Queue

Queue Interface

Tuân thủ theo nguyên tắc fifo, phần tử nào vào trước thì sẽ ra trước

Vào 1 đầu và ra 1 đầu

Deque Interface

tuân thủ nguyên tắc fifo hoặc lifo

Cho phép vào ra 2 đầu

Các class triển khai queue interface:

* **PriorityQueue**
* **ArrayBlockingQueue**
* **LinkedBlockingQueue**
* **PriorityBlockingQueue**
* **ConcurrentLinkedQueue**
* **SynchronousQueue**

**PriorityQueue**

Các phần tử sắp xếp theo thứ tự tự nhiên hoặc theo comparator tuỳ chỉnh

Không tuân thủ theo nguyên tắc fifo và phần tử nào có độ ưu tiên cao hơn sẽ được lấy ra trước.

Phù hợp cho bài toán cần xử lý trước các yêu cầu quan trọng.

**ArrayBlockingQueue**

Là một hàng đợi có kích thước cố định, được sử dụng cho việc đồng bộ hóa giữa các luồng.

Tất cả các thao tác thêm và lấy ra đều được đồng bộ, đảm bảo an toàn trong môi trường đa luồng.

Phù hợp cho bài toàn cần kiểm soát số lượng hàng đợi.

**LinkedBlockingQueue**

Tương tự như ArrayBlockingQueue, nhưng kích thước có thể không cố định

**PriorityBlockingQueue**

Kết hợp giữa PriorityQueue và BlockingQueue, cung cấp khả năng đồng bộ hóa và sắp xếp theo độ ưu tiên.

Phù hợp cho các tác vụ yêu cầu xử lý theo độ ưu tiên, trong khi vẫn cần đồng bộ hóa.

**ConcurrentLinkedQueue**

Một hàng đợi liên kết không giới hạn kích thước, được thiết kế cho việc truy cập đồng thời an toàn.

Cung cấp hiệu suất cao trong môi trường đa luồng với không cần khóa.

**SynchronousQueue**

Là hàng đợi không có khả năng lưu dữ liệu. Mỗi phần tử thêm vào phải được lấy ra ngay lập tức.

Các class triển khai dequeue

**ArrayDeque**

Là một hàng đợi có kích thước thay đổi

Không đồng bộ hóa, do đó không an toàn khi sử dụng trong môi trường đa luồng.

Hàng đợi nằm liên tiếp trong bộ nhớ như 1 mảng

**LinkedList**

Thêm xoá phần tử ở đầu hoặc cuối rất nhanh

**ConcurrentLinkedDeque**

Là một hàng đợi liên kết không giới hạn kích thước, được thiết kế cho việc truy cập đồng thời an toàn.

Hiệu suất làm việc cao.

Không có giá trị null