UJIAN TENGAH SEMESTER

PEMROGRAMAN MOBILE 1

Dosen Pengajar : Ibu
Nova Agustina, ST., M.Kom.

Di susun Oleh :
Nama : Haidir Mirza Ahmad Zacky
NPM : 23552011072

Kelas/Semester : TIF RP 23 CNS B | Semester 4
Program Studi Keahlian : Teknik Informatika
Mata Kuliah : Pemrograman Mobile 1

DEPARTEMEN
TEKNIK INFORMATIKA
UNIVERSITAS TEKNOLOGI BANDUNG
TAHUN 2025

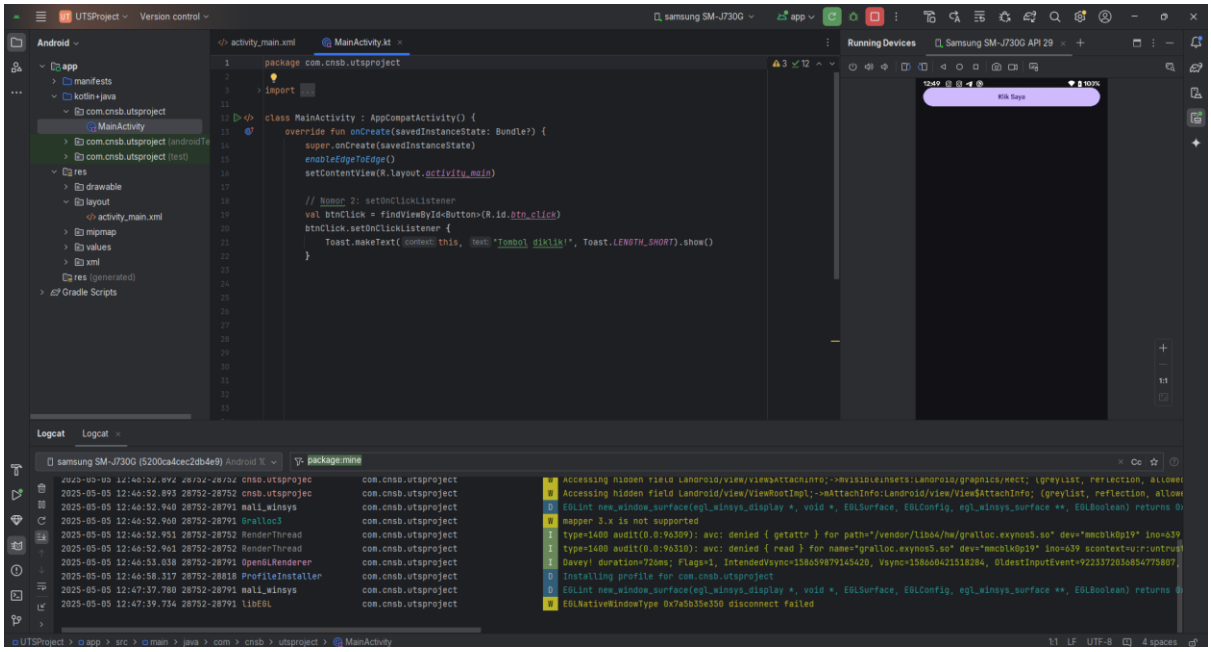
Essay

1. Apa fungsi setOnClickListener?
2. Apa syarat pemanggilan method setOnClickListener? Buat contohnya dan screenshot source code nya!
3. Error apa yang terjadi jika file kotlin salah menginisialisasi findViewById atau objek pada xml belum diinisialisasi? Screenshot logcat-nya!
4. Buat sebuah contoh program untuk menampilkan pesan error NullPointerException!
5. Kumpulkan dalam bentuk pdf di Elearning (Soal essay digabung dengan soal studi kasus cek point 7 Studi Kasus)

Jawaban

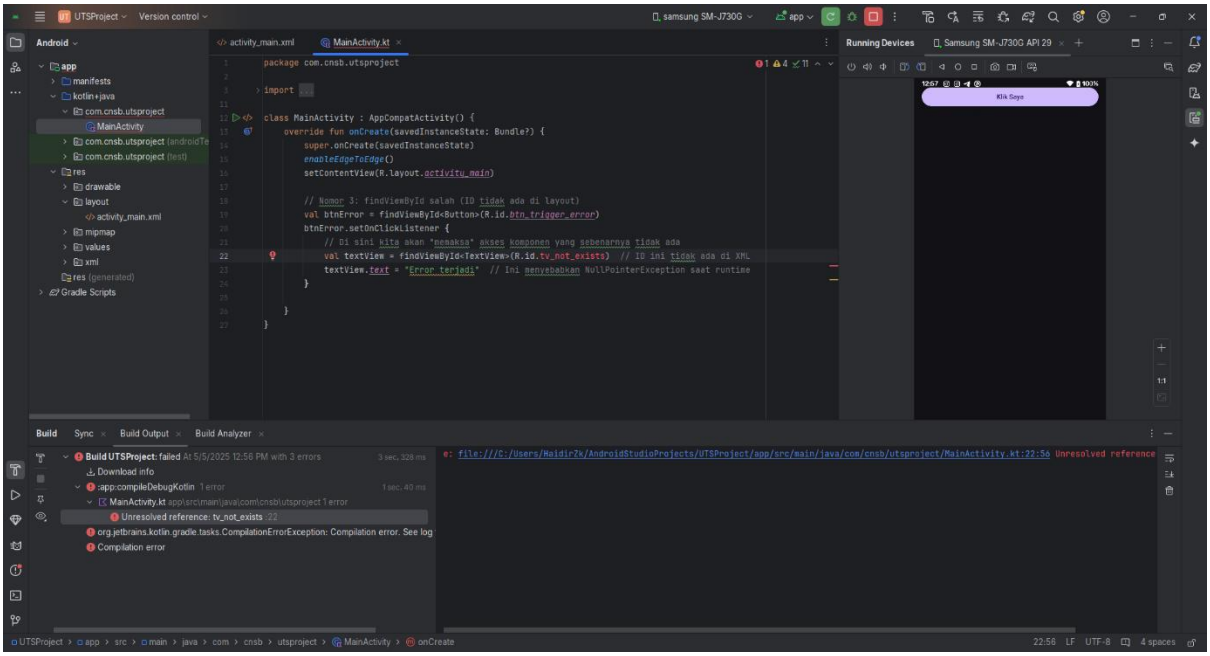
1. setOnClickListener adalah sebuah method di Android yang digunakan untuk menangani event klik pada komponen UI, seperti tombol (Button), gambar (ImageView), dan lainnya. Method ini memungkinkan developer menentukan aksi atau kode apa yang harus dijalankan ketika pengguna mengklik suatu elemen pada aplikasi.

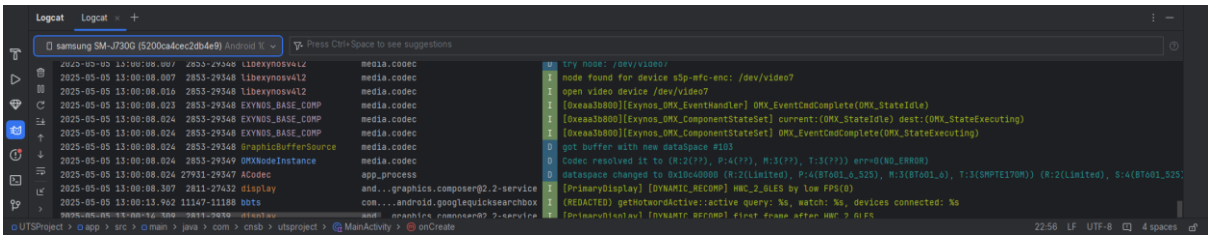
2. **Syarat:** Komponen harus ditemukan (findViewById) dan dipanggil setelah setContentView.



3. Error Umum :

   Jika ID yang dicari tidak ditemukan: java.lang.NullPointerException
   Jika menggunakan ID yang tidak ada di layout: Unresolved reference
   Jika komponen tidak ditemukan pada layout yang aktif: java.lang.NullPointerException: Attempt to invoke virtual method '...' on a null object reference

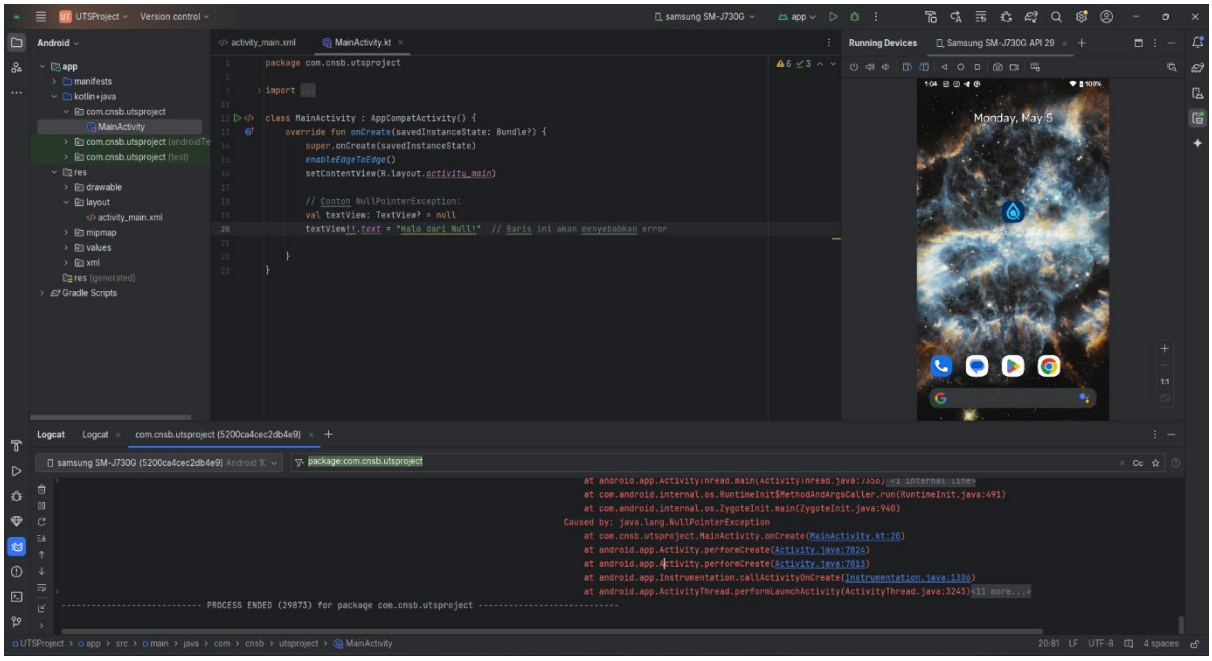




4. class MainActivity : AppCompatActivity() {
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      enableEdgeToEdge()
      setContentView(R.layout.*activity_main*)

      // Contoh NullPointerException:
      val textView: TextView? = null
      textView!!.*text* = "Halo dari Null!" // Baris ini akan menyebabkan error

```
    }
}
```



Studi Kasus

Penjelasan fungsi setiap baris source code pada file kotlin

1. SplashScreenActivity.kt

class SplashScreenActivity : AppCompatActivity() {
Mendeklarasikan kelas SplashScreenActivity yang merupakan turunan dari AppCompatActivity, sebuah kelas dasar untuk aktivitas Android yang kompatibel dengan berbagai versi Android.

private val splashTime: Long = 2500 // 2.5 detik
Mendeklarasikan variabel splashTime bertipe Long dengan nilai 2500 milidetik (2.5 detik). Ini digunakan sebagai durasi tampilan splash screen sebelum berpindah ke halaman login.

override fun onCreate(savedInstanceState: Bundle?) {

onCreate adalah metode siklus hidup (lifecycle) yang dipanggil saat aktivitas pertama kali dibuat. Kata kunci override menunjukkan bahwa metode ini mengoverride versi dari superclass (AppCompatActivity).

super.onCreate(savedInstanceState)
Memanggil implementasi onCreate dari superclass untuk memastikan bahwa inisialisasi standar Android tetap dilakukan.

setContentView(R.layout.activity_splash_screen)
Mengatur layout yang akan digunakan oleh aktivitas ini, yaitu activity_splash_screen.xml.

Handler(Looper.getMainLooper()).postDelayed({
Membuat penundaan (delay) menggunakan Handler. Looper.getMainLooper() memastikan bahwa kode dijalankan di thread UI utama.

startActivity(Intent(this, LoginActivity::class.java))
Setelah delay selesai, akan dibuat Intent untuk memulai LoginActivity, yaitu berpindah ke layar login.

finish()
Menutup SplashScreenActivity agar tidak bisa kembali ke layar ini saat pengguna menekan tombol "back".

}, splashTime)
Menentukan durasi delay berdasarkan nilai splashTime yang telah didefinisikan sebelumnya (2.5 detik).

} }
Menutup blok onCreate dan kelas SplashScreenActivity.

2. LoginActivity.kt

class LoginActivity : AppCompatActivity() {
Mendeklarasikan kelas LoginActivity yang mewarisi dari AppCompatActivity.

private lateinit var emailEditText: EditText private lateinit var passwordEditText: EditText private lateinit var loginButton: ImageButton private lateinit var registerButton: Button private lateinit var sharedPreferences: SharedPreferences
Mendeklarasikan variabel-variabel untuk elemen UI (EditText, Button) dan SharedPreferences (untuk menyimpan data login pengguna).

override fun onCreate(savedInstanceState: Bundle?) {
Fungsi onCreate dipanggil saat Activity dibuat.

super.onCreate(savedInstanceState)
Memanggil onCreate milik superclass dan mengatur tampilan layout dengan activity_login.xml

sharedPreferences = getSharedPreferences("UserData", MODE_PRIVATE)
Menginisialisasi SharedPreferences dengan nama file "UserData" dalam mode privat (hanya aplikasi ini yang bisa mengakses).

val isLoggedIn = sharedPreferences.getBoolean("isLoggedIn", false)
if (isLoggedIn) {
startActivity(Intent(this, ListChatActivity::class.java))
finish()
}

Mengecek apakah pengguna sudah login sebelumnya. Jika ya (isLoggedIn == true), langsung diarahkan ke ListChatActivity, dan LoginActivity ditutup.

emailEditText = findViewById(R.id.editTextEmail) passwordEditText = findViewById(R.id.editTextPassword) loginButton = findViewById(R.id.btnLogin)

registerButton = findViewById(R.id.btnRegister)
Menghubungkan variabel Kotlin dengan elemen UI di layout berdasarkan ID-nya..

loginButton.setOnClickListener {
Memberi aksi saat tombol login ditekan..

val emailInput = emailEditText.text.toString().trim() val passwordInput = passwordEditText.text.toString().trim()
Mengambil dan memproses input email dan password dari pengguna.

val savedEmail = sharedPreferences.getString("email", null) val savedPassword = sharedPreferences.getString("password", null)
Mengambil data email dan password yang tersimpan di SharedPreferences

if (emailInput == savedEmail && passwordInput == savedPassword) {
Mengecek apakah input pengguna sesuai dengan data yang tersimpan.

val editor = sharedPreferences.edit() editor.putBoolean("isLoggedIn", true) editor.apply()
Jika login berhasil, ubah status login menjadi true di SharedPreferences.

Toast.makeText(this, "Login berhasil", Toast.LENGTH_SHORT).show() startActivity(Intent(this, ListChatActivity::class.java)) finish()
Tampilkan pesan "Login berhasil", buka ListChatActivity, dan tutup LoginActivity

} else {

Toast.makeText(this, "Email atau password salah", Toast.LENGTH_SHORT).show() }
Jika login gagal, tampilkan pesan kesalahan.

registerButton.setOnClickListener {
startActivity(Intent(this, RegisterActivity::class.java))
}

Jika tombol register ditekan, buka RegisterActivity untuk membuat akun baru.

3. RegisterActivity.kt

class RegisterActivity : AppCompatActivity() {
Mendeklarasikan kelas RegisterActivity yang mewarisi dari AppCompatActivity.

private lateinit var namaEditText: EditText private lateinit var emailEditText: EditText private lateinit var passwordEditText: EditText private lateinit var registerButton: Button private lateinit var loginLink: TextView private lateinit var sharedPreferences: SharedPreferences
Mendeklarasikan variabel untuk elemen UI (EditText, Button, TextView) dan SharedPreferences

override fun onCreate(savedInstanceState: Bundle?) { super.onCreate(savedInstanceState) setContentView(R.layout.activity_register)
Fungsi `onCreate` dijalankan saat activity dibuat. `setContentView` menampilkan layout `activity_register.xml`

sharedPreferences = getSharedPreferences("UserData", MODE_PRIVATE)
Inisialisasi SharedPreferences untuk menyimpan data user secara lokal di perangkat.

```
        namaEditText = findViewById(R.id.editTextNama)
        emailEditText = findViewById(R.id.editTextEmail)
        passwordEditText = findViewById(R.id.editTextPassword)
        registerButton = findViewById(R.id.btnRegister)
    loginLink = findViewById(R.id.tvLoginLink)
```

Menghubungkan variabel Kotlin dengan elemen UI pada layout berdasarkan ID-nya

```
    registerButton.setOnClickListener {
```

Menangani aksi ketika tombol "Daftar" ditekan.

```
        val nama = namaEditText.text.toString().trim()
        val email = emailEditText.text.toString().trim()
    val password = passwordEditText.text.toString().trim()
```

Mengambil nilai input dari ketiga field dan menghapus spasi berlebih.

```
        if (nama.isEmpty() || email.isEmpty() || password.isEmpty()) {
        Toast.makeText(this, "Semua kolom wajib diisi",
Toast.LENGTH_SHORT).show()
```

Jika salah satu kolom kosong, tampilkan pesan peringatan bahwa semua kolom wajib diisi.

```
        } else {
          val editor = sharedPreferences.edit()
          editor.putString("email", email)
          editor.putString("password", password)
          editor.putBoolean("isLoggedIn", false)
        editor.apply()
```

Jika semua data terisi, simpan data ke SharedPreferences. Status login disetel ke false karena belum login.

```
          Toast.makeText(this, "Registrasi berhasil!",
      Toast.LENGTH_SHORT).show()
       Log.d("Reg", "reg success")
```

Tampilkan pesan sukses dan catat log di Logcat.

startActivity(Intent(this, LoginActivity::class.java)) finish()
Pindah ke halaman login setelah registrasi berhasil, dan tutup RegisterActivity.

loginLink.setOnClickListener {
startActivity(Intent(this, LoginActivity::class.java))
finish()
}

Jika teks "Sudah punya akun? Login" ditekan, kembali ke LoginActivity

4. ListChatActivity.kt

class ListChatActivity : AppCompatActivity() {
Mendeklarasikan kelas ListChatActivity yang merupakan turunan dari AppCompatActivity

private lateinit var recyclerView: RecyclerView private lateinit var chatAdapter: ChatAdapter private lateinit var logoutButton: Button private lateinit var sharedPreferences: SharedPreferences
Variabel untuk menampung view dan data: • recyclerView: komponen daftar chat. • chatAdapter: adapter untuk menampilkan data ke dalam RecyclerView. • logoutButton: (walau tidak dipakai dalam kode ini, kemungkinan disiapkan). • sharedPreferences: menyimpan dan mengelola data pengguna.

override fun onCreate(savedInstanceState: Bundle?) { super.onCreate(savedInstanceState) setContentView(R.layout.activity_list_chat)
Memanggil lifecycle method onCreate() dan menampilkan layout activity_list_chat

```
        val toolbar: Toolbar = findViewById(R.id.toolbar)
    setSupportActionBar(toolbar)
```

Menetapkan custom Toolbar sebagai action bar utama

```
        recyclerView = findViewById(R.id.recyclerViewChat)
    recyclerView.layoutManager = LinearLayoutManager(this)
```

Inisialisasi RecyclerView dan mengaturnya agar tampil secara vertikal menggunakan LinearLayoutManager.

```
        val dummyData = listOf(
            Chat("Grup D-CSRF", "Reminder: Meeting jam 19.00", "5m"),
            ...
            Chat("Bot Reminder", "Backup database kamu sekarang.", "18h")
    )
```

Data dummy (sementara) dalam bentuk daftar Chat, yang akan ditampilkan di daftar percakapan.

```
    sharedPreferences = getSharedPreferences("UserData", MODE_PRIVATE)
```

Mengakses SharedPreferences dengan nama "UserData".

```
        chatAdapter = ChatAdapter(dummyData)
    recyclerView.adapter = chatAdapter
```

Membuat instance ChatAdapter menggunakan data dummy dan menghubungkannya ke RecyclerView.

```
        override fun onCreateOptionsMenu(menu: Menu?): Boolean {
            Log.d("Menu", "Menu created")
            menuInflater.inflate(R.menu.menu_main, menu)
            return true
    }
```

- Fungsi ini memuat file menu menu_main.xml ke dalam toolbar.
- Digunakan untuk menampilkan menu titik tiga di pojok kanan atas.

override fun onOptionsItemSelected(item: MenuItem): Boolean { return when (item.itemId) {
Menentukan aksi saat item menu dipilih.

R.id.action_logout -> { val editor = sharedPreferences.edit() editor.clear() editor.apply()
Saat tombol logout ditekan: • Semua data SharedPreferences dihapus. • Pengguna diarahkan kembali ke LoginActivity.

startActivity(Intent(this, LoginActivity::class.java)) finish() true
Aktivitas login dijalankan dan ListChatActivity ditutup.

R.id.action_materi -> {
startActivity(Intent(this, MateriActivity::class.java))
true
}

Saat menu "Materi" dipilih, akan membuka MateriActivity.

else -> super.onOptionsItemSelected(item) } }
Menangani item menu lain yang tidak dikenali.

5. Chat.kt

data class Chat(val nama: String,

```
    val pesanTerakhir: String,
    val waktu: String
)
```

 data class Chat:
Mendefinisikan sebuah **data class** bernama Chat.
Data class di Kotlin digunakan untuk menyimpan data dan secara otomatis menghasilkan fungsi seperti:

- toString()
- equals()
- hashCode()
- copy()

 val nama: String:
Menyimpan nama pengirim atau nama grup chat.
 val pesanTerakhir: String:
Menyimpan isi pesan terakhir yang dikirim dalam chat tersebut.
 val waktu: String:
Menyimpan waktu saat pesan terakhir dikirim (misalnya: "5m", "1h", "2h").

6. ChatAdapter.kt

```
class ChatAdapter(private val listChat: List<Chat>) :
    RecyclerView.Adapter<ChatAdapter.ChatViewHolder>() {
```

 ChatAdapter adalah adapter khusus RecyclerView untuk menampilkan data bertipe Chat.
 listChat adalah daftar chat yang akan ditampilkan.
 ChatViewHolder adalah kelas internal yang mengatur tampilan item per baris.

```
class ChatViewHolder(view: View) : RecyclerView.ViewHolder(view) {
    val tvNama: TextView = view.findViewById(R.id.tvNamaPengirim)
    val tvPesan: TextView = view.findViewById(R.id.tvPesan)
    val tvWaktu: TextView = view.findViewById(R.id.tvWaktu)
    val imgProfile: ImageView = view.findViewById(R.id.imgProfile)
}
```

ChatViewHolder menyimpan referensi ke komponen UI di layout item_chat.xml:

- tvNama: nama pengirim atau grup.
- tvPesan: isi pesan terakhir.
- tvWaktu: waktu pengiriman.
- imgProfile: gambar profil/icon pengirim.

```
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
ChatViewHolder {
    val view = LayoutInflater.from(parent.context)
```

```
            .inflate(R.layout.item_chat, parent, false)
        return ChatViewHolder(view)
    }
```

- Dipanggil saat RecyclerView butuh baris tampilan baru.
- LayoutInflater digunakan untuk mengubah file XML (item_chat.xml) menjadi objek View.
- Hasilnya dibungkus ke dalam ChatViewHolder.

```
    override fun onBindViewHolder(holder: ChatViewHolder, position:
Int) {
        val chat = listChat[position]
        holder.tvNama.text = chat.nama
        holder.tvPesan.text = chat.pesanTerakhir
        holder.tvWaktu.text = chat.waktu
        holder.imgProfile.setImageResource(R.drawable.profile)
    }
```

- Mengisi data Chat ke dalam komponen UI di baris yang sesuai.
- position menunjukkan indeks item dalam list.
- Gambar profil di-set dengan drawable bernama profile.

```
    override fun getItemCount(): Int = listChat.size
```

Mengembalikan jumlah item yang ada di dalam list Chat.

7. MateriActivity.kt

```
class MateriActivity : AppCompatActivity() {
    private lateinit var recyclerView: RecyclerView
    private lateinit var materiAdapter: MateriAdapter
```

- MateriActivity adalah Activity yang menampilkan halaman daftar materi.
- recyclerView digunakan untuk menampilkan daftar materi.
- materiAdapter adalah adapter khusus yang akan menghubungkan data Materi ke RecyclerView.

```
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_materi)
```

- Fungsi onCreate() dipanggil saat activity pertama kali dibuat.
- setContentView() menampilkan layout activity_materi.xml.

```
        val toolbar: Toolbar = findViewById(R.id.toolbarMateri)
        setSupportActionBar(toolbar)
```

- Toolbar disiapkan agar halaman materi punya title bar.
- setSupportActionBar() menjadikan toolbarMateri sebagai ActionBar

```
        recyclerView = findViewById(R.id.recyclerViewMateri)
        recyclerView.layoutManager = GridLayoutManager(this, 1)
```

- recyclerViewMateri adalah komponen daftar.
- GridLayoutManager(this, 1) berarti tampilannya berupa list 1 kolom (bisa diubah jadi 2 kolom jika ingin grid).

```
        val dummyMateri = listOf(
            Materi(R.drawable.dkotlin, "Dasar Kotlin", "Belajar dasar bahasa
    Kotlin untuk Android"),
            ...
        )
```

- Daftar Materi yang berisi gambar (drawable), judul, dan subjudul.
- Data ini bersifat sementara (dummy) untuk ditampilkan ke layar.

```
        materiAdapter = MateriAdapter(dummyMateri)
        recyclerView.adapter = materiAdapter
    }
```

- MateriAdapter menerima daftar dummyMateri dan menampilkannya ke RecyclerView.

8. Materi.kt

```
    data class Materi(
        val imageResId: Int,
        val judul: String,
        val subjudul: String
    )
```

Kode ini mendefinisikan **data class** bernama Materi, yang merepresentasikan 1 item materi yang akan ditampilkan di dalam daftar materi (RecyclerView).

9. MateriAdapter.kt

```
    class MateriAdapter(private val listMateri: List<Materi>) :
        RecyclerView.Adapter<MateriAdapter.MateriViewHolder>()
```

- MateriAdapter adalah subclass dari RecyclerView.Adapter.

- listMateri adalah daftar data bertipe Materi yang akan ditampilkan.
- Menggunakan inner class MateriViewHolder untuk memegang referensi ke view.

```
class MateriViewHolder(view: View) : RecyclerView.ViewHolder(view) {
    val imgMateri: ImageView = view.findViewById(R.id.imgMateri)
    val tvJudul: TextView = view.findViewById(R.id.tvJudul)
    val tvSubjudul: TextView = view.findViewById(R.id.tvSubjudul)
}
```

- MateriViewHolder adalah penampung view dari 1 item Materi.
- Menyimpan referensi ke elemen-elemen di layout item_materi.xml:
  - imgMateri: gambar materi.
  - tvJudul: judul materi.
  - tvSubjudul: deskripsi singkat.

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
MateriViewHolder {
    val view = LayoutInflater.from(parent.context)
        .inflate(R.layout.item_materi, parent, false)
    return MateriViewHolder(view)
}
```

- Dipanggil ketika RecyclerView butuh membuat ViewHolder baru.
- Meng-inflate layout item_materi.xml sebagai tampilan item list.

```
override fun onBindViewHolder(holder: MateriViewHolder, position: Int)
{
    val materi = listMateri[position]
    holder.imgMateri.setImageResource(materi.imageResId)
    holder.tvJudul.text = materi.judul
    holder.tvSubjudul.text = materi.subjudul
}
```

- Mengikat (bind) data Materi ke tampilan item berdasarkan posisi.
- Mengisi gambar dan teks ke dalam ViewHolder.

```
override fun getItemCount(): Int = listMateri.size
```

- Mengembalikan jumlah item di daftar listMateri.
- Digunakan RecyclerView untuk tahu berapa banyak item yang ditampilkan.

Link GITHUB :

https://github.com/jck-cysec/UTS-PM1-HaidirZacky