Generic trong Java

Nó sẽ rất thú vị nếu chúng ta có thể viết một phương thức sort đơn mà có thể xếp thứ tự các phần tử trong một mảng integer, một mảng String, hoặc một mảng của bất kỳ kiểu nào mà hỗ trợ xếp thứ tự.

Các phương thức **Generic** trong Java và các lớp Generic trong Java cho nhà lập trình khả năng xác định, với một khai báo phương thức đơn, một tập hợp các phương thức liên quan hoặc/với một khai báo lớp đơn, một tập hợp các kiểu liên quan, tương ứng.

Generic trong Java cũng cung cấp tính an toàn về kiểu trong thời điểm biên dịch mà cho phép nhà lập trình bắt các kiểu không hợp lệ tại thời điểm này.

Sử dụng khái niệm Generic trong Java, chúng ta có thể viết một phương thức chung để xếp thứ tự các mảng của đối tượng, sau đó gọi phương thức generic này với các mảng integer, các mảng Double, các mảng String, … để xếp thứ tự các phần tử mảng.

Các phương thức generic trong Java

Bạn có thể viết một khai báo phương thức generic đơn mà có thể được gọi với các tham số của các kiểu khác nhau. Dựa trên các kiểu tham số được truyền tới phương thức generic này, bộ biên dịch xử lý mỗi lần gọi phương thức một cách thích hợp. Dưới đây là các qui tắc để định nghĩa các phương thức Generic:

- Tất cả khai báo phương thức generic có một khu vực tham số kiểu được giới hạn bởi các dấu ngoặc nhọn (< và >) mà đứng trước kiểu trả về của phương thức (< E > trong ví dụ sau đây).

- Mỗi khu vực tham số kiểu chứa một hoặc nhiều tham số kiểu phân biệt nhau bởi dấu phảy. Một tham số kiểu, cũng được biết như là biến kiểu, là một định danh mà xác định một tên kiểu generic.

- Các tham số kiểu có thể được sử dụng để khai báo kiểu trả về và hoạt động như là nơi giữ (placeholder) cho các kiểu của các tham số được truyền tới phương thức generic, mà được biết như là các tham số kiểu thực sự.

- Phần thân phương thức generic được khai báo giống như bất kỳ phương thức nào khác. Ghi chú rằng các tham số kiểu chỉ có thể biểu diễn các kiểu tham chiếu, không phải là các kiểu gốc (như int, double, và char).

Ví dụ:

Ví dụ này minh họa cách chúng ta có thể in mảng các kiểu khác nhau bởi sử dụng một phương thức generic đơn.

```
public class GenericMethodTest
{
   // generic method printArray
   public static < E > void printArray( E[] inputArray )
   {
      // Display array elements
         for ( E element : inputArray ){
            System.out.printf( "%s ", element );
         }
         System.out.println();
    }

    public static void main( String args[] )
    {
        // Create arrays of Integer, Double and Character
        Integer[] intArray = { 1, 2, 3, 4, 5 };
        Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4 };
        Character[] charArray = { 'H', 'E', 'L', 'L', 'O' };

        System.out.println( "Array integerArray contains:" );
        printArray( intArray  ); // pass an Integer array

        System.out.println( "\nArray doubleArray contains:" );
        printArray( doubleArray ); // pass a Double array

        System.out.println( "\nArray characterArray contains:" );
        printArray( charArray ); // pass a Character array
    }
}
```

Nó sẽ cho kết quả:

```
Array integerArray contains:
1 2 3 4 5 6


Array doubleArray contains:
1.1 2.2 3.3 4.4


Array characterArray contains:
H E L L O
```

Các tham số kiểu giới hạn (bounded type) trong Java

Nhiều khi bạn muốn giới hạn các loại kiểu mà được cho phép để truyền tới một tham số kiểu. Ví dụ, một phương thức mà hoạt động trên các số có thể chỉ muốn chấp nhận các Number hoặc các lớp phụ của nó. Đó là lý do có các tham số Bounded Type này.

Để khai báo một tham số bounded type, liệt kê tên của tham số kiểu, được theo sau bởi từ khóa extends, được theo sau bởi giới hạn chữ hoa của nó.

Ví dụ:

Ví dụ sau minh họa cách extends được sử dụng: hoặc “extends” khi trong các lớp hoặc “implements” khi trong các interface. Ví dụ về phương thức generic trả về giá trị lớn nhất trong ba đối tượng Comparable.

```
public 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
   }
   public static void main( String args[] )
   {
      System.out.printf( "Max of %d, %d and %d is %d\n\n",
                  3, 4, 5, maximum( 3, 4, 5 ) );

      System.out.printf( "Maxm of %.1f,%.1f and %.1f is %.1f\n\n",
                  6.6, 8.8, 7.7, maximum( 6.6, 8.8, 7.7 ) );

      System.out.printf( "Max of %s, %s and %s is %s\n","pear",
         "apple", "orange", maximum( "pear", "apple", "orange" ) );
   }
}
```

Nó sẽ cho kết quả sau:

```
Maximum of 3, 4 and 5 is 5

Maximum of 6.6, 8.8 and 7.7 is 8.8

Maximum of pear, apple and orange is pear
```

Các lớp Generic trong Java

Một khai báo lớp Generic trông giống như một khai báo lớp không phải Generic, ngoại trừ là tên lớp được theo sau bởi một khu vực tham số kiểu.

Như với phương thức Generic, khu vực tham số kiểu của một lớp Generic có thể có một hoặc nhiều tham số kiểu phân biệt nhau bởi dấu phảy. Những lớp này còn được biết như là các lớp tham số hóa hoặc các kiểu tham số hóa bởi vì chúng chấp nhận một hoặc nhiều tham số.

Ví dụ:

Ví dụ này minh họa cách chúng ta định nghĩa một lớp Generic trong Java:

```
public class Box<T> {

  private T t;

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

  public T get() {
    return t;
  }

  public static void main(String[] args) {
     Box<Integer> integerBox = new Box<Integer>();
     Box<String> stringBox = new Box<String>();

     integerBox.add(new Integer(10));
     stringBox.add(new String("Hello World"));

     System.out.printf("Integer Value :%d\n\n", integerBox.get());
     System.out.printf("String Value :%s\n", stringBox.get());
  }
}
```

Nó sẽ cho kết quả:

```
Integer Value :10
String Value :Hello World
```