# Đặc điểm của List Interface trong Java

* Lưu trữ các phần tử theo thứ tự mà chúng được thêm vào. Mỗi phần tử trong danh sách có một chỉ mục (index) bắt đầu từ 0.
* Cho phép thêm các phần tử trùng lặp vào danh sách.
* Cho phép truy cập, thêm, xóa phần tử tại một vị trí cụ thể thông qua chỉ mục.
* Các phương thức chính:
* add(): Thêm phần tử vào list.
* get(): Truy cập phần tử trong list tại 1 vị trí chỉ định.
* remove(): Xóa phần tử tại một vị trí chỉ định hoặc theo giá trị.
* set(): Cập nhật phần tử tại một vị trí chỉ định

# Các class triển khai từ List Interface

* Array List
* Linked List
* Vector
* Stack (kế thừa từ Vector)

# Phân biệt và trường hợp sử dụng của từng class

1. **Array List**

* **Cấu trúc bên trong**: Sử dụng mảng động (dynamic array) để lưu trữ các phần tử.
* **Tính năng nổi bật:**
* Truy cập phần tử nhanh do truy cập trực tiếp qua chỉ mục (index).
* Thích hợp cho thao tác đọc và truy cập phần tử thường xuyên.
* Không đồng bộ (không thread-safe).
* **Nhược điểm**: Các thao tác như thêm/xóa ở giữa danh sách có hiệu suất chậm vì phải dịch chuyển các phần tử.
* **Trường hợp sử dụng**: Khi cần truy cập dữ liệu nhanh, chẳng hạn như danh sách sinh viên, danh sách sản phẩm.

1. **Linked List**

* **Cấu trúc bên trong:** Sử dụng danh sách liên kết kép (doubly-linked list).
* **Tính năng nổi bật**:
* Thao tác thêm/xóa phần tử ở bất kỳ vị trí nào trong danh sách nhanh hơn ArrayList, vì chỉ cần thay đổi liên kết.
* Thích hợp cho các thao tác chèn/xóa thường xuyên.
* Hỗ trợ cả danh sách và hàng đợi (queue), vì vậy có thể được sử dụng như một Queue hoặc Deque.
* **Nhược điểm**: Truy cập phần tử chậm hơn so với ArrayList do phải duyệt qua danh sách để tìm phần tử theo chỉ mục.
* **Trường hợp sử dụng**: Khi cần thêm/xóa dữ liệu ở giữa danh sách, chẳng hạn như danh sách lịch sử truy cập web.

1. **Vector**

* **Cấu trúc bên trong:** Sử dụng mảng động như ArrayList.
* **Tính năng nổi bật**:
* Đồng bộ hóa (synchronized), do đó thích hợp cho môi trường đa luồng (multi-threaded).
* Tương tự ArrayList nhưng đảm bảo an toàn khi có nhiều luồng truy cập.
* **Nhược điểm**: Hiệu suất thấp hơn ArrayList do việc đồng bộ hóa làm chậm các thao tác.
* **Trường hợp sử dụng**: Khi cần lưu trữ danh sách các phần tử trong môi trường đa luồng, như danh sách người dùng đăng nhập vào hệ thống đa luồng.

1. **Stack (kế thừa từ Vector)**

* **Cấu trúc bên trong:** Stack là một lớp kế thừa từ Vector và hoạt động dựa trên nguyên tắc LIFO.
* **Tính năng nổi bật**:
* Cung cấp các thao tác như push(), pop(), peek() để thao tác với đỉnh của stack.
* Đồng bộ hóa do kế thừa từ Vector.
* **Nhược điểm**: Tương tự như Vector, hiệu suất bị ảnh hưởng bởi sự đồng bộ.
* **Trường hợp sử dụng**: Khi cần cấu trúc dữ liệu kiểu ngăn xếp như trong các ứng dụng tính toán biểu thức toán học, hoặc quản lý undo/redo trong ứng dụng.