

**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**

2024

DAFTAR ISI

HALAMAN JUDUL	1
DAFTAR ISI.....	ii
DAFTAR GAMBAR.....	iii
BAB I PENDAHULUAN	1
1.1 Latar belakang	1
1.2 Tujuan.....	1
BAB II Konsep Dasar PBO	2
2.1 <i>Inheritance</i>	2
2.2 Polimorfisme	2
2.3 <i>Encapsulation</i>	2
2.4 <i>Getter</i> dan <i>Setter</i>	2
2.5 <i>Interface</i>	2
BAB III Struktur Program	3
3.1 <i>File</i> Utama (ManajemenPegawaiGUI).....	3
3.2 Kelas Pegawai	6
3.3 Kelas <i>Manager</i> dan <i>Staff</i>	7
3.4 <i>Interface</i> PengelolaPegawai	10
3.5 Kelas ManajemenPegawaiGUI (Main Class).....	11
BAB IV Implementasi Program	15
4.1 Menjalankan Program	15
4.2 Menambahkan Produk.....	16
BAB V KESIMPULAN	
DAFTAR PUSTAKA	

DAFTAR GAMBAR

Gambar 3. 1 File Utama Program Main 1	3
Gambar 3. 2 File Utama Program Main 2.....	3
Gambar 3. 3 File Utama Program Main 3.....	4
Gambar 3. 4 Kelas Pegawai 1	6
Gambar 3. 5 Kelas Pegawai 2	6
Gambar 3. 6 Kelas Manager 1.....	8
Gambar 3. 7 Kelas Manager 2.....	8
Gambar 3. 8 Kelas Staff 1	9
Gambar 3. 9 Kelas Staff 2	9
Gambar 3. 10 Interface Pengelola Pegawai	10
Gambar 3. 11 MainPegawaiGUI 1	11
Gambar 3. 12 MainPegawaiGUI 2.....	12
Gambar 3. 13 MainPegawaiGUI 3	12
Gambar 3. 14 MainPegawaiGUI 4.....	12
Gambar 4. 1 Open Neatbeans.....	15
Gambar 4. 2 Open Projects	15
Gambar 4. 3 Arahkan ke project	15
Gambar 4. 4 Open struktur project.....	16
Gambar 4. 5 Run program.....	16
Gambar 4. 6 Tunggu program muncul.....	16
Gambar 4. 7 Menu Pegawai	17
Gambar 4. 8 Menu Staff / Manager	17
Gambar 4. 9 Isi Nama Pegawai.....	17
Gambar 4. 10 Dialog sukses add pegawai	17
Gambar 4. 11 Back to menu main.....	18
Gambar 4. 12 Dialog show pegawai	18
Gambar 4. 13 Back to menu main.....	18

BAB I

PENDAHULUAN

1.1 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.

1.2 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.

1.3 Rumusan Masalah

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

1.4 Manfaat

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

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 antar muka pengguna. Dalam kelas ini, *user* dapat menambahkan pegawai dan menampilkan daftar pegawai..

```
package pegawai.erfiana_maulidiyah;

/*
author : erfiana 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(null, message: "Pilih Operasi", title: "Manajemen Pegawai",
                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 3. 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(null, message: "Pilih Jenis Pegawai", title: "Tambah Pegawai",
            JOptionPane.QUESTION_MESSAGE, icon: null, selectionIndices: JList.DEFAULT_SELECTION_INDEXES, jenisPegawai, jenisPegawai[0]);

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

        // Memeriksa 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("Manager")) {
                Manager manager = new Manager(namaPegawai);
                aplikasi.tambahPegawai(manager);
            } else if (jenis.equals("Staff")) {
                Staff staff = new Staff(namaPegawai);
                aplikasi.tambahPegawai(staff);
            }
        }
        break;

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

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

Gambar 3. 2 File Utama Program Main 2

- 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 3. 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 3. 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 : 3260200012
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 3. 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 3. 7 Kelas *Manager* 2

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*.

```
package pegawai.arfiana_maulidiyah;
/*
author : arfiana maulidiyah
nim : 33602200012
Berikan penjelasan kode ini baris perbaris dengan komentar, bagian polimorfisme, getter setter, constructor, inheritance
*/
// Kelas Staff merupakan turunan dari kelas Pegawai
public class Staff extends Pegawai {
    // Atribut khusus Staff
    private int lembur;

    // Konstruktor Staff dengan parameter nama
    public Staff(String nama) {
        // Memanggil konstruktor kelas induk (Pegawai) dengan kata kunci super
        super(nama, jabatan("Staff"));
        // Menginisialisasi atribut lembur khusus untuk Staff
        this.lembur = 10;
    }

    // Getter dan Setter khusus untuk atribut lembur
    public int getLembur() {
        return lembur;
    }

    public void setLembur(int lembur) {
        this.lembur = lembur;
    }

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

Gambar 3. 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 3. 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 3. 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("Daftar Pegawai:\n");

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

Gambar 3. 11 MainPegawaiGUI 1

- 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.

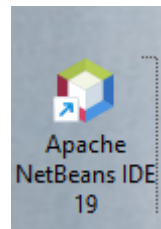
- i. `if (jenis.equals("Manager")) { ... } else if (jenis.equals("Staff")) { ... }:`
Membuat objek *Manager* atau *Staff* sesuai dengan pilihan pengguna.
- j. Case 1: Menampilkan Daftar Pegawai
- k. `aplikasi.tampilkanDaftarPegawai();` Memanggil metode `tampilkanDaftarPegawai()` untuk menampilkan daftar pegawai.
- l. Case 2: Keluar dari Program
- m. `System.exit(0);` Keluar dari program jika pengguna memilih opsi keluar.

BAB IV

Implementasi Program

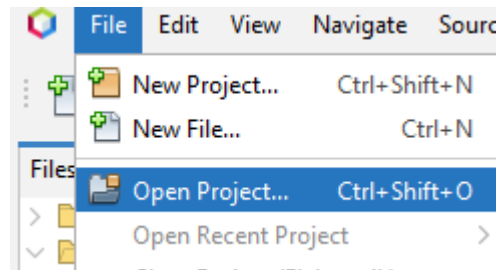
4.1 Menjalankan Program

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

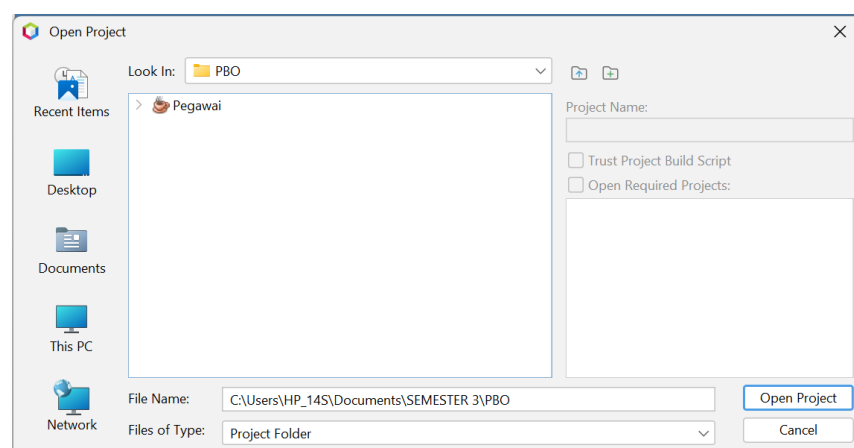


Gambar 4. 1 *Open Neatbeans*

2. Setelah itu arahkan kursor ke arah kiri pada bagian *file*, lalu klik *Open new project*, klik *project* yang akan dijalankan yaitu “pegawai”.

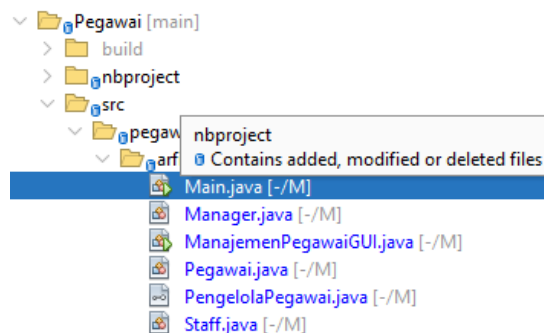


Gambar 4. 2 *Open Projects*



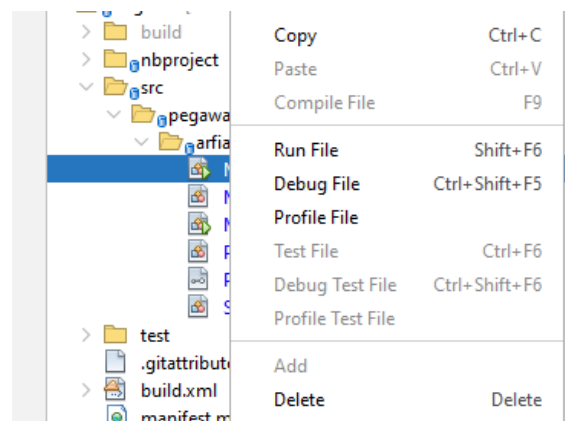
Gambar 4. 3 *Arahkan ke project*

3. Buka src/pegawai/arfiana_maulidiyah



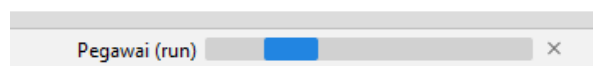
Gambar 4. 4 *Open struktur project*

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



Gambar 4. 5 *Run program*

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



Gambar 4. 6 *Tunggu program muncul*

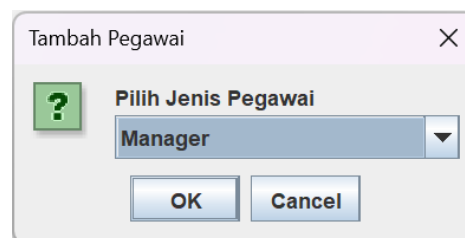
4.2 Menambahkan Pegawai

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

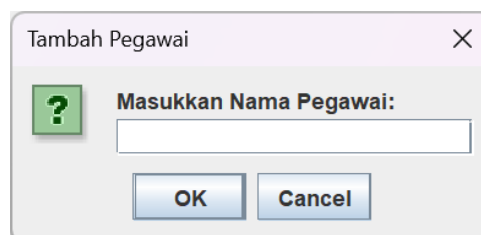


Gambar 4. 7 Menu Pegawai

2. 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 selanjutnya.

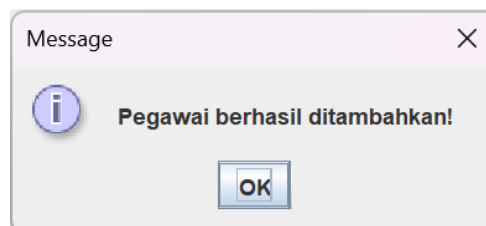
Gambar 4. 8 Menu *Staff* / *Manager*

3. *Inputkan* nama pegawai yang akan ditambahkan pada daftar pegawai, klik ok.

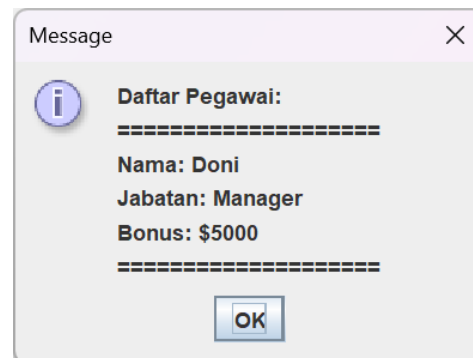


Gambar 4. 9 Isi Nama Pegawai

4. 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

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

Gambar 4. 11 *Back to* menu mainGambar 4. 12 *Dialog show* pegawai

6. 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 menerapkan 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:
 - 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. *Polymorphisme*:
 - 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 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 REPOSITORY 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.