Java Generics

by Sophia

WHAT'S COVERED

In this lesson, you will learn about how generic types and methods are used in Java. Specifically, this lesson covers:

1. Generic Types

As discussed in previous tutorials, the data type for the elements in the array has to be part of the declaration, when declaring an array. Additionally, all of the elements in the array must be of the same data type. This lesson serves as a bridge between these topics and introduces generic types in Java. Though certain Java collections that are a more flexible alternative to traditional arrays will be seen in this tutorial, they will be covered in more detail in a future tutorial.

From time to time, you may have a need to use code for various types of data. **Generic programming** allows programmers to use the same code for different types of data. This alleviates the need to write different versions of the same code when working with different data types. **Generics** are another example and can be used with Java classes, which will be covered in a later tutorial. For certain situations, generics can also be used with methods.

A generic method can work with data of any type. This will be called out in an example in the next section of the tutorial. A generic method returns the value at the midpoint in an array.

KEY CONCEPT

Generics only work with objects (such as `String`), not primitive data types (such as `int, char, double,` etc...).

Fortunately, to address this, there are classes that wrap primitive data types so that they behave like objects.

Here is a list of the primitive types we have worked with so far and the corresponding wrapper classes:

Primitive Type	Wrapper Type	Description
boolean	Boolean	Stores true and false values
char	Character	Stores single letters, digits, and symbols
double	Double	Stores fractional numbers up to 15 decimal digits
float	Float	Stores fractional numbers up to 7 decimal digits
int	Integer	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	Long	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

THINK ABOUT IT

Note how the wrapper types, in the table above, all begin with capital letters. Remember that class names in Java always begin with capital letters. This identifies the wrapper types as classes.

TERM TO KNOW

Generic Programming (also Generic Types or Generics)
A type of programming that allows programmers to use the same code and structures for different data types without having to rewrite the code.

2. Generic Methods

Angle brackets (< >) are used when declaring a **generic method**. This designation indicates that the method is generic.

The variable in the angle brackets below, is a type variable that stands for a data type:

EXAMPLE

> (in the code below, <T>)

HINT

Note how T is then used to specify that the return data type from the method and the data type for the array are the same. Both include whatever T ends up being when the method is called.

When calling a generic method, the call looks like any other method call. The type of data is not specified in angle brackets. The data type is worked out by the compiler.

In the code below, note the following lines:

EXAMPLE

```
String midPointName = getMidPointItem(names);
```

and

EXAMPLE

```
char midPointLetter = getMidPointItem(letters);
```

Among others that show how the generic `getMidPointItem()` method is called, as seen below:

```
import java.util.Arrays;

public class GenericMethod {

    // The angle brackets <> indicate a generic method.
    // T is a type variable that stands for the data type. The method
    // returns a value of the same data type as the data type for the array.
    public static <T> T getMidPointItem(T[] array) {
        // division of an int by an int results in an int quotient - no decimal.
        // Returns array element at index length / 2.
        return array[array.length / 2];
    }

    public static void main(String[] args) {
        String[] names = { "Ann", "George", "Kim", "Pat", "Steve"};
        String midPointName = getMidPointItem(names);
        System.out.print("The middle item in the array " + Arrays.toString(names));
        System.out.println(" is " + midPointName + ".");

        // Instead of primitive type char, use Character - note capital C
        Character[] letters = {'a', 'b', 'c'};
        char midPointLetter = getMidPointItem(letters);
        System.out.print("The middle item in the array " + Arrays.toString(letters));
        System.out.println(" is " + midPointLetter + ".");

        // Instead of primitive type int use Integer - note capital I
        Integer[] agesInYears = {27, 33, 33, 39, 40, 40, 42, 45};
        int midPointAge = getMidPointItem(agesInYears);
```

```
        System.out.print("The middle item in the array " +
          Arrays.toString(agesInYears));
        System.out.println(" is " + midPointAge + ".");

        // Instead of primitive type double use Double - note capital D
        Double[] temperatures = {10.0, 21.5, 22.3, 25.0, 31.85, 35.99};
        double midPointTemp = getMidPointItem(temperatures);
        System.out.print("The middle item in the array " +
          Arrays.toString(temperatures));
        System.out.println(" is " + midPointTemp + ".");
    }
}
```

TRY IT

Directions: Type in the code above in the IDE using a file named GenericMethod.java. The output when the program is run should look like this:

```
The middle item in the array [Ann, George, Kim, Pat, Steve] is Kim.
The middle item in the array [a, b, c] is b.
The middle item in the array [27, 33, 33, 39, 40, 40, 42, 45] is 40.
The middle item in the array [10.0, 21.5, 22.3, 25.0, 31.85, 35.99] is 25.0.
```

REFLECT

Generic methods help programmers be more productive by allowing the writing of methods that can use the same code to work on a variety of data types without needing to write separate methods.

Generic types are used with Java collections and will be discussed in future tutorials.

TERM TO KNOW

Generic Method

A Java method written using generics that allow the same method to work on different data types.

SUMMARY

In this lesson, you learned about the uses of **generic types** in Java. You learned that **generics** are important because they allow the same code and data structures in Java to work with different data types. You learned that this approach helps to avoid repetitive code when a program needs to apply a similar process to different types of data. You learned that **generic methods** are written in Java using angle brackets. Finally, you learned that generic types will be important in future tutorials that include Java collection types.

Source: This content and supplemental material has been adapted from Java, Java, Java: Object-Oriented Problem Solving. Source cs.trincoll.edu/~ram/jjj/jjj-os-20170625.pdf

It has also been adapted from “Python for Everybody” By Dr. Charles R. Severance. Source py4e.com/html3/

TERMS TO KNOW

Generic Method

A Java method written using generics that allow the same method to work on different data types.

Generic Programming (also Generic Types or Generics)

A type of programming that allows programmers to use the same code and structures for different data types without having to rewrite the code.