# Arsitektur JavaFX: Penjelasan Berdasarkan Contoh Kode

Pola arsitektur **Model, Repository, Service, dan Controller** adalah pendekatan terstruktur untuk membangun aplikasi yang kuat. Mari kita bedah pola ini menggunakan contoh kode Member yang telah dibuat.

### Komponen-Komponen Arsitektur dalam Contoh

Setiap lapisan dalam kode kita memiliki peran yang spesifik:

#### 1. Model (com.activa.models.Member)

* **Peran:** Merepresentasikan data seorang anggota (member).
* **Dalam Kode:** Kelas Member adalah representasi data kita. Ia berisi properti seperti id, nim, name, birthdate, isActive, dan lainnya. Kelas ini hanyalah sebuah POJO (*Plain Old Java Object*) yang tugasnya menyimpan data, lengkap dengan *constructors*, *getters*, dan *setters*.

#### 2. Repository (com.activa.repositories.MemberRepository)

* **Peran:** Jembatan antara aplikasi dan sumber data.
* **Dalam Kode:**
  + MemberRepository adalah sebuah *interface* (kontrak) yang mendefinisikan operasi data apa saja yang bisa dilakukan, seperti save, findById, findByNim, dan findAll.
  + InMemoryMemberRepository adalah implementasi nyata dari kontrak tersebut. Alih-alih terhubung ke database, ia menggunakan ArrayList untuk menyimpan objek Member di memori. Ini menyembunyikan kompleksitas penyimpanan data dari lapisan di atasnya.

#### 3. Service (com.activa.services.MemberService)

* **Peran:** Menjalankan logika bisnis (aturan main aplikasi).
* **Dalam Kode:** Kelas MemberService adalah "otak" dari operasi terkait member.
  + Metode registerNewMember(...) adalah contoh sempurna. Ia tidak langsung menyimpan data. Pertama, ia menjalankan **logika bisnis**: memeriksa apakah NIM sudah ada dengan memanggil memberRepository.findByNim(nim).
  + Jika NIM sudah ada, ia melempar IllegalStateException (sebuah aturan bisnis).
  + Jika aman, barulah ia membuat objek Member baru dan mendelegasikan tugas penyimpanan ke memberRepository.save(newMember).

#### 4. Controller (com.activa.controllers.MemberFormController)

* **Peran:** Mengelola interaksi antara pengguna (UI) dan aplikasi.
* **Dalam Kode:** MemberFormController adalah perantara.
  + Ia memiliki referensi ke komponen UI seperti nimField, nameField, dan statusLabel (ditandai dengan @FXML di aplikasi nyata).
  + Metode onSaveButtonClick() adalah *event handler* yang dipicu oleh aksi pengguna. Tugasnya hanya mengambil data dari UI, memanggil metode yang sesuai di MemberService, dan kemudian menampilkan hasilnya (sukses atau error) kembali ke UI melalui statusLabel.

### Siklus dan Alur Kerja Berdasarkan Kode

Mari kita lacak alur kerja saat pengguna menekan tombol "Simpan" di aplikasi kita:

**Langkah 1: View → Controller**

* Pengguna menekan tombol, memicu metode onSaveButtonClick() di MemberFormController.
* Controller mengumpulkan input dari nimField.getText(), nameField.getText(), dll.

**Langkah 2: Controller → Service**

* Controller tidak melakukan validasi atau penyimpanan. Ia langsung memanggil service:  
  memberService.registerNewMember(nim, name, birthdate, address);

**Langkah 3: Service → Repository (Logika Bisnis)**

* MemberService menerima data. Ia menjalankan logika bisnis pertama:

1. Optional<Member> existingMember = memberRepository.findByNim(nim);

2. if (existingMember.isPresent()) {

3. throw new IllegalStateException("NIM " + nim + " sudah terdaftar.");

4. }

* Jika NIM unik, Service membuat objek Member baru dan memanggil repository untuk menyimpannya:

1. Member newMember = new Member(nim, name, birthdate, address);

2. return memberRepository.save(newMember);

**Langkah 4: Repository → Sumber Data (Memori)**

* InMemoryMemberRepository menerima objek newMember. Metode save() di dalamnya menambahkan objek ini ke ArrayList internal yang bertindak sebagai database palsu.

**Langkah 5: Siklus Balik (Menampilkan Hasil)**

* Jika registerNewMember berhasil, ia mengembalikan objek Member yang baru dibuat.
* MemberFormController menangkap hasil ini dan memperbarui UI untuk memberi tahu pengguna:

1. statusLabel.setText("Sukses: Member " + registeredMember.getName() + " berhasil disimpan!");
2. memberTable.refresh(); // Memberi sinyal ke tabel untuk memuat ulang data

* Jika MemberService melempar IllegalStateException, blok catch di controller akan menangkapnya dan menampilkan pesan error ke pengguna:

1. statusLabel.setText("Error: " + e.getMessage());

Dengan alur ini, setiap bagian kode memiliki tanggung jawab yang jelas, membuat aplikasi lebih terstruktur, mudah diuji, dan mudah dikembangkan lebih lanjut.