LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 4

VIEWMODEL AND DEBUGGING

Oleh:

Andra Braputra Akbar Saleh NIM. 2310817210001

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
MEI 2025

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN I
MODUL 4

Laporan Praktikum Pemrograman Mobile Modul 4: ViewModel and Debugging ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman Mobile. Laporan Prakitkum ini dikerjakan oleh:

Nama Praktikan : Andra Braputra Akbar Saleh
NIM : 2310817210001

Menyetujui, Asisten Praktikum	Mengetahui, Dosen Penanggung Jawab Praktikum
Salsabila Syifa NIM. 2010817320004	Andreyan Rizky Baskara, S.Kom., M.Kom. NIP. 19930703 201903 01 011

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

SOAL 1

XML

Soal Praktikum:

1. Lanjutkan aplikasi Android berbasis XML dan Jetpack Compose yang sudah dibuat pada Modul 3 dengan menambahkan modifikasi sesuai ketentuan berikut:

   a. Buatlah sebuah ViewModel untuk menyimpan dan mengelola data dari list item. Data tidak boleh disimpan langsung di dalam Fragment atau Activity.

   b. Gunakan ViewModelFactory untuk membuat parameter dengan tipe data String di dalam ViewModel

   c. Gunakan StateFlow untuk mengelola event onClick dan data list item dari ViewModel ke Fragment

   d. Install dan gunakan library Timber untuk logging event berikut:

      a. Log saat data item masuk ke dalam list

      b. Log saat tombol Detail dan tombol Explicit Intent ditekan

      c. Log data dari list yang dipilih ketika berpindah ke halaman Detail

   e. Gunakan tool Debugger di Android Studio untuk melakukan debugging pada aplikasi. Cari setidaknya satu breakpoint yang relevan dengan aplikasi. Lalu, gunakan fitur Step Into, Step Over, dan Step Out. Setelah itu, jelaskan fungsi Debugger, cara menggunakan Debugger, serta fitur Step Into, Step Over, dan Step Out

A. Source Code

MainActivity.kt

Tabel 1. MainActivity.kt

```
1   package com.example.scrollablexml
2
3   import android.os.Bundle
4   import androidx.appcompat.app.AppCompatActivity
5   import androidx.lifecycle.ViewModelProvider
6   import com.example.scrollablexml.databinding.ActivityMainBinding
7
8   class MainActivity : AppCompatActivity() {
9
10      private lateinit var binding: ActivityMainBinding
11
```

```
    private lateinit var viewModel: FalloutViewModel

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        val factory = FalloutViewModelFactory()
        viewModel = ViewModelProvider(this, factory)[FalloutViewModel::class.java]

        val fragmentManager = supportFragmentManager
        val fragmentTag = MenuFragment::class.java.simpleName

        val existingFragment = fragmentManager.findFragmentByTag(fragmentTag)
        if (existingFragment == null) {
            fragmentManager.beginTransaction()
                .add(binding.fragmentContainer.id, MenuFragment(), fragmentTag)
                .commit()
        }
    }
}
```

activity_main.xml

Tabel 2. Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <FrameLayout
        android:id="@+id/fragment_container"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

MenuFragment.kt

Tabel 3. MenuFragment.kt

```
1   package com.example.scrollablexml
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.os.Bundle
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.ViewModelProvider
11  import androidx.recyclerview.widget.LinearLayoutManager
12  import com.example.scrollablexml.databinding.FragmentMenuBinding
13
14  class MenuFragment : Fragment() {
15
16      private var _binding: FragmentMenuBinding? = null
17      private val binding get() = _binding!!
18
19      private lateinit var viewModel: FalloutViewModel
20
21      override fun onCreateView(
22          inflater: LayoutInflater, container: ViewGroup?,
23          savedInstanceState: Bundle?
24      ): View {
25          _binding = FragmentMenuBinding.inflate(inflater, container, false)
26          return binding.root
27      }
28
29      override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
30          super.onViewCreated(view, savedInstanceState)
31
32          viewModel          =          ViewModelProvider(requireActivity(),
    FalloutViewModelFactory())[FalloutViewModel::class.java]

          if (viewModel.falloutList.value.isNullOrEmpty()) {
              viewModel.setFalloutList(
                  listOf(
                      FalloutItem(1,            getString(R.string.judul1),
    getString(R.string.year1),                getString(R.string.deskripsi1),
    getString(R.string.wiki1), getString(R.string.detail1), R.drawable.img1),
                      FalloutItem(2,            getString(R.string.judul2),
    getString(R.string.year2),                getString(R.string.deskripsi2),
    getString(R.string.wiki2), getString(R.string.detail2), R.drawable.img2),
                      FalloutItem(3,            getString(R.string.judul3),
    getString(R.string.year3),                getString(R.string.deskripsi3),
    getString(R.string.wiki3), getString(R.string.detail3), R.drawable.img3),
                      FalloutItem(4,            getString(R.string.judul4),
    getString(R.string.year4),                getString(R.string.deskripsi4),
    getString(R.string.wiki4), getString(R.string.detail4), R.drawable.img4),
```

```
                    FalloutItem(5,            getString(R.string.judul5),
getString(R.string.year5),                getString(R.string.deskripsi5),
getString(R.string.wikinv),                  getString(R.string.detail5),
R.drawable.imgnv)
                )
            )
        }

        viewModel.falloutList.observe(viewLifecycleOwner) { falloutList -
>
            binding.rvFallout.layoutManager                              =
LinearLayoutManager(requireContext())
            binding.rvFallout.adapter = FalloutAdapter(
                falloutList,
                onDetailClick = { item ->
                    viewModel.selectFallout(item)
                    parentFragmentManager.beginTransaction()
                        .replace(R.id.fragment_container,
DetailFragment())
                        .addToBackStack(null)
                        .commit()
                },
                onWikiClick = { item ->
                    val url = item.wiki
                    if (url.isNotEmpty()) {
                        val  browserIntent  =  Intent(Intent.ACTION_VIEW,
Uri.parse(url))
                        startActivity(browserIntent)
                    }
                }
            )
        }
    }

    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }
}
```

Fragment_menu.xml

Tabel 4. Fragment_menu.xml

```
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:id="@+id/main"
7      android:layout_width="match_parent"
8      android:layout_height="match_parent"
```

```
9       tools:context=".MainActivity">
10
11      <androidx.recyclerview.widget.RecyclerView
12          android:id="@+id/rvFallout"
13          android:layout_width="0dp"
14          android:layout_height="0dp"
15          app:layout_constraintTop_toTopOf="parent"
16          app:layout_constraintBottom_toBottomOf="parent"
17          app:layout_constraintStart_toStartOf="parent"
18          app:layout_constraintEnd_toEndOf="parent"
19          android:background="@color/fallout_bg"/>
20
    </androidx.constraintlayout.widget.ConstraintLayout>
```

DetailFragment.kt

Tabel 5. DetailFragment.kt

```
1   package com.example.scrollablexml
2
3   import android.os.Bundle
4   import android.view.LayoutInflater
5   import android.view.View
6   import android.view.ViewGroup
7   import androidx.fragment.app.Fragment
8   import androidx.fragment.app.activityViewModels
9   import com.example.scrollablexml.databinding.FragmentDetailBinding
10
11  class DetailFragment : Fragment() {
12
13      private var _binding: FragmentDetailBinding? = null
14      private val binding get() = _binding!!
15
16      private val viewModel: FalloutViewModel by activityViewModels {
17  FalloutViewModelFactory() }
18
19      override fun onCreateView(
20          inflater: LayoutInflater, container: ViewGroup?,
21          savedInstanceState: Bundle?
22      ): View {
23          _binding = FragmentDetailBinding.inflate(inflater, container,
24  false)
25          return binding.root
26      }
27
28      override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
29          super.onViewCreated(view, savedInstanceState)
30
31          viewModel.selectedFallout.observe(viewLifecycleOwner) { item ->
32              if (item != null) {
```

```
                binding.tvTitle.text = item.title
                binding.tvYear.text = item.year
                binding.tvDetail.text = item.description
                binding.detailImage.setImageResource(item.imageResId)
            } else {
                binding.tvTitle.text = ""
                binding.tvYear.text = ""
                binding.tvDetail.text = ""
                binding.detailImage.setImageResource(0)
            }
        }
    }

    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }
}
```

Fragment_menu.xml

Tabel 6. Fragment_menu XML

```
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      android:fillViewport="true"
8      android:background="@color/fallout_bg"
9      >
10
11     <ImageView
12         android:id="@+id/detailImage"
13         android:layout_marginTop="15dp"
14         android:layout_width="wrap_content"
15         android:layout_height="wrap_content"
16         android:contentDescription="@string/img"
17         android:scaleType="centerCrop"
18         android:src="@drawable/img1"
19         app:layout_constraintEnd_toEndOf="parent"
20         app:layout_constraintStart_toStartOf="parent"
           app:layout_constraintTop_toTopOf="parent" />

           <TextView
               android:id="@+id/tvTitle"
               android:layout_width="0dp"
               android:layout_height="wrap_content"
               android:layout_marginTop="16dp"
               android:layout_marginStart="10dp"
               android:text="@string/title"
```

```
            android:textColor="@color/fallout_accent"
            android:textSize="22sp"
            android:textStyle="bold"

app:layout_constraintTop_toBottomOf="@+id/detailImage"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toStartOf="@+id/tvYear" />

        <TextView
            android:id="@+id/tvYear"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginEnd="10dp"
            android:text="@string/year"
            android:textColor="@color/fallout_text"
            android:textSize="20sp"

app:layout_constraintBaseline_toBaselineOf="@id/tvTitle"
            app:layout_constraintEnd_toEndOf="parent" />

        <TextView
            android:id="@+id/tvDetail"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            android:layout_marginHorizontal="20dp"
            android:text="@string/description"
            android:textColor="@color/fallout_text"
            android:textSize="16sp"
            app:layout_constraintTop_toBottomOf="@+id/tvTitle"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

FalloutAdapter.kt

Tabel 7. FalloutAdapter.kt

```
1  package com.example.scrollablexml
2
3  import android.view.LayoutInflater
4  import android.view.ViewGroup
5  import androidx.recyclerview.widget.RecyclerView
6  import com.example.scrollablexml.databinding.GameCardBinding
7
8  class FalloutAdapter(
9      private val items: List<FalloutItem>,
10     private val onWikiClick: (FalloutItem) -> Unit,
11     private val onDetailClick: (FalloutItem) -> Unit
12 ) : RecyclerView.Adapter<FalloutAdapter.ViewHolder>() {
13
```

```
14      inner class ViewHolder(val binding: GameCardBinding) :
15          RecyclerView.ViewHolder(binding.root)
16
17      override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
18  ViewHolder {
19          val                                binding                                =
20  GameCardBinding.inflate(LayoutInflater.from(parent.context),      parent,
21  false)
22          return ViewHolder(binding)
23      }
24
25      override fun getItemCount(): Int = items.size
26
27      override fun onBindViewHolder(holder: ViewHolder, position: Int) {
28          val item = items[position]
29          with(holder.binding) {
30              tvTitle.text = item.title
31              tvDescription.text = item.description
32              ivImg.setImageResource(item.imageResId)

                btnDetail.setOnClickListener {
                    onDetailClick(item)
                }

                btnWiki.setOnClickListener {
                    onWikiClick(item)
                }
            }
        }
    }
```

FalloutViewModel.kt

Tabel 8. FalloutViewModel.kt

```
1   package com.example.scrollablexml
2
3   import android.os.Bundle
4   import androidx.appcompat.app.AppCompatActivity
5   import androidx.lifecycle.ViewModelProvider
6   import com.example.scrollablexml.databinding.ActivityMainBinding
7
8   class MainActivity : AppCompatActivity() {
9
10      private lateinit var binding: ActivityMainBinding
11
12      private lateinit var viewModel: FalloutViewModel
13
14      override fun onCreate(savedInstanceState: Bundle?) {
15          super.onCreate(savedInstanceState)
16
17          binding = ActivityMainBinding.inflate(layoutInflater)
```

```
        setContentView(binding.root)

        val factory = FalloutViewModelFactory()
        viewModel                =                ViewModelProvider(this,
factory)[FalloutViewModel::class.java]

        val fragmentManager = supportFragmentManager
        val fragmentTag = MenuFragment::class.java.simpleName

        val                        existingFragment                        =
fragmentManager.findFragmentByTag(fragmentTag)
        if (existingFragment == null) {
            fragmentManager.beginTransaction()
                .add(binding.fragmentContainer.id,        MenuFragment(),
fragmentTag)
                .commit()
        }
    }
}
```

FalloutViewModelFactory.kt

Tabel 9. FalloutViewModeFactoryl.kt

```
package com.example.scrollablexml

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider

class FalloutViewModelFactory : ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        if (modelClass.isAssignableFrom(FalloutViewModel::class.java)) {
            return FalloutViewModel() as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
}
```

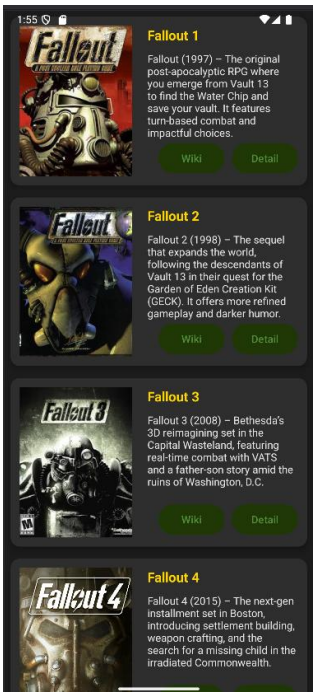
game_card.xml

Tabel 10. game_card.xml

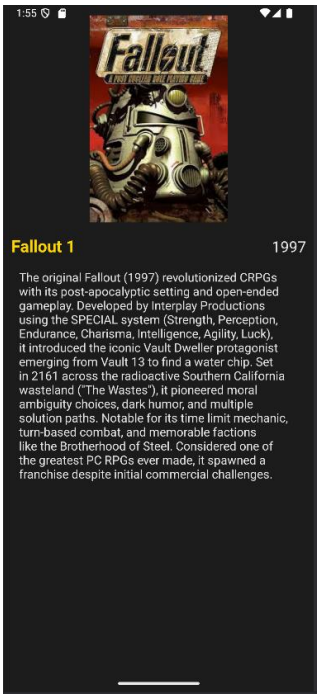
```
package com.example.scrollablexml

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.example.scrollablexml.databinding.GameCardBinding

class FalloutAdapter(
    private val items: List<FalloutItem>,
```

```
    private val onWikiClick: (FalloutItem) -> Unit,
    private val onDetailClick: (FalloutItem) -> Unit
) : RecyclerView.Adapter<FalloutAdapter.ViewHolder>() {

    inner class ViewHolder(val binding: GameCardBinding) :
        RecyclerView.ViewHolder(binding.root)

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
ViewHolder {
        val binding =
GameCardBinding.inflate(LayoutInflater.from(parent.context), parent,
false)
        return ViewHolder(binding)
    }

    override fun getItemCount(): Int = items.size

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        val item = items[position]
        with(holder.binding) {
            tvTitle.text = item.title
            tvDescription.text = item.description
            ivImg.setImageResource(item.imageResId)

            btnDetail.setOnClickListener {
                onDetailClick(item)
            }

            btnWiki.setOnClickListener {
                onWikiClick(item)
            }
        }
    }
}
```

B. Output Program

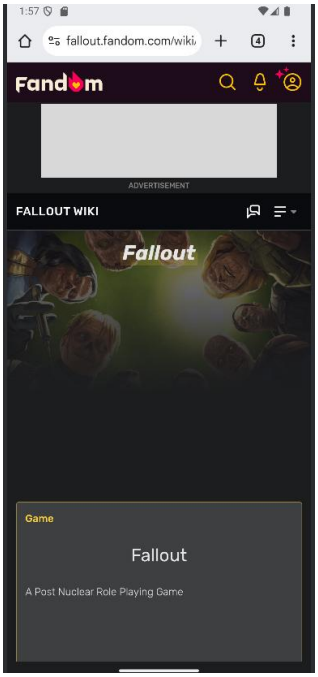


Gambar 1. Screenshot Hasil Jawaban XML



Gambar 2. Screenshot Hasil Jawaban XML

Gambar 3. Screenshot Hasil Jawaban XML



Gambar 4. Screenshot Hasil Jawaban XML

C. Pembahasan

MainActivity.kt:

Line 10: Mendeklarasikan properti viewModel dari tipe FalloutViewModel dan menandainya dengan lateinit, yang berarti akan diinisialisasi kemudian di onCreate.

Line 17: Membuat instance dari FalloutViewModelFactory. Factory ini dibutuhkan jika FalloutViewModel memiliki constructor dengan parameter atau logika pembuatan khusus.

Line 18: Menginisialisasi viewModel dengan ViewModelProvider. Ini akan:

- Memastikan ViewModel hanya dibuat satu kali untuk siklus hidup MainActivity.
- Menggunakan FalloutViewModelFactory sebagai penyedia ViewModel-nya.
- Mengambil instance FalloutViewModel yang sesuai.

MenuFragment.kt

Line 7: Diimport ViewModelProvider untuk menginisialisasi dan mengakses instance FalloutViewModel.

Line 13: Dideklarasikan variabel viewModel bertipe FalloutViewModel yang nantinya akan digunakan untuk menyimpan dan mengambil data Fallout.

Line 26: Menginisialisasi viewModel dengan ViewModelProvider menggunakan requireActivity() agar ViewModel bersifat shared antar Fragment dan Activity, serta menggunakan FalloutViewModelFactory karena ViewModel membutuhkan parameter konstruktor.

Line 28-36: Mengecek apakah falloutList di ViewModel kosong (null atau kosong). Jika kosong, maka mengisi ViewModel dengan data awal berupa list FalloutItem.

Line 38: Melakukan observasi terhadap LiveData falloutList agar UI otomatis update ketika data berubah.

Line 39-52: Pada adapter RecyclerView, ketika tombol Detail ditekan, memanggil fungsi selectFallout di ViewModel untuk menyimpan data item yang dipilih, kemudian membuka DetailFragment.

FalloutViewModel.kt

Line 10: Deklarasi class FalloutViewModel yang merupakan subclass dari ViewModel. Menerima datasource sebagai parameter untuk memuat data.

Line 12: _falloutList adalah MutableStateFlow yang menyimpan daftar game Fallout. Tipe datanya adalah List<Fallout>, awalnya kosong.

Line 13: falloutList adalah StateFlow (versi read-only) agar hanya bisa dibaca oleh UI, tidak bisa diubah langsung dari luar ViewModel.

Line 15: _selectedGame adalah MutableStateFlow yang menyimpan satu game Fallout yang dipilih (bisa null jika belum dipilih).

Line 16: selectedGame adalah versi public (read-only) dari _selectedGame, agar bisa di-observe oleh UI.

Line 18–24: init block dijalankan saat ViewModel dibuat:

- Line 19: Memanggil fungsi loadFallout() dari datasource untuk mengambil data game.
- Line 20: Menyimpan data ke dalam _falloutList.

Line 23–24: Menampilkan setiap item Fallout ke log, termasuk ID dan resource judulnya.

Line 26–30: Fungsi selectGameById(id: Int) digunakan saat user memilih game tertentu dari daftar.

Line 27: Mencari game dengan id yang sesuai dari daftar.

Line 28: Menyimpan game yang dipilih ke _selectedGame.

FalloutViewModelFactory.kt

Line 7–9: Mendeklarasikan class FalloutViewModelFactory yang mengimplementasikan ViewModelProvider.Factory.

- Digunakan saat membuat FalloutViewModel yang membutuhkan parameter Datasource.
- Karena ViewModel secara default harus punya constructor kosong, maka kita butuh factory untuk melewatkan parameter seperti Datasource.

Line 11–12: Override fungsi create() dari ViewModelProvider.Factory.

- @Suppress("UNCHECKED_CAST") digunakan untuk menonaktifkan peringatan cast tidak aman.
- Fungsi ini dipanggil saat ViewModel ingin dibuat oleh sistem.

Line 13–15:

- Mengecek apakah modelClass yang diminta adalah FalloutViewModel.
- Jika ya, buat instance FalloutViewModel dan kirim datasource ke dalamnya, lalu lakukan casting ke T.

Line 16: Jika modelClass bukan FalloutViewModel, maka lempar exception untuk memberi tahu bahwa class ViewModel yang diminta tidak dikenal.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.
https://github.com/Andra-Braputra/PrakMobile/tree/main/Modul4/Modul4XML-main/Modul4XML-main

COMPOSE

A. Source Code

MainActivity.kt

Tabel 11. MainActivity.kt

```
1   package com.example.scrollablecompose
2
3   import android.os.Bundle
4   import androidx.activity.ComponentActivity
5   import androidx.activity.compose.setContent
6   import androidx.lifecycle.ViewModelProvider
7   import androidx.navigation.NavType
8   import androidx.navigation.compose.NavHost
9   import androidx.navigation.compose.composable
10  import androidx.navigation.compose.rememberNavController
11  import androidx.navigation.navArgument
12  import com.example.scrollablecompose.data.Datasource
13  import com.example.scrollablecompose.viewmodel.FalloutViewModel
14  import
15  com.example.scrollablecompose.viewmodel.FalloutViewModelFactory
16  import timber.log.Timber
17
18  class MainActivity : ComponentActivity() {
19
20      private lateinit var viewModel: FalloutViewModel
21
22      override fun onCreate(savedInstanceState: Bundle?) {
23          super.onCreate(savedInstanceState)
24
25          Timber.plant(Timber.DebugTree())
26
27          val factory = FalloutViewModelFactory(Datasource())
28          viewModel              =              ViewModelProvider(this,
29  factory)[FalloutViewModel::class.java]
30
31          setContent {
32              val navController = rememberNavController()
33              NavHost(
34                  navController = navController,
35                  startDestination = Routes.Menu
36              ) {
                    composable(Routes.Menu) {
                        FalloutApp(navController    =    navController,
    viewModel = viewModel)
                    }
                    composable(
                        route = "detail/{gameId}",
                        arguments = listOf(navArgument("gameId") { type
    = NavType.IntType })
                    ) { backStackEntry ->
```

```
                    val                    gameId                    =
backStackEntry.arguments?.getInt("gameId") ?: 0
                    ShowDetail(gameId   =   gameId,   viewModel   =
viewModel)
                }
            }
        }
    }
}
```

Detail.kt

Tabel 12. Detail.kt

```
1  package com.example.scrollablecompose
2
3  import androidx.compose.foundation.Image
4  import androidx.compose.foundation.background
5  import androidx.compose.foundation.layout.Column
6  import androidx.compose.foundation.layout.Row
7  import androidx.compose.foundation.layout.Spacer
8  import androidx.compose.foundation.layout.aspectRatio
9  import androidx.compose.foundation.layout.fillMaxHeight
10 import androidx.compose.foundation.layout.fillMaxSize
11 import androidx.compose.foundation.layout.fillMaxWidth
12 import androidx.compose.foundation.layout.height
13 import androidx.compose.foundation.layout.padding
14 import androidx.compose.foundation.rememberScrollState
15 import androidx.compose.foundation.verticalScroll
16 import androidx.compose.material3.Card
17 import androidx.compose.material3.CardDefaults
18 import androidx.compose.material3.MaterialTheme
19 import androidx.compose.material3.Text
20 import androidx.compose.runtime.Composable
21 import androidx.compose.runtime.LaunchedEffect
22 import androidx.compose.runtime.collectAsState
23 import androidx.compose.runtime.getValue
24 import androidx.compose.ui.Modifier
25 import androidx.compose.ui.layout.ContentScale
26 import androidx.compose.ui.res.colorResource
27 import androidx.compose.ui.res.painterResource
28 import androidx.compose.ui.res.stringResource
29 import androidx.compose.ui.text.style.TextAlign
30 import androidx.compose.ui.unit.dp
31 import
32 com.example.scrollablecompose.viewmodel.FalloutViewModel
33
34 @Composable
35
36
37
```

```
fun ShowDetail(gameId: Int, viewModel: FalloutViewModel) {
    val                    selectedGame                          by
viewModel.selectedGame.collectAsState()

    LaunchedEffect(gameId) {
        viewModel.selectGameById(gameId)
    }

    if (selectedGame == null) {
        Text(
            text = "Game not found",
            modifier = Modifier.padding(16.dp),
            style = MaterialTheme.typography.bodyLarge,
            color = colorResource(R.color.fallout_text)
        )
    } else {
        val game = selectedGame!!
        Column(
            modifier = Modifier
                .fillMaxHeight()
                .background(color                                  =
colorResource(R.color.fallout_bg))
                .padding(16.dp)
                .fillMaxWidth()
                .verticalScroll(rememberScrollState())
        ) {
            Card(
                modifier = Modifier
                    .fillMaxWidth()
                    .aspectRatio(16f / 9f),
                colors                                             =
CardDefaults.cardColors(containerColor                             =
colorResource(R.color.fallout_card))
            ) {
                Image(
                    painter     =     painterResource(id     =
game.imageResId),
                    contentDescription  =  stringResource(id  =
game.titleResId),
                    modifier = Modifier.fillMaxSize(),
                    contentScale = ContentScale.Fit
                )
            }

            Spacer(modifier = Modifier.height(16.dp))

            Row {
```

```
                Text(
                    text      =      stringResource(id      =
game.titleResId),
                    style                                     =
MaterialTheme.typography.headlineLarge,
                    color                                     =
colorResource(R.color.fallout_accent)
                )
                Spacer(modifier = Modifier.weight(1f))
                Text(
                    text = stringResource(id = game.yearId),
                    style                                     =
MaterialTheme.typography.headlineLarge,
                    color                                     =
colorResource(R.color.fallout_text)
                )
            }

            Spacer(modifier = Modifier.height(16.dp))

            Text(
                text = stringResource(id = game.detailResId),
                style = MaterialTheme.typography.bodyLarge,
                modifier = Modifier.fillMaxWidth(),
                textAlign = TextAlign.Justify,
                color = colorResource(R.color.fallout_text)
            )
        }
    }
}
```

Menu.kt

Tabel 13. Menu.kt

```
1  package com.example.scrollablecompose
2
3  import android.os.Bundle
4  import androidx.activity.ComponentActivity
5  import androidx.activity.compose.setContent
6  import androidx.lifecycle.ViewModelProvider
7  import androidx.navigation.NavType
8  import androidx.navigation.compose.NavHost
9  import androidx.navigation.compose.composable
10 import androidx.navigation.compose.rememberNavController
11 import androidx.navigation.navArgument
12 import com.example.scrollablecompose.data.Datasource
13 import com.example.scrollablecompose.viewmodel.FalloutViewModel
```

```
import
com.example.scrollablecompose.viewmodel.FalloutViewModelFactory
import timber.log.Timber

class MainActivity : ComponentActivity() {

    private lateinit var viewModel: FalloutViewModel

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        Timber.plant(Timber.DebugTree())

        val factory = FalloutViewModelFactory(Datasource())
        viewModel             =             ViewModelProvider(this,
factory)[FalloutViewModel::class.java]

        setContent {
            val navController = rememberNavController()
            NavHost(
                navController = navController,
                startDestination = Routes.Menu
            ) {
                composable(Routes.Menu) {
                    FalloutApp(navController    =    navController,
viewModel = viewModel)
                }
                composable(
                    route = Routes.Detail,
                    arguments = listOf(navArgument("gameId") { type
= NavType.IntType })
                ) { backStackEntry ->
                    val                   gameId                   =
backStackEntry.arguments?.getInt("gameId") ?: 0
                    ShowDetail(gameId   =   gameId,   viewModel   =
viewModel)
                }
            }
        }
    }
}
```

Routes.kt

Tabel 14. Routes.kt

```
package com.example.scrollablecompose

object Routes {
    const val Menu = "menu"
    const val Detail = "detail/{gameId}"
```

```
}
```

DataSource.kt

Tabel 15. DataSource.kt

```
package com.example.scrollablecompose.data


import com.example.scrollablecompose.R
import com.example.scrollablecompose.model.Fallout

class Datasource {
    fun loadFallout(): List<Fallout> {
        return listOf(
            Fallout(1,  R.string.judul1,  R.string.deskripsi1,
R.drawable.img1,        R.string.wiki1,        R.string.year1,
R.string.detail1),
            Fallout(2,  R.string.judul2,  R.string.deskripsi2,
R.drawable.img2,        R.string.wiki2,        R.string.year2,
R.string.detail2),
            Fallout(3,  R.string.judul3,  R.string.deskripsi3,
R.drawable.img3,        R.string.wiki3,        R.string.year3,
R.string.detail3),
            Fallout(4,  R.string.judul4,  R.string.deskripsi4,
R.drawable.img4,        R.string.wiki4,        R.string.year4,
R.string.detail4),
            Fallout(5,  R.string.judul5,  R.string.deskripsi5,
R.drawable.imgnv,       R.string.wikinv,       R.string.year5,
R.string.detail5)
        )

    }
}
```

Fallout.kt

Tabel 16. Fallout.kt

```
package com.example.scrollablecompose.model

import androidx.annotation.DrawableRes
import androidx.annotation.StringRes

data class Fallout(
    val id: Int,
    @StringRes val titleResId: Int,
```

```
    @StringRes val descriptionResId: Int,
    @DrawableRes val imageResId: Int,
    @StringRes val wikiResId: Int,
    @StringRes val yearId: Int,
    @StringRes val detailResId: Int
)
```

FalloutViewModel.kt

Tabel 17. FalloutViewModel.kt

```
1  package com.example.scrollablecompose.viewmodel
2
3  import androidx.lifecycle.ViewModel
4  import com.example.scrollablecompose.data.Datasource
5  import com.example.scrollablecompose.model.Fallout
   import kotlinx.coroutines.flow.MutableStateFlow
   import kotlinx.coroutines.flow.StateFlow
   import timber.log.Timber

   class FalloutViewModel(private val datasource: Datasource) :
   ViewModel() {

       private val _falloutList =
   MutableStateFlow<List<Fallout>>(emptyList())
       val falloutList: StateFlow<List<Fallout>> = _falloutList

       private val _selectedGame =
   MutableStateFlow<Fallout?>(null)
       val selectedGame: StateFlow<Fallout?> = _selectedGame

       init {
           val data = datasource.loadFallout()
           _falloutList.value = data

           Timber.i("FalloutViewModel initialized with
   ${data.size} items")
           data.forEach {
               Timber.d("Loaded game: id=${it.id},
   titleResId=${it.titleResId}")
           }
       }

       fun selectGameById(id: Int) {
           val selected = _falloutList.value.firstOrNull { it.id
   == id }
           _selectedGame.value = selected
```

```
        Timber.i("Selected game ID: $id
(titleResId=${selected?.titleResId})")
    }

    fun onWikiClicked(title: String, url: String) {
        Timber.i("Wiki button clicked for: $title ($url)")
    }

    fun onDetailClicked(id: Int, title: String) {
        Timber.i("Detail button clicked: id=$id, title=$title")
    }
}
```

FalloutViewModelFactory.kt

Tabel 18. FalloutViewModelFactory.kt

```
1  package com.example.scrollablecompose.viewmodel
2
3  import androidx.lifecycle.ViewModel
4  import androidx.lifecycle.ViewModelProvider
5  import com.example.scrollablecompose.data.Datasource

class FalloutViewModelFactory(
    private val datasource: Datasource
) : ViewModelProvider.Factory {

    @Suppress("UNCHECKED_CAST")
    override fun <T : ViewModel> create(modelClass: Class<T>):
T {
        if
(modelClass.isAssignableFrom(FalloutViewModel::class.java)) {
            return FalloutViewModel(datasource) as T
        }
        throw   IllegalArgumentException("Unknown   ViewModel
class")
    }
}
```

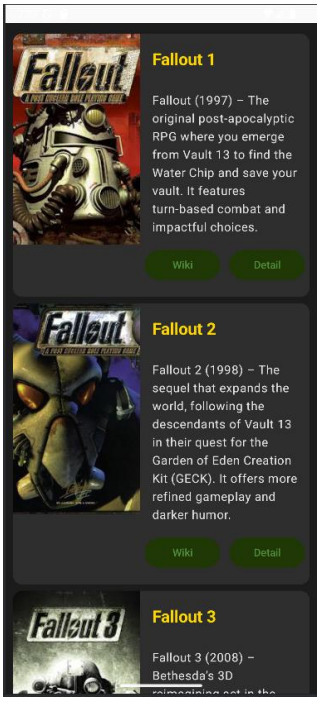
ScrollableApp.kt

Tabel 19. Scrollable.kt

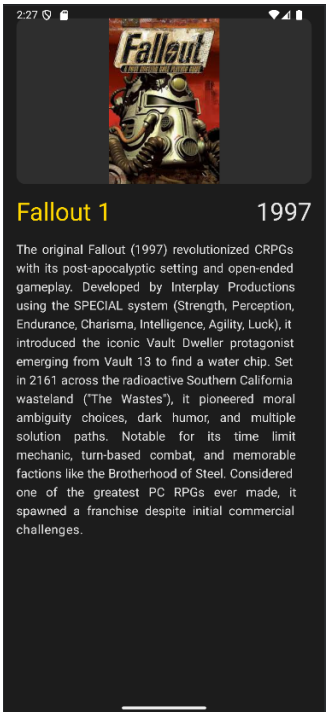
```
1  package com.example.scrollablecompose
2
3  import android.app.Application
4  import timber.log.Timber
5
```

```
class ScrollableApp : Application() {
    override fun onCreate() {
        super.onCreate()
        Timber.plant(Timber.DebugTree())
    }
}
```

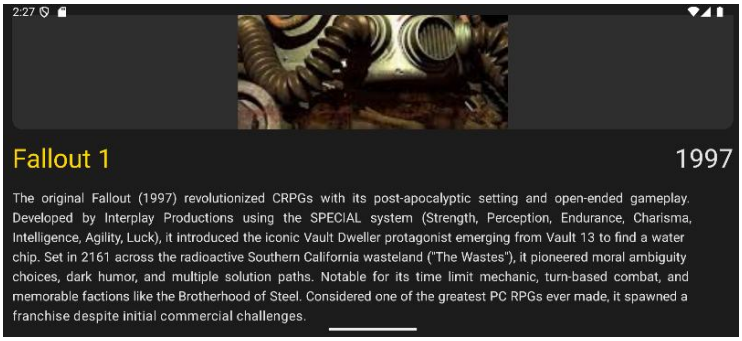
B. Output Program



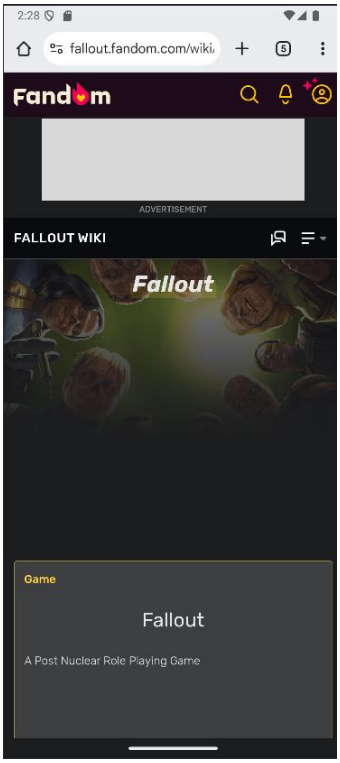
Gambar 4. Source Code Jawaban Compose



Gambar 5. Source Code Jawaban Compose



Gambar 6. Source Code Jawaban Compose



Gambar 7. Source Code Jawaban Compose

C. Pembahasan

MainActivity.kt:

Line 13: Mendeklarasikan properti viewModel bertipe FalloutViewModel dan menandainya dengan lateinit, artinya akan diinisialisasi nanti sebelum digunakan. Variabel ini menyimpan instance ViewModel yang akan digunakan di seluruh activity.

Line 17: Membuat instance FalloutViewModelFactory, dan menyuntikkan Datasource() ke dalamnya. Factory ini bertanggung jawab untuk membuat FalloutViewModel dengan parameter.

Line 18: Menginisialisasi viewModel menggunakan ViewModelProvider:

- this adalah context activity.
- factory digunakan agar FalloutViewModel bisa dibuat dengan constructor yang menerima Datasource.
- [...] digunakan untuk mengambil instance FalloutViewModel dari ViewModelProvider.

Menu.kt

Line 41: FalloutApp menerima parameter viewModel bertipe FalloutViewModel. Ini ViewModel utama yang digunakan untuk mengakses data dan logging interaksi user seperti klik tombol.
Line 45: Mengamati StateFlow dari falloutList menggunakan collectAsState() agar Compose bisa me-recompose saat data berubah.

Data yang ditampilkan adalah daftar game Fallout dari ViewModel.

Line 53 (dalam pemanggilan GameList): ViewModel diteruskan ke composable GameList agar tetap dapat digunakan untuk meneruskan ke FalloutCard.

Line 58 dan Line 70: GameList dan FalloutCard juga menerima FalloutViewModel sebagai parameter agar dapat memanggil fungsi seperti onWikiClicked dan onDetailClicked.

FalloutViewModel.kt

Line 10: Deklarasi class FalloutViewModel yang merupakan subclass dari ViewModel. Menerima datasource sebagai parameter untuk memuat data.

Line 12: _falloutList adalah MutableStateFlow yang menyimpan daftar game Fallout. Tipe datanya adalah List<Fallout>, awalnya kosong.

Line 13: falloutList adalah StateFlow (versi read-only) agar hanya bisa dibaca oleh UI, tidak bisa diubah langsung dari luar ViewModel.

Line 15: _selectedGame adalah MutableStateFlow yang menyimpan satu game Fallout yang dipilih (bisa null jika belum dipilih).

Line 16: selectedGame adalah versi public (read-only) dari _selectedGame, agar bisa di-observe oleh UI.

Line 18–24: init block dijalankan saat ViewModel dibuat:

Line 19: Memanggil fungsi loadFallout() dari datasource untuk mengambil data game.

Line 20: Menyimpan data ke dalam _falloutList.

Line 23–24: Menampilkan setiap item Fallout ke log, termasuk ID dan resource judulnya.

Line 26–30: Fungsi selectGameById(id: Int) digunakan saat user memilih game tertentu dari daftar.

Line 27: Mencari game dengan id yang sesuai dari daftar.

Line 28: Menyimpan game yang dipilih ke _selectedGame.

FalloutViewModelFactory.kt

Line 7–9: Mendeklarasikan class FalloutViewModelFactory yang mengimplementasikan ViewModelProvider.Factory.

- Digunakan saat membuat FalloutViewModel yang membutuhkan parameter Datasource.
- Karena ViewModel secara default harus punya constructor kosong, maka kita butuh factory untuk melewatkan parameter seperti Datasource.

Line 11–12: Override fungsi create() dari ViewModelProvider.Factory.

- @Suppress("UNCHECKED_CAST") digunakan untuk menonaktifkan peringatan cast tidak aman.
- Fungsi ini dipanggil saat ViewModel ingin dibuat oleh sistem.

Line 13–15:

- Mengecek apakah modelClass yang diminta adalah FalloutViewModel.
- Jika ya, buat instance FalloutViewModel dan kirim datasource ke dalamnya, lalu lakukan casting ke T.

Line 16: Jika modelClass bukan FalloutViewModel, maka lempar exception untuk memberi tahu bahwa class ViewModel yang diminta tidak dikenal.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.
https://github.com/Andra-Braputra/PrakMobile/tree/main/Modul4/Modul4Compose-main/Modul4Compose-main

LIBRARY TIMBER

a. Log saat data item masuk ke dalam list

```
init {
    val data = datasource.loadFallout()
    _falloutList.value = data

    Timber.i( message: "FalloutViewModel ada ${data.size} item")
    data.forEach {
        Timber.d( message: "Game: id=${it.id}, titleResId=${it.titleResId}")
    }
}
```

Gambar 8. Screenshot Timber log

```
2025-05-21 14:53:32.191  6314-6314  FalloutViewModel        com.example.scrollablecompose        I  FalloutViewModel ada 5 item
```

Gambar 9. Screenshot Log List masuk

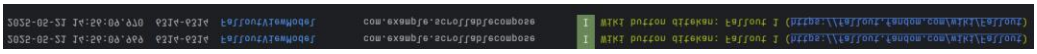
b. Log saat tombol Detail dan tombol Explicit Intent ditekan

```
fun onWikiClicked(title: String, url: String) {
    Timber.i( message: "Wiki button ditekan: $title ($url)")
}

fun onDetailClicked(id: Int, title: String) {
    Timber.i( message: "Detail button ditekan: id=$id, title=$title")
}
```

Gambar 10. Screenshot Timber log

```
2025-05-21 14:54:43.746  6314-6314  FalloutViewModel        com.example.scrollablecompose       I  Detail button ditekan: id=1, title=Fallout 1
2025-05-21 14:54:43.746  6314-6314  FalloutViewModel        com.example.scrollablecompose       I  Detail button ditekan: id=1, title=Fallout 1
```

Gambar 11.. Screenshot Log tombol detail



Gambar 12. Screenshot Log tombol wiki

c. Log data dari list yang dipilih ketika berpindah ke halaman Detail

```
fun selectGameById(id: Int) {
    val selected = _falloutList.value.firstOrNull { it.id == id }
    _selectedGame.value = selected
    Timber.i( message: "Selected game ID: $id (titleResId=${selected?.titleResId})")
}
```

Gambar 13. Screenshot Timber log

```
2025-05-21 14:54:44.039  6314-6314  FalloutViewModel        com.example.scrollablecompose       I  Selected game ID: 1 (titleResId=2131296282)
2025-05-21 14:54:44.041  6314-6314  FalloutViewModel        com.example.scrollablecompose       I  Selected game ID: 1 (titleResId=2131296282)
```

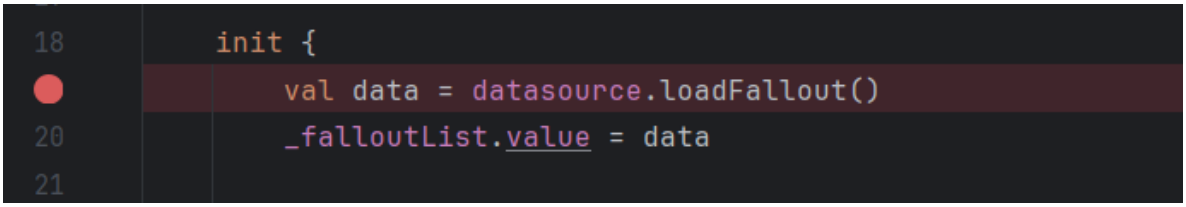
Gambar 14. Screenshot Log ke detail

DEBUGGER

Breakpoint

Breakpoint ini menghentikan eksekusi program saat mencapai baris ini

pemeriksaan state program sebelum/sesudah baris ini dieksekusi



Gambar 15. Screenshot breakpoint

Step Into

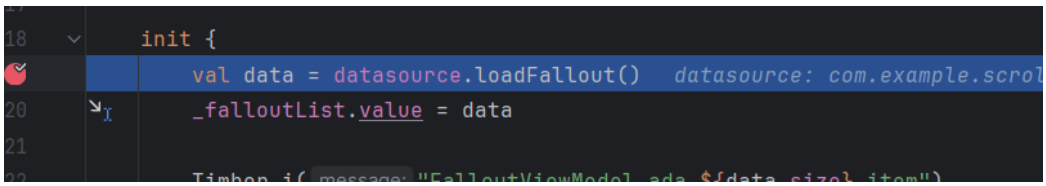
debugger sedang dalam mode step into, yang berarti akan masuk ke dalam fungsi yang dipanggil pada baris berikutnya.

Gambar 16. Screenshot Step Into

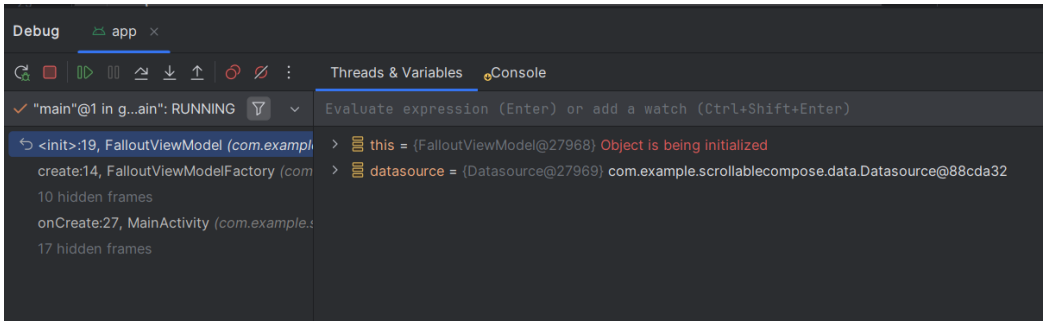
Gambar 17. Screenshot Step Into

Step Out

Jika sudah step into ke loadFallout(), step out akan menyelesaikan fungsi ini dan kembali ke baris setelah pemanggilannya



Gambar 18. Screenshot Step Out



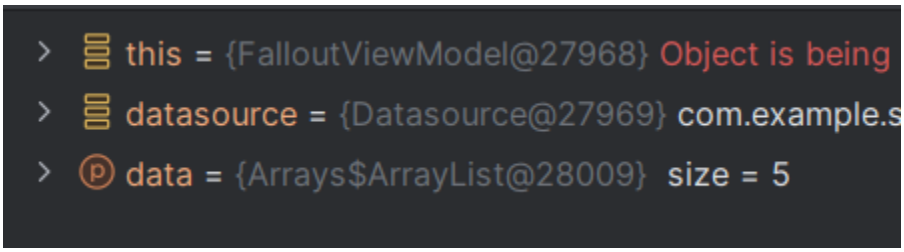
Gambar 19. Screenshot Step Out

Step Over

Di breakpoint yang sama, step over akan mengeksekusi loadFallout() dan langsung ke _falloutList.value = data

Gambar 20. Screenshot Step Over

Data yang didapat dari .loadFallout akan di tampilkan.



Gambar 21. Screenshot Step Over

SOAL 2

2. Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

Jawaban

- Digunakan untuk menginisialisasi library, dependency injection (seperti Dagger/Hilt), database (Room), atau layanan lain sebelum komponen aplikasi dimulai.
- Menyimpan data yang harus bertahan di seluruh lifecycle aplikasi (misalnya, cache, variabel global, atau instance singleton).
- Tempat yang tepat untuk mengatur DI framework
- Mengatur global exception handler untuk menangkap crash