

Modul 8 Inheritance

Tugas Pendahuluan

1. Apa itu konsep "Inheritance" dalam pemrograman berorientasi objek? Berikan definisi singkat dan jelaskan manfaat utama dari penggunaan inheritance.
2. Apa yang dimaksud dengan hubungan "IS-A" dalam konteks inheritance? Berikan contoh konkret untuk mengilustrasikan hubungan ini.
3. Apa perbedaan antara superclass dan subclass dalam inheritance? Berikan penjelasan dan contoh mengenai kedua konsep ini.
4. Bagaimana cara mendefinisikan suatu kelas sebagai subclass dari kelas lain dalam Java? Berikan sintaksis penggunaan kata kunci extends.
5. Jelaskan konsep "Code Reusability" (Penggunaan Kembali Kode) dalam konteks inheritance. Bagaimana inheritance dapat membantu dalam mencapai penggunaan kembali kode?

Materi:

Inheritance adalah konsep dalam pemrograman berorientasi objek di mana suatu kelas (subclass atau child class) dapat mewarisi atribut dan metode dari kelas lain (superclass atau parent class). Dengan inheritance, kita dapat membuat hierarki kelas di mana kelas yang lebih spesifik (subclass) dapat menggunakan dan memperluas fungsionalitas kelas yang lebih umum (superclass). Manfaat utama dari inheritance adalah:

1. **Reusable Code:** Inheritance memungkinkan penggunaan kembali kode yang sudah ada dalam superclass, sehingga mengurangi duplikasi kode dan mempermudah pemeliharaan kode.
2. **Hierarchical Organization:** Kode dapat diorganisasi dalam hierarki, dengan kelas-kelas yang lebih spesifik diwarisi dari kelas-kelas yang lebih umum.
3. **Polymorphism:** Dengan inheritance, objek dari subclass dapat dianggap sebagai objek dari superclass. Ini memungkinkan penggunaan polimorfisme, di mana objek dari berbagai kelas yang berbeda dapat diakses melalui referensi yang sama.

Contoh Inheritance:

```
1. // Superclass (Parent class)
2. class Animal {
3.     String name;
4.
5.     void eat() {
6.         System.out.println(name + " is eating.");
7.     }
8. }
9.
10. // Subclass (Child class) yang mewarisi dari Animal
11. class Dog extends Animal {
12.     Dog(String name) {
```

```

13.         this.name = name;
14.     }
15.
16.     void bark() {
17.         System.out.println(name + " is barking.");
18.     }
19. }
20.
21. public class Main {
22.     public static void main(String[] args) {
23.         Dog dog = new Dog("Buddy");
24.         dog.eat(); // Memanggil method dari superclass
25.         dog.bark(); // Memanggil method dari subclass
26.     }
27. }
28.

```

Dalam contoh di atas, **Animal** adalah superclass, dan **Dog** adalah subclass yang mewarisi dari **Animal**. Subclass **Dog** mendapatkan atribut **name** dan method **eat()** dari superclass **Animal**. Hal ini memungkinkan kita untuk menggunakan kode yang sudah ada dalam **Animal** tanpa perlu mengulangi kode di **Dog**.

Selain inheritance tunggal seperti dalam contoh di atas, Java juga mendukung multiple inheritance melalui interface. Interface adalah kontrak yang mendefinisikan sekumpulan method yang harus diimplementasikan oleh kelas yang mengimplementasikan interface tersebut.

1. Extends Keyword

Kata kunci `extends` digunakan dalam Java untuk mengimplementasikan konsep inheritance, di mana suatu kelas (subclass atau child class) dapat mewarisi atribut dan metode dari kelas lain (superclass atau parent class). Dalam konteks inheritance, kata kunci `extends` mengindikasikan bahwa suatu kelas akan mewarisi sifat dari kelas lain. Sintaksis dasar penggunaan `extends` adalah sebagai berikut:

```

class Subclass extends Superclass {
    // Isi definisi subclass
}

```

Contoh penggunaan `extends`:

```

1. class Animal {
2.     String name;
3.
4.     void eat() {
5.         System.out.println(name + " is eating.");
6.     }
7. }
8.
9. class Dog extends Animal {
10.    void bark() {
11.        System.out.println(name + " is barking.");
12.    }

```

```

13. }
14.
15. public class Main {
16.     public static void main(String[] args) {
17.         Dog dog = new Dog();
18.         dog.name = "Buddy";
19.         dog.eat(); // Memanggil method dari superclass
20.         dog.bark(); // Memanggil method dari subclass
21.     }
22. }
23.

```

Dalam contoh di atas, Dog adalah subclass yang meng-extends kelas Animal. Oleh karena itu, Dog mewarisi atribut name dan method eat() dari kelas Animal. Dengan menggunakan kata kunci extends, Dog memiliki akses ke atribut dan method dari Animal, dan kita juga bisa menambahkan method dan atribut tambahan di Dog.

Namun, perlu diingat bahwa Java mendukung hanya single inheritance secara langsung, artinya sebuah kelas hanya bisa meng-extends satu kelas lain. Jika Anda perlu menggabungkan fungsionalitas dari beberapa kelas, Anda dapat menggunakan konsep interface atau menerapkan multiple inheritance secara tidak langsung melalui hierarki kelas dan interface.

2. The super keyword

Kata kunci super dalam Java digunakan untuk merujuk ke anggota-anggota (atribut atau method) dari superclass saat Anda berada di dalam subclass. Ini memungkinkan Anda untuk mengakses anggota superclass yang diwarisi oleh subclass atau untuk memanggil constructor superclass dari constructor subclass. Ada dua penggunaan utama dari kata kunci super:

1. Mengakses Anggota Superclass: Anda dapat menggunakan super untuk merujuk ke atribut atau method dari superclass saat nama yang sama digunakan di subclass.
2. Memanggil Constructor Superclass: Anda dapat menggunakan super untuk memanggil constructor dari superclass saat Anda membuat objek dari subclass.

Contoh penggunaan super:

```

1. class Animal {
2.     String name;
3.
4.     Animal(String name) {
5.         this.name = name;
6.     }
7.
8.     void eat() {
9.         System.out.println(name + " is eating.");
10.    }
11. }
12.
13. class Dog extends Animal {
14.     Dog(String name) {
15.         super(name); // Memanggil constructor superclass
16.     }

```

```

17.
18.     void bark() {
19.         System.out.println(name + " is barking.");
20.     }
21.
22.     void display() {
23.         super.eat(); // Memanggil method eat() dari superclass
24.     }
25. }
26.
27. public class Main {
28.     public static void main(String[] args) {
29.         Dog dog = new Dog("Buddy");
30.         dog.display(); // Memanggil method display() yang memanggil method eat() dari superclass
31.     }
32. }
33.

```

Dalam contoh di atas, `super(name)` memanggil constructor superclass `Animal`, dan `super.eat()` memanggil method `eat()` dari superclass. Pemanggilan ini memungkinkan kita untuk menggunakan sifat-sifat dan metode dari superclass dalam subclass.

3. IS-A Relationship

Hubungan "IS-A" (adalah) adalah konsep dalam pemrograman berorientasi objek yang menunjukkan bahwa suatu objek adalah contoh dari sebuah kelas atau tipe objek yang lebih umum. Ini menggambarkan hierarki pewarisan dalam konsep inheritance di mana subclass mewarisi sifat dan perilaku dari superclass.

Dalam konteks "IS-A" relationship, subclass dianggap sebagai jenis yang lebih spesifik dari superclass. Jadi, jika objek dari subclass adalah objek dari superclass, kita bisa mengatakan bahwa objek tersebut "adalah" objek dari superclass.

Contoh hubungan "IS-A":

```

1. class Animal {
2.     // atribut dan metode Animal
3. }
4.
5. class Dog extends Animal {
6.     // atribut dan metode Dog
7. }
8.
9. class Cat extends Animal {
10.    // atribut dan metode Cat
11. }
12.

```

Dalam contoh di atas, kita memiliki hubungan "IS-A" antara `Dog` dan `Animal`, serta antara `Cat` dan `Animal`. Ini karena `Dog` dan `Cat` mewarisi sifat dan perilaku dari `Animal`. Konsep hubungan "IS-A" sangat penting dalam pembentukan hierarki kelas dalam inheritance dan digunakan untuk memahami bagaimana objek-objek dalam program berhubungan satu sama lain.

4. Instanceof Keyword

Kata kunci instanceof dalam Java digunakan untuk memeriksa apakah sebuah objek adalah instance dari suatu kelas atau tipe tertentu. Ini memberikan informasi apakah objek yang diperiksa adalah turunan dari kelas tertentu atau mengimplementasikan suatu interface tertentu.

Sintaksis penggunaan instanceof:

```
obj instanceof ClassType
```

Contoh penggunaan instanceof:

```
1. class Animal {
2.     // atribut dan metode Animal
3. }
4.
5. class Dog extends Animal {
6.     // atribut dan metode Dog
7. }
8.
9. class Cat extends Animal {
10.    // atribut dan metode Cat
11. }
12.
13. public class Main {
14.     public static void main(String[] args) {
15.         Animal animal = new Dog();
16.
17.         if (animal instanceof Dog) {
18.             System.out.println("animal adalah instance dari Dog.");
19.         } else if (animal instanceof Cat) {
20.             System.out.println("animal adalah instance dari Cat.");
21.         } else if (animal instanceof Animal) {
22.             System.out.println("animal adalah instance dari Animal.");
23.         }
24.     }
25. }
26.
```

Dalam contoh di atas, instanceof digunakan untuk memeriksa apakah objek animal adalah instance dari kelas Dog, Cat, atau Animal. Karena animal adalah objek Dog, kondisi pertama akan terpenuhi dan pesan "animal adalah instance dari Dog." akan dicetak.

Penggunaan instanceof dapat berguna saat Anda ingin mengidentifikasi tipe objek secara dinamis dalam situasi yang melibatkan polymorphism atau ketika Anda ingin melakukan operasi yang berbeda tergantung pada tipe objek yang diberikan.

5. HAS-A relationship

Hubungan "HAS-A" adalah konsep dalam pemrograman berorientasi objek yang menggambarkan bahwa suatu objek memiliki atau mengandung objek lain sebagai bagian darinya. Ini juga dikenal sebagai komposisi atau agregasi, di mana objek yang satu mengandung objek yang lain sebagai bagian penting.

Dalam konteks "HAS-A" relationship, sebuah kelas dianggap memiliki objek dari kelas lain sebagai atribut atau anggota. Ini menunjukkan bahwa kelas memiliki komponen yang terkait yang merupakan objek dari kelas lain.

Contoh hubungan "HAS-A":

```

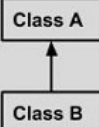
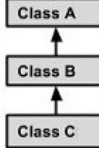
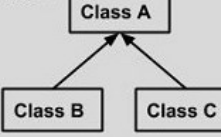
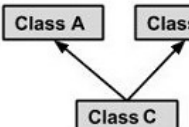
1. class Engine {
2.     // atribut dan metode Engine
3. }
4.
5. class Car {
6.     private Engine engine; // Objek Engine sebagai atribut
7.
8.     Car() {
9.         engine = new Engine(); // Inisialisasi objek Engine
10.    }
11.
12.    // metode-metode Car
13. }
14.

```

Dalam contoh di atas, kelas Car memiliki objek dari kelas Engine sebagai atributnya. Ini adalah contoh hubungan "HAS-A" di mana kelas Car memiliki atau mengandung objek Engine. Konsep hubungan "HAS-A" berguna ketika Anda ingin menggambarkan hubungan antara objek yang lebih besar dan objek yang lebih kecil atau komponen yang ada di dalamnya. Ini membantu dalam memodelkan struktur kompleks dan membuat kode lebih modular.

6. Jenis Inheritance

Berikut adalah jenis-jenis 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>

Fakta yang sangat penting untuk diingat adalah bahwa Java tidak mendukung pewarisan ganda. Ini berarti bahwa sebuah kelas tidak dapat mewarisi dari lebih dari satu kelas. Oleh karena itu, hal berikut dianggap ilegal.

Latihan :

Latihan 1 : Perbaiki program yang salah berikut ini :

```
1. class Vehicle {
2.     String brand;
3.
4.     Vehicle(String brand) {
5.         this.brand = brand;
6.     }
7.
8.     void displayInfo() {
9.         System.out.println("This is a vehicle from " + brand);
10.    }
11. }
12.
13. class Car extends Vehicle {
14.     String modelName;
15.
16.     Car(String brand, String modelName) {
17.         this.modelName = modelName;
18.     }
19.
20.     void displayInfo() {
21.         System.out.println("This is a " + modelName + " car from " + brand);
22.     }
23. }
24.
25. public class Main {
26.     public static void main(String[] args) {
27.         Car car = new Car("Toyota", "Camry");
28.         car.displayInfo();
29.     }
30. }
31.
```

Silahkan perbaiki program diatas, tugasnya termasuk:

1. Memanggil constructor superclass dari constructor subclass Car menggunakan super().
2. Menginisialisasi atribut brand dalam constructor subclass.
3. Menambahkan pemanggilan super.displayInfo() dalam method displayInfo() subclass jika diperlukan.

Latihan 2 : Perbaiki program dibawah ini :

```
1. class Vehicle {
2.     String type;
3.
4.     void showInfo() {
5.         System.out.println("This is a " + type);
6.     }
7. }
8.
```

```

9. class Car extends Vehicle {
10.     String brand;
11.
12.     void showInfo() {
13.         System.out.println("This is a " + brand + " car");
14.     }
15. }
16.
17. public class Main {
18.     public static void main(String[] args) {
19.         Car car = new Car();
20.         car.type = "vehicle";
21.         car.brand = "Toyota";
22.         car.showInfo();
23.     }
24. }
25.

```

Program di atas mengalami kesalahan pada method showInfo() dalam kelas Car. Silahkan perbaiki program tersebut agar menghasilkan output yang diharapkan tanpa error.

Petunjuk :

1. Perhatikan hubungan inheritance antara kelas Vehicle dan Car.
2. Perbaiki method showInfo() dalam kelas Car agar menggabungkan informasi dari superclass dan subclass.
3. Pastikan program tidak mengandung error dan menghasilkan output yang sesuai.

Tugas :

1. Membuat Hierarchy
Buatlah hierarki kelas berdasarkan konsep inheritance dengan minimal dua level pewarisan. Setiap level harus memiliki minimal satu atribut dan satu metode. Anda dapat menggunakan contoh seperti hewan, mobil, manusia, dll.
2. Memperbaiki Kesalahan
Diberikan program di bawah ini, identifikasi kesalahan dalam konsep inheritance dan perbaiki program tersebut agar menghasilkan output yang sesuai:

```

1. class Shape {
2.     String type;
3.
4.     void showInfo() {
5.         System.out.println("This is a " + type);
6.     }
7. }
8.
9. class Circle extends Shape {
10.     double radius;
11.
12.     void showInfo() {
13.         System.out.println("This is a circle with radius " + radius);
14.     }
15. }
16.

```



```
17. public class Main {
18.     public static void main(String[] args) {
19.         Shape shape = new Circle();
20.         shape.type = "shape";
21.         shape.showInfo();
22.     }
23. }
24.
```

3. Menerapkan Polymorphism

Buatlah contoh yang menggunakan konsep polymorphism dengan inheritance. Buat superclass Fruit dengan beberapa atribut dan metode. Kemudian buat beberapa subclass seperti Apple, Banana, dan Orange yang mewarisi dari Fruit. Di dalam main method, buatlah array dari objek-objek Fruit dan panggil metode yang berbeda pada setiap objek.

4. Desain Kelas

Buatlah desain kelas untuk simulasi perpustakaan. Anda bisa memiliki superclass Item yang memiliki atribut seperti judul, penulis, dan tahun terbit. Kemudian, buatlah beberapa subclass seperti Book, Magazine, dan DVD yang mewarisi dari Item. Anda bisa menambahkan atribut tambahan dan metode sesuai dengan jenis item.

Catatan:

1. Anda dapat menggunakan konsep inheritance sesuai dengan tingkat pemahaman Anda.
2. Pastikan program tidak mengandung error saat dijalankan.
3. Tugas ini bertujuan untuk memperkuat pemahaman Anda tentang inheritance dan penggunaannya dalam berbagai situasi.