import android.widget.TextView
9 import androidx.lifecycle.Observer
10 import androidx.fragment.app.Fragment
11 import androidx.lifecycle.ViewModelProviders
12
13 /**
14  * A simple [Fragment] subclass.
15  */
16 class SecondFragment : Fragment() {
17
18     private var communicationViewModel: CommunicationViewModel? = null
19     private var txtName: TextView? = null
20
21     override fun onCreate(savedInstanceState: Bundle?) {
22         super.onCreate(savedInstanceState)
23         communicationViewModel = ViewModelProviders.
24             of(requireActivity()).
25             get(CommunicationViewModel::class.java)
26     }
27
28     override fun onCreateView(
29         inflater: LayoutInflater, container: ViewGroup?,
30         savedInstanceState: Bundle?
31     ): View? {
32         return inflater.inflate(R.layout.fragment_second,
33             container, attachToRoot: false)
34     }
35
36     override fun onViewCreated(view: View, savedInstanceState: Bundle?)
37     {
38         super.onViewCreated(view, savedInstanceState)
39         txtName = view.findViewById(R.id.textViewName)
40         communicationViewModel!!.name.observe(requireActivity(),
41             Observer { s -> txtName!!.text = s })
42     }
43     companion object {
44         fun newInstance(): SecondFragment {
45             return SecondFragment()
46         }
47     }
48 }
```

Penjelasan:

Fragmen ini adalah tempat dimana kita memproses setiap widget fragmen_second.xml. komponennya hampir sama dengan di FirstFragment tadi, bedanya adalah disini kita menambahkan atau import Observer. Observer adalah method yang dipanggil setiap kali objek yang diobservasi diubah. Aplikasi memanggil metode notifyObservers objek Observable agar semua observer objek diberi tahu tentang perubahan tersebut.

12. Buat sebuah adapter dengan nama View PagerAdapter.kt

```
FirstFragment.kt x ViewPagerAdapter.kt x SecondFragment.kt x fragment_second.xml x
1 package com.example.praktikumfragment
2
3 import android.content.Context;
4 import androidx.annotation.StringRes;
```

```

5      import androidx.fragment.app.Fragment;
6      import androidx.fragment.app.FragmentManager;
7      import androidx.fragment.app.FragmentManager;
8
9      class ViewPagerAdapter(private val mContext: Context, fm: FragmentManager) :
10         FragmentPagerAdapter(fm) {
11
12         override fun getItem(position: Int): Fragment {
13             return if (position == 0) {
14                 FirstFragment.newInstance()
15             } else {
16                 SecondFragment.newInstance()
17             }
18         }
19         override fun getPageTitle(position: Int): CharSequence? {
20             return mContext.resources.getString(TAB_TITLES[position])
21         }
22         override fun getCount(): Int {
23             return 2
24         }
25         companion object {
26             @StringRes
27             private val TAB_TITLES = intArrayOf(R.string.tab_text_1,
28                 R.string.tab_text_2)
29         }
30     }

```

Penjelasan:

ViewPager di Android adalah kelas yang memungkinkan pengguna untuk membalik atau menggeser halaman data ke kiri dan kanan. Kelas ini menyediakan fungsionalitas untuk membalik halaman di app. ViewPager adalah widget yang ditemukan di support library. Untuk menggunakannya, kita harus meletakkan elemen di dalam file layout XML. Sementara objek Adapter bertindak sebagai jembatan antara AdapterView dan data yang mendasari tampilan tersebut. Adapter menyediakan akses ke item data. Adapter juga bertanggung jawab untuk membuat tampilan untuk setiap item dalam dataset.

13. Kemudian, buka MainActivity.kt dan tuliskan kode program, sehingga menjadi seperti berikut

```

1      package com.example.praktikumfragment
2
3      import android.os.Bundle
4      import androidx.appcompat.app.AppCompatActivity
5      import kotlinx.android.synthetic.main.activity_main.*
6
7      class MainActivity : AppCompatActivity() {
8
9      override fun onCreate(savedInstanceState: Bundle?) {
10         super.onCreate(savedInstanceState)
11         setContentView(R.layout.activity_main)
12
13         view_pager.adapter = ViewPagerAdapter(
14             mContext: this, supportFragmentManager
15         )
16         tabs.setupWithViewPager(view_pager)
17     }
18 }

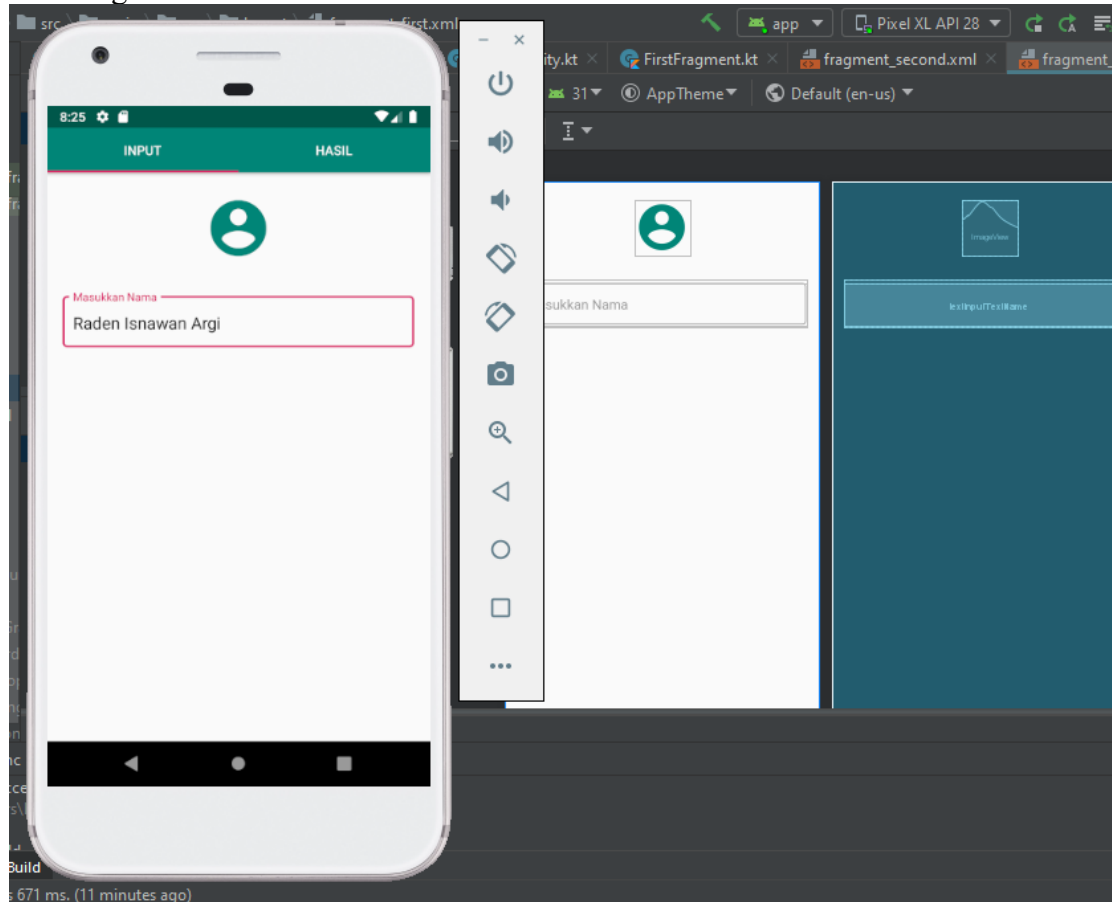
```

Penjelasan:

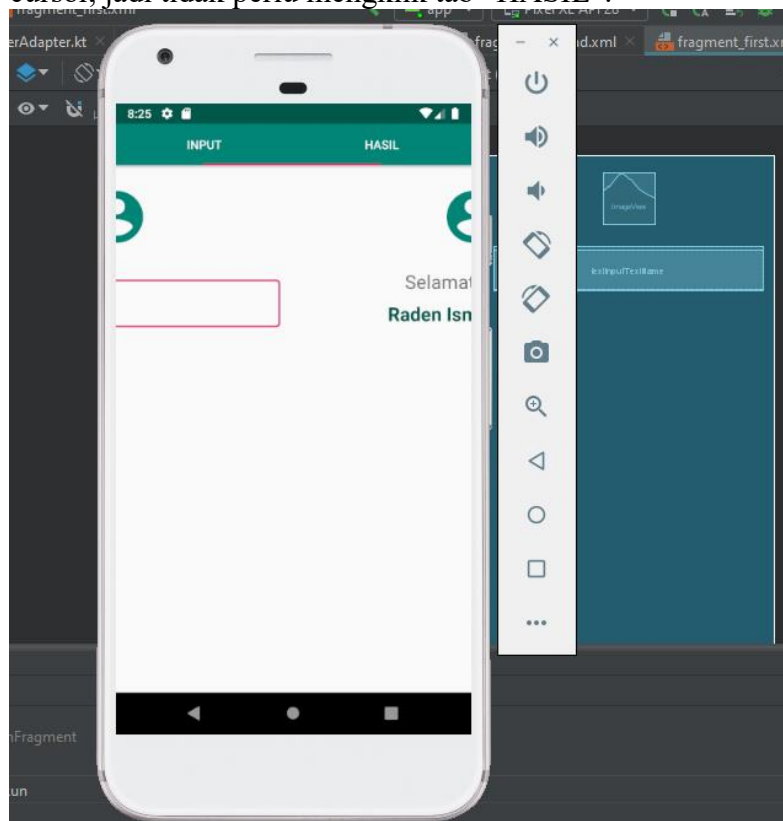
Di dalam MainActivity.kt, kita menginisialisasikan view_pager dengan method adapter yang dihubungkan dengan class ViewPagerAdapter supaya segala kode nya bisa diproses. Kemudian kita tulis kode 'this' lalu dilanjutkan dengan supportFragmentManager. FragmentManager adalah kelas yang bertanggung jawab untuk melakukan tindakan pada fragment aplikasi kita seperti menambahkan, menghapus, mengganti, dan menambahkannya ke back-stack. Terakhir kita tulis **tabs.setupWithViewPager(view_pager)**.

14. Jalankan dan amati hasilnya

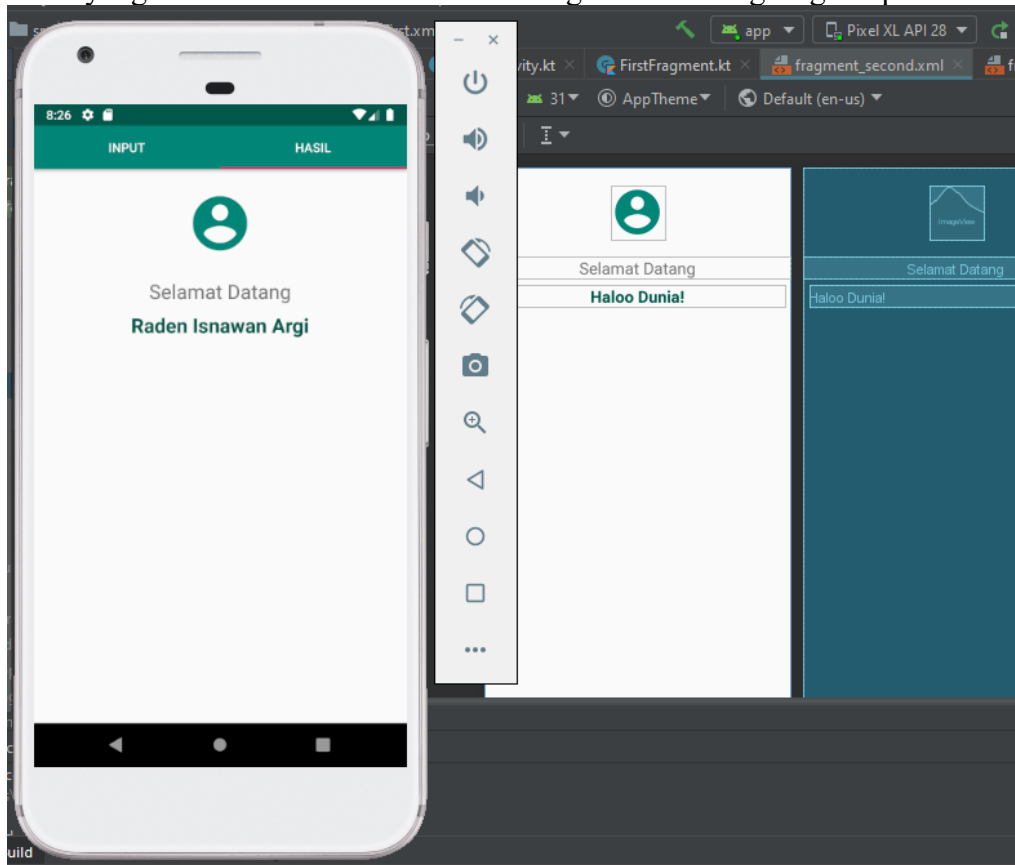
FirstFragment



Kita bisa berpindah dari FirstFragment ke SecondFragment dengan cara swipe menggunakan cursor, jadi tidak perlu mengklik tab “HASIL”.



Kita juga bisa mengklik tab “HASIL” untuk berpindah ke SecondFragment, bisa dilihat jika nama yang sudah kita masukkan di FirstFragment tadi langsung tampil di SecondFragment

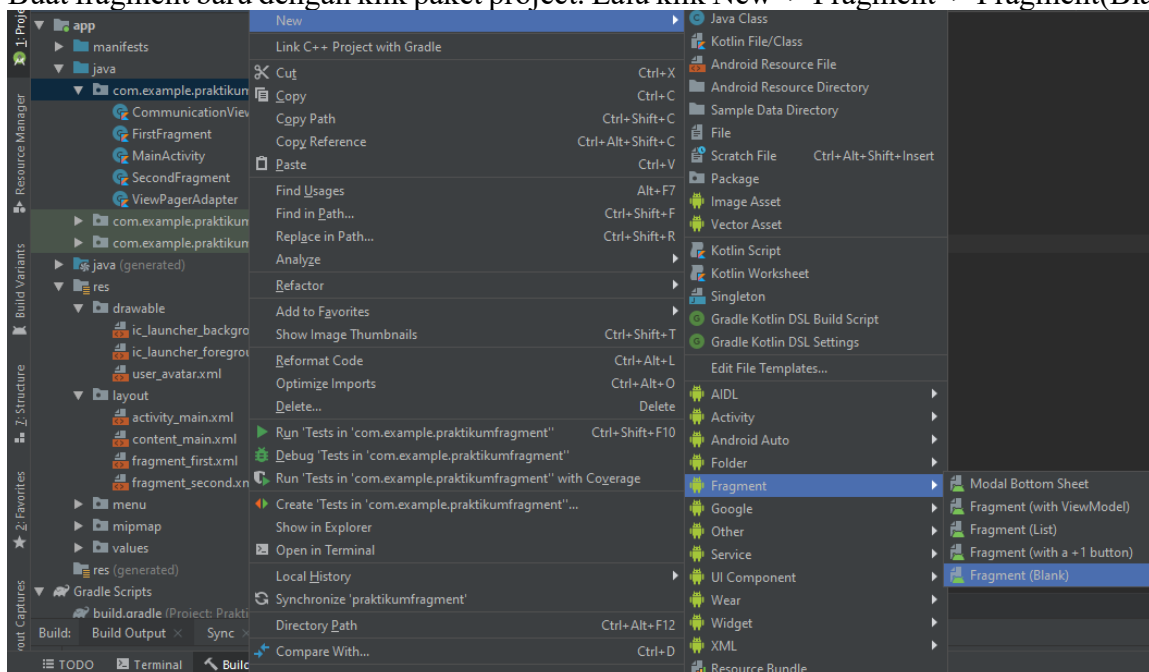


LATIHAN

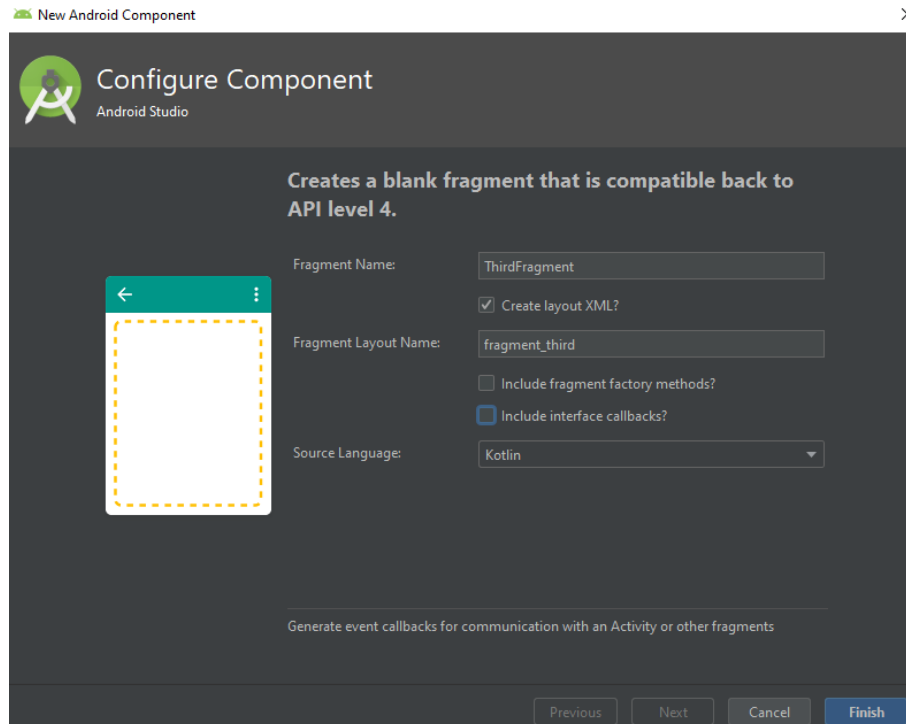
Modifikasilah aplikasi dengan menambahkan satu fragment lagi.

Langkah-langkah:

1. Buat fragment baru dengan klik paket project. Lalu klik New -> Fragment -> Fragment(Blank)



2. Beri nama fragment dengan “ThirdFragment” & nama xml nya dengan “fragment_third.xml”



3. Kita buka fragment third.xml, lalu tambahkan kode berikut

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context=".ThirdFragment">
9
10    <ImageView
11        android:id="@+id/imageView3"
12        android:layout_width="72dp"
13        android:layout_height="72dp"
14        android:layout_marginEnd="8dp"
15        android:layout_marginLeft="8dp"
16        android:layout_marginRight="8dp"
17        android:layout_marginStart="8dp"
18        android:layout_marginTop="24dp"
19        android:src="@drawable/user_avatar"
20        app:layout_constraintEnd_toEndOf="parent"
21        app:layout_constraintStart_toStartOf="parent"
22        app:layout_constraintTop_toTopOf="parent"
23    />
24
25    <TextView
26        android:id="@+id/textViewAddress"
27        android:layout_width="0dp"
28        android:layout_height="wrap_content"
29        android:layout_marginEnd="8dp"
30        android:layout_marginStart="8dp"
31        android:layout_marginTop="8dp"
32        android:gravity="center"
33        android:hint="User Display Address"
34        android:textColor="@color/colorPrimaryDark"
35        android:textSize="22sp"
36        android:textStyle="bold"
37        app:layout_constraintEnd_toEndOf="parent"
38        app:layout_constraintHorizontal_bias="0.0"
39        app:layout_constraintStart_toStartOf="parent"
```

```

40         app:layout_constraintTop_toBottomOf="@+id/textViewAd"
41         tools:text="Hallo Dunia!" />
42
43     <TextView
44         android:id="@+id/textViewAd"
45         android:layout_width="0dp"
46         android:layout_height="wrap_content"
47         android:layout_marginTop="24dp"
48         android:text="Alamat user"
49         android:textAlignment="center"
50         android:textSize="22sp"
51         app:layout_constraintEnd_toEndOf="parent"
52         app:layout_constraintStart_toStartOf="parent"
53         app:layout_constraintTop_toBottomOf="@+id/imageView3"/>

```

Penjelasan:

fragment_third ini adalah fragment untuk menampilkan ImageView yang akan menampilkan vector yang tadi sudah kita buat. Kemudian di bawah vector tersebut, ada tempat atau field untuk menampilkan “Alamat user” dan data alamat yang dimasukkan user (User Display Address).

4. Kita buka file ThirdFragment lalu tambahkan kode berikut

```

fragment_third.xml x ThirdFragment.kt x fragment_second.xml x FirstFragment.kt
1 package com.example.praktikumfragment
2
3
4 import android.os.Bundle
5 import android.view.LayoutInflater
6 import android.view.View
7 import android.view.ViewGroup
8 import android.widget.TextView
9 import androidx.lifecycle.Observer
10 import androidx.fragment.app.Fragment
11 import androidx.lifecycle.ViewModelProviders
12
13 /**
14  * A simple [Fragment] subclass.
15  */
16 class ThirdFragment : Fragment() {
17
18     private var communicationViewModel: CommunicationViewModel? = null
19     private var txtAddress: TextView? = null
20
21     override fun onCreate(savedInstanceState: Bundle?) {
22         super.onCreate(savedInstanceState)
23         communicationViewModel = ViewModelProviders.
24             of(requireActivity()).
25             get(CommunicationViewModel::class.java)
26     }
27
28     override fun onCreateView(
29         inflater: LayoutInflater, container: ViewGroup?,
30         savedInstanceState: Bundle?
31     ): View? {
32         return inflater.inflate(R.layout.fragment_third,
33             container, attachToRoot: false)
34     }
35
36     override fun onViewCreated(view: View, savedInstanceState: Bundle?)
37     {
38         super.onViewCreated(view, savedInstanceState)
39         txtAddress = view.findViewById(R.id.textViewAddress)
40         communicationViewModel!!.address.observe(requireActivity(),
41             Observer { s -> txtAddress!!.text = s })
42     }
43     companion object {
44         fun newInstance(): ThirdFragment {
45             return ThirdFragment()
46         }
47     }
48 }

```


Penjelasan:

Berikut komponen yang ada di dalam class tersebut

- **LayoutInflater**: kelas yang digunakan untuk membuat instance file layout XML ke dalam object view yang sesuai
- **View**: View adalah basic building block UI (User Interface) di android. View adalah kotak persegi kecil yang merespons input pengguna. Contoh nya EditText, Button, CheckBox
- **ViewGroup**: adalah special view yang dapat menampung view-view lain
- **ViewModelProviders**: adalah lifecycle yang berisi utility method untuk kelas ViewModelStore & mengembalikan ViewModelProvider saat kita menggunakan metode of()
- **Observer**: Observer adalah method yang dipanggil setiap kali objek yang diobservasi diubah. Aplikasi memanggil metode notifyObservers objek Observable agar semua observer objek diberi tahu tentang perubahan tersebut

5. Kita tambahkan string ke-3 untuk tab ke-3

```
<string name="tab_text_1">Input</string>
<string name="tab_text_2">Nama</string>
<string name="tab_text_3">Alamat</string>
```

6. Kita ke fragment_first.xml dan tambahkan kode berikut

```
45 <com.google.android.material.textfield.TextInputLayout
46     android:id="@+id/textInputLayout2"
47     android:layout_width="0dp"
48     android:layout_height="wrap_content"
49     android:layout_marginEnd="16dp"
50     android:layout_marginLeft="16dp"
51     android:layout_marginRight="16dp"
52     android:layout_marginStart="16dp"
53     android:layout_marginTop="32dp"
54     app:layout_constraintEnd_toEndOf="parent"
55     app:layout_constraintStart_toStartOf="parent"
56     app:layout_constraintTop_toBottomOf="@+id/textInputLayout"
57     style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox">
58
59 <com.google.android.material.textfield.TextInputEditText
60     android:id="@+id/textInputTextAddress"
61     android:layout_width="match_parent"
62     android:layout_height="wrap_content"
63     android:hint="Masukkan Alamat"
64     />
65 </com.google.android.material.textfield.TextInputLayout>
```

Penjelasan:

Baris-baris kode ini berfungsi sebagai penyedia field untuk memasukkan alamat bagi user. Kita menggunakan material dari google bernama TextInputEditText untuk membuat field editable tersebut. Untuk id nya kita berikan nama **textInputTextAddress** supaya nanti bisa diproses di FirstFragment.kt

7. Pindah ke FirstFragment.kt, kita buat variabel nameEditText2 untuk menampung value dan memproses widget **textInputTextAddress** dari fragment_first.xml

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    val nameEditText = view.findViewById<TextInputEditText>(R.id.textInputTextName)
    val nameEditText2 = view.findViewById<TextInputEditText>(R.id.textInputTextAddress)
    nameEditText.addTextChangedListener{
```

8. Masih di FirstFragment.kt, tambahkan kode berikut
(di halaman selanjutnya)


```

nameEditText2.addTextChangedListener(
    object : TextWatcher {
        override fun beforeTextChanged(
            charSequence: CharSequence, i: Int, il: Int, i2: Int
        ) {
        }

        override fun onTextChanged(
            charSequence: CharSequence,
            i: Int, il: Int, i2: Int
        ) {
            communicationViewModel!!.setAddress(charSequence.toString())
        }

        override fun afterTextChanged(editable: Editable) {
        }
    })

```

Penjelasan:

berforeTextChanged berarti keadaan field sebelum diinput data oleh user. onTextChanged adalah keadaan field saat diinput data oleh user, maka dari itu kita inisialisasikan dengan setAddress di class communicationViewModel. Terakhir adalah afterTextChanged yaitu ketika text sudah diinput, menggunakan editable supaya bisa diedit kembali oleh user.

9. Pindah ke class CommunicationViewModel.kt dan tambahkan kode berikut

```

9      private val mName = MutableLiveData<String>()
10
11      val name: LiveData<String>
12      get() = mName
13
14      val address: LiveData<String>
15      get() = mAddress
16
17      fun setName(name: String) {
18          mName.value = name
19      }
20      fun setAddress(address: String) {
21          mAddress.value = address
22      }

```

Penjelasan:

Di class inilah kita menggunakan metode setter dan getter untuk mendapatkan value dari data yang dimasukkan oleh user ke dalam field “Alamat”. Pertama kita buat sebuah variabel dengan kode **private val mName = MutableLiveData<String>()**. Lalu kita gunakan method getter untuk mendapatkan data yaitu dengan **get() = mName**. Untuk memvalidasi value yang sudah didapatkan, kita gunakan setter dengan fungsi **fun setAddress(address: String)** dan kita inisialisasi value dengan **mAddress.value = address**

10. Kita tambahkan kode berikut di ViewPagerAdapter.kt

```

override fun getItem(position: Int): Fragment {
    return if (position == 0) {
        FirstFragment.newInstance()
    } else if (position == 1) {
        SecondFragment.newInstance()
    } else {
        ThirdFragment.newInstance()
    }
}

```

Penjelasan:

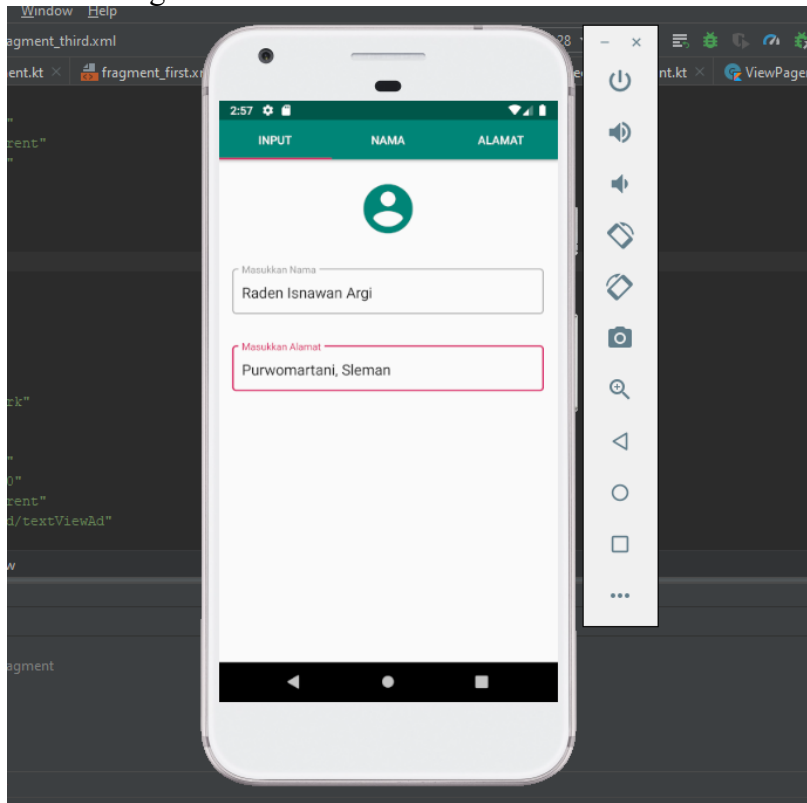
Bisa kita lihat bahwa kita menambahkan seleksi kondisi untuk ThirdFragment. Seleksi pertama yaitu milik FirstFragment adalah “position == 0” yang berarti ada di slide pertama. Seleksi kedua yaitu milik SecondFragment adalah “position == 1” yang berarti ada di slide kedua. Seleksi ketiga yaitu milik ThirdFragment adalah “else” atau “position == 2” yang berarti ada di slide ketiga.

11. Masih di ViewPagerAdapter.kt, tambahkan kode berikut untuk memberi nama tab title dengan string yang sudah kita buat tadi yaitu "Alamat" atau R.string.tab_text_3

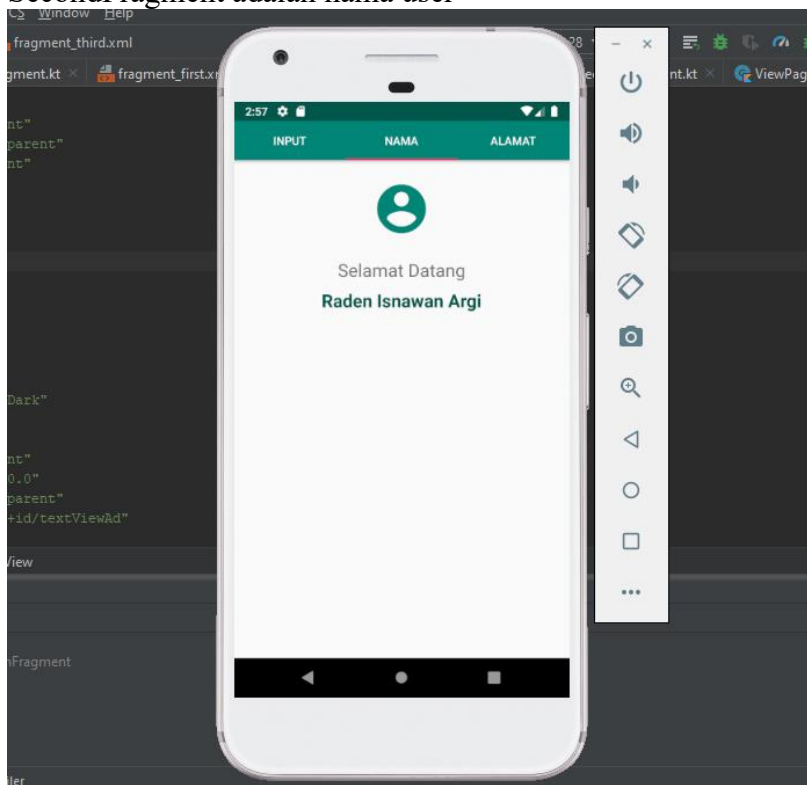
```
private val TAB_TITLES = intArrayOf(R.string.tab_text_1,  
    R.string.tab_text_2, R.string.tab_text_3)
```

12. Fragment ke-3 sudah berhasil ditambahkan. Run aplikasi untuk melihat hasilnya.

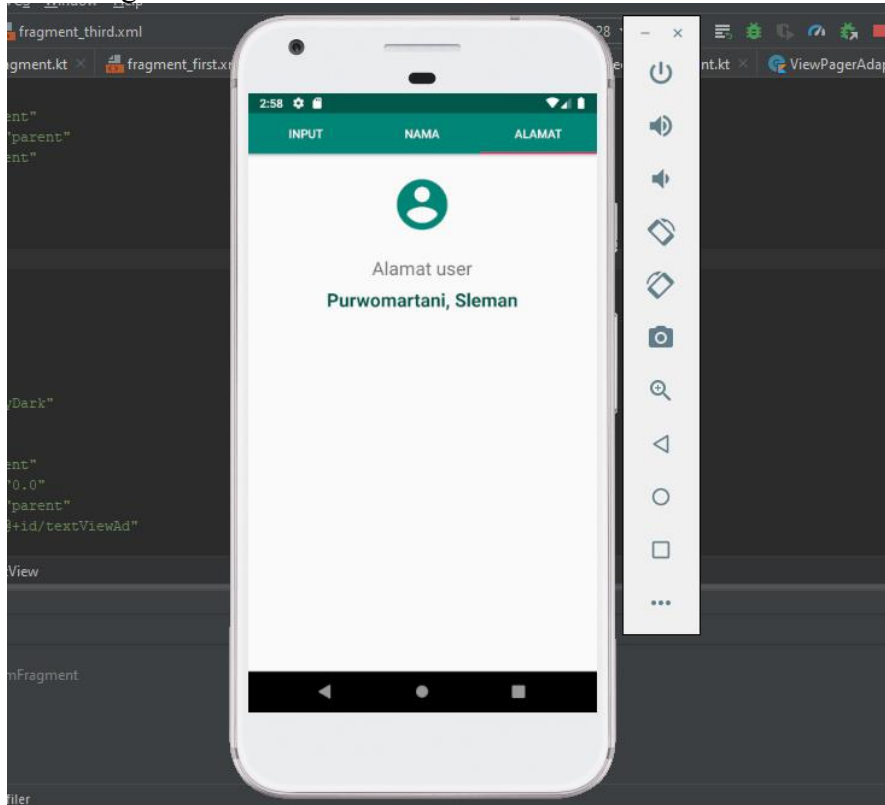
Di FirstFragment ada dua editable field untuk nama dan alamat



SecondFragment adalah nama user



ThirdFragment adalah alamat user



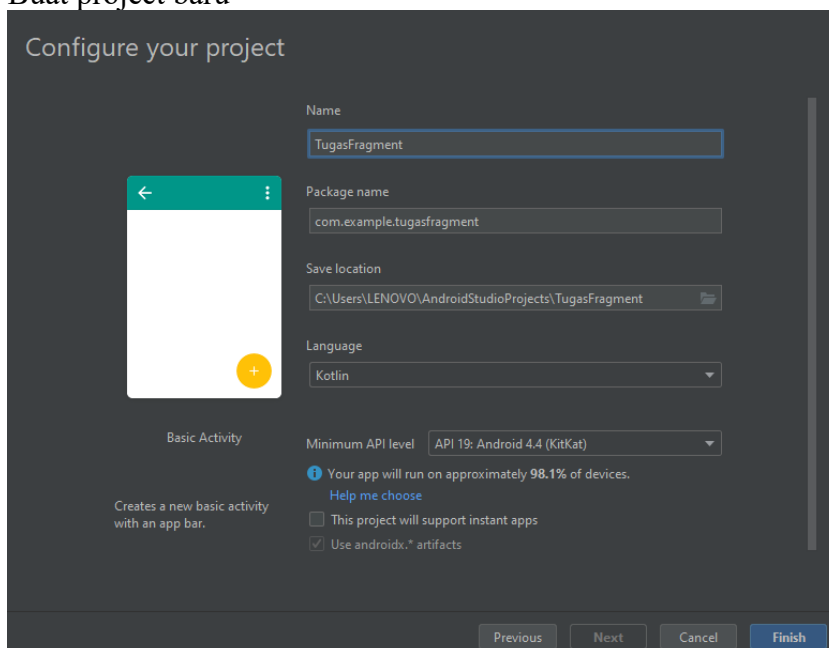
TUGAS

Buat aplikasi baru dengan mengembangkan project diatas

Di tugas ini, saya akan menggunakan 2 fragment saja untuk menampilkan beberapa data user (Nama, Kampus, Prodi) sehingga lebih efisien dan tidak memerlukan lebih dari dua tab atau fragment.

Langkah-langkah:

1. Buat project baru



2. Ubah kode pada activity_main.xml menjadi seperti ini

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.coordinatorlayout.widget.CoordinatorLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context=".MainActivity">
9
10    <com.google.android.material.appbar.AppBarLayout
11        android:layout_width="match_parent"
12        android:layout_height="wrap_content"
13        android:theme="@style/AppTheme">
14
15        <com.google.android.material.tabs.TabLayout
16            android:id="@+id/tabs"
17            android:layout_width="match_parent"
18            android:layout_height="wrap_content"
19            android:background="?attr/colorPrimary"
20            app:tabTextColor="@android:color/background_light"/>
21    </com.google.android.material.appbar.AppBarLayout>
22
23    <androidx.viewpager.widget.ViewPager
24        android:id="@+id/view_pager"
25        android:layout_width="match_parent"
26        android:layout_height="match_parent"
27        app:layout_behavior="com.google.android.material.appbar.AppBarLayout$Scrolli..."
28    />
29 </androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Penjelasan:

CoordinatorLayout adalah super-powered `FrameLayout` (`FrameLayout` dengan fitur dan jangkauan yang lebih banyak dan luas). `CoordinatorLayout` ditujukan untuk dua kasus penggunaan utama: Sebagai dekorasi top-level application atau layout chrome serta juga sebagai container untuk interaksi tertentu dengan satu atau beberapa child views.

AppBarLayout adalah `LinearLayout` vertikal yang mengimplementasikan banyak fitur dari konsep desain material bar aplikasi

TabLayout digunakan untuk mengimplementasikan tab horizontal. `TabLayout` dirilis oleh Android setelah diberhentikannya fitur `ActionBar`. `TabLayout` diperkenalkan di design support library yang berguna untuk mengimplementasikan tab. Tab dibuat menggunakan method `newTab()` dari kelas `TabLayout`.

ViewPager adalah widget yang memungkinkan pengguna untuk menggeser ke kiri atau kanan untuk melihat layar baru. Dengan kata lain, ini `ViewPager` adalah sebuah cara atau method untuk menampilkan banyak tab kepada pengguna. `ViewPager` juga memiliki kemampuan untuk secara dinamis menambah dan menghapus halaman (atau tab) kapan saja.

3. Buat kotlin class CommunicationViewModel untuk viewmodel

```
MainActivity.kt x CommunicationViewModel.kt x activity_main.xml
1 package com.example.tugasfragment
2
3 import androidx.lifecycle.LiveData
4 import androidx.lifecycle.MutableLiveData
5 import androidx.lifecycle.ViewModel
6
7 class CommunicationViewModel : ViewModel() {
8     private val mName = MutableLiveData<String>()
9     private val mKampus = MutableLiveData<String>()
10    private val mProdi = MutableLiveData<String>()
11
12    val name: LiveData<String>
13    get() = mName
14}
```

```

15         val kampus: LiveData<String>
16             get() = mKampus
17
18         val prodi: LiveData<String>
19             get() = mProdi
20
21         fun setName(name: String) {
22             mName.value = name
23         }
24         fun setKampus(kampus: String) {
25             mKampus.value = kampus
26         }
27         fun setProdi(prodi: String) {
28             mProdi.value = prodi
29         }
30     }

```

Penjelasan:

- **LiveData** adalah kelas data holder yang dapat diamati dalam suatu lifecycle. Ini berarti bahwa Observer dapat ditambahkan berpasangan dengan LifecycleOwner, dan Observer akan diberi tahu tentang modifikasi data yang di-wrap hanya jika LifecycleOwner dalam keadaan aktif.
- **MutableLiveData** adalah sebuah kelas yang memperluas class type LiveData. MutableLiveData biasanya digunakan untuk menyediakan metode postValue() , setValue() secara publik, yaitu sesuatu yang tidak disediakan oleh kelas LiveData.
- **ViewModel** adalah kelas yang dirancang untuk menyimpan dan mengelola data terkait UI dengan pada lifecycle

Di dalam class tersebut juga ada metode getter dan setter untuk name, kampus, dan prodi. Getter digunakan untuk menangkap data yang dimasukkan user, sementara setter berguna untuk memvalidasi data yang telah ditangkap oleh getter.

4. Buat vektor dengan kode seperti berikut

```

1 <vector xmlns:android="http://schemas.android.com/apk/res/android"
2     android:width="24dp"
3     android:height="24dp"
4     android:viewportWidth="24.0"
5     android:viewportHeight="24.0">
6     <path
7         android:fillColor="#008577"
8         android:pathData="M12,2C6.48,2 2,6.48 2,12s4.48,10 10,10 10,-4.48 10,-10S17.52,2 12,2zM12,5c1.66,0 3,1.34 3,3s-1.34,3 -3,3 -3,-1.34 -3,-3 1.34,-3 3,-3zM12,19.2c-2.5,0 -4.71,-1.28 -6,-3.22 0.03,-1.99 4,-3.08 6,-3.08 1.99,0 5.97,1.09 6,3.08 -1.29,1.94 -3.5,3.22 -6,3.22z"/>
9     </vector>

```

Penjelasan:

- **android:width** dan **android:height** sebagai attribute yang berfungsi untuk menentukan width dan height dari ukuran VectorDrawable.
- **android:tint** sebagai attribute yang berfungsi untuk menentukan warna dari VectorDrawable.
- **android:viewportWidth** dan **android:viewportHeight** sebagai attribute yang berfungsi untuk menentukan width dan height dari ukuran canvas VectorDrawable. Canvas maksudnya adalah tempat kita melakukan proses penggambarannya.
- **<path>** berfungsi sebagai tag untuk melakukan penggambaran pada canvas.
- **android:fillColor** berfungsi sebagai attribute untuk memberikan isi warna dari path.
- **android:pathData** berfungsi sebagai attribute untuk melakukan gambar melalui command di canvas.

Keterangan pathData:

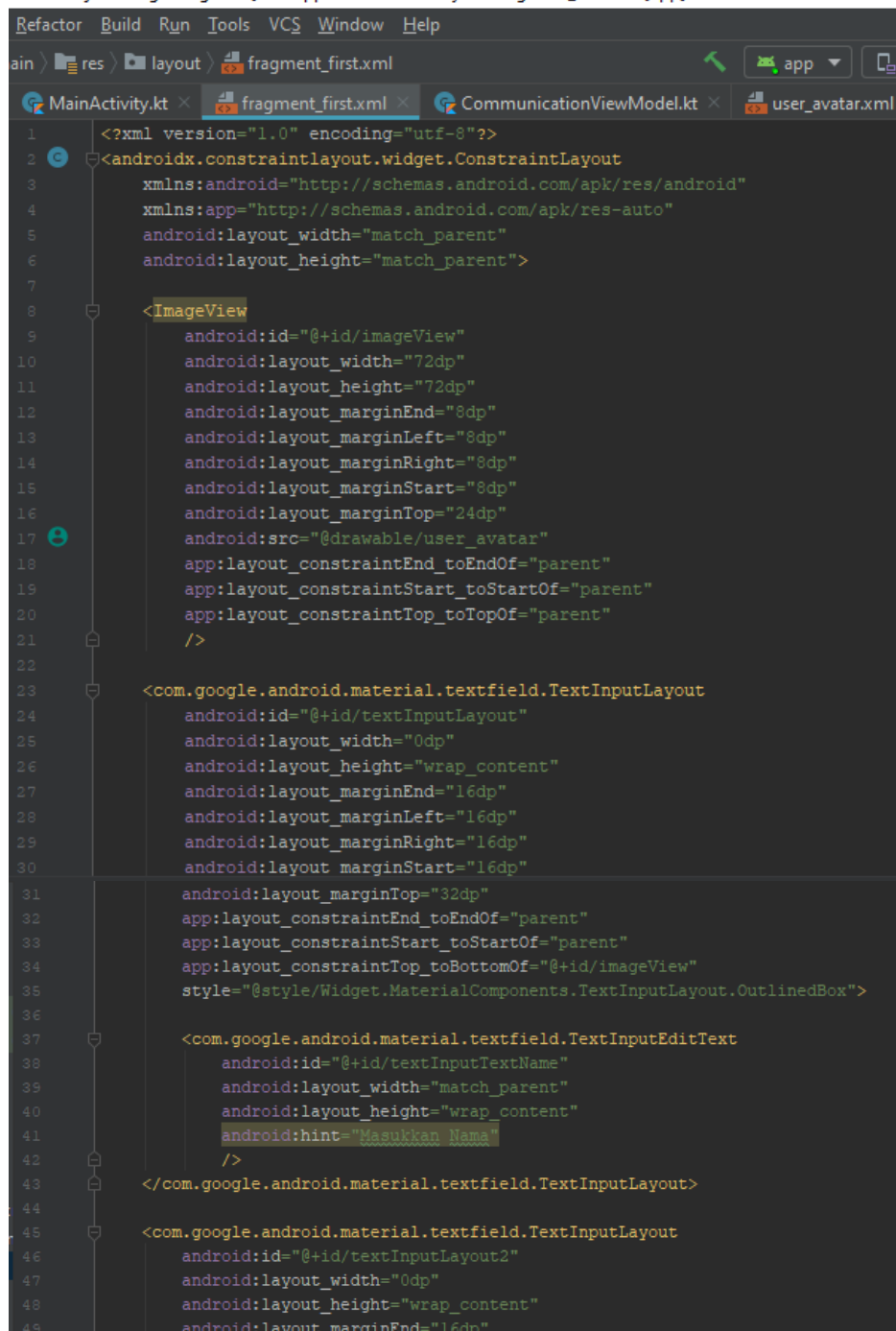
- M atau m (Move to) = memindahkan titik koordinat dari satu titik ke titik lainnya.
- L atau l (Line to) = menggambarkan garis lurus dari satu titik ke titik lainnya.
- H atau h (Horizontal to) = menggambarkan garis horizontal dari satu titik ke titik lainnya.
- V atau v (Vertical to) = menggambarkan garis vertical dari satu titik ke titik lainnya.
- Z atau z (Close path) = menutup path.

5. Buat string Input dan Hasil yang digunakan sebagai nama tab

```
<string name="tab_text_1">Input</string>
<string name="tab_text_2">Hasil</string>
```

6. Buat sebuah fragment, lalu buka xml nya dan tulis kode seperti berikut

StudioProjects\TugasFragment - ...\\app\\src\\main\\res\\layout\\fragment_first.xml [app] - Android Studio



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8     <ImageView
9         android:id="@+id/imageView"
10        android:layout_width="72dp"
11        android:layout_height="72dp"
12        android:layout_marginEnd="8dp"
13        android:layout_marginLeft="8dp"
14        android:layout_marginRight="8dp"
15        android:layout_marginStart="8dp"
16        android:layout_marginTop="24dp"
17        android:src="@drawable/user_avatar"
18        app:layout_constraintEnd_toEndOf="parent"
19        app:layout_constraintStart_toStartOf="parent"
20        app:layout_constraintTop_toTopOf="parent"
21    />
22
23    <com.google.android.material.textfield.TextInputLayout
24        android:id="@+id/textInputLayout"
25        android:layout_width="0dp"
26        android:layout_height="wrap_content"
27        android:layout_marginEnd="16dp"
28        android:layout_marginLeft="16dp"
29        android:layout_marginRight="16dp"
30        android:layout_marginStart="16dp"
31        android:layout_marginTop="32dp"
32        app:layout_constraintEnd_toEndOf="parent"
33        app:layout_constraintStart_toStartOf="parent"
34        app:layout_constraintTop_toBottomOf="@+id/imageView"
35        style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox">
36
37        <com.google.android.material.textfield.TextInputEditText
38            android:id="@+id/textInputTextName"
39            android:layout_width="match_parent"
40            android:layout_height="wrap_content"
41            android:hint="Masukkan Nama"
42        />
43    </com.google.android.material.textfield.TextInputLayout>
44
45    <com.google.android.material.textfield.TextInputLayout
46        android:id="@+id/textInputLayout2"
47        android:layout_width="0dp"
48        android:layout_height="wrap_content"
49        android:layout_marginEnd="16dp"
```

```

50         android:layout_marginLeft="16dp"
51         android:layout_marginRight="16dp"
52         android:layout_marginStart="16dp"
53         android:layout_marginTop="32dp"
54         app:layout_constraintEnd_toEndOf="parent"
55         app:layout_constraintStart_toStartOf="parent"
56         app:layout_constraintTop_toBottomOf="@+id/textInputLayout"
57         style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox">
58
59         <com.google.android.material.textfield.TextInputEditText
60             android:id="@+id/textInputTextKampus"
61             android:layout_width="match_parent"
62             android:layout_height="wrap_content"
63             android:hint="Masukkan Nama Kampus"
64         />
65     </com.google.android.material.textfield.TextInputLayout>
66
67     <com.google.android.material.textfield.TextInputLayout
68         android:id="@+id/textInputLayout3"
69         android:layout_width="0dp"
70         android:layout_height="wrap_content"
71         android:layout_marginEnd="16dp"
72         android:layout_marginLeft="16dp"
73         android:layout_marginRight="16dp"
74         android:layout_marginStart="16dp"
75         android:layout_marginTop="32dp"
76         app:layout_constraintEnd_toEndOf="parent"
77         app:layout_constraintStart_toStartOf="parent"
78         app:layout_constraintTop_toBottomOf="@+id/textInputLayout2"
79         style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox">
80
81         <com.google.android.material.textfield.TextInputEditText
82             android:id="@+id/textInputTextProdi"
83             android:layout_width="match_parent"
84             android:layout_height="wrap_content"
85             android:hint="Masukkan Prodi"
86         />
87     </com.google.android.material.textfield.TextInputLayout>
88 </androidx.constraintlayout.widget.ConstraintLayout>

```

Penjelasan:

ini adalah fragment untuk menampilkan ImageView yang akan menampilkan vector yang tadi sudah kita buat. Kemudian di bawah vector tersebut, ada tempat atau field untuk memasukkan Nama dengan pesan “Masukkan Nama”, field memasukkan nama kampus dengan pesan “Masukkan Nama Kampus”, dan field memasukkan prodi dengan “Masukkan Prodi”.

7. Kemudian buka FirstFragment.kt dan tulis kode berikut

```

1 package com.example.tugasfragment
2
3 import android.os.Bundle
4 import android.text.Editable
5 import android.text.TextWatcher
6 import android.view.LayoutInflater
7 import android.view.View
8 import android.view.ViewGroup
9 import androidx.fragment.app.Fragment
10 import androidx.lifecycle.ViewModelProviders
11
12 import com.google.android.material.textfield.TextInputEditText
13 /**
14  * A simple [Fragment] subclass.
15  */
16 class FirstFragment : Fragment() {
17
18     private var communicationViewModel: CommunicationViewModel? = null
19
20     override fun onCreate(savedInstanceState: Bundle?) {
21         super.onCreate(savedInstanceState)
22         communicationViewModel =
23             ViewModelProviders.of(requireActivity()).
24                 get(CommunicationViewModel::class.java)
25     }

```



```

26
27 override fun onCreateView(
28     inflater: LayoutInflater, container: ViewGroup?,
29     savedInstanceState: Bundle?
30 ): View? {
31     return inflater.inflate(R.layout.fragment_first,
32         container, attachToRoot: false)
33 }
34
35 override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
36     super.onViewCreated(view, savedInstanceState)
37     val nameEditText = view.findViewById<TextInputEditText>(R.id.textInputTextName)
38     val nameEditText2 = view.findViewById<TextInputEditText>(R.id.textInputTextKampus)
39     val nameEditText3 = view.findViewById<TextInputEditText>(R.id.textInputTextProdi)
40     nameEditText.addTextChangedListener(
41         object : TextWatcher {
42             override fun beforeTextChanged(
43                 charSequence: CharSequence, i: Int, il: Int, i2: Int
44             ) {
45             }
46
47             override fun onTextChanged(
48                 charSequence: CharSequence,
49                 i: Int, il: Int, i2: Int
50             ) {
51                 communicationViewModel!!.setName(charSequence.toString())
52             }
53
54             override fun afterTextChanged(editable: Editable) {
55             }
56         })
57     nameEditText2.addTextChangedListener(
58         object : TextWatcher {
59             override fun beforeTextChanged(
60                 charSequence: CharSequence, i: Int, il: Int, i2: Int
61             ) {
62             }
63
64             override fun onTextChanged(
65                 charSequence: CharSequence,
66                 i: Int, il: Int, i2: Int
67             ) {
68                 communicationViewModel!!.setKampus(charSequence.toString())
69             }
70
71             override fun afterTextChanged(editable: Editable) {
72             }
73         })
74     nameEditText3.addTextChangedListener(
75         object : TextWatcher {
76             override fun beforeTextChanged(
77                 charSequence: CharSequence, i: Int, il: Int, i2: Int
78             ) {
79             }
80
81             override fun onTextChanged(
82                 charSequence: CharSequence,
83                 i: Int, il: Int, i2: Int
84             ) {
85                 communicationViewModel!!.setProdi(charSequence.toString())
86             }
87
88             override fun afterTextChanged(editable: Editable) {
89             }
90         })
91 }
92
93 companion object {
94     fun newInstance(): FirstFragment {
95         return FirstFragment()
96     }
97 }
98

```


Penjelasan:

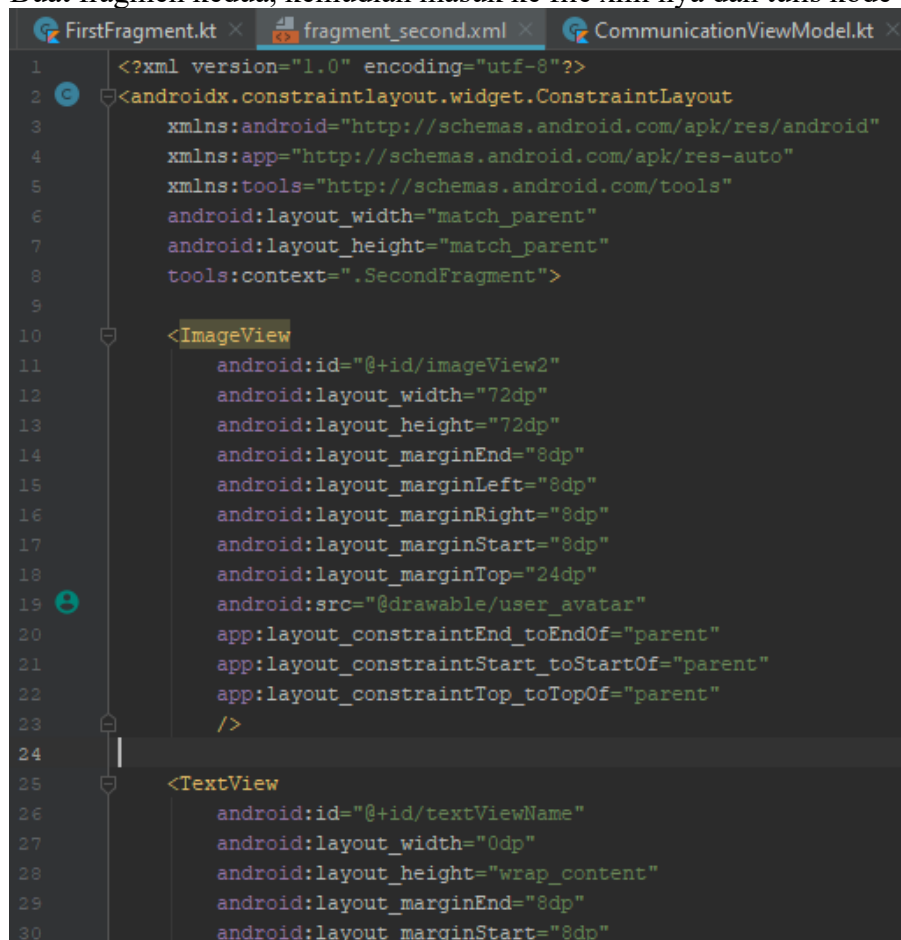
File kotlin ini adalah tempat dimana kita memproses setiap widget fragmen_first.xml.

Berikut komponen yang ada di dalam class tersebut

- **Editable**: kelas untuk teks yang konten dan markup nya dapat diubah. Editable adalah antarmuka untuk teks yang konten dan markupnya dapat diubah
- **TextWatch**: Digunakan untuk melihat input text field dan kita juga dapat langsung memperbarui data pada view lain. **TextWatch** dapat berguna untuk menghitung jumlah karakter yang dimasukkan dalam bidang teks secara instan dan mengukur kekuatan password.
- **LayoutInflater**: kelas yang digunakan untuk membuat instance file layout XML ke dalam object view yang sesuai
- **View**: View adalah basic building block UI (User Interface) di android. View adalah kotak persegi kecil yang merespons input pengguna. Contoh nya EditText, Button, CheckBox
- **ViewGroup**: adalah special view yang dapat menampung view-view lain
- **ViewModelProviders**: adalah lifecycle yang berisi utility method untuk kelas ViewModelStore & mengembalikan ViewModelProvider saat kita menggunakan metode of()

beforeTextChanged berarti keadaan field sebelum diinput data oleh user. onTextChanged adalah keadaan field saat diinput data oleh user, maka dari itu kita inisialisasikan dengan setAddress di class communicationViewModel. Terakhir adalah afterTextChanged yaitu ketika text sudah diinput, menggunakan editable supaya bisa diedit kembali oleh user.

8. Buat fragmen kedua, kemudian masuk ke file xml nya dan tulis kode berikut



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context=".SecondFragment">
9
10    <ImageView
11        android:id="@+id/imageView2"
12        android:layout_width="72dp"
13        android:layout_height="72dp"
14        android:layout_marginEnd="8dp"
15        android:layout_marginLeft="8dp"
16        android:layout_marginRight="8dp"
17        android:layout_marginStart="8dp"
18        android:layout_marginTop="24dp"
19        android:src="@drawable/user_avatar"
20        app:layout_constraintEnd_toEndOf="parent"
21        app:layout_constraintStart_toStartOf="parent"
22        app:layout_constraintTop_toTopOf="parent"
23    />
24
25    <TextView
26        android:id="@+id/textViewName"
27        android:layout_width="0dp"
28        android:layout_height="wrap_content"
29        android:layout_marginEnd="8dp"
30        android:layout_marginStart="8dp"
```

```

31         android:layout_marginTop="8dp"
32         android:gravity="center"
33         android:hint="User Display Name"
34         android:textColor="@color/colorPrimaryDark"
35         android:textSize="22sp"
36         android:textStyle="bold"
37         app:layout_constraintEnd_toEndOf="parent"
38         app:layout_constraintHorizontal_bias="0.0"
39         app:layout_constraintStart_toStartOf="parent"
40         app:layout_constraintTop_toBottomOf="@+id/textView"
41         tools:text="Haloo Dunia!" />
42
43     <TextView
44         android:id="@+id/textViewKampus"
45         android:layout_width="0dp"
46         android:layout_height="wrap_content"
47         android:layout_marginEnd="8dp"
48         android:layout_marginStart="8dp"
49         android:layout_marginTop="8dp"
50         android:gravity="center"
51         android:hint="User Display Kampus"
52         android:textColor="@color/colorPrimaryDark"
53         android:textSize="22sp"
54         android:textStyle="bold"
55         app:layout_constraintEnd_toEndOf="parent"
56         app:layout_constraintHorizontal_bias="0.0"
57         app:layout_constraintStart_toStartOf="parent"
58         app:layout_constraintTop_toBottomOf="@+id/textViewName"
59         tools:text="Haloo Dunia!" />
60
61     <TextView
62         android:id="@+id/textViewProdi"
63         android:layout_width="0dp"
64         android:layout_height="wrap_content"
65         android:layout_marginEnd="8dp"
66         android:layout_marginStart="8dp"
67         android:layout_marginTop="8dp"
68         android:gravity="center"
69         android:hint="User Display Prodi"
70         android:textColor="@color/colorPrimaryDark"
71         android:textSize="22sp"
72         android:textStyle="bold"
73         app:layout_constraintEnd_toEndOf="parent"
74         app:layout_constraintHorizontal_bias="0.0"
75         app:layout_constraintStart_toStartOf="parent"
76         app:layout_constraintTop_toBottomOf="@+id/textViewKampus"
77         tools:text="Haloo Dunia!" />
78
79     <TextView
80         android:id="@+id/textView"
81         android:layout_width="0dp"
82         android:layout_height="wrap_content"
83         android:layout_marginTop="24dp"
84         android:text="Selamat Datang"
85         android:textAlignment="center"
86         android:textSize="22sp"
87         app:layout_constraintEnd_toEndOf="parent"
88         app:layout_constraintStart_toStartOf="parent"
89         app:layout_constraintTop_toBottomOf="@+id/imageView2"/>
90 </androidx.constraintlayout.widget.ConstraintLayout>

```

Penjelasan:

Fragment_second.xml ini adalah fragment untuk menampilkan ImageView yang akan menampilkan vector yang tadi sudah kita buat. Kemudian di bawah vector tersebut, ada tempat untuk menampilkan nama, kampus, dan prodi yang sudah user input di fragment_first. Data ditampilkan dengan hint User Display Name, User Display Kampus, dan User Display Prodi.

9. Kita pindah ke SecondFragment.kt dan tulis kode berikut

```
fragment_second.xml x SecondFragment.kt x CommunicationViewModel.kt x user
1 package com.example.tugasfragment
2
3 import android.os.Bundle
4 import android.view.LayoutInflater
5 import android.view.View
6 import android.view.ViewGroup
7 import android.widget.TextView
8 import androidx.lifecycle.Observer
9 import androidx.fragment.app.Fragment
10 import androidx.lifecycle.ViewModelProviders
11
12 /**
13  * A simple [Fragment] subclass.
14  */
15 class SecondFragment : Fragment() {
16
17     private var communicationViewModel: CommunicationViewModel? = null
18     private var txtName: TextView? = null
19     private var txtKampus: TextView? = null
20     private var txtProdi: TextView? = null
21
22     override fun onCreate(savedInstanceState: Bundle?) {
23         super.onCreate(savedInstanceState)
24         communicationViewModel = ViewModelProviders.
25             of(requireActivity()).
26             get(CommunicationViewModel::class.java)
27     }
28
29     override fun onCreateView(
30         inflater: LayoutInflater, container: ViewGroup?,
31         savedInstanceState: Bundle?
32     ): View? {
33         return inflater.inflate(R.layout.fragment_second,
34             container, attachToRoot: false)
35     }
36
37     override fun onViewCreated(view: View, savedInstanceState: Bundle?)
38     {
39         super.onViewCreated(view, savedInstanceState)
40         txtName = view.findViewById(R.id.textViewName)
41         communicationViewModel!!.name.observe(requireActivity(),
42             Observer { s -> txtName!!.text = s })
43
44         super.onViewCreated(view, savedInstanceState)
45         txtKampus = view.findViewById(R.id.textViewKampus)
46         communicationViewModel!!.kampus.observe(requireActivity(),
47             Observer { s -> txtKampus!!.text = s })
48
49         super.onViewCreated(view, savedInstanceState)
50         txtProdi = view.findViewById(R.id.textViewProdi)
51         communicationViewModel!!.prodi.observe(requireActivity(),
52             Observer { s -> txtProdi!!.text = s })
53     }
54     companion object {
55         fun newInstance(): SecondFragment {
56             return SecondFragment()
57         }
58     }
59 }
```

Penjelasan:

komponen yang ada di dalam class tersebut

- **LayoutInflater**: kelas yang digunakan untuk membuat instance file layout XML ke dalam object view yang sesuai
- **View**: View adalah basic building block UI (User Interface) di android. View adalah kotak persegi kecil yang merespons input pengguna. Contoh nya EditText, Button, CheckBox

- **ViewGroup**: adalah special view yang dapat menampung view-view lain
- **ViewModelProviders**: adalah lifecycle yang berisi utility method untuk kelas ViewModelStore & mengembalikan ViewModelProvider saat kita menggunakan metode of()
- **Observer**: Observer adalah method yang dipanggil setiap kali objek yang diobservasi diubah. Aplikasi memanggil metode notifyObservers objek Observable agar semua observer objek diberi tahu tentang perubahan tersebut

Di dalam class ini, kita mengobserve seluruh variabel yang sudah diberi method setter dan getter di CommunicationViewModel. Untuk meng-observe, kita harus membuat variabel dahulu untuk setiap data yang akan dimasukkan yaitu txtName, txtKampus, dan txtProdi. Kemudian kita inisialisasikan dengan widget yang ada di xml yaitu textViewName, textViewKampus, dan textViewProdi

10. Buat class adapter dengan kode berikut

```

1 package com.example.tugasfragment
2
3 import android.content.Context;
4 import androidx.annotation.StringRes;
5 import androidx.fragment.app.Fragment;
6 import androidx.fragment.app.FragmentManager;
7 import androidx.fragment.app.FragmentPagerAdapter;
8
9 class ViewPagerAdapter(private val mContext: Context, fm: FragmentManager) :
10     FragmentPagerAdapter(fm) {
11
12     override fun getItem(position: Int): Fragment {
13         return if (position == 0) {
14             FirstFragment.newInstance()
15         } else {
16             SecondFragment.newInstance()
17         }
18     }
19
20     override fun getPageTitle(position: Int): CharSequence? {
21         return mContext.resources.getString(TAB_TITLES[position])
22     }
23
24     override fun getCount(): Int {
25         return 2
26     }
27
28     companion object {
29         @StringRes
30         private val TAB_TITLES = intArrayOf("Input",
31             "Hasil")
32     }
33 }

```

Penjelasan:

ViewPager di Android adalah kelas yang memungkinkan pengguna untuk membalik atau menggeser halaman data ke kiri dan kanan. Kelas ini menyediakan fungsionalitas untuk membalik halaman di app. ViewPager adalah widget yang ditemukan di support library.

Untuk menggunakannya, kita harus meletakkan elemen di dalam file layout XML. Sementara objek Adapter bertindak sebagai jembatan antara AdapterView dan data yang mendasari tampilan tersebut. Adapter menyediakan akses ke item data. Adapter juga bertanggung jawab untuk membuat tampilan untuk setiap item dalam dataset.

Di dalam adapter tersebut kita membuat seleksi kondisi. Position 0 atau slide 1 adalah milik FirstFragment, sementara Position 1 (else) atau slide 2 adalah milik SecondFragment. Kemudian kita beri nama pada tab dengan string yang sudah kita buat tadi yaitu INPUT dan HASIL dengan kode private val TAB_TITLES = intArrayOf("Input", "Hasil")

11. Terakhir buka MainActivity kemudian tulis kode berikut

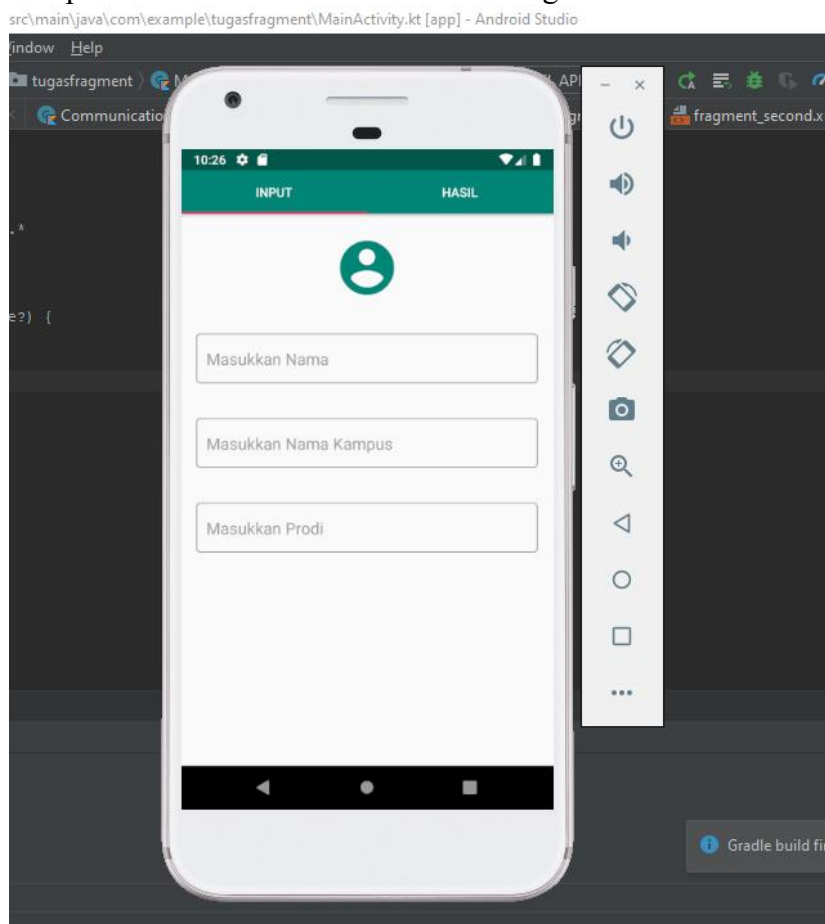
```
MainActivity.kt x fragment_first.xml x FirstFragment.kt x fragm
1 package com.example.tugasfragment
2
3 import android.os.Bundle
4 import androidx.appcompat.app.AppCompatActivity
5 import kotlinx.android.synthetic.main.activity_main.*
6
7 class MainActivity : AppCompatActivity() {
8
9     override fun onCreate(savedInstanceState: Bundle?) {
10         super.onCreate(savedInstanceState)
11         setContentView(R.layout.activity_main)
12
13         view_pager.adapter = ViewPagerAdapter(
14             mContext: this, supportFragmentManager
15         )
16         tabs.setupWithViewPager(view_pager)
17     }
18 }
```

Penjelasan:

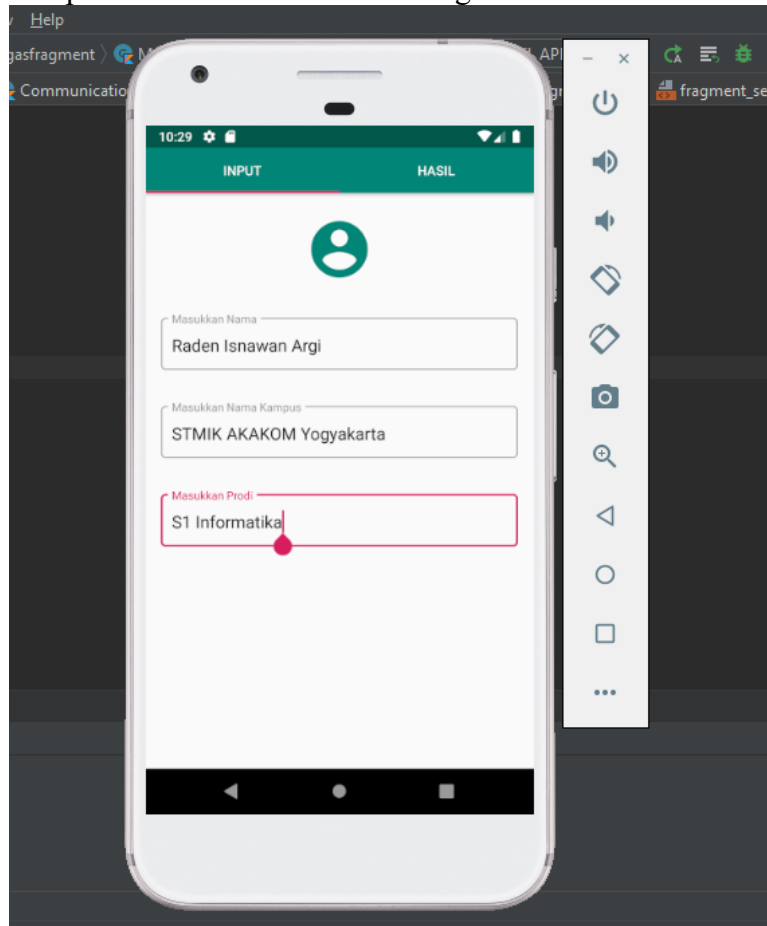
Di dalam MainActivity.kt, kita menginisialisasikan view_pager dengan method adapter yang dihubungkan dengan class ViewPagerAdapter supaya segala kode nya bisa diproses. Kemudian kita tulis kode 'this' lalu dilanjutkan dengan supportFragmentManager. FragmentManager adalah kelas yang bertanggung jawab untuk melakukan tindakan pada fragment aplikasi kita seperti menambahkan, menghapus, mengganti, dan menambahkannya ke back-stack. Terakhir kita tulis **tabs.setupWithViewPager(view_pager)**.

12. Jalankan aplikasi

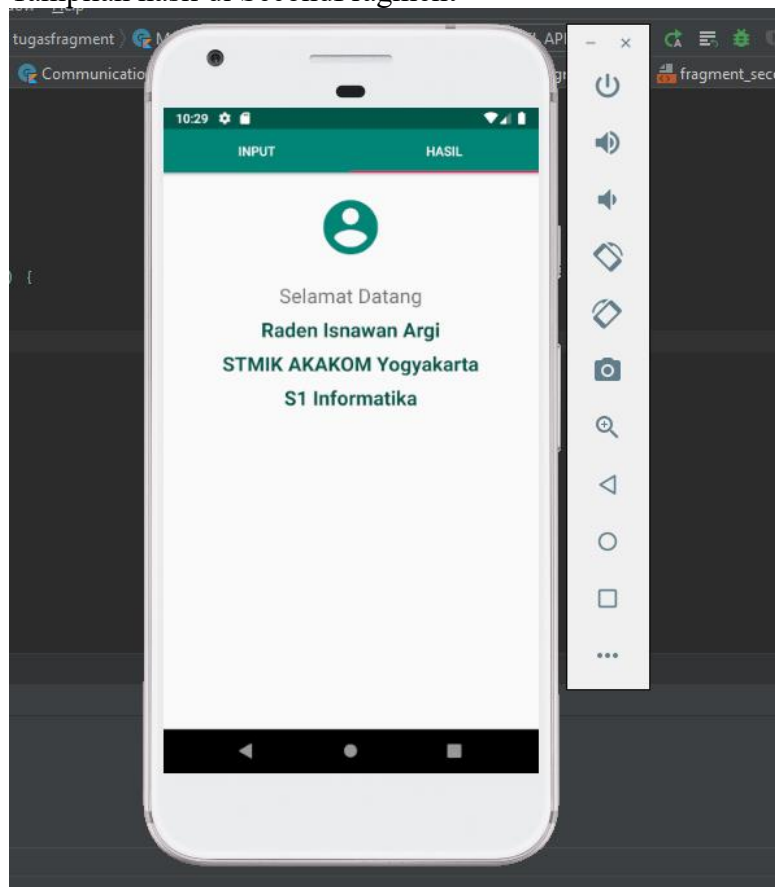
Tampilan saat data belum diisi di FirstFragment



Tampilan saat data diisi di FirstFragment



Tampilan hasil di SecondFragment



KESIMPULAN

Di pertemuan ke-6 (fragment) ini, saya mampu memenuhi tujuan dibuatnya modul ini yaitu dapat membuat aplikasi dengan menggunakan fragment. Saya pun tidak hanya dapat membuat satu fragmen saja, tetapi di praktik saya juga mampu menerapkan beberapa fragmen di beberapa tab dengan menggunakan viewmodel. Di latihan, saya juga mampu menambahkan satu lagi fragmen sehingga total fragmen menjadi 3. Di tugas, saya menyederhanakan fragmen menjadi 2 kembali, tetapi dengan input data yang lebih banyak sehingga aplikasi menjadi lebih efisien dan tidak memerlukan terlalu banyak fragment atau tab untuk menampilkan beberapa data yang sudah diinput oleh user. Walaupun saya masih pemula di pemrograman mobile khususnya Android, tetapi modul ini cukup mudah dipahami dan saya mendapatkan banyak ilmu tentang Android khususnya fragment di modul 6 ini.

Terima Kasih