Cập nhật tháng 8 năm 2024

[Bài đọc] ArrayList và LinkedList

1. **Giới thiệu chung**
- ArrayList
    - Là một **danh sách động** (resizable array) – dùng mảng phía dưới để lưu trữ các phần tử
    - Thuộc **java.util.ArrayList**
    - Cho phép truy cập phần tử nhanh qua chỉ số (**index**)

- LinkedList
    - Là một **danh sách liên kết đôi** (doubly linked list)
    - Mỗi phần tử là một node, chứa giá trị và liên kết đến node trước và sau
    - Thuộc **java.util.LinkedList**

2. **Điểm giống nhau**
- Đều là **class con của List interface**, nên có thể dùng các phương thức chung như: **add()**, **remove()**, **get()**, **size()**, **clear()**, ...
- Cho phép chứa các phần tử trùng lặp
- Duy trì thứ tự chèn phần tử

3. **So sánh chi tiết ArrayList vs LinkedList**

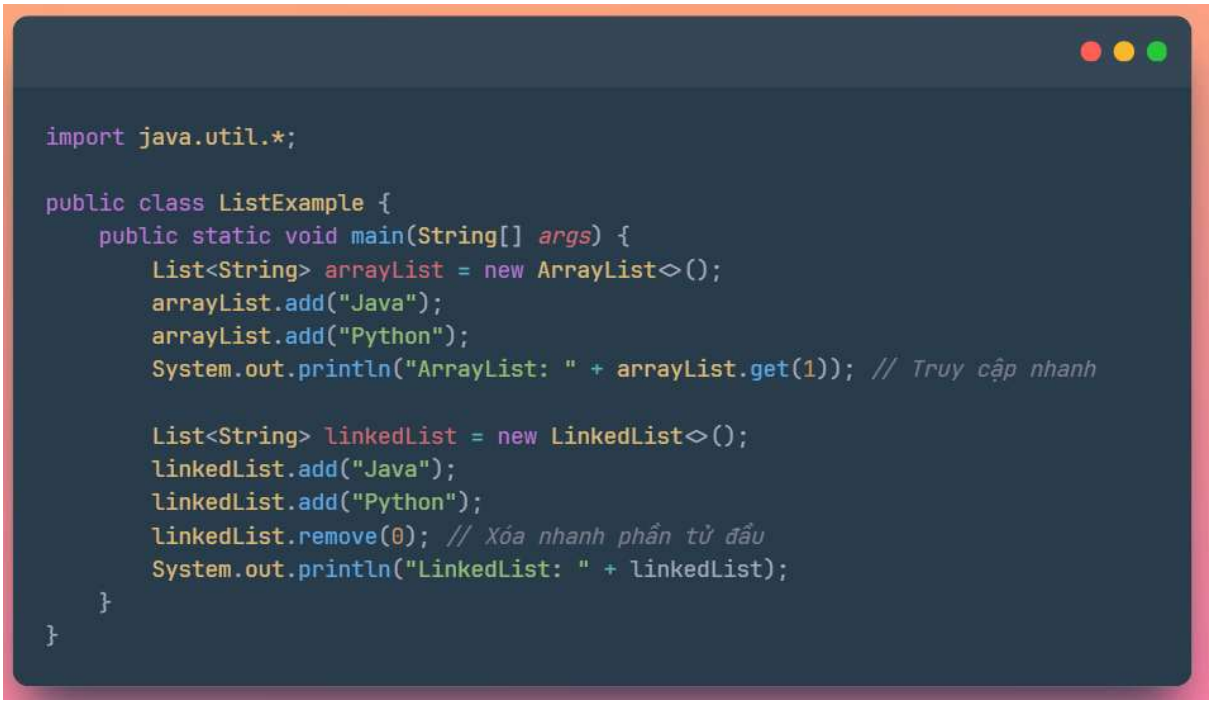
Tiêu chí	**ArrayList**	**LinkedList**
Cấu trúc dữ liệu nền tảng	Mảng (array)	Danh sách liên kết đôi (doubly linked list)
Tốc độ truy cập phần tử	Nhanh (O(1) với get(index))	Chậm (O(n) với get(index))
Thêm vào cuối	Nhanh (O(1) trung bình)	Nhanh (O(1))
Thêm/xóa ở đầu hoặc giữa	Chậm (O(n)) do phải dời phần tử	Nhanh (O(1) nếu biết node)
Bộ nhớ	Ít hơn (chỉ lưu giá trị)	Nhiều hơn (lưu thêm 2 con trỏ mỗi node)

Thích hợp	Truy cập nhanh, thao tác ít thay đổi	Thêm/xóa nhiều ở giữa

4. **Khi nào nên dùng?**

Tình huống	Sử dụng
Truy cập phần tử nhanh qua chỉ số	ArrayList
Thêm/xóa nhiều ở đầu/cuối/danh sách linh hoạt	LinkedList
Không cần chèn nhiều, chủ yếu duyệt	ArrayList
Dữ liệu lớn, thay đổi liên tục	LinkedList (tùy tình huống cụ thể)

5. **Ví dụ minh họa**



```java
import java.util.*;

public class ListExample {
    public static void main(String[] args) {
        List<String> arrayList = new ArrayList<>();
        arrayList.add("Java");
        arrayList.add("Python");
        System.out.println("ArrayList: " + arrayList.get(1)); // Truy cập nhanh

        List<String> linkedList = new LinkedList<>();
        linkedList.add("Java");
        linkedList.add("Python");
        linkedList.remove(0); // Xóa nhanh phần tử đầu
        System.out.println("LinkedList: " + linkedList);
    }
}
```

6. **Tóm tắt**

- **ArrayList** tối ưu cho **truy cập chỉ số** và **thêm vào cuối**
- **LinkedList** tối ưu cho **thêm/xóa liên tục** (đầu/cuối)
- Hiểu đúng nhu cầu sử dụng để chọn cấu trúc phù hợp