TUGAS 3
CLASS&OBJECT PART 2

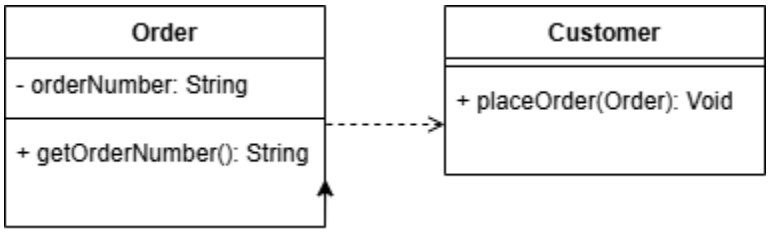
Diajukan untuk memenuhi salat satu tugas Mata kuliah Pemrograman Berorientasi Objek

Disusun Oleh:
Daiva Raditya Pradipa (231511039)

Jurusan Teknik Komputer dan Informatika
Program Studi D-3 Teknik Informatika
Politeknik Negeri Bandung
2024

Class Diagram

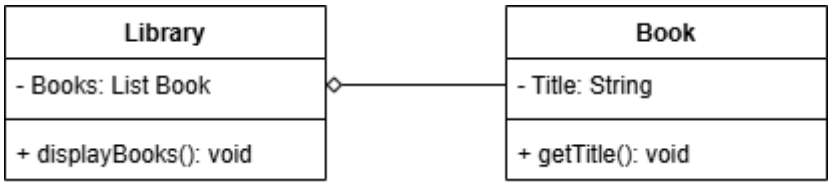
1. Depedency



Dalam OOP (Object-Oriented Programming), dependency terjadi ketika satu class bergantung pada class lain untuk berfungsi. Contohnya dalam kasus ini, terdapat class Customer yang bergantung pada class Order karena untuk menggunakan method placeOrder(), class Customer memerlukan objek bertipe Order sebagai parameter.

Class Customer ini berisi method placeOrder(), yang memerlukan objek bertipe Order untuk berfungsi. Hal ini menunjukkan bahwa Customer memiliki ketergantungan pada class Order. Class Order menyimpan data yaitu orderNumber yang diperlukan oleh method placeOrder() untuk dapat berjalan. Maka dari itu dalam penggunaanya, objek dari class Order harus dibuat terlebih dahulu agar method placeOrder() di class Customer dapat bekerja. Sehingga dalam program dengan source code dibawah, ketika fungsi placeOrder() dipanggil, dia bisa menampilkan orderNumber yang tersimpan dalam class order yang telah dibuat objectnya.

2. Aggregation



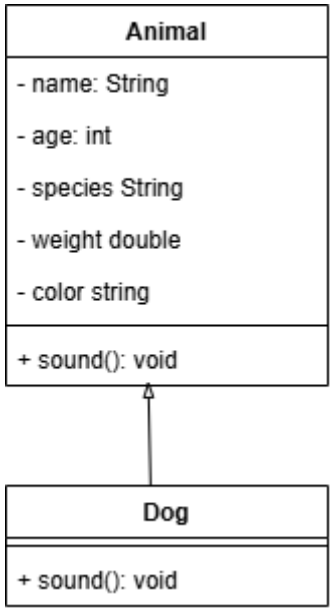
Dalam Object-Oriented Programming (OOP), agregasi adalah jenis hubungan "has-a" (memiliki), yang menunjukkan bahwa satu class dapat mereferensikan class lain. Secara sederhana, class yang mereferensikan class lain dapat menggunakan data dari class yang direferensikan. Biasanya, hubungan ini menggambarkan bahwa class yang mereferensikan dapat memiliki banyak data dari class yang direferensikan. Namun, class yang direferensikan tetap dapat berdiri sendiri dan tidak bergantung pada class yang mereferensikannya.

Sebagai contoh, terdapat class Library yang mereferensikan class Book untuk menyimpan banyak objek Book, tanpa membuat class Book bergantung pada Library.

- Class Library memiliki atribut books yang bertipe List<Book> (di mana Book adalah objek dari class Book).
- Class Library juga memiliki method displayBooks(), yang akan memanggil fungsi getTitle() pada class Book untuk menampilkan judul buku yang ada di dalam koleksi.

Dalam penggunaannya, objek Book harus dibuat terlebih dahulu dalam bentuk List sebelum class Library dapat bekerja dengan objek-objek tersebut. Dengan cara ini, class Library dapat mengakses dan menampilkan informasi dari objek-objek Book yang telah dibuat.

3. Inheritance



Dalam Object-Oriented Programming (OOP), inheritance adalah jenis hubungan "is-a" (adalah), yang menunjukkan bahwa satu class (disebut super class) dapat memiliki subclass yang mewarisi atribut dan method dari superclass-nya. Sebagai contoh, terdapat class Animal dan subclass Dog, yang menunjukkan hubungan bahwa class Dog adalah himpunan bagian dari class Animal.

Pada kasus ini, class Animal dapat mewariskan atribut dan method-nya ke subclass Dog. Hal ini memungkinkan class Dog untuk memiliki atribut dan method yang sama seperti class Animal. Selain itu, class Dog juga bisa menambahkan atribut dan method spesifiknya sendiri. Method yang diwarisi dari class Animal juga bisa diubah isinya di dalam class Dog (proses ini disebut method overriding), sehingga menghasilkan output yang berbeda.

Source Code

Main.java

```
import java.util.Arrays;
import java.util.List;
```

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

        Order order = new Order("12345");
        Customer customer = new Customer();
        customer.placeOrder(order);

        Book book1 = new Book("Java Programming");
        Book book2 = new Book("Design Patterns");
        List<Book> books = Arrays.asList(book1, book2);

        Library library = new Library(books); // Aggregation: Library has books
        library.displayBooks();

        Animal myDog = new Dog(); // Inheritance: Dog inherits from Animal
        myDog.sound();

    }

}
```

Order.Java

```
class Order {
    private String orderNumber;

    public Order(String orderNumber) {
        this.orderNumber = orderNumber;
    }

    public String getOrderNumber() {
        return orderNumber;
    }
}


class Customer {
    public void placeOrder(Order order) {
        System.out.println("Order placed: " + order.getOrderNumber());
    }
}
```

Customer.Java

```
class Customer {
    public void placeOrder(Order order) {
```

```
        System.out.println("Order placed: " + order.getOrderNumber());
    }
}
```

Animal.java

```
public class Animal {
    private String name;
    private int age;
    private String species;
    private double weight;
    private String color;

    // Constructor
    public Animal(String name, int age, String species, double weight, String color) {
        this.name = name;
        this.age = age;
        this.species = species;
        this.weight = weight;
        this.color = color;
    }

    // Getter and Setter methods
    public String getName() {
        return name;
    }

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

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getSpecies() {
        return species;
    }

    public void setSpecies(String species) {
        this.species = species;
    }
```

```
    public double getWeight() {
        return weight;
    }

    public void setWeight(double weight) {
        this.weight = weight;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }

    // Method sound
    public void sound() {
        System.out.println("Animal makes a sound");
    }
}
```

Dog.java

```
class Dog extends Animal {
    @Override
    public void sound() {
        System.out.println("Dog barks");
    }
}
```

Library.java

```
import java.util.List;

class Library {
    private List<Book> books;

    public Library(List<Book> books) {
        this.books = books;
    }

    public void displayBooks() {
        for (Book book : books) {
            System.out.println("Book: " + book.getTitle());
        }
```

```
    }
}
```

Book.java

```
class Book {
    private String title;

    public Book(String title) {
        this.title = title;
    }

    public String getTitle() {
        return title;
    }
}
```