1. Nêu các đặc điểm của **Queue Interface, Dequeue Interface**

Queue interface:

* hàng đợi hoạt động theo quy tắc FIFO first in first out. Phần tử được thêm vào sẽ được loại bỏ trước.
* Các phương thức:
  + add() thêm vào hàng đợi, nếu ko thể thêm sẽ ném ngoại lệ
  + Offer(): thêm vào hàng đợi, nếu ko thể thì trả về false
  + Remove(): xóa phần tử đầu, nếu rỗng ném ngoại lệ
  + Poll():xóa phần tử đầu, nếu rỗng trả về null
  + Element(): trả về phần tử đầu, ném ngoại lệ nếu rỗng
  + Peek():trả về phần tử đầu, trả về null nếu rỗng

Dequeue interface:

* Hàng đợi 2 đầu, cho phép thêm và loại bỏ phần tử 2 đầu
* Hỗ trợ hoạt động FIFO và LIFO
* Một số phương thức:
  + addFirst(): thêm vào đầu
  + addLast(): thêm vào cuối
  + offerFirst():
  + offerLast():
  + removeFirst(): xóa đầu
  + remveLast(): xóa cuối
  + pollFirst():
  + pollLast():
  + getFirst(): lấy đầu
  + getLast(): lấy cuối
  + peekFirst():
  + peekLast():

1. Kể ra các **class**triển khai từ **Queue Interface, Dequeue Interface**, phân biệt trường hợp sử dụng tương ứng

Queue interface:

* Các class triển khai từ Queue Interface là: LinkedList, PriorityQueue, ArrayDeque
  + LinkedList: - triển khai deque interface, cho phép hoạt động 2 đầu
    - Trường hợp sử dụng:
  + PriorityQueue:
    - Là hàng đợi ưu tiên, các phần tử dc sắp xếp theo thứ tự tự nhiên hoặc có thể tùy chỉnh
    - Ko tuân theo FIFO truyền thống
    - Trường hợp sử dụng: Sử dụng khi cần xử lý các phần tử theo mức ưu tiên, như hệ thống điều phối nhiệm vụ, lập lịch CPU, hoặc giải thuật Dijkstra trong đồ thị.
  + ArrayDeque:
    - Có kích thước cố định, Hỗ trợ đồng bộ hóa, giúp quản lý tài nguyên giữa các luồng.
    - Trường hợp sử dụng: Cần một hàng đợi kích thước cố định trong môi trường đa luồng, ví dụ như hàng đợi nhiệm vụ trong mô hình Producer-Consumer.
* Các interface kế thừa: Deque, BlockingQueue, BlockingDeque

Deque:

* Các class triển khai: ArrayDeque và LinkedList
  + ArrayDeque:
    - Là một hàng đợi hai được triển khai bằng cách sử dụng mảng động.
    - Không giới hạn kích thước, nhưng không đồng bộ hóa.
    - Hiệu suất tốt hơn so với LinkedList cho các thao tác thêm/xóa phần tử ở cả hai đầu.
    - Trường hợp sử dụng: Khi cần hàng đợi hoặc ngăn xếp hiệu suất cao, chẳng hạn như trong thuật toán tìm kiếm đường đi hoặc thuật toán sắp xếp, ứng dụng ko yêu cầu xử lý đa luồng.
  + LinkedList:
    - Là một hàng đợi hai đầu được triển khai bằng danh sách liên kết đôi, Hỗ trợ cả FIFO (queue) và LIFO (stack).
    - **Trường hợp sử dụng:** Khi cần hàng đợi hoặc ngăn xếp mà không yêu cầu nhiều hiệu suất, như lưu trữ dữ liệu tạm thời trong các ứng dụng nhỏ hoặc đơn luồng.

1. Khi nào nên sử dụng lớp nào:

LinkedList:

* Thích hợp khi cần chèn và xóa phần tử ở giữa danh sách.
* Không tốt cho việc truy cập các phần tử theo chỉ số.

PriorityQueue:

* Thích hợp khi cần duy trì một tập hợp các phần tử được sắp xếp theo ưu tiên.
* Sử dụng trong các thuật toán sắp xếp, tìm kiếm ưu tiên, ...

ArrayDeque:

* Hiệu quả cho các hoạt động chèn và xóa ở cả hai đầu.
* Thích hợp cho việc triển khai stack, queue, và các cấu trúc dữ liệu khác.