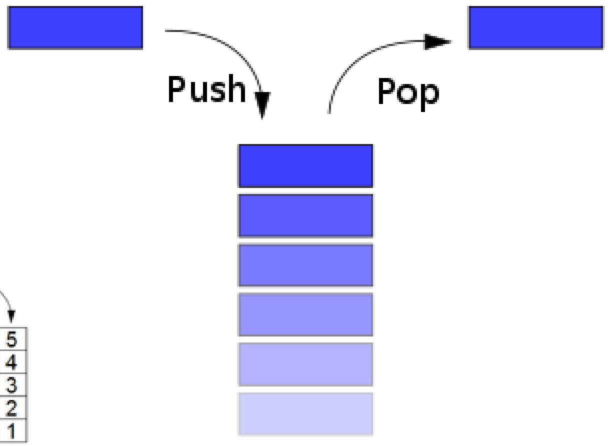
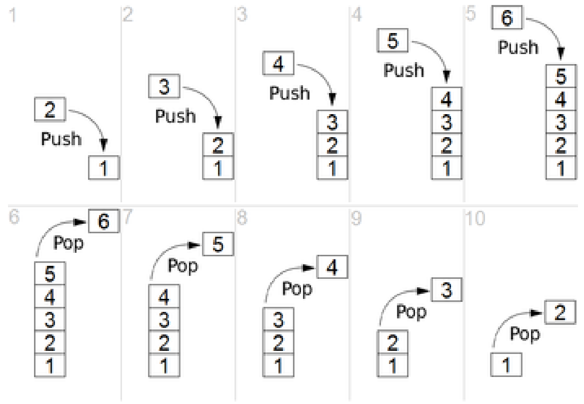
자료구조
Stack

스택(Stack)

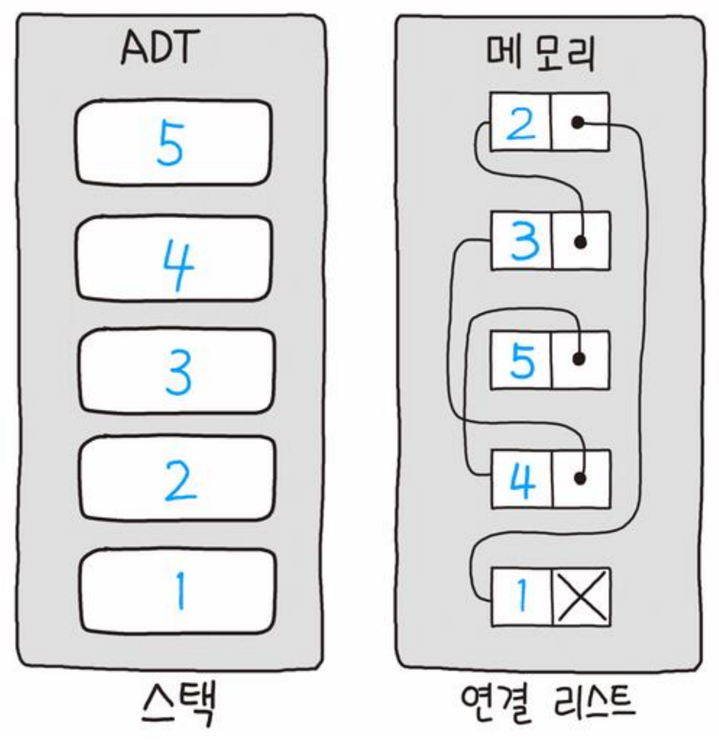


- 바닥이 막힌 상자
  - Last In First Out (LIFO), 후입선출
- 연산
  - Push, Pop, Peek
- index
  - Top
- 스택 관련
  - Alan Turing: 서브루틴 호출(bury), 되돌아오는 과정(unbury)
  - 컴퓨터 프로그램의 서브루틴 호출시 기억 장소: Stack
  - Stack이 가득 차면? error
    - Stackoverflow



스택의 구현

- 구현방법
  - 배열
    - 배열크기를 정하고, index로 top을 표현
  - 리스트
    - 메모리 할당 후 포인터 연결

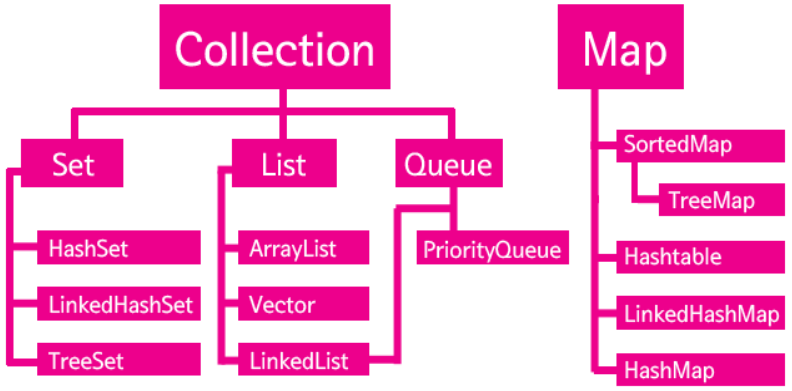


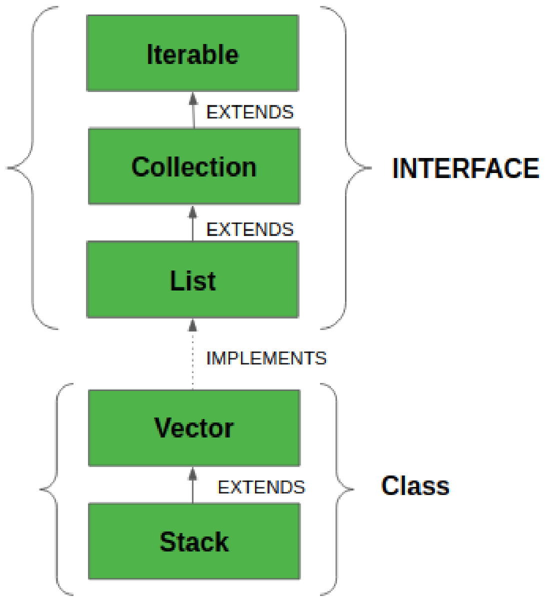
스택의 응용

- 함수 호출
  - 호출시 정보 저장 후 되돌아가기
- Undo 명령어
  - Ctrl+Z

자바, 자료구조의 스택

- 스택 구현 방법
  - Package 활용: java.util.Stack
  - 직접 구현





java.util.Stack

```java
import java.util.Stack;

public class testStack {
    public static void main(String[] args) {
        Stack<String> s = new Stack<String>();

        s.push("A");
        s.push("B");
        s.push("C");

        System.out.println("Pop an element from Stack !!!");
        while(!s.empty()) {
            System.out.println(s.pop());
        }
    }
}

```

Java.util.Stack

- https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Stack.html
- https://www.tutorialspoint.com/java/util/java_util_stack.htm

java.util

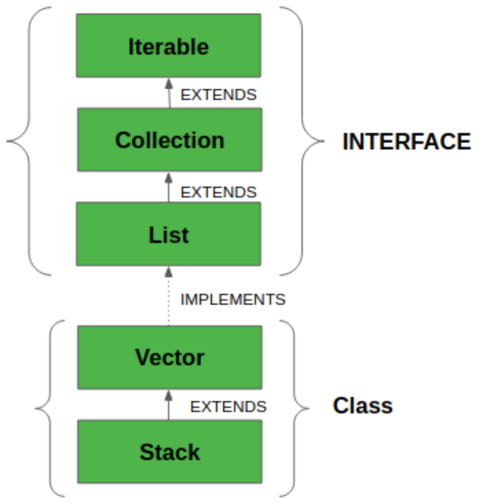
Class Stack<E>

java.lang.Object
    java.util.AbstractCollection<E>
        java.util.AbstractList<E>
            java.util.Vector<E>
                java.util.Stack<E>

All Implemented Interfaces:

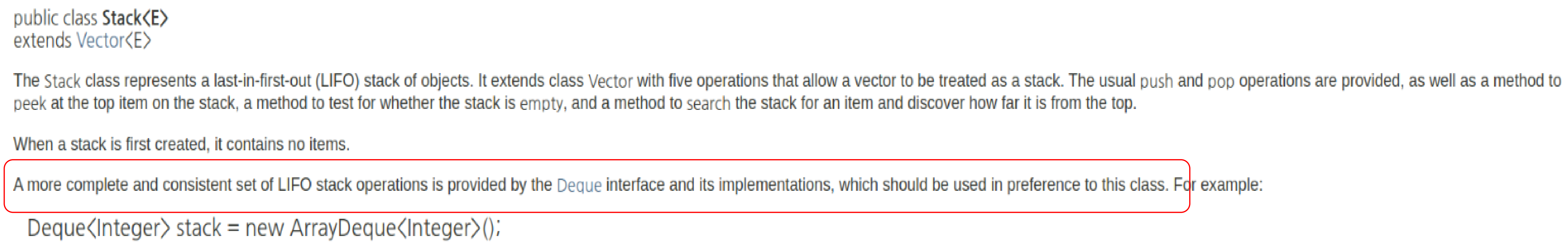
Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess

Modifier and Type	Method	Description
`boolean`	`empty()`	Tests if this stack is empty.
`E`	`peek()`	Looks at the object at the top of this stack without removing it from the stack.
`E`	`pop()`	Removes the object at the top of this stack and returns that object as the value of this function.
`E`	`push(E item)`	Pushes an item onto the top of this stack.
`int`	`search(Object o)`	Returns the 1-based position where an object is on this stack.



Java Stack with Deque Interface

- https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Stack.html



public class **Stack<E>**
extends Vector<E>

The Stack class represents a last-in-first-out (LIFO) stack of objects. It extends class Vector with five operations that allow a vector to be treated as a stack. The usual push and pop operations are provided, as well as a method to peek at the top item on the stack, a method to test for whether the stack is empty, and a method to search the stack for an item and discover how far it is from the top.

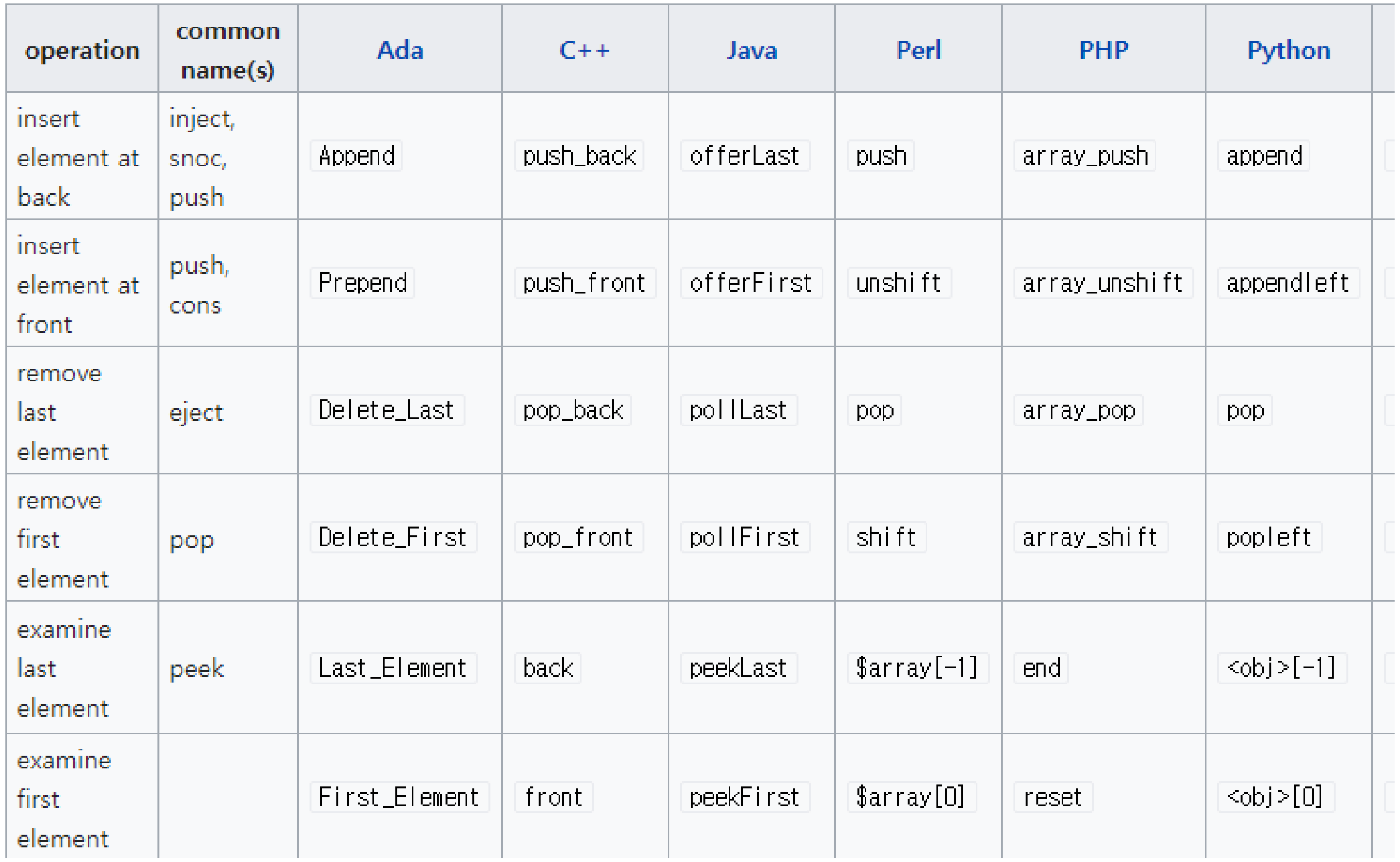
When a stack is first created, it contains no items.

A more complete and consistent set of LIFO stack operations is provided by the Deque interface and its implementations, which should be used in preference to this class. For example:

```
Deque<Integer> stack = new ArrayDeque<Integer>();
```

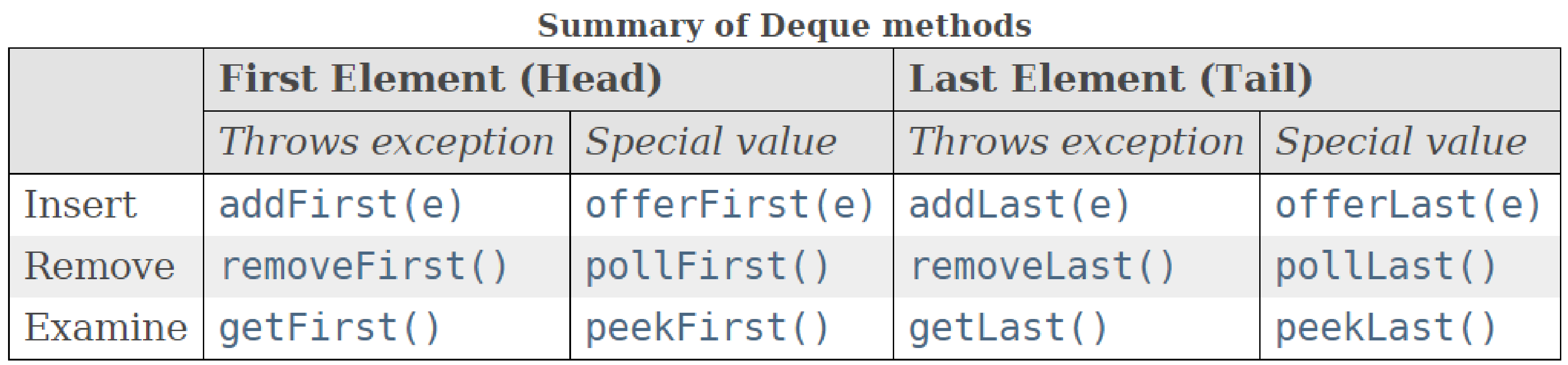
Deque ?

- Double ended queue -> 일반화된 큐
- 이중 큐
  - 스택 + 큐



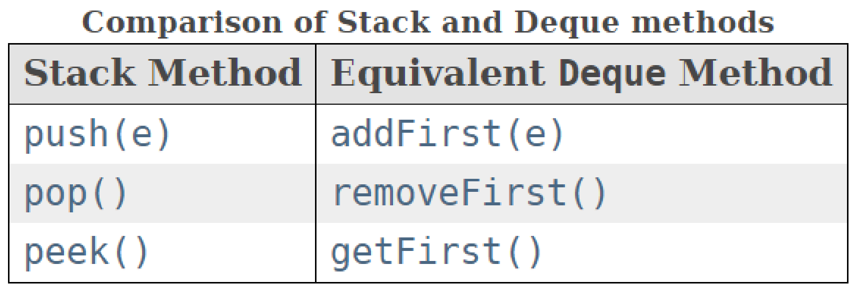
operation	common name(s)	Ada	C++	Java	Perl	PHP	Python
insert element at back	inject, snoc, push	Append	push_back	offerLast	push	array_push	append
insert element at front	push, cons	Prepend	push_front	offerFirst	unshift	array_unshift	appendleft
remove last element	eject	Delete_Last	pop_back	pollLast	pop	array_pop	pop
remove first element	pop	Delete_First	pop_front	pollFirst	shift	array_shift	popleft
examine last element	peek	Last_Element	back	peekLast	$array[-1]	end	<obj>[-1]
examine first element		First_Element	front	peekFirst	$array[0]	reset	<obj>[0]

Deque 메소드



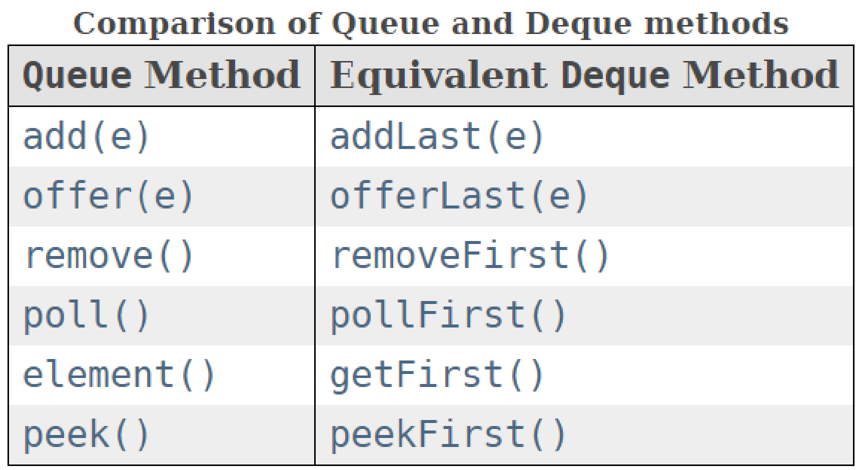
Summary of Deque methods

	First Element (Head)		Last Element (Tail)	
	Throws exception	*Special value*	*Throws exception*	*Special value*
Insert	`addFirst(e)`	`offerFirst(e)`	`addLast(e)`	`offerLast(e)`
Remove	`removeFirst()`	`pollFirst()`	`removeLast()`	`pollLast()`
Examine	`getFirst()`	`peekFirst()`	`getLast()`	`peekLast()`



Comparison of Stack and Deque methods

Stack Method	Equivalent Deque Method
`push(e)`	`addFirst(e)`
`pop()`	`removeFirst()`
`peek()`	`getFirst()`



Comparison of Queue and Deque methods

Queue Method	Equivalent Deque Method
`add(e)`	`addLast(e)`
`offer(e)`	`offerLast(e)`
`remove()`	`removeFirst()`
`poll()`	`pollFirst()`
`element()`	`getFirst()`
`peek()`	`peekFirst()`

자바 스택 활용 코딩테스트 문제 예

- 괄호쌍
- 계산기
- 미로찾기

스택 자료구조 직접 구현

- interface와 클래스 이용
- stack method 구현

```java
package com.cscnu.stack;

public interface Stack {
  public Object peek ();
  public Object pop ();
  public void push (Object object);
  public int size ();
  public boolean isEmpty ();
}
```

```java
package com.cscnu.stack;

import com.cscnu.list.*;

public class ListStack implements Stack {
  private SingleLinkedList list = new SingleLinkedList ();

  public Object peek () {
    if (isEmpty()) throw new IllegalStateException("stack is empty.");
    return list.getLast().data;
  }

```

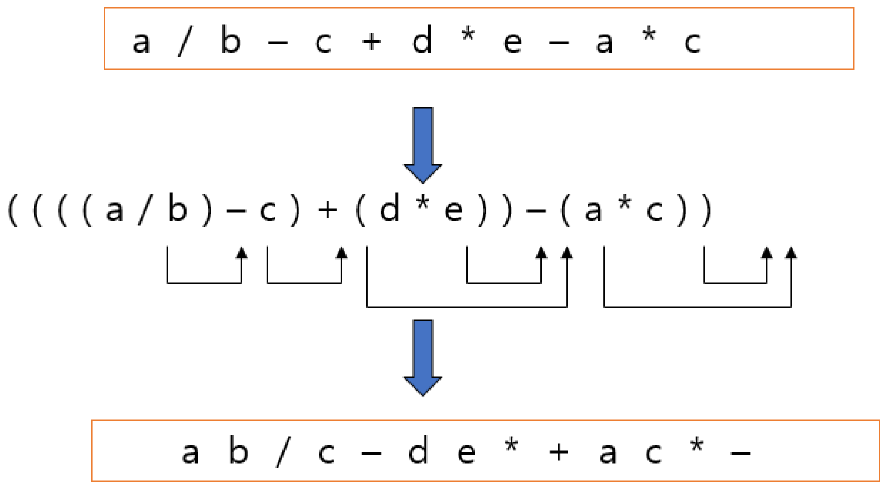
```java
  public Object pop () {
    if (isEmpty()) throw new IllegalStateException("stack is empty.");
    Object item = list.getLast().data;
    list.removeLast();
    return item;
  }

  public void push (Object object) {
    list.insertLast (object);
    return;
  }

  public int size () {
    return list.getSize();
  }

  public boolean isEmpty () {
    return list.isEmpty();
  }
}
```

스택 활용 계산기 문제

- 계산식 표현 변환 문제
  - 사람이 읽기 편한 중위식(infix)
    - 1 + 2 * 3
  - 컴퓨터가 읽기 편한 후위식(postfix)
    - 1 2 3 * +
- 후위식 표현 후 계산 문제
  - 1 2 3 * + 계산 결과는?

중위식 -> 후위식 변환 방법

1. 중위 표기식을 완전하게 괄호로 묶는다.
2. 각 연산자에 해당되는 오른쪽 괄호로 연산자를 이동시킨다.
3. 괄호를 모두 제거한다.
   - 피연산자 순서는 바뀌지 않음


a / b – c + d * e – a * c
((((a / b) – c) + (d * e)) – (a * c))
a b / c – d e * + a c * –

스택이용 변환 방법

Infix to Postfix

6 / 2 – 3 + 4 * 2 -> 6 2 / 3 – 4 2 * +

Token	Stack			Top	Output
	[0]	[1]	[2]		
6				– 1	6
/	/			0	6
2	/			0	6 2
–	–			0	6 2 /
3	–			0	6 2 / 3
+	+			0	6 2 / 3 –
4	+			0	6 2 / 3 – 4
*	+	*		1	6 2 / 3 – 4
2	+	*		1	6 2 / 3 – 4 2
eos				– 1	6 2 / 3 – 4 2 * +

정리

- 스택의 개념
- 스택을 구현하는 방법
  - java.util.Stack 활용
  - 직접 구현