# Dasar-Dasar Pemrograman 2: Generics

**Tim DDP 2 Fasilkom UI**
**Correspondence: Fariz Darari (fariz@cs.ui.ac.id)**

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia
FAKULTAS ILMU KOMPUTER

# Why?

Try this one:

Create a method to print each element of ArrayList of Integers vertically.

# Why?

Try this one:

Create a method to print each element of ArrayList of Strings vertically.

# Why?

Try this one:

Create a method to print each element of ArrayList of Doubles vertically.

# All the methods are similar, the only difference is the element type!

```
public static void printInt(ArrayList<Integer> lst) {
    for(int i = 0; i < lst.size(); i++) {
        System.out.println(lst.get(i));
    }
}

public static void printStr(ArrayList<String> lst) {
    for(int i = 0; i < lst.size(); i++) {
        System.out.println(lst.get(i));
    }
}

public static void printDouble(ArrayList<Double> lst) {
    for(int i = 0; i < lst.size(); i++) {
        System.out.println(lst.get(i));
    }
}
```

# Generics parameterize types!

```
public static <T> void print(ArrayList<T> lst) {
    for(int i = 0; i < lst.size(); i++) {
        System.out.println(lst.get(i));
    }
}
```

# Generics parameterize types!

```
public static <T> void print(ArrayList<T> lst) {
    for(int i = 0; i < lst.size(); i++) {
        System.out.println(lst.get(i));
    }
}
```

Generics are variables, but for types.
They can be used for methods and classes!

# Quiz time: Create a generic method to print any object with "Hello, " + object

```
hello("Hello");
hello(new Integer(5));
String[] strArr = {"1", "2", "3"};
hello(new ArrayList<String>(Arrays.asList(strArr)));
```

**Output:**
Hello, Hello
Hello, 5
Hello, [1, 2, 3]

# Quiz time: Create a generic method to print any object with "Hello, " + object

```
public static <T> void hello(T obj) {
        System.out.println("Hello, " + obj);
}
```

# Quiz time: Create a generic method to print any object with "Hello, " + object

```
public static <Bla> void hello(Bla obj) {
        System.out.println("Hello, " + obj);
}
```

The naming of the generics does not have to be T.

# Quiz time: Create a generic method to print any object with "Hello, " + object

```
 public static <Bla> void hello(Bla obj) {
        System.out.println("Hello, " + obj + " of type "
+ obj.getClass().getName());
 }
```

Print the object's type

# Generics for classes

- **Without generics for ArrayList**

```
ArrayList ageList = new ArrayList();
ageList.add(new Integer(46));
ageList.add("50");

Integer sum = new Integer(0);
for (Object age : ageList) {
    sum = sum + ((Integer) age);
}
```

# Generics for classes

- **Without generics for ArrayList**

```
ArrayList ageList = new ArrayList();
ageList.add(new Integer(46
ageList.add("50");

Integer sum = new Integer(0);
for (Object age : ageList) {
    sum = sum + ((Integer) age);
}
```

**RUNTIME ERROR:**
**Exception in thread "main" java.lang.ClassCastException: class java.lang.String cannot be cast to class java.lang.Integer**

# Generics for classes

- **With generics for ArrayList**

```
ArrayList<Integer> ageList = new ArrayList<Integer>();
ageList.add(new Integer(46));
ageList.add("50");

Integer sum = new Integer(0);
for (Object age : ageList) {
    sum = sum + ((Integer) age);
}
```

# Generics for classes

- **With generics for ArrayList**

```
ArrayList<Integer> ageList = new ArrayList<Integer>();
ageList.add(new Integer(46
ageList.add("50");

Integer sum = new Integer(0);
for (Object age : ageList) {
    sum = sum + ((Integer) age);
}
```

**COMPILE TIME ERROR:**
**java: incompatible types: java.lang.String cannot be converted to java.lang.Integer**

# Generics for classes

- **With generics for ArrayList**

```
ArrayList<Integer> ageList = new ArrayList<Integer>();
ageList.add(new Integer(46
ageList.add("50");
```

**COMPILE TIME ERROR:**
**java: incompatible types: java.lang.String cannot be converted to java.lang.Integer**

```
Integer sum = new Integer(0);
for (Object age : ageList) {
    sum = sum + ((Integer) age);
}
```

Compile time error is better than runtime error, because errors can be detected earlier!

Casting is not needed when using generics!

# Generics for classes

- **With generics for ArrayList**

```
ArrayList<Integer> ageList = new ArrayList<Integer>();
ageList.add(new Integer(46
ageList.add("50");

Integer sum = new Integer(0);
for (Object age : ageList) {
    sum = sum + ((Integer) age);
}
```

**COMPILE TIME ERROR:**
**java: incompatible types: java.lang.String cannot be converted to java.lang.Integer**

Compile time error is better than runtime error, because errors can be detected earlier!

Casting is not needed when using generics!

We know what is the type, so we can directly state it here!

# Generics for classes

- **With generics for ArrayList**

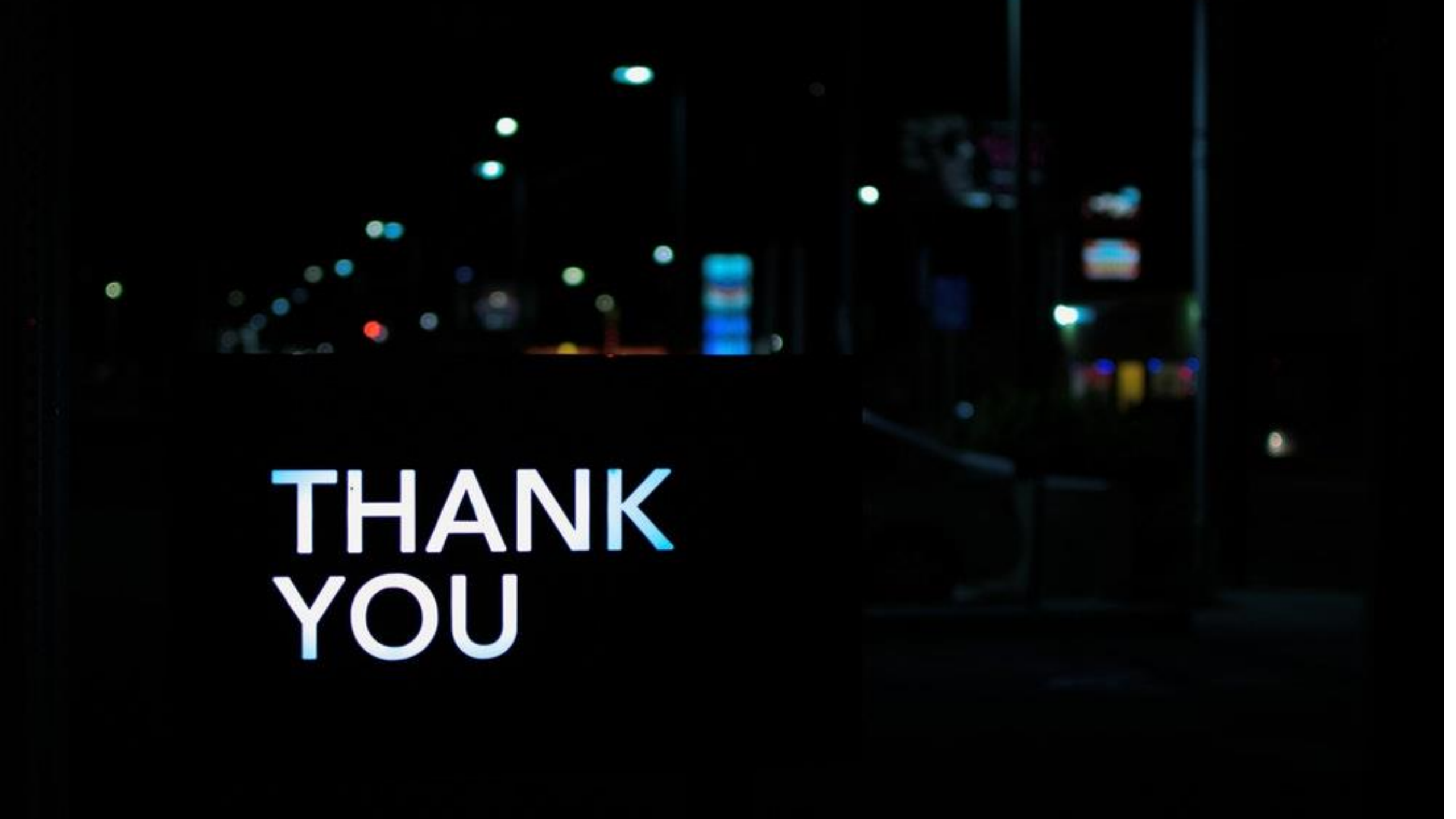
```
ArrayList<Integer> ageList = new ArrayList<Integer>();
ageList.add(new Integer(46
ageList.add("50");
```

**COMPILE TIME ERROR:**
**java: incompatible types: java.lang.String cannot be converted to java.lang.Integer**

```
Integer sum = new Integer(0);
for (Integer age : ageList) {
    sum = sum + age;
}
```

Compile time error is better than runtime error, because errors can be detected earlier!

Fixed, looks better now.


THANK
YOU