

Modul Pemrograman Berbasis Objek (PBO)

Generic Dan Annotation

Pada modul sebelum nya, kita dapat menentukan tipe data pada arraylist, linkedlist, dan lain sebagainya. Hal ini dimungkinkan dengan suatu hal bernama generic.

Generic di java memungkinkan kita membuat suatu class, lalu user dapat menentukan tipe data dari class tersebut. Selain class, kita juga dapat membuat generic pada method dan user dapat menentukan tipe data dari method tersebut.

Mari kita coba langsung pada kodingan. Buatlah folder baru bernama generic, dan buat file bernama Data.java lalu masukan kode dibawah ini pada file tersebut

```
class Data <T> {  
    T dataPertama;  
    T dataKedua;  
  
    public Data (T dataPertama, T dataKedua) {  
        this.dataPertama = dataPertama;  
        this.dataKedua = dataKedua;  
    }  
  
    public void showData () {  
        System.out.println("Data pertama adalah " + dataPertama);  
        System.out.println("Data kedua adalah " + dataKedua);  
    }  
}
```

Dapat kita lihat pada kode diatas, terdapat hal asing yang baru pertama kali kita lihat yaitu <T>

```
class Data <T> {  
    T dataPertama;  
    T dataKedua;
```

<T> menandakan bahwa kita membuat suatu generic class. Nantinya ketika kita membuat suatu objek maka kita dapat menentukan tipe data dari class tersebut. Variabel yang menggunakan tipe data generic dari class tersebut dapat menuliskan huruf sesuai yang digunakan pada generic class.

Huruf <T> pada generic class dapat diganti sesuai yang diinginkan, jadi tidak terbatas hanya T saja. Yang terpenting, jika nantinya variabel juga menggunakan generic dari class tersebut, tipe data nya harus sesuai dengan generic yang diberikan

Contoh

<T> -> Maka variabel nya jadi -> T dataPertama

<E> -> Maka variabel nya jadi -> E dataPertama

Mari kita buat class Main nya, untuk menjalankan kode diatas. Masih pada folder yang sama, buat lah file baru bernama Main, dan masukan kode dibawah ini

```
class Main {  
  
    public static void main(String args []) {  
        Data <Integer> dataInteger = new Data <Integer> (8, 20);  
        Data <String> dataString = new Data <String> ("Nurul", "Fikri");  
  
        dataInteger.showData();  
        dataString.showData();  
    }  
}
```

Dapat kita lihat diatas, kita membuat 2 objek dari class data dengan tipe data yang berbeda. Objek dataInteger menggunakan tipe data <Integer> , sedangkan objek dataString menggunakan tipe data <String>

Mari kita jalan kan kode yang telah kita buat diterminal

```
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Generic$ javac Main.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Generic$ java Main
Data pertama adalah 8
Data kedua adalah 20
Data pertama adalah Nurul
Data kedua adalah Fikri
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Generic$
```

Dapat kita lihat, hasil run berjalan dengan baik dan menampilkan tulisan sesuai dengan yang kita tuliskan di kode.

Pada kode diatas, kita hanya memberikan 1 generic pada class. Kita bisa membuat lebih banyak generic pada class dan tidak terbatas hanya 1 generic saja. Selain memberikan generic pada class, kita juga dapat memberikan generic pada method. Mari kita lihat langsung implementasi pada kodingan

Masih di folder generic, buatlah file baru bernama ListData.java, dan masukan kode dibawah ini pada file tersebut

```
class ListData <K, V> {
    K kunci;
    V nilai;

    public ListData (K kunci, V nilai) {
        this.kunci = kunci;
        this.nilai = nilai;
    }

    public void ListDataLama() {
        System.out.println("Kunci lama adalah = " + this.kunci);
        System.out.println("Nilai lama adalah = " + this.nilai);
    }

    public <K, V> void ListDataBaru (K kunci, V nilai) {
        System.out.println("Kunci baru adalah = " + kunci);
        System.out.println("Nilai baru adalah = " + nilai);
    }
}
```

Pada kode diatas kita telah membuat 2 generic yang dapat digunakan pada class. Untuk membuat lebih dari 1 generic tinggal menambahkan koma, lalu masukan generic lain nya

< K, V > → 2 generic

< K, V, E > → 3 Generic

Selain generic class, kita juga membuat generic method

```
public <K, V> void ListDataBaru (K kunci, V nilai) {  
    System.out.println("Kunci baru adalah = " + kunci);  
    System.out.println("Nilai baru adalah = " + nilai);  
}
```

Sama seperti class, kita tinggal menambahkan tipe data generic yang akan kita gunakan pada method.

Mari kita lihat langsung implementasi dari kode diatas, buka kembali file Main.java, lalu tambahkan kode dibawah ini

```
Jan 9 > Generic > Main.java  
class Main {  
    public static void main(String args []) {  
        Data <Integer> dataInteger = new Data <Integer> (8, 20);  
        Data <String> dataString = new Data <String> ("Nurul", "Fikri");  
  
        dataInteger.showData();  
        dataString.showData();  
  
        ListData <String, Integer> ListDataInteger = new ListData <String, Integer> ("Umur", 15);  
  
        ListDataInteger.ListDataLama();  
        ListDataInteger.ListDataBaru("Nama", "Fikri");  
  
        ListData <String, Boolean> ListDataString = new ListData <String, Boolean> ("Nilai", true);  
        ListDataString.ListDataLama();  
        ListDataString.ListDataBaru("Nilai", 89);  
    }  
}
```

Pada kode diatas, kita telah membuat 2 objek yaitu ListDataInteger dan ListDataString. ListDataInteger memiliki tipe data string dan integer sedangkan ListDataString memiliki tipe data string dan boolean

Nah, coba kita lihat pada method ListDataBaru yang menggunakan generic

```
ListDataInteger.ListDataBaru("Nama", "Fikri");
```

```
ListDataString.ListDataBaru("Nilai", 89);
```

Dapat kita lihat diatas, kita tidak perlu menambahkan tipe data yang ingin digunakan ketika memanggil method. Kita cukup memasukan parameter nya saja, nanti secara otomatis generic akan menentukan tipe data parameter yang kita berikan

“nama”, “fikri” -> string, string

“Nilai”, 89 -> string, integer

Mari kita coba jalankan kode diatas pada terminal

```
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Generic$ javac Main.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Generic$ java Main
Data pertama adalah 8
Data kedua adalah 20
Data pertama adalah Nurul
Data kedua adalah Fikri
Kunci lama adalah = Umur
Nilai lama adalah = 15
Kunci baru adalah = Nama
Nilai baru adalah = Fikri
Kunci lama adalah = Nilai
Nilai lama adalah = true
Kunci baru adalah = Nilai
Nilai baru adalah = 89
```

Kode telah berjalan dengan baik

Notes : Ketika membuat generic pada class Data, kita menggunakan <T>. Tapi, di class ListData kita menggunakan <K> dan <V>. Nah dari kode diatas, kita dapat membuat generic sesuai yang kita inginkan. Kita dapat membuat <A> <C> dan lain sebagai nya sebagai generic.

Ada semacam kebiasaan yang umum dikalangan komunitas, bukan hal yang wajib kalian ikuti, tapi bisa menjadi semacam panduan ketika ingin membuat generic , yaitu

<E> mengindikasikan generic berupa element

<K> mengindikasikan generic berupa Key value

<T> mengindikasikan generic berupa tipe data
<V> mengindikasikan generic berupa value
<S>, <U>, <V>, dll bebas untuk menentukan tipe apa saja

Penulisan generic diatas, hanya panduan umum saja dikalangan komunitas bukan hal yang wajib diikuti.

Berikut referensi dari generic :

<https://www.geeksforgeeks.org/generics-in-java>

Kita telah membahas generic, mari kita beralih ke topik berikutnya yaitu annotation

Untuk menjelaskan annotation, kita akan menggunakan kodingan yang telah kita buat pada modul modul sebelum nya.

Annotation adalah meta data yang digunakan untuk menginformasikan kepada compiler bagaimana ia menangani suatu kode.

Contoh nya pada praktikum inheritance dan interface kita telah menggunakan annotation yaitu @override, mari kita lihat kodingan yang menggunakan override

Kita buka class BangunDatar pada praktikum inheritance

```
class BangunDatar {  
    int panjang;  
    int lebar;  
  
    public BangunDatar(int panjang, int lebar) {  
        this.panjang = panjang;  
        this.lebar = lebar;  
    }  
    double luas() {  
        int nilai = panjang*lebar;  
        return nilai;  
    }  
}
```

```
double keliling() {  
    int nilai = 2*(panjang+lebar);  
    return nilai;  
}  
}
```

Lalu, kita buka class Persegi

```
class Persegi extends BangunDatar {  
    double sisi;  
  
    public Persegi(double sisi, int panjang, int lebar) {  
        super(panjang, lebar);  
        this.sisi = sisi;  
    }  
  
    @Override  
    double luas() {  
        double nilai = this.sisi * this.sisi;  
        return nilai;  
    }  
  
    @Override  
    double keliling() {  
        double nilai = 4 * sisi;  
  
        return nilai;  
    }  
  
    double luasOriginal() {  
        double nilai = super.luas();  
  
        return nilai;  
    }  
}
```

Kita dapat melihat `@Override` pada method luas dan keliling. Annotation `@Override` kita gunakan untuk menginformasikan kepada compiler bahwa method luas dan keliling pada class Persegi digunakan untuk menimpa method luas dan keliling yang diwarisi oleh BangunDatar.

Annotation `@Override` tidak berpengaruh langsung pada kode yang dibuat, tetapi berpengaruh untuk menginformasikan kepada compiler bahwa kodingan pada class child yang kita buat akan menimpa kodingan dari class parent.

Selain `@Override`, terdapat annotation lain yaitu `@Deprecated`. Annotation `@Deprecated` digunakan untuk memberi tahu user bahwa kodingan yang kita buat akan diganti dengan versi baru dimasa yang akan datang. Buka class `PersegiPanjang` dan tambahkan annotation `@Deprecated` pada salah satu method nya

```
class PersegiPanjang extends BangunDatar {
    double panjang = 10;
    double lebar = 20;

    public PersegiPanjang(int panjang, int lebar) {
        super(panjang, lebar);
    }

    @Deprecated
    double luasBangunDatar() {
        return super.luas();
    }

    double luasPersegiPanjang() {
        double nilai = this.panjang * this.lebar;

        return nilai;
    }
}
```

Method `luasBangunDatar` memiliki annotation `@Deprecated`. Hal ini akan memberi tahu user bahwa method yang kita buat akan segera kita ganti dengan versi baru dimasa yang akan datang

Sekarang, mari kita buka class `Main` pada kodingan diatas, dan kita coba jalankan


```

an 9 > Annotation > OverrideDeprecated > Main.java
class Main {
    public static void main(String args[]) {
        Persegi persegi = new Persegi(5, 10, 2);
        double luasPersegi = persegi.luas();
        double luasOriginal = persegi.luasOriginal();
        double kelilingPersegi = persegi.keliling();

        System.out.println("Luas Persegi adalah = " + luasPersegi);
        System.out.println("Luas Original Bangun datar pada class Persegi adalah = " + luasOriginal);
        System.out.println("Keliling persegi adalah = " + kelilingPersegi);

        PersegiPanjang persegiPanjang = new PersegiPanjang(10, 5);
        double luasBangunDatar = persegiPanjang.luasBangunDatar();
        double luasPersegiPanjang = persegiPanjang.luasPersegiPanjang();

        System.out.println("Luas Bangun Datar adalah = " + luasBangunDatar);
        System.out.println("Luas Persegi Panjang adalah = " + luasPersegiPanjang);
    }
}

```

```

ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Annotation/Override
Deprecated$ javac Main.java
Note: Main.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Annotation/Override
Deprecated$ java Main
Luas Persegi adalah = 25.0
Luas Original Bangun datar pada class Persegi adalah = 20.0
Keliling persegi adalah = 20.0
Luas Bangun Datar adalah = 50.0
Luas Persegi Panjang adalah = 200.0
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Annotation/Override
Deprecated$

```

Dapat kita lihat pada kodingan diatas, terdapat warning

“Main.java uses or overrides a deprecated API.”

Kalimat ini muncul karena terdapat annotation `@Deprecated` pada method `luasBangunDatar()`. Compiler memberi tahu bahwa class `Main.java` menggunakan method yang akan segera diperbarui dan dalam contoh diatas yang akan diperbarui adalah method `luasBangunDatar`.

Nah sekarang, bagaimana semisal kita tidak ingin menampilkan warning deprecated pada compiler ? Kita dapat menggunakan annotation `@SuppressWarnings`

```

Jan 9 > Annotation > OverrideDeprecated > Main.java
class Main {
    @SuppressWarnings("deprecation")
    public static void main(String args[]) {
        Persegi persegi = new Persegi(5, 10, 2);
        double luasPersegi = persegi.luas();
        double luasOriginal = persegi.luasOriginal();
        double kelilingPersegi = persegi.keliling();

        System.out.println("Luas Persegi adalah = " + luasPersegi);
        System.out.println("Luas Original Bangun datar pada class Persegi adalah = " + luasOriginal);
        System.out.println("Keliling persegi adalah = " + kelilingPersegi);

        PersegiPanjang persegiPanjang = new PersegiPanjang(10, 5);
        double luasBangunDatar = persegiPanjang.luasBangunDatar();
        double luasPersegiPanjang = persegiPanjang.luasPersegiPanjang();

        System.out.println("Luas Bangun Datar adalah = " + luasBangunDatar);
        System.out.println("Luas Persegi Panjang adalah = " + luasPersegiPanjang);
    }
}

```

```

ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Annotation/Override
Deprecated$ javac Main.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Annotation/Override
Deprecated$ java Main
Luas Persegi adalah = 25.0
Luas Original Bangun datar pada class Persegi adalah = 20.0
Keliling persegi adalah = 20.0
Luas Bangun Datar adalah = 50.0
Luas Persegi Panjang adalah = 200.0
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Annotation/Override
Deprecated$

```

Dengan menambahkan method `@SuppressWarnings("deprecation")`, kita menghilangkan peringatan deprecated pada terminal. Parameter "deprecation" pada annotation `@SuppressWarnings` menandakan bahwa kita ingin menghilangkan peringatan deprecated.

Selain deprecation, terdapat 1 parameter lagi pada annotation `@SuppressWarnings` yaitu `unchecked`.

Mari kita lihat penerapan langsung, pada generic yang telah kita buat

```

class Main {

    public static void main(String args []) {

        Data <Integer> dataInteger = new Data <Integer> (8, 20);
        Data <String> dataString = new Data <String> ("Nurul", "Fikri");

        dataInteger.showData();
        dataString.showData();

    }
}

```

```

ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Annotation/Override
Deprecated$ cd ../../Generic/
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Generic$ javac Main
.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Generic$ java Main
Data pertama adalah 8
Data kedua adalah 20
Data pertama adalah Nurul
Data kedua adalah Fikri
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Generic$

```

Tidak terdapat peringatan apa apa, tetapi bagaimana semisal tipe data pada objek dataInteger kita hapus

```

ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Generic$ javac Main
.java
Note: Main.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Generic$ java Main
Data pertama adalah 8
Data kedua adalah 20
Data pertama adalah Nurul
Data kedua adalah Fikri
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Generic$

```

Terdapat peringatan “Note: Main.java uses unchecked or unsafe operations”.

Peringatan ini muncul karena kita tidak memberikan tipe data pada objek dataInteger. Secara default jika kita tidak memberikan tipe data pada variabel objek yang mengimplemetasikan generic class, maka tipe data nya sendiri akan menjadi objek. Hal ini sama seperti arrayList, linkedlist, dan lain sebagainya yang telah kita pelajari di modul sebelum nya.

Untuk menghilangkan warning tersebut, kita dapat menambahkan tipe data pada objekDataInteger seperti pada contoh pertama. Atau, kita dapat menggunakan @SuppressWarnings(“unchecked”)

```

class Main {
    @SuppressWarnings("unchecked")
    public static void main(String args []) {
        Data dataInteger = new Data (8, 20);
        Data <String> dataString = new Data <String> ("Nurul", "Fikri");

        dataInteger.showData();
        dataString.showData();
    }
}

```

```

ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Generic$ javac Main
.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Generic$ java Main
Data pertama adalah 8
Data kedua adalah 20
Data pertama adalah Nurul
Data kedua adalah Fikri
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Generic$

```

Yap, kali ini tidak terdapat warning pada terminal

Nah, selain menggunakan annotation yang sudah ada kita dapat membuat annotation versi kita sendiri

Mari kita gunakan kodingan praktikum interface.

Buka interface Hewan

```

interface Hewan {
    public String suaraHewan();
    public String jenisMakanan();
}

```

Lalu buka, class Kucing

```

Jan 9 > Annotation > Interface > Kucing.java
@interface test{
    int angka();
}

@interface TestAnnotation
{
    String nama() default "Nurul";
    int nilai();
} // will be retained at runtime

public class Kucing implements Hewan {

    @test(angka=90)
    @Override
    public String suaraHewan() {
        return "meaawww";
    }

    @TestAnnotation(nama="Fikri", nilai=90)
    @Override
    public String jenisMakanan() {
        return "Daging";
    }
}

```

Untuk membuat annotation versi kita sendiri dapat dengan cara

`@interface namaannotaton {}`

Pada kodingan diatas, kita membuat annotation `@test` dan `@testAnnotation`

```
@interface test{
    int angka();
}

@interface TestAnnotation
{
    String nama() default "Nurul";
    int nilai();
} // will be retained at runtime
```

Pada annotation `@test` dan `@TestAnnotation` kita membuat method dengan tipe data sesuai dengan method tersebut

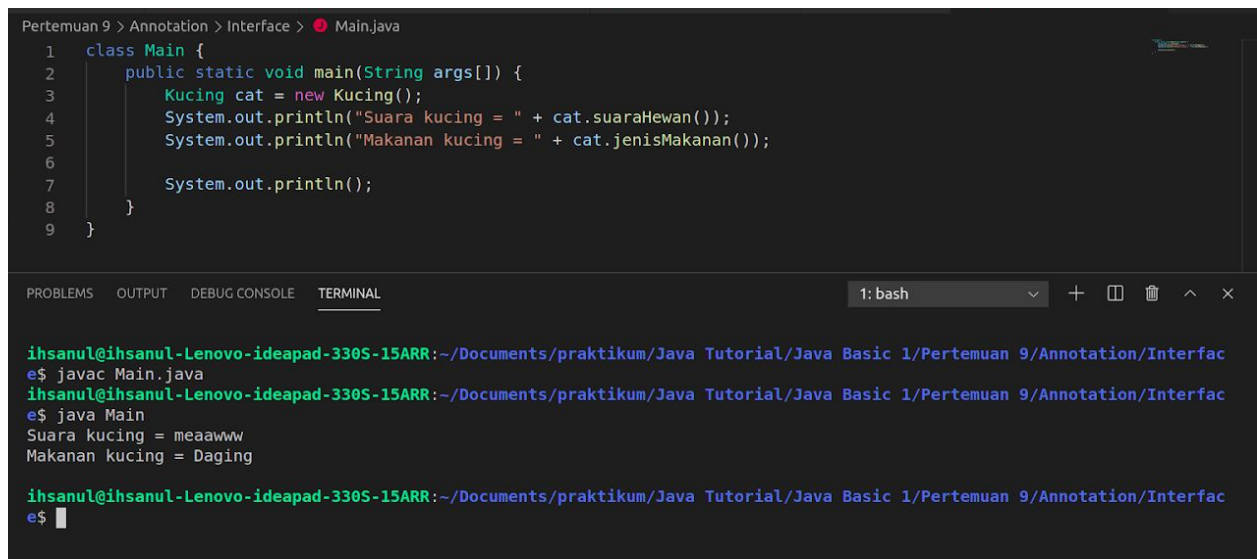
Lalu, kita mencoba mengimplementasikan annotation yang kita buat ke dalam method pada class Kucing

```
@test(angka=90)
@Override
public String suaraHewan() {
    return "meaawww";
}

@TestAnnotation(nama="Fikri", nilai=90)
@Override
public String jenisMakanan() {
    return "Daging";
}
```

Isian parameter annotation `@test` sesuai dengan tipe data yang kita berikan yaitu integer, sedangkan pada annotation `@TestAnnotation` kita memberikan string "Fikri" dan integer 90.

Kita telah membuat annotation versi kita sendiri, mari kita coba jalankan kode diatas



The screenshot shows an IDE window with a Java file named `Main.java`. The code defines a `Main` class with a `main` method that creates a `Kucing` object and prints its `suaraHewan()` and `jenisMakanan()` methods. Below the code editor, the `TERMINAL` tab is active, showing the command `javac Main.java` and the output of `java Main`, which prints `Suara kucing = meaawww` and `Makanan kucing = Daging`.

```
Pertemuan 9 > Annotation > Interface > Main.java
1 class Main {
2     public static void main(String args[]) {
3         Kucing cat = new Kucing();
4         System.out.println("Suara kucing = " + cat.suaraHewan());
5         System.out.println("Makanan kucing = " + cat.jenisMakanan());
6
7         System.out.println();
8     }
9 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash

```
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Annotation/Interfac
e$ javac Main.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Annotation/Interfac
e$ java Main
Suara kucing = meaawww
Makanan kucing = Daging
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 9/Annotation/Interfac
e$
```

Program telah berjalan dengan baik. Mungkin teman teman bertanya, kok ga ada efek apa apa pada terminal? Karena pada annotation yang kita buat tidak terdapat perintah untuk melakukan sesuatu, hanya meta data saja yang kita berikan pada compiler

Annotation sendiri terbagi menjadi 3 tipe

1) Marker Annotation = yaitu annotation tanpa value, contoh nya `@Override`

2) Single-Value Annotation = yaitu annotation dengan 1 value, contohnya

```
@test(angka=90)
```

3) Multi-Value Annotation = yaitu annotation yang memiliki value lebih dari 1, contohnya

```
@TestAnnotation(nama="Fikri", nilai=90)
```

Kita telah membuat annotation versi kita sendiri dan kita juga mengetahui tipe tipe annotation. Nah sekarang kita akan mempelajari mengenai annotation yang digunakan untuk annotation

Kita coba lihat langsung saja pada kodingan, buka kembali kodingan Kucing.java, lalu tambahkan annotation dibawah ini

```
import java.lang.annotation.ElementType;
import java.lang.annotation.Target;

@interface test{
    int angka();
}

@Target(ElementType.METHOD)
@interface TestAnnotation
{
    String nama() default "Nurul";
    int nilai();
} // will be retained at runtime

public class Kucing implements Hewan {

    @test(angka=90)
    @Override
    public String suaraHewan() {
        return "meaawww";
    }

    @TestAnnotation(nama="Fikri", nilai=90)
    @Override
    public String jenisMakanan() {
        return "Daging";
    }
}
```

Pada kodingan diatas, kita mengimport annotation @Target dan ElementType dari library java

Mari kita lihat pada method yang menggunakan annotation @Target

```
@Target(ElementType.METHOD)
@interface TestAnnotation
{
    String nama() default "Nurul";
    int nilai();
} // will be retained at runtime
```

Annotation @Target kita gunakan untuk memberitahu bahwa annotation yang kita buat hanya berfungsi pada element tertentu. Pada contoh diatas, annotation @Target memberitahukan bahwa annotation @TestAnnotation hanya dapat digunakan pada method. Jika annotation @TestAnnotation tidak digunakan pada method, maka akan error

```
@TestAnnotation(nama="Fikri", nilai=90)
@Override
public String jenisMakanan() {
    return "Daging";
}
```

Dapat kita lihat kode diatas, bahwa kita menggunakan @TestAnnotation pada method jenisMakanan

Bagaimana, compiler tahu bahwa @TestAnnotation kita gunakan untuk method ? Karena kita menggunakan annotation @Target, lalu didalam @Target kita menambahkan tipe element yang akan kita gunakan.

```
@Target(ElementType.METHOD)
```

Pada kasus diatas kita menggunakan ElementType.METHOD sebagai tipe element yang digunakan. Nah tipe element itulah yang akan memberitahukan kepada compiler bahwa annotation yang kita buat hanya boleh digunakan pada method.

Selain ElementType.METHOD masih terdapat tipe element yang lain, yaitu

TYPE	Dapat digunakan untuk class, interface, dan enumeration
FIELD	Dapat digunakan untuk fields
METHOD	Dapat digunakan untuk methods
CONSTRUCTOR	Dapat digunakan untuk constructors
LOCAL_VARIABLE	Dapat digunakan untuk local variables
ANNOTATION_TYPE	Dapat digunakan untuk annotation type
PARAMETER	Dapat digunakan untuk parameter

Nah, semisal kita ingin menggunakan type, maka kita tinggal menulis

`@Target(ElementType.TYPE)`

atau semisal kita ingin menggunakan field, maka kita tinggal menulis

`@Target(ElementType.FIELD)`

Dan lain sebagainya

Notes : untuk menggunakan annotation `@Target` dan tipe element, jangan lupa mengimport library yang diperlukan yaitu

```
import java.lang.annotation.ElementType;
import java.lang.annotation.Target;
```

Selain annotation dalam annotation, kita juga dapat membuat annotation level pada annotation. Level yang dimaksud yaitu

RetentionPolicy.SOURCE	Level ini menandakan bahwa annotation hanya berlaku pada source code. Annotation ini akan diabaikan ketika kode dicompile. Annotation ini akan dihilangkan pada hasil kompiler
RetentionPolicy.CLASS	Level ini menandakan bahwa annotation berlaku sampai kode di compile, jadi kode hasil compile nya akan mengandung annotation yang dibuat
RetentionPolicy.RUNTIME	Level ini menandakan bahwa annotation berlaku sampai hasil compile dijalankan

Untuk menggunakan annotation diatas, kita import terlebih dahulu dari library dijava

```
import java.lang.annotation.RetentionPolicy;
```

Lalu, kita dapat langsung menggunakan pada kodingan. Masih di kodingan Kucing.java tambahkan annotation berikut pada kodingan tersebut

```
import java.lang.annotation.ElementType;
import java.lang.annotation.Target;
import java.lang.annotation.Annotation;
import java.lang.annotation.RetentionPolicy;

@interface test{
    int angka();
}

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
@interface TestAnnotation
{
    String nama() default "Nurul";
}
```

```

    int nilai();
} // will be retained at runtime

public class Kucing implements Hewan {
    @test(angka=90)
    @Override
    public String suaraHewan() {
        return "meaawww";
    }

    @TestAnnotation(nama="Fikri", nilai=90)
    @Override
    public String jenisMakanan() {
        return "Daging";
    }
}

```

Kita telah membahas level pada annotation `@TestAnnotation`

```

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
@interface TestAnnotation
{
    String nama() default "Nurul";
    int nilai();
}

```

annotation `@TestAnnotation` memiliki level `RUNTIME`, artinya `@TestAnnotation` digunakan sampai hasil compile dari kode dijalankan

Berikut referensi annotation

<https://www.javatpoint.com/java-annotation>

<https://www.geeksforgeeks.org/annotations-in-java/>

Tugas

1. Buatlah laporan terkait praktikum modul Generic Dan Annotation
2. Screenshot kode dan hasil kode yang dijalankan dari laporan praktikum ini. Masukkan hasil screenshot tersebut dan jelaskan kode nya

Tugas dikirim pada elen, dengan format sebagai berikut

1. Pada penamaan file gunakan format berikut

Praktikum_keberapa_Judul Praktikum_Nama_Nim

Contoh :

Praktikum01_Pengenalan-Java_IhsanulFikriAbiyyu_0220318021

2. Format laporan bertipe pdf

3. Pada cover praktikum, masukan beberapa hal dibawah ini- Praktikum ke berapa

- Judul Praktikum

- Nama

- Nim

4. Tenggat waktu mengerjakan adalah 1 pekan semenjak tugas diberikan

5. Diperbolehkan berkelompok tetapi hanya antara yang tidak punya laptop/komputer dengan yang punya laptop/komputer. Selain itu tidak boleh