JobSheet - Week 2 - Object, Class & Encapsulation

Nama: <Tuliskan Nama Lengkap di sini>

NIM: <Tuliskan NIM>

Kelas: 1A/1B/1C/1D

Repo GitHub: <Tuliskan Tautan Repo GitHub>

Instruksi Pengerjaan:

1. Kerjakan 2 soal di bawah ini dengan melengkapi setiap kolom jawaban yang disediakan pada jobsheet ini.
2. Jawaban setiap soal mencakup source code, screenshot hasil dari program yang ditampilkan full screen termasuk taskbar (tambahkan beberapa screenshot jika diperlukan), penjelasan permasalahan dan solusi yang dihadapi, nama teman yang membantu memecahkan masalah (opsional).
3. Dikumpulkan pada Assignment Classroom sesuai dengan deadline yang tertera pada assignment tersebut.
4. Format penamaan file jobsheet: W2_P_<Kelas 1X>_<3 Digit_NIM_Terakhir>.docx/pdf. Contoh: W2_P_1B_001.docx/pdf.
5. Submit semua jawaban dalam bentuk file java pada repository GitHub masing-masing.

No. 1 Evaluasi & Refactoring Class Restaurant

Soal Praktikum

Diketahui terdapat dua file program yang digunakan untuk menyimpan daftar menu makanan dan stoknya, yaitu Restaurant.java & RestaurantMain.java.

1. Restaurant.java

```
public class Restaurant {
    public String[] nama_makanan;
    public double[] harga_makanan;
    public int[] stok;
    public static byte id = 0;

    public Restaurant() {
        nama_makanan = new String[10];
        harga_makanan = new double[10];
        stok = new int[10];
    }
}

public void tambahMenuMakanan(String nama, double harga, int stok) {
```

```
    this.nama_makanan[id] = nama;
    this.harga_makanan[id] = harga;
    this.stok[id] = stok;
}

public void tampilMenuMakanan() {
    for (int i = 0; i <= id; i++) {
        if (!isOutOfStock(i)) {
            System.out.println(
                nama_makanan[i] + "[" + stok[i] + "]" + "\tRp. " + harga_makanan[i]
            );
        }
    }
}

public boolean isOutOfStock(int id) {
    if (stok[id] == 0) {
        return true;
    } else {
        return false;
    }
}

public static void nextId() {
    id++;
}
```

2. RestaurantMain.java

```
public class RestaurantMain {
    public static void main(String[] args) {
        Restaurant menu = new Restaurant();

        menu.tambahMenuMakanan("Pizza", 250000, 20);
        Restaurant.nextId();

        menu.tambahMenuMakanan("Spaghetti", 80000, 20);
        Restaurant.nextId();

        menu.tambahMenuMakanan("Tenderloin Steak", 60000, 30);
        Restaurant.nextId();

        menu.tambahMenuMakanan("Chicken Steak", 45000, 30);

        menu.tampilMenuMakanan();
    }
}
```

A. Instruksi - Analisis Desain Class:

Amati kode program yang diberikan, kemudian jawab pertanyaan berikut:

1. Apakah class Restaurant sudah menerapkan Encapsulation dengan benar? Anda dapat mengacu pada Buku Ajar Tekpro (Teori) - Chapter 2.4 Enkapsulasi (Lihat di Buku Acuan pada Google Classroom) atau Buku Java Core Design Hint Chapter 4.10.
2. Apakah attribute yang diterapkan saat ini aman dari akses secara

langsung?
3. Apakah terdapat pelanggaran prinsip OOP (misalnya public attribute)?

B. Instruksi - Perbaikan Kode Program:

Lakukan perbaikan desain class dengan ketentuan:
1. Semua attribute harus bersifat private
2. Akses data dilakukan melalui getter dan setter
3. Validasi stok (stok tidak boleh negatif)
4. Pengembangan Fitur (Mini Case) Tambahkan fitur berikut:
   a) Pemesanan menu
   b) Stok otomatis berkurang setelah pemesanan
   c) Pesan ditolak jika stok tidak mencukupi

Source Code
Screenshot Hasil
Penjelasan Permasalahan dan Solusi
Nama Teman Hal yang Dibantu (Opsional)

No. 2 Interaksi Object dan Struktur Class dalam Java

Soal Praktikum
Pada pertemuan sebelumnya, mahasiswa telah mempelajari konsep dasar dalam Pemrograman Berorientasi Objek (Object-Oriented Programming/OOP) yang menjadi fondasi utama dalam pengembangan aplikasi berbasis Java. 1. *Object* Object merupakan representasi nyata dari suatu entitas yang memiliki data (attribute) dan perilaku (method). Object dibuat dari sebuah class dan digunakan untuk menjalankan proses dalam program. 2. *Class* Class adalah cetak biru (blueprint) untuk membuat object. Class mendefinisikan struktur data dan perilaku yang dimiliki oleh object, sehingga memungkinkan pembuatan object dengan karakteristik yang sama. 3. *Message* Message menggambarkan proses komunikasi antar object. Dalam Java, message diwujudkan dalam bentuk pemanggilan method pada suatu object untuk meminta object tersebut melakukan suatu aksi. 4. *Encapsulation* Encapsulation adalah prinsip OOP yang digunakan untuk melindungi data dengan membatasi akses langsung terhadap attribute. Penerapan

encapsulation dilakukan menggunakan access modifier dan accessor method (getter dan setter).

Pada pertemuan kali ini, mahasiswa akan mempelajari praktik beberapa konsep lanjutan dalam Pemrograman Berorientasi Objek menggunakan Java yang berfokus pada interaksi antar object serta struktur dan organisasi program.

1. *Relationship Between Class*
   Membahas hubungan antar class dalam sebuah sistem, seperti bagaimana satu class dapat menggunakan atau memiliki object dari class lain untuk membentuk sistem yang lebih kompleks.
2. *Static Field & Method*
   Mempelajari penggunaan field dan method yang bersifat static, yaitu milik class dan bukan milik object, serta memahami kapan dan mengapa konsep static digunakan.
3. *Method Parameters*
   Membahas cara pengiriman data ke dalam method melalui parameter, termasuk penggunaan object sebagai parameter untuk memungkinkan interaksi antar object.
4. *Object Construction*
   Mempelajari proses pembuatan object melalui constructor, termasuk inisialisasi attribute dan pengaturan kondisi awal object saat pertama kali dibuat.
5. *Packages*
   Membahas pengelompokan class ke dalam package untuk mengatur struktur program, meningkatkan keterbacaan kode, dan menghindari konflik nama class.
6. JAR
   Mempelajari konsep Java Archive (JAR) sebagai media untuk mengemas dan mendistribusikan aplikasi Java agar dapat dijalankan atau dibagikan dengan lebih mudah.

Ketentuan Praktikum:
Pada praktikum ini, mahasiswa diminta untuk memahami contoh program employee dan mengembangkan aplikasi tersebut dengan menerapkan beberapa konsep lanjutan dalam Pemrograman Berorientasi Objek menggunakan Java.

A. Instruksi - Tulis Ulang Kode Program Employee

1. Kelas Department

```
package id.ac.polban.employee.model;

public class Department {
 private String name;

 public Department(String name) {
    this.name = name;
 }

 public String getName() {
    return name;
 }
```

```
 public void setName(String name) {
    this.name = name;
 }
}
```

2. Kelas Employee

```
package id.ac.polban.employee.model;

public class Employee {
    private int id;
    private String name;
    private Department department;
    private EmploymentType type;
    private double salary;

    public Employee(int id, String name, Department department,
EmploymentType type, double salary) {
        this.id = id;
        this.name = name;
        this.department = department;
        this.type = type;
        this.salary = salary;
    }

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Department getDepartment() {
        return department;
    }
    public void setDepartment(Department department) {
        this.department = department;
    }
    public EmploymentType getType() {
        return type;
    }
    public void setType(EmploymentType type) {
        this.type = type;
    }
    public double getSalary() {
        return salary;
    }
    public void setSalary(double salary) {
        this.salary = salary;
    }
}
```

3. Kelas EmployeeType

```
package id.ac.polban.employee.model;

public class EmploymentType {
```

```
    private String type;

    public EmploymentType(String type) {
        this.type = type;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }
}
```

4. Kelas EmployeeService

```
package id.ac.polban.employee.service;

import java.util.HashMap;
import java.util.Map;

import id.ac.polban.employee.model.*;

// mengelola operasi yang berkaitan dengan data dan aturan bisnis
public class EmployeeService {
    private Map<Integer, Employee> employees = new HashMap<>();

    public void addEmployee(Employee emp) {
        employees.put(emp.getId(), emp);
    }

    public Employee getEmployee(int id) {
        return employees.get(id);
    }

    public void raiseSalary(int id, double percent) {
        Employee emp = employees.get(id);
        if (emp != null) {
            emp.setSalary(emp.getSalary() * (1 + percent/100));
        }
    }
}
```

B. Instruksi - Lengkapi Studi Kasus, Rancangan Class Diagram & Penjelasan:

Adapun ketentuan dan tugas praktikum adalah sebagai berikut:

1. Lengkapi studi kasus yang telah dibuat dengan menerapkan penggunaan *static field* dan *static method* secara tepat!
2. Terapkan konsep package dengan membuat minimal dua package, yaitu:
    - `id.ac.polban.employee.model`
    - `id.ac.polban.employee.service`
2. Buat diagram kelas (**class diagram**) yang menggambarkan struktur class, attribute, method, serta hubungan antar class dalam sistem.
3. Implementasikan relasi antar class pada program, yang mencakup:
    - *Dependency*

- *Aggregation*

2. Jelaskan perbedaan dan fungsi masing-masing jenis relasi tersebut berdasarkan kasus yang dibuat!
3. Lakukan proses generate aplikasi ke dalam file JAR!
4. Buat sebuah project Java baru yang hanya memuat file JAR hasil generate, kemudian jalankan aplikasi dari project tersebut untuk memastikan file JAR dapat digunakan dengan benar!
5. Seluruh source code, diagram kelas, serta penjelasan implementasi didokumentasikan pada jobsheet ini! Pastikan source code di submit pada repository GitHub masing-masing!

Source Code
Screenshot Hasil
Penjelasan Permasalahan dan Solusi
Nama Teman dan Hal yang Dibantu (Opsional)