PRAKTIKUM 5

GENERIC 1

A. TUJUAN PEMBELAJARAN

1. Memahami mengenai konsep generic.
2. Mengetahui cara merubah dari bentuk non generic menjadi generic.
3. Memahami generic pada Collection.

B. DASAR TEORI

Generic merupakan cara Java dalam melakukan generalisasi terhadap tipe data tanpa mengurangi kemampuan Java dalam menjaga keamanan penggunaan tipe data.

```
public class NonGen {
  Object ob;

  NonGen(Object o) {
    ob = o;
  }

  Object getob() {
    return ob;
  }
  void showType() {
    System.out.println("Type of ob is " +
                       ob.getClass().getName());
  }
}
```

Pada object Box, kita bisa memasukkan sembarang object karena parameter pada method `add()`adalah Class Object, tapi pada saat mengambil object tersebut harus diubah sesuai dengan tipe dari object tersebut.

```
public class NonGenDemo {
  public static void main(String args[]) {
    NonGen integerObject;
    integerObject = new NonGen(88);

    integerObject.showType();
```

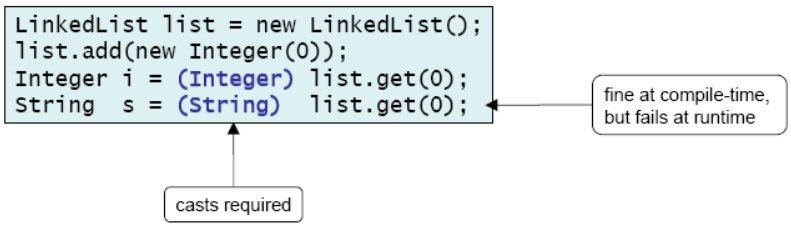
```
    int v = (Integer) integerObject.getob();
    System.out.println("value: " + v);


    NonGen strOb = new NonGen("10");
    strOb.showType();

    String str = (String) strOb.getob();
    System.out.println("value: " + str);

    //menyebabkan exception
    Integer i = (Integer) strOb.getob();
  }
}
```

Terjadi exception karena pada object strOb dimasukkan object 10 tapi dengan tipe String, tapi pada saat mengambil object, diubah menjadi tipe Integer. Tipe data tidak sesuai sehingga menyebabkan terjadinya exception.

Output :

```
Type of ob is java.lang.Integer
value: 88
Exception in thread "main" java.lang.ClassCastException:
java.lang.String cannot be cast to java.lang.Integer
Type of ob is java.lang.String
value: Non-Generics Test
        at com.NonGenDemo.main(NonGenDemo.java:30)
Java Result: 1
```

Permasalahan yang muncul dengan penyimpanan objek yang non homogeneous adalah memerlukan banyak casting. Tidak ada pengecekan pada saat kompile, kesalahan baru bisa terdeteksi pada saat runtime.



Cara mendeklarasikan Class Generic

Ubah class Box Non Generic menjadi class Box Generic. Pendeklarasian type generic dengan mengubah public class Box() menjadi public class Box <T>

– T biasanya disebut parameter type formal (formal type parameter)
– T adalah type parameter yang akan diganti dengan tipe sebenarnya (Type dari T bisa berupa class, interface atau tipe variabel lainnya).
– T adalah nama dari type parameter.

```
public class Gen<T> {
  T ob; // declare an object of type T

  Gen(T o) {
    ob = o;
  }

  T getob() {
    return ob;
  }

  void showType() {
    System.out.println("Type of T is " + ob.getClass().getName());
  }
}
```

```
public class GenDemo {
  public static void main(String args[]) {
    Gen<Integer> iOb;
    iOb = new Gen<Integer>(88);

    iOb.showType();

    int v = iOb.getob();
    System.out.println("value: " + v);

    System.out.println();

    Gen<String> strOb = new Gen<String>("Generics Test");

    String v2 = strOb.getob();
  }
}
```

Objek `iOb` adalah objek dari class Generic, menggunakan tipe data Integer, sehingga pada saat mengambil objek menggunakan method get(), tidak perlu proses casting.

Aturan Penamaan Type Parameter

Nama type parameter biasanya satu huruf dan huruf besar. Jenis nama tipe parameter yang sering digunakan :

– E - Element (biasanya digunakan untuk Collection Framework)
– K – Key
– N – Number
– T - Type
– V - Value
– S,U,V dll. - 2nd, 3rd, 4th types

Generic pada Method

Pada contoh sebelumnya, kita mendefinisikan type parameter pada level class.

Sebenarnya tipe variable ini juga dapat didefinisikan pada level method.

```
public class GenericMethodTest {
    public static<E> void printArray(E[] inputArray){
        for(E element : inputArray)
            System.out.printf("%s",element);
        System.out.println("");
    }
    public static void main(String[] args) {
        Integer[] intArray = {1,2,3,4,5} ;
        Double[] doubleArray = {1.1,2.2,3.3,4.4,5.5};
        Character[] charArray = {'J','A','V','A'};

        printArray(intArray);
        printArray(doubleArray);
        printArray(charArray);
    }
}
```

Subtyping

Jika B adalah suatu subtype dari A dan G adalah suatu tipe data generic, maka tidak berarti G<B> adalah subtype dari G<A>.

Generic pada Collection

• List <E> myList ;

E disebut type variabel, variabel yang diganti dengan type. Jika E adalah class, maka kita bisa melewatkan subclass E. Jika E adalah interface maka kita bisa melewatkan class yang mengimplementasikan E.

```
public class ArrayListGenericDemo {
```

```
  public static void main(String[] args) {
    ArrayList<String> data = new ArrayList<String>();
    data.add("hello");
    data.add("goodbye");
    // data.add(new Date()); This won't compile!
}
```

Objek List data bertipe String, sehingga yang bisa dimasukkan hanya bertipe String saja, jika kita memasukkan objek selain String maka error.

C. TUGAS PENDAHULUAN

Buatlah review mengenai :

- permasalahan pada class non generic
- merubah bentuk class non generic menjadi class generic
- manfaat dari class generics
- generic pada collection.

D. PERCOBAAN

Percobaan 1 : Membuat class NonGeneric, membuat objek dan mengambil nilai dari class NonGeneric

```
public class NonGen {
  Object ob;

  NonGen(Object o) {
    ob = o;
  }

  Object getob() {
    return ob;
  }
  void showType() {
    System.out.println("Type of ob is " +
                       ob.getClass().getName());
  }
}
```

```
public class NonGenDemo {
  public static void main(String args[]) {
    NonGen integerObject;
    integerObject = new NonGen(88);
```

```
    integerObject.showType();

    int v = (Integer) integerObject.getob();
    System.out.println("value: " + v);


    NonGen strOb = new NonGen("Non-Generics Test");
    strOb.showType();

    String str = (String) strOb.getob();
    System.out.println("value: " + str);

    //ini yang menyebabkan excepton
    integerObject = strOb;
    v = (Integer) integerObject.getob();

  }
}
```

Percobaan 2 : Membuat class NonGeneric, membuat objek dan mengambil nilai dari class NonGeneric

```
public class Gen<T> {
  T ob; // declare an object of type T

  Gen(T o) {
    ob = o;
  }

  T getob() {
    return ob;
  }

  void showType() {
    System.out.println("Type of T is " + ob.getClass().getName());
  }
}
```

```
public class GenDemo {
  public static void main(String args[]) {
    Gen<Integer> iOb;
    iOb = new Gen<Integer>(88);

    iOb.showType();

    int v = iOb.getob();
    System.out.println("value: " + v);

    System.out.println();

    Gen<String> strOb = new Gen<String>("Generics Test");
```

```
    strOb.showType();

    String str = strOb.getob();
    System.out.println("value: " + str);
  }
}
```

Percobaan 3 : Class Generic dengan Dua Type Parameter

```
class TwoGen<T, V> {
  T ob1;

  V ob2;

  TwoGen(T o1, V o2) {
    ob1 = o1;
    ob2 = o2;
  }

  void showTypes() {
    System.out.println("Type of T is " + ob1.getClass().getName());
    System.out.println("Type of V is " + ob2.getClass().getName());
  }

  T getob1() {
    return ob1;
  }

  V getob2() {
    return ob2;
  }
}
```

```
public class TwoGenDemo {
  public static void main(String args[]) {
    TwoGen<Integer, String> tgObj = new TwoGen<Integer, String>(88,
"Generics");
    tgObj.showTypes();
    int v = tgObj.getob1();
    System.out.println("value: " + v);
    String str = tgObj.getob2();
    System.out.println("value: " + str);
  }
}
```

Percobaan 4 : Generic pada Method

```
public class GenericMethodTest {
    public static<E> void printArray(E[] inputArray){
        for(E element : inputArray)
```

```
            System.out.printf("%s",element);
        System.out.println("");
    }
    public static void main(String[] args) {
        Integer[] intArray = {1,2,3,4,5} ;
        Double[] doubleArray = {1.1,2.2,3.3,4.4,5.5};
        Character[] charArray = {'J','A','V','A'};

        printArray(intArray);
        printArray(doubleArray);
        printArray(charArray);
    }
}
```

Percobaan 5 : Generic Pada List

```
public class ArrayListGenericDemo {
  public static void main(String[] args) {
    ArrayList<String> data = new ArrayList<String>();
    data.add("hello");
    data.add("goodbye");

    // data.add(new Date()); This won't compile!

    Iterator<String> it = data.iterator();
    while (it.hasNext()) {
      String s = it.next();
      System.out.println(s);
    }
  }
}
```

Percobaan 6 : Generic pada Map

```
public class GenericMap {
    public static void main(String args[])
    {
        HashMap<String,Integer> hm = new HashMap<String,Integer>();

        hm.put("apel", 20);
        hm.put("anggur", 13);
        hm.put("jeruk", 5);
    }
}
```

Percobaan 7 : Subtyping

```
public class GenSubtypingDemo {
    public static void main(String args[]) {
       Gen<String> gen1 = new Gen("abc");
       Gen<Object> gen2 = new Gen("abc");
```

```
        gen2 = gen1 ; //error pada saat dikompile

    }
}
```

E. LATIHAN

1. Buatlah method generic untuk menghitung banyaknya bilangan prima yang tersimpan di collection.

Input : [3,5,7,9,12,15,19] Terdapat 4 bilangan prima.

Alur pengerjaan, dapat mengikuti cara di bawah ini :

```
public interface UnaryPredicate<T> {
    public boolean test(T obj);
}
```

```
class PrimePredicate implements UnaryPredicate<Integer> {
    public boolean test(Integer i) { //isilah }
}
```

```
public final class Algorithm {
public static <T> int countIf(Collection<T> c, UnaryPredicate<T> p) {
        int count = 0;
        for (T elem : c)
            if (p.test(elem))
                ++count;
        return count;
    }
}
```

```
public class Test {
    public static void main(String[] args) {
        Collection<Integer> ci = Arrays.asList(1, 2, 3, 4);
        int count = Algorithm.countIf(ci, new PrimePredicate());
        System.out.println("Number of prime integers = " + count);
    }
}
```

2. Apakah program di bawah ini dapat dikompile ? Jika tidak, jelaskan !

```
public final class Algorithm {
    public static T max(T x, T y) {
```

```
        return x > y ? x : y;
    }
}
```

3. Buatlah method generic untuk menukar posisi dua element yang tersimpan di array
   public static <T> void swap(T[] a, int i, int j) {..}

4. Apakah program di bawah ini dapat dikompile ? Jika tidak, jelaskan !

```
public class Singleton<T> {
    public static T getInstance() {
        if (instance == null)
            instance = new Singleton<T>();
        return instance;
    }
    private static T instance = null;
}
```

5. Apakah program di bawah ini dapat dikompile ? Jika tidak, jelaskan !

```
class Shape { /* ... */ }
class Circle extends Shape { /* ... */ }
class Rectangle extends Shape { /* ... */ }

class Node<T> { /* ... */ }

Node<Circle> nc = new Node<>();
Node<Shape>  ns = nc;
```

6. Apakah program di bawah ini dapat dikompile ? Jika tidak, jelaskan !

```
class Node<T> implements Comparable<T> {
    public int compareTo(T obj) { /* ... */ }
    // ...
}
Node<String> node = new Node<>();
Comparable<String> comp = node;
```

F. LAPORAN RESMI

Kerjakan hasil percobaan(D) dan latihan(E) di atas dan tambahkan analisa.