Menerapkan Pewarisan (*Inheritance*) Antar Class

Definisi Inheritance:

- Inheritance adalah konsep dalam OOP yang memungkinkan sebuah class (subclass atau derived class) untuk mewarisi properti dan method dari class lain (superclass atau base class).

Keuntungan Inheritance:

1. Reusability: Menggunakan kembali kode yang sudah ada tanpa perlu menulis ulang.
2. Organisasi Kode: Membantu dalam mengorganisir dan mengelompokkan kode secara lebih logis.
3. Ekstensi: Memungkinkan pengembangan atau penambahan fungsionalitas baru pada class yang sudah ada.

Hierarki Class dan Subclass:

- Superclass (Base Class): Class yang diwarisi oleh class lain.
- Subclass (Derived Class): Class yang mewarisi properti dan method dari superclass.

Keyword extends:

- Digunakan untuk mendeklarasikan inheritance.
- Sintaks: class Subclass extends Superclass.

Contoh Kode Java yang Menunjukkan Inheritance:

```
// Superclass
public class Animal {
    public void eat() {
        System.out.println("This animal eats food.");
    }
}

// Subclass
public class Dog extends Animal {
    public void bark() {
        System.out.println("The dog barks.");
    }
}

public class animal {
```

```
    public static void main(String[] args) {
        Dog myDog = new Dog();
        myDog.eat(); // Output: This animal eats food.
        myDog.bark(); // Output: The dog barks.
    }
}
```

3.2 Contoh Penggunaan super()

Keyword super():

- Digunakan untuk merujuk pada constructor atau method superclass dari subclass.
- Berguna untuk memanggil constructor superclass dari subclass atau mengakses method yang ditimpa (overridden) dalam subclass.

Memanggil Constructor Superclass:

- super() harus digunakan di baris pertama dari constructor subclass.
- Digunakan untuk memastikan bahwa inisialisasi superclass terjadi sebelum subclass.

Contoh Kode Penggunaan super() dalam Constructor:

```
// Superclass
public class Animal {
    String name;

    // Superclass constructor
    public Animal(String name) {
        this.name = name;
    }
}

// Subclass
public class Dog extends Animal {
    int age;

    // Subclass constructor
    public Dog(String name, int age) {
        super(name); // Call superclass constructor
        this.age = age;
    }

    public void display() {
```

```
        System.out.println("Name: " + name + ", Age: " + age);
    }
}

public class Main {
    public static void main(String[] args) {
        Dog myDog = new Dog("Buddy", 3);
        myDog.display(); // Output: Name: Buddy, Age: 3
    }
}
```

Memanggil Method Superclass yang Ditimpa:

- super.methodName() digunakan untuk memanggil method superclass yang telah di-override dalam subclass.

Contoh Kode Penggunaan super.methodName():

```
// Superclass
public class Animal {
    public void sound() {
        System.out.println("Animal makes a sound");
    }
}

// Subclass
public class Dog extends Animal {
    @Override
    public void sound() {
        super.sound(); // Call superclass method
        System.out.println("Dog barks");
    }
}

public class Main {
    public static void main(String[] args) {
        Dog myDog = new Dog();
        myDog.sound();
        // Output:
        // Animal makes a sound
        // Dog barks
    }
}
```

Kesimpulan Pewarisan (*Inheritance*)

Pada bab ini, kita telah mempelajari konsep inheritance dalam OOP dengan Java. Inheritance memungkinkan kita untuk mewarisi properti dan method dari class lain, meningkatkan reusability dan organisasi kode. Kita juga melihat bagaimana menggunakan keyword super() untuk mengakses constructor dan method dari superclass. Pemahaman konsep inheritance sangat penting untuk membuat aplikasi yang terstruktur dan terorganisir dengan baik.