



2020 학년도 1 학기

컴퓨터 정보과

자료구조(Data Structures)

담당교수 : 김주현

제 3 주차 / 제 2 차시

<https://www.ttsdemo.com/>

»oddcast

Text-to-Speech

Aa

Language

🇰🇷 Korean

▼

Voice

Junwoo

▼

▶ sample

Effect

None



▼

Level


간단한 테스트입니다.

11/600 characters


Say It ▶



Try Our Facecards ▶



Learn About Our Products



❖ 순차 자료구조의 문제점

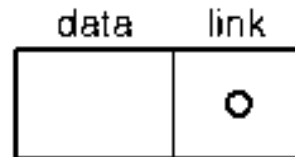
- 삽입연산이나 삭제 연산 후에 연속적인 물리 주소를 유지하기 위해서 원소들을 이동시키는 추가적인 작업과 시간 소요
 - 원소들의 이동 작업으로 인한 오버헤드는 원소의 개수가 많고 삽입 · 삭제 연산이 많이 발생하는 경우에 성능상의 문제 발생
- 순차 자료구조는 배열을 이용하여 구현하기 때문에 배열이 갖고 있는 메모리 사용의 비효율성 문제를 그대로 가짐
- 순차 자료구조에서의 연산 시간에 대한 문제와 저장 공간에 대한 문제를 개선한 자료 표현 방법 필요

연결 리스트(Linked List)

각 노드가 데이터와 포인터를 가지고 한 줄로 연결되어 있는 방식으로 데이터를 저장하는 자료 구조

노드

- 연결 자료구조에서 하나의 원소를 표현하기 위한 단위 구조
- <원소, 주소>의 구조

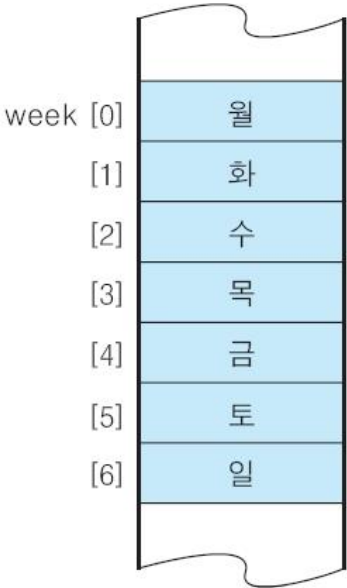


- 데이터 필드(data field)
 - 원소의 값을 저장
 - 저장할 원소의 형태에 따라서 하나 이상의 필드로 구성
- 링크 필드(link field)
 - 다음 노드의 주소를 저장
 - 포인터 변수를 사용하여 주소값을 저장

선형 리스트와 연결 리스트의 비교

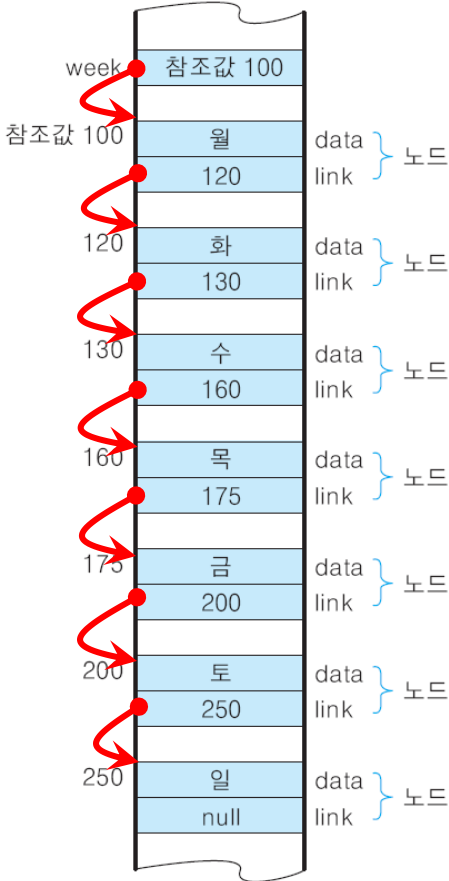
리스트 week=(월, 화, 수, 목, 금, 토, 일)
week에 대한 선형 리스트

	[0]	[1]	[2]	[3]	[4]	[5]	[6]
week	월	화	수	목	금	토	일



(b) 물리구조

리스트 week=(월, 화, 수, 목, 금, 토, 일)
week에 대한 연결 리스트



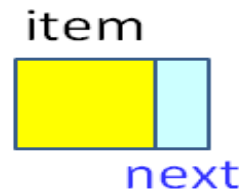
단순 연결 리스트

- 단순연결리스트(Singly Linked List)는 동적 메모리 할당을 이용해 리스트를 구현하는 가장 간단한 형태의 자료구조
- 동적 메모리 할당을 받아 노드(node)를 저장하고, 노드는 레퍼런스를 이용하여 다음 노드를 가리키도록 만들어 노드들을 한 줄로 연결시킴



단순연결리스트의 노드를 위한 Node 클래스

```
01 public class Node <E> {  
02     private E      item;  
03     private Node<E> next;  
04     public Node(E newItem, Node<E> node){ // 생성자  
05         item = newItem;  
06         next = node;  
07     }  
08     // get 과 set 메소드들  
09     public E      getItem() { return item; }  
10     public Node<E> getNext() { return next; }  
11     public void    setItem(E newItem)      { item = newItem; }  
12     public void    setNext(Node<E> newNext){ next = newNext; }  
13 }
```



Node 객체의 간략한 표현

리스트를 단순연결리스트로 구현한 SList 클래스

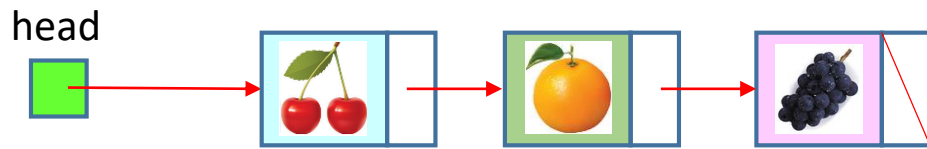
```
01 import java.util.NoSuchElementException;
02 public class SList <E> {
03     protected Node head; // 연결 리스트의 첫 노드 가리킴
04     private int size;
05     public SList(){        // 연결 리스트 생성자
06         head = null;
07         size = 0;
08     }
    // 탐색, 삽입, 삭제 연산을 위한 메소드 선언
}
```

탐색

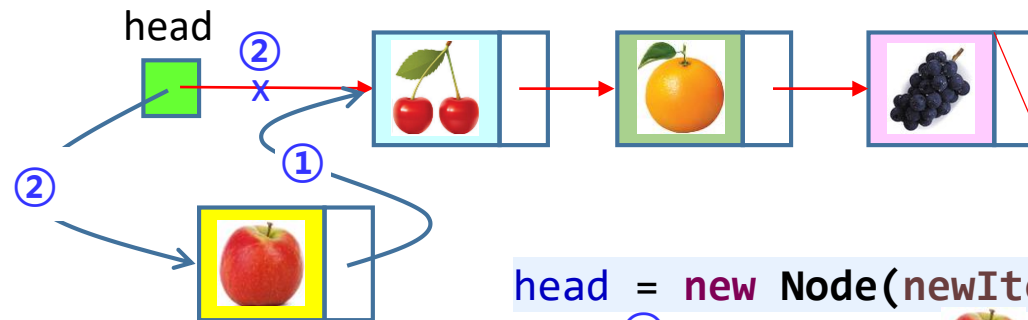
```
01 public int search(E target) { // target을 탐색
02     Node p = head;
03     for (int k = 0; k < size ;k++){
04         if (target == p.getItem()) return k;
05         p = p.getNext();
06     }
07     return -1; // 탐색을 실패한 경우 -1 리턴
08 }
```

삽입 : insertFront() 메소드

```
01 public void insertFront(E newItem){ // 연결리스트 맨 앞에 새 노드 삽입
02     head = new Node(newItem, head);
03     size++;
04 }
```



새 노드 삽입 전

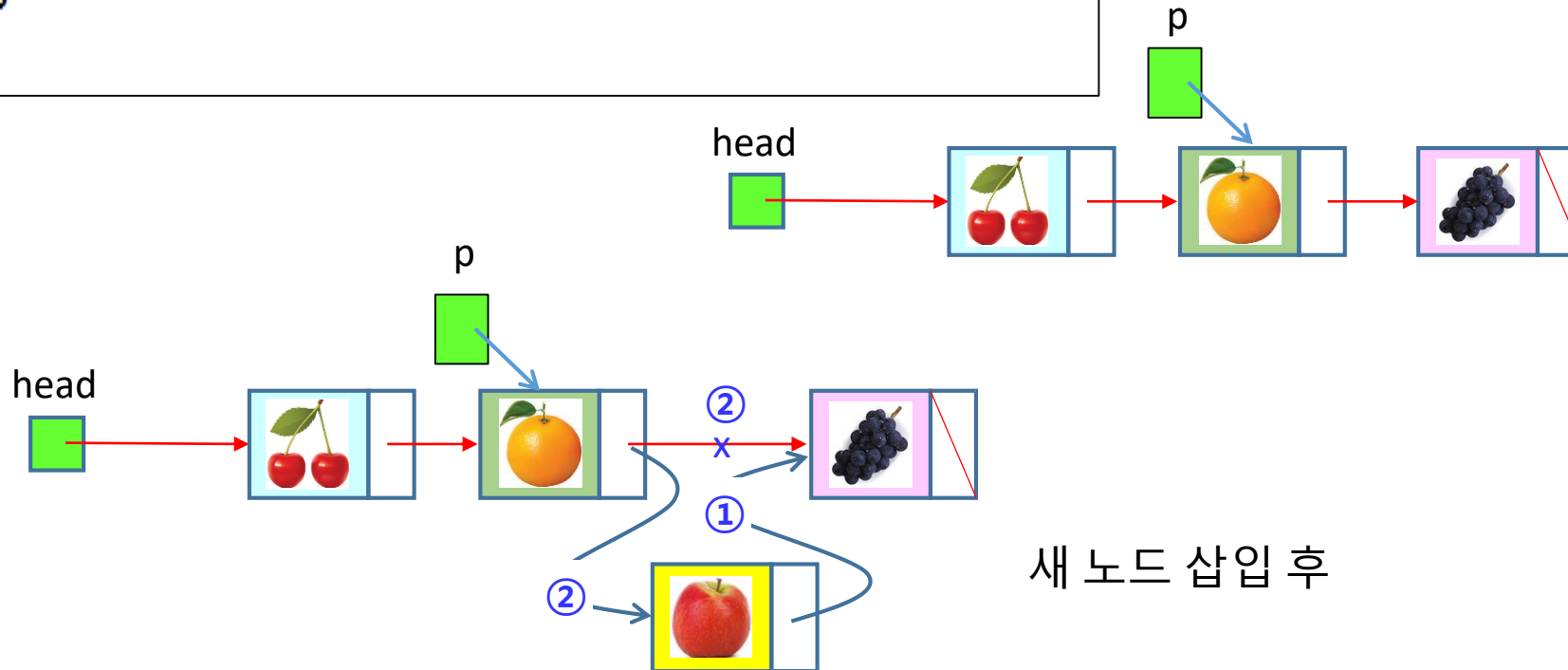


head = new Node(newItem, head);

② ①

삽입 : insertAfter() 메소드

```
01 public void insertAfter(E newItem, Node p){ // 노드 p 바로 다음에 새 노드 삽입
02     p.setNext(new Node(newItem, p.getNext()));
03     size++;
04 }
```



```
p.setNext(new Node(newItem, p.getNext()));
```

②



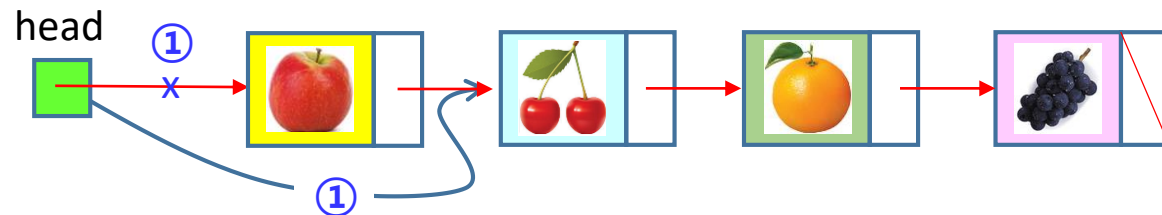
①

삭제 : deleteFront() 메소드

```
01 public void deleteFront(){           // 리스트의 첫 노드 삭제
02     if (size == 0) throw new NoSuchElementException();
03     head = head.getNext();
04     size--;
05 }
```



삭제 전



삭제 후

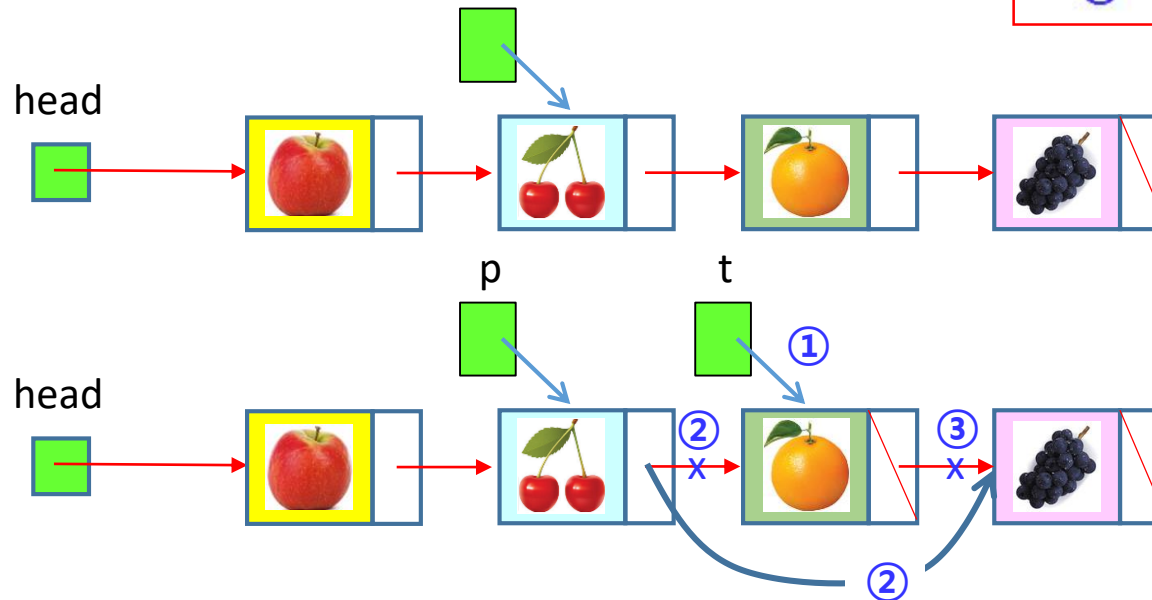
```
head = head.getNext();
```

①

삭제 : deleteAfter() 메소드

```
01 public void deleteAfter(Node p){ // p가 가리키는 노드의 다음 노드를 삭제
02     if (p == null) throw new NoSuchElementException();
03     Node t = p.getNext();
04     p.setNext(t.getNext());
05     t.setNext(null);
06     size--;
07 }
```

- ① Node t = p.getNext();
- ② p.setNext(t.getNext());
- ③ t.setNext(null);



삭제 전

삭제 후

Node

```
public class Node <E extends Comparable<E>>{
    private E    item;
    private Node next;

    public Node(E newItem, Node<E> p){ // 생성자
        item = newItem;
        next = p;
    }
    // get 메소드와 set 메소드
    public E    getItem() { return item;}
    public Node getNext() { return next;}
    public void setItem(E newItem) { item = newItem;}
    public void setNext(Node n)      { next = n;}
}
```

SList

```
import java.util.NoSuchElementException;
public class SList <E extends Comparable<E>> {

    protected Node head; // 연결 리스트의 첫 노드 가리킴
    private int size;
    public Node getHead() { return head; }
    public void setHead(Node n) { head= n;}
    public SList(){ // 연결 리스트 생성자
        head = null;
        size = 0;
    }
    public int size() { return size; }
    public boolean isEmpty() { return size == 0; }

    public void insertFront(E newItem){ // 연결리스트 맨 앞에 새 노드 삽입
        head = new Node(newItem, head);
        size++;
    }

    public void insertAfter(E newItem, Node p){ // 노드 p 바로 다음에 새 노드 삽입
        p.setNext(new Node(newItem, p.getNext()));
        size++;
    }
}
```



```
public void deleteFront(){    // 리스트의 첫 노드 삭제
    if (isEmpty()) throw new NoSuchElementException();
    head = head.getNext();
    size--;
}

public void deleteAfter(Node p){ // p가 가리키는 노드의 다음 노드를 삭제
    if (p == null) throw new NoSuchElementException();
    Node t = p.getNext();
    p.setNext(t.getNext());
    t.setNext(null);
    size--;
}

public int search(E target) { // target을 탐색
    Node p = head;
    for (int k = 0; k < size ;k++){
        if (target == p.getItem()) return k;
        p = p.getNext();
    }
    return -1; // 탐색을 실패한 경우 -1 리턴
}
```

```
public void print(){ // 연결 리스트 노드들의 항목들을 차례로 출력
    for (Node p = head; p != null; p = p.getNext())
        System.out.print(p.getItem()+"\t ");
    System.out.println();
}
```

main

```

1 public class main {
2     public static void main(String[] args) {
3
4         SList<String> s = new SList<String>(); // 연결 리스트 객체 s 생성
5         s.insertFront("orange"); s.insertFront("apple");
6         s.insertAfter("cherry", s.head.getNext());
7         s.insertFront("pear");
8
9         s.print();
10        System.out.println(": s의 길이 = "+s.size()+"\n");
11        System.out.println("체리가 \t"+s.search("cherry")+"번째에 있다.");
12        System.out.println("키위가 \t"+s.search("kiwi")+"번째에 있다.\n");
13        s.deleteAfter(s.head);
14        s.print();
15        System.out.println(": s의 길이 = "+s.size());System.out.println();
16        s.deleteFront();
17        s.print();
18        System.out.println(": s의 길이 = "+s.size());System.out.println();
19
20        SList<Integer> t = new SList<Integer>(); // 연결 리스트 객체 t 생성
21        t.insertFront(500); t.insertFront(200);
22        t.insertAfter(400, t.head);
23        t.insertFront(100);
24        t.insertAfter(300, t.head.getNext());
25        t.print();
26        System.out.println(": t의 길이 = "+t.size());
27    }
28 }

```

Problems @ Javadoc Console Console

<terminated> main (48) [Java Application] C:\Program Files\Java\jdk1.8.0_40\bin\javaw.exe

pear apple orange cherry : s의 길이 = 4

체리가 3번째에 있다.

키위가 -1번째에 있다. ← -1은 리스트에 없다는 의미

pear orange cherry : s의 길이 = 3

orange cherry : s의 길이 = 2

100 200 300 400 500 : t의 길이 = 5

```
class ListNode{
    private String data;
    public ListNode link;
    public ListNode(){
        this.data = null;
        this.link = null;
    }
    public ListNode(String data){
        this.data = data;
        this.link = null;
    }
    public ListNode(String data, ListNode link){
        this.data = data;
        this.link = link;
    }
    public String getData(){
        return this.data;
    }
}
```

```
class LinkedList{
    private ListNode head;
    public LinkedList(){
        head = null;
    }
    public void insertMiddleNode(ListNode pre, String data){
        ListNode newNode = new ListNode(data);
        newNode.link = pre.link;
        pre.link = newNode;
    }
    public void insertLastNode(String data){
        ListNode newNode = new ListNode(data);
        if(head == null){
            this.head = newNode;
        }
        else{
            ListNode temp = head;
            while(temp.link != null) temp = temp.link;
            temp.link = newNode;
        }
    }
}
```

```
public void deleteLastNode(){
    ListNode pre, temp;
    if(head == null) return;
    if(head.link == null){
        head = null;
    }
    else{
        pre = head;
        temp = head.link;
        while(temp.link != null){
            pre = temp;
            temp = temp.link;
        }
        pre.link = null;
    }
}

public ListNode searchNode(String data){
    ListNode temp = this.head;
    while(temp != null){
        if(data == temp.getData())
            return temp;
        else temp = temp.link;
    }
    return temp;
}
```

```
public void reverseList(){
    ListNode next = head;
    ListNode current = null;
    ListNode pre = null;
    while(next != null){
        pre = current;
        current = next;
        next = next.link;
        current.link = pre;
    }
    head = current;
}

public void printList(){
    ListNode temp = this.head;
    System.out.printf("L = (");
    while(temp != null){
        System.out.printf(temp.getData());
        temp = temp.link;
        if(temp != null){
            System.out.printf(", ");
        }
    }
    System.out.println(")");
}
}
```

```
public class Linked_List_Test{
    public static void main(String args[]){
        LinkedList L = new LinkedList();
        System.out.println("(1) 공백 리스트에 노드 3개 삽입하기");
        L.insertLastNode("월");
        L.insertLastNode("수");
        L.insertLastNode("일");
        L.printList();
        System.out.println("(2) 수 노드 뒤에 금 노드 삽입하기");
        ListNode pre = L.searchNode("수");
        if(pre == null)
            System.out.println("검색실패>> 찾는 데이터가 없습니다.");
        else{
            L.insertMiddleNode(pre, "금");
            L.printList();
        }
        System.out.println("(3) 리스트의 노드를 역순으로 바꾸기");
        L.reverseList();
        L.printList();
        System.out.println("(4) 리스트의 마지막 노드 삭제하기");
        L.deleteLastNode();
        L.printList();
    }
}
```

(1) 공백 리스트에 노드 3개 삽입하기
L = (월, 수, 일)
(2) 수 노드 뒤에 금 노드 삽입하기
L = (월, 수, 금, 일)
(3) 리스트의 노드를 역순으로 바꾸기
L = (일, 금, 수, 월)
(4) 리스트의 마지막 노드 삭제하기
L = (일, 금, 수)



Report

앞에서 살펴본 두 개의 단순 연결 리스트 예제를 테스트하고 그 파일들을 깃허브에 업로드하세요

Reference

- [https://ko.wikipedia.org/wiki/%EB%A6%AC%EC%8A%A4%ED%8A%B8_\(%EC%BB%B4%ED%93%A8%ED%8C%85\)](https://ko.wikipedia.org/wiki/%EB%A6%AC%EC%8A%A4%ED%8A%B8_(%EC%BB%B4%ED%93%A8%ED%8C%85))
- 자바로 배우는 쉬운 자료구조, 이지영, 한빛아카데미
- 자바와 함께하는 자료구조의 이해, 양성봉, 생능출판

언제 어디서나 즐공, 열공, 진공하세요.

감사합니다
