

# **LAPORAN UJIAN TENGAH SEMESTER**

## **PEMOGRAMAN MOBILE 1**

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



Disusun Oleh:

Luki Solihin

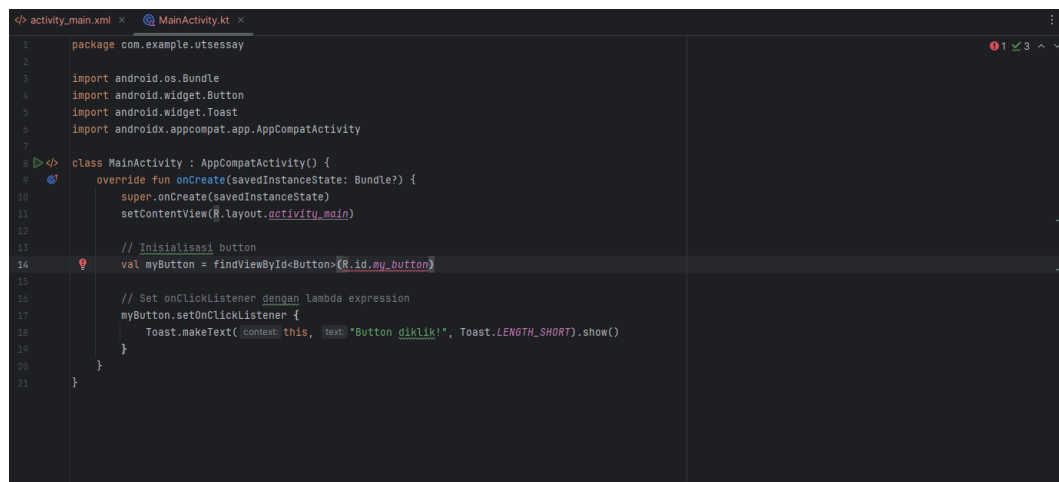
TIF-RP 223 CNS B

23552011413

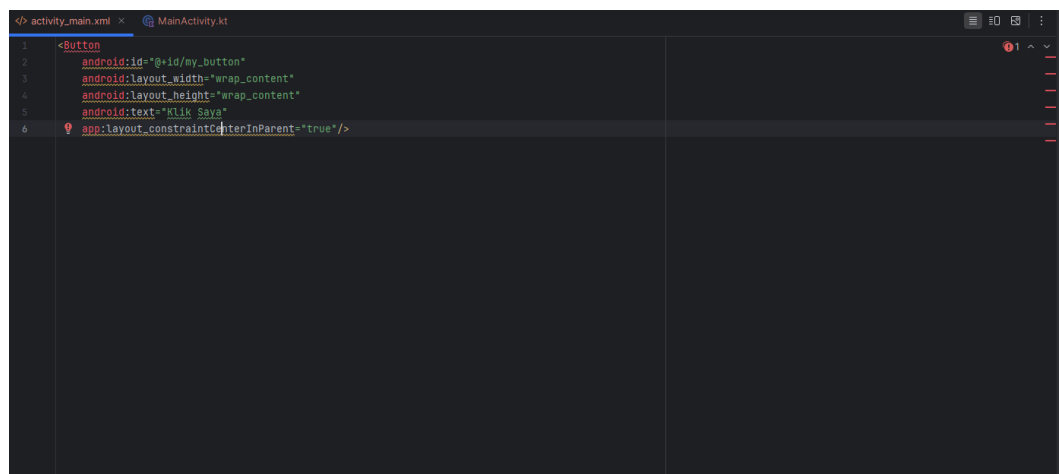
**DEPARTEMEN TEKNIK INFORMATIKA**  
**UNIVERSITAS TEKNOLOGI BANDUNG**

## Essay

1. Apa fungsi `setOnClickListener`?  
method yang digunakan untuk menangani event klik/tap pada sebuah View
2. Apa syarat pemanggilan method `setOnClickListener`? Buat contohnya dan screenshot source code nya!
  - View yang akan diklik harus sudah diinisialisasi dengan `findViewById()` atau View Binding
  - View tersebut harus ada di layout XML yang sedang digunakan
  - Harus mengimplementasikan interface `View.OnClickListener` atau menggunakan lambda expression



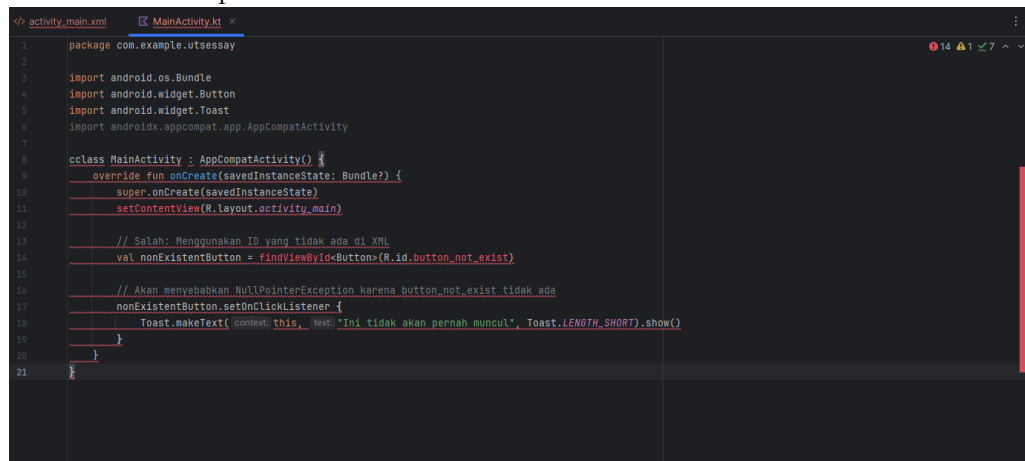
```
<? activity_main.xml x @ MainActivity.kt x
1 package com.example.utsessay
2
3 import android.os.Bundle
4 import android.widget.Button
5 import android.widget.Toast
6 import androidx.appcompat.app.AppCompatActivity
7
8 class MainActivity : AppCompatActivity() {
9     override fun onCreate(savedInstanceState: Bundle?) {
10         super.onCreate(savedInstanceState)
11         setContentView(R.layout.activity_main)
12
13         // Inisialisasi button
14         val myButton = findViewById<Button>(R.id.my_button)
15
16         // Set onClickListener dengan lambda expression
17         myButton.setOnClickListener {
18             Toast.makeText(context=this, text="Button diklik!", Toast.LENGTH_SHORT).show()
19         }
20     }
21 }
```



```
<? activity_main.xml x @ MainActivity.kt
1 <Button
2     android:id="@+id/my_button"
3     android:layout_width="wrap_content"
4     android:layout_height="wrap_content"
5     android:text="Klik Saya"
6     app:layout_constraintBottomInParent="true"/>
```

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`!



```
1 package com.example.utsessay
2
3 import android.os.Bundle
4 import android.widget.Button
5 import android.widget.Toast
6 import androidx.appcompat.app.AppCompatActivity
7
8 class MainActivity : AppCompatActivity() {
9     override fun onCreate(savedInstanceState: Bundle?) {
10         super.onCreate(savedInstanceState)
11         setContentView(R.layout.activity_main)
12
13         // Salah: Menggunakan ID yang tidak ada di XML
14         val nonExistentButton = findViewById<Button>(R.id.button_not_exist)
15
16         // Akan menyebabkan NullPointerException karena button_not_exist tidak ada
17         nonExistentButton.setOnClickListener {
18             Toast.makeText(context, this, "Ini tidak akan pernah muncul", Toast.LENGTH_SHORT).show()
19         }
20     }
21 }
```

5. Kumpulkan dalam bentuk pdf di Elearning (Soal essay digabung dengan soal studi kasus cek point 7 Studi Kasus)

## Studi Kasus

### ChatAdapter.kt

```
class ChatAdapter(private val chatList: List<ChatModel>) :  
RecyclerView.Adapter<ChatAdapter.ChatViewHolder>()
```

- ChatAdapter adalah kelas adapter yang mengatur bagaimana data ditampilkan dalam RecyclerView
- Menerima parameter chatList (List dari ChatModel) yang berisi data chat yang akan ditampilkan
- Meng-extend RecyclerView.Adapter dengan generic type ChatViewHolder

```
class ChatViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  
    val profileImage: ImageView = itemView.findViewById(R.id.profileImage)  
    val nameText: TextView = itemView.findViewById(R.id.nameText)  
    val lastMessageText: TextView = itemView.findViewById(R.id.lastMessageText)  
    val timeText: TextView = itemView.findViewById(R.id.timeText)  
}
```

- ChatViewHolder adalah kelas yang menyimpan referensi ke view-view dalam satu item chat
- Mendeklarasikan 4 komponen UI:

- `profileImage`: `ImageView` untuk foto profil
- `nameText`: `TextView` untuk nama pengirim
- `lastMessageText`: `TextView` untuk pesan terakhir
- `timeText`: `TextView` untuk waktu pesan

```
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 ketika `RecyclerView` membutuhkan `ViewHolder` baru
- Meng-inflate layout item chat (`R.layout.item_chat`) menjadi `View`
- Mengembalikan instance baru dari `ChatViewHolder` dengan `view` yang sudah di-inflate

```
override fun onBindViewHolder(holder: ChatViewHolder, position: Int) {
    val chat = chatList[position]
    holder.profileImage.setImageResource(chat.profileImage)
    holder.nameText.text = chat.name
    holder.lastMessageText.text = chat.lastMessage
    holder.timeText.text = chat.time
}
```

- Dipanggil untuk menampilkan data pada posisi tertentu
- Mengambil data chat dari `chatList` berdasarkan posisi
- Mengisi komponen UI `ViewHolder` dengan data dari model:
  - Mengatur gambar profil
  - Mengatur teks nama
  - Mengatur teks pesan terakhir
  - Mengatur teks waktu

```
override fun getItemCount() = chatList.size
```

- Mengembalikan jumlah total item dalam data set

- Dalam hal ini, mengembalikan ukuran dari chatList

```
data class ChatModel(
    val profileImage: Int, // Resource ID gambar
    val name: String,      // Nama pengirim
    val lastMessage: String, // Isi pesan terakhir
    val time: String       // Waktu pesan
)
```

- RecyclerView meminta jumlah item melalui getItemCount()
- Untuk setiap item yang perlu ditampilkan, RecyclerView memanggil onCreateViewHolder() untuk membuat ViewHolder
- Kemudian memanggil onBindViewHolder() untuk mengisi data ke ViewHolder tersebut
- Proses ini diulang untuk menampilkan semua item dalam daftar chat

### **ChatListActivity.kt**

```
package com.example.myapp
```

```
import androidx.appcompat.app.AppCompatActivity
```

```
import android.os.Bundle
```

```
import androidx.recyclerview.widget.LinearLayoutManager
```

```
import androidx.recyclerview.widget.RecyclerView
```

- Activity ini berada dalam package com.example.myapp
- Mengimpor kelas-kelas yang diperlukan:
  - AppCompatActivity sebagai base class untuk Activity
  - Bundle untuk menyimpan state
  - Komponen RecyclerView dan LayoutManager

```
class ChatListActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

```
setContentView(R.layout.activity_chat_list)
```

- Meng-extend AppCompatActivity untuk kompatibilitas dengan versi Android lama
- onCreate() adalah lifecycle method yang dipanggil saat Activity dibuat
- setContentView() mengatur layout UI dari file activity\_chat\_list.xml

```
val chatRecyclerView = findViewById<RecyclerView>(R.id.chatRecyclerView)
```

```
chatRecyclerView.layoutManager = LinearLayoutManager(this)
```

- Mengambil referensi RecyclerView dari layout menggunakan findViewById()
- Mengatur LinearLayoutManager yang akan mengatur item-item dalam bentuk daftar vertikal

```
val chatList = listOf(
```

```
    ChatModel("John Doe", "Hello there!", "10:30 AM", R.drawable.profile_placeholder),
```

```
    ChatModel("Jane Smith", "How are you doing?", "Yesterday",  
R.drawable.profile_placeholder),
```

```
    ChatModel("Mike Johnson", "Meeting at 3 PM", "Yesterday",  
R.drawable.profile_placeholder),
```

```
    ChatModel("Sarah Williams", "Please review the documents", "Monday",  
R.drawable.profile_placeholder),
```

```
    ChatModel("David Brown", "The project is completed", "Sunday",  
R.drawable.profile_placeholder)
```

```
)
```

- Membuat list berisi 5 item chat menggunakan ChatModel
- Setiap ChatModel berisi:
  - Nama kontak
  - Pesan terakhir
  - Waktu pesan
  - Gambar profil (menggunakan resource drawable yang sama untuk semua)

```
chatRecyclerView.adapter = ChatAdapter(chatList)
```

- Membuat instance ChatAdapter dengan data chatList
- Mengatur adapter ke RecyclerView untuk menampilkan data

## ChatModel.kt

```
data class ChatModel(  
    val name: String,  
    val lastMessage: String,  
    val time: String,  
    val profileImage: Int  
)
```

- data class adalah jenis khusus kelas dalam Kotlin yang secara otomatis menghasilkan fungsi-fungsi standar seperti:
  - equals()/hashCode()
  - toString() dalam format readable
  - copy() untuk membuat salinan objek
  - componentN() functions untuk destructuring declarations

name: String

- Menyimpan nama kontak/pengirim chat
- Bertipe String (teks)

lastMessage: String

- Menyimpan teks pesan terakhir dalam percakapan
- Bertipe String (teks)

time: String

- Menyimpan waktu/waktu relatif dari pesan terakhir
- Bertipe String (teks)

profileImage: Int

- Menyimpan referensi ke resource gambar profil
- Bertipe Int karena menyimpan ID resource drawable

## LoginActivity.kt

```
class LoginActivity : AppCompatActivity() {
```

```
override fun onCreate(savedInstanceState: Bundle?) {
```

```
    super.onCreate(savedInstanceState)
```

```
    setContentView(R.layout.activity_login)
```

- LoginActivity meng-extend AppCompatActivity untuk kompatibilitas dengan versi Android lama
- onCreate() adalah lifecycle method yang dipanggil saat Activity dibuat
- setContentView() mengatur layout UI dari file activity\_login.xml

```
val loginButton = findViewById<Button>(R.id.loginButton)
```

```
val registerText = findViewById<TextView>(R.id.registerText)
```

- Mengambil referensi ke:
  - Tombol login (loginButton) dari layout
  - Teks register (registerText) dari layout
- Menggunakan generic type (<Button> dan <TextView>) untuk type safety

```
loginButton.setOnClickListener {
```

```
    Log.d("LoginActivity", "Login button clicked")
```

```
    Toast.makeText(this, "Login button clicked", Toast.LENGTH_SHORT).show()
```

```
    // Navigate to Chat List
```

```
    val intent = Intent(this, ChatListActivity::class.java)
```

```
    startActivity(intent)
```

```
}
```

- Menambahkan click listener ke tombol login
- Saat diklik:
  1. Mencatat log dengan tag "LoginActivity"
  2. Menampilkan Toast pendek
  3. Membuat Intent untuk navigasi ke ChatListActivity
  4. Memulai Activity baru dengan startActivity()

```
registerText.setOnClickListener {
```



```

val intent = Intent(this, RegisterActivity::class.java)

startActivity(intent)
}

```

- Menambahkan click listener ke teks register
- Saat diklik:
  1. Membuat Intent untuk navigasi ke RegisterActivity
  2. Memulai Activity baru

## RegisterActivity.kt

```

class RegisterActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_register)
    }
}

```

- AppCompatActivity: Digunakan untuk kompatibilitas dengan versi Android lama
- onCreate(): Lifecycle method utama yang dijalankan saat Activity dibuat
- setContentView(): Menghubungkan dengan layout XML (activity\_register.xml)

```

val registerButton = findViewById<Button>(R.id.registerButton)

```

```

val loginText = findViewById<TextView>(R.id.loginText)

```

- Menggunakan findViewById dengan generic type untuk type safety
- Mendapatkan referensi ke:
  - Tombol register (registerButton)
  - Teks login (loginText)

```

registerButton.setOnClickListener {
    Log.d("RegisterActivity", "Register button clicked")
    Toast.makeText(this, "Registration successful!", Toast.LENGTH_SHORT).show()
}

```

```

val intent = Intent(this, LoginActivity::class.java)

startActivity(intent)

```

```
finish()
}
```

- Log.d(): Mencatat event klik untuk debugging
- Toast: Memberikan feedback visual singkat ke user
- Intent Navigation:
  - Membuat intent ke LoginActivity
  - startActivity() memulai Activity baru
  - finish() menutup Activity saat ini (mencegah kembali ke halaman register dengan back button)

```
loginText.setOnClickListener {
    val intent = Intent(this, LoginActivity::class.java)
    startActivity(intent)
    finish()
}
```

- Navigasi langsung ke LoginActivity
- Juga memanggil finish() untuk membersihkan back stack

## **SplashActivity.kt**

```
class SplashScreenActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_splash)
```

- AppCompatActivity: Basis kompatibel untuk semua Activity
- onCreate(): Titik masuk utama saat Activity dibuat
- setContentView(): Menghubungkan dengan layout splash screen (activity\_splash.xml)

```
Handler(Looper.getMainLooper()).postDelayed({
    val intent = Intent(this, LoginActivity::class.java)
    startActivity(intent)
```

```
finish()  
}, 3000)
```

Komponen Penting:

1. Handler + Looper:
  - Handler: Untuk menjadwalkan tugas
  - `Looper.getMainLooper()`: Memastikan kode berjalan di thread UI utama
2. `postDelayed()`:
  - Menunda eksekusi kode selama 3000ms (3 detik)
  - Parameter: Runnable (lambda) dan delay waktu dalam milidetik
3. Navigasi:
  - Membuat Intent ke LoginActivity
  - `startActivity()` memulai Activity baru
  - `finish()` menutup SplashScreen (mencegah kembali dengan back button)