

MODUL PRAKTIKUM

TI03209 – PEMROGRAMAN BERORIENTASI OBJEK

Disusun oleh
Eko Mailansa, S.Kom., M.T.



D3 – TEKNIK INFORMATIKA
POLITEKNIK HASNUR
BARTIO KUALA
TAHUN 2023

MODUL III

Inheritance

III.1. Tujuan Praktikum

Praktikum yang dilakukan pada modul 3 bertujuan untuk:

1. Mengerti dan memahami konsep dari *Inheritance* dalam pemrograman Java
2. Mengerti dan memahami perbedaan antara *IS-A relationship* dan *HAS-A relationship* dalam implementasi *Inheritance*

III.2. Indikator Pencapaian

Setelah melakukan praktikum maka, di harapkan praktikan mampu:

1. Membuat program sederhana dengan mengimplementasikan *extends* dalam *Inheritance*
2. Membuat program sederhana yang mengimplementasikan implementasi *IS-A relationship*
3. Membuat program sederhana yang mengimplementasikan implementasi *HAS-A relationship*

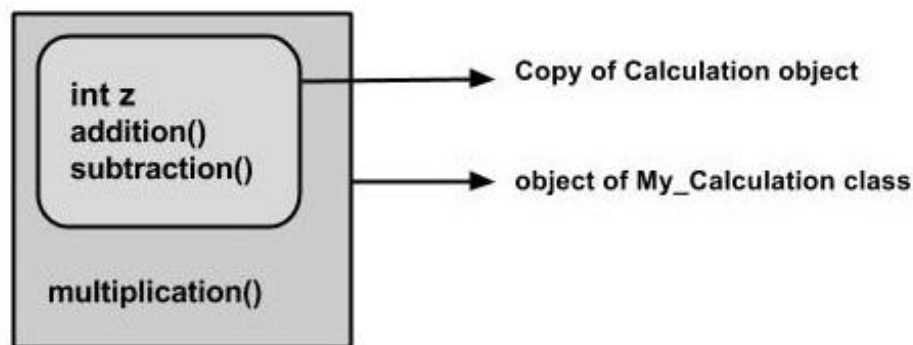
III.3. Materi

Inheritance adalah fitur mendasar dari desain dan pemrograman berorientasi objek. Meskipun penggunaan kelas asosiasi, agregasi dan komposisi sama pentingnya dengan pewarisan.

Ketika mengembangkan program nyata, perbedaan pewarisan adalah dukungan implementasinya harus disediakan oleh bahasa pemrograman berorientasi objek. Dalam dukungan bahasa untuk pewarisan, C++ paling berbeda dari C atau Java berbeda dari bahasa sebelumnya seperti Pascal.

Berikut adalah Contoh yang menunjukkan warisan Java pada (praktikum nomor 1). Dalam Contoh ini, Anda dapat mengamati dua kelas yaitu Calculation dan My_Calculation. Menggunakan kata kunci *extends*, My_Calculation mewarisi metode penambahan() dan Pengurangan() dari kelas Perhitungan.

Dalam program praktikum nomor 1 yang di lakukan, ketika sebuah objek kelas **My_Calculation** dibuat, salinan konten superkelas dibuat di dalamnya. Itu sebabnya, dengan menggunakan objek subkelas Anda dapat mengakses anggota superkelas.



Variabel referensi Superclass dapat menampung objek subclass, tetapi dengan menggunakan variabel tersebut hanya dapat mengakses anggota superclass, sehingga untuk mengakses anggota kedua kelas disarankan untuk selalu membuat variabel referensi ke subclass.

Jika mempertimbangkan program yang dilakukan pada program praktikum nomor 1, Anda dapat membuat instance kelas seperti yang diberikan pada program. Tetapi dengan menggunakan variabel referensi superkelas (**cal** dalam kasus ini) Anda tidak dapat memanggil metode **multiplication()**, yang termasuk dalam subclass **My_Calculation**.

Note - Sebuah subclass mewarisi semua anggota (atribut, metode, dan kelas bersarang) dari superkelasnya. Konstruktor bukan anggota, sehingga tidak diwarisi oleh subclass, namun konstruktor superkelas dapat dipanggil dari subclass.

Keyword super

Kata kunci **super** mirip dengan kata kunci **this**. Berikut adalah skenario penggunaan kata kunci **super**.

- Digunakan untuk **membedakan anggota** superclass dari anggota subclass, jika mereka memiliki nama yang sama.
- Ini digunakan untuk **memanggil konstruktor superkelas** dari subclass.

Membedakan Anggota (*member*)

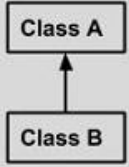
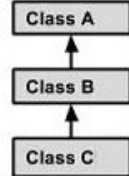
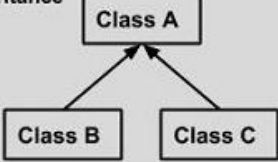
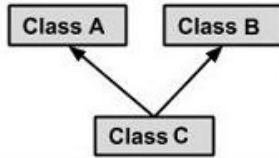
Jika suatu kelas mewarisi properti kelas lain. Dan jika anggota superclass memiliki nama yang sama dengan subkelasnya, untuk membedakan variabel tersebut kita menggunakan kata kunci *super* seperti di bawah ini.

`super.variable`

`super.method();`

Bagian ini memberikan program yang mendemonstrasikan penggunaan kata kunci **super** . Dalam program yang diberikan, Anda memiliki dua kelas yaitu *Sub_class* dan *Super_class* pada program praktikum nomor 2, keduanya memiliki metode bernama `display()` dengan implementasi berbeda dan variabel bernama **num** dengan nilai berbeda. Dilakukan pemanggilan metode `display()` dari kedua kelas dan mencetak nilai variabel **num** dari kedua kelas. Program yang di tulis ini telah menggunakan kata kunci *super* untuk membedakan anggota superkelas dari subkelas.

Types of Inheritance

Single Inheritance  <pre>graph BT; B[Class B] --> A[Class A]</pre>	<pre>public class A { } public class B extends A { }</pre>
Multi Level Inheritance  <pre>graph BT; C[Class C] --> B[Class B]; B --> A[Class A]</pre>	<pre>public class A {} public class B extends A {.....} public class C extends B {..... }</pre>
Hierarchical Inheritance  <pre>graph BT; B[Class B] --> A[Class A]; C[Class C] --> A</pre>	<pre>public class A {} public class B extends A {.....} public class C extends A {..... }</pre>
Multiple Inheritance  <pre>graph BT; A[Class A] --> C[Class C]; B[Class B] --> C</pre>	<pre>public class A {} public class B {.....} public class C extends A,B { } // Java does not support mutiple Inheritance</pre>

III.3.1.IS-A Relationship

IS-A adalah cara untuk mengatakan: Objek adalah tipe dari objek itu. Mari kita lihat bagaimana kata kunci **extends** digunakan untuk mencapai *inheritance*.

```
public class Animal {  
}  
  
public class Mammal extends Animal {  
}  
  
public class Reptile extends Animal {  
}
```

Dengan menggunakan kata kunci **extends**, subkelas akan dapat mewarisi semua properti superkelas kecuali *private properties of the superclass*.

Sekarang bagaimana kata kunci **implements** digunakan untuk mendapatkan hubungan IS-A. Umumnya, kata kunci **implements** digunakan dengan kelas untuk mewarisi properti interface. Interface tidak pernah dapat di **extends** oleh suatu kelas.

```
public interface Animal {  
}  
  
public class Mammal implements Animal {  
}  
  
public class Dog extends Mammal {  
}
```

The instanceof Keyword

Operator **instanceof** di gunakan untuk memeriksa apakah **Mammal** sebenarnya adalah **Animal** dan **Dog** sebenarnya adalah **Animal**.

III.3.2.HAS-A Relationship

Hubungan ini terutama didasarkan pada penggunaan. Ini menentukan apakah kelas tertentu **HAS-A** hal tertentu. Hubungan ini membantu mengurangi duplikasi kode serta bug. Contoh:

```

public class Vehicle{}
public class Speed{}

public class Van extends Vehicle {
    private Speed sp;
}

```

Hal ini menunjukkan bahwa kelas **Van HAS-A Speed**. Dengan memiliki kelas terpisah untuk **Speed**, kita tidak perlu memasukkan seluruh kode milik kecepatan ke dalam kelas **Van**, sehingga memungkinkan untuk menggunakan kembali kelas **Speed** dalam beberapa aplikasi.

Untuk mencapai hal ini, kelas **Van** menyembunyikan detail implementasi dari pengguna kelas **Van**. Jadi, pada dasarnya yang terjadi adalah pengguna akan meminta kelas **Van** untuk melakukan tindakan tertentu dan kelas **Van** akan melakukan pekerjaan itu sendiri atau meminta kelas lain untuk melakukan tindakan tersebut.

III.4. Alat dan Bahan

Peralatan dan bahan-bahan yang digunakan antara lain:

- ✓ Satu set Komputer
- ✓ Aplikasi IDE Netbeans 8.2
- ✓ Modul 3 – Mata kuliah Pemrograman Berorientasi Objek

III.5. Praktikum

1. *Inheritance1*

a. class Calculation

```

public class Calculation {

    int z;

    public void addition(int x, int y) {
        z = x + y;
        System.out.println("x + y : " + z);
    }
}

```



```

        public void Subtraction(int x, int y) {
            z = x - y;
            System.out.println("x - y : " + z);
        }
    }
}

```

b. class My_Calculation

```

public class My_Calculation extends Calculation {

    public void multiplication(int x, int y) {
        z = x * y;
        System.out.println("x * y : " + z);
    }

    public static void main(String args[]) {
        int a = 20;
        int b = 10;
        My_Calculation demo = new My_Calculation();
        demo.addition(a, b);
        demo.Subtraction(a, b);
        demo.multiplication(a, b);
    }
}

```

2. Inheritance2

a. Super_class

```

public class Super_class {

    int num = 20;

    // display method of superclass
    public void display() {
        System.out.println("Ini adalah method of superclass");
    }
}

```

b. Sub_class

```
public class Sub_class extends Super_class {  
  
    int num = 10;  
  
    // display method of sub class  
    public void display() {  
        System.out.println("Ini adalah method of subclass");  
    }  
  
    public void my_method() {  
        // Instantiating subclass  
        Sub_class sub = new Sub_class();  
  
        // Invoking the display() method of sub class  
        sub.display();  
  
        // Invoking the display() method of superclass  
        super.display();  
  
        // printing the value of variable num of subclass  
        System.out.println("Nilai dari variabel num dalam sub class:" + sub.num);  
  
        // printing the value of variable num of superclass  
        System.out.println("Nilai dari variabel num dalam super class:" + super.num);  
    }  
  
    public static void main(String args[]) {  
        Sub_class obj = new Sub_class();  
        obj.my_method();  
    }  
}
```

3. IS-A Relationship

```
class Animal {  
}  
  
class Mamalia extends Animal {  
}  
  
class Reptile extends Animal {  
}  
  
public class Cat extends Mamalia {  
}
```



```

    public static void main(String args[]) {
        Animal a = new Animal();
        Mamalia m = new Mamalia();
        Cat c = new Cat();

        System.out.println(m instanceof Animal);
        System.out.println(c instanceof Mamalia);
        System.out.println(c instanceof Animal);
    }
}

```

4. *IS-A Relationship*

```

interface Animal1 {
}

class Mamalia1 implements Animal1 {
}

public class MyCat extends Mamalia1 {

    public static void main(String args[]) {
        Mamalia1 m = new Mamalia1();
        MyCat c = new MyCat();

        System.out.println(m instanceof Animal1);
        System.out.println(c instanceof Mamalia1);
        System.out.println(c instanceof Animal1);
    }
}

```

III.6. Referensi

III.6.1. Utama

- ✓ Pecinovský, Rudolf, CSc. OOP – Learn Object Oriented Thinking and Programming. ISBN 978-80-904661-8-0 (paperback) & ISBN 978-80-904661-9-7 (PDF). Published in the Czech Republic by Tomáš Bruckner, Řepín – Živonín, Academic Series. 2013.
- ✓ Wiener, Richard (University of Colorado, Colorado Springs) & J. Pinson, Lewis (University of Colorado, Colorado Springs). Fundamentals of OOP and Data Structures in Java. ISBN 0 521 66220 6 hardback and eISBN 0-511-00168 -1 virtual (netLibrary Edition). PUBLISHED BY CAMBRIDGE UNIVERSITY PRESS. 2000.

III.6.2. Pendukung

- ✓ Object Oriented Programming in JAVA. Di akses pada 16 Agustus 2023. <https://www.tutorialspoint.com/object-oriented-programming-in-java/index.asp>
- ✓ Object Oriented Programming (OOPs) Concept In Java - GreekforGreeks. Di akses pada 16 Agustus 2023. <https://www.google.com/amp/s/www.greekforgreeks.com/object-oriented-programming-oops-concept-in-java/>
- ✓ Object Oriented Thinking and Programming. Di akses pada 16 Agustus 2023. https://www.w3schools.in/java/java_oop.asp

III.7. Tugas Tambahan

Dari program praktikum no.1, ubah atau tambahkan kode program agar nilai dari variabel **a** dan **b** bernilai pecahan dan dapat di masukkan oleh *user/pengguna* ketika program di **run** serta buatlah *method* dengan nama **Division** untuk pembagian angka **a** dan **b**. Gunakan **Scanner** atau **Buffered Reader**.

III.8. Laporan

- ✓ Laporan “Praktikum” di tulis sesuai dengan contoh format yang telah di berikan.
- ✓ Laporan “Tugas Tambahan” di tulis sesuai dengan contoh format yang telah di berikan dan di lampirkan di setiap laporan “Praktikum” sesuai dengan Modul Praktikum.

- ✓ Isi laporan praktikum → Praktikum dan Tugas Tambahan.
- ✓ *Rename* file laporan dengan format nama yang benar (nama_modul + nim + kelas) dan berikan ekstensi atau tipe file .pdf.
(contoh: **Modul1_20202020_3B.pdf**)

