Data Structures and Algorithms

Lab 7: Stack

I. Objective

After completing this tutorial, you can:

- Implement a stack with a linked list containing ADT.

II. Introduction

In previous tutorial, we have learned how to construct a linked list of general data type **<E>**. In this tutorial, we demonstrate on stack, which is one of the most popular ADT. The order in which elements come off a stack gives rise to its alternative name, **LIFO** (last in, first out). In stack, we have two important methods:

- *Push*, which adds new element to the top of stack;
- *Pop*, which removes the top element of stack.

Fig. 1 illustrates the two methods, *push* and *pop*.

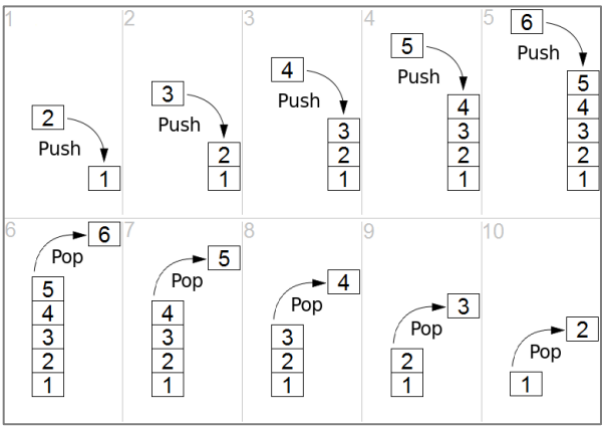
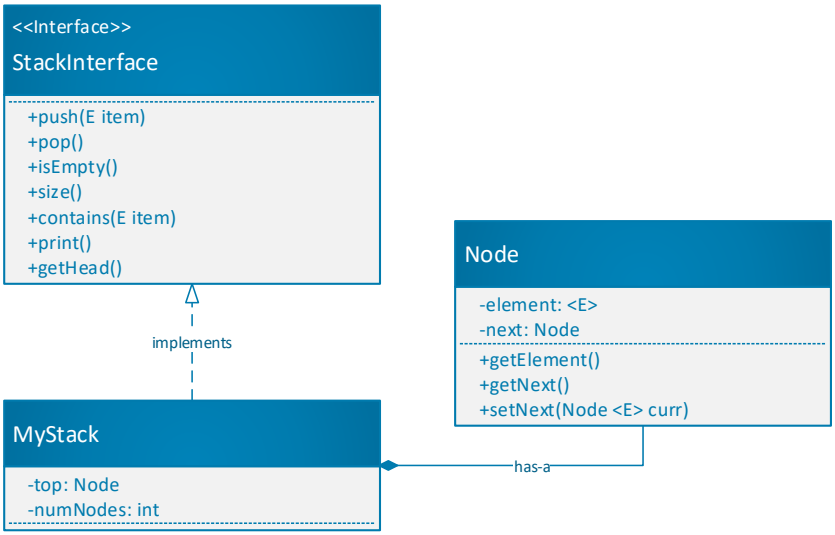


Figure 1 Pop and Push methods in stack

III. UML model



The following figure presents an UML model of stack:

- *StackInterface* represents public functions of stack, *e.g.*, push new item, pop an item.
- *Node* class represents an item (node) in stack.
- *MyStack* class implements *StackInterface* and includes items has *Node* types.

IV. Exercises

1. Based on the previous lab tutorial, you need to implement the stack ADT which contains general data type **`<E>`**. Then, implement **`Fraction`** class and test your program.

2. Compute the result of the following expression by using recursive approach and eliminate recursive by using stack.

$$P(n) = \begin{cases} 2^n + n^2 + P(n-1), & n > 1 \\ 3, & n = 1 \end{cases}$$

3. Write a program that reads in a sequence of characters and prints them in reverse order, using stack.

Write a program that reads in a sequence of characters, and determines whether its parentheses, braces, and curly braces are "balanced". *Hint:* for left delimiters, push onto stack; for right delimiters, pop from stack and check whether popped element matches right delimiter.