

<Code Samples>

General

```
class Container<T> { T value; }
public class GenericsDemo {
    public static void main(String[] args) {
        Container<Integer> obj = new Container<>();
        obj.value = 1;
    }
}
```

//what if we want to fill the "T" with
Number class only?

Limited

```
class Container<T extends Number> {
    T value; }
public class GenericsDemo {
    public static void main(String[] args) {
        Container<Integer> obj = new Container<>();
        Container<Double> obj2 = new Container<>();
        obj2.value = 1.0;

        //Container<String> obj3 = new
            Container<>();
        //Compile Error if uncomment
    }
}
```



<Benefits>

Integer Boolean
String Character

Double
Object
Short
Float
Long
Byte
Type

No Need
For
Individual
Type
Casting

ArrayList Method
HashMap
Class
Attribute

Applicable
to ALL
Algorithms



Code
Reuse

Compile-time
Safety



```
public <T> void safety(
    T obj, T obj2) {
    ArrayList<T> lst =
        new ArrayList<>();
    lst.add(obj);
    lst.add(obj2);
    //Any object can be
        added (No Error)
}
```

<Timeline>



Java
7.0

```
ArrayList<ArrayList
<String>> good = new
ArrayList<>();
```

Birth of
diamond
operator
<>

Generics
was
introduced

```
ArrayList<ArrayList
<String>> mess = new
ArrayList<ArrayList
<String>>();
//REPETITIVE
```

old
Java

No generics
yet

```
ArrayList whatever = new ArrayList();
whatever.add(1);
int obj = whatever.get(0);
//BAD CODE
```

Java
5.0



Generics

do not exist at runtime,
but is a compile-time concept



allows users to write
a MORE COMMON,



LESS WORDY code

that is EASIER to
read and maintain

General

receives any type arguments

e.g. ArrayList<T>



Limited / Parameterized

receives fixed argument according
to the type parameter

e.g. ArrayList<Integer>



T
y
p
e
s

GLEND
E SUTANT

HARNINDYTO
WICAKSANA

WULAN
MANTIRI

EXTENDS

DR. FARIZ
DARARI

IMPLEMENTS



FACULTY OF
COMPUTER
SCIENCE

