LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK
MODUL IV
OBJECT ORIENTED PROGRAMMING JAVA I

Disusun Oleh:

Mohamad Dandung Sadat

PROGRAM STUDI ILMU KOMPUTER

FAKULTAS SAINS DAN KOMPUTER

UNIVERSITAS PERTAMINA

2025

I. Pendahuluan

Di era industri kontemporer, efektivitas dan ketepatan dalam pengelolaan sumber daya manusia (SDM) sangat penting, terutama dalam hal perhitungan gaji karyawan. Sistem penggajian manual yang masih umum dipakai oleh perusahaan kecil hingga menengah sering menyebabkan beragam masalah, seperti kesalahan dalam perhitungan, keterlambatan dalam pembayaran, serta kurangnya transparansi antara perusahaan dan karyawan. Oleh karena itu, diperlukan sistem otomatis yang dapat mendukung bagian keuangan dan administrasi dalam menghitung gaji dengan mempertimbangkan berbagai variabel seperti total jam kerja, shift kerja, tingkat kehadiran, serta kompensasi lembur.

Bahasa pemrograman Java sering dipilih sebagai opsi utama dalam pengembangan sistem penggajian karena sifatnya yang adaptif, memiliki dokumentasi yang komprehensif, serta mendukung paradigma pemrograman yang berorientasi objek. Program yang dihasilkan dalam laporan ini, yaitu GajiKaryawanPabrik, adalah penerapan dari sistem penggajian berbasis Java yang mengalkulasi gaji karyawan berdasarkan shift kerja (pagi, siang, malam), total jam kerja per minggu, serta jumlah hari ketidakhadiran tanpa keterangan. Selain itu, program ini juga menghitung pengurangan jika jam kerja terlalu rendah dan menambahkan gaji lembur jika jam kerja melebihi 40 jam dalam seminggu.

Beberapa penelitian mendukung signifikansi pengembangan aplikasi seperti ini. Pratama (2022) di dalam publikasinya mengungkapkan bahwa pemanfaatan Java untuk sistem penggajian terbukti berhasil dalam meningkatkan efisiensi proses pembayaran dan pencatatan keuangan perusahaan. Hal serupa juga diungkapkan oleh Pratama et al. (2024), yang menyatakan bahwa penerapan metodologi waterfall dalam pengembangan aplikasi desktop Java memberikan kerangka kerja yang sistematis dan mudah untuk diuji. Oleh sebab itu, pengembangan program seperti ini tidak hanya penting dalam konteks akademis, tetapi juga aplikatif dalam dunia kerja langsung.

II. Variabel

No	Nama Variabel	Tipe data	Fungsi
1	*nama*	String	Menyimpan nama mahasiswa yang diinput user
2	Nim	String	Menyimpan Nomor Induk Mahasiswa
3	Jurusan	String	Menyimpan jurusan mahasiswa
4	Ipk	Double	Menyimpan Indeks Prestasi Kumulatif mahasiswa
5	kodeProduk	String	Menyimpan kode unik produk
6	Nama produk	String	Menyimpan nama produk
7	harga	Double	Menyimpan harga per unit produk
8	stok	Int	Menyimpan jumlah stok produk yang tersedia
9	idPelanggan	String	Menyimpan ID unik pelanggan
10	email	String	Menyimpan alamat email pelanggan
11	saldo	Double	Menyimpan jumlah saldo akun pelanggan
12	idTransaksi	String	Menyimpan ID unik transaksi
13	pelanggan	Pelanggan	Menyimpan objek pelanggan yang melakukan transaksi
14	produk	Produk	Menyimpan objek produk yang dibeli
15	jumlahBeli	Int	Menyimpan jumlah produk yang dibeli dalam transaksi
16	totalHarga	double	Menyimpan total harga transaksi

III. Constructor dan Method

No	Nama Metode	Jenis Metode	Fungsi
1	Mahasiswa(String nama, String nim, String jurusan, double ipk)	*(constructor)*	Menginisialisasi objek Mahasiswa dengan nama, NIM, jurusan, dan IPK.
2	tampilkanInfo()	*(Procedural)*	Mencetak semua informasi detail mahasiswa.
3	cekLulus()	*(Functional)*	Mengembalikan true jika IPK >= 3.0, false jika tidak.
4	Produk(String kodeProduk, String namaProduk, double harga, int stok)	*(Constructor)*	Menginisialisasi objek Produk dengan kode, nama, harga, dan stok.
5	tampilkanInfoProduk()	*(Procedural)*	Menampilkan detail (kode, nama, harga, stok) produk.
6	kurangiStok(int jumlah)	*(Functional)*	Mengurangi stok produk sejumlah jumlah jika tersedia. Mengembalikan true jika berhasil, false jika gagal.
7	Pelanggan(String idPelanggan, String nama, String email, double saldo)	*(Constructor*	Menginisialisasi objek Pelanggan dengan ID, nama, email, dan saldo awal.
8	tampilkanInfoPelanggan()	*(Procedural)*	Menampilkan informasi pribadi (ID, nama, email, saldo) pelanggan.
9	topUpSaldo(double jumlah)	*(Procedural)*	Menambah saldo akun pelanggan sejumlah jumlah.
10	kurangiSaldo(double jumlah)	*(Functional)*	Mengurangi saldo akun pelanggan sejumlah jumlah jika saldo mencukupi. Mengembalikan true jika berhasil, false jika gagal.
11	Transaksi(String idTransaksi, Pelanggan	*Constructor*	Menginisialisasi objek Transaksi dengan ID,

	pelanggan, Produk produk, int jumlahBeli)		pelanggan, produk, jumlah beli, dan menghitung total harga.
12	prosesTransaksi()	*Functional*	Memproses transaksi: mengecek saldo dan stok, mengurangi keduanya jika cukup. Mengembalikan true jika transaksi berhasil, false jika gagal.
13	tampilkanDetailTransaksi()	*Procedural*	Menampilkan detail (pelanggan, produk, jumlah, total harga) transaksi.
14	hitungDiskon(double totalBiayaDasar, int durasiSewaHari)	*Functional*	Menghitung total biaya setelah diskon 10% jika durasi sewa lebih dari 5 hari.
15	formatMataUang(double amount)	*Functional*	Mengonversi angka ke format mata uang Rupiah.

IV. Dokumentasi dan Pembahasan Code

4.1.Class Produk

4.1.1.Attribut Produk

```
public class Produk {
    private String kodeProduk;
    private String namaProduk;
    private double harga;
    private int stok;
```

Baris-baris ini adalah deklarasi kelas Produk dan atribut-atributnya. Kelas Produk merepresentasikan data produk di toko online.

4.1.2.Constructor Produk

```
public Produk(String kodeProduk, String namaProduk, double harga, int stok) {
    this.kodeProduk = kodeProduk;
    this.namaProduk = namaProduk;
    this.harga = harga;
    this.stok = stok;
}
```

Ini adalah konstruktor untuk kelas Produk. public Produk(...) adalah metode khusus yang dipanggil saat objek Produk baru dibuat

4.1.3.Getter dan Setter Produk

```
public String getKodeProduk() {
    return kodeProduk;
}
```

Metode-metode seperti public String getKodeProduk() adalah getter. Metode ini digunakan untuk mengizinkan akses baca ke nilai atribut private

4.1.4. Metode Procedural Produk

```
public void tampilkanInfoProduk() {
    System.out.println("--- Detail Produk ---");
    System.out.println("Kode Produk : " + kodeProduk);
    System.out.println("Nama Produk : " + namaProduk);
    System.out.println("Harga       : Rp" + String.format("%.2f", harga));
    System.out.println("Stok        : " + stok);
    System.out.println("--------------------");
}
```

Metode public void tampilkanInfoProduk() adalah metode procedural. Metode ini mencetak detail lengkap dari sebuah objek Produk ke konsol, termasuk kode produk, nama, harga (diformat dengan dua angka desimal), dan stok.

4.1.5.Metode Fungsional

```
public boolean kurangiStok(int jumlah) {
    if (this.stok >= jumlah) {
        this.stok -= jumlah;
        return true;
    } else {
        System.out.println("Stok " + namaProduk
        return false;
    }
}
```

Metode ini mencoba mengurangi stok produk sejumlah jumlah yang diminta.

4.2.Class Pelanggan

4.2.1.Attribut Pelanggan

```
public class Pelanggan {
    private String idPelanggan;
    private String nama;
    private String email;
    private double saldo;
```

Deklarasi kelas Pelanggan dan atribut-atributnya. Kelas Pelanggan merepresentasikan data pengguna toko online.

4.2.2.Konstruktor Pelanggan

```
public Pelanggan(String idPelanggan, String nama, String email, double saldo) {
    this.idPelanggan = idPelanggan;
    this.nama = nama;
    this.email = email;
    this.saldo = saldo;
}
```

Konstruktor untuk kelas Pelanggan. Metode ini dipanggil saat objek Pelanggan baru dibuat. Ini menginisialisasi atribut idPelanggan, nama, email, dan saldo dengan nilai yang diteruskan sebagai parameter.

4.2.3.Metode Procedural Pelanggan

```
public void tampilkanInfoPelanggan() {
    System.out.println("--- Informasi Pelanggan ---");
    System.out.println("ID Pelanggan: " + idPelanggan);
    System.out.println("Nama        : " + nama);
    System.out.println("Email       : " + email);
    System.out.println("Saldo Akun  : Rp" + String.format("%.2f", saldo));
    System.out.println("---------------------------");
}
```

Metode public void tampilkanInfoPelanggan() adalah metode procedural. Metode ini mencetak semua informasi detail pelanggan (ID, nama, email, saldo) ke konsol. Tipe kembaliannya adalah void karena tidak mengembalikan nilai.

```
public void topUpSaldo(double jumlah) {
    if (jumlah > 0) {
        this.saldo += jumlah;
        System.out.println("Top-up saldo sebesar Rp" + String.format(
    } else {
        System.out.println("Jumlah top-up harus positif.");
    }
}
```

Metode public void topUpSaldo(double jumlah) adalah metode procedural. Metode ini berfungsi untuk menambah saldo pelanggan. Ini memeriksa apakah jumlah top-up positif

4.2.4.Metode Fungsional Pelanggan

```
public boolean kurangiSaldo(double jumlah) {
    if (this.saldo >= jumlah) {
        this.saldo -= jumlah;
        return true;
    } else {
        System.out.println("Saldo tidak mencukupi. Saldo saat ini: Rp" + String
        return false;
    }
}
```

Metode public boolean kurangiSaldo(double jumlah) adalah metode fungsional. Metode ini mencoba mengurangi saldo pelanggan sejumlah jumlah.

4.3.Kelas Transaksi

4.3.1.Attribut Transaksi

```
public class Transaksi {
    private String idTransaksi;
    private Pelanggan pelanggan;
    private Produk produk;
    private int jumlahBeli;
    private double totalHarga;
```

Deklarasi kelas Transaksi dan atribut-atributnya. Kelas Transaksi merepresentasikan sebuah transaksi pembelian.

4.3.2.Konstruktor Transaksi

```
public Transaksi(String idTransaksi, Pelanggan pelanggan, Produk produk,
    this.idTransaksi = idTransaksi;
    this.pelanggan = pelanggan;
    this.produk = produk;
    this.jumlahBeli = jumlahBeli;
    this.totalHarga = produk.getHarga() * jumlahBeli;
}
```

Konstruktor untuk kelas Transaksi. Konstruktor ini menginisialisasi ID transaksi, objek pelanggan yang terlibat, objek produk yang dibeli, dan jumlah produk yang dibeli.

4.3.3.Metode Fungsional Transaksi

```
public boolean prosesTransaksi() {
    System.out.println("\n--- Memproses Transaksi " + idTransaksi + " ---");
    if (pelanggan.getSaldo() < totalHarga) {
        System.out.println("Transaksi GAGAL: Saldo pelanggan tidak mencukupi.
        return false;
    }
```

Metode public boolean prosesTransaksi() adalah metode fungsional yang merupakan inti dari manajemen transaksi. Bagian ini pertama-tama mencetak pesan bahwa transaksi sedang diproses.

```
    if (produk.getStok() < jumlahBeli) {
        System.out.println("Transaksi GAGAL: Stok produk "
        return false;
    }
```

Bagian ini adalah validasi kedua. Ini mengecek apakah stok dari produk yang akan dibeli mencukupi untuk jumlahBeli yang diinginkan. Jika tidak, pesan gagal dicetak dan metode mengembalikan false.

```
    if (pelanggan.kurangiSaldo(totalHarga) && produk.kurangiStok(jumlahBeli)) {
        System.out.println("Transaksi BERHASIL!");
        tampilkanDetailTransaksi();
        return true;
    } else {
        System.out.println("Transaksi GAGAL: Terjadi masalah saat mengurangi sa
        return false;
    }
}
```

Ini adalah logika utama keberhasilan transaksi. Jika kedua validasi di atas lolos (saldo cukup dan stok cukup), maka metode kurangiSaldo() dari objek pelanggan dan kurangiStok() dari objek produk akan dipanggil.

4.3.4.Metode Procedural Transaksi

```
public void tampilkanDetailTransaksi() {
    System.out.println("\n--- Detail Transaksi " + idTransaksi + " ---");
    System.out.println("Pelanggan   : " + pelanggan.getNama() + " (ID: " + pela
    System.out.println("Produk      : " + produk.getNamaProduk() + " (Kode: " +
    System.out.println("Jumlah Beli : " + jumlahBeli);
    System.out.println("Total Harga : Rp" + String.format("%.2f", totalHarga));
    System.out.println("---------------------------------");
}
```

Metode public void tampilkanDetailTransaksi() adalah metode procedural. Metode ini mencetak ringkasan detail transaksi ke konsol, termasuk informasi pelanggan, produk yang dibeli, jumlah, dan total harga. Tipe kembaliannya adalah void.

4.4.Kelas Main

4.4.1.Metode Main

```
public class Main {
    public static void main(String[] args) {
```

Ini adalah kelas Main dan metode main-nya. Seperti yang dijelaskan sebelumnya, public static void main(String[] args) adalah titik awal eksekusi program Java. Semua skenario pengujian dan interaksi antar objek akan terjadi di sini.

4.4.2.Membuat Objek Kelas Produk

```
System.out.println("===== MANAJEMEN PRODUK =====");
Produk pensil = new Produk("P001", "Pensil 2B", 3500.0, 50);
Produk bukuTulis = new Produk("B002", "Buku Tulis A5", 7000.0, 30);
Produk penghapus = new Produk("H003", "Penghapus Faber", 2000.0, 10);
```

Bagian ini mendemonstrasikan Manajemen Produk. Baris-baris ini membuat tiga objek Produk baru (pensil, bukuTulis, penghapus) dengan menggunakan konstruktor Produk(), menginisialisasi mereka dengan kode, nama, harga, dan stok awal.

```
pensil.tampilkanInfoProduk();
bukuTulis.tampilkanInfoProduk();
penghapus.tampilkanInfoProduk();
```

Baris-baris ini memanggil metode tampilkanInfoProduk() pada masing-masing objek Produk yang telah dibuat, untuk menampilkan detail awal dari setiap produk ke konsol.

```
System.out.println("\n--- Memperbarui Stok Pensil ---");
pensil.setStok(60);
pensil.tampilkanInfoProduk();
```

Bagian ini menunjukkan fitur pembaruan produk. Metode setStok(60) dipanggil pada objek pensil untuk mengubah nilai stoknya menjadi 60. Kemudian, tampilkanInfoProduk() dipanggil lagi untuk menunjukkan stok yang sudah diperbarui.

4.4.3.Membuat Objek Kelas Pelanggan

```
System.out.println("\n===== MANAJEMEN PELANGGAN =====");
Pelanggan pelanggan1 = new Pelanggan("CST001", "Budi Santoso",
Pelanggan pelanggan2 = new Pelanggan("CST002", "Siti Aminah",
```

Bagian ini mendemonstrasikan Manajemen Pelanggan. Dua objek Pelanggan baru (pelanggan1 dan pelanggan2) dibuat menggunakan konstruktor Pelanggan(), menginisialisasi mereka dengan ID, nama, email, dan saldo awal.

```
pelanggan1.tampilkanInfoPelanggan();
pelanggan2.tampilkanInfoPelanggan();
```

Memanggil metode tampilkanInfoPelanggan() pada kedua objek Pelanggan untuk menampilkan informasi detail awal mereka.

```
System.out.println("\n--- Top-up Saldo Pelanggan 1 ---");
pelanggan1.topUpSaldo(10000.0);
pelanggan1.tampilkanInfoPelanggan();
```

Bagian ini menunjukkan fitur top-up saldo. Metode topUpSaldo(10000.0) dipanggil pada pelanggan1 untuk menambahkan Rp10.000,00 ke saldonya. Kemudian, info pelanggan ditampilkan lagi untuk mengonfirmasi perubahan saldo.

4.4.4.Membuat Objek Kelas Transaksi

```
System.out.println("\n===== MANAJEMEN TRANSAKSI =====");

// Skenario 1: Transaksi Berhasil
Transaksi tr1 = new Transaksi("TRX001", pelanggan1, pensil, 3);
tr1.prosesTransaksi();
System.out.println("\n--- Info Terbaru Setelah Transaksi TRX001 ---");
pelanggan1.tampilkanInfoPelanggan();
pensil.tampilkanInfoProduk();
```

Ini adalah Skenario 1 untuk Manajemen Transaksi. Sebuah objek Transaksi bernama tr1 dibuat, merepresentasikan pembelian 3 pensil oleh pelanggan1. Metode tr1.prosesTransaksi() dipanggil untuk menjalankan logika transaksi.

```
// Skenario 2: Transaksi Gagal (Saldo tidak cukup)
Transaksi tr2 = new Transaksi("TRX002", pelanggan2, bukuTulis, 5);
tr2.prosesTransaksi();
System.out.println("\n--- Info Terbaru Setelah Transaksi TRX002 (Gagal) ---");
pelanggan2.tampilkanInfoPelanggan();
bukuTulis.tampilkanInfoProduk();
```

Ini adalah Skenario 2, transaksi yang diharapkan gagal karena saldo pelanggan tidak mencukupi. Objek Transaksi tr2 dibuat. prosesTransaksi() dipanggil, dan program akan menampilkan pesan gagal karena pelanggan2 tidak memiliki cukup saldo untuk membeli 5 bukuTulis.

```
// Skenario 3: Transaksi Gagal (Stok tidak cukup)
Transaksi tr3 = new Transaksi("TRX003", pelanggan1, penghapus, 15);
tr3.prosesTransaksi();
System.out.println("\n--- Info Terbaru Setelah Transaksi TRX003 (Gagal) ---");
pelanggan1.tampilkanInfoPelanggan();
penghapus.tampilkanInfoProduk();
```

Ini adalah Skenario 3, transaksi yang diharapkan gagal karena stok produk tidak mencukupi. Objek Transaksi tr3 dibuat. prosesTransaksi() dipanggil, dan program akan menampilkan pesan gagal karena stok penghapus (hanya 10) tidak cukup untuk jumlah beli 15. Info pelanggan dan produk ditampilkan, menunjukkan tidak ada perubahan.

```
        // Skenario 4: Transaksi Berhasil (dengan sisa stok yang sedikit)
        System.out.println("\n--- Top-up Saldo Pelanggan 2 untuk mencoba lagi ---");
        pelanggan2.topUpSaldo(50000.0);
        Transaksi tr4 = new Transaksi("TRX004", pelanggan2, penghapus, 8);
        tr4.prosesTransaksi();
        System.out.println("\n--- Info Terbaru Setelah Transaksi TRX004 ---");
        pelanggan2.tampilkanInfoPelanggan();
        penghapus.tampilkanInfoProduk();
    }
}
```

Ini adalah Skenario 4. Pertama, pelanggan2 di-top-up saldonya agar mencukupi. Kemudian, objek Transaksi tr4 dibuat untuk pembelian 8 penghapus oleh pelanggan2. Metode prosesTransaksi() dipanggil. Kali ini, transaksi diharapkan berhasil karena saldo sudah cukup dan stok penghapus (yang masih 10 dari awal) mencukupi untuk 8. Info terbaru pelanggan dan produk ditampilkan untuk menunjukkan pengurangan saldo dan stok.

V. Kesimpulan

Program GajiKaryawanPabrik yang dibuat dengan bahasa pemrograman Java ini telah berhasil mengotomatiskan proses penghitungan gaji karyawan secara terstruktur dan rasional. Program ini memperhitungkan berbagai elemen penting dalam penggajian, seperti tarif per jam sesuai shift kerja, total jam kerja mingguan, jumlah hari ketidakhadiran tanpa alasan, serta kompensasi untuk lembur. Perhitungan yang dilakukan oleh program sangat berkaitan dengan praktik industri, khususnya dalam konteks perusahaan manufaktur yang menggunakan sistem kerja shift. Dengan penerapan validasi input melalui teknik exception handling, program ini juga dapat mencegah kesalahan pengguna saat memasukkan data, sehingga menjadi dapat diandalkan dan efisien.

Selain efisiensi, program ini juga memberikan keterbukaan dalam pelaporan remunerasi pegawai. Hasil keluaran berupa tabel yang menampilkan ID, nama, shift, jam kerja, jumlah ketidakhadiran, dan total gaji bersih, sehingga informasi tersebut mudah dibaca dan dilacak. Walaupun begitu, program ini tetap memiliki batasan seperti belum adanya penyimpanan data yang permanen (database atau file), tidak adanya sistem otentikasi pengguna, serta belum tersedianya antarmuka pengguna grafis (GUI). Sebagai akibatnya, pengembangan lebih lanjut sangat disarankan agar program ini dapat diterapkan di dunia usaha yang sesungguhnya.

Secara keseluruhan, program ini adalah prototipe yang sangat menjanjikan untuk dikembangkan menjadi sistem informasi penggajian yang lebih komprehensif. Seperti yang diungkapkan oleh studi-studi sebelumnya, sistem penggajian yang berbasis Java tidak hanya menyederhanakan proses administrasi internal, tetapi juga berpotensi meningkatkan kepercayaan serta kepuasan karyawan terhadap manajemen perusahaan.

VI. Daftar Pustaka

Pratama, Y. (2022, September). *Perancangan aplikasi penggajian karyawan menggunakan Java*. *Jurnal Manajamen Informatika Jayakarta, 2*(4), 347–354. https://doi.org/10.52362/jmijayakarta.v2i4.884

Pratama, M. B. A., Hilabi, S. S., Ihsan, M. M., Ferdiansyah, I., & Nizar, H. S. (2024). *Application of the Waterfall Method in Creating Payroll Applications Based on Java Netbeans*. *Jurnal Multimedia dan Teknologi Informasi (Jatilima), 6*(1). https://doi.org/10.54209/jatilima.v6i01.432

Neliti. (n.d.). *Sistem Informasi Penggajian Karyawan Berbasis Java Desktop*. Retrieved June 2025, from https://www.neliti.com/publications/466691/…