**LAPORAN**

**WEEK 2 PEMROGRAMAN BERBASIS OBJEK**

Dibuat untuk memenuhi salah satu tugas mata kuliah Pemrograman Berbasis Objek yang diampu oleh Bapak Ardhian Ekawijana

Oleh:

**NAMA : Hafiz Zulhakim**

**NIM : 241511044**

**KELAS : 2B**

**PROGRAM STUDI : D3 TEKNIK INFORMATIKA**

**JURUSAN : TEKNIK KOMPUTER DAN INFORMATIKA**



**POLITEKNIK NEGERI BANDUNG**

**Tahun Ajaran 2025/2026**

**Main.java**

public class Main {

    public static void main(String[] args){

        // set pasien

        Pasien pasien1 = new Pasien("tomi", "Demam");

        // Buat obat

        Obat paracetamol = new Obat("Paracetamol", 3);

        // Buat dokter

        Dokter drBudi = new Dokter("Budi", "Umum");

        // Proses pemeriksaan

        drBudi.periksaPasien(pasien1, paracetamol);

        // Lihat sisa stok

        System.out.println("Sisa stok " + paracetamol.getNama() + ": " + paracetamol.getStock());

    }

}

**Obat.java**

public class Obat{

    private String nama;

    private int stock;

    //Construction

    public Obat(String nama, int stock){

        this.nama = nama;

        this.stock = stock;

    }

    //Getter and Setter

    public String getNama(){

        return nama;

    }

    public void setNama(String nama){

        this.nama =  nama;

    }

    public int getStock(){

        return stock;

    }

    public void setStock(int stock){

        this.stock =  stock;

    }

    //Method mengurangi stock barang

    public void kurangiStock(int jumlah){

        if (stock >= jumlah){

            stock -= jumlah;

        } else{

            System.out.println(nama + " Stock obat sudah habis");

        }

    }

}

**Pasien.java**

public class Pasien{

    private String nama;

    private String penyakit;

    //Constructor

    public Pasien(String nama, String penyakit){

        this.nama = nama;

        this.penyakit = penyakit;

    }

    //Getter and Setter

    public String getNama(){

        return nama;

    }

    public void setNama(String nama){

        this.nama = nama;

    }

    public String getPenyakit(){

        return penyakit;

    }

    public void setPenyakit(String NIM){

        this.penyakit = penyakit;

    }

}

**Dokter.java**

public class Dokter {

    private String nama;

    private String spesialis;

    //Constructor

    public Dokter(String nama, String spesialis){

        this.nama = nama;

        this.spesialis = spesialis;

    }

    //Getter

    public String getNama(){

        return nama;

    }

    public String getSpesialis(){

        return spesialis;

    }

    //Method memeriksa pasien dan memberi obat

    public void periksaPasien(Pasien pasien, Obat obat) {

        System.out.println("Dokter " + nama +" (spesialis "+ spesialis +") memeriksa " +pasien.getNama() + " yang sakit " + pasien.getPenyakit());

        obat.kurangiStock(1);

        System.out.println("Obat " + obat.getNama() + " diberikan kepada " + pasien.getNama());

   }

}

A black background with white text

AI-generated content may be incorrect.

**Lesson Learn**

Dari pengerjaan tugas ini, dapat dipelajari bahwa penggunaan atribut private dan method public (getter/setter) memberikan kendali penuh terhadap bagaimana data dalam objek diakses dan dimodifikasi. Konsep enkapsulasi ini mencegah perubahan data secara sembarangan, misalnya stok barang hanya bisa berkurang melalui method khusus, bukan langsung diubah nilainya. Interaksi antar objek, seperti antara Petugas, Peminjam, dan NamaBarang, membentuk alur kerja yang jelas, di mana setiap class memiliki perannya masing-masing. Constructor memastikan setiap objek diinisialisasi dengan data awal yang valid, sehingga tidak ada objek yang dibuat dalam keadaan kosong. Pemisahan tanggung jawab antar class juga memberikan manfaat besar, karena Peminjam hanya fokus pada data mahasiswa yang meminjam, NamaBarang fokus pada detail barang yang bisa dipinjam, dan Petugas bertugas melayani transaksi peminjaman. Dengan struktur ini, kode menjadi lebih terorganisir, mudah dipahami, serta lebih mudah untuk dikembangkan di masa depan, sesuai dengan praktik pemrograman berorientasi objek yang baik.