LAPORAN RESMI

Praktikum 10 Inheritance 2

Mata Kuliah: Praktek Pemrograman Berbasis Objek

Disusun oleh:

M. Ainur Ramadhan (3122500047)

2 D3 Teknik Informatika B

Dosen Pengampu: Yanuar Risah Prayogi S.Kom., M.Kom.

PROGRAM STUDI D3 TEKNIK INFORMATIKA

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

2023/2024

A. TUGAS PENDAHULUAN

1. Ada berapa modifier untuk pengontrolan akses? Jelaskan masing-masing!
   Jawab:
   Di Java, terdapat 4 modifier yang digunakan untuk mengontrol akses terhadap variabel, metode, atau kelas, diantaranya:
   1. Public: Modifier public digunakan untuk memberikan akses penuh ke variabel, metode, atau kelas dari mana saja dalam program Java. Variabel atau metode yang diberi modifier public dapat diakses secara langsung dari mana saja, termasuk dari kelas lain yang berada di paket yang berbeda.
   2. Private: Modifier private digunakan untuk membatasi akses ke variabel atau metode hanya dari dalam kelas itu sendiri. Variabel atau metode dengan modifier private tidak dapat diakses dari luar kelas tersebut.
   3. Protected: Modifier protected memungkinkan akses ke variabel atau metode dari dalam kelas itu sendiri dan juga dari kelas turunan (subkelas). Variabel atau metode yang diberi modifier protected dapat diakses dari kelas yang mewarisi (inherit) kelas tersebut.
   4. Default (Tidak ada modifier): Jika tidak ada modifier yang ditentukan (misalnya, tidak menggunakan public, private, atau protected), maka akses akan mengikuti aturan default dalam Java. Aturan default dalam Java memungkinkan akses dari kelas dalam paket yang sama, tetapi tidak dari paket yang berbeda.

2. Apakah kegunaan kata kunci super? Jelaskan !
   Jawab:
   Kata kunci "super" digunakan untuk merujuk ke anggota (variabel atau metode) dari kelas induk (superclass) dari suatu kelas turunan (subclass). Berikut beberapa kegunaan utama dari kata kunci "super":
   1. Mengakses Anggota Kelas Induk: Kita dapat menggunakan "super" untuk mengakses variabel anggota atau metode dari kelas induk, terutama ketika ada nama yang sama antara anggota di kelas anak dan kelas induk. Ini membantu dalam menghindari ambigu atau konflik nama antara anggota.
   2. Memanggil Konstruktor Kelas Induk: Dalam konstruktor kelas anak, "super" dapat digunakan untuk memanggil konstruktor kelas induk. Hal ini berguna ketika kelas anak memerlukan inisialisasi yang diperlukan oleh kelas induk sebelum melakukan inisialisasi sendiri.
   3. Menghindari Keterbatasan Nama: Ketika terdapat nama yang sama antara variabel atau metode di kelas induk dan kelas anak, "super" digunakan untuk membedakan antara keduanya. Ini memungkinkan kita untuk merujuk ke anggota kelas induk secara eksplisit.

3. Apakah yang dimaksud dengan konstruktor tidak diwariskan?
   Jawab:
   Konstruktor tidak diwariskan berarti bahwa kelas anak (subclass) harus mendefinisikan konstruktor-konstruktornya sendiri, dan konstruktor dari kelas induk (superclass) tidak secara otomatis diwariskan oleh kelas anak. Namun, kelas anak dapat memanggil konstruktor kelas induk menggunakan "super" untuk melakukan inisialisasi kelas induk sebelum inisialisasi kelas anak.

B. LATIHAN

Latihan1 ➔ Konstruktor tidak diwariskan

Buatlah class berikut ini

```
class Base{
    Base(int i){
      System.out.println("base constructor");
    }
    Base(){
    }
}
```

```
public class Sup extends Base{
    public static void main(String argv[]){
      Sup s= new Sup();
      //baris 1
    }

    Sup(){
      // baris 2
    }

    public void derived(){
      // baris 3
    }
}
```

Modifikasilah class Sup (di bagian **//baris 1, //baris 2** *atau* **//baris 3)** sedemikian hingga konstruktor kelas **Base** (konstruktor **Base(int i)**) dipanggil dan menampilkan string "base constructor" ke layar!

Base.java

```
public class Base {
    Base(int i) {
        System.out.println("Base constructor");
    }

    Base() {
    }
}
```

Sup.java

```
public class Sup extends Base {
    public static void main(String argv[]) {
```

```
        Sup s = new Sup(10);
    }

    Sup(int i) {
        super(i);
    }

    public void derived() {
    }
}
```

Output

```
PS C:\Users\USER\OOP\Praktikum\Praktikum10\Latihan\Latihan1> Javac *.java
PS C:\Users\USER\OOP\Praktikum\Praktikum10\Latihan\Latihan1> java Sup
Base constructor
PS C:\Users\USER\OOP\Praktikum\Praktikum10\Latihan\Latihan1>
```

Penjelasan

Dengan perubahan kode, saya mengintegrasikan konsep pewarisan (inheritance) dalam hierarki kelas. Sebelum perubahan, kelas Sup hanya memiliki konstruktor tanpa parameter dan tidak dapat menerima nilai i saat objeknya dibuat. Namun, dengan menambahkan konstruktor baru Sup(int i) yang menerima parameter, hal ini dapat memungkinkan objek Sup untuk diinisialisasi dengan nilai i, yang sekarang diturunkan dari kelas Base. Sehingga memungkinkan kelas turunan Sup untuk mewarisi konstruktor dari kelas Base dan menambahkan fungsionalitas tambahan atau mengubah perilaku dengan cara yang sesuai dengan kebutuhan kelas turunan.

Latihan2 ➔ Konstruktor tidak diwariskan

```
private class Base{
    Base(){
      int i = 100;
      System.out.println(i);
    }
}
```

```
public class Pri extends Base{
    static int i = 200;
    public static void main(String argv[]){
        Pri p = new Pri();
        System.out.println(i);
    }
}
```

Kompile dan jalankan program diatas! Apa yang terjadi? Jelaskan !

Jawab:

Apabila kode tersebut di compile terdapat error yang muncul

```
PS C:\Users\USER\OOP\Praktikum\Praktikum10\Latihan\latihan2> javac *.java
Base.java:1: error: modifier private not allowed here
private class Base {
        ^
1 error
```

Agar kode tersebut dapat berjalan dengan baik kita harus mengganti acces modifier class Base menjadi public bukan private, karena jika dideklarasikan dengan private akan hanya bisa diakses dalam lingkup yang sama. Berikut kode yang telah saya perbaiki.

Base.java

```java
public class Base {
    Base() {
        int i = 100;
        System.out.println(i);
    }
}
```

Pri.java

```java
public class Pri extends Base {
    static int i = 200;

    public static void main(String argv[]) {
        Pri p = new Pri();
        System.out.println(i);
    }
}
```

Output

```
PS C:\Users\USER\OOP\Praktikum\Praktikum10\Latihan\Latihan2> javac *.java
PS C:\Users\USER\OOP\Praktikum\Praktikum10\Latihan\Latihan2> java Pri
100
200
```

Penjelasan

Dalam kode di atas, terdapat dua kelas: Base dan Pri. Pri adalah subclass dari Base. Kemudian objek Pri dibuat dan mencetak nilai dari variabel i.

Akan tetapi, ada beberapa hal yang perlu diperhatikan:

1. Variabel i pada kelas Base dan variabel i pada kelas Pri adalah dua variabel yang berbeda. Mereka terletak pada cakupan yang berbeda. Variabel i dalam konstruktor kelas Base hanya memiliki cakupan (scope) lokal di dalam konstruktor itu sendiri, dan variabel i pada kelas Pri adalah variabel statis milik kelas Pri.
2. Saat objek Pri dibuat dalam metode main, konstruktor kelas Base akan dipanggil karena Pri adalah subclass dari Base. Di dalam konstruktor Base, variabel i dengan nilai 100 dicetak. Namun, ini tidak berdampak pada variabel i dalam kelas Pri.

Jadi, program akan mencetak 100 dari konstruktor kelas Base, dan kemudian mencetak 200 dari variabel i dalam kelas Pri. Variabel i dalam konstruktor Base dan variabel i dalam kelas Pri adalah dua variabel yang berbeda dan berada dalam cakupan yang berbeda.

Latihan3 ➔ Apa yang tampil di layar bila kode dibawah ini dijalankan?

```
class X{
            Y b = new Y();
            X(){
                System.out.print("X");
            }
}
```

```
class Y{
            Y(){
                System.out.print("Y");
            }
}
```

```
public class Z extends X{
            Y y = new Y();
            Z(){
                System.out.print("Z");
            }
            public static void main(String[] args){
                new Z();
            }
}
```

Jawab:

X.java

```
public class X {
    Y b = new Y();

    X() {
        System.out.println("X");
    }
}
```

Y.java

```
public class Y {
    Y() {
        System.out.println("Y");
    }
}
```

Z.java

```
public class Z extends X {
    Y y = new Y();

    Z() {
        System.out.println("Z");
    }

    public static void main(String[] args) {
        new Z();
```

```
    }
}
```

Output

```
PS C:\Users\USER\OOP\Praktikum\Praktikum10\Latihan\Latihan3> javac *.java
PS C:\Users\USER\OOP\Praktikum\Praktikum10\Latihan\Latihan3> java Z
Y
X
Y
Z
PS C:\Users\USER\OOP\Praktikum\Praktikum10\Latihan\Latihan3>
```

Penjelasan

Kode tersebut merupakan contoh dari konsep pewarisan (inheritance) dalam pemrograman Java. Kode tersebut terdiri dari tiga kelas: `X`, `Y`, dan `Z`.

- Kelas `Y` memiliki konstruktor yang mencetak "Y" ketika objeknya dibuat.
- Kelas `X` memiliki konstruktor yang mencetak "X" dan menginisialisasi objek dari kelas `Y` (objek `b`).
- Kelas `Z` adalah subkelas dari `X` dan memiliki konstruktor sendiri yang mencetak "Z" dan menginisialisasi objek dari kelas `Y` (objek `y`).

Berikut penjelasan jika kode tersebut dijalankan

1. Konstruktor kelas `Z` (`Z()`) dipanggil terlebih dahulu. Ini mencetak "Z" dan menginisialisasi objek `y` dari kelas `Y`.
2. Selanjutnya, konstruktor kelas `X` (`X()`) dipanggil karena `Z` adalah subkelas dari `X`. Ini mencetak "X" dan menginisialisasi objek `b` dari kelas `Y`.
3. Pada tahap inisialisasi objek `Y` (`b` dan `y`), konstruktor kelas `Y` (`Y()`) dipanggil dua kali. Masing-masing mencetak "Y" saat objeknya dibuat.
4. Akhirnya, program selesai dieksekusi, dan hasil keluaran yang diharapkan adalah: "ZYXY".

Jadi, urutan cetakan "Z", "X", "Y", "Y" adalah hasil dari urutan pemanggilan konstruktor dan inisialisasi objek dalam hierarki kelas.

Latihan4 ➔ Kompile dan jalankan program berikut! Apa yang terjadi? Jelaskan !

```
public class Hope{
    public static void main(String argv[]){
        Hope h = new Hope();
    }

    protected Hope(){
        for(int i =0; i <10; i ++){
            System.out.println(i);
        }
    }
}
```

Jawab:

Output

```
PS C:\Users\USER\OOP\Praktikum\Praktikum10\Latihan\Latihan4> javac *.java
PS C:\Users\USER\OOP\Praktikum\Praktikum10\Latihan\Latihan4> java Hope
0
1
2
3
4
5
6
7
8
9
PS C:\Users\USER\OOP\Praktikum\Praktikum10\Latihan\Latihan4>
```

Penjelasan

Dalam metode main, objek "Hope" dibuat dengan Hope h = new Hope();. Ini menciptakan sebuah objek dari kelas "Hope."

Kelas "Hope" memiliki sebuah konstruktor protected Hope(). Konstruktor ini memiliki tingkat akses protected, yang berarti hanya dapat diakses oleh kelas dalam paket yang sama atau oleh kelas yang merupakan subkelas dari "Hope."

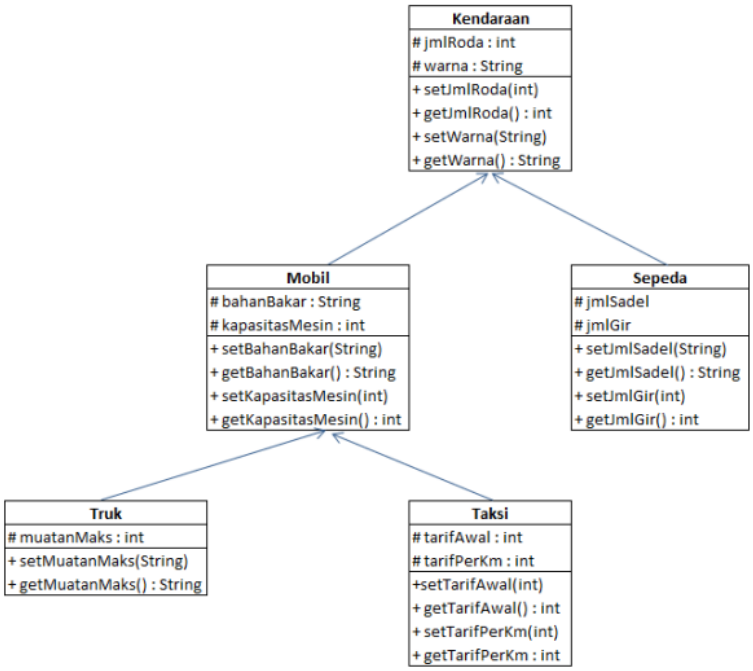
Di dalam konstruktor "Hope," terdapat sebuah loop for yang mencetak angka dari 0 hingga 9. Setiap angka akan dicetak ke konsol dalam urutan dari 0 hingga 9.

Ketika program dijalankan, metode main akan membuat objek "Hope." Saat objek tersebut dibuat, konstruktor "Hope" akan dijalankan, dan loop for akan mencetak angka dari 0 hingga 9 ke konsol. Hasil cetakan ini akan mengikuti urutan angka dari 0 hingga 9.

C. TUGAS

Mengimplementasikan UML class diagram dalam program.

a. Buatlah kelas-kelas berdasarkan UML class diagram dibawah ini!



Jawab:

Kendaraan.java

```
public class Kendaraan {
    protected int jmlRoda;
    protected String warna;

    public void setJmlRoda(int jmlRoda) {
        this.jmlRoda = jmlRoda;
    }

    public int getJmlRoda() {
        return jmlRoda;
    }

    public void setWarna(String warna) {
        this.warna = warna;
    }

    public String getWarna() {
        return warna;
    }
}
```

Mobil.java

```
public class Mobil extends Kendaraan {
    protected String bahanBakar;
    protected int kapasitasMesin;

    public void setBahanBakar(String bahanBakar) {
        this.bahanBakar = bahanBakar;
    }

    public String getBahanBakar() {
        return bahanBakar;
    }

    public void setKapasitasMesin(int kapasitasMesin) {
        this.kapasitasMesin = kapasitasMesin;
    }

    public int getKapasitasMesin() {
        return kapasitasMesin;
    }
}
```

Sepeda.java

```
public class Sepeda extends Kendaraan {
    protected int jmlSadel;
```

```
    protected int jmlGir;

    public void setJmlSadel(int jmlSadel) {
        this.jmlSadel = jmlSadel;
    }

    public int getJmlSadel() {
        return jmlSadel;
    }

    public void setJmlGir(int jmlGir) {
        this.jmlGir = jmlGir;
    }

    public int getJmlGir() {
        return jmlGir;
    }
}
```

Taksi.java

```
public class Taksi extends Mobil {
    protected int tarifAwal;
    protected int tarifPerKm;

    public void setTarifAwal(int tarifAwal) {
        this.tarifAwal = tarifAwal;
    }

    public int getTarifAwal() {
        return tarifAwal;
    }

    public void setTarifPerKm(int tarifPerKm) {
        this.tarifPerKm = tarifPerKm;
    }

    public int getTarifPerKm() {
        return tarifPerKm;

    }
}
```

Truk.java

```
public class Truk extends Mobil {
    protected int muatanMaks;

    public void setMuatanMaks(int muatanMaks) {
```

```
        this.muatanMaks = muatanMaks;
    }

    public int getMuatanMaks() {
        return muatanMaks;
    }
}
```

b. Selanjutnya buatlah kelas Tes.java yang membuat obyek-obyek serta mengeset nilai variabel seperti pada Tabel 9.2.dan tampilkan data per obyek.

Tabel 9.2. Data obyek

Obyek	jmlRoda	warna	bahanBakar	kapasitasMesin	muatanMaks	
truk1	4	kuning	solar	1500	1000	
truk2	6	merah	solar	2000	5000	
					tarifAwal	tarifPerKm
taksi1	4	oranye	bensin	1500	10000	5000
taksi1	4	biru	bensin	1300	7000	3500
			jmlSadel	jmlGir		
sepeda1	3	hitam	1	2		
sepeda2	2	putih	2	5		

Jawab :

Tes.java

```
public class Tes {
    public static void main(String[] args) {
        Truk truk1 = new Truk();
        truk1.setJmlRoda(4);
        truk1.setWarna("kuning");
        truk1.setBahanBakar("solar");
        truk1.setKapasitasMesin(1500);
        truk1.setMuatanMaks(1000);
        System.out.println("\nObjek Truk 1");
        System.out.println("Jumlah roda: " + truk1.getJmlRoda());
        System.out.println("Warna: " + truk1.getWarna());
        System.out.println("Bahan bakar: " + truk1.getBahanBakar());
        System.out.println("Kapasitas mesin: " +
truk1.getKapasitasMesin());
        System.out.println("Muatan maks: " + truk1.getMuatanMaks());
        Truk truk2 = new Truk();
        truk2.setJmlRoda(6);
        truk2.setWarna("merah");
        truk2.setBahanBakar("solar");
        truk2.setKapasitasMesin(2000);
        truk2.setMuatanMaks(5000);
        System.out.println("\nObjek Truk 2");
        System.out.println("Jumlah roda: " + truk2.getJmlRoda());
```

```
        System.out.println("Warna: " + truk2.getWarna());
        System.out.println("Bahan bakar: " + truk2.getBahanBakar());
        System.out.println("Kapasitas mesin: " +
truk2.getKapasitasMesin());
        System.out.println("Muatan maks: " + truk2.getMuatanMaks());
        Taksi taksi1 = new Taksi();
        taksi1.setJmlRoda(4);
        taksi1.setWarna("oranye");
        taksi1.setBahanBakar("bensin");
        taksi1.setKapasitasMesin(1500);
        taksi1.setTarifAwal(10000);
        taksi1.setTarifPerKm(5000);
        System.out.println("\nObjek Taksi 1");
        System.out.println("Jumlah roda: " + taksi1.getJmlRoda());
        System.out.println("Warna: " + taksi1.getWarna());
        System.out.println("Bahan bakar: " + taksi1.getBahanBakar());
        System.out.println("Kapasitas mesin: " +
taksi1.getKapasitasMesin());
        System.out.println("Tarif awal: " + taksi1.getTarifAwal());
        System.out.println("Tarif per km: " + taksi1.getTarifPerKm());
        Taksi taksi2 = new Taksi();
        taksi2.setJmlRoda(4);
        taksi2.setWarna("biru");
        taksi2.setBahanBakar("bensin");
        taksi2.setKapasitasMesin(1300);
        taksi2.setTarifAwal(7000);
        taksi2.setTarifPerKm(3500);
        System.out.println("\nObjek Taksi 2");
        System.out.println("Jumlah roda: " + taksi2.getJmlRoda());
        System.out.println("Warna: " + taksi2.getWarna());
        System.out.println("Bahan bakar: " + taksi2.getBahanBakar());
        System.out.println("Kapasitas mesin: " +
taksi2.getKapasitasMesin());
        System.out.println("Tarif awal: " + taksi2.getTarifAwal());
        System.out.println("Tarif per km: " + taksi2.getTarifPerKm());
        Sepeda sepeda1 = new Sepeda();
        sepeda1.setJmlRoda(3);
        sepeda1.setWarna("hitam");
        sepeda1.setJmlSadel("1");
        sepeda1.setJmlGir(2);
        System.out.println("\nObjek Sepeda 1");
        System.out.println("Jumlah roda: " + sepeda1.getJmlRoda());
        System.out.println("Warna: " + sepeda1.getWarna());
        System.out.println("Jumlah sadel: " + sepeda1.getJmlSadel());
        System.out.println("Jumlah gir: " + sepeda1.getJmlGir());
        Sepeda sepeda2 = new Sepeda();
        sepeda2.setJmlRoda(2);
        sepeda2.setWarna("putih");
```

```
        sepeda2.setJmlSadel("2");
        sepeda2.setJmlGir(5);
        System.out.println("\nObjek Sepeda 2");
        System.out.println("Jumlah roda: " + sepeda2.getJmlRoda());
        System.out.println("Warna: " + sepeda2.getWarna());
        System.out.println("Jumlah sadel: " + sepeda2.getJmlSadel());
        System.out.println("Jumlah gir: " + sepeda2.getJmlGir());
    }
}
```

Output

```
PS C:\Users\USER\OOP\Praktikum\Praktikum10\Tugas> javac *.java
PS C:\Users\USER\OOP\Praktikum\Praktikum10\Tugas> java Tes

Objek Truk 1
Jumlah roda: 4
Warna: kuning
Bahan bakar: solar
Kapasitas mesin: 1500
Muatan maks: 1000

Objek Truk 2
Jumlah roda: 6
Warna: merah
Bahan bakar: solar
Kapasitas mesin: 2000
Muatan maks: 5000

Objek Taksi 1
Jumlah roda: 4
Warna: oranye
Bahan bakar: bensin
Kapasitas mesin: 1500
Tarif awal: 10000
Tarif per km: 5000

Objek Taksi 2
Jumlah roda: 4
Warna: biru
Bahan bakar: bensin
Kapasitas mesin: 1300
Tarif awal: 7000
Tarif per km: 3500

Objek Sepeda 1
Jumlah roda: 3
Warna: hitam
Jumlah sadel: 1
Jumlah gir: 2

Objek Sepeda 2
Jumlah roda: 2
Warna: putih
Jumlah sadel: 2
Jumlah gir: 5
PS C:\Users\USER\OOP\Praktikum\Praktikum10\Tugas>
```

Penjelasan

Pada kode tersebut terdapat kelas Truk, Taksi, dan Sepeda. Masing-masing kelas ini memiliki atribut-atribut yang mendefinisikan karakteristik kendaraan tersebut, seperti jumlah roda, warna, bahan bakar, kapasitas mesin, tarif awal (untuk taksi), dan atribut khusus lainnya.

Setiap kelas memiliki metode setter (misalnya, setJmlRoda, setWarna, dll.) untuk mengatur nilai atribut objek dan metode getter (misalnya, getJmlRoda, getWarna, dll.) untuk mengambil nilai-nilai atribut objek.

Dalam metode main, beberapa objek dari masing-masing kelas (Truk, Taksi, dan Sepeda) dibuat menggunakan operator new. Kemudian, atribut-atribut objek diatur dengan nilai tertentu menggunakan metode setter.

Setelah mengatur atribut objek, kode mencetak informasi kendaraan ke konsol dengan menggunakan metode getter. Ini memungkinkan untuk melihat karakteristik dari masing-masing objek yang telah dibuat.

D. KESIMPULAN

Inheritance adalah salah satu konsep utama dalam pemrograman berorientasi objek (OOP) yang memungkinkan pembuatan hierarki kelas dengan hubungan "is-a." Dalam Java, ada empat modifier pengontrolan akses utama: `public`, `protected`, `default`, dan `private`, yang mengatur visibilitas anggota dalam inheritance. Kata kunci `super` digunakan untuk mengakses metode yang diwarisi dari kelas dasar, terutama dalam konteks overriding, dan untuk memanggil konstruktor kelas dasar dalam konstruktor kelas anak. Namun, konstruktor tidak diwariskan, yang berarti bahwa kelas anak harus secara eksplisit memanggil konstruktor kelas dasar menggunakan `super` untuk memastikan inisialisasi kelas dasar terjadi. Konsep ini penting dalam pemodelan hubungan antar objek dan pengorganisiran kode dalam Java, dan dengan pemahaman yang baik tentang inheritance, programmer dapat merancang hierarki kelas yang kuat dan efisien.