# LAPORAN TUGAS BESAR
# PEMROGRAMAN BERORIENTASI OBJEK
# SISTEM PEGAWAI


**Disusun Oleh :**

**NAMA : ARFIANA MAULIDIYAH**

**NIM : 32602200012**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INDUSTRI**

**UNIVERSITAS ISLAM SULTAN AGUNG**

**SEMARANG**

**2022**

**DAFTAR ISI**

## DAFTAR GAMBAR

# BAB I Pendahuluan

## 1.1 Latar belakang

Pengelolaan pegawai merupakan aspek krusial dalam suatu organisasi. Dalam rangka meningkatkan efisiensi dan keteraturan data pegawai, sistem manajemen pegawai berbasis pemrograman berorientasi objek (PBO) menjadi solusi yang tepat. Pada laporan ini, akan dibahas implementasi sistem manajemen pegawai dengan menggunakan konsep dasar PBO.

## 1.2 Tujuan

Tujuan utama dari laporan ini adalah memberikan pemahaman mengenai konsep dasar PBO yang diterapkan dalam pengelolaan pegawai. Selain itu, laporan ini bertujuan untuk memberikan panduan langkah-langkah dalam menjalankan program dan menambahkan data pegawai.

# BAB II Konsep Dasar PBO

## 2.1 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.

## 2.2 Polimorfisme

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

## 2.3 Encapsulation

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

## 2.4 Getter dan Setter

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

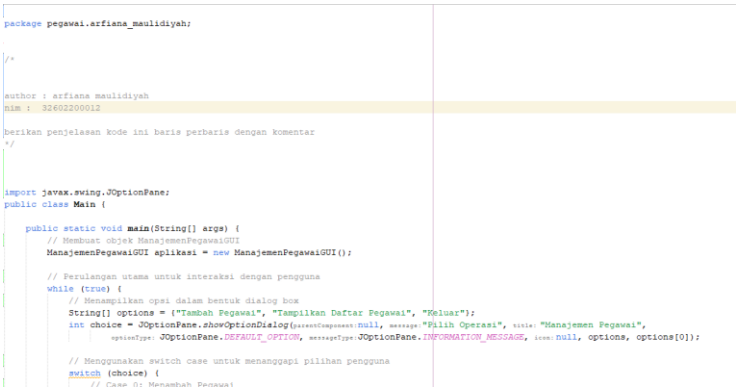
## 2.5 Interface

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

# BAB III Struktur Program

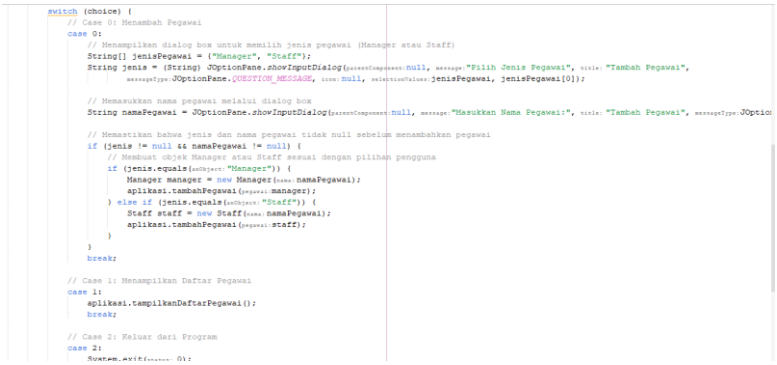
## 3.1 File Utama ( ManajemenPegawaiGUI )

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



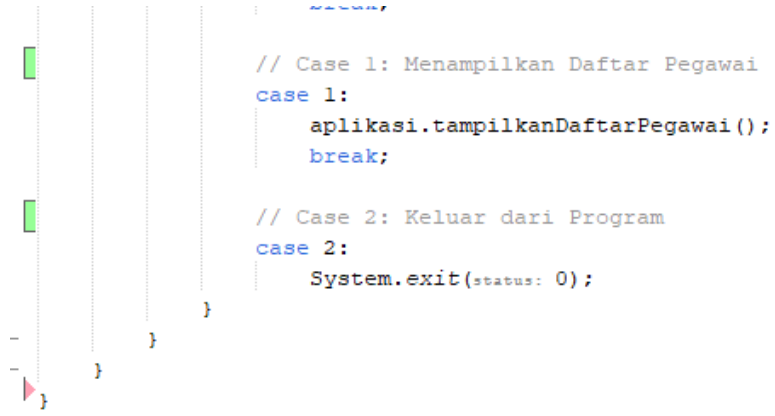
```
package pegawai.arfiana_maulidiyah;

/*

author : arfiana maulidiyah
nim :  32602200012

berikan penjelasan kode ini baris perbaris dengan komentar
*/


import javax.swing.JOptionPane;
public class Main {

    public static void main(String[] args) {
        // Membuat objek ManajemenPegawaiGUI
        ManajemenPegawaiGUI aplikasi = new ManajemenPegawaiGUI();

        // Perulangan utama untuk interaksi dengan pengguna
        while (true) {
            // Menampilkan opsi dalam bentuk dialog box
            String[] options = {"Tambah Pegawai", "Tampilkan Daftar Pegawai", "Keluar"};
            int choice = JOptionPane.showOptionDialog(parentComponent:null, message:"Pilih Operasi", title: "Manajemen Pegawai",
                    optionType: JOptionPane.DEFAULT_OPTION, messageType:JOptionPane.INFORMATION_MESSAGE, icon: null, options, options[0]);

            // Menggunakan switch case untuk menanggapi pilihan pengguna
            switch (choice) {
                // Case 0: Menambah Pegawai
```

Gambar 1. 1 File Utama Program Main 1



```
    switch (choice) {
        // Case 0: Menambah Pegawai
        case 0:
            // Menampilkan dialog box untuk memilih jenis pegawai (Manager atau Staff)
            String[] jenisPegawai = {"Manager", "Staff"};
            String jenis = (String) JOptionPane.showInputDialog(parentComponent:null, message:"Pilih Jenis Pegawai", title: "Tambah Pegawai",
                    messageType:JOptionPane.QUESTION_MESSAGE, icon: null, selectionValues:jenisPegawai, jenisPegawai[0]);

            // Memasukkan nama pegawai melalui dialog box
            String namaPegawai = JOptionPane.showInputDialog(parentComponent:null, message:"Masukkan Nama Pegawai:", title: "Tambah Pegawai", messageType:JOptio

            // Memastikan bahwa jenis dan nama pegawai tidak null sebelum menambahkan pegawai
            if (jenis != null && namaPegawai != null) {
                // Membuat objek Manager atau Staff sesuai dengan pilihan pengguna
                if (jenis.equals(anObject:"Manager")) {
                    Manager manager = new Manager(name:namaPegawai);
                    aplikasi.tambahPegawai(pegawai:manager);
                } else if (jenis.equals(anObject:"Staff")) {
                    Staff staff = new Staff(name:namaPegawai);
                    aplikasi.tambahPegawai(pegawai:staff);
                }
            }
            break;

        // Case 1: Menampilkan Daftar Pegawai
        case 1:
            aplikasi.tampilkanDaftarPegawai();
            break;

        // Case 2: Keluar dari Program
        case 2:
            System.exit(status: 0);
```

Gambar 1. 2 File Utama Program Main 2




Gambar 1. 3 File Utama Program Main 3

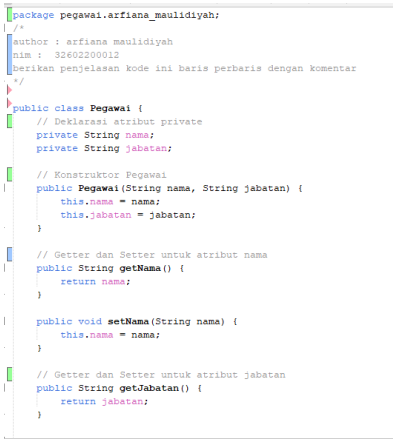
Penjelasan Kode :

1. Package dan Import
   a. package pegawai.arfiana_maulidiyah;: Mendefinisikan package untuk kelas-kelas dalam program.
   b. import javax.swing.JOptionPane;: Mengimpor kelas JOptionPane dari paket javax.swing untuk penggunaan dialog box.
2. Deklarasi Kelas Main
   a. public class Main {: Mendeklarasikan kelas utama dengan nama Main.
3. Metode main
   a. public static void main(String[] args) {: Mendeklarasikan metode main sebagai titik masuk utama program.
4. Inisialisasi Objek ManajemenPegawaiGUI
   a. ManajemenPegawaiGUI aplikasi = new ManajemenPegawaiGUI();: Membuat objek ManajemenPegawaiGUI untuk mengelola pegawai.
5. Perulangan Utama
   a. while (true) {: Membuat perulangan tanpa batas untuk terus berinteraksi dengan pengguna.
6. Opsi Pilihan Pengguna
   a. String[] options = {"Tambah Pegawai", "Tampilkan Daftar Pegawai", "Keluar"};: Membuat array string berisi opsi pilihan pengguna.

7. Dialog Box Pilihan Operasi
    a. int choice = JOptionPane.showOptionDialog(...);: Menampilkan dialog box opsi dan mengambil pilihan pengguna.
8. Switch Case untuk Memproses Pilihan Pengguna
    a. switch (choice) {: Memulai struktur kontrol switch case untuk menanggapi pilihan pengguna.
9. Case 0: Menambah Pegawai
    a. case 0:: Bagian untuk menambah pegawai.
    b. String[] jenisPegawai = {"Manager", "Staff"};: Membuat array string berisi opsi jenis pegawai.
    c. String jenis = (String) JOptionPane.showInputDialog(...);: Menampilkan dialog box untuk memilih jenis pegawai.
    d. String namaPegawai = JOptionPane.showInputDialog(...);: Menampilkan dialog box untuk memasukkan nama pegawai.
    e. if (jenis != null && namaPegawai != null) {: Memastikan input jenis dan nama pegawai tidak null.
    f. if (jenis.equals("Manager")) { ... } else if (jenis.equals("Staff")) { ... }: Membuat objek Manager atau Staff berdasarkan pilihan pengguna.
10. Case 1: Menampilkan Daftar Pegawai
    a. case 1:: Bagian untuk menampilkan daftar pegawai.
    b. aplikasi.tampilkanDaftarPegawai();: Memanggil metode untuk menampilkan daftar pegawai.
11. Case 2: Keluar dari Program
    a. case 2:: Bagian untuk keluar dari program.
    b. System.exit(0);: Mengakhiri program.
12. Penutup Metode main
    a. } // Akhir switch case: Menutup blok switch case.
    b. } // Akhir while loop: Menutup blok while loop.
    c. } // Akhir main method: Menutup blok main method.
    d. } // Akhir kelas Main: Menutup blok kelas Main.

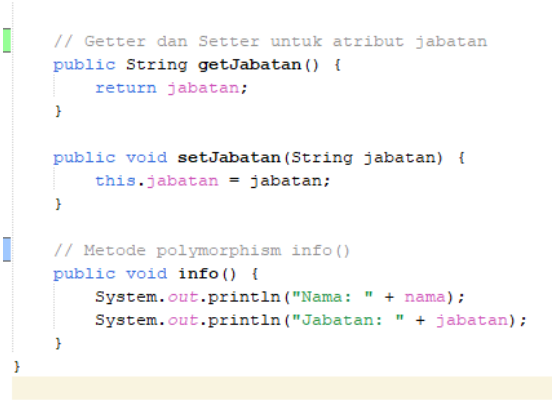
## 3.2 Kelas Pegawai

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



```
package pegawai.arfiana_maulidiyah;
/*
author : arfiana maulidiyah
nim :  32602200012
berikan penjelasan kode ini baris perbaris dengan komentar
*/

public class Pegawai {
    // Deklarasi atribut private
    private String nama;
    private String jabatan;

    // Konstruktor Pegawai
    public Pegawai(String nama, String jabatan) {
        this.nama = nama;
        this.jabatan = jabatan;
    }

    // Getter dan Setter untuk atribut nama
    public String getNama() {
        return nama;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    // Getter dan Setter untuk atribut jabatan
    public String getJabatan() {
        return jabatan;
    }
```

Gambar 1. 4 Kelas Pegawai 1



```
    // Getter dan Setter untuk atribut jabatan
    public String getJabatan() {
        return jabatan;
    }

    public void setJabatan(String jabatan) {
        this.jabatan = jabatan;
    }

    // Metode polymorphism info()
    public void info() {
        System.out.println("Nama: " + nama);
        System.out.println("Jabatan: " + jabatan);
    }
}
```

Gambar 1. 5 Kelas Pegawai 2

Penjelasan Kode :

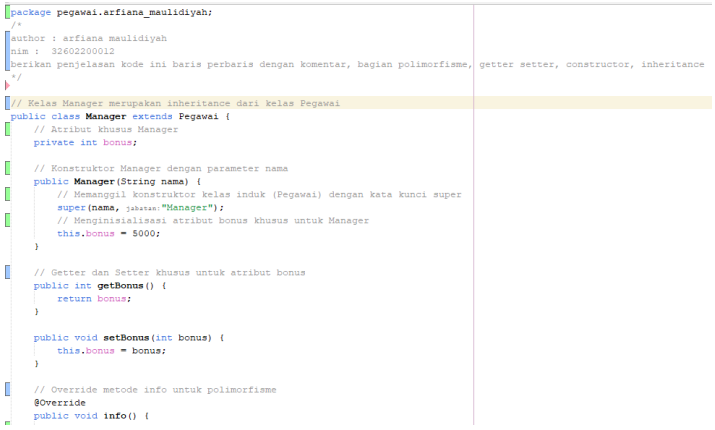
1. Deklarasi Package dan Komentar Header
   a. package pegawai.arfiana_maulidiyah;: Mendefinisikan package tempat kelas Pegawai berada.
   b. Komentar berisi informasi tentang penulis dan NIM.
2. Deklarasi Kelas Pegawai
   a. public class Pegawai {: Mendeklarasikan kelas Pegawai.
3. Atribut Kelas Pegawai

a. private String nama;: Atribut private untuk menyimpan nama pegawai.
b. private String jabatan;: Atribut private untuk menyimpan jabatan pegawai.

4. Konstruktor Kelas Pegawai
   a. public Pegawai(String nama, String jabatan) {: Konstruktor untuk inisialisasi objek Pegawai dengan nama dan jabatan.
5. Getter dan Setter
   a. public String getNama() { return nama; }: Getter untuk mendapatkan nilai atribut nama.
   b. public void setNama(String nama) { this.nama = nama; }: Setter untuk mengubah nilai atribut nama.
   c. public String getJabatan() { return jabatan; }: Getter untuk mendapatkan nilai atribut jabatan.
   d. public void setJabatan(String jabatan) { this.jabatan = jabatan; }: Setter untuk mengubah nilai atribut jabatan.
6. Metode Polymorphism info()
   a. public void info() { ... }: Metode polymorphism yang menampilkan informasi nama dan jabatan pegawai.

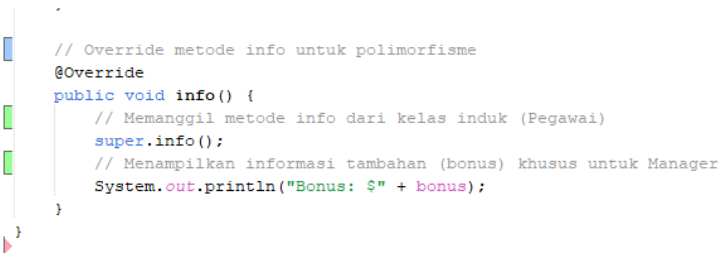
### 3.3 Kelas Manager dan Staff

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



```
package pegawai.arfiana_maulidiyah;
/*
author : arfiana maulidiyah
nim :  32602200012
berikan penjelasan kode ini baris perbaris dengan komentar, bagian polimorfisme, getter setter, constructor, inheritance
*/

// Kelas Manager merupakan inheritance dari kelas Pegawai
public class Manager extends Pegawai {
    // Atribut khusus Manager
    private int bonus;

    // Konstruktor Manager dengan parameter nama
    public Manager(String nama) {
        // Memanggil konstruktor kelas induk (Pegawai) dengan kata kunci super
        super(nama, jabatan:"Manager");
        // Menginisialisasi atribut bonus khusus untuk Manager
        this.bonus = 5000;
    }

    // Getter dan Setter khusus untuk atribut bonus
    public int getBonus() {
        return bonus;
    }

    public void setBonus(int bonus) {
        this.bonus = bonus;
    }

    // Override metode info untuk polimorfisme
    @Override
    public void info() {
```

Gambar 1. 6 Kelas Manager 1



```
    // Override metode info untuk polimorfisme
    @Override
    public void info() {
        // Memanggil metode info dari kelas induk (Pegawai)
        super.info();
        // Menampilkan informasi tambahan (bonus) khusus untuk Manager
        System.out.println("Bonus: $" + bonus);
    }
}
```

Gambar 1. 7 Kelas Manager 2

Penjelasan Kode :

1. Package dan Komentar Header
   a. Komentar berisi informasi tentang penulis dan NIM.
2. Deklarasi Kelas Manager dan Inheritance
   a. public class Manager extends Pegawai {: Mendeklarasikan kelas Manager yang merupakan turunan dari kelas Pegawai.
3. Atribut Kelas Manager
   a. private int bonus;: Atribut khusus untuk Manager.
4. Konstruktor Kelas Manager
   a. public Manager(String nama) { ... }: Konstruktor Manager dengan parameter nama.
   b. super(nama, "Manager");: Memanggil konstruktor kelas induk (Pegawai) dengan kata kunci super.
   c. this.bonus = 5000;: Menginisialisasi atribut bonus khusus untuk Manager.
5. Getter dan Setter untuk Atribut Bonus

a. public int getBonus() { return bonus; }: Getter untuk mendapatkan nilai atribut bonus.
b. public void setBonus(int bonus) { this.bonus = bonus; }: Setter untuk mengubah nilai atribut bonus.

6. Override Metode info() untuk Polimorfisme
   a. @Override: Menandakan bahwa metode berikut ini di-override dari kelas induk.
   b. super.info();: Memanggil metode info dari kelas induk (Pegawai).
   c. System.out.println("Bonus: $" + bonus);: Menampilkan informasi tambahan (bonus) khusus untuk Manager.


Gambar 1. 8 Kelas Staff 1

```
    // Override metode info untuk polimorfisme
    @Override
    public void info() {
        // Memanggil metode info dari kelas induk (Pegawai)
        super.info();
        // Menampilkan informasi tambahan (lembur) khusus untuk Staff
        System.out.println("Lembur: " + lembur + " jam");
    }
}
```

Gambar 1. 9 Kelas Staff 2

Penjelasan Kode :

1. Package dan Komentar Header
   a. Komentar berisi informasi tentang penulis dan NIM.
2. Deklarasi Kelas Staff dan Inheritance
   a. public class Staff extends Pegawai {: Mendeklarasikan kelas Staff yang merupakan turunan dari kelas Pegawai.

3. Atribut Kelas Staff
    a. private int lembur;: Atribut khusus untuk Staff.
4. Konstruktor Kelas Staff
    a. public Staff(String nama) { ... }: Konstruktor Staff dengan parameter nama.
    b. super(nama, "Staff");: Memanggil konstruktor kelas induk (Pegawai) dengan kata kunci super.
    c. this.lembur = 10;: Menginisialisasi atribut lembur khusus untuk Staff.
5. Getter dan Setter untuk Atribut Lembur
    a. public int getLembur() { return lembur; }: Getter untuk mendapatkan nilai atribut lembur.
    b. public void setLembur(int lembur) { this.lembur = lembur; }: Setter untuk mengubah nilai atribut lembur.
6. Override Metode info() untuk Polimorfisme
    a. @Override: Menandakan bahwa metode berikut ini di-override dari kelas induk.
    b. super.info();: Memanggil metode info dari kelas induk (Pegawai).
    c. System.out.println("Lembur: " + lembur + " jam");: Menampilkan informasi tambahan (lembur) khusus untuk Staff.

## 3.4 Interface PengelolaPegawai

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

```
package pegawai.arfiana_maulidiyah;
/*
author : arfiana maulidiyah
nim :  32602200012
berikan penjelasan kode ini baris perbaris dengan komentar
*/

// Interface PengelolaPegawai
public interface PengelolaPegawai {
    // Metode untuk menambah pegawai
    void tambahPegawai(Pegawai pegawai);

    // Metode untuk menampilkan daftar pegawai
    void tampilkanDaftarPegawai();
}
```

Gambar 1. 10 Interface Pengelola Pegawai

Penjelasan Kode :

1. Package dan Komentar Header

   Komentar berisi informasi tentang penulis dan NIM.

2. Deklarasi Interface PengelolaPegawai

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

3. Metode tambahPegawai

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

4. Metode tampilkanDaftarPegawai

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

5. Penutup Interface

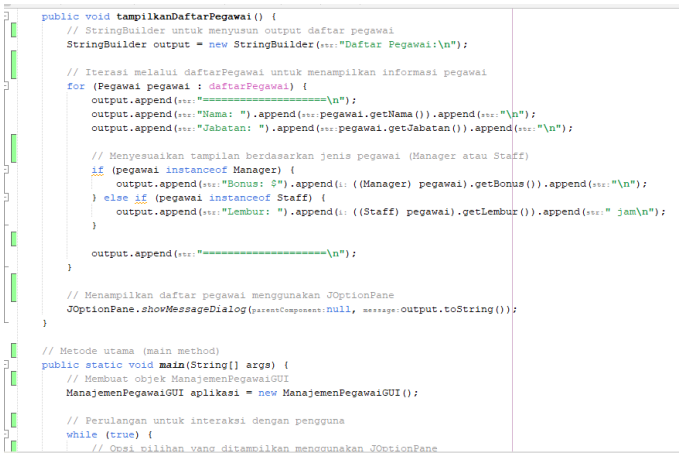
   }: Menutup deklarasi interface PengelolaPegawai.

## 3.5 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..

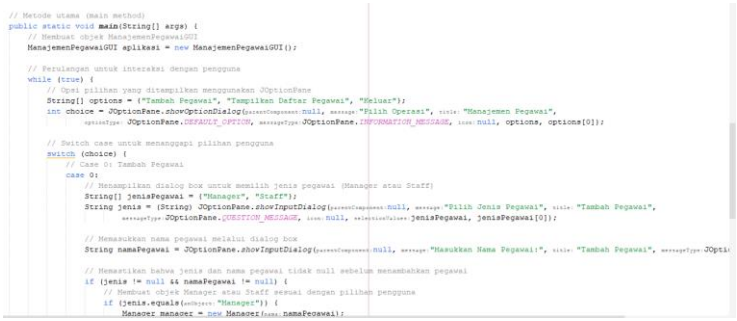
```
package pegawai.arfiana_maulidiyah;
/*
author : arfiana maulidiyah
nim :  32602200012
berikan penjelasan kode ini baris perbaris dengan komentar
*/

import java.util.ArrayList;
import javax.swing.JOptionPane;

// Kelas ManajemenPegawaiGUI implementasi dari interface PengelolaPegawai
public class ManajemenPegawaiGUI implements PengelolaPegawai {

    // Atribut untuk menyimpan daftar pegawai menggunakan ArrayList
    private ArrayList<Pegawai> daftarPegawai = new ArrayList<>();

    // Implementasi metode tambahPegawai dari interface PengelolaPegawai
    @Override
    public void tambahPegawai(Pegawai pegawai) {
        // Menambah pegawai ke dalam daftarPegawai
        daftarPegawai.add(e: pegawai);
        // Menampilkan pesan sukses menggunakan JOptionPane
        JOptionPane.showMessageDialog(parentComponent:null, message:"Pegawai berhasil ditambahkan!");
    }

    // Implementasi metode tampilkanDaftarPegawai dari interface PengelolaPegawai
    @Override
    public void tampilkanDaftarPegawai() {
        // StringBuilder untuk menyusun output daftar pegawai
        StringBuilder output = new StringBuilder(str:"Daftar Pegawai:\n");

        // Iterasi melalui daftarPegawai untuk menampilkan informasi pegawai
```

Gambar 1. 11 MainPegawaiGUI 1



```
    public void tampilkanDaftarPegawai() {
        // StringBuilder untuk menyusun output daftar pegawai
        StringBuilder output = new StringBuilder("Daftar Pegawai:\n");

        // Iterasi melalui daftarPegawai untuk menampilkan informasi pegawai
        for (Pegawai pegawai : daftarPegawai) {
            output.append("====================\n");
            output.append("Nama: ").append(pegawai.getNama()).append("\n");
            output.append("Jabatan: ").append(pegawai.getJabatan()).append("\n");

            // Menyesuaikan tampilan berdasarkan jenis pegawai (Manager atau Staff)
            if (pegawai instanceof Manager) {
                output.append("Bonus: $").append((Manager) pegawai).getBonus()).append("\n");
            } else if (pegawai instanceof Staff) {
                output.append("Lembur: ").append((Staff) pegawai).getLembur()).append(" jam\n");
            }

            output.append("====================\n");
        }

        // Menampilkan daftar pegawai menggunakan JOptionPane
        JOptionPane.showMessageDialog(null, output.toString());
    }

    // Metode utama (main method)
    public static void main(String[] args) {
        // Membuat objek ManajemenPegawaiGUI
        ManajemenPegawaiGUI aplikasi = new ManajemenPegawaiGUI();

        // Perulangan untuk interaksi dengan pengguna
        while (true) {
            // Opsi pilihan yang ditampilkan menggunakan JOptionPane
```

Gambar 1. 12 MainPegawaiGUI 2



```
// Metode utama (main method)
public static void main(String[] args) {
    // Membuat objek ManajemenPegawaiGUI
    ManajemenPegawaiGUI aplikasi = new ManajemenPegawaiGUI();

    // Perulangan untuk interaksi dengan pengguna
    while (true) {
        // Opsi pilihan yang ditampilkan menggunakan JOptionPane
        String[] options = {"Tambah Pegawai", "Tampilkan Daftar Pegawai", "Keluar"};
        int choice = JOptionPane.showOptionDialog(parentComponent:null, message:"Pilih Operasi", title: "Manajemen Pegawai",
                optionType: JOptionPane.DEFAULT_OPTION, messageType:JOptionPane.INFORMATION_MESSAGE, icon:null, options, options[0]);

        // Switch case untuk menanggapi pilihan pengguna
        switch (choice) {
            // Case 0: Tambah Pegawai
            case 0:
                // Menampilkan dialog box untuk memilih jenis pegawai (Manager atau Staff)
                String[] jenisPegawai = {"Manager", "Staff"};
                String jenis = (String) JOptionPane.showInputDialog(parentComponent:null, message:"Pilih Jenis Pegawai", title: "Tambah Pegawai",
                        messageType: JOptionPane.QUESTION_MESSAGE, icon:null, selectionValues: jenisPegawai, jenisPegawai[0]);

                // Memasukkan nama pegawai melalui dialog box
                String namaPegawai = JOptionPane.showInputDialog(parentComponent:null, message:"Masukkan Nama Pegawai:", title: "Tambah Pegawai", messageType: JOpti

                // Memastikan bahwa jenis dan nama pegawai tidak null sebelum menambahkan pegawai
                if (jenis != null && namaPegawai != null) {
                    // Membuat objek Manager atau Staff sesuai dengan pilihan pengguna
                    if (jenis.equals(anObject:"Manager")) {
                        Manager manager = new Manager(nama:namaPegawai);
```

Gambar 1. 13 MainPegawaiGUI 3

```
                    if (jenis.equals(anObject: "Manager")) {
                        Manager manager = new Manager(nama: namaPegawai);
                        aplikasi.tambahPegawai(pegawai:manager);
                    } else if (jenis.equals(anObject: "Staff")) {
                        Staff staff = new Staff(nama: namaPegawai);
                        aplikasi.tambahPegawai(pegawai:staff);
                    }
                }
                break;

            // Case 1: Tampilkan Daftar Pegawai
            case 1:
                aplikasi.tampilkanDaftarPegawai();
                break;

            // Case 2: Keluar dari Program
            case 2:
                System.exit(status: 0);
        }
    }
}
```

Gambar 1. 14 MainPegawaiGUI 4

Penjelasan Kode :

1. Package dan Komentar Header
    a. Komentar berisi informasi tentang penulis dan NIM.
2. Import Paket
    a. import java.util.ArrayList;: Mengimpor kelas ArrayList dari paket

java.util.

b. import javax.swing.JOptionPane;: Mengimpor kelas JOptionPane dari paket javax.swing.

3. Deklarasi Kelas ManajemenPegawaiGUI

   a. public class ManajemenPegawaiGUI implements PengelolaPegawai {: Mendeklarasikan kelas ManajemenPegawaiGUI yang mengimplementasi interface PengelolaPegawai.

4. Atribut Kelas ManajemenPegawaiGUI

   a. private ArrayList<Pegawai> daftarPegawai = new ArrayList<>();: Membuat ArrayList untuk menyimpan daftar pegawai.

5. Implementasi Metode dari Interface

   a. @Override: Menandakan bahwa metode berikut diimplementasi dari interface.

   b. void tambahPegawai(Pegawai pegawai) { ... }: Menambahkan pegawai ke dalam daftar dan menampilkan pesan dengan JOptionPane.

   c. void tampilkanDaftarPegawai() { ... }: Menampilkan daftar pegawai dengan informasi sesuai jenis pegawai menggunakan JOptionPane.

6. Metode Main

   a. public static void main(String[] args) { ... }: Metode utama untuk menjalankan aplikasi.

   b. ManajemenPegawaiGUI aplikasi = new ManajemenPegawaiGUI();: Membuat objek ManajemenPegawaiGUI.

   c. Perulangan while (true) untuk terus berinteraksi dengan pengguna.

   d. String[] options = {"Tambah Pegawai", "Tampilkan Daftar Pegawai", "Keluar"};: Opsi pilihan yang ditampilkan menggunakan JOptionPane.

   e. int choice = JOptionPane.showOptionDialog(...);: Meminta pengguna untuk memilih operasi menggunakan JOptionPane.

   f. Switch case untuk menanggapi pilihan pengguna.

   g. Case 0: Menambah Pegawai

   h. String[] jenisPegawai = {"Manager", "Staff"};: Menampilkan dialog box untuk memilih jenis pegawai.

i. if (jenis.equals("Manager")) { ... } else if (jenis.equals("Staff")) { ... }: Membuat objek Manager atau Staff sesuai dengan pilihan pengguna.

j. Case 1: Menampilkan Daftar Pegawai

k. aplikasi.tampilkanDaftarPegawai();: Memanggil metode tampilkanDaftarPegawai() untuk menampilkan daftar pegawai.

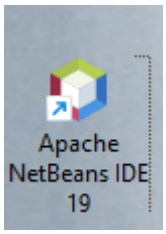
l. Case 2: Keluar dari Program

m. System.exit(0);: Keluar dari program jika pengguna memilih opsi keluar.

# BAB IV Implementasi Program

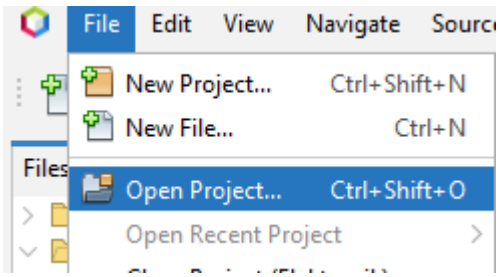
## 4.1 Menjalankan Program

1. Untuk menjalankan program klick neatbeans

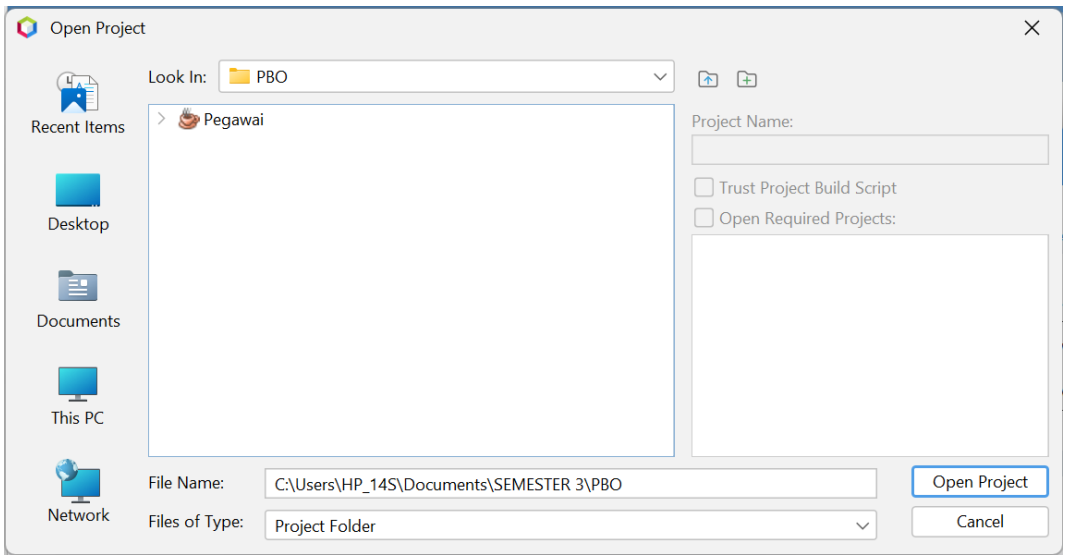



Gambar 2. 1 Open Neatbeans

2. Open new project, arahkan ke pegawai

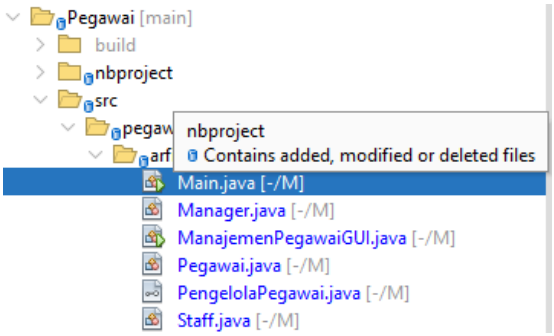



Gambar 2. 2 Open Projects



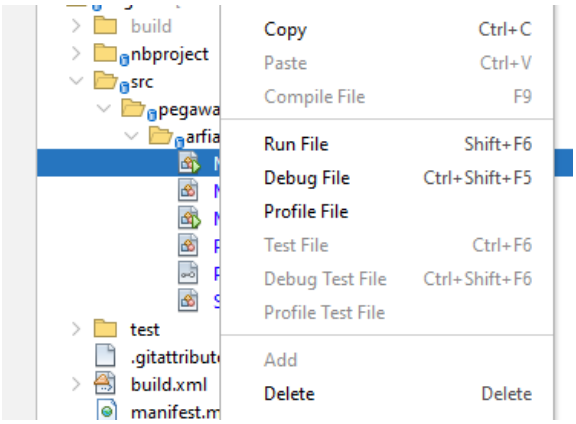

Gambar 2. 3 Arahkan ke project

3. Buka src/pegawai/arfiana_maulidiyah



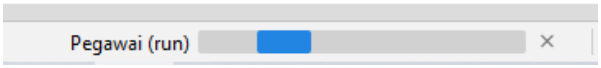

Gambar 2. 4 Open struktur project

4. Klick file Main.java, lalu klick kanan run file atau shift + f6




Gambar 2. 5 Run program

5. Tunggu program muncul




Gambar 2. 6 Tunggu program muncul

## 4.2 Menambahkan Pegawai

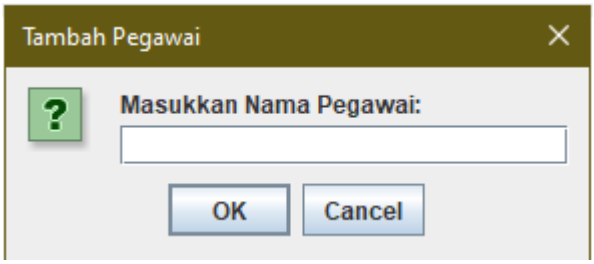
1. Klick Tombol Tambah Pegawai


Gambar 2. 7 Menu Pegawai

2. Pilih Manajer / Staff, lalu klick ok

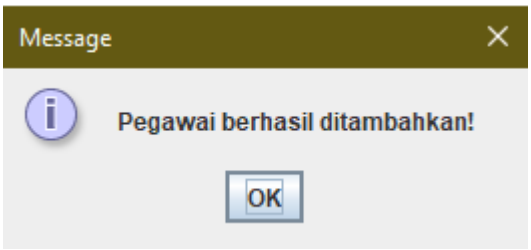

Gambar 2. 8 Menu Staff / Manager

3. Masukan isi nama pegawai, klick ok.




Gambar 2. 9 Isi Nama Pegawai

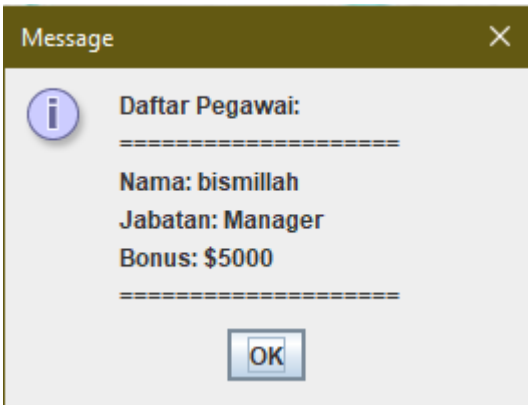
4. Lalu akan muncul dialog sukses berhasil menambahkan. Klick Ok




Gambar 2. 10 Dialog sukses add pegawai

5. Akan Kembali ke menu awal, klick tampilkan daftar pegawai, Jika diklick ok, maka akan Kembali menu awal.


Gambar 2. 11 Back to menu main




Gambar 2. 12 Dialog show pegawai

6. Klick keluar, program selesai berhenti


Gambar 2. 13 Back to menu main

# BAB V Kesimpulan

Dari kode yang diberikan, kita dapat menyimpulkan beberapa hal:

1. Struktur Kode:
   a. Kode terstruktur dengan baik, menggunakan konsep OOP (Object-Oriented Programming) dengan inheritance dan interface.
   b. Dibuat hierarki kelas Pegawai, Manager, dan Staff yang menggunakan konsep inheritance.
   c. Menggunakan interface PengelolaPegawai untuk memastikan implementasi metode tertentu pada kelas ManajemenPegawaiGUI.
2. Polimorfisme:
   a. Menerapkan polimorfisme dengan menggunakan metode info() yang di-override pada kelas Manager dan Staff.
   b. Polimorfisme memungkinkan pemanggilan metode info() dari objek Pegawai, tanpa harus mengetahui jenis sebenarnya (Manager atau Staff).
3. Encapsulation:
   a. Menggunakan encapsulation dengan mendeklarasikan atribut kelas sebagai private dan menyediakan getter dan setter untuk mengakses dan mengubah nilai atribut tersebut.
4. Getter dan Setter:
   a. Setiap kelas (Pegawai, Manager, Staff) memiliki getter dan setter untuk mengakses dan mengubah nilai atribut yang bersifat private.
5. Inheritance:
   a. Memanfaatkan inheritance untuk membuat kelas Manager dan Staff yang merupakan turunan dari kelas Pegawai.
   b. Inheritance memungkinkan kelas turunan untuk mewarisi sifat dan perilaku kelas induknya.
6. Antarmuka Pengguna (GUI):

a. Menerapkan antarmuka pengguna sederhana menggunakan JOptionPane untuk berinteraksi dengan pengguna.
b. Menyediakan opsi untuk menambah pegawai, menampilkan daftar pegawai, dan keluar dari program.

7. Dinamika Program:
   a. Program memiliki dinamika interaktif di mana pengguna dapat memilih operasi yang ingin dilakukan.
   b. 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 (OOP) dengan baik, seperti inheritance, polimorfisme, encapsulation, getter dan setter, serta antarmuka pengguna (GUI).

# DAFTAR PUSTAKA

(Czajkowski and Von Eicken, 1998; Lumpe, Mahmud and Vasa, 2010; Ari Yuana, S.Si, M.Kom, 2015)

Ari Yuana, S.Si, M.Kom, R. (2015) 'Pemrograman Java', *Informatika*, pp. 1–77. Available at: https://d1wqtxts1xzle7.cloudfront.net/33519053/JavaIndo.pdf?1398090002=&response-content-disposition=inline%3B+filename%3DDitulis_oleh.pdf&Expires=1644744989&Signature=hOHnqB2lSAg3O37OjxRlo8mzhmE2pUyaB6R8UbZ75XMmwTHll9mbtKfvvdS4S7MgZTY9R46hVzgrAIej4uG5YJK.

Czajkowski, G. and Von Eicken, T. (1998) 'JRes: A Resource Accounting Interface for Java', *SIGPLAN Notices (ACM Special Interest Group on Programming Languages)*, 33(10), pp. 21–35. Available at: https://doi.org/10.1145/286942.286944.

Lumpe, M., Mahmud, S. and Vasa, R. (2010) 'On the use of properties in Java applications', *Proceedings of the Australian Software Engineering Conference, ASWEC*, pp. 235–244. Available at: https://doi.org/10.1109/ASWEC.2010.35.