Politeknik Caltex Riau

LAPORAN PRAKTIKUM

Pemrograman Mobile

Dosen Pengampu
Shumaya Resty Ramadhani, S.ST., M.Sc.

AIL
Puspita Aisyah Asnur S.Tr.Kom

Identitas :

No Absen : 13

Nama Lengkap : Laurin Madelau

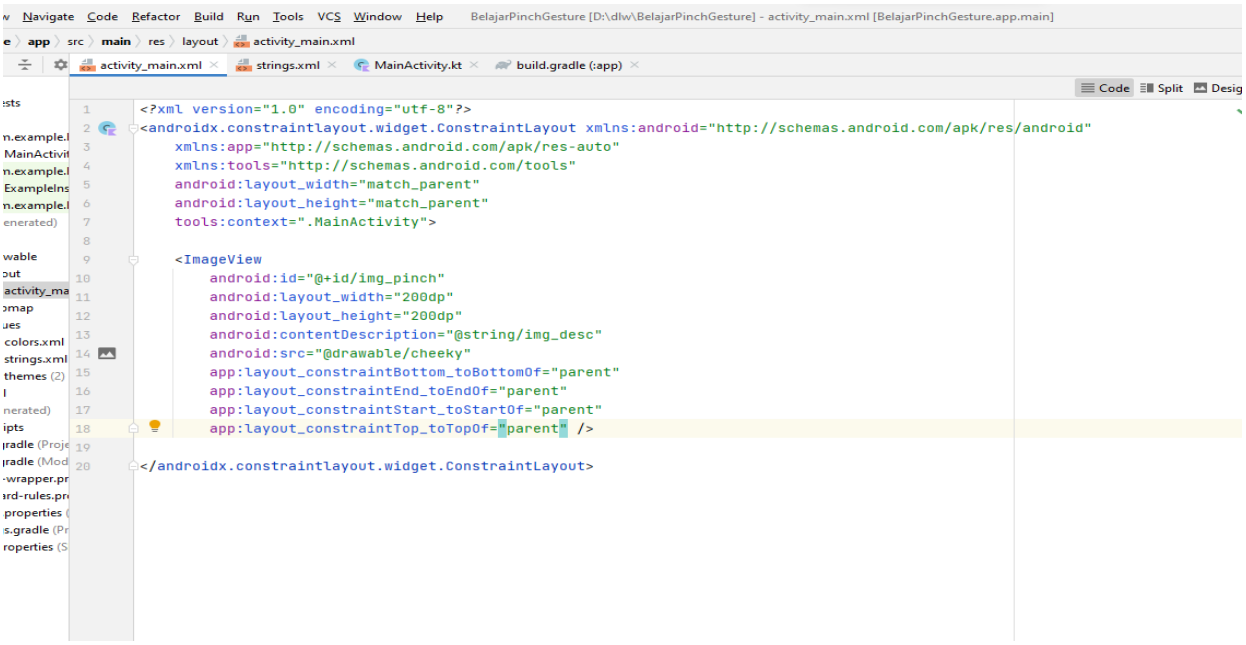
NIM : 2055301067

Kelas : 3 TI C

Pertemuan 13

Gesture

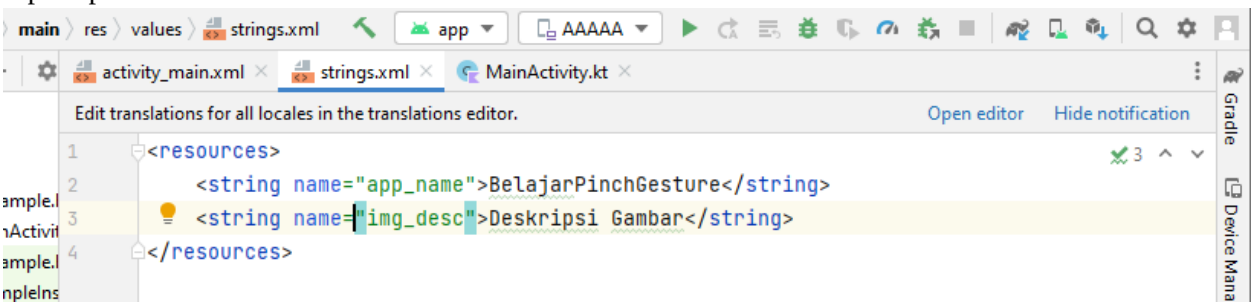
Untuk percobaan kali ini kita akan mencoba mengaplikasikan fungsi pinch tersebut kedalam sebuah aplikasi android. Silahkan buat sebuah projek baru bernama “BelajarPinchGesture”. Kemudian buka halaman activity_main.xml untuk membuat tampilan awal. Silahkan tambahkan sebuah icon/gambar ke dalam folder drawable untuk dapat digunakan pada halaman ini. Kita akan menggunakan gambar tersebut untuk menjalankan aksi pinch ini.



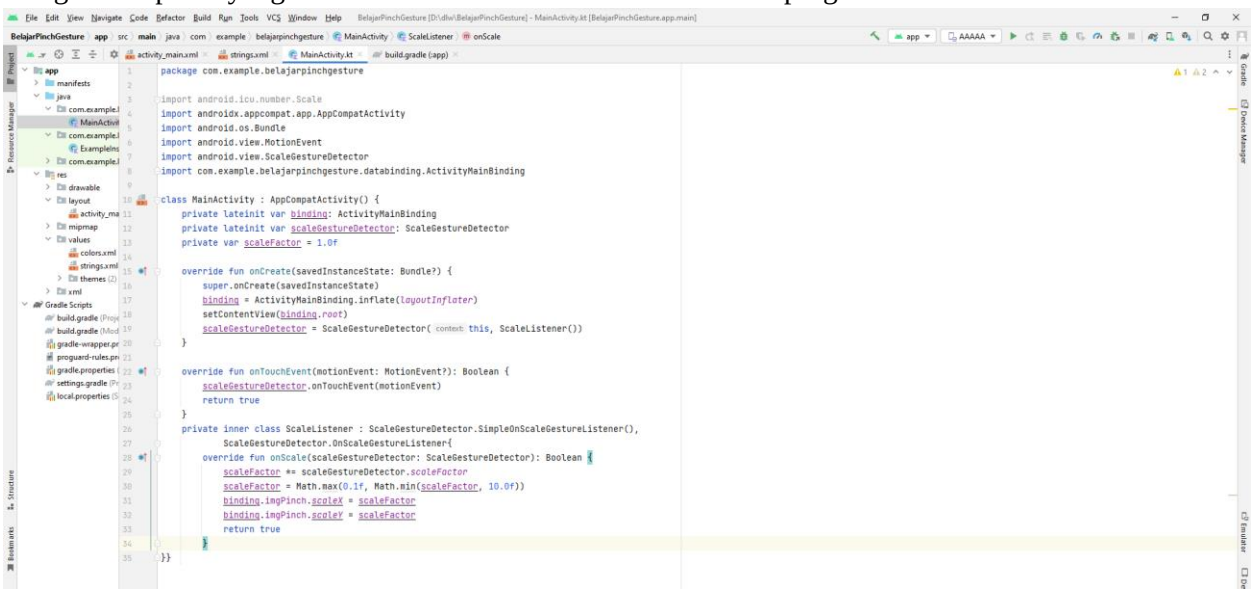
Analisa:

Pada file diatas menambahkan sebuah ImageView yang akan ditampilkan pada aplikasi, dengan memanggil gambar pada @drawable/cheeky.

Kemudian silahkan tambahkan content description yang disimpan dalam file strings.xml seperti pada contoh di bawah ini.



Setelah itu, buka file MainActivity.kt untuk menambahkan kode program kotlin. Terlihat pada kode di bawah bahwa kita menggunakan class ScaleGestureDetector agar aplikasinya dapat mengenali inputan yang diberikan user. Silahkan tambahkan kode program di bawah ini.

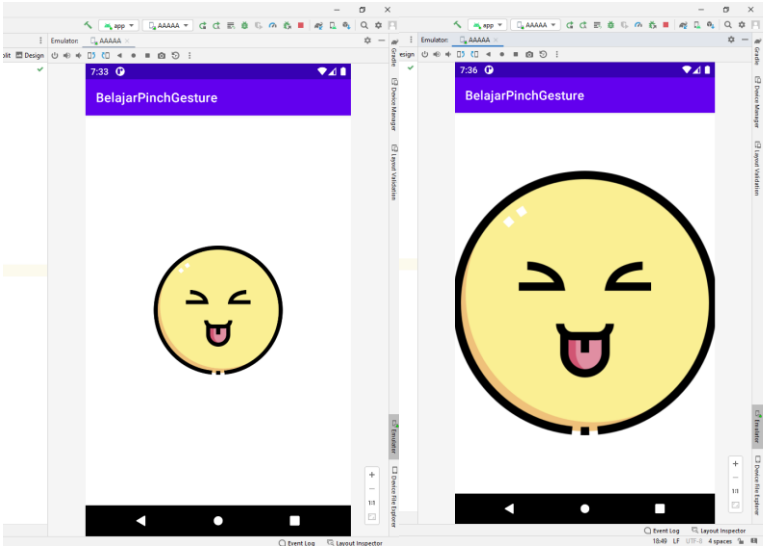


```kotlin
package com.example.belajarpinchgesture

import android.icu.number.Scale
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.MotionEvent
import android.view.ScaleGestureDetector
import com.example.belajarpinchgesture.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding
    private lateinit var scaleGestureDetector: ScaleGestureDetector
    private var scaleFactor = 1.0f

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
        scaleGestureDetector = ScaleGestureDetector( context: this, ScaleListener())
    }

    override fun onTouchEvent(motionEvent: MotionEvent?): Boolean {
        scaleGestureDetector.onTouchEvent(motionEvent)
        return true
    }
    private inner class ScaleListener : ScaleGestureDetector.SimpleOnScaleGestureListener(),
            ScaleGestureDetector.OnScaleGestureListener{
        override fun onScale(scaleGestureDetector: ScaleGestureDetector): Boolean {
            scaleFactor *= scaleGestureDetector.scaleFactor
            scaleFactor = Math.max(0.1f, Math.min(scaleFactor, 10.0f))
            binding.imgPinch.scaleX = scaleFactor
            binding.imgPinch.scaleY = scaleFactor
            return true
        }
}}
```

Analisa :

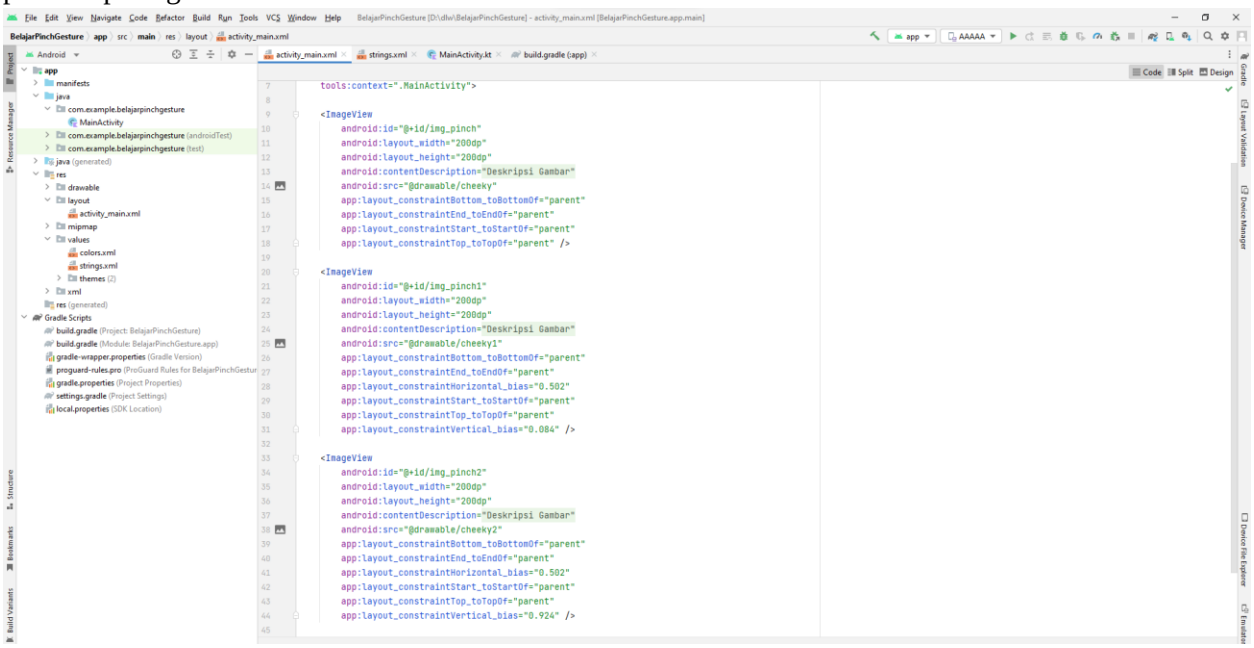
Pada file diatas terdapat ScaleGestureDetector yang mana digunakan untuk melakukan scaling pada aplikasi. Terdapat function onTouchEvent() yang berfungsi untuk mengidenntifikasi jari atau sentuhan ketika kita menyentuh layar. MotionEvent berfungsi untuk memberika detai detiap interaksi yang tejadi pada sentuhan layar.

Setelah selesai, maka silahkan jalankan aplikasi anda. Jika berhasil, maka ketika anda lakukan aksi cubit/pinch di layar/emulator, gambar akan membesar dan mengecil seperti pada contoh dibawah.

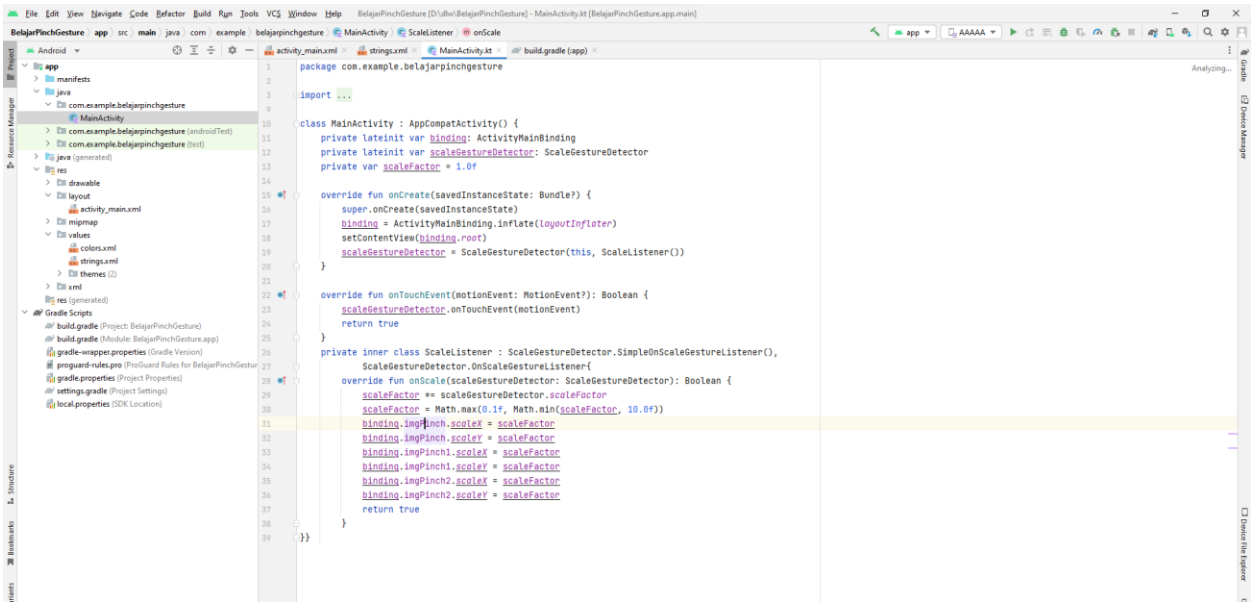


Tugas Analisa tambahan :

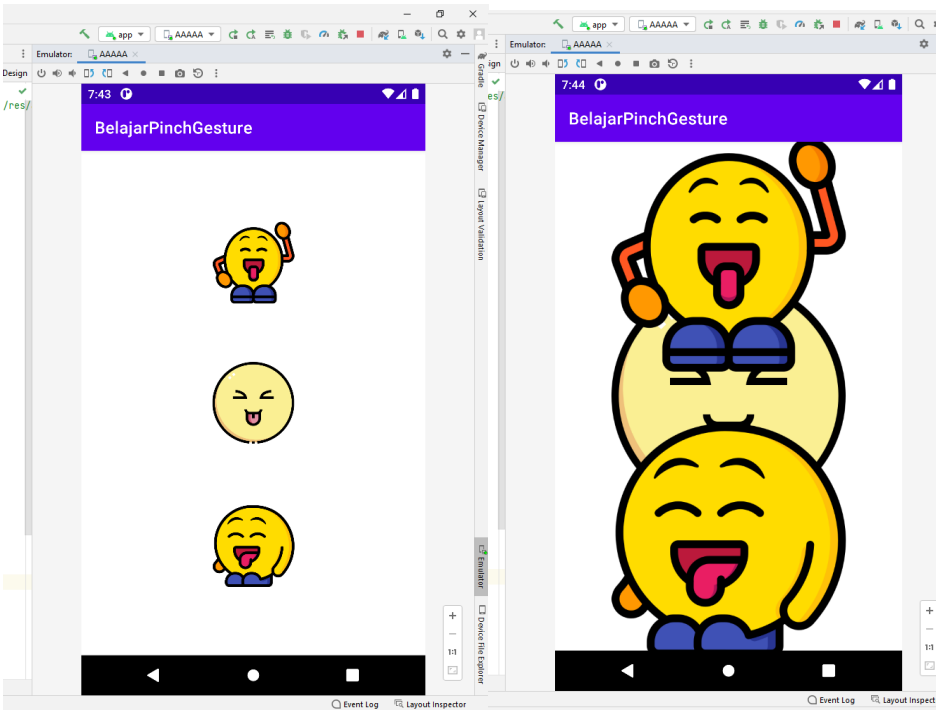
1. Silahkan tambahkan dua buah gambar lagi pada projek ini (gambar bebas), dan aktifkan fungsi pinch kepada gambar tersebut.



```
tools:context=".MainActivity">

<ImageView
    android:id="@+id/img_pinch"
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:contentDescription="Deskripsi Gambar"
    android:src="@drawable/cheeky"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<ImageView
    android:id="@+id/img_pinch1"
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:contentDescription="Deskripsi Gambar"
    android:src="@drawable/cheeky1"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.502"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.084" />

<ImageView
    android:id="@+id/img_pinch2"
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:contentDescription="Deskripsi Gambar"
    android:src="@drawable/cheeky2"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.502"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.924" />
```

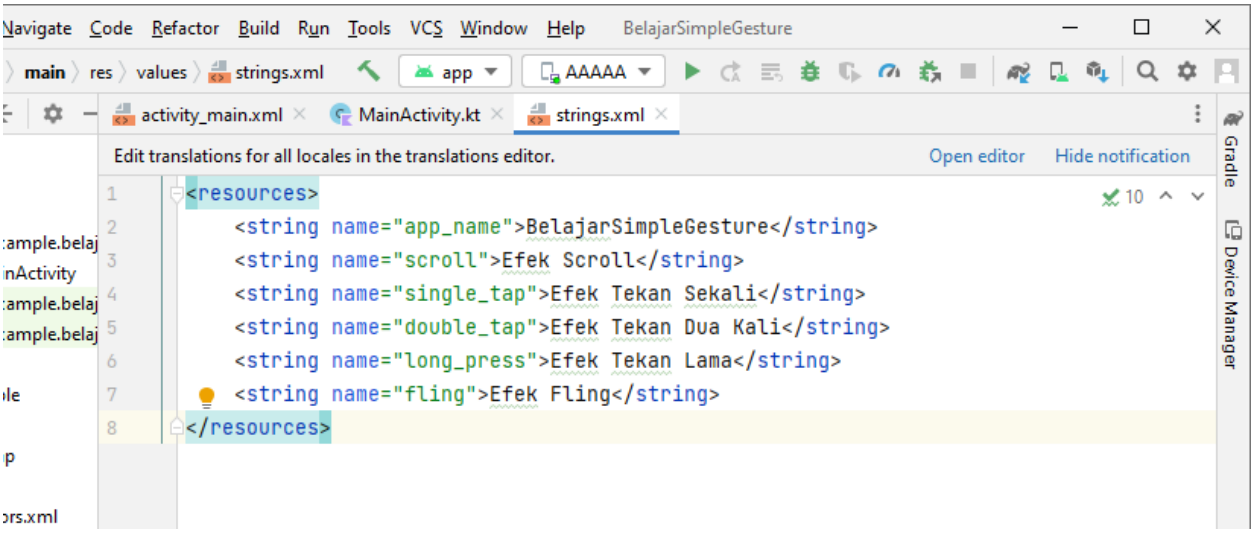


```
package com.example.belajarpinchgestureimport ...class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding
    private lateinit var scaleGestureDetector: ScaleGestureDetector
    private var scaleFactor = 1.0f

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
        scaleGestureDetector = ScaleGestureDetector(this, ScaleListener())
    }

    override fun onTouchEvent(motionEvent: MotionEvent?): Boolean {
        scaleGestureDetector.onTouchEvent(motionEvent)
        return true
    }
    private inner class ScaleListener : ScaleGestureDetector.SimpleOnScaleGestureListener(),
            ScaleGestureDetector.OnScaleGestureListener{
        override fun onScale(scaleGestureDetector: ScaleGestureDetector): Boolean {
            scaleFactor *= scaleGestureDetector.scaleFactor
            scaleFactor = Math.max(0.1f, Math.min(scaleFactor, 10.0f))
            binding.imgPinch.scaleX = scaleFactor
            binding.imgPinch.scaleY = scaleFactor
            binding.imgPinch1.scaleX = scaleFactor
            binding.imgPinch1.scaleY = scaleFactor
            binding.imgPinch2.scaleX = scaleFactor
            binding.imgPinch2.scaleY = scaleFactor
            return true
        }
}}
```

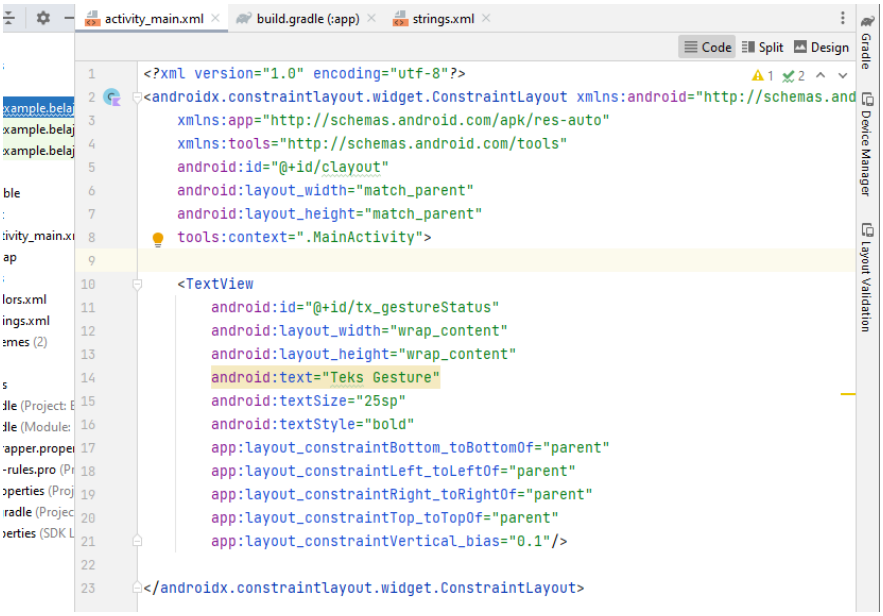

7:43
BelajarPinchGesture
7:44
BelajarPinchGesture

Belajar Gesture 2

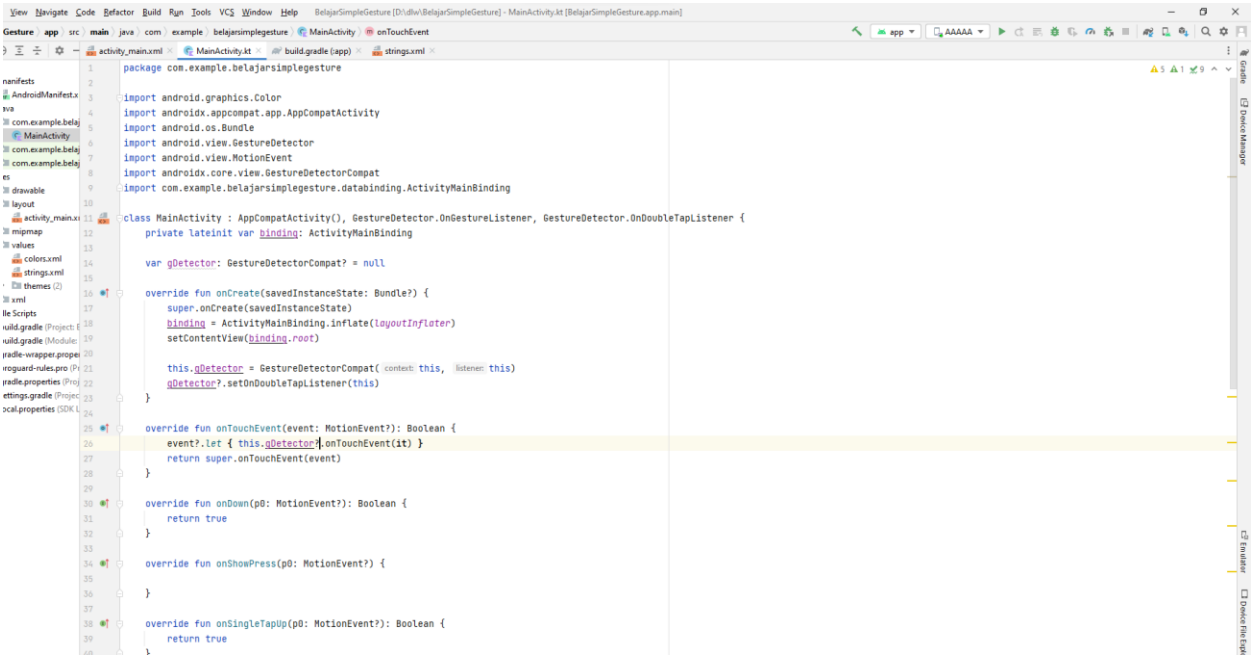
Pada percobaan kedua ini, kita akan mencoba mendeteksi berbagai macam bentuk sentuhan yang kerap digunakan pada aplikasi. Dengan menggunakan onGestureListener dan OnDoubleTapListener, maka kita harus mengimplementasikan berbagai macam bentuk gesture pada saat pembuatan aplikasi. Jika tidak, maka pembuatan program akan mengalami compilation error karena ada method yang belum terimplementasi. Silahkan buat sebuah projek baru bernama “BelajarSimpleGesture”. Kemudian tambahkan beberapa teks berikut pada halaman strings.xml yang akan digunakan di aplikasi.



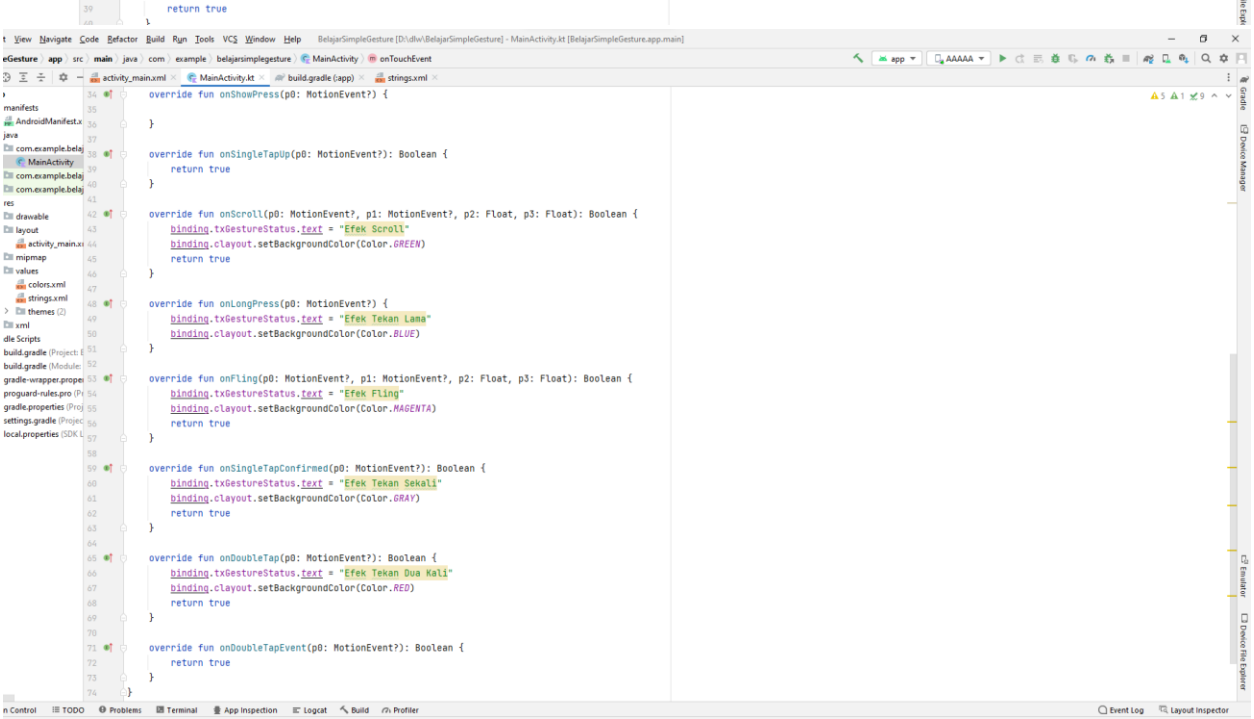
Setelah itu, kita akan membuat UI sederhana dengan menggunakan sebuah TextView sebagai penanda gesture apa yang kita lakukan pada aplikasi. Buka halaman activity_main.xml.



Kemudian buka file MainActivity.kt, kemudian tambahkan kode program berikut ini. Seperti yang telah disinggung diatas bahwa ketika mengimplementasikan OnGesture dan OnDoubleTapListener, maka beberapa method harus dibuat. Antara lain onTouchEvent, onDown, onShowPress, onSingleTap, dll. Meski begitu, method tersebut tidak harus diisi dengan event. Kita dapat membiarkannya kosong dengan mengisi method yang digunakan saja. Pada contoh ini, setiap kali event terjadi, maka akan menampilkan teks dan mengganti warna latar pada tampilan aplikasi. Anda dapat melakukan modifikasi sesuai dengan keinginan.

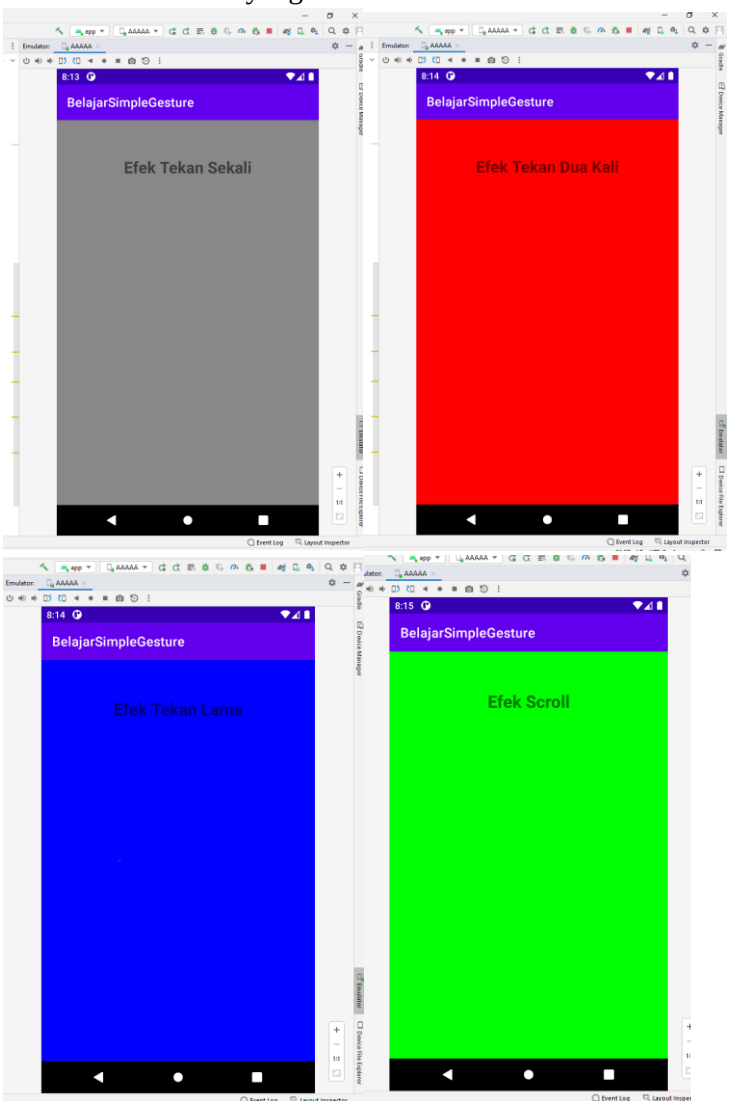


```
package com.example.belajarsimplegesture

import android.graphics.Color
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.GestureDetector
import android.view.MotionEvent
import androidx.core.view.GestureDetectorCompat
import com.example.belajarsimplegesture.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity(), GestureDetector.OnGestureListener, GestureDetector.OnDoubleTapListener {
    private lateinit var binding: ActivityMainBinding

    var gDetector: GestureDetectorCompat? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        this.gDetector = GestureDetectorCompat( context: this,  listener: this)
        gDetector?.setOnDoubleTapListener(this)
    }

    override fun onTouchEvent(event: MotionEvent?): Boolean {
        event?.let { this.gDetector?.onTouchEvent(it) }
        return super.onTouchEvent(event)
    }

    override fun onDown(p0: MotionEvent?): Boolean {
        return true
    }

    override fun onShowPress(p0: MotionEvent?) {

    }

    override fun onSingleTapUp(p0: MotionEvent?): Boolean {
        return true
    }
```



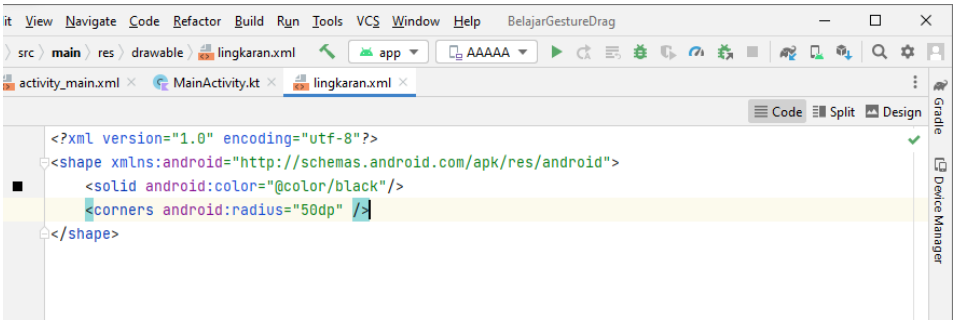
```
    override fun onShowPress(p0: MotionEvent?) {

    }

    override fun onSingleTapUp(p0: MotionEvent?): Boolean {
        return true
    }

    override fun onScroll(p0: MotionEvent?, p1: MotionEvent?, p2: Float, p3: Float): Boolean {
        binding.txGestureStatus.text = "Efek Scroll"
        binding.clayout.setBackgroundColor(Color.GREEN)
        return true
    }

    override fun onLongPress(p0: MotionEvent?) {
        binding.txGestureStatus.text = "Efek Tekan Lama"
        binding.clayout.setBackgroundColor(Color.BLUE)
    }

    override fun onFling(p0: MotionEvent?, p1: MotionEvent?, p2: Float, p3: Float): Boolean {
        binding.txGestureStatus.text = "Efek Fling"
        binding.clayout.setBackgroundColor(Color.MAGENTA)
        return true
    }

    override fun onSingleTapConfirmed(p0: MotionEvent?): Boolean {
        binding.txGestureStatus.text = "Efek Tekan Sekali"
        binding.clayout.setBackgroundColor(Color.GRAY)
        return true
    }

    override fun onDoubleTap(p0: MotionEvent?): Boolean {
        binding.txGestureStatus.text = "Efek Tekan Dua Kali"
        binding.clayout.setBackgroundColor(Color.RED)
        return true
    }

    override fun onDoubleTapEvent(p0: MotionEvent?): Boolean {
        return true
    }
}
```

Jika sudah selesai, maka silahkan jalankan aplikasi. Coba lakukan berbagai macam aksi seperti menekan layar sekali, dua kali, atau menahan lebih lama. Maka pada setiap aksi akan menghasilkan teks dan warna latar yang berbeda-beda.

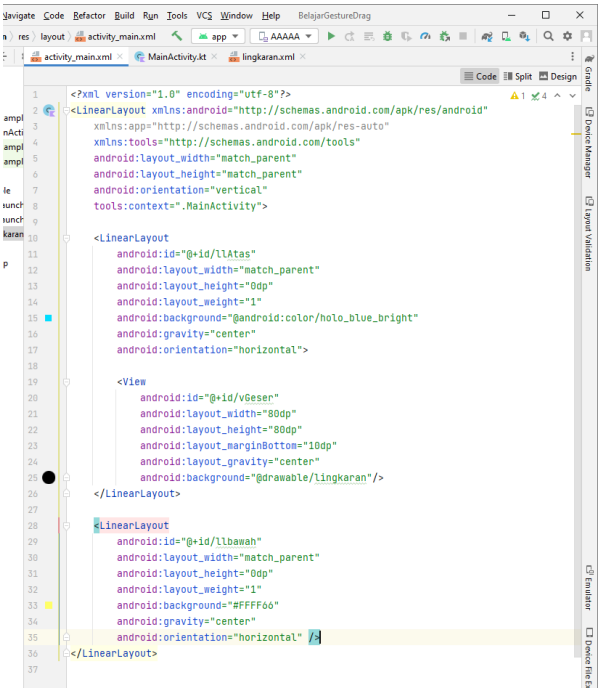


Belajar Gesture 3

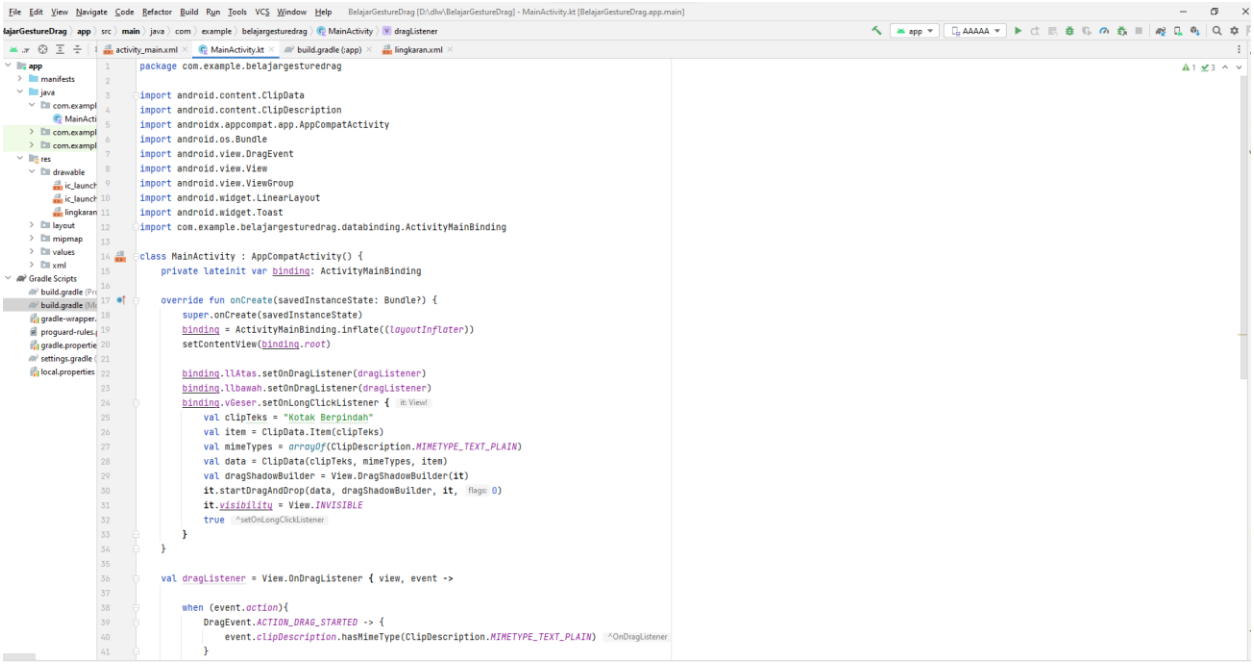
Pada percobaan ini kita akan membuat sebuah aplikasi yang mengimplementasikan fungsi drag dan drop untuk memindahkan sebuah objek. Agar dapat memanfaatkan fitur ini, kita harus menggunakan OnDragListener yang merupakan turunan dari DragEvent. Dengan menggunakan event ini, maka sistem dapat mendeteksi sentuhan pengguna pada layar hp. Silahkan buat sebuah projek baru bernama “BelajarGestureDrag”. Kita akan memindahkan sebuah objek berbentuk lingkaran dari satu layout ke layout lainnya pada sebuah halaman. Silahkan buat sebuah file layout baru di dalam folder drawable dan beri nama lingkaran.xml. Lalu ketikkan kode xml berikut ini.



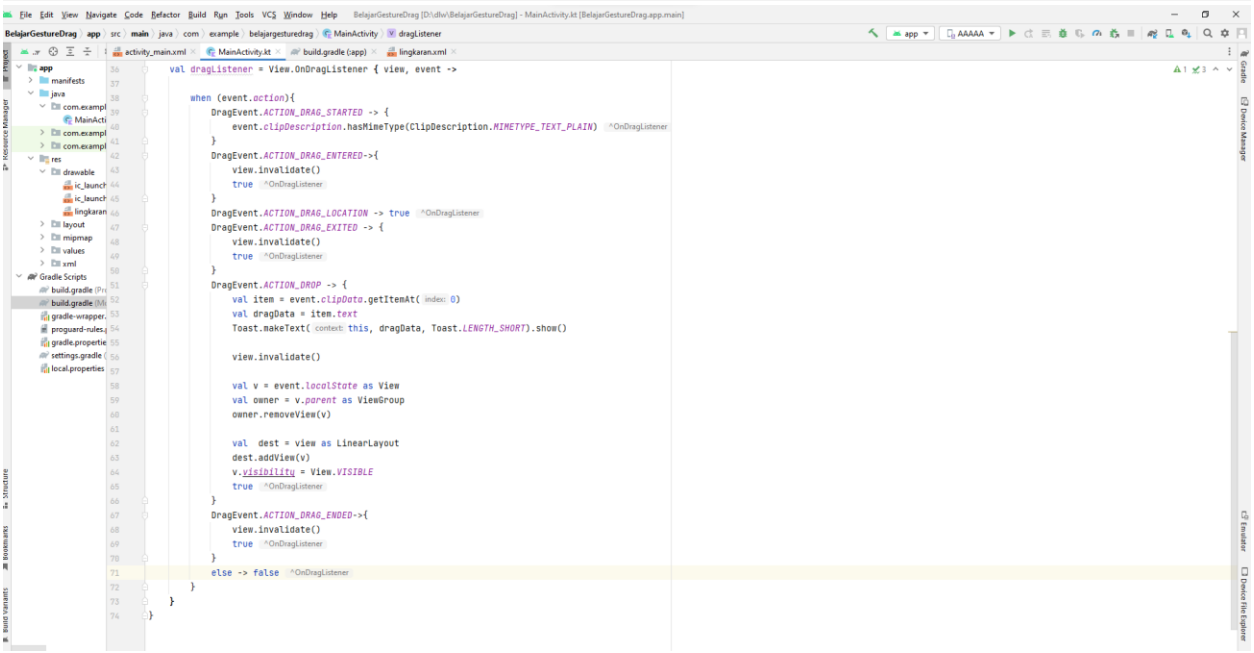
Setelahnya, kita akan membuka kembali file activity_main.xml. Lalu kita tambahkan dua buah layout. Lalu kita menambahkan sebuah view yang merupakan objek lingkaran yang akan berpindah antar layout. View ini akan menggunakan file lingkaran diatas untuk membuat objek berbentuk oval.



Setelahnya, buka file MainActivity.kt. Pada file ini kita akan menggunakan onDragListener. Event ini akan diterapkan pada objek lingkaran yang telah kita buat dalam bentuk view pada file xml diatas. Ketika event ini diimplementasikan pada objek view, maka objek view dapat mendeteksi sentuhan dari awal layar ditekan hingga objek diajak berpindah antar layout.

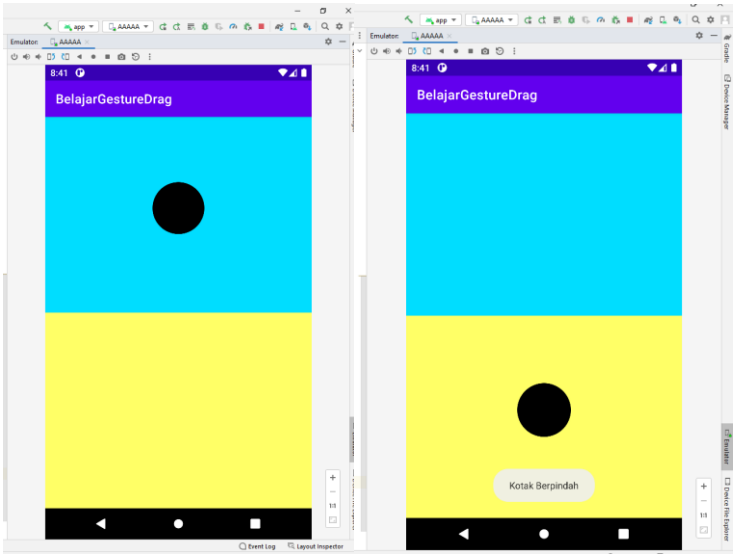


```
package com.example.belajargesturedrag

import android.content.ClipData
import android.content.ClipDescription
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.DragEvent
import android.view.View
import android.view.ViewGroup
import android.widget.LinearLayout
import android.widget.Toast
import com.example.belajargesturedrag.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.llAtas.setOnDragListener(dragListener)
        binding.llbawah.setOnDragListener(dragListener)
        binding.vGeser.setOnLongClickListener {
            val clipTeks = "Kotak Berpindah"
            val item = ClipData.Item(clipTeks)
            val mimeTypes = arrayOf(ClipDescription.MIMETYPE_TEXT_PLAIN)
            val data = ClipData(clipTeks, mimeTypes, item)
            val dragShadowBuilder = View.DragShadowBuilder(it)
            it.startDragAndDrop(data, dragShadowBuilder, it, 0)
            it.visibility = View.INVISIBLE
            true
        }
    }

    val dragListener = View.OnDragListener { view, event ->

        when (event.action){
            DragEvent.ACTION_DRAG_STARTED -> {
                event.clipDescription.hasMimeType(ClipDescription.MIMETYPE_TEXT_PLAIN)
            }
```



```
    val dragListener = View.OnDragListener { view, event ->

        when (event.action){
            DragEvent.ACTION_DRAG_STARTED -> {
                event.clipDescription.hasMimeType(ClipDescription.MIMETYPE_TEXT_PLAIN)
            }
            DragEvent.ACTION_DRAG_ENTERED->{
                view.invalidate()
                true
            }
            DragEvent.ACTION_DRAG_LOCATION -> true
            DragEvent.ACTION_DRAG_EXITED -> {
                view.invalidate()
                true
            }
            DragEvent.ACTION_DROP -> {
                val item = event.clipData.getItemAt(0)
                val dragData = item.text
                Toast.makeText(this, dragData, Toast.LENGTH_SHORT).show()

                view.invalidate()

                val v = event.localState as View
                val owner = v.parent as ViewGroup
                owner.removeView(v)

                val  dest = view as LinearLayout
                dest.addView(v)
                v.visibility = View.VISIBLE
                true
            }
            DragEvent.ACTION_DRAG_ENDED->{
                view.invalidate()
                true
            }
            else -> false
        }
    }
}
```

Kemudian jika sudah selesai mengetik kode program diatas, maka silahkan jalankan aplikasi. Kita menekan agak lama pada objek lingkaran tersebut, dan arahkan ke layout yang telah diberikan warna berbeda. Maka setelah objek berpindah, akan menampilkan toast yang berbeda seperti pada tampilan aplikasi di bawah ini.



Tugas Analisa :
1. Apa fungsi dari masing-masing aksi ini :
a. ACTION_DRAG_STARTED,
b. ENTERED,
c. LOCATION,
d. EXITED,
e. DROP,
f. ENDED

Jawab :

a. ACTION_DRAG_STARTED → menampilkan bayangan seret, yang dapat berupa bayangan atau tampilan sebenarnya yang dapat diseret, pada perangkat.

b. ENTERED → Pemroses peristiwa drag objek View menerima jenis tindakan peristiwa ini saat bayangan drag baru saja memasuki kotak pembatas View. Ini adalah jenis tindakan peristiwa pertama yang diterima pemroses saat bayangan drag memasuki kotak pembatas.

c. LOCATION → Pemroses peristiwa drag objek View menerima jenis tindakan peristiwa ini setelah menerima peristiwa ACTION_DRAG_ENTERED sementara bayangan drag masih berada dalam kotak pembatas View.

d. EXITED → Pemroses peristiwa drag objek View menerima jenis tindakan peristiwa ini setelah menerima ACTION_DRAG_ENTERED dan minimal satu peristiwa ACTION_DRAG_LOCATION, dan setelah pengguna memindahkan bayangan drag ke luar kotak pembatas View.

e. DROP → Pemroses peristiwa drag objek View menerima jenis tindakan peristiwa ini saat pengguna men-drop bayangan drag di atas objek View. Jenis tindakan ini hanya dikirim ke pemroses objek View jika pemroses menampilkan boolean true sebagai respons terhadap peristiwa drag ACTION_DRAG_STARTED.

f. ENDED → Pemroses peristiwa drag objek View menerima jenis tindakan peristiwa ini saat sistem mengakhiri operasi drag. Jenis tindakan ini tidak perlu didahului oleh peristiwa ACTION_DROP. Jika sistem mengirim ACTION_DROP, penerimaan jenis tindakan ACTION_DRAG_ENDED tidak berarti bahwa operasi drop berhasil.