● Stack 두 개를 이용하여 Queue를 구현해라.

▶enqueue()를 구현할 때 첫 번째 stack을 사용하고, dequeue()를 구현할 때 두 번째 Stack을 사용하면 queue를 구현할 수 있습니다. 일단 첫번째 stack에 push를 하여 데이터를 순차적으로 저장하는 것으로 enqueue를 구현합니다. 그리고 dequeue를 구현하려면, 만약 두번재 stack이 비어 있다면 첫번째 stack에서 데이터를 pop하여 두번째 stack에 push하는 것으로 먼저 모든 데이터를 옮깁니다. 그러면 결과적으로 첫번째 stack에 가장 먼저 push됐던 데이터가 두번째 stack의 가장 마지막, top에 위치하게 됩니다. 이때 두번째 stack에서 데이터를 pop하면 가장 먼저 top에 있던 데이터가 추출되기 때문에, 결론적으로는 첫번째 stack에 가장 먼저 추가되었던 데이터가 두번째 stack에서 가장 먼저 추출되고, 아는 FIFO 선입선출로 Queue를 구현하게 됩니다.

```
import java.util.Stack;

class realizeQueue<T> {
    Stack<T> stackFirst, stackSecond;

    // 생성자에서 두 개의 스택을 만든다
    realizeQueue() {
        stackFirst = new Stack<T>();
        stackSecond = new Stack<T>();
    }

    // enqueue: 첫번째 stack에 데이터를 추가한다
    public void enqueue(T value) {
        stackFirst.push(value);
    }

    // dequeue: 두번째 stack이 비어 있을 경우, 첫번째 stack이 완전히 빌 때까지
    // 두번째 stack으로 데이터를 옮기고, 두번째 stack에서 데이터를 pop한다.
    public T dequeue() {
        if (stackSecond.isEmpty()) {
            while (!stackFirst.isEmpty()) {
                stackSecond.push(stackFirst.pop());
            }
        }
        return stackSecond.pop();
    }
}

class Ex {
    public static void main(String[] args) {
        realizeQueue<Integer> que = new realizeQueue<>();
        que.enqueue(1);
        que.enqueue(2);
        que.enqueue(3);
        System.out.println(que.dequeue());
        System.out.println(que.dequeue());
        System.out.println(que.dequeue());
    }
}
```

● 시간 복잡도는 어떻게 되는가?

▶ enqueue() 는 첫번째 stack에 push()를 한 번만 하면 되기 때문에, 시간복잡도는 O(1)입니다.

▶dequeue()는 두 가지 경우가 있습니다. worst case는 두번째 스택이 비어있는 경우입니다. 이 때는 첫번째 스텍에 있는 n개의 데이터를 pop()을 한 이후에 두번째 스택에 push()를 해줘야 합니다. 따라서 총 2*n 번의 operation이 실행되어야 하므로 O(n)의 시간복잡도를 갖습니다. 하지만 두번째 스택이 비어 있지 않는 경우에는 두번째 스택에서 pop()만 해주면 되기 때문에, O(1)의 시간복잡도를 갖습니다. 두 가지 경우를 종합했을 때, amortized O(1)의 시간복잡도를 갖는다고 할 수 있습니다.

Resources: inflearn 개발남 노씨