NAMA : ANISSYA AULIAN SP

NIM : 1227050023

KELAS : IF (F)

UTS PRATIKUM PBO

INPUT

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        try (Scanner scanner = new Scanner(System.in)) {
            List<Karyawan> daftarKaryawan = new ArrayList<>();

            // Tambahkan beberapa karyawan
            daftarKaryawan.add(new Manager("John", 5000));
            daftarKaryawan.add(new Programmer("Alice", 3000));
            daftarKaryawan.add(new Programmer("Bob", 2500));

            // Tampilkan daftar karyawan
            System.out.println("Daftar Karyawan:");
            for (Karyawan karyawan : daftarKaryawan) {
                System.out.println(karyawan.infoKaryawan());
            }

            // Tanyakan apakah ingin menghitung gaji karyawan
            System.out.print("\nApakah Anda ingin menghitung gaji karyawan?
(ya/tidak): ");
            String jawaban = scanner.nextLine();

            if (jawaban.equalsIgnoreCase("ya")) {
                for (Karyawan karyawan : daftarKaryawan) {
                    karyawan.hitungGaji();
                }
            } else {
                System.out.println("Terima kasih atas kunjungan Anda.");
            }
        }
    }
}

// Interface untuk Karyawan
interface Karyawan {
```

```
    String infoKaryawan();
    void hitungGaji();
}

// Class Manager sebagai subclass dari Karyawan
class Manager implements Karyawan {
    private String nama;
    private int gajiPokok;
    private int tunjangan;

    public Manager(String nama, int gajiPokok) {
        this.nama = nama;
        this.gajiPokok = gajiPokok;
        this.tunjangan = 1000;
    }

    @Override
    public String infoKaryawan() {
        return "Manager: " + nama + " - Gaji Pokok: " + gajiPokok;
    }

    @Override
    public void hitungGaji() {
        int totalGaji = gajiPokok + tunjangan;
        System.out.println("Gaji " + nama + ": " + totalGaji);
    }
}

// Class Programmer sebagai subclass dari Karyawan
class Programmer implements Karyawan {
    private String nama;
    private int gajiPokok;

    public Programmer(String nama, int gajiPokok) {
        this.nama = nama;
        this.gajiPokok = gajiPokok;
    }

    @Override
    public String infoKaryawan() {
        return "Programmer: " + nama + " - Gaji Pokok: " + gajiPokok;
    }

    @Override
    public void hitungGaji() {
        System.out.println("Gaji " + nama + ": " + gajiPokok);
    }
}
```

OUTPUT

```
PS C:\PRATIKUM PBO\UTS\UTS PRAT PBO>  & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C
:\PRATIKUM PBO\UTS\UTS PRAT PBO\bin' 'Main'
Daftar Karyawan:
Manager: John - Gaji Pokok: 5000
Programmer: Alice - Gaji Pokok: 3000
Programmer: Bob - Gaji Pokok: 2500

Apakah Anda ingin menghitung gaji karyawan? (ya/tidak): ya
Gaji John: 6000
Gaji Alice: 3000
Gaji Bob: 2500
PS C:\PRATIKUM PBO\UTS\UTS PRAT PBO>
```

Penjelasan

Program ini merupakan simulasi manajemen karyawan yang memanfaatkan konsep-konsep OOP seperti Encapsulation, Inheritance, Polymorphism, dan Interface.

Program dimulai dengan mendeklarasikan sebuah list daftarKaryawan yang akan menyimpan objek-objek karyawan.

Kemudian, tiga objek karyawan (dua programmer dan satu manajer) ditambahkan ke dalam list tersebut.

Setelah itu, program menampilkan daftar karyawan beserta informasi gaji pokok mereka.

Pengguna kemudian diminta untuk menghitung gaji karyawan atau tidak.

Jika pengguna memilih untuk menghitung gaji, program akan mengiterasi melalui setiap objek karyawan dan memanggil metode hitungGaji() pada masing-masing objek. Metode hitungGaji() akan menghitung dan menampilkan gaji sesuai dengan tipe karyawan (manager atau programmer).

Jika pengguna tidak ingin menghitung gaji, program akan menampilkan pesan terima kasih dan berakhir.

konsep-konsep dalam program yang telah dibuat:

Encapsulation:

Konsep encapsulation terlihat dalam pembuatan atribut-atribut kelas yang dideklarasikan sebagai private, seperti nama dan gajiPokok pada kelas Manager, serta nama dan gajiPokok pada kelas Programmer. Dengan demikian, atribut-atribut tersebut tidak dapat diakses secara langsung dari luar kelas, namun hanya dapat diakses melalui metode-metode publik yang disediakan.

Contoh: Atribut nama dan gajiPokok pada kelas Manager dan Programmer.

Inheritance:

Konsep inheritance terjadi saat class Manager dan Programmer mewarisi sifat dan perilaku dari interface Karyawan. Dengan demikian, kelas Manager dan Programmer memiliki akses ke metode yang didefinisikan dalam interface Karyawan.

Contoh: Manager dan Programmer mengimplementasikan interface Karyawan.

Polymorphism:

Polymorphism terjadi ketika objek kelas Manager dan Programmer dapat diperlakukan secara polimorfik sebagai objek Karyawan. Ini memungkinkan kita untuk menggunakan objek kelas Manager dan Programmer dengan cara yang sama seperti objek Karyawan.

Contoh: Dalam loop for-each di Main, kita memperlakukan setiap objek Karyawan (yang mungkin adalah objek Manager atau Programmer) secara seragam, menggunakan metode infoKaryawan() dan hitungGaji().

Interface:

Interface Karyawan digunakan untuk menetapkan perilaku yang diharapkan dari kelas-kelas yang menerapkannya. Dalam hal ini, setiap kelas yang mengimplementasikan interface Karyawan diharapkan memiliki metode infoKaryawan() dan hitungGaji().

Contoh: Interface Karyawan digunakan sebagai kontrak untuk kelas-kelas Manager dan Programmer, memastikan bahwa keduanya memiliki metode infoKaryawan() dan hitungGaji() yang sesuai.