LAPORAN PRAKTIKUM PENGEMBANGAN APLIKASI BERGERAK

RESPONSI 1

Disusun oleh:

Nama : Humam Alwi Ahmad

Nim : L0122075

Kelas : B

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET

2024

1. Screenshot Source Code
    A. Data Class

```
data class DataClass(
    val Image: Int,
    val Title: String?,
    val Des: String?,
    val Detail: ArrayList<DetailDataClass>
)
```

Kode di atas mendefinisikan sebuah kelas data bernama DataClass yang terdiri dari empat properti: Image, Title, Des, dan Detail. Properti Image bertipe integer, digunakan untuk menyimpan ID gambar. Title dan Des bertipe nullable string, digunakan untuk menyimpan judul dan deskripsi. Properti Detail bertipe ArrayList yang berisi objek-objek DetailDataClass, digunakan untuk menyimpan rincian detail terkait.

B. Adapter Class

```
class AdapterClass(private val dataList: ArrayList<DataClass>) :
    RecyclerView.Adapter<AdapterClass.ViewHolderClass>() {

    var onItemClick :((DataClass) -> Unit)? = null
    class ViewHolderClass(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val rvImage: ImageView = itemView.findViewById(R.id.image)
        val rvTitle: TextView = itemView.findViewById(R.id.title)
        val rvDescription: TextView = itemView.findViewById(R.id.description)
        val rvCard: CardView = itemView.findViewById(R.id.kru_card)
    }
```

Kode di atas mendefinisikan sebuah kelas AdapterClass yang bertugas untuk mengelola data serta tampilan item di dalam sebuah RecyclerView. Kelas ini menerima sebuah parameter dataList, yaitu sebuah ArrayList yang berisi objek-objek DataClass. Di dalam kelas AdapterClass terdapat sebuah kelas ViewHolderClass yang menginisialisasi tampilan-tampilan yang ada dalam setiap item, seperti ImageView, TextView, dan CardView.

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolderClass {
    val itemView = LayoutInflater.from(parent.context).inflate(R.layout.krulist, parent, attachToRoot: false)
    return ViewHolderClass(itemView)
}
```

Fungsi onCreateViewHolder bertugas untuk meng-inflate layout item (krulist.xml) dan mengembalikan sebuah objek ViewHolderClass yang berisi tampilan-tampilan tersebut.

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

Fungsi getItemCount mengembalikan jumlah item yang ada di dalam dataList.

```
override fun onBindViewHolder(holder: ViewHolderClass, position: Int) {
    val currentItem = dataList[position]
    holder.rvImage.setImageResource(currentItem.Image)
    holder.rvTitle.text = currentItem.Title
    holder.rvDescription.text = currentItem.Des
    holder.rvCard.setOnClickListener { it: View!
        val context = holder.itemView.context
        val intent = Intent(context, DetailActivity::class.java).apply { this: Intent
            putExtra( name: "IMAGE", currentItem.Image)
            putExtra( name: "TITLE", currentItem.Title)
            putExtra( name: "DESCRIPTION", currentItem.Des)
            putParcelableArrayListExtra( name: "DETAIL_LIST", currentItem.Detail)
        }
        context.startActivity(intent)
    }
}
```

Fungsi onBindViewHolder bertugas untuk mengikat data dari objek DataClass ke tampilan-tampilan yang ada dalam ViewHolderClass. Dalam method ini, data dari objek DataClass yang berada pada posisi tertentu akan diambil dan digunakan untuk mengatur gambar (rvImage), judul (rvTitle), dan deskripsi (rvDescription) dari item tersebut. Selain itu, terdapat sebuah OnClickListener yang diterapkan pada CardView (rvCard). Ketika item di-klik, listener ini akan membuat sebuah Intent baru yang mengarah ke DetailActivity. Data dari objek DataClass yang di-klik akan dimasukkan ke dalam Intent sebagai ekstra dengan menggunakan putExtra dan putParcelableArrayListExtra. Setelah itu, Intent tersebut akan digunakan untuk memulai DetailActivity, memungkinkan pengguna untuk melihat detail dari item yang dipilih.

C. DashboardFragment

```
class DashboardFragment : Fragment() {

    private lateinit var adapter: AdapterClass
    private lateinit var recyclerView: RecyclerView
    private lateinit var array: ArrayList<DataClass>

    private lateinit var Image: Array<Int>
    private lateinit var Title: Array<String>
    private lateinit var Des: Array<String>
    private lateinit var Detail: Array<ArrayList<DetailDataClass>>

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        return inflater.inflate(R.layout.fragment_dashboard, container, attachToRoot: false)
    }
```

Kode di atas mendefinisikan sebuah kelas DashboardFragment yang bertugas untuk menampilkan daftar data dalam bentuk RecyclerView. Kelas ini mendefinisikan beberapa variabel seperti adapter, recyclerView, dan array yang merupakan daftar data utama yang akan ditampilkan. Selain itu, ada beberapa array yang digunakan untuk

menyimpan data gambar, judul, deskripsi, dan detail. Pada fungsi onCreateView, layout fragment di-inflate menggunakan fragment_dashboard.

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    dataInitialize()
    val layoutManager = LinearLayoutManager(context)
    recyclerView = view.findViewById(R.id.dashboard)
    recyclerView.layoutManager = layoutManager
    recyclerView.setHasFixedSize(true)
    adapter = AdapterClass(array)
    recyclerView.adapter = adapter

    adapter.onItemClick = { data ->
        val intent = Intent(context, DetailActivity::class.java).apply { this: Intent
            putParcelableArrayListExtra( name: "DETAILS", data.Detail)
            putExtra( name: "IMAGE", data.Image)
            putExtra( name: "TITLE", data.Title)
            putExtra( name: "DESCRIPTION", data.Des)
        }
        startActivity(intent)
    }
}
```

Setelah itu, pada fungsi onViewCreated, data diinisialisasi melalui dataInitialize(), di mana data gambar, judul, deskripsi, dan detail diatur. RecyclerView diatur menggunakan LinearLayoutManager dan AdapterClass yang mengelola bagaimana data ditampilkan di dalam RecyclerView. Adapter ini kemudian dipasang ke RecyclerView. Setiap kali sebuah item di dalam RecyclerView diklik, sebuah Intent baru dibuat untuk memulai DetailActivity. Data yang terkait dengan item yang diklik, seperti gambar, judul, deskripsi, dan detail, dimasukkan ke dalam Intent menggunakan method putExtra dan putParcelableArrayListExtra. Aktivitas baru kemudian dimulai dengan startActivity(intent).

```
private fun dataInitialize() {
    array = ArrayList()

    val KruImagesList = arrayListOf(
        arrayListOf(
            R.drawable.edwardnewgate,
            R.drawable.marco,
            R.drawable.portgasdace,
```

```
val KruNamesList = arrayListOf(
    arrayListOf(
        "Edward Newgate", "Marco", "Portgas D. Ace", "Jozu", "Izou", "Kozuki Oden", "Thatch", "
    ),
```

```
val indexList = listOf(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

Image = arrayOf(
    R.drawable.krushirohige,
    R.drawable.kruroger,
    R.drawable.krushanks,
    R.drawable.krubigmom,
    R.drawable.krukaido,
    R.drawable.krukurohige,
    R.drawable.krutopijerami,
    R.drawable.krulaw,
    R.drawable.krukid,
    R.drawable.krudofi
)

Title = resources.getStringArray(R.array.kru_titles)
Des = resources.getStringArray(R.array.kru_descriptions)

Detail = Array(indexList.size) { index ->
    val kruImages = KruImagesList[indexList[index]]
    val kruNames = KruNamesList[indexList[index]]
    val detailDataList = kruImages.mapIndexed { j, imageRes ->
        DetailDataClass(imageRes, kruNames[j])
    } as ArrayList<DetailDataClass>
    detailDataList ^Array
```

Dalam fungsi dataInitialize, beberapa daftar data seperti KruImagesList dan KruNamesList didefinisikan dan diisi dengan gambar dan nama karakter. Data ini kemudian diubah menjadi daftar objek DetailDataClass. Variabel Image, Title, Des, dan Detail diisi dengan data yang sesuai dari sumber daya dan daftar yang telah dibuat. Terakhir, objek-objek DataClass dibuat dengan menggunakan data ini dan ditambahkan ke dalam array yang akan digunakan oleh adapter.

D. DetailDataClass

```
@Parcelize
data class DetailDataClass(
    val image: Int,
    val name: String?,
): Parcelable
```

Kode di atas mendefinisikan sebuah kelas data DetailDataClass yang menggunakan anotasi @Parcelize. Kelas ini memiliki dua properti: image bertipe integer dan name bertipe nullable string. Anotasi @Parcelize digunakan untuk secara otomatis mengimplementasikan antarmuka Parcelable pada kelas ini. Antarmuka Parcelable memungkinkan objek dari kelas ini untuk di-serialize dan di-deserialize,

yang sangat berguna saat objek perlu dikirim antar-komponen Android seperti aktivitas atau fragment

E. DetailAdapter

```
class DetailAdapter(private val detailList: ArrayList<DetailDataClass>) :
    RecyclerView.Adapter<DetailAdapter.DetailViewHolder>() {

    inner class DetailViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val imageView: ImageView = itemView.findViewById(R.id.imagedetailvw)
        val nameTextView: TextView = itemView.findViewById(R.id.titledetailvw)
    }
```

Kode di atas mendefinisikan sebuah kelas DetailAdapter yang berfungsi untuk mengelola tampilan item-item dalam sebuah RecyclerView berdasarkan data yang diberikan dalam bentuk ArrayList<DetailDataClass>. Kelas ini memiliki sebuah inner class DetailViewHolder yang menginisialisasi tampilan-tampilan dalam setiap item, seperti ImageView dan TextView, yang masing-masing ditemukan dengan menggunakan findViewById.

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): DetailViewHolder {
    val view = LayoutInflater.from(parent.context).inflate(R.layout.krulist2, parent, attachToRoot: false)
    return DetailViewHolder(view)
}

override fun onBindViewHolder(holder: DetailViewHolder, position: Int) {
    val detailData = detailList[position]
    holder.imageView.setImageResource(detailData.image)
    holder.nameTextView.text = detailData.name
}

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

Fungsi onCreateViewHolder bertanggung jawab untuk meng-inflate layout item dari krulist2.xml dan mengembalikan sebuah objek DetailViewHolder yang berisi referensi ke tampilan-tampilan tersebut. Fungsi onBindViewHolder mengikat data dari detailList ke tampilan-tampilan dalam DetailViewHolder. Pada fungsi ini, data dari objek DetailDataClass yang berada pada posisi tertentu akan diambil dan digunakan untuk mengatur gambar (imageView) dan teks nama (nameTextView) dari item tersebut. Fungsi getItemCount mengembalikan jumlah item yang ada di dalam detailList, yang menunjukkan berapa banyak item yang akan ditampilkan dalam RecyclerView.

F. DetailActivity

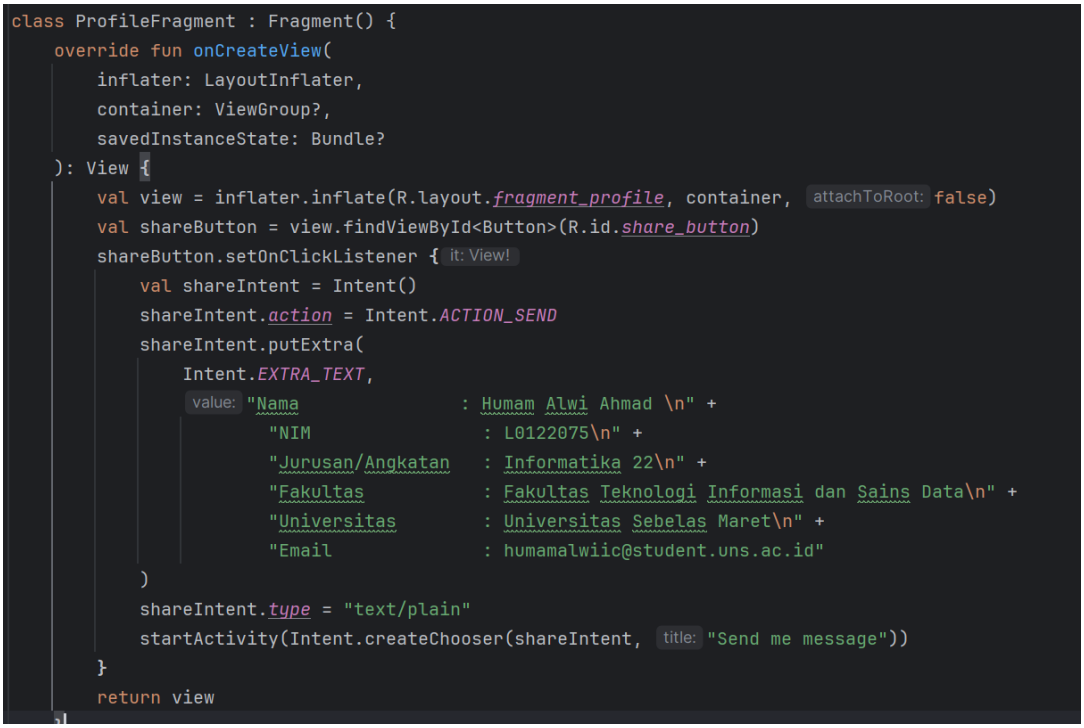
```
class DetailActivity: AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        supportActionBar?.hide()
        setContentView(R.layout.detail_kru)
        val image = intent.getIntExtra( name: "IMAGE",  defaultValue: 0)
        val title = intent.getStringExtra( name: "TITLE")
        val description = intent.getStringExtra( name: "DESCRIPTION")
        val Detail = intent.getParcelableArrayListExtra<DetailDataClass>( name: "DETAIL_LIST")
```

Kode di atas mendefinisikan sebuah kelas DetailActivity yang digunakan untuk menampilkan detail dari item yang dipilih sebelumnya di DashboardFragment. Pada fungsi onCreate, pertama-tama memanggil super.onCreate untuk menjalankan logika inisialisasi default dari AppCompatActivity. Kemudian, supportActionBar?.hide() digunakan untuk menyembunyikan action bar. Setelah itu, layout detail_kru diatur sebagai konten tampilan aktivitas dengan setContentView. Data yang diteruskan melalui intent, seperti gambar (IMAGE), judul (TITLE), deskripsi (DESCRIPTION), dan daftar detail (DETAIL_LIST), diambil menggunakan fungsi getIntExtra, getStringExtra, dan getParcelableArrayListExtra.

```
    val imageView: ImageView = findViewById(R.id.imageDetail)
    val titleView: TextView = findViewById(R.id.titleDetail)
    val descriptionView: TextView = findViewById(R.id.desDetail)

    imageView.setImageResource(image)
    titleView.text = title
    descriptionView.text = description

    val detailRecyclerView: RecyclerView = findViewById(R.id.details)
    detailRecyclerView.layoutManager = GridLayoutManager( context: this,  spanCount: 2)
    detailRecyclerView.adapter = DetailAdapter( detailList: Detail ?: ArrayList())


}
```

Tampilan-tampilan seperti ImageView, TextView untuk judul dan deskripsi diinisialisasi menggunakan findViewById, dan data yang diambil dari intent kemudian diatur ke dalam tampilan-tampilan tersebut. Gambar diatur menggunakan setImageResource, dan teks judul serta deskripsi diatur menggunakan setText. Selanjutnya, sebuah RecyclerView untuk menampilkan daftar detail diinisialisasi dan diatur dengan GridLayoutManager yang mengatur item dalam dua kolom. Adapter DetailAdapter diatur pada RecyclerView tersebut dengan data Detail yang diambil dari intent. Jika Detail null, maka adapter akan diinisialisasi dengan sebuah ArrayList kosong.

G. ProfileFragment



```
class ProfileFragment : Fragment() {
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        val view = inflater.inflate(R.layout.fragment_profile, container, attachToRoot: false)
        val shareButton = view.findViewById<Button>(R.id.share_button)
        shareButton.setOnClickListener { it: View!
            val shareIntent = Intent()
            shareIntent.action = Intent.ACTION_SEND
            shareIntent.putExtra(
                Intent.EXTRA_TEXT,
                value: "Nama                : Humam Alwi Ahmad \n" +
                        "NIM                 : L0122075\n" +
                        "Jurusan/Angkatan    : Informatika 22\n" +
                        "Fakultas            : Fakultas Teknologi Informasi dan Sains Data\n" +
                        "Universitas         : Universitas Sebelas Maret\n" +
                        "Email               : humamalwiic@student.uns.ac.id"
            )
            shareIntent.type = "text/plain"
            startActivity(Intent.createChooser(shareIntent, title: "Send me message"))
        }
        return view
    }
```

Kode di atas mendefinisikan sebuah kelas ProfileFragment yang bertugas untuk menampilkan profil pengguna dalam sebuah fragment dan menyediakan fitur berbagi informasi profil melalui aplikasi lain. Fungsi onCreateView bertanggung jawab untuk meng-inflate layout fragment_profile dan mengembalikan tampilan tersebut sebagai objek View. Dalam fungsi ini, terdapat sebuah tombol dengan ID share_button yang diinisialisasi menggunakan findViewById. Kemudian, sebuah OnClickListener diterapkan pada tombol ini untuk menangani aksi ketika tombol diklik.

Ketika tombol shareButton diklik, sebuah Intent baru dibuat dengan aksi Intent.ACTION_SEND. Data yang akan dibagikan, yaitu informasi profil pengguna seperti nama, NIM, jurusan, fakultas, universitas, dan email, dimasukkan ke dalam intent menggunakan putExtra dengan kunci Intent.EXTRA_TEXT. Tipe data intent diatur sebagai text/plain. Selanjutnya, startActivity(Intent.createChooser(shareIntent, "Send me message")) digunakan untuk memulai aktivitas berbagi dengan menampilkan aplikasi-aplikasi yang mendukung aksi berbagi teks sehingga pengguna dapat memilih aplikasi yang ingin digunakan untuk berbagi informasi profil tersebut.

H. Screenshot Terminal

Berikut adalah hasil pada terminal apabila source code dijalankan.

a. Mode Terang

b. Mode Gelap

I. Kesimpulan

Kesimpulan dari praktikum responsi ini adalah bahwa aplikasi ini memanfaatkan berbagai komponen seperti DataClass dan DetailDataClass yang digunakan untuk menyimpan dan mengelola data, sedangkan AdapterClass dan DetailAdapter menampilkan data tersebut dalam RecyclerView. DashboardFragment menampilkan daftar item utama dan mengirimkan data ke DetailActivity untuk

menampilkan detail item yang dipilih. Selain itu, ProfileFragment menyediakan fitur berbagi informasi profil pengguna.