e-learning

2020 학년도 1 학기

컴퓨터 정보과

자료구조(Data Structures)

담당교수 : 김주현

제 4 주차 / 제 2 차시

DIT 동의과학대학교
DONG-EUI INSTITUTE OF TECHNOLOGY

스택 자료구조의 응용

- 역순 문자열 만들기
- 시스템 스택
- 수식의 괄호 검사
- 후위 표기법
- 미로 찾기
- 트리의 방문
- 그래프의 깊이 우선 탐색
- etc

문자열을 역으로 출력하는 다양한 방법

```java
import java.util.Scanner;


public class ReverseString {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String str = scanner.nextLine();

        for (int i = str.length()-1; i>=0; i--){
            System.out.print(str.charAt(i));
        }

    }

}
```

```java
import java.util.Stack;

public class Rev_str {
    public static void main(String[] args) {
        Stack<Character> stack = new Stack<Character>();
        String str = "String reverse using Stack";

        for(int i=0; i<str.length();i++)
            stack.push(str.charAt(i));

        while(!stack.empty())
            System.out.print(stack.pop());
    }
}
```

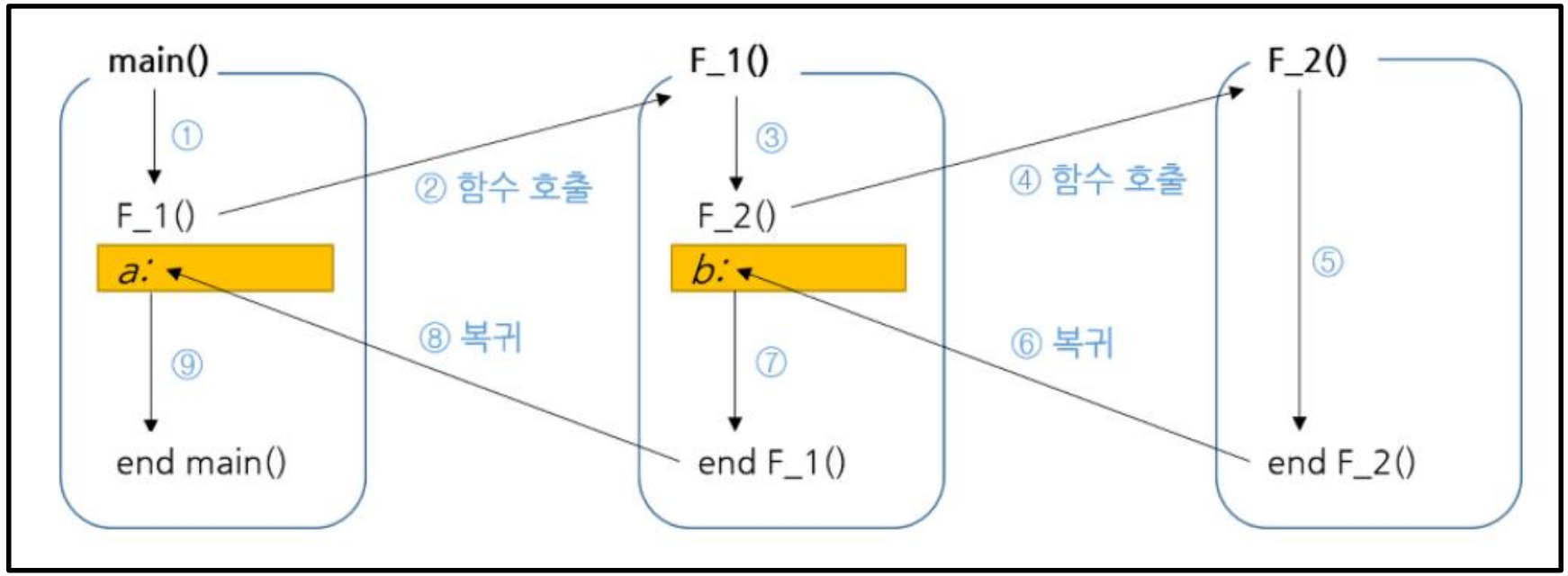
```java
import java.util.Scanner;

public class ReverseString {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String str = scanner.nextLine();

        StringBuffer sb = new StringBuffer();
        sb.append(str);

        System.out.println(sb.reverse());

    }
}
```

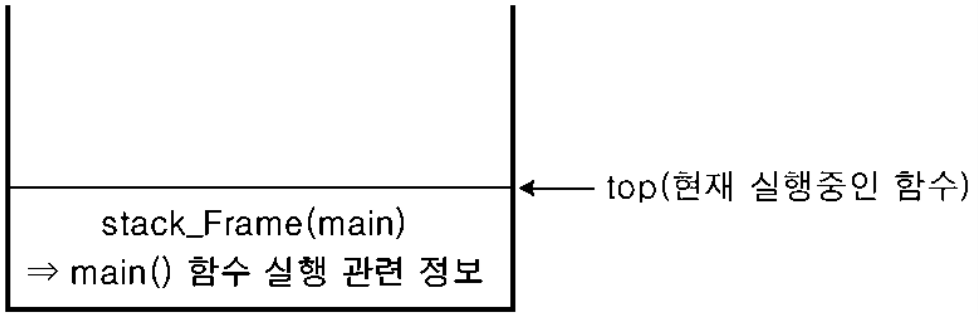
https://docs.oracle.com/en/java/javase/14/

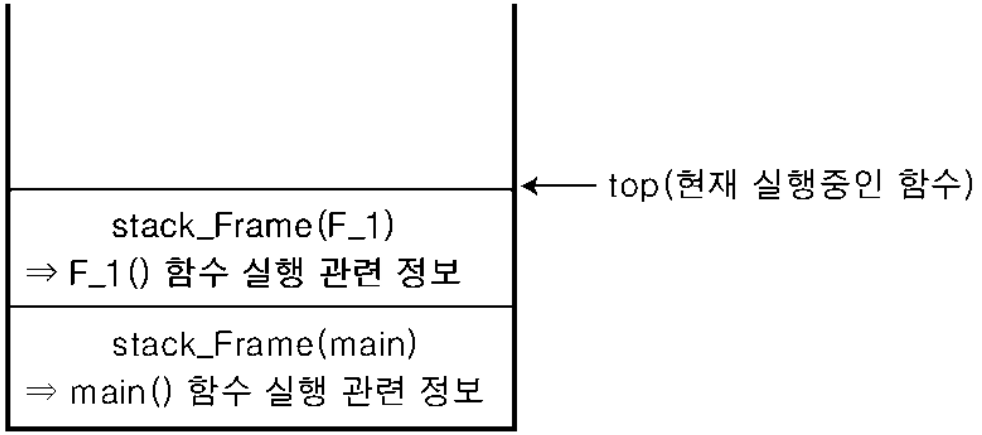
API Documentation

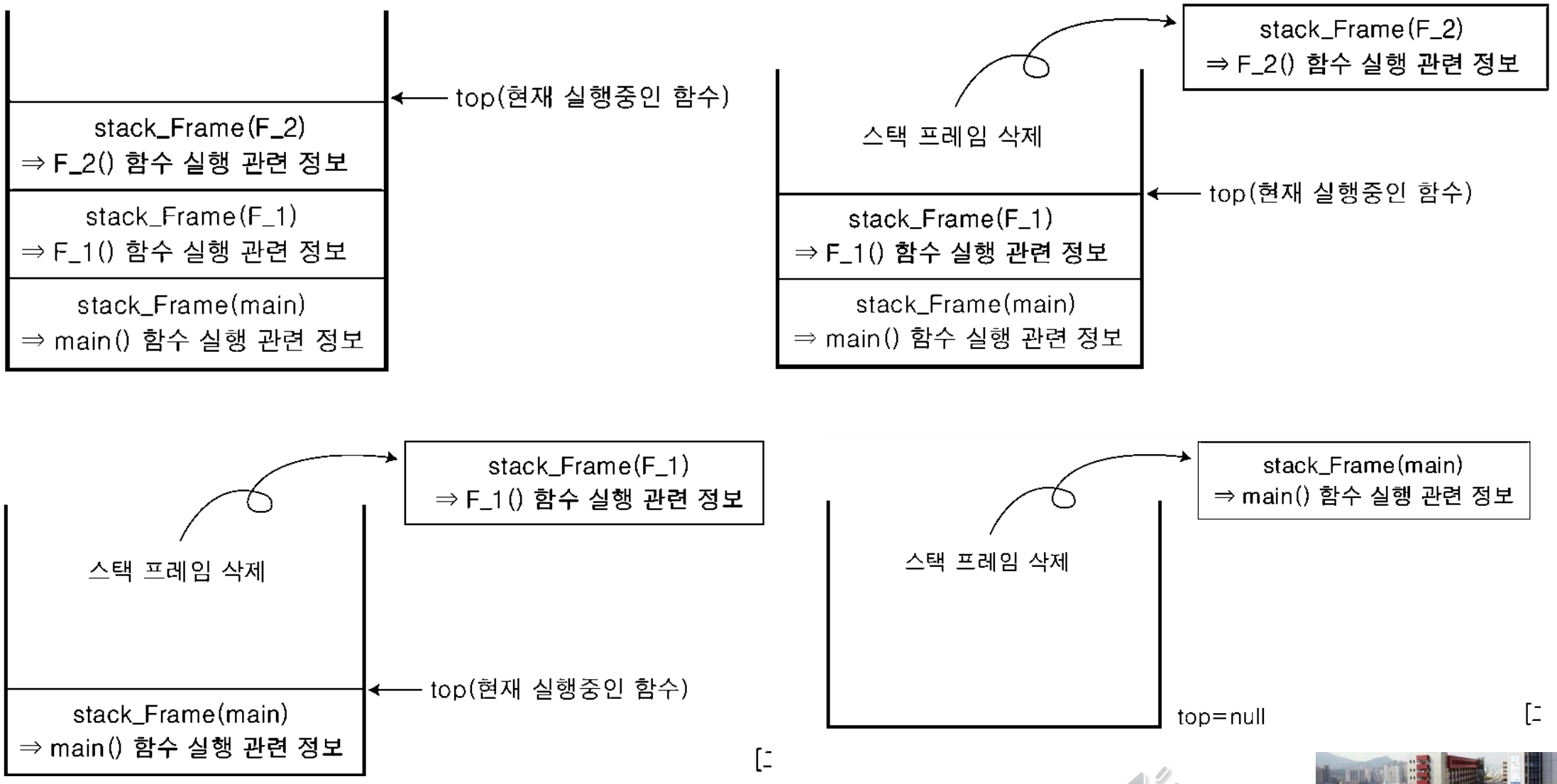
DIT 동의과학대학교
DONG-EUI INSTITUTE OF TECHNOLOGY

시스템 스택

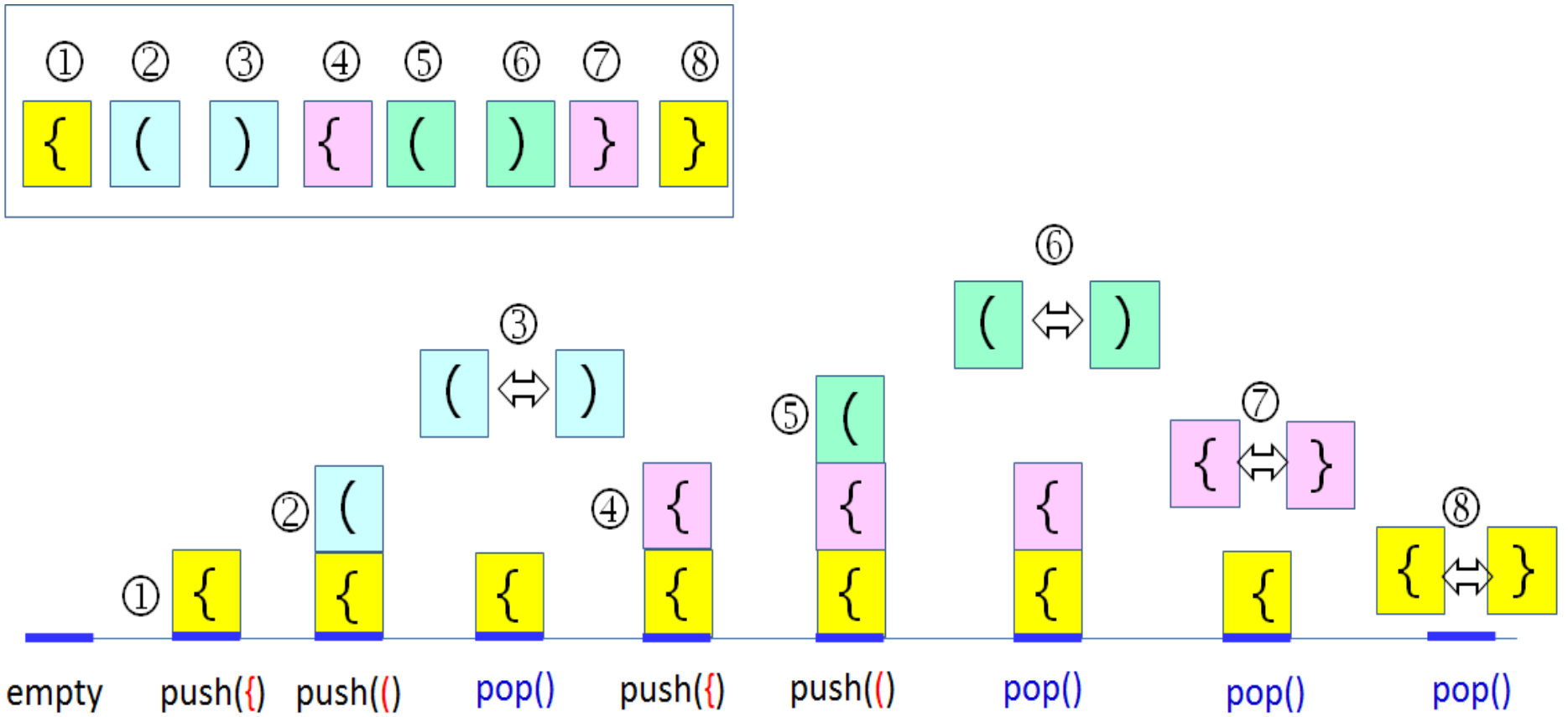

main()
①
F_1()
a:
⑨
end main()
② 함수 호출
⑧ 복귀
F_1()
③
F_2()
b:
⑦
end F_1()
④ 함수 호출
⑥ 복귀
F_2()
⑤
end F_2()


stack_Frame(main)
⇒ main() 함수 실행 관련 정보
top(현재 실행중인 함수)


top(현재 실행중인 함수)
stack_Frame(F_1)
⇒ F_1() 함수 실행 관련 정보
stack_Frame(main)
⇒ main() 함수 실행 관련 정보


stack_Frame(F_2)
⇒ F_2() 함수 실행 관련 정보
stack_Frame(F_1)
⇒ F_1() 함수 실행 관련 정보
stack_Frame(main)
⇒ main() 함수 실행 관련 정보
top(현재 실행중인 함수)
스택 프레임 삭제
top=null

수식의 괄호 검사



```
interface Stack{
        boolean isEmpty();
        void push(char item);
        char pop();
        void delete();
        char peek();
}

class StackNode{
        char data;
        StackNode link;
}
```

```
class LinkedStack implements Stack{
        private StackNode top;

        public boolean isEmpty(){
                return (top == null);
        }

        public void push(char item){
                StackNode newNode = new StackNode();
                newNode.data = item;
                newNode.link = top;
                top = newNode;
        }
```

```
public char pop(){
        if(isEmpty()) {
                System.out.println("Deleting fail! Linked Stack is empty!!");
                return 0;
        }
        else{
                char item = top.data;
                top = top.link;
                return item;
        }
}

public void delete(){
        if(isEmpty()){
                System.out.println("Deleting fail! Linked Stack is empty!!");
        }
        else {
                top = top.link;
        }
}
```

```
        public char peek(){
                if(isEmpty()){
                        System.out.println("Peeking fail! Linked Stack is empty!!");
                        return 0;
                }
                else
                        return top.data;
        }

        public void printStack(){
                if(isEmpty())
                        System.out.printf("Linked Stack is empty!! %n %n");
                else{
                        StackNode temp = top;
                        System.out.println("Linked Stack>> ");
                        while(temp != null){
                                System.out.printf("\t %c \n", temp.data);
                                temp = temp.link;
                        }
                        System.out.println();
                }
        }
}
```

```
class OptExp{
        private String exp;
        private int expSize;
        private char testCh, openPair;

        public boolean testPair(String exp){
                this.exp = exp;
                LinkedStack S = new LinkedStack();
                expSize = this.exp.length();
                for(int i=0; i<expSize; i++){
                        testCh = this.exp.charAt(i);
                        switch(testCh){
                                case '(' :
                                case '{' :
                                case '[' :
                                        S.push(testCh); break;
                                case ')' :
```

```
                        case '}' :
                        case ']' :
                                if(S.isEmpty()) return false;
                                else{
                                        openPair = S.pop();
                                        if((openPair == '(' && testCh != ')') ||
                                           (openPair == '{' && testCh != '}') ||
                                           (openPair == '[' && testCh != ']'))
                                           return false;
                                   else break;
                                }
                        }
                }
                if (S.isEmpty()) return true;
                else return false;
        }
}
```

```
class Bracket_test{
        public static void main(String args[]){
                OptExp opt = new OptExp();
                String exp = "(3*5)-(6/2)";

                System.out.println(exp);

                if(opt.testPair(exp))
                        System.out.println("괄호 맞음!");
                else
                        System.out.println("괄호 틀림!!!");

        }
}
```

중위 표기법, 후위 표기법, 전위 표기법 수식

중위표기법	후위표기법	전위표기법
A + B	A B +	+ A B
A + B – C	A B + C –	+ A – B C
A + B * C – D	A B C * + D –	– + A * B C D
(A + B) / (C – D)	A B+ C D – /	/ + A B – C D

중위표기법 수식을 후위표기법으로 변환 방법

① 왼쪽 괄호를 만나면 무시하고 다음 문자를 읽는다.

② 피연산자를 만나면 **출력**한다.

③ 연산자를 만나면 스택에 **push**한다.

④ 오른쪽괄호를 만나면 스택을 **pop**하여 출력한다.

⑤ 수식이 끝나면, 스택이 공백이 될 때까지 **pop**하여 출력한다.

중위표기법 수식을 후위표기법으로 변환 알고리즘

```
infix_to_postfix(exp)
    while(true) do {
        symbol ← getSymbol(exp);
        case {
            symbol = operand : // 피연산자 처리
                print(symbol);
            symbol = operator :     // 연산자 처리
                push(stack, symbol);
            symbol = ")" :          // 오른쪽 괄호 처리
                print(pop(stack));
            symbol = null :         // 중위 수식의 끝
                while(top > -1) do
                    print(pop(stack));
            else :
        }
    }
end infix_to_postfix( )
```

자바로 구현하고 테스트 해 보세요

스택을 이용하여 후위 표기법 수식 계산

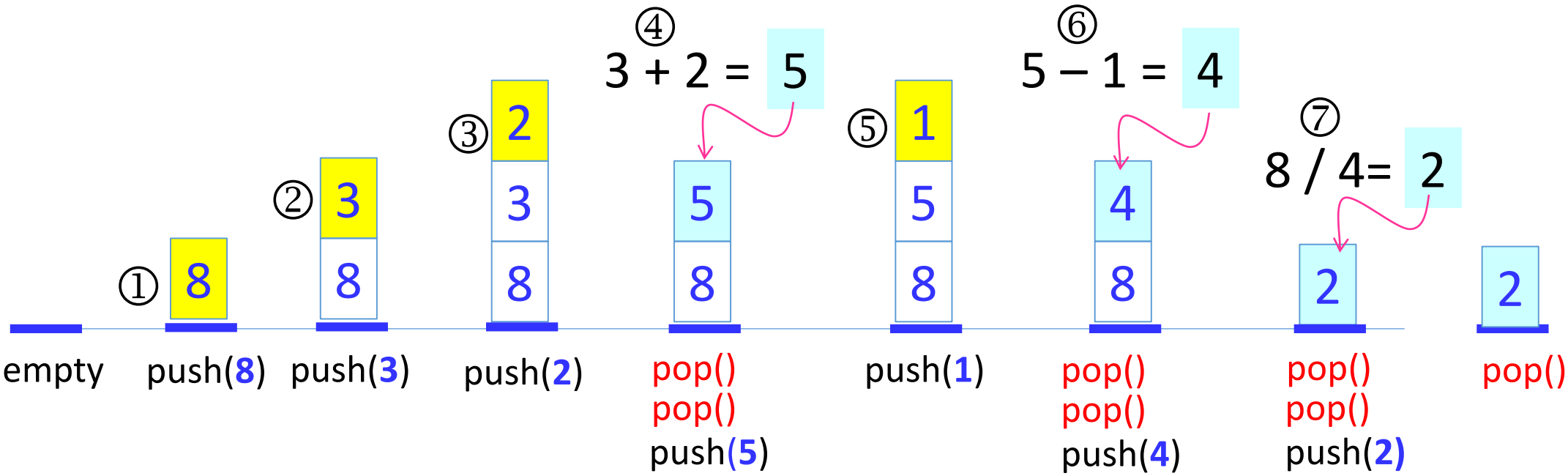
① **피연산자**를 만나면 스택에 **push** 한다.

② **연산자**를 만나면 필요한 만큼의 피연산자를 스택에서 **pop**하여 연산하고, **연산결과**를 다시 스택에 **push** 한다.

③ 수식이 끝나면, 마지막으로 스택을 **pop**하여 출력한다.

수식이 끝나고 스택에 마지막으로 남아있는 원소는 전체 수식의 연산 결과 값이 된다

①	②	③	④	⑤	⑥	⑦
8	3	2	+	1	−	/



후위표기법 수식의 연산 알고리즘

```
evalPostfix(exp)
    while (true) do {
        symbol ← getSymbol(exp);
        case {
            symbol = operand : // 피연산자 처리
                push(Stack, symbol);
            symbol = operator : // 연산자 처리
                opr2 ← pop(Stack);
                opr1 ← pop(Stack);
                result ← opr1 op(symbol) opr2;
                // 스택에서 꺼낸 피연산자들을 연산자로 연산
                push(Stack, result);
            symbol = null : // 후위 수식의 끝
                print(pop(Stack));
        }
    }
end evalPostfix()
```

```
class StackNode{
        int data;
        StackNode link;
}

class LinkedStack{
        private StackNode top;

        public boolean isEmpty(){
                return (top == null);
        }

        public void push(int item){
                StackNode newNode = new StackNode();
                newNode.data = item;
                newNode.link = top;
                top = newNode;
        }
```

```
        public int pop(){
                if(isEmpty()) {
                        System.out.println("Deleting fail! Linked Stack is empty!!");
                        return 0;
                }
                else{
                        int item = top.data;
                        top = top.link;
                        return item;
                }
        }
}
```

```
class OptExp2{
        private String exp;

        public int evalPostfix(String postfix){
                LinkedStack S = new LinkedStack();
                exp = postfix;
                int opr1, opr2, value;
            char testCh;
                for(int i=0; i<7; i++){
                        testCh = exp.charAt(i);
                        if(testCh != '+' && testCh != '-' && testCh != '*' && testCh != '/'){
                                value = testCh - '0';
                                S.push(value);
                        }
                        else{
                                opr2 = S.pop();
```

```
                                opr1 = S.pop();
                                switch(testCh){
                                        case '+' : S.push(opr1 + opr2); break;
                                        case '-' : S.push(opr1 - opr2); break;
                                        case '*' : S.push(opr1 * opr2); break;
                                        case '/' : S.push(opr1 / opr2); break;
                                }
                        }
                }
                return S.pop();
        }
}
```

```
class Postfix_test{
        public static void main(String args[]){
                OptExp2 opt = new OptExp2();
                int result;
                String exp = "35*62/-";
                System.out.printf("\n후위표기식 : %s", exp);
                result = opt.evalPostfix(exp);
                System.out.printf("\n 연산결과 =  %d \n", result);
        }
}
```

회문 검사하기

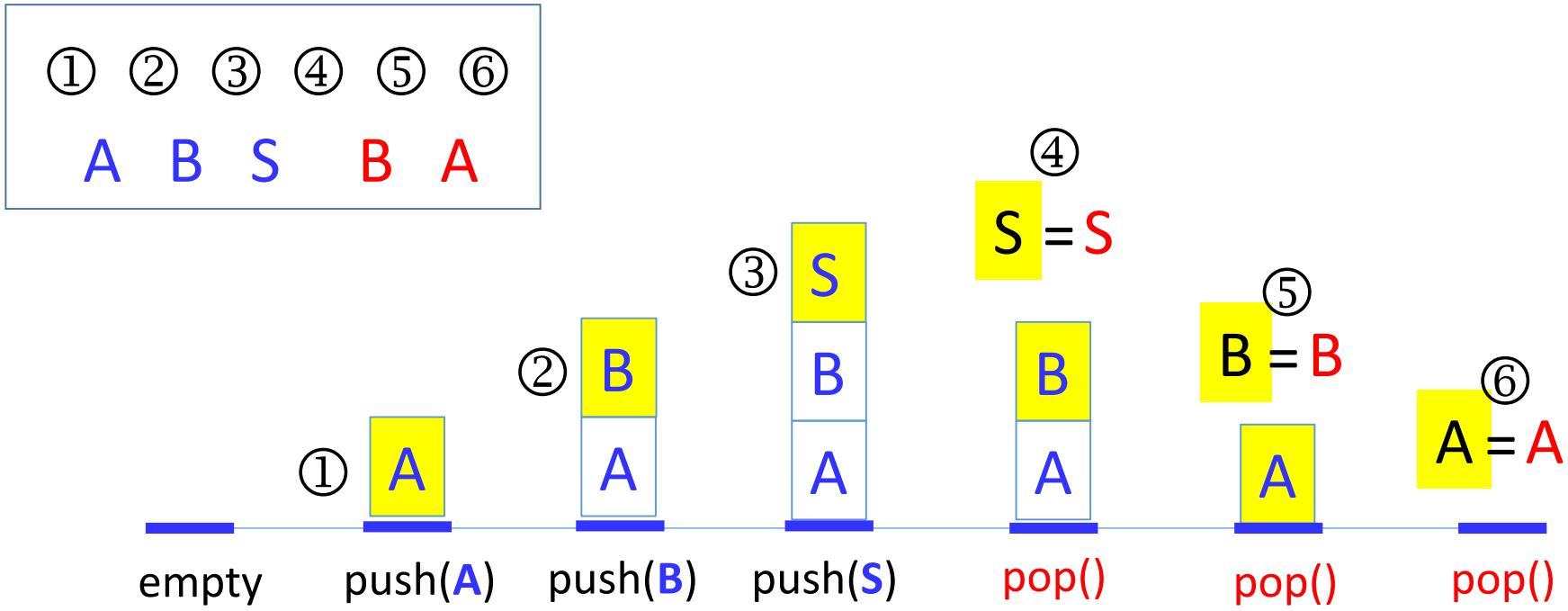
회문(Palindrome): 앞으로부터 읽으나 뒤로부터 읽으나 동일한 스트링

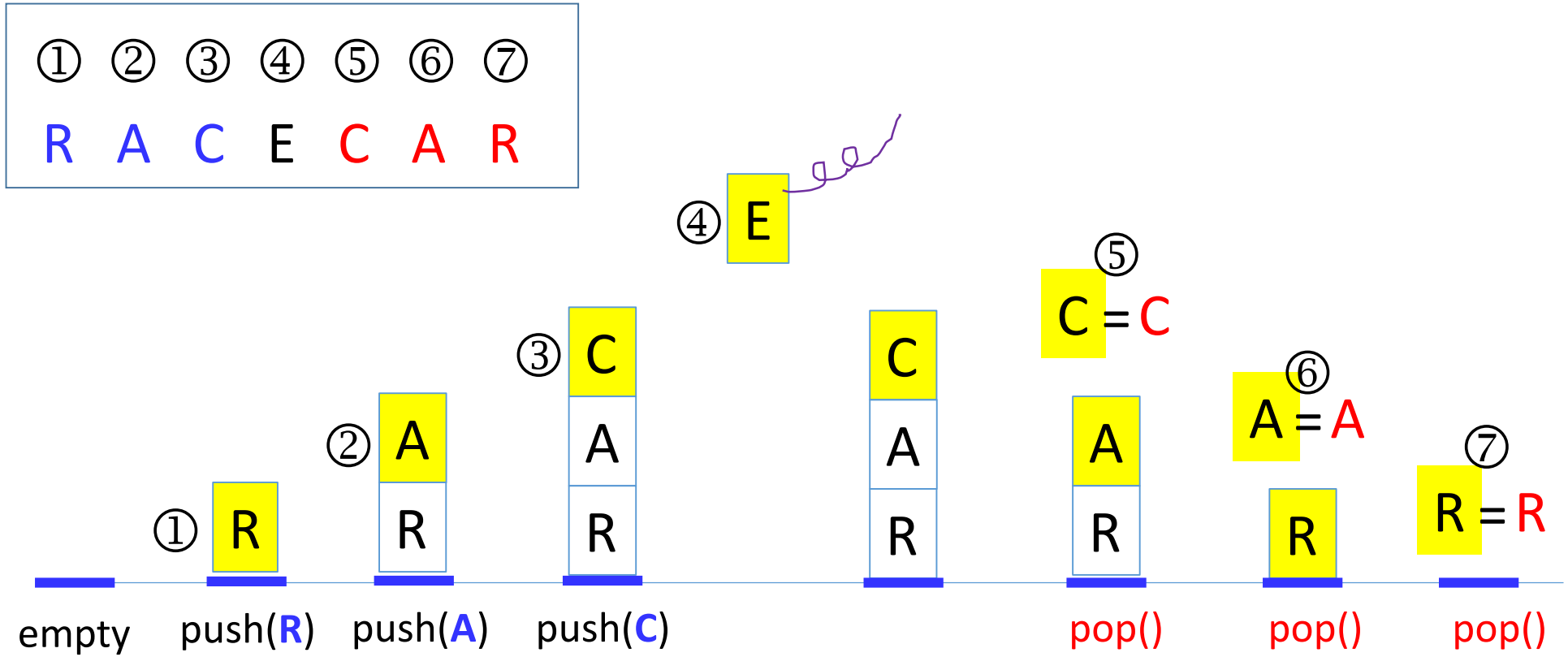
[핵심 아이디어]
전반부의 문자들을 스택에 push한 후, 후반부의 각 문자를 차례로 pop한 문자와 비교

- 회문 검사하기는 주어진 스트링의 앞부분 반을 차례대로 읽어 스택에 push한 후, 문자열의 길이가 짝수이면 뒷부분의 문자 1 개를 읽을 때마다 pop하여 읽어 들인 문자와 pop된 문자를 비교하는 과정을 반복 수행

- 만약 마지막 비교까지 두 문자가 동일하고 스택이 empty가 되면, 입력 문자열은 회문

- 문자열의 길이가 홀수인 경우, 주어진 스트링의 앞부분 반을 차례로 읽어 스택에 push한 후, 중간 문자를 읽고 버린다. 이후 짝수 경우와 동일하게 비교 수행




① ② ③ ④ ⑤ ⑥ ⑦
R A C E C A R
④ E
③ C
② A
① R
C
A
R
⑤
C = C
⑥
A = A
⑦
R = R
empty
push(R)
push(A)
push(C)
pop()
pop()
pop()

Report

문자열 역으로 출력하기, 수식의 괄호검사, 중위표기법을 후위표기법으로 변환하기,
후위 표기법 수식 계산하기, 회문을 구현 및 테스트하고 깃허브에 업로드하면 됩니다.

Reference

- https://ryumin13.tistory.com/entry/%EC%9E%90%EB%A3%8C%EA%B5%AC%EC%A1%B0-%EC%8A%A4%ED%83%9D%EC%9D%98-%EC%9D%91%EC%9A%A9
- https://aroundlena.tistory.com/5
- https://gbsb.tistory.com/239
- https://javaconceptoftheday.com/
- 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