1. Nêu ra các đặc điểm **List Interface**:

List là một giao diện (interface) trong Java, thuộc gói java.util, dùng để lưu trữ một tập hợp các phần tử có thứ tự, cho phép các phần tử trùng lặp.

Thứ tự: theo thứ tự thêm vào. Mỗi phần tử đều có một (index) bắt đầu từ 0.

Cho phép trùng lặp: List cho phép thêm các phần tử có giá trị giống nhau.

Truy cập ngẫu nhiên: Có thể truy cập và sửa đổi phần tử thông qua chỉ số của nó.

Phương thức chính:

add(): Thêm phần tử vào cuối danh sách.

remove(Object o): Xóa phần tử đầu tiên có giá trị tương ứng trong danh sách

get(): Lấy phần tử ở vị trí index

set(int index, E element): Thay thế phần tử ở vị trí chỉ định bằng giá trị mới

size(): Trả về số lượng phần tử trong danh sách.

contains(Object o): Kiểm tra xem danh sách có chứa phần tử hay không.

indexOf(Object o): Trả về chỉ số của phần tử đầu tiên xuất hiện trong danh sách.

1. Kể ra các **class**triển khai từ **List Interface**

Các class triển khai từ List Interface:

ArrayList

LinkedList

Vector

Stack (là một lớp con của Vector)

3.  Phân biệt rõ trường hợp sử dụng của từng **class**đó

ArrayList:

Đặc điểm:

Dựa trên một mảng động (dynamic array) giúp quản lý các phần tử một cách hiệu quả.

Hỗ trợ truy cập ngẫu nhiên (random access) nhanh với thời gian O(1) vì các phần tử được lưu trong một mảng.

Tuy nhiên, việc thêm hoặc xóa phần tử ở giữa danh sách có thể tốn thời gian vì phải dịch chuyển các phần tử khác (O(n)).

Trường hợp sử dụng:

Khi cần truy cập phần tử thường xuyên và việc thêm/xóa không xảy ra nhiều.

Ví dụ: Dùng khi cần một danh sách để lưu trữ và truy cập nhanh các phần tử theo chỉ số.

2. LinkedList:

Đặc điểm:

Dựa trên cấu trúc danh sách liên kết (doubly linked list), các phần tử không lưu trữ liên tiếp mà liên kết với nhau thông qua các con trỏ.

Truy cập phần tử qua chỉ số chậm hơn ArrayList vì cần phải duyệt qua các phần tử (O(n)).

Tuy nhiên, việc thêm/xóa phần tử ở đầu, cuối, hoặc giữa danh sách nhanh hơn (O(1) khi biết vị trí).

Trường hợp sử dụng:

Khi cần thêm/xóa nhiều phần tử ở đầu hoặc giữa danh sách.

Ví dụ: Dùng khi xử lý hàng đợi (queue) hoặc ngăn xếp (stack).

3. Vector:

Đặc điểm:

Giống như ArrayList, Vector cũng dựa trên mảng động nhưng có một điểm khác biệt quan trọng là thread-safe. Điều này có nghĩa là các phương thức của Vector được đồng bộ hóa (synchronized), nên có thể sử dụng an toàn trong môi trường đa luồng (multithreading).

Tuy nhiên, việc đồng bộ hóa các phương thức khiến Vector có hiệu suất kém hơn ArrayList trong các chương trình đơn luồng (single-threaded).

Trường hợp sử dụng:

Khi làm việc với môi trường đa luồng và cần một danh sách an toàn cho việc truy cập đồng thời (concurrent access).

Ví dụ: Dùng trong các ứng dụng yêu cầu quản lý danh sách an toàn trong môi trường đa luồng.

4. Stack:

Đặc điểm:

Stack là một lớp con của Vector và tuân theo nguyên tắc LIFO (Last In, First Out - vào sau ra trước).

Có các phương thức đặc trưng như push(), pop(), peek().

Trường hợp sử dụng:

Khi cần một cấu trúc dữ liệu tuân theo nguyên tắc LIFO.

Ví dụ: Dùng trong các thuật toán đệ quy, kiểm tra dấu ngoặc, xử lý hồi quy.