

**Nama:** Raihana Aisha Az-Zahra

**NIM:** 241511056

**Prodi/Kelas:** D3/2B

## **Tugas – 2: Object & Class**

### **Deskripsi Aplikasi**

Aplikasi ini adalah program sederhana berbasis Java Console untuk mengelola data Karyawan dan Departemen di sebuah perusahaan.

### **Fitur Utama:**

#### **1. Tambah Departemen**

- User dapat memasukkan jumlah departemen yang ingin ditambahkan
- Setiap departemen memiliki ID otomatis, nama departemen, dan deskripsi
- Data departemen disimpan dalam **ArrayList departemenList**

#### **2. Tambah Karyawan**

- User dapat memasukkan jumlah karyawan yang ingin ditambahkan
- Setiap karyawan memiliki ID otomatis, nama karyawan, gaji, dan harus dipetakan ke salah satu departemen yang sudah ada.
- Data karyawan disimpan dalam **ArrayList karyawanList**

#### **3. Lihat Data**

- Menampilkan semua data Departemen dan Karyawan yang telah dimasukkan.
- Data karyawan ditampilkan lengkap dengan informasi departemen yang menaunginya.

### **Konsep yang Digunakan:**

#### **1. Encapsulation**

Semua atribut di class Karyawan dan Departemen bersifat private, hanya bisa diakses melalui getter dan setter.

#### **2. Auto Increment ID**

Atribut id pada Karyawan dan Departemen dibuat otomatis bertambah dengan bantuan variabel static counter.

#### **3. ArrayList**

Digunakan untuk menyimpan data dinamis (jumlah karyawan/departemen tidak dibatasi).

#### 4. Menu Looping

Menu utama berjalan berulang hingga user memilih keluar.

#### Penjelasan Program

Gambar	Konsep/Fitur	Deskripsi
<div>Karyawan.java</div> <pre>public class Karyawan{     private static int counter = 1;     private int id;     private String name;     private Departemen departemen;     private int gaji;      // Constructor     public Karyawan() {         this.id = counter++;     }      // Getter     public int getId() {         return id;     }      public String getName(){         return name;     }      public Departemen getDepartemen(){         return departemen;     }      public int getGaji(){         return gaji;     }      // Setter     public void setName(String newName){         name = newName;     }     public void setDepartemen(Departemen newDepartemen){         departemen = newDepartemen;     }     public void setGaji(int newGaji){         gaji = newGaji;     } }</pre> <div>Departemen.java</div>	Enkapsulasi	<p>Semua atribut (id, name, departemen, gaji, desc) dibuat private.</p> <p>Akses ke atribut hanya bisa lewat getter dan setter.</p>

<pre>public class Departemen{     private static int counter = 1;     private int id;     private String name;     private String desc;      // Constructor     public Departemen(){         this.id = counter++;     }      // Getter     public int getId() {         return id;     }     public String getName() {         return name;     }     public String getDesc() {         return desc;     }      // Setter     public void setName(String newName){         name = newName;     }     public void setDesc(String newDesc){         desc = newDesc;     } }</pre>		
<p>Karyawan.java</p> <pre>// Constructor public Karyawan() {     this.id = counter++; }</pre> <p>Departemen.java</p> <pre>// Constructor public Departemen(){     this.id = counter++; }</pre>	<p>Constructor</p>	<p>Constructor digunakan untuk menginisialisasi id karyawan dan id departemen secara otomatis dengan auto increment.</p>
<p>Karyawan.java</p> <pre>// Display public void display() {     System.out.println("ID: " + id);     System.out.println("Name: " + name);     if (departemen != null) {         System.out.println("Departemen: " + departemen.getName());         System.out.println("Deskripsi: " + departemen.getDesc());     }     System.out.println("Gaji: " + gaji);     System.out.println("-----"); }</pre> <p>Departemen.java</p>	<p>Method/Behavior</p>	<p>Method display() dipakai untuk menampilkan data karyawan dan departemen dengan format rapi.</p> <p>Hal Ini merupakan contoh <b>encapsulation</b></p>

<pre>// Display public void display() {     System.out.println("ID: " + id);     System.out.println("Name: " + name);     System.out.println("Deskripsi: " + desc);     System.out.println("-----"); }</pre>		<p>+ <b>behavior:</b> objek menyimpan datanya sendiri, lalu bisa menampilkan datanya sendiri.</p>
<p>Main.java</p> <pre>// Inisiasi Object Scanner scanner = new Scanner(System.in); ArrayList&lt;Karyawan&gt; karyawanList = new ArrayList&lt;&gt;(); ArrayList&lt;Departemen&gt; departemenList = new ArrayList&lt;&gt;(); int jumlahKaryawan, jumlahDepartemen, pilihDept, pilih;</pre>	<p>Inisialisasi object &amp; variable</p>	<p>scanner =&gt; untuk input dari user</p> <p>karyawanList =&gt; menyimpan semua data karyawan</p> <p>departemenList =&gt; menyimpan semua data departemen</p>
<p>Main.java</p> <pre>// Menu Aplikasi do {     System.out.println("\nSelamat Datang di Aplikasi Manajemen Karyawan!");     System.out.println("\n1. Tambah Departemen");     System.out.println("2. Tambah Karyawan");     System.out.println("3. Lihat Data");     System.out.println("4. Keluar");     System.out.print("Silakan Pilih Menu: ");     pilih = scanner.nextInt();     scanner.nextLine();      switch (pilih) {         case 1 -&gt; { ...         case 2 -&gt; {             System.out.print("Masukan jumlah karyawan: ");             jumlahKaryawan = scanner.nextInt();             scanner.nextLine();              for (int i = 0; i &lt; jumlahKaryawan; i++) { ...             }         case 3 -&gt; { ...         case 4 -&gt; {             System.out.println("Keluar dari program...");         }         default -&gt; { ...         }     } while (pilih != 4);</pre>	<p>Looping menu utama</p>	<p>Program berjalan terus-menerus sampai user memilih Keluar (4)</p>
<p>Main.java</p>	<p>Case 1: Tambah Departemen</p>	<p>User bisa menambahkan beberapa departemen sekaligus</p>

<pre> switch (pilih) {     case 1 -&gt; {         System.out.print("Masukan jumlah departemen: ");         jumlahDepartemen = scanner.nextInt();         scanner.nextLine();          for (int i = 0; i &lt; jumlahDepartemen; i++) {             Departemen departemen = new Departemen();             System.out.println("\nDepartemen ke - " + (i + 1));              System.out.print("Nama Departemen: ");             departemen.setName(scanner.nextLine());              System.out.print("Deskripsi: ");             departemen.setDesc(scanner.nextLine());              departemenList.add(departemen);         }     } } </pre>		<p>Data yang diminta untuk input adalah nama departemen dan deskripsinya</p> <p>Objek departemen baru dibuat lalu ditambahkan ke departemenList</p>
<p>Main.java</p> <pre> case 2 -&gt; {     System.out.print("Masukan jumlah karyawan: ");     jumlahKaryawan = scanner.nextInt();     scanner.nextLine();      for (int i = 0; i &lt; jumlahKaryawan; i++) {         Karyawan karyawan = new Karyawan();         System.out.println("\nKaryawan ke - " + (i + 1));          System.out.print("Nama Karyawan: ");         karyawan.setName(scanner.nextLine());          System.out.print("Gaji: ");         karyawan.setGaji(scanner.nextInt());         scanner.nextLine();          // Pilih Departemen         System.out.println("Pilih Departemen: ");         for (int j = 0; j &lt; departemenList.size(); j++) {             System.out.println((j + 1) + ". " + departemenList.get(j).getName());         }         System.out.print("Masukan nomor departemen: ");         pilihDept = scanner.nextInt() - 1;         scanner.nextLine();          if (pilihDept &gt;= 0 &amp;&amp; pilihDept &lt; departemenList.size()) {             karyawan.setDepartemen(departemenList.get(pilihDept));         }         karyawanList.add(karyawan);     } } </pre>	<p>Case 2: Tambah Karyawan</p>	<p>User bisa menambahkan beberapa karyawan</p> <p>Data yang diminta adalah nama dan gaji</p> <p>Lalu user diminta memilih departemen dari daftar departemenList</p> <p>Setelah itu, karyawan ditambahkan ke karyawanList</p>
<p>Main.java</p> <pre> case 3 -&gt; {     System.out.println("===== Data Departemen =====");     for (Departemen departemen : departemenList) {         departemen.display();     }     System.out.println("===== Data Karyawan =====");     for (Karyawan karyawan : karyawanList) {         karyawan.display();     } } </pre>	<p>Case 3: Melihat data karyawan dan data departemen</p>	<p>Pemanggilan display() menampilkan detail objek masing-masing.</p>

## Hasil

Gambar	Fitur
--------	-------

<p>Selamat Datang di Aplikasi Manajemen Karyawan!</p> <ol style="list-style-type: none"> <li>1. Tambah Departemen</li> <li>2. Tambah Karyawan</li> <li>3. Lihat Data</li> <li>4. Keluar</li> </ol> <p>Silakan Pilih Menu:</p>	<p>Menu utama</p>
<p>Masukan jumlah departemen: 2</p> <p>Departemen ke - 1 Nama Departemen: R&amp;D Deskripsi: Riset dan Pengembangan</p> <p>Departemen ke - 2 Nama Departemen: Pemasaran Deskripsi: Memasarkan Produk</p>	<p>Tambah Departemen</p>
<p>Masukan jumlah karyawan: 2</p> <p>Karyawan ke - 1 Nama Karyawan: Alice Gaji: 500000 Pilih Departemen: <ol style="list-style-type: none"><li>1. R&amp;D</li><li>2. Pemasaran</li></ol>Masukan nomor departemen: 2</p> <p>Karyawan ke - 2 Nama Karyawan: Bob Gaji: 560000 Pilih Departemen: <ol style="list-style-type: none"><li>1. R&amp;D</li><li>2. Pemasaran</li></ol>Masukan nomor departemen: 1</p>	<p>Tambah Karyawan</p>

<pre> Silakan Pilih Menu: 3 ===== Data Departemen ===== ID: 1 Name: R&amp;D Deskripsi: Riset dan Pengembangan ----- ID: 2 Name: Pemasaran Deskripsi: Memasarkan Produk ----- ===== Data Karyawan ===== ID: 1 Name: Alice Departemen: Pemasaran Deskripsi: Memasarkan Produk Gaji: 500000 ----- ID: 2 Name: Bob Departemen: R&amp;D Deskripsi: Riset dan Pengembangan Gaji: 560000 ----- </pre>	<p>Menampilkan data departemen dan data karyawan</p>
--	--

## Lesson Learned

### 1. Praktik Langsung Lebih Efektif

Dengan mencoba langsung membuat class dan object, pemahaman jadi lebih mendalam dibanding hanya membaca teori. Praktik nyata membantu mengerti alur bagaimana object dibuat, bagaimana atribut dan method saling berhubungan, serta bagaimana konsep OOP diterapkan dalam kode.

### 2. Pemahaman Getter dan Setter

Saya jadi paham bahwa getter digunakan untuk mengambil nilai atribut secara aman, sementara setter dipakai untuk mengubah nilai atribut dengan kendali tertentu. Konsep ini membantu menjaga prinsip enkapsulasi, supaya atribut tidak bisa diakses/diubah sembarangan dari luar class.

### 3. Tidak Semua Atribut Perlu Setter

Dari implementasi class Karyawan, saya belajar bahwa tidak semua atribut harus bisa diubah. Contohnya id, saya membuatnya auto increment dengan counter static. Artinya setiap object Karyawan otomatis punya id unik, dan nilainya tidak boleh diubah manual. Hal ini menegaskan pentingnya access modifier dan perancangan atribut sesuai kebutuhan.

#### **4. Eksplorasi Object dalam Array dan ArrayList**

Saya menemukan bahwa object tidak hanya dibuat satu per satu, tetapi juga dapat disimpan dalam bentuk Array maupun ArrayList. Array cocok untuk jumlah data yang tetap, sedangkan ArrayList lebih fleksibel karena ukurannya bisa berubah. Dari sini saya belajar bagaimana mengelola Kumpulan object dengan lebih dinamis

#### **Referensi**

<https://www.geeksforgeeks.org/java/how-to-create-array-of-objects-in-java/>

[https://www.w3schools.com/java/java\\_arraylist.asp](https://www.w3schools.com/java/java_arraylist.asp)

<https://stackoverflow.com/questions/3982550/creating-an-arraylist-of-objects>