LAPORAN PRATIKUM 5

APLIKASI MOBILE

Dosen Pengampu: Nurfiah, S.ST, M.Kom

DISUSUN OLEH :

Rehan Khairuno

2211533003

DEPARTEMEN INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

2024

A. Mainactivity

```
package com.rehan.conecttojson;

import ...

public class MainActivity extends AppCompatActivity {

    private static final String TAG = "MainActivity";  3 usages
    private static final String url = "http://192.168.1.25/BelajarAPI/costumer.php";  2 usages
    private TextView textView;  3 usages

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        textView = findViewById(R.id.textView);

        // Check if the root view exists in layout XML
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });

        Log.d(TAG, msg: "Requesting data from URL: " + url);

        JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(
                Request.Method.GET,
                url,
                jsonRequest: null,
                response -> {
                    Log.d(TAG, msg: "Response received: " + response.toString());
                    textView.setText("Response: " + response.toString());
                },
                error -> {
                    Log.e(TAG, msg: "Error: " + error.toString());
                    textView.setText("Error: " + error.toString());
                }
        );

        // Adding request to request queue
        MySingleton.getInstance( context: this).addToRequestQueue(jsonObjectRequest);
    }
}
```

Penjelasan Kode:

Kode ini adalah implementasi dari MainActivity di Android, di mana aplikasi menggunakan library *Volley* untuk mengirim permintaan HTTP GET ke API, mengakses data JSON dari URL yang ditentukan, dan menampilkan respons atau kesalahan pada tampilan teks di aplikasi.

Bagian-Bagian Kode:

1. **Deklarasi Konstanta dan Variabel:**
    - TAG : Digunakan untuk menandai log sehingga lebih mudah dilacak di logcat.
    - url : URL endpoint API yang dituju (http://192.168.1.25/BelajarAPI/costumer.php).
    - textView : Elemen UI untuk menampilkan respons atau pesan kesalahan.
2. **Method onCreate:**
    - **Mengatur Layout:** setContentView(R.layout.activity_main); menetapkan layout dari activity_main.xml untuk tampilan MainActivity.
    - **Inisialisasi textView:** Variabel textView dihubungkan ke elemen dengan id textView pada layout activity_main.xml.
    - **Pengaturan Padding untuk Insets:** Listener ViewCompat.setOnApplyWindowInsetsListener digunakan untuk mengatur padding sesuai insets, menyesuaikan tampilan agar tidak terpotong oleh *system bars* seperti *status bar* atau *navigation bar*.
3. **Mengirim Permintaan JSON:**
    - **Membuat Request JSON:** JsonObjectRequest digunakan untuk mengirim permintaan GET ke API.
        - Request.Method.GET: Menetapkan metode HTTP sebagai GET.
        - url: URL tujuan API.

- response -> { ... }: *Callback* yang akan dipanggil saat respons berhasil diterima.
  - Log.d(TAG, "Response received: " + response.toString()); : Menampilkan respons pada log.
  - textView.setText("Response: " + response.toString()); : Menampilkan data JSON respons ke textView.
- error -> { ... }: *Callback* yang akan dipanggil saat ada kesalahan.
  - Log.e(TAG, "Error: " + error.toString());: Mencatat pesan kesalahan pada log.
  - textView.setText("Error: " + error.toString()); : Menampilkan pesan kesalahan di textView.

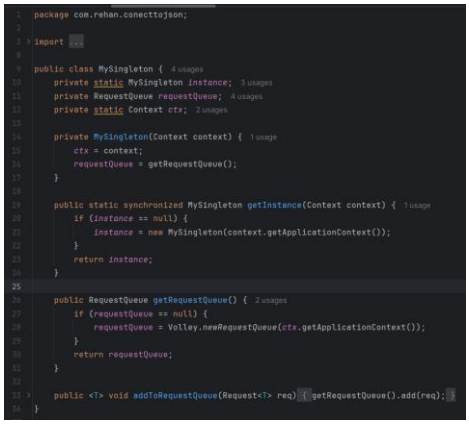
4. **Menambahkan Permintaan ke Request Queue:**
   - MySingleton.getInstance(this).addToRequestQueue(jsonObjectRequest);: Permintaan ditambahkan ke *request queue* menggunakan *singleton* MySingleton untuk mengelola antrean permintaan.

Menyalin Kode ke Word

Untuk memasukkan kode ini ke Microsoft Word, ikuti langkah-langkah berikut:

1. Salin kode di atas dengan menyorot keseluruhan kode dan menekan Ctrl + C (Windows) atau Command + C (Mac).
2. Buka dokumen Word dan letakkan kursor di lokasi yang diinginkan.
3. Tempel kode dengan Ctrl + V (Windows) atau Command + V (Mac).
4. Agar lebih rapi, Anda bisa menggunakan *Font* khusus seperti Courier New atau Consolas, dan mengatur ukuran font lebih kecil agar kode lebih mudah dibaca.

B. MySingleton



```
package com.rehan.conecttojson;

import ...

public class MySingleton {  4 usages
    private static MySingleton instance;  3 usages
    private RequestQueue requestQueue;  4 usages
    private static Context ctx;  2 usages

    private MySingleton(Context context) {  1 usage
        ctx = context;
        requestQueue = getRequestQueue();
    }

    public static synchronized MySingleton getInstance(Context context) {  1 usage
        if (instance == null) {
            instance = new MySingleton(context.getApplicationContext());
        }
        return instance;
    }

    public RequestQueue getRequestQueue() {  2 usages
        if (requestQueue == null) {
            requestQueue = Volley.newRequestQueue(ctx.getApplicationContext());
        }
        return requestQueue;
    }

    public <T> void addToRequestQueue(Request<T> req) { getRequestQueue().add(req); }
}
```

Penjelasan Kode:
Kode ini mendefinisikan sebuah kelas singleton bernama MySingleton yang berfungsi untuk mengelola RequestQueue di seluruh aplikasi. Dengan menggunakan pola *singleton*, aplikasi dapat memastikan hanya ada satu instance RequestQueue, yang memudahkan pengelolaan permintaan jaringan dan menghemat sumber daya.

Bagian-Bagian Kode:

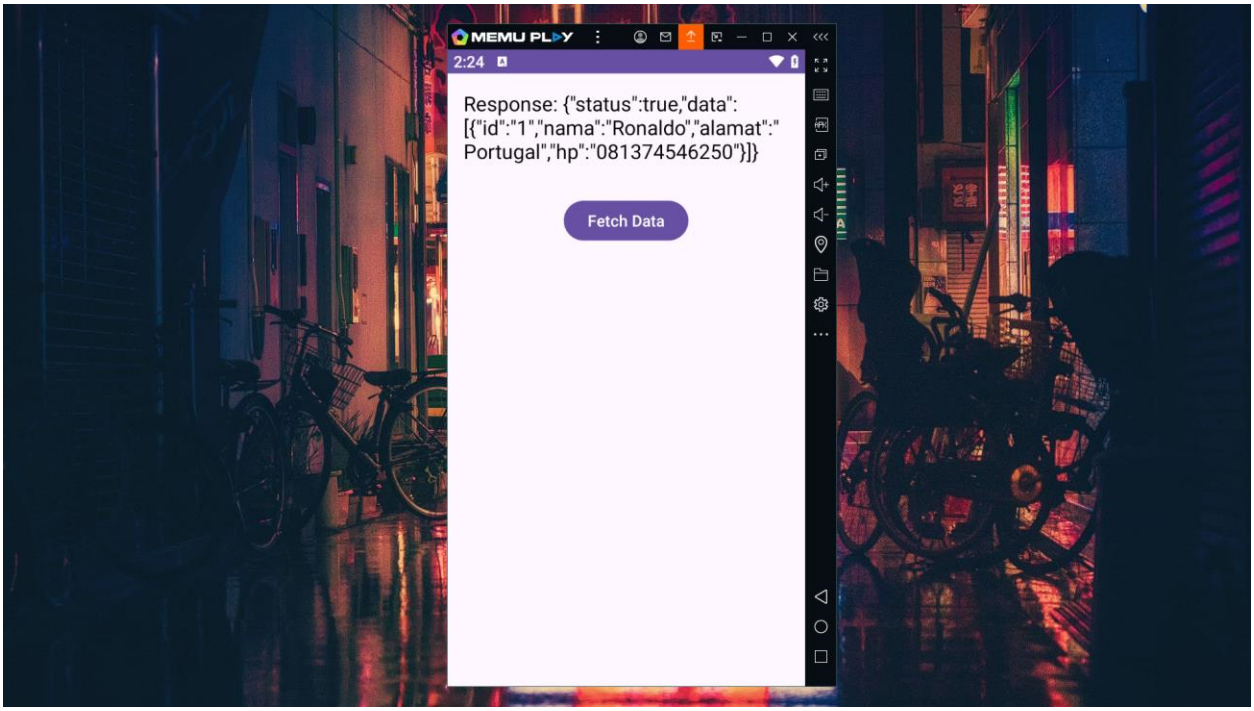
1. **Deklarasi Variabel:**
    - instance : Menyimpan satu-satunya instance dari MySingleton.
    - requestQueue : Menyimpan RequestQueue yang akan digunakan untuk menangani permintaan jaringan.
    - ctx : Menyimpan konteks aplikasi, yang digunakan untuk menginisialisasi RequestQueue.
2. **Konstruktor MySingleton:**
    - private MySingleton(Context context): Konstruktor ini bersifat *private* untuk mencegah instansiasi langsung dari kelas ini. Hanya metode getInstance() yang dapat memanggil konstruktor ini.
    - ctx = context; : Menyimpan konteks aplikasi yang diteruskan sebagai parameter.
    - requestQueue = getRequestQueue(); : Memastikan bahwa requestQueue diinisialisasi ketika instance MySingleton dibuat.
3. **Metode getInstance:**
    - public static synchronized MySingleton getInstance(Context context): Metode ini memastikan bahwa hanya ada satu instance MySingleton yang dibuat.
        - synchronized : Memastikan metode ini aman diakses oleh beberapa thread secara bersamaan.
        - Jika instance masih null, maka sebuah objek baru MySingleton akan dibuat menggunakan konteks aplikasi (context.getApplicationContext()), dan disimpan dalam instance.
        - return instance; : Mengembalikan instance MySingleton yang sudah ada atau baru dibuat.
4. **Metode getRequestQueue:**
    - public RequestQueue getRequestQueue(): Mengembalikan RequestQueue untuk permintaan jaringan.
        - Jika requestQueue masih null, maka sebuah RequestQueue baru akan dibuat dengan memanggil Volley.newRequestQueue(ctx.getApplicationContext());, dan disimpan dalam requestQueue.
        - return requestQueue; : Mengembalikan RequestQueue yang sudah ada atau baru dibuat.
5. **Metode addToRequestQueue:**
    - public <T> void addToRequestQueue(Request<T> req): Menambahkan permintaan jaringan ke dalam RequestQueue.
        - getRequestQueue().add(req); : Memanggil getRequestQueue() untuk mendapatkan antrean permintaan, kemudian menambahkan req ke antrean tersebut.

C. activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <!-- TextView for displaying response message -->
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Response will appear here"
        android:textColor="@android:color/black"
        android:textSize="18sp"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="16dp"/>

    <!-- Button to fetch data -->
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Fetch Data"
        android:layout_below="@id/textView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="16dp"/>
```

```xml
    <!-- Button to fetch data -->
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Fetch Data"
        android:layout_below="@id/textView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="16dp"/>

    <!-- RecyclerView for displaying list of customers -->
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@id/button"
        android:layout_marginTop="16dp"/>
</RelativeLayout>
```

Output :


MEMU PLAY
2:24
Response: {"status":true,"data":
[{"id":"1","nama":"Ronaldo","alamat":"
Portugal","hp":"081374546250"}]}
Fetch Data