List Interface trong Java

List Interface trong Java kế thừa **Collection** và khai báo các hành vi của một collection mà lưu giữ một dãy các phần tử.

- Các phần tử có thể được chèn hoặc được truy cập thông qua vị trí của chúng trong danh sách, bởi sử dụng chỉ mục xây dựng bắt đầu từ 0.
- Một list có thể chứa nhiều bản sao phần tử.
- Ngoài các phương thức được định nghĩa bởi **Collection**, List Interface định nghĩa một số phương thức riêng, mà được liệt kê trong bảng dưới đây.
- Một số phương thức của List Interface sẽ ném một UnsupportedOperationException nếu collection không thể bị sửa đổi, và ném một ClassCastException nếu một đối tượng là không tương thích với đối tượng khác.

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**.

STT	Phương thức và Miêu tả
1	**void add(int index, Object obj)** Chèn obj vào trong List đang gọi tại index đã cho. Bất kỳ phần tử nào đã tồn tại trước tại hoặc trên điểm chèn bị bỏ qua. Vì thế, không có phần tử nào bị ghi đè
2	**boolean addAll(int index, Collection c)** Chèn tất cả phần tử của c vào trong List đang gọi tại chỉ mục đã cho. Bất kỳ phần tử nào đã tồn tại trước tại hoặc trên điểm chèn bị bỏ qua. Vì thế, không có phần tử nào bị ghi đè. Trả về true nếu List đang gọi thay đổi và nếu không trả về false
3	**Object get(int index)** Trả về đối tượng được lưu giữ tại index đã cho bên trong Collection đang gọi

4	**int indexOf(Object obj)** Trả về index của sự xuất hiện đầu tiên của obj trong List đang gọi. Nếu obj không là một phần tử trong list, -1 được trả về
5	**int lastIndexOf(Object obj)** Trả về index của sự xuất hiện cuối cùng của obj trong List đang gọi. Nếu obj không là một phần tử trong list, -1 được trả về
6	**ListIterator listIterator( )** Trả về một iterator tới phần đầu của List đang gọi
7	**ListIterator listIterator(int index)** Trả về một iterator tới List đang gọi tại index đã cho
8	**Object remove(int index)** Gỡ bỏ phần tử tại index từ List đang gọi và trả về phần tử bị xóa đó. List kết quả được compact lại. Đó là, các chỉ mục của dãy phần tử phụ bị lượng giảm đi 1
9	**Object set(int index, Object obj)** Gán obj tới vị trí được xác định bởi index bên trong List đang gọi
10	**List subList(int start, int end)** Trả về một list mà bao gồm các phần tử từ start tới end-1 trong List đang gọi. Các phần tử trong list trả về cũng được tham chiếu bởi đối tượng đang gọi

Ví dụ

List Interface trên đã triển khai trong các lớp đa dạng như ArrayList hoặc LinkedList, …. Sau đây là ví dụ giải thích một số phương thức của List 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);
   }
}
```

Nó sẽ cho kết quả sau:

```
 ArrayList Elements
        [Zara, Mahnaz, Ayan]
 LinkedList Elements
        [Zara, Mahnaz, Ayan]
```