# Queue Interface

1. **Đặc điểm**

* Là một cấu trúc dữ liệu dạng hàng đợi (FIFO - First In First Out), trong đó các phần tử được thêm vào cuối và lấy ra từ đầu hàng đợi.
* Không cho phép chèn vào giữa hay truy cập ngẫu nhiên các phần tử, chỉ có thể thao tác với phần tử đầu và cuối hàng đợi.
* Các phương thức chính:
* add(E e): Thêm phần tử vào hàng đợi, ném ngoại lệ nếu hàng đợi đầy.
* offer(E e): Thêm phần tử vào hàng đợi, trả về false nếu hàng đợi đầy.
* remove(): Lấy và xóa phần tử đầu tiên của hàng đợi, ném ngoại lệ nếu hàng đợi rỗng.
* poll(): Lấy và xóa phần tử đầu tiên, trả về null nếu hàng đợi rỗng.
* element(): Trả về phần tử đầu tiên mà không xóa, ném ngoại lệ nếu hàng đợi rỗng.
* peek(): Trả về phần tử đầu tiên mà không xóa, trả về null nếu hàng đợi rỗng.

1. **Class triển khai**

* LinkedList:
* Triển khai cả Queue lẫn Deque.
* Thích hợp khi cần các thao tác thêm và xóa ở cả hai đầu của hàng đợi.
* PriorityQueue:
* Cung cấp một hàng đợi ưu tiên, phần tử được xếp theo thứ tự tự nhiên hoặc theo thứ tự được xác định bởi Comparator.
* Dùng khi cần lấy phần tử theo độ ưu tiên, không theo thứ tự chèn.
* ArrayDeque
* Một cách triển khai của Deque, cũng có thể được dùng như một hàng đợi thông thường (FIFO).
* Thích hợp cho các thao tác cần hiệu năng cao mà không cần đồng bộ.

1. **Trường hợp sử dụng**

* Sử dụng khi cần một hàng đợi truyền thống (FIFO), nơi phần tử được xử lý theo thứ tự chúng được thêm vào. Ví dụ: quản lý luồng công việc, hàng đợi tác vụ.

# Deque Interface

1. **Đặc điểm**

* Deque (Double Ended Queue) là một hàng đợi hai đầu, cho phép thêm và xóa phần tử ở cả hai đầu (FIFO hoặc LIFO - Last In First Out).
* Các phương thức chính:
* addFirst(E e), addLast(E e): Thêm phần tử vào đầu hoặc cuối hàng đợi.
* offerFirst(E e), offerLast(E e): Thêm phần tử vào đầu hoặc cuối, không ném ngoại lệ.
* removeFirst(), removeLast(): Lấy và xóa phần tử đầu hoặc cuối, ném ngoại lệ nếu rỗng.
* pollFirst(), pollLast(): Lấy và xóa phần tử đầu hoặc cuối, trả về null nếu rỗng.
* getFirst(), getLast(): Trả về phần tử đầu hoặc cuối mà không xóa, ném ngoại lệ nếu rỗng.
* peekFirst(), peekLast(): Trả về phần tử đầu hoặc cuối mà không xóa, trả về null nếu rỗng.

1. **Class triển khai**

* LinkedList:
* Triển khai cả Queue lẫn Deque.
* Thích hợp khi cần thao tác với cả hai đầu của hàng đợi.
* ArrayDeque
* Thích hợp cho các thao tác như hàng đợi hai đầu với hiệu suất cao, không có giới hạn kích thước.
* Có thể được dùng như ngăn xếp (stack) hoặc hàng đợi thông thường.

1. **Trường hợp sử dụng**

* Sử dụng khi cần linh hoạt với việc thêm hoặc lấy phần tử từ cả hai đầu của hàng đợi, hoặc muốn dùng nó như ngăn xếp (LIFO). Ví dụ: thực hiện Undo/Redo, xử lý luồng công việc theo nhiều thứ tự (FIFO/LIFO).