LAPORAN

TUGAS PROJEK

PEMROGRAMAN BERORIENTASI OBJEK

“APLIKASI PERHITUNGAN UPAH PEGAWAI HARIAN”

Oleh:

Nama : Shelvina Eka Julianti

NIM : 2400018132

Kelas : C

Link github : https://github.com/shelvinaeka/TugasakhirPBO-C132

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS AHMAD DAHLAN

YOGYAKARTA

2025

A. DESKRIPSI PROJEK

a. Latar Belakang

Aplikasi ini merupakan sistem perhitungan upah pegawai yang dikembangkan menggunakan bahasa pemrograman Java dengan pendekatan Object-Oriented Programming (OOP) serta antarmuka berbasis Graphical User Interface (GUI)**.**
Aplikasi dirancang untuk menghitung total gaji pegawai berdasarkan jenis pegawai**,** jumlah hari kerja, serta faktor tambahan berupa jam lembur atau potongan gaji. Dengan adanya GUI, pengguna dapat memasukkan data dan memperoleh hasil perhitungan secara interaktif dan mudah digunakan.

b. Tujuan

Tujuan pembuatan aplikasi ini adalah menerapkan konsep Object-Oriented Programming (OOP) dalam sebuah aplikasi desktop berbasis Java GUI untuk menghitung upah pegawai.

c. Manfaat

Aplikasi ini bermanfaat sebagai media pembelajaran PBO, simulasi sistem penggajian, serta mempermudah proses perhitungan gaji.

B. RUANG LINGKUP APLIKASI

Fitur-fitur utama yang tersedia dalam aplikasi ini meliputi:

a. Pemilihan Jenis Pegawai
   Pengguna dapat memilih jenis pegawai melalui GUI, yaitu:
   - Pegawai Harian: Memiliki upah pokok dan tambahan upah lembur.
   - Pegawai Kontrak: Memiliki upah pokok dengan pengurangan berupa potongan tertentu.

b. Input Data Pegawai
   Aplikasi menyediakan form input berbasis GUI untuk memasukkan data berikut:
   - Nama pegawai
   - Jumlah hari kerja
   - Upah per hari
   - Jam lembur (untuk pegawai harian)
   - Potongan gaji (untuk pegawai kontrak)
   - Field input akan menyesuaikan dengan jenis pegawai yang dipilih.

c. Perhitungan Gaji (Polimorfisme)
   Aplikasi menerapkan konsep polimorfisme melalui method hitungGaji(), dengan perhitungan sebagai berikut:
   - Pegawai Harian
     Gaji = (Hari Kerja × Upah per Hari) + (Jam Lembur × Tarif Lembur)

   - Pegawai Kontrak
     Gaji = (Hari Kerja × Upah per Hari) − Potongan

d. Output Hasil
   Aplikasi menampilkan hasil perhitungan gaji secara terformat melalui GUI, meliputi:
   - Nama pegawai
   - Jumlah hari kerja
   - Upah per hari
   - Jam lembur atau potongan sesuai jenis pegawai
   - Total gaji yang diterima

e. Validasi Input
   Aplikasi menggunakan exception handling untuk memastikan data yang dimasukkan valid, antara lain:
   - Nama pegawai tidak boleh kosong
   - Jumlah hari kerja harus lebih dari 0
   - Upah per hari harus lebih dari 0
   - Jam lembur atau potongan tidak boleh bernilai negatif
   - Jika terjadi kesalahan input, sistem akan menampilkan pesan error kepada pengguna.

C. SPESIFIKASI APLIKASI

- Bahasa Pemrograman: Java
- Paradigma: Object-Oriented Programming
- Antarmuka: Java Swing GUI
- IDE: IntelliJ IDEA / NetBeans
- Jenis Pegawai: Pegawai Harian dan Pegawai Kontrak
- Output: Total gaji pegawai

D. RANCANGAN ANTARMUKA

Rancangan antarmuka aplikasi dibuat sederhana dan user friendly menggunakan Java Swing. Antarmuka terdiri dari beberapa komponen utama, yaitu label, text field, combo box, dan button.

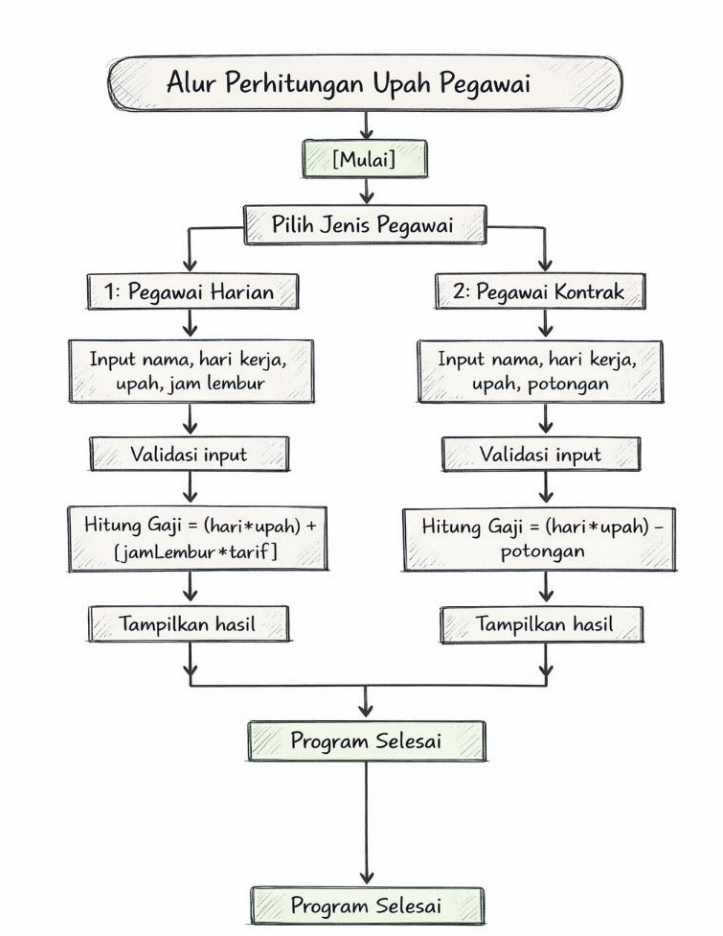
Form input digunakan untuk memasukkan data pegawai seperti nama, jumlah hari kerja, upah per hari, serta jam lembur atau potongan yang akan menyesuaikan dengan jenis pegawai yang dipilih.

Tombol "Hitung Gaji" berfungsi untuk memproses perhitungan dan menampilkan hasil gaji dalam bentuk dialog box.

E. DIAGRAM PROSES APLIKASI

Alur proses aplikasi dimulai dari pengguna membuka aplikasi, kemudian memasukkan data pegawai melalui antarmuka GUI. Pengguna memilih jenis pegawai, yaitu pegawai harian atau pegawai kontrak. Sistem akan memvalidasi seluruh input yang dimasukkan. Jika data valid, sistem akan memanggil method hitungGaji() sesuai dengan jenis pegawai menggunakan konsep polimorfisme.

Hasil perhitungan gaji kemudian ditampilkan kepada pengguna melalui dialog box. Jika terjadi kesalahan input, sistem akan menampilkan pesan error.



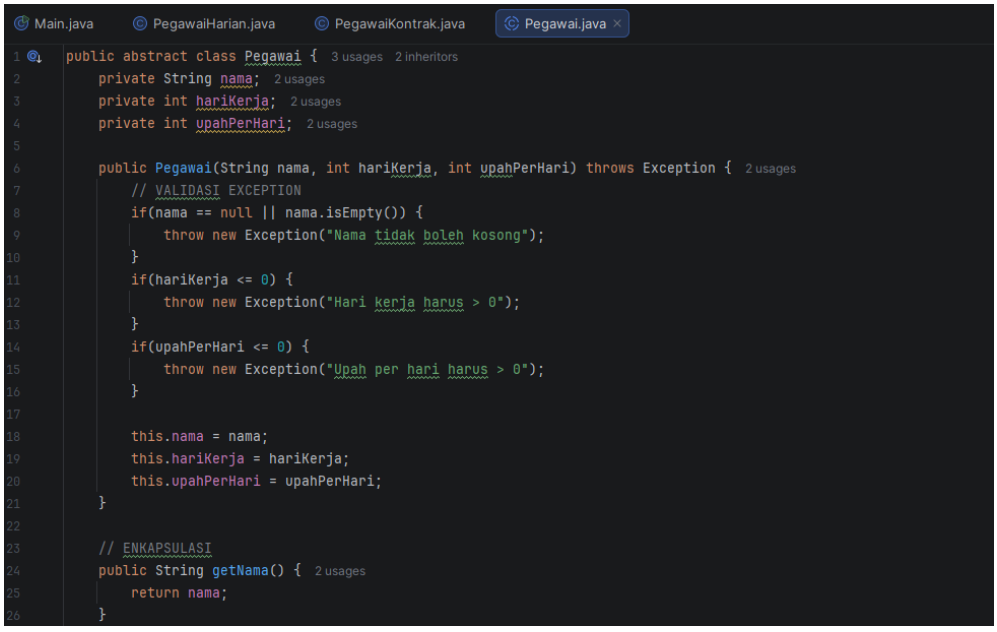
F. KODE PROGRAM DAN PENJELASANNYA

Kode program pada aplikasi ini disusun dengan menerapkan konsep Object-Oriented Programming (OOP). Struktur program terdiri dari class induk Pegawai dan dua class turunan yaitu PegawaiHarian dan PegawaiKontrak. Konsep inheritance dan polimorfisme diterapkan pada method hitungGaji().

1. Kode program
   - **Class induk Pegawai**

```
// Class abstract Pegawai sebagai blueprint pegawai
public abstract class Pegawai {

    // Atribut private untuk enkapsulasi data
    private String nama;       // nama pegawai
    private int hariKerja;     // jumlah hari kerja
    private int upahPerHari;   // upah per hari kerja

    // Constructor untuk inisialisasi objek Pegawai
    // Menerima nama, hari kerja, dan upah per hari sebagai parameter
    // Melempar exception jika data tidak valid
    public Pegawai(String nama, int hariKerja, int upahPerHari) throws Exception {

        // Validasi: nama tidak boleh null atau kosong
        if(nama == null || nama.isEmpty()) {
            throw new Exception("Nama tidak boleh kosong");
        }

        // Validasi: hari kerja harus lebih dari 0
        if(hariKerja <= 0) {
            throw new Exception("Hari kerja harus > 0");
        }

        // Validasi: upah per hari harus lebih dari 0
        if(upahPerHari <= 0) {
            throw new Exception("Upah per hari harus > 0");
        }

        // Assign nilai ke atribut objek menggunakan this
        this.nama = nama;
```

```
    this.hariKerja = hariKerja;
    this.upahPerHari = upahPerHari;
}

// Getter untuk mendapatkan nama pegawai
public String getNama() {
    return nama;
}

// Getter untuk mendapatkan jumlah hari kerja pegawai
public int getHariKerja() {
    return hariKerja;
}

// Getter untuk mendapatkan upah per hari pegawai
public int getUpahPerHari() {
    return upahPerHari;
}

// Method abstract untuk menghitung gaji pegawai
// Harus diimplementasikan di subclass yang mewarisi Pegawai
public abstract int hitungGaji();
```



```
    // ENKAPSULASI
    public String getNama() {  2 usages
        return nama;
    }

    public int getHariKerja() {  4 usages
        return hariKerja;
    }

    public int getUpahPerHari() {  4 usages
        return upahPerHari;
    }

    // ABSTRACT METHOD → harus dioverride di class turunan
    public abstract int hitungGaji();  2 usages  2 implementations
}
```

- **Class PegawaiHarian**

```
// Kelas PegawaiHarian turunan dari Pegawai
// Mengimplementasikan method hitungGaji() khusus untuk pegawai harian
public class PegawaiHarian extends Pegawai {

    // Atribut tambahan khusus pegawai harian
    private int jamLembur;          // jumlah jam lembur
    private final int tarifLembur = 20000;  // tarif per jam lembur, tetap

    // Constructor untuk inisialisasi PegawaiHarian
    public PegawaiHarian(String nama, int hariKerja, int upahPerHari, int
jamLembur) throws Exception {
        super(nama, hariKerja, upahPerHari);  // memanggil constructor Pegawai
(superclass)

        // Validasi: jam lembur tidak boleh negatif
        if(jamLembur < 0) throw new Exception("Jam lembur tidak boleh negatif");

        // Assign nilai jamLembur ke atribut
        this.jamLembur = jamLembur;
    }

    // Getter untuk mengambil jumlah jam lembur
    public int getJamLembur() {
        return jamLembur;
    }
```

```
    // POLIMORFISME: implementasi method abstract hitungGaji()
    @Override
    public int hitungGaji() {
        // Gaji total = gaji pokok + gaji lembur
        // gaji pokok = hariKerja * upahPerHari (dari superclass)
        // gaji lembur = jamLembur * tarifLembur
        return super.getHariKerja() * super.getUpahPerHari() + (jamLembur *
tarifLembur);
    }
```

Main.java | PegawaiHarian.java | PegawaiKontrak.java | Pegawai.java

```
public class PegawaiHarian extends Pegawai {  no usages
    private int jamLembur;  3 usages
    private final int tarifLembur = 20000;  1 usage

    public PegawaiHarian(String nama, int hariKerja, int upahPerHari, int jamLembur) throws Exception {  no usages
        super(nama, hariKerja, upahPerHari);
        if(jamLembur < 0) throw new Exception("Jam lembur tidak boleh negatif");
        this.jamLembur = jamLembur;
    }

    public int getJamLembur() {  no usages
        return jamLembur;
    }

    // POLIMORFISME
    @Override  no usages
    public int hitungGaji() {
        return super.getHariKerja() * super.getUpahPerHari() + (jamLembur * tarifLembur);
    }
}
```

- **Class PegawaiKontrak**

```
// Class PegawaiKontrak merupakan turunan (inheritance) dari class Pegawai
public class PegawaiKontrak extends Pegawai {

    // Atribut khusus pegawai kontrak
    private int potongan;

    // Constructor PegawaiKontrak
    // Menerima nama, hari kerja, upah per hari, dan potongan
    public PegawaiKontrak(String nama, int hariKerja, int upahPerHari, int
potongan) throws Exception {
        // Memanggil constructor dari class induk (Pegawai)
        super(nama, hariKerja, upahPerHari);

        // Validasi potongan tidak boleh negatif
```

```
        if (potongan < 0) {
            throw new Exception("Potongan tidak boleh negatif");
        }

        // Mengisi nilai potongan
        this.potongan = potongan;
    }

    // Method untuk menghitung gaji pegawai kontrak
    // Polimorfisme: method ini berbeda implementasi dengan PegawaiHarian
    @Override
    public int hitungGaji() {
        // Rumus gaji pegawai kontrak:
        // (hari kerja × upah per hari) − potongan
        return (getHariKerja() * getUpahPerHari()) - potongan;
    }
```

```
public class PegawaiKontrak extends Pegawai {  2 usages
    private int potongan;  2 usages

    public PegawaiKontrak(String nama, int hariKerja, int upahPerHari, int potongan) throws Exception {  1 usage
        super(nama, hariKerja, upahPerHari);
        if (potongan < 0) {
            throw new Exception("Potongan tidak boleh negatif");
        }
        this.potongan = potongan;
    }

    @Override  2 usages
    public int hitungGaji() {
        return (getHariKerja() * getUpahPerHari()) - potongan;
    }
}
```

- **Class Main**

```
import javax.swing.*;
import java.awt.*;

public class Main {

    public static void main(String[] args) {

        // Membuat frame utama aplikasi
        JFrame frame = new JFrame("Aplikasi Penggajian Pegawai");
        frame.setSize(400, 350);
```

```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setLayout(new GridLayout(6, 2, 5, 5));

// TextField untuk input data pegawai
JTextField txtNama = new JTextField();
JTextField txtHari = new JTextField();
JTextField txtUpah = new JTextField();
JTextField txtExtra = new JTextField(); // jam lembur / potongan

// Label yang akan berubah sesuai jenis pegawai
JLabel lblExtra = new JLabel("Jam Lembur:");

// ComboBox untuk memilih jenis pegawai
String[] jenis = {"Pegawai Harian", "Pegawai Kontrak"};
JComboBox<String> cbJenis = new JComboBox<>(jenis);

// Tombol untuk menghitung gaji
JButton btnHitung = new JButton("Hitung Gaji");

// Menambahkan komponen ke frame
frame.add(new JLabel("Nama Pegawai:"));
frame.add(txtNama);

frame.add(new JLabel("Jenis Pegawai:"));
frame.add(cbJenis);

frame.add(new JLabel("Hari Kerja:"));
frame.add(txtHari);

frame.add(new JLabel("Upah per Hari:"));
frame.add(txtUpah);

frame.add(lblExtra);
frame.add(txtExtra);

frame.add(new JLabel(""));
frame.add(btnHitung);

// Event ketika jenis pegawai diganti
// Label akan berubah sesuai jenis pegawai
```

```
cbJenis.addActionListener(e -> {
    if (cbJenis.getSelectedIndex() == 0) {
        lblExtra.setText("Jam Lembur:");
    } else {
        lblExtra.setText("Potongan:");
    }
});

// Event ketika tombol "Hitung Gaji" ditekan
btnHitung.addActionListener(e -> {
    try {
        // Mengambil input dari text field
        String nama = txtNama.getText();
        int hari = Integer.parseInt(txtHari.getText());
        int upah = Integer.parseInt(txtUpah.getText());
        int extra = Integer.parseInt(txtExtra.getText());

        int gaji;

        // Jika pegawai harian
        if (cbJenis.getSelectedIndex() == 0) {
            PegawaiHarian ph = new PegawaiHarian(nama, hari, upah, extra);
            gaji = ph.hitungGaji();
        }
        // Jika pegawai kontrak
        else {
            PegawaiKontrak pk = new PegawaiKontrak(nama, hari, upah, extra);
            gaji = pk.hitungGaji();
        }

        // Menampilkan hasil perhitungan gaji
        JOptionPane.showMessageDialog(frame,
                "Nama: " + nama +
                "\nGaji: Rp " + gaji);

    } catch (Exception ex) {
        // Menampilkan pesan error jika input tidak valid
        JOptionPane.showMessageDialog(frame,
                ex.getMessage(),
                "Error",
```

```
                    JOptionPane.ERROR_MESSAGE);
                }
            });

            // Menampilkan frame di tengah layar
            frame.setLocationRelativeTo(null);
            frame.setVisible(true);
        }
    }
```

```java
import javax.swing.*;
import java.awt.*;

public class Main {

    public static void main(String[] args) {
        JFrame frame = new JFrame( title: "Aplikasi Perhitungan upah Pegawai");
        frame.setSize( width: 400,  height: 350);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new GridLayout( rows: 6,  cols: 2,  hgap: 5,  vgap: 5));

        JTextField txtNama = new JTextField();
        JTextField txtHari = new JTextField();
        JTextField txtUpah = new JTextField();
        JTextField txtExtra = new JTextField();

        JLabel lblExtra = new JLabel( text: "Jam Lembur:");

        String[] jenis = {"Pegawai Harian", "Pegawai Kontrak"};
        JComboBox<String> cbJenis = new JComboBox<>(jenis);

        JButton btnHitung = new JButton( text: "Hitung Gaji");
```

```java
        JButton btnHitung = new JButton( text: "Hitung Gaji");

        // Layout
        frame.add(new JLabel( text: "Nama Pegawai:"));
        frame.add(txtNama);

        frame.add(new JLabel( text: "Jenis Pegawai:"));
        frame.add(cbJenis);

        frame.add(new JLabel( text: "Hari Kerja:"));
        frame.add(txtHari);

        frame.add(new JLabel( text: "Upah per Hari:"));
        frame.add(txtUpah);

        frame.add(lblExtra);
        frame.add(txtExtra);

        frame.add(new JLabel( text: ""));
        frame.add(btnHitung);
```

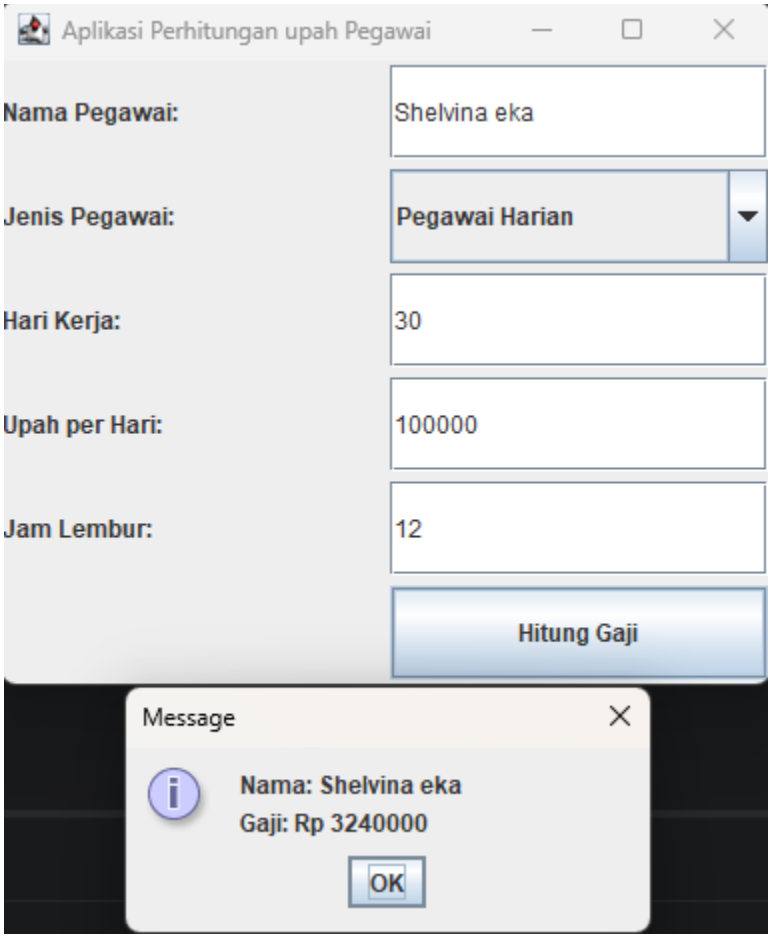
```
        // Ganti label sesuai jenis pegawai
        cbJenis.addActionListener( ActionEvent e -> {
            if (cbJenis.getSelectedIndex() == 0) {
                lblExtra.setText("Jam Lembur:");
            } else {
                lblExtra.setText("Potongan:");
            }
        });

        // Event tombol
        btnHitung.addActionListener( ActionEvent e -> {
            try {
                String nama = txtNama.getText();
                int hari = Integer.parseInt(txtHari.getText());
                int upah = Integer.parseInt(txtUpah.getText());
                int extra = Integer.parseInt(txtExtra.getText());

                int gaji;

                if (cbJenis.getSelectedIndex() == 0) {
```

```
                if (cbJenis.getSelectedIndex() == 0) {
                    PegawaiHarian ph = new PegawaiHarian(nama, hari, upah, extra);
                    gaji = ph.hitungGaji();
                } else {
                    PegawaiKontrak pk = new PegawaiKontrak(nama, hari, upah, extra);
                    gaji = pk.hitungGaji();
                }

                JOptionPane.showMessageDialog(frame,
                        message: "Nama: " + nama +
                                "\nGaji: Rp " + gaji);

            } catch (Exception ex) {
                JOptionPane.showMessageDialog(frame,
                        ex.getMessage(),
                        title: "Error",
                        JOptionPane.ERROR_MESSAGE);
            }
        });
```

```

            frame.setLocationRelativeTo(null);
            frame.setVisible(true);
        }
    }
```

G. PENJELASAN HASIL CAPTURE APLIKASI

Capture UI/Hasil Output



Penjelasan:

1. Input Data Pegawai

Pada tampilan aplikasi, pengguna telah mengisi data sebagai berikut:

- Nama Pegawai: Shelvina eka
- Jenis Pegawai: Pegawai Harian
- Hari Kerja: 30 hari
- Upah per Hari: Rp100.000
- Jam Lembur: 12 jam

Data tersebut dimasukkan melalui komponen input berupa *text field* dan *combo box* yang tersedia pada antarmuka aplikasi.

2. Proses Perhitungan Gaji

Karena jenis pegawai yang dipilih adalah Pegawai Harian, maka sistem melakukan perhitungan gaji menggunakan rumus:

Gaji = (Hari Kerja × Upah per Hari) + (Jam Lembur × Tarif Lembur)

Dengan asumsi tarif lembur yang digunakan dalam program adalah Rp20.000 per jam, maka perhitungan gaji adalah sebagai berikut:

- Gaji pokok = 30 × 100.000 = Rp3.000.000
- Upah lembur = 12 × 20.000 = Rp240.000

Total gaji = Rp3.240.000

3. Output Hasil

Setelah tombol **"Hitung Gaji"** ditekan, aplikasi menampilkan hasil perhitungan dalam bentuk **dialog box (JOptionPane)** yang berisi:

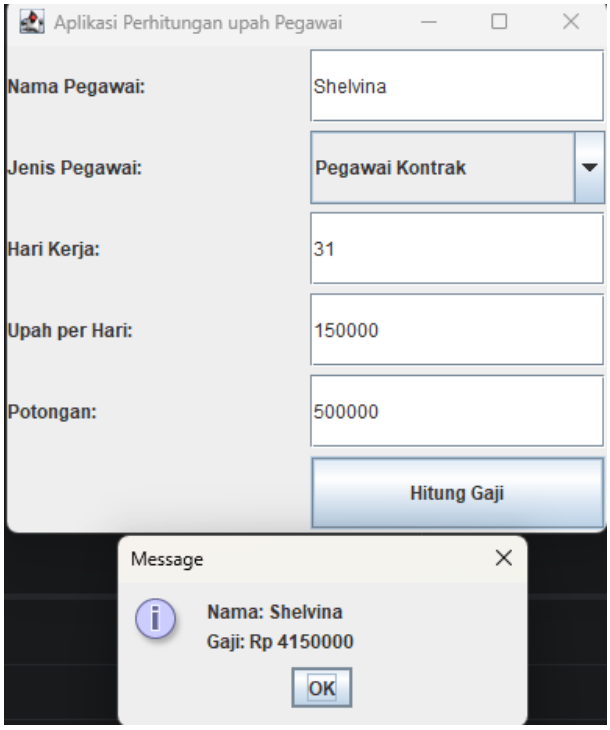
- Nama pegawai
- Total gaji yang diterima

Pada contoh ini, sistem menampilkan hasil:
Nama : Shelvina eka
Gaji : Rp 3.240.000

4. Kesimpulan

Berdasarkan hasil tampilan tersebut, dapat disimpulkan bahwa aplikasi berhasil:

- Menerima input data pegawai melalui GUI
- Menentukan jenis pegawai yang dipilih
- Melakukan perhitungan gaji secara otomatis sesuai aturan yang berlaku
- Menampilkan hasil perhitungan secara jelas dan informatif



Penjelasan:

1. Input Data Pegawai

Pada tampilan aplikasi, pengguna mengisi data pegawai dengan rincian sebagai berikut:

- Nama Pegawai: Shelvina
- Jenis Pegawai: Pegawai Kontrak
- Hari Kerja: 31 hari
- Upah per Hari: Rp150.000
- Potongan: Rp500.000

Input data dilakukan melalui komponen antarmuka berupa *text field* dan *combo box* yang tersedia pada aplikasi.

2. Proses Perhitungan Gaji

Karena jenis pegawai yang dipilih adalah **Pegawai Kontrak**, maka sistem melakukan perhitungan gaji menggunakan rumus:

Gaji = (Hari Kerja × Upah per Hari) − Potongan

Berdasarkan data yang dimasukkan:

- Gaji pokok = 31 × 150.000 = Rp4.650.000
- Potongan = Rp500.000
  Total gaji = Rp4.150.000

3. Output Hasil

Setelah pengguna menekan tombol **“Hitung Gaji”**, aplikasi menampilkan hasil perhitungan gaji dalam bentuk **dialog box (JOptionPane)** yang berisi:

- Nama pegawai
- Total gaji yang diterima

Hasil yang ditampilkan adalah:

Nama : Shelvina

Gaji : Rp 4.150.000

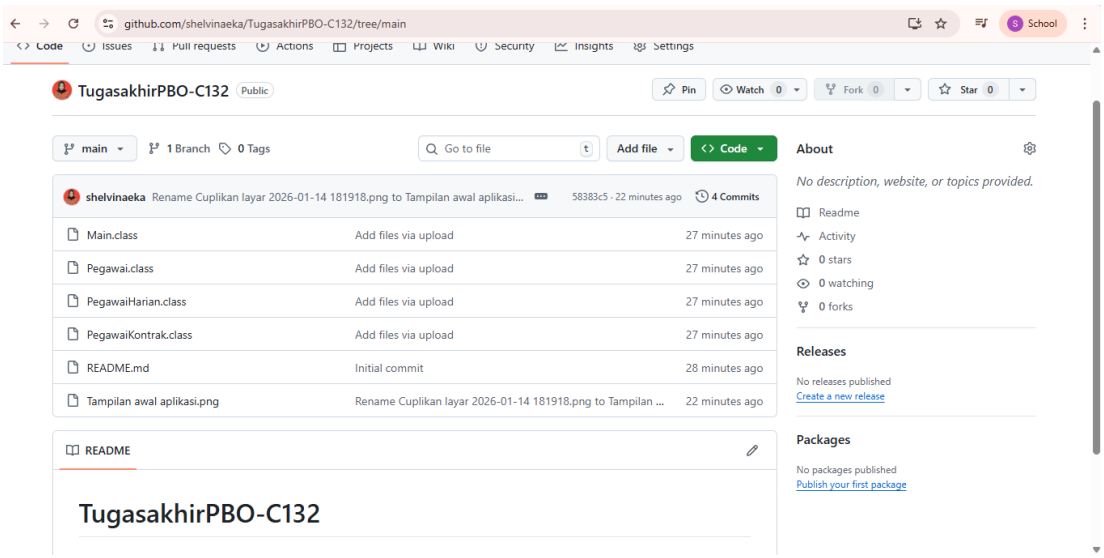
4. Kesimpulan

Berdasarkan hasil tampilan tersebut, dapat disimpulkan bahwa aplikasi:

- Berhasil membedakan perhitungan gaji berdasarkan jenis pegawai
- Menerapkan konsep **polimorfisme** melalui method hitungGaji()
- Menghitung dan menampilkan gaji pegawai kontrak secara tepat
- Menyajikan hasil perhitungan secara informatif melalui GUI

H. SCREENSHOT HASIL GITHUB

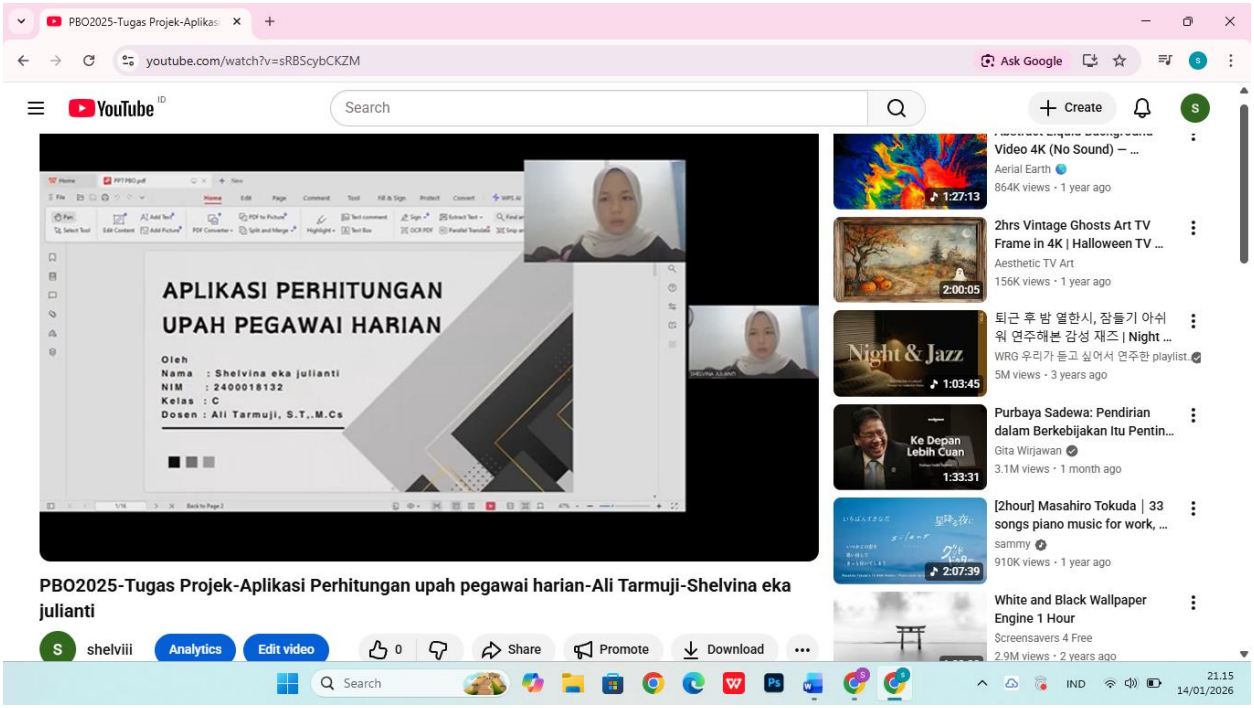
Link Github: https://github.com/shelvinaeka/TugasakhirPBO-C132



Pada tahap akhir pengembangan, seluruh source code aplikasi telah diunggah kerepository GitHub. Proses upload dilakukan menggunakan fitur commit dan push hingga versi final projek. Screenshot di atas menunjukkan repository telah berisi seluruh file program, struktur folder, tampilan aplikasi, serta riwayat commit sebagai bukti penyelesaian projek.

I. SCREENSHOT STATUS HASIL PUBLISH YOUTUBE

Link youtube: https://youtu.be/sRBScybCKZM



Aplikasi yang telah dibuat kemudian didemonstrasikan melalui video presentasi yang diunggah ke platform YouTube. Video tersebut menampilkan proses penggunaan aplikasi, mulai dari input data hingga hasil perhitungan gaji. Screenshot menunjukkan video telah berhasil dipublish dan dapat diakses secara publik sebagai bukti dokumentasi projek.

ANALISIS PENGERJAAN PROJEK

Pengerjaan projek aplikasi penggajian pegawai ini dilakukan dalam beberapa tahapan, yaitu perancangan konsep, pembuatan struktur class, implementasi GUI, pengujian, dan dokumentasi. Dari sisi waktu, projek dapat diselesaikan sesuai dengan jadwal yang ditentukan. Seluruh spesifikasi utama berhasil diimplementasikan dengan baik sesuai studi kasus. Dari sisi biaya, projek ini tidak memerlukan biaya tambahan karena menggunakan perangkat lunak gratis dan open-source. Kendala utama yang dihadapi adalah pengaturan event pada GUI dan validasi input pengguna, namun dapat diatasi dengan penerapan exception handling. Ke depan, aplikasi ini dapat dikembangkan dengan penambahan fitur penyimpanan data pegawai ke database dan laporan gaji dalam bentuk file.