e-learning

2020 학년도 1 학기

컴퓨터 정보과

자료구조(Data Structures)

담당교수 : 김주현

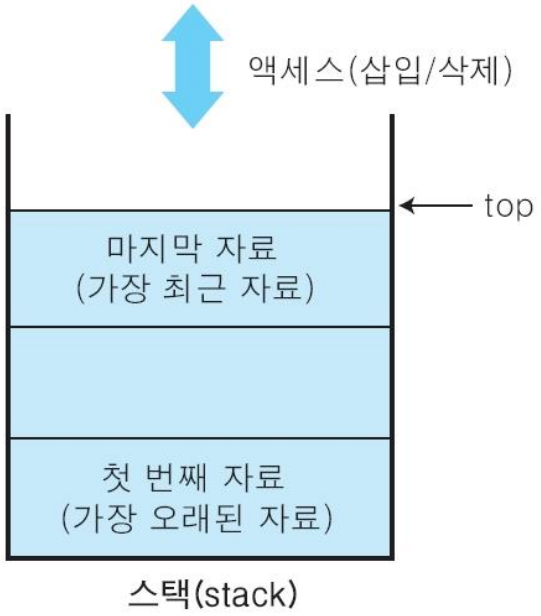
제 4 주차 / 제 1 차시

DIT 동의과학대학교
DONG-EUI INSTITUTE OF TECHNOLOGY

스택(Stack)

스택(Stack)

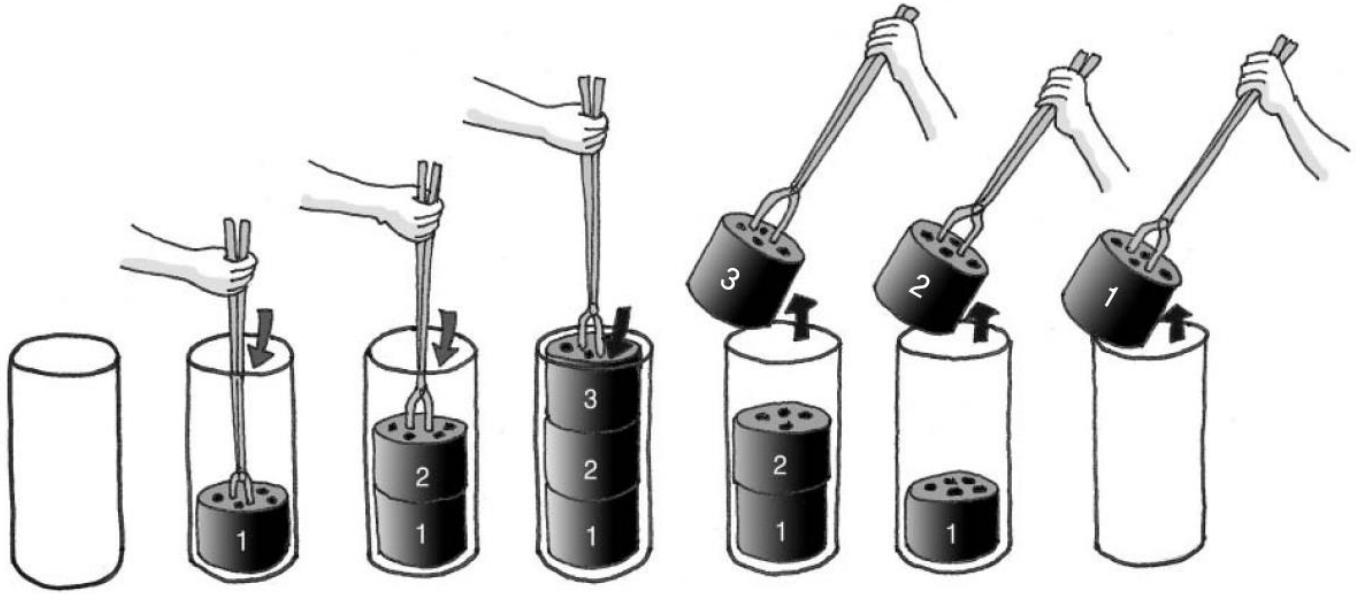
- 한 쪽 끝에서만 item(항목)을 삭제하거나 새로운 item을 저장하는 자료구조
- 마지막에 삽입(Last-In)한 원소는 맨 위에 쌓여 있다가 가장 먼저 삭제(First-Out) 된다. ☞ 후입선출 구조 (LIFO, Last-In-First-Out)



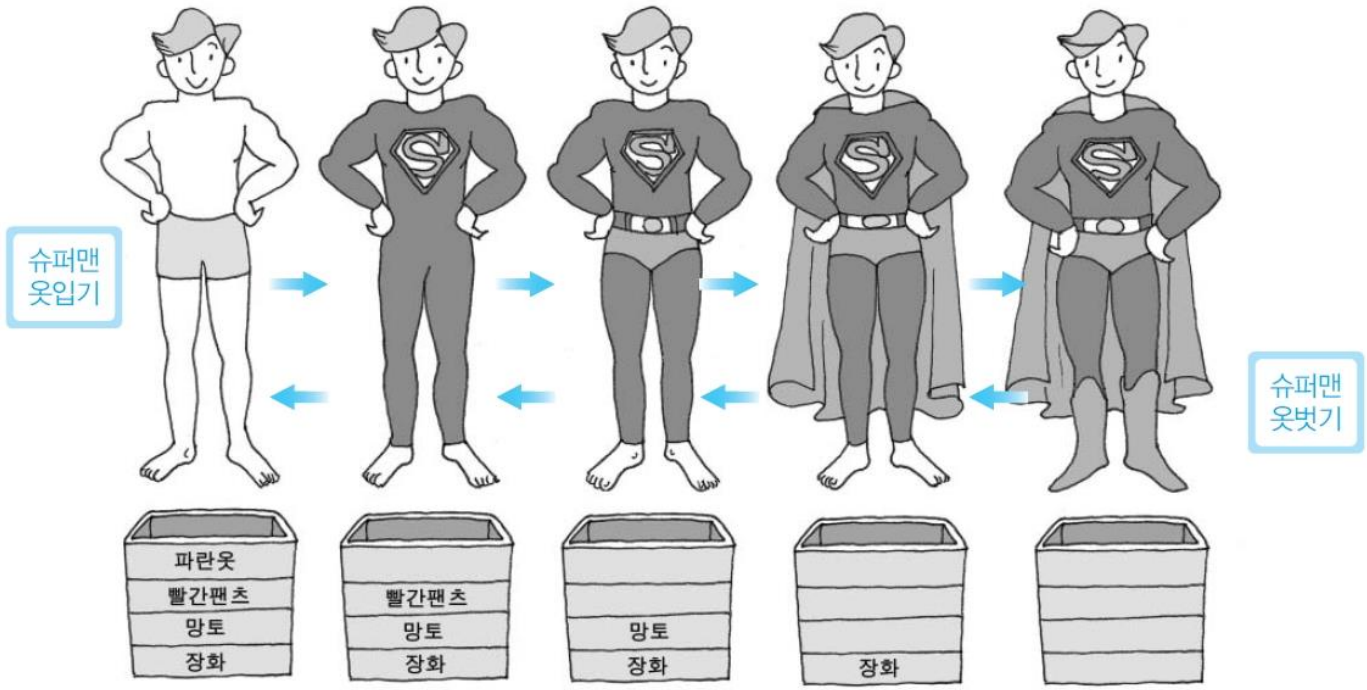
스택(stack)

- 후입선출 구조의 예1 : 연탄 아궁이

  - 연탄을 하나씩 쌓으면서 아궁이에 넣으므로 마지막에 넣은 3번 연탄이 가장 위에 쌓여 있다.
  - 연탄을 아궁이에서 꺼낼 때에는 위에서부터 하나씩 꺼내야 하므로 마지막에 넣은 3번 연탄을 가장 먼저 꺼내게 된다.

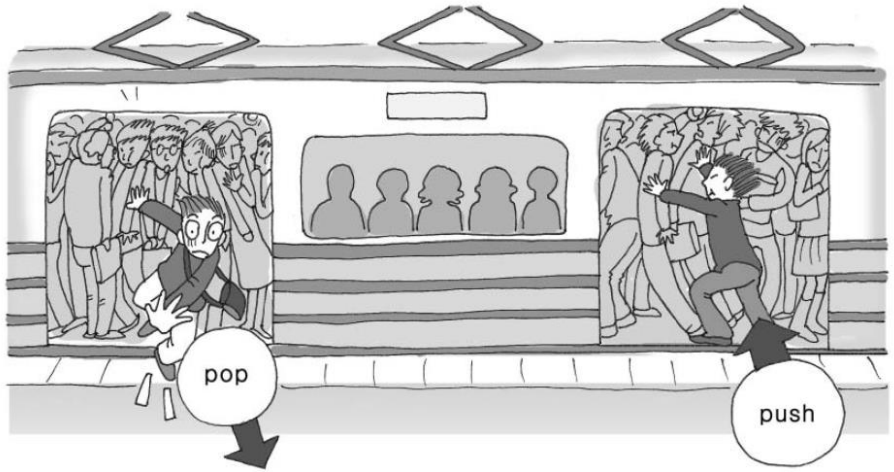


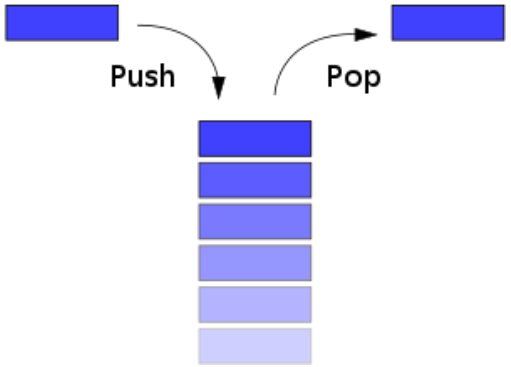
- 후입선출 구조의 예2 : 슈퍼맨의 옷 갈아입기
  - 수퍼맨이 옷을 벗는 순서
    ① 장화 → ②망토 → ③빨간팬츠 → ④파란옷
  - 슈퍼맨이 옷을 입는 순서
    ④ 파란옷 → ③빨간팬츠 → ②망토 → ①장화

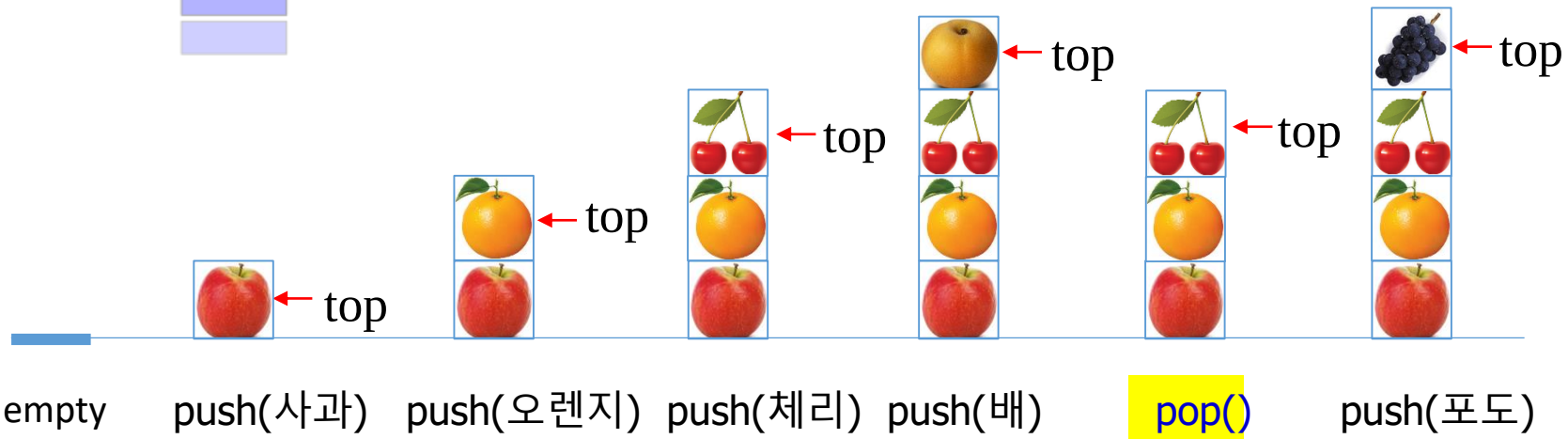


❖ 스택의 연산

- 스택에서의 삽입 연산 : push
- 스택에서의 삭제 연산 : pop







❖ 스택의 push 알고리즘

① top ← top+1;

- 스택 S에서 top이 마지막 자료를 가리키고 있으므로 그 위에 자료를 삽입하려면 먼저 top의 위치를 하나 증가
- 만약 이때 top의 위치가 스택의 크기(stack_SIZE)보다 크다면 오버플로우(overflow)상태가 되므로 삽입 연산을 수행하지 못하고 연산 종료

② S(top) ← x;

- 오버플로우 상태가 아니라면 스택의 top이 가리키는 위치에 x 삽입

❖스택의 pop 알고리즘

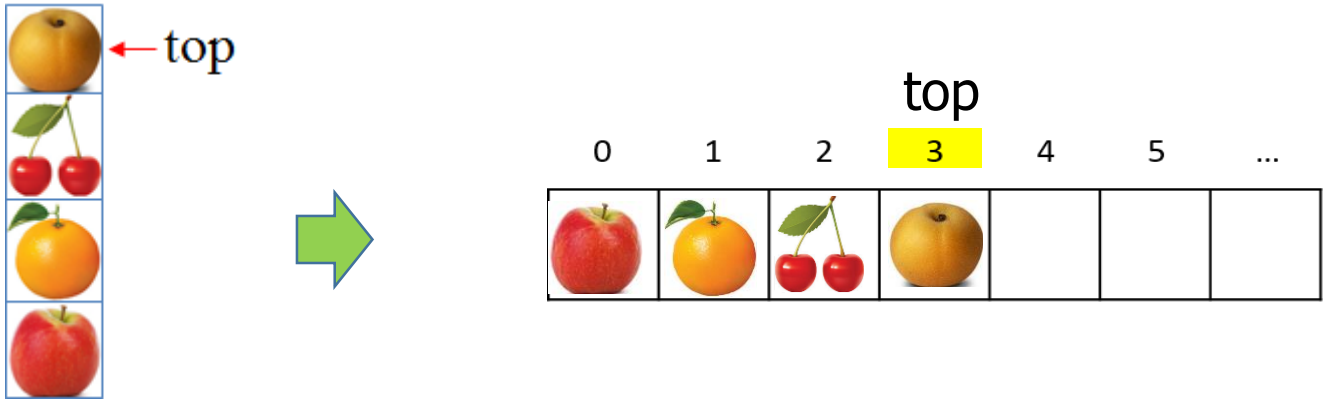
① return S(top);

- 스택이 공백 스택이 아니라면 top이 가리키는 원소를 먼저 반환

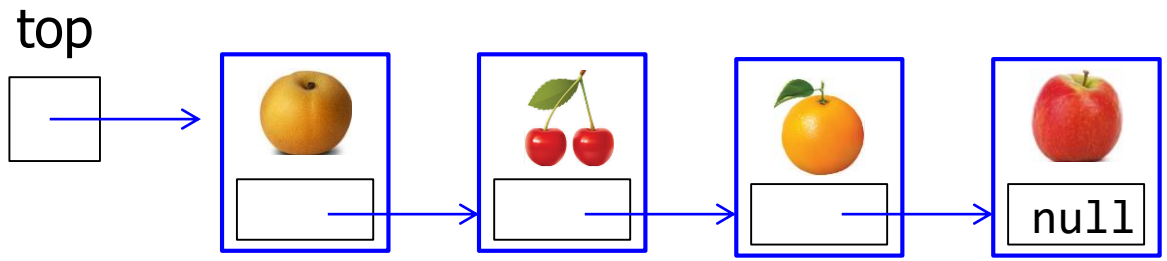
② top ← top-1;

- 스택의 top 원소를 반환하였으므로 top의 위치는 그 아래의 원소로 변경하기 위해 top의 위치를 하나 감소

배열로 구현된 스택



단순연결리스트로 구현된 스택



❖순차 자료구조를 이용한 스택의 구현

- 순차 자료구조인 1차원 배열을 이용하여 구현
  - 스택의 크기 : 배열의 크기
  - 스택에 저장된 원소의 순서 : 배열 원소의 인덱스
    - 인덱스 0번 : 스택의 첫번째 원소
    - 인덱스 n-1번 : 스택의 n번째 원소

- 변수 top : 스택에 저장된 마지막 원소에 대한 인덱스 저장
  - 공백 상태 : top = -1 (초기값)
  - 포화 상태 : top = n-1

순차 자료구조 방식을 이용하여 구현한 스택 프로그램

```
01      interface Stack{
02              boolean isEmpty( );
03              void push(char item);
04              char pop( );
05              void delete( );
06              char peek( );
07      }
08
09      class ArrayStack implements Stack{
10                      private int top;
11                      private int stackSize;
12                      private char itemArray[];
13
14                      public ArrayStack(int stackSize){
15                              top = -1;
16                              this.stackSize = stackSize;
17                              itemArray = new char[this.stackSize];
18                      }
```

```
19
20                  public boolean isEmpty(){
21                          return (top == -1);
22                  }
23
24                  public boolean isFull(){
25                          return (top == this.stackSize-1);
26                  }
27
28                  public void push(char item){
29                          if(isFull()){
30                                  System.out.println("Inserting fail! Array Stack is full!!");
31                          }
32                          else{
33                                  itemArray[++top] = item;
34                                  System.out.println("Inserted Item : " + item);
35                          }
36                  }
```

```
37
38                      public char pop(){
39                              if(isEmpty()) {
40                                      System.out.println("Deleting fail! Array Stack is empty!!");
41                                      return 0;
42                              }
43                              else{
44                                      return itemArray[top--];
45                              }
46                      }
47
48                      public void delete(){
49                              if(isEmpty()){
50                                      System.out.println("Deleting fail! Array Stack is empty!!");
51                              }
52                              else {
53                                      top--;
54                              }
55                      }
```

```
56
57                      public char peek(){
58                              if(isEmpty()){
59                                      System.out.println("Peeking fail! Array Stack is empty!!");
60                                      return 0;
61                              }
62                              else
63                                      return itemArray[top];
64                      }
65
66                      public void printStack(){
67                              if(isEmpty())
68                                      System.out.printf("Array Stack is empty!! %n %n");
69                              else{
70                                      System.out.printf("Array Stack>> ");
71                                      for(int i=0; i<=top; i++)
72                                              System.out.printf("%c ", itemArray[i]);
73                                      System.out.println(); System.out.println();
74                              }
75                      }
76      }
77
```

```
78      class Array_Stack{
79                  public static void main(String args[]){
80                              int stackSize = 5;
81                              char deletedItem;
82                              ArrayStack S = new ArrayStack(stackSize);
83
84                              S.push('A');
85                              S.printStack( );
86
87                              S.push('B');
88                              S.printStack();
89
90                              S.push('C');
91                              S.printStack( );
92
93                              deletedItem = S.pop();
94                              if(deletedItem != 0)
95                                      System.out.println("deleted Item : " + deletedItem);
96                              S.printStack();
97                  }
98      }
```

❖ 순차 자료구조로 구현한 스택의 장점

- 순차 자료구조인 1차원 배열을 사용하여 쉽게 구현

❖ 순차 자료구조로 구현한 스택의 단점

- 물리적으로 크기가 고정된 배열을 사용하므로 스택의 크기 변경 어려움
- 순차 자료구조의 단점을 그대로 가지고 있다.

배열로 구현한 ArrayStack 클래스

```
01 import java.util.EmptyStackException;
02 public class ArrayStack<E> {
03     private E   s[];       // 스택을 위한 배열
04     private int top;       // 스택의 top 항목의 배열 원소 인덱스
05     public ArrayStack() { // 스택 생성자
06         s = (E[]) new Object[1];  // 초기에 크기가 1인 배열 생성
07         top = -1;
08     }
09     public int     size()    { return top+1;}        // 스택에 있는 항목의 수를  리턴
10     public boolean isEmpty() { return (top == -1);} // 스택이 empty이면 true 리턴

       // peek(), push(), pop(), resize() 메소드 선언

   }
```

```
01 public E peek() {  // 스택 top 항목의 내용만을 리턴
02     if (isEmpty()) throw new EmptyStackException(); // underflow 시 프로그램 정지
03     return s[top];
04 }
```

```
01  public void push(E newItem) { // push 연산
02      if (size() == s.length)
03          resize(2*s.length);    // 스택을 2배의 크기로 확장
04      s[++top] = newItem;        // 새 항목을 push
05  }
```

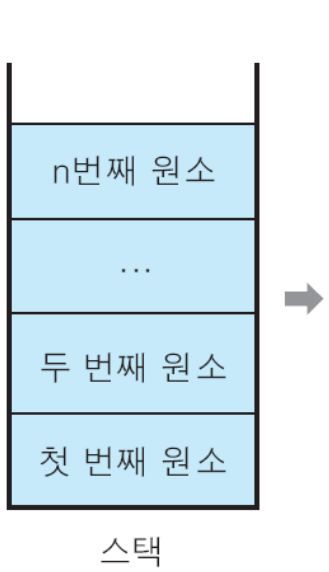
```
01 public E pop() {  // pop 연산
02     if (isEmpty()) throw new EmptyStackException(); // underflow시 프로그램 정지
03     E item = s[top];
04     s[top--] = null;          // null로 초기화
05     if (size() > 0 && size()==s.length/4)
06         resize(s.length/2);  // 스택을 1/2 크기로 축소
07     return item;
08 }
```

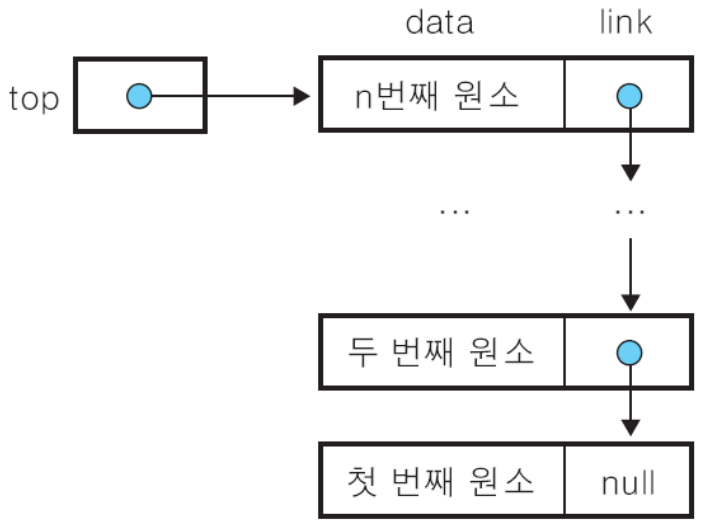
```java
public class main {
    public static void main(String[] args) {
        ArrayStack<String> stack = new ArrayStack<String>();

        stack.push("apple");
        stack.push("orange");
        stack.push("cherry");
        System.out.println(stack.peek());
        stack.push("pear");
        stack.print();
        stack.pop();
        System.out.println(stack.peek());
        stack.push("grape");
        stack.print();
    }
}
```

```
cherry
apple    orange   cherry   pear
cherry
apple    orange   cherry   grape
```

❖ 연결 자료구조를 이용한 스택의 구현

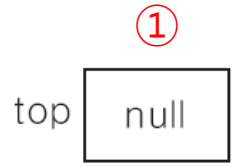
- **단순 연결 리스트를 이용하여 구현**
  - ➢ 스택의 원소 : 단순 연결 리스트의 노드
    - ✓ 스택 원소의 순서 : 노드의 링크 포인터로 연결
    - ✓ push : 리스트의 마지막에 노드 삽입
    - ✓ pop : 리스트의 마지막 노드 삭제
  - ➢ 변수 top : 단순 연결 리스트의 마지막 노드를 가리키는 포인터 변수
    - ✓ 초기 상태 : top = null



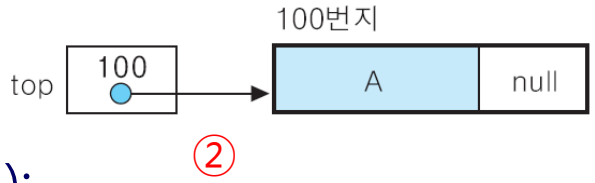


단순 연결 리스트의 스택에서의 연산 수행 과정

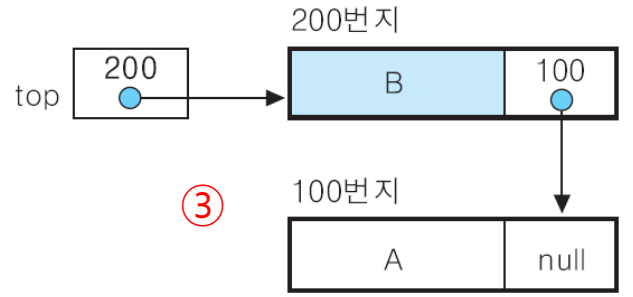
① 공백 스택 생성 : create(stack);



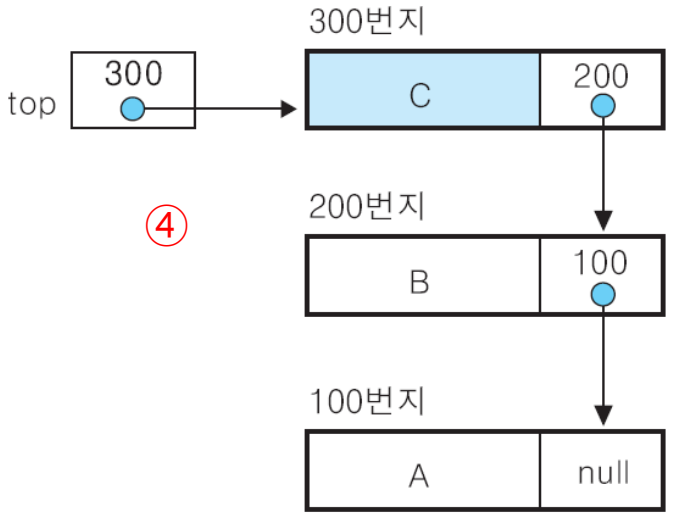
② 원소 A 삽입 : push(stack, A);



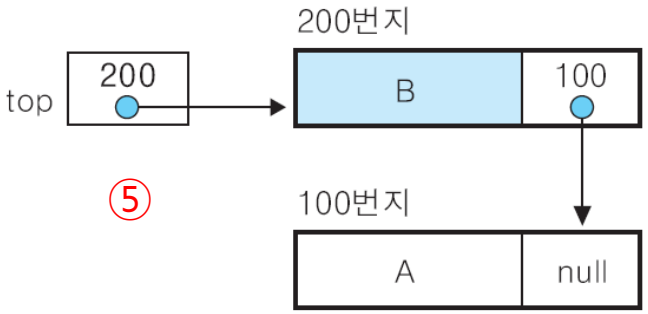
③ 원소 B 삽입 : push(stack, B);



④ 원소 C 삽입 : push(stack, C);



⑤ 원소 삭제 : pop(stack);



연결 자료구조 방식을 이용하여 구현한 스택 프로그램

```
01      interface Stack{
02              boolean isEmpty();
03              void push(char item);
04              char pop();
05              void delete();
06              char peek();
07      }
08
09      class StackNode{
10              char data;
11              StackNode link;
12      }
```

```
13
14        class LinkedStack implements Stack{
15                  private StackNode top;
16
17                  public boolean isEmpty(){
18                            return (top == null);
19                  }
20
21                  public void push(char item){
22                            StackNode newNode = new StackNode();
23                            newNode.data = item;
24                            newNode.link = top;
25                            top = newNode;
26                            System.out.println("Inserted Item : " + item);
27                  }
28
```

```
29              public char pop(){
30                      if(isEmpty()) {
31                              System.out.println("Deleting fail! Linked Stack is empty!!");
32                              return 0;
33                      }
34                      else{
35                              char item = top.data;
36                              top = top.link;
37                              return item;
38                      }
39              }
40
41              public void delete(){
42                      if(isEmpty()){
43                              System.out.println("Deleting fail! Linked Stack is empty!!");
44
45                      }
46                      else {
47                              top = top.link;
48                      }
49              }
```

```
50
51              public char peek(){
51                      if(isEmpty()){
52                              System.out.println("Peeking fail! Linked Stack is empty!!");
53                              return 0;
54                      }
55                      else
56                              return top.data;
57              }
58
```

```
59                  public void printStack(){
60                          if(isEmpty())
61                                  System.out.printf("Linked Stack is empty!! %n %n");
62                          else{
63                                  StackNode temp = top;
64                                  System.out.println("Linked Stack>> ");
65                                  while(temp != null){
66                                          System.out.printf("\t %c \n", temp.data);
67                                          temp = temp.link;
68                                  }
69                                  System.out.println();
70                          }
71                  }
72          }
73
```

```
74      class linked_Stack{
75              public static void main(String args[]){
76                      char deletedItem;
77                      LinkedStack LS = new LinkedStack();
78
79                      LS.push('A');
80                      LS.printStack();
81
82                      LS.push('B');
83                      LS.printStack();
84
85                      LS.push('C');
86                      LS.printStack();
87
88                      deletedItem = LS.pop();
89                      if(deletedItem != 0)
90                              System.out.println("deleted Item : " + deletedItem);
91                      LS.printStack();
92              }
93      }
```

```
public class Node <E> {
        private E       item;
        private Node<E> next;
        public Node(E newItem, Node<E> node){  // 생성자
                item = newItem;
                next = node;
        }
        // get 메소드들과 set 메소드들
        public E       getItem() { return item; }
        public Node<E> getNext() { return next; }
        public void    setItem(E newItem)      { item = newItem; }
        public void    setNext(Node<E> newNext){ next = newNext; }
}
```

```
import java.util.EmptyStackException;
public class ListStack <E> {
        private Node<E> top;    // 스택 top 항목을 가진  Node를 가리키기 위해
        private int size;       // 스택의 항목 수
        public ListStack() {    // 스택 생성자
                top = null;
                size = 0;
        }
        public int     size()    { return size;}      // 스택의 항목의 수를 리턴
        public boolean isEmpty() { return size == 0;} // 스택이 empty이면 true 리턴
        public void push(E newItem){  // 스택 push 연산
                Node newNode = new Node(newItem, top);  // 리스트 앞부분에 삽입
                top = newNode;                          // top이 새 노드 가리킴
                size++;                                 // 스택 항목 수 1 증가
        }
```

```
        public E peek() {  // 스택 top 항목만을 리턴
                if (isEmpty()) throw new EmptyStackException(); // underflow 시 프로그램 정지
                return top.getItem();
        }
        public E pop() {   // 스택 pop연산
                if (isEmpty()) throw new EmptyStackException(); // underflow 시 프로그램 정지
                E topItem = top.getItem();   // 스택 top 항목을 topItem에 저장
                top = top.getNext();         // top이 top 바로 아래 항목을 가리킴
                size--;                      // 스택 항목 수를 1 감소
                return topItem;
        }
        public void print() { // 스택의 항목들을 top부터 차례로 출력
                if (isEmpty()) System.out.print("스택이 비어있음.");
                else
                        for (Node p = top; p != null; p = p.getNext())
                                System.out.print(p.getItem()+"\t ");
                System.out.println();
        }
```

```
        public static void main(String[] args) {
                ListStack<String> stack = new ListStack<String>();
                stack.push("apple"); stack.push("orange"); stack.push("cherry");
                System.out.println(stack.peek());
                stack.push("pear");     stack.print();
                stack.pop();                    System.out.println(stack.peek());
                stack.push("grape");    stack.print();
        }
}
```

Report

앞에서 언급한 소스들을 테스트하고 깃허브에 업로드 하세요!!!

Reference

- 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