VERSION 1.0

FEBRUARI 10, 2024

[STRUKTUR DATA]

MODUL 1, GENERICS JAVA

DISUSUN OLEH:
IZZA IHSAN FATHONY
MOCH IQBAL ARIZKI WIDYANSYAKH

DIAUDIT OLEH:
MUHAMMAD ILHAM PERDANA. S.TR.T., M.T.

LAB. INFORMATIKA

UNIVERSITAS MUHAMMADIYAH MALANG

[STRUKTUR DATA]

PERSIAPAN MATERI

Mahasiswa diharapkan mempelajari materi sebelum mengerjakan tugas, materi yang tercakup antara lain:

1. Bahasa Pemrograman Java
2. Class dan Object
3. Enkapsulasi
4. Array

TUJUAN

Mahasiswa mampu menguasai dan menjelaskan konsep dari Struktur Data Generics serta mengetahui fungsi generics dalam beberapa kasus nyata.

TARGET MODUL

Mahasiswa mampu memahami:

1. Menerapkan Array of Object
2. Generics Class
3. Generics Method
4. Wildcard
5. Enumerated Types

PERSIAPAN SOFTWARE/APLIKASI

1. Java Development Kit
2. Java Runtime Environment
3. IDE (Intellij IDEA, Eclipse, Netbeans, dll)

REFERENSI MATERI

Oracle iLearning Java Programming section 6-1 Generics

Geekforgeeks: https://www.geeksforgeeks.org/generics-in-java/

Youtube (Indonesia): https://www.youtube.com/watch?v=bvWRDAl30Gs

Youtube (English): https://www.youtube.com/watch?v=K1iu1kXkVoA

Javapoint: https://www.javatpoint.com/generics-in-java

Programiz: https://www.programiz.com/java-programming/generics

Note: Dari referensi tersebut mungkin terdapat sedikit perbedaan satu sama yang lain, cukup pahami konsepnya dan terapkan pada kasus di modul ini.

MATERI POKOK

1. Pengertian Generics

Generics merupakan sebuah cara yang digunakan untuk membuat tipe bersifat umum. Generics juga dilakukan untuk menambah stabilitas pada kode program sehingga dapat membantu mempermudah deteksi bug pada saat kompilasi.

2. Mengapa mempelajari Generics

Generics memberikan keamanan tipe pada saat kompilasi yang memungkinan pemrograman mengetahui tipe yang tidak valid. Generics memungkinkan tipe (class dan interface) menjadi parameter Ketika mendefinisikan suatu class, interface, dan method. Sama seperti parameter formal yang lebih akrab digunakan dalam deklarasi method, parameter memberikan cara untuk menggunakan Kembali kode yang sama dengan input yang berbeda. Perbedaannya terdapat pada input yang dilakukan ke parameter formal adalah suatu nilai, sedangkan input untuk parameter adalah suatu tipe.

Penerapan Generics memiliki banyak kegunaan dalam penerapan pada *real project* seperti pembuatan Frontend Website, Mobile Application, Backend Application, karena dapat membuat kode kita menjadi modular dan *reusable* sehingga kode kita menjadi rapi dan mudah dipahami oleh *programmer* lainnya serta meminimalisir penulisan kode yang berulang. Oleh karena itu materi Generics ini menjadi salah satu *basic* Struktur Data yang harus kita pahami dengan dalam karena tidak sedikit *programmer* yang menerapkan konsep ini di perusahaan atau di proyeknya.

3. Contoh perbedaan Dalam Pemanfaatan Generics:

Berikut adalah kode program yang dibuat tanpa menggunakan metode generics (metode yang dipakai adalah metode casting yang digunakan untuk menerima value dari variable):

```
List list = new ArrayList();
list.add("Halo dek");
String s = (String) list.get(0);
```

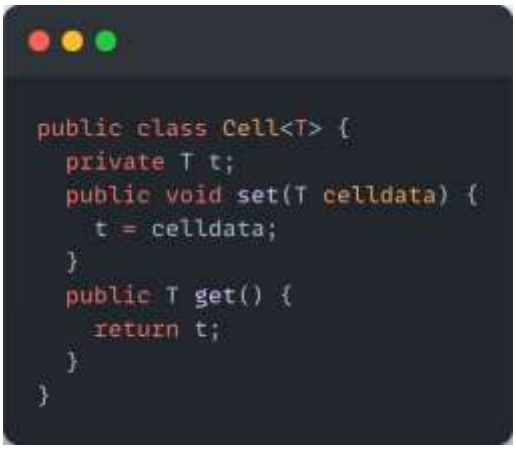
Jika menggunakan metode generics:

```
List<String> list = new ArrayList<String>();
list.add("Halo dek");
String s = list.get(0);
```

Dengan metode generics, pemrogram dapat menerapkan algoritma generics yang bekerja pada koleksi dari berbagai jenis, dapat disesuaikan, aman dan lebih mudah dibaca. Tipe generics dapat dideklarasikan menggunakan angled brackets (tanda <>) dengan didalamnya diisi tipe pengembalian yang ingin didapatkan.

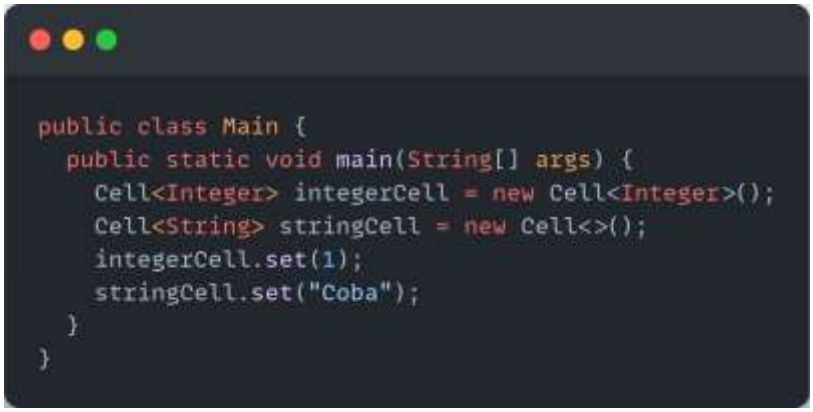
4. Contoh Deklarasi Class Menggunakan Metode Generics:

Penggunaan pada class ditandai dengan angled brackets (tanda <>) lalu dituliskan di dalamnya seperti class Cell berikut:



```
public class Cell<T> {
  private T t;
  public void set(T celldata) {
    t = celldata;
  }
  public T get() {
    return t;
  }
}
```

Untuk pembuatan object dan menerima object yang menggunakan class dapat menggunakan cara sebagai berikut:



```
public class Main {
  public static void main(String[] args) {
    Cell<Integer> integerCell = new Cell<Integer>();
    Cell<String> stringCell = new Cell<>();
    integerCell.set(1);
    stringCell.set("Coba");
  }
}
```

Deklarasi generics dapat dilakukan dengan cara yang lebih ringkas seperti yang dapat dilihat pada class Main diatas, yaitu pada baris ke-3 new Cell<Integer> dan baris ke-4 diringkas menjadi new Cell<>, kedua penulisan tersebut benar semua.

5. Konvensi Penamaan Parameter Pada Generics:

Penamaan parameter dapat dibuat sesuai keinginan *programmer*, berikut penamaan parameter yang paling umum digunakan:

- E – Element (digunanakan secara luas oleh Java Collections Framework)
- K – Key
- N – Number
- T – type
- V – Value

6. Generics Method:

Generics method adalah metode yang digunakan dengan memanfaatkan tipe parameter mereka sendiri. Cara ini mirip dengan mendeklarasikan tipe generics, tetapi cakupan tipe parameter terbatas pada metode yang dideklarasikan. Method static, non-static, dan constructor class generics dapat menggunakan metode ini. Berikut adalah contoh penggunaan generics method:

```
class Util {
  public static <K, V> boolean compare(Pair<K, V> p1, Pair<K, V> p2) {
    return p1.getKey().equals(p2.getKey()) && p1.getValue().equals(p2.getValue());
  }
}

public class Pair<K, V> {
  private K key;
  private V value;

  public Pair(K key, V value) {
    this.key = key;
    this.value = value;
  }

  public void setKey(K key) { this.key = key; }
  public void setValue(V value) { this.value = value; }
  public K getKey() { return key; }
  public V getValue() { return value; }
}
```

Pemanggilan dan pembuatan object dapat dilakukan pada class main seperti berikut:

```
public class Main {
  public static void main(String[] args) {
      Pair<Integer, String> p1 = new Pair<>(1, "Apple");
      Pair<Integer, String> p2 = new Pair<>(2, "Pear");
      boolean same = Util.compare(p1, p2);
  }
}
```

7. Wildcards:

Wildcard dituliskan dalam tanda tanya (?) yang memiliki arti tipe data belum diketahui. Wildcard dapat digunakan dalam beberapa situasi, seperti jenis parameter, field, variabel lokal atau tipe pengembalian. Dalam wildcard terdapat istilah yang disebut dengan upper bound wildcard, yaitu digunakan untuk memberi Batasan pada variabel. Berikut adalah contoh penggunaannya:

```java
import java.util.Arrays;
import java.util.List;

public class Wildcard {
    public static void printList(List<?> list) {
        for (Object elm : list) {
            System.out.print(elm + " ");
        }
        System.out.println();
    }
    public static void main(String[] args) {
        List<Integer> dataInt = Arrays.asList(7, 9, 10);
        List<String> dataString = Arrays.asList("Tujuh", "Sembilan", "Sepuluh");
        printList(dataInt);
        printList(dataString);
    }
}
```

8. Enumerated Types:

Enumerated types adalah tipe data khusus yang memungkinkan variabel menjadi Kumpulan konstanta yang telah ditentukan. Variabel harus sama dengan salah satu nilai yang telah ditentukan sebelumnya. Enumerated types ini bisa digunakan untuk arah kompas (NORTH, SOUTH, EAST, WEST) dan lainnya. Berikut adalah contoh Enumerated types untuk membuat deskripsi nama hari:

```java
public enum Day {
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY
}
```

```java
public class EnumType {
    Day day;

    public EnumType(Day day) {
        this.day = day;
    }

    public void aboutThisDay() {
        switch (day) {
            case MONDAY:
                System.out.println("Mondays are bad");
                break;
            case FRIDAY:
                System.out.println("Fridays are better");
                break;
            case SATURDAY: case SUNDAY:
                System.out.println("Weekend are best");
                break;
            default:
                System.out.println("Midweek are :(");
                break;
        }
    }

    public static void main(String[] args) {
        EnumType firstDay = new EnumType(Day.MONDAY);
        firstDay.aboutThisDay();
        EnumType thirdDay = new EnumType(Day.WEDNESDAY);
        thirdDay.aboutThisDay();
        EnumType fifthDay = new EnumType(Day.FRIDAY);
        fifthDay.aboutThisDay();
        EnumType sixDay = new EnumType(Day.SATURDAY);
        sixDay.aboutThisDay();
    }
}
```

CODELAB

LATIHAN 1

Kita akan membuat sebuah kelas generik bernama **Box**, yang dapat menyimpan nilai dari satu jenis data. Dalam contoh ini, kita akan menggunakan tipe data integer.

```
class Box<T> {
    private T value;

    public void setValue(T value) {
        this.value = value;
    }

    public T getValue() {
        return value;
    }

    public void display() {
        System.out.println("Box contains: " + value);
    }
}

public class Main {
    public static void main(String[] args) {
        Box<Integer> integerBox = new Box<>();

        integerBox.setValue(42);

        int intValue = integerBox.getValue();
        System.out.println("Integer value: " + intValue);

        Box<String> stringBox = new Box<>();

        stringBox.setValue("Hello, Generics!");

        String stringValue = stringBox.getValue();
        System.out.println("String value: " + stringValue);
    }
}
```

1. **Class Box<T>**
    a. **Class Box** adalah kelas generik dengan parameter tipe **T**, yang memungkinkan kita untuk menggunakan tipe data yang berbeda tanpa perlu membuat kelas yang berbeda.
    b. Kelas ini memiliki satu atribut private **value** dengan tipe **T** untuk menyimpan nilai generik.

2. **Method setValue, getValue, dan display**
    a. **setValue** digunakan untuk menyimpan nilai ke dalam objek **Box**.
    b. **getValue** digunakan untuk mengambil nilai dari objek **Box**.
    c. **display** digunakan untuk menampilkan nilai yang disimpan dalam objek **Box**.
3. **Method main**
    a. Membuat objek **Box<Integer>** dan **Box<String>** bernama **integerBox** dan **stringBox** untuk menyimpan nilai bertipe **Integer** dan **String**.
    b. Memanggil method **setValue(42)** dan **setValue("Hello generics")** untuk menyimpan nilai ke dalam **integerBox** dan **stringBox**.
    c. Memanggil method **getValue** untuk mengambil nilai dari **integerBox** dan **stringBox**.

TUGAS

TUGAS 1

Buatlah program input barang menggunakan Java generics yang memungkinkan pengguna untuk menangani berbagai barang. Berikut adalah ketentuan tugas dari kode tersebut:

1. Class Barang harus memiliki dua generics yang berbeda.
2. Class Barang wajib memiliki method get dan konstruktor.
3. Parameter pada konstruktor Barang harus terdiri dari dua generics dan satu enum. Pengaplikasiannya bebas.
4. Program harus mampu menginputkan dua generics dan memilih enum saat melakukan operasi.
5. Hanya diperbolehkan menggunakan class Barang, Main, dan enum dalam program.

Berikut adalah contoh output program:

```
Masukkan Nama: Wagyu Steak
Masukkan Harga: 1500000
Pilih Jenis Barang:
0. SANDANG
1. PANGAN
2. PAPAN
Masukkan pilihan: 1

Informasi Barang:
Nama : Wagyu Steak
Harga : 1500000
Jenis : PANGAN
```

Mahasiswa diperbolehkan melakukan improvisasi dari tugas yang diberikan dengan syarat tidak mengurangi ketentuan yang ada.

KRITERIA & DETAIL PENILAIAN

Kriteria	Nilai
Codelab	20
Tugas	60
Pemahaman	20
Total	**100**