PRAKTIKUM 09

GENERIC AND ANNOTATION

MUHAMMAD SYA`BANDI ABDILLAH

0110219052

STT TERPADU NURUL FIKRI

PROGRAM STUDI TEKNIK INFORMATIKA

DEPOK

2020

PERKENALAN GENERIC

Generic adalah metode di dalam java yang memungkin untuk membuat suatu class, lalu user dapat menentukan tipe data dari class tersebut. Selain class generic juga dapat diterapkan pada method agar user dapat menentukan tipe data dari method tersebut.

```
1  public class Data <T> {
2      T dataPertama;
3      T dataKedua;
4
5      public Data (T dataPertama, T dataKedua) {
6          this.dataPertama = dataPertama;
7          this.dataKedua = dataKedua;
8      }
9
10     public void showData() {
11         System.out.println("Data pertama adalah " + dataPertama);
12         System.out.println("Data pertama adalah " + dataKedua);
13     }
14 }
```

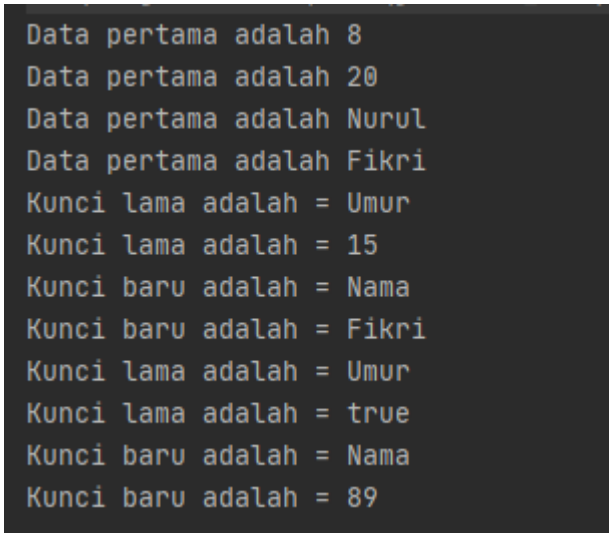
Dari yang bisa kita liat tersebut terdapat tanda **<T>** menandakan bahwa kita membuat sutau generic class. Tanda tersebutlah yang akan kita isi tipe data untuk variable yang terkait. Tanda **<T>** diatas dapat diganti dengan huruf lainnya.

```java
public 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();

        ListData<String, Integer> ListDataInteger = new ListData<String, Integer>("Umur", 15);

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

        ListData<String, Boolean> ListDataString = new ListData<String, Boolean>("Umur", true);

        ListDataString.ListDataLama();
        ListDataString.ListDataBaru("Nama", 89);
    }
}
```

Dan saat diimplementasikan ke dalam kodingan, maka kita bisa memberikan tipe data yang berbeda seperti kode diatas

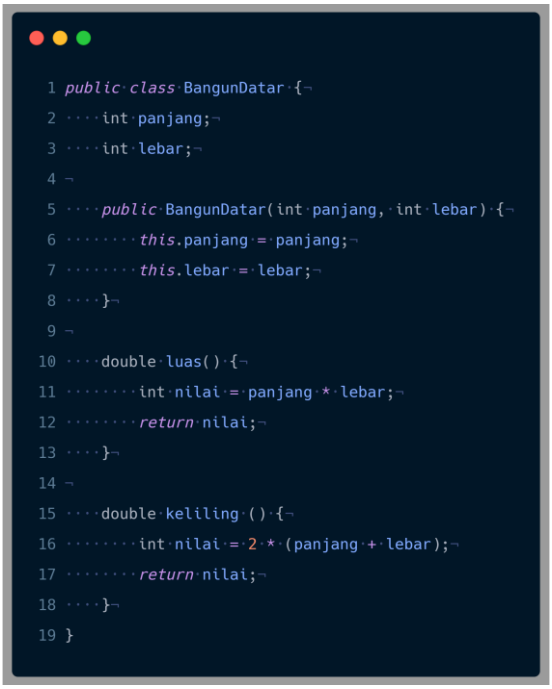
```java
public class ListData <K, V> {

    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("Kunci lama adalah = " + this.nilai);
    }

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

Pada kode diatas adalah implementasi generic class lebih dari satu. Selain class, kita bisa menggunakan generic pada method. Saat dijalankan maka hasilnya adalah :

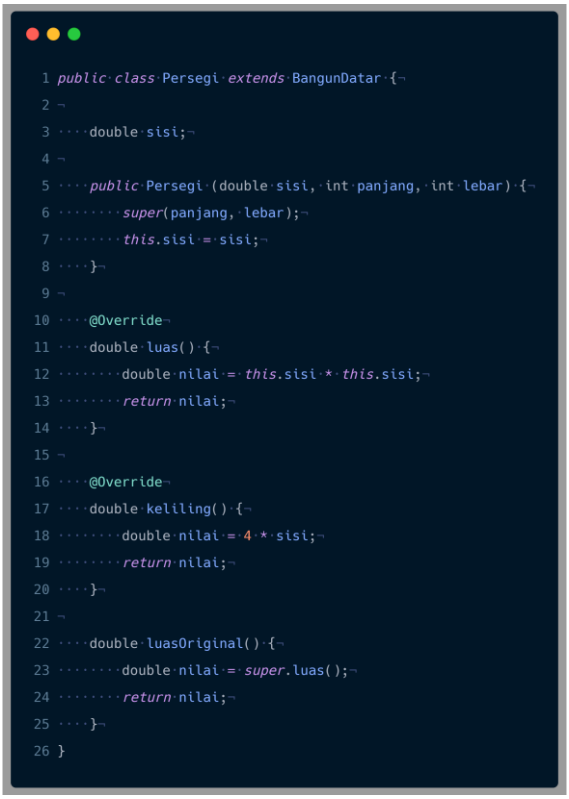


PERKENALAN ANNOTATION

Annotation adalah meta data yang digunakan untuk menginformasikan kepada compiler bagaimana ia menangani suatu kode. Contohnya terdapat pada praktikum sebelumnya telah menggunakan annotation @Override.



```
public 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;
    }
}
```



```
public 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;
    }
}
```

Pada kode diatas kita bisa melihat implementasi @Override apada method luas dan keliling. Annotattion @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 Bangun Datar.

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.

```
public 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 diatas yang menggunakan annotation @Deprecated untuk memberitahu user bahwa method yang dibuat akan segera diganti dengan versi baru dimasa yang akan datang.

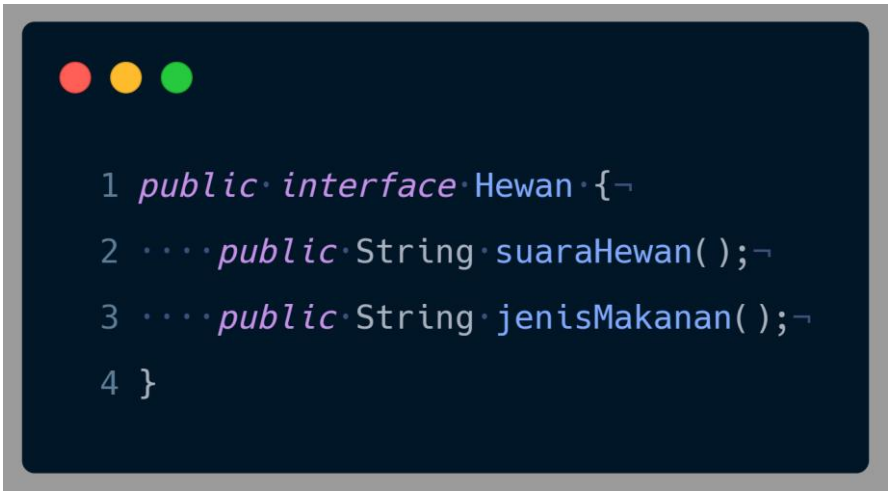
```java
public class MainBangunDatar {
    @SuppressWarnings("deprecation")
    public static void main(String[] args) {
        Persegi persegi = new Persegi(5, 10, 2);
        double luasPersegi = persegi.luas();
        double luasOriginal = persegi.luasOriginal();
        double kelilingPersegi = 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);
    }
}
```

```
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

Process finished with exit code 0
```

Di dalam kode diatas terdapat warning yang disebabkan penggunaan annotation @Deprecated. Untuk menghilangkan warning tersebut kita bisa menggunakan annotation @SuppressWarnings.

```
public 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();

        ListData<String, Integer> ListDataInteger = new ListData<String, Integer>("Umur", 15);

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

        ListData<String, Boolean> ListDataString = new ListData<String, Boolean>("Umur", true);

        ListDataString.ListDataLama();
        ListDataString.ListDataBaru("Nama", 89);
    }
}
```

Selain itu, terdapat 1 parameter lagi pada annotation @SuppressWarning yaitu unchecked. Yang telah kita implementasikan pada kode diatas yang berfugsi untuk menghapus warning yang terjadi ketika kita tidak memasukkan tipe data pada variable dataInteger. Mari kita pelajari lagi annotation yang lain.



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

```java
import java.lang.annotation.*;

@interface test {
    int angka();
}

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

public class Kucing implements Hewan {

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

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

Nah disini kita membuat annotation versi kita sendiri dengan cara @interface namaannotation {}. Pada kode diatas kita membuat annotation sendiri yakni @test dan @TestAnnotation. Pada annotation tersebut kita juga membuat method dengan tipe data sesuai dengan method tersebut. Lalu setelah dibuat, kita bisa implementasikan ke class kucing.

Yang perlu diketahui adalah annotation terbagi menjadi 3 tipe yakni :

1. Marker Annotation = yaitu annotation tanpa value, contohnya adalah @Override.

2. Single-Value Annotation = yaitu annotation dengan 1 value, contohnya adalah @test (angka = 90).
3. Multi-Value Annotation = yaitu annotation yang memiliki value lebih dari 1, contohnya adalah @TestAnnotation (nama = “Fikri”, nilai=90).

Pada kode kita diatas, kita juga mengimport annotation @Target yang berfungsi untuk memberitahu bahwa annotation yang kita buat hanya berfungsi pada element tertentu. Pada contoh diatas @Target untuk memberitahu bahwa annotation bersangkutan hanya dapat digunakan pada method.