Collection Interface trong Java

Collection Interface trong Java là nền tảng mà trên đó Collection Framework được xây dựng. Nó khai báo các phương thức core mà tất cả Collection sẽ có. Những phương thức này được tổng hợp trong bảng dưới đây.

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**.

Bởi vì tất cả tập hợp triển khai Collection, tương tự như các phương thức của nó là cần thiết cho Framework đó. Một số phương thức này có thể ném một**UnsupportedOperationException**.

STT	Phương thức và Miêu tả
1	**boolean add(Object obj)** Thêm obj tới Collection đang gọi. Trả về true nếu obj được thêm tới Collection đó. Trả về false nếu obj đã là một thành viên trong Collection đó, hoặc nếu Collection đó không cho phép các bản sao
2	**boolean addAll(Collection c)** Thêm tất cả phần tử của c tới Collection đang gọi. Trả về true nếu hoạt động này thành công (ví dụ: phần tử được thêm thành công). Nếu không là false
3	**void clear( )** Gỡ bỏ tất cả phần tử từ Collection đang gọi
4	**boolean contains(Object obj)** Trả về true nếu obj là một phần tử của Collection đang gọi. Nếu không là false
5	**boolean containsAll(Collection c)** Trả về true nếu Collection đang gọi chứa tất cả phần tử của c. Nếu không là false

6	**boolean equals(Object obj)** Trả về true nếu Collection đang gọi và obj là cân bằng nhau. Nếu không là false
7	**int hashCode( )** Trả về hash code cho Collection đang gọi này
8	**boolean isEmpty( )** Trả về true nếu Collection đang gọi là trống. Nếu không là false
9	**Iterator iterator( )** Trả về một iterator cho Collection đang gọi
10	**boolean remove(Object obj)** Gỡ bỏ một instance của obj từ Collection đang gọi. Trả về true nếu phần tử bị gỡ bỏ. Nếu không là false
11	**boolean removeAll(Collection c)** Gỡ bỏ tất cả phần tử của c từ Collection đang gọi. Trả về true nếu Collection đã thay đổi (ví dụ: các phần tử bị gỡ bỏ). Nếu không là false
12	**boolean retainAll(Collection c)** Trả về tất cả phần tử từ Collection đang gọi ngoại trừ những phần tử trong c. Trả về true nếu Collection đã thay đổi (ví dụ: các phần tử bị gỡ bỏ). Nếu không là false
13	**int size( )** Trả về số phần tử được giữ trong Collection đang gọi
14	**Object[ ] toArray( )** Trả về một mảng mà chứa tất cả phần tử được lưu trong Collection đang gọi. Các

	phần tử mảng này được sao chép từ các phần tử trong Collection
15	**Object[ ] toArray(Object array[ ])** Trả về một mảng chỉ chứa các phần tử của Collection mà có kiểu đã so khớp với mảng đó

Ví dụ

Ví dụ sau minh họa một số phương thức từ việc triển khai các lớp đa dạng của Collection Interface trong Java:

```
import java.util.*;


public class CollectionsDemo {


   public static void main(String[] args) {
      List a1 = new ArrayList();
      a1.add("Zara");
      a1.add("Mahnaz");
      a1.add("Ayan");
      System.out.println(" ArrayList Elements");
      System.out.print("\t" + a1);


      List l1 = new LinkedList();
      l1.add("Zara");
      l1.add("Mahnaz");
      l1.add("Ayan");
      System.out.println();
      System.out.println(" LinkedList Elements");
      System.out.print("\t" + l1);


      Set s1 = new HashSet();
      s1.add("Zara");
```

```
      s1.add("Mahnaz");
      s1.add("Ayan");
      System.out.println();
      System.out.println(" Set Elements");
      System.out.print("\t" + s1);

      Map m1 = new HashMap();
      m1.put("Zara", "8");
      m1.put("Mahnaz", "31");
      m1.put("Ayan", "12");
      m1.put("Daisy", "14");
      System.out.println();
      System.out.println(" Map Elements");
      System.out.print("\t" + m1);
   }
}
```

Nó sẽ cho kết quả sau:

```
ArrayList Elements
       [Zara, Mahnaz, Ayan]
LinkedList Elements
       [Zara, Mahnaz, Ayan]
Set Elements
       [Zara, Mahnaz, Ayan]
Map Elements
       {Mahnaz=31, Ayan=12, Daisy=14, Zara=8}
```