Stack,Queue,Deck

Stack

스택은 후입선출(Last In First Out)의 개념을 통해 만들어지는 자료구조입니다.

```
protected Object[] elementData;
```

java 기준 vector의 elementdata라는 오브젝트 배열을 가지고 구현되어 있습니다. vector는 기본적으로 모든 오퍼레이션에 동기화 키워드가 붙은 Thread safe한 구조를 가지고 있습니다. 다만 아래 글을 보면 java에서 제공하는 stack의 사용을 권장하지 않는 이유에 대해서 나와있습니다.

4.1. Using the Java *Stack*

Java Collections has a legacy implementation for thread-safe *Stack*, based on *Vector* which is basically a synchronized variant of *ArrayList*.

However, the official doc itself suggests considering using *ArrayDeque*. Hence we won't get into too much detail.

Although the Java *Stack* is thread-safe and straight-forward to use, there are major disadvantages with this class:

- It doesn't have support for setting the initial capacity
- It uses locks for all the operations. This might hurt the performance for single threaded executions.

4.2. Using *ArrayDeque*

Using the *Deque* interface is the most convenient approach for LIFO data structures as it provides all the needed stack operations. *ArrayDeque* is one such concrete implementation.

Since it's not using locks for the operations, single-threaded executions would work just fine. But for multi-threaded executions, this is problematic.

Stack은 기본적으로 세가지 메서드를 가지고 있어야합니다. 각각의 메서드가 의미하는 행위에 따라 구현될 경우 해당 객체는 stack의 종류로 구분될 수 있습니다.

1. push() - 가장 상단부터 쌓는다.
2. pop() - 가장 상단의 데이터를 뽑고 지운다.
3. peek() - 가장 상단의 데이터를 뽑기만 한다..

Queue

큐라는 자료구조는 FIFO 구조를 가집니다. 가장 먼저 들어간게 가장 먼저 나오는 구조로 되어 있는 자료구조를 의미합니다.

큐의 경우 아래의 메서드들을 의미에 맞게 구현했을 때 큐에 해당하는 자료구조로 볼 수 있습니다.

1. offer() : 뒤에 추가합니다.
2. poll() : 맨 앞부터 차근 차근 뽑습니다.(지우기까지)
3. peek() : 맨 앞의 수를 뽑기만 합니다.

```
@Override
public Iterator<T> iterator() {
    return elements.iterator();
}

@Override
public int size() {
    return elements.size();
}

@Override
public boolean offer(T t) {
    if(t == null) return false;
    elements.add(t);
    return true;
}

@Override
public T poll() {
    Iterator<T> iter = elements.iterator();
    T t = iter.next();
    if(t != null){
        iter.remove();
        return t;
    }
    return null;
}

@Override
public T peek() {
    return elements.getFirst();
}
```

Deck(Dequeue)

Stack과 Queue가 섞인 자료구조를 의미합니다. 양 사이드로 데이터를 추가/삭제 할 수 있는 자료구조입니다.

Operation	Method	Method throwing Exception
Insertion from Head	*offerFirst(e)*	*addFirst(e)*
Removal from Head	*pollFirst()*	*removeFirst()*
Retrieval from Head	*peekFirst()*	*getFirst()*
Insertion from Tail	*offerLast(e)*	*addLast(e)*
Removal from Tail	*pollLast()*	*removeLast()*
Retrieval from Tail	*peekLast()*	*getLast()*

메서드명처럼 처음과 끝을 사용자가 정할 수 있습니다. 다만 예외를 던지는 메서드와 던지지 않는 메서드들이 다르게 구성되어 있습니다.