Sử dụng Iterator trong Java

Thường thì, bạn sẽ muốn tuần hoàn qua các phần tử trong một tập hợp. Ví dụ, bạn có thể muốn hiển thị mỗi phần tử.

Cách đơn giản nhất để thực hiện điều này là thuê một Iterator, là một đối tượng mà triển khai hoặc Iterator hoặc Listlerator interface.

Iterator cho bạn khả năng để tuần hoàn qua một tập hợp, kiếm được và gỡ bỏ các phần tử. ListIterator kế thừa Iterator để cho phép "vọc" song hướng một danh sách và sửa đổi các phần tử.

Trước khi bạn có thể truy cập một Collection thông qua một Iterator, bạn phải có được nó. Mỗi lớp Collection cung cấp một phương thức **iterator()** mà trả về một iterator tới phần bắt đầu của Collection. Bởi sử dụng đối tượng Iterator, bạn có thể truy cập mỗi phần tử trong Collection, từng phần tử một tại một thời điểm.

Để hiểu sâu hơn các khái niệm được trình bày trong chương này, mời bạn tham khảo loạt bài: **Ví dụ về Collection trong Java**.

Nói chung, để sử dụng một iterator để tuần hoàn qua các nội dung của một Collection, bạn theo các bước sau:

- Đạt được một iterator tới phần đầu của Collection bằng cách gọi phương thức iterator() của Collection trong Java.
- Thiết lập một vòng lặp mà tạo triệu hồi tới hasNext(). Vòng lặp này lặp đi lặp lại tới khi hasNext() trả về true.
- Trong vòng lặp, thu được mỗi phần tử bởi triệu hồi phương thức next().

Với các Collection mà triển khai List, bạn cũng có thể thu được một interator bởi triệu hồi ListIterator.

Phương thức được khai báo bởi Iterator trong Java

STT	Phương thức và Miêu tả
1	**boolean hasNext( )**

	Trả về true nếu có nhiều phần tử. Nếu không là false
2	**Object next( )** Trả về phần tử kế tiếp. Ném NoSuchElementException nếu không có một phần tử kế tiếp
3	**void remove( )** Gỡ bỏ phần tử hiện tại. Ném IllegalStateException nếu cố gắng gọi phương thức remove() mà không được đặt trước một triệu hồi tới next()

Phương thức được khai báo bởi ListIterator trong Java

STT	Phương thức và Miêu tả
1	**void add(Object obj)** Chèn obj vào trong List ở trước phần tử mà sẽ được trả về bởi lần triệu hồi tiếp theo tới next()
2	**boolean hasNext( )** Trả về true nếu có một phần tử kế tiếp. Nếu không là false
3	**boolean hasPrevious( )** Trả về true nếu có một phần tử ở trước. Nếu không là false
4	**Object next( )** Trả về phần tử kế tiếp. Ném NoSuchElementException nếu không có phần tử đó
5	**int nextIndex( )** Trả về chỉ mục của phần tử kế tiếp. Nếu không có phần tử này, trả về kích cỡ của list

6	**Object previous( )** Trả về phần tử trước. Ném NoSuchElementException nếu không có phần tử đó
7	**int previousIndex( )** Trả về chỉ mục của phần tử ở trước. Nếu không có phần tử này, trả về -1
8	**void remove( )** Gỡ bỏ phần tử hiện tại từ list. Ném IllegalStateException nếu remove() được triệu hồi trước khi next() hoặc previous() được gọi
9	**void set(Object obj)** Gán obj tới phần tử hiện tại. Đây là phần tử cuối cùng được trả về bởi một triệu hồi tới hoặc next() hoặc previous()

Ví dụ

Sau đây là ví dụ minh họa cả Iterator và ListIterator. Nó sử dụng một đối tượng ArrayList, nhưng các qui tắc chung áp dụng tới bất kỳ kiểu Collection nào.

Tất nhiên, ListIterator chỉ có sẵn cho các Collection mà triển khai List Interface trong Java:

```
import java.util.*;


public class IteratorDemo {


   public static void main(String args[]) {
      // Create an array list
      ArrayList al = new ArrayList();
      // add elements to the array list
      al.add("C");
      al.add("A");
      al.add("E");
      al.add("B");
```

```
      al.add("D");
      al.add("F");

      // Use iterator to display contents of al
      System.out.print("Original contents of al: ");
      Iterator itr = al.iterator();
      while(itr.hasNext()) {
         Object element = itr.next();
         System.out.print(element + " ");
      }
      System.out.println();

           // Modify objects being iterated
      ListIterator litr = al.listIterator();
      while(litr.hasNext()) {
         Object element = litr.next();
         litr.set(element + "+");
      }
      System.out.print("Modified contents of al: ");
      itr = al.iterator();
      while(itr.hasNext()) {
         Object element = itr.next();
         System.out.print(element + " ");
      }
      System.out.println();

      // Now, display the list backwards
      System.out.print("Modified list backwards: ");
      while(litr.hasPrevious()) {
         Object element = litr.previous();
         System.out.print(element + " ");
       }
       System.out.println();
```

```
    }
}
```

Nó sẽ cho kết quả sau:

```
Original contents of al: C A E B D F
Modified contents of al: C+ A+ E+ B+ D+ F+
Modified list backwards: F+ D+ B+ E+ A+ C+
```