Pencarian Dependency Aggregation Inheritance
Pertemuan 4

Diajukan untuk memenuhi salat satu tugas praktikum

Mata kuliah PBO

Disusun Oleh:
Alqan Nazra (231511068)

Jurusan Teknik Komputer dan Informatika

Program Studi D-3 Teknik Informatika
Politeknik Negeri Bandung
2024

Judul :

Hari, Tanggal :

Dependency

```
public
class
Application
{
                  public static void main(String[] args) {
                    var container = new ServiceContainer();
                    container.initialize();
                    container.start();
                  }
                }

                class ServiceContainer {
                  UserRepository userRepository;
                  AuthService authService;
                  OrderService orderService;

                  public void initialize() {
                    userRepository = new UserRepository();
                    authService = new AuthService(userRepository);
                    orderService = new OrderService(authService);
                  }

                  public void start() {
                    orderService.acceptRequests();
                  }
                }

                class OrderService {
                  private AuthService authService;

                  public OrderService(AuthService authService) {
                    this.authService = authService;
                  }

                  // Methods...
                }
```

```
class AuthService {
  private UserRepository userRepository;

  public AuthService(UserRepository userRepository) {
    this.userRepository = userRepository;
  }

  // Methods
}

class UserRepository {}
```

Penjelasan

Konteks Dependency dalam dependency injection adalah ketergantungan antara satu instantsi class dengan yang lain.Dependency dapat juga berarti sesuatu library yang diugnakan dalam suatu aplikasi dapat juga berarti ketergantungan antara satu aplikasi dengan aplikasi lain. Dependendency dapat terjadi karena kertergantungan yang kuat antara satu class dengan class hal ini yang harus dihindari sebuah class harus bisa mandiri agar tidak terlalu merusak sistem inti,Dpendency injection merupakan Solusi dari permsalahan ini dimana setiap class harus menginstansi masing-masing dependecnt sendiri sendiri,dependency injenction artinya membuat satu class baru yang bertanggung jawab untuk mengnstansi class-class lain yang membutuhkan dalam waktu runtime.

Dari contoh diatas tedapat claa yang dimana servisecontaner sebagai orang ketiga sebagai penghubung antara kelas lain,sehingga container ini sebagai penyambung antara kelas lain sehingga tidak perlu untuk melakukan Inisialisasi langsung pada setiap kelas bila ingin menggunakan behaivour kelas itu contoh

Pada orderservice kita ingin menggunakan kelas UserResository tidak perlu menggunakan UserResository user = new UserResository () namun hanya menggunakan container. UserResository untuk dapat memanggilnya hal ini dapat terhadi karena kelas servisecontainer sudah disiapkan Inisialisasi sehingga memudahkan dalam pemanggilan dan membuat kelas tidak saling ketergantungan.

Sumber : https://bilbatez.medium.com/mengenal-konsep-dependency-dependency-inversion-dan-dependency-injection-956d87090224

Aggregation

```
// Class Mesin yang bisa eksis terpisah dari Mobil
class Mesin {
    private String tipeMesin;

    public Mesin(String tipeMesin) {
        this.tipeMesin = tipeMesin;
    }

    public String getTipeMesin() {
        return tipeMesin;
    }
}

// Class Mobil yang memiliki Mesin (Agregasi)
class Mobil {
    private String merek;
    private Mesin mesin; // Agregasi: Mobil memiliki Mesin, tapi Mesin bisa eksis terpisah

    public Mobil(String merek, Mesin mesin) {
        this.merek = merek;
        this.mesin = mesin;
    }

    public void tampilkanInfo() {
        System.out.println("Mobil: " + merek + ", Mesin: " + mesin.getTipeMesin());
    }
}

public class Main {
    public static void main(String[] args) {
        // Membuat objek Mesin
        Mesin mesin1 = new Mesin("V8 Turbo");

        // Mobil menggunakan Mesin (Agregasi)
        Mobil mobil1 = new Mobil("Toyota", mesin1);

        mobil1.tampilkanInfo();

        // Mesin tetap ada walaupun mobilnya dihapus
        mobil1 = null; // Menghapus mobil1, tapi mesin1 masih ada

        System.out.println("Mesin masih ada: " + mesin1.getTipeMesin());
    }
}
```

Penjelasan

Agregasi berarti menempatkan objek bersama-sama untuk menjadikannya objek yang lebih besar. Barang-barang atau benda-benda manufaktur biasanya merupakan contoh dari bentuk agregasi. Secara lebih jelas konsep menggambarkan hubungan "has-a" atau "part-whole" antara dua objek. Ini berarti bahwa suatu objek terdiri dari objek lain, tetapi objek tersebut dapat hidup secara independen dari objek yang memilikinya.

Code ini memiliki kenyataan bahwa

1. Mesin dan Mobil adalah class yang terpisah
2. Mobil memiliki Mesin, tapi mesin bisa tetap ada meskipun Mobil dihapus
3. Agregasi menggambarkan bahwa **Mobil** *memiliki* **Mesin**, tapi tidak bertanggung jawab penuh atas keberadaan **Mesin**. Jika mobil dihapus, **Mesin** masih bisa digunakan oleh objek lain.

Pada program ini menyatakan bahwa setiap kelas saling berhubungan namun hubungan ini bersifat kepemilikan bukan kebutuhan dimana sifat kepemilikan pada dasarnya kepemilikan bersifat tidak pasti dimana bisa hilang dan tidak namun hilang tidaknya barang tidak mempergaruhi secara krusial pada kita, dimana pada PBO ini kelas dapat bergantung pada kelas lain namun ketika kelas ini dihapus atau tidak dipakai maka, kelas yang memiliki sebelumnya tidak terpengaruh secara parah dan masih bisa berjalan.

Sumber : https://medium.com/skyshidigital/hubungan-antar-kelas-pada-java-bbe2131e592b

Sumber : https://www.codepolitan.com/blog/aggregation-dan-association-pada-java-58a1991f9b52c/

Inheritance

inheritance/Pegawai.java

```
package pewarisan_sifat;
public class pegawai{
String nama ;
int id_pegawai;
String gaji;
public void menampilkan(){
System.out.println(“Menampilkan nama, id pegawai, gaji dan tugas.”);
System.out.println("-------------------------------------------------------");
}
}
```

inheritance/manager.java

```
        package pewarisan_sifat;
        public class manager extends pegawai{
        //tambahkan @Override diatas fungsi void menampilkan().
        //override sendiri berfungsi sebagai pembuatan ulang method dari superclass untuk subclass.
        @Override
        public void menampilkan()
        {
        //untuk nilai dari void menampilkan bisa berbeda dari nilai yang ada pada superclass.
        System.out.println(“Nama : ”+nama);
        System.out.println(“Id Pegawai : ”+id_pegawai);
        System.out.println(“Gaji : ”+gaji);
        }
        //untuk fungsi dari void tugas() merupakan fungsi spesifik yang hanya dimiliki oleh class
//manager.
        public void tugas(){
        System.out.println(“Tugas : Melakukan manajemen untuk franchise”);
        System.out.println(“-----------------------------------------------”);
        }
        }
```

inheritance/Kasir.java

```
package pewarisan_sifat;
public class kasir extends pegawai{
@Override
public void menampilkan()
{
System.out.println(“Nama : ”+nama);
System.out.println(“Id Pegawai : ”+id_pegawai);
System.out.println(“Gaji : ”+gaji);
}
public void tugas(){
System.out.println(“Tugas : Melakukan transaksi dengan pembeli”);
System.out.println(“-----------------------------------------------”);
}
}
```

inheritance/Koki.java

```
package pewarisan_sifat;
public class kasir extends pegawai{
@Override
public void menampilkan()
{
System.out.println(“Nama : ”+nama);
System.out.println(“Id Pegawai : ”+id_pegawai);
System.out.println(“Gaji : ”+gaji);
}
public void tugas(){
System.out.println(“Tugas : Memasak makanan dan Membuat minuman”);
System.out.println(“-----------------------------------------------”);
}
}
```

inheritance/pelayan.java

```
package pewarisan_sifat;
public class pelayan extends pegawai{
@Override
public void menampilkan()
{
System.out.println(“Nama : ”+nama);
System.out.println(“Id Pegawai : ”+id_pegawai);
System.out.println(“Gaji : ”+gaji);
}
public void tugas(){
System.out.println(“Tugas : Melayani dan Menyajikan pesanan pembeli”);
System.out.println(“-----------------------------------------------”);
}
}
```

inheritance/satpam.java

```
package pewarisan_sifat;
public class satpam extends pegawai{
public void menampilkan()
{
System.out.println(“Nama : ”+nama);
System.out.println(“Id Pegawai : ”+id_pegawai);
System.out.println(“Gaji : ”+gaji);
}
public void tugas(){
System.out.println(“Tugas : Menjaga keamanan didalam dan diluar frenchise”);
System.out.println(“-----------------------------------------------”);
}
}
```

```
        bangunDatar.luas();
        bangunDatar.keliling();

        persegi.luas();
        persegi.keliling();

        lingkaran.luas();
        lingkaran.keliling();

        persegiPanjang.luas();
        persegiPanjang.keliling();

        mSegitiga.luas();
        mSegitiga.keliling();
    }
}
```

inheritance/Main_project.java

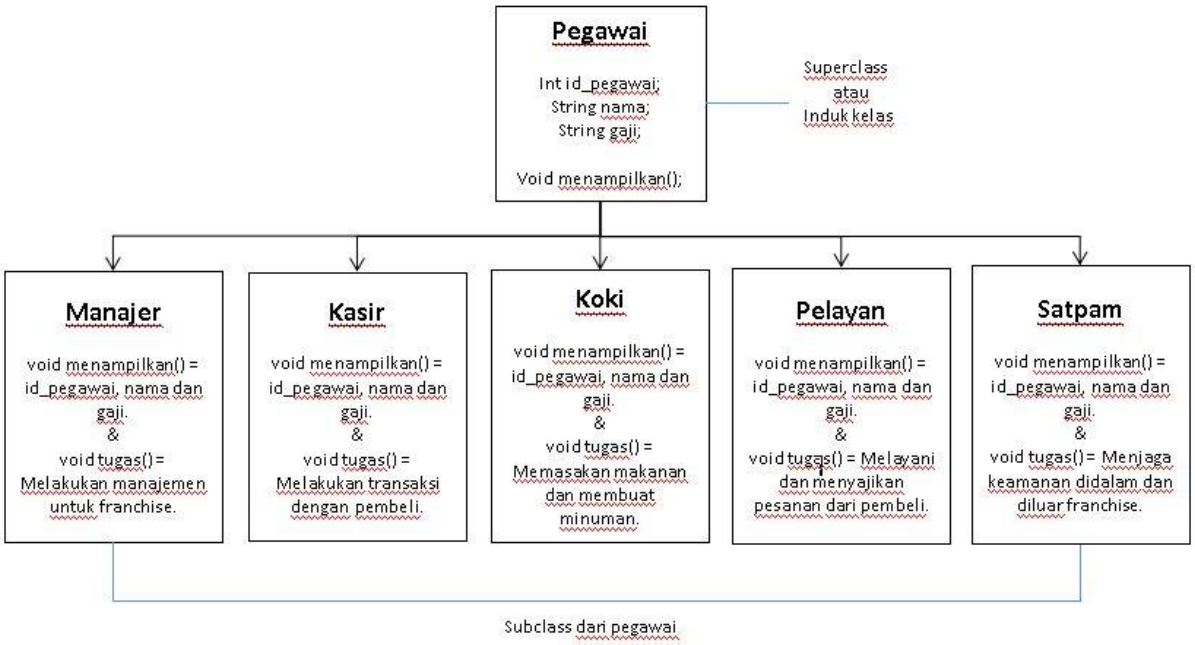
```
package  pewarisan_sifat;
public class Main_project{
public static void main(String[] args){
//membuat sebuah object
// "pegawai" merupakan class, lalu "Pegawai" merupakan objek yang akan dibuat, lalu "new pegawai();" merupakan constructor.
pegawai Pegawai = new pegawai();
manager Manager = new manager();
kasir Kasir = new kasir();
koki Koki = new koki();
pelayan Pelayan = new pelayan();
satpam Satpam = new satpam();

//memasukkan nilai variabel menggunakan objek.
Manager.nama = "sifa";
Manager.id_pegawai = 1;
Manager.gaji = "7 Juta";
Kasir.nama = "Aldi";
Kasir.id_pegawai = 2;
Kasir.gaji = "1,2 Juta";
Koki.nama = "Reza";
Koki.id_pegawai = 3;
Koki.gaji = "2 Juta";
Pelayan.nama = "Dean";
Pelayan.id_pegawai = 4;
Pelayan.gaji = "1,2 Juta";
Satpam.nama = "Aldi";
Satpam.id_pegawai = 2;
Satpam.gaji = "1 Juta";
//nilai tersebut akan dimasukkan kedalam variabel yang ada pada superclass.

//memanggil fungsi pada superclass
Pegawai.menampilkan();

//memanggil nilai variabel pada superclass dan memasukkannya kedalam fungsi yang ada pada class;
Manager.menampilkan();
Manager.tugas();
Kasir.menampilkan();
Kasir.tugas();
Koki.menampilkan();
Koki.tugas();
Pelayan.menampilkan();
Pelayan.tugas();
Satpam.menampilkan();
Satpam.tugas();
}
}
```

Penjelasan

Inheritance (pewarisan) adalah sebuah konsep pewarisan sifat berupa variabel dan fungsi yang dimiliki oleh class untuk diwariskan kepada kelas-kelas yang lain.Konsep dari inheritance sendiri yaitu untuk membuat struktur class pada pemrograman yang dimana struktur tersebut terdapat sebuah *Parentclass* atau *Superclass* sebagai induk kelas dan *Subclass* sebagai anak kelas. Konsep tersebut merupakan sebuah percabangan dari sebuah class (*Superclass*) yang memiliki sifat umum menjadi sebuah class(*Subclass*) yang memiliki sifat lebih spesifik.

Pada program ini menjelaskan bahwa sebuah restaurant makanan terdiri dari beberapa pegawai seperti manajer,pelayan,koki,kasir dan satpam,Setiap pegawai memiliki sebuah nama,ID pegawai dan gaji.Untuk pegawai dengan id pegawai = 01 memiliki tanggung jawab sebagai manager dengan gaji 7 juta. Pegawai dengan id pegawai 02 memiliki tugas untuk melakukan transaksi dengan pembeli dan gaji sebesar 1 juta ,Pegawai dengan ID 03 memiliki tugas untuk memasak makanan dengan gaji sebesar 2 juta. Pegawai dengan id pegawai 4 memiliki tugas dan menyajikan setiap makanan yang dipesan oleh pembeli dengan gaji 1 juta. Pegawai dengan id pegawai 05 memiliki tugas untuk menjaga keamanan didalam atau diluar frenchise dengan gaji 1 juta.

Lewat bisnis rule diatas maka sangat buruk ketika semuanya disimpan dalam satu class ini dikarenakan akan memiliki banyak objeck pada satu kelas yang membuatnya sangat tergantungan satu sama lain dan bila dibuat dalam kelas berbeda dengan file yang sama maka program akan sulit untuk dibaca karena dalam satu file memiliki kelas yang berbeda-beda dengan behavoir yang berbeda sehingga membuatnya menjadi kurang efektif. Sehingga cara terbaik untuk bisnis rule dengan banyak case ini dengan memisahkan setiap class pada file yang berbeda namun file-file ini disatukan dengan file lagi bila dianalogika ini seperti struktur data tree dimana Parent memiliki anak-anak,parent bisa memiliki banyak namun anak tidak bisa memiliki banyak parent, hal ini bertujua untuk sentralisasi pada program,manfaat dari metode ini setiap kelas bersifat sangat independent tapi tetap saling berhubungan dengan gambar grafik seperti gambar dibawah:



Sumber : https://www.petanikode.com/java-oop-inheritance/

Sumber : https://www.gamelab.id/news/1171-java--menggunakan-konsep-inheritance-agar-kode-pemrograman-menjadi-lebih-terstruktur