Queue Interface

**1. Queue Interface:**

**Đặc điểm:**

* Queue là một interface trong Java, đại diện cho một collection lưu trữ các phần tử theo thứ tự chèn vào (FIFO - First In, First Out). Tuy nhiên, thứ tự này có thể thay đổi tùy thuộc vào việc triển khai cụ thể.
* Các phương thức phổ biến:
  + add(E e): Thêm phần tử vào hàng đợi, nếu không thể thêm thì ném ra ngoại lệ.
  + offer(E e): Thêm phần tử vào hàng đợi, trả về true nếu thành công, false nếu không thêm được.
  + remove(): Xóa và trả về phần tử đầu tiên, ném ngoại lệ nếu hàng đợi rỗng.
  + poll(): Xóa và trả về phần tử đầu tiên, trả về null nếu hàng đợi rỗng.
  + element(): Lấy nhưng không xóa phần tử đầu tiên, ném ngoại lệ nếu hàng đợi rỗng.
  + peek(): Lấy nhưng không xóa phần tử đầu tiên, trả về null nếu hàng đợi rỗng.

**Các lớp triển khai từ Queue Interface:**

* **PriorityQueue**: Hàng đợi ưu tiên, các phần tử được sắp xếp theo thứ tự tự nhiên hoặc thông qua comparator.
  + Sử dụng khi cần truy cập phần tử có độ ưu tiên cao nhất (không phải FIFO).
* **LinkedList**: Triển khai danh sách liên kết kép có thể hoạt động như một hàng đợi.
  + Sử dụng khi cần hàng đợi FIFO đơn giản hoặc khi cần cả thao tác như Deque.
* **ArrayBlockingQueue**: Hàng đợi có giới hạn kích thước cố định, sử dụng trong môi trường đa luồng.
  + Sử dụng khi cần hàng đợi có dung lượng giới hạn và khả năng thread-safe.
* **ConcurrentLinkedQueue**: Hàng đợi không giới hạn kích thước, thread-safe, không chặn.
  + Sử dụng khi cần hàng đợi thread-safe, không cần kích thước cố định.

**2. Deque Interface:**

**Đặc điểm:**

* Deque (Double Ended Queue) là một hàng đợi hai đầu, cho phép thêm và xóa phần tử từ cả hai đầu.
* Hỗ trợ cả hai kiểu FIFO và LIFO (Last In, First Out).
* Các phương thức bổ sung:
  + 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 nếu không thêm được.
  + removeFirst(), removeLast(): Xóa và trả về phần tử ở đầu hoặc cuối, ném ngoại lệ nếu rỗng.
  + pollFirst(), pollLast(): Xóa và trả về phần tử ở đầu hoặc cuối, trả về null nếu rỗng.
  + getFirst(), getLast(): Lấy nhưng không xóa phần tử ở đầu hoặc cuối, ném ngoại lệ nếu rỗng.
  + peekFirst(), peekLast(): Lấy nhưng không xóa phần tử ở đầu hoặc cuối, trả về null nếu rỗng.

**Các lớp triển khai từ Deque Interface:**

* **LinkedList**: Triển khai cả Queue và Deque, hỗ trợ thao tác từ cả hai đầu.
  + Sử dụng khi cần một cấu trúc dữ liệu có thể hoạt động cả như hàng đợi FIFO và LIFO.
* **ArrayDeque**: Một triển khai dựa trên mảng, không có giới hạn kích thước, cho phép thao tác ở cả hai đầu.
  + Sử dụng khi cần hàng đợi hai đầu hiệu suất cao mà không cần thread-safe.
* **LinkedBlockingDeque**: Triển khai hàng đợi hai đầu có giới hạn kích thước, hỗ trợ đa luồng.
  + Sử dụng khi cần hàng đợi thread-safe hai đầu trong môi trường đa luồng.

**Phân biệt trường hợp sử dụng:**

* **Queue**: Dùng khi chỉ cần thao tác theo FIFO đơn giản, như trong hệ thống xử lý công việc (job scheduling).
* **Deque**: Dùng khi cần thao tác từ cả hai đầu (FIFO và LIFO), ví dụ như cài đặt stack và queue cùng một lúc.