

# **LAPORAN PRAKTIKUM PERTEMUAN-11**

Diajukan untuk memenuhi salah satu tugas praktikum Mata kuliah Pemrograman Berorientasi Objek

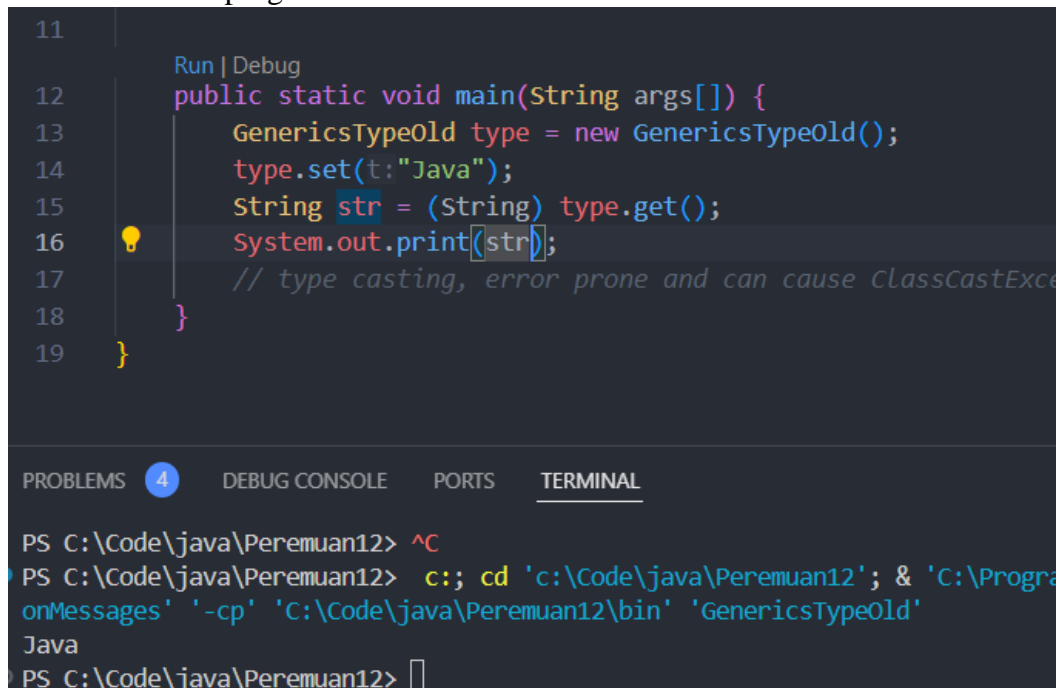


**Disusun Oleh:  
Daiva Raditya Pradipa (231511039)**

**Jurusan Teknik Komputer dan Informatika  
Program Studi D-3 Teknik Informatika  
Politeknik Negeri Bandung  
2024**

## 1. GenericTypeOld

### a. Screenshot hasil program



The screenshot shows an IDE with a Java file named `GenericTypeOld.java`. The code is as follows:

```
11  
12     Run | Debug  
13     public static void main(String args[]) {  
14         GenericTypeOld type = new GenericTypeOld();  
15         type.set(t:"Java");  
16         String str = (String) type.get();  
17         System.out.print(str);  
18         // type casting, error prone and can cause ClassCastException  
19     }
```

Below the code editor, the **TERMINAL** tab is active, showing the command prompt output:

```
PS C:\Code\java\Peremuan12> ^C  
PS C:\Code\java\Peremuan12> c::; cd 'c:\Code\java\Peremuan12'; & 'C:\Program Files\Java\jdk-11.0.2\bin\java.exe' -cp 'C:\Code\java\Peremuan12\bin' 'GenericTypeOld'  
Java  
PS C:\Code\java\Peremuan12> 
```

### b. Penjelasan

Pada program ini class `GenericTypeOld` yang memiliki atribut `t` dengan tipe object, kemudian terdapat method getter dan setter untuk mengambil dan manipulasi atribut `t` tersebut. Kemudian terdapat main program untuk meninstansikan class `GenericTypeOld` menjadi sebuah object. Kemudian terdapat pemanggilan method `set` pada object `type` yang dengan parameter string “java”, mengapa hal ini bisa dilakukan meskipun pada deklarasi method parameter method ini bertipe object ? Hal ini dikarenakan dalam Java, semua kelas adalah subclass dari `Object`, termasuk `String`. Karena itu, `String` bisa digunakan sebagai argumen untuk parameter `Object`. Kemudian terdapat typecasting untuk mengasign nilai dari `t` pada class `GenericTypeOld` ke variabel dengan tipe string menggunakan type casting string.

### c. Masalah

### d. Solusi

### e. Teman Yang Membantu

## 2. GenericsType

### a. Screenshot hasil program

```
11
12 Run | Debug
13 public static void main(String args[]) {
14     GenericsType<String> type = new GenericsType<>();
15     type.set(t1:"Java"); // valid
16     GenericsType type1 = new GenericsType(); // raw type
17     type1.set(t1:"Java"); // valid
18     type1.set(t1:10); // valid and autoboxing support
19     System.out.print(type1.get());
20 }
```

PROBLEMS 4 DEBUG CONSOLE PORTS TERMINAL

```
PS C:\Code\java\Peremuan12> ^C
PS C:\Code\java\Peremuan12> c:: cd 'c:\Code\java\Peremuan12'; & 'C:\Program
onMessages' '-cp' 'C:\Code\java\Peremuan12\bin' 'GenericsType'
10
PS C:\Code\java\Peremuan12> 
```

b. Penjelasan

Pada kelas `GenericsType`, implementasinya mirip dengan kelas `GenericsTypeOld` sebelumnya. Perbedaannya adalah pada kelas ini diterapkan *generic programming* menggunakan *generic class*. Penerapan *generic programming* ditandai dengan adanya sintaks `<T>` setelah deklarasi nama kelas. Hal serupa juga terjadi pada atribut `t` yang dideklarasikan bertipe generik yang secara otomatis membuat konstruktor dan method getter dan setter untuk atribut `t` di dalam kelas tersebut menjadi bertipe generik.

Kemudian pada main program terdapat instansiasi class `GenericsType` menjadi sebuah class namun ditambahkan dengan tipe generik `String` yang membuat class object hasil instansiasi memiliki atribut `t` bertipe `string`, method `set` dengan parameter bertipe `string`, dan method `get` yang akan mengembalikan nilai `string`. Maka dari itu saat object memanggil method `set` dengan mengisi nilai parameter dengan nilai `string` tidak terjadi error. Kemudian terdapat instansiasi class `GenericsType` kembali menjadi object, namun kali ini tanpa mendefinisikan tipe generiknya. Hal ini membuat atribut `t` beserta method `set` dan `get` dapat memiliki tipe apapun. Contoh pada line code dibawahnya terdapat pemanggilan method `set` pertama dengan mengisi parameter dengan nilai `string` dan pemanggilan method `set` kedua dengan mengisi parameter dengan nilai `integer` atau `number`. Hasilnya program dapat berjalan dan menghasilkan output dengan nilai `10` saat method `get` dipanggil. Namun, terdapat peringatan berupa

```
Type safety: The method set(Object) belongs to the raw type GenericsType. References to generic
type GenericsType<T> should be parameterized Java(16777747)
```

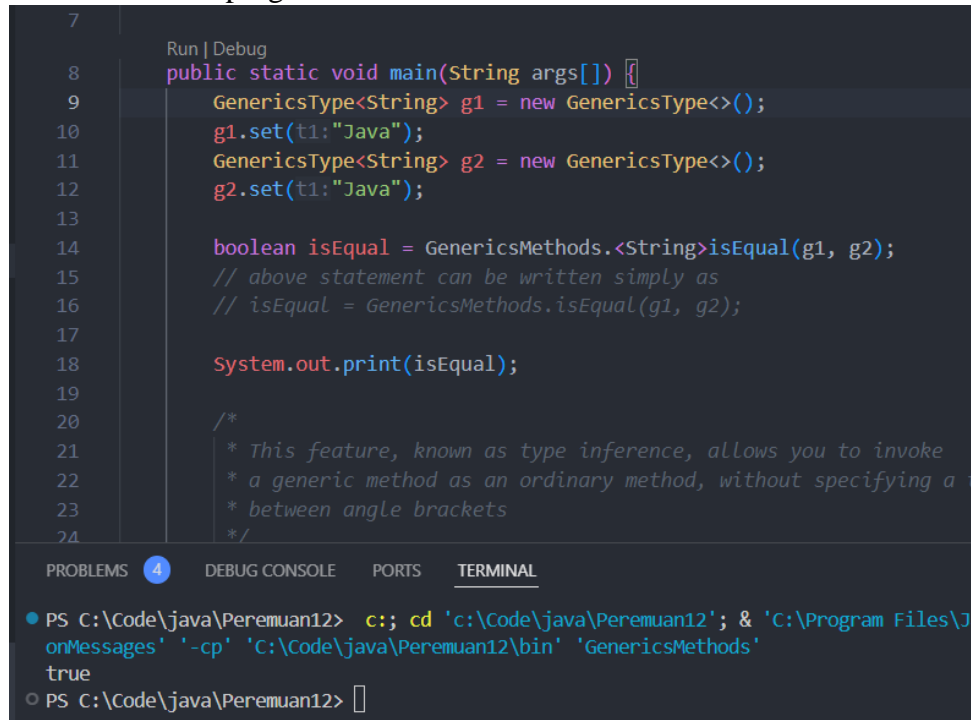
Yang mengindikasikan bahwa method tersebut menggunakan kelas generik tanpa menentukan tipe parametriknya, yang dapat mengurangi type safety atau keamanan

tipe. Hal ini dikarenakan kompilator tidak bisa memverifikasi tipe data yang di-set atau diambil dari objek tersebut.

- c. Masalah
- d. Solusi
- e. Teman Yang Membantu

### 3. GenericsMethods

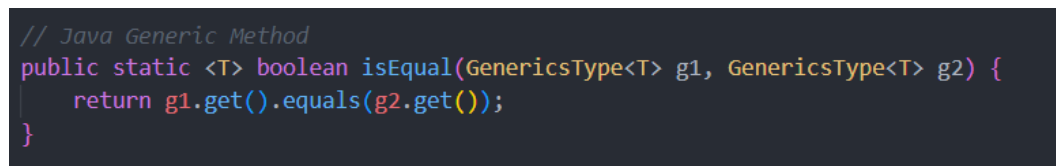
#### a. Screenshot hasil program



```
7
8      Run | Debug
9      public static void main(String args[]) {
10         GenericsType<String> g1 = new GenericsType<>();
11         g1.set(t1:"Java");
12         GenericsType<String> g2 = new GenericsType<>();
13         g2.set(t1:"Java");
14
15         boolean isEqual = GenericsMethods.<String>isEqual(g1, g2);
16         // above statement can be written simply as
17         // isEqual = GenericsMethods.isEqual(g1, g2);
18
19         System.out.print(isEqual);
20
21         /*
22          * This feature, known as type inference, allows you to invoke
23          * a generic method as an ordinary method, without specifying a
24          * between angle brackets
25          */
26     }
27 }
28
29 PROBLEMS 4 DEBUG CONSOLE PORTS TERMINAL
30
31 PS C:\Code\java\Peremuan12> c:: cd 'c:\Code\java\Peremuan12'; & 'C:\Program Files\J
32 onMessages' '-cp' 'C:\Code\java\Peremuan12\bin' 'GenericsMethods'
33 true
34 PS C:\Code\java\Peremuan12> }
```

#### b. Penjelasan

Pada class GenericsMethod terdapat method isEqual yang dideklarasikan sebagai berikut



```
// Java Generic Method
public static <T> boolean isEqual(GenericsType<T> g1, GenericsType<T> g2) {
    return g1.get().equals(g2.get());
}
```

Di sini, kita dapat melihat bahwa metode isEqual adalah metode generik, yang ditandai dengan sintaks <T> sebelum deklarasi method. Terdapat pula arameter g1 dan g2 bertipe GenericsType<T>, di mana T adalah tipe generik yang akan ditentukan saat metode ini dipanggil.

Kemudian terdapat main program terdapat instansiasi dua object dari class GenericsType dengan tipe generics string menjadi object g1 dan g2. Kemudian g1 dan g2 memanggil method set dan masing masing diisikan nilai parameternya dengan nilai string “java”. Kemudian terdapat pemanggilan method isEqual dari class GenericsMethod ini sebagai berikut

```
boolean isEqual = GenericsMethods.<String>isEqual(g1, g2);
// above statement can be written simply as
// isEqual = GenericsMethods.isEqual(g1, g2);
```

Disini kita bisa lihat bahwa pemanggilan method isEqual ini diikuti dengan pendefinisian tipe generic pada method tersebut yaitu string. Meskipun begitu, pemanggilan method ini dapat disimplifikasi menjadi GenericsMethods.isEqual(g1,g2) dengan hasil yang sama untuk masing masing cara pemanggilan yaitu true.

- c. Masalah
- d. Solusi
- e. Teman yang membantu

#### 4. Java Generic Interface

##### a. Screenshot hasil program

The screenshot shows an IDE with a Java file named Main.java. The code defines a public class Main with a static void main method. Inside main, it creates two arrays: Integer inums and Character chs. It then creates two objects, MyClass<Integer> a and MyClass<Character> b, using the arrays. Finally, it prints the result of a.max() and b.max(). The output in the terminal shows the number 8 and the character 'w'.

```
src > Main.java > Main > main(String[])
1 public class Main {
    Run | Debug
2     public static void main(String args[]) {
3         Integer inums[] = { 3, 6, 2, 8, 6 };
4         Character chs[] = { 'b', 'r', 'p', 'w' };
5         MyClass<Integer> a = new MyClass<Integer>(inums);
6         MyClass<Character> b = new MyClass<Character>(chs);
7         System.out.println(a.max());
8         System.out.println(b.max());
9     }
10 }
```

PROBLEMS 4 DEBUG CONSOLE PORTS TERMINAL

```
PS C:\Code\java\Peremuan12> & 'C:\Program Files\Java\jdk-22\bin\java.exe'
remuan12\bin' 'Main'
8
w
PS C:\Code\java\Peremuan12>
```

##### b. Penjelasan

Pada program Java Generic Interface ini terdapat sebuah interface dengan nama MinMx sebagai berikut

```
interface MinMax<T extends Comparable<T>> {
    T max(); /* w w w . java2 s . co m */
}
```

Disini kita bisa lihat bahwa interface ini merupakan interface generic ditandai dengan adanya syntax <T setelah nama interface yang mengekstend ke class Comparable yang merupakan sebuah generic class pula. Lalu pada interface ini terdapat method max dengan tipe T atau dapat kita sebut method generic.

Kemudian terdapat pula class MyClass yang mengimplementasikan interface MinMax ini

```
class MyClass<T extends Comparable<T>> implements MinMax<T> {
    T[] vals;

    MyClass(T[] o) {
        vals = o;
    }

    public T max() {
        T v = vals[0];
        for (int i = 1; i < vals.length; i++) {
            if (vals[i].compareTo(v) > 0) {
                v = vals[i];
            }
        }
        return v;
    }
}
```

Class MyClass disini juga merupakan class generic ditandai dengan syntax<T> seleah deklarasi nama class yang mengekstend ke class generic Comparable dan mengimplement generic interface MinMax. Pada class ini terdapat atribut generic array dengan nama vals beserta constructor dengan tipe parameter array generic. Kemudian terdapat method generic max() yang berfungsi mencari nilai terbesar dalam array vals dengan menggunakan method compateTo dari class comparable.

Kemudian pada main program terdapat instansiasi class MyClass ini menjadi 2 objek dengan tipe generic character dengan parameter variabel chs dengan Kumpulan nilai elemen karakter dan integer dengan parameter variable inums yang merupakan array dengan Kumpulan elemen integer. Yang kemudian masing masing objek memanggil method max yang menghasilkan output 8 dan w. Method max ini dapat dipanggil masing masing objek meskipun tanpa mendefinisikan tipe generiknya karena dia otomatis menentukan tipe methodnya berdasarkan input pada saat intansiasi class menjadi objek.

5. Java Bounded Class
  - a. Screenshot hasil program

```
src > J BoundedClass.java > BoundedClass > main(String[])
1 public class BoundedClass {
    Run | Debug
2     public static void main(String a[]) {
3
4         // Creating object of sub class C and
5         // passing it to Bound as a type parameter.
6         Bound<C> bec = new Bound<C>(new C());
7         bec.doRunTest();
8
9         // Creating object of sub class B and
10        // passing it to Bound as a type parameter.
11        Bound<B> beb = new Bound<B>(new B());
12        beb.doRunTest();
13
14        // similarly passing super class A
15        Bound<A> bea = new Bound<A>(new A());
16        bea.doRunTest();
17
18    }
19 }
```

PROBLEMS 4 DEBUG CONSOLE PORTS TERMINAL

```
PS C:\Code\java\Peremuan12> & 'C:\Program Files\Java\jdk-22\bin\java.exe'
remuan12\bin' 'BoundedClass'
Inside sub class C
Inside sub class B
Inside super class A
```

b. Penjelasan

Pada program ini terdapat class Bound yang merupakan class generik yang mengekstend ke class A

```
class Bound<T extends A> {
    private T objRef;

    public Bound(T obj) {
        this.objRef = obj;
    }

    public void doRunTest() {
        this.objRef.displayClass();
    }
}

class A {
    public void displayClass() {
        System.out.println(x:"Inside super class A");
    }
}
```

Class Bound memiliki atribut objRef yang merupakan atribut dengan tipe generic yang membuat otomatis constructor juga memiliki parameter dengan tipe generic. Kemudian terdapat method void doRunTest() yang akan memanggil method displayClass() dari atribut objRef.

Kemudian pada main program terdapat instansiasi 3 objek dari class Bound ini dengan implementasi sebagai berikut

```
public static void main(String a[]) {  
  
    // Creating object of sub class C and  
    // passing it to Bound as a type parameter.  
    Bound<C> bec = new Bound<C>(new C());  
    bec.doRunTest();  
  
    // Creating object of sub class B and  
    // passing it to Bound as a type parameter.  
    Bound<B> beb = new Bound<B>(new B());  
    beb.doRunTest();  
  
    // similarly passing super class A  
    Bound<A> bea = new Bound<A>(new A());  
    bea.doRunTest();  
}
```

Dimana masing masing objek merupakan instansiasi dengan tipe mendefinisikan tipe generic yaitu berupa class A, B, dan C dan parameter berupa object dari class C, B, dan A. Sehingga nilai yang bisa di assign atau dikirimkan pada parameter dibatasi tipenya yaitu berupa objek class dari class A, B, dan C. Kemudian masing masing objek menjalankan method doRunTest() yang dimana implementasi method ini adalah memanggil method displayClass yang ada pada masing masing class A, B, dan C melalui objek yang didismpn pada atribut class Bound.

- c. Masalah
  - d. Solusi
  - e. Teman yang membantu
6. Java Generic WildCard
- a. Screenshot hasil program



```
10
11 */
12 public class WildCardSimpleExample {
13     public static void printCollection(Collection<?> c) {
14         for (Object e : c) {
15             System.out.println(e);
16         }
17     }
18
19     Run | Debug
20     public static void main(String[] args) {
21         Collection<String> collection = new ArrayList<>();
22         collection.add(e:"ArrayList Collection");
23         printCollection(collection);
24         Collection<String> collection2 = new LinkedList<>();
25         collection2.add(e:"LinkedList Collection");
26         printCollection(collection2);
27         Collection<String> collection3 = new HashSet<>();
28
29 PROBLEMS 4 DEBUG CONSOLE PORTS TERMINAL
30
31 HashSet Collection
32 PS C:\Code\java\Peremuan12>
33
34 c:: cd 'c:\Code\java\Peremuan12'; & 'C:\Program File
35 onMessages' '-cp' 'C:\Code\java\Peremuan12\bin' 'WildCardSimpleExample'
36 ArrayList Collection
37 LinkedList Collection
38 HashSet Collection
39 PS C:\Code\java\Peremuan12>
```

b. Penjelasan

Pada class WildCardSimpleExample terdapat method static printCollection yang memiliki parameter Collection dengan tipe *unbounded wildcard* ditandai dengan syntax <?>. Deklarasi parameter generic dengan sintaks <?> disebut sebagai *unbounded wildcard* dalam bahasa pemrograman Java. Ini berarti bahwa tipe yang diizinkan untuk parameter tersebut bisa merupakan tipe apapun, tanpa batasan spesifik. Selain itu apabila kita mendefinisikan *unbounded wildcard* pada suatu method sebagai contoh pada method printCollection() kita tidak dapat menambahkan element baru koleksi yang dikirimkan melalui parameter sebagai berikut

```
public static void printCollection(Collection<?> c) {
    c.add(e:"asdasdasdas");
    for (Object e : c) {
        System.out.println(e);
    }
}
```

Disini kita bisa lihat bahwa apabila kita mencoba menambah elemen koleksi di dalam method printCollection maka akan muncul peringatan error yang menyatakan bahwa

```
The method add(capture#1-of ?) in the type Collection<capture#1-of ?> is not applicable for the arguments (String) Java(67108979)
boolean java.util.Collection.add(Object e)
Ensures that this collection contains the specified element (optional operation). Returns true if this collection changed as a result of the call. (Returns false if this collection does not permit duplicates and already contains the specified element.)
Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. Collection classes should clearly specify in their documentation any restrictions on what elements may be added.
If a collection refuses to add a particular element for any reason other than that it already contains the element, it must
```

Kemudian pada main method, terdapat tiga jenis koleksi berbeda didefinisikan: ArrayList, LinkedList, dan HashSet, masing-masing bertipe String. Setiap koleksi ditambahkan satu elemen teks yang mencantumkan jenis koleksinya. Kemudian, metode printCollection dipanggil untuk masing-masing koleksi, yang mencetak elemen-elemen dari ketiga koleksi ini secara terpisah.

- c. Masalah
- d. Solusi
- e. Nama teman yang membantu