Teori Pertemuan 4

Disusun untuk memenuhi tugas mata kuliah Pemrograman Berorientasi Objek (Teori)

Disusun Oleh:
Radja Restu Arsita (231511061) 2B – D3

Jurusan Teknik Komputer dan Informatika

Politeknik Negeri Bandung

2024

DAFTAR ISI

SOURCE CODE

1.1. Dependency

a. Source code

```
// Layanan Email (Dependency)
class EmailService {
    public void sendEmail(String message, String recipient) {
        System.out.println("Email sent to " + recipient + " with message: "
+ message);
    }
}

// Kelas Notifikasi yang bergantung pada EmailService (Dependency Injection)
class NotificationService {
    private EmailService emailService; // Dependency

    // Dependency Injection melalui konstruktor
    public NotificationService(EmailService emailService) {
        this.emailService = emailService;
    }

    public void sendNotification(String message, String recipient) {
        // Menggunakan EmailService untuk mengirim email
        emailService.sendEmail(message, recipient);
    }
}

// Program Utama
public class Main {
    public static void main(String[] args) {
        // Membuat dependency (EmailService)
        EmailService emailService = new EmailService();

        // Injecting dependency ke NotificationService
        NotificationService notificationService = new
NotificationService(emailService);

        // Mengirim notifikasi
        notificationService.sendNotification("Hello, this is a test
notification!", "user@example.com");
    }
}
```

b. Penjelasan

Jadi pada Dependency itu nanti ada sebuah class yang bergantung pada class lainnya sesuai dengan program diatas yang mana notification service tidak akan berjalan tanpa ada nya email service sehingga objek NotificationService tidak akan berjalan sesuai tanpa adanya objek EmailService

1.2 Aggregation

a. Source code

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

class Department {
    private String name;

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

    // Other department-related methods
}

class University {
    private String name;
    private List<Department> departments; // Aggregation

    public University(String name) {
        this.name = name;
        this.departments = new ArrayList<>();
    }

    public void addDepartment(Department department) {
        departments.add(department);
    }

    // Other university-related methods
}
```

b. Penjelasan

Pada aggregation yang perlu di ingat adalah kata kunci nya yaitu has-a yang mana objek departemen berada di bawah naungan objek university namun objek departemen tidak memiliki ketergantungan terhadap objek university jadi mau ada atau tidak nya university itu tidak masalah namun demi menjaga agar code menjadi clean maka dibuat metode aggregasi agar membuat objek menjadi lebih mudah di kelompokan

1.3 Inheritance

a. Source Code

```
class Animal {

    // field and method of the parent class
    String name;
```

```
    public void eat() {
        System.out.println("I can eat");
    }
}

// inherit from Animal
class Dog extends Animal {

    // new method in subclass
    public void display() {
        System.out.println("My name is " + name);
    }
}

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

        // create an object of the subclass
        Dog labrador = new Dog();

        // access field of superclass
        labrador.name = "Rohu";
        labrador.display();

        // call method of superclass
        // using object of subclass
        labrador.eat();

    }
}
```

b. Penjelasan

Jadi pada inti nya bahwa inheritance itu merupakan penurunan dari class sebelum nya sebagai contoh program diatas yaitu class animal yang inheritance nya itu merupakan dog dan class dog ini karena merupakan turunan dari class animal maka dia bisa menggunakan fungsi pada class animal yang pada contoh diatas adalah eat.

LESSON LEARNED

Jadi apa yang saya pelajari dari kelas teori ini yaitu terdapat tiga relasi dalam class yang mana sebagai berikut :

1. Dependency

   Sebuah class memiliki ketergantungan terhadap class lainnya sehingga ketika class yang digantungi menghilang class yang memiliki ketergantungan tidak dapat bekerja dengan baik

2. Aggregation

   Sebuah class yang dibentuk hanya untuk mencakup class lainnya agar mudah diolah namun tanpa ada nya class tersebut untuk mencakup kelas lainnya class yang di cakup tidak terpengaruh oleh apapun

3. Inheritance

   Penurunan class dari class ibu maksud dari class ibu disini adalah class yang merupakan asal muasal dari class turunan nya yang mana class turunan memiliki akses untuk menggunakan function yang berada di dalam class ibu seperti contoh yang sudah diberikan yaitu animal (class ibu) dan dog (class turunan)

REFERENCE

- Salvi Priya, "Java Aggregation and Composition Explained with Examples", *Medium*, diakses pada 17 September 2024, https://medium.com/@salvipriya97/java-aggregation-and-composition-explained-with-examples-66cbffd21b9c.

- "Java Inheritance", *Programiz*, diakses pada 17 September 2024, https://www.programiz.com/java-programming/inheritance.