Introduction to Generics

In this lesson, an explanation is provided to get started with Generics Methods (functions) in Java.

We'll cover the following

Definition

"Generics allow the reusability of code, where one single method can be used for different `data-types` of variables or objects."

The idea is to allow different types like `Integer`, `String`, … etc and user-defined types to be a parameter to methods, classes, and interfaces.

For example, classes like `HashSet`, `ArrayList`, `HashMap`, etc use generics very well. We can use them for any type.

The following example illustrates three **non-generic** (type-sensitive) functions for finding maximum out of 3 inputs:

```
class Main {
    public static void main(String[] args) {
        System.out.printf("Max of %d, %d and %d is %d\n\n", 3, 4, 5,
            MaximumTest.maximum(3, 4, 5));
        System.out.printf("Max of %.1f,%.1f and %.1f is %.1f\n\n", 6.6, 8.8, 7.7,
            MaximumTest.maximum(6.6, 8.8, 7.7));
        System.out.printf("Max of %s, %s and %s is %s\n", "pear", "apple", "orange",
            MaximumTest.maximum("pear", "apple", "orange"));
    }
}
class MaximumTest {
    // determines the largest of three Comparable objects
    public static int maximum(int x, int y, int z) {
        int max = x; // assume x is initially the largest
        if (y > max) {
            max = y; // y is the largest so far
        }

        if (z > max) {
            max = z; // z is the largest now
        }
        return max; // returns the largest object
    }

    public static double maximum(double x, double y, double z) {
        double max = x; // assume x is initially the largest

        if (y > max) {
            max = y; // y is the largest so far
        }

        if (z > max) {
            max = z; // z is the largest now
        }
        return max; // returns the largest object
    }
    public static String maximum(String x, String y, String z) {
        String max = x; // assume x is initially the largest

        if (y.compareTo(max) > 0) {
            max = y; // y is the largest so far
        }

        if (z.compareTo(max) > 0) {
            max = z; // z is the largest now
        }
        return max; // returns the largest object
    }
}
```

Three *methods* that do exactly the same thing, but cannot be defined as a single method because they use *different* data types. (`int`, `double` & `String`)

Generic methods

To use generic methods, we use the following syntax.

Syntax

```
<T> void MyFirstGenericMethod(T element)
```

`T` is our *generic* data type's name, and when the method is to be called, it would be the same as if `T` was a `typedef` for your datatype.

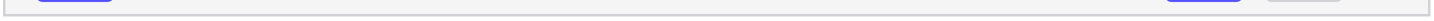
The following example now illustrates how the `multiply` *method* would be written using a *template*:

```
class Main {
    public static void main(String[] args) {
        System.out.printf("Max of %d, %d and %d is %d\n\n", 3, 4, 5,
            MaximumTest.maximum(3, 4, 5));
        System.out.printf("Max of %.1f,%.1f and %.1f is %.1f\n\n", 6.6, 8.8, 7.7,
            MaximumTest.maximum(6.6, 8.8, 7.7));
        System.out.printf("Max of %s, %s and %s is %s\n", "pear", "apple", "orange",
            MaximumTest.maximum("pear", "apple", "orange"));
    }
}

class MaximumTest {
    // determines the largest of three Comparable objects
    public static < T extends Comparable < T >> T maximum(T x, T y, T z) {
        T max = x; // assume x is initially the largest

        if (y.compareTo(max) > 0) {
            max = y; // y is the largest so far
        }

        if (z.compareTo(max) > 0) {
            max = z; // z is the largest now
        }
        return max; // returns the largest object
    }
}
```

Generic methods with multiple type parameters

In the above code, all of the arguments to `maximum()` must be the same type. Optionally, a template can have more type options, and the syntax is pretty simple. For a template with **three** types, called `T1`, `T2` and `T3`, we have:

```
class Generics {
    public static < T1, T2, T3 > void temp(T1 x, T2 y, T3 z) {
        System.out.println("This is x =" + x);
        System.out.println("This is y =" + y);
        System.out.println("This is z =" + z);

    }
    public static void main(String args[]) {
        temp(1, 2, 3);
    }
}
```



In the next lesson, we’ll take a look at templates in some more detail!