Cập nhật tháng 8 năm 2024

[Bài đọc] Giới thiệu List và các phương thức làm việc với List

1. List là gì?

List là một giao diện (interface) trong Collection Framework của Java, nằm trong gói **java.util.List** là một tập hợp các phần tử có thứ tự, cho phép chứa các phần tử trùng lặp. Điều này làm cho **List** trở thành một lựa chọn phổ biến khi cần lưu trữ và xử lý dữ liệu tuần tự.

2. Đặc điểm của List

- **Duy trì thứ tự chèn**: Các phần tử trong List được lưu trữ theo thứ tự chèn vào.
- **Cho phép trùng lặp**: List hỗ trợ các phần tử trùng lặp, tức là bạn có thể thêm nhiều bản sao của cùng một phần tử.
- **Hỗ trợ truy cập theo chỉ số (index)**: Mỗi phần tử trong List được đánh chỉ số bắt đầu từ 0, cho phép truy cập trực tiếp đến các phần tử bằng chỉ số.

3. Các lớp triển khai của List

- **ArrayList**: Một lớp được triển khai dựa trên mảng động, cung cấp khả năng truy cập nhanh nhưng thêm/xóa chậm ở giữa danh sách.
- **LinkedList**: Triển khai danh sách liên kết kép, hỗ trợ thêm/xóa nhanh ở giữa nhưng truy cập chậm hơn ArrayList.
- **Vector**: Một phiên bản đồng bộ (synchronized) của ArrayList.
- **Stack**: Kế thừa từ Vector, hoạt động như một ngăn xếp (Stack).

4. Các phương thức chính của List

Dưới đây là một số phương thức quan trọng trong giao diện List:

Phương thức	Mô tả
add(E e)	Thêm phần tử vào danh sách.
add(int index, E element)	Chèn phần tử vào vị trí chỉ định.
get(int index)	Lấy phần tử tại vị trí chỉ định.
set(int index, E element)	Gán giá trị mới cho phần tử tại vị trí chỉ định.

remove(int index)	Xóa phần tử tại vị trí chỉ định.
remove(Object o)	Xóa phần tử đầu tiên khớp với đối tượng chỉ định.
size()	Trả về số lượng phần tử trong danh sách.
isEmpty()	Kiểm tra danh sách có rỗng hay không.
contains(Object o)	Kiểm tra danh sách có chứa phần tử chỉ định không.
indexOf(Object o)	Trả về chỉ số của phần tử đầu tiên khớp với đối tượng chỉ định.
lastIndexOf(Object o)	Trả về chỉ số của phần tử cuối cùng khớp với đối tượng chỉ định.
clear()	Xóa tất cả phần tử trong danh sách.
subList(int fromIndex, int toIndex)	Trả về một danh sách con (sublist) từ chỉ số fromIndex đến toIndex.

5. **Ví dụ minh họa**

Ví dụ sử dụng ArrayList:

```java
import java.util.ArrayList;
import java.util.List;

public class ListExample {
    public static void main(String[] args) {
        // Tạo một danh sách
        List<String> list = new ArrayList<>();

        // Thêm phần tử
        list.add("Java");
        list.add("Python");
        list.add("C++");

        // Chèn phần tử vào vị trí 1
        list.add(1, "JavaScript");

        // Truy cập phần tử
        System.out.println("Phần tử tại vị trí 2: " + list.get(2));

        // Xóa phần tử
        list.remove("Python");

        // In danh sách
        System.out.println("Danh sách: " + list);
    }
}
```

Kết quả:

```
Phần tử tại vị trí 2: Python
Danh sách: [Java, JavaScript, C++]
```

6. **Khi nào nên sử dụng List**
   - Khi bạn cần lưu trữ dữ liệu có thứ tự.
   - Khi bạn cần truy cập và chỉnh sửa dữ liệu theo chỉ số.
   - Khi dữ liệu có thể có giá trị trùng lặp.

7. **Một số lưu ý**
   - Chọn ArrayList khi hiệu suất truy cập là ưu tiên.
   - Chọn LinkedList khi thao tác thêm, xóa ở giữa danh sách là quan trọng.
   - Hạn chế sử dụng Vector vì nó đã lỗi thời, trừ khi bạn cần đồng bộ hóa.

Link tài nguyên đọc thêm: https://www.geeksforgeeks.org/list-interface-java-examples/

Danh sách các bài học