**<스택 Stack>**

* 일시적으로 저장을 하기 위한 자료구조

건물, 앉아있는, 어두운, 텔레비전이(가) 표시된 사진

자동 생성된 설명

데이터의 입출력 순서는 LIFO(Last In First Out)-> 맨 나중에 들어온 데이터가 제일 먼저 나간다.

Class IntStack{

int max; //스택의 용량

int ptr; //스택의 포인터 (현재 쌓여있는 데이터 수)

int[ ] stk; //스택 본체

}

* 메서드

1. 푸시 push

푸시는 스택에 데이터를 추가하는 메서드이다. 가득 차서 푸시할 수 없는 경우 예외처리한다. 전달 받은 데이터를 stk[ptr]에 저장하고 ptr을 증가시킨다.

1. 팝 pop

스택의 꼭대기 데이터를 제거 하고 그 값을 return 시키는 메서드이다. 스택이 비어있어 팝을 할 수 없을 경우 예외처리한다. ptr값을 감소시키고 stk[ptr]에 저장되어 있는 값을 반환한다.

1. 피크 peak

스택의 꼭대기 데이터를 몰래 보는 메서드이다. 비어있는 경우 예외처리한다. 비어있지 않은 경우 stk[ptr-1]값을 반환한다.

1. 검색 indexOf

Stk에 x라는 데이터가 포함되어 있는 지 조사하는 메서드이다. 검색은 꼭대기에서 바닥 쪽으로 검색한다. 검색에 성공하면 요소의 인덱스를 반환하고 실패하면 -1을 반환한다.

1. 모든 요소 삭제 clear

스택에 쌓여있는 모든 데이터를 삭제한다.

1. 용량 확인 capacity

스택의 용량(max값)을 반환하는 메서드이다.

1. 데이터 수 확인 size

현재 스택에 쌓여있는 데이터 수를 반환하는 메서드이다.

1. 비어 있는지 검사 IsEmpty

스택이 비어 있는지 검사하는 메서드이다. 비어 있으면 true, 그렇지 않으면 false를 반환한다.

1. 가득 찼는지 검사 IsFull

스택이 가득 찼는지 검사하는 메서드이다. 가득 찼으면 true, 그렇지 않으면 false를 반환한다.

1. 스택 안의 모든 데이터 표시 dump

모든 데이터를 바닥에서 꼭대기 순으로 표시하는 메서드이다.

%첨부파일 IntStack과 IntStackTester를 참고하시오.

**<큐 Queue>**

그리기이(가) 표시된 사진

자동 생성된 설명

가장 먼저 넣은 데이터를 가장 먼저 꺼내는 선입선출 구조로 되어있다.

class IntQueue {  
 private int max; // 큐의 용량  
private int front; // 첫 번째 요소 커서  
private int rear; // 마지막 요소 커서  
private int num; // 현재 데이터 수  
 private int[] que; // 큐 본체

}

* 메서드

1. 인큐 enque

큐가 가득 차서 인큐할 수 없으면 예외처리 한다. 맨 마지막 커서(rear)의 뒤로 데이터를 넣어주고 rear의 값은 증가한다. 데이터 개수(num)는 -1 해준다. Rear의 값이 max와 같아지면 rear를 0으로 만들어준다.

1. 디큐 deque

큐가 비어있어 디큐할 수 없으면 예외처리 한다. 첫번째 커서(front)의 값을 반환해준다. 인큐와 마찬가지로 front의 값이 max와 같아지면 front의 값을 0으로 만들어준다.

1. 피크 peak

Que[front]값을 반환하여 맨 위의 데이터를 보여준다. 큐가 비어 있으면 예외처리 한다.

1. 검색 indexOf

검색의 시작 부분은 front이다. 인덱스 계산은 (i + front)%max로 한다. 검색을 성공하면 요소의 인덱스를 반환하고 실패하면 -1을 반환한다.

1. 모든 요소 삭제 clear
2. 용량 확인 capacity
3. 데이터 수 확인 size
4. 비어 있는지 검사 IsEmpty
5. 가득 찼는지 검사 IsFull
6. 스택 안의 모든 데이터 표시 dump

(다 쓰기 귀찮…..)

%첨부파일 IntQueue과 IntQueueTester를 참고하시오.