Inheritance

Pada dasarnya setiap objek yang berbeda sering memiliki kesamaan atau kemiripan tertentu. Kita ambil contoh kelas hewan. Misalnya, kucing dan anjing memiliki kemiripan sebagai hewan. Kesamaan yang dimiliki kucing dan anjing misalnya sama-sama hewan mamalia. Namun suara atau cara komunikasi kucing dan anjing berbeda. Begitu juga dengan hewan lain, misalnya harimau dan domba keduanya juga hewan mamalia dengan suara serta komunikasi yang berbeda.

Dari contoh di atas bisa dipahami kalau kucing, anjing, harimau, dan domba adalah hewan. Maka dalam bahasa Java kucing, anjing, harimau, dan domba adalah turunan (*inheritance*) dari hewan. Pada pemrograman berorientasi objek atau OOP, konsep *inheritance* menjadi salah satu topik yang penting. Suatu objek diwariskan dengan menggunakan *keyword* `extends`. Pada pemrograman Java setiap objek pada kenyataannya adalah turunan dari class `Object`. Walaupun tidak secara eksplisit, ini bisa dibuktikan dengan operator `instanceof`.

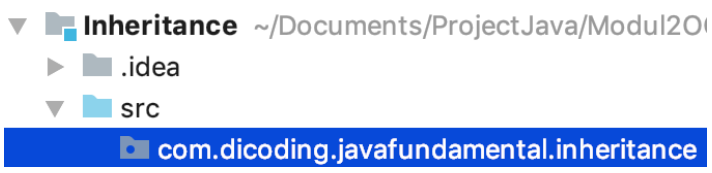
Beberapa terminologi yang penting:

- Super Class atau Parent Class, kelas yang semua fiturnya di wariskan kepada kelas turunannya.
- Sub Class atau Child Class, kelas turunan yang mewarisi semua fitur dari kelas lain. Sub class dapat menambah field dan metodenya sendiri sebagai tambahan dari kelas yang memberi warisan.
- Reusability, yaitu ketika kita ingin membuat kelas baru dan sudah ada kelas yang berisi kode yang kita inginkan, kita bisa menurunkan kelas baru tersebut dari kelas yang sudah ada. Dengan begitu, kita menggunakan kembali field dan metode dari kelas yang telah ada.

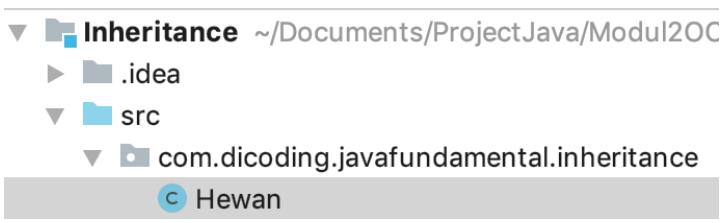
Codelab Inheritance

Untuk lebih detailnya kita coba langsung praktikan pada kode program berikut ini.

1. Buatlah proyek baru dengan nama Inheritance dengan nama package `com.dicoding.javafundamental.inheritance` di dalamnya:



2. Buatlah kelas baru dengan nama `Hewan` di dalam package tersebut.



3. Tambahkan kode berikut di dalam kelas `Hewan`:

```java
package com.dicoding.javafundamental.inheritance;

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

4. Buatlah kelas `Kucing`, *inherit* dari class `Hewan` menggunakan `extends`.

```java
package com.dicoding.javafundamental.inheritance;

public class Kucing extends Hewan {
    public Kucing() {
        super(); // akan tetap memanggil constructor dari parent Class
        System.out.println("construct Kucing");
    }
}
```

5. Selanjutnya buatlah sebuah class baru dengan nama `Main`, lalu tambahkan kode berikut:

```java
package com.dicoding.javafundamental.inheritance;

public class Main {
    public static void main(String[] args) {
        Hewan hewan = new Hewan(); // memanggil constructor
        System.out.println("Apakah hewan IS-A Objek -> " + (hewan instanceof Object));
        System.out.println("Apakah hewan IS-A Hewan -> " + (hewan instanceof Hewan));
        System.out.println("Apakah hewan IS-A Kucing -> " + (hewan instanceof Kucing));

        System.out.println("------------------------");

        Kucing kucing = new Kucing(); // memanggil constructor
        System.out.println("Apakah hewan IS-A Objek -> " + (kucing instanceof Object));
        System.out.println("Apakah kucing IS-A Hewan -> " + (kucing instanceof Hewan));
        System.out.println("Apakah kucing IS-A Kucing -> " + (kucing instanceof Kucing));
```

6. Selanjutnya jalankanlah kode di atas pada IDE yang kalian gunakan. Bila sukses, seharusnya Console akan menampilkan output seperti ini.

   Output-nya akan seperti di bawah ini.

```
construct Hewan
Apakah hewan IS-A Objek -> true
Apakah hewan IS-A Hewan -> true
Apakah hewan IS-A Kucing -> false
----------------------
construct Hewan
construct Kucing
Apakah hewan IS-A Objek -> true
Apakah kucing IS-A Hewan -> true
Apakah kucing IS-A Kucing -> true
```

Dari output di atas bisa kita simpulkan bahwa seekor kucing adalah (IS-A) hewan . Tapi seekor hewan belum tentu kucing dan pastinya hewan adalah instans dari kelas `Hewan` dan `kucing` adalah instans dari kelas Kucing yang keduanya adalah (IS-A) Object.

7. Perhatikan output berikut:

```
...
construct Hewan
construct Kucing
...
```

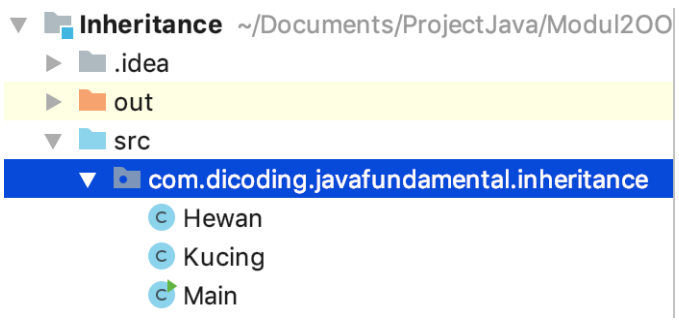
Ketika membuat instans objek `kucing`, constructor dari class `Hewan` akan tetap dipanggil walaupun sengaja kita komentari baris `super()`. Coba jadikan komentar di baris tersebut dan jalankan ulang class `Main` maka hasilnya akan sama saja.

Overriding dan Overloading

Setiap kelas yang `extends` kelas lain akan mewarisi semua field dan metode yang ada di parent class tersebut.

Codelab Overriding dan Overloading

1. Bukalah kembali proyek Inheritance.



2. Masukkan kode berikut ke dalam kelas `Hewan`:

```java
package com.dicoding.javafundamental.inheritance;

public class Hewan {
    public Hewan() {
        System.out.println("construct Hewan");
    }

    public void makan() {
        System.out.println("Hewan sedang makan..");
    }
}
```

3. Tambahkan kode berikut di kelas `Main`:

```java
package com.dicoding.javafundamental.inheritance;

public class Main {
    public static void main(String[] args) {
        Hewan hewan = new Hewan(); // memanggil constructor
        System.out.println("Apakah hewan IS-A Objek -> " + (hewan instanceof Object));
        System.out.println("Apakah hewan IS-A Hewan -> " + (hewan instanceof Hewan));
        System.out.println("Apakah hewan IS-A Kucing -> " + (hewan instanceof Kucing));

        System.out.println("-----------------------");

        Kucing kucing = new Kucing(); // memanggil constructor
        System.out.println("Apakah hewan IS-A Objek -> " + (kucing instanceof Object));
        System.out.println("Apakah kucing IS-A Hewan -> " + (kucing instanceof Hewan));
        System.out.println("Apakah kucing IS-A Kucing -> " + (kucing instanceof Kucing));
```

4. Selanjutnya jalankanlah kode di atas pada IDE yang kalian gunakan. Bila sukses, seharusnya Console akan menampilkan output seperti ini.

```
construct Hewan
Apakah hewan IS-A Objek -> true
Apakah hewan IS-A Hewan -> true
Apakah hewan IS-A Kucing -> false
-----------------------
construct Hewan
construct Kucing
Apakah hewan IS-A Objek -> true
Apakah kucing IS-A Hewan -> true
Apakah kucing IS-A Kucing -> true
-----------------------
Hewan sedang makan..
```

5. Ada kalanya metode yang diwariskan perlu diubah sesuai kebutuhan spesifik dari sub-class tersebut. Perubahan dapat dilakukan dengan 2 cara yaitu overriding dan overloading. Overriding adalah pembuatan metode baru pada subclass yang sama persis dengan superclassnya, sedangkan overloading adalah pembuatan metode baru pada subclass yang sama dengan method superclass namun parameternya berbeda. Untuk lebih jelasnya, bukalah kelas `Hewan`.

```java
package com.dicoding.javafundamental.inheritance;

public class Hewan {
    public Hewan() {
        System.out.println("construct Hewan");
    }

    public void makan() {  // base method
        System.out.println("Hewan sedang makan..");
    }
}
```

Selanjutnya bukalah kelas `Kucing` dan tambahkan metode berikut:

```java
package com.dicoding.javafundamental.inheritance;

public class Kucing extends Hewan {
    public Kucing() {
        //super(); // akan tetap memanggil constructor dari parent Class
        System.out.println("construct Kucing");
    }

    public void makan() { // overriding
        System.out.println("Kucing sedang makan..");
    }

    public void makan(String food) { // overloading
        System.out.println("Kucing makan " + food);
    }
```

6. Selanjutnya bukalah kelas Main dan tambahkan kode berikut:

```java
package com.dicoding.javafundamental.inheritance;

public class Main {
    public static void main(String[] args) {
        Hewan hewan = new Hewan(); // memanggil constructor
        System.out.println("Apakah hewan IS-A Objek -> " + (hewan instanceof Object));
        System.out.println("Apakah hewan IS-A Hewan -> " + (hewan instanceof Hewan));
        System.out.println("Apakah hewan IS-A Kucing -> " + (hewan instanceof Kucing));

        System.out.println("-----------------------");

        Kucing kucing = new Kucing(); // memanggil constructor
        System.out.println("Apakah hewan IS-A Objek -> " + (kucing instanceof Object));
        System.out.println("Apakah kucing IS-A Hewan -> " + (kucing instanceof Hewan));
        System.out.println("Apakah kucing IS-A Kucing -> " + (kucing instanceof Kucing));
```

7. Maka jika dijalankan, akan jadi seperti ini:

```
construct Hewan
Apakah hewan IS-A Objek -> true
Apakah hewan IS-A Hewan -> true
Apakah hewan IS-A Kucing -> false
-----------------------
construct Hewan
construct Kucing
Apakah hewan IS-A Objek -> true
Apakah kucing IS-A Hewan -> true
Apakah kucing IS-A Kucing -> true
-----------------------
Hewan sedang makan..
Kucing sedang makan..
Kucing makan daging ikan
```

8. Seperti dibahas di awal, setiap objek pada kenyataannya adalah turunan dari class Object. Maka setiap objek akan otomatis mewarisi method dari class `Object`. Sebagian metode yang sering digunakan adalah metode `toString()` dan `equals()`. Metode `toString()` bisa kita gunakan untuk merepresentasikan objek dalam bentuk String yang sangat berguna salah satunya untuk *debugging*. Sedangkan metode `equals()` digunakan untuk membandingkan antara dua objek, apakah mereka sama. Metode `equals()` biasanya kita override karena kebutuhan khusus . Misalnya kita mau dua objek `Kucing` adalah sama jika rasnya sama, meski habitatnya beda. Bukalah kelas `Kucing` dan tambahkan kode di dalamnya menjadi seperti ini:

```java
package com.dicoding.javafundamental.inheritance;

public class Kucing extends Hewan {

    private String ras;
    private String habitat;

    public Kucing(String ras, String habitat) {
        this.ras = ras;
        this.habitat = habitat;
    }

    @Override
    public String toString() {
        return "Kucing ras: " + ras + ", habitat: " + habitat;
```

9. Tambahkan juga kode berikut di dalam kelas `Main`.

```java
package com.dicoding.javafundamental.inheritance;

public class Main {
    public static void main(String[] args) {
        Hewan hewan = new Hewan(); // memanggil constructor
        System.out.println("Apakah hewan IS-A Objek -> " + (hewan instanceof Object));
        System.out.println("Apakah hewan IS-A Hewan -> " + (hewan instanceof Hewan));
        System.out.println("Apakah hewan IS-A Kucing -> " + (hewan instanceof Kucing));

        System.out.println("------------------------");

        Kucing kucing = new Kucing(); // memanggil constructor
        System.out.println("Apakah hewan IS-A Objek -> " + (kucing instanceof Object));
        System.out.println("Apakah kucing IS-A Hewan -> " + (kucing instanceof Hewan));
        System.out.println("Apakah kucing IS-A Kucing -> " + (kucing instanceof Kucing));
```

10. Jika dijalankan, outputnya akan seperti di bawah ini:

```
construct Hewan
Apakah hewan IS-A Objek -> true
Apakah hewan IS-A Hewan -> true
Apakah hewan IS-A Kucing -> false
-----------------------
construct Hewan
construct Kucing
Apakah hewan IS-A Objek -> true
Apakah kucing IS-A Hewan -> true
Apakah kucing IS-A Kucing -> true
-----------------------
Hewan sedang makan..
Kucing sedang makan..
Kucing makan daging ikan
-----------------------
construct Hewan
construct Hewan
construct Hewan
Kucing ras: Ocicat, habitat: tropis
Kucing ras: Ocicat, habitat: subtropis
Kucing ras: Anggora, habitat: subtropis
meow equals puss ? true
meow equals popo ? false
```

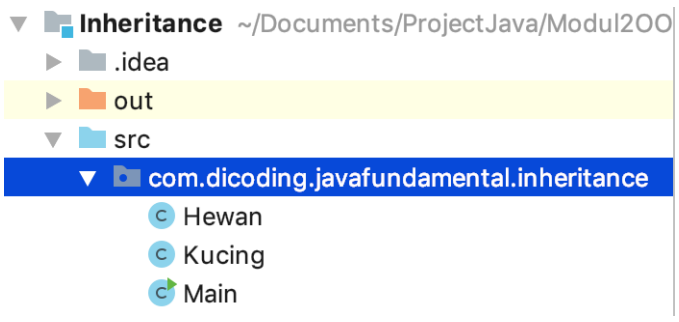
Polymorphism

Polymorphism artinya banyak bentuk. Contohnya, kucing adalah seekor Hewan, kucing juga adalah sebuah objek. Artinya kucing bisa dikenali dalam banyak bentuk. Lalu apa maknanya bagi pemrograman khususnya Java?

Codelab Polymorphism

Kita coba lihat di koding yuk! Kita akan pakai kode yang sebelumnya, agar biar lebih mudah dipahami.

1. Bukalah kembali proyek Inheritance.



2. Kita akan melihat kembali kelas Hewan,

```java
package com.dicoding.javafundamental.inheritance;

public class Hewan {
    public Hewan() {
        System.out.println("construct Hewan");
    }

    public void makan() {  // base method
        System.out.println("Hewan sedang makan..");
    }
}
```

dan Kucingberikut:

```java
package com.dicoding.javafundamental.inheritance;

public class Kucing extends Hewan {

    private String ras;
    private String habitat;

    public Kucing(String ras, String habitat) {
        this.ras = ras;
        this.habitat = habitat;
    }

    @Override
    public String toString() {
        return "Kucing ras: " + ras + ", habitat: " + habitat;
```

3. Selanjutnya bukalah kelas `Main` dan tambahkan kode berikut:

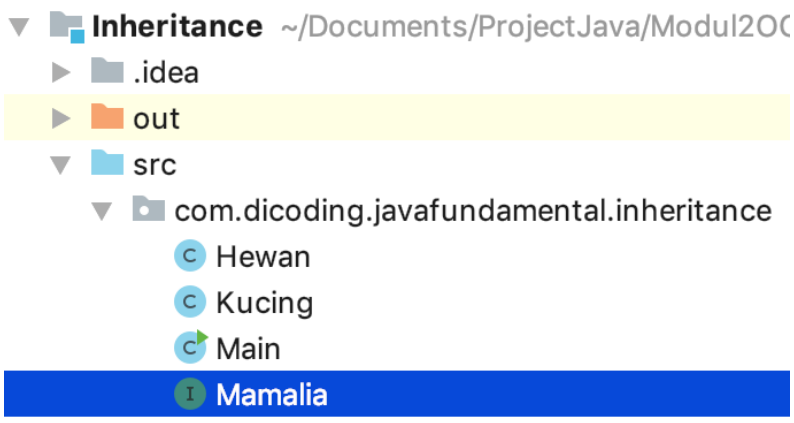
```java
package com.dicoding.javafundamental.inheritance;

public class Main {
    public static void main(String[] args) {
        Hewan hewan = new Hewan(); // memanggil constructor
        System.out.println("Apakah hewan IS-A Objek -> " + (hewan instanceof Object));
        System.out.println("Apakah hewan IS-A Hewan -> " + (hewan instanceof Hewan));
        System.out.println("Apakah hewan IS-A Kucing -> " + (hewan instanceof Kucing));

        System.out.println("------------------------");

        Kucing kucing = new Kucing(); // memanggil constructor
        System.out.println("Apakah hewan IS-A Objek -> " + (kucing instanceof Object));
        System.out.println("Apakah kucing IS-A Hewan -> " + (kucing instanceof Hewan));
        System.out.println("Apakah kucing IS-A Kucing -> " + (kucing instanceof Kucing));
```

4. Coba jalankan kode di atas lalu lihat hasilnya. Dari kode tersebut bisa dilihat *instance* yang lebih spesifik bisa langsung di-*assign* ke class yang lebih umum (misal *instance* kucing ke *instance* object atau hewan),. Tapi sebaliknya tidak berlaku.

```
construct Hewan
Apakah hewan IS-A Objek -> true
Apakah hewan IS-A Hewan -> true
Apakah hewan IS-A Kucing -> false
-----------------------
construct Hewan
construct Kucing
Apakah hewan IS-A Objek -> true
Apakah kucing IS-A Hewan -> true
Apakah kucing IS-A Kucing -> true
-----------------------
Hewan sedang makan..
Kucing sedang makan..
Kucing makan daging ikan
-----------------------
construct Hewan
construct Hewan
construct Hewan
Kucing ras: Ocicat, habitat: tropis
Kucing ras: Ocicat, habitat: subtropis
Kucing ras: Anggora, habitat: subtropis
meow equals puss ? true
meow equals popo ? false
-----------------------
construct Hewan
construct Kucing
construct Hewan
construct Kucing
construct Hewan
construct Kucing
Kucing sedang makan..
Kucing sedang makan..
Kucing sedang makan..
Kucing sedang makan..
construct Hewan
Exception in thread "main" java.lang.ClassCastException: class
com.dicoding.javafundamental.inheritance.Hewan cannot be cast to class
com.dicoding.javafundamental.inheritance.Kucing
(com.dicoding.javafundamental.inheritance.Hewan and
com.dicoding.javafundamental.inheritance.Kucing are in unnamed module of loader 'app')
at com.dicoding.javafundamental.inheritance.Main.main(Main.java:56)
```

Perlu dilakukan casting untuk mengubah instance hewan menjadi instance kucing. Casting sendiri adalah mengubah suatu tipe data menjadi tipe data yang lain. Pada inheritance sendiri, casting dapat dilakukan terhadap kelas yang mempunyai hubungan *parent-child*. Analoginya kucing adalah pasti hewan tetapi belum tentu hewan adalah kucing. Bisa jadi harimau atau elang atau yang lainnya. bukan? Sehingga jika casting gagal akan muncul sebagai `ClassCastException`.

5. Interface juga bisa diterapkan dalam penggunaan polymorphism ini. Buatlah interface dengan nama `Mamalia`.



6. Lalu kita pasang interface tersebut ke class `Kucing`.

```
1.  package com.dicoding.javafundamental.inheritance;
2.
3.  public class Kucing extends Hewan implements Mamalia{
4.      ...
5.  }
```

7. Sehingga kita bisa mendeklarasikan instance kucing menggunakan interface `Mamalia`.

```
1.  Mamlia mamalia = new Kucing();
```

Pada praktiknya banyak kode java menggunakan cara deklarasi di atas yaitu menggunakan interface. Coba ingat kembali materi Code to Interface di materi Interface.

Perbedaan Antara Inheritance Dengan Interface

Sekilas penggunaan Inheritance dan interface mirip-mirip. Keduanya sama-sama akan menurunkan field/method. Lalu apa bedanya antara Inheritance dan interface? Hal ini sering ditanyakan dalam *interview* programmer Java :) Jawaban yang sering dilontarkan adalah “interface menggunakan Implements sedangkan Inheritance menggunakan extends.”

Sebenarnya, jawaban tersebut tidak cocok. Alih-alih, jawaban tersebut cocok untuk pertanyaan bagaimana cara menggunakan interface dan menggunakan Inheritance.

Konsep inheritance digunakan untuk abstraksi dari yang paling umum ke yang lebih spesifik. Misalnya class `Hewan` adalah bentuk yang umum, lalu class `Kucing` adalah turunannya yang lebih spesifik.

Sedangkan interface digunakan sebagai kontrak atau aturan. Class yang menerapkan suatu interface wajib override semua method dari interface tersebut. Artinya class tersebut harus mengikuti aturan atau spesifikasi yang ada di interface.

Sampai sini, Anda sudah paham perbedaan Inheritance dan Interface? Ingat saja kata kuncinya inheritance adalah abstraksi sedangkan interface adalah kontrak. Happy coding :)

Sebelumnya

Selanjutnya

dicoding

Dicoding Space
Jl. Batik Kumeli No.50, Sukaluyu,
Kec. Cibeunying Kaler, Kota Bandung
Jawa Barat 40123

Penghargaan

Decode Ideas
Discover Potential

Blog
Reward
Showcase
Tentang Kami
Hubungi Kami
FAQ