세상의 속도를
따라잡고 싶다면

Do it!

알고리즘 코딩 테스트

자바 편

이지스퍼블리싱

02

선형자료구조

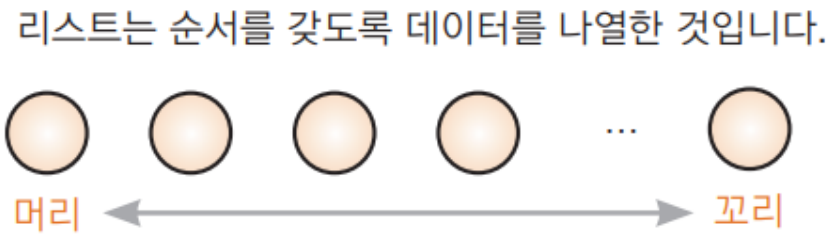
01 리스트

02 스택

03 큐

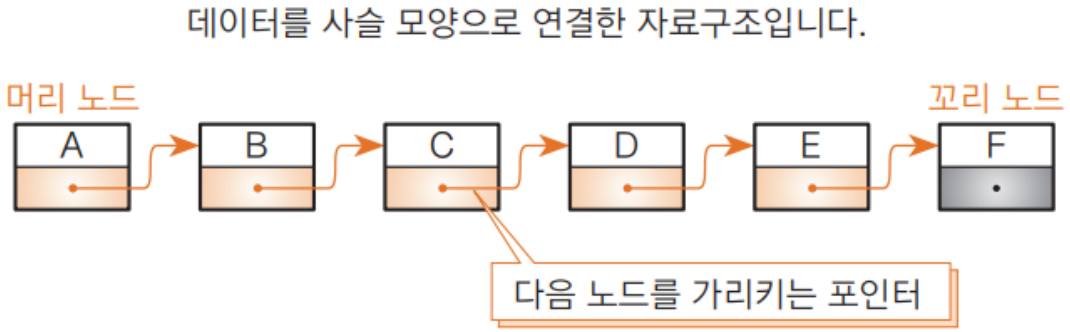
리스트

- 리스트는 데이터를 순서대로 나열한(줄지어 늘어놓은) 자료구조
  - 선형 리스트 : 데이터가 배열처럼 연속하는 메모리 공간에 저장된 자료구조
  - 연결 리스트 : 데이터가 메모리 공간에 연속적으로 저장되어 있지 않더라도 노드 포인터를 이용해 연결된 자료구조



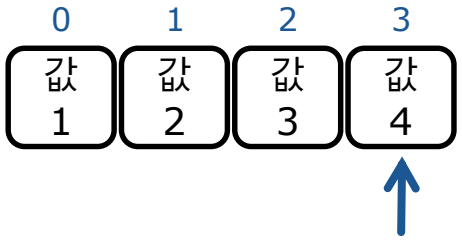
연결 리스트

- 메모리 동적 할당을 기반으로 구현된 자료구조
- 각 데이터들이 연속적으로 저장되지 않지만 데이터 안에 다음 데이터에 대한 정보를 갖는 자료구조
- 동적할당 기반이므로 사이즈를 미리 지정할 필요 없음



연결 리스트

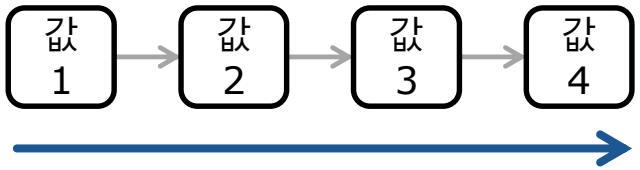
배열의 데이터 조회



인덱스로 데이터 바로 접근

시간복잡도 O(1)

리스트의 데이터 조회

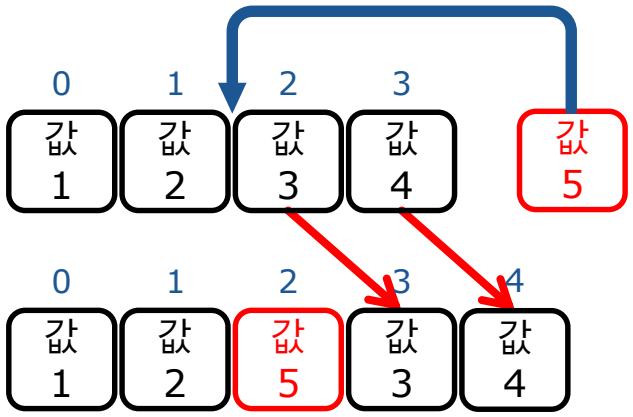


첫 노드부터 순차적으로 접근

시간복잡도 O(N)

연결 리스트

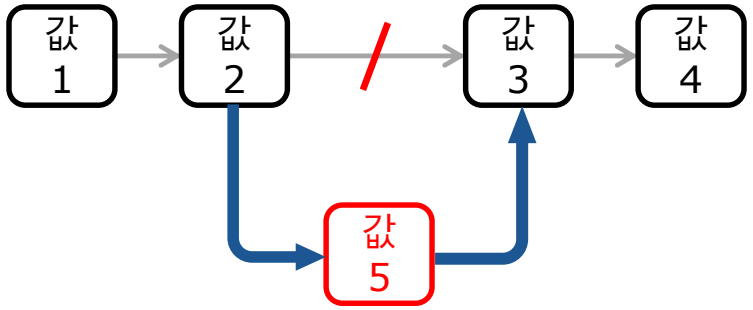
배열의 데이터 삽입



삽입 후 나머지 데이터들 값 이동

시간복잡도 O(N)

리스트의 데이터 삽입

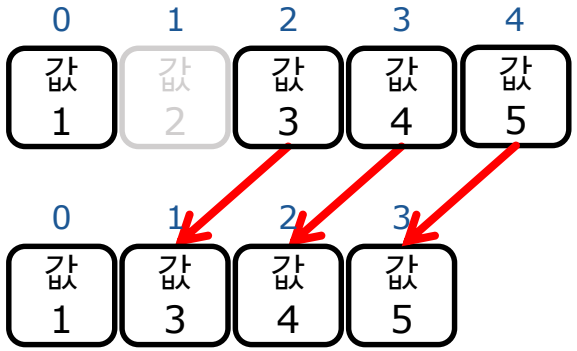


삽입 위치 전후 데이터의 연결만 변경

시간복잡도 O(1)

연결 리스트

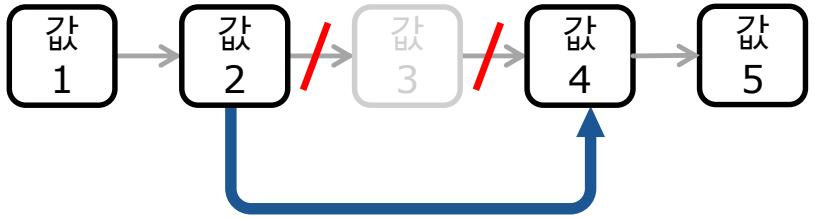
배열의 데이터 삭제



삭제 후 나머지 데이터들 값 이동

시간복잡도 O(N)

리스트의 데이터 삭제

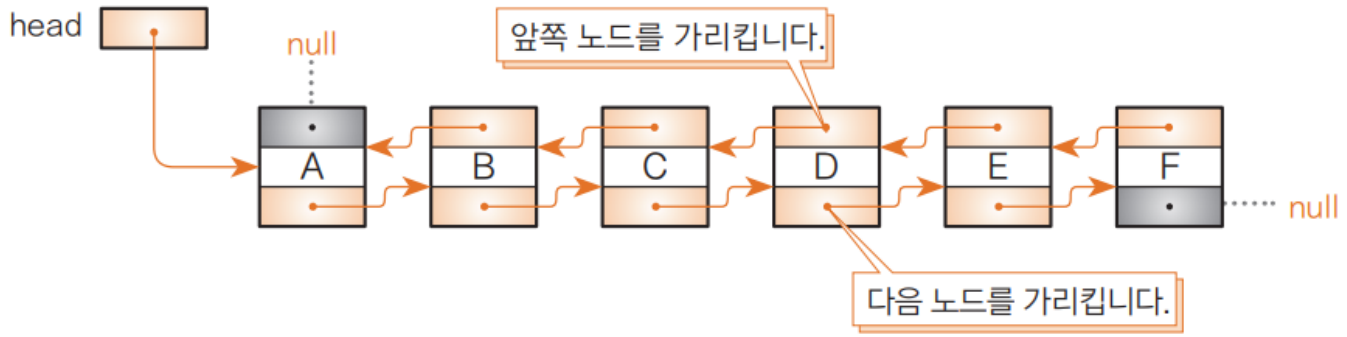


삭제 위치 이전 데이터의 연결만 변경

시간복잡도 O(1)

이중 연결 리스트

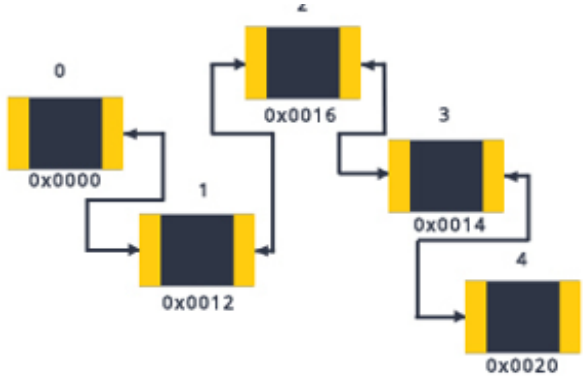
- 이중 연결 리스트는 각 노드마다 이전 노드/ 다음 노드에 대한 포인터가 존재
- 양방향으로 연결되어 노드를 양방향으로 탐색 가능
- API 에 구현된 연결리스트는 대부분 이중 연결 리스트 사용



배열과 리스트

- 배열의 특징
  - 인덱스를 이용해서 바로 접근 가능
  - 새로운 값을 삽입하거나 특정 인덱스 값 삭제 비효율적
  - 배열의 크기는 선언할 때 지정. 크기를 늘리거나 줄이려면 재할당 필요
  - 구조가 간단하여 쉽게 사용 가능

- 리스트의 특징
  - 인덱스가 없으므로 값에 접근 시 Head 포인터부터 순차적 접근 (접근속도 느림)
  - 포인터로 연결되어 있으므로 데이터 삽입/삭제 연산 빠름
  - 리스트는 크기가 정해져 있지 않으므로, 크기가 변하기 쉬운 데이터를 다룰 때
  - 포인터를 저장할 공간이 필요하므로 배열보다 구조가 복잡하고 메모리를 더 많 . . .



배열과 리스트

- 기본 배열
  - 간단하게 사용 가능
  - Primitive/Object 타입 모두 넣을 수 있음
  - 사이즈 고정

```
int [] arr = new int [5];
```

- java.util.ArrayList
  - 배열 기반의 자료구조
  - Object 타입만 넣을 수 있음(Integer, Boolean 등)
  - 사이즈 가변적

```
ArrayList <Integer> arr = new ArrayList<>();
```

- java.util.LinkedList
  - 이중 연결 리스트 기반의 자료구조
  - Object 타입만 넣을 수 있음(Integer, Boolean 등)
  - 사이즈 가변적

```
LinkedList <Integer> link_list = new LinkedList<>();
```

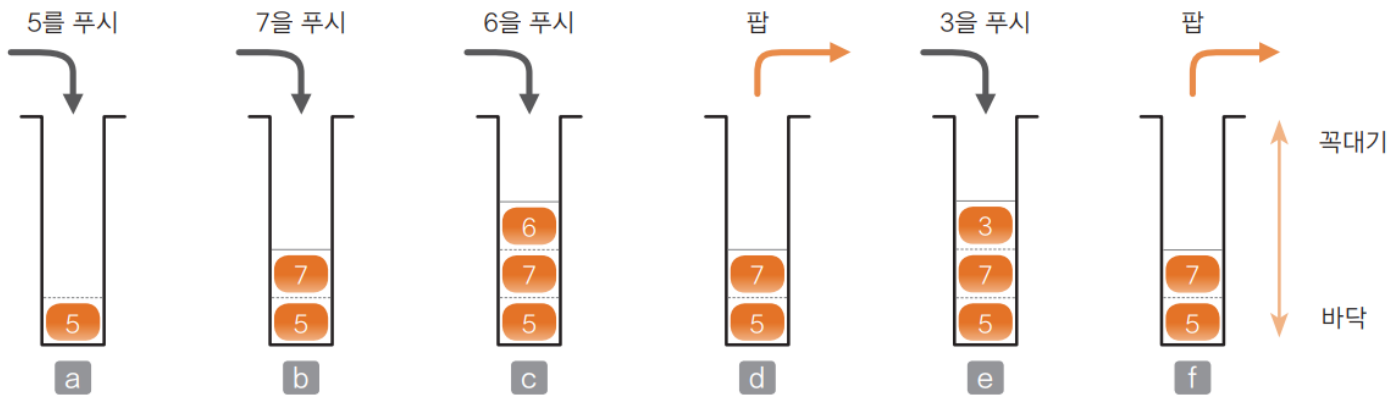
연습문제

- 배열(ArrayList)과 리스트(LinkedList)를 생성하고 다음을 수행하세요.
  - 반복문을 사용하여 배열의 0번 인덱스에 값을 추가하세요.
  - 반복문을 사용하여 리스트의 0번 인덱스에 값을 추가하세요.
  - 각 반복문의 실행 시간을 계산하여 출력하세요.
  - 반복문의 횟수를 10000번, 100000번, 500000번으로 변경하면서 실행 시간을 확인하세요.

```
long start, end;
start = System.currentTimeMillis(); // 시작 시간
// 여기에 반복문을 작성하세요.
end = System.currentTimeMillis(); // 종료 시간
System.out.println("반복문 실행시간(ms) : " + (end - start));
```

스택

- 스택(stack)은 데이터를 쌓아 올린 형태의 자료구조
- 데이터의 입출력 순서는 후입선출(LIFO : Last In First Out) 로 가장 나중에 넣은 데이터를 가장 먼저 꺼냄
- 스택에 데이터를 넣는 작업을 푸시(push)라 하고, 꺼내는 작업을 팝(pop)이라고 한다.



스택

- java.util.Stack
  - push() / add() : 스택에 데이터 삽입
  - pop() : 스택의 맨 위 데이터를 제거하고 그 값을 반환
  - peek() : 스택의 맨 위에 있는 요소의 값을 반환
  - isEmpty() : 스택이 비어 있으면 true를, 그렇지 않으면 false를 반환
  - toString() : 스택에 저장된 값을 문자열로 변환

연습문제

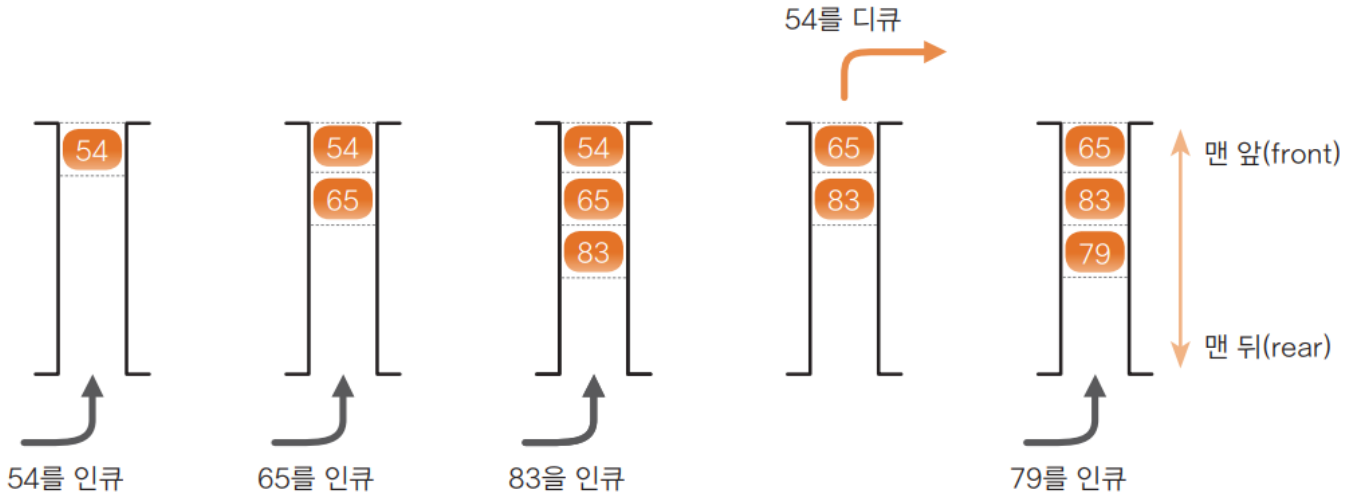
- 스택을 생성하고 다음을 수행하세요.
  - 100, 200, 300, 400, 500을 순서대로 추가하고 스택을 출력하세요.
  - 스택의 맨 위의 값을 제거하고 제거된 값을 변수 top에 저장하고 top과 스택을 출력하세요.
  - 스택의 맨 위의 값을 출력하세요.
  - while 반복문을 사용하여 스택의 모든 값을 제거하고 값이 하나씩 제거될 때마다 스택을 출력하세요.

연습문제

- 백준 온라인 저지 – 17608 막대기

큐

- 큐(queue)는 일렬로 늘어선 데이터를 일시적으로 쌓아 두는 기본 자료구조이다.
- 가장 먼저 넣은 데이터를 가장 먼저 꺼내는 선입선출(FIFO : First In First Out) 구조
- 큐에 데이터를 넣는 작업을 인큐(en-queue)라 하고, 꺼내는 작업을 디큐(de-queue)라고 한다.



큐

- java.util.Queue
  - offer(), add() : 큐에 데이터를 인큐(추가)
  - remove(), poll() : 큐에서 데이터를 디큐(삭제)
  - peek() : 맨 앞의 데이터를 반환
  - isEmpty() : 큐가 비어 있으면 true를, 그렇지 않으면 false를 반환
  - toString() : 큐에 저장된 값을 문자열로 변환

연습문제

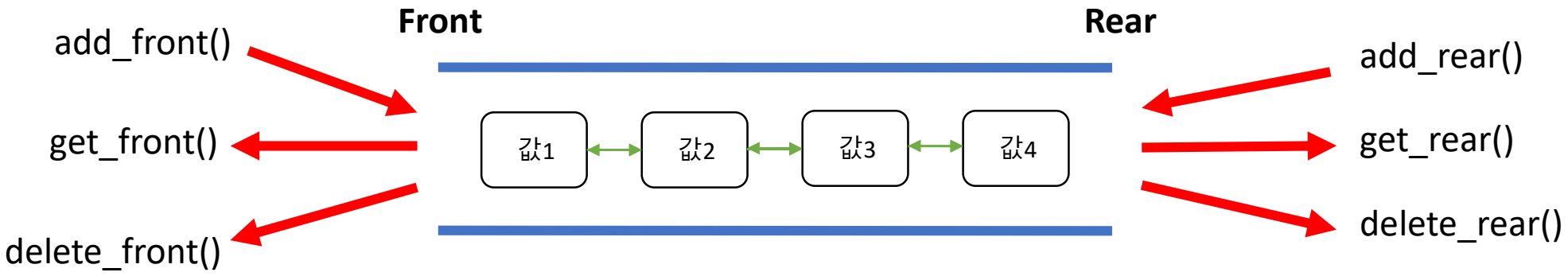
- 큐를 생성하고 다음을 수행하세요.
  - 100, 200, 300, 400, 500을 순서대로 추가하고 큐를 출력하세요.
  - 큐의 맨 앞의 값을 제거하고 제거된 값을 변수 front에 저장 후 front와 큐를 출력하세요.
  - 큐의 맨 앞의 값을 출력하세요.
  - while 반복문을 사용하여 큐의 모든 값을 제거하고 값이 하나씩 제거될 때마다 큐를 출력하세요.

연습문제

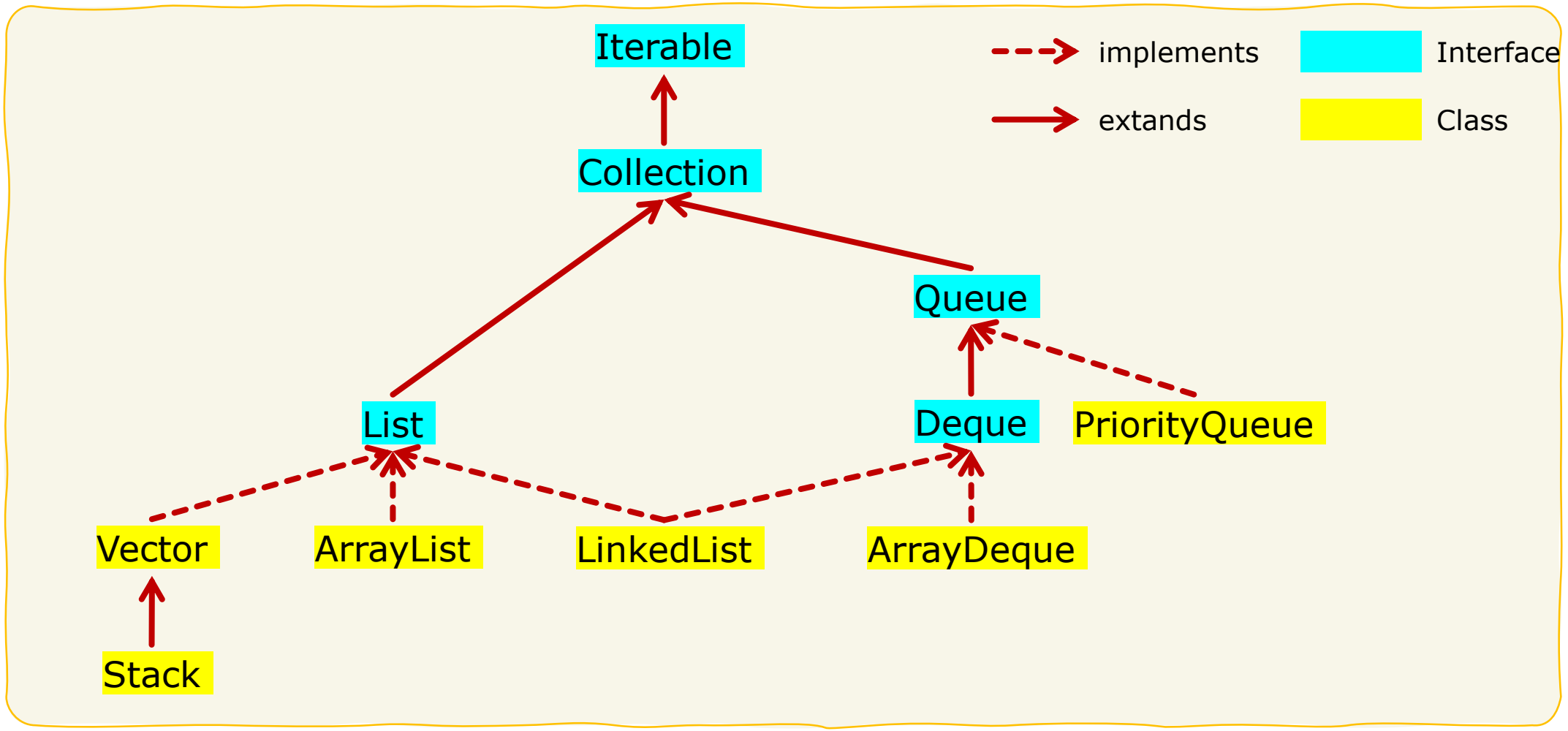
- 백준 온라인 저지 – 2164 카드2

Deque(이중 연결 리스트)

- 덱(Deque) 은 Double-Ended Queue 로 큐의 앞부분(front)과 뒷부분(rear)에서 모두 삽입과 삭제가 가능한 자료구조
- 이중 연결 리스트로 구현
- java.utill.Deque 인터페이스로 구현 가능
- 가장 앞부분을 가리키는 Head 과 가장 뒷부분을 가리키는 Tail 변수가 존재



Java 선형자료구조


Iterable
implements
Interface
extands
Class
Collection
Queue
List
Deque
PriorityQueue
Vector
ArrayList
LinkedList
ArrayDeque
Stack

연습문제

- 백준 온라인 저지 – 1874 스택 수열
- 백준 온라인 저지 – 1158 요세푸스 문제
- 백준 온라인 저지 – 2493 탑