CS 445

Rec 9

Agenda

1. **Topic of This Week**
    a. Introducing Stack
    b. Representing Stack with Linked List
2. **Working Session: Lab 8**

Stack

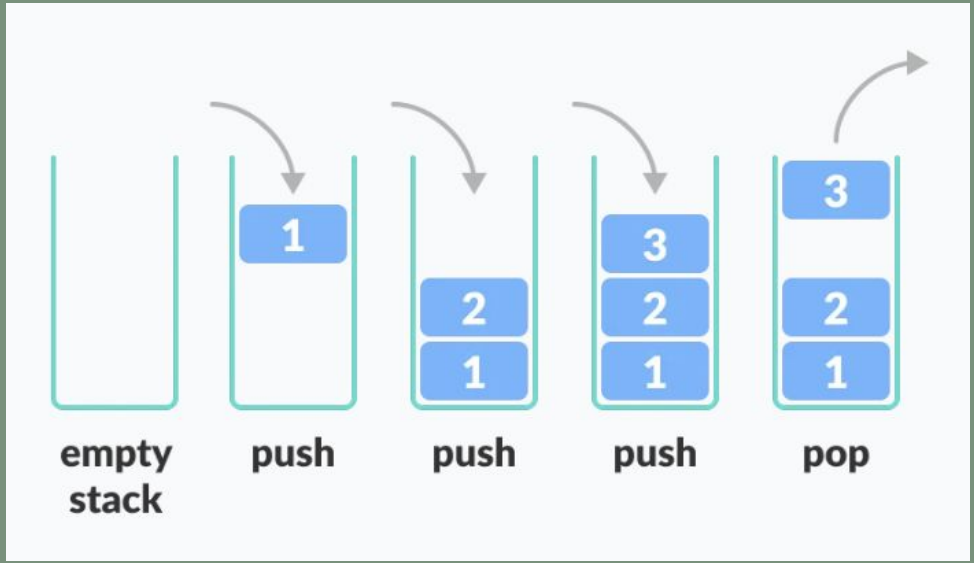
- A stack is a linear data structure that follows the principle of Last In First Out (LIFO)
  - Meaning that the last element inserted is the first one to remove
- Think of it as a pile of plates on top of one another
  - You can:
    - Put a new plate on top
    - Remove the top plate
- With stack, we perform operations (insertion, retrieval, deletion, etc.) on *one data item* at a time
-

We only Operations are perform *at one end* (which we call the “top”)

- Common stack operations:
  - push(), pop(), peak(), isEmpty()
  - All are O(1) in terms of complexity because no

Stack

- **With stack, operations (insertion, retrieval, deletion, etc.) are performed**
  - **on one data item at a time**
  - **at one end only (the "top")**
- **Common stack operations:**
  - push(), pop(), peak(), isEmpty()
  - **All are O(1) in terms of complexity because no traversal is needed**



Lab 8

In Lab 8, you will be representing a tower of disks to get you set up for the Towers of Hanoi Problem, where you implement two methods push() **and** pop() **in Tower.java**

ADT:	**Tower (with Disks)**
Principle/rules:	**Stack (LIFO)**
Implementation:	**Linked-List (with Nodes)**

Implementing a Stack with Linked-List

```
class Disk<T>
{
	T label;
	Disk<T> next;

	Disk(T data)
	{
		this( data, null );
	}

	Disk(T label, Disk<T> next)
	{
		this.label = label;
		this.next = next;
	}

}
```

```
class Node<T>
{
	T data;
	Node<T> next;

	Node(T data)
	{
		this( data, null );
	}

	Node(T data, Disk<T> next)
	{
		this.data = data;
		this.next = next;
	}

}
```

Implementing a Stack with Linked-List

```
public class Tower<T>
{
	private Disk<T> base;
	private Disk<T> top;
	public Tower()
	{
		base = null;
	}

	public boolean empty()
	{
		return (base==null);
	}

	public void push(T label)
	{
		// you implement this
	}

	public Disk<T> pop()
	{
		// you implement this
	}
}
```

```
public class LinkedList<T>
{
	private Node<T> head;
	private Node<T> tail;
	public LinkedList()
	{
		head = null;
	}

	public boolean empty()
	{
		return (head==null);
	}

	public void InsertAtTail(T data)
	{
		// you did this in Lab 3
	}

	public Node<T> RemoveAtTail()
	{
		// you did this in Lab 3
	}
}
```