e-learning

2020 학년도 1 학기

컴퓨터 정보과

자료구조(Data Structures)

담당교수 : 김주현

제 3 주차 / 제 1 차시

DIT 동의과학대학교
DONG-EUI INSTITUTE OF TECHNOLOGY

리스트(List)

- 컴퓨터 과학에서 같은 값이 한 번 이상 존재할 수 있는 일련의 값이 모여있는 추상적 자료형
- 객체 지향 프로그래밍 언어에서 리스트는 제네릭 "리스트" 클래스의 서브클래스의 인스턴스로 제공
- 리스트의 구현:
  - 배열(선형 리스트)
  - 단순연결리스트
  - 이중연결리스트
  - 환형연결리스트

선형 리스트(Linear List)

- 순서 리스트(Ordered List)
- 자료들 간에 순서를 갖는 리스트
- 선형 리스트의 예

동창 선형 리스트	
1	김좌진
2	신채호
3	안중근
4	이봉창
5	한용운

좋아하는 음식 선형 리스트	
1	김치찌개
2	닭볶음탕
3	된장찌개
4	잡채
5	북어국

오늘의 할일 선형 리스트	
1	운동
2	자료구조 수업
3	동아리 공연 연습
4	과제 제출
5	방청소

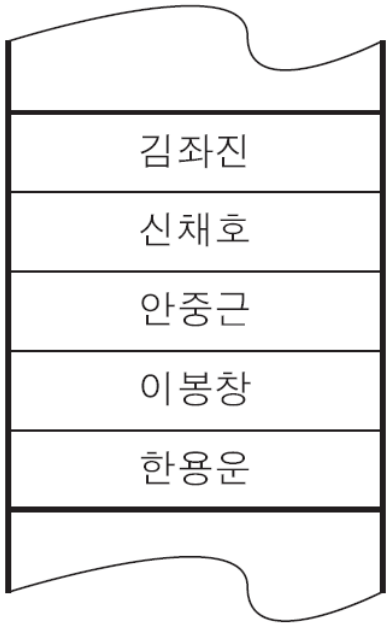
선형 리스트의 표현

리스트이름 = (원소 1, 원소 2, ···, 원소 n)

- 선형 리스트에서 원소를 나열한 순서는 원소들의 순서가 된다

선형 리스트의 저장

- 원소들의 논리적 순서와 같은 순서로 메모리에 저장
- 순차 자료구조
  원소들의 논리적 순서 = 원소들이 저장된 물리적 순서

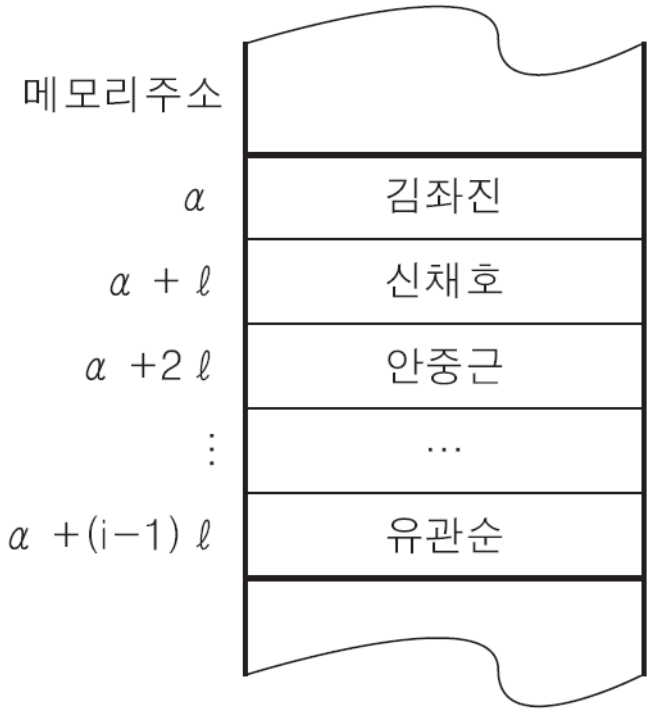


순차 자료구조의 원소 위치 계산

선형 리스트가 저장된 시작 위치 : α

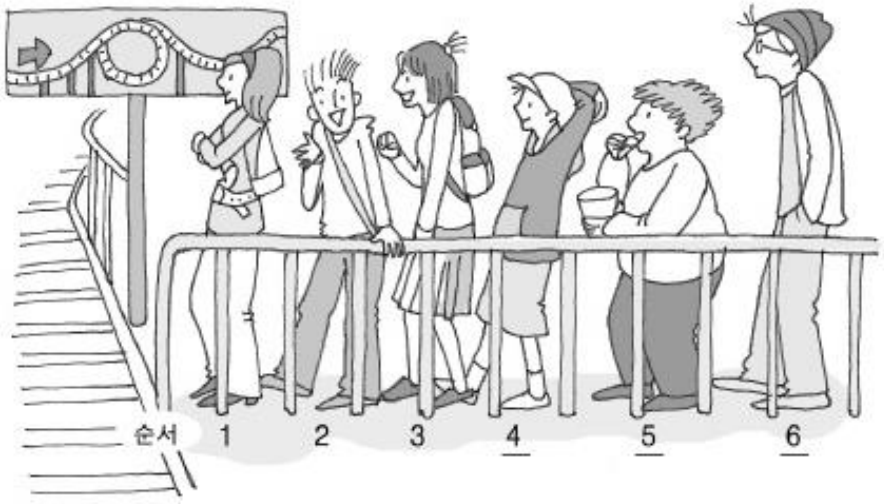
원소의 길이 : ℓ

i번째 원소의 위치 = α + (i-1) x ℓ

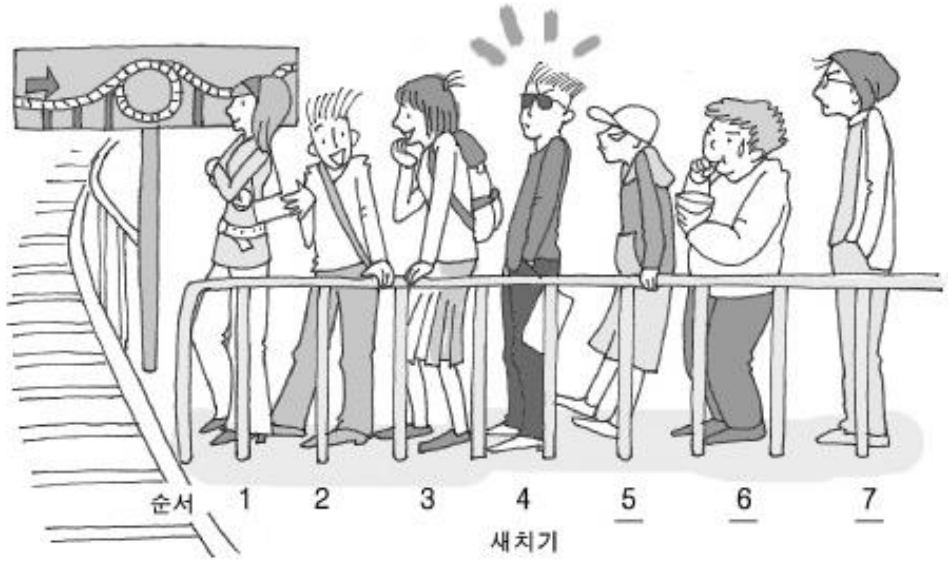


선형 리스트에서의 원소 삽입

- 선형리스트 중간에 원소가 삽입되면, 그 이후의 원소들은 한자리씩 자리를 뒤로 이동하여 물리적 순서를 논리적 순서와 일치시킨다



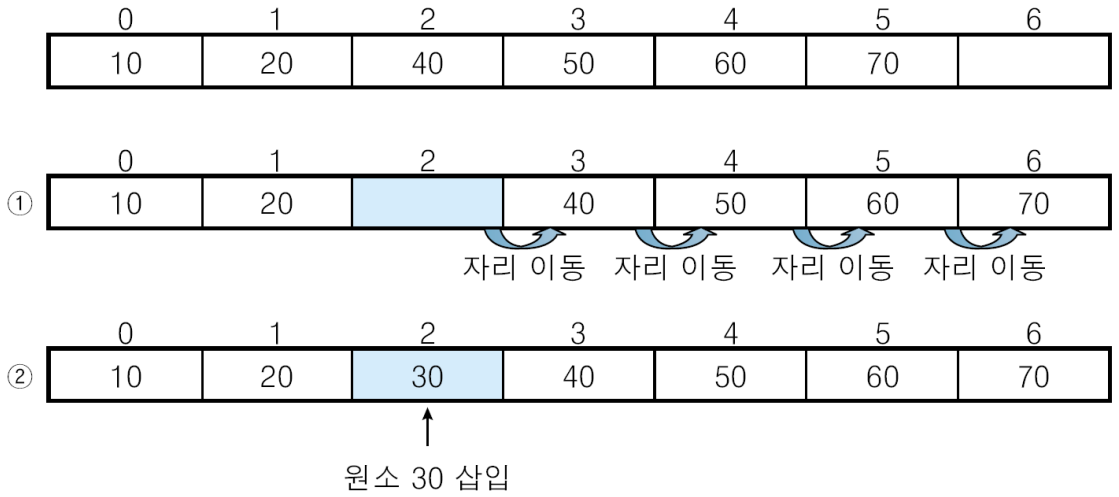
(a) 새치기 전



(b) 새치기 후

- 원소 삽입 방법
  ① 원소를 삽입할 빈 자리 만들기
  ☞ 삽입할 자리 이후의 원소들을 한자리씩 뒤로 자리 이동 시키기

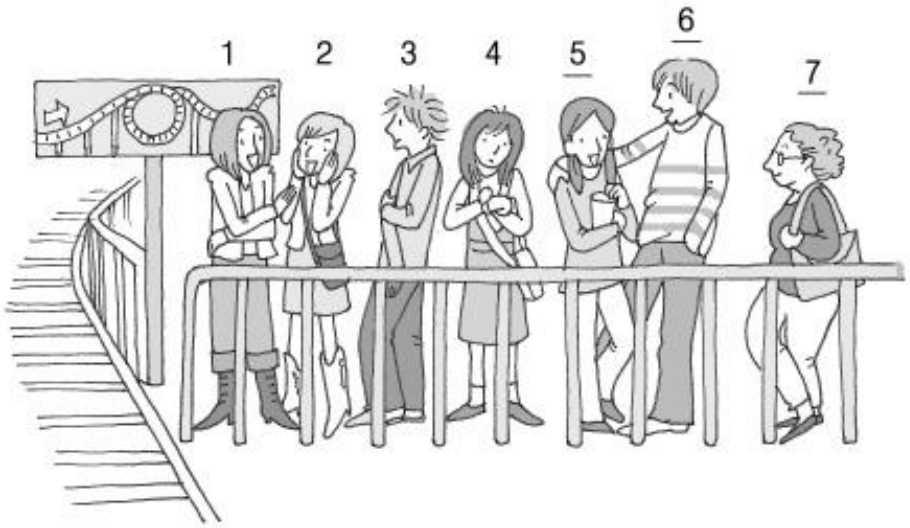
  ② 준비한 빈 자리에 원소 삽입하기



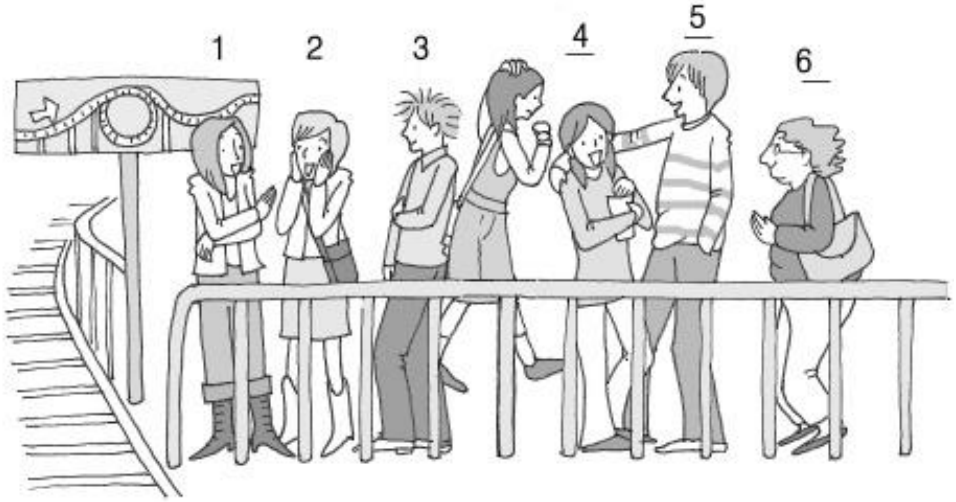
- 삽입할 자리를 만들기 위한 자리이동 횟수
  (n+1)개의 원소로 이루어진 선형 리스트에서 k번 자리에 원소를 삽입하는 경우 : k번 원소부터 마지막 n번 원소까지 (n-k+1)개의 원소를 이동
  이동횟수 = n-k+1 = 마지막 원소의 인덱스 - 삽입할 자리의 인덱스 +1

선형 리스트에서의 원소 삭제

- 선형리스트 중간에서 원소가 삭제되면, 그 이후의 원소들은 한자리씩 자리를 앞으로 이동하여 물리적 순서를 논리적 순서와 일치시킴



(a) 나가기 전

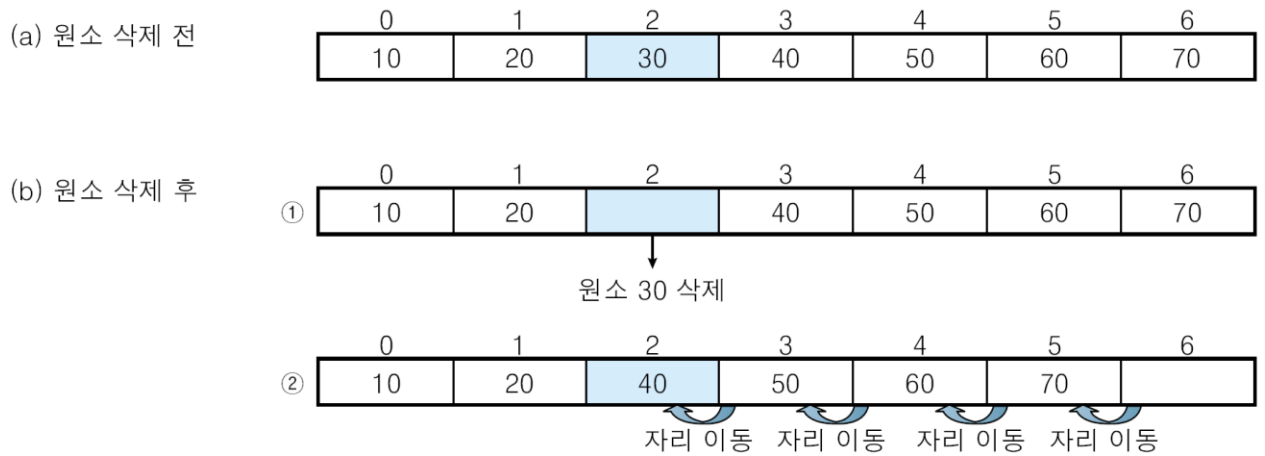


(b) 나간 후

- 원소 삭제 방법
  ① 원소 삭제하기

  ① 삭제한 빈 자리 채우기

  ☞ 삭제한 자리 이후의 원소들을 한자리씩 앞으로 자리 이동시키기



- 삭제 후, 빈 자리를 채우기 위한 자리이동 횟수
  (n+1)개 원소로 이루어진 선형 리스트에서 k번 자리의 원소를 삭제한 경우
  (k+1)번 원소부터 마지막 n번 원소까지 (n-(k+1)+1)개의 원소를 이동
  이동횟수 = n-(k+1)+1 = n-k
  = 마지막 원소의 인덱스-삭제한 자리의 인덱스

선형 리스트의 구현

- 순차 구조의 배열을 사용
  - ✓ 배열 - <인덱스, 원소>의 순서쌍의 집합
  - ✓ 배열의 인덱스 – 배열 원소의 순서

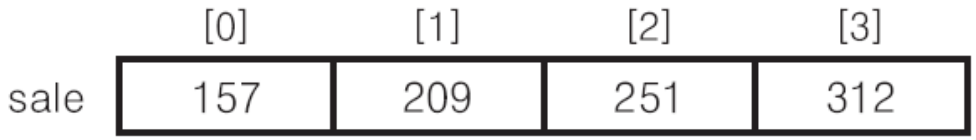
- **1차원 배열을 이용한 선형 리스트의 구현**
  - 예: 분기별 노트북 판매량

분기	1/4분기	2/4분기	3/4분기	4/4분기
판매량	157	209	251	312

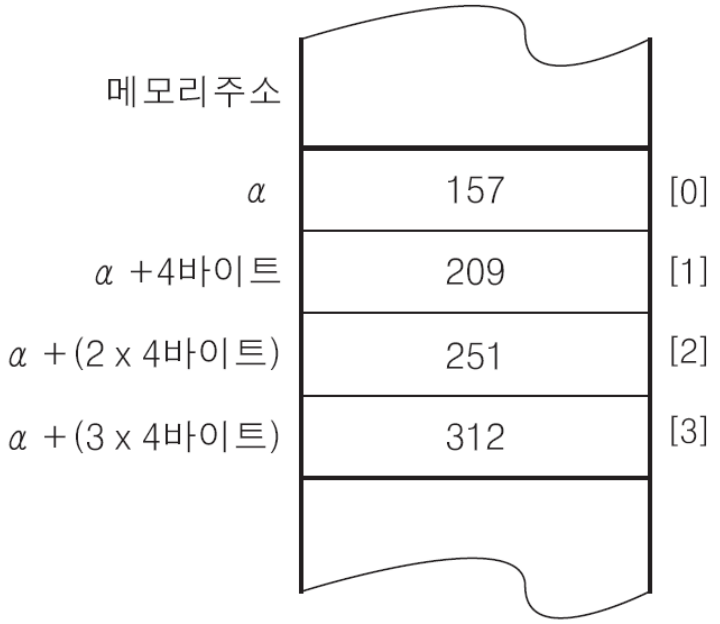
  - 1차원 배열을 이용한 구현

```
int sale[] = new int[] {157, 209, 251, 312}
```

➲ 논리적 구조



➲ 물리적 구조



- 분기별 판매량 리스트 프로그램

```
01      class Sale_1{
02              public static void main(String srgs[]){
03                      int sale[] = new int[]{157, 209, 251, 312};
04
05              for(int i=0; i<4; i++)
06                      System.out.printf("%d/4분기 : sale[%d]= %d %n", i+1, i, sale[i]);
07              }
08      }
```

❖ 2차원 배열을 이용한 선형 리스트의 구현

- 예 : 2007~2008년 분기별 노트북 판매량

년 \ 분기	1/4분기	2/4분기	3/4분기	4/4분기
2007년	63	84	140	130
2008년	157	209	251	312

- 2차원 배열을 이용한 구현

```
int sale[][] = new int[][]{{63, 84, 140, 130},
                           {157, 209, 251, 312}};
```

❶ 논리적 구조

	[0]	[1]	[2]	[3]
sale [0]	63	84	140	130
[1]	157	209	251	312

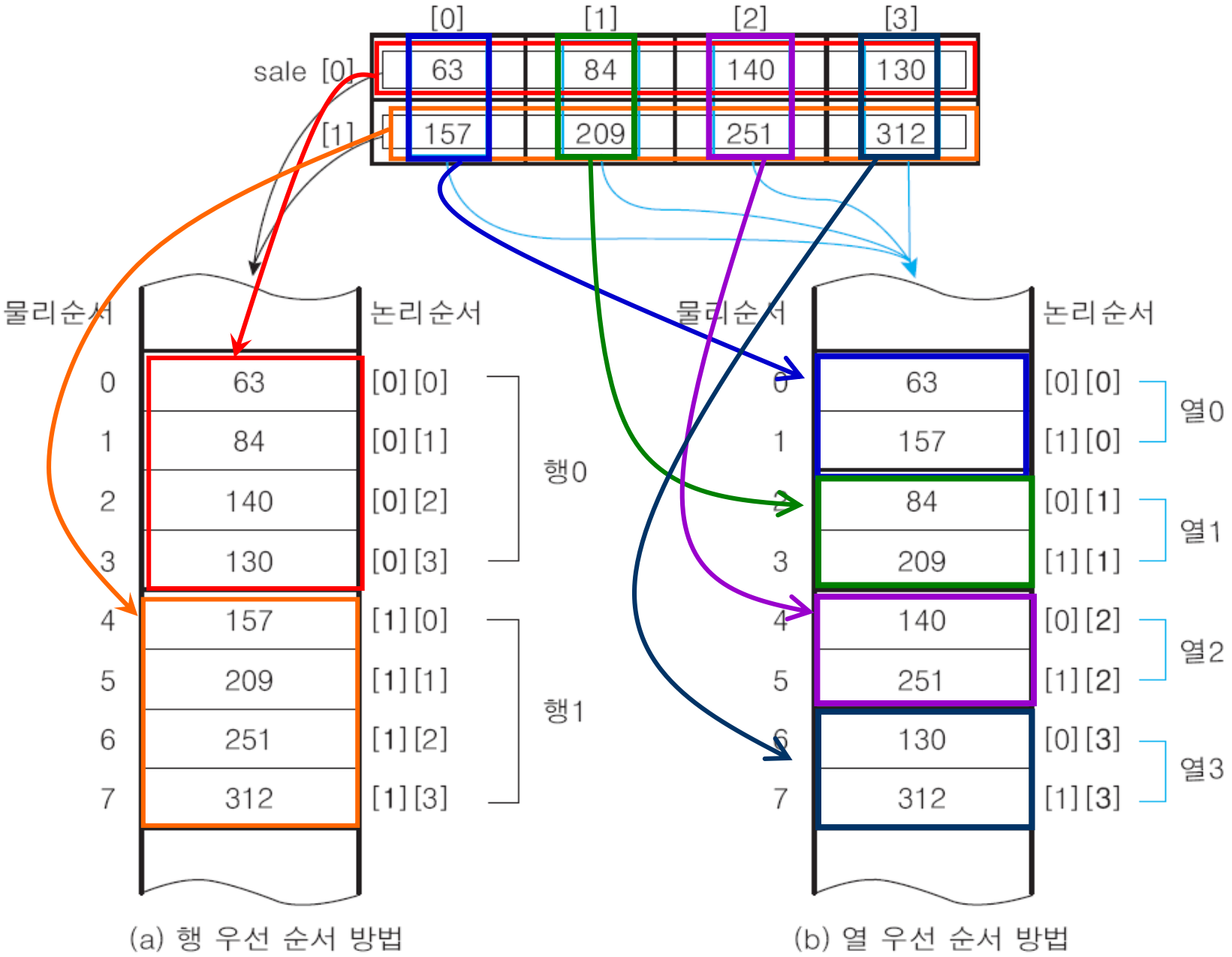
❖2차원 배열의 물리적 저장 방법

- 2차원의 논리적 순서를 1차원의 물리적 순서로 변환하는 방법 사용

- 행 우선 순서 방법(row major order)
  - 2차원 배열의 첫 번째 인덱스인 행 번호를 기준으로 사용하는 방법
    sale[**0**][0]=63, sale[**0**][1]=84, sale[**0**][2]=140, sale[**0**][3]=130, sale[**1**][0]=157,
    sale[**1**][1]=209, sale[**1**][2]=251, sale[**1**][3]=312
  - 원소의 위치 계산 방법 : $\alpha + (i \times n_j + j) \times \ell$
    행의 개수가 n_i이고 열의 개수가 n_j인 2차원 배열 $A[n_i][n_j]$의 시작주소가 α이고 원소의 길이가 ℓ 일 때, i행 j열 원소 즉, A[i][j]의 위치

- 열 우선 순서 방법(column major order)
  - 2차원 배열의 마지막 인덱스인 열 번호를 기준으로 사용하는 방법
    sale[0][**0**]=63, sale[1][**0**]=157, sale[0][**1**]=84, sale[1][**1**]=209, sale[0][**2**]=140,
    sale[1][**2**]=251, sale[0][**3**]=130, sale[1][**3**]=312
  - 원소의 위치 계산 방법 : $\alpha + (j \times n_i + i) \times \ell$

➲ 물리적 구조


[0]
[1]
[2]
[3]
sale [0]
63
84
140
130
[1]
157
209
251
312
물리순서
논리순서
0
63
[0][0]
1
84
[0][1]
2
140
[0][2]
3
130
[0][3]
행0
4
157
[1][0]
5
209
[1][1]
6
251
[1][2]
7
312
[1][3]
행1
(a) 행 우선 순서 방법
물리순서
논리순서
0
63
[0][0]
1
157
[1][0]
열0
2
84
[0][1]
3
209
[1][1]
열1
4
140
[0][2]
5
251
[1][2]
열2
6
130
[0][3]
7
312
[1][3]
열3
(b) 열 우선 순서 방법

- 2007, 2008년 분기별 판매량 선형 리스트 프로그램

```
01      class Sale_2{
02              public static void main(String srgs[]){
03                  int sale[][] = new int[][]{{63, 84, 140, 130},
04                                                  {157, 209, 251, 312}};
05
06                  for(int i=0; i<2; i++){
07                          for(int j=0; j<4; j++)
08                              System.out.printf("%d/4분기 : sale[%d][%d]= %d %n", j+1, i, j, sale[i][j]);
10                          System.out.println();
11                          }
12              }
13      }
```

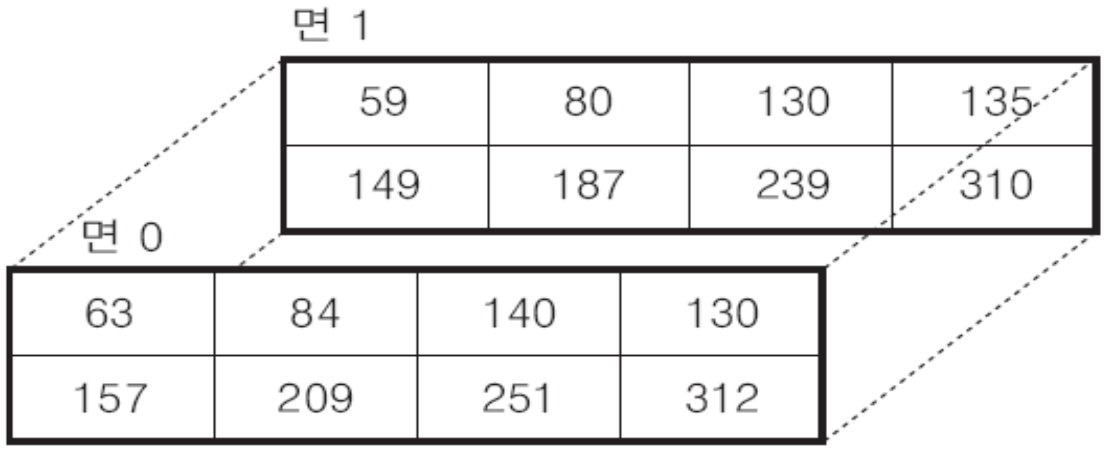
❖ 3차원 배열을 이용한 선형 리스트의 구현

- 예 : 2007~2008년, 1팀과 2팀의 분기별 노트북 판매량

1팀				
년 \ 분기	1/4분기	2/4분기	3/4분기	4/4분기
2007년	63	84	140	130
2008년	157	209	251	312

2팀				
년 \ 분기	1/4분기	2/4분기	3/4분기	4/4분기
2007년	59	80	130	135
2008년	149	187	239	310

- 3차원 배열을 이용한 구현
  - int sale[][][] = new int [][][]{{{63, 84, 140, 130},
    {157, 209, 251, 312}},
    {{59, 80, 130, 135},
    {149, 187, 239, 310}}
    };



논리적 구조

❖3차원 배열의 물리적 저장 방법

- 3차원의 논리적 순서를 1차원의 물리적 순서로 변환하는 방법 사용

- 면 우선 순서 방법
  - ➢ 3차원 배열의 첫 번째 인덱스인 면 번호를 기준으로 사용하는 방법
  - ➢ 원소의 위치 계산 방법 : $\alpha + \{(i \times n_j \times n_k) + (j \times n_k) + k\} \times \ell$
    - ✓ 면의 개수가 n_i이고 행의 개수가 n_j이고, 열의 개수가 n_k 인 3차원 배열 $A[n_i][n_j][n_k]$
    - ✓ 시작주소가 α이고 원소의 길이가 ℓ 일 때, i면 j행 k열 원소 즉, A[i][j][k]의 위치

- 열 우선 순서 방법
  - ➢ 3차원 배열의 마지막 인덱스인 열 번호를 기준으로 사용하는 방법
  - ➢ 원소의 위치 계산 방법 : $\alpha + \{(k \times n_j \times n_i) + (j \times n_i) + i\} \times \ell$

➲ 물리적 구조

물리 순서		논리 순서	
0	63	[0][0][0]	면0
1	84	[0][0][1]	
2	140	[0][0][2]	
3	130	[0][0][3]	
4	157	[0][1][0]	
5	209	[0][1][1]	
6	251	[0][1][2]	
7	312	[0][1][3]	
8	59	[1][0][0]	면1
9	80	[1][0][1]	
10	130	[1][0][2]	
11	135	[1][0][3]	
12	149	[1][1][0]	
13	187	[1][1][1]	
14	239	[1][1][2]	
15	310	[1][1][3]	

(a) 면 우선 순서 방법

물리 순서		논리 순서	
0	63	[0][0][0]	열0
1	59	[1][0][0]	
2	157	[0][1][0]	
3	149	[1][1][0]	
4	84	[0][0][1]	열1
5	80	[1][0][1]	
6	209	[0][1][1]	
7	187	[1][1][1]	
8	140	[0][0][2]	열2
9	130	[1][0][2]	
10	251	[0][1][2]	
11	239	[1][1][2]	
12	130	[0][0][3]	열3
13	135	[1][0][3]	
14	312	[0][1][3]	
15	310	[1][1][3]	

(b) 열 우선 순서 방법

```
01  class Sale_3{
02      public static void main(String srgs[]){
03          int sale[][][] = new int [][][]{{{63, 84, 140, 130},
04                                           {157, 209, 251, 312}},
05                                          {{59, 80, 130, 135},
06                                           {149, 187, 239, 310}}
07                                          };
08
09          for(int i=0; i<2; i++){
10              System.out.printf("<< %d 팀 >> %n", i+1);
11              for(int j=0; j<2; j++){
12                  for(int k=0; k<4; k++)
13                      System.out.printf("%d/4분기 : sale[%d][%d][%d]
14                                         = %d %n", k+1, i, j, k, sale[i][j][k]);
15                  System.out.println("-------------------------");
16              }
17              System.out.println();
18          }
19      }
20  }
```

Overflow

- 배열은 미리 정해진 크기의 메모리 공간을 할당 받은 뒤 사용해야 하므로, 빈자리가 없어 새 항목을 삽입할 수 없는 상황(Overflow) 발생

- Overflow 가 발생하면 에러 처리를 하여 프로그램을 정지시키는 방법이 주로 사용된다. 하지만 프로그램의 안정성을 향상시키기 위해 다음과 같은 방법을 사용

[핵심 아이디어]
- 배열에 overflow가 발생하면 배열 크기를 2배로 확장한다. 또한 배열의 3/4이 비어 있다면 배열 크기를 1/2로 축소한다.

Overflow

- 프로그램이 실행되는 동안에 할당된 배열

ArrList 클래스

```
01 import java.util.NoSuchElementException;
02 public class ArrList <E> {
03     private  E a[];      // 리스트의 항목들을 저장할 배열
04     private  int size;   // 리스트의 항목 수
05     public  ArrList() { // 생성자
06         a = (E[]) new Object[1];   // 최초로 1개의 원소를 가진 배열 생성
07         size = 0;                  // 항목 수를 0으로 초기화
08     }
       // 탐색, 삽입, 삭제 연산을 위한 메소드 선언
   }
```

insertLast(), insert() method

```
01  public void insertLast(E newItem) { // 가장 뒤에 새 항목 삽입
02       if (size == a.length)          // 배열에 빈 공간이 없으면
03            resize(2*a.length);       // 배열 크기 2배로 확장
04        a[size++] = newItem;          // 새 항목 삽입
05  }
06 public void insert(E newItem, int k) { // 새 항목을 k-1번째 항목 다음에 삽입
07     if (size == a.length)             // 배열에 빈 공간이 없으면
08         resize(2*a.length);           // 배열 크기 2배로 확장
09     for (int i = size-1; i >= k; i--)  a[i+1] = a[i];  // 한 칸씩 뒤로 이동
10     a[k] = newItem;
11     size++;
12 }
```

resize(), delete() method

```
01 private void resize(int newSize) {        // 배열 크기 조절
02     Object[] t = new Object[newSize];     // newSize 크기의 새로운 배열 t 생성
03     for (int i = 0; i < size; i++)
04         t[i] = a[i];                      // 배열 s를 배열 t로 복사
05     a = (E[]) t;                          // 배열 t를 배열 s로
06 }
```

```
01 public E delete(int k) {  // k번째 항목 삭제
02     if (isEmpty()) throw new NoSuchElementException(); // underflow 경우에 프로그램 정지
03     E item = a[k];
04     for (int i = k; i <size; i++)  a[i] = a[i+1];  // 한 칸씩 앞으로 이동
05     size--;
06     if (size > 0 && size == a.length/4) // 배열에 항목들이 1/4만 차지한다면
07         resize(a.length/2);             // 배열을 1/2 크기로 축소
08     return item;
09 }
```

```
import java.util.NoSuchElementException;
public class ArrList <E> {
            private  E a[];     // 리스트의 항목들을 저장할 배열
    private  int size;  // 리스트의 항목 수

    public  ArrList() { // 생성자
        a = (E[]) new Object[1];  // 최초로 1개의 원소를 가진 배열 생성
        size = 0;                 // 항목 수를 0으로 초기화
    }
    public boolean isEmpty() {return size == 0;} // 리스트가 empty이면 true 리턴

    public void insertLast(E newItem) {          // 가장 뒤에 새 항목 삽입
        if (size == a.length)           // 배열에 빈 공간이 없으면
            resize(2*a.length);         // 배열 크기 2배로 확장
         a[size++] = newItem;           // 새 항목 삽입
    }

    public void insert(E newItem, int k) { // 새 항목을 k-1번째 항목 다음에 삽입
            if (size == a.length)                    // 배열에 빈 공간이 없으면
                    resize(2*a.length);                                  // 배열 크기 2배로 확장
            for (int i = size-1; i >= k; i--)  a[i+1] = a[i];  // 한 칸씩 뒤로 이동
            a[k] = newItem;
            size++;
    }
```

```
    public E delete(int k) {  // k번째 항목 삭제
                        if (isEmpty()) throw new NoSuchElementException(); // underflow 경우에 프로그램 정지
                        E item = a[k];
                        for (int i = k; i <size; i++)  a[i] = a[i+1];  // 한 칸씩 앞으로 이동
                        size--;
                        if (size > 0 && size == a.length/4) // 배열에 항목들이 1/4만 차지한다면
                                    resize(a.length/2);                                    // 배열을 1/2 크기로 축소
                        return item;
    }
    public E peek(int k) {  // k번째 항목을 리턴, 단순히 읽기만 한다.
                        if (isEmpty()) throw new NoSuchElementException(); // underflow 경우에 프로그램 정지
                        return a[k];
      }
    private void resize(int newSize) {                          // 배열 크기 조절
                        Object[] t = new Object[newSize];   // newSize 크기의 새로운 배열 t 생성
                        for (int i = 0; i < size; i++)
                                    t[i] = a[i];                    // 배열 s를 배열 t로 복사
                        a = (E[]) t;                        // 배열 t를 배열 s로
              }
    public void print() { // 배열의 항목들을 출력
                        if (isEmpty())
                                    System.out.print("배열이 비어있음.");
                        else
                                    for(int i = 0; i < a.length; i++)      System.out.print(a[i]+"₩t ");
                        System.out.println();
              }
}
```

```
public class main {
	public static void main(String[] args) {
		ArrList<String> s = new ArrList<String>();
		s.insertLast("apple");   s.print();   s.insertLast("orange"); s.print();
		s.insertLast("cherry");  s.print();   s.insertLast("pear");   s.print();
		s.insert("grape",1);     s.print();   s.insert("lemon",4);    s.print();
		s.insertLast("kiwi");    s.print();
		s.delete(4); s.print();  s.delete(0); s.print();
		s.delete(0); s.print();  s.delete(3); s.print();
		s.delete(0); s.print();

		System.out.println("1번째 항목은 "+s.peek(1)+"이다."); System.out.println();

		ArrList<Integer>  t = new ArrList<Integer>();
		t.insertLast(100); t.insertLast(200); t.insertLast(300); t.insertLast(400); t.print();
		t.insert(350,3);          t.print();
		t.insert(250,2);          t.print();
		t.insertLast(500);        t.print();
	}
}
```

Problems Javadoc Console Console

<terminated> main (49) [Java Application] C:₩Program Files₩Java₩jdk1.8.0_40₩bin₩javaw.exe

```
apple
apple   orange                                          ← 2배로 확장
apple   orange  cherry  null                            ← 2배로 확장
apple   orange  cherry  pear
apple   grape   orange  cherry  pear    null    null    null    ← 2배로 확장
apple   grape   orange  cherry  lemon   pear    null    null
apple   grape   orange  cherry  lemon   pear    kiwi    null
apple   grape   orange  cherry  pear    kiwi    null    null
grape   orange  cherry  pear    kiwi    null    null    null    null
orange  cherry  pear    kiwi    null    null    null    null    null
orange  cherry  pear    null    null    null    null    null
cherry  pear    null    null                                  ← 1/2로 축소
1번째 항목은 pear이다.
```

삭제

Report

1. 1, 2, 3 차원 배열에 대한 예제 파일을 만들고 테스트한 결과 파일을 작성하시오.
2. ArrList와 테스트한 파일을 작성하시오.

1과 2에서 만든 파일을 Github에 업로드하면 됩니다.

Reference

- 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

DIT 동의과학대학교
DONG-EUI INSTITUTE OF TECHNOLOGY

DIT 동의과학대학교
DONG-EUI INSTITUTE OF TECHNOLOGY