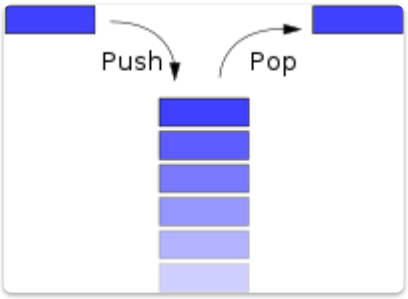
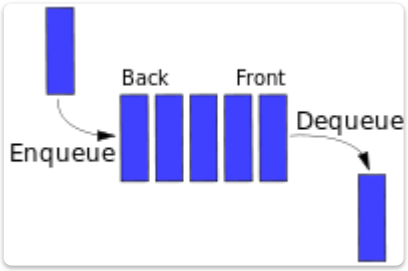
Stack & Queue & Heap

📌 스택 (Stack)



- LIFO (Last In First Out) 형태로 데이터를 저장하는 구조
- 주요 동작
  - `push` : 스택에 자료를 삽입한다.
  - `pop` : 스택에서 자료를 삭제한다. (최상단의 값 삭제)
  - `peek` : 스택에서 최상단의 값을 가져온다.
- 사용 사례
  - stack memory
  - stack frame

📌 큐 (Queue)



- FIFO (FIrst In First Out) 형태로 데이터를 저장하는 구조
- 주요 동작
  - `enqueue` : 큐에 자료를 삽입한다.
  - `dequeue` : 큐에서 자료를 삭제한다. (가장 먼저 들어간 값)
  - `peek` : 큐에서 가장 앞에 있는 값을 가져온다.
- 사용 사례
  - producer/consumer architecture

💡 기술 문서에서 큐를 만났을 때 팁

- 항상 FIFO를 의미하지는 않음.
- 대기 공간으로서의 큐를 의미하기도 함.

💡 스택/큐 관련 에러와 해결 방법

- StackOverflowError
  - 스택 메모리 공간을 다 썼을 때 발생하는 에러
  - 재귀함수(recursive function)에서 탈출 못해서 발생
  - 탈출 조건을 잘 작성해주면 된다.
- OutOfMemoryError
  - Java의 힙(heap) 메모리를 다 썼을 때 발생
  - 큐에 데이터가 계속 쌓이기만 하고 consume 이 되지 않아 발생
  - 큐 사이즈를 고정
  - 만약 큐가 다 찼다면 ?
    - 예외(exception) 던지기
    - 특별한 값 (null or false) 을 반환
    - 성공할 때까지 영원히 스레드 블락 (block)
    - 제한된 시간만 블락되고 그래도 안되면 포기
    - LinkedBlockingQueue (위 사항을 구현해줌)

📌 우선순위 큐 (Priority Queue)

- 큐와 유사하지만 우선순위가 높은 아이템이 먼저 처리됨.
- 주요 동작
  - `insert` : 큐에 자료를 삽입한다. (우선순위 정보도 함께 넣는다.)
  - `delete` : 큐에서 가장 우선순위가 높은 것을 삭제한다.
  - `peek` : 큐에서 가장 우선순위가 높은 자료의 값을 가져온다.

📌 힙 (heap)

- 주로 이진 트리(binary tree) 기반으로 구현
  - 트리(tree) : `부모-자녀` 처럼 계층적인 형태를 가지는 구조

💡 max heap

- 부모 노드의 키(key)가 자식 노드(들)의 키(key)보다 크거나 같은 트리

💡 min heap

- 부모 노드의 키(key)가 자식 노드(들)의 키(key)보다 작거나 같은 트리
- 주요 동작
  - `insert`
  - `delete`
  - `peek`

💡 Priority Queue 와 Heap 의 관계

- 힙(heap)의 키(key)를 우선순위(priority)로 사용한다면 **힙은 우선순위 큐(priority queue)의 구현체가 된다.**
- Priority queue = ADT
- Heap = data structure

💡 우선순위 큐와 힙의 사용 사례

1. 프로세스 스케줄링 (process scheduling) - Priority Queue Scheduling
2. 힙 정렬 (heap sort)

---

https://hihiha2.tistory.com/175
https://velog.io/@yyj8771/%EC%9E%90%EB%A3%8C%EA%B5%AC%EC%A1%B0-%ED%81%90-Queue