

Generics

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

Generic Class Declaration

- Pendeklarasian tipe generics dengan mengubah `public class Box` menjadi `public class Box<T>`
- T biasanya disebut parameter tipe formal (formal type parameter)
- T adalah tipe parameter yang akan diganti dengan tipe sebenarnya (Tipe dari T bisa berupa class, interface atau tipe variabel lainnya).
- T adalah nama dari tipe parameter.

```
public class Box<T> {
    private T t; // T stands for type
    public Box(T t){
        this.t = t;
    }
    public void add(T t){
        this.t = t;
    }
    public T get(){
        return this.t;
    }
}
```

Subtyping

- Jika B adalah subtype dari A dan G adalah data generics maka tidak berarti `G` adalah subtype dari `G<A>`

```
Object o ;
String s ;

Box<String> gS = new Box<String>();
Box<Object> gO = new Box<Object>();

o = s // benar
gO = gS /** Error Box<String> bukan
subtype dari Box<Object> */
```

Generic Method

- Contohnya method `inspect` pada class `Box`
- Contoh lainnya method `isIn` di bawah ini

```
static <T,V extends T> boolean isIn(T, V[] y){
    for(int i = 0; i < y.length; i++){
        if(x.equals(y[i]))
            return true
    }
    return false
}

public static void main(String[] args){
    Integer[] nums = {1,2,3};
    if(isIn(2,nums)){
        System.out.println("2 is in nums");
    }
    String[] strings = {"a","b","c"};
    if(isIn("a",strings)){
        System.out.println("a is in strings");
    }
}
```

Generics in List

- `List<E> myList;`
- E disebut tipe variabel, variabel yang diganti dengan tipe.
- Jika E adalah class, maka kita bisa melewati subclass E.
- Jika E adalah interface maka kita bisa melewati class yang mengimplementasikan E.

```
List<E> myList = new ArrayList<E>();
```

Raw Type

- Java membolehkan class generic digunakan tanpa tipe argument tetapi akan mengeluarkan warning saat di-compile.

```
Box rawType = new Box(new Double(89.5));
```

Bounded Type Parameter

- Jika kita ingin memberikan batasan tipe yang diperbolehkan untuk dilewatkan ke tipe parameter.
- Contoh method dengan parameter `Number`, hanya menerima object dari class `Number` dan subclassnya. Hal ini yang disebut bounded type parameter.
- Cara:
 - <U extends Number>
- Jika terdapat interface yang harus diimplementasikan gunakan &:
 - <U extends Number & MyInterface>
- Tidak boleh menggunakan "super"

```
// Tambahkan method inspect di Box

public <U extends Number> void inspect (U u){
    System.out.println(u);
}

Box<Integer> box = new Box<Integer>();
box.add(10);
box.inspect("text"); // Error
```