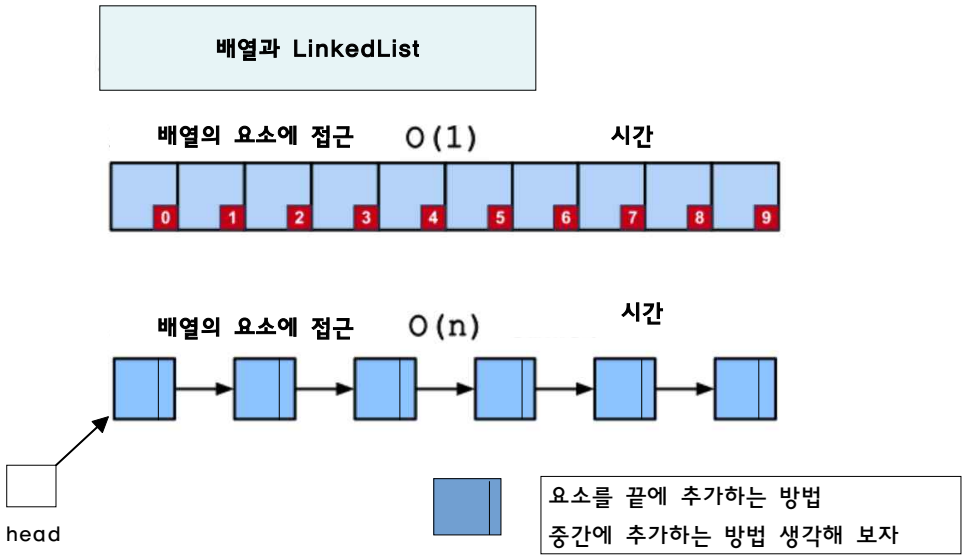
◓ LinkedList

▼ 어레이 vs. 연결리스트



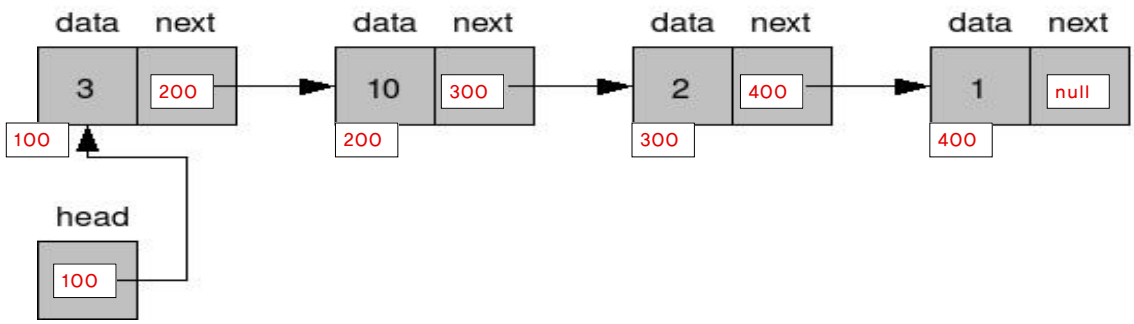
◉ Array vs LinkedList 비교

(O(1) : 빅오1 , O(N) :빅오N : 알고리즘 성능 표기법)

	Array	LinkedList
특정 원소 조회	O(1) : 빅오1	O(N) :빅오N
중간에 삽입 삭제	O(N)	O(1)
데이터 추가	데이터 추가 시 모든 공간이 다 차버렸다면 새로운 메모리 공간을할당받아야 한다	모든 공간이 다 찼어도 맨 뒤의 노드만 동적으로 추가하면 된다.
정리	데이터에 접근하는 경우가 빈번하다면 Array를 사용하자	삽입과 삭제가 빈번하다면 LinkedList를 사용하는것이 더 좋다.

```
class ListNode {
    public int val
    public ListNode next

    public ListNode() {}

    public ListNode(int val, ListNode next) {
        this.val = val
        this.next = next
    }
}

public class MyLinkedList{
    public ListNode head

    public LinkedListTest() {
        this.head = null
    }

    public void append(int val) {    //뒤에 추가하기
        if (head == null) {
            head = new ListNode(val, null);
            return
        }

        ListNode node = head
        while (node.next != null) {
            node = node.next
        }

        node.next = new ListNode(val, null);
    }

    public void delete(int val) {
        if (head == null) {
            return
        }

        if (head.val == val) {
            head = head.next
            return
        }

        ListNode current = head
        ListNode prev = null

        while (current != null && current.val != val) {
            prev = current
            current = current.next
        }

        if (current == null) {
            return
        }

        prev.next = current.next
    }

 public static void main(String[] args) {
        MyLinkedList linkedList = new MyinkedList();

        linkedList.append(1);
        linkedList.append(2);
        linkedList.append(3);

        System.out.println("Linked List before deletion:");
        displayLinkedList(linkedList);

         linkedList.delete(3);

         System.out.println("\nLinked List after deletion:");
         displayLinkedList(linkedList);
    }

     private static void displayLinkedList(MyLinkedList linkedList) {
        ListNode current = linkedList.head
        while (current != null) {
            System.out.print(current.val + " ");
            current = current.next
        }
        System.out.println();
    }
}
```

◉ 2를 가진 노드를 삭제하고 싶다면 어떻게 해야 할지 생각해 봅시다 !!



자신이 좋아하는 음식 5개를 링크드리스트로 표현하시오
음식하나를 (특정 음식 뒤에) 추가하시오
기존음식에서 하나를 삭제하시오

```java
import java.util.LinkedList;

public class FavoriteFoods {
    public static void main(String[] args) {
        // 링크드 리스트 생성
        LinkedList<String> favoriteFoods = new LinkedList<>();

        // 음식 추가
        favoriteFoods.add("짜장면");
        favoriteFoods.add("피자");
        favoriteFoods.add("초밥");
        favoriteFoods.add("스테이크");
        favoriteFoods.add("아이스크림");

        // 링크드 리스트 출력
        System.out.println("좋아하는 음식 목록: " + favoriteFoods);

        // 음식 추가
        addFood(favoriteFoods, "파스타");

        // 링크드 리스트 출력
        System.out.println("음식 추가 후 목록: " + favoriteFoods);

        // 음식 삭제
        removeFood(favoriteFoods, "초밥");

        // 링크드 리스트 출력
        System.out.println("음식 삭제 후 목록: " + favoriteFoods);
    }

    // 음식 추가 메서드
    private static void addFood(LinkedList<String> list, String food) {
        // 특정 음식 뒤에 추가
        list.add(list.indexOf("스테이크") + 1, food);
    }

    // 음식 삭제 메서드
    private static void removeFood(LinkedList<String> list, String food) {
        // 특정 음식 삭제
        list.remove(food);
    }
}
```

```
class Node {
    String data;
    Node next;

    public Node(String data) {
        this.data = data;
        this.next = null;
    }
}

class LinkedList {
    Node head;

    public LinkedList() {
        this.head = null;
    }

    // 맨 뒤에 노드 추가
    public void addNode(String data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
        } else {
            Node current = head;
            while (current.next != null) {
                current = current.next;
            }
            current.next = newNode;
        }
    }

    // 특정 노드 뒤에 노드 추가
    public void addNodeAfter(String existingData, String newData) {
        Node newNode = new Node(newData);
        Node current = head;
        while (current != null) {
            if (current.data.equals(existingData)) {
                newNode.next = current.next;
                current.next = newNode;
                break;
            }
            current = current.next;
        }
    }

    // 노드 삭제
    public void deleteNode(String data) {
        if (head == null) {
            return;
        }

        if (head.data.equals(data)) {
            head = head.next;
            return;
        }

        Node current = head;
        while (current.next != null) {
            if (current.next.data.equals(data)) {
                current.next = current.next.next;
                break;
            }
            current = current.next;
        }
    }
```

LinkedList 직접 만들어서 사용

```
    // 링크드 리스트 출력
    public void printList() {
        Node current = head;
        while (current != null) {
            System.out.print(current.data + " ");
            current = current.next;
        }
        System.out.println();
    }
}

public class FavoriteFoods {
    public static void main(String[] args) {
        LinkedList favoriteFoods = new LinkedList();

        // 음식 추가
        favoriteFoods.addNode("짜장면");
        favoriteFoods.addNode("피자");
        favoriteFoods.addNode("초밥");
        favoriteFoods.addNode("스테이크");
        favoriteFoods.addNode("아이스크림");

        // 링크드 리스트 출력
        System.out.print("좋아하는 음식 목록: ");
        favoriteFoods.printList();

        // 음식 추가
        favoriteFoods.addNodeAfter("스테이크", "파스타");

        // 링크드 리스트 출력
        System.out.print("음식 추가 후 목록: ");
        favoriteFoods.printList();

        // 음식 삭제
        favoriteFoods.deleteNode("초밥");

        // 링크드 리스트 출력
        System.out.print("음식 삭제 후 목록: ");
        favoriteFoods.printList();
    }
}
```