

Let's say we want to
print all elements in our **arrays**.

`Boolean [] bool = { false, true, false }`

`Integer [] num = { 1, 2, 3, 4, 5, 6 }`

`String [] fakultas = { "fasilkom", "fib", "feb", "fk" }`

```
public static void printbol(Boolean[] arr){
    for (Boolean bool : arr){
        System.out.println(bool);
    }
}
```

WE MUST
IMPLEMENT
ALL OF THESE
METHODS!

```
public static void printint(Integer[] arr){
    for (Integer num : arr){
        System.out.println(num);
    }
}
```

```
public static void printstr(String[] arr){
    for (String fakultas : arr){
        System.out.println(fakultas);
    }
}
```

what if we have **1000 types** of objects?

do we have to implement 1000 methods? **NO!** just use:

Generics is a type or method to operate on objects of various types while providing compile-time type safety.

It was introduced in JDK 5.0 with the aim of reducing bugs and adding an extra layer of abstraction over types.

GENERICS

those methods can be replaced by:

```
public static <T> void printdir(T[] arr){
    for (T element:arr){
        System.out.println(element);
    }
}
```

⚠ All instructions inside should be compatible to **all types of object**. Don't use any type's specific method

ex:

- `object.replaceAll()` only can be applied to `String`

⚠ Generics only allow **non-primitive** data type!

Directed by: **Fariz Darari, S.Kom, M.Sc., Ph.D.**