UJIAN TENGAH SEMESTER

“PEMROGRAMAN MOBILE 1”

Dosen Pengampu: Nova Agustina, ST., M.Kom.

UNIVERSITAS
TEKNOLOGI BANDUNG

Dibuat Oleh:

Wildam Pramudiya Alif TIF RP 223PA (23552011235)

PROGRAM STUDI TEKNIK INFORMATIKA

UNIVERSITAS TEKNOLOGI BANDUNG

2025

Essay

1. Apa fungsi findViewById?
2. Apa syarat pemanggilan method findViewById? Buat contohnya dan screenshot source code nya!
3. Error apa yang terjadi jika file kotlin salah menginisialisasi findViewById atau objek pada xml belum diinisialisasi?
4. Buat sebuah contoh program untuk menampilkan pesan error Resources.NotFoundException! Screenshot logcat-nya!

Jawab:

1. findViewById berfungsi untuk mengambil (mengakses) komponen UI dari file layout XML ke dalam kode program, agar bisa memberikan aksi terhadap komponen tersebut melalui kode.
2. Syarat pemanggilan method findViewById:
   - Layout harus sudah dimuat dengan setContentView()
   - ID komponen harus ada dan benar
   - Pemanggilan dilakukan di dalam konteks yang sesuai
   - Tipe data harus sesuai dengan komponen

```
<ImageView
    android:id="@+id/imageLogo"
    android:layout_width="match_parent"
    android:layout_height="180dp"
    android:src="@drawable/logo2"
    android:layout_alignParentTop="true"
    android:scaleType="centerCrop" />
```

Gambar di atas merupakah komponen XML yang sudah sesuai syarat, yaitu mempunyai id

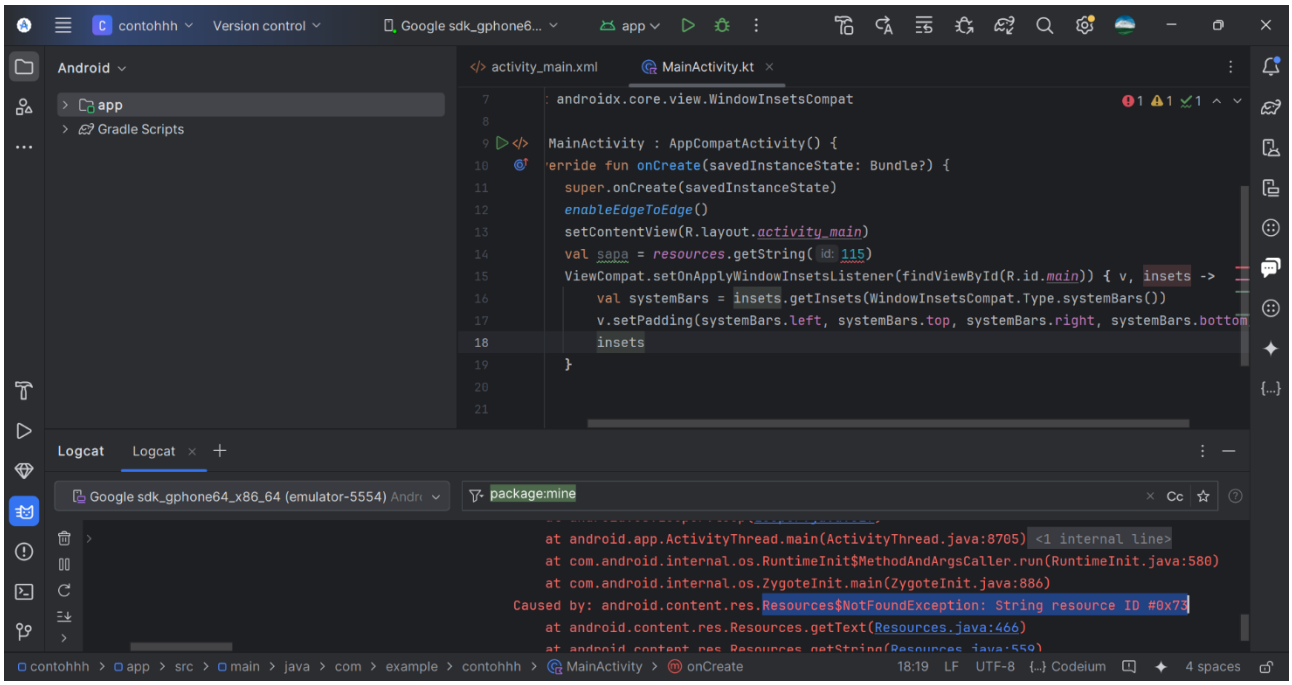
```
class SplashScreenActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_splash)
        val logo = findViewById<ImageView>(R.id.logoImage)
        logo.setImageResource(R.drawable.logo2)
```

Gambar di atas merupakan contoh pemanggilan image view di file Kotlin

3. Error jika file kotlin salah menginisialisasi findViewById

- **NullPointerException:** Disebabkan jika ID tidak ditemukan, tapi objek tetap digunakan
- **Resources.NotFoundException:** Disebabkan jika Layout XML yang dimaksud tidak ada / salah
- **ClassCastException:** Disebabkan jika View yang ditemukan tidak cocok dengan tipe yang di-cast

4. Contoh program untuk menampilkan pesan error Resources.NotFoundException



Penjelasan Source Code Pada Studi Kasus:

1. SplaceScreenActivity.kt

```
class SplashScreenActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_splash)
        val logo = findViewById<ImageView>(R.id.logoImage)

        Handler(Looper.getMainLooper()).postDelayed({
            val intent = Intent( packageContext: this, MainActivity::class.java)
            startActivity(intent)
            finish()
        }, delayMillis: 3000)
    }
}
```

Menampilkan logo untuk splacescreen yang telah dibuat pada file xml:

- *setContentView(R.layout.activity_splash):* Yaitu code untuk menetapkan tampilan/layout dari activity dengan file XML bernama activity_splash.xml.
- *val logo = findViewById<ImageView>(R.id.logoImage):* Yaitu code untuk Mencari komponen ImageView di dalam layout (dengan ID logoImage) dan menyimpannya ke dalam variabel logo. Karena logo/seluruh tampilan untuk splacescreennya sudah termasuk di dalam imageview, maka tidak harus memanggil logonya lagi untuk tampil pada file Kotlin ini.
- *Handler(Looper.getMainLooper()).postDelayed({ ... }, 3000):* Yaitu kode untuk *Handler(Looper.getMainLooper()):* Membuat handler yang berjalan di UI thread (main thread) dan *postDelayed({ ... }, 3000):* Menjalankan blok kode setelah delay 3000 milidetik (3 detik), sehingga setelah 3 detik imageview tampil, kemudian akan beralih ke halaman selanjutnya.
- *val intent = Intent(this, MainActivity::class.java):* Yaitu kode untuk Membuat Intent agar berpindah dari SplashScreenActivity ke MainActivity.
- *startActivity(intent):* Yaitu kode untuk menjalankan MainActivity
- *finish():* Yaitu kode untuk Menutup SplashScreenActivity agar tidak bisa dikembalikan dengan tombol back.

2. MainActivity.kt(halaman ke2)

```kotlin
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val emailEditText = findViewById<EditText>(R.id.email)
        val passwordEditText = findViewById<EditText>(R.id.password)
        val loginButton = findViewById<Button>(R.id.login_button)
        val daftarSekarang = findViewById<TextView>(R.id.tvDaftarSekarang)

        loginButton.setOnClickListener {
            val inputEmail = emailEditText.text.toString().trim()
            val inputPassword = passwordEditText.text.toString().trim()

            val sharedPref = getSharedPreferences( name: "user_pref", MODE_PRIVATE)
            val savedEmail = sharedPref.getString( key: "email",  defValue: null)
            val savedPassword = sharedPref.getString( key: "password",  defValue: null)


            if (inputEmail == savedEmail && inputPassword == savedPassword) {

                val intent = Intent( packageContext: this, DashboardActivity::class.java)
                intent.putExtra( name: "email", savedEmail)
                startActivity(intent)
                finish()
            } else {
                Toast.makeText( context: this,  text: "Email atau Password salah!", Toast.LENGTH_SHORT).show()
            }
        }

        daftarSekarang.setOnClickListener {
            val intent = Intent( packageContext: this, RegisterActivity::class.java)
            startActivity(intent)
        }
    }
}
```

Implementasi Login Sederhana Menggunakan SharedPreferences di Android:

- *setContentView(R.layout.activity_main):* untuk mengatur tampilan layout activity dari file XML activity_main.xml.
- *val emailEditText = findViewById<EditText>(R.id.email)*
  val passwordEditText = findViewById<EditText>(R.id.password)
  val loginButton = findViewById<Button>(R.id.login_button)
  val daftarSekarang = findViewById<TextView>(R.id.tvDaftarSekarang)
  Ke empat kode diatas digunakan untuk menghubungkan komponen-komponen di XML ke kode Kotlin:
  - emailEditText: kolom input email.
  - passwordEditText: kolom input password.
  - loginButton: tombol login.
  - daftarSekarang: teks untuk pindah ke halaman daftar.
- *loginButton.setOnClickListener {:* Digunakan untuk menangani event saat tombol login diklik.
- *val inputEmail = emailEditText.text.toString().trim()*
  val inputPassword = passwordEditText.text.toString().trim()
  Mengambil teks dari input dan menghapus spasi di awal/akhir:
  - inputEmail: email yang diketik user.
  - inputPassword: password yang diketik user.

- *val sharedPref = getSharedPreferences("user_pref", MODE_PRIVATE):* Mengakses data yang disimpan di SharedPreferences bernama user_pref yang digunakan untuk mengambil email dan password yang disimpan saat registrasi.
- *val savedEmail = sharedPref.getString("email", null)*
  val savedPassword = sharedPref.getString("password", null)
  Kode di atas digunakan untuk mengambil data email dan password yang disimpan sebelumnya. Jika belum disimpan, nilainya akan null.
- *if (inputEmail == savedEmail && inputPassword == savedPassword) {:* Digunakan untuk mengecek apakah email dan password yang diinput cocok dengan yang tersimpan.
- *val intent = Intent(this, DashboardActivity::class.java)*
  intent.putExtra("email", savedEmail)
  startActivity(intent)
  finish()
  Digunakan untuk jika cocok:
  - Buat Intent untuk pindah ke DashboardActivity.
  - Kirim email ke activity tujuan lewat putExtra.
  - Panggil startActivity untuk membuka dashboard.
  - Gunakan finish() agar user tidak bisa kembali ke login dengan tombol back.

- *Toast.makeText(this, "Email atau Password salah!", Toast.LENGTH_SHORT).show():*
  Berfungsi Jika email atau password salah maka akan menampilkan pesan peringatan (Toast).
- *daftarSekarang.setOnClickListener {*
  val intent = Intent(this, RegisterActivity::class.java)
  startActivity(intent)}
  Berfungsi Jika pengguna mengklik “Daftar Sekarang”, maka akan membuat Intent untuk membuka halaman RegisterActivity dan menjalankan activity tersebut.

3.RegisterAktivity.kt

```
val emailEditText = findViewById<EditText>(R.id.email)
val passwordEditText = findViewById<EditText>(R.id.password)
val loginButton = findViewById<Button>(R.id.login_button)
val loginNowText = findViewById<TextView>(R.id.tvDaftarSekarang)
```

Ini merupakan Inisialisasi Komponen yang Menghubungkan elemen UI dari XML ke dalam kode input email, input password, tombol "Daftar", TextView "Sudah punya akun? dan Login sekarang"

```
loginButton.setOnClickListener {
    val email = emailEditText.text.toString().trim()
    val password = passwordEditText.text.toString().trim()
```

Saat tombol daftar ditekan, maka kode ini akan mengambil input dari user.

```
if (email.isEmpty() || password.isEmpty()) {
    Toast.makeText( context: this,  text: "Semua data harus diisi!", Toast.LENGTH_SHORT).show()
} else {
```

Melakukan validasi jika email atau password kosong, tampilkan peringatan.

```
val sharedPref = getSharedPreferences( name: "user_pref", MODE_PRIVATE)
val editor = sharedPref.edit()
editor.putString("email", email)
editor.putString("password", password)
editor.apply()
```

Jika sudah terpenuhi semuanya maka kode ini akan menyimpan input email dan password ke penyimpanan lokal agar bisa dipakai saat login nanti.

```
Toast.makeText( context: this,  text: "Akun berhasil dibuat!", Toast.LENGTH_SHORT).show()
val intent = Intent( packageContext: this, MainActivity::class.java)
startActivity(intent)
finish()
```

Berfungsi untuk setelah selesai melakukan penginputan, maka saat menekan tombol daftar akan menampilkan sebuah toast, dan akan dilanjutkan ke halaman login dan menutup halaman register.

```
loginNowText.setOnClickListener {
    startActivity(Intent( packageContext: this, MainActivity::class.java))
    finish()
```

Berfungsi saat user menekan tombol daftar sekarang yang dimana akan langsung menuju ke halaman register, dan si user baru sadar bahwa dia sudah memiliki akun, maka untuk Kembali ke halaman login cukup tekan “Login Sekarang”.

4. DashboardActivity.kt

```
val recyclerView = findViewById<RecyclerView>(R.id.recycler_view)
recyclerView.layoutManager = GridLayoutManager( context: this,  spanCount: 1)
recyclerView.setHasFixedSize(true)
```

Digunakan untuk menyiapkan RecyclerView yang merupakan komponen untuk menampilkan list data. GridLayoutManager(this, 1) artinya list ditampilkan dalam 1 kolom (seperti list biasa). setHasFixedSize(true) digunakan jika ukuran item tidak berubah, supaya performa lebih baik.

```
val itemList = listOf(
    ItemList( judul: "Gear Set", subjudul: " komponen penggerak rantai pada sepeda motor yang terdiri
            "\n1. Gear depan (sprocket depan): terhubung ke mesin." +
            "\n2. Gear belakang (sprocket belakang): terhubung ke roda belakang." +
            "\n3. Rantai: menghubungkan gear depan dan belakang untuk menyalurkan tenaga dari mesi
        R.drawable.gearset),
    ItemList( judul: "Komponen Stir(KomStir)", subjudul: "Bagian dari sistem kemudi pada sepeda motor
        R.drawable.komstir),
    ItemList( judul: "Roller", subjudul: "komponen penting pada motor matic yang berada di dalam CVT (
        R.drawable.roller)
)
```

Berfungsi untuk menampilkan list gambar dan keterangannya dengan mengambil gambar dari drawable

```
val adapter = AdapterList(itemList)
recyclerView.adapter = adapter
```

Berfungsi membuat adapter untuk menghubungkan data (itemList) ke tampilan RecyclerView dan menetapkan adapter agar datanya ditampilkan.

5. ItemList.kt

```
class ItemList (
    var judul: String,
    var subjudul: String,
    var imageUrl: Int
)
```

Berfungsi sebagai blueprint untuk item yang akan ditampilkan di RecyclerView dan akan digunakan di DashboardActivity sebagai struktur data untuk setiap item yang ditampilkan dalam daftar. Setiap objek ItemList merepresentasikan satu komponen motor dengan nama komponen, Penjelasan/deskripsi, dan Gambar.

6. AdapterList

```
class AdapterList(private val itemList: kotlin.collections.List<ItemList>)
```

Berfungsi untuk Menerima data berupa list dari objek ItemList, yaitu data yang akan ditampilkan di setiap baris.

```
val view = LayoutInflater.from(parent.context).inflate(R.layout.item_data, parent, attachToRoot: false)
return ViewHolder(view)
```

Method ini digunakan untuk **membuat tampilan (View)** dari setiap item, menggunakan LayoutInflater untuk mengubah file XML item_data.xml menjadi objek View, kemudian dibungkus dalam ViewHolder.

```
val item = itemList[position]
holder.judul.text = item.judul
holder.subJudul.text = item.subjudul
Glide.with(holder.imageView.context).load(item.imageUrl).into(holder.imageView)
```

Method ini bertugas mengisi data ke tampilan berdasarkan posisi (position) dalam list, dan Glide.with(...).load(...).into(...) digunakan untuk menampilkan gambar.

```
class ViewHolder (@NonNull itemView: View) : RecyclerView.ViewHolder(itemView) {
    val imageView: ImageView = itemView.findViewById(R.id.item_image)
    val judul: TextView = itemView.findViewById(R.id.title)
    val subJudul: TextView = itemView.findViewById(R.id.sub_title)
}
```

ViewHolder berfungsi untuk **mereferensikan komponen UI** dalam layout item_data.xml, dan ini dilakukan agar proses scroll lebih efisien tanpa harus terus-menerus mencari view.

```
override fun getItemCount(): Int {
    return itemList.size
```

Berfungsi untuk memberitahu RecyclerView berapa banyak item yang akan ditampilkan.