Java Generics

Parametric Polymorphism

CASE STUDY: GENERIC SORTING PROGRAM

DR. ERIC CHOU

IEEE SENIOR MEMBER

Sort Algorithm on Generic Data Collection

- Sort algorithm is a good example to demonstrate the flexibility of generic method.
- For a data type to be qualified for sorting, it must be comparable. That means there is certain order among the elements. In Java objects, the data class must implement **Comparable** Interface or the data type must have a method working like compareTo or those basic comparison operators (==, >=, >, <, <=, !=).
- To make a generic method for sorting, it will be easier for object types which support **Comparable** Interface.

Case Study: Sorting an Array of Objects

You can develop a generic method for sorting an array of Comparable objects.

- The algorithm for the sort method is the same as in **SelectionSort.java**.
- The sort method in that program sorts an array of double values. The sort method in this example can sort an array of any object type, provided that the objects are also instances of the Comparable interface.
- The generic type is defined as **<E extends Comparable<E>>**. This has two meanings. First, it specifies that **E** is a subtype of **Comparable**. Second, it specifies that the elements to be compared are of the **E** type as well.
- The sort method uses the compareTo method to determine the order of the objects in the array. Integer, Double, Character, and String implement Comparable, so the objects of these classes can be compared using the compareTo method. The program creates arrays of Integer objects, Double objects, Character objects, and String objects and invoke the sort method to sort these arrays.

Generic Sort

Demo Program: GenericSort.java

Go BlueJ!