VERSI 2.2

SEPTEMBER, 2020

[STRUKTUR DATA]

MODUL 1, GENERICS

TIM PENYUSUN: - DOSEN
- DICKY PRABOWO OCTIANTO

PRESENTED BY: LAB. TEKNIK INFORMATIKA

UNIVERSITAS MUHAMMADIYAH MALANG

[STRUKTUR DATA]

CAPAIAN PEMBELAJARAN MATA KULIAH

Mahasiswa mampu menguasai & menjelaskan konsep dari struktur data Generics

SUB CAPAIAN PEMBELAJARAN MATA KULIAH

Mahasiswa mampu memahami:

1. Menerapkan array of object
2. Generic class
3. Generic method
4. Wildcard
5. Enumerated types

PERSYARATAN PEMAHAMAN

1. Class dan Object
2. Enkapsulasi
3. Array

KEBUTUHAN HARDWARE & SOFTWARE

- Java Development Kit
- Java Runtime Environment
- IDE (Intellij IDEA, Eclipse, Netbeans, dll.)

REFERENSI MATERI

Oracle iLearning Java Programming section 6-1 Generics

MATERI POKOK

Pengertian:

Generic merupakan teknik untuk menambah stabilitas pada kode program yang dapat membantu untuk mepermudah deteksi bug pada waktu kompilasi.

Kenapa memakai generics?:

Singkatnya, generic memungkinkan tipe (class dan interface) menjadi parameter ketika mendefinisikan 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 adalah input ke parameter formal adalah nilai, sedangkan input untuk mengetik parameter adalah tipe.

Contoh perbedaan:

Ketika code ditulis tidak menggunakan metode generic maka diperlukan metode casting untuk menerima value dari variable tersebut.

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

Sedangkan jika menggunanakan metode generic:

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

Dengan menggunakan metode generic, pemrogram dapat menerapkan algoritma generics yang bekerja pada koleksi dari berbagai jenis, dapat disesuaikan, aman dan lebih mudah dibaca.

Tipe generic dapat dideklarasikan menggunakan angled brackets (tanda <>) didalamnya diisi dengan tipe pengembalian yang ingin didapatkan.

Contoh deklarasi kelas menggunakan metode generic:

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

Sehingga untuk membuat object dan menerima object yang menggunakan kelas tersebut dapat menggunakan cara sebagai berikut:

```
public class CellDriver{
    public static void public static void main(String[] args) {
        Cell<Integer> integetCell = new Cell<Integer>();
        Cell<String> stringCell = new Cell<String>();
        integerCell.set(1);
        stringCell.set("Test");
        int num = integerCell.get();
        String str = stringCell.get();
    }
}
```

Untuk membuat deklarasi generic lebih ringkas juga dapat menggunakan cara:

```
Example<String> showMe = new Example<String>();
```

menjadi

```
Example<String> showMe = new Example();
```

Konvensi penamaan parameter pada generics:

Penamaan parameter yang paling umum biasa digunakan:

- E - Element (digunakan secara luas oleh Java Collections Framework).
- K - Key
- N - Number
- T - Type
- V – Value

Anda juga dapat mendefenisikan parameter dalam bentuk lain.

Generic methods:

Metode generic (Generic methods) adalah metode dengan parameter tipe mereka sendiri. Ini mirip dengan mendeklarasikan tipe generics, tetapi cakupan parameter tipe terbatas pada metode yang dideklarasikan. Method static dan non-static diperbolehkan, dan juga constructor kelas generic juga dapat menggunakan metode generic ini.

Contoh penggunaan generics method:

```
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; }
}
```

Sehingga deklarasi dan pemanggilannya dapat menggunakan cara:

```
Pair<Integer, String> p1 = new Pair<>(1, "apple");
Pair<Integer, String> p2 = new Pair<>(2, "pear");
boolean same = Util.<Integer, String>compare(p1, p2);
```

Wildcards:

Dalam kode generics, tanda tanya (?), dibaca wildcard, mewakili tipe yang tidak dikenal. Wildcard dapat digunakan dalam berbagai situasi: sebagai jenis parameter, field, atau variabel lokal; bisa juga sebagai tipe pengembalian.

Upper bounded wildcards:

Anda dapat menggunakan Upper bounded wildcards untuk memberi Batasan pada variabel. Misalnya, pemrogram ingin menulis metode yang berfungsi pada List <Integer>, List <Double>, dan List <Number>.

Contoh penggunaannya adalah sebagai berikut:

Method sumOfList akan memberikan pengembalian penjumlahan dari nomor yang ada dalam variable list.

```
public static double sumOfList(List<? extends Number> list) {
    double s = 0.0;
    for (Number n : list)
        s += n.doubleValue();
    return s;
}
```

Sehingga pemanggilan method sumOfList jika menggunakan elemen Integer dapat dilakukan dengan cara sebagai berikut:

```
List<Integer> li = Arrays.asList(1, 2, 3);
System.out.println("sum = " + sumOfList(li));
```

Enumerated types:

Tipe enum adalah tipe data khusus yang memungkinkan variabel menjadi seperangkat konstanta yang telah ditentukan. Variabel harus sama dengan salah satu nilai yang telah ditentukan sebelumnya. Contoh umum termasuk arah kompas (nilai NORTH, SOUTH, EAST, dan WEST) dan hari-hari dalam seminggu.

Berikut contoh Enumerated types untuk membuat deskripsi nama hari:

```
public class EnumTest {
    Day day;

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

    public void tellItLikeItIs() {
        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("Weekends are best.");
                break;

            default:
                System.out.println("Midweek days are so-so.");
                break;
        }
```

```
    }

    public static void main(String[] args) {
        EnumTest firstDay = new EnumTest(Day.MONDAY);
        firstDay.tellItLikeItIs();
        EnumTest thirdDay = new EnumTest(Day.WEDNESDAY);
        thirdDay.tellItLikeItIs();
        EnumTest fifthDay = new EnumTest(Day.FRIDAY);
        fifthDay.tellItLikeItIs();
        EnumTest sixthDay = new EnumTest(Day.SATURDAY);
        sixthDay.tellItLikeItIs();
        EnumTest seventhDay = new EnumTest(Day.SUNDAY);
        seventhDay.tellItLikeItIs();
    }
}
```

MATERI PRAKTIKUM

LATIHAN 1.1 ARRAY OF OBJECT MENGGUNAKAN GENERICS CLASS:

Biasakan untuk mengetik secara manual tidak menggunakan copy paste sehingga mengerti alur proses dari kode.

1. Membuat kelas generics untuk konsumsi:

```
class Konsumsi<M, I> {
  private M m;
  private I i;

  public M getM() {
    return m;
  }

  public I getI() {
    return i;
  }

  public void setKonsumsi(M makanan, I minuman) {
    this.m = makanan;
    this.i = minuman;
  }
}
```

2. Membuat Hidangan:

```
class Hidangan {
  protected String namaHidangan;

  public String getNamaHidangan() {
    return namaHidangan;
  }

  public void setNamaHidangan(String namaHidangan) {
    this.namaHidangan = namaHidangan;
  }

  public String disantap() {
    return "Makanan Dihidangkan";
  }
}
```

3. Membuat kelas Minuman yang di extends ke kelas Hidangan

```
class Minuman extends Hidangan{
  public String disantap() {
    return this.getNamaHidangan() + " diminum";
  }
}
```

4. Membuat kelas Makanan yang di extends ke kelas Hidangan

```
class Makanan extends Hidangan {
  public String disantap() {
    return this.getNamaHidangan() + " dimakan";
  }
}
```

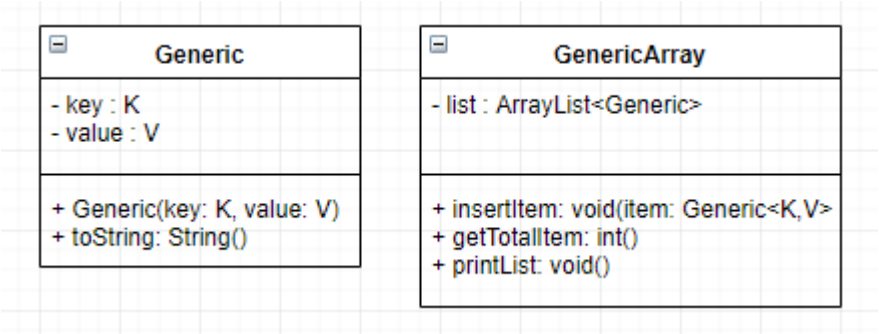
5.Setelah semua kelas telah dibuat uji coba menggunakan kelas main

```
public class Main {
    public static void main(String[] args){
     ArrayList<Konsumsi> listKonsumsi = new ArrayList<>();
     Konsumsi<Makanan,Minuman> breakfast = new Konsumsi<>();
     Konsumsi<Makanan,Minuman> lunch = new Konsumsi<>();

     Makanan roti = new Makanan();
     roti.setNamaHidangan("Roti Tawar");
     Minuman susu = new Minuman();
     susu.setNamaHidangan("Susu Sapi");
     breakfast.setKonsumsi(roti,susu);
     listKonsumsi.add(breakfast);

     Makanan nasi = new Makanan();
     nasi.setNamaHidangan("Nasi Putih");
     Minuman air = new Minuman();
     air.setNamaHidangan("Air Putih");
     lunch.setKonsumsi(nasi,air);
     listKonsumsi.add(lunch);

     System.out.println("Menu Konsumsi");
     for (Konsumsi<Makanan,Minuman> konsumsi: listKonsumsi ) {
       Makanan makanan = konsumsi.getM();
       Minuman minuman = konsumsi.getI();

       System.out.println(makanan.disantap());
       System.out.println(minuman.disantap());
     }
   }
}
```

LEMBAR KERJA

KEGIATAN 1

Buat class generics bernama Generic yang menyimpan data menggunakan parameter Kunci (key) dan Nilai (value). Kemudian buat lagi kelas bernama GenericArray sehingga gambaran diagramnya akan tampak seperti dibawah berikut.



Fungsi dalam kelas Generic antara lain:

Tipe	Nama Method	Deskripsi
constructor	Generic(K key, V value)	Inisialisasi object key dan value
String	toString()	Mengembalikan nilai variable key dan value.

Fungsi dalam kelas GenericArray antara lain:

Tipe	Nama Method	Deskripsi
String	printList()	Menampilkan semua item dengan informasi kunci (Value) dan nilainya (value).
Int	getTotalItem()	Menampilkan jumlah data (instances) yang sudah dimasukkan
Void	insertItem(K Value, V value)	Mengisi data.

Isi minimal 3 data dalam object tersebut sehingga jika ditampilkan akan terdapat 3 instances (data) dari object tersebut.

Hewan	Jenis
Harimau	Karnivora
Ikan Lele	Omnivora
Sapi	Herbivora

Praktikkan semua method diatas dalam driver Class (Main method).

KEGIATAN 2

Buat kelas generik bernama “Kubus” yang akan menyimpan nilai panjang, lebar, dan tinggi dengan syarat hanya menerima jenis Nomer (Numbers).
Fungsi dalam kelas Kubus antara lain:

Tipe	Nama Method	Deskripsi
Constructor	Kubus(T tinggi, T panjang, T lebar)	Inisialisasi variable tinggi, panjang, dan lebar.
String	toString()	Menampilkan nilai dari variable panjang, tinggi, dan lebar secara bersamaan dalam bentuk string.
Long	getResultAsLong()	Mendapatkan hasil perhitungan volume dalam bentuk tipe long
Int	getResultAsInt ()	Mendapatkan hasil perhitungan volume dalam bentuk tipe integer.
Double	getResultAsDouble ()	Mendapatkan hasil perhitungan volume dalam bentuk tipe double.

Sehingga contoh inisialisasi object akan seperti berikut:

```
Kubus<Double> kubusTipeDouble = new Kubus<>();
Kubus<Integer > kubusTipeInteger = new Kubus<>();
Kubus<String > kubusTipeString = new Kubus<>(); //error type parameter; should extend ‘java.lang.Number’
```

Praktikkan semua method diatas dalam driver Class (Main method).

CATATAN

aturan umum penulisan bahasa JAVA agar mudah di koreksi oleh asisten:

1. Untuk nama kelas,interface,enum, dan yang lainnya biasakan menggunakan gaya CamelCase (diawali dengan huruf besar pada tiap kata untuk mengganti spasi) seperti: **K**ursi , **J**alan**R**aya, **P**arkiran**G**edung, dan lain seterusnya.
2. Untuk penulisan nama method, dan attribute diawali dengan huruf kecil di awal katadan menggunakan huruf besar untuk kata setelahnya, seperti: **g**et**N**ama**J**alan, **n**ama**J**alan, **h**arga, **s**et**N**ama**J**alan, dan lain seterusnya.
3. Jika menggunakan IDE Intellij jangan lupa untuk memformat penulisan kode agar terlihat rapi menggunakan menu code -> show reformat file dialog -> centang semua field dan klik ok.

Silahkan dikerjakan tanpa copy – paste

RUBRIK PENILAIAN

Soal	Nilai
Kegiatan 1:	50%
Kegiatan 2:	50%