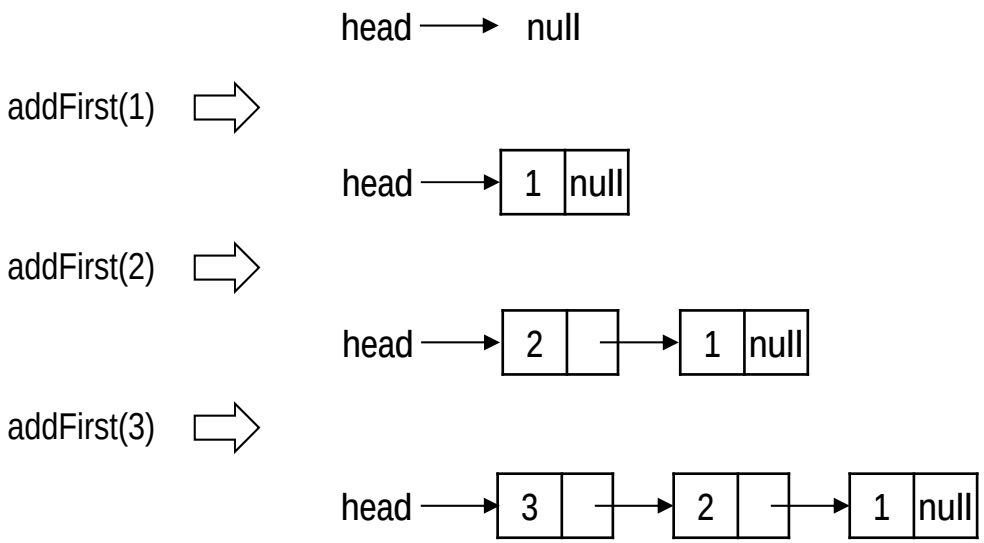
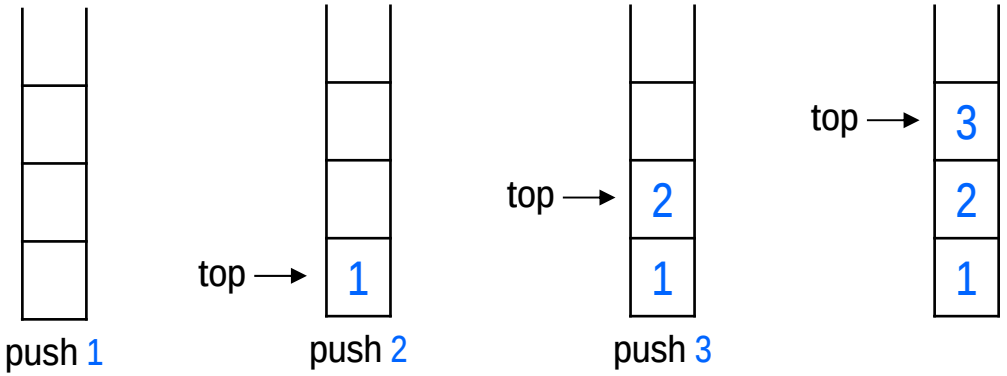
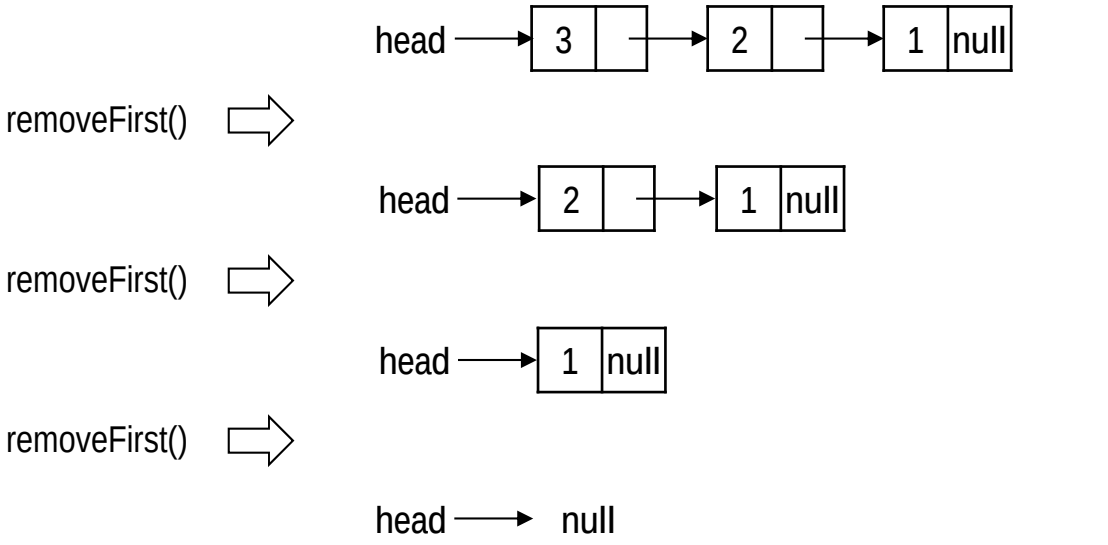
스택

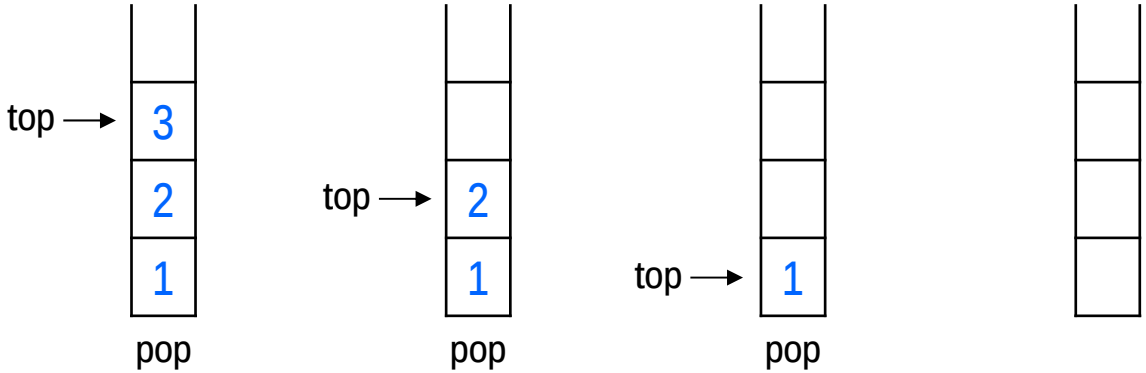
연결리스트와 스택


head
null
addFirst(1)
head
1
null
addFirst(2)
head
2
1
null
addFirst(3)
head
3
2
1
null


top
1
top
2
1
top
3
2
1
push 1
push 2
push 3

연결리스트와 스택


head
3
2
1
null
removeFirst()
head
2
1
null
removeFirst()
head
1
null
removeFirst()
head
null


top
3
2
1
pop
top
2
1
pop
top
1
pop

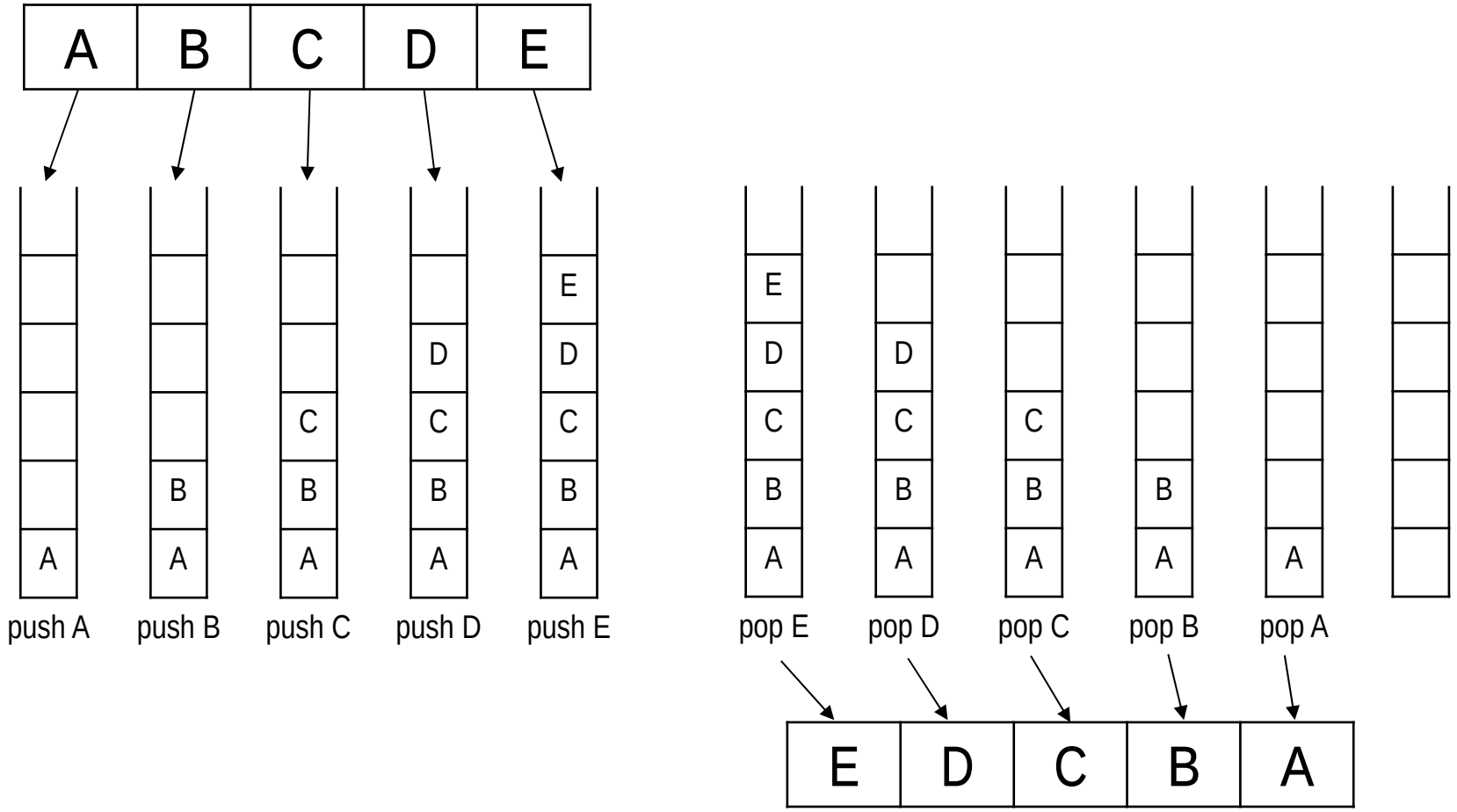
스택

- 스택(stack)
  - 자료의 삽입과 삭제가 한 쪽 끝(top)에서만 발생하는 리스트
  - 삽입, 삭제가 발생하는 위치를 스택의 top이라고 부름
- 스택의 연산
  - Push
    - 스택의 top 위에 새로운 자료를 삽입
  - Pop
    - 스택의 top 위치에 있는 자료를 삭제 (및 반환)
  - Peek
    - 스택의 top 위치에 있는 자료를 삭제하지 않고 반환
  - isEmpty
    - 스택이 비어 있는지 여부를 반환
  - isFull
    - 스택이 포화 상태인지 여부를 반환

Reference: https://en.wikipedia.org/wiki/Stack_(abstract_data_type)

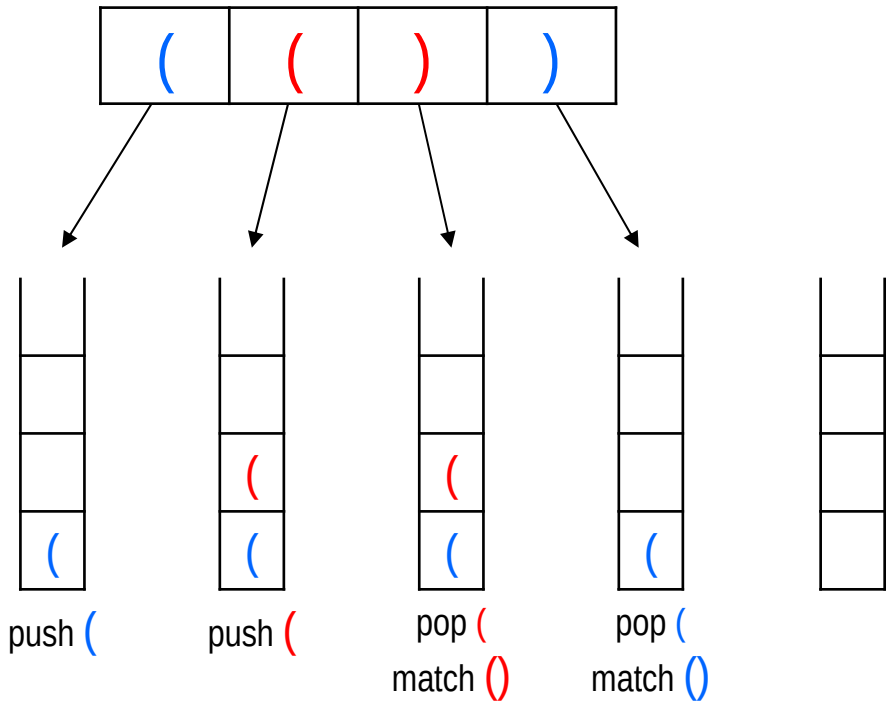
스택 응용: 역순 처리

- 역순 처리



스택 응용: 괄호쌍 매칭

- 괄호쌍 매칭
  - 여는 괄호 → 스택 push
  - 닫는 괄호 → 스택 pop한 괄호와 매치

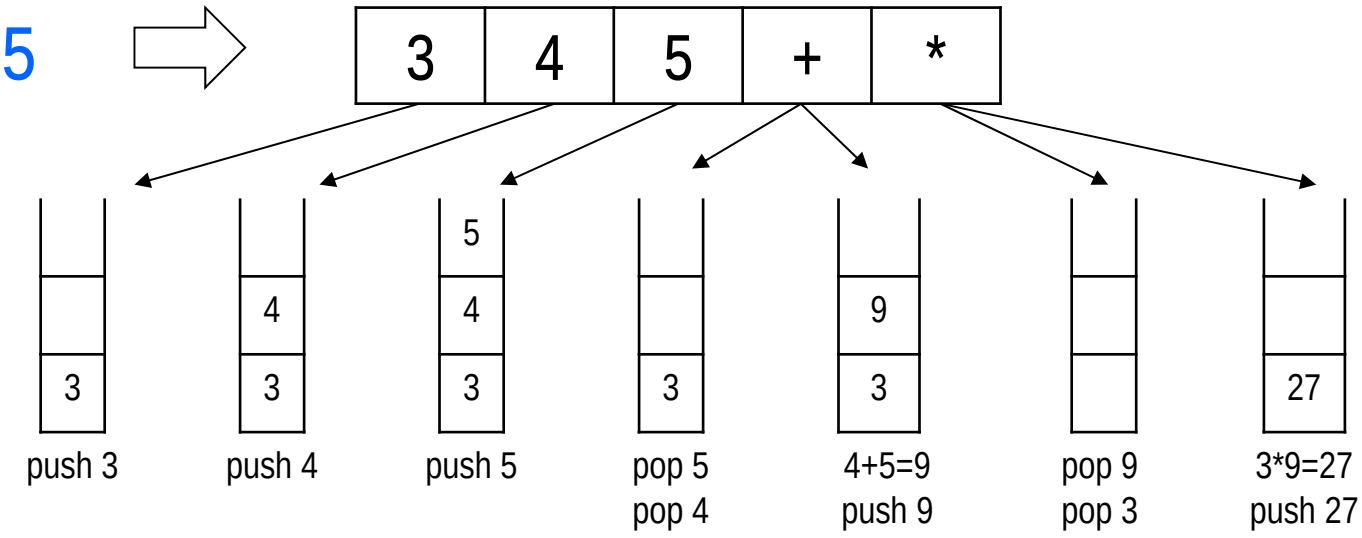


스택 응용: 후위표현식 계산

Reference: https://en.wikipedia.org/wiki/Reverse_Polish_notation

- 후위표현식 계산
  - 피연산자
    - 스택 push
  - 연산자
    - 스택에서 피연산자 pop → pop한 피연산자(들)에 연산자 적용 → 연산 결과 스택 push

3 + 4 * 5



스택 응용: 후위표현식 계산 예시


```
public class Test {
	public static void main(String[] args) {
		int	n1, n2;
		Stack<Integer>		stack=new Stack<>();
		// 후위표현식: 3 4 5 + *
		stack.push(3); // 피연산자 3 => push
		stack.push(4); // 피연산자 4 => push
		stack.push(5); // 피연산자 5 => push
		// 연산자 +
		n2=stack.pop();
		n1=stack.pop();
		stack.push(n1+n2);
		// 연산자 *
		n2=stack.pop();
		n1=stack.pop();
		stack.push(n1*n2);
		System.out.println("계산결과="+stack.pop());
	}
}
```

배열 기반 스택 구현

- 배열 기반 스택 구현
  - top 초기값은 -1
  - push → top을 1 증가시킨 후 top 위치에 자료 삽입
  - pop → top 위치의 자료를 추출하고 top을 1 감소
  - isEmpty → top의 값이 -1이면 true
  - isFull → top의 값이 스택 최대 크기보다 1 적은 값이면 true

```
int    stack[]=new int[100]; // 크기 100의 스택 생성
int    top=-1;
stack[++top]=5; // push 5
stack[++top]=9; // push 9
stack[++top]=1; // push 1
for (int i = 0; i <= top; i++) System.out.print(stack[i]+" "); // 스택 출력
System.out.println();
System.out.println(stack[top--]); // pop
for (int i = 0; i <= top; i++) System.out.print(stack[i]+" "); // 스택 출력
```

배열 기반 스택 클래스 구현

- 배열 기반 스택 클래스 구현
  - 고정 크기 스택
  - 포화 및 공백 스택 검사 부재

```
public class SimpleStack {
        int     stack[];
        int     top=-1;
        public SimpleStack(int size) {
                stack=new int[size];
        }
        public void push(int data) {
                stack[++top]=data;
        }
        public int pop() {
                return stack[top--];
        }
        @Override
        public String toString() {
                return "top="+top+", stack="+Arrays.toString(stack);
        }
}
```

```
SimpleStack stack=new SimpleStack(10);

stack.push(5);
stack.push(9);
stack.push(1);

System.out.println(stack);
int     data=stack.pop();
System.out.println("Data deleted from stack:"+data);

System.out.println(stack);
```

동적 배열 기반 스택 클래스 구현

- 동적 배열(dynamic array) 기반 스택 클래스 구현
  - 포화 스택 → 배열 2배 확장
  - 배열 축소 코드 추가 필요
    - ¼ 포화 시 ½ 크기로 축소

```
SimpleStack stack=new SimpleStack();

for (int i = 0; i < 10; i++){
        stack.push(i);
        System.out.println(stack);
}

for (int i = 0; i < 10; i++){
        int     data=stack.pop();
        System.out.println("Data from stack:"+data);
        System.out.println(stack);
}
```

```
public class SimpleStack {
        int     DefaultSize=1, MaxSize;
        int     stack[];
        int     top=-1;
        public SimpleStack() {
                stack=new int[DefaultSize];
                MaxSize=DefaultSize;
        }
        private boolean empty() { return top==-1; }
        private boolean full() { return top==MaxSize-1; }
        public void push(int data) {
                if(full()){
                        MaxSize*=2;
                        stack=Arrays.copyOf(stack, MaxSize);
                }
                stack[++top]=data;
        }
        public int pop() {
                if(empty()) throw new RuntimeException("stack empty");
                return stack[top--];
        }
        @Override
        public String toString() {
                return "top="+top+", stack="+Arrays.toString(stack);
        }
}
```

동적 배열 기반 스택 클래스 구현

- 동적 배열(dynamic array) 기반 스택 클래스 구현
  - 삽입 연산의 시간복잡도 O(n)

References

- C로 쓴 자료구조론 (Fundamentals of Data Structures in C, Horowitz et al.). 이석호 역. 사이텍미디어. 1993.
- 쉽게 배우는 알고리즘: 관계 중심의 사고법. 문병로. 한빛아카데미. 2013.
- C언어로 쉽게 풀어 쓴 자료구조. 천인국 외 2인. 생능출판사. 2017.
- https://introcs.cs.princeton.edu/java/assignments/
- https://en.wikipedia.org/wiki/Stack_(abstract_data_type)
- https://en.wikipedia.org/wiki/Reverse_Polish_notation
- Introduction to Algorithms, Cormen et al., 3rd Edition (The MIT Press)