Riza Satria Perdana, S.T., M.T.

Teknik Informatika - STEI ITB

Generics

Pemrograman Berorientasi Objek

Pengantar

- Pada proyek pengembangan perangkat lunak sering muncul bug. Dengan perencanaan, programming, dan testing yang baik akan mereduksi munculnya bug.
- Ada bug yang lebih mudah dideteksi yaitu compile-time bug (dibandingkan run-time bug)

Pengantar

- Dengan menggunakan konsep Generik akan menambah stabilitas kode dengan membuat bug terdeteksi saat kompilasi

Contoh kasus: Simple Box class

```java
public class Box {
    private Object object;

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

    public Object get() {
        return object;
    }
}
```

```java
public class BoxDemo1 {
    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 kasus: Simple Box class

```
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)
```

Generic Type

- Generic type declaration

```
/**
 * 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;
    }
}
```

Generic Type

- Generic type invocation

```
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);
    }
}
```

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

Generic Methods and Constructors

- Type parameter dapat juga digunakan pada method dan konstruktor menjadi *generic methods* dan *generic constructors*

Contoh Generic Methods

```
public class Box<T> {
    private T t;

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

    public T get() {
        return t;
    }

    public <U> 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");
    }
}
```

Contoh Type Inference

```
public static <U> void fillBoxes(U u, List<Box<U>> boxes) {
    for (Box<U> box : boxes) {
        box.add(u);
    }
}


Crayon red = ...;
List<Box<Crayon>> crayonBoxes = ...;

Box.<Crayon>fillBoxes(red, crayonBoxes);

Box.fillBoxes(red, crayonBoxes); // compiler infers that U is Crayon
```

Terima Kasih