

LAPORAN
WEEK 3 PEMROGRAMAN BERBASIS OBJEK

Dibuat untuk memenuhi salah satu tugas mata kuliah Pemrograman Berbasis Objek yang diampu
oleh Bapak Ardhian Ekawijana, M.T.

Oleh:

Nama	: Raihana Aisha Az-Zahra
NIM	: 241511056
Kelas	: 2B
Program Studi	: D3 Teknik Informatika
Jurusan	: Teknik Komputer dan Informatika

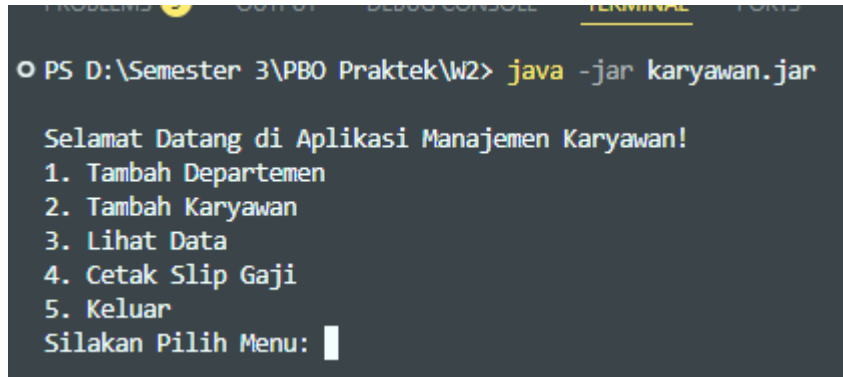


POLITEKNIK NEGERI BANDUNG
KOTA BANDUNG
2025

BAB I

HASIL Pengerjaan

Link Github: <https://github.com/raihanaiooo/PBO-Praktek-Sem3>



```
PS D:\Semester 3\PBO Praktek\W2> java -jar karyawan.jar

Selamat Datang di Aplikasi Manajemen Karyawan!
1. Tambah Departemen
2. Tambah Karyawan
3. Lihat Data
4. Cetak Slip Gaji
5. Keluar
Silakan Pilih Menu: █
```

Berhasil membuat file jar dan di run

A. Struktur Package

Program dibagi ke dalam dua package utama:

- id.ac.polban.model → berisi class Karyawan dan Departemen
- id.ac.polban.service → berisi class KaryawanService

B. Implementasi Program

- Class Karyawan (id.ac.polban.model.Karyawan)

Class karyawan merepresentasikan data karyawan, dimulai dari id, nama, gaji, dan departemen. Memiliki getter, setter, inputData, dan method display untuk menampilkan detail karyawan.

- Class Departemen (id.ac.polban.model.Departemen)

Class Departemen merepresentasikan data departemen, dimulai dari id, nama, dan deskripsi. Memiliki getter, setter, inputData, dan method display untuk menampilkan detail departemen

- Class KaryawanService (id.ac.polban.service.KaryawanService)

Bertugas mengelola proses tambahDepartemen(), tambahKaryawan(), tampilkanData(), cetakSlipGajiMenu()

- **Aggregation** → ArrayList<Karyawan> dan ArrayList<Departemen> disimpan terpisah dari service. Artinya, Karyawan & Departemen tetap bisa hidup walau service dihentikan

- **Dependency** → Method cetakSlipGaji(Karyawan karyawan) bergantung langsung pada objek Karyawan.
- **Class main**
Bertugas sebagai entry point aplikasi, menyediakan menu interaktif, memanggil service untuk melakukan operasi Tambah Departemen, Tambah Karyawan, Lihat Data, Cetak Slip Gajis

C. Relasi Antar Kelas

- **Aggregation**
main dan KaryawanService menyimpan banyak objek Karyawan dan Departemen dalam ArrayList. Tapi kalau main berhenti, objek Karyawan & Departemen masih bisa hidup di memori atau digunakan lagi.
- **Dependency**
karyawanService bergantung pada Karyawan saat mencetak slip gaji. main bergantung pada KaryawanService untuk menjalankan menu.
- **Static**
counter di Karyawan dan Departemen digunakan untuk memberi nomor ID unik otomatis. Di main, variable scanner, karyawanList, departemenList, dan service dideklarasikan static agar bisa dipakai oleh semua method.

D. Static Field dan Method

- **private static int counter = 1;** → artinya nilainya dibagi bersama oleh semua objek Karyawan, bukan milik satu objek saja.
- **private static final Scanner scanner = new Scanner(System.in);** → agar tidak perlu bikin objek Scanner baru setiap kali input, cukup pakai yang ini.
- **private static final ArrayList<Karyawan> karyawanList = new ArrayList<>();** → menjaga supaya data karyawan konsisten dan bisa diakses dari mana saja dalam program.
- **private static final ArrayList<Departemen> departemenList = new ArrayList<>();** → jadi “wadah global” departemen untuk aplikasi.
- **private static final KaryawanService service = new KaryawanService();** → service ini berisi method (tambah, lihat, cetak) dan dijadikan helper tetap untuk menjalankan menu.

E. Deployment JAR File

Langkah-langkah pembuatan jar:

1. Kompilasi kode ke folder bin

```
javac -d bin id\ac\polban\model\*.java id\ac\polban\service\*.java main.java
```

Mengompilasi semua file .java dan menyimpan file .class ke folder bin.

2. Buat file manifest.txt

Isinya menentukan kelas utama (entry point aplikasi)

Main-Class: main

(Pastikan ada enter di akhir baris)

3. Buat file jar

```
jar cfm karyawan.jar manifest.txt -C bin .
```

Mengemas semua file .class di dalam folder bin menjadi employee.jar dan menyertakan manifest.

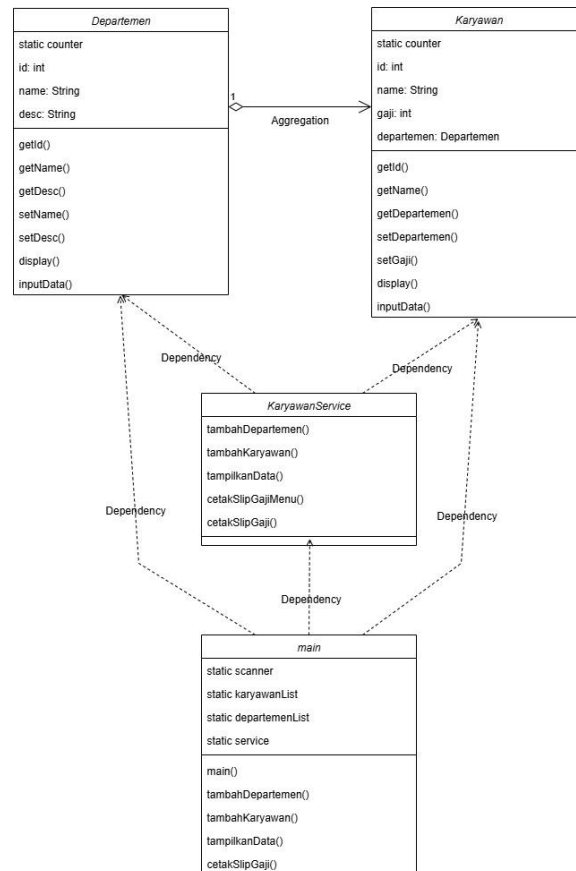
4. Jalankan jar

```
java -jar karyawan.jar
```

Menjalankan aplikasi langsung melalui file employee.jar tanpa perlu menyebutkan classpath.

BAB II

CLASS DIAGRAM



1. Relasi Departemen dengan Karyawan

Class **Departemen** berelasi dengan class **Karyawan** melalui hubungan **aggregation**. Artinya, sebuah departemen dapat memiliki banyak karyawan yang tergabung di dalamnya, namun objek karyawan tetap bisa ada meskipun departemen dihapus. Hal ini mencerminkan situasi nyata di mana karyawan bisa saja berpindah atau tetap ada walaupun departemennya tidak ada lagi.

2. Relasi KaryawanService dengan Departemen dan Karyawan

Class **KaryawanService** memiliki hubungan **dependency** terhadap class **Departemen** dan **Karyawan**. Hal ini karena service membutuhkan data dari departemen maupun karyawan untuk menjalankan fungsinya, seperti menambah karyawan baru ke dalam suatu

departemen, menampilkan daftar karyawan, atau memperbarui informasi. Tanpa keberadaan objek Departemen dan Karyawan, service ini tidak dapat berfungsi dengan baik.

3. Relasi Main dengan KaryawanService

Class Main memiliki relasi *dependency* dengan KaryawanService karena seluruh logika utama program yang berhubungan dengan pengelolaan data karyawan dan departemen dilakukan melalui service ini. Main bertugas menerima input dari pengguna, kemudian memanggil method yang ada di dalam KaryawanService agar data dapat diproses sesuai kebutuhan.

4. Relasi Main dengan Karyawan dan Departemen

Selain melalui KaryawanService, class Main juga berelasi langsung dengan Karyawan dan Departemen karena menyimpan daftar objek dari kedua class tersebut dalam bentuk koleksi (seperti ArrayList). Relasi ini menunjukkan bahwa Main tidak hanya mengandalkan service, tetapi juga mengelola penyimpanan objek karyawan dan departemen secara langsung agar data dapat diakses dan dimodifikasi sesuai kebutuhan.

BAB III

LESSON LEARNED

Dari implementasi program manajemen karyawan dan departemen ini, saya belajar bahwa setiap class memiliki peran dan relasi yang berbeda. Class **Departemen** dan **Karyawan** menunjukkan konsep *aggregation* karena objek karyawan bisa tetap ada meskipun departemennya dihapus. Class **KaryawanService** memperlihatkan pentingnya *dependency*, yaitu bagaimana sebuah service memanfaatkan class lain untuk menjalankan fungsinya. Class **Main** berperan sebagai pengelola alur utama program yang menghubungkan input pengguna dengan service maupun data. Selain itu, saya juga memahami kegunaan **static field dan method** yang memungkinkan perhitungan atau pemanggilan data secara global tanpa harus membuat objek baru. Dengan begitu, konsep OOP seperti *aggregation*, *dependency*, *static field/method*, dan penggunaan *package* dapat diterapkan secara nyata dalam program sederhana ini.