

PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK

Nama	Ruben Kristanto	No. Modul	3
NPM	2306214624	Tipe	TP

1. Encapsulation adalah salah satu konsep OOP yang membungkus field/atribut dan method dalam sesuatu yang disebut class, di dalam class sendiri kemudian ada yang disebut dengan visibility modifier yang membantu kita untuk membungkus data lebih erat lagi tergantung kepada modifier yang digunakan. Tujuan dari encapsulation adalah data hiding dimana field dan method tidak bisa diakses secara langsung dari luar kode.

Berikut adalah beberapa kegunaan atau keuntungan dari encapsulation:

- Membantu mengontrol value yang diberikan kepada variabel-variabel yang ada di fields.
- Dapat berfungsi sebagai error prevention/validation yang handle apakah data yang diinput sudah valid dan sesuai.
- Sebagai data transformer dimana data input dapat diubah menjadi unit/format yang sesuai untuk disimpan pada variabel tersebut.
- Data Hiding, dimana akses terhadap implementasi sebuah class disembunyikan dari luar.

Referensi :

[1] netlab "DASTE Modul 3: Encapsulation, Polymorphism, and UML," Praktikum Pemrograman Berorientasi Objek, 2024. Available: /mnt/data/DASTE MODUL 3.pdf.

2. Getter adalah jenis method yang digunakan untuk mengambil atau mereturn suatu value dari luar suatu class yang fieldnya hanya bisa diakses oleh class itu atau subclassnya sendiri.
Setter adalah method yang kita gunakan untuk menetapkan atau mengubah sebuah nilai dari atribut suatu objek yang isi classnya hanya bisa diakses oleh class itu atau subclassnya sendiri.

Contohnya adalah :

```
Public class Shape{  
    Private String nama;  
  
    Public Shape(String nama){  
        This.nama = nama;  
    }  
    Public String getNama(){  
        Return nama;  
    }  
    Public void setNama(String nama){  
        This.nama = nama  
    }  
}
```

```
Class utama{  
    Public static void main(String[] args){  
        Shape bentuk = new Shape("Segitiga");  
        //set atau menetapkan nama menjadi kotak yang sebelumnya segitiga  
        Bentuk.setNama("kotak");  
        //untuk ngeprint nama dari objek bentuk dengan class Shape, yang akan diprint  
        adalah "kotak"  
        System.out.println(bentuk.getNama());  
    }  
}
```

Referensi :

[1] netlab "DASTE Modul 3: Encapsulation, Polymorphism, and UML," Praktikum Pemrograman Berorientasi Objek, 2024. Available: /mnt/data/DASTE MODUL 3.pdf.

3. Inheritance adalah sebuah konsep oop yang sesuai namanya mewariskan sifat-sifat berupa field dan method suatu class yang disebut parent atau super class kepada class lain yang disebut child class dengan keyword extends sedangkan encapsulation adalah konsep oop dengan tujuan data hiding dimana field dan method tersebut tidak bisa diakses langsung dari luar kode.

Referensi :

[1] netlab "DASTE Modul 3: Encapsulation, Polymorphism, and UML," Praktikum Pemrograman Berorientasi Objek, 2024. Available: /mnt/data/DASTE MODUL 3.pdf.

4. Polimorfisme adalah salah satu konsep utama dalam Pemrograman Berorientasi Objek (OOP) yang memungkinkan aksi yang sama dilakukan dengan cara berbeda. Ini memungkinkan sebuah kelas yang mewarisi dari kelas lain untuk menggunakan properti dari kelas induk sesuai kebutuhan. Selain itu, polimorfisme memungkinkan metode dengan nama yang sama memiliki parameter yang berbeda. Polimorfisme dapat dianalogikan dengan bagaimana seseorang bisa berperan sebagai orang tua, kakak, atau teman dalam situasi yang berbeda. Polimorfisme dibagi menjadi dua jenis: Polimorfisme Statis, yang dicapai melalui overloading metode, dan Polimorfisme Dinamis, yang dicapai melalui overriding metode.

Referensi :

[1] netlab "DASTE Modul 3: Encapsulation, Polymorphism, and UML," Praktikum Pemrograman Berorientasi Objek, 2024. Available: /mnt/data/DASTE MODUL 3.pdf.

5. Polimorfisme Statis

Polimorfisme statis juga dikenal sebagai polimorfisme waktu kompilasi, yang berarti bahwa kita dapat menulis banyak metode dalam program dengan nama yang sama namun melakukan tugas yang berbeda.

Namun, hal ini memberikan keuntungan kepada pengguna atau pengembang perangkat lunak berupa efisiensi dan keterbacaan kode yang lebih baik.

Polimorfisme Runtime (Polimorfisme Dinamis)

Polimorfisme ini juga dikenal sebagai Polimorfisme Dinamis. Ini adalah proses di mana

pemanggilan metode yang di-override diselesaikan saat runtime, itulah sebabnya disebut polimorfisme runtime.

Perbedaan Utama

- Pada Polimorfisme Statis, pemanggilan diselesaikan oleh compiler, sedangkan pada Polimorfisme Runtime, pemanggilan tidak diselesaikan oleh compiler.
- Polimorfisme Statis juga disebut sebagai Polimorfisme Waktu Kompilasi dan Pengikatan Awal (Early Binding), sedangkan Polimorfisme Runtime disebut sebagai Pengikatan Dinamis (Dynamic Binding), Pengikatan Akhir (Late Binding), dan juga Overriding.
- Overloading adalah polimorfisme waktu kompilasi di mana lebih dari satu fungsi memiliki nama yang sama dengan parameter atau tanda tangan yang berbeda dan tipe pengembalian yang berbeda, sedangkan Overriding adalah polimorfisme runtime yang memiliki fungsi yang sama dengan parameter atau tanda tangan yang sama, tetapi terkait dengan kelas dan subclass-nya.
- Polimorfisme Statis dicapai melalui overloading fungsi dan overloading operator, sedangkan Polimorfisme Runtime dicapai melalui pointer dan fungsi virtual.
- Pada Polimorfisme Statis, karena dianalisis lebih awal pada waktu kompilasi, eksekusinya lebih cepat, sedangkan Polimorfisme Runtime lebih lambat karena dianalisis saat runtime.
- Pada Polimorfisme Statis, semua proses dikelola dan dieksekusi pada waktu kompilasi yang membuatnya kurang fleksibel, sedangkan Polimorfisme Runtime lebih fleksibel karena dieksekusi pada waktu runtime.

Referensi :

[1]

“Difference Between Static And Dynamic Polymorphism | Programmerbay,” Mar. 13, 2019.

<https://programmerbay.com/difference-between-static-and-runtime-polymorphism/>

Accessed : September 15 2024

6. Contoh Overloading dimana method add memiliki 2 nama yang sama dengan isi yang berbeda

```
class MethodOverloadingEx {  
  
    static int add(int a, int b) { return a + b; }  
  
    static int add(int a, int b, int c)  
    {  
        return a + b + c;  
    }  
  
    // Main Function  
    public static void main(String args[])  
    {  
        System.out.println("add() with 2 parameters");  
        // Calling function with 2 parameters  
        System.out.println(add(4, 6));  
  
        System.out.println("add() with 3 parameters");  
        // Calling function with 3 Parameters  
        System.out.println(add(4, 6, 7));  
    }  
}
```

```
class Animal {  
    void eat() {  
        System.out.println("eat() method of base class");  
        System.out.println("Animal is eating.");  
    }  
}  
  
// Derived Class  
class Dog extends Animal {  
    @Override  
    void eat() {  
        System.out.println("eat() method of derived class");  
        System.out.println("Dog is eating.");  
    }  
  
    // Method to call the base class method  
    void eatAsAnimal() {  
        super.eat();  
    }  
}
```

```
}
```

```
// Driver Class
```

```
class MethodOverridingEx {
```

```
    // Main Function
```

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

```
        Dog d1 = new Dog();
```

```
        Animal a1 = new Animal();
```

```
        // Calls the eat() method of Dog class
```

```
        d1.eat();
```

```
        // Calls the eat() method of Animal class
```

```
        a1.eat();
```

```
        // Polymorphism: Animal reference pointing to Dog object
```

```
        Animal animal = new Dog();
```

```
        // Calls the eat() method of Dog class
```

```
        animal.eat();
```

```
        // To call the base class method, you need to use a Dog reference
```

```
        ((Dog) animal).eatAsAnimal();
```

```
    }
```

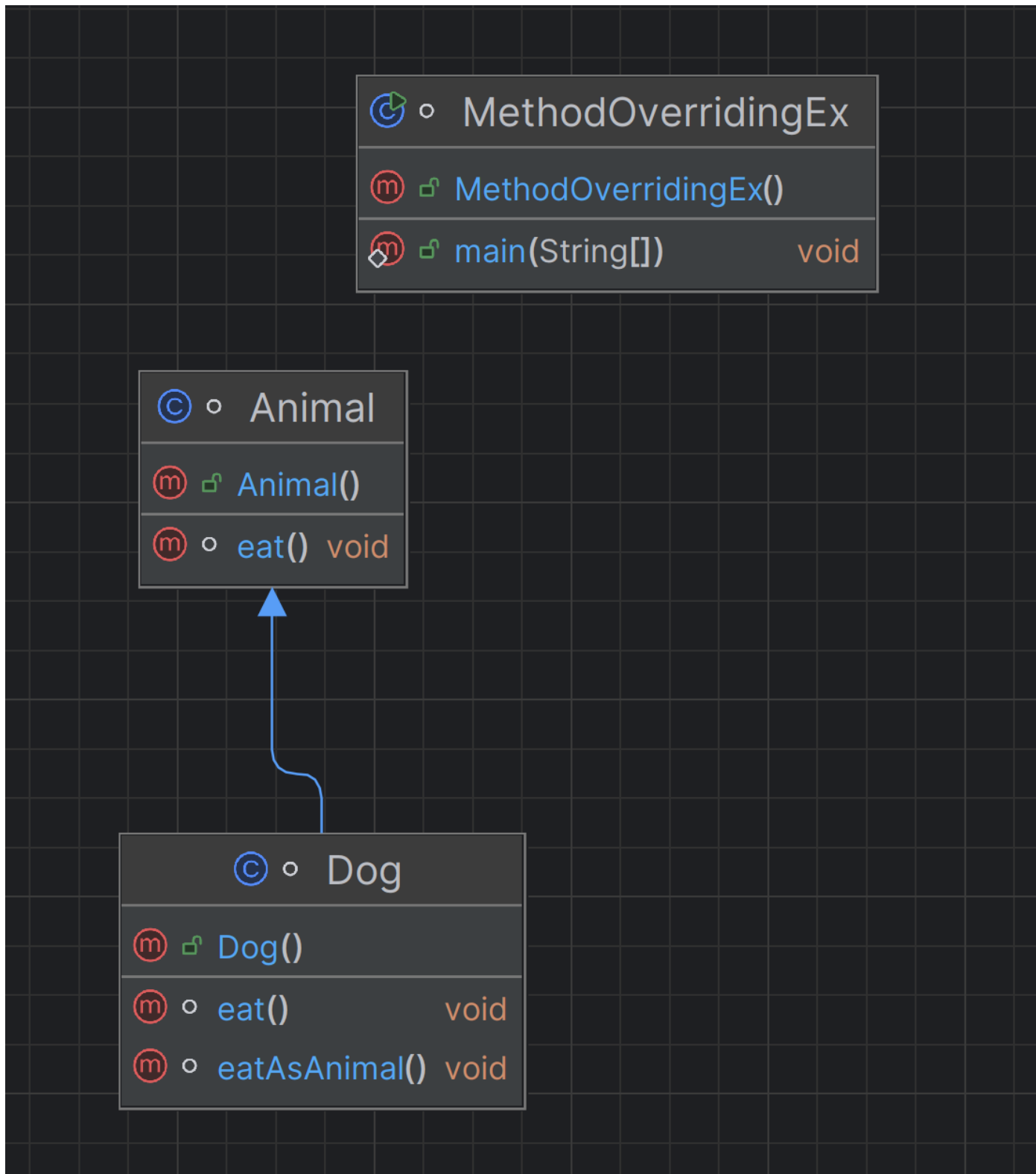
```
}
```

overriding adalah ketika super dan child memiliki method dengan nama dan parameter yang sama tetapi isi yang berbeda.

Referensi :






[1]

“Difference between Method Overloading and Method Overriding in Java,” *GeeksforGeeks*, Oct. 03, 2019. <https://www.geeksforgeeks.org/difference-between-method-overloading-and-method-overriding-in-java/>



7.

Untuk overriding

	MethodOverloadingEx	
	MethodOverloadingEx()	
	add(int, int)	int
	add(int, int, int)	int
	main(String[])	void

overloading