


Pemrograman Berbasis Objek

# GENERICIS

Yuliana Setiowati

Politeknik Elektronika Negeri Surabaya




Pemrograman Berbasis Objek

# Topik

- Definisi Generics
- Non Generic Class
- Cara mendeklarasikan Type Generics
- Aturan Penamaan Type Parameter
- Subtyping
- Class Generic dengan Dua Type Parameter
- Generics pada List
- Nested generic type
- Raw Types
- Generics and Collections
- Menggunakan Interface Generic Comparable
- Type parameter yang dibatasi
- Menggunakan ? Wildcard
- Menggunakan Bounded Wildcard dalam Method

Politeknik Elektronika Negeri Surabaya


2


Pemrograman Berbasis Objek

## Generics

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

Politeknik Elektronika Negeri Surabaya


Pemrograman Berbasis Objek

## Non Generic Class

```

public class Box {
    private Object object;

    public void add(Object object) {
        this.object = object;
    }

    public Object get() {
        return object;
    }
}

```

Buatlah 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 BoxDemol {
    public static void main(String[] args) {
        // ONLY place Integer objects into this box!
        Box integerBox = new Box();

        integerBox.add(new Integer(10));
        Integer someInteger = (Integer)integerBox.get();
        System.out.println(someInteger);
    }
}

```

Contoh object Box yaitu integerBox diberikan object Integer, pada saat mengambil harus diubah menjadi Integer

Politeknik Elektronika Negeri Surabaya


**Pemrograman Berbasis Objek**

```

public class BoxDemo2 {
    public static void main(String[] args) {

        // ONLY place Integer objects into this box!
        Box integerBox = new Box();


        // Imagine this is one part of a large application
        // modified by one programmer.
        integerBox.add("10"); // note how the type is now String

        // ... and this is another, perhaps written
        // by a different programmer
        Integer someInteger = (Integer)integerBox.get();
        System.out.println(someInteger);
    }
}

Exception in thread "main"
    java.lang.ClassCastException:
        java.lang.String cannot be cast to java.lang.Integer
        at BoxDemo2.main(BoxDemo2.java:6)

```

Politeknik Elektronika Negeri Surabaya


**Pemrograman Berbasis Objek**


## Penjelasan

- Mengapa Error ?  
 Pada object IntegerBox dimasukkan object 10 tapi dengan tipe String, tapi pada saat mengambil object, diubah menjadi tipe Integer. Tipe data tidak sesuai sehingga error

Politeknik Elektronika Negeri Surabaya

6

Pemrograman Berbasis Objek



## Non Generic Class

- Permasalahan ?
- no homogeneous collections
  - memerlukan banyak casting
- Tidak ada pengecekan pada saat compile, kesalahan baru bisa terdeteksi pada saat runtime.

```

LinkedList list = new LinkedList();
list.add(new Integer(0));
Integer i = (Integer) list.get(0);
String s = (String) list.get(0);


```

casts required

fine at compile-time,  
but fails at runtime

Politeknik Elektronika Negeri Surabaya

Pemrograman Berbasis Objek



## Cara mendeklarasikan Type Generics

- Ubah class Box menggunakan generics
- Pendeklarasian type generics dengan mengubah public class Box → 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.

```

/**
 * Generic version of the Box class.
 */
public class Box<T> {

    private T t; // T stands for "Type"

    public void add(T t) {
        this.t = t;
    }

    public T get() {
        return t;
    }
}

```

Membuat Object

```

Box<Integer> integerBox;


integerBox = new Box<Integer>();

atau

Box<Integer> integerBox = new Box<Integer>();

```

Negeri Surabaya


Pemrograman Berbasis Objek

## Cara mendeklarasikan Type Generics


- Tidak perlu proses casting pada saat menggunakan fungsi get().

```
public class BoxDemo3 {
    public static void main(String[] args) {
        Box<Integer> integerBox = new Box<Integer>();
        integerBox.add(new Integer(10));
        Integer someInteger = integerBox.get(); // no cast!
        System.out.println(someInteger);
    }
}
```

- Tapi jika kita menambahkan ke box dengan tipe yang tidak sesuai (misal : String) maka pada saat di compile akan mengalami error.

```
BoxDemo3.java:5: add(java.lang.Integer) in Box<java.lang.Integer>
cannot be applied to (java.lang.String)
    integerBox.add("10");
                  ^
1 error
```

Politeknik Elektronika Negeri Surabaya


Pemrograman Berbasis Objek

## 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

Politeknik Elektronika Negeri Surabaya

Pemrograman Berbasis Objek

```

class GenericClass<T> {
    T ob;

    GenericClass(T o) {
        ob = o;
    }

    T getob() {
        return ob;
    }

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

public class MainClass {
    public static void main(String args[]) {
        // Create a Gen reference for Integers.
        GenericClass<Integer> iOb = new GenericClass<Integer>(88);
        iOb.showType();

        // no cast is needed.
        int v = iOb.getob();
        System.out.println("value: " + v);

        // Create a Gen object for Strings.
        GenericClass<String> strOb = new GenericClass<String>("Generics Test");
        strOb.showType();

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

```

```

Type of T is java.lang.Integer
value: 88
Type of T is java.lang.String
value: Generics Test

```

Pemrograman Berbasis Objek

## Subtyping

```

public static void main(String[] args) {
    Box<String> boxS = new Box<String>();
    Box<Object> boxO = new Box<Object>();
    boxS.add("abc");
    boxO.add(new Object());


    Object A=null ;
    String B="" ;
    A = B ;
    boxO = boxS ;
}

```

- G<Object> gA ; }
- G<String> gB;
- gA = gB // error karena G<String> bukan subtype dari G<Object>
- Secara umum jika B adalah dari A dan G adalah suatu tipe data generics maka tidak berarti G<B> adalah subtype dari G<A>.

Politeknik Elektronika Negeri Surabaya 12

Pemrograman Berbasis Objek



## 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 MainClass {
    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);
    }
}

```


```

Type of T is java.lang.Integer
Type of V is java.lang.String
value: 88
value: Generics

```

Fakultas Teknik Elektronika Negeri Surabaya

Pemrograman Berbasis Objek



## Class Generic dengan Dua Type Parameter

```

class Pair<KeyType, ValueType> {
    // Constructor
    public Pair(KeyType aKey, ValueType aValue) {
        key = aKey;
        value = aValue;
    }

    // Get the key for this pair
    public KeyType getKey() {
        return key;
    }

    // Get the value for this pair
    public ValueType getValue() {
        return value;
    }


    // Set the value for this pair
    public void setValue(ValueType aValue) {
        value = aValue;
    }

    private KeyType key;

    private ValueType value;
}

```

abaya




Pemrograman Berbasis Objek

## Class Generic dengan Dua Type Parameter

```
public class MainClass {  
  
    public static void main(String[] a) {  
        Pair<Integer, String> p = new Pair<Integer, String>(1, "A");  
  
        System.out.println(p.getKey().getClass().getName());  
    }  
}
```

`java.lang.Integer`

Politeknik Elektronika Negeri Surabaya



Pemrograman Berbasis Objek


## Generics pada List

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

Politeknik Elektronika Negeri Surabaya



Pemrograman Berbasis Objek



## Generics List

```

import java.util.ArrayList;
import java.util.List;

public class MainClass {
    public static void main(String[] args) {
        List stringList1 = new ArrayList();
        stringList1.add("Java 5");
        stringList1.add("with generics");

        String s1 = (String) stringList1.get(0);
        System.out.println(s1.toUpperCase());

        List<String> stringList2 = new ArrayList<String>();
        stringList2.add("Java 5");
        stringList2.add("with generics");


        String s2 = stringList2.get(0);
        System.out.println(s2.toUpperCase());
    }
}

```

JAVA 5  
 JAVA 5

Politeknik Elektronika Negeri Surabaya

Pemrograman Berbasis Objek



## Nested generic type

- A generic type is itself a type and can be used as a type variable
 

```
List<List<String>> myListOfListsOfStrings;
```
- Cara untuk mendapatkan string dari list pertama :
 

```
String s = myListOfListsOfStrings.get(0).get(0);
```


```

import java.util.ArrayList;
import java.util.List;

public class MainClass {
    public static void main(String[] args) {
        List<String> listOfStrings = new ArrayList<String>();
        listOfStrings.add("Hello again");
        List<List<String>> listOfLists = new ArrayList<List<String>>();
        listOfLists.add(listOfStrings);
        String s = listOfLists.get(0).get(0);
        System.out.println(s); // prints "Hello again"
    }
}

```

Pemrograman Berbasis Objek



## Type Generic dapat menerima lebih dari satu type variabel

```


import java.util.HashMap;
import java.util.Map;

public class MainClass {
    public static void main (String[] args) {
        Map<String, String> map = new HashMap<String, String>();
        map.put ("key1", "value1");
        map.put ("key2", "value2");
        String value1 = map.get("key1");
    }
}

```

Politeknik Elektronika Negeri Surabaya

Pemrograman Berbasis Objek



## Raw Types

- Java membolehkan class generic digunakan tanpa type argument.

```

// Demonstrate a raw type.
class Gen<T o> {
    T ob;

    Gen(T o) {
        ob = o;
    }

    T getob() {
        return ob;
    }
}

public class MainClass {
    public static void main(String args[]) {
        Gen<Integer> iOb = new Gen<Integer>(88);
        Gen<String> strOb = new Gen<String>("Generics Test");

        Gen raw = new Gen(new Double(98.6));

        // Cast here is necessary because type is unknown.
        double d = (Double) raw.getob();
        System.out.println("value: " + d);

        strOb = raw; // OK, but potentially wrong
        String str = strOb.getob();

        // This assignment also overrides type safety.
        raw = iOb; // OK, but potentially wrong
        d = (Double) raw.getob();
    }
}

```


```

Exception in thread "main" value: 98.6
java.lang.ClassCastException: java.lang.Double
at MainClass.main(MainClass.java:26)

```

Politeknik Elektronika Negeri Surabaya

Pemrograman Berbasis Objek



## Generics and Collections: ArrayList

```

import java.util.ArrayList;
import java.util.Iterator;

public class MainClass {
    public static void main(String args[]) {
        ArrayList<String> list = new ArrayList<String>();
        list.add("one");
        list.add("two");
        list.add("three");
        list.add("four");

        Iterator<String> itr = list.iterator();

        while(itr.hasNext()) {
            String str = itr.next();

            System.out.println(str + " is " + str.length() + " chars long.");
        }
    }
}

```


```

one is 3 chars long.
two is 3 chars long.
three is 5 chars long.
four is 4 chars long.

```

Surabaya

Pemrograman Berbasis Objek



## Array

```

class Person {
    private String lastName;
    private String firstName;

    private int age;

    public Person(String last, String first, int a) {
        lastName = last;
        firstName = first;
        age = a;
    }


    public String toString() {
        return "Last name: " + lastName + " First name: " + firstName + " Age: " + age;
    }

    public String getLast() {
        return lastName;
    }
}

```

Politeknik Elektronika Negeri Surabaya

Pemrograman Berbasis Objek



# Array

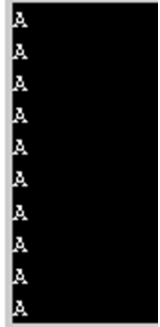
```

public class MainClass {
    public static void main(String[] args) {
        Person[] persons = new Person[10];

        for(int i=0;i<persons.length;i++){
            persons[i] = new Person("A", "B", 10);
        }


        for(Person p: persons){
            System.out.println(p.getLast());
        }
    }
}

```



Politeknik Elektronika Negeri Surabaya

Pemrograman Berbasis Objek



# Menggunakan Interface Generic Comparable

```

import java.util.Arrays;

class Person implements Comparable<Person> {
    public Person(String firstName, String surname) {
        this.firstName = firstName;
        this.surname = surname;
    }

    public String toString() {
        return firstName + " " + surname;
    }

    public int compareTo(Person person) {
        int result = surname.compareTo(person.surname);
        return result == 0 ? firstName.compareTo(((Person) person).firstName) : result;
    }

    private String firstName;

    private String surname;
}

```

Class Person terdapat dua variabel yaitu firstName dan surname, untuk mengurutkan data maka class Person harus mengimplementasikan interface Comparable dan mengimplementasikan method compareTo().  
Data2 pada class Person diurutkan berdasarkan surname, tetapi jika surname sama maka data diurutkan berdasarkan firstName

Politeknik Elektronika Negeri Surabaya

**Pemrograman Berbasis Objek**

Memasukan beberapa object Person pada object array author, selanjutnya data pada array tersebut diurutkan berdasarkan surname

```

public class MainClass {
    public static void main(String[] args) {
        Person[] authors = {
            new Person("D", "S"),
            new Person("G", "G"),
            new Person("T", "C"),
            new Person("C", "S"),
            new Person("P", "C"),
            new Person("B", "B") };

        Arrays.sort(authors); // Sort using Comparable method

        System.out.println("\nOrder after sorting into ascending sequence:");
        for (Person author : authors) {
            System.out.println(author);
        }

        Person[] people = {
            new Person("C", "S"),
            new Person("N", "K"),
            new Person("T", "C"),
            new Person("C", "D") };

        int index = 0;
        System.out.println("\nIn search of authors:");

        for (Person person : people) {
            index = Arrays.binarySearch(authors, person);
            if (index >= 0) {
                System.out.println(person + " was found at index position " + index);
            } else {
                System.out.println(person + "was not found. Return value is " + index);
            }
        }
    }
}
  
```

```

Order after sorting into ascending sequence:
B B
P C
T C
J G
C S
D S

In search of authors:
C S was found at index position 4
N K was not found. Return value is -5
T C was found at index position 2
C D was not found. Return value is -4
  
```

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

**Pemrograman Berbasis Objek**

## Type parameter yang dibatasi

- Jika kita ingin memberikan batasan type yang diperbolehkan untuk dilewatkan ke type parameter. Contoh method dengan parameter number, hanya menerima object dari class Number dan subclass. Hal ini yang disebut *bounded type parameter*.
- Cara
 

```
<U extends Number>
```
- Jika terdapat interface yang harus diimplementasikan gunakan &
 

```
<U extends Number & MyInterface>
```

Politeknik Elektronika Negeri Surabaya

Pemrograman Berbasis Objek

```

public class Box<T> {
    private T t;

    public void add(T t) {
        this.t = t;
    }

    public T get() {
        return t;
    }

    public <U extends Number> void inspect(U u){
        System.out.println("T: " + t.getClass().getName());
        System.out.println("U: " + u.getClass().getName());
    }

    public static void main(String[] args) {
        Box<Integer> integerBox = new Box<Integer>();
        integerBox.add(new Integer(10));
        integerBox.inspect("some text"); // error: this is still String!
    }
}

```

Yang menjadi parameter dari method inspect() adalah semua object yang merupakan anak dari class Number

```

Box.java:21: <U>inspect(U) in Box<java.lang.Integer> cannot
    be applied to (java.lang.String)
        integerBox.inspect("10");
                        ^
1 error

```

Politeknik Elektronika Negeri Surabaya

Pemrograman Berbasis Objek

## Generic Method

```

public class MainClass {
    static <T, V extends T> boolean isIn(T x, 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, 4, 5 };

        if (isIn(2, nums)) {
            System.out.println("2 is in nums");
        }
        if (!isIn(7, nums)) {
            System.out.println("7 is not in nums");
        }

        // Use isIn() on Strings.
        String strs[] = { "one", "two", "three", "four", "five" };

        if (isIn("two", strs))
            System.out.println("two is in strs");

        if (!isIn("seven", strs))
            System.out.println("seven is not in strs");
    }
}


```

```

2 is in nums
7 is not in nums
two is in strs
seven is not in strs

```

Pemrograman Berbasis Objek



## Generic Constructor

```

class GenericClass {
    private double val;

    <T extends Number> GenericClass(T arg) {
        val = arg.doubleValue();
    }

    void showValue() {
        System.out.println("val: " + val);
    }
}

public class MainClass {
    public static void main(String args[]) {

        GenericClass test = new GenericClass(100);
        GenericClass test2 = new GenericClass(123.5F);

        test.showValue();
        test2.showValue();
    }
}

```


Yang menjadi parameter dari method inspect() adalah semua object yang merupakan anak dari class Number misal object Integer, Float, Double

```

val: 100.0
val: 123.5

```

Pemrograman Berbasis Objek



```

public class MainClass {
    // generic method printArray
    public static <E> void printArray(E[] inputArray) {
        // display array elements
        for (E element : inputArray)
            System.out.printf("%s ", element);

        System.out.println();
    }

    public static void main(String args[]) {
        // create arrays of Integer, Double and Character
        Integer[] integerArray = { 1, 2, 3, 4, 5, 6 };
        Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7 };
        Character[] characterArray = { 'H', 'E', 'L', 'L', 'O' };


        System.out.println("Array integerArray contains:");
        printArray(integerArray); // pass an Integer array
        System.out.println("\nArray doubleArray contains:");
        printArray(doubleArray); // pass a Double array
        System.out.println("\nArray characterArray contains:");
        printArray(characterArray); // pass a Character array
    } // end main
}

```

```

Array integerArray contains:
1 2 3 4 5 6

```


Pemrograman Berbasis Objek

```

class Stats<T extends Number> {
    T[] nums;

    Stats(T[] o) {
        nums = o;
    }

    double average() {
        double sum = 0.0;

        for(int i=0; i < nums.length; i++)
            sum += nums[i].doubleValue();

        return sum / nums.length;
    }
}

public class MainClass {
    public static void main(String args[]) {

        Integer inums[] = { 1, 2, 3, 4, 5 };
        Stats<Integer> iob = new Stats<Integer>(inums);
        double v = iob.average();
        System.out.println("iob average is " + v);


        Double dnums[] = { 1.1, 2.2, 3.3, 4.4, 5.5 };
        Stats<Double> dob = new Stats<Double>(dnums);
        double w = dob.average();
        System.out.println("dob average is " + w);
    }
}

```

## Type parameter yang dibatasi

```
iob average is 3.0
dob average is 3.3
```

Surabaya


Pemrograman Berbasis Objek

## Menggunakan ? Wildcard

- Jika kita mendeklarasikan sebuah List<aType>, isi List adalah object dengan tipe aType, maka kita bisa menyimpan di List object dengan tipe :
  - Object dari aType
  - Object dari subclass aType, jika aType adalah class
  - Object dari class yang mengimplementasikan aType, jika aType adalah interface

Politeknik Elektronika Negeri Surabaya



**Pemrograman Berbasis Objek**

 `List<Object> list1 = new ArrayList<Object>();`  
`List<String> list2 = new ArrayList<String>();`

list1 mengacu ke List dengan tipe `java.lang.Object` dan list2 mengacu ke List String. Meskipun String merupakan subclass dari Object, List<String> tidak ada hubungannya dengan List<Object>, sehingga melewatkan List<String> ke method yang memiliki parameter List<Object> akan menyebabkan **error**.

```
package com.brainysoftware.jdk5.app16;
import java.util.ArrayList;
import java.util.List;

public class AllowedTypeTest {
    public static void doit(List<Object> l) {
    }
    public static void main(String[] args) {
        List<String> myList = new ArrayList<String>();
        // this will generate a compile error
        doit(myList);
    }
}
```

Politeknik Elektronika Negeri Surabaya

**Pemrograman Berbasis Objek**

 **Menggunakan ? Wildcard**

- Bagaimana penyelesaiannya ?
- Menggunakan ? Wildcard. List<?> berarti list dengan object tipe sembarang.

```
public static void doit(List<?> l) {
}


package com.brainysoftware.jdk5.app16;
import java.util.ArrayList;
import java.util.List;

public class wildCardTest {
    public static void printList(List<?> list) {
        for (Object element : list) {
            System.out.println(element);
        }
    }
    public static void main(String[] args) {
        List<String> list1 = new ArrayList<String>();
        list1.add("Hello");
        list1.add("world");
        printList(list1);

        List<Integer> list2 = new ArrayList<Integer>();
        list2.add(100);
        list2.add(200);
        printList(list2);
    }
}
```

```
Hello
World
100
200
```

Pemrograman Berbasis Objek



## Menggunakan ? Wildcard

- Illegal menggunakan wildcard pada saat create type generic.


```
List<?> myList = new ArrayList<?>(); // illegal
```

- Jika list bisa menerima sembarang object, gunakan class Object sebagai type variabel.

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

Politeknik Elektronika Negeri Surabaya

Pemrograman Berbasis Objek



## Menggunakan ? Wildcard

```
// Use a wildcard.
class GenericStats<T extends Number> {
    T[] nums;

    GenericStats(T[] o) {
        nums = o;
    }

    double average() {
        double sum = 0.0;
        for(int i=0; i < nums.length; i++){
            sum += nums[i].doubleValue();
        }
        return sum / nums.length;
    }

    boolean sameAvg(GenericStats<?> ob) {
        if(average() == ob.average())
            return true;

        return false;
    }
}
```

Pemrograman Berbasis Objek

```

public class MainClass {
    public static void main(String args[]) {
        Integer inums[] = { 1, 2, 3, 4, 5 };
        GenericStats<Integer> iob = new GenericStats<Integer>(inums);
        double v = iob.average();
        System.out.println("iob average is " + v);

        Double dnums[] = { 1.1, 2.2, 3.3, 4.4, 5.5 };
        GenericStats<Double> dob = new GenericStats<Double>(dnums);
        double w = dob.average();
        System.out.println("dob average is " + w);

        Float fnums[] = { 1.0F, 2.0F, 3.0F, 4.0F, 5.0F };
        GenericStats<Float> fob = new GenericStats<Float>(fnums);
        double x = fob.average();
        System.out.println("fob average is " + x);

        System.out.print("Averages of iob and dob ");
        if(iob.sameAvg(dob)){
            System.out.println("are the same.");
        }else{
            System.out.println("differ.");
        }

        System.out.print("Averages of iob and fob ");
        if(iob.sameAvg(fob)){
            System.out.println("are the same.");
        }else{
            System.out.println("differ.");
        }
    }
}

```

```

iob average is 3.0
dob average is 3.3
fob average is 3.0
Averages of iob and dob differ.
Averages of iob and fob are the same.

```

Pemrograman Berbasis Objek


## Menggunakan Bounded Wildcard dalam Method

- Sintak upper bound untuk wildcard:  
**GenericType <? extends upperBoundType>**
- Contoh :  
**List <? extends Number>**
- Berarti kita bisa melewati object dari List<Integer>, List<Double>, List<Float>
- Sintak lower bound untuk wildcard:  
**GenericType <? super Integer>**

berarti kita bisa melewati List<Integer> atau list yang berisi object yang merupakan superclass dari class Integer.

Politeknik Elektronika Negeri Surabaya

Pemrograman Berbasis Objek



## Menggunakan Bounded Wildcard dalam Method

```


package com.brainysoftware.jdk5.app16;
import java.util.ArrayList;
import java.util.List;
public class BoundedwildcardTest {
    public static double getAverage(List<? extends Number> numberList)
    {
        double total = 0.0;
        for (Number number : numberList)
            total += number.doubleValue();
        return total/numberList.size();
    }

    public static void main(String[] args) {
        List<Integer> integerList = new ArrayList<Integer>();
        integerList.add(3);
        integerList.add(30);
        integerList.add(300);
        System.out.println(getAverage(integerList)); // 111.0
        List<Double> doubleList = new ArrayList<Double>();
        doubleList.add(3.0);
        doubleList.add(33.0);
        System.out.println(getAverage(doubleList)); // 18.0
    }
}

```

111.0  
18.0

Pemrograman Berbasis Objek



## Bounded Wildcards

```

class Two {
    int x, y;

    Two(int a, int b) {
        x = a;
        y = b;
    }
}

class Three extends Two {
    int z;

    Three(int a, int b, int c) {
        super(a, b);
        z = c;
    }
}

class Four extends Three {
    int t;

    Four(int a, int b, int c, int d) {
        super(a, b, c);
        t = d;
    }
}


class Gen<T extends Two> {
    T[] coords;

    Gen(T[] o) {
        coords = o;
    }
}

```

a Negeri Surabaya

Pemrograman Berbasis Objek



## Bounded Wildcards

```

public class MainClass {
    static void showTwo(Gen<?> c) {
        System.out.println("X Y Coordinates:");
        for (int i = 0; i < c.coords.length; i++)
            System.out.println(c.coords[i].x + " " + c.coords[i].y);
        System.out.println();
    }


    static void showThree(Gen<? extends Three> c) {
        System.out.println("X Y Z Coordinates:");
        for (int i = 0; i < c.coords.length; i++)
            System.out.println(c.coords[i].x + " " + c.coords[i].y + " " + c.coords[i].z);
        System.out.println();
    }

    static void showAll(Gen<? extends Four> c) {
        System.out.println("X Y Z T Coordinates:");
        for (int i = 0; i < c.coords.length; i++)
            System.out.println(c.coords[i].x + " " + c.coords[i].y + " " + c.coords[i].z + " "
                + c.coords[i].t);
        System.out.println();
    }
}

```

Politeknik Elektronika Negeri Surabaya

Pemrograman Berbasis Objek



## Bounded Wildcards

```

public static void main(String args[]) {
    Two td[] = { new Two(0, 0), new Two(7, 9), new Two(18, 4), new Two(-1, -23) };

    Gen<Two> tdlocs = new Gen<Two>(td);

    System.out.println("Contents of tdlocs.");
    showTwo(tdlocs); // OK, is a TwoD

    Four fd[] = { new Four(1, 2, 3, 4), new Four(6, 8, 14, 8), new Four(22, 9, 4, 9),
        new Four(3, -2, -23, 17) };

    Gen<Four> fdlocs = new Gen<Four>(fd);

    System.out.println("Contents of fdlocs.");
    // These are all OK.
    showTwo(fdlocs);
    showThree(fdlocs);
    showAll(fdlocs);
}
}

```

Contents of tdlocs.	X Y Z Coordinates:
X Y Coordinates:	1 2 3
0 0	6 8 14
7 9	22 9 4
18 4	3 -2 -23
-1 -23	
Contents of fdlocs.	X Y Z T Coordinates:
X Y Coordinates:	1 2 3 4
1 2	6 8 14 8
6 8	22 9 4 9
22 9	3 -2 -23 17
3 -2	

Politeknik Elektronika Negeri Surabaya