**LAPORAN TUGAS BESAR**

**PEMROGRAMAN BERORIENTASI OBJEK**

**SISTEM PEGAWAI**

# HALAMAN JUDUL



**Disusun Oleh :**

**NAMA : Arfiana Maulidiyah**

**NIM : 32602200012**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INDUSTRI**

**UNIVERSITAS ISLAM SULTAN AGUNG**

**SEMARANG**

**2024**

# DAFTAR ISI

[**HALAMAN JUDUL** 1](#_Toc154989973)

[**DAFTAR ISI** ii](#_Toc154989974)

[**DAFTAR GAMBAR** iii](#_Toc154989975)

[**BAB I PENDAHULUAN** 1](#_Toc154989976)

[1.1 Latar belakang 1](#_Toc154989977)

[1.2 Tujuan 1](#_Toc154989978)

[**BAB II Konsep Dasar PBO** 2](#_Toc154989979)

[2.1 *Inheritance* 2](#_Toc154989980)

[2.2 Polimorfisme 2](#_Toc154989981)

[2.3 *Encapsulation* 2](#_Toc154989982)

[2.4 *Getter* dan *Setter* 2](#_Toc154989983)

[2.5 *Interface* 2](#_Toc154989984)

[**BAB III Struktur Program** 3](#_Toc154989985)

[3.1 *File* Utama ( ManajemenPegawaiGUI ) 3](#_Toc154989986)

[3.2 Kelas Pegawai 6](#_Toc154989987)

[3.3 Kelas *Manager* dan *Staff* 7](#_Toc154989988)

[3.4 *Interface* PengelolaPegawai 10](#_Toc154989989)

[3.5 Kelas ManajemenPegawaiGUI ( Main Class ) 11](#_Toc154989990)

[**BAB IV Implementasi Program** 15](#_Toc154989991)

[4.1 Menjalankan Program 15](#_Toc154989992)

[4.2 Menambahkan Produk 16](#_Toc154989993)

[**BAB V KESIMPULAN**](#_Toc154989994)

[**DAFTAR PUSTAKA**](#_Toc154989995)

# DAFTAR GAMBAR

[Gambar 3. 1 File Utama Program Main 1 3](#_Toc155477250)

[Gambar 3. 2 File Utama Program Main 2 3](#_Toc155477251)

[Gambar 3. 3 File Utama Program Main 3 4](#_Toc155477252)

[Gambar 3. 4 Kelas Pegawai 1 6](#_Toc155477253)

[Gambar 3. 5 Kelas Pegawai 2 6](#_Toc155477254)

[Gambar 3. 6 Kelas Manager 1 8](#_Toc155477255)

[Gambar 3. 7 Kelas Manager 2 8](#_Toc155477256)

[Gambar 3. 8 Kelas Staff 1 9](#_Toc155477257)

[Gambar 3. 9 Kelas Staff 2 9](#_Toc155477258)

[Gambar 3. 10 Interface Pengelola Pegawai 10](#_Toc155477259)

[Gambar 3. 11 MainPegawaiGUI 1 11](#_Toc155477260)

[Gambar 3. 12 MainPegawaiGUI 2 12](#_Toc155477261)

[Gambar 3. 13 MainPegawaiGUI 3 12](#_Toc155477262)

[Gambar 3. 14 MainPegawaiGUI 4 12](#_Toc155477263)

[Gambar 4. 1 Open Neatbeans 15](#_Toc155477264)

[Gambar 4. 2 Open Projects 15](#_Toc155477265)

[Gambar 4. 3 Arahkan ke project 15](#_Toc155477266)

[Gambar 4. 4 Open struktur project 16](#_Toc155477267)

[Gambar 4. 5 Run program 16](#_Toc155477268)

[Gambar 4. 6 Tunggu program muncul 16](#_Toc155477269)

[Gambar 4. 7 Menu Pegawai 17](#_Toc155477270)

[Gambar 4. 8 Menu Staff / Manager 17](#_Toc155477271)

[Gambar 4. 9 Isi Nama Pegawai 17](#_Toc155477272)

[Gambar 4. 10 Dialog sukses add pegawai 17](#_Toc155477273)

[Gambar 4. 11 Back to menu main 18](#_Toc155477274)

[Gambar 4. 12 Dialog show pegawai 18](#_Toc155477275)

[Gambar 4. 13 Back to menu main 18](#_Toc155477276)

# BAB I PENDAHULUAN

## Latar belakang

Perkembangan teknologi yang sangat cepat memberikan pengaruh yang sangat besar di berbagai aspek kehidupan manusia. Pengelolaan pegawai merupakan aspek krusial dalam suatu organisasi. Dalam rangka meningkatkan efisiensi dan keteraturan data pegawai, sistem manajemen pegawai berbasis pemrograman berorientasi objek menjadi solusi yang tepat. Pada laporan ini, akan dibahas implementasi sistem manajemen pegawai dengan menggunakan konsep dasar pemrograman berorientasi Objek.

## Tujuan

1. Untuk mengetahui dan sebagai media pembelajaran tentang pengaplikasian bahasa pemrograman berorientasi objek pada pengelolaan pegawai.
2. Untuk memberikan panduan langkah-langkah dalam menjalankan program dan menambahkan data pegawai
3. Untuk memenuhi tugas besar pemrograman berorientasi objek pada jurusan Teknik Informatika semester 3.

## Rumusan Masalah

1. Bagaimana pengaplikasian Bahasa pemrograman java pada pengelolaan data pegawai.

## Manfaat

1. Mempermudah penggunaan dalam melakukan perhitungan matematika sehari-hari
2. Menyediakan antarmuka yang intutif untuk pengalaman pengguna yang lebih baik.

# BAB II Konsep Dasar PBO

## *Inheritance*

Konsep *inheritance* diterapkan dalam pembuatan kelas Pegawai, *Manager*, dan *Staff*. Kelas *Manager* dan *Staff* merupakan turunan dari kelas Pegawai, mewarisi atribut dan metode dari kelas induknya.

## Polimorfisme

Polimorfisme diimplementasikan melalui penggunaan metode info() pada kelas Pegawai, yang kemudian di-override oleh kelas *Manager* dan *Staff* sesuai dengan kebutuhan masing-masing.

## *Encapsulation*

*Encapsulation* diaplikasikan dengan menyembunyikan implementasi internal kelas Pegawai dan memberikan akses melalui metode *getter* dan *setter*.

## *Getter* dan *Setter*

Metode *getter* dan *setter* digunakan untuk mengakses dan mengubah nilai atribut pada kelas Pegawai, *Manager*, dan *Staff* dengan prinsip *encapsulation*.

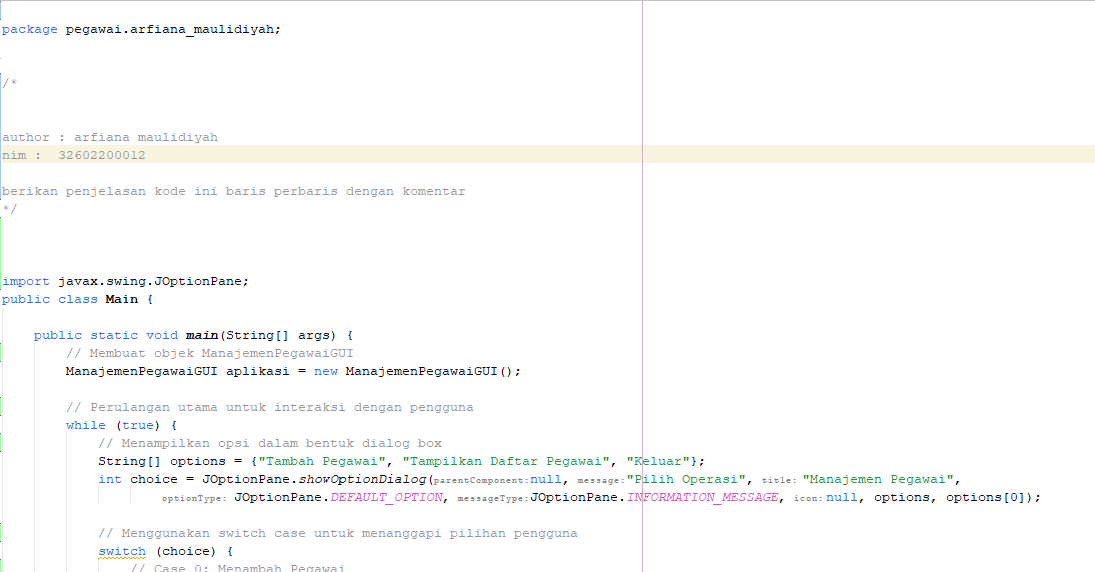
## *Interface*

*Interface* PengelolaPegawai diimplementasikan pada kelas ManajemenPegawaiGUI untuk memastikan bahwa metode untuk menambah pegawai dan menampilkan daftar pegawai telah diimplementasikan.

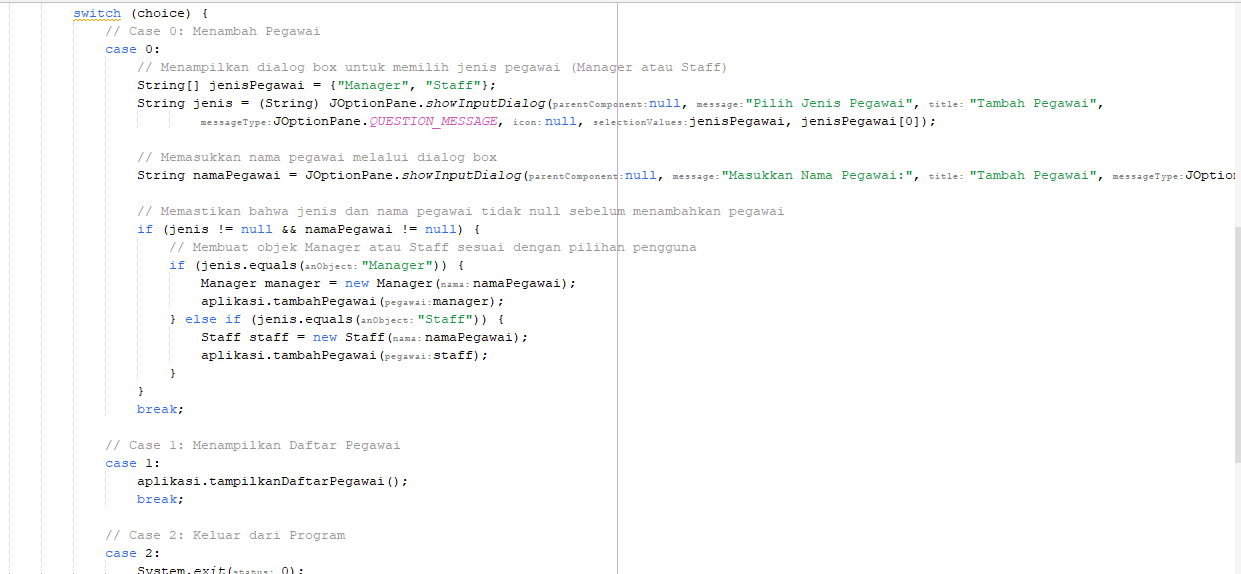
# BAB III Struktur Program

## *File* Utama ( ManajemenPegawaiGUI )

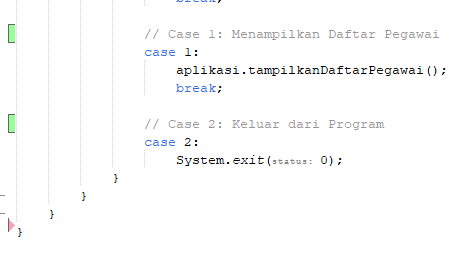
ManajemenPegawaiGUI merupakan kelas utama yang berfungsi sebagai antar muka pengguna. Dalam kelas ini, *user* dapat menambahkan pegawai dan menampilkan daftar pegawai..



Gambar 3. 1 File Utama Program Main 1



Gambar 3. 2 File Utama Program Main 2



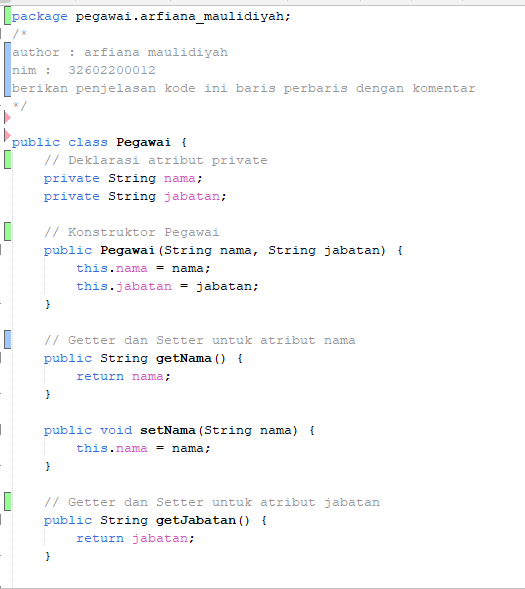
Gambar 3. 3 File Utama Program Main 3

Penjelasan Kode :

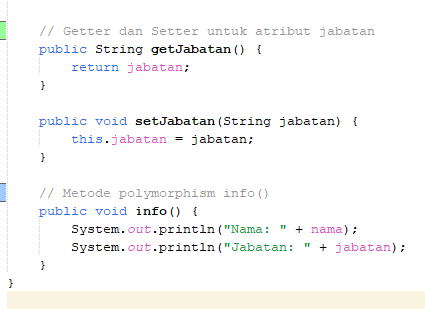
1. Package dan Import
2. package pegawai.arfiana\_maulidiyah;: Mendefinisikan package untuk kelas-kelas dalam program.
3. import javax.swing.JOptionPane;: Mengimpor kelas JOptionPane dari paket javax.swing untuk penggunaan dialog box.
4. Deklarasi Kelas Main
5. public class Main {: Mendeklarasikan kelas utama dengan nama Main.
6. Metode main
7. public static void main(String[] args) {: Mendeklarasikan metode main sebagai titik masuk utama program.
8. Inisialisasi Objek ManajemenPegawaiGUI
9. ManajemenPegawaiGUI aplikasi = new ManajemenPegawaiGUI();: Membuat objek ManajemenPegawaiGUI untuk mengelola pegawai.
10. Perulangan Utama
11. while (true) {: Membuat perulangan tanpa batas untuk terus berinteraksi dengan pengguna.
12. Opsi Pilihan Pengguna
13. String[] options = {"Tambah Pegawai", "Tampilkan Daftar Pegawai", "Keluar"};: Membuat array string berisi opsi pilihan pengguna.
14. Dialog Box Pilihan Operasi
15. int choice = JOptionPane.showOptionDialog(...);: Menampilkan dialog box opsi dan mengambil pilihan pengguna.
16. Switch Case untuk Memproses Pilihan Pengguna
17. switch (choice) {: Memulai struktur kontrol switch case untuk menanggapi pilihan pengguna.
18. Case 0: Menambah Pegawai
19. case 0:: Bagian untuk menambah pegawai.
20. String[] jenisPegawai = {"*Manager*", "*Staff*"};: Membuat array string berisi opsi jenis pegawai.
21. String jenis = (String) JOptionPane.showInputDialog(...);: Menampilkan dialog box untuk memilih jenis pegawai.
22. String namaPegawai = JOptionPane.showInputDialog(...);: Menampilkan dialog box untuk memasukkan nama pegawai.
23. if (jenis != null && namaPegawai != null) {: Memastikan input jenis dan nama pegawai tidak null.
24. if (jenis.equals("*Manager*")) { ... } else if (jenis.equals("*Staff*")) { ... }: Membuat objek *Manager* atau *Staff* berdasarkan pilihan pengguna.
25. Case 1: Menampilkan Daftar Pegawai
26. case 1:: Bagian untuk menampilkan daftar pegawai.
27. aplikasi.tampilkanDaftarPegawai();: Memanggil metode untuk menampilkan daftar pegawai.
28. Case 2: Keluar dari Program
29. case 2:: Bagian untuk keluar dari program.
30. System.exit(0);: Mengakhiri program.
31. Penutup Metode main
32. } // Akhir switch case: Menutup blok switch case.
33. } // Akhir while loop: Menutup blok while loop.
34. } // Akhir main method: Menutup blok main method.
35. } // Akhir kelas Main: Menutup blok kelas Main.

## Kelas Pegawai

Kelas Pegawai sebagai superclass berisi atribut dan metode umum yang digunakan oleh kelas turunannya, yaitu *Manager* dan *Staff*.



Gambar 3. 4 Kelas Pegawai 1



Gambar 3. 5 Kelas Pegawai 2

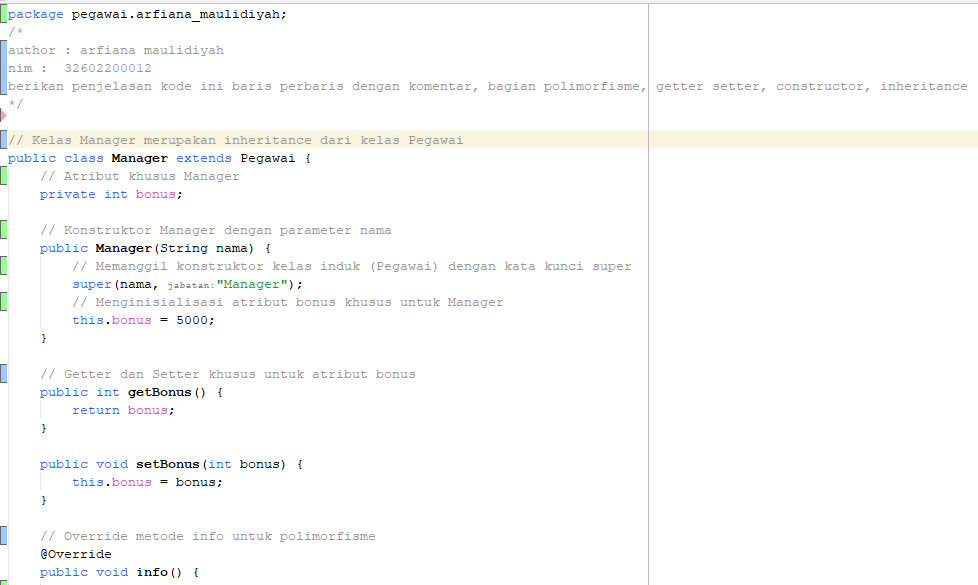
Penjelasan Kode :

1. Deklarasi Package dan Komentar Header
2. package pegawai.arfiana\_maulidiyah;: Mendefinisikan package tempat kelas Pegawai berada.
3. Komentar berisi informasi tentang penulis dan NIM.
4. Deklarasi Kelas Pegawai
   1. public class Pegawai {: Mendeklarasikan kelas Pegawai.
5. Atribut Kelas Pegawai
   1. private String nama;: Atribut private untuk menyimpan nama pegawai.
   2. private String jabatan;: Atribut private untuk menyimpan jabatan pegawai.
6. Konstruktor Kelas Pegawai
   1. public Pegawai(String nama, String jabatan) {: Konstruktor untuk inisialisasi objek Pegawai dengan nama dan jabatan.
7. *Getter* dan *Setter*
   1. public String getNama() { return nama; }: *Getter* untuk mendapatkan nilai atribut nama.
   2. public void setNama(String nama) { this.nama = nama; }: *Setter* untuk mengubah nilai atribut nama.
   3. public String getJabatan() { return jabatan; }: *Getter* untuk mendapatkan nilai atribut jabatan.
   4. public void setJabatan(String jabatan) { this.jabatan = jabatan; }: *Setter* untuk mengubah nilai atribut jabatan.
8. Metode Polymorphism info()

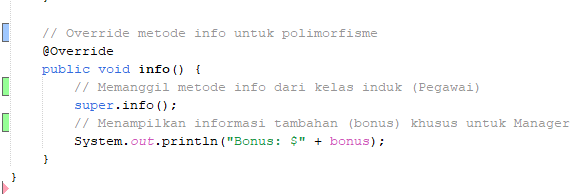
a. public void info() { ... }: Metode polymorphism yang menampilkan informasi nama dan jabatan pegawai.

## Kelas *Manager* dan *Staff*

Kelas *Manager* dan *Staff* sebagai subclass dari Pegawai, mewarisi atribut dan metode dari kelas Pegawai dan menambahkan atribut khusus masing-masing..



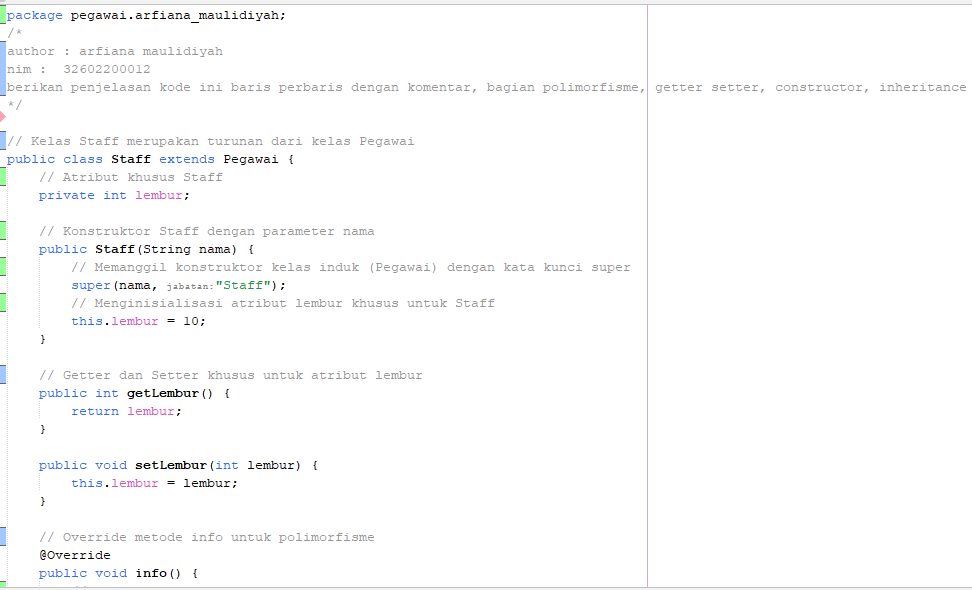
Gambar 3. 6 Kelas Manager 1



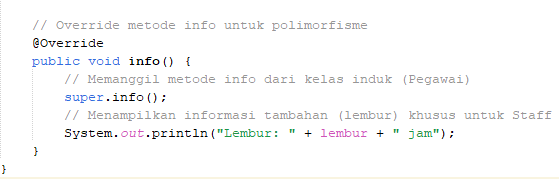
Gambar 3. 7 Kelas Manager 2

* 1. Package dan Komentar Header

1. Komentar berisi informasi tentang penulis dan NIM.
   1. Deklarasi Kelas *Manager* dan *Inheritance*
2. public class *Manager* extends Pegawai {: Mendeklarasikan kelas *Manager* yang merupakan turunan dari kelas Pegawai.
   1. Atribut Kelas *Manager*
3. private int bonus;: Atribut khusus untuk *Manager*.
   1. Konstruktor Kelas *Manager*
4. public *Manager*(String nama) { ... }: Konstruktor *Manager* dengan parameter nama.
5. super(nama, "*Manager*");: Memanggil konstruktor kelas induk (Pegawai) dengan kata kunci super.
6. this.bonus = 5000;: Menginisialisasi atribut bonus khusus untuk *Manager*.
   1. *Getter* dan *Setter* untuk Atribut Bonus
7. public int getBonus() { return bonus; }: *Getter* untuk mendapatkan nilai atribut bonus.
8. public void setBonus(int bonus) { this.bonus = bonus; }: *Setter* untuk mengubah nilai atribut bonus.
   1. Override Metode info() untuk Polimorfisme
9. @Override: Menandakan bahwa metode berikut ini di-override dari kelas induk.
10. super.info();: Memanggil metode info dari kelas induk (Pegawai).
11. System.out.println("Bonus: $" + bonus);: Menampilkan informasi tambahan (bonus) khusus untuk *Manager*.



Gambar 3. 8 Kelas Staff 1



Gambar 3. 9 Kelas Staff 2

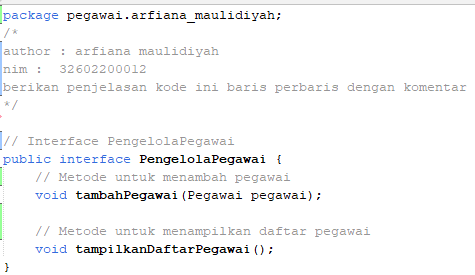
Penjelasan Kode :

* 1. Package dan Komentar Header

1. Komentar berisi informasi tentang penulis dan NIM.
   1. Deklarasi Kelas *Staff* dan *Inheritance*
2. public class *Staff* extends Pegawai {: Mendeklarasikan kelas *Staff* yang merupakan turunan dari kelas Pegawai.
   1. Atribut Kelas *Staff*
3. private int lembur;: Atribut khusus untuk *Staff*.
   1. Konstruktor Kelas *Staff*
4. public *Staff*(String nama) { ... }: Konstruktor *Staff* dengan parameter nama.
5. super(nama, "*Staff*");: Memanggil konstruktor kelas induk (Pegawai) dengan kata kunci super.
6. this.lembur = 10;: Menginisialisasi atribut lembur khusus untuk *Staff*.
   1. *Getter* dan *Setter* untuk Atribut Lembur
7. public int getLembur() { return lembur; }: *Getter* untuk mendapatkan nilai atribut lembur.
8. public void setLembur(int lembur) { this.lembur = lembur; }: *Setter* untuk mengubah nilai atribut lembur.
   1. Override Metode info() untuk Polimorfisme
9. @Override: Menandakan bahwa metode berikut ini di-override dari kelas induk.
10. super.info();: Memanggil metode info dari kelas induk (Pegawai).
11. System.out.println("Lembur: " + lembur + " jam");: Menampilkan informasi tambahan (lembur) khusus untuk *Staff*.

## *Interface* PengelolaPegawai

*Interface* yang memuat metode untuk menambah pegawai dan menampilkan daftar pegawai..



Gambar 3. 10 Interface Pengelola Pegawai

Penjelasan Kode :

1. Package dan Komentar Header

Komentar berisi informasi tentang penulis dan NIM.

1. Deklarasi *Interface* PengelolaPegawai

public *interface* PengelolaPegawai {: Mendeklarasikan sebuah *interface* dengan nama PengelolaPegawai.

1. Metode tambahPegawai

void tambahPegawai(Pegawai pegawai);: Metode untuk menambahkan objek Pegawai ke dalam suatu implementasi.

1. Metode tampilkanDaftarPegawai

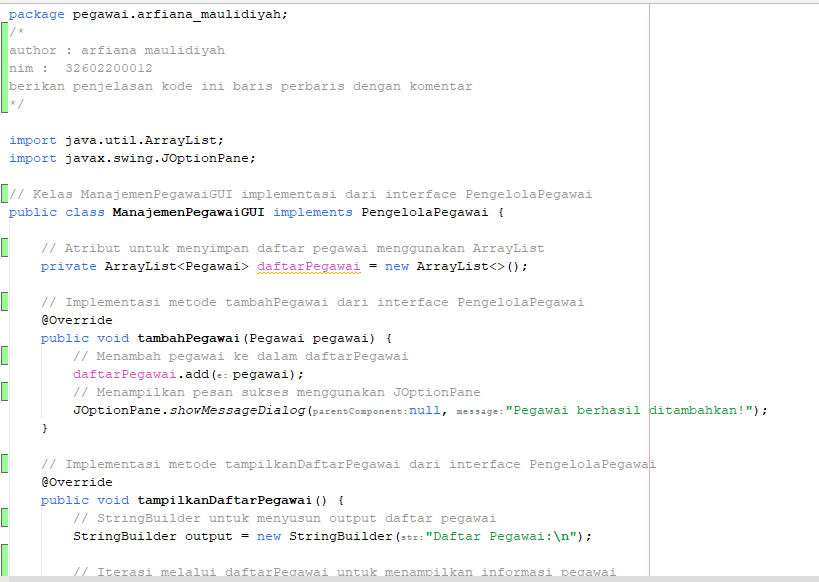
void tampilkanDaftarPegawai();: Metode untuk menampilkan daftar pegawai dalam suatu implementasi.

1. Penutup *Interface*

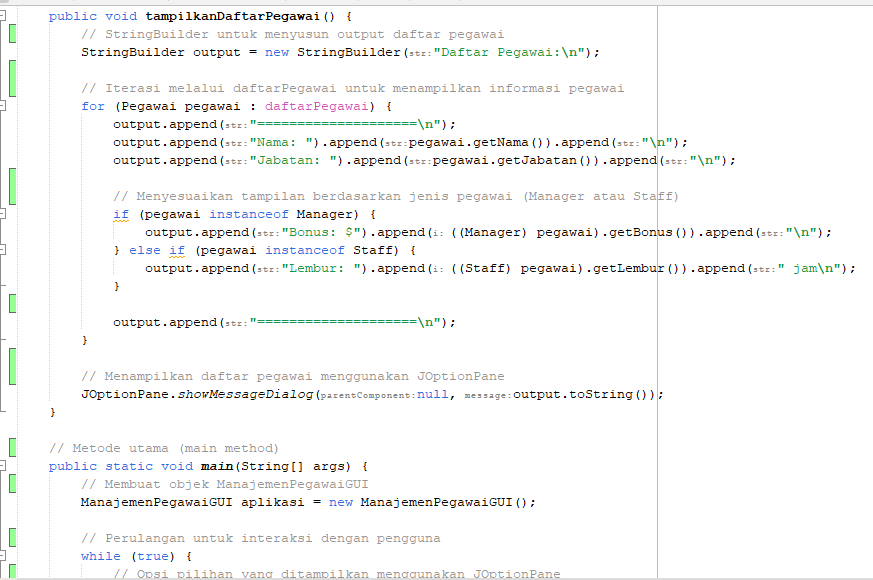
}: Menutup deklarasi *interface* PengelolaPegawai.

## Kelas ManajemenPegawaiGUI ( Main Class )

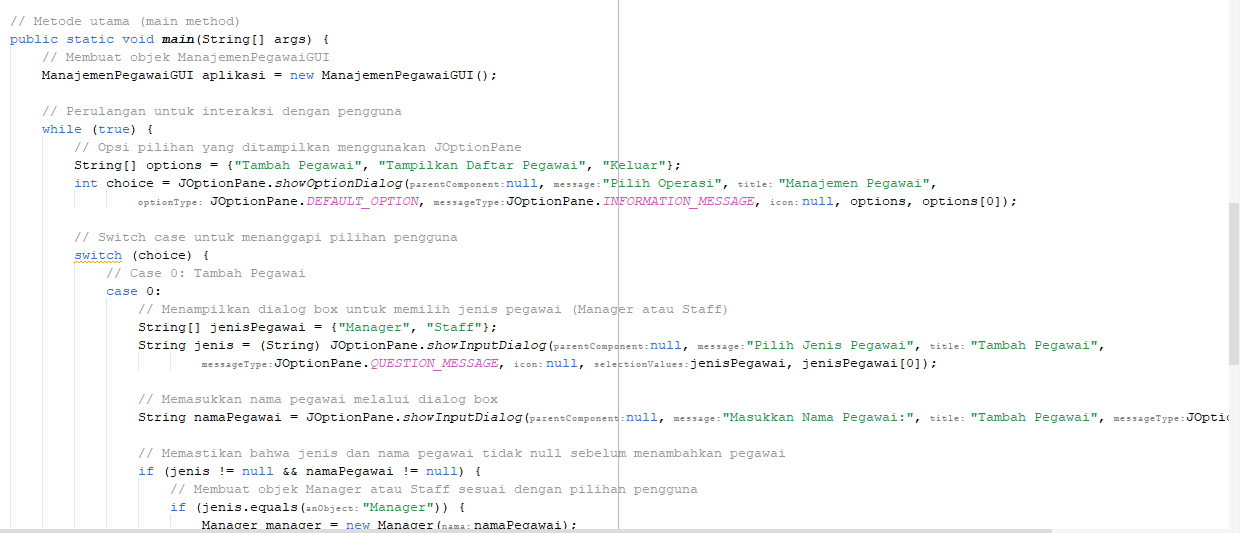
Kelas ini berfungsi sebagai antarmuka utama yang mengintegrasikan seluruh sistem manajemen pegawai. Pengguna dapat memilih operasi untuk menambah pegawai atau menampilkan daftar pegawai.



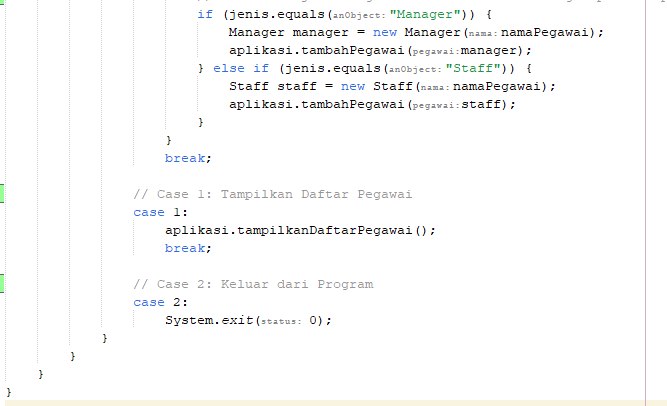
Gambar 3. 11 MainPegawaiGUI 1



Gambar 3. 12 MainPegawaiGUI 2



Gambar 3. 13 MainPegawaiGUI 3



Gambar 3. 14 MainPegawaiGUI 4

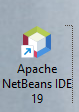
Penjelasan Kode :

1. Package dan Komentar Header
   1. Komentar berisi informasi tentang penulis dan NIM.
2. Import Paket
3. import java.util.ArrayList;: Mengimpor kelas ArrayList dari paket java.util.
4. import javax.swing.JOptionPane;: Mengimpor kelas JOptionPane dari paket javax.swing.
5. Deklarasi Kelas ManajemenPegawaiGUI
6. public class ManajemenPegawaiGUI implements PengelolaPegawai {: Mendeklarasikan kelas ManajemenPegawaiGUI yang mengimplementasi interface PengelolaPegawai.
7. Atribut Kelas ManajemenPegawaiGUI
   1. private ArrayList<Pegawai> daftarPegawai = new ArrayList<>();: Membuat ArrayList untuk menyimpan daftar pegawai.
8. Implementasi Metode dari Interface
9. @Override: Menandakan bahwa metode berikut diimplementasi dari interface.
10. void tambahPegawai(Pegawai pegawai) { ... }: Menambahkan pegawai ke dalam daftar dan menampilkan pesan dengan JOptionPane.
11. void tampilkanDaftarPegawai() { ... }: Menampilkan daftar pegawai dengan informasi sesuai jenis pegawai menggunakan JOptionPane.
12. Metode Main
13. public static void main(String[] args) { ... }: Metode utama untuk menjalankan aplikasi.
14. ManajemenPegawaiGUI aplikasi = new ManajemenPegawaiGUI();: Membuat objek ManajemenPegawaiGUI.
15. Perulangan while (true) untuk terus berinteraksi dengan pengguna.
16. String[] options = {"Tambah Pegawai", "Tampilkan Daftar Pegawai", "Keluar"};: Opsi pilihan yang ditampilkan menggunakan JOptionPane.
17. int choice = JOptionPane.showOptionDialog(...);: Meminta pengguna untuk memilih operasi menggunakan JOptionPane.
18. Switch case untuk menanggapi pilihan pengguna.
19. Case 0: Menambah Pegawai
20. String[] jenisPegawai = {"Manager", "Staff"};: Menampilkan dialog box untuk memilih jenis pegawai.
21. if (jenis.equals("Manager")) { ... } else if (jenis.equals("Staff")) { ... }: Membuat objek Manager atau Staff sesuai dengan pilihan pengguna.
22. Case 1: Menampilkan Daftar Pegawai
23. aplikasi.tampilkanDaftarPegawai();: Memanggil metode tampilkanDaftarPegawai() untuk menampilkan daftar pegawai.
24. Case 2: Keluar dari Program
25. System.exit(0);: Keluar dari program jika pengguna memilih opsi keluar.

# BAB IV Implementasi Program

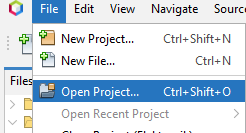
## Menjalankan Program

1. Untuk menjalankan program, pertama klik ikon aplikasi neatbeans pada layer dekstop

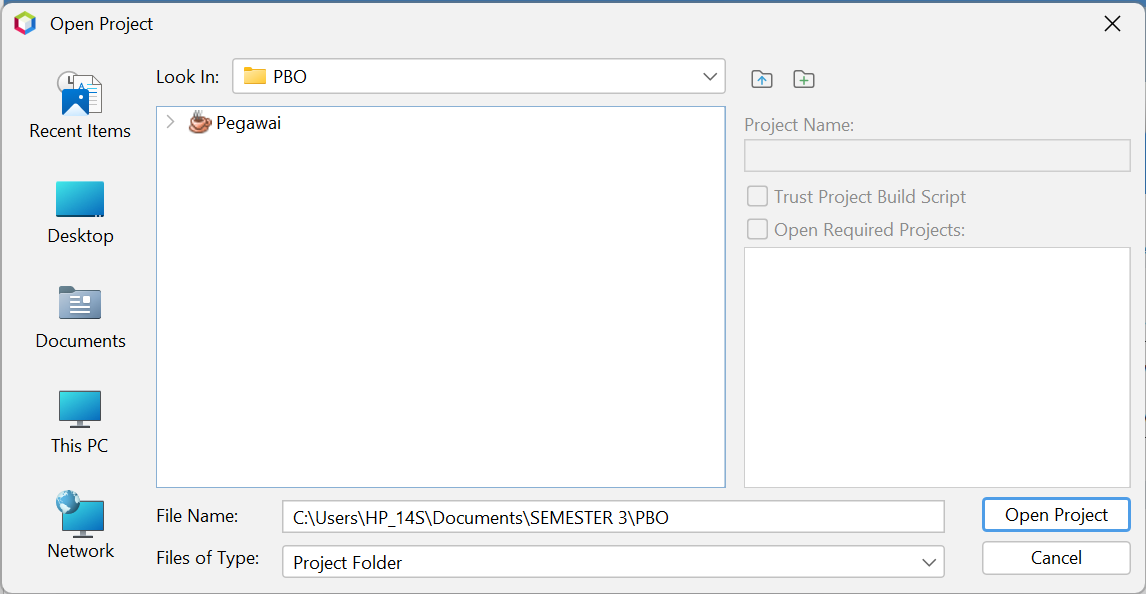


Gambar 4. 1 Open Neatbeans

1. Setelah itu arahkan kursor kea rah kiri pada bagian *file*, lalu klik *Open new projec*t, klik *project* yang akan dijalankan yaitu “pegawai”.

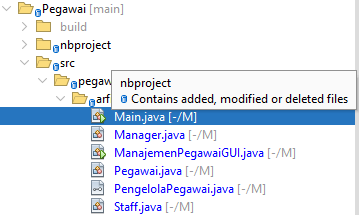


Gambar 4. 2 Open Projects



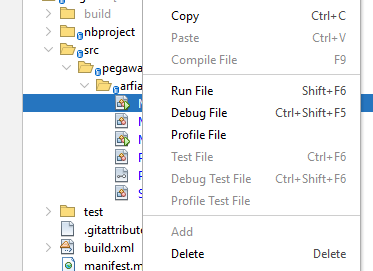
Gambar 4. 3 Arahkan ke project

1. Buka src/pegawai/arfiana\_maulidiyah



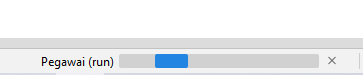
Gambar 4. 4 Open struktur project

1. Klik *file* Main.java, lalu klik kanan *run* *file* atau shift + f6 untuk mengetahui hasil atau *output* dari program.



Gambar 4. 5 Run program

1. Setelah muncul tampilan seperti dibawah, tunggu hingga program muncul.



Gambar 4. 6 Tunggu program muncul

## Menambahkan Pegawai

1. Setelah me-*run* program maka akan menu pegawai di bawah ini yang terdiri dari beberapa pilihan operasi. Lalu klik Tombol Tambah Pegawai unutk menambahkan daftar nama pegawai



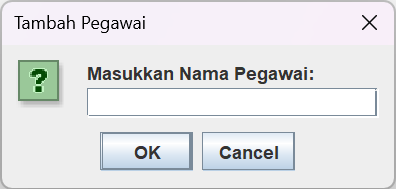
Gambar 4. 7 Menu Pegawai

1. Setelah memilih opsi tambah pegawai maka akan muncul tampilan seperti dibawah ini, disini diberi pilihan untuk menambahkan pegawai berdasarkan jenisnya. Pilih Manajer / *Staff*, lalu klik ok untuk melanjutkan proses selanjtunya.



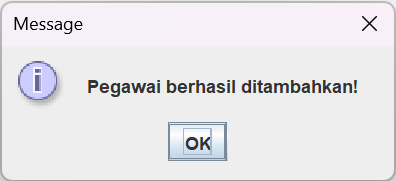
Gambar 4. 8 Menu Staff / Manager

1. *Input*kan nama pegawai yang akan ditambahkan pada daftar pegawai, klik ok.



Gambar 4. 9 Isi Nama Pegawai

1. Lalu akan muncul tampilan seperti berikut jika telah berhasil menambahkan nama pegawai pada daftar pegawai yang telah dibuat sebelumnya. Klik Ok

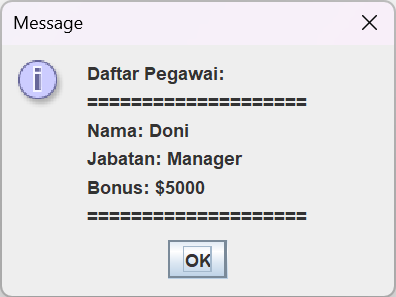


Gambar 4. 10 Dialog sukses add pegawai

1. Kemudian akan Kembali ke menu awal, klik tampilkan daftar pegawai, Jika diklik ok, maka akan kembali menu awal.



Gambar 4. 11 Back to menu main



Gambar 4. 12 Dialog show pegawai

1. Klik keluar, maka program selesai berhenti



Gambar 4. 13 Back to menu main

# BAB V KESIMPULAN

Kesimpulan dari laporan tugas besar ini menunjukkan bahwa pengembangan aplikasi dengan merapkan konsep-konsep pemrograman berorientasi objek seperti *inheritance, polymorphisme, encapsulation, getter* dan *setter*, dan *interface*, telah berhasil memberikan struktur yang efisien dan fleksibel pada aplikasi yang dikembangkan.

Dari kode yang diberikan, kita dapat menyimpulkan beberapa hal:

* 1. Struktur Kode:

1. Kode terstruktur dengan baik, menggunakan konsep OOP (Object-Oriented Programming) dengan *inheritance* dan *interface*.
2. Dibuat hierarki kelas Pegawai, *Manager*, dan *Staff* yang menggunakan konsep *inheritance*.
3. Menggunakan *interface* PengelolaPegawai untuk memastikan implementasi metode tertentu pada kelas ManajemenPegawaiGUI.
   1. *Polymorphisme*:
4. Menerapkan polimorfisme dengan menggunakan metode info() yang di-override pada kelas *Manager* dan *Staff*.
5. Polimorfisme memungkinkan pemanggilan metode info() dari objek Pegawai, tanpa harus mengetahui jenis sebenarnya (*Manager* atau *Staff*).
   1. *Encapsulation* :
6. Menggunakan *encapsulation* dengan mendeklarasikan atribut kelas sebagai *private* dan menyediakan *getter* dan *setter* untuk mengakses dan mengubah nilai atribut tersebut.
   1. *Getter* dan *Setter* :
7. Setiap kelas (Pegawai, *Manager*, *Staff*) memiliki *getter* dan *setter* untuk mengakses dan mengubah nilai atribut yang bersifat *private*.
   1. *Inheritance* :
8. Memanfaatkan *inheritance* untuk membuat kelas *Manager* dan *Staff* yang merupakan turunan dari kelas Pegawai.
9. *Inheritance* memungkinkan kelas turunan untuk mewarisi sifat dan perilaku kelas induknya.
   1. Antarmuka Pengguna (GUI):
10. Menerapkan antarmuka pengguna sederhana menggunakan JOptionPane untuk berinteraksi dengan pengguna.
11. Menyediakan opsi untuk menambah pegawai, menampilkan daftar pegawai, dan keluar dari program.
    1. Dinamika Program:
12. Program memiliki dinamika interaktif di mana pengguna dapat memilih operasi yang ingin dilakukan.
13. Pengguna dapat menambahkan *Manager* atau *Staff*, dan melihat daftar pegawai yang telah ditambahkan.

Dengan demikian, keseluruhan implementasi menunjukkan penggunaan konsep-konsep dasar Pemrograman Berorientasi Objek dengan baik, seperti *inheritance, polimorfisme, encapsulation, getter* dan *setter*, serta antarmuka pengguna (GUI).

# DAFTAR PUSTAKA

Bastari, D.I., Pradana, F. and Priyambadha, B. (2017) ‘Pengembangan Sistem Pembelajaran Pemrograman Java yang Atraktif Berbasis Website’, *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 1(12), pp. 1493–1499.

Herwanto, H.W. and Febrita, R.E. (2014) ‘Pengembangan Media Pembelajaran Berbasis Web pada Matakuliah Pemrograman Berorientasi Objek’, *Tekno*, 21(1), pp. 11–18.

Ramadhan, R. (2020) ‘RANCANG BANGUN E-LEARNING DAN REPOSITORI DIGITAL DENGAN PLATFORM GITHUB ( Studi Kasus : Centratama Telekomunikasi Indonesia )’.

Tengah, J.R., Rebo, P. and Timur, J. (2022) ‘Perancangan Sistem Penilaian Kinerja Pegawai Pada’, (2012), pp. 139–145.