LAPORAN PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK

MODUL III

PEMROGRAMAN JAVA DASAR II

Disusun Oleh:

Mohamad Dandung Sadat

PROGRAM STUDI ILMU KOMPUTER

FAKULTAS SAINS DAN KOMPUTER

UNIVERSITAS PERTAMINA

2025

I. Pendahuluan

Di zaman industri kontemporer, efektivitas dalam pengelolaan sumber daya manusia (SDM) menjadi elemen yang sangat krusial dalam mendukung produktivitas dan kelangsungan operasional perusahaan. Salah satu elemen penting dalam manajemen SDM adalah sistem remunerasi atau payroll system. Sistem ini berfungsi untuk menghitung dan mendistribusikan gaji karyawan dengan akurat, adil, serta sesuai dengan ketentuan yang berlaku. Perhitungan gaji tidak hanya sekadar merangkum jam kerja, tetapi juga perlu memperhatikan berbagai faktor seperti shift kerja, lembur, kehadiran, dan pengurangan akibat ketidakhadiran. Oleh sebab itu, sistem remunerasi yang efisien dan tepat sangat diperlukan, terutama di sektor industri manufaktur yang memiliki pola kerja yang rumit dan dinamis.

Isu yang kerap timbul dalam sistem penggajian manual di pabrik mencakup kesalahan dalam memasukkan data, perhitungan yang tidak tepat, keterlambatan dalam pembayaran, serta kurangnya transparansi pada laporan. Oleh karena itu, diperlukan solusi yang berlandaskan teknologi untuk mengotomatiskan proses itu. Salah satu metode yang diterapkan adalah pengembangan software berbasis Java yang dapat menghitung gaji pegawai berdasarkan sejumlah parameter tertentu secara terstruktur.

Program GajiKaryawanPabrik yang dibuat dengan bahasa pemrograman Java ini adalah contoh sederhana dari penerapan sistem penggajian secara digital. Program ini dibuat untuk mendukung bagian administrasi atau HRD dalam menghitung gaji pegawai berdasarkan shift kerja (pagi, siang, malam), total jam kerja per minggu, total hari absen tanpa alasan, serta memberikan tambahan gaji lembur jika diperlukan. Pemilihan bahasa Java dilakukan karena karakteristiknya yang dapat dipindahkan, tangguh, serta memiliki kemampuan yang baik dalam pengelolaan input/output dan penanganan pengecualian.

Dengan sistem ini, perhitungan gaji menjadi lebih jelas, dapat dipantau, dan efisien, serta mengurangi peluang terjadinya kesalahan manusia. Program ini juga bisa digunakan sebagai landasan untuk pengembangan lanjutan menjadi sistem penggajian berbasis database atau aplikasi web yang lebih rumit.

II. Variabel

No	Nama Variabel	Tipe data	Fungsi
1	*Scanner*	Scanner	Menerima input dari pengguna
2	jumlahKaryawan	Int	Menyimpan jumlah karyawan yang datanya akan dimasukkan
3	inputValid	Boolean	Menandai apakah input dari pengguna valid atau tidak
4	idKaryawan	String[]	Menyimpan ID masing-masing karyawan
5	namaKaryawan	String[]	Menyimpan nama masing-masing karyawan
6	shiftKaryawan	String[]	Menyimpan shift kerja masing-masing karyawan
7	jamKerjaMingguan	Int[]	Menyimpan total jam kerja dalam seminggu untuk setiap karyawan |
8	hariAbsen	Int[]	Menyimpan jumlah hari absen tanpa alas an
9	gajiBersih	Double[]	Menyimpan hasil perhitungan gaji bersih untuk setiap karyawan
10	tarifPerJam	Double	Menyimpan tarif upah per jam berdasarkan shift kerja
11	gajiPokok	Double	Menyimpan gaji pokok (jam kerja x tarif per jam)
12	potonganAbsen	Double	Menyimpan total potongan gaji karena absen
13	gajiLembur	Double	Menyimpan total tambahan gaji karena lembur
14	potonganKurangKerja	double	Menyimpan potongan karena jam kerja di bawah 30 jam

III. Constructor dan Method

No	Nama Metode	Jenis Metode	Fungsi
1	Main	*(Procedural)*	Merupakan titik awal eksekusi program yang mengatur seluruh alur kerja dari input, perhitungan hingga output
2	nextInt, nextLine, dll (dalam Scanner)	Built in	Digunakan untuk membaca input pengguna dari konsol

IV. Dokumentasi dan Pembahasan Code

4.1. Struktur Program dan Inisialisasi

```java
package Tht;

import java.util.InputMismatchException;
import java.util.Scanner;

public class GajiKaryawanPabrik {

    Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println(x:"=== Program Penghitung Gaji Karyawan Pabrik ===");

```

Program dimulai dengan mengimpor kelas-kelas yang diperlukan:

- java.util.InputMismatchException untuk menangani jenis input yang tidak sesuai dan
- java.util.Scanner untuk menerima input dari pengguna.
- package Tht;: Baris ini menentukan paket tempat kelas berada.

- import java.util.Input MismatchException;: Mengimpor kelas pengecualian yang digunakan untuk menangani kasus di mana pengguna memasukkan input dengan tipe yang tidak diharapkan (misalnya, teks alih-alih angka).

- import java.util.Scanner;: Mengimpor kelas Scanner, yang penting untuk membaca input dari konsol.

- public class GajiKaryawanPabrik { ... }: Mendefinisikan kelas utama program.

- public static void main(String[] args) { ... }: Ini adalah titik masuk program Java di mana eksekusi dimulai.

- Scanner scanner = new Scanner(System.in);: Sebuah objek dari kelas Scanner dibuat, memungkinkan program untuk membaca input dari standard input stream (System.in), yang biasanya adalah keyboard.

4.2. Meminta Jumlah Karyawan

```
int jumlahKaryawan = 0;
boolean inputValid = false;
while (!inputValid) {
    try {
        System.out.print(s:"Masukkan jumlah karyawan yang akan dihitung gajinya: ");
        jumlahKaryawan = scanner.nextInt();
        if (jumlahKaryawan <= 0) {
            System.out.println(x:"Jumlah karyawan harus lebih dari 0.");
        } else {
            inputValid = true;
        }
    } catch (InputMismatchException e) {
        System.out.println(x:"Input tidak valid. Harap masukkan angka bulat.");
        scanner.next(); // Membersihkan input yang salah
    }
}
scanner.nextLine(); // Membersihkan newline setelah nextInt()
```

Program pertama-tama meminta pengguna untuk memasukkan total jumlah karyawan yang gajinya akan dihitung. Ini mencakup validasi input yang kuat untuk memastikan bahwa angka bulat positif dimasukkan.

- while (!inputValid): Loop ini terus berjalan sampai input yang valid (bilangan bulat positif untuk jumlahKaryawan) diterima.
- Blok try-catch: Ini digunakan untuk penanganan pengecualian.

- try: Mencoba membaca bilangan bulat menggunakan scanner.nextInt().
- catch (InputMismatchException e): Jika pengguna memasukkan sesuatu yang bukan bilangan bulat, InputMismatchException akan tertangkap. Pesan kesalahan ditampilkan, dan scanner.next() dipanggil untuk mengonsumsi input yang tidak valid, mencegah loop tak terbatas.
- Validasi Input: if (jumlahKaryawan <= 0) memeriksa apakah jumlah karyawan yang dimasukkan adalah positif.
- scanner.nextLine();: Setelah nextInt(), karakter newline akan tetap berada di buffer input. Baris ini mengonsumsi newline yang tersisa tersebut, mencegah masalah dengan panggilan scanner.nextLine() berikutnya.

4.3. Penyimpanan Data Menggunakan Array

```
String[] idKaryawan = new String[jumlahKaryawan];
String[] namaKaryawan = new String[jumlahKaryawan];
String[] shiftKaryawan = new String[jumlahKaryawan];
int[] jamKerjaMingguan = new int[jumlahKaryawan];
int[] hariAbsen = new int[jumlahKaryawan];
double[] gajiBersih = new double[jumlahKaryawan];
```

Array dideklarasikan untuk menyimpan detail setiap karyawan. Ukuran *array* ini ditentukan oleh jumlahKaryawan yang diperoleh dari pengguna.

Setiap *array* akan menyimpan sepotong informasi spesifik untuk semua karyawan, diindeks dari 0 hingga jumlahKaryawan - 1.

- idKaryawan: ID Karyawan (String).
- namaKaryawan: Nama Karyawan (String).
- shiftKaryawan: Shift Kerja (String).
- jamKerjaMingguan: Jam Kerja Mingguan (int).
- hariAbsen: Jumlah Hari Absen Tanpa Alasan (int).
- gajiBersih: Gaji Bersih (double).

4.4. Loop Input Data Karyawan

Sebuah *loop* for berulang sebanyak jumlahKaryawan kali, meminta detail untuk setiap karyawan. Setiap bidang input juga menyertakan validasinya sendiri untuk memastikan kualitas data.

```
for (int i = 0; i < jumlahKaryawan; i++) {
    System.out.println("\n--- Data Karyawan ke-" + (i + 1) + " ---");

    System.out.print(s:"ID Karyawan: ");
    idKaryawan[i] = scanner.nextLine();

    System.out.print(s:"Nama Karyawan: ");
    namaKaryawan[i] = scanner.nextLine();
```

4.4.1.Validasi Input Shift

```
inputValid = false;
while (!inputValid) {
    System.out.print(s:"Shift Kerja (Pagi/Siang/Malam): ");
    String shiftInput = scanner.nextLine().trim();
    if (shiftInput.equalsIgnoreCase(anotherString:"Pagi") ||
        shiftInput.equalsIgnoreCase(anotherString:"Siang") ||
        shiftInput.equalsIgnoreCase(anotherString:"Malam")) {
        shiftKaryawan[i] = shiftInput;
        inputValid = true;
    } else {
        System.out.println(x:"Shift tidak valid. Harap masukkan 'Pagi', 'Siang', atau 'Malam'.");
    }
}
```

Ini memastikan shift yang dimasukkan adalah "Pagi", "Siang", atau "Malam" (tidak peka huruf besar-kecil karena equalsIgnoreCase). trim() menghapus spasi di awal/akhir.

4.4.2.Validasi Total Jam Kerja Mingguan

```
inputValid = false;
while (!inputValid) {
    try {
        System.out.print(s:"Total Jam Kerja dalam Seminggu: ");
        jamKerjaMingguan[i] = scanner.nextInt();
        if (jamKerjaMingguan[i] < 0 || jamKerjaMingguan[i] > (7 * 24)) { // Maksimal 7 hari x 24 jam
            System.out.println(x:"Jam kerja tidak masuk akal. Harap masukkan angka antara 0 dan 168.");
        } else {
            inputValid = true;
        }
    } catch (InputMismatchException e) {
        System.out.println(x:"Input tidak valid. Harap masukkan angka bulat untuk jam kerja.");
        scanner.next(); // Membersihkan input yang salah
    }
}
scanner.nextLine(); // Membersihkan newline setelah nextInt()
```

Menggunakan try-catch untuk InputMismatchException dan memvalidasi bahwa jam kerja antara 0 dan 168 (7 hari * 24 jam).

4.4.3.Validasi Hari Absen

```
inputValid = false;
while (!inputValid) {
    try {
        System.out.print(s:"Jumlah Hari Absen Tanpa Alasan (0-7 hari): ");
        hariAbsen[i] = scanner.nextInt();
        if (hariAbsen[i] < 0 || hariAbsen[i] > 7) {
            System.out.println(x:"Jumlah hari absen tidak valid. Harap masukkan angka antara 0 dan 7.");
        } else {
            inputValid = true;
        }
    } catch (InputMismatchException e) {
        System.out.println(x:"Input tidak valid. Harap masukkan angka bulat untuk hari absen.");
        scanner.next(); // Membersihkan input yang salah
    }
}
scanner.nextLine(); // Membersihkan newline setelah nextInt()
```

Menggunakan try-catch untuk InputMismatchException dan memvalidasi bahwa hari absen antara 0 dan 7.

4.5. Logika Perhitungan Gaji

```
double tarifPerJam = 0;
switch (shiftKaryawan[i].toLowerCase()) {
    case "pagi":
        tarifPerJam = 50000; // Contoh tarif
        break;
    case "siang":
        tarifPerJam = 55000; // Contoh tarif
        break;
    case "malam":
        tarifPerJam = 60000; // Contoh tarif
        break;
}

double gajiPokok = jamKerjaMingguan[i] * tarifPerJam;
double potonganAbsen = hariAbsen[i] * 100000; // Potongan Rp100.000 per hari absen
double gajiLembur = 0;
double potonganKurangKerja = 0;

// Hitung lembur
if (jamKerjaMingguan[i] > 40) {
    int jamLembur = jamKerjaMingguan[i] - 40;
    gajiLembur = jamLembur * (tarifPerJam * 1.5); // Tarif lembur 1.5x dari tarif normal
}

// Hitung potongan jika kurang dari 30 jam
if (jamKerjaMingguan[i] < 30) {
    potonganKurangKerja = gajiPokok * 0.10; // Potongan 10%
}

// Hitung gaji bersih
gajiBersih[i] = gajiPokok + gajiLembur - potonganAbsen - potonganKurangKerja;

// Pastikan gaji tidak negatif
if (gajiBersih[i] < 0) {
    gajiBersih[i] = 0;
```

- (Tarif Per Jam):

  Pernyataan switch menentukan tarif per jam berdasarkan shift karyawan. Shift yang berbeda memiliki contoh tarif yang berbeda (Pagi: Rp 50.000, Siang: Rp 55.000, Malam: Rp 60.000).

- (Gaji Pokok):

  Dihitung dengan mengalikan jamKerjaMingguan (jam kerja mingguan) dengan tarifPerJam.

- (Potongan Absensi):

  Potongan tetap sebesar Rp 100.000 diterapkan untuk setiap hariAbsen (hari absen tanpa alasan).

- (Gaji Lembur):

  Jika jamKerjaMingguan melebihi 40 jam, jam di atas 40 dianggap lembur.

  Gaji lembur dihitung 1,5 kali tarifPerJam.

- (Potongan Kurang Kerja):

  Jika jamKerjaMingguan kurang dari 30 jam, potongan 10% diterapkan pada gajiPokok.

- (Gaji Bersih):

  Dihitung sebagai gajiPokok + gajiLembur - potonganAbsen - potonganKurangKerja.

- Gaji Non-negatif:

  if (gajiBersih[i] < 0) { gajiBersih[i] = 0; } memastikan bahwa gaji bersih akhir yang dihitung tidak kurang dari nol.

V. Kesimpulan

Dari hasil pengembangan dan implementasi program GajiKaryawanPabrik dapat disimpulkan bahwa pemanfaatan bahasa pemrograman Java dalam sistem penggajian karyawan pabrik memberikan solusi yang efektif dan efisien dalam proses perhitungan gaji. Program ini mampu mengakomodasi berbagai komponen penting dalam penggajian, seperti tarif shift kerja, jumlah jam kerja mingguan, potongan akibat absen tanpa keterangan, serta insentif lembur. Validasi input yang dilakukan program juga meningkatkan keandalan sistem dan meminimalkan kesalahan perhitungan.

Secara fungsional, program ini berhasil memberikan gambaran sederhana namun aplikatif mengenai bagaimana sistem penggajian otomatis dapat dibangun menggunakan paradigma pemrograman prosedural. Kelebihan utama dari program ini terletak pada kemampuan adaptasinya terhadap jumlah karyawan yang fleksibel serta keluaran laporan yang informatif dan mudah dibaca.

Namun demikian, program ini masih memiliki keterbatasan, seperti tidak tersedianya fitur penyimpanan data jangka panjang (misalnya ke dalam database atau file eksternal), tidak adanya autentikasi pengguna, dan belum tersedia antarmuka pengguna grafis (GUI). Oleh karena itu, pengembangan lebih lanjut sangat disarankan, misalnya dengan mengintegrasikan sistem ini dengan database MySQL, membuat versi GUI menggunakan JavaFX atau Swing, atau mengonversi program ini ke dalam platform berbasis web atau mobile agar dapat digunakan dalam skala industri yang lebih luas.

VI. Referensi

- Nugroho, A. (2010). *Rekayasa Perangkat Lunak Menggunakan UML dan Java*. Yogyakarta: Andi.
- McConnell, S. (2004). *Code Complete: A Practical Handbook of Software Construction*. Microsoft Press.
- Rizky, F. (2021). *Analisis dan Perancangan Sistem Informasi Penggajian Karyawan Berbasis Desktop*. Jurnal Teknologi Informasi dan Ilmu Komputer, 8(1), 45-52.
- Wilson, G., & Oram, A. (2007). *Beautiful code: Leading programmers explain how theythink*."O'ReillyMedia,Inc."https://books.google.co.id/books?hl=en&lr=&id=gJrmszNHQV4C&oi=fnd&pg=PR7&dq=related:aStDDaR4koJ:scholar.google.com/&ots=rPTWrxP5oe&sig=Cz1kLk9krauzDYnFZEpATOwqveI&redir_esc=y#v=onepage&q&f=false

VII. Daftar Pustaka