Lab 2
Stack and Queue

Quang D. C.
`dcquang@it.tdt.edu.vn`

August 31, 2020

In this tutorial, we will approach two new data structures: Stack and Queue which you can implement by using Linked list.
After completing this tutorial, you can:

- Implement a Stack with a Linked list containing ADT (Abstract Data Type).
- Implement a Queue with a Linked list containing ADT (Abstract Data Type).

1. Introduction

1.1. Stack

First, we demonstrate 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.

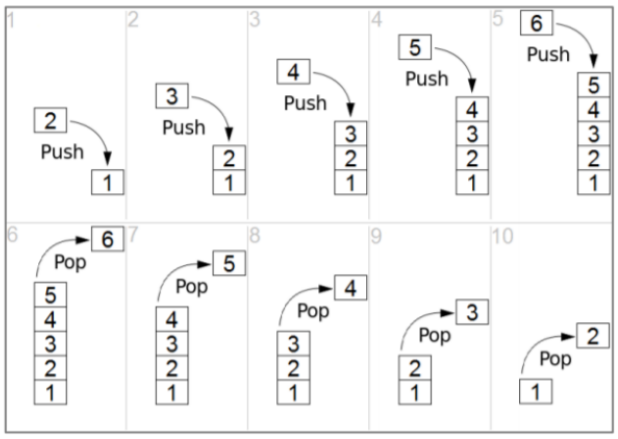


Figure 1: Push and Pop methods in Stack

Figure 1 illustrates the two methods, Push and Pop

1.2. Queue

Next, we consider Queue, which is one of the most popular ADT like Stack. The order in which elements come off a Queue gives rise to its alternative name, **FIFO** (first in, first out). In Queue, we have two important methods:

- enQueue, which adds new element to the Queue;
- deQueue, which removes the first element of Queue.

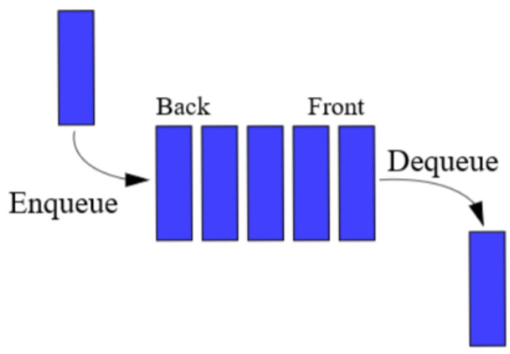


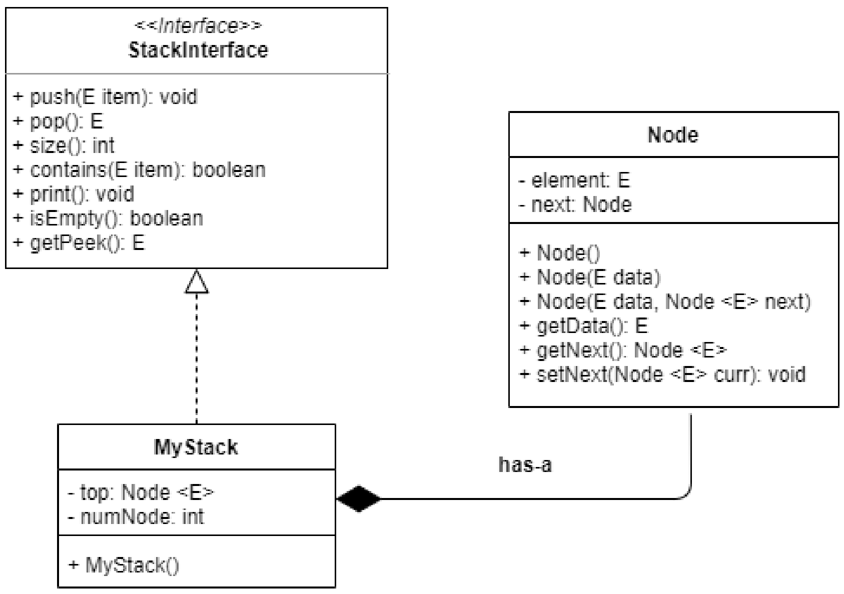
Figure 2: euQueue and deQueue methods in Queue

Figure 2 illustrates the two methods, enQueue and deQueue. We must notice that queue maintains and tracks two positions, *front* and *rear*.

In the next section, we will consider the UML model of Stack and Queue.

2. UML Model

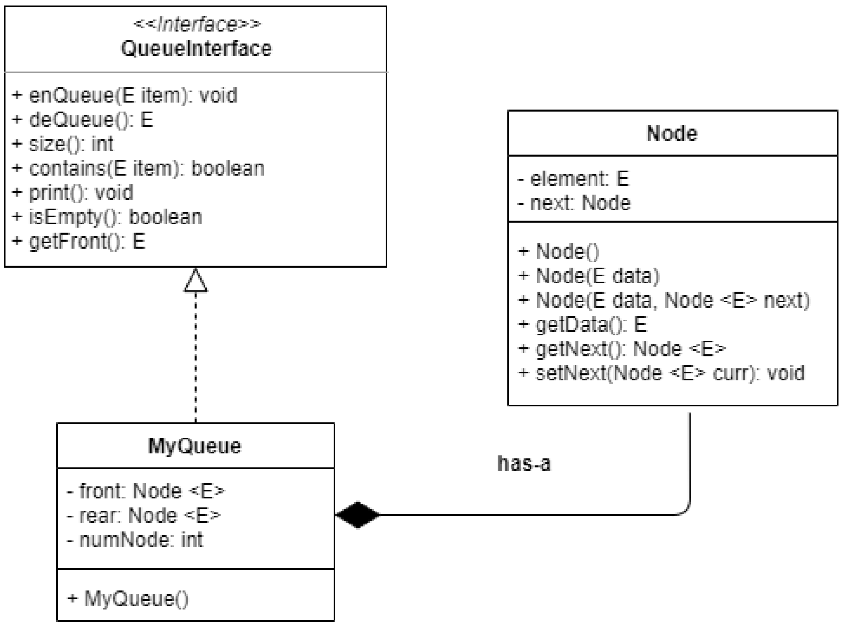
2.1. Class Diagram of Stack



The following figure presents an UML model of Stack:

- *StackInterface* represents public functions of Stack, e.g., push a new item, pop an item.
- *Node* class represents an item (node) in Stack.
- *MyStack* class implements *StackInterface* and includes items that have *Node* type.

2.2. Class Diagram of Queue



The following figure presents an UML model of Queue:

- *QueueInterface* represents public functions of Queue, e.g., enqueue new item, dequeue an item.
- *Node* class represents an item (node) in Queue.
- *MyQueue* class implements *QueueInterface* and includes items have *Node* type.

3. Excercise

Exercise 1

Based on the code in Lab 1, you need to implement the **Stack** ADT which contains general data type <**E**>. Then, implement **Fraction** class and test your program

Exercise 2

Based on the code in Lab 1, you need to implement the **Queue** ADT which contains general data type <**E**>. Then, implement **Fraction** class and test your program

Exercise 3

Compute the result of the following expression by using recursive approach and eliminate recursive by using **Stack**.

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

Exercise 4

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

Exercise 5

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.

Exercise 6

Show how to implement a **Queue** using two **Stack**.

Exercise 7

A palindrome is a word or a phrase that is spelled the same forward and backward. For example, ”dad” is a palindrome; ”A man, a plan, a canal: Panama” is a palindrome if you take out the spaces and ignore the punctuation.

Implement a program to determine whether an input is palindrome using one **<Character> Stack** and one **<Character> Queue**.